DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

KOS.content

01 | 2014

Ergebnisse der Untersuchungen des
Kompetenzzentrum Open Source der DHBW-Stuttgart

Frühjahr 2014
band.2

Editorial

Lieber Leser,

vor Ihnen liegt der erste Ergebnisband mit studentischen Ausarbeitungen, die im Rahmen des Forschungsprojekt KOS entstanden sind. KOS steht für Kompetenzzentrum Open Source und bezeichnet ein kooperatives Forschungsprojekt im Studiengang Wirtschaftsinformatik der DHBW Stuttgart, das in Zusammenarbeit mit den dualen Partnern Allianz Deutschland, HALLESCHE Krankenversicherung und Deutsche Rentenversicherung Baden-Württemberg den Einsatz von Open Source Software/Techniken zur Optimierung von Geschäftsprozessen in Versicherungsunternehmen untersucht.

Die Ursprünge des Forschungsprojekts KOS gehen auf das Jahr 2009 zurück, in dem die Duale Hochschule Baden-Württemberg (DHBW) nicht nur den Hochschulstatus erhielt, sondern damit verbunden auch einen Forschungsauftrag. Im Studiengang Wirtschaftsinformatik startete man damals zwar sofort mit den ersten Überlegungen zu möglichen Formen der kooperativen Forschung, es dauerte dann aber doch noch zwei Jahre, bis man sich mit drei dualen Partnern einig darüber war, wie eine Zusammenarbeit auf Forschungsebene aussehen könnte.

Die zwei Vorbereitungsjahre haben dem Projekt gut getan. Denn innerhalb kürzester Zeit entwickelte sich im Projekt KOS eine Form der lehreintegrierten kooperativen Forschung wie es sich alle Beteiligten idealerweise vorgestellt hatten:

Die dualen Partner liefern aus deren betrieblichen Abläufen heraus die zu untersuchenden Fragestellungen, wobei es oft um einen Vergleich des Einsatzes kommerzieller Software mit dem Einsatz von Open-Source-Produkten geht. Die jeweiligen Fragestellungen werden dann in Seminaren an der DHBW von studentischen Arbeitsgruppen analysiert, wobei nicht nur die Dozenten, sondern auch Fachexperten der dualen Partner die Studierenden wissenschaftlich leiten.

Am Ende eines jeden Seminars präsentieren die Studierenden die Untersuchungsergebnisse vor den Vertretern der beteiligten Unternehmen. Meist geht dabei um generische Lösungskonzepte, die von den beteiligten dualen Partnern in konkrete Lösungen für das eigene Unternehmen umgesetzt werden können. Diese Abschlusspräsentationen sind nicht nur für die Unternehmen, sondern auch für die Studierenden etwas Besonderes, da sie ihre Seminarergebnisse vor einem recht großen fachkundigem Publikum „verkaufen“ müssen.

Ein halbes Jahr nach den Ergebnispräsentationen werden die studentischen Ausarbeitungen schließlich in Form eines Sammelbandes veröffentlicht. Das vorliegende Dokument ist der erste Band in dieser Reihe und fasst die Ergebnisse aus dem Seminar im Wintersemester 2013/2014 zusammen. Weitere Bände werden semesterweise folgen, nicht zuletzt deshalb, weil die dualen Partner im Dezember 2013 das zweijährige Projekt KOS um zwei weitere Jahre verlängert haben.

Wir wünschen allen Lesern eine spannende Lektüre und hoffen, dass der vorliegende Sammelband viele interessante Erkenntnisse aus dem Open-Source-Bereich für Sie bereithält.

Prof. Dr. Niko Preiß _ Wissenschaftlicher Leiter
Dipl.-Inform. Michael Hitz _ Projektleiter
Prof. Dr. Thomas Kessel _ Wissenschaftlicher Leiter

INHALT BAND.2

Das Kompetenzzentrum Open Source (KOS)

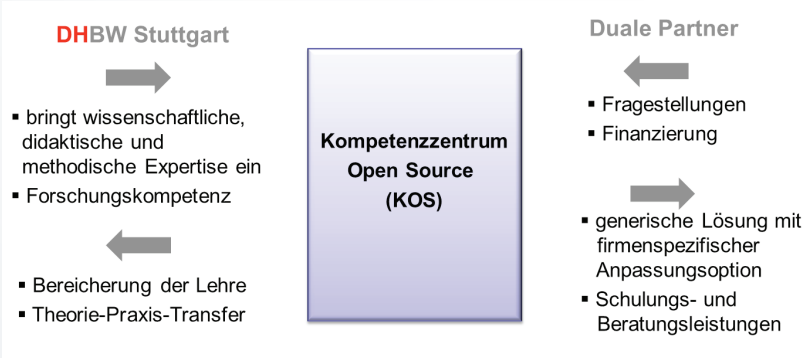
Ziel des Projektes

Das Projekt Kompetenzzentrum Open Source der DHBW Stuttgart wurde mit der Zielsetzung ins Leben gerufen, die Einsatzfelder für Open Source Software in Unternehmen zu identifizieren und durch den Einsatz quelloffener Produkte und deren kostengünstigen Einsatzmöglichkeiten Optimierungen in ausgewählten Geschäftsbereichen zu erzielen.

Dies bedeutet konkret, dass z.B. Open Source Software evaluiert wird, um Lizenzkosten zu reduzieren, bewertet wird, ob sie diverse Qualitätskriterien erfüllt und erfolgreich(er) und effizient(er) in Unternehmen genutzt werden kann. Das Ziel des Projektes ist es hierbei, allgemeingültige Lösungskonzepte für Problemstellungen zu erarbeiten, welche von den am Projekt beteiligten Unternehmen zu firmenspezifischen Lösungen weiterentwickelt werden können. Die beteiligten Unternehmen partizipieren so an den Ergebnissen des Projekts.

Zusammenarbeit mit den Dualen Partnern

Die Zusammenarbeit mit den Dualen Partnern gestaltet sich entlang deren Anforderungen und Bedürfnissen. Sie sind die Themengeber für betriebliche Fragestellungen, die im Rahmen des Projekts untersucht werden. Die DHBW steuert die wissenschaftliche, didaktische und methodische Expertise und Forschungskompetenz bei und untersucht die identifizierten Themenfelder.



Im Rahmen des Projektes steuert die DHBW Stuttgart die wissenschaftliche Expertise und Forschungskompetenz bei zur Bearbeitung der betrieblichen Fragestellungen der Dualen Partner. Es entstehen generische Lösungen, welche von den Partnern an Ihre Situation angepasst werden kann.
Im Rahmen der Arbeit entstehen (generische) Lösungen, an denen die Partner teilhaben können indem sie diese auf ihre spezifische Unternehmenssituation anpassen. Zudem fließen die Ergebnisse in die Arbeit der DHBW ein, sodass hier dem Anspruch an eine hohe Anwendungs- und Transferorientierung ganz im Sinne einer kooperativen Forschung Rechnung getragen wird.

An den Ergebnissen des Projekts partizipieren die Dualen Partner Allianz Deutschland AG, die Deutsche Rentenversicherung Baden-Württemberg und die HALLESCHE Krankenversicherung a.G.

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Asynchrone Servertechnologien zur optimalen Auslastung von Webservern

Zusammenfassung der Projektergebnisse

Christian Brummer
Michael Busam
Thomas Scherer
Matthias Welz

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
WI2011I

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
API	Application Programming Interface
COC	Convention over configuration
CPU	Central Processing Unit
DI	Dependency Injection
DSL	Domain specific language
IoC	Inversion of Control
IP	Internet Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JPA	Java Persistence API
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MVC	Model-View-Controller
ORM	Object-Relational Mapping
REST	Representational State Transfer
SOA	Service-oriented Architecture
SOLID	Single Responsibility Prinzip, Open-Closed Prinzip, Liskovsches Substitutionsprinzip, Interface Segregation Prinzip, Dependency Inversion
SQL	Structured Query Language
STS	Spring Tool Suite
TCP	Transmission Control Protocol
TDD	Test driven development
UML	Unified Modeling Language

Abbildungsverzeichnis

Verzeichnis der Listings

Verzeichnis der Tabellen

1 Einleitung: Struktur der Arbeit (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

Die Arbeit beginnt mit einem umfassenden Theorieteil zur Erläuterung der fachlichen Grundlagen, Motivation und Methodik. Kapitel 2 beginnt mit einer Einführung in die heutige Situation der Informationstechnologie (IT) in Unternehmen, wobei der besondere Fokus auf Webanwendungen liegt. Neben der Struktur von Anfragen und Problemen bei deren Abarbeitung folgt eine Einführung in Java und Multithreading. In Kapitel 3 werden die Grundlagen synchroner Frameworks anhand des Beispiels Spring MVC erläutert, sowie Probleme synchroner Frameworks aufgezeigt. Es folgt in Kapitel 4 eine Einleitung zu asynchroner Verarbeitung. In einer Marktbetrachtung wird im Anschluss das derzeitige Angebot derartiger Frameworks ermittelt und im Hinblick auf die Implementierung die Grundlagen sowie Vor- und Nachteile von Vert.x genauer erläutert. Als Referenz dient dabei der Marktführer Node.js. Danach werden mögliche Anwendungsbereiche asynchroner Frameworks und Fallstricke bei deren Nutzung aufgezeigt.

Kapitel 5 mit den theoretischen Grundlagen zur Szenario- und Anwendungsfallentwicklung bildet den Übergang zum praktischen Teil des Projektes. Dieser ist in drei Teile untergliedert:

In Kapitel 6 wird das gewählte Szenario einer Bank zunächst anhand detaillierter Anwendungsfallbeschreibungen und ausgewählter Anwendungsfalldiagramme illustriert.

Die Umsetzung umfasst Kapitel 7, in dem die Architektur der Anwendung mit Schwerpunkt auf dem Realitätsbezug erläutert wird, sowie Kapitel 8, in dem die Implementierung von synchronem und asynchronem Server, Datenbank sowie dem Benchmarktool zur Performancemessung erklärt wird. Auch auf Schwierigkeiten im Zusammenhang mit der Implementierung wird eingegangen.

Kapitel 9 enthält die Ergebnisse des Leistungsvergleichs und deren Interpretation, insbesondere der Aussagekraft. Zuerst werden jedoch die zugrundeliegenden Messszenarien und Einflussfaktoren erläutert.

Das Fazit in Kapitel 10 dient schließlich zur Zusammenfassung und kritischen Reflexion der Arbeit. Der Code der Implementierung und die umfassenden Ergebnisse werden der Arbeit in digitaler Form beigelegt.

2 IT in Unternehmen (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

2.1 Webanwendungen

Im Zeitalter digitaler Medien gewinnt das Internet und damit verbundene Anwendungen immer mehr an Bedeutung. Die Ubiquität, d.h. Allgegenwärtigkeit dieser Technologien prägt immer stärker das alltägliche Leben. Auch Unternehmen sind im besonderen Maße davon betroffen.

2.1.1 Arten von Webanwendungen (Differenzierung)

Zum genaueren Verständnis werden in diesem Unterkapitel grundlegende Begriffe definiert und voneinander abgegrenzt. Generell ist die Abgrenzung vom **Internet** zum **Word Wide Web (WWW)** nötig. Ersteres zielt dabei auf die eigentliche Verknüpfung von Rechnern auf globaler Ebene. Dementsprechend umfasst das Internet die Vernetzung verschiedenster technischer Endgeräte in einem globalen Netzwerk. Das Konzept des WWW baut dabei auf diese Definition auf. Es beschreibt im Allgemeinen die Ansammlung von Dokumenten und Applikationen im Internet.[1] Folglich beschreibt es den Inhalt des Internets während jenes die eigentliche Vernetzung definiert.

Ein weiterer, im WWW vertretener Oberbegriff ist der der **Website**, welche einen Rahmen für die Dokumente und Applikationen im Internet darstellt. Eine Website beschreibt dabei die Gesamtheit einer hinter einer Adresse stehenden Seite im Word Wide Web.[2] Damit stellt eine Website die Präsenz einer Entität (z.B. Unternehmen, Privatpersonen, Organisationen, etc.) innerhalb des WWW dar. Dabei kann eine einzelne Seite im Internet die Präsenz bereits darstellen, doch in der Regel besteht eine Website aus mehreren Unterseiten, die als Webseiten definiert sind. Diese sind in der Regel in einer Hierarchie angeordnet und über eine URL aufrufbar. Der Domainname sorgt für die eindeutige Zuordnung zur Website.[3] Die Zahl der Domains mit der Endung .de, die Deutschland zugeordnet sind und von der Denic überwacht werden, stieg in den letzten knapp 20 Jahre stark an.[4] Der Inhalt einer solchen Webseite ist frei definierbar und kann aus ein beispielsweise den eingangs angesprochenen Dokumenten und Applikationen des Internets bestehen.

---

[1] Vgl. Bauer, G. (2009), S. 1 ff.

[2] Vgl. Duden auf *http://www.duden.de/rechtschreibung/Website*

[3] Vgl. dazu ausführlich *http://www.itwissen.info/definition/lexikon/Website-website.html*

[4] Vgl. dazu *http://de.statista.com/statistik/daten/studie/39530/umfrage/entwicklung-der-domainzahl-mit-endung-de/*

Letztere sind auch als **Webanwendungen** bekannt Eine Webanwendung kann dabei als eine Software beschrieben werden, die auf Basis von unmittelbar mit dem Internet zusammenhängenden Technologien und Protokollen (z.B. TCP/IP, HTTP) agiert. Ein Benutzer kann mittels eines Webbrowser auf eine Webanwendung zugreifen, die typischerweise auf (mindestens) einem Webserver installiert ist. Der Inhalt (engl. *Content*) kann zuteilen aus statischen und dynamischen Teilen bestehen.[5] Diese lose Beschreibung einer Webanwendung lässt es zu, mehrere spezifischere Ausprägungen unter diesen Begriff zu fassen. In Folge werden die Begriffe Portal und Webservice genauer erläutert.

Ein **Portal** ist eine auf Web-Technologien basierte Applikation, die einen zentralen Zugriff auf eine personalisierbare Plattform bietet.[6] Es handelt sich also um ein Anwendungssystem, das mehrere Funktionen für dessen Benutzer zur Verfügung stellt (z.B. Navigation, Personalisierung). Die Realisierung geschieht über ein Portlet. Auf weitere Funktionalitäten wird an dieser Stelle nicht eingegangen, da dies für die weitere Arbeit nicht relevant ist.[7]

Besonderer Fokus wird nun auf **Webservices** gelegt, da diese die Grundlage für die praktische Umsetzung legen. Dabei ist zu beachten, dass ein Webservice nicht gleichzeitig eine vollständige Webanwendung darstellt. Diese kann jedoch einen Webservice nutzen, bietet aber zusätzlich eine Schnittstelle für Benutzerinteraktionen (Eingaben und Ausgaben) an. Der Arbeitskreis Web Services der Gesellschaft für Informatik definiert einen Webservice als eine selbstbeschreibende- und gekapselte Softwarekomponente, die eine Schnittstelle anbietet, über die ihre Funktion entfernt aufgerufen werden kann, und die lose durch den Austausch von Nachrichten gekoppelt ist.[8] Damit wird auch deren Wiederverwendbarkeit als Dienst beschrieben. Im Zuge dessen sind verschiedene Standards im Bereich der Webservices definiert. Als zentral sind dabei SOAP, WSDL und UDDI anzusehen.

Simple Object Access Protocol (**SOAP**) dient als Austauschprotokoll und ist vom W3 Consortium standardisiert.[9] Im Regelfall wird zur Realisierung XML verwendet, da diese Auszeichnungssprache plattform- und protokollunabhängig[10] ist. Damit erlaubt es, beliebig formatierte Nachrichten zu übertragen. Dies ist ein großer Vorteil von XML. Jedoch wird zur Realisierung ein großer Overhead benötigt, was zu einem relativ großen Datenvolumen bei

[5] Vgl. Bauer, G. (2009), S. 4 ff.
[6] Vgl. Möbus, C. u.a. (2006), S. 93 f.
[7] Vgl. dazu ausführlich: *docs.spring.io/spring/docs/current/spring-framework-reference/html/portlet.html*
[8] Vgl. Finger, P. / Zeppenfeld, K. (2009), S. 40
[9] Vgl. dazu ausführlich *http://www.w3.org/TR/soap12-part1/*
[10] Vgl. Finger, P. / Zeppenfeld, K. (2009), S. 46

der Übertragung von Nachrichten jeglicher Größe führt.[11] Besonders bei vielen aufeinanderfolgenden Anfragen ist dies bemerkbar.

Als Alternative für XML wird nun häufiger **JSON**, eine Teilmenge des JavaScript-Sprachumfangs, verwendet. Es ermöglicht eine platzsparende Speicherung und Übertragung strukturierter Daten. Dabei können die Nachrichten als JavaScript-Objekte umgewandelt und weiterverwendet werden,[12] was für die aktuelle Beliebtheit von JavaScript im Bereich der Webentwicklung von großen Vorteil ist. Auch das relativ geringe Datenvolumen bei der Übertragung spricht für JSON bei Anwendungen mit vielen, sehr kurzen Nachrichten.[13]

Damit ist SOAP für den Zugriff zuständig. Weitere Möglichkeiten sind in Form eines Remote Procedure Calls (RPC) und Representational State Transfer (**REST**) gegeben. Letzteres wurde in der Dissertation von R. Fielding im Jahre 2000 vorgestellt und sorgte für ein Umdenken in der Strukturierung von Webanwendungen. Der Grundgedanke von REST ist dabei, dass jede vorhandene Ressource über eine einheitliche Syntax (URL) adressierbar ist. Der Zugriff auf die jeweilige Ressource ist durch einheitliche Operationen definiert, die auf den Verben zur Beschreibung von Anfragen im HTTP-Protokoll basieren, nämlich GET, POST, PUT und DELETE.[14] REST ist dabei lediglich eine Richtlinie zur Strukturierung von Webanwendungen. Feste Regeln sind nicht gegeben.

Ein weiterer Standard bei Webservice ist die Webservice Description Language (**WSDL**). Diese dient zur Schnittstellenbeschreibung von Webservices. Als Schnittstelle ist dabei die Gesamtheit aller Operationen, die von einem Webserivce angeboten werden, zu betrachten.[15] Die Beschreibung basiert auf den XML-Standard und ist somit protokollunabhängig. Dadurch ermöglicht WSDL den Zugriff auf fremde Web-Services.

Universal Description, Discovery and Integration (**UDDI**) ist ein zentraler Verzeichnisdienst für angebotene Web Services zur Auffindung anderer Web-Services im Internet. In SOA orientierten Architekturen kann der Einsatz von UDDI entfallen, wenn die angebotenen Dienste ausschließlich unternehmensintern genutzt werden,[16] da alle vorhandenen Webservices bekannt sind und manuell referenziert werden können. Eine genauere Standardisierung wie bei SOAP und WSDL gibt es für UDDI nicht.

[11] Vgl. Jäger, K. (2008), S. 207 f.
[12] Vgl. Ebenda, S. 226
[13] Vgl. Ebenda, S. 237
[14] Vgl. Ebenda, S. 224 f.
[15] Vgl. Ebenda, S. 209
[16] Vgl. Finger, P. / Zeppenfeld, K. (2009), S. 40 ff.

2.1.2 Architektur von Webanwendungen / Aufbau einer Webanwendung

Eine Webanwendung basiert immer auf dem Client-Server-Modell. Das **Client-Server-Modell** (siehe Abbildung 1) beschreibt einen generischen Aufbau von zwei Programmen in Form eines Clients und eines Servers, die miteinander kommunizieren. Dabei stellt der Client eine Anfrage (Request) an den Server, der passiv auf Anfragen wartet. Bei Eingang einer Anfrage bearbeitet der Server diese und sendet eine Antwort (Reply) zurück. Die Kommunikation wird dabei immer vom Client initiiert. Daher kann ein Programm als Client beschrieben werden, wenn es von einem anderen Programm einen Dienst anfordert und dabei den Kontakt aufnimmt. Ein Server ist demzufolge ein Programm, was einen Dient für ein anderes Programm (Client) zur Verfügung stellt.[17] In Bezug auf eine Webanwendung initiiert ein Anwender mittels seines Webbrowsers eine Anfrage an den (Web-)Server, der die angeforderte Information bereitstellt.

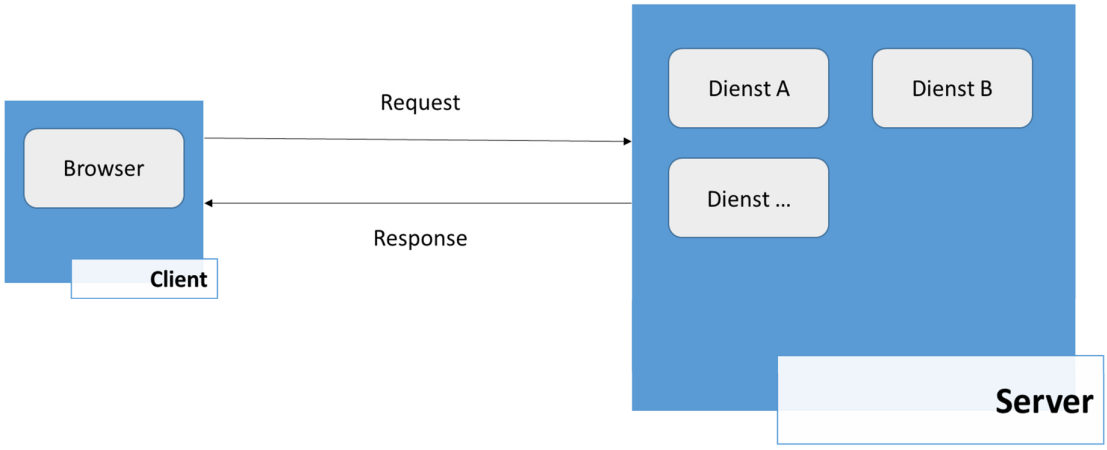


Abbildung 1: Client-Server-Modell

Die Webanwendung, mittels derer die Kommunikation zwischen Client und Server abläuft, kann dabei nach verschiedenen Architekturmodellen entwickelt sein. In der Geschäftswelt hat die Serviceorientierung an Bedeutung gewonnen und so auch die *Service-orientierte Architektur* (**SOA**). SOA beschreibt dabei ein Konzept, um verteilte IT-Services in Geschäftsanwendungen einzubinden oder diese anzubieten. Der Grundgedanke hierbei liegt darin, unterschiedliche Systeme über Services lose miteinander zu koppeln, also das Maß der Abhängigkeit möglichst gering zu halten. Dadurch ist es möglich, neue Anwendungen mittels Kompositionen zu entwickeln.[18] Dies wird durch eine Unterteilung der Anwendung in verschiedene Bestandteile ermöglicht. In der klassischen Software-Architektur ist eine Anwendung gemäß dem 3-Schichten-Architekturmodell aufgebaut:

[17] Vgl. Brothuhn, R. / Chantelau, K. (2010), S. 40
[18] Vgl. Finger, P. / Zeppenfeld, K. (2009), S. 3

- Präsentationsschicht (Darstellung)
- Anwendungslogik (Verarbeitung)
- Persistenzschicht (Datenspeicherung)

Damit wird beispielsweise die Möglichkeit geschaffen, die Aufgaben einer Webanwendung durch ein verteiltes System erledigen zu lassen. Einzelne Komponenten werden von der Anwendung gelöst und verteilt. Die Gesamtheit aller Teile ergibt die Webanwendung.[19]

Bei einer SOA ist eine solche Aufteilung anders angedacht. Kafzig definiert SOA in seinem Buch „Enterprise SOA“ folgendermaßen: „Eine Serviceorientierte Architektur (SOA) ist eine Softwarearchitektur, die auf den Schlüsselkonzepten Anwendungs-Frontend, Service, Service-Repository und Service-Bus basiert.“[20] Abbildung 2 zeigt den Zusammenhang der Bestandteile einer SOA nach obiger Definition.

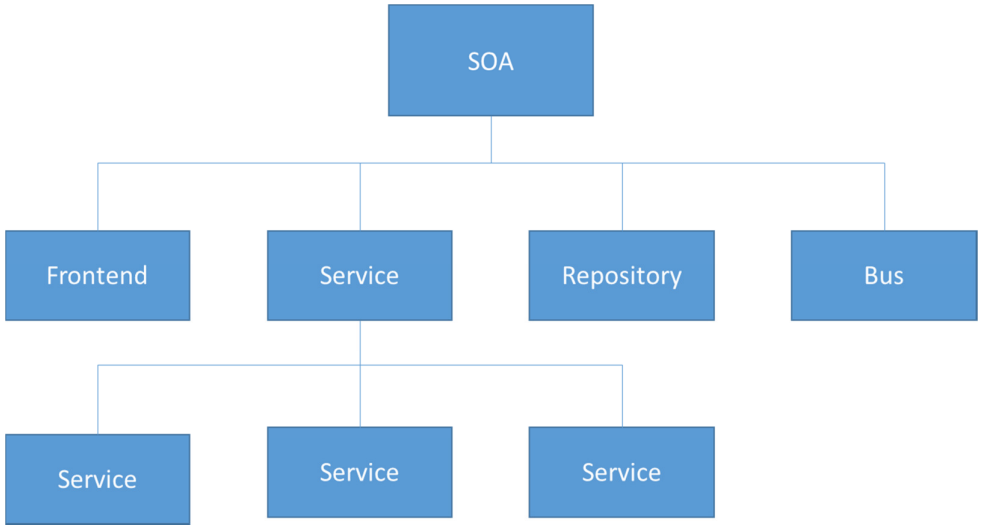


Abbildung 2: Bestandteile einer SOA[21]

Das Anwendungs-Frontend ist analog zur Präsentationsschicht zu verstehen. Es übernimmt die Darstellung von Nachrichten und dient daher der Interaktion zwischen Anwender und einem Service. Letztere sind in einer SOA zentral und werden von zwei Parteien genutzt, dem Anbieter (Provider) und dem Nutzer (Consumer). Beide schließen einen Vertrag über den Service ab, der die Leistung detailliert beschreibt (Funktionen, Beschränkung, etc.). Die Implementierung eines Service umfasst die Geschäftslogik (analog Anwendungslogik) und

[19] Vgl. Brothuhn, R. / Chantelau, K. (2010), S. 38
[20] Banke, K. / Kafzig, D. / Slama, D. (2007), S. 84
[21] Abbildung mit Änderungen übernommen aus Finger, P. / Zeppenfeld, K. (2009), S. 9

die Datenspeicherung (analog Persistenzschicht). In einer SOA wird, damit eine übersichtliche Architektur gewährleistet werden kann, ein Service vier unterschiedlichen Schichten zugeordnet:

- ***Enterprise Layer*** (Geschäftsdienste): Nach außen verfügbare Services, die zugleich die Interaktion mit den Anwender definieren.
- ***Process Layer*** (Prozesszentrierter Dienst): Geschäftslogik bestehend aus mehreren Basis- bzw. Zwischendiensten
- ***Intermediate Layer*** (Zwischendienste): Vermittlung zwischen Diensten (als *Technology Gateway, Adapter, Façade, Functionality Service*)
- ***Basic Layer*** (Basisdienst): elementar (wiederverwendbar) Funktion (daten- oder logikzentriert)

Schnittstellen ermöglichen es, Operationen eines Service nach außen anzubieten. Das Service-Repository dient zur Erkennung von anderen Services und Anforderung von Informationen zur Nutzung ebendieser. Der Service-Bus ist die zentrale Plattform einer SOA. Durch ihn sind alle Bestandteile verbunden. Die zentralen Funktionen eines Service-Bus sind die Nachrichtenübermittlung, die Datentransformation (zur Gewährleistung der technischen Unabhängigkeit) und das Routing der Nachrichten. Das grundlegende Prinzip wird in Abbildung 3 veranschaulicht.[22]

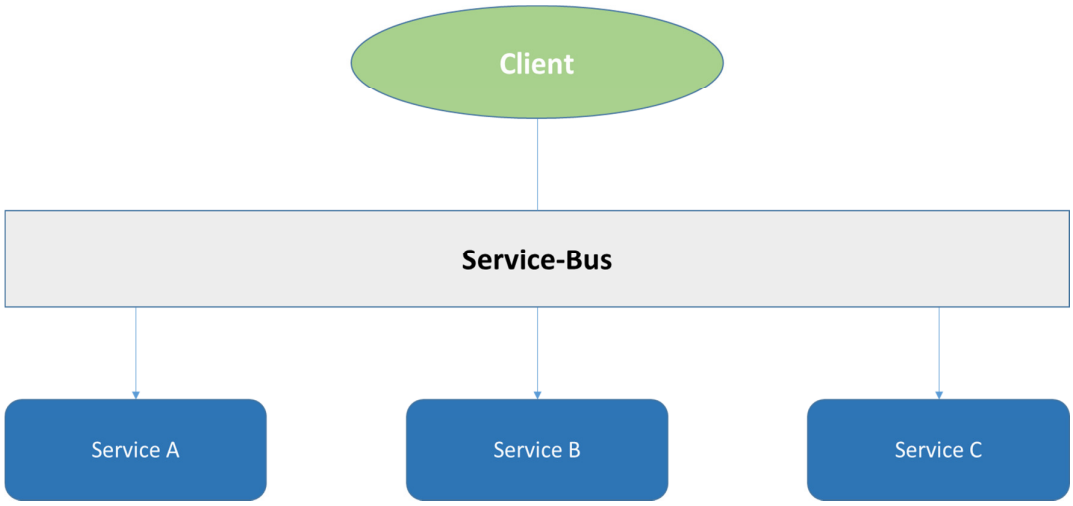


Abbildung 3: Service-.Bus einer SOA[23]

[22] Vgl. Finger, P. / Zeppenfeld, K. (2009), S. 9 ff.
[23] Abbildung mit Änderungen übernommen aus Finger, P. / Zeppenfeld, K. (2009), S. 26

Die Vorteile einer SOA sind vor allem durch deren grundlegendes Konzept gegeben. Die lose Kopplung der Services erhöht die Flexibilität der Architektur und lässt sich dadurch leicht ändern. Durch die oben genannten vier Schichten wird eine Redundanz vermieden. Im Idealfall ist also keine Funktion mehrfach implementiert, was die Wartung einer Anwendung erleichtert. Zudem können durch Zwischendienste bestehende Systeme, auch unternehmensfremde, eingebunden werden. Eigene Services können zudem kostenpflichtig angeboten werden. Der Service-Bus ermöglicht eine Skalierung der Anwendung, da neue Services bei Bedarf flexibel eingebunden werden können.

Die Umsetzung einer SOA erfolgt in der Praxis häufig mit Web-Services, da diese von Natur aus den Grundgedanken einer losen Kopplung unterstützen. Dabei sind zentrale Bestandteile einer SOA durch die in Webservice verwendeten Standards abgedeckt. Der Service-Vertrag beispielsweise ist über WSDL beschrieben. Auch das Service-Repository findet sich in UDDI wieder.[24]

Zur Modellierung der Interaktionen von Benutzer und Anwendungen auf der Präsentationsschicht wird häufig das Konzept von *Modell-View-Controller* (**MVC**) verwendet. MVC sieht in seiner ursprünglichen Auslegung eine strikte Trennung von Benutzerinteraktion, darauf basierende Datenveränderungen und deren Darstellung vor. Die Umsetzung dessen lässt sich anhand der drei Komponenten beschreiben. Der *Controller* ist für die Verarbeitung der Eingaben und deren Kommunikation an das Modell zuständig. Das *Modell* gibt die grundlegende Datenstruktur wieder und kann vom Controller befragt und modifiziert werden. Die eigentliche Darstellung wird im *View* beschrieben, der sich auf Grundlage des Modells anpasst. Abbildung 4 veranschaulicht den Zusammenhang der einzelnen Komponenten. Daraus wird ersichtlich, dass die Beziehung von View und Controller klar definiert ist, während hingegen ein Modell von beliebig vielen View-/Controller-Paaren betreut werden können. Dadurch ist es möglich, dasselbe Modell mit unterschiedlichen Schichten zu bedienen. Durch die von MVC vorgeschlagene Trennung wird ein zentraler Aspekt objektorientier Programmierung umgesetzt, nämlich das betroffene Objekte sich nicht zwangsläufig gegenseitig kennen müssen.[25]

[24] Vgl. Zeppenfeld, K. / Finger, P. (2009), S. 7 ff.
[25] Vgl. Lahres, B. / Rayman, G. (2009)

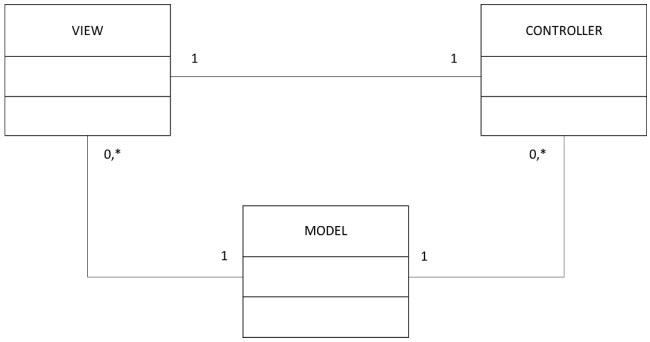


Abbildung 4: Beziehung zwischen Model, View und Controller[26]

2.2 Struktur von Anfragen

In Abbildung 1 wurde ein generischer Aufbau einer Webanwendung gezeigt. In diesem Kapitel wird auf die eigentliche Kommunikation zwischen Client und Server näher eingegangen, nachdem im vergangenen Kapitel der Fokus auf der Anwendung selbst lag.

Für die Kommunikation im WWW ist das TCP/IP-Referenzmodell maßgebend. Es umfasst die im Internet gängigen Kommunikationsprotokolle, die feste Regeln für die Kommunikation vorgeben. Dabei modellierst es einzelne Schichten der Kommunikation. Das TCP/IP-Referenzmodell besteht aus vier Schichten, die jeweils die Aufgaben und Funktionalität des gleichen Abstraktionslevels bündeln.[27]

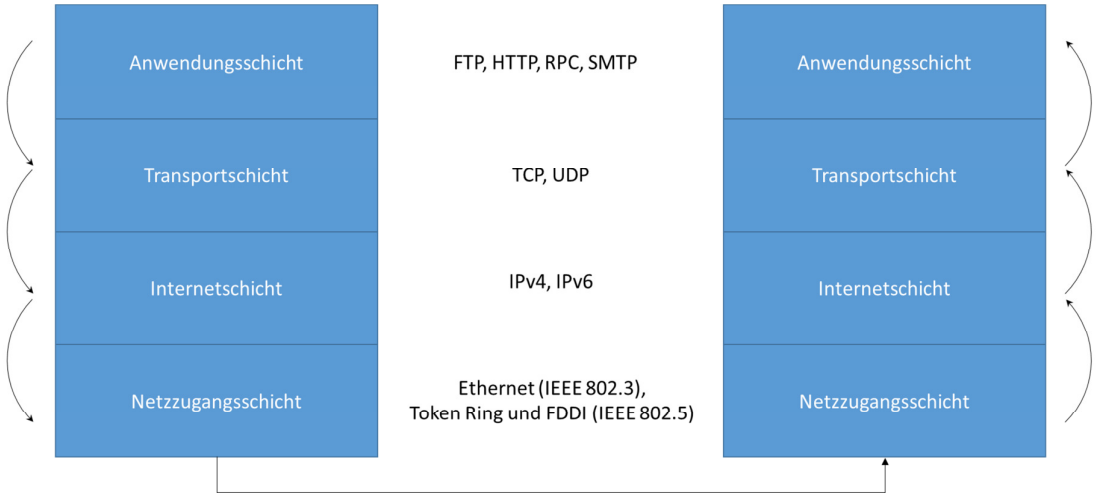


Abbildung 5: TCP/IP Modell

[26] Abbildung mit Änderungen übernommen aus Lahres, B. / Rayman, G. (2009)
[27] Vgl. ausführlich Meinel, C. / Sack, H. (2012), S. 31 ff.

Für das Thema der asynchronen Behandlung von Anfragen ist das Verständnis der Struktur ebendieser wichtig. Dafür ist die vierte Schicht des TCP/IP-Referenzmodells, die Anwendungsschicht, von Bedeutung. Zur Datenübertragung im WWW wird das Hypertext Transfer Protocol (HTTP) verwendet, welches nach dem Client-/Server-Kommunikationsprinzip funktioniert. Dieses beschreibt einen aktiven Client, der einen passiv wartenden Server kontaktiert und eine Anfrage (Request) stellt. Der Server nimmt diese Anfrage entgegen, verarbeitet sie und sendet dem Client eine Antwort (Response).[28] So wird beim Aufrufen einer Webseite mittels Uniform Resource Locater (URL) über den Browser eine Anfrage an einen Webserver gesendet. Der Informationssuchende Browser muss auf die angeforderte Ressource berechtigt und diese auf dem Server verfügbar sein. Wenn dies gegeben ist, sendet der Server die aufgeforderte Ressource (in diesem Fall ein Hypermediadokument) und einen positiven Statuscode zurück. Falls es sich bei der Anfrage um zum Beispiel spezifische Informationen für einen Account (Warenkorb, Einstellungen, etc.) handelt, werden diese über eine Session verifiziert. Cookies dienen dabei, serverseitig einen Browser (Nutzer) zu erkennen, in dem diese der Anfrage des Browsers hinzugefügt wird. Dies dienst beispielsweise dazu, personifizierten Inhalt (Werbung) einzublenden.

Die vom Server gesendeten Statuscodes sind folgendermaßen definiert.[29]

- 1xx → Informative Rückmeldung
- 2xx → Erfolgreiche Operation
- 3xx → Weiterleitung
- 4xx → Client Error
- 5xx → Server Error

In der Praxis läuft die Interaktion zwischen Browser und Server oft komplizierter, da verschiedene Zwischensysteme eingebunden sein können. Dabei handelt es sich um Proxy-Server, Gateways und Tunnels. Dies ist für das Verständnis der weiteren Arbeit nicht nötig und es wird daher auf weiterführende Literatur verwiesen.[30]

Eine HTTP Nachricht lässt sich in zwei Kategorien einteilen; die Anfrage (Request) und die Antwort (Response). Beide Typen bestehen aus einem Header und können einen Body

[28] Vgl. Meinel, C. / Sack, H. (2012), S. 61 f.
[29] Vgl. dazu ausführlich Meinel, C. / Sack, H. (2004), S. 750
[30] Vgl. Ebenda, S. 736 ff.

haben. Folgende Befehle eines Requests sind Bestandteil von HTTP 1.1 nach dem W3Consortium.[31]

- OPTIONS
- ***GET***
- HEAD
- ***POST***
- ***PUT***
- ***DELETE***
- TRACE
- CONNECT

Für das Anfordern einer Ressource wird in der Regel der GET-Operator verwendet. Dieser Operator gilt als idempotent, d.h. er verändert den Status des Servers bei beliebig häufiger Ausführung nicht. Zum Ändern oder Erstellen einer Ressource wird der PUT-Operator, zum Löschen der DELETE-Operator verwendet. Der POST-Operator funktioniert ähnlich unddient primär zur Übermittlung von Formulardaten an ein Programm. Die restlichen Operatoren sind von geringerer Bedeutung für die Arbeit und werden nicht genauer erläutert.

Eine Antwort vom Server (Response) enthält, wenn der Request erfolgreich ausgeführt worden konnte, die angeforderte Information als Body der Nachricht.

Der Nutzer eines Browser wird in der Regel nie mit HTTP in Verbindung kommen, da der Browser die notwendigen Befehle ermittelt. Die Abfolge protokollarischer HTTP-Befehle bleibt dem Nutzer dabei vollständig verborgen.[32]

2.3 **Probleme bei der Abarbeitung von Anfragen**

Ein Webserver unterliegt stetig schwankenden Auslastungen. Je nach Nutzergruppe und der Art der Website können sogenannte *Peaks* entstehen. Zu diesen Zeiten versuchen mehr Nutzer als sonst auf den Webserver zuzugreifen. Wenn dieser nicht die nötigen Ressourcen aufweisen kann, führt eine solche Situation zu beispielsweise Verbindungsabbrüchen beim anfragenden Nutzer.

[31] Vgl. dazu ausführlich http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5
[32] Vgl. Meinel, C. / Sack, H. (2012), S. 736

Der Fokus dieser Arbeit liegt jedoch nicht auf der reinen Bewältigung von vielen, gleichartigen Anfragen. Je nach Art der angebotenen Webanwendung können ungleiche Anfragen auftreten. Meist korreliert dies mit dem Angebot einer Webanwendung und derer Verknüpfung mit weiteren, teils unternehmensfremden Systemen. Dabei entstehen unter anderem kurze Anfragen, die der Webserver schnell abarbeiten kann. Somit können viele kurze Anfragen in relativ kurzer Zeit bewältigt werden. Das Gegenstück hierzu sind lange Anfragen. Zu diesen kommt es, wenn das Bearbeiten einer Anfrage auf die Antwort eines anderen Systems angewiesen ist. Die dadurch entstehende Wartezeit führt dazu, dass der Webserver länger mit einer Anfrage beschäftigt ist. Das generelle Unterscheidungsmerkmal ist somit die Dauer, die ein Webserver benötigt, eine Anfrage vollständig zu bearbeiten. Diese Differenzierung kann weiter verfeinert werden, jedoch ist die Unterscheidung in kurze und lange Anfragen für den Zweck dieser Arbeit ausreichend.

Die Problematik, die sich aus kurzen und langen Anfragen ergibt, liegt in der Natur von letzteren. Durch die Wartezeit bei der Bearbeitung ist der Webserver belegt. In dieser Zeit kann dieser also keine weitere Anfrage entgegennehmen und bearbeiten. Unabhängig von der Rechenleistung eines Webservers können also relativ wenige lange Anfragen dazu führen, dass dieser nur noch sehr langsam Anfragen entgegennehmen kann oder im schlimmsten Fall blockiert wird. So werden wenige lange Anfragen bedient, während viele kurze Anfragen auf eine Bearbeitung warten. Somit wurde eine Situation kreiert, die sich ähnlich verhält, wie ein zu ausgelasteter, ressourcenarmer Webserver bei Stoßzeiten. Der Unterschied ist hier nur, dass die Ressourcen verfügbar sind, aber durch lange Bearbeitungszeiten belegt werden. In der IT wird die Problematik der *Resource Starvation* thematisiert, bei der ein Prozess in einer Multitasking Umgebung dauerhaft nicht die nötigen Ressourcen zugeteilt bekommt. Folglich kann der Prozess nie beendet werden. Dies entsteht in der Regel bei einer fälschlichen Priorisierung, die einer Art von Prozessen eine höhere Bedeutung zuordnet.[33] Eine solche Priorisierung ist im oben genannten Szenario nicht gegeben. Jedoch ist der Grundgedanke valide. Die wenigen langen Anfragen lassen die vielen kurzen Anfragen „verhungern“, in dem die nötigen Ressourcen zur Bearbeitung blockiert werden.

[33] http://web.eecs.umich.edu/~aprakash/482/notes/deadlocks.pdf

2.4 **Java**

Java ist eine von Sun Microsystems, heute Oracle, entwickelte objektorientierte Programmiersprache. Sie gehört mittlerweile zu den populärsten Programmiersprachen unserer Zeit.[34] Dies wird vor allem durch die unzählige Literatur zu Java verdeutlicht.[35] Auch im Bildungsbereich wird Java gerne als erste zu erlernende Programmiersprache für Studenten und Schüler aufgrund ihres relativ kleinen Befehlsumfang gewählt.

Diese Popularität ist durch die Eigenschaften von Java begründet. Dabei spielt die klare objektorientierte Struktur eine eher unter geordnete Rolle. Java zeichnet sich vor allem durch seine Plattformunabhängigkeit aus. Das bedeutet, dass in Java geschriebene Programme unabhängig von Betriebssystem und Prozessor des ausführenden Rechners laufen können. Der Ausdruck „Write once – run anywhere“ ist daher zum Slogan von Java geworden.

Die Realisierung der Plattformunabhängigkeit ist in der Funktionsweise gegeben. Abbildung 6 veranschaulicht diese. Dabei wird der in Java geschriebene Quellcode durch den Compiler in einen maschinenverständlichen Bytecode übersetzt. Als Maschinensprache sind hierbei elementare Anweisungen zu verstehen, die direkt von einem Prozessor ausgeführt werden können. Demzufolge ist der Java-Bytecode unabhängig von der Rechnerarchitektur für jeglichen Prozessor bzw. Rechner verständlich. Einzige Voraussetzung hierfür ist das Vorhandensein der Laufzeitumgebung Java Virtual Machine (JVM). Diese stellt das softwaretechnische Umfeld dar, in dem die Ausführung eines Programms sich vollzieht.[36] Es äußert sich in einer abstrakten (virtuellen) Schicht zwischen den auszuführenden Bytecode und der ausführenden Einheit. Dabei überwacht die JVM die Ausführung und kann hierbei Maßnahmen zur Leistungssteigerung oder zum Schutz des Systems einleiten. Aus dieser Gegebenheit ergibt sich auch ein Nachteil von Java. Durch die Übersetzung von Quellcode zu Bytecode gibt es Leistungseinbußen gegenüber maschinennahen Sprachen wie C. Dies ist aber bei den meisten Anwendungen vernachlässigbar.

---

[34] Vgl. Tiobe Index für Januar 2014, *http://www.tiobe.com/index.php/content/paperinfo//tpci/index.html* und RedMonk Programming Language Raking Juni 2013, *http://redmonk.com/sogrady/2013/07/25/language-rankings-6-13/*

[35] Vgl. diesbezüglich ausführlich Abts, D. (2010), Block, M. (2010), Deininger, M. / Faust, G. / Kessel, T. (2009) und Ullenboom, C. (2011)

[36] Vgl. Deck, K. / Neuendorf, H. (2007), S. 14 ff.

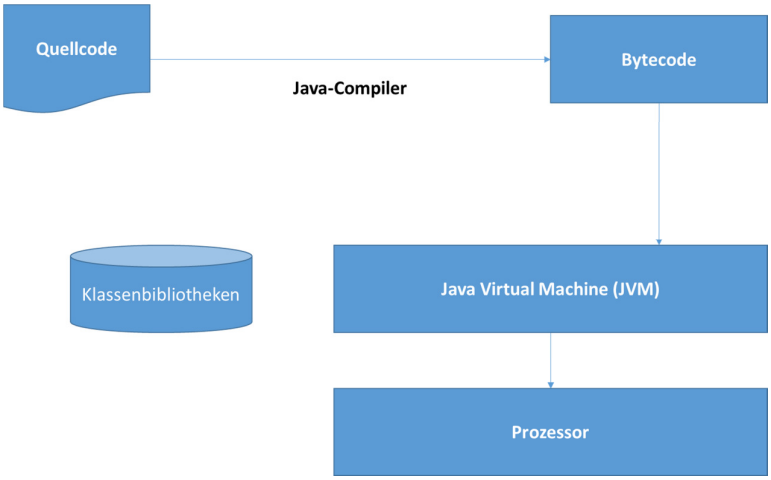


Abbildung 6: Ausführung unter Java[37]

Die Einfachheit der Sprache ist neben dem übersichtlichen Sprachumfang auch durch reine Objektorientierung gegeben. Als Grundlage für Java wird häufig C++ angesehen, welche die Weiterentwicklung um objektorientierte Anteile der prozeduralen Sprache C ist. Java wurde von Beginn an rein objektorientiert entworfen. Auch viele systemnahe Anforderungen wie die Speicherverwaltung sind dem Entwickler bei Java bewusst abgenommen. Diese werden von der Laufzeitumgebung automatisch erfasst (z.B. Garbage Collection).[38]

Für die Entwicklung von Java-Programmen ist eine Entwicklungsumgebung nötig, die die JVM enthält. Auf der Homepage von Java wird das Java Development Kit (JDK) angeboten.[39] Dieses Softwarepaket enthält für die Entwicklung nötige Tools (Compiler, Java Run-Time Environment). Ein gesonderter Editor wird nicht gegeben. Für die Entwicklung wird häufig eine integrierte Entwicklungsumgebung (Integrated Development Environment, IDE) genutzt, die neben den nötigen Tools auch einen Editor bieten. Die zwei bekanntesten IDE sind das von der Eclipse Foundation geschaffene quelloffene *Eclipse*[40] und das von Oracle stammende Open Source Projekt *NetBeans*.[41]

Für Unternehmen ist Java besonders attraktiv. Durch die Plattformunabhängigkeit ergibt sich ein sehr breites Anwendungsspektrum von in Java geschriebenen Anwendungen. Vor allem im Zusammenhang mit dem Internet kann Java von seinen Stärken Gebrauch machen. Anwendungsarten umfassen Webanwendungen, Applets und Apps. Erstere können vollständig oder teilweise (nur Serverteil) in Java geschrieben werden. Ein bekanntes

[37] Abbildung mit Änderungen übernommen aus Abts, D. (2010), S. 4
[38] Vgl. Deck, K. / Neuendorf, H. (2007), S. 15
[39] *http://www.oracle.com/technetwork/java/javase/downloads/index.html*
[40] *http://www.eclipse.org/*
[41] *https://netbeans.org/*

Beispiel ist die Plattform Twitter, die vollständig in Java geschrieben ist.[42] Applets sind in HTML-Seiten referenzierte Java-Programme, die in einem Browser ausgeführt werden. Im Zuge des Trends mobiler Endgeräte ist die Entwicklung von Apps stark angestiegen.[43] Das mobile Betriebssystem Android ermöglicht die Entwicklung von Java Anwendungen.

2.5 **Prozesse, Threads und Multithreading**

Bevor die Eigenschaften und Beispiele synchroner und asynchroner Servertechnologien ausgeführt werden, müssen zunächst einige grundlegende Begriffe geklärt werden. Was sind Prozesse und Threads? Was bedeutet Single- und Multithreading und was wird unter ereignisgesteuerter Entwicklung verstanden?

Ein Programm ist als formale Beschreibung von Maschinenbefehlen in Form eines Textdokuments zunächst statisch, das heißt zeit- und zustandsunabhängig. Ein Prozess wird vom Betriebssystem dann angelegt, wenn ein Programm ausgeführt werden soll. So findet die Ausführung einer Instanz eines Webservers in einem eigenen Prozess statt, der über eine Prozesskennung eindeutig identifizierbar ist und der über einen dezidierten Bereich im Arbeitsspeicher zur Speicherung von jeweils aktuellen Variableninhalten verfügt. Ein Prozess ist also zeitabhängig mit definiertem Anfang und Ende und kann verschiedene Zustände durchlaufen. Threads – oder Leichtgewichtprozesse – sind schließlich Teilprozesse. Sie repräsentieren nicht die Ausführung eines komplett eigenständigen Programms, sondern einer Teilfunktion des Programms.[44]

Threads sind notwendig zur Umsetzung des Konzepts der nebenläufigen Verarbeitung (Nebenläufigkeit; von engl.: concurrency) innerhalb eines Prozesses. Im Gegensatz zu Singlethreading bedeutet (softwareseitiges) Multithreading, dass in einer Anwendung sich mehrere Threads oder Unterprozesse gleichzeitig in der Abarbeitung befinden können. Im Rahmen dieser Arbeit wird dieses Konzept nur unter dem Gesichtspunkt der Nebenläufigkeit betrachtet, bei der die Abarbeitung durch die Zuteilung der Rechenleistung einer CPU (Central Processing Unit) stückchenweise nacheinander erfolgt[45], und wird getrennt von der

[42] Vgl. *http://www.oraclejavamagazine-digital.com/javamagazine/july_august_2013?sub_id=D1tTW2hUQ7Hsk&folio=18#pg19*

[43] Vgl. *http://de.statista.com/statistik/daten/studie/74368/umfrage/anzahl-der-verfuegbaren-apps-im-google-play-store/*

[44] Vgl. Weigend, M. (2010), S. 538f.

[45] Vgl. Lewis, B. / Berg, D. J. (2000), S. 12ff.

programmatischen Umsetzung des Parallelismus, wo mehrere Threads auf verschiedenen Prozessoren tatsächlich gleichzeitig abgearbeitet werden.[46]

Bei der softwareseitigen Umsetzung von Multithreading können sehr leicht Speicherinkonsistenzen und Deadlocks auftreten, da verschiedene Threads zur gleichen Zeit auf Ressourcen zugreifen. Dies kann beispielsweise durch Sperren (engl.: spinlocks) verhindert werden, wodurch eine **Synchronisation** der Threads erreicht wird.[47]

Ein Ansatz, nebenläufige Programme zu erstellen, ohne inkonsistente Zustände zu provozieren, ist die sogenannte ereignisgesteuerte Entwicklung oder Programmierung (engl.: event-driven programming). Ereignisgesteuerte Entwicklung bietet sich besonders bei langlaufenden Prozessen an, innerhalb derer viele kürzere Teilprozesse, teilweise auch nebenläufig, abgearbeitet werden müssen. Dies trifft beispielsweise auf grafische Benutzeroberflächen zu, die ständige angezeigt werden und für verschiedene Benutzereingaben verfügbar sein müssen, oder eben auch auf Webserver, die ständig zur Abarbeitung auch gleichzeitiger Requests verfügbar sein müssen.[48]

Eine sogenannte Ereignisschleife läuft als Thread innerhalb des Hauptprozesses dauerhaft. Ihre Aufgabe ist es, ununterbrochen auf neue Ereignisse zu testen, etwa einen HTTP-Request, und zur Verarbeitung dieses Events die entsprechende Callbackfunktion (Eventhandler) aufzurufen, die daraufhin in einem nebenläufigen Thread abgearbeitet wird, sodass die Eventschleife für neue Ereignisse verfügbar bleibt. Nach Abarbeitung des Requests wird der Callback wiederum im Thread der Ereignisschleife bearbeitet, sodass immer nur ein Callback zur gleichen Zeit verarbeitet werden kann, was Inkonsistenzen vermeidet.[49]

[46] Breshears, C. (2009), S. 3
[47] Vgl. ibidem, S. 10
[48] Vgl. Welch, B. B. (2003), S.227
[49] Vgl. ibidem, S. 228

3 Synchrone Frameworks (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

Bevor im Folgenden die Performance synchroner und asynchroner Frameworks empirisch verglichen werden kann, werden zunächst einige Grundlagen beider Herangehensweisen untersucht. Dieser Abschnitt soll einen Überblick über die Grundlagen synchroner Frameworks und einen kurzen Einblick in deren technischen Hintergrund geben. Außerdem werden Probleme und Einschränkungen synchroner Frameworks aufgezeigt und diskutiert.

3.1 Synchronität

Synchronität beschreibt eine mögliche Art, wie Threads anstehende Aktionen durchführen können. Dabei ist ein Thread für die Ausführung einer Anfrage zuständig. Das impliziert ein Besetzen des ausführenden Threads über die Lebensdauer der Anfrage, das heißt vom Eingang der Anfrage bis zum Abschluss der Bearbeitung.

Abbildung 7 zeigt schematisch eine synchrone Vorgehensweise. Hierbei ist ein Thread für die Abarbeitung einer beliebigen Anfrage zuständig. Zur Bearbeitung der Anfrage ist es nötig, Daten aus einem anderen System einzufordern. In dieser Zeit benötigt der Thread keine Rechenleistung, jedoch wartet dieser auf den Eingang der nötigen Daten. Daher ist es bei einer synchronen Verarbeitung nicht möglich, den Thread für weitere Anfragen freizugeben.

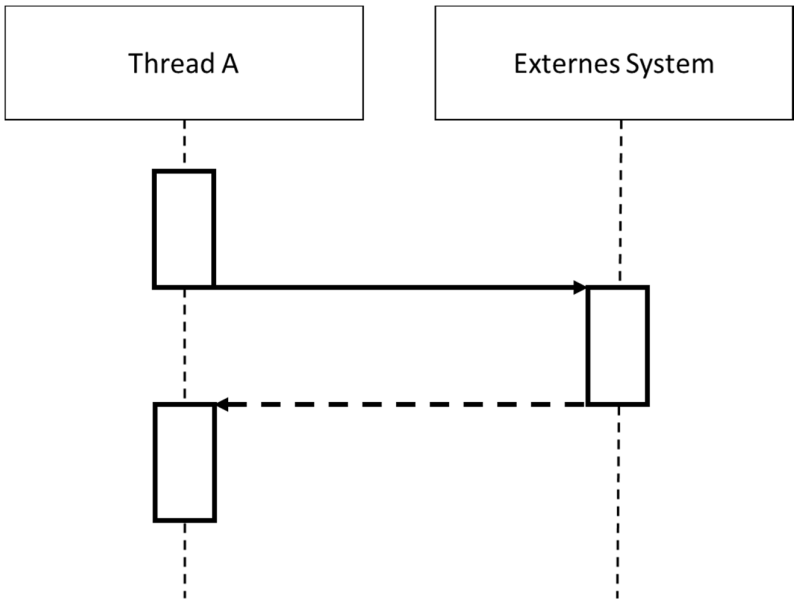


Abbildung 7: Synchrone Verarbeitung

3.2 **Technischer Hintergrund – Spring MVC**

Eine synchrone Herangehensweise kann beispielsweise mittels dem Spring Framework[50] umgesetzt werden. Der in der Beispielimplementierung entwickelte synchrone Server basiert auf diesem Framework. Das Spring Framework zeichnet sich vor allem durch sein Konzept der Invertierungskontrolle (engl. *Inversion of Controll* – IoC) bzw. der Injektion von Abhängigkeiten (engl. *Dependency Injection*) aus. Hierbei ist es von zentraler Bedeutung, dass Objekte ihre Abhängigkeiten zu anderen Objekten durch Argumente bei der jeweiligen Instanziierung setzen. Der *IoC container* injiziert dabei die festgesetzten Abhängigkeiten beim Erstellen der Objekte, der sogenannten *beans*. Diese legen den Grundstock der Anwendung. Der IoC container instanziiert und verwaltet diese Objekte. Der gesamte Prozess der Injektion der Abhängigkeiten ist dabei spiegelbildlich zu der normalen Herangehensweise, bei der das Objekt selbst seine Instanziierung und die damit verbundenen Abhängigkeiten bestimmt.[51]

Springs Web-Modul zum Erstellen von Webanwendungen besteht aus den Modulen WebSocket, Web-Servlet und Web-Portlet.[52] Für den technischen Hintergrund bei der Realisierung eines synchronen Ansatzes ist das Web-Servlet von großer Bedeutung. Dieses basiert auf dem MVC-Konzept (vgl. dazu auch Kapitl 2.1.2):[53]

1) ***Model(s)*** sind Domain Objekte, die entweder auf die Geschäftslogik (Datenverarbeitung) oder die Datenspeicherung (Persistenzschicht) bezogen sind.

2) ***Views*** sind in der Regel als JavaServer Pages (JSP) implementiert.

3) ***Controllers*** definieren die Interaktion mit den Modellen auf der Basis der Geschäftslogik

Das Spring Web-Framework funktioniert dabei in ähnlicher Weise wie gängige Vertreter: Es ist um ein zentrales Servlet entworfen, dessen Funktionsweise von Anfragen gesteuert wird. Spring definiert dafür das *DispatcherServlet*, welches zusätzlich mit dem Spring IoC container verknüpft ist und daher von dessen Funktionalität (z.B. Dependency Injection) Gebrauch machen kann. Abbildung 8 veranschaulicht die Funktionsweise des Servlets. Dieses ist dabei Bestandteil des *Front Controller*, der eine Erweiterung des MVC Konzepts um einen zentralen Einstiegspunkt in eine Webanwendung darstellt.

---

[50] *http://projects.spring.io/spring-framework/*
[51] Vgl. Johnson, R. / Hoeller, J. / Donald, K. et. Al. (2013)
[52] Vgl. Ebenda (2013)
[53] *http://www.mkyong.com/spring-mvc/spring-mvc-hello-world-example/*

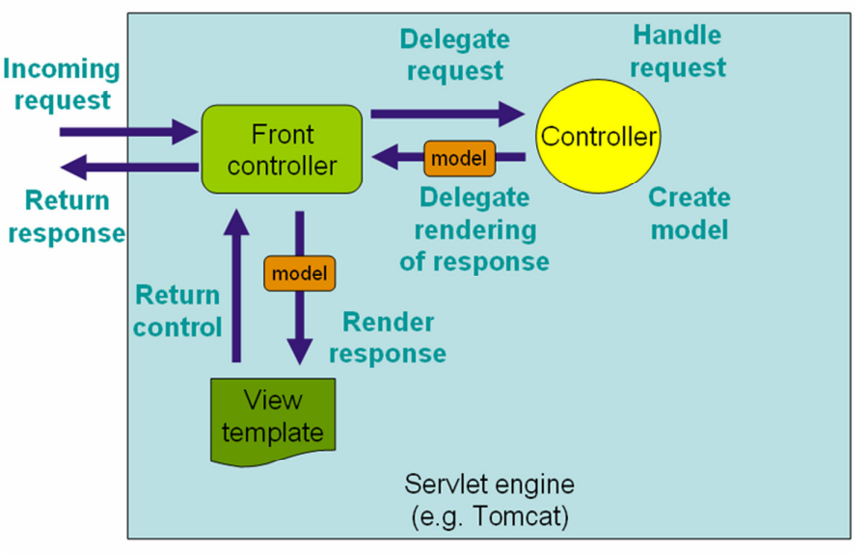


Abbildung 8: Funktionsweise Spring DispatcherServlet[54]

Hierbei wird die synchrone Arbeitsweise deutlich. Der Front Controller (damit einhergehend das DispatcherServlet) fungiert als Vermittlungsknoten (engl. *Router*) zwischen Controller und der View. Der Controller agiert dabei als Schnittstelle zu den zu Grunde liegenden Modellen. Das Generieren der Antwort auf der Präsentationsschicht wird über den Front Controller geleitet. Dabei wird es deutlich, dass das DispatcherServlet stetig aktiv an der Abarbeitung einer Anfrage beteiligt ist. Während des gesamten Prozess nimmt es also keine weiteren Anfragen entgegen. Abbildung 9 verdeutlicht die Aufgabe des DispatcherServlets als aktiven Dirigent in der Abarbeitung einer Anfrage. Hierbei ist der Prozess im Spring Framework detaillierter beschrieben.

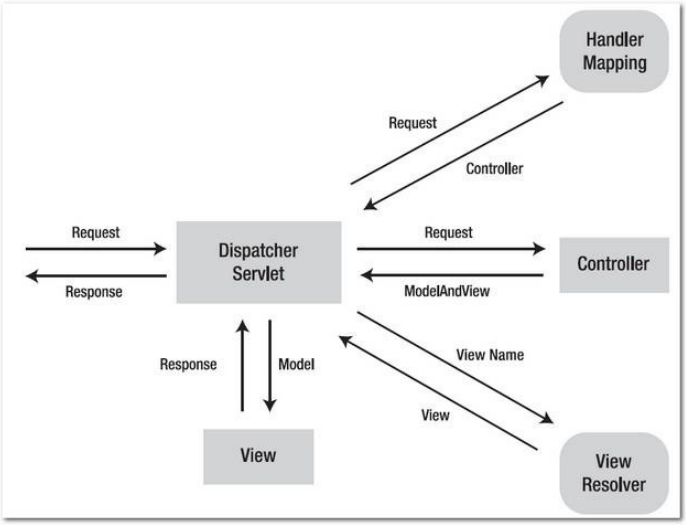


Abbildung 9: Funktionsweise Spring DispatcherServlet (2)[55]

[54] Abbildung übernommen von Johnson, R. / Hoeller, J. / Donald, K. et. Al. (2013)

3.3 **Probleme / Einschränkungen**

Insbesondere im Feld der Webprogrammierung wird die Notwendigkeit nebenläufiger Verarbeitung deutlich: Eine Vielzahl von Clients kann gleichzeitig auf einen Server zugreifen und Ressourcen anfordern. Zunächst sind synchrone Technologien eine Möglichkeit, überhaupt nebenläufige Verarbeitung durch Multithreading zu ermöglichen. Jede nebenläufige Operation in jedem Thread ändert jedoch möglicherweise Variablenwerte im Arbeitsspeicher oder kommuniziert mit anderen Threads. Es muss verhindert werden, dass Inkonsistenzen oder Deadlocks auftreten, weshalb die Synchronisation von Threads ressourcenintensiv und fehleranfällig ist und oft zu komplexem Code führt.[56]

Das Hauptproblem jedoch ist, dass auf einem Webserver nur eine begrenzte Anzahl an Threads zur Verfügung steht. So ist beispielsweise bei einem Apache Tomcat 6 die maximale Anzahl standardmäßig auf 200 begrenzt.[57] Synchrone Technologien binden jedoch jeden Request an einen eigenen Thread, welcher blockiert bleibt bis zur vollständigen Abarbeitung des Requests. Dies wird bei kurzlaufenden Anfragen, wie dem Anfordern einer Ressource auf dem Server selbst, selten zum Problem. Bei langlaufenden Requests, meistens wenn zur Abarbeitung die Kommunikation mit externen Systemen erforderlich ist, bleiben die entsprechenden Threads jedoch besetzt, obwohl keine Rechenkapazität benötigt wird, sodass die Abarbeitung weiterer Requests verhindert wird. [58] In diesem Fall steht zwar ausreichend Rechenkapazität zur Verfügung, allerdings können externe Daten, z.B. von der lokalen Festplatte oder einem Datenbankserver, nicht schnell genug herbeigeschafft werden. Der Thread wartet (blockiert).[59] Die Leistung des Webservers wird nachhaltig eingeschränkt und neu eintreffende Requests müssen abgewiesen werden.

Nun könnte man einwenden, dass diesem Problem einfach durch eine Erhöhung der Threadzahl begegnet werden kann. Jedoch belegt jeder weitere Thread Arbeitsspeicher alleine zur Verwaltung seines derzeitigen Zustands. Darüber hinaus führt eine Erhöhung der Threadzahl dazu, dass die verfügbare Rechenleistung der CPU unter allen Threads aufgeteilt werden muss, was mit einer hohen Zahl an Kontextwechseln einhergeht. Bei jedem Kontextwechsel durch das Betriebssystem müssen Speicheroperationen, also wiederum langsame Ein- und Ausgabeoperationen, durchgeführt werden, beispielsweise die Speicherung bzw. das Auslesen relevanter Register.[60]

---

[55] Abbildung von *http://www.mkyong.com/spring-mvc/spring-mvc-hello-world-example/*
[56] Vgl. Breshears, C. (2009), S. 10
[57] Vgl. o. A. (o. J.a)
[58] Vgl. Arranz, J. M. (2011)
[59] Vgl. o. A. (2003)
[60] o. A. (o. J.b), S.4

4 Asynchrone Frameworks (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

Die Stärke asynchroner (nicht blockierender) Frameworks besteht darin, die oben beschriebene „Synchronisierung“ von Request und Thread, also deren Bindung bis zur vollständigen Abarbeitung auch aufwändiger Requests, aufzulösen. Ein Request blockiert nicht länger einen Thread, die Abarbeitung erfolgt asynchron.[61] Dies wird mit Hilfe der sogenannten ereignisgesteuerten Programmierung erreicht (siehe Kapitel 2.5).

Asynchrone Verarbeitung, asynchroner Programmierstill oder ereignisgesteuerte Programmierung werden oftmals synonym gebraucht.[62] Bei der asynchronen Verarbeitung sind die einzelnen, gleichzeitigen Teilaufgaben eines Prozesses, etwa die Abarbeitung verschiedener Requests, wie beim Multithreading mit synchronen Technologien ebenfalls miteinander verzahnt. Jedoch gibt es in Form der Eventschleife nur einen einzigen Steuerungsthread, sodass ein großer Nachteil des Multithreadings mit synchronen Technologien, die komplexe Koordination der Threads untereinander, entfällt.[63]

Indem ein- oder ausgabeintensive Operationen von Eventhandlern in nebenläufigen Threads bearbeitet werden, wird die Eventschleife nicht blockiert. Sie kann weiterhin ihre Hauptfunktionen erfüllen: Die Abfrage neuer Events und der Aufruf des passenden Eventhandlers.[64]

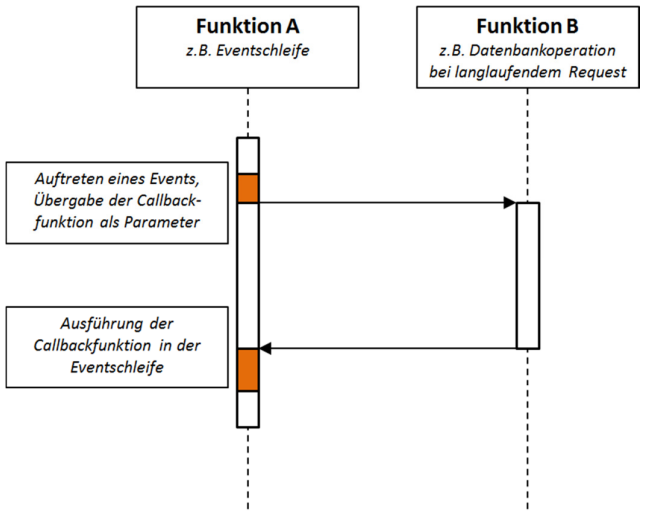


Abbildung 10: Schematischer Ablauf eines asynchronen Aufrufs

[61] Vgl. Arranz, J. M. (2011)
[62] Vgl. Teixeira, P. (2013), S.16
[63] Vgl. o. A. (o. J.b), S. 2
[64] Vgl. Teixeira, P. (2013), S.16f.

Asynchrone Frameworks können dabei helfen, viele der Einschränkungen von synchronen Frameworks zu überwinden, sind jedoch auch kein Allheilmittel für die Erstellung performanter Webserver. Dieser Abschnitt stellt einige Beispiele für asynchrone Frameworks sowie deren Vor- und Nachteile vor und zeigt Fallstricke bei der Nutzung und Anwendungsbereiche im Unternehmenskontext auf. Danach soll eine Abgrenzung zu anderen Technologien vorgenommen und ein Marktüberblick über asynchrone Frameworks gegeben werden.

4.1 **Marktbetrachtung**

Bevor verschiedene asynchrone Frameworks näher untersucht werden können, ist es wichtig, sich zunächst einmal einen Marktüberblick zu verschaffen. Die Idee asynchroner Verarbeitung auf der Serverseite ist nicht neu und es gibt eine große Anzahl entsprechender Ansätze für fast alle gängigen Programmiersprachen, umgesetzt als eigenständiges Framework oder als Erweiterung eines bestehenden Frameworks. Dazu zählen New I/O von Java, welches neue APIs (Application Programming Interfaces) unter anderem für Dateiein- und -ausgaben definiert[65], Netty, welches New I/O unterstützt[66], Node.js mit JavaScript[67], Perl Object Environment von Perl[68], libevent für das Management von Callbackfunktionen in C[69], die Reactive Extensions (Rx) für Microsoft's NET[70], die Bibliothek EventMachine zur ereignisgesteuerten Programmierung für Ruby, Twisted für Python[71], die epoll API für Linux[72] und schließlich Vert.x, welches auf der JVM läuft.[73] Die Liste gibt für die jeweiligen Sprachen populäre Frameworks und Bibliotheken an, erhebt jedoch keinen Anspruch auf Vollständigkeit.

Eine gute Möglichkeit, die derzeitige Popularität von Konzepten und Begriffen im Internet herauszufinden, ist GoogleTrends, welches die Häufigkeit von populären Anfragen und verwandten Wörtern indiziert darstellt.[74] In Abbildung 11 wird deutlich, dass das JavaScript-basierte Node.js derzeit das dominierende Framework für asynchrone Webserver zu sein scheint und die anderen marginalisiert. Für die Auswahl der fünf verglichenen Frameworks (Java NIO, Node.js, Libevent, Vert.x und Twisted) war entscheidend, dass sie

[65] Vgl. *http://docs.oracle.com/javase/1.5.0/docs/guide/nio/index.html*
[66] Vgl. *http://netty.io/*
[67] Vgl. *http://nodejs.org/*
[68] Vgl. *http://poe.perl.org/?What_POE_Is*
[69] Vgl. *http://libevent.org/*
[70] Vgl. http://blogs.msdn.com/b/pfxteam/archive/2011/09/17/10212961.aspx
[71] Vgl. *https://twistedmatrix.com/trac/*
[72] Vgl. *http://man7.org/linux/man-pages/man7/epoll.7.html*
[73] Vgl. o. A. (o. J.c)
[74] Vgl. Google, Inc. (2014)

gewöhnlicherweise nicht auf derselben Plattform entwickelt und betrieben werden und also verschiedene Konzepte darstellen. Leider lassen sich anhand der Zahlen von Google Trends nur relative, aber keine absoluten Aussagen treffen.

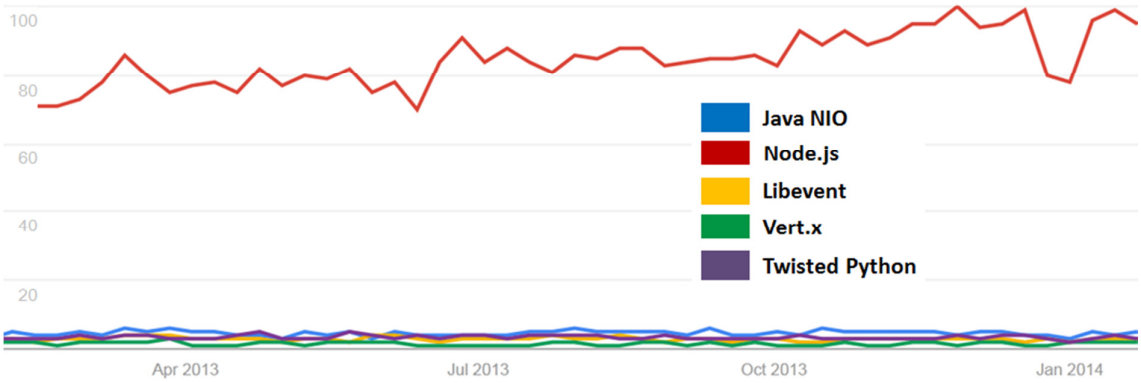


Abbildung 11: Vergleich des Google Suchvolumens asynchroner Frameworks[75]

Das Interesse für ein Konzept speziell in der Entwicklergemeinde lässt sich sehr gut visualisieren über die Häufigkeit der Konzepte bei Plattformen, die eher von Entwicklern besucht werden. Gleichzeitig lässt sich daran abschätzen, wie viele Menschen nicht nur Interesse, sondern auch tiefgreifende Fähigkeiten in den jeweiligen Technologien haben. Abbildung 12 vergleicht dazu, wie oft die jeweiligen Frameworks jemals in Beiträgen auf Stack Overflow getaggt wurden, sowie die Anzahl der GitHub Projekt-Repositories, die entsprechend getaggt wurden. Stack Overflow als Forum für Entwickler ist dabei ein guter Indikator für die aktuelle Entwicklungsaktivität, während die Anzahl der GitHub Repositories eher auf die Anzahl an fertigen Projekten und Komponenten schließen lässt. Auch auf diesen beiden Plattformen ist das Interesse für Node.js mit Abstand am größten, jedoch lassen sich auch Unterschiede zwischen den anderen Frameworks ausmachen. Dabei sind die Verhältnisse auf Stack Overflow und GitHub mit Ausnahme von Java NIO weitestgehend gleich.

[75] Datengrundlage von:
http://www.google.com/trends/explore#q=Java%20NIO%2C%20Libevent%2C%20Vert.x%2C%20Twisted%20Python%2C%20Node.js&date=today%2012-m&cmpt=q

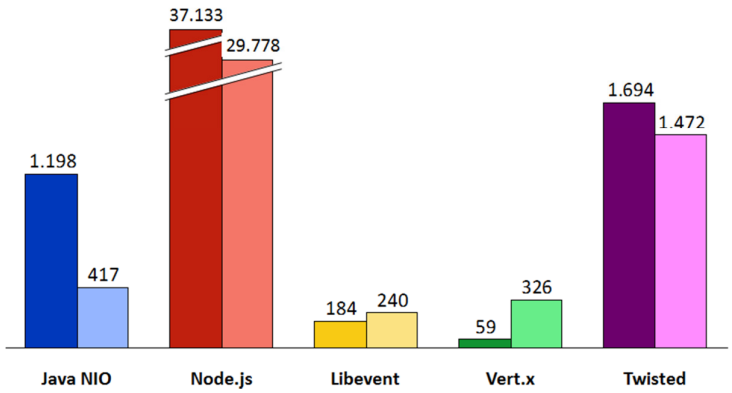


Abbildung 12: Tagzahlen von Java NIO, Node.js, Libevent, Vert.x und Twisted bei Stack Overflow (dunkel)76 und Anzahl Repositories bei Github, Stand 27.01.2014 (transparent)77

Die Frage ist, warum das Interesse für Node.js so viel höher ist und Vert.x, welches in der weiteren Arbeit als asynchrones Framework genutzt wird, derart selten gesucht und genutzt wird. Dafür gibt es mehrere mögliche Antworten, von denen drei im Folgenden erläutert werden:

Aus der Analyse von GitHub und Stack Overflow erzeugt auch der Analyst Redmonk sein Ranking der populärsten Programmiersprachen in der Entwicklergemeinde. Dieses Ranking wird schon seit über zwei Jahren von JavaScript angeführt, im Januar 2014 gefolgt von Java, PHP und C#. Die hohe Verbreitung von JavaScript könnte über Synergieeffekte zur Verbreitung von Node.js beitragen.[78]

Als Ursache könnte ebenso das spätere Auftreten von Vert.x in Betracht kommen, welches erst ab 2011 entwickelt wird. Node.js, welches bereits 2009 zuerst herausgegeben wurde[79], hat sich bis zu diesem Zeitpunkt mitunter bereits in der Entwicklergemeinde etabliert, sodass aufgrund der ähnlichen Funktionalität von Vert.x und Node.js der Bedarf für ein derartiges Framework bereits weitestgehend gedeckt war.

Zuletzt ein technischer Aspekt: Trotz einer Node.js sehr ähnlichen Architektur (siehe auch Kapitel 4.2) nutzt Vert.x zahlreiche Java-Bibliotheken, die selbst nicht nach asynchronen Prinzipien aufgebaut sind. Bei Vert.x besteht somit im Gegensatz zu Node.js, das auf einer ebenfalls nach asynchronen Prinzipien aufgebauten JavaScript-Bibliothek basiert, die Gefahr, synchronen Code auszuführen, was mitunter viele Entwickler abschreckt.[80]

---

[76] Datengrundlage von: *http://stackoverflow.com/tags*
[77] Datengrundlage von: *https://github.com/search?q=&type=Repositories&ref=searchresults*
[78] Vgl. RedMonk (2014)
[79] Vgl. *https://github.com/joyent/node/tags?after=v0.0.4*
[80] Vgl. Bineytioglu, C. P., Boden, J. A., Schulz, R. u. A. (2013), S. 7

4.2 **Einführung in Vert.x**

In der Beispielimplementierung zum Leistungsvergleich asynchroner mit synchronen Servertechnologien wird ein Vert.x Webserver genutzt (ab Kapitel 7). Deshalb liegt der Schwerpunkt dieses Kapitels auf der Vorstellung von Vert.x und dem Vergleich mit dem derzeitigen Marktführer unter den asynchronen Servertechnologien, Node.js.

Aufgrund der geringen Verbreitung von Vert.x gibt es dazu kaum Fachliteratur und nur wenige unabhängige Artikel. Deshalb muss an vielen Stellen die offizielle Dokumentation des Frameworks als primäre Quelle genutzt werden.[81]

Vert.x ist ein asynchrones, ereignisgesteuertes Framework, das auf der Java Virtual Machine (JVM) ausgeführt wird. Dadurch kann ein Vert.x Webserver mehrere Prozessorkerne nutzen, ohne dass der Benutzer manuell neue Prozesse initiieren oder sich um die Kommunikation zwischen den Prozessen kümmern muss. Zusammen mit dem asynchronen Programmierstil ermöglicht dies dem Nutzer, den Programmcode einsträngig zu schreiben, als würden keine nebenläufigen Threads genutzt.[82]

Zudem ist Vert.x polyglott, was bedeutet, dass Anwendungen in Ruby, Groovy, Python, Java, JavaScript und seit Sommer 2013 auch in Clojure und Scala geschrieben werden können.[83]

Vert.x basiert auf einem modularen System, das die Wiederverwendbarkeit von Funktionen einer Serveranwendung ermöglicht. Module können außerdem über die öffentliche Vert.x Module Registry[84] mit anderen Nutzern geteilt werden.

Abbildung 13 illustriert die Kernkonzepte von Vert.x: Das Kernelement von Vert.x sind die sogenannten Verticles, die zu Modulen gepackt werden können. Selbst Verticles innerhalb eines Moduls können in verschiedenen Sprachen geschrieben sein. Mehrere Verticles können innerhalb einer Vert.x-Instanz ausgeführt werden, die wiederum in einer eigenen JVM-Instanz ausgeführt wird. Vert.x ordnet jedes Verticle genau einer Eventschleife zu und verhindert damit, dass eine Verticle-Instanz zur gleichen Zeit in mehr als einem Thread ausgeführt wird, sodass ein Entwickler Aspekte des Multithreading nicht berücksichtigen muss. Für den Fall, dass ein Verticle ein- oder ausgabeintensiven (blockierenden) Code enthält, kann es als sogenanntes „Worker Verticle“ deklariert werden. Es ist dann keiner

[81] Vgl. o. A. (o. J.c)
[82] Vgl. ibidem
[83] Vgl. Fox, T. (2013)
[84] Vgl. *http://modulereg.vertx.io/*

Eventschleife, sondern wird einem speziellen „Workerthread“ aus einem internen Threadpool zugeordnet. Dadurch wird eine Blockade der Eventschleife effektiv verhindert.[85]

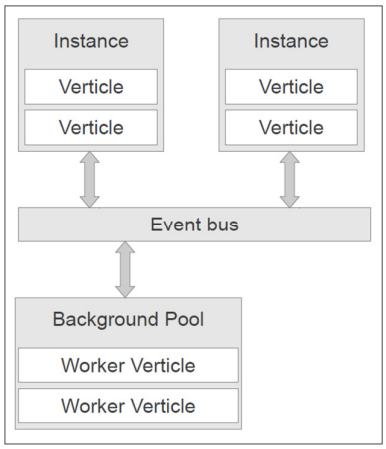


Abbildung 13: Zusammenhang grundlegender Konzepte in Vert.x[86]

Die Kommunikation zwischen den einzelnen, mitunter in verschiedenen Sprachen geschriebenen Prozessen übernimmt der Vert.x Event Bus, der Funktionen des In-Memory Datennetzes Hazelcastm nutzt.[87] Durch die Nutzung einer In-Memory-Lösung ist der Zugriff auf ein Relationales Datenbankmanagementsystem in vielen Fällen nicht nötig, was den Informationsaustausch deutlich beschleunigt. Zusammen mit der Tatsache, dass Hazelcastm ein verteiltes Datennetz ist, stellt dies ebenfalls einen großen Vorteil in einer verteilten Umgebung mit vielen verschiedenartigen Klienten dar.[88] Insgesamt verfügt eine Vert.x Anwendung über drei Threadpools:

1) Für jeden Port wird ein Akzeptorthread (engl. Acceptor) angelegt, der ein Websocket akzeptieren kann.

2) Die Eventschleifen werden in einem zweiten Threadpool verwaltet. Für jeden Prozessorkern wird eine Eventschleife angelegt, die bei einem auftretenden Event den entsprechenden Handler aufruft.

3) Die von den Eventschleifen aufgerufenen Handler (oder Worker-Verticles) werden in gesonderten Workerthreads abgearbeitet, die in einem Hintergrundpool (engl. Background) gehalten werden. Standardmäßig verfügt Vert.x über 20 Threads dieser Art.[89]

[85] Vgl. o. A. (o. J.c)
[86] Vgl. Bineytioglu, C. P., Boden, J. A., Schulz, R. u. A. (2013)
[87] Vgl. Kim, J. (2013)
[88] Vgl. ibidem
[89] Vgl. Kim, J. (2013)

Neben der asynchronen Programmierung ist eine der Grundideen von Node.js, die in der clientseitigen Programmierung dominierende Skriptsprache JavaScript auch serverseitig zu verwenden. Zur Nutzung der gleichen Sprache auf Client und Server gab es auch deutlich früher bereits Ansätze wie den Netscape Enterprise Server[90], die sich allerdings nicht auf Dauer etablieren konnten. Dies hat sich durch die Weiterentwicklung von JavaScript, insbesondere durch die Entwicklung der V8 JavaScript Engine von Google geändert, die Just-In-Time (JIT) Kompilierung von JavaScript ermöglicht und den Kern von Node.js bildet.[91]

Die Entwicklung des wie Vert.x vollständig ereignisgesteuerten Node.js begann 2009, zwei Jahre vor Vert.x, und das Projekt wird heute stark unterstützt von Joyent Inc., einem amerikanischen Anbieter von Clouddienstleistungen.[92]

Vergleichbar zu den Modulen bei Vert.x gibt es bei Node.js eine ganze Reihe vorgefertigter Erweiterungen und Anwendungen in Form von Paketen, die über den Node.js Package Manager (npm) erhältlich sind und die Entwicklung komplexer Anwendungen erleichtern.[93] Im Gegensatz zu Vert.x sind alle Node.js Module, Bibliotheken und Funktionen selbst asynchron.

Abbildung 14 ist eine gute Illustration der Funktionsweise nicht nur von Node.js, sondern von asynchronen Frameworks im Allgemeinen. Node.js läuft in nur einem Thread, der Eventschleife. Jeder eingehende HTTP-Request erreicht zunächst die Eventschleife, die sich durch Aufruf des entsprechenden Eventhandlers um die Abarbeitung in einem sogenannten Workerthread kümmert. Insbesondere bei langlaufenden Requests wird so eine Blockade der Eventschleife verhindert. Die Callbackfunktion, die bei Aufruf des Eventhandlers als Parameter übergeben wird, wird nach der Eventbehandlung in der Eventschleife ausgeführt.

[90] Vgl. *http://docs.oracle.com/cd/E19957-01/816-5648-10/es351jpg.html*
[91] Vgl. Braun, H. (2014)
[92] Vgl. Thornsby, J. (2010)
[93] Vgl. *https://npmjs.org/*

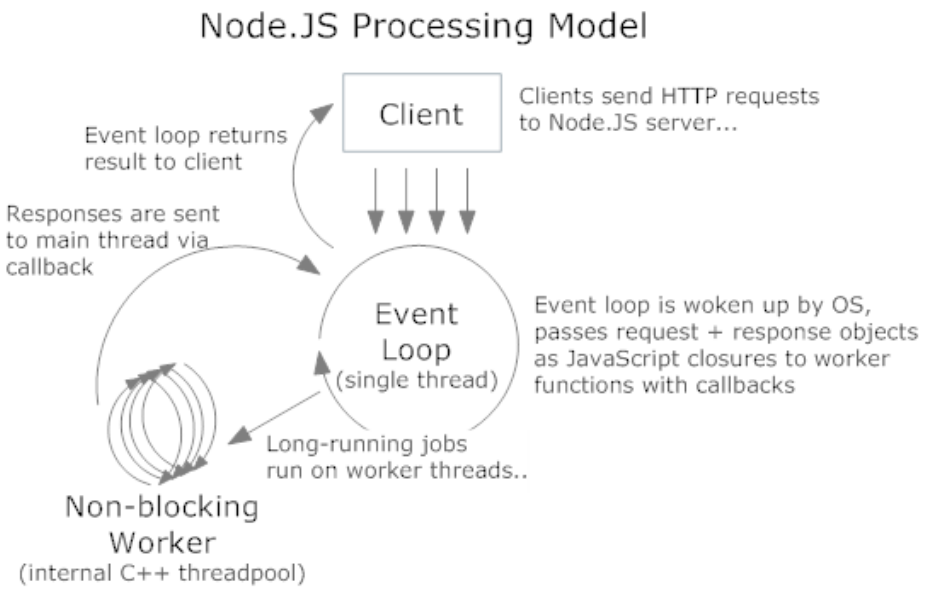


Abbildung 14: Funktionsweise von Node.js[94]

Vert.x unterscheidet sich im Wesentlichen in zwei Hauptpunkten von diesem Modell: Zum einen stehen neben JavaScript weitere Sprachen zur Verfügung, zum anderen bietet Vert.x Unterstützung von Mehrkernprozessoren, sodass mehrere Eventschleifen gleichzeitig ausgeführt werden können.

4.3 **Vor- / Nachteile asynchroner Servertechnologien**

Der Vergleich mit synchronen Technologien und der asynchronen Frameworks untereinander offenbart zahlreiche Vor- und Nachteile asynchroner Servertechnologien. Zunächst werden einige generelle Vor- und Nachteile erläutert.

Der Entwickler hat bei asynchroner Verarbeitung im Gegensatz zu synchroner Verarbeitung, wo letztlich das Betriebssystem die Ressourcenzuteilung steuert, eine größere **Kontrolle über die Threadzuteilung und –verwaltung** durch die explizite Übergabe einer neuen Aufgabe durch Aufruf des Eventhandlers und die Beendung jeder Aufgabe mit einer Callbackfunktion.[95]

Darüber hinaus gibt es bei asynchronen Anwendungen aus Entwicklersicht mit der Eventschleife nur einen einzigen Thread, während die Delegation von Aufgaben an „Hintergrundthreads“ von der jeweiligen Technologie selbst verwaltet wird. Damit entfällt die

[94] Vgl. o.A. (2011b)
[95] Vgl. o. A. (o. J.b), S.2

oftmals aufwendige **Synchronisation** verschiedener Threads zur Deadlockvermeidung und nebenläufige Threads können problemlos auf die gleichen Datenstrukturen zugreifen.[96] Dies reduziert zudem den Overhead bei hoher Anfragenzahl, da die Prozess- bzw. Threadinitialisierung durch das Betriebssystem entfällt.[97]

Eine Schwierigkeit stellt bei einigen asynchronen Frameworks jedoch der **Zugriff auf relationale Datenbanken** dar, der von synchronen Frameworks wie Ruby on Rails mit bereits integriertem objektrelationalem Mapping deutlich weiter entwickelt ist.[98] Lösungen wie der objektrelationaler Mapper Sequelize[99], der komplett in JavaScript geschrieben ist, bieten jedoch Alternativen, in diesem Fall besonders geeignet für Node.js.

Neben den generellen Vor- und Nachteilen haben die einzelnen Frameworks jeweils individuelle Stärken und Schwächen. Diese sollen hier für vert.x, das für die Beispielimplementierung verwendet werden wird, und node.js, dem derzeit am weitesten verbreiteten asynchronen Framework (vgl. Kapitel 4.1, Marktbetrachtung), aufgeführt werden.

Bei der **Dokumentation** zeigen sich Unterschiede zwischen den Frameworks, die sich größtenteils auf die stark unterschiedliche Verbreitung zurückführen lassen. Bereits die offizielle Dokumentation von Node.js ist sehr ausführlich[100], und durch die starke Beteiligung der Webentwicklergemeinde besteht darüber hinaus eine umfangreiche „sekundäre“ Dokumentation in Form von Artikeln und Foreneinträgen (vergleiche auch unterschiedliche Aktivität bei Stack Overflow, Kapitel 4.1). Die offizielle Dokumentation von Vert.x umfasst alle unterstützten Sprachen und ist ebenfalls ausführlich[101], sekundäre Informationen fehlen jedoch weitestgehend, sodass die Informationsbeschaffung deutlich schwieriger ist.

Eine **Erweiterung der Kernfunktionalität** ist bei beiden Frameworks möglich. Die Anzahl der bereits verfügbaren Erweiterungen zeigt jedoch wiederum, dass Vert.x und Node.js unterschiedlich lange bestehen und das Interesse an Node.js größer ist. Vert.x bietet auf seiner Website eine Module Registry[102], in der zum Stand Ende Januar 2014 nur 117 Module registriert waren, während das „Vert.x Central Module Repository“ auf GitHub[103] nur Module für Version 1 beinhaltet und bereits seit einiger Zeit verwaist zu sein scheint. Bei Node.js können hingegen bereits umfangreiche Erweiterungen der Kernfunktionalität genutzt werden

[96] Vgl. Braun, H. (2014)
[97] Vgl. Vgl. o.A. (2011b)
[98] Vgl. Capan, T. (o. J.)
[99] Vgl. Webauftritt: *http://sequelizejs.com/*
[100] Vgl. *http://nodejs.org/api/*
[101] Vgl. o. A. (o. J.c)
[102] Vgl. *http://modulereg.vertx.io/*
[103] Vgl. *https://github.com/vert-x/vertx-mods*

über den „Nodes Package Manager“ (npm)[104], der (Stand Ende Januar 2014) 56.884 Packages enthält.

Der **Einstieg in die serverseitige Entwicklung** wird clientseitigen Webentwicklern sowohl von Node.js, welches mit JavaScript eine clientseitige Programmiersprache auf der Serverseite unterstützt, als auch mit Vert.x erleichtert, welches neben JavaScript u.a. mit Ruby und Java noch weitere weitverbreitete Sprachen unterstützt.

Insbesondere die Javaunterstützung ermöglicht bei Vert.x zusätzlich eine unkomplizierte **Einbindung in bestehende Java-Applikationen** und die **Nutzung bestehender Javabibliotheken** und APIs, wodurch der Mangel an eigenen Vert.x-Modulen teilweise wieder ausgeglichen werden kann.

Darüber hinaus bietet Vert.x problemlose **Skalierbarkeit über mehrere Prozessorkerne** unter Nutzung mehrerer Eventschleifen, wodurch sich der Programmierer auf die (einsträngige) Entwicklung der Anwendungslogik konzentrieren kann.[105] Bei Node.js müssen zur Nutzung weiterer Kerne eigenhändig zusätzliche Instanzen erzeugt werden, was den Entwicklungsaufwand und insbesondere die Fehleranfälligkeit erhöht, da damit wieder Probleme der Synchronisation beim Multithreading auftreten können. Jedoch befindet sich auch bei Node.js die Unterstützung von Mehrkernprozessoren in der Entwicklung.[106]

Beide Frameworks, Vert.x wie auch Node.js, erlauben die **kommerzielle Nutzung**, wobei die von Vert.x genutzte Apache License in der Version 2.0[107] einige Einschränkungen in der Form von Informationspflichten auferlegt, während die von Node.js genutzte MIT License[108] keine Bedingungen stellt.

4.4 **Anwendungsbereiche im Unternehmenskontext**

Dieses Kapitel soll Anwendungsfälle asynchroner Servertechnologien im Unternehmensumfeld aufzeigen, ohne sich dabei auf nur eine Technologie festzulegen. Wegen der Dominanz wird in der Literatur oft nur explizit Node.js behandelt[109], viele dabei aufgezeigte Anwendungsbereiche lassen sich jedoch verallgemeinern.

[104] Vgl. *https://npmjs.org/*
[105] Vgl. Kim, J. (2013)
[106] Vgl. Braun, H. (2014)
[107] Vgl. *http://www.apache.org/licenses/LICENSE-2.0.html*
[108] Vgl. *http://opensource.org/licenses/MIT*
[109] Vgl. Capan, T. (o. J.)

Im Allgemeinen sind asynchrone Frameworks gut geeignet für Anwendungen, deren Engpass die Ein- und Ausgabe an externe Systeme ist. Anstatt beispielsweise auf die Rückmeldung eines Datenbankservers zu warten, kann die Eventschleife bei Anwendung asynchroner Technologien in der Zwischenzeit für die Abarbeitung kurzlaufender Requests genutzt werden und weiterhin Eventhandler aufrufen, während er bei synchroner Herangehensweise für die gesamte Dauer des Hintergrundrequests blockiert bleibt – obwohl bis zur Rückmeldung keinerlei Berechnungen nötig sind.[110]

Die ereignisgesteuerte Verarbeitung asynchroner Modelle ist häufig hilfreich bei Echtzeitanwendungen. Ein Standardbeispiel dafür sind **Chatanwendungen** oder **Instant Messaging Anwendungen**, die im Rahmen der virtuellen Zusammenarbeit auch innerhalb von Unternehmen immer wichtiger werden. Eine Chatanwendung erfüllt weitere Kriterien, die die Verwendung asynchroner Technologien nahelegen, da sie eine hohe Zahl von Requests zu verarbeiten haben, die kaum Rechenleistung benötigen, dafür aber oft größere Datenmengen beinhalten, die gelesen und ausgegeben werden müssen.[111] Eine Chatanwendung, die mit synchronen Technologien umgesetzt ist, wäre nicht mehr verfügbar, sobald alle Threads belegt sind. Beim Einsatz asynchroner Technologien bleibt die Eventschleife verfügbar, während die Nachrichten versendet werden, und kann nach Empfang einer Nachricht durch einen Client die entsprechende Callbackfunktion ausführen. Gleichzeitig entfällt bei asynchroner Verarbeitung durch die singuläre Eventschleife die aufwändige Synchronisierung der nebenläufigen Threads und Inkonsistenzen können vermieden werden.[112]

Des Weiteren können asynchrone Frameworks genutzt werden zum Aufbau eines **Systemüberwachungsdashboards**. Die Eventschleife testet dabei andauernd asynchron die Status der entsprechenden Services und nutzt Websockets, um die Daten zu den Kunden zu senden und sie über den derzeitigen Status zu informieren. Eine solche Technologie hat vielfältige Einsatzmöglichkeiten im Unternehmensumfeld: Ein Cloudanbieter im Bereich Software as a Service kann damit seine Kunden über die aktuelle Verfügbarkeit informieren, ein Unternehmen seine internen Systeme in einem Network Operations Center überwachen oder ein Börsenbetreiber den Händlern aktuelle Kurse sowie entsprechende Analysen und Diagramme zur Verfügung stellen.[113]

In Unternehmen sind viele Mitarbeiter auf Informationen angewiesen, die über interne Netzwerke in Form zahlreicher Dateien geteilt werden. Eine langsame Up- und

[110] Vgl. o. A. (2003)
[111] Vgl. Capan, T. (o. J.)
[112] Vgl. Bineytioglu, C. P., Boden, J. A., Schulz, R. u. A. (2013), S. 17
[113] Vgl. Capan, T. (o. J.)

Downloadgeschwindigkeit kann schnell zu empfindlichen Produktivitätsrückgängen führen. **Streaming von Dateien** ist ein weiterer möglicher Einsatzbereich für asynchrone Technologien, wobei es zwei grundlegende Einsatzszenarien gibt: (1) HTTP-Anfragen und –Antworten sind Streams, werden in älteren Frameworks jedoch oft wie isolierte Ereignisse behandelt.[114] Node.js ermöglicht die Ausnutzung der Stream-Eigenschaft über die Stream API[115] und ermöglicht damit die Verarbeitung von Dateien, bevor diese vollständig hochgeladen sind. Diese Eigenschaft wird bereits vom Unternehmen *Transloadit* in dessen *Realtime Encoding* Lösung genutzt, welche auf Node.js aufbaut.[116] (2) Node.js kann durch Buffering entstehende Arbeitsspeicherprobleme verhindern, insbesondere beim Dateiupload. Die Stream API von Node.js unterscheidet zwischen „readable“ (ablesbaren) und „writable“ (beschreibbaren) Streams. Ein readable Stream (Produzent) entsteht etwa durch das Auslesen einer Datei von der lokalen Festplatte, was im Vergleich zum writable Stream (Konsument), in dem die Datei über ein Netzwerk zum Client geschickt wird, oft relativ schnell funktioniert. Der kurzzeitige Datenüberschuss muss im Arbeitsspeicher gebuffert werden. Node.js verhindert das Auftreten des Buffers, indem der Produzent pausiert werden kann.[117] Dasselbe gilt natürlich beim Herunterladen einer Datei und Speicherung im lokalen Speicher, wobei meist das Netzwerk der limitierende Faktor ist und dieser Fall somit selten auftritt.

4.5 **Fallstricke bei der Nutzung asynchroner Servertechnologien**

Im Anschluss an die Behandlung einiger Szenarien, in denen sich die asynchrone Verarbeitung gut eignet, sollen hier einige „Fallstricke“ asynchroner Frameworks aufgezeigt werden. Dies beinhaltet sowohl den Aspekt der falschen Anwendung asynchroner Frameworks für geeignete Problemstellungen, andererseits nicht geeignete Anwendungsbereiche.

Ein Paradebeispiel für den unvorteilhaften Einsatz von asynchronen Servertechnologien ist die Verwendung von Callback-Funktionen, die Arbeitsspeicher oder Prozessor stark auslasten und dadurch die Eventschleife an der Abarbeitung neu eintreffender Requests hindern. Jede Callbackfunktion unterbricht die Verarbeitung von Requests in der Eventschleife. Dauert die Behandlung der Callbackfunktion besonders lange, hat dies denselben Effekt wie die ein- oder ausgabebedingte Blockade bei der synchronen Verarbeitung. Nicht abgearbeitete Requests sammeln sich an und das System wird aus

[114] Vgl. o. A. (2011a)
[115] Vgl. Dokumentation: *http://nodejs.org/api/stream.html#stream_stream*
[116] Vgl. Webauftritt: *http://transloadit.com/blog/2010/12/realtime-encoding-over-150x-faster/*
[117] Vgl. Teixeira, P. (2013), S. 80f.

Nutzersicht langsam und im Extremfall unerreichbar.[118] Die Konsequenz daraus ist die Notwendigkeit schlanker, ressourcensparender Callbackfunktionen, die die Abarbeitung in der Eventschleife nur für möglichst kurze Zeit unterbrechen.

Die Vorteile asynchroner Frameworks kommen besonders dann zum Tragen, wenn der Webserver eine Vielzahl gleichzeitiger ein- und ausgabeintensiver Anfragen bewältigen muss. Sind Interaktionen mit Drittsystemen hingegen kaum notwendig, so zahlt sich dieser Vorteil nicht aus. Dies kann sogar gegenteilige Folgen haben und die Abarbeitung der Requests verlangsamen, da die Eventschleife dann mit der Ereigniserkennung und dem Eventhandling überlastet ist, während die Workerthreads kaum gebraucht werden.[119] Beim synchronen Multithreading würden die Requests hingegen nebenläufig abgearbeitet. Dies trifft beispielsweise auf einfache serverseitige Webanwendungen zu, die einzig als Schnittstelle für ein objektrelationales Modell dienen und weder mit externen Systemen kommunizieren, noch besonders rechenintensiv sind. In diesem Fall können synchrone Frameworks wie Ruby on Rails oder ASP.Net MVC performanter sein.[120] Bei hohen Threadzahlen treten jedoch auch bei asynchronen Frameworks Verzögerungen durch die häufigen Kontextwechsel auf.[121]

Auch, wenn die Requests nicht (nur) ein- oder ausgabeintensiv sind, sondern sehr hohe Anforderungen an die Prozessorleistung stellen, können asynchrone Technologien ungeeignet sein.[122] Wird die CPU durch die Abarbeitung rechenintensiver Requests in Anspruch genommen, so ist die Auslagerung in (auf derselben CPU verarbeitete) Workerthreads wenig sinnvoll. In diesem Fall kann nur echte Parallelisierung auf mehreren Prozessen Abhilfe schaffen.[123] Ein Beispiel hierfür sind Echtzeitanalysetools wie beispielsweise eine Business Intelligence Oberfläche, die vom Nutzer personalisiert werden kann.

4.6 **Bisherige Betrachtungen**

Das Aufkommen asynchroner Frameworks wie Node.js, Twisted oder Vert.x bringt das Konzept der Ereignissteuerung auf die Serverseite. Synchrone Frameworks wie etwa Spring MVC oder Apache Tomcat ermöglichen durch Multithreading zwar ebenfalls die nebenläufige Abarbeitung von Requests, die notwendige Synchronisation ist mitunter jedoch aufwändig und zahlreiche gleichzeitige Requests können die Threads des Webservers blockieren, insbesondere wenn sie ein- oder ausgabeintensiv sind.

[118] Vgl. Teixeira, P. (2013), S. 48
[119] Vgl. Roden, G. (2012),, S. 14
[120] Vgl. Capan, T. (o. J.)
[121] Vgl. Kim, J. (2013)
[122] Vgl. Roden, G. (2012), S. 15
[123] Vgl. Semerau, R. (2011)

Asynchrone Frameworks hingegen gewährleisten die Konsistenz der Threads, indem eine Ereignisschleife das Threadmanagement über den Aufruf von Eventhandlern und die Verarbeitung von Callbackfunktionen übernimmt. Bei richtiger Anwendung ist das Risiko einer Blockade bei einem asynchronen Framework deutlich geringer. Anwendungsfälle finden sich im Unternehmensumfeld etwa bei Streaming, Instant Messaging oder der Systemüberwachung. Asynchrone Frameworks können jedoch auch falsch eingesetzt werden, etwa, wenn die Callbackfunktionen rechenintensiv sind und die Eventschleife blockieren.

Am Markt gibt es eine Vielzahl asynchroner Frameworks für verschiedene Sprachen, wobei das auf JavaScript basierende Node.js den Markt seit einigen Jahren dominiert. An dieser Stelle kann nun auch besser erklärt werden, warum aus der Menge an asynchronen Frameworks ausgerechnet das verhältnismäßig unbekannte, kleine Vert.x zum Vergleich mit einer weitverbreiteten synchronen Servertechnologie wie Apache Tomcat genutzt wird:

Das erste Release von Vert.x liegt weniger als zwei Jahre zurück und Vert.x ist damit eines der neuesten asynchronen Frameworks, das im Vergleich zu älteren Frameworks inklusive Node.js auch einige neue Features bietet. So ist Vert.x polyglott und nicht an eine Programmiersprache gebunden. Gerade die Java-Unterstützung ist im Unternehmensumfeld von besonderer Wichtigkeit, da Java dort noch immer eine der dominierenden Sprachen ist (vergleiche Kapitel 2.4). Vert.x ermöglicht dadurch die Nutzung bereits vorhandener Kenntnisse, kann helfen, die Umstiegskosten zu verringern und eröffnet Java-Programmierern den Zugang zu serverseitiger asynchroner Programmierung.

Außerdem wird Vert.x auf der JVM ausgeführt, die eigentlich nicht prädestiniert ist für asynchrone Anwendungen. Dies ermöglicht aber im Gegensatz zu den meisten anderen Frameworks neben der Nebenläufigkeit auf einem Prozessorkern zusätzlich die parallele Nutzung weiterer Eventschleifen auf weiteren Prozessorkernen, wobei sich der Programmierer um die technische Umsetzung keinerlei Gedanken machen muss und einsträngig programmieren kann.

Diese Fülle an interessanten, neuen Eigenschaften legt trotz der geringen Verbreitung des jungen Frameworks einen Leistungsvergleich mit bestehenden Technologien nahe. Auch nach langer Recherche ist aber der einzige Leistungsvergleich, dass die Autoren bei der Recherche finden konnten, ein HTTP benchmarking mit Node.js, das jedoch von einem Vert.x Entwickler durchgeführt wurde und eine deutliche Überlegenheit von Vert.x gegenüber Node.js ergab.[124]

[124] Vgl. o. A. (2012)

5 Szenarien (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

Um einen Kontext für einen Versuchsmechanismus zu geben, werden Szenarios erstellt, wo die Servertechnologien im Unternehmensumfeld zum Einsatz kommen könnten. Die Szenarioerstellung, -konkretisierung und die Darstellung mit Hilfe von Anwendungsfällen schlägt die Brücke von den theoretischen Überlegungen zu Webservices, -servern und Servertechnologien zur tatsächlichen Implementierung, die im vierten Teil der Arbeit schließlich die Grundlage für den Leistungsvergleich sein wird.

Eine genaue Definition von Szenario lautet:

„Ein Szenario (engl. Scenario) beschreibt ein eine tatsächliche oder denkbare Aktions- und Ereignisreihenfolge. Im Kontext von Informationssystemen können Szenarien prinzipiell auf unterschiedlichen Abstraktionsebenen eingesetzt werden. Sie können beispielsweise die Einbettung eines Informationssystemen in einem organisatorischen Umfeld beschrieben, oder sie können ein Anwendungsszenario des Systems darstellen, oder sie können auch die für menschlichen Benutzer nicht direkt sichtbaren Abläufen zwischen den einzelnen Komponenten innerhalb eines Informationssystem abbilden“.[125]

Zum Verallgemeinern von verschiedenen möglichen konkreten Szenarien werden unter anderem Use-Cases benutzt. Diese bieten eine abstrakte Betrachtungsmöglichkeit.[126]

5.1 Use-Case Diagramme

5.1.1 Einleitung

Use-Case Diagramme geben eine externe Nutzersicht auf das zu entwickelnde System. Sie zeigen die Beziehungen zwischen Aktoren und Use-Cases. Aktoren können Personen oder auch andere Systeme wie zum Beispiel Schnittstellen sein. Systemreaktionen auf Ereignisse aus dessen Umwelt werden durch Use-Cases abgebildet. Uses-Cases werden im Deutschen als Anwendungsfall bezeichnet. Sie sind gekapselt und beinhalten verschiedene Aktionen die in einer definierten Reihenfolge ablaufen. Beim Ablauf der Anwendungsfälle werden von den Aktoren verschiedene Daten oder Signale an das System gesendet welches den Use-Case realisiert. Dieses zeigt dann an was, aber nicht wie etwas auf den Aktor reagiert. Wird zum Beispiel eine Website aufgerufen, beschreibt der Use-Case zwar dass diese aufgerufen wird aber weder wie das genau geschieht noch wie diese

[125] Hansen, H /Neumann,G (2010), S.312
[126] Vgl. ibidem, S.312

dargestellt wird. Deshalb werden auch keine Klasse oder Objekte in einem solchen Diagramm dargestellt. [127]

5.1.2 Motivation für den Einsatz von Use Cases

Use-Cases sind anwendernah gestaltet und bieten eine Sicht auf das System und deren Umwelt ohne sich mit technischen Aspekten zu befassen. Use-Cases werden verwendet wenn:

- Das System in kleinere logische Teile zerlegt werden sollen um somit eine bessere Handhabung zu schaffen und somit auch „auf einen Blick“ die Dienste des Systems bzw. der Komponenten des Systems zu zeigen.
- Die Kommunikation von System und Umwelt, wie Schnittstellen oder Personen dargestellt werden soll.
- Kleinere planbare Einheiten für die Entwicklung benötigt werden.

Es ist eine Black Box Sicht auf das System und ermöglicht eine verständliche Darstellung auf einem hohen Abstraktionsniveau. Gerade deswegen eignen sich Use-Case Diagramme in der Anforderungsanalyse, da Sie auch von Personen die mit der eigentlichen Entwicklung nichts zu tun haben leicht verstanden werden. Vor allem bei einer inkrementellen Entwicklung, eigenen sich die Use-Case Diagramme. Einzelne Fälle werden Prioritäten zugeordnet und danach entwickelt und implementiert und anschließend getestet.[128]

5.1.3 Grafische Notation

Aktoren sind entweder menschliche Benutzer oder andere Systeme und werden mit einem Strichmännchen dargestellt. Sie werden mit einem durchgezogenen Strich ohne Pfeilspitzem mit den Use-Cases verbunden, die von den jeweiligen Aktoren ausgeführt werden.

Generalisierungsbeziehungen können zwischen Aktoren eingetragen werden. Dadurch können verschiedene Aktoren, welche die gleichen Anwendungsfälle aufrufen, verallgemeinert werden. Eigene besondere Funktionen werden dann bei dem spezifischen Aktor ausgeführt. Die Generalisierung wird durch einen durchgezogenen Pfeil vom speziellen Aktor zum generalisierten Aktor dargestellt. Eine weitere Darstellungsmöglichkeit des Aktors ist ein Rechteckmit der Beschriftung <<actor>> und dann der Name des Aktors.

[127] Vgl.:Jeckle, M (2013), S.2-4
[128] Vgl. ibidem, S. 5

Ein Beispiel für die Generalisierung ist, dass ein Filialleiter genau die gleichen Anwendungsfälle wie ein normaler Sachbearbeiter aufrufen kann und zusätzlich noch weitere Anwendungsfälle hat, die einem Sachbearbeiter nicht zur Verfügung stehen.

Die Notation des Use-Case ist eine Ellipse, in welcher die Anwendung kurz beschrieben wird. Zwischen einzelne Use-Cases können verschiedene Beziehungen bzw. Spezialfälle bestehen:

- Include: Ein Use-case enthält einen anderen Use-Case. Dadurch können gemeinsame Abläufe zentral modelliert werden. Beim Ausführen von inkludierenden Use-Cases werden die inkludierten Use-Cases ebenfalls ausgeführt. Die Notation ist ein mit <<include>> beschrifteter Pfeil.
- Extend: Hierbei wird ein allgemeingehaltener Use-Case um einen weiteren, spezielleren erweitert. Dieser hat noch weiter Funktionen im Gegensatz zum Basisfall.

Die Umgebung wird in der Notation durch ein Rechteck begrenzt.

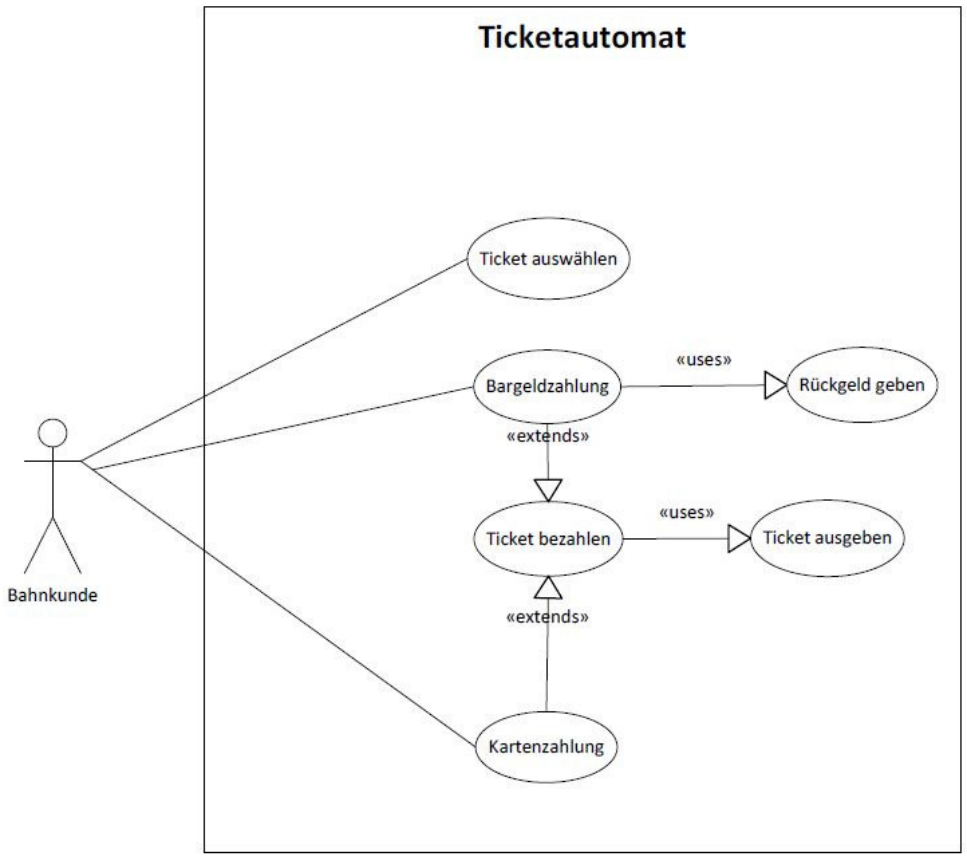


Abbildung 15: Beispiel eines Use-Case Diagramms

Use-Cases werden häufig auch noch in Textform beschrieben. Hierfür werden Templates wie folgendes benutzt:

Name des Use-Case:	<eindeutiger Name>
Zugehöriges Benutzerziel:	<Ziel, zu dessen Erreichung der Use-Case beiträgt>
Prioritäten	<beispielsweise hoch, mittel, niedrig>
Kurzbeschreibung	<kurze Textbeschreibung des Use Cases>
Aktoren	<beteiligte Aktoren>
Auslöser(Trigger)	<das den Use Case auslösende Ereignis>
Vorbedingungen	<Bedingung, die vor Durchführung des Use-Case erfüllt sein müssen>
Nachbedingungen	<Bedingungen, die nach Durchführung des Use Case erfüllt sein müssen>
Szenarien	<mehrere Szenarien die die Durchführung des Use Case beschreiben>
Zusatzinformationen	<Zusatzinformationen die für das Verständnis dienlich sein können>

Tabelle 1: Template für Use Cases

Um einzelne Szenarien zu konkretisieren, können zum Beispiel Sequenzdiagramme oder auch ereignisgesteuerte Prozessketten verwendet werden.[129]

[129] Vgl.: Hansen, H /Neumann,G (2010), S.312-315

6 Praktische Umsetzung: Szenario (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

Um die Vor- und Nachteile der asynchronen Servertechnologien auszutesten haben wir uns entschieden für ein Szenario im Bankenumfeld entschieden. Die praktische Umsetzung wird in einem späteren Kapitel erläutert. Um dieses einfach und leicht verständlich darzustellen verwenden wir Use-Case Diagramme. Aktoren sind hierbei Kunden und Bankmitarbeiter, diese sind generalisiert in dem Aktor Benutzer, ein Kunde hat nicht zwar nicht mehr Funktionen als ein Benutzer, er wurde lediglich zur Übersicht erstellt.

Zunächst einmal das Use-Case Diagramm und die Beschreibung der einzelnen Use-Cases.

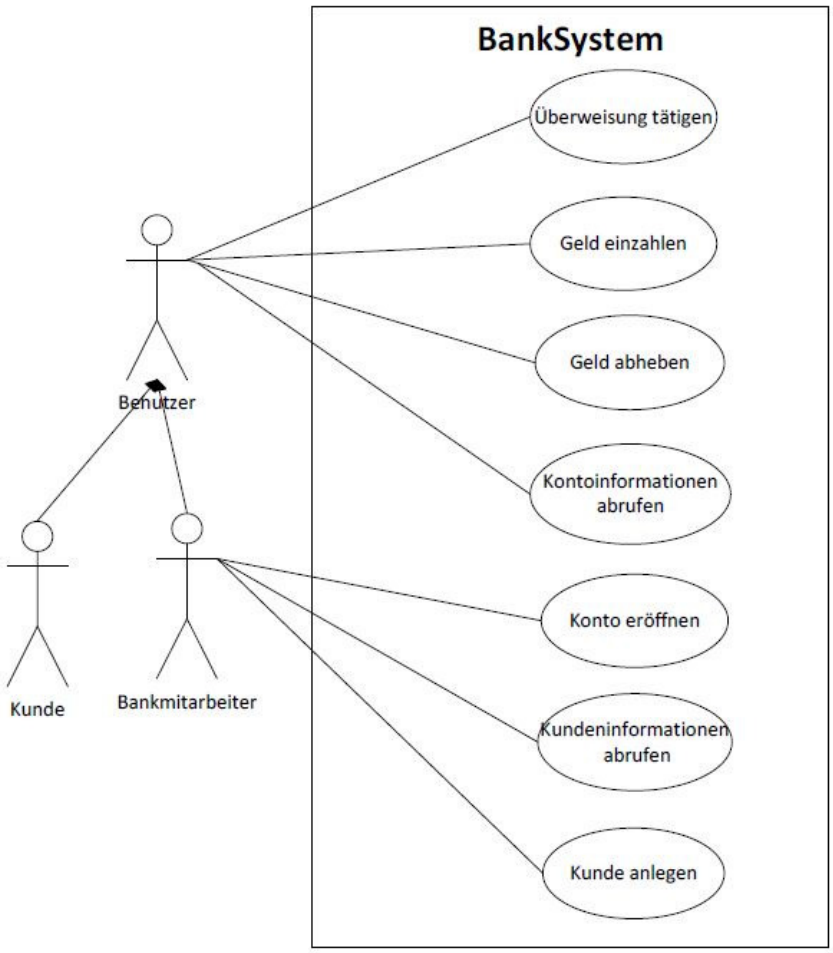


Abbildung 16: Use-Case Szenario

Name des Use-Case:	Überweisung tätigen
Zugehöriges Benutzerziel:	Geld wird auf ein anderes Konto überwiesen
Kurzbeschreibung	Ein Betrag wird von einem Konto auf ein anderes übertragen
Aktoren	Benutzer
Auslöser(Trigger)	Benutzer möchte Überweisen
Vorbedingungen	Konten (sowohl Ausgangs- und Zielkonto) müssen existent sein
Nachbedingungen	Geld muss jeweils gutgeschrieben bzw. abgebucht worden sein
Szenarien	Ein Kunde oder ein Mitarbeiter soll im Auftrag dessen eine Überweisung tätigen. Dies kann über Onlinebanking, am Bankautomat oder am Schalter geschehen.
Zusatzinformationen	Laufzeit: kurz

Tabelle 2: Use-Case Überweisung tätigen

Name des Use-Case:	Geld einzahlen
Zugehöriges Benutzerziel:	Geld wird auf ein Konto eingezahlt
Kurzbeschreibung	Bargeld wird entgegengenommen und auf Konto gutgeschrieben
Aktoren	Benutzer
Auslöser(Trigger)	Benutzer möchte einzahlen
Vorbedingungen	Konto muss existent sein
Nachbedingungen	Geld muss gutgeschrieben worden sein
Szenarien	Ein Kunde oder ein Mitarbeiter soll im Auftrag dessen eine Buchung auf ein Konto tätigen. Dies kann am Bankautomat oder am Schalter geschehen.
Zusatzinformationen	Laufzeit: kurz

Tabelle 3: Use-Case Geld einzahlen

Name des Use-Case:	Geld abheben
Zugehöriges Benutzerziel:	Geld wird von einem Konto ausgezahlt
Kurzbeschreibung	Ein Geldbetrag wird von einem Konto abgebucht und in Bargeld ausgezahlt
Aktoren	Benutzer
Auslöser(Trigger)	Benutzer möchte Geld abheben
Vorbedingungen	Konto muss existent sein, Guthaben/ Kreditrahmen muss ausreichen
Nachbedingungen	Geld muss abgebucht worden sein
Szenarien	Ein Kunde oder ein Mitarbeiter soll im Auftrag dessen einen Geldbetrag abheben. Dies kann am Bankautomat oder am Schalter geschehen.
Zusatzinformationen	Laufzeit: kurz

Tabelle 4: Use-Case Geld abheben

Name des Use-Case:	Kontoinformationen einholen
Zugehöriges Benutzerziel:	Informationen über das Konto werden ausgegeben
Kurzbeschreibung	Kontoinformationen wie Umsätze, Stand und Bedingungen werden ausgegeben
Aktoren	Benutzer
Auslöser(Trigger)	Benutzer möchte Informationen
Vorbedingungen	Konto muss existent sein
Nachbedingungen	Informationen müssen ausgegeben werden
Szenarien	Ein Kunde oder ein Mitarbeiter soll im Auftrag dessen Informationen einholen. Dies kann über Onlinebanking, am Bankautomat oder am Schalter geschehen.
Zusatzinformationen	Laufzeit: kurz

Tabelle 5: Use-Case Kontoinformationen einholen

Name des Use-Case:	Konto eröffnen
Zugehöriges Benutzerziel:	Konto wird eröffnet
Kurzbeschreibung	Neues Konto wird für einen neuen oder bestehenden Kunden eröffnet.
Aktoren	Bankmitarbeiter
Auslöser(Trigger)	Bankmitarbeiter möchte im Auftrag eines Kunden ein Konto eröffnen
Vorbedingungen	Kunde muss vorhanden sein
Nachbedingungen	Neues Konto muss existieren
Szenarien	Ein Kunde oder ein Mitarbeiter soll im Auftrag dessen ein neues Konto eröffnen. Dies kann nur am Schalter erfolgen.
Zusatzinformationen	Laufzeit: lang

Tabelle 6: Use-Case Konto eröffnen

Name des Use-Case:	Kundeninformationen einholen
Zugehöriges Benutzerziel:	Konto wird eröffnet
Kurzbeschreibung	Neues Konto wird für einen neuen oder bestehenden Kunden eröffnet.
Aktoren	Bankmitarbeiter
Auslöser(Trigger)	Bankmitarbeiter möchte im Auftrag eines Kunden ein Konto eröffnen
Vorbedingungen	Kunde muss vorhanden sein
Nachbedingungen	Neues Konto muss existieren
Szenarien	Ein Kunde oder ein Mitarbeiter soll im Auftrag dessen ein neues Konto eröffnen. Dies kann nur am Schalter erfolgen.
Zusatzinformationen	kurz

Tabelle 7: Use-Case Kundeninformationen einholen

Name des Use-Case:	Neuer Kunde anlegen
Zugehöriges Benutzerziel:	Kundendaten werden angelegt
Kurzbeschreibung	Ein neuer Kunde soll ins System eingetragen werden
Aktoren	Bankmitarbeiter
Auslöser(Trigger)	Bankmitarbeiter möchte einen Neukunden anlegen
Vorbedingungen	Alle Informationen müssen vorhanden sein
Nachbedingungen	Kunde soll in Datenbank existieren.
Szenarien	Ein Kunde wird durch Akquise in die Datenbank der Bank gespeichert. Dies kann jedoch nur vom Bankmitarbeiter getätigt werden.
Zusatzinformationen	lang

Tabelle 8: Use-Case Neuer Kunde anlegen

Dieses Szenario könnte man noch durch weitere Use-Cases erweitern. Es soll auch nicht den komplexen Sachverhalt einer Bank wiederspiegeln, sondern lediglich eine vereinfachte Darstellung bieten um sich das Einsatzgebiet von asynchronen bzw. synchronen Servertechnologien vorzustellen. Hierbei wichtig ist, dass es Prozesse gibt die länger bzw. kürzer dauern, da unterschiedliche Laufzeiten simuliert werden sollen. Dieses Use-Case Diagramm war der gedankliche Hintergrund für die das Testsystem. In der tatsächlichen Versuchsumgebung gibt es keine Aktoren Mitarbeiter usw. sondern lediglich ein Client der diese ersetzt.

7 Architektur (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

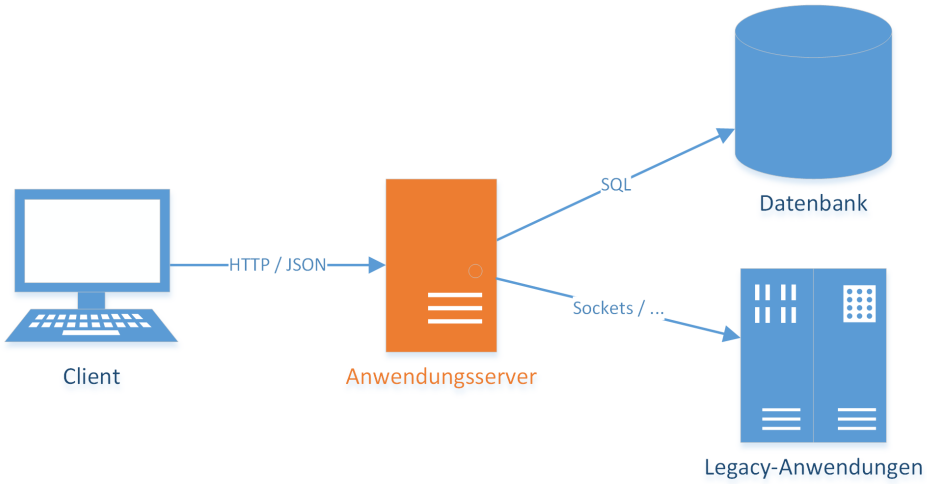


Abbildung 17: Architektur des Messszenarios

7.1 Überblick

Die Implementierung des Szenarios erfolgt mittels SOA: Die Funktionen der Bank werden in einem von einem Anwendungsserver gehosteten Webservice bereitgestellt, welcher dann von einem Klienten genutzt wird. Dieser wiederrum greift auf eine Datenbank zur Speicherung von Stamm- und Bewegungsdaten, sowie auf Legacy-Anwendungen zu.

Im Beispiel existiert nur ein generischer Client, welcher auf einen einzigen Webservice zugreift. Dieser lässt sich jedoch parametrisieren, um verschiedene Arten “echter” Klienten zu simulieren. Als Datenbanksystem wird MySQL verwendet. Hingegen wird der Zugriff auf die Legacy-Anwendungen lediglich durch Wartezeiten simuliert.

Die Kommunikation zwischen Webservice und Client erfolgt über HTTP (und somit letztlich via TCP/IP). Das Protokoll auf Anwendungsebene basiert auf REST und JSON und ist in einem nachfolgenden Abschnitt genauer beschrieben.

7.2 Vergleich mit der Realität

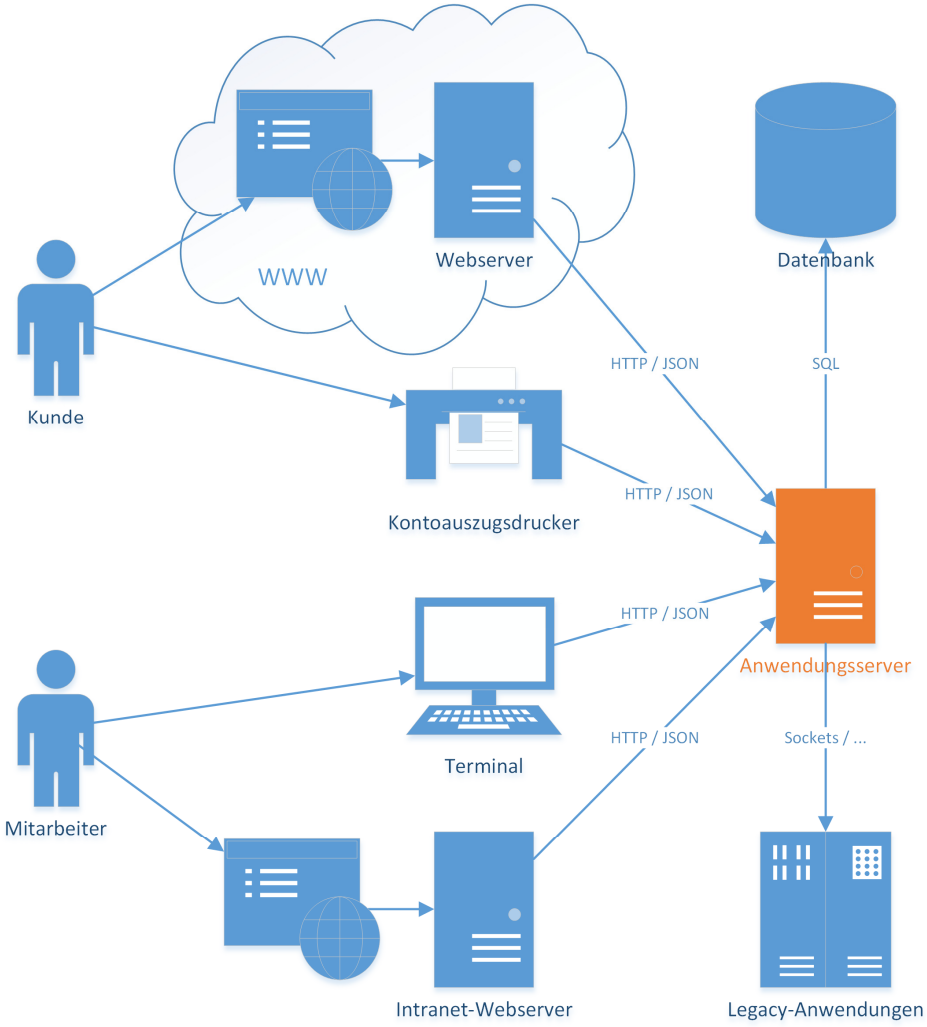


Abbildung 18: Schematische Erweiterung der Architektur des Messszenarios

In der Praxis lässt sich diese Architektur einfach erweitern wie in Abbildung 18 angedeutet: Anstelle des einzelnen Klienten treten eine ganze Reihe unterschiedlicher Client, beispielsweise Kontoauszugsdrucker oder Geldautomaten. Diese Klienten nutzen jeweils nur einen Teilbereich des Webservice: Während ersterer ausschließlich auf Methoden zum Erstellen von Transaktionsübersichten zugreifen würde, wäre letzterer auf die Nutzung der Methoden zur Abbuchung beschränkt. Außerdem ist eine Webschnittstelle (z.B. für Onlinebanking) angedeutet. Diese würde zur Vermeidung von Redundanzen und zur Sicherstellung der Datenintegrität nicht direkt auf die Datenbank zugreifen, sondern ebenfalls den Services nutzen, und deren Ergebnisse in HTML in einer für den Endnutzer zugänglichen Form aufbereiten. Auch die Annahme, dass nur ein Webservice bereitgestellt wird, lässt sich in der Praxis nicht halten: Sinnvoll wäre beispielsweise eine Aufteilung des

Webservices nach Funktionalitäten. Auf diese Weise kann unter anderem die Ausfallsicherheit erhöht werden (wenn der Webservice für Kontoauszüge ausfällt, kann weiterhin Geld abgehoben werden und umgekehrt).

Das Beispielszenario wurde jedoch bewusst einfach gehalten: So würde sich eine Aufteilung des Webservices auf mehrere Webservices kontraproduktiv auf die Aussage der Messergebnisse auswirken (hier soll ja gerade eine Lastsituation auf einem einzelnen Webserver provoziert werden). Auch eine Bereitstellung unterschiedlicher Clients würde den Rahmen dieser Arbeit sprengen und die Aussagekraft der Messungen nicht wesentlich erhöhen, wenn nicht sogar senken.

7.3 **Schnittstelle**

Die Schnittstelle basiert auf dem HTTP-Protokoll und verwendet JSON als Austauschformat. Dabei werden diverse Endpunkte bzw. Servicemethoden angeboten, die jeweils mittels POST angesprochen werden. Dabei wird als Inhalt der Anfrage ein JSON-Objekt verschickt und als Antwort ebenfalls ein JSON-Objekt zurückgeliefert.

Im Folgenden werden die einzelnen Servicemethoden beschrieben.

7.3.1 Überweisung durchführen

<table>
<tr><td>URL</td><td>/transfer</td></tr>
<tr><td>Methode</td><td>POST</td></tr>
<tr><td>Beschreibung</td><td>Führt eine Überweisung von einem Bankkonto auf ein anderes durch.</td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr>
<tr><td>sourceAccount</td><td>int</td><td>Kontonummer des Auftraggebers</td></tr>
<tr><td>targetAccount</td><td>int</td><td>Kontonummer des Überweisungszieles</td></tr>
<tr><td>amount</td><td>double</td><td>Zu überweisender Betrag in Euro</td></tr>
<tr><td>comment</td><td>String</td><td>Verwendungszweck</td></tr>
</table>
</td></tr>
<tr><td>Rückgabewert</td><td>N/A</td></tr>
<tr><td>Vorbedingungen</td><td>• sourceAccount und targetAccount müssen gültige Kontonummern sein, die sich unterscheiden
• amount muss echt größer Null sein
• Wenn sourceAccount oder targetAccount ein Tagesgeldkonto</td></tr>
</table>

	bezeichnet, müssen beide Konten den gleichen Inhaber besitzen • Der Betrag von sourceAccount inkl. eines eventuell vorhandenen Disporahmens muss größer oder gleich des zu überweisenden Betrages sein
Fehler	• InvalidAccountException: Es wurde eine ungültige Kontonummer angegeben • InvalidAmountException: Es wurde ein ungültiger Betrag spezifiziert • InvalidTransferException: Es wurde versucht, Geld von einem Tagesgeldkonto auf ein Konto eines anderen Benutzers zu überweisen bzw. umgekehrt • InsufficientFundsException: Der Kontostand reicht nicht aus, um die Transaktion abzuwickeln

7.3.2 Einzahlung verbuchen

URL	/deposit
Methode	POST
Beschreibung	Verbucht eine Einzahlung auf einem angegebenen Konto
Parameter	<table><tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr><tr><td>accountNumber</td><td>int</td><td>Kontonummer</td></tr><tr><td>amount</td><td>double</td><td>Betrag in Euro</td></tr></table>
Rückgabewert	N/A
Vorbedingungen	• accountNumber muss ein gültiges Girokonto sein • amount muss echt größer Null sein
Fehler	• InvalidAccountException: Es wurde eine ungültige Kontonummer angegeben • InvalidAmountException: Es wurde ein ungültiger Betrag spezifiziert • InvalidTransferException: Es wurde versucht, Geld auf ein Tagesgeldkonto einzuzahlen

7.3.3 Abhebung verbuchen

<table>
<tr><td>URL</td><td>/draw</td></tr>
<tr><td>Methode</td><td>POST</td></tr>
<tr><td>Beschreibung</td><td>Verbucht eine Abhebung auf einem angegebenen Konto</td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr>
<tr><td>accountNumber</td><td>int</td><td>Kontonummer</td></tr>
<tr><td>amount</td><td>double</td><td>Betrag in Euro</td></tr>
</table>
</td></tr>
<tr><td>Rückgabewert</td><td>N/A</td></tr>
<tr><td>Vorbedingungen</td><td>

accountNumber muss ein gültiges Giroonto sein
amount muss echt größer Null sein
Der Kontostand von accountNumber inkl. eines eventuell vorhandenen Disporahmens muss größer oder gleich des abzuhebenden Betrages sein

</td></tr>
<tr><td>Fehler</td><td>

InvalidAccountException: Es wurde eine ungültige Kontonummer angegeben
InsufficientFundsException: Der Kontostand reicht nicht aus, um die Transaktion abzuwickeln
InvalidAmountException: Es wurde ein ungültiger Betrag spezifiziert
InvalidTransferException: Es wurde versucht, Geld von einem Tagesgeldkonto abzuheben

</td></tr>
</table>

7.3.4 Konto anlegen

<table>
<tr><td>URL</td><td>/openAccount</td></tr>
<tr><td>Methode</td><td>POST</td></tr>
<tr><td>Beschreibung</td><td>Legt ein neues Konto an</td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr>
<tr><td>Customer</td><td>Int</td><td>Kundennumer</td></tr>
</table>
</td></tr>
<tr><td>Rückgabewert</td><td>
<table>
<tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr>
<tr><td>AccountNumber</td><td>int</td><td>Kontonummer</td></tr>
<tr><td>CustomerId</td><td>int</td><td>Kundennummer</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td>CreditLine</td><td>double</td><td>Disporahmen</td></tr><tr><td>Balance</td><td>double</td><td>Kontostand</td></tr></table>
Vorbedingungen	• Customer muss eine gültige Kundennummer sein
Fehler	• InvalidCustomerException: Es wurde eine ungültige Kundennummer angegeben

7.3.5 Kunde anlegen

URL	/newCustomer
Methode	POST
Beschreibung	Legt einen neuen Kunden an
Parameter	<table><tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr><tr><td>FirstName</td><td>String</td><td>Vorname des Kunden</td></tr><tr><td>LastName</td><td>String</td><td>Nachname des Kunden</td></tr><tr><td>Email</td><td>String</td><td>E-Mailadresse des Kunden</td></tr><tr><td>Birthday</td><td>String</td><td>String</td></tr></table>
Rückgabewert	<table><tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr><tr><td>Id</td><td>int</td><td>Kundennummer</td></tr><tr><td>FirstName</td><td>String</td><td>Vorname des Kunden</td></tr><tr><td>LastName</td><td>String</td><td>Nachname des Kunden</td></tr><tr><td>Email</td><td>String</td><td>E-Mailadresse des Kunden</td></tr><tr><td>Birthday</td><td>String</td><td>String</td></tr><tr><td>AccountIds</td><td>int[]</td><td>Kontonummern aller Konten des Kunden</td></tr></table>
Vorbedingungen	• Die Daten des Kunden müssen gültig sein • Der Kunde muss älter als 18 Jahre sein
Fehler	• InvalidDataException: Es wurden ungültige Kundendaten angegeben • CustomerNotAcceptedException: Der Kunde wurde aufgrund einer Kundenprüfung nicht als Kunde akzeptiert

7.3.6 Kontoauszug erzeugen

URL	/statement
Methode	POST
Beschreibung	Erzeugt einen Kontoauszug zu einem angegebenen Konto
Parameter	<table><tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr><tr><td>AccountId</td><td>int</td><td>Kontonummer</td></tr></table>
Rückgabewert	<table><tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr><tr><td>accountNumber</td><td>int</td><td>Kontonummer</td></tr><tr><td>startBalance</td><td>double</td><td>Anfangsstand</td></tr><tr><td>endBalance</td><td>double</td><td>Endstand</td></tr><tr><td>start</td><td>Date</td><td>Anfangsdatum</td></tr><tr><td>end</td><td>Date</td><td>Enddatum</td></tr><tr><td>entries</td><td>StatementEntry[]</td><td>Einträge</td></tr></table>
Vorbedingungen	• AccountId muss eine gültige Kontonummer sein
Fehler	• InvalidAccountException: Es wurde eine ungültige Kontonummer angegeben

7.3.7 Kontoinformationen abrufen

URL	/account
Methode	POST
Beschreibung	Legt ein neues Konto an
Parameter	<table><tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr><tr><td>AccountId</td><td>Int</td><td>Kontonummer</td></tr></table>
Rückgabewert	<table><tr><td>Name</td><td>Typ</td><td>Beschreibung</td></tr><tr><td>AccountNumber</td><td>int</td><td>Kontonummer</td></tr><tr><td>CustomerId</td><td>int</td><td>Kundennummer</td></tr><tr><td>CreditLine</td><td>double</td><td>Disporahmen</td></tr></table>

	Balance | double | Kontostand
Vorbedingungen	• AccountId muss eine gültige Kontonummer sein
Fehler	• InvalidAccountException: Es wurde eine ungültige Kontonummer angegeben

7.3.8 Kundeninformationen abrufen

URL	/customer
Methode	POST
Beschreibung	Ruft die Informationen zu einem Kunden ab
Parameter	Name | Typ | Beschreibung CustomerId | int | Kundennummer
Rückgabewert	Name | Typ | Beschreibung Id | int | Kundennummer FirstName | String | Vorname des Kunden LastName | String | Nachname des Kunden Email | String | E-Mailadresse des Kunden Birthday | String | String AccountIds | int[] | Kontonummern aller Konten des Kunden
Vorbedingungen	• Kundennummer muss gültig sein
Fehler	• InvalidCustomerException: Es wurde eine ungültige Kundennummer angegeben

8 Implementierung (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

8.1 Allgemeines

Das gesamte Projekt wurde in Java entwickelt. Als Entwicklungsumgebung wurde die Spring Tool Suite (STS) verwendet, eine auf Eclipse basierte IDE zur Entwicklung von Spring-basierten Anwendungen130. Alle Projekte nutzen Spring-Context zur Dependency Injection (DI), um Abhängigkeiten innerhalb der einzelnen Komponenten aufzulösen.

Externe Abhängigkeiten und der Buildprozess werden mittels Maven verwaltet131. Je Projekt sind in einer sogenannten POM-Datei (für „project object model“) diverse für den Buildprozess relevante Eigenschaften abgelegt, sowie alle externen Abhängigkeiten aufgeführt. Wird nun ein Build durchgeführt, werden die externen Abhängigkeiten automatisch aus dem zentralen Maven Repository heruntergeladen und in das Projekt eingebunden. Im Anschluss wird der Code kompiliert und entsprechend in eine jar- bzw. war-Datei verpackt. Der Buildprozess ist in einer README-Datei im Stammverzeichnis des Projektes beschrieben.

Die Gruppen-Id aller Projekte lautet „de.dhbwstuttgart.wi2011i.sbank“, was dem gemeinsamen Namensraum aller Klassen in den Projekten entspricht.

Zur Quellcodeverwaltung wurde das verteilte Versionsverwaltungssystem GIT eingesetzt.

8.2 Datentypen

Das Projekt „de.dhbwstuttgart.wi2011i.sbank.common“ enthält sämtliche Datentypen, die auf Ebene der Schnittstelle zwischen Client und Server definiert sind. Im Namensraum „de.dhbwstuttgart.wi2011i.sbank.parameters“ findet sich je Servermethode ein entsprechender Typ, welcher als deren Parametertyp verwendet wird. Entsprechend befinden sich im Namensraum „de.dhbwstuttgart.wi2011i.sbank.results“ die Typen, welche der Server jeweils als Ergebnis einer Servermethode zurückliefert.

8.3 Server

Um Vergleichbarkeit zu gewährleisten, sind beide Server hinsichtlich des Protokolls vollständig kompatibel zueinander. Beide Server stellen somit ein HTTP-Frontend zur Verfügung, welches JSON als Austauschformat nutzt.

---

[130] Vgl. http://spring.io/tools/sts

[131] https://maven.apache.org/

Auch auf Ebene der Implementierung greifen beide Projekte auf eine gemeinsame Codebasis zu. Hier sind vor allem die Datentypdefinitionen und die Serviceklassen zu nennen, in welchen die eigentliche der Serverfunktionalität implementiert ist. Unterschiede ergeben sich somit ausschließlich auf oberster Architekturebene (des Servers).

Der gemeinsame Code der Server befindet sich im Projekt „de.dhbwstuttgart.wi2011i.sbank.server.common“. Je Servicemethode existiert ein Service für die Implementierung. Außerdem befinden sich hier diverse unterstützende Services, sowie der Code zur Kommunikation mit der Datenbank. Im Sinne von SOLID und DRY befindet sich die Logik der einzelnen Servicemethoden nicht direkt in der Serverimplementierung, sondern in jeweils eigenständigen Serviceklassen, deren Abhängigkeiten mittels DI - einer Anwendung des IoC-Grundsatzes - injiziert werden. Dies senkt die Komplexität und würde eine Entwicklung unter Zuhilfenahme von TDD ermöglichen.

In der Datei „times.properties“ kann festgelegt werden, wie lange die einzelnen Aktionen dauern (in Millisekunden). Dabei existieren je Aktion zwei Zeilen, wobei die „waitTime“ mittels Thread.Sleep und die busyWaitTime mittels einer Schleife realisiert werden.

8.4 **Datenbank**

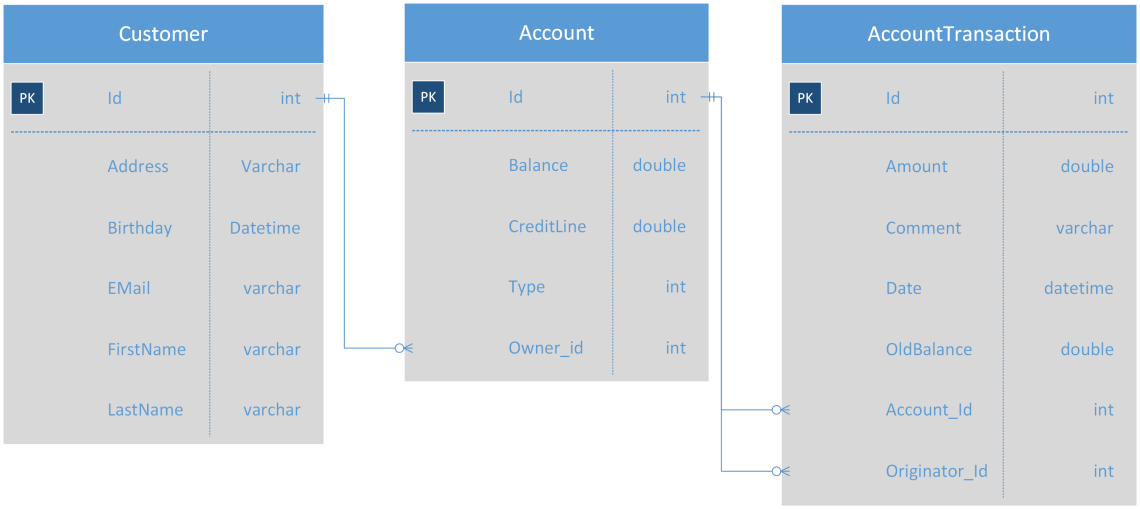


Abbildung 19: Relationenmodell

Da der Datenbankcode beider Serverprojekte identisch ist, können beide auf die gleiche Datenbank zugreifen. Deren Adresse kann in einer Konfigurationsdatei geändert werden.

Zum Zugriff auf die Datenbank wird Spring-Data verwendet, welches intern Hibernate nutzt. Hierbei wird die Datenbank automatisch aufgrund von annotierten Klassen generiert. Ein

ERD der Datenbank ist oben abgebildet. Innerhalb der weiter oben beschriebenen Services erfolgt der Zugriff auf die Datenbank über sogenannte Repositories mithilfe des ORM-Standards JPA. Hierbei werden sämtliche SQL-Statements durch Verwendung vordefinierte Methoden, Annotations sowie COC und einer eigenen, SQL-ähnlichen DSL automatisch generiert.

8.5 „Klassischer“ Webserver

Architektonisch nutzt der klassische Webserver das MVC-Pattern, dessen Verwendung von dem verwendeten Framework Spring MVC forciert wird.

HomeController ist die Controllerklasse der Anwendung: Sie ist für die Zuordnung von Requests zu den Services zuständig. Diese Aufgabe wird hauptsächlich mithilfe der entsprechenden Annotations automatisiert erledigt; ein Beispiel für sogenannte deklarative Programmierung.

8.6 vert.x Server

Grundelemente einer vert.x-Anwendung stellen die sogenannten Verticles da. Vert.x unterscheidet zwischen normalen Verticles und sogenannten Worker-Verticles. Während sich in ersteren ausschließlich nicht-blockierender Code befinden sollte, sind letztere vor allem für blockierende Zugriffe wie Datenbank- oder Dateisystemoperationen gedacht.

Architektonisch ergibt sich eine Aufteilung in das `MainVerticle`, welches die Anwendung initialisiert und die Worker Verticles startet. Zudem wird hier das `RequestVerticle` deployt, dass Anfragen an den Server entgegennimmt und an den Event-Bus weiterleitet.

```
container.deployVerticle("sbank.RequestVerticle", 4);

container.deployWorkerVerticle("sbank.workers.AccountWorker", new
JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.CustomerWorker", new
JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.DepositWorker", new
JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.DrawWorker", new
JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.NewCustomerWorker",
new JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.OpenAccountWorker",
new JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.StatementWorker",
new JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.TransferWorker", new
JsonObject(), 1, true);
```

Listing 1: Abarbeitung der Aktionen in multithreaded Background Vehicles

Außerdem nimmt es die Ergebnisse entgegen und bereitet diese auf.

Die Autoren haben sich aufgrund der Datenbankoperationen in jeder Aktion zunächst dazu entschieden, jede Aktionen in ein eigenes multithreaded Worker-Verticle auszulagern. **Error! Reference source not found.** zeigt den hierzu nötigen Code innerhalb der start Methode

```
container.deployVerticle("sbank.RequestVerticle", 4);

container.deployWorkerVerticle("sbank.workers.AccountWorker", new
JsonObject(), 1, false);
container.deployWorkerVerticle("sbank.workers.CustomerWorker", new
JsonObject(), 1, false);
container.deployWorkerVerticle("sbank.workers.DepositWorker", new
JsonObject(), 1, false);
container.deployWorkerVerticle("sbank.workers.DrawWorker", new
JsonObject(), 1, false);
container.deployWorkerVerticle("sbank.workers.NewCustomerWorker",
new JsonObject(), 11, false);
container.deployWorkerVerticle("sbank.workers.OpenAccountWorker",
new JsonObject(), 3, false);
container.deployWorkerVerticle("sbank.workers.StatementWorker",
new JsonObject(), 1, false);
container.deployWorkerVerticle("sbank.workers.TransferWorker", new
JsonObject(), 1, false);
```

Listing 2: Abarbeitung der Aktionen in konfigurierten Background Vehicles

des MainVerticles.

Im weiteren Verlauf hat sich jedoch gezeigt, dass diese Herangehensweise das Programmiermodell von vert.x eher suboptimal ausnutzt, da die Worker alle auf demselben Threadpool laufen und sich somit gegenseitig blockieren können.

```
container.deployVerticle("sbank.RequestVerticle", 4);

container.deployVerticle("sbank.workers.DepositWorker");
container.deployWorkerVerticle("sbank.workers.AccountWorker", new
JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.CustomerWorker", new
JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.DrawWorker", new
JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.NewCustomerWorker",
new JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.OpenAccountWorker",
new JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.StatementWorker",
new JsonObject(), 1, true);
container.deployWorkerVerticle("sbank.workers.TransferWorker", new
JsonObject(), 1, true);
```

Listing 3: Abarbeitung einer Aktion in der Haupteventschleife

Daraufhin wurden zwei weitere Varianten eingeführt, welche eine differenziertere Betrachtung ermöglichen.

Listing 3 verlagert die Abarbeitung einer Anfragenart in die Haupteventschleife, sodass diese Anfragenart auch abgearbeitet werden kann, wenn alle Workerthreads belegt sind.

In Listing 2 werden die multithreaded Worker Verticles durch singlethreaded Worker Verticles ersetzt und dafür die Zahl entsprechend der Anfragenhäufigkeit und zu erwartender Dauer angepasst. Dies führt dazu, dass in jedem Fall immer mindestens ein Thread für jede Abfragenart zur Verfügung steht.

8.7 **Benchmarktool**

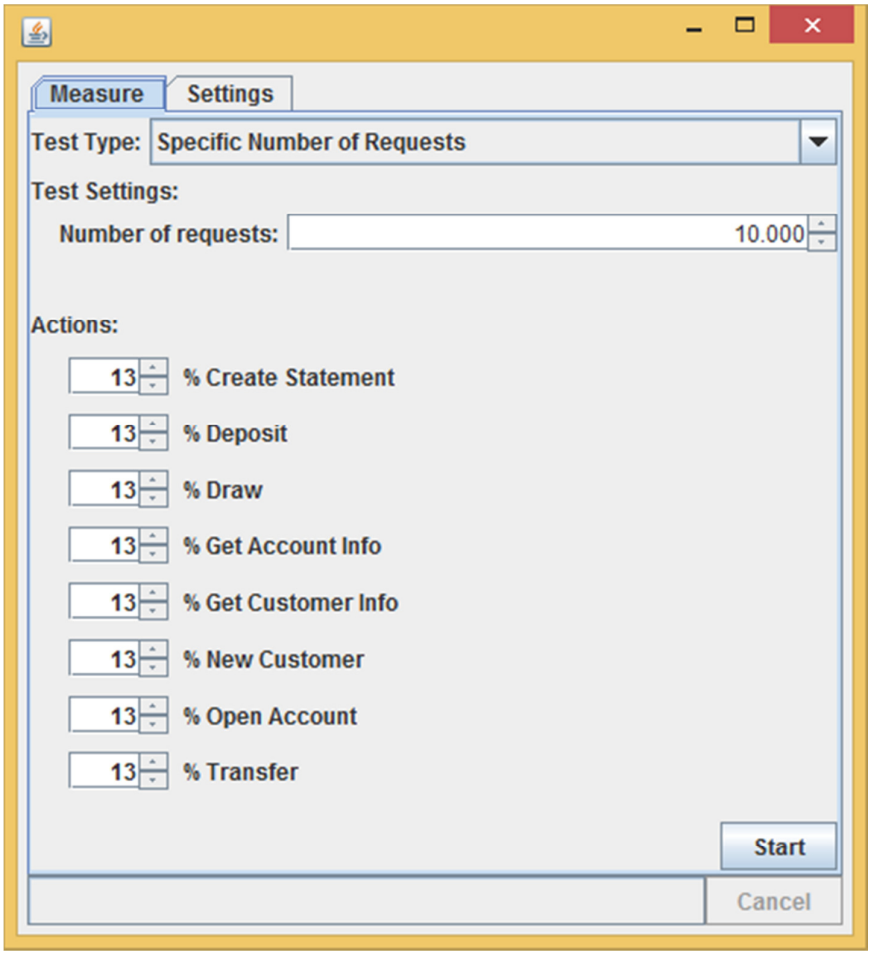


Abbildung 20: Screenshot des Benchmarktools

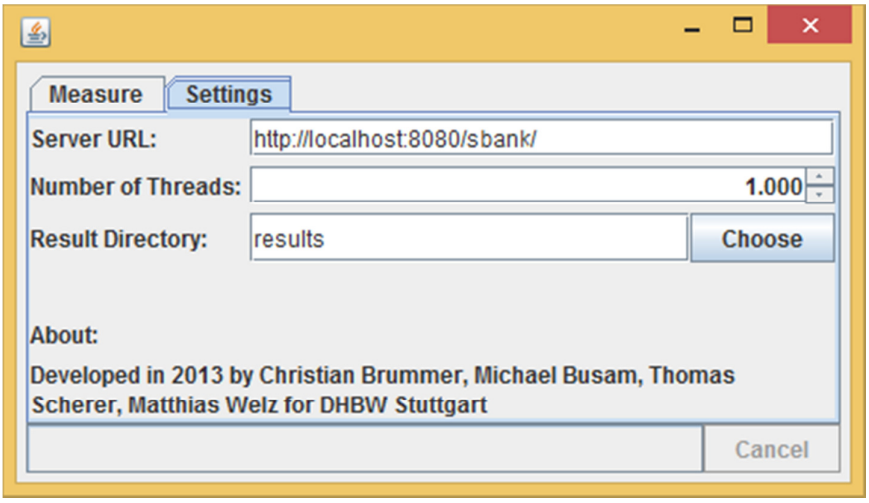


Abbildung 21: Einstellungen des Benchmarktools

Das Benchmarktool stellt das Gegenstück zu den beiden Servern dar. Es ermöglicht, eine frei wählbare Anzahl an Anfragen durch eine simulierte Anzahl an Klienten an den Server zu senden.

Die Benutzeroberfläche ist in zwei Reiter eingeteilt: Messung und Einstellungen.

Im Reiter Messung („Measure“ – siehe **Error! Reference source not found.**) kann zwischen zwei verschiedenen Testarten gewählt werden und die Zusammensetzung der Anfragen festgelegt werden. Die Testart „Specific Number of Rerquests“ setzt dabei eine vorbestimmte Anzahl an den Server ab; die Testart „Timed Test“ setzt über einen wählbaren Zeitraum Anfragen in einer einstellbaren Frequenz ab.

Die Anfragen werden dabei zufällig generiert, wobei die prozentuale Verteilung ebenfalls eingestellt werden kann.

In den Einstellungen (siehe **Error! Reference source not found.**) kann sowohl die Serveradresse festgelegt werden, als auch die Anzahl der Threads, die der Client zur Performancemessung verwendet. Dabei entspricht dieser Wert der Anzahl der Klienten, welche der Testclient simuliert: Maximal diese Anzahl an Anfragen findet gleichzeitig statt.

Für jede Messung legt der Testclient eine CSV -Datei mit den Messergebnissen im gewählten Verzeichnis an. Dabei wird zu jeder Anfrage protokolliert, um welche Art der Anfrage es sich handelt und wie lange die Abarbeitung der Anfrage gedauert hat. Falls es zu einer Exception kommen sollte, wird diese ebenfalls geloggt. CSV wurde als Dateiformat gewählt, da sich dieses Format zwecks weiteren Untersuchungen leicht nach Excel importieren lässt und dabei einfach unter Java erzeugt werden kann.

8.8 **Probleme durch Datenbankanbindung**

Im Zuge erster Tests während der Entwicklung stellte sich heraus, dass bereits die einfache Form der Datenbankanbindung das Messergebnis zusehends beeinflusste: So wurde beispielsweise eine hohe Prozessorauslastung durch den Datenbankprozess beobachtet. Zudem traten gelegentlich Deadlocks auf.

Der Realismusgrad der angebotenen Servermethoden wirkte ebenfalls negativ auf die Messergebnisse: Ein Großteil der vom Testclient gesendeten Anfragen wurden abgelehnt, da die entsprechenden Konten nicht ausreichend gedeckt waren oder sonstige ungültige Eingaben gesendet wurden. Dieses Problem hätte sich nur mit sehr viel Mehraufwand lösen lassen.

Da das Ziel dieser Arbeit jedoch in der Analyse und im Vergleich verschiedener Webservertechnologien liegen soll, entschlossen sich die Autoren aus diesen Gründen schließlich dazu, die Datenbankanbindung komplett zu deaktivieren und stattdessen vollständig mit Wartezeiten zu arbeiten.

Für die folgenden Messungen wurde daher die Implementierung der Servermethoden komplett durch Dummy-Implementierungen ersetzt, die neben der Wartezeit lediglich die Antwortnachrichten entsprechend mit statischem Inhalt füllen. Der alte Code inklusiver der Datenbankanbindung findet sich jedoch weiterhin im GIT-Repository des Projektes.

8.9 **Schwierigkeiten bei der Entwicklung**

Die größte Schwierigkeit bei der Entwicklung stellte die korrekte Konfiguration und Einbindung sämtlicher externer Technologien dar: Angefangen mit Spring-MVC und Spring-Data benötigt jedes Framework seine eigene, sehr spezielle Konfiguration, bis es fehlerfrei läuft. Dokumentation hierzu und insbesondere Unterstützung bei Fehlern ist häufig sehr vage, veraltet oder nur sehr schwer zu finden. Auch der korrekte Buildprozess mittels Maven eröffnete neue Herausforderungen und Probleme.

Bei vert.x setzte sich die Problematik fehlender oder veralteter Dokumentation fort. Viele Ideen und Konzepte sind nur sehr schlecht dokumentiert oder werden in der Dokumentation nur grob umrissen. Insbesondere Beispielprojekte zu erweiterten Features ließen sich nicht finden. Hinzu kommt, dass sich nur sehr schwer beurteilen ließ, wie existierende Konzepte am besten nach vert.x übertragen werden können. Von den Autoren wurde letztlich ein Ansatz gewählt, der sich unter Beibehaltung von möglichst viel Code realisieren ließ (Verwendung der Service-Klassen in Worker-Verticles).

9 Ergebnisse (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

9.1 Abarbeitungszeit als leistungsbestimmender Faktor

Einen Vert.x- und einen Apache Tomcat 7 Server zu vergleichen stellt einige Herausforderungen. Der Tomcat 7 ist standardmäßig auf 200 Threads beschränkt, kann theoretisch aber beliebig skaliert werden – allerdings nur im Voraus.[132] Der Vert.x Server hingegen belegt je Eventschleife bei Ausführung eines Verticles einen Thread, und instantiiert davon sinnvollerweise eine je Prozessorkern. Die Workerthreads werden in einem Hintergrundpool gehalten und können bei Bedarf dynamisch genutzt werden.

Der Aufbau der Versuche und deren Durchführung hängen in erster Linie von der Zielsetzung ab. Dazu muss zunächst definiert werden, welcher Faktor als leistungsbestimmendes Merkmal der Servertechnologien angesehen wird und deshalb verglichen werden soll.

Bei Client-Server-Beziehungen im Webumfeld ist aus Klientensicht vor allem die Reaktionsfähigkeit und -zeit des Servers der sichtbare und bestimmende Leistungsfaktor, und für Unternehmen ist eine Optimierung dieser Faktoren zur Erhöhung der Kundenzufriedenheit bestimmend.

Als guter Bestimmungsfaktor für die Gesamtleistungsfähigkeit kann somit sowohl die gesamte Abarbeitungsdauer aller gleichzeitig ankommenden Requests dienen, als auch die Aufteilung auf die einzelnen Requestarten. Sobald keine weiteren Threads mehr zur Abarbeitung ankommender Requests zur Verfügung stehen, werden weitere eintreffende Requests im Stack vorgehalten, wodurch Verzögerungen entstehen. Je stärker also die Belastung des Webservers, desto länger werden die Antwortzeiten an den Client. Einerseits kann die Vergleichbarkeit der Servertechnologien der Servertechnologien beeinträchtigt werden, wenn zum Vergleich des synchronen Tomcat 7 mit dem asynchronen Vert.x die Antwortzeit bei der Abarbeitung eines Requests oder einer bestimmten Art von Requests, im Bankbeispiel etwa zur Anlage eines neuen Kunden, herangezogen wird, da Verzögerungen bei der Abarbeitung nur bei bestimmten Requests, etwa besonders ein- oder ausgabeintensiven, auftreten könnten. Andererseits können gerade auch Verzögerungen bei bestimmten Requesttypen entscheidend sein, da sie die Leistungserfahrung der Nutzer besonders beeinträchtigen. Deshalb ermöglicht der Performancemessungsclient die Messung der gesamte Abarbeitungsdauer aller Requests, aufgeteilt auf die sieben Requestarten.

[132] Vgl. o. A. (o. J.a)

Für alle Versuche wird die gesamte Bearbeitungszeit die abhängige, leistungsbestimmende Variable sein. Die Frage, in Abhängigkeit welcher Faktoren man die Antwortzeiten nun vergleichen will, stellt sich als komplex dar, insbesondere da nur begrenzte Ressourcen und Zeit für das Gesamtprojekt zur Verfügung stehen.

9.2 Messszenarien

9.2.1 Einflussfaktoren und Annahmen

Welche Messszenarien sollen also gewählt werden, um valide, vergleichbare Ergebnisse zu erhalten? Bei der Definition des Aufbaus der verschiedenen Versuche gilt es zunächst, eine unabhängige Variable zu wählen. Danach müssen alle anderen Einflussfaktoren festgesetzt werden und gemäß dem Grundsatz der sonst gleichen Bedingungen während einer Versuchsreihe beibehalten werden.

Im Folgenden werden die relevanten Variablen beschrieben und, soweit nicht in den jeweiligen Versuchsreihen abweichend definiert, deren Ausprägung beschrieben. Diese sind im Einzelnen:

1) Gesamtzahl gleichzeitiger Requests der Clients (Anzahl der Clients)
2) Gesamtzahl der abzuarbeitenden Requests
3) Dauer der Abarbeitung der einzelnen Requesttypen
4) Verhältnis der verschiedenen Requesttypen zueinander
5) Hardware von Client und Server
6) Anzahl der verfügbaren Prozessorkerne des Webservers
7) Anzahl der verfügbaren Threads des Webservers
8) Konfiguration des Tomcat 7 HTTP Connectors
9) Konfiguration der Vert.x Threadverwaltung

Eine aussagekräftige **(1) Gesamtzahl gleichzeitiger Requests der Clients** entspricht der Anzahl in der Realität der Anzahl der Clients, die gleichzeitig mit jeweils einem Request auf den Server zugreifen wollen. Bis zu einer bestimmten Anzahl von gleichzeitigen Anfragen genügen Threadzahl und Rechenleistung der verwendeten Hardware sowohl bei Einsatz des Vert.x Webservers, als auch beim Tomcat 7 für eine unproblematische, schnelle

Abarbeitung. Dies sollte insbesondere dann der Fall sein, wenn die Anzahl gleichzeitiger Requests kleiner ist als die Zahl der verfügbaren Threads des Webservers.

Ebenso bedeutend für die gesamte Prozessierungszeit ist die **(2) Gesamtzahl der abzuarbeitenden Requests**. Über eine Änderung der Requestanzahl lässt sich die Dauer der Last auf die Webserver bestimmen.

Die **(3) Dauer der Abarbeitung der einzelnen Requesttypen** ist ein wichtiger Faktor, da Vorteile der asynchronen Verarbeitung erwartungsgemäß besonders beim Auftreten von langlaufenden, ein- und ausgabeintensiven Requests sichtbar werden, da im Gegensatz zur synchronen Verarbeitung die Blockierung der Eventschleife durch die Zuweisung an Workerthreads verhindert wird.

In einer früheren Version der Beispielimplementierung konnte die Dauer der Abarbeitung der Requests für die langlaufenden Requests jeweils als Mindestangabe in einer Konfigurationsdatei festgelegt werden, die der kurzlaufenden entsprach der tatsächlichen Abarbeitungsdauer. Die Abarbeitung der langlaufenden Anfragen erfolgte – ähnlich der Realität – direkt auf dem angebundenen Datenbankserver. Um auszuschließen, dass die Messergebnisse durch eine verzögerte, über die festgelegte Mindestdauer hinausgehende Antwort des Datenbankservers verfälscht werden, wird in der aktuellen Version die Abarbeitungsdauer aller Requests programmatisch festgelegt. Dabei wird in allen Versuchsreihen nur die waitTime bestimmt, um ein- und ausgabeintensive Aufgaben zu simulieren, die die CPU kaum belasten. Die busyWaitTime zur Simulation CPU-intensiver Anfragen wird nicht gesetzt, da bei prozessorintensiven Aufgaben bei keiner der Servertechnologien ein Vorteil zu erwarten ist. Im Messszenario werden folgende Werte für die WaitTime verwendet (in Millisekunden):

Requesttyp	**WaitTime in ms**
Kurzlaufende Requests	
AccountWorker	25
CustomerWorker	25
DepositWorker	15
DrawWorker	35
StatementWorker	100
TransferWorker	50
Langlaufende Requests	
NewCustomerWorker	20.000
OpenAccountWorker	5.000

Tabelle 9: WaitTime der verschiedenen Anfragearten

Darüberhinaus ist das **(4) Verhältnis der verschiedenen Requesttypen zueinander** entscheidend. Wie bereits ausgeführt, besteht die Stärke asynchroner Technologien in der effizienteren Abarbeitung aller, insbesondere aber der kurzlaufenden Requests bei einer gewissen Zahl langlaufender, ein- und ausgabeintensiver Requests, sodass bei einem hohen Anteil langlaufender Anfragen unter allen gleichzeitig eintreffenden Requests ein stärkerer Vorteil des Vert.x Webservers zu erwarten ist.

Um die Realität möglichst genau abzubilden, werden jedoch zwei Arten von langlaufenden und sechs Arten von kurzlaufenden Requests simuliert. Während das Verhältnis der kurz- zu den langlaufenden Requests sich ändert, bleibt zur Verminderung der Komplexität das Verhältnis der kurz- und langlaufenden Requesttypen untereinander in allen Szenarien jeweils konstant. Die festgelegten Verhältnisse entsprechen jedoch nur Wahrscheinlichkeiten für das Auftreten eines entsprechenden Requests, wodurch die tatsächlichen Verhältnisse zu einem gewissen Maße dem Zufall unterworfen sind. Deshalb werden in jeder Versuchsausprägung drei Messungen durchgeführt, deren Mittelwert als Vergleichswert herangezogen wird.

Die **(5) Hardware von Client und Server** muss bei beiden Versuchsaufbauen gleich sein, um die Leistungstests ceteris paribus durchführen zu können. Client und Server auf jeweils eigener Hardware auszuführen würde einerseits die Prozessor- und Speichernutzung des Clients als verfälschenden Einflussfaktor auf die Antwortzeit ausschließen, würde jedoch die Leistungsfähigkeit des Netzwerks als weitere Unbekannte hinzufügen. Für die Tests in dieser Arbeit wurde auf eine physische Trennung von Client und Server verzichtet.

Als Hardware wurde für alle Tests ein Thinkpad T420 gewählt:

Betriebssystem	Windows 7 Professional 64-bit
Prozessor	Intel Core i5-2520M, 2 physikalische und 2 virtuelle Kerne, 2,5 GHz
Arbeitsspeicher	4 GB

Tabelle 10: Merkmale der Testhardware

Für einen Webserver stellt die gewählte Hardware eine verhältnismäßig schwache Ausstattung dar, was im Wesentlichen mangelnden Ressourcen geschuldet ist. Dies hat jedoch den Vorteil, bereits bei relativ geringen Request- und Threadzahlen eine hohe Belastung des Servers zu erreichen.

Des Weiteren muss die **(6) Anzahl der verfügbaren Prozessorkerne der Webserver** festgelegt werden und übereinstimmen. Indem beide Webserver nur einen Kern zur Verfügung haben, kann verhindert werden, dass die verschiedene Allokation der Rechenleistung auf die Threads die Ergebnisse beeinträchtigt. Die Serverprozesse laufen

dabei nie gleichzeitig und werden beide Kern 3 der CPU zugewiesen. Zudem wird darauf geachtet, dass zum Ausführungszeitpunkt neben dem Performancemessungsclient und dem jeweiligen Server keine weiteren Anwendungen ausgeführt werden, um Schwankungen in der Ressourcenverfügbarkeit zu vermeiden.

Zudem muss eine Festlegung hinsichtlich der **(7) Anzahl der verfügbaren Threads des Webservers** getroffen werden. Vert.x instantiiert für jede verfügbare CPU eine Eventschleife und legt standardmäßig einen Hintergrundpool von 20 Workerthreads an.[133] Da in den meisten Versuchsreihen Vert.x alle Anfragen in Workerthreads abarbeitet und die Eventschleife nicht zur Unterstützung genutzt wird, muss zur Vergleichbarkeit auch die Threadzahl des Tomcat auf 20 beschränkt werden. Die Gesamtthreadzahl kann durch Setzen des Attributs *maxThreads* recht einfach beschränkt werden. Standardmäßig hält der Tomcat jedoch nur 10 Threads bereit. Um die Threadinitialisierung als Quelle von Verfälschungen auszuschließen wird das Attribut *minSpareThreads* auf 20 gesetzt (vergleiche Listing 4: Konfiguration des Tomcat HTTP Connectors).[134]

Durch die Änderung der **(8) Konfiguration des Tomcat 7 HTTP Connectors** können weitere verfälschende Faktoren ausgeschlossen werden. Es soll in allen Fällen die tatsächliche Abarbeitungszeit aller eintreffenden Requests als Vergleichswerkzeug dienen, somit muss ausgeschlossen werden, dass Anfragen von vornherein abgewiesen werden oder nach einiger Wartezeit die Verbindung wegen eines Timeout geschlossen wird. Dies wird durch Setzen der Attribute *AcceptCount* auf die maximale Zahl gleichzeitiger Requests von 400 und *connectionTimeout* auf -1 (unendlich) erreicht.[135]

```
<Connector
      connectionTimeout="-1"
      port="8080"
      maxThreads="20"
      protocol="HTTP/1.1"
      redirectPort="8443"
      acceptCount="10000"
      minSpareThreads = "20"
     />
```

Listing 4: Konfiguration des Tomcat HTTP Connectors

[133] www.cubrid.org/blog/dev-platform/understanding-vertx-architecture-part-2/
[134] http://tomcat.apache.org/tomcat-5.5-doc/config/http.html
[135] http://tomcat.apache.org/tomcat-7.0-doc/config/http.html

Zuletzt besteht schließlich die Möglichkeit, durch die **(9) Konfiguration der Vert.x Threadverwaltung** das Ergebnis maßgeblich zu beeinflussen. Dabei spielt insbesondere eine Rolle, ob ein Request als „normales“ Verticle oder als Workerverticle ausgeführt wird und ob der Pool an Workerthreads den Workerverticles dezidiert zugeteilt wird oder nicht.

Aufgrund der Fülle an resultierenden Varianten und Szenarien ist es im Rahmen des Projekts nicht möglich, die Leistungsfähigkeit der Webserver auf die Änderung jeder einzelnen der Variablen hin zu untersuchen. Es muss eine sinnvolle Auswahl getroffen werden. Im Folgenden werden die tatsächlich durchgeführten Versuche beschrieben.

9.2.2 Versuchsreihe 1: Abhängigkeit von der Anzahl gleichzeitiger Requests

In dieser Versuchsreihe soll die Abarbeitungszeit in Abhängigkeit von der Anzahl gleichzeitig eintreffender Requests bestimmt werden. Eine Fragestellung, die dieser Versuchsreihe zu Grunde liegt, ist, ob Vert.x Leistungsvorteile erzielen kann, wenn die Eigenschaft der Ereignissteuerung nicht voll ausgenutzt wird und alle Requestarten als WorkerVerticles in den Workerthreads abgearbeitet werden.

Das erwartete Ergebnis für diesen Fall ist, dass sich kein signifikanter Unterschied zwischen den Ergebnissen von Tomcat 7 und Vert.x zeigt, dass bei beiden Frameworks die Abarbeitungszeit jedoch mit zunehmenden Requestzahlen überproportional ansteigt. Gäbe es einen signifikanten Unterschied, würde dies darauf hindeuten, dass eine der Technologien die verfügbaren (identischen) Systemressourcen besser ausnutzt, etwa aufgrund der unterschiedlichen Architektur.

Die oben beschriebenen Einflussgrößen werden dazu folgendermaßen kalibriert:

Die **(1) Gesamtzahl gleichzeitiger Requests der Clients (Anzahl der Clients)** soll groß genug sein, damit immer alle Requests gleichzeitig zu Beginn ankommen und orientiert sich damit an **(2), der Gesamtzahl der abzuarbeitenden Requests**. Für die verwendete Konstellation von Hardware und verfügbaren Prozessorkernen sowie Threads begannen die Antwortzeiten ab 100 Requests überproportional zu steigen. Als kritischer Bereich wurde folglich der Bereich von 100 bis 400 gleichzeitigen Requests in Schritten von 100 Requests untersucht. Zusätzlich wurde zur Absicherung der Messwerte nach unten zusätzlich für 50 gleichzeitige Requests gemessen.

Zudem sollen zwei **Verhältnisse der verschiedenen Requesttypen zueinander (4)** untersucht werden, erstens mit einem sehr hohen Anteil an kurzlaufenden Requests von 90% zu 10% für die langlaufenden Requests und zweitens mit erhöhtem Anteil an langlaufenden Requests von 40% zu 60% kurzlaufenden Anfragen.

Die **(9) Konfiguration der Vert.x Threadverwaltung** wird im Vergleich zu Standardkonfiguration nicht verändert, sodass allen Anfragen zur Abarbeitung alle 20 Workerthreads zur Verfügung stehen, aber keine Abarbeitung in der Eventschleife direkt erfolt.

9.2.3 Versuchsreihe 2: Abhängigkeit von der Anzahl der Klienten

Die Leistung soll hier in Abhängigkeit von der Anzahl der gleichzeitigen Klienten erfolgen. Als Ergebnis wird erwartet, dass die Abarbeitungsdauer im Bereich von 20 Klienten optimal ist, während sie bei weniger Klienten leicht zunimmt, da die Server keinen Stack an bereits eingegangenen Anfragen mehr halten können, und bei mehr als 20 Klienten ähnlich dem Verhalten in Versuchsreihe 1 überproportional zunimmt, da lange Wartezeiten entstehen.

Weitere Faktoren, die in diesem Szenario berücksichtigt werden müssen:

Die **(1) Gesamtzahl gleichzeitiger Requests der Clients (Anzahl der Clients)** soll mit einem Wert kleiner als 20 beginnen, um den Fall zu simulieren, dass es weniger gleichzeitige Klienten als verfügbare (Worker-)Threads gibt. Es werden sieben Versuche durchgeführt mit Clientzahlen von 10, 20, 30, 40, 50, 100 und 200.

In allen Versuchen wird die **(2) Gesamtzahl der abzuarbeitenden Requests** 500 betragen, um eine gewisse Belastung zu erzeugen, aber gleichzeitig die Testdauern nicht unnötig stark zu erhöhen.

Das **(4) Verhältnis der verschiedenen Requesttypen** zueinander soll in allen Fällen 90% für die kurzen Requests und 10% für die langen Requests betragen.

Auch in diesem Szenario werden in der **(9) Konfiguration der Vert.x Threadverwaltung** zunächst keine der besonderen Eigenschaften von Vert.x genutzt, sodass alle Requests in den Workerthreads abgearbeitet werden und diese auch keinen speziellen Requesttypen zugewiesen werden.

9.2.3.1 Abwandlung zu Versuchsreihe 2: Zuteilung des Vert.x Workerthreadpools

Diese Versuchsreihe stellt eine Abwandlung von Versuchsreihe 2 in der **(9) Konfiguration der Vert.x Threadverwaltung** dar. Indem der Hintergrundpool von 20 Workerthreads den acht Requesttypen nun eindeutig zugeordnet wird, soll vermieden werden, dass langlaufende Requests die kurzlaufenden weiter blockieren können. Die Aufteilung der Workerthreads folgt dem Schema in Tabelle 11.

Berechnung der Zuweisung der Workerthreads							
Requestart	**Ausarbei-tungszeit**	**Anteile**	**"Gewicht"**	**Anteil am Gesamtgewicht**	**1 Thread je kurzem Typ**	**Aufteilung des Rests auf die langen Typen**	**Verteilung**
DepositWorker	15	15	225	0,03	1		**1**
AccountWorker	25	15	375	0,06	1		**1**
CustomerWorker	25	15	375	0,06	1		**1**
DrawWorker	35	15	525	0,08	1		**1**
TransferWorker	50	15	750	0,12	1		**1**
StatementWorker	100	15	1500	0,23	1		**1**
OpenAccountWorker	5000	5	25000	3,88	14	2,8	**3**
NewCustomerWorker	20000	5	100000	15,53		11,2	**11**
			128750				

Tabelle 11: Ermittlung der Threadzuteilung

Zunächst wird aus dem Produkt von relativer Häufigkeit und jeweiliger Abarbeitungszeit ein „Gewicht“ eines jeden Requesttyps ermittelt sowie ein Anteil am Gesamtgewicht, multipliziert mit der Gesamtzahl an Workerthreads (20). Daraus ergibt sich, dass jedem kurzlaufenden Requesttyp nur der Bruchteil eines Threads zustünde, sodass jedem letztlich ein Thread zugewiesen wird. Die restlichen 14 Threads werden wiederum nach dem Gewichten auf die beiden langlaufenden Requesttypen verteilt, sodass sich die Threadverteilung in der letzten Spalter ergibt.

Das erwartete Ergebnis ist eine deutliche Verbesserung der Abarbeitungszeiten insbesondere der kurzen Requests, da diese nun auf dezidierten Threads schnell abgearbeitet werden können und nicht mehr von langlaufenden Threads blockiert werden können.

Die neuen Tests werden nur für Vert.x durchgeführt. Es sind keine weiteren Faktoren zu bestimmen.

9.2.3.2 Abwandlung zu Versuchsreihe 2: Abarbeitung einer nichtblockierenden Requestart in der Eventschleife

Eine weitere Abwandlung von Versuchsreihe 2, wiederum nur für Vert.x, basiert ebenfalls nur auf einer Änderung der **(9) Konfiguration der Vert.x Threadverwaltung**. In diesem Fall soll die Eigenschaft der Ereignissteuerung mit nicht-blockierender Eventschleife vollumfänglich ausgenutzt werden, indem der kürzeste (und somit auch am kürzesten blockierende) Request, die Deposit Action (Kontostand anzeigen), direkt in der Eventschleife abgearbeitet werden soll. Diese Anfragenart wird deshalb nicht mehr als WorkerVerticle deklariert. Alle weiteren Requests teilen sich analog zu Versuchsreihe 2 die Workerthreads, es findet keine Zuweisung wie in Abwandlung 1 statt.

Als Ergebnis wird erwartet, dass die Abarbeitung der Deposit Action stark verbessert wird, da sie mit der Eventschleife immer einen freien, nicht blockierten Thread zur Verfügung haben sollte, während sich die Abarbeitungszeiten aller anderen Requestarten kaum ändern. Tendenziell ist eher eine Verschlechterung zu erwarten, da die Eventschleife in ihrer Hauptaufgabe, der Eventdelegation, beeinträchtigt werden könnte.

9.3 **Messergebnisse**

Um bei verschiedenen Anteilen der verschiedenen Anfragearten an der Gesamtheit der Anfragen Fehler in der Durchschnittsberechnung aufgrund der unterschiedlichen, programmatisch festgelegten Ausführungszeiten zu vermeiden, zeigen alle folgenden Diagramme die Mittelwerte der gesamten Abarbeitungszeit abzüglich der jeweiligen vorprogrammierten Ausführungszeit, also die reine zusätzliche Latenz durch die Serveraktivität. Diese wird in der Folge auch als Abarbeitungsdauer oder -zeit bezeichnet.

Die Messergebnisse werden mit Hilfe eines einheitlichen Diagrammtyps verdeutlicht. In diesem wird die jeweils die mittlere Abarbeitungszeit aller kurzen Anfragen, aller langen Anfragen und der Anfragen vom Typ “Kontostand anzeigen“ (Deposit Action, der kürzeste aller Requests mit 15ms) mit jeweils einem Balken illustriert. Die durchschnittliche Abarbeitungsdauer aller Requests erscheint als grauer Balken. Entweder es werden tatsächliche Abarbeitungsdauern oder die Differenz zwischen zwei Messungen gezeigt. In einem zweiten zum Zur Durchschnittsberechnung wird das arithmetische Mittel verwendet.

9.3.1 Ergebnisse zu Versuchsreihe 1

Abbildung 22 veranschaulicht die Abarbeitungszeiten des Tomcat 7. Erwartungsgemäß steigen die Zeiten bei steigender Anfragenzahl stark und überproportional bis auf über sieben Sekunden an, während 50 gleichzeitige Anfragen noch mit geringer Latenz bearbeitet werden können. Zwischen den verschiedenen Anfragetypen zeigen sich kaum Unterschiede, die Requests werden also nicht priorisiert und „diskriminierungsfrei“ in chronologischer Reihenfolge an die freiwerdenden Threads verteilt.

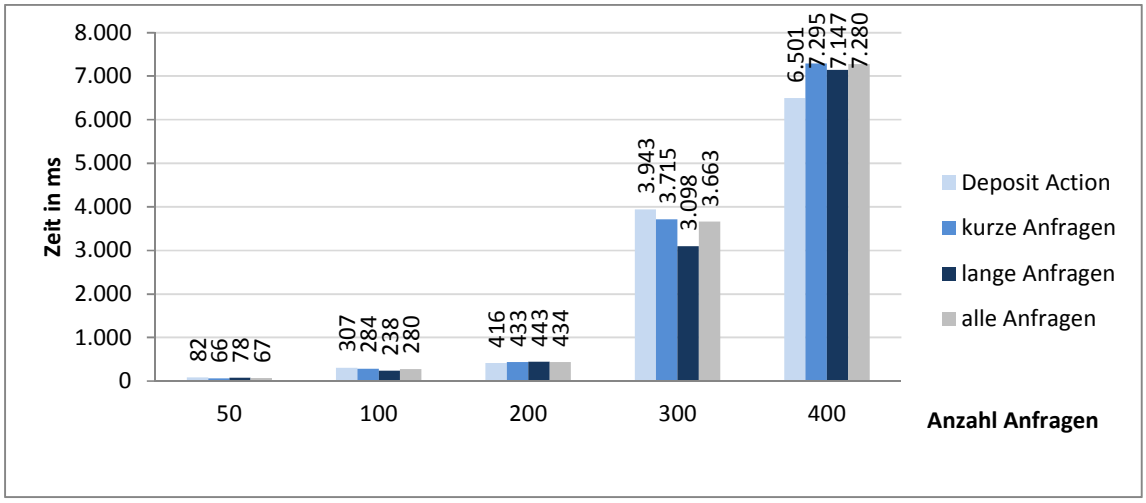


Abbildung 22: Durchschnittliche Abarbeitunszeiten Tomcat 7 (Versuchsreihe 1, 90% kurzlaufende Requests)

Vert.x zeigt ein ähnliches Verhalten. Abbildung 23 vergleicht die Abarbeitungszeiten von Vert.x im Vergleich mit Tomcat 7. Insbesondere bei großen Requestzahlen zeigt sich ein leichter Vorteil für Vert.x.

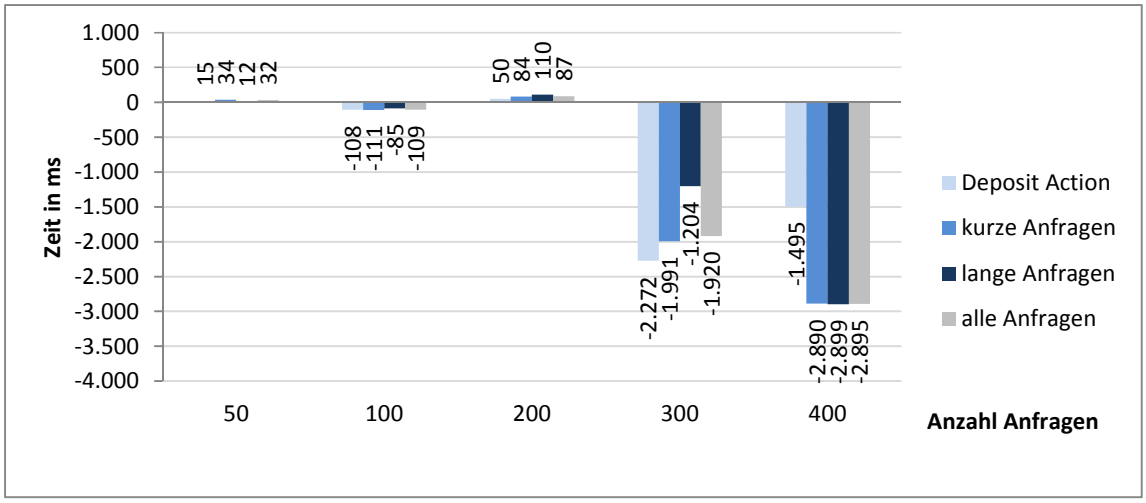


Abbildung 23: Differenz Abarbeitungszeit Vert.x zu Tomcat 7 (Versuchsreihe 1, 90% kurzlaufende Requests)

Ob dafür aber wirklich Eigenschaften des Frameworks selbst ursächlich sind, kann leider nicht abschließend geklärt werden, da ab etwa 300 gleichzeitigen Anfragen Tomcat die Abarbeitung trotz anderslautender Konfiguration eingestellt hat und Anfragen abgewiesen hat. Die Ursache dafür kann in einer zu schwachen Hardware liegen, konnte aber nicht abschließend geklärt werden, weshalb insbesondere der letzte Messpunkt kaum verwertbar ist.

Bei einem höheren Anteil längerer Requests von kumuliert 40% ergibt sich bei deutlich höheren Abarbeitungszeiten ein ähnliches Muster und es lässt sich kein Vorteil für eine der Technologien erkennen. Der Tomcat 7 beginnt bei diesem Szenario bereits ab 250 Requests Anfragen abzuweisen.

9.3.2 Ergebnisse zu Versuchsreihe 2

Bei Versuchsreihe 2 ergibt sich ein ähnliches Bild wie bei Versuchsreihe 1. Abbildung 24 zeigt die durchschnittlichen Abarbeitungszeiten des Tomcat 7, die mit zunehmender Anzahl an Klienten überproportional ansteigen, jedoch auch für alle Requestarten ähnlich sind.

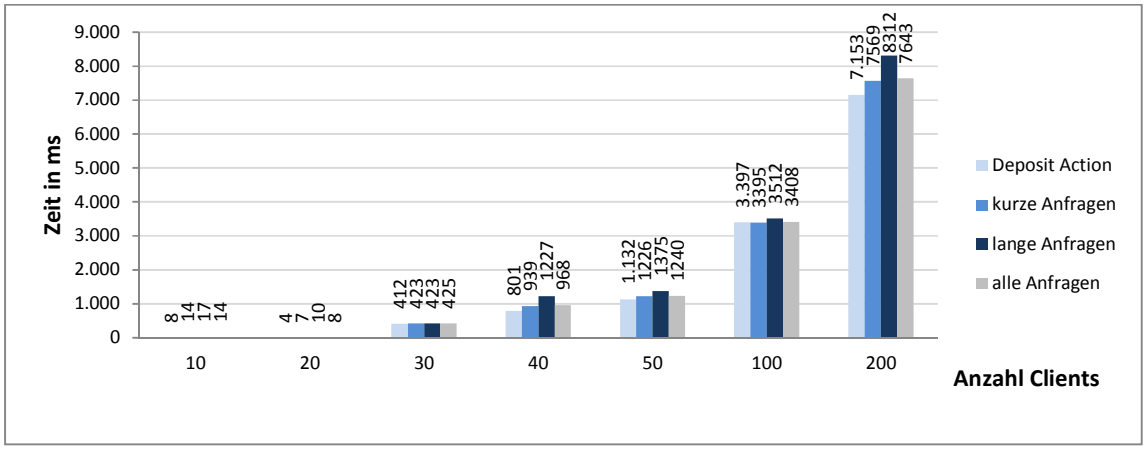


Abbildung 24: Durchschnittliche Abarbeitunszeiten Tomcat 7 (Versuchsreihe 2)

Abbildung 25 zieht wiederum den Vergleich zu Vert.x und zeigt auch bei Vert.x ähnliche Zeiten für kurz- und langlaufende Requests. Vert.x ist auch hier meist gleich auf mit Tomcat 7, zeigt bei hohen Clientzahlen – und damit einer hohen Anzahl gleichzeitiger Requests – jedoch wiederum leichte Zeitersparnisse von etwa zwei Sekunden im Mittel bei allen Requesttypen. In diesem Fall sind alle Testdaten vollständig.

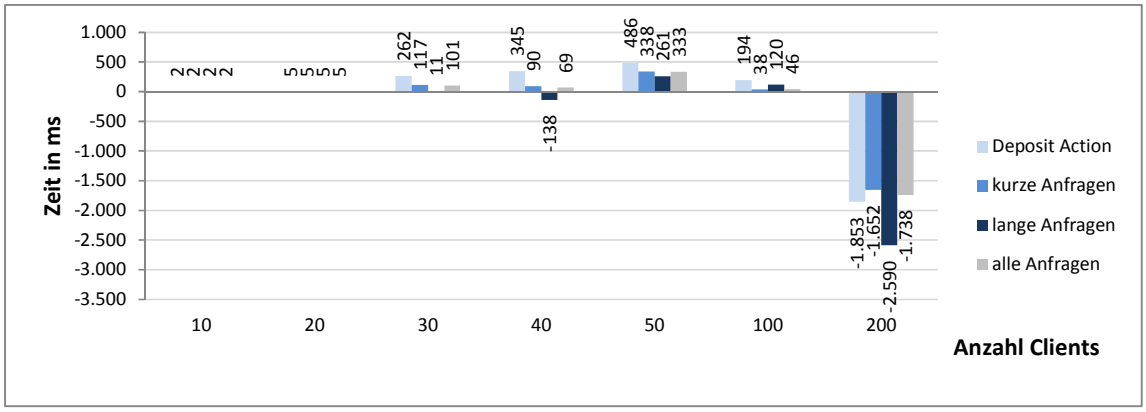


Abbildung 25: Differenz Abarbeitungszeit Vert.x zu Tomcat 7 (Versuchsreihe 2)

9.3.3 Ergebnisse zu Abwandlung 2.1

Abbildung 26 zeigt zunächst, dass die Aufteilung der Workerthreads auf die einzelnen Requestarten gemäß der unterschiedlichen Gewichte zu einer erheblichen Verbesserung der durchschnittlichen Abarbeitungszeit der kürzesten Requestart, der Deposit Action, führt. Hat sich in Versuchsreihe 2 noch kaum ein Unterschied zum Tomcat gezeigt, schafft es Vert.x nun, die Deposit Action auch bei einer hohen Zahl von Clients in Sekundenbruchteilen abzuarbeiten.

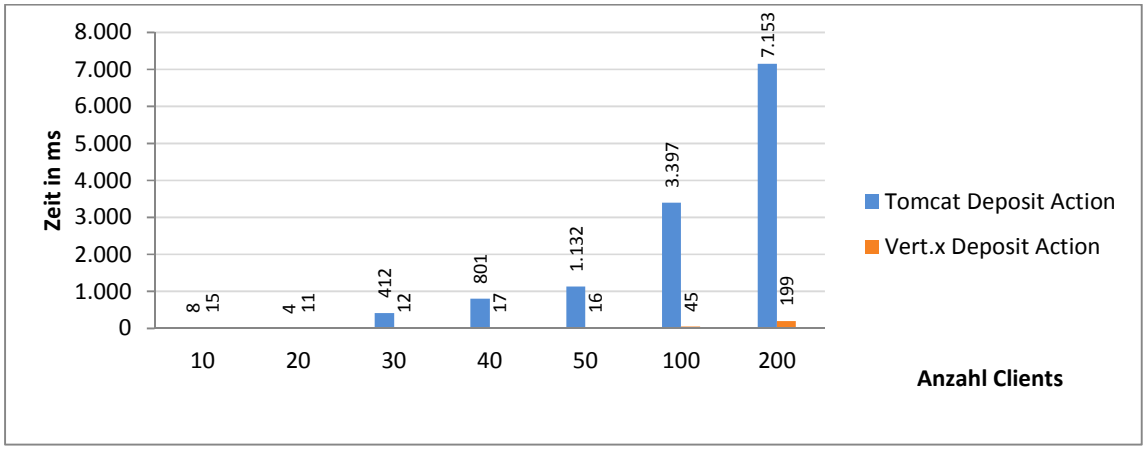


Abbildung 26: Vergleich der Abarbeitungszeit für Deposit Action (Tomcat 7 aus Versuchsreihe 2; Vert.x aus Versuchsreihe 2.1)

Abbildung 27 enthält wiederum den Vergleich der Ergebnisse von Vert.x mit zugeordneten Workerthreads mit den Ergebnissen des Tomcat aus Versuchsreihe 2. Es wird deutlich, dass die durchweg deutlich schnellere Abarbeitung aller kurzen Requests zunächst stark zu

Lasten der langen Requests geht, auch wenn sich dieser Nachteil bei hohen Klientenzahlen zunehmend abschwächt. Aufgrund des hohen Anteils an kurzen Requests zeigt sich jedoch im Mittel aller Requestarten ein eindeutiger Vorteil für die Vert.x Konfiguration aus Szenario 2.1.

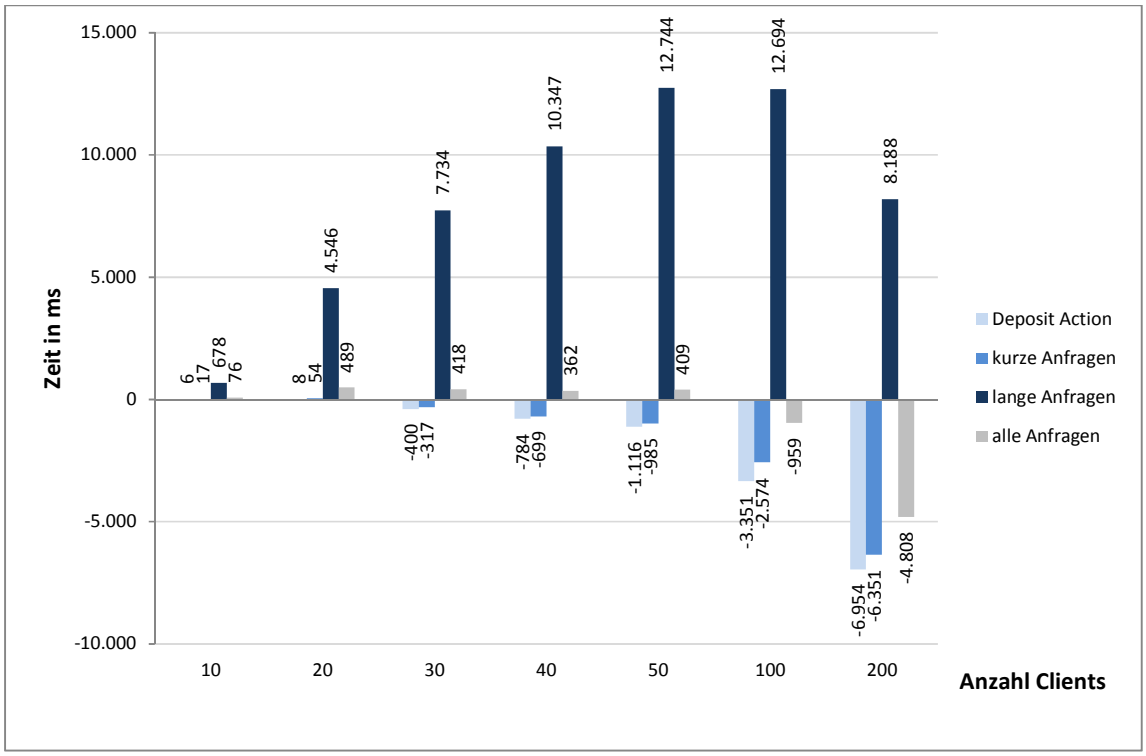


Abbildung 27: Differenz Abarbeitungszeit Vert.x zu Tomcat 7 (Tomcat 7 aus Versuchsreihe 2; Vert.x aus Versuchsreihe 2.1)

9.3.4 Ergebnisse zu Abwandlung 2.2

Auch, wenn die Deposit Action, anstatt in einem dezidierten Workerthread, direkt in der Eventschleife abgearbeitet wird, zeigt sich ein ähnlich deutlicher Vorteil für Vert.x im Vergleich zum Tomcat 7, wie Abbildung 28 zeigt.

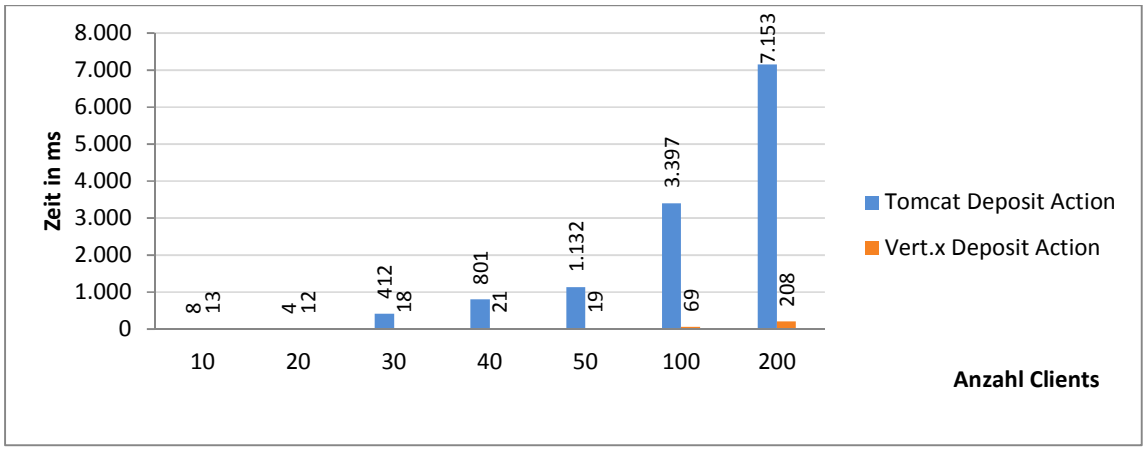


Abbildung 28: Vergleich der Abarbeitungszeit für Deposit Action (Tomcat 7 aus Versuchsreihe 2; Vert.x aus Versuchsreihe 2.2)

Wie Abbildung 29 zeigt, fällt der Vergleich aller Requestsarten mit den Tomcat-Ergebnissen aus Versuchsreihe 2 hier jedoch noch positiver für Vert.x aus, da die starken Verzögerungen bei den langen Requests nicht zu Stande kommen.

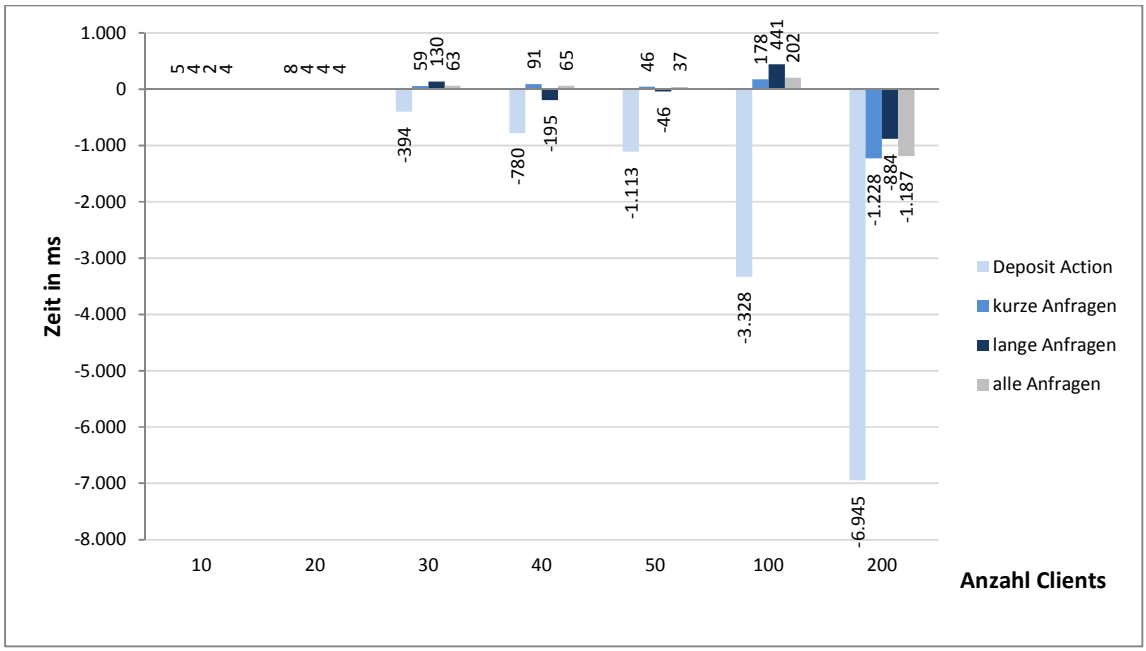


Abbildung 29: Differenz Abarbeitungszeit Vert.x zu Tomcat 7 (Tomcat 7 aus Versuchsreihe 2; Vert.x aus Versuchsreihe 2.2)

Ein letzter Vergleich in Abbildung 30 stellt die Ergebnisse von Vert.x aus den Szenarien 2.2 und 2 gegenüber. Hier zeigen sich insgesamt leichte Verschlechterungen, während die Deposit Action jedoch weiterhin große Verbesserungen erzielen kann.

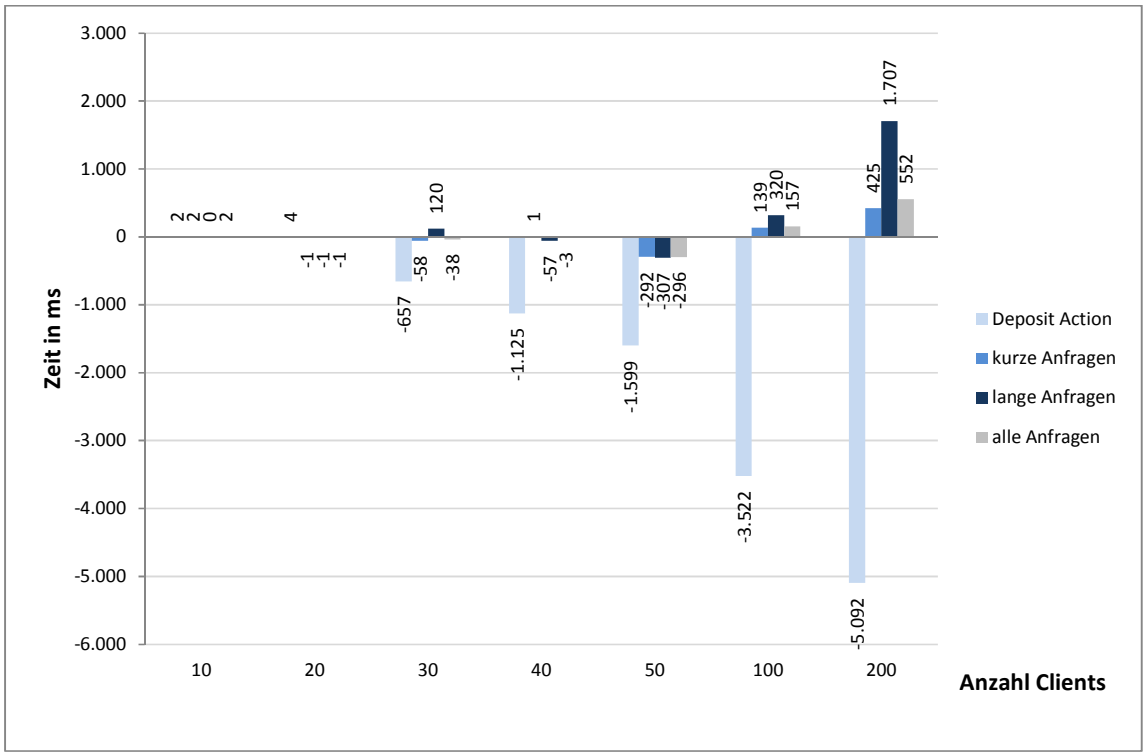


Abbildung 30: Differenz Abarbeitungszeit Vert.x aus Versuchsreihe 2 mit Vert.x aus Versuchsreihe 2.2

9.4 **Interpretation**

Für den Fall, dass Vert.x alle Workerthreads allen Requestarten zur Verfügung stellt und keine Requests in der Eventschleife abarbeitet, konnten keine statistisch signifikanten Vorteile des Vert.x gegenüber Tomcat 7 festgestellt werden, wie Versuchsreihe 1 zeigt. Es scheint also keine der beiden Technologien irgendwelche Vorteile „per se“ zu haben, die auch ohne Einsatz der Ereignissteuerung zum Tragen kommen. Es zeigt sich allenfalls das Erwartbare, dass beide Server ab einer gewissen Zahl von gleichzeitigen Requests in ihren Antwortzeiten deutlich nachlassen. Dass der Tomcat 7 ab einer gewissen Requestzahl begonnen hat, Requests abzuweisen, kann man als leichten Vorteil für Vert.x deuten. Es besteht allerdings die Wahrscheinlichkeit von Konfigurationsfehlern beim Tomcat oder Problemen mit der Hardware, sodass hier kein abschließendes Fazit gezogen werden kann.

Ein ähnlich Ergebnis folgt aus Versuchsreihe 2, wo ebenfalls keines der Frameworks signifikant bessere Ergebnisse erzielt hat.

In den ersten beiden Versuchsreihen wurden die Möglichkeiten des Vert.x und das Konzept der Ereignissteuerung allerdings nicht sinnvoll genutzt.

Die erste Abwandlung von Versuchsreihe 2 zeigt, dass die Möglichkeit, den Hintergrund-Threadpool speziellen Requestarten zuzuweisen, Vert.x bei richtiger Anwendung zu deutlichen Leistungsvorteilen insbesondere bei kurzen Requests verhelfen kann. Dies geht zwar zunächst zu Lasten der langlaufenden Requests, was in der Gesamtbetrachtung allerdings eine untergeordnete Rolle spielt, da die kurzlaufenden Anfragen – wie auch in der Realität – deutlich häufiger auftreten. Für die Mehrheit der Nutzer ergäbe sich bei einer Konfiguration ähnlich Szenario 2.1 also eine deutliche Verbesserung der Nutzungserfahrung. Dieses Modell ist auch gut skalierbar für größere Threadpools, setzt allerdings eine gewisse Berechenbarkeit der Abarbeitungsdauer der Requestarten voraus, damit nicht ein unerwartet langer Request den einzigen Thread für diese Anfrageart blockiert. Einschränkend muss man hinzufügen, dass auch hier die Ereignisgetriebenheit von Vert.x nur eine untergeordnete Rolle gespielt hat und jede synchrone Technologie ähnliche Leistungsverbesserungen erzielen kann, wenn der Threadpool entsprechend aktiv allokiert werden kann. Insbesondere in Abwandlung 2.1 ist es wichtig, die Workerthreads im richtigen Verhältnis auf die Requestarten zuzuteilen. Hier wären mitunter noch weitere Leistungssteigerungen für Vert.x möglich gewesen, wäre die Konfiguration in weiteren Tests optimiert worden.

In der zweiten Abwandlung der zweiten Versuchsreihe konnten nun eindeutig die Leistungsverbesserungen durch das Nutzen der nicht-blockierenden Eventschleife von Vert.x belegt werden. Die Ausführung eines kurzen, minimal blockierenden Requests

(Ausführungsdauer 15ms) direkt in der Eventschleife führt zu deutlichen Leistungsverbesserungen bei der Abarbeitung dieses Requests – ohne die restlichen Requests notwendigerweise einzuschränken, da schließlich entsprechend weniger Requests in den Workerthreads prozessiert werden müssen. Dieses Modell ist in dieser Form nur mit einer asynchronen Technologie möglich und nicht auf den Tomcat übertragbar. Allerdings erfordert es einige Kenntnisse über die Eigenarten der Ablaufzeiten der verschiedenen Typen von Anfragen, da eine einzige außergewöhnlich lang laufende Anfrage – in der Eventschleife abgearbeitet – zu einer Blockade des ganzen Servers führen kann. Vert.x kann diese Problematik durch die Multikern-Unterstützung mit mehreren Verticlen umgehen, was auf andere asynchrone Frameworks wie Node.js jedoch nicht zutrifft.

9.5 **Bewertung der Aussagekraft**

Die in dieser Arbeit durchgeführten Tests können nur einen ersten Eindruck von den Möglichkeiten synchroner und asynchroner Verarbeitung geben und erheben keinen Anspruch auf Vollständigkeit. Es existieren zahlreiche Unsicherheitsfaktoren und viele Testszenarien konnten aufgrund der begrenzten Ressourcen nicht geprüft werden. Dieser Abschnitt soll einen Einblick in mögliche Fehlerquellen und Unsicherheiten bieten, gleichzeitig aber auch aufzeigen, welche weiteren Untersuchungen von Interesse sein könnten.

Eine Unbekannte in der verwendeten Testumgebung ist die Zufälligkeit der tatsächlich gesendeten Anfragen. Die Anteile der verschiedenen Requestarten sind als Wahrscheinlichkeiten implementiert, sodass in einigen wenigen Fällen bei gleichem Testszenario Abweichungen bei der Gesamtabarbeitungsdauer um bis zu Faktor 2 vorkommen können. Diese hohen Gesamtzeiten können schon durch eine unglückliche Kombination von besonders langen Requestarten zu Beginn der Ausführung zu Stande kommen. Jedoch darf der Gesamtabarbeitungsdauer keine allzu große Wichtigkeit beigemessen werden. Was entscheidend ist, sind die durchschnittlichen Latenzzeiten der Requests, die durch einige wenige sehr lang laufende Requests kaum beeinträchtigt werden. Zudem wurde bei den Messergebnissen eine automatische „Outlier-Detection“ durchgeführt, stark abweichende Messwerte, die auf eine besonders ungünstige Veteilung der Requests zurückgeführt werden konnten, wurden in seltenen Fällen also nicht in die Messungen miteinbezogen, um den tatsächlichen Mittelwerten näher zu kommen.

Bezüglich der Technologien basieren alle Testszenarien auf der Standard-Threadzahl von Vert.x, den 20 Workerthreads, auf die die des Tomcat angepasst wurde. Dies könnte eine Benachteiligung des Tomcat darstellen, das dies nur 10% seiner standardgemäßen

Threadzahl entspricht. Da die Messergebnisse insbesondere in den Abänderungen 2.1 und 2.2 jedoch sehr deutlich zugunsten von Vert.x ausgefallen sind, kann die Gesamtaussage schwer angezweifelt werden.

Darüber hinaus konnte nicht abschließend geklärt werden, ob Vert.x in den Tests wirklich nur eine Eventschleife zur Verfügung hatte, da standardmäßig eine Schleife je Prozessorkern instantiiert wird. Wie sich Vert.x in dieser Hinsicht bei virtuellen Prozessorkernen und manueller Zuweisung an einen Kern über den Windows Taskmanager verhält, konnte nicht geklärt werden. Die Ergebnisse der Szenarien 1 und 2 legen jedoch nahe, dass Vert.x keine zusätzlichen Rechenkapazitäten zur Verfügung hatte.

Darüber hinaus wurden in allen Versuchsreihen Client und Server nicht physisch getrennt und wurden auf derselben Hardware ausgeführt. Da für alle Tests jedoch derselbe Client verwendet wurde, sollte dessen Ressourcennutzung beide Server gleichermaßen beeinträchtigt haben, sodass dies keine größeren Auswirkungen auf die Ergebnisse haben sollte. Dennoch ist bei weiteren Tests eine Trennung von Client und Server anzudenken.

10 Fazit und Ausblick (Brummer C./ Busam, M. / Scherer, T. / Welz, M.)

Diese Seminararbeit gibt einen Überblick über die theoretischen Grundlagen von Webservern, synchronen sowie asynchronen Technologien und schlägt die Brücke zu einer realitätsnahen Implementierung einer ideellen Bank, bei der ähnliche Probleme auftreten wie in vielen Unternehmen. Auf den synchronen Webservern treffen zahlreiche Requests zur gleichen Zeit ein und müssen auf einer relativ kleinen Threadzahl bearbeitet werden. Jedoch verzögert sich die Abarbeitung durch einen verhältnismäßig kleinen Anteil an langlaufenden, ein- oder ausgabeintensiven Requests, die Webserverthreads ohne jegliche CPU-Belastung blockieren. Die Unerreichbarkeit oder langsame Reaktionszeit von Webservern kann nicht im Sinne eines auf Kundenzufriedenheit bedachten modernen Unternehmens sein. Eine Lösung dafür können asynchrone Frameworks sein.

Die zugrundeliegende Frage war, ob eine spezielle, recht neue und noch kaum untersuchte asynchrone Technologie – Vert.x – in genau diesen Situationen Leistungsvorteile gegenüber einer traditionellen, synchronen Technologie bietet.

Als Nebenprodukt ist insbesondere der praktische Teil der Arbeit auch die Dokumentation der erfolgreichen Durchführung eines Softwareprojektes. Das Hauptergebnis dieser Arbeit aber ist der Nachweis, dass eine asynchrone Technologie wie Vert.x tatsächlich eine deutlich bessere Leitung als ein klassischer, synchroner Tomcat 7 erbringen kann, wenn sie richtig eingesetzt wird.

Dazu ist es nötig, entweder die definierende Eigenschaft der Asynchronität auszunutzen, und sehr kurze, nicht-blockierende Requests direkt in der Eventschleife abzuarbeiten, oder den Hintergrundpool an Workerthreads, über den Vert.x verfügt, geschickt auf die verschiedenen Requestarten aufzuteilen, um eine Blockade zu verhinden.

Es ist zu erwarten, dass mit weiteren Untersuchungen noch viel tiefgreifendere Erkenntnisse über Vert.x gewonnen werden können. Nicht geklärt werden konnte beispielsweise, ob eine Kombination beider Möglichkeiten noch bessere Ergebnisse und Antwortzeiten ermöglichen kann. Die Abhängigkeit der Leistungsfähigkeit von vielen Parametern, wie den Ausführungsdauern der acht simulierten Requesttypen oder dem Verhalten der Klienten konnten aufgrund der begrenzten Ressourcen nicht untersucht werden. Bis zu welcher Anzahl bringen zusätzliche Workerthreads eine weitere Leistungssteigerung? Was ist das ideale Verhältnis von Workerthreads zu Eventschleifen, und wie verhält sich Vert.x, wenn es in großem Maßstab skaliert wird und nicht einige hundert, sondern vielleicht zehntausende Klienten bedienen muss? Ebenfalls sollte die Anzahl der verfügbaren Prozessorkerne variiert werden. Zu erwarten wäre, dass dies weitere Vorteile von Vert.x offenbart, da das

Multithreading der Eventschleife als eine der großen Neuerungen und Stärken des Frameworks gepriesen wird.

Diese Arbeit kann bei Weitem nicht auf alle Fragen eine Antwort geben, sondern nur grundsätzliche Ergebnisse mit begrenztem Aussagegehalt liefern. Sie zeigt etwa, dass der Einsatz asynchroner Frameworks helfen kann, kurze Requests schneller abzuarbeiten – ohne dabei lange Anfragen auszubremsen.

Ebenso konnten große Vorteile aufgezeigt werden für den Fall, dass Workerthreads an einzelne Requesttypen zugewiesen werden, eine Eigenschaft, die vielleicht auch synchronen Frameworks zu einer besseren Performance verhelfen kann

Wichtiger aber ist vielleicht die Implementierung der Server, des Performance-Messungs-Clients und der abstrahierten Unternehmenslandschaft einer Bank. Damit stehen die Werkzeuge für weitere Tests zur Verfügung, in denen die Umgebungsvariablen ergänzt und ihre Ausprägung geändert werden kann.

Mit Blick in die Zukunft ist zu überlegen, ob asynchrone Technologien nicht intelligenter werden können und ihre Leistungsfähigkeit selbst optimieren können, etwa durch eine Art automatisches Loadbalancing. Je nach historischer Häufigkeit und Abarbeitungsdauer einer Requestart wird entschieden, in welchen Workerthreads Anfragen dieser Art ausgeführt werden – oder ob die Eventschleife ihre Abarbeitung übernimmt.

Bei alledem dürfen jedoch auch die sogenannten weichen Faktoren nicht vergessen werden: Momentan findet man sicherlich nicht viele Entwickler, die sofort mit Vert.x umgehen können, auch wenn die Mehrsprachigkeit dabei ein eindeutiger Vorteil ist. Wer leistet Support für Vert.x Server, und inwiefern ist ein Skilltransfer möglich? Die offene Apache-Lizenz von Vert.x reduziert zumindest die Einstiegskosten, jedoch ist die Dokumentation derzeit mangelhaft, was insbesondere die Sekundärliteratur betrifft.

Anhang

Anhangverzeichnis

Anhang 1: Implementierung des Projekts

Der Quellcode der Implementierung des Projektes befindet sich im Verzeichnis *Async_Webserver_Implementierung* auf dem eingereichten Datenträger.

Anhang 2: Detaillierte Ergebnisse des Leistungsvergleichs

Die detaillierten Ergebnisse der während der Arbeit durchgeführten Versuchsreihen wurden in einer Microsoft Excel-Datei abgelegt und befinden sich auf dem eingereichten Datenträger im Verzeichnis *Async_Webserver_Messergebnisse*.

Anhang 3: Citavi-Datenbank der verwendeten Literatur und Quellen

Die Citavi-Datenbank der für die Arbeit verwendeten Literatur befindet sich auf dem gemeinsam mit der Arbeit eingereichten Datenträger. Die Bezeichnung des zip-Verzeichnisses lautet *Async_Webserver_CitaviDB*.

Quellenverzeichnisse

Literaturverzeichnis

Abts, D. (2010): Grundkurs Java, 6. Auflage, Wiesbaden: Vieweg + Teubner

Banke, K. / Krafzig, D. / Slama, D. (2007): Enterprise SOA, 1. Auflage, Heidelberg: Redline

Bauer, G. (2009): Architekturen für Web-Anwendungen, 1. Auflage, Wiesbaden: Vieweg + Teuber

Bineytioglu, C. P., Boden, J. A., Schulz, R. u. A. (2013) Evaluation of Asynchronous Server Technologies, Stuttgart: DHBW Baden-Württemberg

Block, M. (2010): Java-Intensivkurs, Xpert.press, 2. Auflage, Berlin/Heidelberg: Springer

Braun, H. (2014): Node-Lösung, JavaScript-Anwendungen für den Server mit Node.js, in: „c't" Nr. 4 vom 27.01.2014, Seite 172-177

Breshears, C. (2009) The Art of Concurrency, A Thread Monkey's Guide to Writing Parallel Applications, Farnham: O'Reilly

Brothuhn, R. / Chantelau, K. (2010): Multimediale Client-Server-Systeme, eXamen.press, Heidelberg/New York: Springer

Deck, K. / Neuendorf, H. (2007): Java-Grundkurs für Wirtschaftsinformatiker, 1. Auflage, Wiesbaden: Vieweg

Deininger, M. / Faust, G. / Kessel, T. (2009): Java leicht gemacht, Wirtschaftsinformatik kompakt, München: Oldenbourg

Finger, P. / Zeppenfeld, K. (2009): SOA und WebServices, Informatik im Fokus, Berlin/Heidelberg: Springer

Hansen, R./Neumann, G.(2009): Wirtschaftsinformatik 1, 10.Auflage, Stuttgart: Lucius & Lucius

Jäger, K. (2008): Ajax in der Praxis, Xpert.press, Berlin: Springer

Lahres, B. / Rayman, G. (2011):Objektorientierte Programmierung, Galileo Computing, Bonn: Galileo Press

Lewis, B. / Berg, D. J. (2000) Multithreaded programming with Java technology, in: The Sun Microsystems Press Java series, Upper Saddle River, NJ: Prentice Hall

Meinel, C. / Sack, H. (2012): Internetworking, X.media.press, Heidelberg/New York: Springer

Möbus, C. u.a. (2006): Web-Kommunikation mit OpenSource Chatbots, Virtuelle Messen, Rich-Media-Content, Xpert.press, Berlin/Heidelberg: Springer

Nguyen, D. (2012) Jump Start Node.js, Collingwood: SitePoint Pty.

Roden, G. (2012) Node.js & Co: Skalierbare, hochperformante und echtzeitfähige Webanwendungen professionell in JavaScript entwickeln, 1. Auflage, Heidelberg: dpunkt-Verlag

Teixeira, P. (2013) Professional Node.js: Building Javascript Based Scalable Software, 1. Auflage, New York: Wiley, J.

Ullenboom, C. (2011): Java ist auch eine Insel, Galileo Computing, 10. Auflage, Bonn: Galileo Press

Weigend, M. (2010) Objektorientierte Programmierung mit Python 3: Einstieg, Praxis, professionelle Anwendung, 4., aktualisierte Auflage, Heidelberg: mitp

Welch, B. B. (2003) Practical Programming in Tcl/Tk, 4. Auflage, Upper Saddle River, NJ: Prentice Hall/PTR

Verzeichnis der Internet- und Intranet-Quellen

Arranz, J. M. (2011) The “joy” of asynchronous programming, http://www.theserverside.com/discussions/thread.tss?thread_id=61693, Abruf: 28.01.2014

Capan, T. (o. J.) Why The Hell Would I Use Node.js? A Case-by-Case Introduction, http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js, Abruf: 28.01.2014

Fox, T. (2013) The Vert.x Blog, http://vertxproject.wordpress.com/, Abruf: 28.01.2014

Google, Inc. (2014) Where Trends data comes from, https://support.google.com/trends/answer/4355213?hl=en&ref_topic=4365599, 28.01.2014

Jeckle, M (2013): UML Glasklar, http://www.jeckle.de/umlglasklar-/UseCaseDiagramm.pdf, Abruf: 15.12.2013

Johnson, R. u.a. (2013): Spring Framework Reference Documentation, http://docs.spring.io/spring/docs/current/spring-framework-reference/html/index.html, Abruf: 30.01.2014

Kim, J. (2013) Understanding Vert.x Architecture – Part II, http://www.cubrid.org/blog/dev-platform/understanding-vertx-architecture-part-2/, Abruf: 28.01.2014

Maryka, S. (2009) What is the Asynchronous Web and How is it Revolutionary?, http://www.theserverside.com/news/1363576/What-is-the-Asynchronous-Web-and-How-is-it-Revolutionary, Abruf: 28.01.2014

Mykong.com (Hrsg.) (2010): Spring MVC Hello World Example, http://www.mkyong.com/spring-mvc/spring-mvc-hello-world-example/, Abruf: 30.01.2014

Pivotal Software Inc. (Hrsg.) (2014): Spring Tool Suite, http://spring.io/tools/sts, Abruf: 30.01.2014

RedMonk (2014) The RedMonk Programming Language Rankings: January 2014, http://redmonk.com/sogrady/2014/01/22/language-rankings-1-14/, Abruf: 28.01.2014

Semerau, R. (2011) Node.js is good for solving problems I don’t have, http://xquerywebappdev.wordpress.com/2011/11/18/node-js-is-good-for-solving-problems-i-dont-have/, Abruf: 28.01.2014

The Apache Software Foundation (Hrsg.) (2014): Welcome to Apache Maven, https://maven.apache.org/, Abruf: 30.01.2014

Thornsby, J. (2010) Node.js Moves to Joyent, http://jaxenter.com/node- jsmoves-to-joyent-32530.html, Abruf: 28.01.2014

Toub, S. (2011)	Parallel Programming with .NET, http://blogs.msdn.com/b/pfxteam/archive/2011/09/17/10212961.aspx, Abruf: 28.01.2014
W3C (Hrsg.) (1999):	Hypertext Transfer Protocol – HTTP/1.1, http://www.w3.org/Protocols/rfc2616/rfc2616.html, Abruf: 30.01.2014
o. A. (2003)	Identifiying disk I/O-bound systems, http://osr507doc.sco.com/en/PERFORM/ident_IO_bound.html, Abruf: 28.01.2014
o. A. (2011a)	Felix's Node.js Convincing the boss guide, http://nodeguide.com/convincing_the_boss.html, Abruf: 28.01.2014
o.A. (2011b)	Intro to Node.JS for .NET Developers, http://www.aaronstannard.com/post/2011/12/14/Intro-to-NodeJS-for-NET-Developers.aspx, Abruf: 28.01.2014
o. A. (2012)	Vert.x vs node.js simple HTTP benchmarks, vertxproject.wordpress.com/2012/05/09/vert-x-vs-node-js-simple-http-benchmarks/, Abruf: 28.01.2014
o. A. (2013)	Node.js v0.10.25 Manual & Documentation, http://nodejs.org/api/, Abruf: 28.01.2014
o. A. (o. J.a)	Apache Tomcat Configuration Reference, http://tomcat.apache.org/tomcat-7.0-doc/config/http.html, Abruf 28.01.2014
o. A. (o. J.b)	Introduction to Asynchronous Programming, http://cs.brown.edu/courses/cs196-5/f12/handouts/async.pdf, Abruf: 28.01.2014
o. A. (o. J.c)	Vert.x Manual, http://vertx.io/manual.html#introduction, http://vertx.io/manual.html, Abruf: 28.01.2014

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Migration von Open Source Content Management Systemen

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Integrationsseminar“

Vorgelegt von

Philipp Richter,
Maurice Görges

am 31.01.2014

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
WWI2011I

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BPM	Business Process Management
CMIS	Content Management Interoperability Services
CMS	Content Management System
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DM	Dokumentenmanagement
ECM	Enterprise Content Management
HTML	Hypertext Markup Language
JCR	Content Repository for Java Technology API
RM	Record Management
WCM	Web Content Management
WCMS	Web Content Management System
WF	Workflow Management
XML	eXtensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung (Philipp Richter / Maurice Görges)

1.1 Einleitung (Philipp Richter / Maurice Görges)

Open Source Lösungen im Enterprise-Umfeld haben sich in den vergangenen Jahren mehr und mehr etabliert und sind inzwischen in Anwendungsbereiche vorgedrungen, die bisher von kommerziellen Produkten besetzt waren. Insbesondere beim Aufbau von Web-Portalen werden Open Source-Lösungen in vielen Bereichen eingesetzt.

Ein wichtiges Gebiet beim Aufbau eines Portals ist das Erfassen, Speichern, Verwalten und Bereitstellen von Inhalten für einen Webauftritt.

Diese Inhalte werden im Falle von Unternehmenswebseiten meist von Content Management Systemen (CMS) verwaltet und bereitgestellt. Das CMS kümmert sich dabei um die redaktionelle Bearbeitung, die Strukturierung der Website, sowie das Verwalten mehrerer Sprachversionen oder Varianten, die für unterschiedliche Geräte geliefert werden müssen.

Im Open Source Umfeld existiert eine ganze Reihe von Implementierungen von CMS-Systemen (z.B. OpenCMS, TYPO3, Joomla!, etc.). Die Entscheidung für ein CMS-System bedeutet gegebenenfalls aber, dass man sich auf lange Zeit einem System „verschreibt“. Kommen neue Systeme auf den Markt oder wird die Entwicklung des CMS von der Community nicht weiterverfolgt, hat das in einem Unternehmenskontext hohe Kosten zur Folge, wenn man die erstellten Inhalte nicht retten und in ein neues CMS überführen kann.

Ziel der Arbeit ist es, zu untersuchen, inwieweit bestehende CMS-Implementierungen im Open Source-Bereich die Migration von Inhalten auf ein neues System zu unterstützen, welche Optionen es gibt und wie diese bewertet werden können.

1.2 Zielsetzung (Philipp Richter / Maurice Görges)

Es soll aufgedeckt werden inwieweit ein bestehendes Content-Management-System in einem Unternehmen durch ein neues System ersetzt werden kann und welche Inhalte dabei migriert werden können und müssen. Dementsprechend soll geklärt werden, welche Optionen es zur Migration von Daten und Inhalten vorhanden sind und welche unternehmensrelevanten CMS Systeme es auf dem Markt gibt, die diese Migrationsfunktion unterstützten.

1.3 Vorgehensweise (Philipp Richter / Maurice Görges)

Im zweiten Kapitel werden zunächst die theoretischen Grundlagen, die für das bessere Verständnis der folgenden Kapiteln nötig sind, beschrieben. Ausgehend von einer Definition, Klassifikation und Funktionsbeschreibung von Content-Management-Systemen werden insbesondere auch der Aufbau von Inhalten und die Marktsituation auf dem CMS Markt betrachtet. Grundlagen der Migration von CMS, Kriterienkatalogen und Unternehmensrelevanz schaffen die Voraussetzungen für Kapitel 3 und 4.

In Kapitel 3 sollen verschiedene Strategien zur Migration von Content-Management-Systemen erarbeitet werden. Es werden zunächst Automatisierte und Manuelle Migration unterschieden, bevor spezielle Technologien und Angebote genauer vorgestellt werden. Insbesondere handelt es sich hierbei um: Standards, Austauschformate, Plug-Ins und Erweiterungen, Skripte und Migration as a Service.

Im vierten Kapitel werden die gewonnen Erkenntnisse für den Aufbau eines Kriterienkatalogs angewandt. Basierend auf der grundlegenden Theorie aus Kapitel 2 und erarbeiteten Migrationsstrategien aus dem vorangegangenen Kapitel wird ein Kriterienkatalog zur Bewertung der Migrierbarkeit von Content-Management-Systemen erstellt. In einem letzten Schritt werden exemplarisch drei CMS auf ihre Migrierbarkeit hin untersucht.

2 Grundlagen – Open Source Content Management Systeme (Philipp Richter / Maurice Görges)

2.1 Content Management Systeme (Philipp Richter / Maurice Görges)

2.1.1 Definition und Klassifizierung (Philipp Richter / Maurice Görges)

Bei der Anfertigung dieser Arbeit wurde deutlich, dass gewisse Begrifflichkeiten immer wieder in Zusammenhang mit CMS angeführt werden. Die einzelnen Abgrenzungen sind dabei nicht immer genau zu erkennen.

Oftmals wird unter CMS das eigentliche Web-Content Management (WCM) verstanden, da der Schwerpunkt eines CMS oft auf dem Management von Websites liegt. Genau genommen ist WCM allerdings lediglich eine Komponente von CMS. Weitere häufig aufzufindende Begriffe im Zusammenhang zu CMS sind Wissensmanagement, Dokumentenmanagement (DM) und Enterprise-Content-Management (ECMS). Daher soll in diesem Kapitel eine Begriffsbestimmung und Klassifikation der unterschiedlichen Typen von Content Management Systemen

erfolgen. Vorweg soll allerdings eine Unterscheidung der Begriffe „Content“, „Content Management“ sowie „Content Management Systeme“ ein besseres Bild verschaffen.

Content (Inhalt) kann als Information oder Informationspaket verstanden werden, das mit Hilfe eines Mediums (Papier, TV, Computer, etc.) weitergegeben werden kann.[1]

Unter Content Management können somit Prozesse wie Erstellung, Speicherung, Abruf, Präsentation, Verwaltung, und Verbreitung von Content verstanden werden.[2]

Unter einem Content-Management-System wird eine Software verstanden, die zur Erstellung, Bearbeitung und Organisation von Inhalten dient und somit die Aufgaben des Content Management in elektronischer Form unterstützt.[3]

Heute auf dem Markt befindliche Content Management Systeme setzen innerhalb der oben aufgezeigten Prozesse unterschiedliche Schwerpunkte und können aufgrund ihres Funktionsumfangs und des systemspezifischen Verständnisses von Content Management in folgende Systemkategorien klassifiziert werden:

Web-Content-Management: WCM Systeme basieren auf der Trennung von Inhalt und Programmierung und dienen der Verwaltung von Portalen und Websites im Internet und Intranet.[4] Inhalte können so ohne Programmierkenntnisse dezentral eingegeben werden.[5] Somit ist ein WCMS eine besondere Ausprägung eines CMS und dient v.a. der redaktionellen Bereitstellung von Webinhalten und der Pflege von Websites.

Dokumentenmanagement: DM Systeme dienen der Verwaltung aller unternehmensweiten Dokumente. Dabei werden Funktionen wie Scannen, Erstellen, Verwalten, Weiterleiten, Ablegen, Archivieren, Suchen und Abrufen von Dokumenten erfasst, wobei diese durch die Metadaten beschrieben werden.[6] Zu beachten ist, dass auch nicht rechnerunterstützte Dokumente bspw. in Papierformat mit einbezogen werden.

[1] Spörrer 2009, S.5ff
[2] Ebenda, S.6ff
[3] Christ 2003, S.135ff
[4] Kocakurt 2010, S.7
[5] Spörrer 2009, S.27
[6] Fröschle, Reich 2007, S.109

Record Management: „Unter Records Management werden allgemein die Methoden, Verfahren und Anwendungen verstanden, die zu einer geordneten Verwaltung, Erschließung, Aufbewahrung, Sicherung und Löschung elektronischer Informationen dienen.“[7]

Collaboration: Collaboration Systeme dienen v.a. der Kommunikation im Unternehmen und nutzen Informations- und Kommunikationstechnologien zur Unterstützung von Teamarbeit wie bspw. Telefon- und Videokonferenzsysteme, firmeninterne Communicator oder Sharepoints zur gemeinsamen Erstellung und Abruf von Dokumenten.[8] Zudem fällt auch das Wissensmanagement (Knowledge Management) unter Collaboration, welches „[…] alle Aktivitäten, die sich mit Daten, Informationen und Knowledge beschäftigen“[9] befasst.

Worklflow & Business Process Management: Workflow (WF) und Business Process Management (BPM) besitzen eine wichtige Verbindungs-, Steuerungs- und Kontrollfunktion in einem ECM.[10] Ein Beispiel hierfür wäre die Kontrolle und Dokumentation der Bearbeitungsstände, Laufwege und Ergebnisse der Bearbeitung. Während WF die weitgehende Automatisierung von Prozessen mit Einbindung aller notwendigen Ressourcen zum Ziel hat, soll ein BMP der vollständigen Integration aller Unternehmensanwendungen mit Kontrolle der Prozesse und Zusammenführung aller benötigten Informationen dienen.[11]

Enterprise Content Management: ECM umfasst die Methoden, Techniken und Werkzeuge zur Erfassung, Verwaltung, Speicherung, Bewahrung und Bereitstellung von Content und Dokumenten zur Unterstützung organisatorischer Prozesse im Unternehmen.[12]

2.1.2 Funktionen eines CMS am Beispiel ECMS (Philipp Richter / Maurice Görges)

Die sogenannten ECM-Komponenten können in die zuvor beschriebenen Anwendungsfelder Document Management (DM), Collaboration, Web-Content-Management (WCM), Record Management (RM) sowie Workflow (WF) & Business Process Management (BPM) unterteilt werden. In Abbildung 1 wird der Content-Lebenszyklus innerhalb eines ECMS mit seinen jeweiligen Komponenten beschrieben und zeigt die folgenden Funktionalitäten auf:

[7] Riggert 2009, S.8
[8] Springer Gabler Wirtschaftslexikon
[9] Hüttenegger 2006, S.3
[10] Kampffmeyer 2011a
[11] Vom Brocke, Simons 2013, S. 23f
[12] Association for Information and Image Management 2013

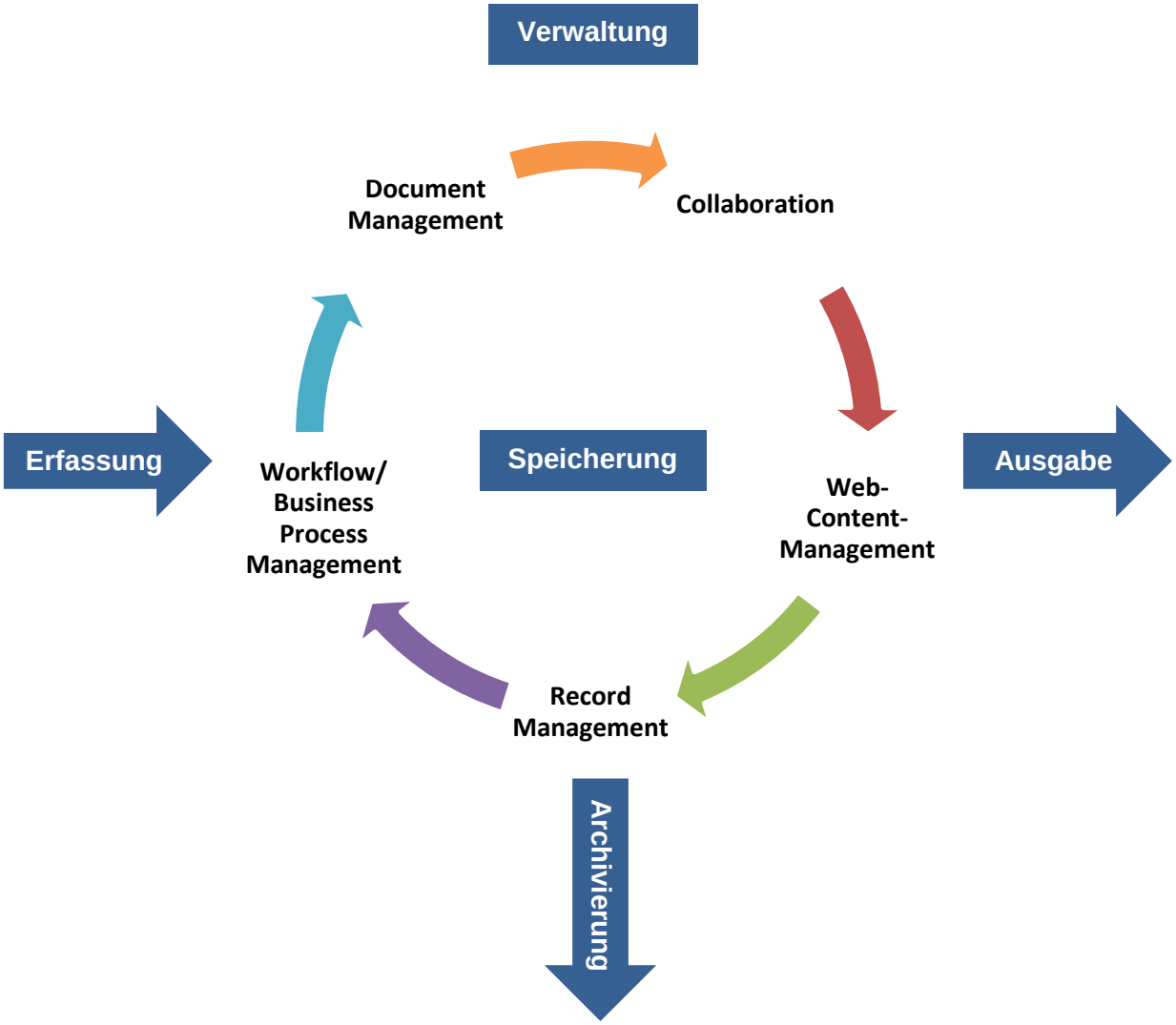


Abbildung 1 ECMS[13]

Erfassung: Input-Komponente zur Erstellung, Aufbereitung und Verarbeitung von analogen und elektronischen Informationen[14] z.B. Digitalisieren von analogen Medien durch Scannen.

Verwaltung: Komponente zur Bearbeitung und Nutzung der Informationen in allen Anwendungsbereichen (WCM, DM, etc.) z.B. mit Hilfe von Datenbanken.[15]

Speicherung: Komponente zur „[…] temporären Speicherung von Informationen, die nicht archivierungswürdig oder archivierungspflichtig sind.“[16]

[13] Vom Autor erstellt in Anlehnung an Kampffmeyer 2011c
[14] Kampffmeyer 2011a
[15] Kampffmeyer 2011d
[16] Kampffmeyer 2011f

Archivierung: Komponente zur langfristigen und statischen Sicherung von Informationen.[17]

Ausgabe: Output-Komponente, die die zuvor in „Verwaltung“ bearbeiteten Informationen bereitstellt: z.B. Präsentation von Inhalten über Webportale, eBooks, etc.[18]

Eingrenzung:

Der Fokus der Arbeit soll von nun an auf unternehmensrelevanten **Web-Content Management Systemen** liegen. Darum werden andere in der Klassifikation genannte Formen von CMS, die der besseren Übersicht über die diversen Anwendungsarten diente, fortan keine Erwähnung mehr finden.

2.1.3 Aufbau von Inhalten (Philipp Richter / Maurice Görges)

Definition

Unter Inhalten (Content) werden Informationselemente verstanden, welche durch eine bestimmte Informationsart (z.B. Text, Audio, Bild, Grafik, Video) dargestellt werden.[19] Diese können alle digital verarbeitet und zu übermittelt werden und zusätzlich in statische Informationsarten (Text, Grafik, Bild) oder dynamische Informationsarten (z.B. animierte Grafik, Audio und Video) unterteilt werden.[20,21]

Neben diesen beschreibenden Substanzformen umfasst Content allerdings auch Metadaten wie Autor, Titel und Inhaltsbeschreibung.[22] Zusätzlich können beispielsweise auch Informationen wie Benutzer-Daten und Zugangsberechtigungen zu den Inhalten gezählt werden.[23]

Bearbeitung von Inhalten in Web Content Management System

Durch die Trennung von Inhalt und Layout werden dem Anwender bei einem WCMS gestalterische Entscheidungen erspart.[24] Diese werden in einem zuvor gestalteten Layout festgelegt und dem Anwender als Template (Layoutvorgabe) vorgegeben. Inhaltselemente wie Texte und Bilder werden dann in die genannten Templates eingesetzt.[25] In einem per Login gesicherten

[17] Kampffmeyer 2011e
[18] Kampffmeyer 2011b
[19] Bodendorf 2006, S.95
[20] Ebenda, S.95
[21] Mauthe, Thomas 2003, S.23f
[22] Spörrer 2009, S.5
[23] Hazlett 2013
[24] Krüger, Kopp 2002, S.22
[25] Zschau et al. 2002, S.59

Bereich (Backend) ist es so dem Anwender ohne HTML-Kenntnisse möglich, Inhalte einzugeben oder zu verändern.[26] Die Templates sorgen dann für eine korrekte Darstellung in Ihrem individuellen Design auf der Webseite, was in diesem Zusammenhang als Frontend verstanden wird.[27]

Die dadurch zur Verfügung stehende Arbeitsoberfläche, mit der ein Anwender an einer Webseite arbeitet, ähnelt so bspw. einem Textverarbeitungsprogramm, wie z.B. Microsoft Word.

Vorteile

Aus den Eigenschaften vieler CMS lassen sich folgende Vorteile ableiten:

- Keine Software zur Benutzung muss installiert werden, sondern das Programm läuft in jedem üblichen Browser einer neueren Generation (z.B. den "Internet Explorer 7 bzw. 8" oder "Firefox"). Über das Internet kann die Website dadurch jederzeit und von jedem Ort aktualisiert und bearbeitet werden.
- Um nachträglich Änderungen an einer Webseite durchzuführen, werden keine Spezialisten mehr benötigt, wodurch Kosten gespart und Arbeitsabläufe vereinfacht werden.
- Der Zugriff und die Pflege von Inhalten durch verschiedene Anwender kann durch die Vergabe von Benutzerrechten erleichtert werden. Bspw. werden definierte Bereiche mit Bearbeitungsrechten bestimmt und so Teile der Website nur für bestimmte Anwender freigegeben.
- Durch eine Bestimmung von allgemeinen Zugriffsrechten kann so ein Intranet oder Extranet auf einer Internet-Seite geschaffen werden. Interne sowie externe Prozesse wie bspw. Datenaustausch, News oder Unternehmens-Interna können so in einem WCMS gut verwaltet werden.

Bei der Migration von Inhalten von einem Content Management System auf ein anderes muss somit die Sicherung folgender Informationen sichergestellt werden:

- Text (Posts/Seiten)
- Kategorien
- Bilder
- User
- Kommentare
- Tags
- Interne Links
- Anhänge
- Meta-Daten
- Templates

[26] Krüger, Kopp 2002, S.22f
[27] Ebenda, S.23

2.1.4 Marktsituation – Proprietär vs. Open Source (Philipp Richter / Maurice Görges)

Der CMS-Markt lässt sich hauptsächlich in kommerzielle Systeme und Open Source Software unterteilen.

Während bei kommerzieller Software eine Veränderung des Quelltextes i.d.R. nicht gestattet ist (proprietäre Software) und bereits für die Lizenzierung Gebühren entstehen, ist bei Open Source Software der Quellcode frei verfügbar und kann beliebig geändert werden.[28,29]

Wikipedia.org listet derzeit 108 vollwertige, Open Source CMS und 40 kommerzielle Lösungen.[30] Cmscrawler.com zählt im Gegensatz zu Wikipedia über 830 verschieden Content Management Systeme auf. Auf cmsmatrix.org werden 1200 verschiedene CMS aufgeführt.[31] Webkalkulator.com erfasst gar mehr als 13000 verschiedene Systeme.[32] Diese Unterschiede in der Anzahl der Ergebnisse lassen sich auf die unterschiedlich enge Definition von Content Management zurückführen. Alle Listen zeigen jedoch unabhängig von der Gesamtanzahl der Systeme einen deutlichen Trend: Open Source Produkte dominieren die kommerzielle Konkurrenz. Die Gründe hierfür sind in der mindestens gleichen Leistungsfähigkeit, problemloser Erweiterbarkeit und Individualisierbarkeit und natürlich den geringeren Kosten zu suchen.

System	Anzahl	Marktanteil	Typ
1 WordPress	4.186.559	46,8%	Open Source
2 Joomla!	1.173.413	13,1%	Open Source
3 Drupal	496.456	5,5%	Open Source
4 Microsoft FrontPage	369.543	4,1%	Kommerziell
5 TYPO3	312.804	3,5%	Open Source
6 Magento	104.556	1,2%	Open Source
7 Blogger	92.358	1,0%	Open Source

Tabelle 1 CMS Marktführer[33]

[28] Wikimedia Foundation, Inc. 2014e
[29] Open Source Initiative 2014
[30] Wikimedia Foundation, Inc. 2014d
[31] CMS Matrix
[32] Bösiger Engineering AG 2014
[33] Eigene Darstellung basierend auf Daten von CMS Crawler 2014

Tabelle 1 zeigt die am meist verwendeten Vertreter beider Kategorien und ihren Marktanteil gemessen an der Anzahl an Webseiten die auf dem jeweiligen System basieren. Die Daten stützen sich auf eine Untersuchung von 45 Mio. Internetseiten weltweit. Von diesen konnten 8 949 450 Seiten einem CMS Tool zugeordnet werden.[34] Insgesamt wurden 830 unterschiedliche Systeme identifiziert. Die Grafik (Tabelle 1) zeigt nur diejenigen die einem Marktanteil von mindestens 1% haben. Es zeigt sich die Dominanz der Open Source Systeme, denn mit Microsoft FrontPage ist nur ein Vertreter der kommerziellen Systeme aufgeführt.

2.2 Grundlagen der Migration (Philipp Richter / Maurice Görges)

In dem Kontext dieser Arbeit bezieht sich Migration auf die Überführung von Web-Inhalten von einer Plattform auf eine andere. Dieser Vorgang findet meistens während eines Wechsels oder Upgrades von Content Management Systemen statt.

Bei einem Wechsel des Content Management Systems ändert sich oftmals auch die Struktur der Inhalte. Die größte Herausforderung ist hier die Überführung aller relevanten Inhalte (siehe. 2.1.3) in die Strukturen des neuen CMS.

Zur systematischen Überführung alter Seitenbestandteile in das neue Layout dient ein Migrationskonzept: Es gilt, den Ziel-Webauftritt sowie die einzelnen Planungsschritte inhaltlich, zeitlich und organisatorisch zu definieren.[35] Zusätzlich soll der der Ist- mit dem Soll-Zustand in seinen einzelnen Bestandteilen der Navigations- und Inhaltsstruktur abgeglichen werden und im Migrationskonzept festgelegt werden.[36] Veraltete und doppelte Inhalte sollten zusätzlich konsolidiert werden.

Um die Migration zu vereinfachen, sollten während dem gesamten Prozess der Überführung keinerlei Änderungen am eigentlichen Inhalt der Seiten gemacht werden, um Daten-Inkonsistenz zu vermeiden (auch Content Freeze genannt).[37]

[34] CMS Crawler 2014, Stand 22.01.2014
[35] HowTo.gov 2013
[36] Ebenda
[37] Ebenda

2.3 Unternehmensrelevanz (Philipp Richter / Maurice Görges)

Nicht alle Open Source Content Management Systeme sind für den Einsatz in Unternehmen geeignet. Im Folgenden soll anhand von verschiedenen Kriterien eine Auswahl von Content Management Systemen getroffen werden, die in Unternehmen einsetzbar sind und exemplarisch in dieser Arbeit untersucht werden.

2.3.1 Kriterien (Philipp Richter / Maurice Görges)

Bei der Auswahl der Content Management Systeme wurde insbesondere Wert auf die Erfüllung folgender Kriterien, die bei dem Einsatz in großen Unternehmen wichtig sind, gelegt:[38,39]

- Installationsaufwand
- Benutzerfreundlichkeit
- Strukturelle Flexibilität
- Community / Web 2.0 Funktionalität
- (Sicherheits-) Updates und Kompatibilität
- Konfigurationsaufwand
- Suchmaschinenoptimierung
- Rechte, Rollen und Workflows
- Erweiterung und Integration
- Support und Größe der Community

2.3.2 Auswahl relevanter Systeme (Philipp Richter / Maurice Görges)

Aufgrund der großen Anzahl an verschiedenen Systemen ist es im Rahmen dieser Arbeit nicht möglich alle CMS Systeme zu begutachten. Darum wurde eine Auswahl getroffen um die danach gewählten Systeme detaillierter betrachten zu können. Da Bestandteil der Arbeit lediglich die Untersuchung von Open Source Lösungen ist, werden kommerzielle Systeme wie bspw. „Microsoft FrontPage“ nicht berücksichtigt. Ein weiteres Kriterium für die Auslese der relevanten Systeme ist die für ein Unternehmen notwendige Voraussetzung einer großen Programmier-Community. So kann eine stetige Verbesserung und möglicher Support zur Fehlerbehebung und Wartung sichergestellt werden. Darum werden nur die Systeme vorgestellt, die als Marktführer bezeichnet werden können. Zusätzlich hängt die Auswahl mit der Funktionalität und der Technologie, die von den verschiedenen Systemen genutzt werden zusammen.

In Abstimmung mit den Ergebnissen des Projektes Marktanalysen: *Open Source Content Management-Systeme (CMS)* werden nach der Bewertung durch die in 2.3.1 genannten Kriterien

[38] Jischke 2012
[39] Onasch 2006

Joomla!, Typo3, und Magnolia als exemplarische Content Management Systeme für diese Arbeit ausgewählt. WordPress, Blogger, Drupal und Magento stellen nicht den unternehmensnotwendigen Funktionsumfang zur Verfügung.

Joomla!

Joomla! Ist ein Open-Source CMS zur Erstellung und Verwaltung von ambitionierten Webprojekten verschiedener Art und Größenordnungen.[40] Das CMS basiert auf der Programmiersprache PHP 5 in Kombination mit dem Datenbanksystem MySQL.[41] Die Anwendung als Redaktionssystem erfolgt über den Web-Browser. Die Installation benötigt allerdings wegen der Einrichtung eines Webservers sowie der SQL Datenbank gewisse Fachkenntnisse. Joomla! ist ein stark erweiterbares CMS, welches sich auf eine große Community berufen kann und stellt somit die Basis eines Management Systems für Unternehmen oder Informationsportale dar.[42] Neben Kernfunktionen wie Benutzerverwaltung, Menüs, Artikel, Formulare, etc., sind im CMS diverse standardmäßige Erweiterungen als Komponenten, Module und Plugins umgesetzt.[43] Über das „Joomla Extension Directory“ (JED) (offizielles Erweiterungsverzeichnis), können dem System zudem noch weitere Funktionen hinzugefügt werden.[44] Das System wird weltweit für die Erstellung und Verwaltung von Websites und Portalen oder Intranets und Extranets für Firmen oder öffentliche Einrichtungen eingesetzt.[45]

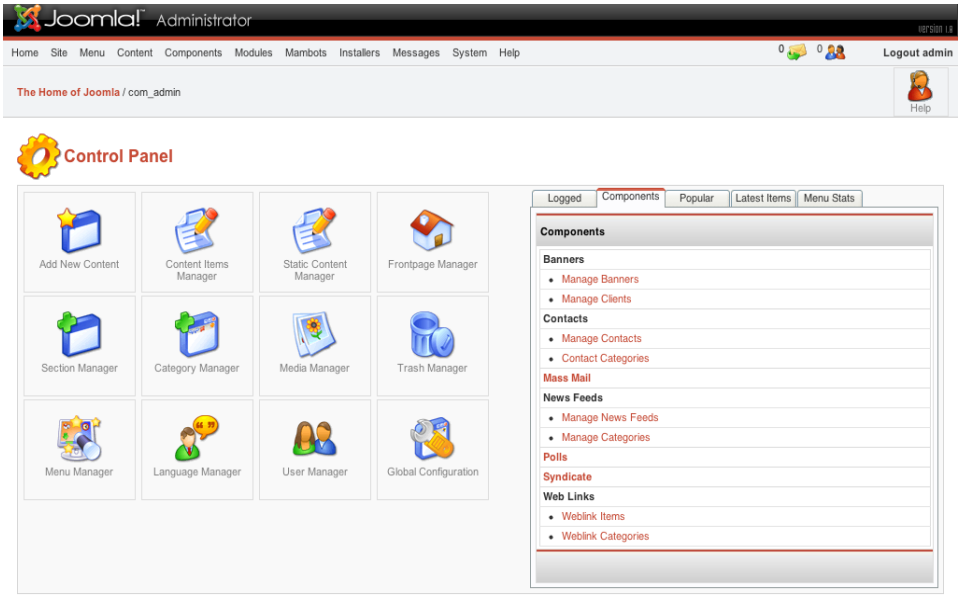


Abbildung 2 Joomla! Control Panel[46]

[40] JandBeyond e.V. 2014
[41] Ebenda
[42] Black Duck Inc. 2014a
[43] JandBeyond e.V. 2014
[44] Open Source Matters Inc. 2014
[45] JandBeyond e.V. 2014
[46] Apps Kostenloser 2014

TYPO3

TYPO3 ist in PHP geschrieben und läuft auf mehr als 220.000 Servern weltweit.[47] Die Anwendung wurde in 45 Sprachen übersetzt und wird aktiv in einer Community von ca. 40.000 Nutzern in 50 Ländern weiterentwickelt. Das CMS unterstützt die volle Flexibilität und Erweiterbarkeit und bietet eine große Auswahl an vorgefertigten Interfaces, Funktionen und Modulen.[48] TYPO3 dient als umfangreiches Open Source CMS für große Internetplattformen, Marktplätze und Portallösungen und unterstützt die Erstellung von Website-Designs, sowie die Eingabe von Website Inhalt.[49] Zudem bietet Typo3 die Option, Inhalte, neben dem Backend, auch direkt über das Frontend der Website zu bearbeiten.[50] Das System wird nach Installation auf einem Webserver über einem Webbrowser genutzt, allerdings ist die Konfiguration von TYPO3 mittels TypoScript ohne einen fachkundigen Entwickler schwierig. Weiterentwickelt wird das System von zwei so genannten „Core-Teams“, bei denen eines für TYPO3 CMS und das andere für TYPO3 Neos verantwortlich ist.[51]

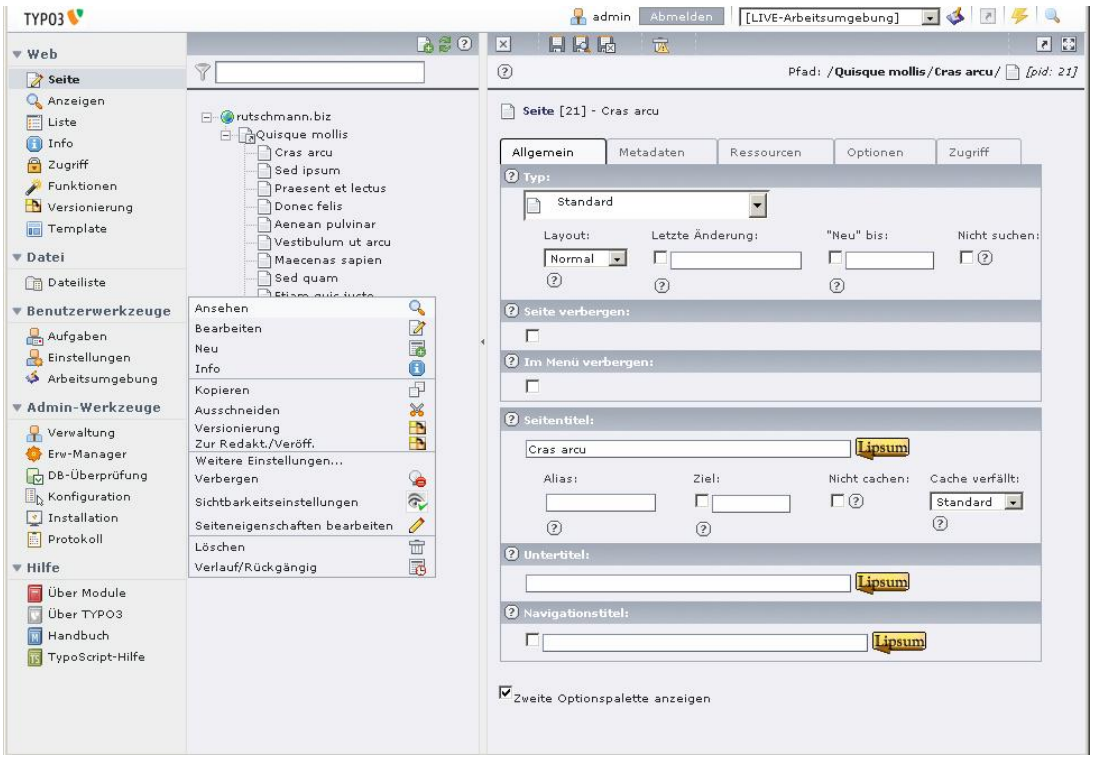


Abbildung 3 TYPO3 Backend[52]

[47] Black Duck Inc. 2014b
[48] Ebenda
[49] Typo3 Association 2014b
[50] Ebenda
[51] Typo3 Association 2014a
[52] der-webentwickler.net 2011

Magnolia CMS

Magnolia CMS ist ein offenes Java CMS mit einer HTML5 Benutzeroberfläche. Mit Magnolia CMS können bspw. Unternehmen Online-Marketing, Online-Services und Online-Vertrieb über alle digitalen Kanäle hinweg koordinieren.[53] Durch erweiterbare, aufgabenzentrierte Apps, Benachrichtigungen und Favoriten für den schnellen Zugriff auf den Workspace wird es als Unternehmenslösungen angeboten.[54] Die Open Source-Technologie unterstützt offene Standards bspw. CMIS und JCR, sowie die Integration mit dem Spring Framework.[55] Somit ermöglicht es Programmierern, Backend-Systeme und Anwendungen von Drittanbietern voll einzusetzen.[56] Laut Hersteller wird das System in über 100 Ländern im öffentlichen Sektor und in führenden Fortune 500 Unternehmen eingesetzt. Kundenbeispiele sind u.a. die Allianz Gruppe, EADS, Pirelli, REWE oder Sony. Magnolia International Ltd. bietet neben einer Open Source Community Edition auch eine lizenzierte Enterprise Edition des CMS an.

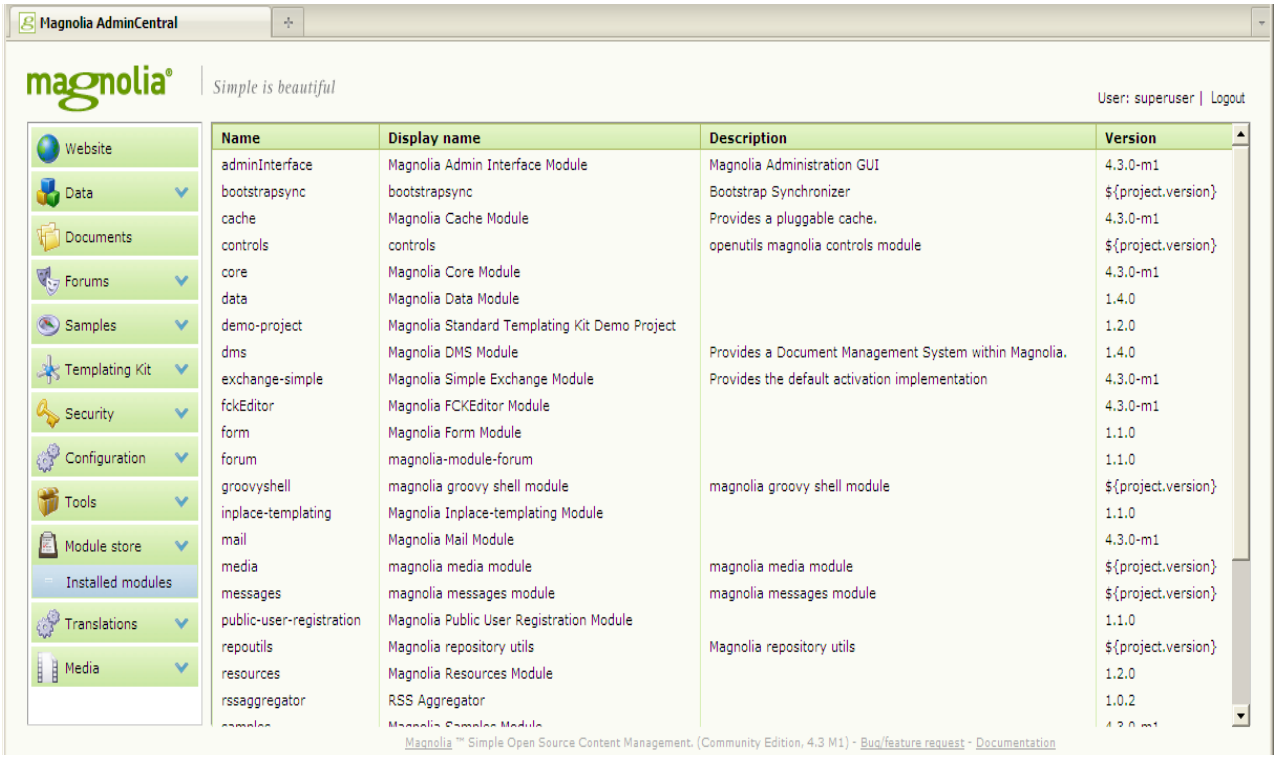


Abbildung 4 Magnolia Admin Central[57]

[53] Magnolia International Ltd. 2014
[54] Ebenda
[55] Ebenda
[56] Ebenda
[57] Altassian 2012

2.4 Kriterienkatalog (Philipp Richter / Maurice Görges)

Computergestützten Software-Anwendungen kommt heutzutage eine immer größere Bedeutung zu. Die meist große Auswahl an existierenden Programmen stellt darum Käufer und Anwender beim Vergleich und der Auswahl der geeigneten Software vor Probleme. Somit wird ein Verfahren zur schnellen und kostengünstigen Qualitätsbeurteilung benötigt. Diesem Bedarf versuchen sog. Kriterienkataloge gerecht zu werden.

Definition

„Ein Kriterienkatalog ist eine hierarchische, geordnete und logische Sammlung von Kriterien für Indikatoren und Kennzahlen, um ein System möglichst genau, standardisiert und wiederholbar evaluieren zu können“[58]

Kennzahlen liefern Werte, die eine Bewertung von Systemen zulassen. Somit können Kriterienkataloge Zusammenstellungen von Fragen und Einschätzungsskalen enthalten, die zur standardisierten Beschreibung und Beurteilung von Aspekten der technischen, inhaltlichen oder anwendungsfreundlichen Qualität von Software dienen. Synonyme für Kriterienkataloge sind oftmals Checkliste oder Kriterienraster.

Methodik

Grundlage der Qualitätsbeurteilung sind von den Katalogentwicklern aufgestellte Qualitätskriterien. Diese thematisieren die für die Produktqualität als bedeutsam zu erachtenden Merkmale z.B. Grundfunktionen.[59]

Durch die Einbeziehung sog. K.O.-Kriterien z.B. unternehmensrelevante Kriterien (s. 2.3) kann mit Hilfe von Kriterienkatalogen eine kostengünstige und schnelle Vorselektion von Software getroffen werden. Diese alles-oder-nichts-Attribute zeichnen sich dadurch aus, dass sie leicht durch eine ja/nein Abfrage beantwortet werden können.[60]

Praktisch wird also eine Software von einem Bewertenden anhand von Kriterien oder Fragen analysiert. Häufig sind Antworten bereits vorgegeben und durch ein entsprechendes Research z.B. im Internet zu klären. Ziel ist immer die Bewertung eines Produktes. Somit kann das Ergebnis bspw. für Verbesserungsvorschläge oder zu Auswahl einer bestimmten Software dienen. Durch eine immer gleiche und schrittweise Abarbeitung der aufgestellten Kriterien kann

[58] Meier 1995, S.150
[59] Baumgartner 1995, S.242ff
[60] Ebenda, S.242ff

das Verfahren objektiv und methodisch sauber durchgeführt werden.[61] Die Durchführung kann in folgende Schritte unterteilt werden.

1. Auswahl der Evaluationskriterien:

   Meier listet nach einem Vergleich mehrerer Kriterienkataloge folgende Anforderungen auf, die Kriterien erfüllen sollten.[62]

- Die Kriterien sind sachlich korrekt, verständlich, treffend und knapp formuliert.
- Die Kriterien liegen in strukturierter Form vor und sind in Kriteriumskategorien unterteilt.
- Die Kriteriumskategorien sind einzeln als Prüfinstrumente anwendbar.

2. Die Sichtung und Dokumentation der am Markt befindlichen Programme

   Betrachtung des Marktes und Auswahl von Programmen, welche anschließend Anhand der Kriterien beurteilt werden.

3. Auswertung und Ergebnis

   Zusammenfassung der Ergebnisse und Programmempfehlung anhand von gewonnenen Erkenntnissen.

3 Arbeitsteil – Strategien zur Migration von Content Management Systemen (Philipp Richter / Maurice Görges)

In diesem Kapitel sollen Strategien zur Migration von Content Management Systemen erarbeitet werden. Hierzu werden zunächst allgemeine Strategien – Manuelle-, Automatische- und Hybrid-Migration vorgestellt. Anschließend werden verschiedene Technologien und Angebote vorgestellt, die bei der Migration von Content Management Systemen eine entscheidende Rolle spielen. Die in diesem Kapitel erarbeiteten Strategien bilden die Grundlage für die Entwicklung eines Kriterienkataloges in Kapitel 4.

[61] Ebenda, S.242ff
[62] Meier 1995, S.188ff

3.1 Manuelle vs. Automatisierte Migration (Philipp Richter / Maurice Görges)

Generell lassen sich zwei Migrationsstrategien unterscheiden[63]: Manuelle und Automatisierte Migration. Die Größe und Komplexität der zur migrierenden Website bestimmt ob bestimmte Teile oder gar die komplette Migration automatisiert werden können oder ob Content ganz oder zum Teil manuell überführt werden muss.

Wie in *2.2 Grundlagen der Migration* erwähnt, sollten während der gesamten Migration keine Inhalte verändert werden um Daten Konsistenz zu gewährleisten und den Verlust von Daten zu verhindern. Eine manuelle Migration kann einen Content Freeze von mehreren Tagen bis hin zu mehreren Wochen bedeuten; eine automatisierte Migration kann diese Zeit drastisch verringern, erhöht aber unter Umständen die Vorbereitungszeit.

Generell ist eine manuelle Migration bei Content Management Systemen, die bis zu 500 Seiten verwalten, möglich. Da bei größeren Unternehmen – die vor allem in dieser Arbeit betrachtet werden sollen – häufig wesentlich größere Datenbestände vorliegen, ist in diesem Bereich hauptsächlich eine automatisierte Migration vorzuziehen. Die verschiedenen Migrationsstrategien werden im Folgenden dargestellt.

Manuelle Migration

Bei einer manuellen Migration werden Inhalte hauptsächlich via „Copy & Paste“ von einem Content Management System zum anderen übertragen.[64] Die benötigte Zeit für diesen Vorgang kann, in Abhängigkeit von der Komplexität der zu migrierenden Seite und den Verfügbaren Ressourcen, bei kleineren Webseiten (weniger als 1.000 Seiten) wenige Wochen oder Monate dauern.[65] Bei großen Seiten (mehr als 50.000 Seiten) hingegen kann dies – von der ersten Planung bis hin zur Fertigstellung – wesentlich länger dauern. Generell wird eine manuelle Migration für Webauftritte mit 300 – 500 Seiten, abhängig von der Komplexität der Seiten, empfohlen.[66]

Bei einer manuellen Migration sind u.a. folgende Vorgänge zu beachten:

1. Selektion des betreffenden Textes und Eingabe in das entsprechende Eingabefeld des Ziel-CMS
2. Tabellarische Inhalte und Sonderformatierungen rekonstruieren
3. Bilder und Grafiken in geeigneten Formaten erstellen, hochladen und mit Metainformationen versehen
4. Benutzer und Rechte neu anlegen und zuweisen

---

[63] Hildebrand 2007, S.7f

[64] Magnolia International Ltd. 2013b

[65] Ebenda

[66] Ebenda

5. Templates kopieren und ggf. anpassen

Vorteile einer manuellen Migration liegen vor allem in der Qualität der Ergebnisse. Nachteilig wirken sich bei dieser Strategie allerdings die Geschwindigkeit und damit auch die Dauer des Content Freeze aus. Dem kann bei größeren Seiten durch ein automatisiertes Vorgehen Abhilfe geschaffen werden.

Automatisierte Migration

Der Umfang und die Komplexität der Seiteninhalte macht eine automatisierte Content-Migration bei großen Webauftritten häufig unumgänglich. Bei einer automatisierten Migration werden Software-Tools benutzt, um Content von einer bestehenden Seite auf eine neue zu übertragen.

Dies benötigt in der Regel eine im Vergleich zur manuellen Migration höhere Vorbereitungszeit. Da sich nur Inhalte in strukturierter Form, z.B. im XML oder CSV Format, automatisiert übertragen lassen, müssen vorhandene Dateien, Dokumente und Seiten zunächst in eine systematische und einheitliche Struktur gebracht werden (siehe auch Kapitel 3.3 Austauschformate).[67]

Im nächsten Schritt erfolgt die Entwicklung und/oder Auswahl von geeigneten Migrations-Tools, wie z.B. Skripten. Diese übersetzen die Struktur der Daten des alten CMS in die Struktur des neuen und fügen die Inhalte automatisch in die richtigen Bereiche ein.[68] Je populärer die beiden beteiligten Content Management Systeme sind, desto grösser ist die Wahrscheinlichkeit, dass bereits Skripte für diese Aufgaben existieren.[69] Abhängig von den Anforderungen einer Migration, ist eine Eigenentwicklung oder Anpassung des Migrationsskriptes allerdings häufig unumgänglich. Sind alle Vorbereitungen abgeschlossen, können die Datenbankinhalte in die Datenbankstruktur des neuen CMS übertragen werden.[70]

Durch diese Vorbereitung lässt sich die eigentliche Migration der Daten beschleunigen und die Dauer des Content Freeze drastisch verkürzen. [71]

Die Vorteile der automatisierten Content-Migration liegen eindeutig in der Schnelligkeit und der reduzierten Dauer des Content Freezes. Bei den Nachteilen ist zunächst das benötigte Knowhow zu nennen. Eine automatisierte Migration lässt sich oftmals nicht ohne fundierte

[67] Short 2008
[68] Ebenda
[69] Kelleher 2013
[70] Ebenda
[71] ADselect GmbH 2014

Kenntnisse im Bereich CMS, Datenbanken und verschiedenen Programmiersprachen durchführen. Ein weiterer Nachteil ist oft die Qualität der Ergebnisse. Nicht selten muss nach einer automatisierten Migration manuell nachgearbeitet werden.

Hybrid-Migration

Durch eine automatisierte Content-Migration entfallen “Copy&Paste”-Arbeiten größtenteils, in vielen Fällen wird jedoch eine Kombination aus Automatisierter und Manueller Migration verwendet. Dies kann folgende Gründe haben:

- Unstrukturierter Content lässt sich größtenteils nur manuell übertragen.[72]
- Grundsätzlich empfiehlt sich nach einer automatisierten Migration eine manuelle Anpassung zur weiteren Optimierung der Seiteninhalte. Eine stichprobenartige Prüfung deckt hier mögliche Fehlerquellen auf.[73]
- Aufgrund sehr unterschiedlicher Strukturen der beiden Systeme lassen sich nur Teile der Inhalte automatisiert übertragen und es muss manuell nachgearbeitet werden.

Kommen bei einer Content Migration sowohl manuelle als auch automatisierte Migrationstechniken zum Einsatz, wird diese Strategie oftmals als Hybrid-Migration bezeichnet.[74]

3.2 Standards (Philipp Richter / Maurice Görges)

Um einen einfachen und einheitlichen Zugriff auf strukturierte Daten zu ermöglichen, sind Standards erforderlich. Sie definieren die Struktur von Daten und beschreiben gemeinsame Protokolle die für den Zugriff auf diese Daten genutzt werden können. Um bei einer Migration von Content Management Systemen diese Standards nutzen zu, müssen sowohl das alte als auch das neue CMS diese Technologien unterstützen. Die folgenden beiden Standards sind im Umfeld von Content Management Systemen zu berücksichtigen:

CMIS

Die Content Management Interoperability Services (CMIS) sind ein offener Standard, die es verschiedenen Content Management Systemen ermöglichen über das Internet zu interagieren. CMIS definiert insbesondere eine Abstraktionsschicht, mit verschiedenen Web-Protokollen, Datenmodellen und Services, über die Daten zwischen Content Management Systemen ausgetauscht werden können. Durch die Verwendung von REST (Representational State Transfer) und SOAP (Simple Object Access Protocol) ist CMIS Programmiersprachen unabhängig. Ein weiterer Vorteil ist, dass CMIS Web Service und Content trennt und somit auch

[72] Branson 2011
[73] Ebenda
[74] Ebenda

auf bereits bestehende Content Repositories zugreifen kann[75]. CMIS wird derzeit von Folgenden OpenSource Content Management Systemen unterstützt[76]:

Alfresco	*Magnolia*	*OpenCms*
dotCMS	*Drupal*	*Hippo CMS*
LibreOffice	*Liferay*	*Sense*
SilverStripe CMS	*TYPO3*	*WordPress*

Tabelle 2 CMIS Kompatible Systeme[77]

JCR

Content Repository for Java Technology API (JCR) ist eine Spezifikation für eine Java-Plattform-API, mit deren Hilfe einheitlich auf Inhalte zugegriffen werden kann.[78] Content Repositories werden von den unterschiedlichsten Informationssystemen genutzt, die beliebige Dokumente zusammen mit Metadaten verwalten, wie z. B. Web Content Management Systeme oder Enterprise Content Management Systeme.

Ziel der API ist es, die vielen Content-Inseln mit den jeweils eigenen, proprietären Schnittstellen zukünftig über eine Schnittstelle anzusteuern. So kann eine Anwendung, die die JCR-API nutzt, den in ihr verwendeten Content in beliebigen anderen, JCR-konforme Content Repositories speichern. Damit werden Anwendungen als auch die Anwender selbst unabhängiger von einem Content-Repository-Hersteller.[79] Ein wichtiger Aspekt zur Unterstützung der genannten Unabhängigkeit ist die Migration der Inhalte eines Content Repository mittels Import und Export der Daten in einem standardisierten XML-Format.

Da es sich bei JCR um eine Spezifikation für eine Java-Plattform-API handelt, wird sie größtenteils nur von Java-basierten Content Management Systemen unterstützt. Aufgrund der großen Vorteile dieses Standards, wird allerdings auch an Implementierungen für andere Programmiersprachen gearbeitet, z.B. TYPO3CR, eine PHP Implementierung der JCR API.

[75] Oasis 2014
[76] Wikimedia Foundation, Inc. 2014b
[77] Wikimedia Foundation, Inc. 2014b
[78] Oracle Corporation 2013
[79] Magnolia International Ltd. 2013a

3.3 Austauschformate (Philipp Richter / Maurice Görges)

Bei der Migration von Inhalt zwischen Content Management Systemen, ist es entscheidend, dass einheitliche und strukturierte Dateiformate zur Verfügung stehen, die von beiden Systemen unterstützt werden. Die folgenden Formate sollten von jedem CMS unterstützt werden und spielen bei der Migration eine entscheidende Rolle.

HTML & CSS

Die Hypertext Markup Language (HTML), ist eine textbasierte Auszeichnungssprache zur Strukturierung von Inhalten wie Texten, Bildern und Hyperlinks in Dokumenten. HTML-Dokumente werden von einem Webbrowser dargestellt und bilden damit die Grundlage des Internets. Neben den vom Browser angezeigten Inhalten einer Webseite enthält HTML zusätzliche Angaben in Form von Metainformationen, die z. B. über die im Text verwendete Sprache oder den Autor Auskunft geben oder den Inhalt des Textes zusammenfassen.[80]

HTML wurde im Laufe der Jahre um durch die Definition von Cascading Style Sheets (CSS) erweitert. Sie befassen sich mit der Gestaltung des Dokuments orientieren sich damit an der ursprünglichen Idee der Trennung von Struktur und Layout. So soll das Aussehen bzw. die Darstellung des Dokuments in einer separaten Datei, dem sogenannten Stylesheet, festgelegt werden. [81]

Im Umfeld von Content Management Systemen kommen HTML und CSS bei der Definition von sogenannten Templates zum Einsatz. Sie bestimmen das spätere Aussehen der Webseite und stellen das Framework da, in dem die Inhalte dargestellt werden. Bei einer Migration von Content Management Systemen lassen sich diese Templates häufig relativ einfach übernehmen. Oft muss jedoch eine Anpassung der Bezeichnung der verschiedenen Felder – in Übereinstimmung mit der Syntax des neuen CMS – vorgenommen werden.

XML

Die eXtensible Markup Language (XML) ist eine Sprache zur Beschreibung der Struktur von Inhalten. Mit Hilfe von Markierungen innerhalb einer XML-Datei, so genannter Markups, wird der Verwendungszweck bzw. die Bedeutung der einzelnen Informationselemente der Datei definiert.[82] XML unterscheidet sich von HTML unter anderem dadurch, dass es in der Regel nur die Struktur von Inhalten beschreibt und keine Informationen über Formatierungen oder das Layout der Inhalte enthält.

---

[80] Wikimedia Foundation, Inc. 2014c
[81] Wikimedia Foundation, Inc. 2014a
[82] Rothfuss, Ried 2003, S.40

Damit ein IT-System eine importierte XML-Datei richtig interpretieren kann, wird in so genannten XML-Schemata die Syntax definiert, nach der die XML-Datei aufgebaut wurde. Diese unterscheiden sich häufig von CMS zu CMS. Um die XML-Dateien in ein für das Ziel-CMS lesbares Format zu übersetzen, kommen sogenannte XML-Parser zum Einsatz. Verwenden beide Content Management Systeme den JCR Standard, lassen sich die Inhalte der CMS aufgrund der gleichen Struktur der Daten beliebig importieren und exportieren.[83]

CSV

Das Dateiformat CSV (Comma-Separated Values) beschreibt den Aufbau einer Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten. In CSV-Dateien können Tabellen oder eine Liste unterschiedlich langer Listen abgebildet werden. Kompliziertere, beispielsweise geschachtelte Datenstrukturen, können durch zusätzliche Regeln oder in verketteten CSV-Dateien gespeichert werden. Um sie in einer Datei abzuspeichern, eignen sich jedoch andere Formate, wie z.B. XML, besser.[84]

Im Migrationsumfeld von Content Management Systemen wird CSV häufig für den Export und Import einfach strukturierter Daten (wie z.B. Benutzerdaten) verwendet. Wie bei XML ist der Unterschiedliche Aufbau der Daten bei unterschiedlichen CMS zu beachten. Unter Umständen muss auch hier zunächst ein sog. Mapping, also eine Restrukturierung der Daten stattfinden.

3.4 Tools, Plug-Ins, Erweiterungen (Philipp Richter / Maurice Görges)

Zusätzlich zu den Standard-Funktionalitäten von Content Management Systemen lassen sich in den meisten Fällen zusätzliche Features installieren, die den Funktionsumfang erweitern. Auch zum Thema Migration gibt es unzählige Erweiterungen. Meist handelt es sich hierbei um sehr spezifische Funktionen, wie z.B. ein Tool zum Exportieren von Benutzerdaten als CSV-Datei. Generell gilt: Je grösser die Community eines Open Source CMS und die damit verbundene Auswahl an Tools, Plug-Ins und Erweiterungen, desto grösser ist die Wahrscheinlichkeit, dass bei einem Migrationsprojekt auf bereits bestehende Lösungen zurückgegriffen werden. Aufgrund des enormen Angebotes und der spezialisierten Angebote verschiedener CMS, muss die Auswahl potentiell relevanter Erweiterungen immer im Einzelfall erfolgen.

[83] Bundesverwaltungsamt 2004

[84] Rothfuss, Ried 2003, S.40

3.5 Skripte (Philipp Richter / Maurice Görges)

Skripte sind ein wichtiges Werkzeug bei der Migration von Content Management Systemen und v.a. bei größeren Projekten unverzichtbar. Computerskripte werden geschrieben, um den Inhalt von ganzen Seiten oder bestimmten Feldern (wie z.B. Seitentitel, Text, Metadaten, oder andere Inhalte) in die entsprechenden Felder des neuen CMS zu übertragen. Die Hauptaufgabe besteht hierbei in dem bereits angesprochenen „Mapping“ der Inhalte.[85] Durch die häufig abweichende Struktur der Inhalte, muss durch ein Skript sichergestellt werden, dass alle Inhalte in die korrekten Felder übertragen werden. Die Entwicklung geschieht oft anhand einer Beispielseite und lässt sich dann bei entsprechender Vorbereitung beliebig oft reproduzieren. Entscheidend ist, dass alle zu migrierenden Daten in einheitlich strukturierter Form vorliegen.

Bei beliebten und weit verbreiteten CMS Systemen gibt es oftmals bereits von der Community vorgefertigte Skripte. In Abhängigkeit von der Komplexität der Migration und der vorhandenen Inhalte ist eine Anpassung und / oder Eigenentwicklung der Skripte oft unumgänglich.

3.6 Migration as a Service (Philipp Richter / Maurice Görges)

Inhaltsmigration ist ein komplexer Prozess. Es gibt viele Aktivitäten und Abhängigkeiten, die sorgfältig verwaltet werden müssen, um einen nahtlosen Übergang von einer Plattform zur anderen zu gewährleisten. Jede Phase der Migration, von der Erfassung der Anforderungen bis hin zur eigentlichen Migration, birgt Risiken. Jegliche Fehler können Auswirkungen auf die Projektabwicklung haben und Ausfälle oder Verzögerungen verursachen. Darum ist in Migrationsprojekten, ohne die Betreuung und Verwaltung durch einen Fachmann, oft mit zeitlichen Verzögerungen und Budget Überschreitungen zu rechnen.

Diese Problematik wurde von einigen Unternehmen erkannt und durch professionelle Beratungs- und Durchführungsservices zur Content Migration adressiert. Dieser Service wird auch als „Migration as a Service“ verstanden.

Bei der Auswahl eines Migrationsdienstleisters sollten Unternehmen auf vor allem auf Referenzen und Zertifizierungen achten. Manche Anbieter können Erfahrung aus über 200 erfolgreichen Migrationen und einer Zertifizierung als "Trusted Advisor" vorweisen und sind dadurch für eine Migration im Unternehmensumfeld besonders geeignet.

[85] Magnolia International Ltd. 2013b

Zu den Leistungen zählen meistens die unternehmensinterne Beratung und Unterstützung des Projektteams während einer Migration.[86] Zudem wird oftmals auch die Übernahme des gesamten Migrations-Projekt-Managements als Service angeboten. Dabei können Inhalte sowohl von einem unstrukturierten Datenspeicher zu einem Content Management System, als auch von CMS zu CMS migriert werden.

Das Unternehmen „CMS2CMS“, als ein Dienstleiter *im Migration as a Service* Umfeld, bietet Leistungen, wie die automatisierte CMS & Forum Migration unter anderem als Webtool an.[87] Dabei wird die Migration auf Systeme wie Joomla!, Typo3, Drupal, WordPress, phpBB, vBulletin und Kunena unterstützt. CMS2CMS bietet eine kostenfreie Demo für potenzielle Interessenten um sich von der Arbeitsweise überzeugen zu können.[88] Zudem gibt es eine Preiskalkulation um die ungefähren Kosten einer Migration abschätzen zu können.[89]

4 Anwendung (Philipp Richter / Maurice Görges)

In diesem Kapitel soll zunächst ein Kriterienkatalog anhand der in Kapitel 3 erarbeiteten Strategien erstellt werden. Im Anschluss daran werden die drei ausgewählten Content Management Systeme (TYPO3, Joomla!, Magnolia CMS) anhand der gewählten Kriterien bewertet.

4.1 Aufbau eines Kriterienkataloges zur Migrierbarkeit von CMS Systemen (Philipp Richter / Maurice Görges)

Der Kriterienkatalog unterteilt sich in fünf Kategorien und orientiert sich dabei an den zuvor ausgearbeiteten Migrationsstrategien. Im Einzelnen sollen die Kategorien Allgemeines, Standards, Austauschformate, Tools, Plug-Ins, Erweiterungen, und Migration as a Service betrachtet werden.

Allgemein

In dieser Kategorie soll zunächst die Anzahl der Entwicklern (oder Contributors), Zukunftsaussichten und die überwiegende Programmiersprache untersucht werden.

Es ist davon auszugehen, dass mit steigender Verbreitung und Größe der Community auch das Ausmaß der verfügbaren Vorarbeit zum Thema Migration steigt. Das Kriterium verwendete

[86] T-Systems Limited 2014
[87] MagneticOne 2014b
[88] MagneticOne 2014a
[89] MagneticOne 2014c

Programmiersprache gibt Aufschluss über die Wahrscheinlichkeit der Unterstützung bestimmter Standards (z.B. JCR).

Standards

In dieser Kategorie soll die Unterstützung von JCR und CMIS Standards geprüft werden.

Austauschformate

In der Kategorie Austauschformate soll zunächst überprüft werden, ob sich Inhalte und User-Daten generell importieren und exportieren lassen. Danach wird die Unterstützung verschiedener Datei-Typen untersucht.

Tools, Plug-Ins, Erweiterungen

Aufgrund der enormen Anzahl verfügbarer Erweiterungen kann in dieser Kategorie nur die generelle Verfügbarkeit bewertet werden. Es soll zunächst die generelle Erweiterbarkeit geprüft werden und anschließend die Anzahl verfügbarer Plug-Ins bestimmt werden. Es kann davon ausgegangen werden, dass bei steigender Anzahl an verfügbaren Erweiterungen auch die Wahrscheinlichkeit steigt, dass Migrationsspezifische Plug-Ins angeboten werden.

Migration as a Service

Da die Angebote im Bereich Migration as a Service meist sehr Plattform spezifisch und / oder individuell sind, lassen sich hier nur exemplarische Kriterien untersuchen. Zunächst soll die Analyse von Google Suchergebnissen zu der Anfrage „ *CMS* Migration“ (z.B. „TYPO3 Migration“) Aufschluss über die Verbreitung dieses Themas geben. Danach wird exemplarisch die Unterstützung durch verschiedene Anbieter geprüft.

Die folgende Tabelle zeigt den ausgearbeiteten Kriterienkatalog.

Kriterium	Wichtigkeit
Allgemein	
Anzahl an Entwicklern	+
Zukunftsaussichten	-
Programmiersprache	o
Standards	
JCR konforme Architektur	o
CMIS Support	o
Austauschformate	
Import / Export Funktion für Content	+
Content exportierbar als XML-Datei	+
HTML / CSS basierte Templates	o
Import/Export Funktion für Benutzerdaten	+
Tools, Plug-Ins, Erweiterungen	
Durch zusätzliche Plug-Ins erweiterbar	+
Anzahl verfügbarer Plug-Ins/Erweiterungen	+
Migration as a Service	
Google Suchergebnisse " 'CMS' Migration"	-
CMS2CMS.com Unterstützung	-

Tabelle 3 Ausgearbeiteter Kriterienkatalog

Legende zur Wichtigkeit:

\+ *Sehr wichtig* o *Neutral* - *Weniger wichtig*

4.2 Exemplarische Bewertung ausgewählter Content Management Systeme (Philipp Richter / Maurice Görges)

In einem letzten Schritt wurden nun die ausgewählten CMS anhand der aufgestellten Kriterien bewertet. Die folgende Tabelle zeigt die Ergebnisse.

Kriterium	Wichtigkeit	TYPO3	Joomla!	Magnolia
Allgemein				
Anzahl an Entwicklern	+	158	216	33
Zukunftsaussichten	-	+	+	+
Programmiersprache	o	PHP	PHP	Java
Standards				
JCR konforme Architektur	o	TYPO3CR	-	JSR 168, JSR 170
CMIS Support	o	Ja	Nein	Ja (Modul)
Austauschformate				
Import / Export Funktion für Content	+	Ja	Ja (Erweiterung)	Ja
Content exportierbar als XML-Datei	+	Ja	Ja (Erweiterung)	Ja
HTML / CSS basierte Templates	o	Ja	Ja	Ja
Import/Export Funktion für Benutzerdaten	+	Ja	Ja (Erweiterung)	Ja
Tools, Plug-Ins, Erweiterungen				
Durch zusätzliche Plug-Ins erweiterbar	+	Ja	Ja	Ja
Anzahl verfügbarer Plug-Ins/Erweiterungen	+	6045	7590	49
Migration as a Service				
Google Suchergebnisse " 'CMS' Migration"	-	2,990	80,200	14,100
CMS2CMS.com unterstuetzung	-	Ja	Ja	Nein

Tabelle 4 Exemplarische Bewertung von TYPO3, Joomla!, Magnolia CMS

Joomla!

Joomla! ist ein PHP basiertes CMS, das von allen drei Content Management Systemen auf die größte Entwicklerbasis zurückgreifen kann. Joomla! unterstützt keine der beiden vorgestellten Standards. Standardmäßig werden nur wenige Import und Export Funktionen unterstützt. Durch die sehr hohe Anzahl verfügbarer Extensions, kann Joomla! diesen Nachteil jedoch wieder ausgleichen. Die Migration von Joomla! scheint zudem eine sehr hohe Nachfrage im World Wide Web zu haben und wird auch von Anbietern wie CMS2CMS.com unterstützt.

TYPO3

TYPO3 zeichnet sich zunächst durch eine große Anzahl an Entwicklern und einer großen Community aus. Darüber hinaus hat es von allen drei getesteten Systemen den größten Marktanteil. Als eines der wenigen PHP-basierten Content Management Systemen unterstützt es in der neuesten Version, TYPO3 Neos, auch den JCR Standard. Inhalte, User und Templates lassen sich in eine Reihe von Formaten exportieren. Für den Import dieser Daten in ein anderes CMS müssen diese allerdings zunächst in die benötigte Form gebracht werden. Durch die große Community und die hohe Anzahl an bereits verfügbaren Erweiterungen können hierfür eventuell bereits bestehende Lösungen verwendet werden. Bei einem Umstieg auf ein unterstütztes System, kann außerdem auf CMS2CMS.com als Service Provider zurückgegriffen werden.

Magnolia

Magnolia ist ein Java basiertes CMS, das von den getesteten CMS die geringste Anzahl an Entwicklern und den geringsten Marktanteil hat. Dies soll aber keinesfalls über den Funktionsumfang von Magnolia im Migrationskontext hinwegtäuschen. Als einziges untersuchtes CMS unterstützt Magnolia nativ JCR und CMIS Standards und ist damit für eine Migration bestens gerüstet. Zusätzlich wird der Import und Export von Inhalten, Templates und Usern unterstützt. Obwohl Magnolia prinzipiell erweiterbar ist, werden offiziell nur 49 Extensions angeboten. Diesen Nachteil kann Magnolia allerdings durch den von Haus aus sehr großen Funktionsumfang wettmachen.

5 Fazit (Philipp Richter / Maurice Görges)

Open Source Content-Management-Systeme haben sich auch im Enterprise-Umfeld etabliert und sind inzwischen in Anwendungsbereiche vorgedrungen, die bisher von kommerziellen Produkten besetzt waren. Im Open Source Umfeld existiert eine ganze Reihe von Implementierungen von CMS-Systemen. Die Entscheidung für ein CMS-System bedeutet gegebenenfalls aber, dass man sich auf lange Zeit einem System „verschreibt“. Kommen neue Systeme auf den Markt oder wird die Entwicklung des CMS von der Community nicht weiterverfolgt, hat das in einem Unternehmenskontext hohe Kosten zur Folge, wenn man die erstellten Inhalte nicht retten und in ein neues CMS überführen kann.

Ziel der Arbeit war es, zu untersuchen, inwieweit bestehende CMS-Implementierungen im Open Source-Bereich die Migration von Inhalten auf ein neues System zu unterstützen, welche Optionen es gibt und wie diese bewertet werden können.

Content-Management-Systeme zeichnen sich durch die Trennung von Inhalten und Darstellung aus. Ausgehend von diesem Aufbau, gilt es bei einer Migration vor allem folgenden Daten zu migrieren: Inhalte (Texte, Bilder, Links, etc.), Templates zur Darstellung der Website, User-Daten und Meta-Daten.

Generell kann man hierbei manuell oder automatisiert vorgehen. Während bei kleineren Webseiten mit 300 – 500 eine manuelle Migration sinnvoll ist, ist im Unternehmensbereich aufgrund kürzerer Ausfallzeiten fast immer eine automatisierte Migration vorzuziehen. Meistens muss allerdings auch bei einer automatisierten Vorgehensweise manuell nachgearbeitet werden (Hybrid – Migration).

Es stehen verschiedene Technologien und Services zur Verfügung, die bei der Migration eines CMS beachtet werden sollten. Standards wie JCR und CMIS erleichtern die Kommunikation zwischen CMS Systemen und vereinheitlichen die Struktur von Inhalten, werden allerdings nur von wenigen Produkten unterstützt. HTML, CSS, XML und CSV sind gängige Dateiformate für den Austausch von Daten bei der Migration von Content-Management-Systemen und werden von den meisten Systemen unterstützt. Häufig muss bei der Migration jedoch die Struktur der Daten an das neue CMS angepasst werden. Hierfür werden sogenannte Skripte eingesetzt. Diese werden verwendet, um den Inhalt von ganzen Seiten oder bestimmten Feldern (wie z.B. Seitentitel, Text, Metadaten, oder andere Inhalte) in die entsprechenden Felder des neuen CMS zu übertragen. Zuletzt gibt es Anbieter, die die Migration als Dienstleistung übernehmen. Hierbei gibt es verschiedene Lösungen – vom der standardisierten Online-Tool bis hin zur komplett maßgeschneiderten Lösung.

Ausgehend von diesen Strategien wurde ein Kriterienkatalog entwickelt, der Systeme anhand der Kategorien Allgemeines, Standards, Austauschformate, Tools, Plug-Ins, Erweiterungen, und Migration as a Service bewertet.

In einer Vorauswahl unternehmensrelevanter Content-Management-Systeme wurden TYPO3, Joomla! und Magnolia für eine exemplarische Bewertung ausgewählt. Es hat sich gezeigt, dass alle drei ihre individuellen Stärken und Schwächen haben.

Joomla! unterstützt keine der untersuchten Standards. Viele Export und Import Funktionen werden erst durch Erweiterungen unterstützt. Diese Nachteile kann Joomla! durch große Verbreitung und Unterstützung durch Migrationsportale wettmachen.

TYPO3 zeichnet sich durch große Verbreitung, eine hohe Anzahl an Erweiterungen und die Unterstützung der beurteilten Import und Export Funktionen aus. Obwohl es sich um ein PHP basiertes System handelt, wird der JCR Standard unterstützt.

Magnolia unterstützt sowohl CMIS und JCR als auch alle untersuchten Import und Export Funktionen. Das System wird allerdings von einer vergleichsweise kleinen Community und Entwicklerbasis unterstützt.

Abschließend lässt sich festhalten, dass es sich bei der Migration von Content-Management-Systemen im Unternehmensumfeld um ein sehr komplexes Projekt handelt. Zwar gibt es einige wenige Standards, die jedoch bei weitem nicht von allen Systemen unterstützt werden. So hängen der Aufwand und die Vorgehensweise schlussendlich immer sehr von beiden beteiligten Systemen ab. Eine konkrete Produktempfehlung kann deshalb nicht bzw. nur eingeschränkt gegeben werden. Die Migration von Content-Management-Systemen erfordert in den meisten Fällen ein sehr gutes Verständnis der beiden Systeme, von Datenbanken und Programmiersprachen im Allgemeinen. Um einen reibungslosen Ablauf zu gewährleisten ist deshalb eine Zusammenarbeit mit Spezialisten bei einer Migration im Unternehmensumfeld immer ratsam.

Quellenverzeichnis

Literaturquellen

Baumgartner, Peter (1995): Didaktische Anforderungen an (multimediale) Bildungssoftware. Information und Lernen mit Multimedia. Edited by L. J. Issing, P. Klimsa. Weinheim: Psychologie.

Bodendorf, Freimut (2006): Daten- und Wissensmanagement. 2., aktual.u. erw. Aufl. Berlin: Springer. Available online at http://www.worldcat.org/oclc/62229930.

Christ, Oliver (2003): Content-Management in der Praxis. Erfolgreicher Aufbau und Betrieb unternehmensweiter Portale. Berlin [u.a.]: Springer. Available online at http://www.worldcat.org/oclc/248849071.

Fröschle, Hans-Peter; Reich, Siegfried (2007): Enterprise Content Management. Heidelberg: Dpunkt-Verl. (44.2007,258). Available online at http://www.worldcat.org/oclc/254747809.

Hildebrand, Knut (2007): IT-Industrialisierung & Migration. Heidelberg: Dpunkt-Verl. (H. 257. Jg. 44). Available online at http://www.worldcat.org/oclc/263418781.

Hüttenegger, Georg (2006): Open Source Knowledge Management. Berlin: Springer. Available online at http://www.worldcat.org/oclc/403810675.

Jischke, Holger (2012): ECM Blog. Enterprise-Anforderungen an ECM-Lösungen. Edited by Zöller & Partner. Stuttgart. Available online at http://www.zoeller.de/index.php/blog/item/87-enterprise-anforderungenanecm-loesungen, updated on 11/1/2012, checked on 12/17/2013.

Kocakurt, Demet (2010): Web-Content-Management-Systeme im Wandel der Zeit. Bewertung von Open-Source-Anwendungen für kleine und mittelständische Unternehmen. München: GRIN Verlag.

Krüger, Joerg-Dennis; Kopp, Matthias (2002): Web-Content managen. Professioneller Einsatz von Content-Management-Systemen. München: Markt & Technik Verlag. Available online at http://www.worldcat.org/oclc/52477986.

Mauthe, Andreas Ulrich; Thomas, Peter (2003): Multimedia content management. Chichester: Wiley. Available online at http://www.worldcat.org/oclc/52358309.

Meier, A. (1995): Wer braucht Kriterienkataloge? Evaluation multimedialer Lernprogramme und Lernkonzepte. Edited by P. Schenkel, H. Holz. Nürnberg: BW Bildung und Wissen Verlag und Software GmbH.

Riggert, Wolfgang (2009): Enterprise content Management. 1. Aufl. Wiesbaden: Vieweg + Teubner. Available online at http://www.worldcat.org/oclc/444839652.

Rothfuss, Gunther; Ried, Christian (2003): Content Management mit XML. Grundlagen und Anwendungen. 2., überarb. Aufl. Berlin [u.a.]: Springer. Available online at http://www.worldcat.org/oclc/51712010.

Spörrer, Stefan (2009): Content Management Systeme. Begriffsstruktur und Praxisbeispiel. Köln: Kölner Wissenschaftsverlag. Available online at http://www.worldcat.org/oclc/845630523.

Vom Brocke, Jan; Simons, Alexander (Eds.) (2013): Enterprise Content Management in Information Systems Research. Foundations, Methods and Cases. Berlin: Springer Berlin. Available online at http://www.worldcat.org/oclc/864536177.

Zschau, Oliver; Traub, Dennis; Zahradka, Rik (2002): Web Content Management. Websites professionell planen und betreiben ; [mit detaillierter Marktübersicht ; komplett überarbeitete und erweiterte Neuauflage]. 2., überarb. und erw. Aufl. Bonn: Galileo Press. Available online at http://www.worldcat.org/oclc/50973471

Verzeichnis der Internet- und Intranet-Quellen

ADselect GmbH (2014): Automatisierte Migration der Seiten. contentmanager.de. Available online at http://www.contentmanager.de/wp-content/uploads/Automatisierte-Migration-der-Seiten.jpg, checked on 1/17/2014.

Altassian (2012): Magnolia Community Wiki. Available online at http://wiki.magnolia-cms.com/download/attachments/37847123/installed-modules.png?version=1&modificationDate=1265891268000, updated on 2/4/2012, checked on 1/15/2014.

Apps Kostenloser (2014): Joomla! Administratior. Available online at http://darcy-norman.net/images/joomla-screenshots/joomla-1-control-panel.png, checked on 1/18/2014.

Association for Information and Image Management (Ed.) (2013): What is enterprise content management (ECM)? Available online at http://www.aiim.org/What-is-ECM-Enterprise-Content-Management, checked on 11/28/2013.

Black Duck Inc. (2014a): Joomla! Ohloh.net. Available online at http://www.ohloh.net/p/joomla, checked on 1/15/2014.

Black Duck Inc. (2014b): Typo3. Ohloh.net. Available online at http://www.ohloh.net/p?ref=homepage&q=typo3, checked on 1/15/2014.

Bösiger Engineering AG (2014): CMS Vergleich. CMS Marktanteile & CMS Budgets. Webkalkulator.com. Available online at http://www.webkalkulator.com/cmsvergleich, updated on 1/28/2014, checked on 1/16/2013.

Branson, Aaron (2011): Moving to a New CMS? Don't Overlook the Content Migration Plan! Roundedcube, Inc. Available online at http://www.roundedcube.com/WhatsNew/Blog/moving-to-a-new-cms-dont-overlook-the-content-migration-plan, updated on 5/26/2011.

Bundesverwaltungsamt (2004): Kompetenzzentrum CMS Toolbox. Leitfaden Information Sharing. Available online at http://www.bva.bund.de/SharedDocs/Downloads/DE/BIT/GSB/Information_Sharing.pdf?__blob=publicationFile&v=2, updated on 12/8/2004, checked on 1/25/2014.

CMS Crawler (2014): Browse by tool. The complete list. CMS Crawler. Available online at http://www.cmscrawler.com/tool/, updated on 1/22/2014, checked on 1/22/2014.

CMS Matrix: Compare Content Management Systems. Available online at http://www.cmsmatrix.org/, checked on 12/19/2013.

der-webentwickler.net (2011): Typo3 Backend. Available online at http://www.der-webentwickler.net/wp-content/uploads/2010/07/typo3-seitenbaum.jpg, checked on 1/15/2014.

Hazlett, Maril (2013): Best Practices for a successful Content Migration. Part 1. degigital.com. Available online at http://www.degdigital.com/blog/best-practices-for-a-successful-content-migration-part-i/, updated on 9/4/2013, checked on 12/15/2013.

HowTo.gov (2013): Migrating Content to a CMS. Available online at http://www.howto.gov/web-content/technology/content-management-systems/migrating-content-to-a-cms, updated on 10/31/2013, checked on 12/2/2013.

JandBeyond e.V. (2014): Was ist Joomla!? Available online at http://www.joomla.de/joomla-entdecken, checked on 1/10/2014.

Kampffmeyer, Ulrich (2011a): Capture. Available online at http://project-consult.net/ecm/content/capture, checked on 12/17/2013.

Kampffmeyer, Ulrich (2011b): Deliver. Available online at http://project-consult.net/ecm/content/deliver, checked on 12/12/2013.

Kampffmeyer, Ulrich (2011c): ECM Komponenten. Available online at http://project-consult.net/ecm/content/ecm_kategorien, checked on 12/9/2013.

Kampffmeyer, Ulrich (2011d): Manage. Available online at http://project-consult.net/ecm/content/manage, checked on 12/12/2013.

Kampffmeyer, Ulrich (2011e): Preserve. Available online at http://project-consult.net/ecm/content/preserve, checked on 12/12/2013.

Kampffmeyer, Ulrich (2011f): Store. Available online at http://project-consult.net/ecm/content/store, checked on 12/9/2013.

Kelleher, Michael (2013): 6 Key Considerations in Content Migration. Simpler Media Group, Inc. cmswire.com. Available online at http://www.cmswire.com/cms/web-publishing/no-small-task-migrating-content-to-a-new-cms-002437.php, updated on 10/2/2013.

MagneticOne (2014a): CMS2CMS Services. Free Demo. With assistance of CMS2CMS. Available online at http://app.cms2cms.com/wizard/, checked on 12/13/2013.

MagneticOne (2014b): CMS2CMS Services. Provided Services. With assistance of CMS2CMS. Available online at http://www.cms2cms.com/provided-services/, checked on 12/13/2013.

MagneticOne (2014c): CMS2CMS Services. Pricing. With assistance of CMS2CMS. Available online at http://www.cms2cms.com/pricing/, checked on 12/13/2013.

Magnolia International Ltd. (2013a): Magnolia Documentation. Content storage and structure. Magnolia Community. Available online at http://documentation.magnolia-cms.com/display/DOCS/Content+storage+and+structure, updated on 11/21/2013, checked on 1/21/2014.

Magnolia International Ltd. (2013b): Magnolia Documentation. Importing and exporting. Magnolia Community. Available online at http://documentation.magnolia-cms.com/display/DOCS/Importing+and+exporting#Importingandexporting-Sitemigration, updated on 11/21/2013, checked on 1/23/2014.

Magnolia International Ltd. (2014): Magnolia CMS. Available online at http://www.magnolia-cms.com/product.html, checked on 1/15/2014.

Oasis (2014): OASIS Content Management Interoperability Services (CMIS) Technical Committee. Available online at https://www.oasis-open.org/committees/cmis/charter.php, checked on 08.01.204.

Onasch, Lars (2006): Spezielle Anforderungen an Content Management Systeme für den Mittelstand. Edited by contentmanager.de. Available online at http://www.contentmanager.de/magazin/spezielle_anforderungen_an_content_management_systeme_fuer.html, updated on 3/6/2006, checked on 12/17/2013.

Open Source Initiative (2014): Open Source Definition. Available online at http://opensource.org/docs/osd, checked on 1/11/2014.

Open Source Matters Inc. (2014): The Joomla! Extension Directory. Available online at http://extensions.joomla.org/, checked on 1/10/2014.

Oracle Corporation (2013): JSR 93: JavaTM API for XML Registries 1.0 (JAXR). Java Community Process. Available online at https://www.jcp.org/en/jsr/detail?id=93, checked on 1/13/2014.

Short, Kyle (2008): No Small Task: Migrating Content to a New CMS. Simpler Media Group, Inc. cmswire.com. Available online at http://www.cmswire.com/cms/web-publishing/no-small-task-migrating-content-to-a-new-cms-002437.php, updated on 3/18/2008.

Springer Gabler Wirtschaftslexikon (Ed.): Groupware. Available online at http://wirtschaftslexikon.gabler.de/Archiv/76195/groupware-v7.html, checked on 12/12/2013.

T-Systems Limited (2014): CMS Migration Program Management. With assistance of Vamosa Enterprise Content Governance. Available online at http://www.vamosa.com/cms-migration-program-management-a414, checked on 1/22/2014.

Typo3 Association (2014a): Typo3 Core Development Team. Available online at http://typo3.org/teams/core-development-team/, checked on 1/15/2014.

Typo3 Association (2014b): Typo3 Features. Available online at http://typo3.org/about/features/, checked on 1/15/2014.

Wikimedia Foundation, Inc. (2014a): Cascading Style Sheets. Wikipedia.org. Available online at http://de.wikipedia.org/wiki/Cascading_Style_Sheets, updated on 1/15/2014, checked on 12/19/2013.

Wikimedia Foundation, Inc. (2014b): Content Management Interoperability Services. CMIS Servers. Wikipedia.org. Available online at http://en.wikipedia.org/wiki/Content_Management_Interoperability_Services#Client_applications, updated on 1/24/2014, checked on 12/8/2013.

Wikimedia Foundation, Inc. (2014c): Hypertext Markup Language. Wikipedia.org. Available online at http://de.wikipedia.org/wiki/Hypertext_Markup_Language, updated on 1/18/2014, checked on 12/19/2013.

Wikimedia Foundation, Inc. (2014d): List of Content Management Systems. Wikipedia.org. Available online at http://en.wikipedia.org/wiki/List_of_content_management_systems, updated on 1/27/2014, checked on 12/19/2013.

Wikimedia Foundation, Inc. (2014e): Proprietäre Software. Wikipedia.org. Available online at http://de.wikipedia.org/wiki/Propriet%C3%A4re_Software, updated on 1/4/2014, checked on 1/11/2014.

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Geschäftsmodelle und Lizenzen von Open Source Anbietern im Vergleich zu proprietären Software Firmen

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Integrationsseminar“

Vorgelegt von

Janine Jakob
Anastasiia Shelukhina

am 31.01.2014

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
Kurs WWI2011I

Inhaltsverzeichnis

Abkürzungsverzeichnis

B2B	Business-to-Business
ERP	Enterprise Resource Planning
IT	Informationstechnik
OS	Open Source
OSBA	Open Source Business Alliance
OSS	Open Source Software
PSG	Professional Services Group

Abbildungsverzeichnis

1 Einleitung

Weltweit werden immer mehr Anwendungen wie das Betriebssystem Linux oder das Textverarbeitungsprogramm Open Office[1] mit Freude verwendet, da sie die Lösung zu den Anforderungen der Anwender wie beispielsweise ein lauffähiger Computer oder das Erstellen eines digitalen Dokuments darstellen. Dabei sind sich viele Nutzer nicht bewusst, dass es sich hierbei um Open Source Programme handelt. Das bedeutet, sie haben das Recht den öffentlich zugänglichen Quellcode zu verändern und weiterzugeben. Trotzdem sind die Programme qualitativ hochwertig.

Selbst bei der Diskussion um die aktuellen Spähaffären des US-Geheimdienstes NSA gilt ein Teil der Aufmerksamkeit den verschiedenen Softwaretypen. In der Technologiebranche galt der amerikanische Geheimdienst eine lange Zeit als Kämpfer für eine sichere Infrastruktur, was sich durch das Einkaufen von Sicherheitslücken in proprietärer Software auf dem Graumarkt eher wiederlegen lässt. Das Widersprüchliche ist, dass die NSA dabei gewollt diese Lücken erhalten möchte.[2] Die NSA entwickelte beispielsweise einen Code, welche als Sicherungserweiterung des Linuxkernels dient, zudem wurde ihre IT-Expertise von amerikanischen Großfirmen wie Google gerne zu Rate gezogen. Googles Android Anwendung weist Sicherheitsfunktionen auf, welche spezielle von der NSA entwickelt wurden.[3] Dabei stellt sich die Frage, welche Software nun die Rolle der bösen Software mit den lückenhaften Sicherheitsfunktionen spielt.

Um unternehmerische Entscheidungen treffen oder um eine kompetente Beratung durchführen zu können, sollte man Informationen zu den Angeboten auf dem Markt und ihren Anbieterunternehmen verfügbar haben.[4] Dieses Projekt dient genau diesem Zweck. Es stellt eine Marktstudie dar, zu den momentan in Baden-Württemberg befindlichen Anbieterfirmen von Open Source und proprietärer Software. Der Schwerpunkt dieser Thesis liegt dabei auf Open Source Produkten/Lösungen und ihren Lizenzen, welche des öfteren mit proprietärer Software verglichen werden. Die Autorinnen haben sich zudem auf kleine bis mittelständische Unternehmen spezialisiert. Außerdem soll die Unternehmensanalyse ein besseres Verständnis über die ausgewählten Geschäftsmodelle, Herausforderungen und Eigenschaften der Anbieterfirmen vermitteln. Die Erkenntnisse sollen zugleich dem zukünftigen Beratungszentrum der Medien-und Filmgesellschaft (MFG) Stuttgart dienen.

---

[1] Vgl. Bandel, G. (2010), S. 10

[2] Vgl. Lischka, K. (2013)

[3] Vgl. Lischka, K. (2013):

[4] Vgl. Fleig, J. (2009):

Zu Beginn werden essentielle Begriffe näher erläutert, um eine allgemeine Basis des Verständnisses von Open Source Software (OSS) sowie von proprietärer Software zu schaffen. Das neu erworbene Wissen wird durch die möglichen Vor- und Nachteile dieser zwei unterschiedlichen Softwaretypen und möglichen Lizenzen abgerundet, um später bei der Auswertung der Fragebögen eine Beurteilung abgeben zu können. Bevor es zu einer ausführlichen Evaluation der Antworten kommt, wird die Vorgehensweise dieser Marktstudie in Kapitel drei näher erläutert. Das vierte Kapitel beschreibt den Kern aus dem praktischen Teil der Unternehmensanalyse. Es wird auf die Repräsentativität dieser Studie eingegangen. Danach folgt eine Evaluation der Daten aus dem Fragebogen mit Fokus auf die verwendeten Geschäftsmodelle und der am meisten verwendeten Lizenzen. Hinzu kommt ein Zukunftsausblick wie sich dich Reputation von Open Source Software weiter entwickeln wird. Außerdem gibt es einen Vergleich zweier Anbieterfirmen von ERP-Systemen auf Basis von proprietärer Software und OSS. Vor dem endgültigen Fazit gibt Kapitel fünf eine kritische Reflektion des gesamten Projektverlaufs mit ein paar Verbesserungsvorschlägen für zukünftige Studien unter ähnlichen Bedingungen ab. Das Fazit unter Kapitel sechs liefert eine Zusammenfassung der relevantesten Erkenntnisse aus dieser Marktstudie, welche mit einer persönlichen Meinung der Autorinnen vervollständigt wird. Im Anhang befinden sich die ausführlichen Antworten der befragten Unternehmen, jedoch zum Teil etwas anonymisiert, um die Gedankenflüsse und Beurteilungen der Autorinnen besser nachvollziehen zu können.

2 Grundlagen

2.1 Open Source Software

Das Konzept, nach dem Computerprogramme mit ihrem Quellcode ausgeliefert werden, nennt sich Open Source. Diesen Quellcode darf jeder sehen und verändern. Open Source Software wird auch oft als „kostenlose Software“ genannt, was aber nicht immer der Wahrheit entspricht. Die Schnittmenge von sogenannter freier Software und Open Source Software ist sehr groß, aber nicht identisch. Es gibt verschiedene Kriterien, um Open Source genauer zu definieren und von anderer Software zu unterscheiden.[1]

Eine Software muss 10 verschiedene Kriterien erfüllen, um Open Source genannt werden zu dürfen. Diese Kriterien wurden von der Open Source Initiative definiert: [2]

1. Freie Weitergabe:

   Die Lizenz darf nicht verbieten, das Software-Paket, welches aus Programmen mit unterschiedlichem Ursprung besteht, zu verschenken oder zu verkaufen. Die Lizenz darf keine Lizenz- oder andere Gebühren festschreiben.

2. Der Quellcode:

   Das Programm muss den Quellcode beinhalten. Es soll zugelassen werden, den Quellcode, sowie eine kompilierte Form weiterzugeben. In dem Fall, wenn das Programm ohne Quellcode weitergegeben wird, muss es eine allgemein bekannte Möglichkeit geben, diesen Quellcode zu bekommen. Dieser Quellcode soll zum Selbstkostenpreis oder, vorzugsweise, gebührenfrei als Datei zum Herunterladen im Internet zur Verfügung stehen. Der Quellcode soll in der Form eines Programms sein, die ein Programmierer bevorzugt bearbeitet. Der Quellcode, der mit Absicht unverständlich geschrieben wurde, ist nicht zulässig.

3. Abgeleitete Software:

   Die Lizenz muss Veränderungen und eine abgeleitete Software zulassen, um sie zu denselben Lizenzbestimmungen wie die Ausgangssoftware weiterzugeben.

4. Integrität des Quellcodes des Autors:

   Die Weitergabe des Quellcodes in veränderter Form darf durch eine Lizenz nur dann eingeschränkt werden, wenn sie vorsieht, dass zusammen mit dem Quellcode Patch Files weitergegeben werden dürfen, die den Code während der Kom-

[1] Vgl. Gabler Wirtschaftslexikon (o.J.); Vgl. MacFarlane, F. (2013), S. 6

[2] Vgl. Open Source Initiative (2014); Vgl. MacFarlane, F. (2013), S. 7 ff.

pilierung verändern. Die Lizenz muss eine Weitergabe der Software erlauben, welche aus verändertem Quellcode entstanden ist. Außerdem kann sie verlangen, dass der abgeleiteten Software ein anderer Name oder eine andere Versionsnummer gegeben wird.

5. Keine Diskriminierung von Personen oder Gruppen:

   Die Lizenz darf niemanden diskriminieren.

6. Keine Einschränkung bezüglich des Einsatzfeldes:

   Die Lizenz darf niemanden verhindern, die Software in einem bestimmten Bereich einzusetzen. Sie darf beispielsweise nicht verbieten, das Programm in einem Geschäft oder in einer Genforschung einzusetzen.

7. Weitergabe der Lizenz:

   Die Rechte an einer Software müssen auf alle Personen übergehen, an die diese weitergegeben wurde, ohne dass für diese Personen die Notwendigkeit bestehen würde, eine zusätzliche Lizenz zu erwerben.

8. Die Lizenz darf nicht auf ein bestimmtes Produkt beschränkt werden:

   Die Rechte an einer Software dürfen nicht davon abhängig sein, ob diese Software Teil eines bestimmten Software-Paketes ist. In dem Fall, wenn das Programm aus dem Paket herausgenommen wird und nach Bedingungen der dazugehörigen Lizenz benutzt oder verbreitet wird, so müssen alle Personen, an die das Programm weitergegeben wurde, die gleichen Rechte haben, die auch in Verbindung mit dem ursprünglichen Software-Paket gewährt wurden.

9. Die Lizenz darf andere Software nicht einschränken:

   Die Lizenz darf keine Software einschränken, die zusammen mit lizenzierter Software verbreitet wird. Die Lizenz darf z.B. nicht verlangen, dass andere Software, die auf dem gleichen Medium verbreitet wird, auch Open Source sein muss.

10. Die Lizenz muss technologieneutral sein:

    Die Lizenz darf nicht an bestimmte Technologien oder Stile eines Interfaces gebunden sein.

Es gibt eine Reihe von typischen Vor- und Nachteilen, mit denen der Einsatz von Open Source Software verbunden ist. Die Bedeutung von diesen ist in jeder einzelnen Situation unterschiedlich und vom jeweiligen individuellen Anwendungsfall abhängig.[1]

[1] Vgl. Schimpf, S./ Kugler, A. (2007), S. 8

2.1.1 Vorteile

Zu den größten Vorteilen der Open Source Software zählen: [1]

- Wirtschaftlichkeit: Bei der Beschaffung werden Kosten eingespart.
- Anpassbarkeit: Dank dem offenen Quellcode lässt sich Software den individuellen Bedürfnissen jedes Nutzers oder Unternehmens anpassen.
- Unabhängigkeit: Der Nutzer ist vom Hersteller unabhängig und hat ihm gegenüber keine Verpflichtungen.
- Sicherheit: Durch den offenen Quellcode lassen sich verschiedene Fehler und Sicherheitslücken rechtzeitig identifizieren.

Eine Trendstudie aus dem Jahr 2009 bestätigt diese Aussage und nennt auch weitere Beweggründe für den Einsatz einer Open Source Lösung. Wie die Abbildung 1 veranschaulicht, steht für 87 Prozent der Studienteilnehmer das Motiv der „Lizenzkosten sparen“ ganz oben. Danach kommen Wünsche, wie „offene Standards“ und „Herstellerunabhängigkeit“. Die schon früher erwähnte Anpassbarkeit und Sicherheit sind für die Anwender auch sehr wichtig und stehen ganz oben in der Wunschliste. Bemerkenswert ist auch der Fakt, dass nur 18 Prozent der befragten Unternehmen den Wert auf die Bekanntheit der Softwaremarke legen.[2]

Alle Motive in Abbildung 1 lassen sich in vier verschiedene Motivgruppen unterteilen:

- Freiheit: Hier werden alle Gründe umfasst, die sich auf solche genuine Vorteile, wie die Herstellerunabhängigkeit und die Freiheit durch Open Source Lizenzen abzielen.
- Offene Quellen: Das sind alle Vorteile, die der offene Quellcode mit sich bringt.
- Qualität: Das umfasst solche Motive, wie die Zuverlässigkeit und die Sicherheit der Software.
- Pragmatismus*:* Dies bezieht sich zum Beispiel auf die Möglichkeit des vorherigen Testens und die Bekanntheit der Marke ab.

[1] Vgl. Schimpf, S./ Kugler, A. (2007), S. 8

[2] Vgl. Diedrich, O. (2009), S. 4

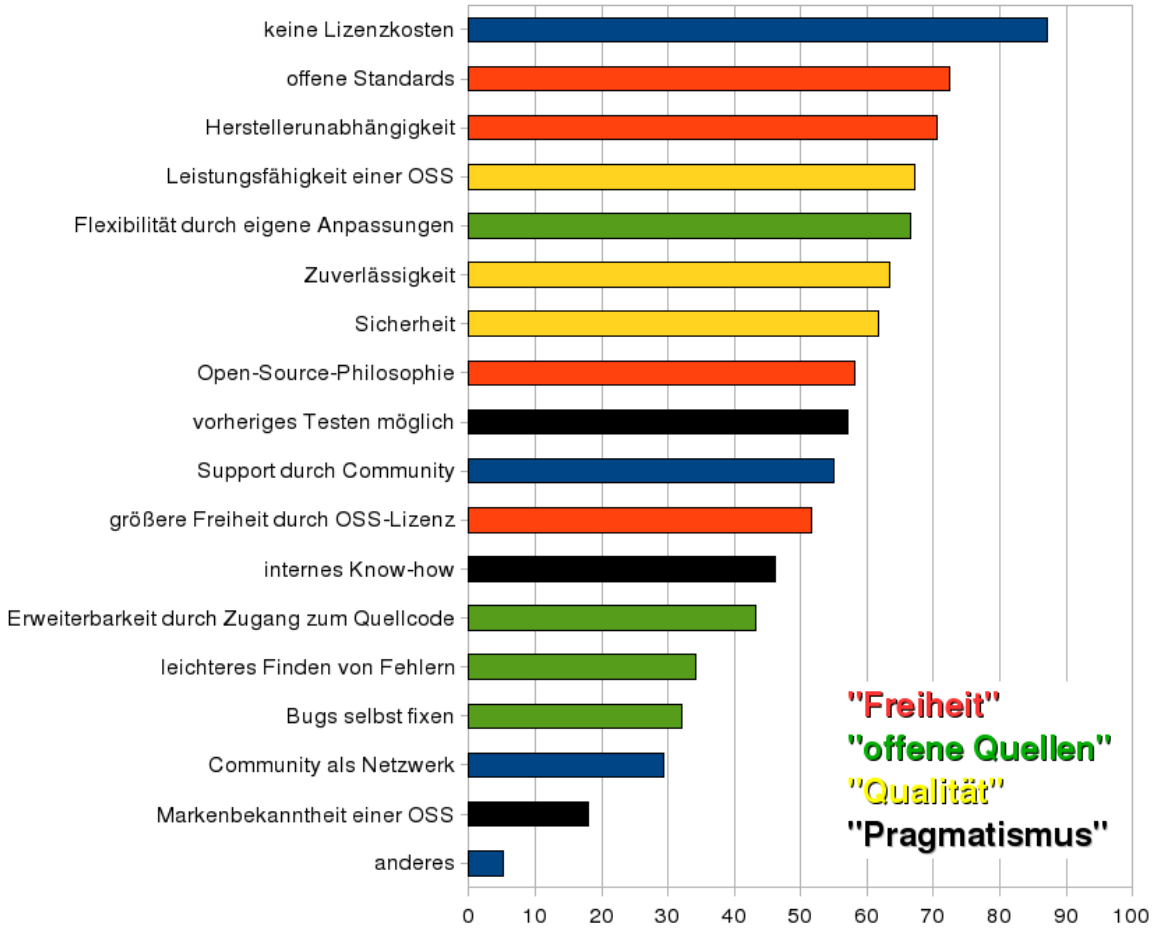


Abb. 1: Beweggründe für den Einsatz von Open Source Software[1]

2.1.2 Nachteile

Obwohl der Einsatz von Open Source Software viele Vorteile mit sich bringt, gibt es auch einige Nachteile, die beachtet werden sollten. Oft werden an der ersten Stelle die fehlende Garantie und Support genannt. Grundsätzlich können keine Garantien oder andere Ansprüche gegenüber dem Hersteller geltend gemacht werden. Die Praxis zeigt aber, dass es andere Nachteile für viele Unternehmen gibt, die mehr von Bedeutung sind.[2]

Wie es die Abbildung 2 veranschaulicht, sehen viele Unternehmen die fehlende Treiberunterstützung und uneinheitliche Bedienoberfläche als größte Nachteile der OSS. Fehlender Support wird nur von einer Minderheit als Nachteil gesehen.[3]

[1] Vgl. Diedrich, O. (2009), S. 4

[2] Vgl. Schimpf, S./ Kugler, A. (2007), S. 9; Vgl. Diedrich, O. (2009), S. 5

[3] Vgl. Diedrich, O. (2009), S. 5

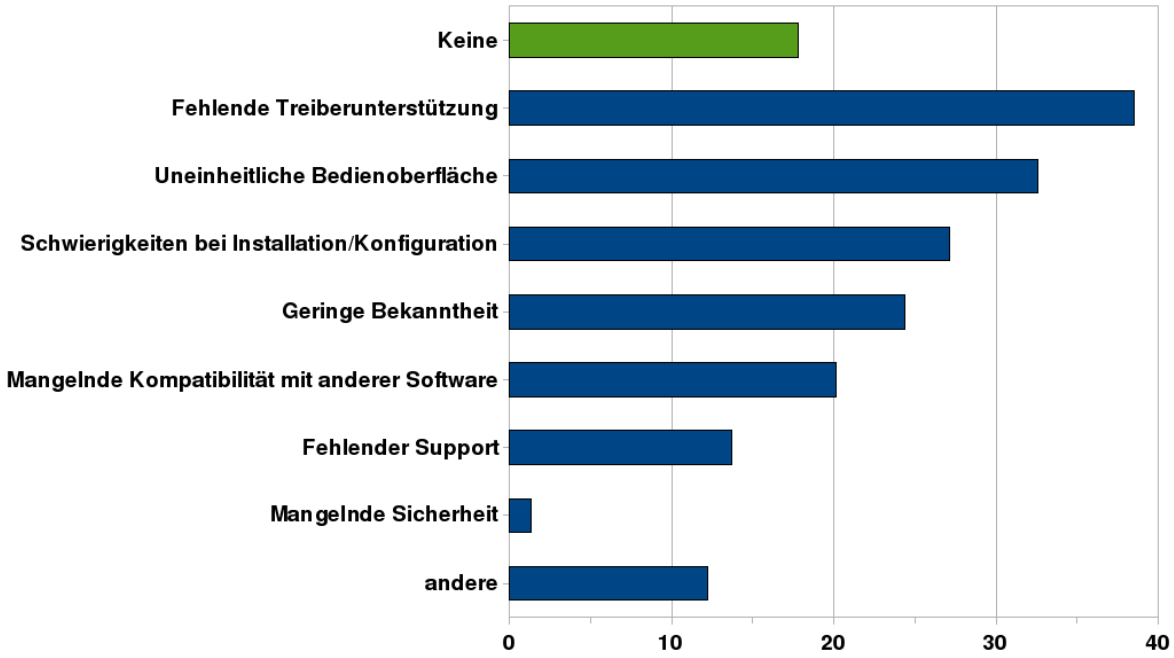


Abb. 2: Allgemeine Nachteile von Open Source[1]

Neben den in Abbildung 2 gezeigten Nachteilen, gibt es auch weitere Kritikpunkte. Ein Grund dafür ist zum Beispiel die Kosteneinsparung, warum Unternehmen sich für Open Source Software entscheiden. Wie es aber oben schon erwähnt wurde, ist Open Source nicht immer kostenlos. Und auch wenn die Lizenz für OSS kostenlos ist, können weitere Dienstleistungen wie Beratung, Implementierung, Wartung, Support und Schulungen einem teuer zu stehen kommen.[2]

2.2 Proprietäre Software

Proprietäre Software wird als Synonym für unfreie Software oder kommerzielle Software verwendet. Proprietär beschreibt den Zustand, bei dem eine Person oder eine Firma die exklusiven Copyright Rechte an einer Software hat und anderen den Zugang zum Quellcode verhindert. Die Software zu kopieren, zu verändern und zu studieren, ist ebenfalls verboten. Diese Rechte und der Zugriff auf den Quellcode können in der Regel nur gegen Geld gewährt werden.[3]

Genauso wie Open Source Software, hat auch proprietäre oder kommerzielle Software ihre Vor- und Nachteile.

[1] Vgl. Diedrich, O. (2009), S. 5

[2] Vgl. Geiger, J. (2009), S. 2

[3] Vgl. Uni-Protokolle.de (2014); Vgl. GNU Betriebssystem (2013)

2.2.1 Vorteile

Der erste Vorteil ist die deutlich höhere Zukunftssicherheit. Die Weiterentwicklung einer proprietären Software kann nicht jederzeit eingestellt werden. Dies kann auch für einen gewissen Zeitraum vertraglich abgesichert werden. Der zweite Vorteil umfasst die bessere Stabilität und Performance der Software. Kommerzielle Software ist in der Regel besser getestet worden. Selbst wenn die Anwendung Probleme bereiten sollte, hat der Nutzer den Anspruch auf Wartung und Reparatur. Als dritter Vorteil wird der oben beschriebene Support genannt. Das Supportangebot ist im kommerziellen Bereich professioneller und die Reaktionszeiten bei Problemen sind deutlich kürzer bzw. können vertraglich vereinbart werden. Die Nichterfüllung dieser Leistung kann hohe Vertragsstrafen nach sich ziehen.[1]

2.2.2 Nachteile

Als größter Nachteil der proprietären Software wird die Tatsache angesehen, dass jemand dafür bezahlen muss. Proprietäre Software ist kostenpflichtig. Darüber hinaus können auch jährliche Wartungsgebühren und Kosten für verschiedene Updates anfallen. Kommerzielle Software ist meistens auch „Closed Source Software“. Das bedeutet, dass der Nutzer keine Möglichkeit hat, in den Quellcode reinzuschauen, um z.B. Fehler oder Schwächen der Software vorab zu entdecken.[2]

2.3 Open Source Lizenzen

Genauso wie bei proprietärer Software, werden bei Open Source Software die Lizenzbedingungen festgelegt, die beschreiben, was der Nutzer mit der Software machen darf. Diese Lizenzen werden nach Copyleft-Prinzip in Copyleft und Non-Copyleft unterteilt.

Copyleft wird vom Wort Copyright abgeleitet. Dieses Prinzip stellt sicher, dass Open Source Software auch in weiterentwickelten Versionen von jedem vervielfältigt, verändert, verbreitet und öffentlich zugänglich gemacht werden kann.[3]

Die verschiedenen Typen von Copyleft sind folgende:[4]

- Strenges Copyleft: Die modifizierte Software darf wie die ursprüngliche Software nur unter der gleichen Lizenz veröffentlicht und weitergegeben werden.

[1] Vgl. Pansch, C. (2013)

[2] Vgl. Pansch, C. (2013)

[3] Vgl. Bandel, G. (2010), S. 10; Vgl. Schaaf, A. (2013), S. 17

[4] Vgl. Schaaf, A. (2013), S. 17

- Beschränktes Copyleft: Nur das Ursprungswerk verliert die Lizenz nicht. Die Modifikationen, die in einer eigenen Datei gespeichert werden, können unter einer anderen Lizenz als die restliche Software weitergegeben werden.
- Non-Copyleft: Die modifizierte Software muss nicht unter der Lizenz des Ursprungswerks stehen.

Zu den berühmtesten und wichtigsten Open Source Lizenzen zählen GNU General Public License (GPL) Version 2 und Version 3, GNU Lesser General Public License (LGPL) und Apache License.[1]

- GNU General Public License: Das ist eine Lizenz mit strengem Copyleft. Etwa 60% der Open Source Software untersteht dieser Lizenz, darunter auch so ein bekanntes Programm wie Linux. Sie ist die verbreitetste und wichtigste Open Source Lizenz.
- Lesser General Public License: Diese Lizenz ist speziell für Programmbibliotheken gedacht und hat ein beschränktes Copyleft. Sie erlaubt die Einbindung von Software nicht nur mit anderer Open Source Software, sondern auch mit proprietären Programmen.
- Apache License: Sie gehört zu den Non-Copyleft Lizenzen und wurde von der Apache Software Foundation für ihre eigene OSS entwickelt. Der Apache Webserver ist das wichtigste Programm, das dieser Lizenz untersteht.

[1] Vgl. ifrOSS (2014)

3 Vorgehensweise

Ziel dieses Projektes war es, eine Marktstudie zu den Anbietern von Open Source und proprietärer Software in Baden-Württemberg durchzuführen und die daraus folgenden Ergebnisse zu analysieren.

Um einen Überblick über die am Markt befindlichen Unternehmen zu erhalten, wurde von den Autorinnen zuerst eine Internetrecherche durchgeführt. Dabei wollte man mithilfe von einzelnen Stichworten über Google an die benötigten Unternehmensinformationen gelangen. Es ist schnell klar geworden, dass diese Suche keine effektive Methode dargestellt hat. Die Hoffnung, dass die gesuchten Ergebnisse sofort geliefert werden, wurde nicht erfüllt. Daraufhin wurden die zwei Projektberater der Autorinnen zu Rate gezogen. Der Studiengangleiter der Autorinnen und der Auftraggeber der MFG empfahlen die Internetseiten der Open Source Business Alliance (OSBA)[1] und des Software Zentrums Böblingen/Sindelfingen e.V.[2] auf die passenden Unternehmen für die Marktstudie durchzusuchen.

Auf den oben genannten Internetseiten wurden knapp 100 verschiedene Unternehmen vorgestellt. Laut der Aufgabenstellung musste die Analyse auf ungefähr 20 Unternehmen begrenzt werden, darunter ein paar wenige Firmen mit dem Angebot von kommerzieller Software. Dementsprechend musste die Auswahl durch eine bestimmte Kategorisierung von Unternehmen in zwei Schritten durchgeführt werden.

Im ersten Schritt wurde die Internetliste nach Firmen mit einem Standort in Baden-Württemberg durchsucht. Alle anderen Firmen wurden sofort ausrangiert. Die Autorinnen sind anfangs davon ausgegangen, dass alle aufgelisteten Unternehmen bei der OSBA nur Open Source Software anbieten, was oftmals nicht der Fall war. Nur im Falle, dass man nicht genügend vorrangig OS-Anbieter in Baden-Württemberg findet, wurden ein paar Firmen aus Süddeutschland als Backup notiert, was später verworfen werden konnte. Im zweiten Schritt wurden alle ausgewählten Unternehmen nach unterschiedlichen Produkt- und Lösungsangeboten differenziert. Aus den Unternehmen, die dasselbe Produkt- oder Lösungssegment anbieten, wurden anhand der Homepage und der subjektiven Meinung der Autorinnen diverse OSS Firmen ausgesucht und die Anzahl sank auf 16.

Da diese Analyse in einem relativ kurzen Zeitraum von nur zwei Personen durchgeführt werden musste, einigten sich die Projektberater auf eine Auswahl von etwa zehn OSS-Anbietern und zum Vergleich noch fünf Unternehmen mit kommerziellen Lösungen.

---

[1] Vgl. Open Source Business Alliance

[2] Vgl. Software Zentrum Böblingen/Sindelfingen e.V.

Danach sollten von Autorinnen weitere proprietäre Software Anbieter ausgesucht werden, die den Anbietern mit gleichen Lösungen aus dem Open Source Bereich gegenübergestellt werden mussten. Es wurde auch festgestellt, dass zwei der vorher ausgewählten OS Anbieter zusätzlich proprietäre Software vertreiben. Um effizient und zeitsparend zu bleiben, entschieden sich die Autorinnen, diese beiden Unternehmen auch zu ihrer Liste hinzu zu fügen. Daraus folgend mussten nur noch drei andere Anbieter von kommerzieller Software ausgewählt werden. Diese wurden über die Internetrecherche und einer Empfehlung eines unserer Experten ausgesucht. Zum Schluss ergab sich eine Liste mit insgesamt 13 diversen Unternehmen aus Baden-Württemberg.

Der nächste Schritt war die Erstellung des Fragebogens. Zuerst musste die Entscheidung getroffen werden, welche Fragen den Unternehmen gestellt werden müssen. Dafür haben die Autorinnen nochmal genauer die Aufgabe des Projektes analysiert und die ersten Fragen herausgezogen. Die nächsten ausformulierten Fragen mussten dazu dienen, repräsentative Erkenntnisse zu liefern, um weitere Vergleiche daraus ziehen zu können. Als letztes wurde nach einem Feedback und weiteren Wünsche des Auftraggebers der MFG gefragt und es wurde eine finale Version des Fragebogens erstellt. Die erstellte Umfrage enthält 30 Fragen, die innerhalb von ca. 15 Minuten beantwortet werden können. Alle Fragen wurden in drei verschiedene Kategorien geteilt: Unternehmen allgemein, Unternehmenstätigkeit und Kennzahlen.

Nach der Erstellung des Fragebogens konnte mit der Umfrage gleich angefangen werden. Drei Kontakte stellte der Auftraggeber den Autorinnen zur Verfügung, um das Marktstudie und die erste Kontaktaufnahme mit den Unternehmen zu beschleunigen. Aufgrund der Vorweihnachtszeit und Feiertage waren viele Unternehmen sehr schwer zu erreichen, deshalb wurden die meisten Kontaktaufnahmen erst im Januar weitergeführt. Nach mehreren Telefonaten, verfassten E-Mails und später auch Erinnerungsschreiben wurde festgestellt, dass der Rücklauf insgesamt sechs Antworten betrug. Eine Erinnerungsemail oder kurzes Telefonat hat die Quote nicht weiter erhöht.

Um noch an mehr repräsentative Informationen zu gelangen, wurde entschieden eine weitere Internetrecherche durchzuführen. Als alle Informationen gesammelt wurden, wurde die Endanalyse durchgeführt. Dabei wurden verschiedene Erkenntnisse über die Unternehmen herausgezogen und manche Verbesserungsvorschläge gemacht, die in den nachfolgenden Kapiteln näher erläutert werden. Vor der Veröffentlichung dieser Thesis gab es als letzten Schritt eine weitere Kontaktaufnahme mit den Teilnehmern der Befragung, um sicher zu gehen, dass die hier verwendeten Angaben ihrem Wissen entsprechenden und es keine falschen Interpretationen von seitens der Autorinnen gibt. Falls doch, konnten sie ein Veto einlegen und es korrigieren.

Die Abbildung 3 veranschaulicht die einzelnen Schritte der Vorgehensweise:

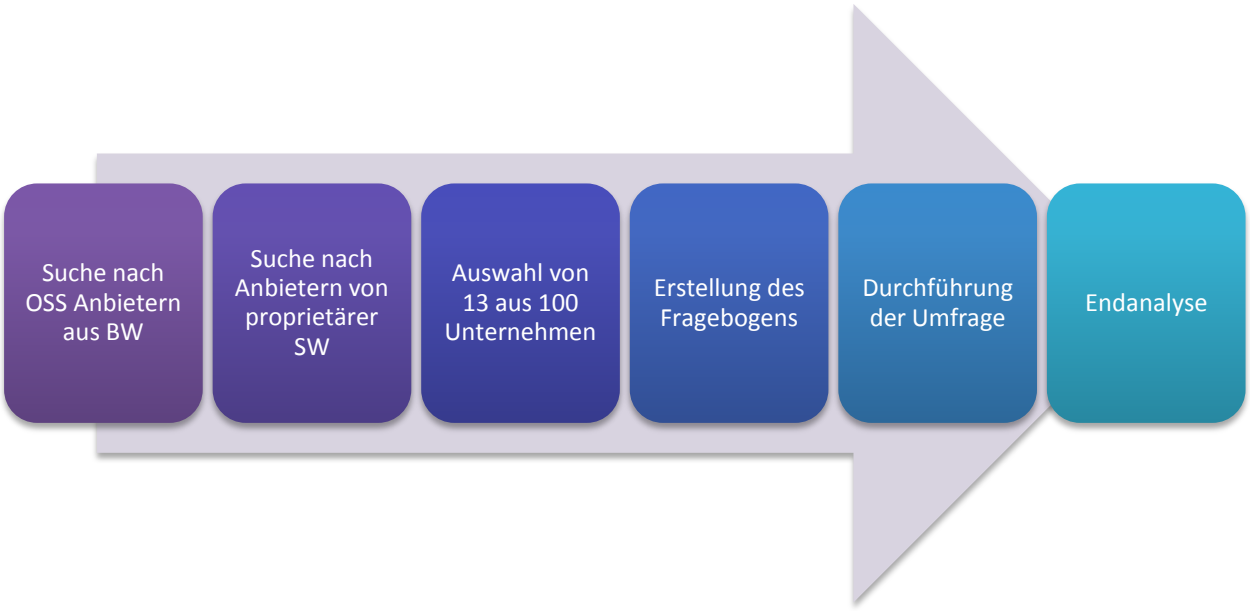


Abb. 3: Vorgehensweise

4 Evaluation der gewonnenen Daten

Bevor die Daten evaluiert werden und voreilige Schlüsse und Verbesserungsvorschläge aus den Antworten der Teilnehmer gezogen werden können, sollte man sich überlegen, ob man die Studie als repräsentativ bezeichnen kann.

4.1 Repräsentativität der Marktstudie

Da es keine einheitliche Formel zur eindeutigen Bestimmung von Repräsentativität gibt, müssen eigene Kriterien zur Prüfung gefunden werden.[1] Jedoch kann man allgemein sagen, dass eine Studie dann als besonders repräsentativ angesehen wird, wenn sie so nah wie möglich die Meinung der Gesamtheit bzw. die Realität abbildet.[2]

Anbei werden einige Faktoren definiert, die hier als Entscheidungshilfe dienen sollen, ob die durchgeführte Marktstudie als ausschlaggebend angesehen werden kann und um somit das weitere Vorgehen mit den Daten bestimmen zu können:

- Anzahl der Befragten und die Rücklaufquote: Je größer die Anzahl, umso mehr verschiedene oder sich wiederholende Antworten erhält man, um diese besser beurteilen zu können. Eine weitere Frage, die sich hier stellt ist, wie viele Personen unter ihnen gaben auswertbare Antworten ab.
- Auswahl der teilnehmenden Unternehmen und der befragten Ansprechpartner: Zum Beispiel durch eine Stichprobe, Vollerhebung oder eine gezielte Auswahl nach bestimmten Kriterien, wie letzteres zumindest bei dieser Marktstudie stattfand.
- Richtigkeit der eingetragenen Daten: In der Marktforschung darf die Fehlerquote der Ergebnisse nicht 5 Prozent übersteigen, um bei den Marktforschern als repräsentativ angesehen zu werden.
- Instrument, womit die Daten gewonnen wurden: Beispielsweise durch eine Onlineumfrage, telefonisch, per E-Mail, aus Statistiken usw. Denn jedes Instrument bringt verschiedene Vorteile und Risiken mit sich.

Diese Kriterien sind nur einige von vielen möglichen Faktoren, die bei der Definition helfen können. Tiefer kann auf dieses Thema hier nicht weiter eingegangen werden, da dafür eine eigene ausführliche wissenschaftliche Arbeit benötigt werden würde.

Wie bereits erwähnt, wurde ein standardisierter Fragebogen für telefonische Interviews und zur Befragung per E-Mail mit den Anbieterunternehmen von proprietärer genauso wie

[1] Vgl. Fleig, J. (2009)

[2] Vgl. Fleig, J. (2009)

von Open Source Software verwendet. Von 13 auserlesenen Unternehmen gab es einen Rücklauf von sechs davon.

Obwohl die Autoren sich selbst mit dem Thema ausführlich beschäftigt haben und sie in der Lage sind, den Großteil der Antworten der Befragten auf richtig und falsch zu prüfen, kann man nicht von einem hundertprozentigen Sicherheitsgrad ausgehen. Daher ist es schwierig zu berechnen, wie hoch die Fehlerquote bei der Unternehmensanalyse liegt, ob auf alle verschiedenen Märkten der ausgewählten Produkte und Lösungen in Baden-Württemberg dieselbe Meinung, dieselben Herausforderungen, Geschäftsmodelle etc. geteilt werden. Die Autoren haben bei ihrer Umfrage einige Tendenzen für Erkenntnisse herausgehört. Trotzdem wird der Leser dazu gezwungen die nun nachfolgenden Ergebnisse weiterhin kritisch zu hinterfragen.

4.2 Die ausgewählten Unternehmen im Überblick

Zunächst sollen die auserwählten Unternehmen mit einigen Eckdaten näher vorgestellt werden, damit der Leser sich einen Überblick über die Kandidaten verschaffen kann, bevor die konkreten Antworten ausgewertet werden. Um eine gute Übersicht und flüssige Leserlichkeit der Thesis zu gewährleisten, befinden sich die konkreten Antworten der Teilnehmer tabellarisch im Anhang. Darin enthalten gibt es weitere wichtige Antworten, die auch für das zukünftige Open Source Beratungszentrum der Medien- und Filmgesellschaft Stuttgart sehr interessant sein werden. Aufgrund der zeitlichen Beschränkung kann nicht auf jede Frage und Antwort im Detail eingegangen werden. Noch wichtig zu erwähnen ist, dass sich zwei der sechs Unternehmen für eine Anonymisierung entschieden haben, weshalb sie deshalb in dieser Thesis als „Unternehmen X aus Baden-Württemberg“ und „Unternehmen Y aus Baden-Württemberg“ bezeichnet werden.

Die Eckdaten der Firmen wurden mündlich oder schriftlich von den Ansprechpartnern geäußert. Sie sind zudem aber auch problemlos auf den Webseiten der Software Anbieter auffindbar.

Folgende Unternehmen in Baden-Württemberg haben an dieser Marktstudie teilgenommen:

1) Comundus GmbH, Waiblingen
2) DASEQ GmbH, Stuttgart-Fellbach
3) GRAU DATA AG, Schwäbisch Gmünd
4) Polarion Software GmbH, Stuttgart
5) Unternehmen X aus Baden-Württemberg *(ausgedachter Name zur Gewährleistung der Anonymität)*

6) Unternehmen Y aus Baden-Württemberg *(ausgedachter Name zur Gewährleistung der Anonymität)*

- Alle 6 Unternehmen sind lokal in Baden-Württemberg vertreten.
- 5 von 6 Firmen, welche auch Open Source Software anbieten, wurden zwischen 1998 und 2005 gegründet. Das Unternehmen mit der proprietären Software gab es schon viel früher nämlich seit 1980.
- 4 von den 6 Unternehmen haben zwischen 13 und 30 Mitarbeiter. Dabei hat nur eins von ihnen ein paar Standorte im Ausland. Polarion Software GmbH hat hierbei als Ausnahme bereits 90 Mitarbeiter, darunter an mehreren internationalen Standorten. Das anonyme Unternehmen X besitzt bereits 850 Mitarbeiter (unter anderem bei Vertriebspartnern) und ist in 28 anderen Ländern vertreten.
- Die Palette von Produkten und Lösungen ist sehr unterschiedlich und nicht identisch mit einem der anderen gewählten Unternehmen. Nur ERP-Systeme werden zweimal präsentiert. Jedoch handelt es sich hierbei einmal um ein System als kommerzielle Software des anonymen Unternehmens X und das anonyme Unternehmen Y setzt auf eine Eigenherstellung als Open Source Software.
- Weitere proprietäre Software und OSS Produkte und Lösungen der vorgestellten Firmen beinhalten Web Content Management-, Portal-, Archivsysteme und weiteres.
- Aufgrund der diversen Angebote werden auch sehr unterschiedliche Branchen und (inter-)nationale Märkte bedient, um einen einheitlichen Schluss daraus zu ziehen.
- Alle 6 Unternehmen haben gemeinsam, dass sie den Wettbewerb am Markt mit als die größte Herausforderung im Vertrieb ansehen.

➔ Bereits anhand der ersten Vergleiche sieht man, in wie vielen Faktoren sich das Unternehmen X mit den rein proprietären Lösungen von den anderen Unternehmen abhebt, welche sich entweder komplett oder nur teilweise auf Open Source spezialisieren. Dies verdeutlicht, dass die Anforderungen an das Unternehmen X in Baden-Württemberg anders zu sein scheint als für die aufgeführten Unternehmen.

Teil des Fragebogens waren die Angabe von Kennzahlen der Unternehmen wie Eigenkapital, ROI, EBIT usw. Jedoch hat jedes Unternehmen nur ein paar wenige Zahlen angegeben. Der Hintergrund für die Frage nach den Kennzahlen war, dass die Autorinnen sich einen besseren Ergebnisvergleich unter den Unternehmen erhofft haben, um bewerten zu können, ob ein gewisses Geschäftsmodell besonders oder weniger erfolgreich ist. Dieser Punkt muss übersprungen werden, um Falschaussagen zu vermeiden.

4.3 Zwei Anbieter von ERP-Systemen im Vergleich

Zwei von sechs befragten Unternehmen bieten beide Enterprise Resource Planning Systeme an. Zufälligerweise waren dies die einzigen zwei Unternehmen, deren Angaben anonymisiert behandelt werden sollten. Das muss aber nicht dementsprechend auf einen bestimmten Grund aus dem ERP-System-Umfeld schließen. Anbei werden einige interessante Gemeinsamkeiten und Unterschiede zusammengefasst. Doch zunächst folgt die Definition:

ERP-Systeme entsprechen einer Software, das in einem Unternehmen (fast) alle Geschäftsprozesse informationstechnisch unterstützt, jedoch aus betriebswirtschaftlicher Sicht.[1]

Unterscheiden tun sich die Aussagen der Ansprechpartner der zwei Firmen enorm, wie die anschließende Abbildung zeigt. Gemeinsamkeiten scheint es auf den ersten Blick kaum welche zu geben. Das Mitarbeiterwachstum in den Firmen stieg bei Beiden um circa 30 Prozent innerhalb der letzten drei Jahre an, welches für einen Wachstumsmarkt oder eine steigende Nachfrage nach ERP-Systemen mit sich führt. Das Unternehmenswachstum entsprach bei Beiden 10 Prozent. Anhand der fehlenden betrieblichen Kennzahlen und der Repräsentation von nur zwei Unternehmen, weiß man nicht, welcher Typ von ERP-Systemen stärker nachgefragt wird, da es unternehmensabhängig sein kann.

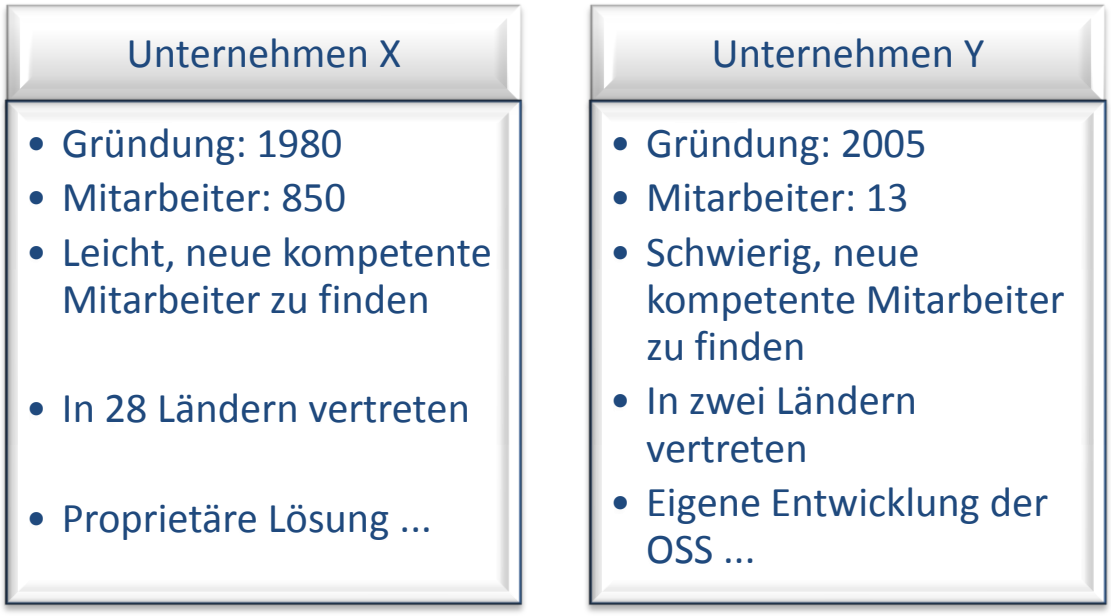


Abb. 4: Unterschiede zwischen den ERP-System-Anbietern Unternehmen X und Unternehmen Y

[1] Vgl. MacFarlane, M. (2013), S. 15

Wie in Kapitel 4.1 erwähnt, ist es in diesem Fall schwierig eine hinreichende Analyse mit wahrheitsgemäßen Aussagen und Verbesserungsvorschlägen zu verfassen. Beide Firmen haben sich für unterschiedliche Lizenzen und genauso Geschäftsmodelle entschieden und während der Eine den lokalen Markt hauptsächlich bedient, versorgt der andere große Teile im Ausland. Diese Kriterien benötigen unterschiedliche Strategien, die hier nicht weiter ausgelegt werden können.

4.4 Geschäftsmodelle

Aufgrund der oft kostenlosen Verfügbarkeit von OS Anwendungen kann das Einführen von dem Modell, indem man Lizenzkosten erhebt, nicht immer als sinnvoll angesehen werden, vor allem in einem Markt mit großer Konkurrenz.[1] Deshalb wurden für OSS ein paar neue Modelle entwickelt, die folglich vorgestellt werden.

4.4.1 Verschiedene Geschäftsmodelle

1. Service-Modell: Wenn man sich verschiedene Unternehmen anschaut, wie beispielsweise Hewlett-Packard, erkennt man, dass das große Geld heutzutage nicht mehr durch den Verkauf von Produkten erzielt wird, sondern, um es bildlich zu beschreiben: es handelt sich um das Sahnehäubchen auf dem Speiseeis, dass die Lösung erst vollkommen macht. Durch verschiedene Konzepte werden Schulungen, Beratungen oder auch Wartungen beim Kunden durchgeführt.
2. Support-Modell: Den Aspekt des Supports kann man auch als ein eigenständiges Geschäftsmodell betrachten. Wobei viele Unternehmen eine Kombination aus mehreren Geschäftsmodellen für sich zusammen clustern.[2] Beispielsweise können hier externe Mediatoren eingesetzt werden, die als Vermittler zwischen den Entwicklern und dem OSS Kunden dient.
3. Subskription; Dieses Modell beruht darauf, ergänzende Dienstleistungen zusätzlich zum Open Source Produkt/Lösung zu verkaufen. Dabei soll eine langfristige Kundenbindung aufgebaut werden. Zudem soll ein Wechsel zu einem Angebot der Konkurrenz erschwert werden. Gegen ein pauschales Entgelt, das monatlich oder jährlich bezahlt werden kann, werden regelmäßige Updates durchgeführt oder Sicherheitslücken werden verschlossen. Das Schließen von Sicherheitslücken ist hier im Vergleich zu der NSA Spähaffäre auf jeden Fall gewollt und auch empfehlenswert. Weitere Funktionalitäten können mit einem ergänzenden Serviceangebot

[1] Vgl. Keßler, S. (2012), S.18

[2] Vgl. Keßler, S. (2012), S.18

in Anspruch genommen werden. Red Hat und Novell Suse sehen sich hier als Serviceanbieter an.[1]

4. Duallizensierung: Der Erwerb einer weiteren Lizenz, um damit zum Beispiel eine weniger strikte Benutzung eines Programmes möglich zu machen.[2]
5. Softwareintegration: Dieses Modell ist speziell angelegt für Unternehmen wie Distributoren, die einen weiten Kundenstamm ansprechen möchten. Red Hat Enterprise Linux lässt sich als gutes Beispiel für so einen Distributoren vorzeigen.

Die Liste mit den unterschiedlichsten Geschäftsmodellen könnte noch weiter ergänzt werden, was irgendwann den Rahmen sprengen würden. Weitere Modelle lauten Hardwareintegration, Spenden, Wertsteigerung und Verteilung von SW-Produkten.

4.4.2 Das beliebteste Geschäftsmodell

Anhand der dreizehn auserlesenen Unternehmen, bei denen die Unternehmensanalyse geplant war, konnte man durch ihre Antworten oder durch die Internetrecherche von ihrer Homepage ein favorisiertes Geschäftsmodell entdecken. Die folgende Angabe ist eine Beobachtung der Autorinnen, welche keineswegs eine Ausgabe über alle am Markt befindlichen Anbieterfirmen von OSS und proprietärer Software in Baden-Württemberg darstellen soll. Daher muss diese Angabe vom Leser noch einmal kritisch hinterfragt werden, wie es ausführlich im Kapitel 4.1 „Repräsentativität der Marktstudie“ bereits näher ausgeführt wurde.

Die Äußerung der Teilnehmer ergibt die Tendenz, dass das favorisierte Geschäftsmodell dem Service-Modell entspricht. Das Support-Modell wird hier einfach zum Service-Modell zugehörig mit angegeben, da der Support auch ein Teil des Services Modells darstellt. Dabei scheint es keinen Unterschied zu geben, ob ein Unternehmen nur proprietäre Software wie das Unternehmen X aus Baden-Württemberg anbietet, nur Open Source Lösungen wie die Comundus GmbH oder eine Mischung aus Beidem wie die anderen vier befragten Softwareanbieter. Daraus ergibt sich die Frage, weshalb eine Fokussierung auf das Konzept von Beratung, Schulung und/oder Support so beliebt zu sein scheint. Das globale IT-Unternehmen Hewlett-Packard GmbH ist ein weiteres von vielen Beispielen, die mittlerweile mehr Potential im Dienstleistungssektor sehen, als im Vergleich zu früher nur der Vertrieb von Hardware einen hohen Umsatz eingebracht hat. Die Fokussierung auf die Hardware statt der Software war für lange Zeit geschichtlich gesehen ein Grund,

[1] Vgl. Mundhenke, J. (2007), S.133

[2] Vgl. Keßler, S. (2012), S.19

warum sich Open Source auch erst so spät durchgesetzt hat und z.B. in den 1980-er Jahren noch nicht so beliebt war.

Nach dem Service-Modell besonders beliebt scheinen Subskription und die Wertsteigerung und Verteilung von Software-Paketen zu sein, mit einem Ranking von zwei aus sechs präsentierten Unternehmen aus der Marktstudie. Für eine bessere, erweiterte Wertschöpfung für den Kunden wird das Service-Modell oder auch andere Geschäftsmodelle gerne auch mit weiteren Ansätzen kombiniert.

Zudem wurde speziell von der DASEQ GmbH betont, dass man genau überlegen sollte, bei der Entscheidungsfindung über das Geschäftsmodell, den Kundenkreis und bei der richtigen Lizenzwahl, ob sich beispielsweise eine Eigenherstellung oder ein Fremdbezug von Software lohnt. Die DASEQ GmbH zählt zu einem von vielen Firmen, die sich gegen eine eigene Softwareentwicklung entschieden haben und sich darauf spezialisiert haben ihre Software fremd zu beziehen. 2 von 4 der befragten Unternehmen mit Open Source Lösungen haben diese nicht selbst hergestellt.

4.5 Die am häufigsten verwendeten Lizenzen

Eine der Aufgaben dieses Projektes war es, zu analysieren, welche Lizenztypen am häufigsten von Open Source Anbietern in Baden-Württemberg genutzt werden. Die Ergebnisse, die in dieser Arbeit beschrieben werden, sind nicht nur aus den Umfrageergebnissen entstanden, sondern wurden auch durch Internetrecherche und Nachforschung erarbeitet.

So wurde festgestellt, dass die beliebteste Lizenz, die bereits beschriebene GNU General Public License ist. Sie zählt auch zu den bekanntesten Open Source Lizenzen

Unter anderem werden von mehreren Unternehmen die GNU Affero General Public License (AGPL) und bereits vordefinierte Apache Lizenzen verwendet.

GNU Affero General Public License ist eine Lizenz, die zu großen Teilen der GPLv3 entspricht, nur diese um einen wichtigen Punkt erweitert. AGPL enthält zusätzliche Regeln und Definitionen für Software, die über Netzwerk verwendet wird. So kann jeder mithilfe dieser Lizenz den Quellcode von Webanwendungen nutzen und verändern.[1] Die AGPL wird beispielsweise von Unternehmen Y angewendet, da es die beste vorgehenswiese ist Webanwendungen zu veröffentlichen. Das Unternehmen sieht darüber hinaus Vorteile darin, dass die Community verpflichtet ist, alle Erweiterungen und Änderungen an der Software zur Verfügung zu stellen. (vgl. Anhang 1)

[1] Vgl. GFDF (2014)

4.6 Die zukünftige Rolle von Open Source Anwendungen

Schon 2007 erkannten Saugatuck Technology, welche eine Marktforschungs- und Beratungsfirma aus den USA ist und zur Jahrtausendwende von ehemaligen Gartner-Analysten gegründet wurden, die ansteigende Benutzung und Bedeutung von Open Source Anwendungen in verschiedenen Märkten weltweit.

In ihrer Studie zu "Open Source: The Next Disruptive IT Influence" beobachteten sie, dass die zunehmende Verwendung quelloffener Software das Verhalten, Planen und Denken der Anbieterfirmen auch von proprietärer Software verändern wird. Durch die internationale Befragung der 200 Business- und IT-Verantwortlichen und aus Einzelinterviews leiten die Analysten ab, dass bereits 2007 über 80 Prozent der Firmen Open Source Software einsetzen. Immer öfter befinden sich auch Komponenten der OSS Anwendung in bereits hergebrachten Systemen wieder. Laut einer Unternehmensbefragung im Frühjahr 2005 in Baden-Württemberg über O und IT-Sicherheit ergab sich, dass momentan in Deutschland nur 25 Prozent der Unternehmen und öffentlichen Einrichtungen OSS als eine besonders wichtige Komponente der IT-Infrastruktur dargestellt wird.[1]

Dass die Prognose und Annahmen bezüglich des Zukunftsmarktes von OSS von Saugatuck Technology nicht weit hergeholt waren, unterstützen beispielsweise die Angaben der sechs befragten Unternehmen. Wie man an ihren Antworten sehen kann, fand in jedem der sechs Unternehmen in den letzten drei Jahren ein Wachstum statt, unter anderen ein Mitarbeiterwachstum von zehn bis 200 Prozent statt. Zusätzlich rechnen Sie auch weiterhin mit einer steigenden Nachfrage nach ihren Open Source Produkten und Lösungen und hoffen darauf, dass Sie den kommenden Anforderungen mit ihrer Unternehmensaufstellung gerecht werden können.

Brands wie Red Hat haben sich auf quelloffene Software spezialisiert und werden von den hier befragten Unternehmen auch gerne weiter verbreitet, wie z.B. von der DASEQ GmbH. Die globalen Großunternehmen wie HP, IBM oder Microsoft investieren ebenfalls mehr Geld in die Entwicklung von OSS, jedoch gelten sich auf dem Markt der quelloffenen Software noch nicht als die dominierenden Teilnehmer.[2]

Die Meinung der Experten aus den gewählten Unternehmen schätzen Open Source Anwendungen mit noch weiter steigendem Potential ein, das noch nicht erreicht ist. Comundus Waiblingen GmbH sieht der Zukunft positiv entgegen und berichtet von einem Wachstumsmarkt. Die GRAU DATA AG plant die Eröffnung einer weiteren Niederlassung im Ausland und ist 2014 Brasilien vorgesehen. Zudem befinden sie eine hohe strategi-

[1] Vgl. Döbler, T (2005), S.77

[2] Vgl. Herrmann, W. (2007)

sche Ausrichtung der Unternehmen als besonders wichtig. Das anonymisiert Unternehmen Y aus Baden-Württemberg sieht eine stetige Weiterentwicklung der OSS als entscheidend an, nur dadurch kann das Vertrauen in die Lösungen gewonnen werden. Die Polarion Software GmbH aus Stuttgart erkennt bei einem Rückblick in die Vergangenheit, dass Open Source einen hohen Stellenwert mittlerweile eingenommen hat, jedoch eher in dem Bereich zur Nutzung der Software für eine Weiterentwicklung anstatt einer reinen Nutzung des Endanwenders. Das Unternehmen vermutet, dass das daran liegt, da die benutzerfreundliche Komponente des Kunden noch entwicklungsbedürftig ist, da der Kunde andere Ansprüche an die Nutzung bringt. Momentan sind in dem Bereich eher technische Personen und Entwickler am Werk, die sich mit der Materie auskennen.

Eine Änderung der angebotenen Lizenzen oder des Geschäftsmodells sei im Moment nicht vorgesehen, da man mit seiner Unternehmensaufstellung zufrieden ist und die befragten Unternehmen denken, dass sie für die kommenden Jahre weiterhin gut aufgestellt sind, um den Anforderungen am Markt gerecht zu werden. Nur die Software Polarion COUD sieht mehrere Variationen der Lizenzen. Die Herausforderungen im Vertrieb bleiben bestehen, vor allem der hohe Wettbewerbsdruck, ob bei ERP-Systemen oder im Portalbereich.

Auch in Zukunft erhoffen sich die Open Source Nutzer weiterhin Kostenvorteile und eine geringere Abhängigkeit von den Herstellern.[1] Der Preis bleibt ein starkes Kriterium bei der Kaufentscheidung.

[1] Vgl. Herrmann, W. (2007)

5 Reflexion des Projektverlaufs und Tipps zur Verbesserung

Nach Meinung der Autorinnen gab es bereits am Anfang einige kritische Faktoren, die von vornherein eine reibungslose Durchführung der Marktstudie als schwieriger gestalteten. Zudem ergaben sich währenddessen weitere Kriterien, von welchen angenommen wird, dass mit einem Ausbau von Fähigkeiten diese Kriterien in Zukunft besser gemeistert werden können.

Kritische Faktoren, welche zu Beginn gegeben waren:

1. Ein ambitionierter Zeitrahmen: Die Verfügbarkeit der Studenten zur Durchführung dieser Marktstudie wurde auf ungefähr einem Monat vor bis einem Monat nach dem Jahreswechsel 2013 zu 2014 gelegt. Dabei lag der Großteil der von der Hochschule eingeplanten Zeit nach den Weihnachtsfeiertagen und Silvester. Zudem lagen dieses Jahr die gesetzlichen Weihnachts-/Neujahrsfeiertage 2013 besonders günstig für den Arbeitnehmer. Dahingehend gab es für die Majorität der Arbeitnehmer weniger Werktage, die wir für eine Befragung benötigt hätten und um eine längere Frist zur Beantwortung geben zu können.

2. Prioritäten der Arbeitnehmer: Aus dem ambitionierten Zeitfenster und dem Arbeitnehmer freundlichen Feiertagen, leitet sich ab, dass sich Arbeit auf ihren Schreibtischen gestapelt haben und die vermutlich priorisiert werden, da die Arbeitnehmer sonst ihre eigenen Fristen/Abhängigkeiten nicht einhalten können. Durch die Abhängigkeit der Kooperation mit den ausgewählten Unternehmen wurden die für das Projekt erwarteten ausgefüllten Umfragebögen erst später als erwartet zugesendet.

   Tipp: Zu Beginn des Projektes sollte man sich einen ausführlichen Zeitplan, mit Einberechnung von Risikofaktoren und einem Puffer zum Beispiel mithilfe von Microsoft Project oder zumindest Microsoft Excel als Strukturierungshilfe zu eigen machen.

Kritische Faktoren, die sich während des Projektes ergaben:

Obwohl bei dem Telefonat mit den auserwählten Firmen nichts verkauft werden sollte und nur ein paar schnelle Informationen benötigt wurden, kann man die Gespräche mit einer Telefonakquise vergleichen, denn dieselben Herausforderungen galten bei diesen telefonischen Befragung, z.B.

1. Fehlendes Interesse: an einer Teilnahme, z.B. da kein Mehrwert zu sehen war.

2. Verweigerung von Bekanntgabe von unternehmensinternen Zahlen: Manche Informationen, vor allem die Kennzahlen von Eigen- und Fremdkapital, ROI, EBIT usw. wollten die Ansprechpartner der Öffentlichkeit nicht Preis geben. Jedoch hatten sich die Autorinnen diese Zahlen für ihre Bewertung der Profitabilität der Unternehmen gewünscht gehabt.

   Tipp: Über den Bundesanzeiger hat man teilweise Zugriff auf die öffentlichen Kennzahlen der Unternehmen wie beispielsweise das Eigenkapital. Jedoch bleibt der Großteil der aussagekräftigeren Zahlen wie EBIT im Verborgenen. In Zukunft muss man sich andere Kriterien zum Vergleich aussuchen, wobei Zahlen schlüssiger zu vergleichen sind, anstatt nur Aussagen, welchen einen großen Interpretationsspielraum mit sich führen.

3. Zeitdefizit: Ein zeitliches Defizit bestand während des spontanen Anrufs bei den Firmen, da vorab kein Termin für das Gespräch ausgemacht wurde.

4. Wissen über Lizenzen: Besonders erschreckend war die Tatsache, dass die Kontaktpersonen wenig detailliertes Wissen über die in dem Unternehmen verwendeten Lizenzen mit sich brachten.
   Tipp: Nach dem Grundsatz „Vertrauen ist gut, Kontrolle ist besser“, sollten die Antworten kritischer hinterfragt und geprüft werden bzw. sollten bei Unklarheiten mehr Fragen während des Interviews oder nachträglich per E-Mail gesendet werden, um Missverständnisse sofort auszuschließen.
5. Kein fester Ansprechpartner: Durch das fehlende Wissen über den Namen und die direkte Kontaktmöglichkeit des richtigen Ansprechpartners im Unternehmen, musste man sich manchmal telefonisch über mehrere Personen hinweg kämpfen, um eine Kontaktmöglichkeit mit der richtigen Person zu erhalten. Trotzdem blieb das manchmal auch komplett aus.

6. Verschätzte Bearbeitungszeit des Fragebogens (Motivationsfaktor): Den Ansprechpartnern selbst teilten die Autorinnen eine benötigte Bearbeitungszeit des Fragbogens – egal ob telefonisch oder schriftlich – von ca. 20 min mit. 20 min gelten als eine demotivierende Zahl. Zudem hat die Erfahrung gezeigt, dass 10 bis 15 min völlig ausreichend gewesen wären.

   Tipp: Bei der Bearbeitungszeit sollte man eine realisierbare, doch eher kleine Zahl verwenden, um die Kontaktpersonen zur Teilnahme zu motivieren.

7. Mangelhafte Kommunikations- bzw. Motivationsfähigkeiten der Interviewer: Diese Fähigkeiten gelten als besonders essentielle Kriterien, die eine definitiv höhere Erfolgsquote bei richtiger Anwendung mit sich führen.

   a. Wie bei der Kaltakquise sollte der Anrufer in der Lage sein dem Befragten sofort einen wichtigen Nutzen für das eigene Unternehmen mithilfe der Befragung aufzuzeigen. Zum Beispiel: „Die von Ihnen erhaltenen Daten dienen zur Beratung zukünftiger Unternehmen, die auf der Suche nach einer Lösung sind. Ihre Software könnte die Antwort zu dem Problem sein. Dadurch können Sie neue Geschäfte und Kunden ohne zusätzlichen Aufwand dank uns generieren."

   b. Zudem gibt es Tipps und Tricks, um die Telefonzentralen der ausgewählten Unternehmen geschickt und schnell zu umgehen, anstatt sich gleich abwimmeln zu lassen. Bei 11 von 15 Unternehmen blieb eine Weiterleitung des Anrufs zu der entsprechenden, unbekannten Kontaktperson aus. Zudem hat man sich als Anrufer zu oft damit zufrieden gegeben, den Fragebogen elektronisch an das Sekretariat zu versenden, anstatt auf einen telefonischen Kontakt zu verharren. Zuletzt blieb nur die Hoffnung auf Ausfüllen des Fragebogens auch ohne direkten Ansprechpartner, welche vergeblich war.

      Tipp: Jeder Verkäufer hat seine eigene Taktik bei der Telefonakquise, die sich mit zunehmender Übung als weniger erfolgreich erweist oder ihn bestätigt. Beispielsweise empfiehlt Tim Taxis eine Kombination aus seiner entwickelten „Ehklar-Technik" und einer Verwendung von Schlüsselwörtern wie „strategisch, politisch, entscheiden, Sondierung etc."[1] Es handelt sich dabei um eine gebündelte, kurze Erklärung zur Frage „Worum es denn geht?" und zugleich die Verwendung der überzeugenden Signalwörter. Mehr Taktiken von unterschiedlichen Ratgebern können aufgrund der Bündelung dieser Arbeit nicht erwähnt werden, da es ein breites Thema für eine weitere wissenschaftliche Arbeit darstellt. Jedoch hat die eigene Erfahrung der Autorinnen in einem anderen Kontext gezeigt, dass solche Ratgeber einige brauchbare Tipps aufweisen.

[1] Vgl. Taxis, T. (2012), S. 38

6 Fazit

Ziel der Befragung war es eine Marktstudie mit ausgewählten Unternehmen durchzuführen. Die Firmen sollten proprietäre und/oder Open Source Software mit einer Unternehmensniederlassung in Baden-Württemberg aufweisen. Durch eine Internetrecherche und Befragungen von ausgewählten Unternehmen, mithilfe eines selbsterstellten Fragebogens, kamen die Autoren zu der Schlussfolgerung, dass seit einigen Jahren eine zunehmende Nachfrage nach Open Source Software besteht. Von den dreizehn befragten Unternehmen und den sechs Unternehmen, die tatsächlich an der Umfrage teilgenommen haben, hat auch das einzige rein kommerzielle Software anbietende Unternehmen in dieser Studie einen zehnprozentigen Wachstum bzw. Mitarbeiterwachstum von dreißig Prozent innerhalb von drei Jahren.

Die befragten Unternehmen, aus dem vorrangig kleinen und mittelständischen Sektor mit teilweise internationalen Vertriebsstandorten oder Partnern sehen der Zukunft von Open Source Software sehr positiv entgegen. Die meisten der ausgewählten Unternehmen folgen dem Service-Geschäftsmodell und legen ihren Schwerpunkt auf Beratung, Schulung und/oder Support. Zudem wird dieses Modell auch gerne mit anderen kombiniert, wie beispielsweise mit einer Wertsteigerung oder Subskription.

Die Tendenz der beliebtesten Lizenzen war bei den sechs teilnehmenden Unternehmen nicht ganz so eindeutig wie bei den Geschäftsmodellen. Doch zusammen mit einer Internetrecherche über die anderen verbleibenden sieben Firmen und ihre Lizenztypen, konnten die GPL besonders oft identifiziert werden. Zudem scheinen auch Apache und AGPL oft verwendet zu werden.

Diese Unternehmen bieten ihre Produkte, Lösungen und Dienstleistungen im Bereich unter anderem von ERP-, Archiv-, Portal- und Content Management Systemen an. Die zwei Firmen Unternehmen X und Unternehmen Y bieten beide ERP-Systeme einmal als Open Source und als proprietäre Software an. Bis auf ihr Unternehmenswachstum hatten die Beiden wenig miteinander gemein und differenzieren sich stark. Aufgrund der moderaten Ergebnisse und einer streitwürdigen Repräsentativität dieser Studie, erlaubten die Autorinnen sich Verbesserungsvorschläge oder Hypothesen wegzulassen, um keine falschen Ergebnisse zu verbreiten.

Genauso wie die Ansprechpartner der befragten Unternehmen sehen die Autorinnen einen weiteren Wachstum und eine Zunahme an Bedeutung von Open Source bei den verschiedensten Branchen und (inter-)nationalen Märkten. Ein Vergleich der Präsenz von Open Source in ihrem eigenen Leben mit einem Blick in die Vergangenheit und danach in die Gegenwart zeigt, dass OSS allgegenwärtig ist. Zudem benutzt man mehr Open Sour-

ce Software, als dass man das als Laie manchmal weiß. Im privaten, zum Beispiel mit Open Office sowie im beruflichen Umfeld, z.B. mit Linux haben sie bisher gute Erfahrungen gemacht. Außerdem sind ihnen bis jetzt keine Sicherheitslücken aufgefallen, wie sie beispielsweise von dem amerikanischen Geheimdienst NSA gerne ausgenutzt werden, was weltweit einen Skandal ausgelöst hat. Auch daran sieht man wieder die Omnipräsenz von dem Thema der Open Source Software an.

Anhang

Anhangverzeichnis

Anhang 1: Ergebnisse der Unternehmensbefragungen

Ergebnisse der Unternehmensbefragungen							
1	Name des Unternehmens und Hauptstandort	Comundus Waiblingen GmbH	Polarion Software GmbH Stuttgart	DASEQ GmbH, Stuttgart-Fellbach	GRAU DATA AG, Schwäbisch Gmünd	Unternehmen X aus Baden-Württemberg *(ausgedachter Name zur Gewährleistung der Anonymität)*	Unternehmen Y aus Baden-Württemberg *(ausgedachter Name zur Gewährleistung der Anonymität)*
2	Gründungsjahr	2001	2004	2005	1998	1980	2005
3	Anzahl der Mitarbeiter	30 (alle vier Unternehmensgruppen zusammen insgesamt 100)	90	20	30	850 u.a. bei Vertriebspartnern	13
4	Wie hoch ist der Mitarbeiterzuwachs in den letzten 3 Jahren?	10 % (2 Personen)	200 %	Von 5 auf 20 Personen	50 %	30 %	4 Personen
5	Fällt es Ihrer Firma leicht neue kompetente Mitarbei-	Nein, da kleines Unternehmen, geringe Gehaltsfrage und flache	Kommt auf die Position an: im technischen Bereich eher schwer, da die Aus-	Nicht schwer, da gute Kontakte in der Community.	Ja, aufgrund der hohen Flexibilität: Homeoffice-Möglichkeit und flexib-	Ja, da gute Zusammenarbeit mit Universität, Unternehmen auf verschiedenen Plattformen	Nein, die Ausbildungen sind schlecht, bzw. werden die guten Mitarbeiter oft von

	ter anzuwerben? Warum?	Hierarchie, auch wenn interessante Projekte vorhanden sind.	wahl an Firmen im Raum Stuttgart sehr groß ist.		le Arbeitszeiten	vertreten, Straßenbahn Werbung etc.	großen Firmen abgesaugt.
6	Weitere Niederlassungen (evtl. im Ausland)	keine	Polarion Standorte: USA, Tschechien, Schweiz Partner: u.a. in Frankreich, Spanien, Finnland, China, Brasilien usw. Eine Übersicht dazu: http://www.polarion.com/company/partners/index.php	keine	USA, Frankreich, Brasilien (2014)	Weltweit u.a. über Partner in 28 Ländern vertreten & eigene Standorte im Ausland	Stuttgart als Vertriebstochter
Unternehmenstätigkeit							
7	Auf welchen Märkten agieren Sie?	Mittelstand ab 500 Mitarbeiter (zumindest das Ziel), branchenunabhängig, NGOs, öffentliche Einrichtungen (Kirchen,	Weltweite Märkte im Business-to-Business in den Branchen: Software, Engineering, Electronics, Medical Devices, Automotive, Aerospace, Defense, Financial	Wesentlich in den deutschsprachigen Ländern: Deutschland, Österreich, Schweiz. RedHat Consultants im Ausland auch weltweit	Verstärkt europäischer Markt; B2B	Unternehmenssoftware ERP-Systeme in Zentraleuropa, China, USA, Indien	DACH

		Diakonien)	Services, Telecommunication, Transportation & Logistics u.v.m.				
8	Angebotene Lösungen/ Produkte (Open Source und/oder proprietäre Software)	Open Source Lösungen: Portalsysteme (Liferay) und Web Content Management (OpenCms), Enterprise Search (Solr), Webshop (KonaKart)	Application Lifecycle Management Software Dazu gehören: • Open Source: SVN Importer, Webclients Und die kostenpflichtigen Produkte: • Polarion ALM • Polarion REQUIREMENTS • Polarion QA • Polarion CLOUD	Beides, Security sehr viel proprietär. Red Hat Enterprise Linux, Univention für Mittelstand	Archivierungsprodukte: ArchiveManager, FileLock, Fileserver Archiver, OpenArchive, XtreemStore / cloud Produkt: DataSpace	ERP-Systeme als proprietäre Software	ERP System *„Unternehmensname“* (CE, EE) Logistik Club, Touchkasse
9	Welche Lösungen/ Produkte davon sind Open Source?	alle	SVN Importer, Webclients	Red Hat Enterprise Linux, Univention für Mittelstand	OpenArchive	keine	Eigenes ERP System
10	Falls Sie im Open Source	2001	2004	Seit 2005. Von Beginn im Bereich	2010	--	2010

	Bereich tätig sind, seit wann?			Security, 2011 neue Mitarbeiter			
11	Wie schätzen Sie die Entwicklung/ Rolle von Open Source Software in Zukunft ein?	Positiv, ein Wachstumsmarkt. Ob man richtig aufgestellt ist mit seinen Produkten glaubt man, weiß man aber nie	Hat gegenwärtig hohen Stellenwert eingenommen, allerdings mehr im Bereich Nutzung für die Weiterentwicklung, weniger die reine Nutzung als Endanwender. Hier fehlt oftmals die Kunden/Usability orientierte Komponente, da hier Entwickler/ technisch orientierte Personen unterwegs sind, die andere Ansprüche an die Nutzung haben.	Glauben an Erfolg von OpenSource, große Zukunft, wird sehr erfolgreich. Große Kunden haben Open Source Strategie.	Hohe strategische Ausrichtung für Unternehmen, Rolle wird zunehmend wichtiger	--	Wird immer wichtiger werden, da nur dadurch das Vertrauen in die Lösungen wieder gewonnen werden kann
12	Angebotene Lizenztypen	Keine Angaben*	Named und Concurrent/Floating	Subscription, weil keine eigene Software. Nur	Bei Archivierung: Lizenzierung nach Speichervolumen, bei	Normale Lizenzmodelle, gestaffeltes Concurrent User Model	Community Edition in der AGPL, verschiedene Ausprägungen

				Dienstleistungen	Cloud: nach Userzahlen		der Enterprise Edition
13	Warum haben Sie sich für diese Lizenztypen entschieden?	--	Um den Kunden eine individuell anpassbare Toollandschaft zu ermöglichen.	Nur Modell	--	--	AGPL, weil es die vernünftigste Variante ist und die Community verpflichtet wird, die Erweiterungen „zurückzugeben“/ zur Verfügung zu stellen.
14	Zukunftsaussicht: Lizenzänderungen im Unternehmen	--	Polarion CLOUD in mehreren Variationen erhältlich	Werden nicht verändert	Keine Änderungen	Jetzige bleiben	Derzeit keine Änderungen geplant
15	Mitgliedschaft in Gruppen (z.B. Entwicklungscommunities, Open Source Business Alliance usw.)	Keine offiziellen Mitgliedschaften, außer opencms.org	Siehe unsere Partner: http://www.polarion.com/company/partners/index.php	OSBA	OSBA, Active Archive Alliance, Eco Cloud System usw.	Zusammenarbeit mit und Engagement an der Universität	OSBA, OSEG (Open Source Experts Group)
16	Geschäftsmodell (Wie werden Einnah-	Durch Customizing, Weiterentwicklung der	Software Lieferant (Einnahmen durch Lizenzverkauf, Main-	Durch Service-Modell, Subscription auch	Service Modell	Lizenzen verkaufen, Implementierungsprojekte (Dienstleistungs-	Subscription, Consulting, Enterprise Edition

	men erzielt?)	OSS, dazu Beratung, Konzeption und Entwicklung der Lösung	tenance und Services)	dazu. Support kaufen, nicht das Produkt		geschäft), Wartungsgeschäft, Upgrade Verträge	
17	Warum haben Sie sich für dieses Geschäftsmodell entschieden?	IT Dienstleister machen es alle so	Es ist ein sehr skalierfähiges Modell, da sich Software beliebig oft, ohne weitere Kosten kopieren lässt.	Einfach Kunden zu bekommen, sehr transparent. Wegen den Mitarbeitern, die sehr gut sich in dem Bereich auskennen, deshalb Consulting und Schulungsgeschäft. Sie sind dafür ausgebildet. Wollen keine Software selber entwickeln. Schulung oder Unterstützung durch Software-Architekten. Software Design.	Bindung an das Unternehmen	--	Es ist die fairste Variante

				Kernstärke: das Know-how der Mitarbeiter!			
18	Was hebt Ihr Lösungs-/ Produktangebot vom Mitbewerb mit gleichen Lösungen ab?	Hohe Flexibilität und Schnelligkeit, da im Vertrieb schnell agieren kann. Ziel: innerhalb von 24 Stunden soll der Kunde eine Antwort erhalten.	Wir unterstützen eine Lösung, bei der Sie alle Phasen des Application Lifecycle (Requirements Management, Testmanagement, Taskmanagement, Changemanagement ...) in einer einzigen Lösung verwalten können. Polarion ist ein TÜV Nord zertifiziertes Tool für ISO 26262/IEC 61508, 100% webbasiert und unterstützt jeden Prozess und jede Methode (Agile,	Open Source Lösung die leicht mit anderen Open Source Lösungen verbunden werden kann. Man bindet sich nicht an Unternehmen an, sorgen für Integration. (Unternehmen hat sich für 2 Monitoring Lösungen entschieden, aus 20 Guten). Möglichst viele Bausteine aus Standard Komponenten kombiniert. Sie binden die Kun-	Kein Wettbewerb	Sehr international aufgestellt, lokalisiert für verschiedene Länder, rechtliche und marktty-pische Anforderungen angeboten, anpassungsfähiges System, Upgrade fähig	*Produkt „Unterneh-mensname*“ ist das technologisch modernste ERP System und im Open Source Umfeld auf jeden Fall auch das Umfassendste.

			Wasserfall, regulatorischen, hybrid) Volle Traceability und Versionskontrolle. Details unter folgendem Video: http://www.polarion.com/company/video/value.php	den an sich nicht an! Kunde hat die Wahl. Hat nicht den „Login“.			
19	Herausforderun-gen im Vertrieb	Quantitativ und qualitativ überall starker Wettbewerb (im Portalbereich, bei Internetauftritten und Business Apps). Bei OpenCms sind besser aufgestellt, da bereits sehr lange in dem Business dabei.	Sichtbarkeit am Markt	Marketingphasen. Noch nicht ganz bekannt, da noch sehr junges Unternehmen.	Keine Free Ware, Service Vertrag in Verbindung mit Produkt	Wettbewerbssituation (ca. 300 andere Anbieter neben den Großen wie SAP)	Viele KMUs sehen die absolute Notwendigkeit eines umfassenden ERP-Systems nicht ein. Im Open Source Bereich kommt es dazu, OSS = kostenlos (dass so auf keinen Fall stimmt), OSS wird auch oft mit Nicht-Hersteller betreut, also nur Chaos gleichgesetzt. Hier versuchen

							wir auf die Hersteller-unterstützung hinzuarbeiten.
20	Mögliche Abhängigkeiten in Ihrem Business	Konjuktur, sonst unabhängig	Nein, außer dem Klassiker, der Investitionsbereitschaft unserer Kunden/Interessenten	Mieten Räume, aber eigentlich kein Stress und keine Probleme damit	keine	keine	Partner und auch Präsentationsräumlichkeiten sind beide wichtig. Wobei die Partner eine umfassende Unternehmenskompetenz mitbringen müssen, was (siehe Mitarbeiter) entsprechend selten / schwierig zu finden ist.
Kennzahlen							
21	Eigenkapital	0,00	--	--	92.161,86[1]	EK Quote: 65% (2012)	--
22	Fremdkapital	--	--	--	--	Kein FK, unabhängig von Banken und Investoren, eine Mitarbeiter AG, nicht börsennotiert	0,-
23	Umsatz	--	--	--	--	--	720.000,-
24	Gewinn	--	--	--	--	--	+- 0,-
25	Investitionen	In Mitarbeiter, in	--	--	--	--	In die Weiterentwick-

[1] Vgl. Bundesanzeiger (2014)

		Marketing und Knowledge, da ein Diestlesitungsunternehmen und daher keine Investitionen in Produkte. Interne Akademie mit kontinuierlichen Schulungen zur Weiterbildung der Mitarbeiter, ROI.					lung ca. 100.000 bis 150.000€ jährlich
26	Wachstum in %	2 oder 3 Prozent, kleine Tendenz nach oben (Unternehmen ist konservativ aufgestellt)	--	--	--	10 %	10%
27	ROI	--	--	--	--	--	--
28	EBIT	--	--	--	--	--	--

* Offizielle Antwort am Telefon: keine Lizenzen, da es sich um Open Source handelt.

Quellenverzeichnisse

Literaturverzeichnis

Bandel, G. (2010): Open Source Software: Fördert oder hemmt diese Art der Softwareentwicklung den Wettbewerb?, Hamburg: Diplomica

Bertschek, I./ Döbler, T. (2005): Open Source Software und IT-Sicherheit: Unternehmensbefragung Fruhjahr 2005 in Baden-Württemberg“, o.O.: o. Verl.

Henkel, J. (2007): Offene Innovationsprozesse, Die kommerzielle Entwicklung von Open-Source-Software, 1. Aufl., Wiesbaden: GWV

Keßler, S. (2013): Anpassung von Open-Source-Software in Anwenderunternehmen, Wiesbaden: Springer

MacFarlane, F. (2013): Open Source Enterprice-Ressource-Planning (ERP) Software, Evaluation, Installation und Test – Eine Machbarkeitsstudie, Hamburg: Diplomica

Mundhenke, J. (2007): Wettbewerbswirkungen von Open-Source-Software und offenen Standards auf Softwaremärkten, Berlin/Heidelberg: Springer

Saleck, T. (2005): Chefsache Open Source, Kostenvorteile und Unabhängigkeit durch Open Source, 1. Aufl., Wiesbaden: Vieweg & Sohn

Schaaf, A. (2013): Open-Source-Lizenzen, Untersuchung der GPL, LGPL, BSD und Artistic License, Hamburg: Diplomica

Schimpf, S./ Kugler, A. (2007): Webbasierte Open Source-Kollaborationsplattforme, Eine Studie der Fraunhofer Gesellschaft, Stuttgart: Fraunhofer IRB

Taxis, T. (2012): Heiß auf Kaltakquise in 45 Minuten, München: o.Verl.

Verzeichnis der Internetquellen

ARD.de (2013): PRISM, „Tempora“ und viele Wanzen, http://www.tagesschau.de/ausland/chronologie-prism-tempora100.html, Abruf: 29.01.2014

Bundesanzeiger (2014): Wegen Anonymität der Daten, kann die genaue Seite nicht angegeben werden. Abruf: 26.01.2014

Diedrich, O. (2009): Trendstudie Open Source, http://www.heise.de/open/artikel/Trendstudie-Open-Source-221696.html, Abruf: 27.01.2014

Fleig, J. (2009): Umfragen planen und durchführen, http://www.business-wissen.de/artikel/weitsicht-umfragen-planen-und-durchfuehren/, Abruf: 28.01.2014

Gabler Wirtschaftslexikon (o.J.): Open Source, http://wirtschaftslexikon.gabler.de/Definition/open-source.html, Abruf: 26.01.2014

Geiger, J. (2009): Offen, sparsam, professionell, http://business.chip.de/artikel/Open-Source-im-Unternehmen_39764376.html, Abruf: 27.01.2014

GFDF (2014): Die AGPL V3 Lizenz, http://gfdf.de/agpl-1.php, Abruf: 30.01.2014

GNU Betriebssystem (2013): Kategorien freier und unfreier Software, http://www.gnu.org/philosophy/categories.de.html, Abruf: 26.01.2014

Herrmann, W. (2007): Die Zukunft von Open Source, http://www.computerwoche.de/a/die-zukunft-von-open-source,1849336, Abruf: 27.01.2014

ifrOSS (2014): Welches sind die wichtigsten Open Source Lizenzen und welchem Lizenztyp gehören sie an?, http://www.ifross.org/welches-sind-wichtigsten-open-source-lizenzen-und-welchem-lizenztyp-gehoeren-sie, Abruf: 28.01.2014

Lischka, K. (2013): Cyber-Angriffe: So gefährdet die NSA die Internetsicherheit, http://www.spiegel.de/netzwelt/netzpolitik/cyber-angriffe-so-gefaehrdet-die-nsa-die-internet-sicherheit-a-919759.html, Abruf: 29.01.2014

Open Source Initiative (2014): The Open Source Definition, http://opensource.org/osd, Abruf: 26.01.2014

Pansch, C. (2013): Open Source und kommerzielle Lösungen gegenübergestellt, http://www.christian-pansch.de/mein-wissen/rund-um-typo3-und-content-management-systeme/open-source-und-kommerzielle-loesungen-gegenuebergestellt/, Abruf: 27.01.2014

Toprak, M. (2008): Sicherheitsrisiko: Open Source im Unternehmen, http://www.netzwelt.de/news/78207-sicherheitsrisiko-open-source-unternehmen.html, Abruf: 28.01.2014

Uni-Protokolle.de (2014): Proprietäre Software, http://www.uni-protokolle.de/Lexikon/Propriet%E4re_Software.html, Abruf: 26.01.2014

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

In-Memory Datenbanken – wichtige Merkmale und deren praxisnahe Umsetzung

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt“

Vorgelegt von

Alexander Croll,
Tobias Dietz,
Philipp Koch,
Tim Soetebeer

am 31.01.2014

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
WWI2011I

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACID	Atomicity-Consistency-Isolation-Durability
CEP	Complex Event Processing
CPU	Central Processing Unit
DB	Datenbank
DBMS	Datenbank-Management-System
DBS	Datenbank-System
DHBW	Duale Hochschule Baden-Württemberg
ECC	Error Checking and Correction
ECP	Enterprise Caché Protocol
ERP	Enterprise Resource Planning
ETL	Extract-Transform-Load
HANA	High Performance Analytic Appliance
IBM	International Business Machines
IMDB	In-Memory Datenbanken
IMDBS	In-Memory Datenbanksystem
IMDBMS	In-Memory Datenbankmanagementsystem
MDX	Multidimensional Expressions
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
RAID	Redundant Array of Independent Disks
RAM	Random-Access Memory
RPO	Recovery Period Objective
RTO	Recovery Time Objective
SSD	Solid State Drive
USB	Universal Serial Bus

Abbildungsverzeichnis

Abstract
(A. Croll, T. Dietz, P. Koch, T. Soetebeer)

Die Begriffe Big Data und Business Analytics sind heutzutage in aller Munde. Die Datenbestände wachsen überall unablässig – als Folge suchen Unternehmen zunehmend nach Möglichkeiten, die existierenden Daten zu nutzen und aus ihnen einen Wettbewerbsvorteil zu generieren. In diesem Zusammenhang gewinnen In-Memory Lösungen als Alternative zur traditionellen Datenspeicherung mit Hilfe von relationalen Datenbanken immer mehr an Bedeutung. Besonders SAP HANA ist hier als führendes System zu nennen, doch auch andere große IT-Unternehmen wie IBM und Oracle lassen nicht lange auf entsprechende Konkurrenzprodukte warten. Diese Thematik hat den Anlass der nachfolgenden Ausarbeitung gegeben. Ziel war es vordergründig, dem verantwortlichen Fachpersonal in Form dieser Untersuchung eine Entscheidungshilfe an die Hand zu geben, durch die die potenziellen Einsatzmöglichkeiten der In-Memory Datenbanktechnologie für das eigene Unternehmen bewertet werden können. Hinsichtlich der zentralen Merkmale wie der Spaltenorientierung, der Datenkompression und der Parallelverarbeitung wurde im Laufe der Untersuchung erläutert, wie SAP HANA diese Eigenschaften umsetzt. Mit Hilfe praxisnaher Rechenbeispiele wurde zudem aufgezeigt, wie sich der Einsatz dieser Merkmale auf die Performance und die Zugriffszeiten des Systems auswirkt. Darüber hinaus wurde deutlich, dass innerhalb von SAP HANA bereits diverse Maßnahmen existieren, durch die ein möglicher Datenverlust aufgrund der Speicherung im Hauptspeicher im Falle eines Systemausfalls verhindert werden kann.

Abschließend zeigte sich vor allem die Notwendigkeit, bei der Bewertung der Einsatzmöglichkeiten von In-Memory Datenbanken besonders die entsprechenden Anwendungsfälle und –szenarien im jeweiligen Geschäftsumfeld zu berücksichtigen. Je nach Einsatzbereich spielt diese Datenbanktechnologie ihre Vorteile daher nicht immer voll aus. Geht es aus Sicht der geschäftlichen Anforderungen aber beispielsweise darum, komplexe analytische Abfragen wie bei OLAP durchzuführen, stellt In-Memory eine zukunftsweisende Technologie dar.

Die wissenschaftliche Untersuchung wurde im Rahmen der Lehrveranstaltung „Projekt“ von Studenten im Alter von 20-23 Jahren erstellt, die gemeinsam im 5. Semester das Studienfach „International Management for Business and Information Technology“ an der DHBW Stuttgart studieren. Dies sind die vier Teammitglieder:


Alexander Croll

Tobias Dietz

Philipp Koch

Tim Soetebeer

1 Einleitung
(A. Croll, T. Dietz, P. Koch, T. Soetebeer)

"With In-Memory Databases we can do things we couldn't do in the last 25 years" – Hasso Plattner, CEO und Mitgründer SAP.

Die kontinuierlich wachsenden Datenmengen im internen und externen Unternehmensumfeld stellen nicht nur eine Herausforderung für Unternehmen dar - sie bieten auch das Potential, Big Data als Wettbewerbsvorteil zu nutzen. Dies bedarf einer effizienten Datenspeicherung sowie eines schnellen Datenzugriffs. Mit der Entwicklung von In-Memory Datenbanktechniken geht in diesem Zusammenhang, bezogen auf das einleitende Zitat, ein alternativer Ansatz zur Datenverarbeitung in traditionellen Datenbanksystemen einher.

Das Thema In-Memory wird vor allem seit dem Aufkommen von SAP HANA in Kombination mit sich verändernden Anforderungen an Geschäftsanwendungen für das aktuelle IT-Geschäftsumfeld immer präsenter. Unternehmen stellen sich dabei zunehmend die Frage nach den Eigenschaften von In-Memory Datenbanken sowie deren Möglichkeiten für das jeweilige Geschäftsumfeld.

Auf Basis dieser Problemstellung ergibt sich die Zielsetzung der folgenden Ausarbeitung. Diese soll durch einen Überblick über die technischen Merkmale von In-Memory Datenbanken und eine Abgrenzung zu traditionellen Datenbanksystemen dazu dienen, eine Bewertung für mögliche Anwendungsszenarien vornehmen zu können.

Die erläuterte Zielsetzung spiegelt sich im Aufbau dieser Ausarbeitung folgendermaßen wieder: Einleitend sollen die Merkmale von traditionellen Datenbanksystemen aufgezeigt werden, um anschließend auf die Umsetzung dieser Merkmale bei In-Memory Datenbanken einzugehen und die Unterschiede zu traditionellen Systemen herauszuarbeiten. Der eigene Wertbeitrag der Autoren besteht neben der Auswahl der relevanten Merkmale für In-Memory Datenbanken und deren Strukturierung vor allem in der anschließenden praxisnahen Betrachtung der konkreten Umsetzung dieser Merkmale anhand einer ausgewählten Beispieldatenbank. Darüber hinaus wird durch die Autoren ein Überblick über einige marktrelevante Produkte gegeben. Abschließend ist eine Bewertung durch eine Reflexion der Arbeitsergebnisse vorgesehen. Diese bezieht sich zudem auf mögliche Alternativen zu In-Memory Datenbanken und gibt einen Ausblick auf zukünftig zu erwartende Entwicklungen.

2 Einführung In-Memory-Datenbanken (A. Croll, T. Dietz, P. Koch, T. Soetebeer)

Im folgenden Kapitel werden zunächst die Grundlagen von Datenbanken allgemein erläutert, bevor zu In-Memory-Datenbanken übergeleitet wird. Hierbei wird sowohl eine kurze Einführung in IMDBs, als auch eine detaillierte Beschreibung der einzelnen Grundlagen und Merkmale gegeben.

2.1 Grundsätze von Datenbanken

2.1.1 Datenbankhistorie

Das Bedürfnis, Wissen geordnet festzuhalten, existiert bereits seit Jahrhunderten und war schon vor langer Zeit in Form von beispielsweise Bibliotheken ausgeprägt.

Die erste halbwegs elektronische Datenverarbeitung ermöglichten dann zum Ende des 19. Jahrhunderts erstmals die Lochkarten[1], welche vom International Business Machines (IBM) Vorgänger, „Tabulating Machine Company“ und seinem Gründer Herman Hollerith auf den Markt gebracht wurden. Sie ermöglichten eine schnellere Datenerfassung z.B. bei Volkszählungen.[2]

Den Durchbruch zur tatsächlichen Speicherung von Datensätzen, welche später auch wieder in Computer eingelesen werden konnten, lieferte das 1898 erfundene und in den 1930er Jahren massentauglich gemachte Magnetband.[3]

Die Erfindung der Festplatte im Jahr 1956[4] machte es nun möglich, nicht mehr ausschließlich sequentiell, sondern sogar direkt auf gespeicherte Datensätze zuzugreifen.[5] Diese Erleichterung sollte später maßgeblichen Anteil an der Entwicklung von modernen Datenbankverfahren, wie wir sie heute noch nutzen, haben.

Im Laufe der Zeit haben sich daraufhin eine Reihe von verschiedenen Datenmodellen-/banken herausgebildet, welche im folgenden Abschnitt kurz zum Verständnis erläutert werden.

[1] Vgl. Kreitz, C. (1995), S. 5
[2] Vgl. Süddeutsche Zeitung (2011)
[3] Vgl. Kreitz, C. (1995), S. 5
[4] Vgl. Tagesanzeiger (2011)
[5] Vgl. Sieprath, S. (2005), S. 2

2.1.2 Bestehende Datenbankmodelle

Hierarchisches Datenbankmodell

Das hierarchische Datenmodell ist das älteste aller Datenmodelle und existiert bereits seit den 1960er Jahren.[6] Zu dieser Zeit wurde es entwickelt, um die gespeicherten Dateien besser zu strukturieren. Auch hier war IBM von entscheidender Rolle, denn das Modell kam erstmalig in Information Management Systemen (Datenbanken) auf Mainframes vor.[7]

Heutzutage findet man dieses Modell noch bei Banken und Versicherungen, welche noch nicht auf modernere Systeme umgestellt haben.[8]

Der Aufbau basiert, wie der Name vermuten lässt, auf einer Hierarchie. Diese Hierarchie kann auch als Baumstruktur beschrieben werden. Die Baumstruktur geht von einem Wurzelelement aus und hat dann verschiedene in Verbindung stehende Tochterelemente. Jedes Tochterelement kann wiederum selbst weitere Tochterelemente besitzen, sodass sich die Baumstruktur schnell nach unten hin ausbreiten kann.

Die Beziehungen, welche mit diesem System dargestellt werden können, umfassen 1:1 bzw. 1:n, denn die Abhängigkeit von zwei verbundenen Elementen muss unmittelbar bestehen.

Dies lässt auch sofort die Schwäche des hierarchischen Datenbankmodells erkennen: es sind keine (bzw. nur sehr kompliziert) n:m-Beziehungen modellierbar. Um n:m-Beziehungen herzustellen, müssen Daten oft redundant gehalten werden. Deshalb kommt dieses Modell bei der heutigen Datenverarbeitung nur noch vereinzelt zum Einsatz.[9]

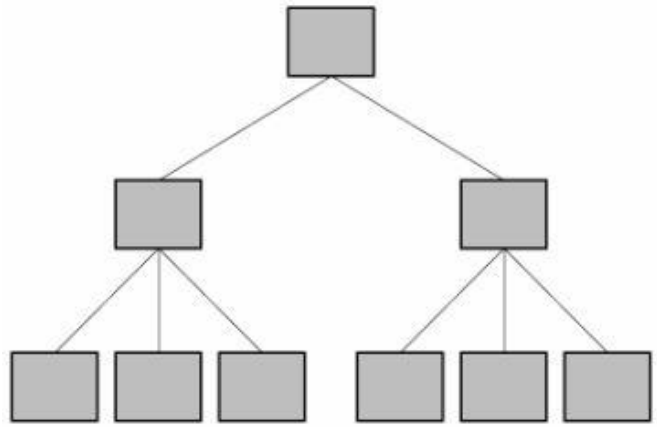


Abbildung 1: Hierarchisches Datenbankmodell[10]

[6] Vgl. Löhl, J. u.a. (2005), S. 3 ff.
[7] Vgl. Puff, M. (2008), S. 5
[8] Vgl. Löhl, J. u.a. (2005), S. 3 ff.
[9] Vgl. Puff, M. (2008), S. 5
[10] Enthalten in: Puff, M. (2008), S. 5

Netzwerkdatenbankmodelle

Das Netzwerkdatenbankmodell ging aus dem hierarchischen Datenbankmodell hervor und wurde entwickelt, um dessen Schwachstellen zu beheben. Ein Element konnte nun in verschiedene Richtungen mit anderen verbunden sein, was die redundante Datenhaltung in verschiedenen Zweigen des hierarchischen Datenbankmodells eliminierte. Allerdings war die Navigation durch das Netzwerkmodell, wie auch schon bei seinem Vorgänger, noch immer recht kompliziert, sodass der Anwender sich gut mit der Datenbank auskennen musste, um dessen Inhalte ansteuern und verarbeiten zu können.[11]

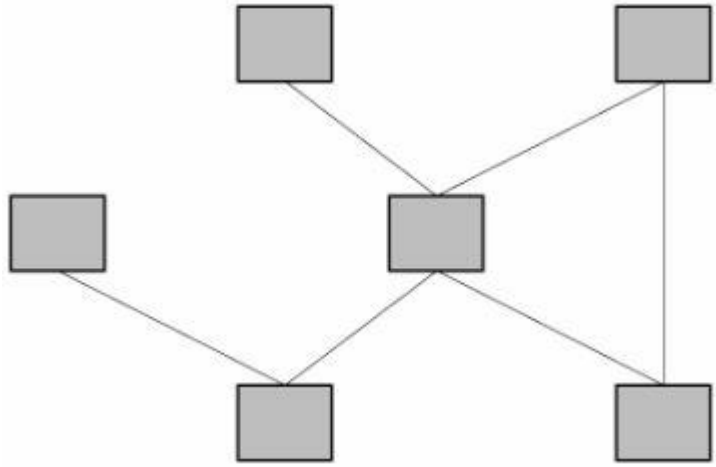


Abbildung 2: Netzwerkdatenbankmodell[12]

Relationale Datenbankmodelle

Das relationale Datenbankmodell ist seit 1970 die wichtigste Entwicklung der Datenbankmodelle und so umfassend, dass sie auch heute noch flächendeckend zum Einsatz kommt.

Die Basis für die Datenspeicherung bei diesem Modell bilden verschiedene Tabellen, in denen die Daten gehalten werden und die über Relationen miteinander verknüpft werden. Genauer gesagt gibt es mehrere Tabellen, z.B. Nutzer, Bücher und Ausleihe. In einer Bibliothek könnten in der Nutzertabelle Personendaten und in der Büchertabelle Informationen zu verfügbaren Büchern gehalten werden. Dabei repräsentiert eine Zeile je einen zusammenhängenden Datensatz. Kommt nun eine Ausleihe zustanden, werden die Daten aus beiden Tabellen miteinander verknüpft. Um Redundanz zu vermeiden, gibt es in beiden Tabellen ein sogenanntes Schlüsselattribut, welches einmalig ist und somit den entsprechenden Datensatz jederzeit wieder identifizieren kann. Der Schlüssel der Person X wird zusammen mit dem

[11] Vgl. Puff, M. (2008), S. 6
[12] Enthalten in: Puff, M. (2008), S. 6

Schlüssel des ausgeliehenen Buches Y in die Ausleihtabelle geschrieben und dokumentiert somit diese Transaktion.

Dieses Prinzip lässt sich auf eine Vielzahl, auch größerer, Transaktionen anwenden und wird auch noch heute weitverbreitet genutzt.

Für den Nutzer ist die Bedienung dieses Datenbankmodells angenehmer, denn es muss kein Wissen darüber vorliegen, welche Logik hinter der Datenspeicherung liegt. Die Zugriffe erfolgen automatisch über die Schlüsselattribute.[13]

Buchnummer	Autor	Verlag	Verlagsjahr	Titel	Datum
123456	Hans Vielschreiber	Musterverlag	2007	Wir lernen SQL	13.01.2007
123457	J. Gutenberg	Gutenberg und Co.	1452	Drucken leicht gemacht	01.01.1452
123458	G. I. Caesar	Handschrift Verlag	-44	Mein Leben mit Asterix	16.03.-44
123459	Galileo Galilei	Inquisition International	1640	Eppur si muove	1641
123460	Charles Darwin	Vatikan Verlag	1860	Adam und Eva	1862

Nutzernummer	Vorname	Nachname
3546	Hans	Vielleser
3547	Jens	Mittelleser
3548	Erich	Wenigleser

Nutzernummer	Buchnummer
3546	123456
3546	123458
3546	123459
3547	123457
3547	123460

Abbildung 3: Tabellen eines relationalen Datenbanksystems[14]

2.1.3 Merkmale

Datenbanksysteme (DBS), also die Zusammensetzung aus einer Datenbank (DB) und dem dazugehörigen Datenbankmanagementsystem (DBMS),[15] haben einige Charakteristika, welche erfüllt sein müssen, um einen ordnungsgemäßen Betrieb zu garantieren. Diese sollen im Folgenden erläutert werden.

- Der ***Datenbestand*** einer Datenbank muss alle relevanten Daten in ordentlicher Struktur beinhalten. Das bedeutet, dass die Daten strukturiert und dementsprechend

[13] Vgl. Puff, M. (2008), S. 6 ff.
[14] Enthalten in: Puff, M. (2008), S. 7-8
[15] Vgl. Sieprath, S. (2005), S. 8

auswählbar sind sowie ebenfalls Metadaten, also Beschreibungen über die eigentlich abgespeicherten Daten, enthalten.[16]

- Eine ***Mehrfachnutzung*** des Datenbanksystems muss möglich sein. Ein Datenbanksystem ist erst dann richtig produktiv, wenn mehrere Benutzer gleichzeitig daran arbeiten können. Hierfür ist besonders wichtig, dass die Benutzer mit den richtigen (aktuellen) Daten arbeiten und sich bei gleichzeitiger Bearbeitung von Datensätzen „nicht im Weg stehen". Durch Zugriffsrechte können verschiedene Nutzer außerdem mit unterschiedlichen Ansichten bzw. Daten ausgestattet werden.[17] [18]

- Die ***Datenintegrität*** einer Datenbank garantiert, dass die Informationen in der Datenbank jederzeit korrekt logisch verknüpft sind und ebenso abgelegt werden. Außerdem sorgt Datenintegrität dafür, dass die Datenregeln wie z.B. Format eingehalten werden. Redundante Speicherung wird ebenso ausgeschlossen wie die Abhängigkeit von der spezifischen Kombination einer Anwendung mit einer Datenbank.[19] [20]

- Die ***Sicherheit*** einer Datenbank ist gleich in zweifacher Hinsicht zu beachten: Schutz vor Zugriff von außen und Schutz vor Datenverlust. Entsprechende Zugriffsrechte geben den richtigen Nutzern Zugang zum Datenbanksystem, während sichergestellt werden sollte, dass andere nicht die Daten auslesen können. Im Falle eines Systemabsturzes oder Stromausfalles muss weiterhin gewährleistet werden, dass Daten nicht verloren gehen. Eine Datensicherung in angemessenem Abstand muss demnach durchgeführt werden.[21] [22]

Transaktionen sind Vorgänge, bei denen Datensätze in eine Datenbank geschrieben bzw. geändert oder gelöscht werden. Heutzutage können Transaktionen teils sehr komplexe Vorgänge darstellen, die vieler Schritte bedürfen. Daher ist es wichtig, dass Transaktionen ordnungsgemäß ablaufen und auch abgeschlossen werden. Um dies zu garantieren, wurde das Atomicity-Consistency-Isolation-Durability-Modell (ACID) entwickelt, welches die folgenden Merkmale umfasst:

[16] Vgl. Philipps Universität Marburg (o.J.)
[17] Vgl. ebenda
[18] Vgl. Sieprath, S. (2005), S. 15 ff.
[19] Vgl. Philipps Universität Marburg (o.J.)
[20] Vgl. Sieprath, S. (2005), S. 15 ff.
[21] Vgl. HTWK Leipzig (o.J.)
[22] Vgl. Sieprath, S. (2005), S. 16

- ***Atomicity*** bedeutet, dass Transaktionen entweder ganz oder überhaupt nicht durchgeführt werden. Sie sind demnach unteilbar, schlägt auch nur ein Teil einer Transaktion fehl, muss die komplette Transaktion abgebrochen werden. Am Beispiel einer Überweisung wird dies deutlich: Einer der Transaktionsschritte wäre „Geld von meinem Konto abbuchen“, ein weiterer „Geld auf Fremdkonto einbuchen“. Schlägt der zweite Schritt fehl, würde der Empfänger irgendwann das fehlende Geld vom Sender einfordern, während es bei diesem jedoch längst abgebucht wurde. In diesem Falle wäre es also verloren gegangen.

- ***Consistency*** bedeutet, dass sich die Datenbank nur von einem korrekten Zustand in einen anderen korrekten Zustand bewegt. Sollten Daten zu irgendeinem Zeitpunkt einer Transaktion inkonsistent sein (falsches Zahlenformat, etc.), muss die komplette Transaktion abgebrochen werden. Der dauerhaft konsistente Zustand einer Datenbank wird durch Datenintegrität garantiert.

- ***Isolation*** bedeutet, dass Transaktionen einzeln gesehen werden müssen und auch so durchgeführt werden sollten. Zwei Transaktionen, die zur selben Zeit von der Datenbank bearbeitet werden sollen, sollten ihre Resultate nicht gegenseitig beeinflussen, sprich die eine Transaktion sollte keine unvollständigen Daten der anderen Transaktion lesen. Daher muss in diesem Fall zuerst eine Transaktion vollständig abgeschlossen werden, bevor die andere ausgeführt wird. Weiterhin sagt Isolation aus, dass, wenn eine Transaktion isoliert funktioniert, sie ebenso mit anderen Transaktionen zusammen (in entsprechender Reihenfolge) funktioniert.

- ***Durability*** bedeutet, dass jede Transaktion, die einmal durchgeführt und in die Datenbank geschrieben („commit“) wurde, niemals wieder verloren werden kann. Diese Eigenschaft bleibt auch dann gültig, falls ein Datenbank-/Festplattenfehler oder sogar ein Systemabsturz auftritt. Durch Logging und ausreichend Backups wird sichergestellt, dass eine Wiederherstellung jederzeit möglich ist.[23]

[23] Vgl. Chapple, M. (o.J.)

2.2 Grundsätze von In-Memory-Datenbanken

Als Einstieg in das Thema der In-Memory-Datenbanken (IMDB) soll zunächst die Bedeutung des Begriffs „In-Memory“ geklärt werden.

Als In-Memory Technologie wird grundsätzlich alles bezeichnet, was sich mit der Speicherung von Daten im Arbeitsspeicher befasst. Das bedeutet, dass der flüchtige Arbeitsspeicher hier als der primäre Datenspeicher dient.[24]

Arbeitsspeicher oder auch Hauptspeicher bezieht sich auf den Speicher, „der die gerade auszuführenden Programme oder Programmteile und die [dabei] benötigten Daten enthält. Der Hauptspeicher ist eine Komponente der Zentraleinheit (CPU). Da der Prozessor unmittelbar auf den Hauptspeicher zugreift, beeinflussen dessen Leistungsfähigkeit und Größe in wesentlichem Maße die Leistungsfähigkeit der gesamten Rechenanlage“.[25]

Wie aus der Einführung bereits hervorging, ist die In-Memory-Technologie seit einigen Jahren sehr populär und gewinnt durch den Trend um Big Data immer mehr an Beliebtheit. In den folgenden Unterkapiteln sollen daher die Bedeutung und die Grundzüge von In-Memory erarbeitet werden.

Als IMDB werden Datenbanken bezeichnet, bei denen der Arbeitsspeicher als primäres Speichermedium dient. Damit unterscheiden sie sich von herkömmlichen oder traditionellen DBMS, bei welchen die Datenspeicherung auf persistenten Speichermedien stattfindet.[26]

Bereits seit Mitte der 1980er Jahre existieren Produkte, welche nach dem In-Memory Prinzip arbeiten. Damals war es die Datenbank TM1, welche seit 2008 als IBM Cognos TM1 zu IBM gehört.[27] Allerdings existiert der große IT Trend um IMDBs erst seit einigen Jahren. Denn seit 2008 haben die führenden IT-Unternehmen im Bereich Datenbanken (IBM, Oracle und SAP) neue Versionen von In-Memory Datenbanken auf den Market gebracht, welche sich für den großflächigen Einsatz in Unternehmen eignen.

Wie bereits erwähnt, hängt dies eng mit der Entwicklung um Big Data zusammen, denn seither haben sich die Anforderungen an Datenbanken verändert. Es bestand schon lange das Potenzial einer dauerhaften Speicherung von Daten sowie die Möglichkeit, diese ständig verfügbar zu machen.[28] Allerdings ist es heutzutage immer entscheidender, wie schnell auf

[24] Vgl. Bauer, A. (2013), S. 174
[25] Duden (1993), S. 296
[26] Vgl. McObject (2014)
[27] Vgl. Gruenes, J./Ilacqua, C./Oehler, K. (2012)
[28] Vgl. Funk, M./Marinkov, B./Paar, M. (2012)

diese Daten zugegriffen werden kann und wie schnell diese wachsenden Datenmengen analysiert werden können.[29]

Die Hauptunterschiede von IMDBs im Vergleich zu traditionellen Datenbanksysteme beziehen sich auf die Art der Datenspeicherung. Dadurch ergeben sich große Unterschiede im Bezug auf die Persistenz der Daten sowie auf die Geschwindigkeit, mit welcher auf die Daten zugegriffen werden kann bzw. Abfragen durchgeführt werden können.

Einige der wichtigsten Grundsätze, die sich bei IMDBs ergeben und aus welchen besondere Merkmale und Eigenschaften resultieren, werden im Folgenden beschrieben.

Wenn man die Grundsätze der IMDBs betrachtet, kann man bei dieser Analyse zwei unterschiedliche Grundaspekte beleuchten: zum Einen die Hardwaresicht und zum Anderen die konzeptionelle Umsetzungssicht.

Durch verschiedene Neuerungen im Bereich der Hardware ergeben sich erst die Potenziale und Voraussetzungen für die Umsetzung von In-Memory Technologien. Allerdings treten in diesem Zusammenhang auch gewisse Limitationen auf, welche es zu umgehen gilt.

Die erste Wandlung in diesem Bereich ist eine Multi-Core Architektur, wodurch sich auf einem normalen Serverboard mittlerweile zwischen 64 und 128 Kernen befinden. Diese einzelnen Serverboards werden meist mit weiteren sog. Nodes in einem Cluster zusammengeschlossen. Dies bildet die Anforderung und Grundlage an eine konsequent umgesetzte Parallelisierung der Rechenvorgänge und ermöglicht eine hohe Skalierbarkeit, wodurch die Rechenleistung deutlich steigt. Trotz dieser Parallelisierung wurde eine wesentliche Optimierung durch die Beschränkungen traditioneller Festplatten verpasst. Das Problem an dieser Stelle ist, dass sich die Geschwindigkeiten von CPUs unproportional schneller entwickelt haben als die der Festplatten. Daher stellte diese Schnittstelle des Festplattenzugriffs bis dato die eigentliche Flaschenhalsproblematik dar.[30]

[29] Vgl. Cao, J./Gupta, U. (2013)

[30] Vgl. Boncz, P./Kersten, M. L./Manegold, S. (o.J.)

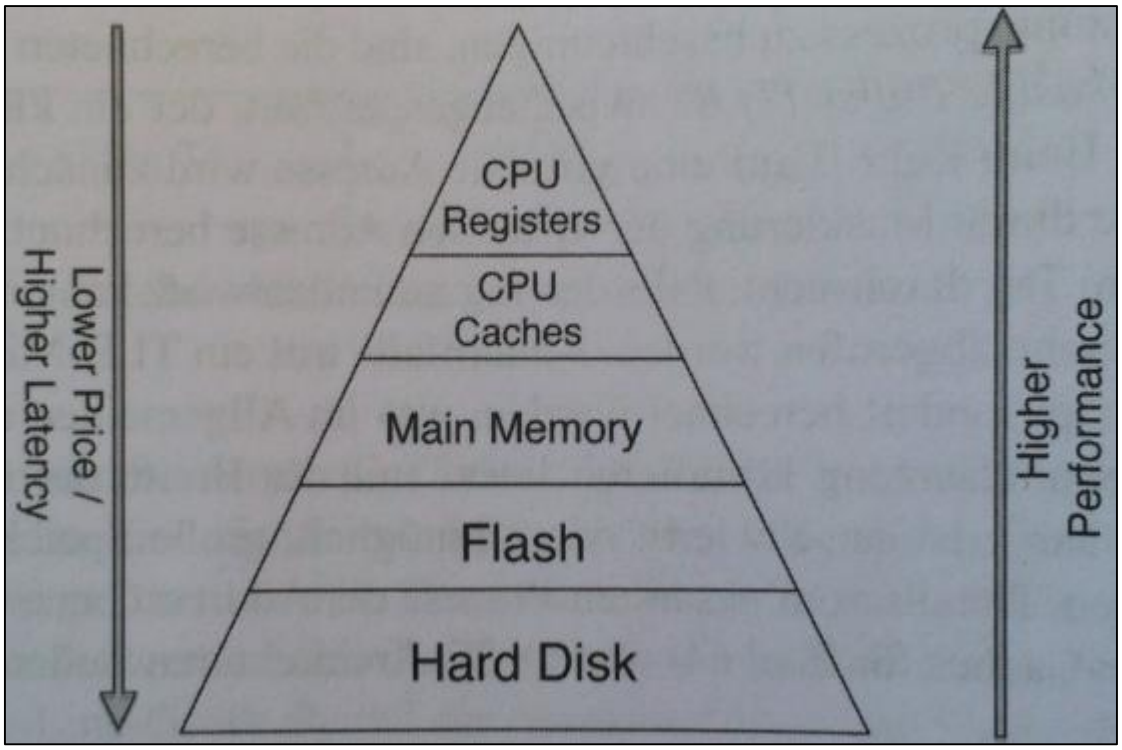


Abbildung 4: Speichermedien (Überblick)[31]

In die Betrachtung der Hardware fallen diverse Komponenten, wobei an dieser Stelle vor allem die Flaschenhalsproblematik ausschlaggebend ist. Unter Rücksichtnahme der Speicherhierarchie wird deutlich, dass die oben beschriebene Flaschenhalsproblematik nicht mehr besteht, da es bei IMDBs nicht mehr zu einem Zugriff auf die Festplatte als primäres Speichermedium kommt. Es wird ebenfalls deutlich, dass die unteren Schichten in der Hierarchie bei einer In-Memory Anwendung nicht relevant sind. Dadurch steigen einerseits die Übertragungsgeschwindigkeit, aber andererseits auch die Kosten.

Bisher war die Übertragungszeit durch die Schreibgeschwindigkeit der Festplatte limitiert. Da diese nun wegfällt, stellt sich die Frage, ob das neue System generell keine Flaschenhalsproblematik mehr aufweist. Dies ist allerdings nicht der Fall. Die CPU-Frequenzen sind zwar entscheidend angestiegen, allerdings nicht die der Speicherbusse, wodurch die Latenzzeiten immer noch relativ hoch sind im Vergleich zu denen, die durch die CPUs möglich wären. Daraus folgt, dass der Speicherzugriff länger benötigt und die Kosten für Speicherzugriffe steigen, da viele CPU-Zyklen nicht genutzt werden können. Daher stellt die Schnittstelle zwischen CPU und Hauptspeicher den neuen Flaschenhals dar.

[31] Enthalten in: Plattner, H. (2013a)

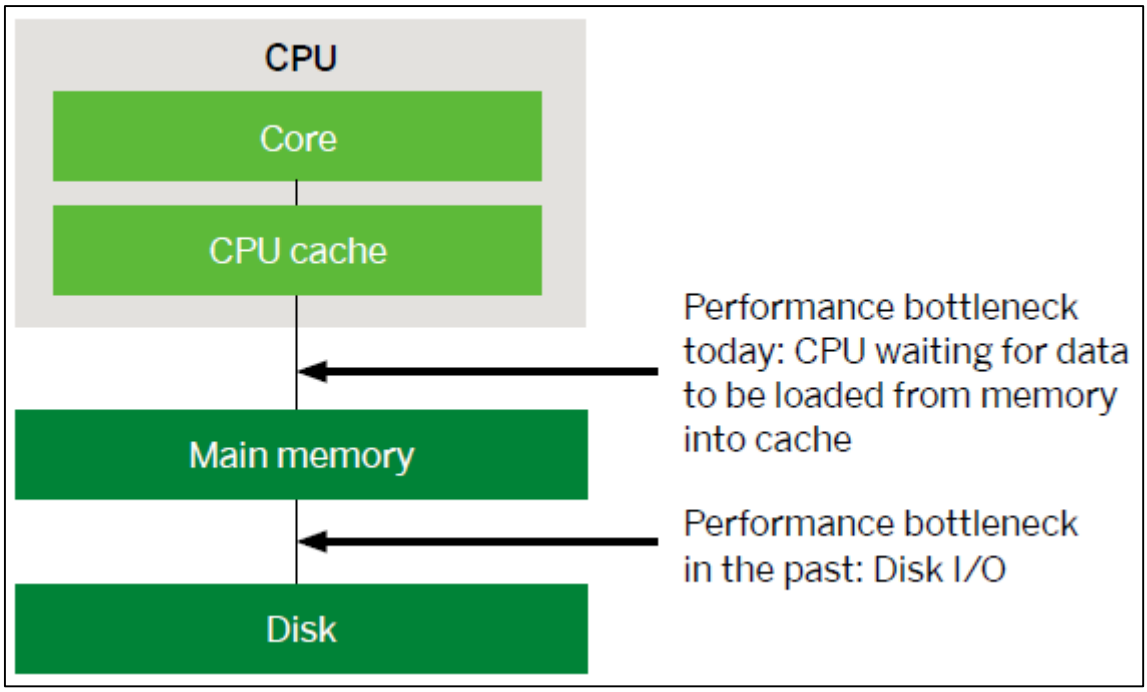


Abbildung 5: Flaschenhalsproblematik [32]

Zunächst einmal sieht es so aus, als würde das bedeuten, dass die CPU eine Rechenleistung vorlegt, die durch den proportional dazu langsameren Speicherzugriff limitiert ist. Daraus lässt sich wie in Abbildung 5 gezeigt wird folgern, dass sich die Flaschenhalsproblematik lediglich verschoben hat.

Neben der Sicht auf die Hardware-spezifischen Komponenten existieren als Teil der IMBD Architektur diverse Merkmale und Konzepte, deren strukturelle Umsetzung versucht, diesen Flaschenhals aufzulösen.

Hierbei kann man allgemein zwei Problemfelder unterscheiden. Zum Einen besteht die Problematik der Persistenz, zum Anderen existiert die Notwendigkeit zur Performancesteigerung um den Flaschenhals zu umgehen.

Die Leistungsoptimierung einer Datenbank, speziell einer IMDB, hängt stark mit folgenden Merkmalen und Kriterien zusammen: Einerseits spielt hier die veränderte Speicherarchitektur in Form von Spaltenorientierung eine Rolle, sowie die darauf basierenden Komprimierungsverfahren. Andererseits sind jedoch Parallelisierung und Partitionierung sowie Indexierung besonders relevant für die Behebung der Flaschenhalsproblematik, hierfür wird die Grundlage durch die neue Hardwarearchitektur gegeben, welche oben bereits beschrieben wurde.

[32] Enthalten in: SAP HANA (2012)

Abseits der Optimierung existieren weitere Ansätze zur Behebung der Probleme, die sich rund um das Thema Persistenz ergeben. Hier geht es hauptsächlich darum, dass der Arbeitsspeicher, der bei IMDBs als primäres Speichermedium gilt, ein flüchtiger Speicher ist. Daher existieren diverse Ansätze, mit denen versucht wird für Persistenz zu sorgen, um somit einen Datenverlust zu verhindern. Im Zusammenhang mit Persistenz existiert bei vielen IMDBs ein theoretisches Konzept, welches sich auf aktive und passive Daten bezieht. Dies bildet hierbei eine der Grundlagen für das Umsetzen der Persistenz bei IMDBs. Dieser Grundsatz der aktiven und passiven Daten stellt sich folgendermaßen dar.

Aktive Daten beziehen sich auf Geschäftsprozesse, die noch nicht abgeschlossen sind, während sich passive Daten auf bereits abgeschlossene Geschäftsprozesse beziehen. Bei diesen finden keine Änderungen mehr statt. Aus diesem Grund werden passive Daten direkt auf langsamere Speichermedien geschrieben. Im Gegensatz dazu wird der Hauptspeicher für aktive Daten verwendet. Dadurch kann die Größe des Hauptspeichers verringert werden, was sich hauptsächlich in den Kosten widerspiegelt. Zusätzlich dazu findet eine Speicherung von Logs und Snapshots (Schnappschüssen) im nichtflüchtigen Speicher statt, damit im Falle eines Systemausfalls die Daten gesichert sind. Diese Trennung zwischen aktiven und passiven Daten sorgt ebenfalls dafür, dass die Antwortzeiten nicht sonderlich variieren, da sich die aktiven Daten, also die Daten die aktuell verwendet werden, im Hauptspeicher befinden und somit ein schnellerer Zugriff möglich ist.[33]

Die nachfolgende Grafik stellt einen beispielhaften Aufbau eines In-Memory Datenbank Managementsystems (IMDBMS) dar. Mit Hilfe dieser Grafik werden die bereits beschriebenen Grundsätze aus Sicht der Hardware sowie konzeptionellen Umsetzung erneut verdeutlicht.

[33] Kapitel orientiert an Plattner, H. (2013a), S. 19 ff.

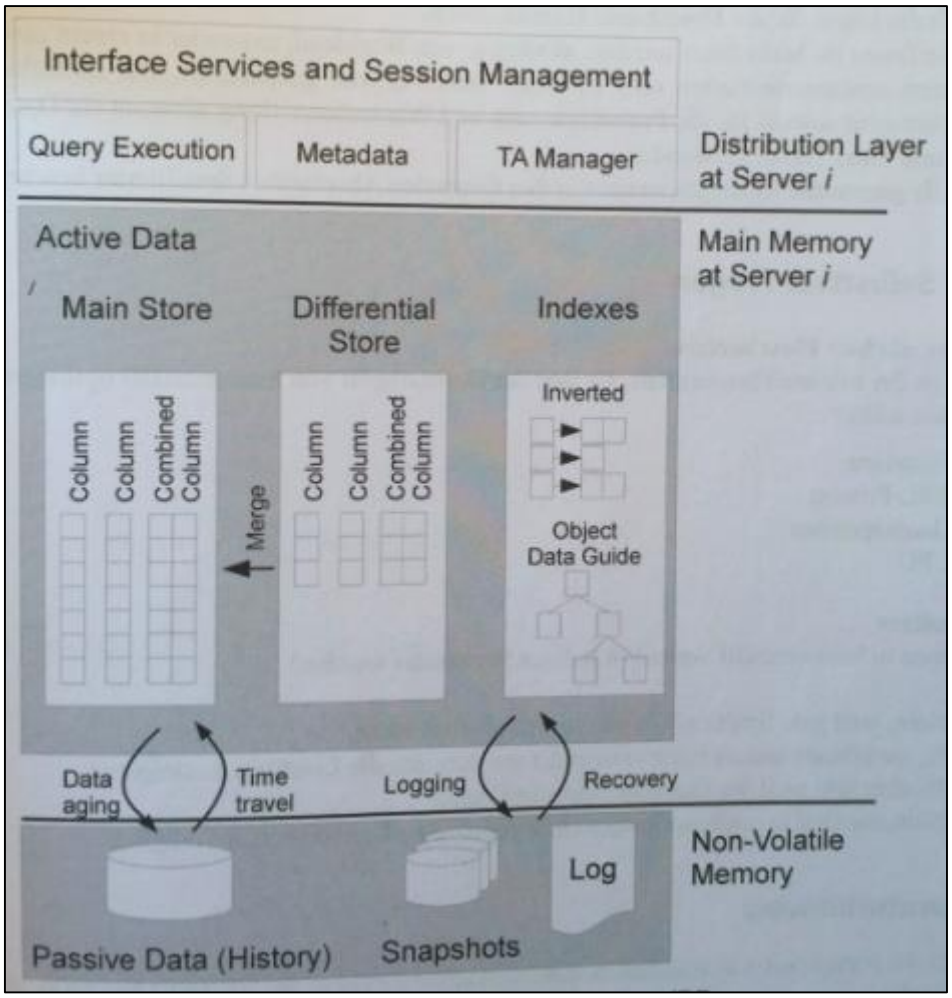


Abbildung 6: Architekturüberblick IMDB[34]

Diese kurze Einführung sollte einen Überblick über die entscheidenden Merkmale von IMDBs geben, um diese in einen Kontext zu setzen. Im Folgenden werden diese dann noch näher erklärt und ausgeführt.

2.3 Umsetzung der DB-Merkmale bei In-Memory-Datenbanken

Nachdem bereits die grundsätzlichen Merkmale bestehender Datenbanksysteme wie die Integrität von Daten oder ihre Sicherheit erläutert sowie die einzelnen Elemente des ACID-Modells kurz behandelt wurden, soll der Fokus der vorliegenden Ausarbeitung nun auf die Anwendung dieser Merkmale bei In-Memory Datenbanken gelegt werden. Dazu werden im Folgenden verschiedene Eigenschaften betrachtet, anhand dieser verdeutlicht werden kann, inwieweit beispielsweise die Anforderungen des ACID-Modells bei In-Memory Datenbanken erfüllt werden. Zur genaueren Untersuchung liegt eine Auswahl an Merkmalen zu Grunde, die

[34] Enthalten in: Plattner, H. (2013a)

besonders relevant für die Abgrenzung von In-Memory Datenbanken gegenüber herkömmlichen Technologien sind. Dazu wurde die Fachliteratur mit Hinblick auf die übereinstimmenden Merkmale studiert, die entscheidend zu den Performanceunterschieden beitragen.

2.3.1 Spaltenorientierte Speicherung und Verarbeitung der Datensätze

Big Data und Business Analytics gehören derzeit zu den wichtigsten Trends der IT-Branche. Überall wachsen die Datenmengen ununterbrochen – Unternehmen suchen daher nach Möglichkeiten, diese Daten zu nutzen und aus ihnen einen Wettbewerbsvorteil zu generieren. Der Bedarf nach Business-Intelligence-Lösungen und einer damit verbundenen schnellen Analyse und Auswertung von Daten ist in den letzten Jahren enorm gestiegen.[35]

Ein mögliches Mittel zur Erfüllung dieser Anforderung ist die Verwendung von spaltenorientierten Datenbanken. Diese bilden in ihrer Organisation eine Alternative zur zeilenorientierten Speicherung und Verarbeitung von Datensätzen, welche vor allem bei traditionellen Datenbanken gängig sind. Wie der Name bereits suggeriert, werden die Daten hier nicht zeilenweise, sondern spaltenweise abgespeichert. In herkömmlichen Datenbanken liegen alle Attribute eines Datensatzes, also alle Zeilen, hintereinander (Tupel). Wird dann eine Auswertung durchgeführt, muss immer die gesamte Datenbank mit allen Zeilen und Spalten durchsucht werden, um die geforderten Daten zu finden. Wie Abbildung 7 verdeutlicht, hat die vertikale Verarbeitungsrichtung der zeilenorientierten Verarbeitung zur Folge, dass komplette Spalten, die für die Auswertung nicht relevant sind, trotzdem analysiert werden und sich damit die Verarbeitungszeit unnötigerweise erhöht. Ein konkreter Vergleich verdeutlicht diesen Unterschied noch deutlicher: Bei zeilenorientierten Datenbanken handelt es sich um ein Karteikartensystem. Will man eine Auswertung aller Karteikarten durchführen, muss man jede einzelne Karteikarte betrachten und die benötigten Informationen notieren.[36] Diese Methode bietet sich im Beispiel unten an, wenn für einen bestimmten „Key“ sämtliche Informationen benötigt werden, wie es häufig beim klassischen Online Transaction Processing (OLTP) der Fall ist.[37] Bei spaltenorientierten Datenbanken hingegen würden die gleichartigen Informationen, also die gleichen Attribute, alle auf einer einzigen Karte stehen. Möchte man dann, wie beim Online Analytical Processing (OLAP) häufig gefordert, in der Datenbank aus Abbildung 7 beispielweise eine Auswertung über den gesamten Umsatz aller Verkäufe machen, müsste man einzig die Karteikarte mit der Spalte „Sales“ betrachten und auswerten,

[35] Vgl. Strehlitz, M. (2011)
[36] Vgl. eliteinformatiker (2011)
[37] Vgl. Herden, O. (2012)

wodurch die Verarbeitungszeit enorm verringert wird und nicht, wie bei der zeilenorientierten Speicherung, unnötige Daten ebenfalls durchsucht werden müssen.

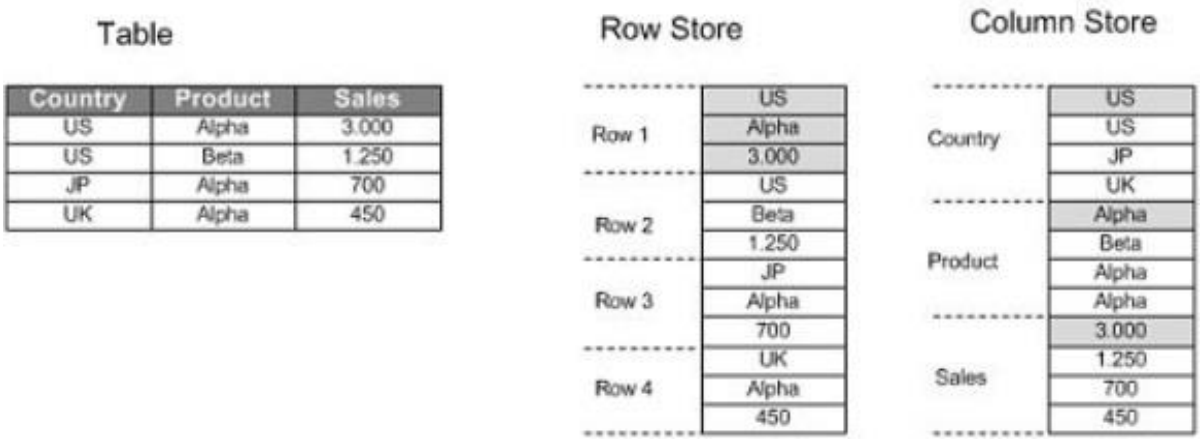


Abbildung 7: Zeilenorientierte vs. spaltenorientierte Speicherung[38]

Zusammenfassend eignen sich spaltenorientierte Datenbanken also, wenn

- eine Auswertung über viele Zeilen, aber nur über wenige Spalten stattfinden soll:

  Beispiel: SELECT SUM(Sales) FROM Table;

- eine Spalte gleichzeitig für alle Werte neu beschrieben werden soll:

  Beispiel: UPDATE Table SET Sales = Sales*1.19;

Zeilenorientierte Datenbanken hingegen sind von Vorteil, wenn

- sämtliche Attribute einer oder weniger Zeilen ausgelesen werden sollen:

  Beispiel: SELECT * FROM Table WHERE Country=JP;

- eine neue Zeile eingefügt werden soll, bei der sämtliche Attribute bereits vorliegen:

  Beispiel: INSERT INTO Table (Country, Product, Sales) VALUES (‘US’, ‘Beta’, 999);

2.3.2 Komprimierungsverfahren zur effizienten Speicherung und Verarbeitung der Datensätze

Im Rahmen der Spaltenorientierung einer Datenbank verwenden diverse In-Memory Datenbanken überdies ein zusätzliches Verfahren zur weiteren Reduktion des benötigten Speicherplatzes sowie zur Verkürzung der Übertragungszeit der Daten. Mit Hilfe von Kompression können nicht nur mehr Informationen zur selben Zeit prozessiert werden, auch

[38] Enthalten in: Datenbanken Online Lexikon (2013)

der Transfer von Daten zwischen Hauptspeicher und Prozessor wird deutlich reduziert.[39] Bei der Komprimierung, auch Kompression genannt, wird von der Ähnlichkeit benachbarter Daten und einer häufig übereinstimmenden Semantik dieser Daten bei einer spaltenorientierten Speicherung Gebrauch gemacht. Konkret existieren mehrere Kompressionsverfahren, von denen die drei bekanntesten im Folgenden erläutert werden sollen. Die Wörterbuchkompression (Dictionary-Encoding), Lauflängencodierung und Differenzspeicherung erreichen eine Komprimierung der Daten durch jeweils verschiedene Herangehensweisen.

Bei der Wörterbuchkompression von Lempel und Ziv werden sich häufig wiederholende Muster wie z.B. Wörter durch einen kürzeren Schlüssel ersetzt und in einer Art Wörterbuch hinterlegt.[40] Zusätzlich wird der Wert in der Spalte durch den Schlüssel ersetzt, welcher den ursprünglichen Wert im Wörterbuch eindeutig referenziert.[41] Abbildung 8 verdeutlicht dies anhand eines Beispiels. Die unkomprimierte Speicherung weist eine Vielzahl von sich wiederholenden Wörtern auf. Diese werden innerhalb des Wörterbuchs durch kurze Zeichen beschrieben, so dass das Wort „Frucht“ dann den Schlüssel „01“ besitzt. In der ursprünglichen Spalte werden die entsprechenden Wörter durch die Schlüssel ersetzt, wodurch folglich der erforderliche Speicherplatz enorm sinkt. Anstelle von 17 Einträgen mit 139 Zeichen á 1 Byte sind komprimiert noch 4 Einträge á 2 Byte nötig, sodass 97% des Speicherplatzes reduziert werden kann.

Die Lauflängencodierung erweist sich nur dann als vorteilhaft, wenn in einer Spalte lange Folgen sich wiederholender Symbole auftreten. In diesem Fall werden die einzelnen Abschnitte gleicher Zeichen in Form der Anzahl des jeweiligen Zeichens als Wertpaar (Wert und Wertanzahl) zusammengefasst.[42] Wie in Abbildung 8 erkennbar, wird aus „aaaaa“ nach der Komprimierung „5a“. Im Beispiel unten kann der Speicherplatz um 56% verringert werden.

Abschließend bietet auch die Differenzspeicherung unter bestimmten Voraussetzungen ein deutliches Komprimierungspotenzial. Wichtigste Voraussetzung ist, dass die einzelnen Einträge der Spalte bis auf wenige Zeichen exakt übereinstimmen. Nur der Anfangswert und die jeweilige Differenz zum Vorgänger muss dann in die Auswertung einbezogen werden. Auch hier zeigt Abbildung 8, dass anstelle der Zahlenketten „121355“, „121356“ usw. ausschließlich die Differenz zwischen den jeweiligen Zahlen, also hier z.B. „1“ verwendet werden muss. Im genannten Beispiel ist damit ein Ersparnispotenzial von 75% vorhanden.

[39] Vgl. Wittwer, C. (2009), S. 40
[40] Vgl. Krueger, J. u.a. (o.J.), S. 5
[41] Vgl. Krueger, J. u.a. (o.J.), S. 5
[42] Vgl. ebenda, S. 5

	Wörterbuchkompression	Lauflängencodierung	Differenzspeicherung
Unkomprimierte Speicherung	Kleidung|Kleidung|Frucht| Foto|Foto|Kleidung|Frucht| Tiefkühlwaren|Tiefkühlwaren| Frucht|Frucht|Frucht| Kleidung|Foto|Tiefkühlwaren| Tiefkühlwaren|Tiefkühlwaren|	aaaaabbbccccccaaa	121355 121356 121358 121361 121362 121364 121366
Komprimierte Speicherung	Wörterbuch: Kleidung 00 Frucht 01 Tiefkühlwaren 10 Foto 11 Daten: 00|00|01|11|11|00|01|10|10|01 |01|01|00|11|10|10|10|	5a3b7c3a	… 1 2 3 1 2 2
Kompressionsfaktor	Unkomprimierte Speicherung: 17 Einträge, 139 Zeichen á 1 Byte Komprimiert: Pro Eintrag 2 Bit KF = 32,7	Unkomprimiert: 18 Zeichen Komprimiert: 8 Zeichen KF = 2,25	Unkomprimiert: 7 Integer-Werte á 4 Byte Komprimiert: 7 Byte-Werte KF = 4
Speicherplatz-ersparnis	97%	56%	75%

Abbildung 8: Kompressionsverfahren in Datenbanken[43]

Bei jedem der erläuterten Komprimierungsverfahren ist zu beachten, dass im Falle von Datenmodifkationen eine erneute Komprimierung des Datenbestandes nötig wird. Dieser Umstand lässt sich jedoch umgehen, indem ein Puffer eingesetzt wird, der die Datensätze unkomprimiert ablegt und Änderungen sammelt.[44] Die Datenbank liefert als Ergebnis auf eine Anfrage dann die komprimierten Daten aus dem Hauptspeicher sowie die unkomprimierten Änderungen aus dem Puffer. Dennoch müssen die Änderungen zur Integration in den Hauptspeicher zusätzlich asynchron mit dem bestehenden Datenbestand zusammengeführt werden.[45]

Unabhängig davon, für welches Kompressionsverfahren sich ein Unternehmen letztlich entscheidet – es muss immer Rücksicht auf ein ausgeglichenes Verhältnis zwischen der Kompressionseffektivität (mit dem Ergebnis einer erhöhten Speicherkapazität), dem Umfang des erforderlichen Datentransfers (als Ursache für eine erhöhte Performance) und dem Einsatz von Ressourcen genommen werden.

2.3.3 Indexierung

Indexierung, also das Erstellen von Indizes auf die Datensätze einer Datenbank, bezeichnet allgemein den Vorgang, das Suchen und Sortieren von Daten zu beschleunigen. Eine

[43] Enthalten in: Herden, O. (2012)
[44] Vgl. Krueger, J. u.a. (o.J.), S. 6
[45] Vgl. ebenda, S. 6

grundlegende Indexierung findet beim Erstellen der Datenbank schon durch die Auswahl der adäquaten Primärschlüssel statt. Diese kann jedoch weiter verfeinert und optimiert werden. Ein Index wird als Zeiger verwendet, über den das Verhältnis zwischen mehreren Spalten beziehungsweise Zeilen definiert wird. Anstelle eines sequentiellen Vorgangs bei Such- und Sortiervorgängen tritt durch optimierte Indexierung ein Algorithmus meist in Form von B+ Bäumen auf.[46]

Bei zeilenorientierter Speicherung dienen Indizes dazu, die Zugriffsgeschwindigkeit zu erhöhen, indem man verhindert, dass bei einer Abfrage die komplette Tabelle untersucht wird, wie bereits in Punkt 2.3.1 erläutert.

Spaltenorientierte Speicherung selbst ist bereits ein Vorgang zur Optimierung von Such- und Sortierprozessen, daher kann theoretisch auf die Verwendung von zusätzlichen Indizes verzichtet werden, was ansonsten in jedem Fall die Pflege eben dieser bedarf.[47] Der Vorteil von Komprimierungsverfahren bei spaltenorientierter Datenspeicherung zeigt sich nicht bei der Festlegung der Spaltenreihenfolge über Indizes, da bei Spaltenorientierung jede Spalte für sich optimiert werden kann, sondern bei der Kompressionsrate von zusammengesetzten Indizes.

Es besteht daher die Möglichkeit, beispielsweise auf Bitmap-Indizes zurückzugreifen, die die Komprimierung weiter verbessern können. Der Bitmap-Index speichert mehrdimensionale Daten in sogenannten Bitmustern (Bitmap). Dabei ist es ratsam, auf Tabellenspalten mit geringer Kardinalität zurückzugreifen, um nach diesen Schlüsseln in aufsteigender Reihenfolge (zum Beispiel Geschlecht, Wohnort, Alter, Nachname, Vorname) zu sortieren.[48]

Bei einigen DBMS ist es möglich, jede beliebige Spalte nach den gewünschten Attributen zu durchsuchen, um so die Treffer zu bestimmen. Der Versatz (Offset) der gefundenen Treffer wird dann als Index verwendet, um den Wert weiterer Attribute feststellen zu können. Dies führt wiederum dazu, dass nicht benötigte Daten nicht gelesen werden. Das Ergebnis ist letztendlich ein Zustand, in dem komplexe Objekte über ihre Attribute gefiltert werden können.[49]

Der Index behält jedoch eine komplexe Position, wenn es um Umsortierung zur Kompressionssteigerung geht. Ein Beispiel in diesem Zusammenhang ist der Fall, dass der Index das Geschlecht und den Wohnort aller Personen umfasst. Dann muss die Kompression auf diese Attribute durch Umsortieren gesteigert werden. Bei der Suche nach nur einem dieser

[46] Vgl. Kemper, A.; Eickler, A. (2009): Datenbanksysteme, München: Oldenbourg Verlag, S. 218
[47] Vgl. Datenbanken Online Lexikon (2013)
[48] Vgl. Chan, C.-Y.; Ioannidis, Y. (Hrsg.) (1998)
[49] Vgl. Hasso Plattner Institut (Hrsg.) (o.J.)

Attribute wird der Index allerdings nutzlos.[50] Von einem leistungsorientierten Standpunkt ist das Erstellen von Indizes jedoch durchaus sinnvoll, da die erreichbare Performancesteigerung in den Fällen, in denen der Index nützlich ist, die Fälle, in denen Indizes nutzlos oder redundant sind, in der Regel deutlich übertrifft.

2.3.4 Parallelisierung und Parallele Operationen bei IMDBS

Bei Parallelisierung von Datenbanken muss grundlegend zwischen "Pipelined Parallelism" und "Data Parallelism" unterschieden werden. Pipelined Parallelism bedeutet, dass ein zweiter Prozess startet, während ein erster Prozess noch nicht abgeschlossen ist, verwendet wird also eine auch aus dem Projektmanagement bekannte 'lead'-Zeit. Data Parallelism bedeutet wiederum die Partitionierung der Datensätze, sodass diese individuell bearbeitet und im Anschluss wieder zusammengeführt werden können. Darüber hinaus ist bei DBS eine weitere Parallelisierung über intra- und inter-query Parallelisierung möglich. Intra-query Parallelisierung parallelisiert die Operatoren innerhalb der Abfrage, wohingegen inter-query für die Parallelisierung mehrerer Abfragen über die Reihenfolge deren Ausführung bewirkt. Die Parallelisierung bei IMDBS wird im Folgenden auf der Basis von Data Parallelism aufgezeigt, da es sich hierbei um einen essentiellen Aspekt der Leistungsoptimierung bei IMDBS handelt.[51]

Es gibt verschiedene Bereiche und Operationen, in die Parallelisierung im Hauptspeicher untergliedert werden muss, um den Gesamtnutzen dieser Operationen zu ermöglichen. Hierzu zählen unter anderem: "Parallel Join", "Parallel Aggregation", "Parallel Select" und "Parallel Data Processing", welche nachfolgend erklärt und analysiert werden.

Joins bezeichnen die Zusammenführung eines kartesischen Produkts und einer Selektion, wobei das kartesische Produkt generell die Kombination zweier oder mehrerer Relationen ausdrückt, auf die im Anschluss eine Selektion durchgeführt wird.

Um Parallelisierung in diesem Bereich aufzuzeigen, wird der sogenannte "Hash-Join" als Beispielfall verwendet. Ein Hash-Join erstellt eine Hashtabelle, in der die jeweiligen Join-Attribute festgehalten werden. Verweist der ausgewählte Wert auf ein Bucket in der Hashtabelle (Hashtable) welches NOT NULL liefert, so wird dieses Bucket mit dem definierten Tupel verglichen.[52]

---

[50] Vgl. Association for computing Machinery (Hrsg.) (2008)
[51] Vgl. Plattner, H. (2013)
[52] Vgl. Hasso Plattner Institut (Hrsg.) (o.J.)

Der Hash-Join wird für verschiedene Join Methoden verwendet und ist durch seinen strukturellen Aufbau eine vergleichsweise einfache Joinausführung. Zur Parallelisierung von Joins gibt es verschiedene Varianten wie zum Beispiel Sequential Hashing ("Sequentielles Zerkleinern") und anschließendes Parallel Probing ("Paralleles Untersuchen"). Sequential Hashing bedeutet, dass jeweils ein Thread (Prozess) den jeweiligen Hashtable konstruiert und danach die zusammengeführte Spalte in einer kleineren, geeigneteren Tabelle prozessiert wird. Der Hashwert jedes Elements wird im Hashtable gespeichert. Danach wird im Parallel Probing eine horizontale Partitionierung der Untersuchungstabelle (Probetable) vorgenommen. [53]

Der Nachteil bei Join-Parallelisierung auf diese Art und Weise ist, dass die Hashtabelle an alle beteiligten Kerne/Threads kopiert werden muss, die Ergebnisse werden anschließend in einer lokalen Tabelle gespeichert und zusammengeführt. Durch eine große Anzahl an residenten Hashes und zufälligen Zugriffen auf zwei Kerne in der gleichen Central Processing Unit (CPU) besteht somit die Gefahr, dass Hashes Daten vom jeweiligen CPU Cache verdrängen. Eine mögliche Lösung wäre folglich, die Hashthreads zu einer anderen CPU zu verlagern.[54]

Es muss also diskutiert werden, ob es ebenso möglich ist, die Hashing Phase zu parallelisieren. Hierzu wird die Tabelle unter den Threads partitioniert, aus denen diese wiederum eine Hashtabelle erzeugen. Diese werden an die CPU Cachegröße angepasst, um Hauptspeichersuchläufe zu verhindern. Bei Erreichen des Speicherlimits wird die Hashtabelle in den Speicher ("global Buffer") kopiert und eine neue, leere Hashtabelle wird erstellt. Zum Schluss werden mittlere Tabellen aus dem global Buffer mit Hilfe von "Merger Threads" je nach Hashwert zusammengeführt, bevor eine globale Hashtabelle erstellt wird.[55] Der Prozess wird in der folgenden Grafik veranschaulicht:

[53] Vgl. Plattner, H. (o.J. a)
[54] Vgl. Plattner, H (o.J. d)
[55] Vgl. Plattner, H. (o.J. a)

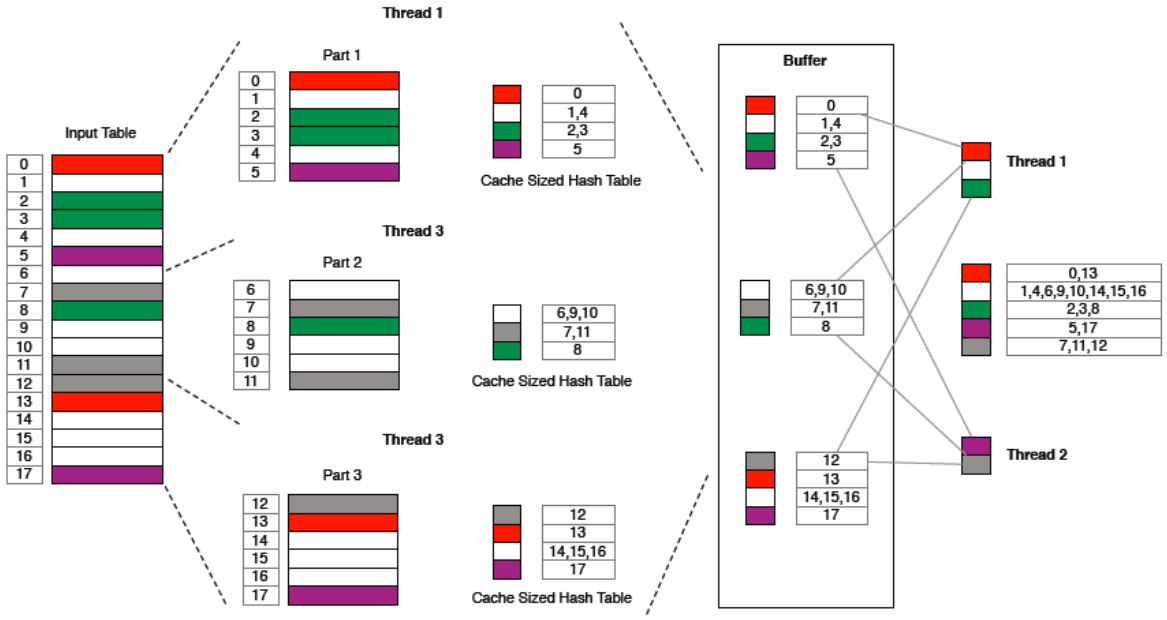


Abbildung 9: Durchführung eines parallelen Joins[56]

Parallel Aggregation nutzt einen sehr ähnlichen Vorgang wie Parallel Joins und ist daher thematisch eng verknüpft. Aggregatfunktionen einer Datenbank geben zusammengefasste Informationen über die Daten einer Datenbank aus. Dabei werden Gruppierungen ("Grouping") verwendet, die sich sehr ähnlich dem Hashing verhalten. Für erfolgreiches Grouping müssen zunächst Gruppen mit dem gleichen Wert identifiziert werden, was bei Parallel Aggregation geschieht. Durch Hashing sind Threads in der Lage, gleiche Werte zu erkennen und diese in einem global verfügbaren Feld zusammenzuführen. Gleich den Parallel Joins werden die Gruppen an einen globalen Buffer ausgelagert, sobald sie die Cachegröße erreichen. Nach der Zusammenführung der Gruppen kann eine Aggregation stattfinden. Es ist auch denkbar, die Aggregationsfunktion in die Hashingoperation zu integrieren. Hierbei muss, sobald ein gleicher Wert beim Hashen erkannt wird, die Aggregationsoperation angewendet werden. Problematisch bei der integrierten Variante ist jedoch, dass zum Zeitpunkt der Aggregation das Wertespektrum unvollständig ist, wodurch keine verlässlichen Aussagen über beispielsweise Median getroffen werden können.[57] Es ist somit sehr wichtig, das geeignete Parallelisierungsverfahren in Bezug auf die geplanten Operationen zu wählen. Im Folgenden wird dies veranschaulicht:

[56] Enthalten in: Plattner, H. (o.J. d)
[57] Vgl. Plattner, H. (o.J. b)

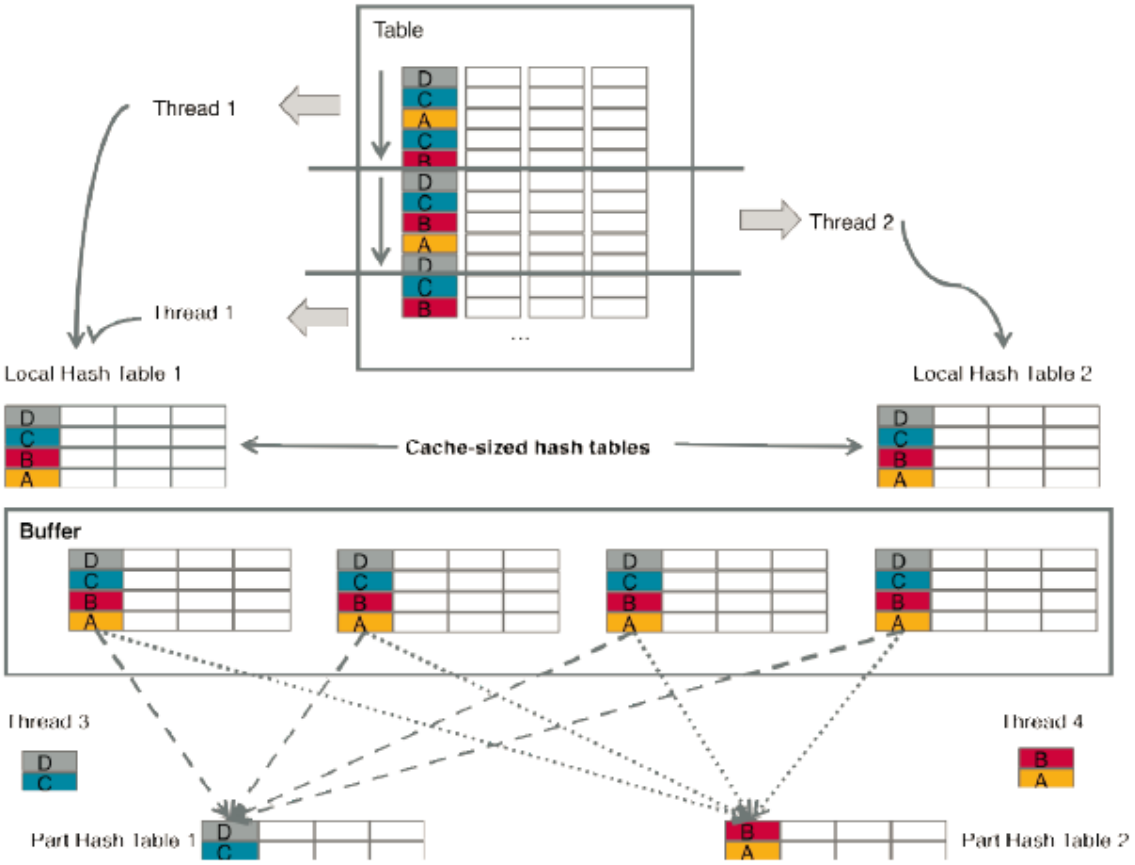


Abbildung 10: Parallele Aggregation eines Datensatzes[58]

Der einfachste Weg, über Joins und Aggregation hinaus entsprechende Operationen wie einen Parallel Select durchzuführen, geht über getrennte Partitionen. Eine Tabelle wird also auf mehrere Partitionen verteilt, sodass eine zu prozessierende Abfrage, die eine verteilte Tabelle anspricht, an alle Partitionen gerichtet und dort ausgeführt werden muss, um dann über einen UNION Befehl zusammengeführt zu werden. Soll dieser Prozess innerhalb der Nodes ausgeführt werden, und nicht über eine Anzahl an Nodes verteilt, müssen die Prozessorkerne der Nodes parallelisiert werden. Allerdings werden die Daten nicht über eine Anzahl von Nodes partitioniert, sondern der Node komplett zur Verfügung gestellt. Dann wird die Abfrage geteilt und jede entstandene Teilabfrage erhält einen anderen Ausschnitt der Tabelle. Die Anzahl an verfügbaren Kernen stellt also die Anzahl der Partitionen innerhalb der Node dar, auf die dann jeweils eine Unterabfrage gestartet wird. Durch einen UNION Befehl werden nach Beendigung sämtlicher Unterabfragen die Ergebnisse zusammengeführt. Möglich ist auch eine integrierte Lösung von beiden vorgestellten Optionen. Das bedeutet zunächst die Partitionierung auf

[58] Enthalten in: Plattner, H. (o.J. b)

Node-Ebene und anschließend eine Aufteilung auf die Node-Kerne.[59] Die folgende Rechnung zeigt beispielhaft eine erreichbare Parallelität:

25 Nodes * 40 Kerne pro Node = 1000-fache Parallelität.

Diese Optimierungsmöglichkeiten lassen sich durch die Zugriffsgeschwindigkeit und die bereits besprochene Indexierung der Tabellen maßgeblich (negativ) beeinflussen. Durch In-Memory Datenbanken können diese Zugriffsgeschwindigkeiten und damit die kritischen Flaschenhälse umgangen werden. Wie aus der folgenden Grafik hervorgeht, wird die Tabelle entsprechend der Anzahl an parallel zugreifenden Threads partitioniert.[60]

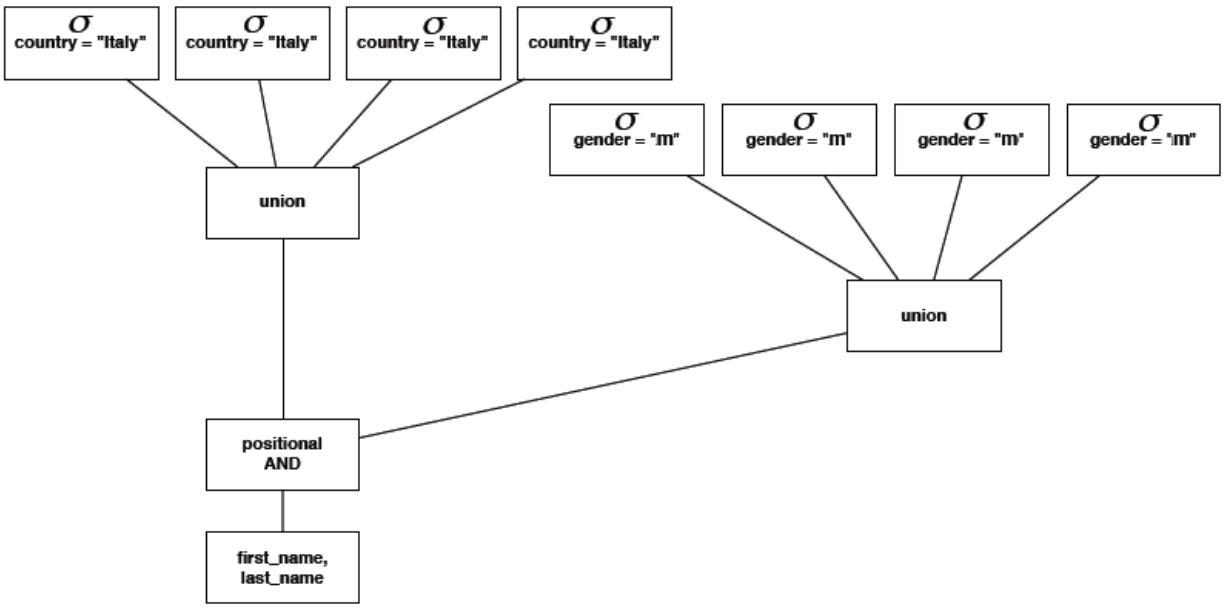


Abbildung 11: Paralleler Select einer partitionierten Tabelle[61]

Angekommen an einem Punkt der maximalen Optimierung, vor allem bei IMDBs, kann auf SIMD Operationen zurückgegriffen werden. Hierbei wird der Zugriff auf Daten durch den Hauptspeicher parallelisiert. Im Gegensatz zu RISK CPUs, Prozessoren mit reduziertem Befehlssatz, werden vektorisierte Operatoren verwendet. Diese werden in die CPU integriert, um Operationen über verschiedene Datensätze innerhalb von speziellen Registern durchzuführen.[62] Die Quintessenz der Parallelisierung in In-Memory Datenbanken bleibt jedoch die bereits beschriebene Partitionierung.

[59] Vgl. Plattner, H. (o.J. c)
[60] Vgl. Plattner, H. (o.J. e)
[61] Vgl. ebenda
[62] Vgl. Plattner, H. (o.J. e)

2.3.5 Persistenz bei In-Memory-Datenbanken - Wiederherstellungsverfahren bei Systemausfall

Betrachtet man das Konzept der In-Memory Datenbanken grundlegend, so stellt man fest, dass sie in der Basisvariante nicht vollständig dem ACID Konzept folgen. Konkretes Problem ist hier die "Durability", also die dauerhafte Speicherung (Persistenz), da In-Memory Datenbanken auf flüchtigen Speicher zurückgreifen. Dieser verliert die Daten, sobald er von der Stromquelle getrennt wird. Um trotzdem Persistenz gewährleisten zu können, greifen In-Memory Datenbanksysteme (IMDBS) also auf alternative Methoden zurück:

- Snapshots / Checkpoints
  Hierbei wird der jeweils aktuelle Stand der Datenbank über einen Snapshot gesichert. Allerdings wird das IMDBS hierbei nur begrenzt persistent, da Änderungen, die nicht erfasst werden, trotzdem verloren gehen. Der folgende Fall ist denkbar: Eine Datenbank wird jeweils beim Herunterfahren über einen Snapshot gesichert. Im laufenden Prozess fährt die Datenbank unkontrolliert herunter und es gehen alle Daten bis zum letzten gültigen Snapshot verloren.

- Protokolldateien (Logs)
  In einer Protokoll- oder Logdatei werden, wie bei traditionellen Datenbanken, Änderungen an den Datensätzen direkt erfasst. In Falle eines Absturz können sämtliche Änderungen nachvollzogen und zum Beispiel durch einen Recoverymanager wiederhergestellt werden.

- Nichtflüchtiger RAM und ROM Speicher
  Statischer RAM-Speicher verfügt über eine unabhängige Stromversorgung in Form einer Batterie (battery RAM), welche den Speicher "nichtflüchtig" werden lässt. Alternativ kann auch electrically reasable programmable ROM verwendet werden. In beiden Fällen können Daten aus dem letztgültigen, konsistenten Zustand wieder hergestellt werden.[63]

- Replikation auf herkömmlichen Speicher (Hybride)
  IMDBS können zur Herstellung von Persistenz auf nichtflüchtigen Speicher wie zum Beispiel Flash zurückgreifen, um darauf Replikationen des aktuellen Dateizustands durchzuführen. Diese Replikation kann allerdings auch auf klassischem Speicher stattfinden. Der Hauptspeicher ist somit weiterhin in der Lage, schnelle Lesezugriffe zu bewirken, während die aktuellen Daten (auf die

[63] Vgl. Narayanan, D./Hodson, O. (2012)

kein Lesezugriff erfolgt) auf einen klassischen Speicher geschrieben werden.[64] Vorstellbar ist ein Modell aus 3 Instanzen, bestehend aus CPU, Flash und konventionellem Speicher. Der Flashspeicher hält die Daten, die im Prozess häufig von der CPU benötigt werden, während das klassische Speichermedium für selten genutzte Daten und Backups verwendet wird.

[64] Vgl. Cole, B. (2007)

3 Umsetzung der Merkmale anhand konkreter Beispiele (A. Croll, T. Dietz, P. Koch, T. Soetebeer)

Im folgenden Kapitel werden die in Kapitel 2 beschriebenen Merkmale von In-Memory-Datenbanksystemen anhand konkreter Beispiele praktisch erläutert.

Für die beispielhafte Darstellung wurde an dieser Stelle die IMDB von SAP, SAP HANA ausgewählt. HANA steht hierbei für „High Performance Analytic Appliance“.[65] Zunächst einmal gab es einige Punkte, die zu der Entscheidung für SAP HANA geführt haben. Dazu gehört vor allem die Bedeutung, welche SAP mit HANA am Markt einnimmt. So haben diese mit insgesamt etwas über 20% den größten Marktanteil im Bereich In-Memory Computing.[66] Des Weiteren hat sich bei einer anfänglichen Untersuchung gezeigt, dass SAP HANA nahezu alle relevanten In-Memory Merkmale umsetzt, die bereits eingeführt wurden. Im Gegensatz zu einigen anderen kommerziell verfügbaren Systemen existiert hier in der Literatur fast überall eine konkrete Umsetzung, was die beispielhafte Darstellung wesentlich erleichtert. Letztlich stellt sich ein weiterer Vorteil heraus, durch welchen die Wahl auf dieses System gefallen ist. Hasso Plattner, einer der Mitbegründer von SAP, ist besonders aktiv im Umfeld der Entwicklung von In-Memory Technologie. Die Entwicklung eines Datenbanksystems (SanssouciDB) durch das Hasso Plattner Institut bildet daher die Grundlagen von SAP HANA ab. Dieses System wird im Rahmen studentischer Lehrveranstaltungen, sowie verschiedener Literatur für die Öffentlichkeit zugänglich gemacht.

Die Wahl auf SAP HANA fiel also primär auf Basis dieser drei Gründe: Der Stellung am Markt, der technischen Umsetzung und der öffentlich verfügbaren Literatur zu diesem Thema.

Die folgende Ausarbeitung orientiert sich vor allem an den Zusammenhängen, die in der Einführung zu IMDBs bereits vorgestellt wurden.

3.1 Datenspeicherung bei SAP HANA

Wie aus der Einführung in das Thema der In-Memory Datenbanken sowie derer Merkmale bereits hervorgegangen ist, ist eines der wichtigsten Themen im Zusammenhang mit IMDBs die Datenspeicherung. Auf das neue Konzept der Spaltenorientierung wurde in dieser Arbeit bereits intensiv eingegangen. Des Weiteren ist die Bedeutung dessen von großer Wichtigkeit für die Umsetzung der IMDB spezifischen Merkmale.

[65] Wörtlich aus SAP (2014)
[66] Vgl. Kurzlechner, W. (2013)

Es ist bereits angeklungen, dass IMDBs hauptsächlich auf das Konzept der spaltenorientierten Speicherung setzen. Allerdings werden auch einige hybride Lösungen verwendet, die aus einer Kombination aus Spalten- und Zeilenorientierung bestehen. Grundsätzlich wird vorgeschlagen, dass SAP HANA unterschiedliche Datensätze auf unterschiedliche Art und Weise abspeichert.

Das heißt, dass zu einer spaltenorientierten Speicherung dann geraten wird, wenn eine Tabelle viele Spalten hat, eine Durchsuchung nach einer bzw. weniger Spalten durchgenommen werden soll, die Spalten im Gegensatz zu den Zeilen nur wenige unterschiedliche Werte beinhalten, oder sobald spaltenorientierte Berechnungen durchgeführt werden sollen. Zu einer zeilenorientierten Speicherung wird im Umkehrschluss dann geraten, wenn die Tabelle wenige Reihen besitzt, immer nur ein einzelner Eintrag prozessiert und bearbeitet werden muss, immer auf jede Spalte eines Eintrages zugegriffen werden muss, die Spalten viele verschiedene Werte beinhalten, oder schnelle Durchsuchungen und Aggregationen nicht benötigt sind.[67]

Wie bereits aus diesen Anforderungen ersichtlich wird, erfüllen die meisten Unternehmensanwendungen sowie Datenbanken eher die Anforderungen an eine spaltenorientierte Speicherung. Daher kann man auch sagen, dass die Aufteilung je nach Umsetzung bei SAP HANA ca. 95% spaltenorientierte Speicherung umfasst.[68]

Zeilenorientierte Speicherung wird bei SAP HANA hauptsächlich für die Ablage von Metadaten, Application Server internen Daten und Konfigurationsdaten verwendet. Zusätzlich erlaubt SAP HANA, dass spaltenorientiert sowie zeilenorientiert gespeicherte Daten miteinander verbunden („Join“) werden können. Allerdings ist es effizienter, nur einheitlich gespeicherte Daten miteinander zusammenzuführen. Daher werden transaktionale Daten meist spaltenorientiert gespeichert, sofern diese häufig mit analytischen Daten angereichert werden.

Wie bereits ausführlich beschrieben, entstehen durch die Spaltenorientierung diverse Vorteile. Diese beinhalten bei SAP HANA ganz konkret folgende: Höhere Performance bei spaltenorientierten Operationen, höhere Kompressionsraten, Eliminierung zusätzlicher Indizes und Parallelisierung.

Eine zusätzliche Funktion von SAP HANA ist die Verwendung von „Temporal Tables“. Diese ermöglichen es, dass man historische Daten bearbeiten und prozessieren kann. Alle

[67] Vgl. SAP HANA (2012), S. 13 ff.
[68] Vgl. SAP HANA (2013), S. 13 ff.

Schreibanweisungen, die historischen Daten betreffen, ändern bzw. überschreiben diese nicht, sondern werden mit Zeitstempel zusätzlich hinterlegt.[69]

Um das Konzept der Spaltenorientierung, sowie deren Umsetzung noch einmal aufzeigen zu können, soll an dieser Stelle noch auf ein kurzes Beispiel eingegangen werden, welches sich in einem Unternehmensumfeld auftun könnte und die beispielhafte Umsetzung der Spaltenorientierung und deren Auswirkung auf eine Unternehmensanwendung bei SAP HANA zeigt.

Es wird von einem Beispielunternehmen ausgegangen, welches 20.000 Mitarbeiter hat. Zu jedem Mitarbeiter werden Attribute wie Vor- und Nachname, Geschlecht, Wohnort, Nationalität und Geburtsdatum gespeichert. Die angenommene Gesamtlänge für ein Tupel beträgt 200 Byte. Eine weitere Kennzahl sind die technischen Anforderungen an die Scangeschwindigkeit von 2MB/ms/Kern und eine Cache-Linien-Größe von 64 Byte.

Zunächst einmal wird die Scangeschwindigkeit bei einer zeilenorientierten Speicherung betrachtet. Hier muss die gesamte Tabelle gescannt werden und jedes Tupel muss nach dem Geschlechterattribut ausgewertet werden. Das bedeutet in Zahlen:

Zu scannender Datenbestand: 200 Byte * 20 000 = 4 MB

Antwortzeit = 4 MB / (2 MB/s) = 2s

Die Antwortzeit bei einem vollständigen Tabellenscan auf einem Kern beträgt 2 Sekunden. Dies erscheint an dieser Stelle nicht sehr viel, ist allerdings unter Berücksichtigung des kleinen Datensatzes doch ein relativ hoher Wert.

Es wird weiterhin von einer zeilenorientierten Speicherung ausgegangen, nun aber von nicht nur einem Kern auf der CPU sondern von einer Architektur mit vier Kernen. Das verändert das Beispiel folgendermaßen:

Antwortzeit = 4 MB / (4* 2 MB/s) = 0,5s

Dadurch wird die Antwortzeit auf ein Viertel reduziert, allerdings sollten wir nach wie vor den relativen überschaubaren Datensatz beachten. Auf zeilenorientierte Systeme können noch diverse andere Optimierungsverfahren angewendet werden, allerdings nicht in einer Weise, die einen merklichen Vorteil liefert.

Anschließend wird von einer spaltenorientierten Architektur ausgegangen, daraus lässt sich folgendes erkennen: Es bestehen diverse Kompressionsmöglichkeiten, da die Attribute des

[69] SAP HANA (2012), S.13 ff.

gleichen Typs nacheinander gespeichert werden und da es wahrscheinlicher ist, das nächste zu scannende Attribut in den Cache zu laden, wenn die Attribute nacheinander gespeichert werden. Dies macht sich deutlich bei der Scangeschwindigkeit bemerkbar.

Für das Beispiel bedeutet dies, dass nun von einer Wörterbuchcodierung ausgegangen wird. Diese Art der Kompression wurde bereits beschrieben und wird im Folgenden noch ausführlicher erklärt. Für den Zusammenhang hier ist zunächst einmal entscheidend, dass durch die Wörterbuchcodierung das Geschlechter Attribut ausschließlich einen Speicherplatz von 1 Bit benötigt, um die beiden Möglichkeiten „m“ und „w“ abzubilden. Für das Datenvolumen bedeutet das:

20 000 * 1 Bit = 20 000 Bits = 2,44 KB

Bei der Annahme eines vollständigen Spaltenscans für einen Kern:

Antwortzeit = 2,44 KB / (2 MB/s) = 2,44 KB / (2048 KB/s) = 0,0012 s

Für einen Scan der gleichen Spalte mit nur einem Kern, wird bei einer spaltenorientierten Architektur lediglich eine Hundertstelsekunde benötigt. Diese Zeit würde bei einem Scan mit einer Quad-Core Architektur ein Viertel der Zeit betragen. Es bleibt allerdings zu beachten, dass bei höherer Kardinalität der Attribute der Vorteil durch die Spaltenorientierung sinkt. Allerdings zeigt sich hier der Performanceunterschied, den eine spaltenorientierte Speicherung gegenüber einer zeilenorientierten Speicherung hat, wenn es um oft angewendete analytische Scanabfragen geht.[70]

Eine weitere Veränderung, die sich bei SAP HANA aus der Spaltenorientierung im Gegensatz zu anderen IMDBs ergibt ist, dass keine stringente Trennung zwischen Online-Transactional-Processing (OLTP) und Online-Analytical-Processing (OLAP) stattfindet. Dies ist ein entscheidendes Merkmal, welches auch aus den neuen Anforderungen an Unternehmensanwendungen, sowie aus einer veränderten Speicherarchitektur hervorgeht. OLTP und OLAP Anfragen unterscheiden sich in der Art der Unternehmensdaten, auf die diese zugreifen, sowie in den Zugriffen an sich. Bisher fand eine strenge Trennung beider Bereiche statt, allerdings war hier dann ein aufwendiger und teurer Extract-Transform-Load (ETL) Prozess notwendig, um beide Datenbestände zu synchronisieren.

Der Ansatz von SAP HANA wiederum ist es, dass keine Trennung von OLTP und OLAP mehr stattfindet. Das heißt, dass sowohl analytische, als auch transaktionale Abfragen auf dem gleichen Server mit dem gleichen Datenbestand stattfinden. Dieses Verfahren wird erst durch moderne Hardware ermöglicht, Mittel die bisher zur Optimierung verwendet werden mussten

[70] Beispiel orientiert sich an Plattner, H. (2013a)

werden hingegen nicht mehr nötig. Die Abfragen auf der gemischten Workload können nun sowohl zeilenbasierte Abfragen, als auch spaltenorientierte Analysen durchführen.[71]

Aus Architektursicht wird die Grundlage für das Verschmelzen von analytischen und transaktionalen Daten durch die Verwendung einer spaltenorientierten Speicherung gegenüber einer zeilenorientierten Speicherung gelegt.

3.2 Komprimierungs-und Codierungsverfahren bei SAP HANA

Der Schwerpunkt dieses Abschnitts soll auf dem Thema Wörterbuchcodierung liegen. Wie bereits erwähnt und beschrieben wurde, handelt es sich hierbei um die entscheidende Form der Komprimierung bei In-Memory Datenbanken.

SAP HANA setzt grundsätzlich drei Methoden der Komprimierung ein: Neben der Wörterbuchcodierung sind es auch die bereits anderen beschriebenen Verfahren der Lauflängencodierung und der Cluster-Codierung. Allerdings handelt es sich bei der Wörterbuchcodierung um die meistverwendete Form der Kompression. Des Weiteren bildet diese die Grundlage für die anderen Kompressionsverfahren. Bei traditionellen Unternehmensanwendungen, wie bei SAP HANA häufig der Fall, werden mit dieser Codierungsform die besten Ergebnisse erzielt.

Bezüglich der Begriffsbezeichnungen ist es noch wichtig zu erwähnen, dass in der Literatur der Begriff Wörterbuchcodierung meist verwendet wird, Wörterbuchkomprimierung allerdings auch regelmäßig auftaucht. Inhaltlich unterscheiden sich diese jedoch nicht.

Die Wörterbuchcodierung stellt die Grundlage für alle weiteren Komprimierungslösungen dar und ist somit der Ansatz zur Lösung der Flaschenhalsproblematik, was die Speicherkapazität und Bandbreite bei IMDBs betrifft.[72] Wie bereits bekannt, liegt der Vorteil der Wörterbuchcodierung darin, lange Werte, bspw. Texte, durch kurze ganzzahlige Integer darzustellen. Dadurch wird viel Speicherplatz gespart.[73] Das ist auch der Grund, warum dies die entscheidende Komprimierungsmethode bei SAP HANA und bei den führenden anderen IMDBs ist.

Wichtig im Zusammenhang mit Komprimierungsverfahren ist, zu verstehen, dass bei traditionellen, relationalen Datenbanksystemen das Problem bestand, dass mit komprimierten Daten nicht gearbeitet werde konnte. Das hat sich bei spaltenorientierten IMDBs verändert.

[71] Vgl. Plattner, H. (2013), S. 39
[72] Vgl. Plattner, H. (2013a), S. 38 ff.
[73] Vgl. Plattner, H. (2013a), S. 38 ff.

Werte, die mit Hilfe der Wörterbuchcodierung komprimiert wurden, können nun auch in komprimierter Form verarbeitet werden. Dadurch wird erneut die Flaschenhals-Problematik reduziert.[74]

Da es sich bei der Wörterbuchcodierung sowie bei den anderen Komprimierungsverfahren um grundsätzliche Elemente der Datenhaltung und von IMDBs handelt, existiert hierzu keine konkrete Dokumentation über die Umsetzung bei SAP HANA. Das Prinzip wird bei SAP HANA genauso wie bei anderen IMDBs angewandt. Anschaulich soll die Umsetzung der Kompression bei SAP HANA wieder durch ein beispielhaftes Szenario aufgezeigt werden.

Es wird erneut von dem bereits eingeführten Beispiel ausgegangen. Das komplette Datenvolumen für die Tabelle wird hier noch einmal in Erinnerung gerufen, es beträgt:

200 Byte * 20 000 (Einträge) = 4 MB

Nun soll mit Hilfe des Feldes des Vornamens gezeigt werden, was die Kompression durch Wörterbuchcodierung bedeutet.

Bei 20.000 Mitarbeitern gehen wir von 300 verschiedenen Vornamen aus. Die bisherige Spaltengröße für die Vornamenspalte ist 49 Byte.

Zunächst soll bestimmt werden, wie viele Bits nötig sind, um all diese 300 einmaligen Werte anzuzeigen.

$\log_2(300) = 8{,}23 \approx 9$ Bits

Zuvor war der benötigte Speicherplatz folgender:

20 000 * 49 Byte = 980 000 Byte = 957 KB

Nun werden die Größen der beiden zu verwendenden Vektoren berechnet. Einerseits wird der Attributvektor verwendet, welcher die Verbindung zwischen der eigentlichen Tabelle und dem Wörterbuch darstellt.

20 000 * 9 Bits = 180 000 Bits = 22 500 Byte = 21,97 KB

Das zusätzlich einzuführende Wörterbuch benötigt einen Speicherplatz von:

300 * 49 Byte = 14 700 Byte = 14,35 KB

[74] Vgl. Hasso Plattner Institut (2013)

Somit beträgt der neue benötigte Speicherplatz für die Spalte „Vorname“ lediglich 36,32 KB. Dies steht im Gegensatz zu 957 KB in einer zeilenorientierten, nicht komprimierten Speichermethodik. Daraus ergibt sich ein Kompressionsfaktor von:

unkomprimierte Größe / komprimierte Größe = 957 KB / (21,97 + 14,35) KB ≈ 26

Damit werden die enormen Leistungsvorteile dieser Komprimierungsmethodik klar. Des Weiteren zeigt sich, dass bei steigender Größe der Datensätze auch ein Einsatz der Wörterbuchcodierung immer sinnvoller wird.[75]

Laut einer Erhebung des Hasso Plattner Instituts besitzen Unternehmensdaten eine sehr geringe Entropie. Das liegt hauptsächlich daran, dass ein Großteil der Daten nicht genutzt werden. Daher weisen Unternehmensdatenbanken meist eine hohe Anzahl an Spalten auf. Des Weiteren ist die Kardinalität vieler Spalten sehr niedrig.[76] Diese Faktoren begünstigen den Einsatz von Komprimierungsverfahren enorm und, wie das Beispiel gezeigt hat, ergeben sich dadurch gute Komprimierungsfaktoren.

Eine weitere Optimierung bei Datenscans bieten sortierte Wörterbücher. Das heißt, dass Wörter mittels einer binären Suche schneller gefunden werden können, da die Wörterbücher sortiert sind und neu eingefügte Werte nicht einfach am Ende eingefügt werden. Das Problem liegt hier bei großen Datensätzen, die beim Einfügen neuer Werte immer überarbeitet werden müssen.[77]

Dieses Prinzip der Wörterbuchcodierung kommt bei jeder durchzuführenden Operation zum Einsatz. Das heißt, beim Einfügen, Löschen oder Zugreifen auf Daten spiegeln sich Merkmale der Wörterbuchcodierung wieder. Die Entlastung des „Flaschenhals“, bestehend aus Hauptspeicher und Bandbreite, ist dabei bedeutend. Allerdings wird gleichzeitig der Prozessor durch die andauernden Übersetzungstätigkeiten stärker beansprucht. Dies ist jedoch in diesem Zusammenhang zur Entlastung durchaus akzeptabel. Außerdem kann auch diese Belastung durch gezielte und gutformulierte Abfragen verringert werden.[78]

[75] Beispiel orientiert sich an Plattner, H. (2013a)
[76] Vgl. Föll H. (o.J.)
[77] Vgl. Plattner, H. (2013), S. 37 ff.
[78] Vgl. Hasso Platner Institut (2013)

3.3 Indexierung bei SAP HANA

Die Indexierung wird bei SAP HANA vom Index Server übernommen. Anders als bei anderen In-Memory-Database-Systemen werden hier keine speziellen Indexierungsverfahren wie bspw. Hash oder auch Bitmap Indexierung unterstützt.[79] Über die Technik, welche SAP HANA bei der Indexierung im Index Server anwendet, ist nur sehr wenig in der Literatur zu finden. Vereinzelt gibt es in Foren Aussagen über die Möglichkeiten von SAP HANA. Demnach werden bei den spaltenorientierten Tabellen keine zusätzlichen Indexierungsverfahren genutzt, bei traditionell zeilenorientierten Tabellen können jedoch weitere Indizes in Form von B+ Bäumen angewendet werden.[80]

Die Indexierung per Index Server wird hauptsächlich genutzt, um Daten in der Datenbank entsprechend zu sortieren. Dieser kann aber auch zu schnelleren Abfragen führen.

Der Index Server ist der Teil von SAP HANA, welcher die schnellen Berechnungen und Analysen durchführt. Im Folgenden wird dieser in seiner Bestandteilen kurz beschrieben.

- Connection and Session Management

  SAP HANA bietet eine sehr umfangreiche Möglichkeit, um die Connection and Session Management Parameter an entsprechende Situationen im Bereich von Sicherheit bzw. Datentransfer anzupassen.

- Authentication

  Bei Abfragen in einer Datenbank müssen Firmen immer vor unberechtigtem Zugriff auf sensible Daten geschützt werden. Dies betrifft auch die eigenen Mitarbeiter und wird anhand von speziellen Rollen und Privilegien in der SAP HANA Datenbank festgelegt. Entsprechend der Zugriffsrechte können Nutzer dann SQL Abfragen ausführen.

- SQL Processor

  Hier wird dafür gesorgt, dass Datenabfragen an die entsprechenden Stellen weitergeleitet werden und somit bestmöglich bearbeitet werden. Hierzu bedient sich HANA bzw. der SQL Processor der folgenden Engines:

  - Multidimensional Expressions (MDX) ist für Abfragen im mehrdimensionalen Bereich von OLAP Data Cubes.
  - Planning Engine wird für die finanzielle Planung innerhalb SAP HANA genutzt

[79] Vgl. Dontcheff, J. (2013)
[80] Vgl. Bernard, M. (2011)

- Stored Procedure Processor führt vorkonfigurierte SQL Befehle aus
- Calculation Engine bereitet Kalkulationspläne vor, um das parallele Prozessieren von Daten zu unterstützen.

- Relational Stores

  Die HANA Datenbank segmentiert die gespeicherten Daten. Daten, welche nicht dringend benötigt werden, werden vom RAM in einen Disk-Speicher transferiert. Dabei gibt es vier verschiedene Relational Stores:

  - Row Store ist optimiert für zeilenbasierte Datenverarbeitung und am besten geeignet für das Schreiben (bzw. Manipulieren) von ganzen Datensätzen.
  - Column Store ist optimiert für spaltenbasierte Datenverarbeitung und am besten geeignet für das Schreiben in ganzen Spalten bzw. sehr schnelle Datenanalyse.
  - Object Store ist eine Integration von SAP LiveCache und verantwortlich für extrem schnelle Datenverarbeitung.
  - Disk Based Store speichert die nicht aktuell benötigten Daten aus dem RAM und gibt sie dorthin frei, sobald sie gebraucht werden.

- Transaction Manager

  Dieser prozessiert einzelne SQL Statements als Transaktionen, arbeitet mit dem Persistency Layer zusammen und stellt so sicher, dass Transaktionen konform durchgeführt werden.

- Repository

  Diese dient zur Kontrolle der Metadatenobjekte wie z.B. Attribute, Analytical Views oder Stored Procedures und unterstützt den Im-/Export von Daten.[81]

[81] Vgl. Somers, K. (2012)

Die folgende Grafik gibt noch einmal einen Überblick den SAP HANA Index Server.

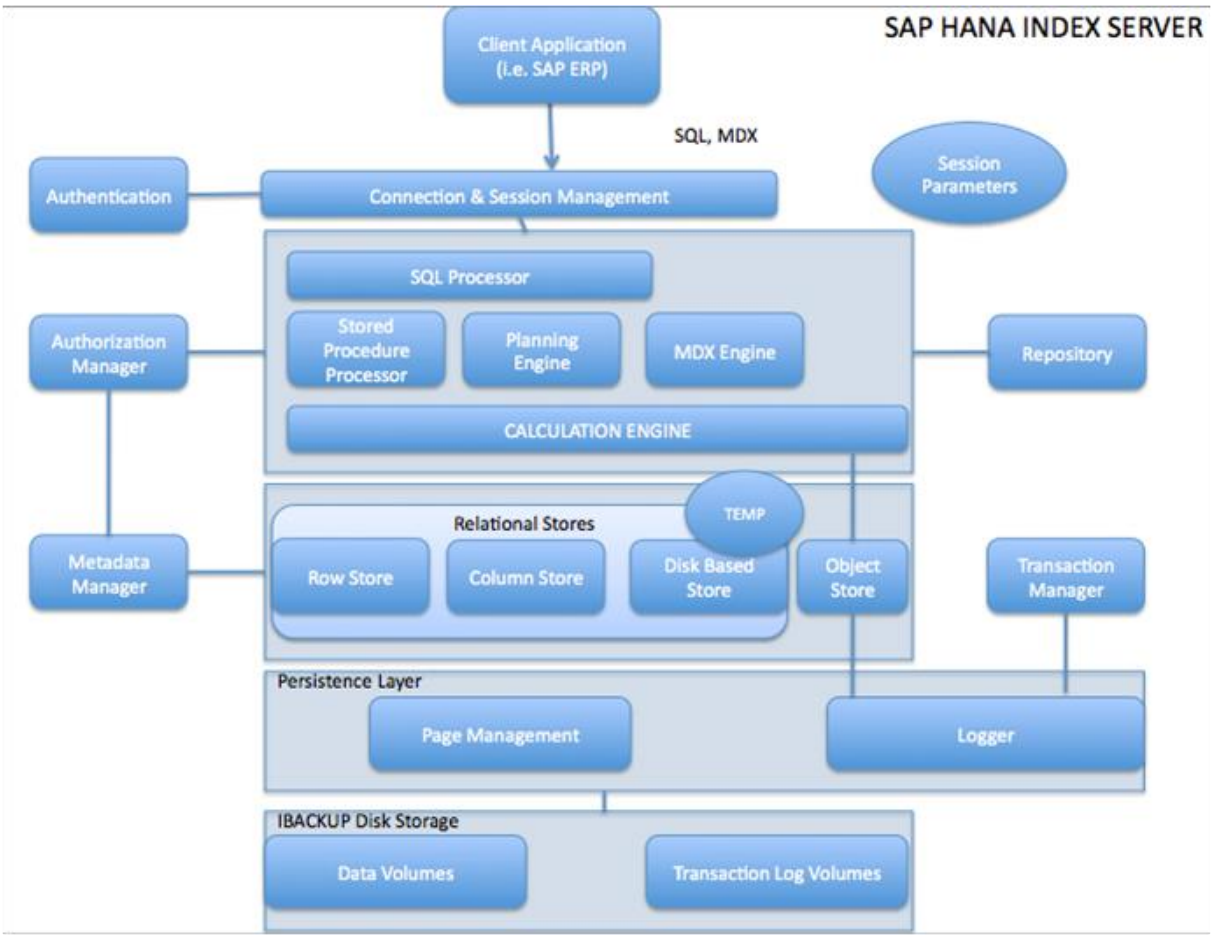


Abbildung 12: Architektur SAP HANA Index Server[82]

3.4 Parallelisierung bei SAP HANA

Wie bereits beschrieben, ist die Parallelisierung ein wichtiger Bestandteil der IMDBs. Sie stellt eine der Grundeigenschaften und Vorteile der IMDBs dar. Dabei hängt die parallele Datenverarbeitung eng mit den Grundsätzen der Indexierung und Partitionierung zusammen.

Was die Umsetzung der Parallelisierung bei SAP HANA betrifft, lässt sich diese in zwei Ebenen unterscheiden. Zum Einen die zu Grunde liegende Hardware-Schicht, zum Anderen die darauf aufsetzende konzeptionelle Umsetzung.

Ein entscheidender Vorteil der Parallelisierung ist das Vermeiden der bekannten Flaschenhälse. Die Schnittstelle zwischen Hauptspeicher und CPU stellt die eigentliche

[82] Enthalten in: Somers, K. (2012)

Flaschenhalsproblematik heutzutage dar. Da heutige Server mittlerweile bis zu 80 CPU Cores besitzen, in Zukunft womöglich bis zu 128, können diese wesentlich mehr Daten in kurzer Zeit verarbeiten. Daher versucht SAP HANA, diese Problematik zu bearbeiten, indem es nicht ausschließlich das Caching in den Hauptspeicher optimiert, aber vor allem den Speicherzugriff pro CPU Core.[83] Dieser Zusammenhang ging bereits aus der Einführung hervor, ebenso wie die Bedeutung, die der Hardware-Schicht bei der Umsetzung all dieser Merkmale zukommt.

Wenn man den konzeptionellen Aufbau von SAP HANA betrachtet, dann fällt auf, dass ein zusätzliche Schicht existiert, welche als „Parallel calculation engine“ dient. Diese hilft dabei, die SQL Anweisungen in ein optimiertes Modell zu strukturieren, sodass die Vorteile mehrerer Cores ausgenutzt werden können. Um hier Parallelisierung durchführen zu können, ist natürlich eine Partitionierung der Daten von besonderer Bedeutung. Bei sehr großen Tabellen ist sogar eine Partitionierung möglich, um die Inhalte von verschiedenen Hosts prozessieren zu lassen.[84]

Folgende Grafik zeigt einen von Intel durchgeführten Test, welcher einen „near-linear scale“, auf SAP HANA, in Abhängigkeit von der Anzahl an Threads darstellt.

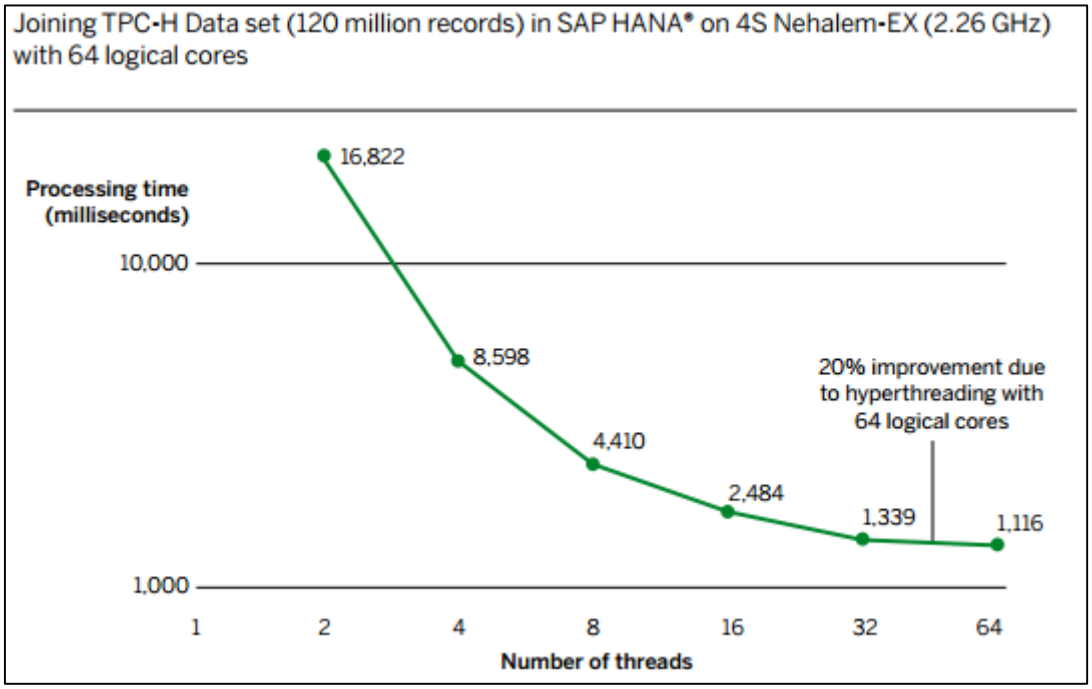


Abbildung 13: Performancevorteile durch Parallelisierung[85]

[83] Vgl. SAP HANA (2012), S. 10
[84] Vgl. SAP HANA (2012), S. 10 ff.
[85] Enthalten in: SAP HANA (2012), S. 9

Near-linear scalability bezeichnet die Fähigkeit eines Systems, nahezu linear skalierbar zu sein. Lineare Skalierbarkeit bedeutet, dass ein System im Prinzip für einen Fixpreis pro Einheit beliebig weit hochskaliert werden kann, ohne einen Performanceverlust durch einen fehlenden Systemupgrade zu erleiden.[86]

Wie die Grafik allerdings zeigt, nimmt die Processing-Time bei steigender Anzahl an Threads deutlich ab. Das zeigt eindeutig, dass durch eine effiziente Ausnutzung der multi-core Systeme SAP HANA einen enormen Performance-Vorteil durch Parallelisierung erreicht.

Die folgende Grafik zeigt die Prozess-Kette innerhalb SAP HANA als ein konzeptionelles Design.

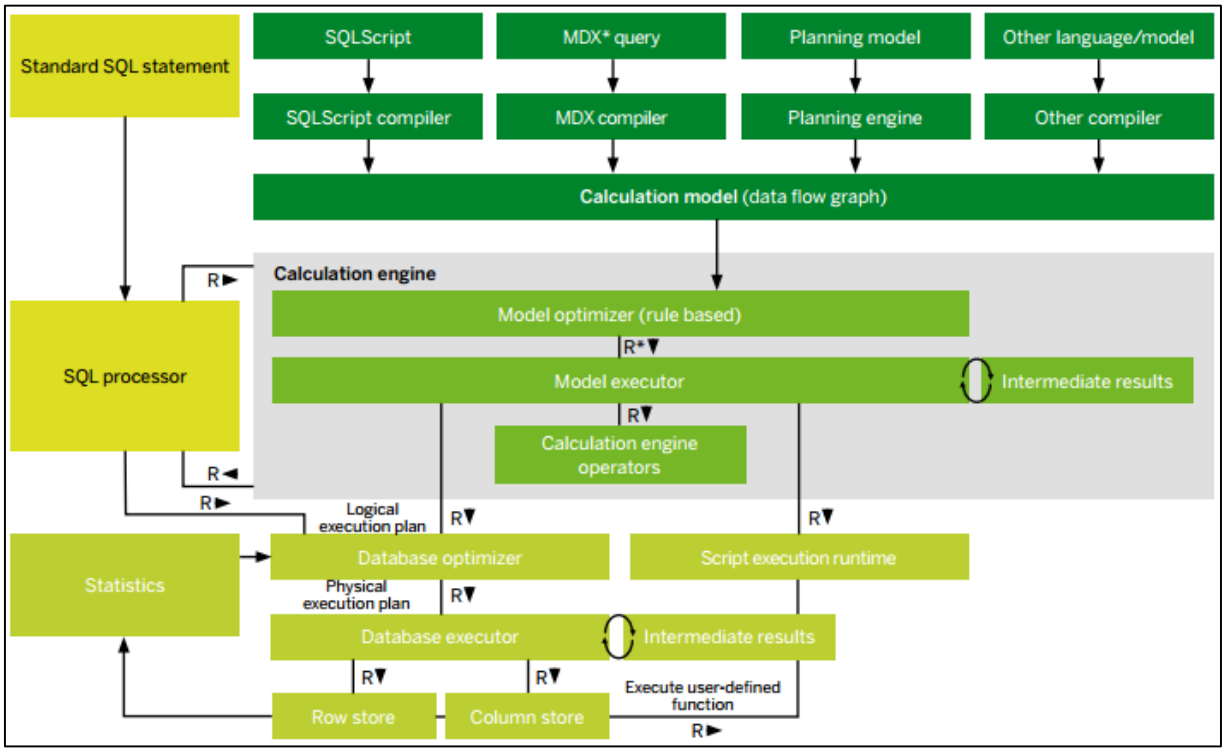


Abbildung 14: Konzeptioneller Aufbau von SAP HANA[87]

Das Calculation Model ist eine Art Datenflussdiagram. Hier werden verschiedene Operationen, wie die bereits bekannten Aggregationen und Joins, zu den unterschiedlichen Inputs definiert. Dieser Ansatz des Verzichts auf Rekursion und Schleifen ermöglicht eine Parallelisierung der Datenverarbeitung. Innerhalb des sogenannten Calculation Engine werden nun wiederum die Modelle in eigentliche Operationen unterteilt, die parallel zueinander ablaufen können. Diese

[86] Vgl. SAP HANA (2012), S. 11
[87] Enthalten in: SAP HANA (2012), S. 10

werden an den Database Optimizer weitergegeben, welcher festlegt, nach welchem Ansatz diese bearbeitet werden (entweder zeilen- oder spaltenorientiert).[88]

Um den Grundsatz der Parallelisierung, nämlich die parallele Verarbeitung der Daten, umzusetzen, muss das Calculation Model aber noch weitere Funktionen beinhalten. Zunächst einmal müssen Join- und Split-Funktionen vorhanden sein. Mit Hilfe der Split-Funktionen wird der vorhandene Datensatz partitioniert, sodass diese Partitionen nun parallel verarbeitet werden können. Dies ist einer der wichtigen Grundsätze denen SAP HANA folgt. Ein weiterer ist die Vermeidung der sequentiellen Verarbeitung von Daten. Das heißt, auf die Skalierbarkeit und Ausnutzung der vorhandenen Multi-Core Architektur wird wie bei anderen IMDBs großer Wert gelegt.

Als Beispiel hierzu wird noch der Ablauf einer parallelen Aggregation angeführt. Die Grundkonzepte der Aggregation und des Hash Tables wurden bereits eingeführt und werden daher hier nicht mehr tiefgreifend erläutert.

SAP HANA veranlasst innerhalb des geteilten Speichers einer Server-Node die Erstellung verschiedener Threads, welche dann wiederum parallel ablaufen. All diese Threads haben gleichen Zugriff auf den geteilten Speicher. Wie die untenstehende Grafik verdeutlicht, folgen diese Aggregationen dem gleichen Ablauf.

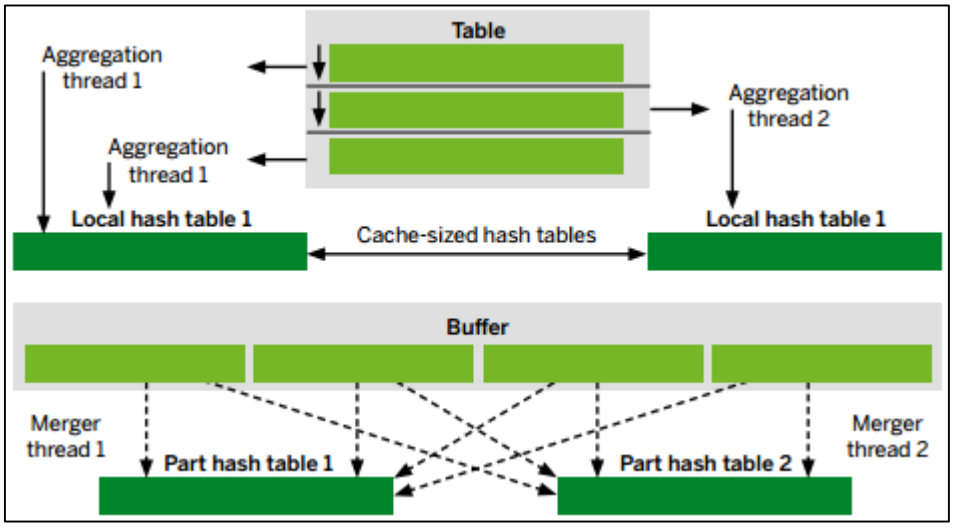


Abbildung 15: Darstellung der Parallelisierung[89]

Zunächst holt sich jeder Thread eine Partition der Input-Relation. Dann findet die eigentliche Aggregation dieser Teil-Relation statt. Dieser Ablauf wird so oft wiederholt, bis der komplette Input verarbeitet wurde. Am Ende werden dann die Ergebnisse mittels einer merge-Funktion

[88] Vgl. SAP HANA (2012), S. 12
[89] Enthalten in: SAP HANA (2012), S. 12

wieder zusammengeführt. Dies geschieht durch sogenannte merger-Threads, die auf die verschiedenen Hash-Tables zugreifen.[90]

Die vorherigen Paragraphen haben gezeigt, wie SAP HANA die Parallelisierung konzeptionell umsetzt. Des Weiteren zeigt sich hier erneut, wie eng die Parallelisierung mit den anderen Merkmalen der Partitionierung und auch der Indexierung zusammenhängt, um die Performance einer IMDB zu optimieren.

3.5 Persistenz bei SAP HANA

Hochverfügbarkeit und Persistenz sind wichtige Begriffe, wenn es um geschäftskritische Anwendungen geht, denn jeder Datenverlust oder auch schon jede Nichtverfügbarkeit von Unternehmensanwendungen kann ein Unternehmen viel Geld kosten und unangenehme Folgen haben.

Bei SAP HANA wird diese Persistenz durch ein speziell bereitgehaltenes Persistency Layer, welche u.a. dafür sorgt, dass alle Transaktionen geloggt und im Notfall nachvollziehbar sind, umgesetzt. Persistenz und Hochverfügbarkeit sind jedoch auf weitere, umfassendere Maßnahmen zurückzuführen, welche im Folgenden erläutert werden.

Persistenz wird dadurch erreicht, dass mögliche Schwachstellen eines Systems eliminiert werden. Da es aber, unabhängig von der Qualität von Hardware oder Software, immer zu Fehlern (auch bedingt von äußeren Umständen) kommen kann, ist die beste Möglichkeit, Ausweichlösungen für den Fall von Ausfällen bereitzuhalten. Diese Redundanz wird in den Teilen Hardware, Netzwerk sowie Data Center verfolgt:[91]

- **Hardware-Redundanz**

  Die zertifizierten Hardware-Partner für SAP HANA bieten Hardware-Konfigurationen an, welche verschiedenste Ebenen von Redundanz anbieten. Alle Bauteile der Hardware (z.B. Kühlung, Stromzufuhr, Netzwerkzufuhr, etc.) werden doppelt oder sogar häufiger vorgehalten. Dies ist vergleichbar mit der Bauweise eines Mainframes, bei welchem ebenfalls alle Teile doppelt vorhanden sind, sodass beim Ausfall eines Teils dieses im laufenden Betrieb ohne Downtime oder Datenverlust ausgetauscht werden kann. Außerdem gibt es spezielle Speichertechniken für die Ablage von Daten. Hier ist besonders die Redundant Array of Indepent Disks (RAID)-Speichertechnik zu nennen, welche dafür sorgt, dass Daten gleichzeitig auf verschiedene Festplatten

[90] Vgl. SAP HANA (2012), S. 12
[91] Vgl. Bendelac, C. (2014)

geschrieben werden („mirroring“) sowie die Mitgabe von zusätzlichen Bits, welche das Erkennen und Beheben von Fehlern ermöglicht („parity“).[92]

- **Netzwerk-Redundanz**

  Für die Netzwerk-Redundanz wird ein SAP HANA-System mit zwei unabhängigen Internet-Zugängen verbunden, um bei Ausfall des einen (Anbieter A) auf den anderen (Anbieter B) zurückgreifen zu können. Die Netzwerk-Architektur wird meist durch das sogenannte Spanning Tree Protocol sowie das Border Gateway Protocol umgesetzt.[93]

- **Datencenter-Redundanz**

  Redundante Datencenter werden installiert, um die Vorteile der Redundanz von Hardware und Netzwerk zusammenzuführen. Die einzelnen Komponenten (z.B. Kühlung, Stromzufuhr, Netzwerk, etc.) liegen demnach also doppelt vor.

  Einige Unternehmen betreiben sogar vollständig redundante Datencenter an verschiedenen Orten, um bei Naturkatastrophen und anderen unvorhergesehenen Ereignissen abgesichert zu sein.[94]

Bevor entsprechende Desaster-Wiederherstellungs-Maßnahmen eingeleitet werden können, müssen Daten natürlich auch entsprechend gespeichert werden. Obwohl SAP HANA eine In-Memory Datenbank ist, loggt und speichert sie dennoch jede einzelne Transaktion, damit diese auch persistent ist bzw. bleibt (ACID-Prinzip). Hierfür nutzt SAP HANA zwei verschiedene Wege: jede einzelne Änderung an der Datenbank wird in den Redo Log geschrieben. Der Redo Log dient dazu, bei Systemausfall die letzten Transaktionen noch einmal durchlaufen zu lassen, um den aktuellen Status der Datenbank zu rekonstruieren. Der Redo Log wird durch sogenannte Savepoints unterbrochen, welche in gewissen Zeitabständen (meist max. 5min) alle Änderungen fest auf Festplatten schreiben. Dabei werden ältere Savepoints (und damit auch der log) von neueren Savepoints überschrieben (aktueller Stand der Datenbank zählt). Savepoints können jedoch ebenfalls gespeichert werden, um einen gewissen Datenbankstand zu einem genauen Zeitpunkt festzuhalten. In diesem Fall spricht man von Snapshots. In der

[92] Vgl. Bendelac, C. (2014), S. 4
[93] Vgl. Bendelac, C. (2014), S. 4
[94] Vgl. ebenda

folgenden Grafik wird die Thematik von Backups bei SAP HANA noch einmal grafisch dargestellt.[95]

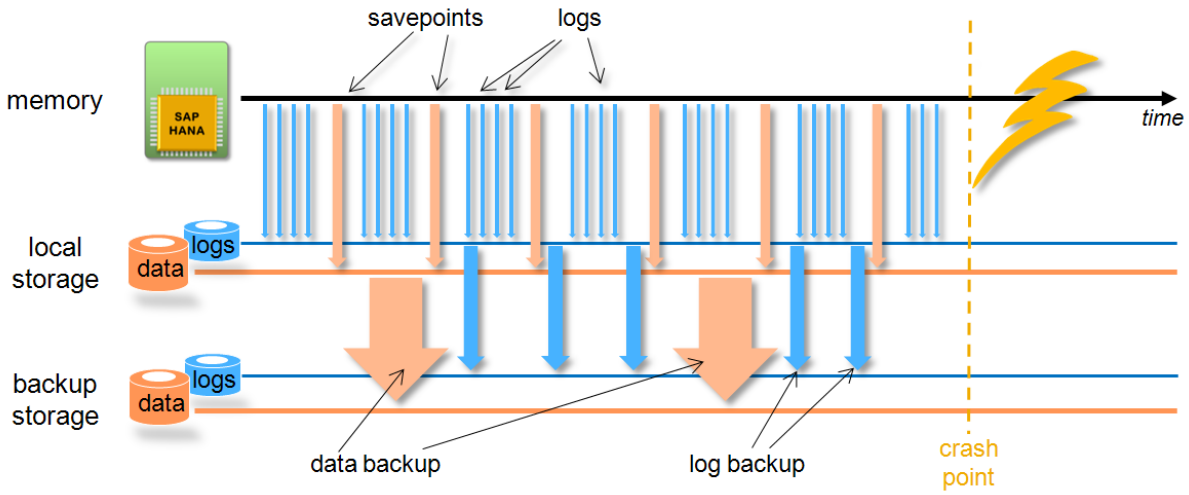


Abbildung 16: Sichere Backups bei SAP HANA[96]

Werden Backups ausschließlich lokal gespeichert, besteht weiterhin die Gefahr eines Datenverlustes bei Systemausfällen. Daher werden Backups oft in Abständen bspw. per LKW in andere Lokationen gebracht.[97] Es gibt jedoch auch Möglichkeiten, dies zu umgehen und dennoch eine sichere Speicherung der Backups im Desaster-Fall zu garantieren. In diesem Fall spricht man von Replikation. Es gibt mehrere Arten von Replikationen in diesem Zusammenhang, von denen zwei im Folgenden kurz erläutert werden: Storage und System Replikation.

- **Storage Replikation**

  Die Replikation des Speichers hat den Vorteil, dass alle Transaktionen sofort in einer anderen Lokation sicher gespeichert sind und somit bei einem Systemausfall des HANA-Systems das Recovery ohne Datenverlust durchgeführt werden kann. Wichtig ist hierbei lediglich, dass die zweite Speicherlokation innerhalb von 100 km des eigentlichen Systems liegt und eine Internetverbindung mit sehr hoher Bandbreite zuverlässig verfügbar ist, da sonst die sogenannte synchrone Speicherreplikation nicht durchgeführt werden kann. Spezielle Konfigurationen dieser Replikation limitieren die

[95] Vgl. Bendelac, C. (2014), S. 5
[96] Vgl. Bendelac, C. (2014), S. 6
[97] Vgl. ebenda

Gültigkeit von Transaktionen sogar darauf, dass diese nur als vollständig bezeichnet werden, wenn sie sowohl lokal als auch remote gespeichert wurden.[98]

- **System Replikation**

  Die Replikation eines vollständigen Systems bedeutet, dass zu einem primären HANA-System noch ein zweites parallel geschaltet wird. Dieses zweite System wird mit dem primären synchronisiert, also alle Komponenten mit den primären Komponenten verbunden. Sobald das sekundäre System den Betrieb aufnimmt (Live Replication Mode), erhält es die notwendigen Snapshots vom primären System, um auf den aktuellen Stand der Datenbank zu kommen. Ähnlich zur Storage Replikation muss dieses System in einem 100 km-Umkreis stehen und über eine entsprechende Netzwerkverbindung verfügen. Im Live Replication Mode sind drei verschiedene Synchronisationsmodi möglich: synchron, synchron-in-memory und asynchron.

  - Bei der synchronen System-Replikation werden Transaktionslogs an das sekundäre System geschickt und müssen dort erst persistent verarbeitet werden, bevor die Operation im primären System fortgesetzt wird.
  - Bei der synchronen-in-memory System-Replikation werden Transaktionslogs an das sekundäre System geschickt, müssen jedoch nicht persistent verarbeitet werden, sondern lediglich im Arbeitsspeicher vorliegen, bevor die Operation im primären System fortgesetzt wird. Dadurch wird die Wartezeit verringert.
  - Bei der asynchronen System-Replikation werden Transaktionslogs an das sekundäre System geschickt, jedoch wird sofort mit der Operation fortgefahren. Dies ist vor allem sinnvoll, wenn sich beide Systeme sehr weit auseinander befinden.

Bei Systemausfall wird dann auf das sekundäre System umgeschaltet. Nachdem die entsprechenden Snapshots geladen und die neuen Transaktionslogs erneut durchgespielt wurden, lädt das System noch Tabellen nach und ist dann bereit für Anfragen.[99]

[98] Vgl. Bendelac, C. (2014), S. 6
[99] Vgl. Bendelac, C. (2014), S. 7 ff.

Bei Fehlern wie bspw. Softwarefehlern oder aber bewusstem Eingreifen von Systemadministratoren gibt es zwei Recovery-Optionen:

- **Service Auto-Restart**

  Wird ein HANA-Service (Index Server, Name Server, etc.) automatisch oder manuell abgeschaltet, wird dieser Service von der „watchdog“ Funktion erneut gestartet. Dies führt zu keinem Datenverlust, allerdings ist mit einiger Downtime zu rechnen, während der Service hochgefahren wird.[100]

- **Host Auto-Failover**

  Diese Variante ist ähnlich zur System-Replikation, denn auch hier wird ein komplettes HANA-System parallel zu den aktiven Systemen geschaltet. Allerdings werden keine Daten oder Logs an dieses passive System übertragen. Auch Anfragen beantwortet es nicht. Es bleibt im Standby-Modus, bis es aufgrund eines Ausfalls benötigt wird. Dann werden sofort alle Anfragen und Datenflüsse des ausgefallenen Systems an das Standby-System umgeleitet. Besonders wichtig ist, dass alle Verbindungen korrekt umgeleitet werden und das Primärsystem auch tatsächlich ausgefallen ist, da sonst möglicherweise beide System parallel arbeiten, was zu schweren Problemen führen könnte.[101]

Um die Terminologien der Persistenz bei SAP HANA noch einmal in Zusammenhang zu bringen, wird anhand von Abbildung 17 der Zyklus eines funktionierenden Systems, über den Systemfehler bis hin zur Wiederherstellung illustriert.

Zunächst läuft das HANA-System im Normalbetrieb. Dabei werden regelmäßige Backups gemacht, sodass die Daten entsprechend gesichert sind. Hier kommt nun der Begriff Recovery Period Objective (RPO) hinzu. Dieser gibt die maximale Zeitspanne zwischen letztem Backup und möglichem Systemfehler an, bei der Daten verloren gehen dürfen. Ein RPO von fünf Minuten bedeutet demnach, dass Backups alle fünf Minuten gemacht werden müssen, um nicht Daten eines längeren Zeitraums zu verlieren. Nachdem der Fehler nun erkannt wurde, beginnt das Recovery Time Objective (RTO), welches festlegt, wie lange es nach einem Systemfehler maximal dauern darf, bis das System wieder auf den Zustand genau vor dem Systemfehler hergestellt ist.[102]

[100] Vgl. Bendelac, C. (2014), S. 9 ff.
[101] Vgl. ebenda
[102] Vgl. Bendelac, C. (2014), S. 5

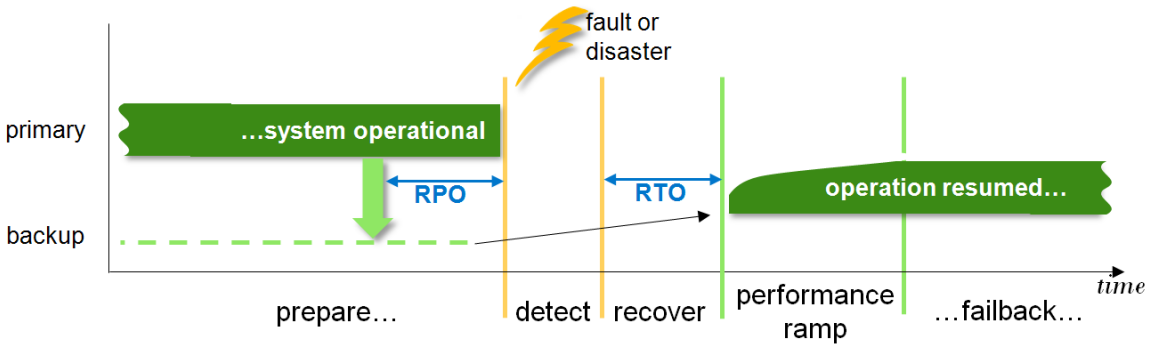


Abbildung 17: SAP HANA Backup and Recovery Scheme[103]

[103] Enthalten in: Bendelac, C. (2014), S. 5

4 In-Memory Datenbanksysteme Marktübersicht (A. Croll, T. Dietz, P. Koch, T. Soetebeer)

Durch die Entwicklung von Datenbanksystemen wie SAP HANA hat sich der Datenbankmarkt stark neu strukturiert. Noch vor wenigen Jahren war dieser Markt recht übersichtlich, es gab eine überschaubare Anzahl an Datenbanksystemen, die sich wiederum aufteilten in universell einsetzbare relationale Datenbanken, Spezial-Datenbanken wie zum Beispiel Objektdatenbanken oder Key-Value-Store Datenbanken, und Data Warehouses. Die eindimensionale Welt von Unternehmensdatenbanken hat sich jedoch bedeutend verändert. Die Gründe hierfür sind vielfältig, lassen sich aber auf einige mit besonderer Relevanz herunterbrechen. Die Mehrdimensionierung des Datenbankmarkts wurde durch das Aufkommen von Open Source merklich vorangetrieben, hier entstanden mit MySQL und HBase leistungsfähige Datenbanksysteme, die mit ihren Lizenzmodellen und Entwicklungspotentialen den Marktführern Konkurrenz machen konnten. Weiterhin wurde die Nachfrage nach Webcontent und Analysemöglichkeiten größer, es entstanden skalierbare Lösungen für das 'Internet Zeitalter'. Gleichzeitig änderten sich auch die Infrastrukturgegebenheiten, was am Beispiel der niedrigen Preise für CPUs und Speicher deutlich wird.[104]

Diese Faktoren begünstigen die Entwicklung von IMDBS und strukturieren den aktuellen Markt deutlich. Im Bereich IMDBS ist eine deutlich vielfältige Forschungs- und Entwicklungsaktivität charakteristisch, was allerdings nicht bedeutet, dass der Markt leicht überschaubar ist. Es gibt bereits über 30 erwähnenswerte IMDBS, die zum Teil kommerzielle, zum Teil Open Source Lösungen darstellen. Die relevanten Systeme gliedern den Markt aktuell grob wie folgt.

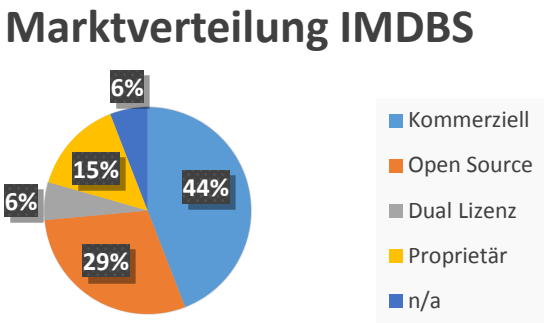


Abbildung 18: Marktverteilung IMDBs[105]

Um einen Einblick in die Segmente Open Source und Kommerziell zu ermöglichen, wird im Folgenden eine Einführung in eine Auswahl der elaboriertesten, am Markt verfügbaren

[104] Vgl. Quinn, E. (2013)

[105] Auswahl von 34 In-Memory Datenbanken die sich nach Recherche als marktrelevant herausgestellt haben.

Systeme, neben SAP HANA, gegeben. Diese Auswahl wird nach Abwägung der Marktanteile und technischen Leistungsfähigkeit getroffen. Hierbei werden zwei Vertreter der Open Source Kategorie, SQLite und HSQLDB, und zwei weitere Vertreter der kommerziellen Systeme, Oracle TimesTen und IBM SolidDB vorgestellt.

SQLite ist, wie bereits erwähnt, eine Open Source IMDBS Lösung und eine der am weitesten verbreiteten SQL Datenbanksysteme weltweit.[106] Es richtet sich mit den Potentialen vor allem im Bereich der Datenbankgröße überwiegend an kleine bis mittelgroße IT Landschaften. Auch der Kosten-Faktor spielt natürlich eine Rolle. Da SQLite unter einer Public-Domain-Lizenz läuft, kann es von Anwendern kostenlos bezogen werden. Die Datenbank ist zusätzlich nur ein paar hundert Kilobyte groß, da sie keine eigenen Serverprozesse enthält. Daraus entwickelt sich ein weiterer Vorteil für den jeweiligen Anwender, da weder Installation noch Konfiguration benötigt wird.[107] Die Konsequenz daraus ist, dass SQLite ausschließlich im Arbeitsspeicher existiert. Dies bedeutet, die Existenz der Datenbank endet mit der Verbindung zur DB, der implementierende Entwickler muss folglich Maßnahmen zur Dauerhaftigkeit und Sicherung der Daten unternehmen.[108] Die Lösung bietet trotzdem vielfältige Einsatzmöglichkeiten, so zum Beispiel für Applikationen, die Cross-Plattform-Formate speichern oder auch als integrierte Datenbank für kleine Websites oder mobile Endgeräte. Die Möglichkeiten der parallelen Nutzung von SQLite sind recht eingeschränkt, trotzdem wird es als Persistenz-Schicht für beispielsweise Firefox und Skype eingesetzt.[109]

Ein weiterer Open-Source Vertreter ist HSQLDB, die auch im kommerziellen Bereich weite Verbreitung erfährt. Auch HSQLDB ist speicherschonend und nimmt mit 1,4 Megabyte unwesentlich mehr Raum ein als SQLite. Die IMDBS Lösung ist komplett in Java geschrieben und ist somit auf allen Java-fähigen Plattformen einsetzbar. Der offene Java-Code ist auch Grund für die relativ große Anzahl an unterschiedlichen Anbietern von HSQLDB. Wie SQLite findet die Datenbank häufig als Persistenz-Schicht Einsatz wie zum Beispiel bei "OpenOffice" und "Mathematica". Der Nutzer hat die Wahl, die Tabellen als reine In-Memory-Tabellen anzulegen, persistente und Disk-basierende Speicherung ist jedoch ebenso möglich.[110] Auf Grund des recht geringen Funktionsumfangs ist auch HSQLDB in der Regel keine Option für große Unternehmensinfrastrukturen.

Für große IT Landschaften wird in den meisten Fällen ein komplexer Funktionskatalog inklusive Wartung benötigt, den Open Source Hersteller nicht zur Genüge erfüllen können. Leistungsfähigkeit, Skalierbarkeit, hohe Verfügbarkeit und nahtlose Implementierung sind

[106] Vgl. Choinyambuu, S. (2010)
[107] Vgl. Ott, J./Stirnimann, R. (2012)
[108] Vgl. Choinyambuu, S. (2010)
[109] Vgl. Ott, J./Stirnimann, R. (2012)
[110] Vgl. Ott, J./Stirnimann, R. (2012)

Anforderungen, denen High-Tech Unternehmen gegenüber stehen. Oracle sieht sich mit Oracle TimesTen dieser Herausforderung gewachsen.

TimesTen ist eine Akquisition von Oracle aus dem Jahre 2005, an der Oracle nach wie vor forscht und entwickelt. Es handelt sich hierbei um eine kommerzielle IMDBS Lösung, die für Windows, Linux und Unix Plattformen verfügbar ist. TimesTen kann in zwei große Funktionsbereiche untergliedert werden: Zum Einen in den Betrieb als Standalone-Datenbank, zum Anderen als Cache für traditionelle Oracle Tabellen. Durch vollständiges oder teilweises Laden in die TimesTen "Cache Groups" können Prozesse dafür sorgen, dass Daten synchron oder asynchron in eine Oracle Datenbank geschrieben werden. Der offensichtliche Vorteil hieraus ist ein beschleunigter Zugriff auf die jeweils relevanten Daten selbst bei hoher Anfragendichte. Weiterhin können zusätzliche Daten automatisch aus relationalen DBMS nachgeladen werden, hierbei werden aktuell jedoch nur Oracle Plattformen unterstützt. Außerdem ist die Portierung von relationalen DBMS Applikationen in ein solches TimesTen Umfeld mit nicht zu unterschätzendem Anpassungsaufwand verbunden. Oracles IMDBS zeichnet sich jedoch vor allem durch eine hohe Anzahl von Replizierungsverfahren aus, die bei Ausfällen eine unterbrechungsfreie Nutzung ermöglichen und ist somit vor allem bei der Verarbeitung von kritischen und aktuellen Unternehmensdaten zu finden.[111]

Ähnliche Einsatzgebiete wie Oracle TimesTen finden sich bei der IMDBS Lösung von IBM. IBM SolidDB ist das Ergebnis aus IBMs Übernahme von SolidDB im Jahr 2008. Das System entwickelt seine Dauerhaftigkeit, indem es zwei separate aber synchrone Kopien der Datenbank führt. Zusätzlich verwendet SolidDB eine permanente Log Datei, die auf der Disk gespeichert wird. So löst IBM das Problem des flüchtigen Hauptspeichers und verhindert somit konkret den Datenverlust. Laut IBM kann im Falle eines Absturzes eine Wiederherstellung über die zusätzliche Datenbank ohne jeglichen Datenverlust in wenigen Sekunden vollzogen werden.[112] SolidDB ist für die IBM Power7 Prozessoren optimiert und kann sowohl auf AIX als auch auf Linux, Windows und Solaris Plattformen betrieben werden. SolidDB macht neuen Nutzern die Umgewöhnung vergleichsweise leicht. Grund dafür ist die Ähnlichkeit zu bekannten DB2-Instanzen, was sich zum Beispiel in der Verwendung des „SolidDB Universal Cache“, welcher auch als Cache von Tabellen aus traditionellen relationalen Datenbanken bekannt ist, zeigt. Im direkten Vergleich zu Oracle besticht SolidDB vor allem durch die Anzahl unterstützter Plattformen.[113]

Deutlich wird, dass der Markt von kommerziellen Systemen dominiert wird und somit auch mit weiteren Investitionen in Entwicklungen und Innovationen von In-Memory Lösungen zu

[111] Vgl. Ott, J./Stirnimann, R. (2012)
[112] Vgl. IBM Germany (Hrsg.) (o.J.)
[113] Vgl. Ott, J./Stirnimann, R. (2012)

rechnen ist. Zu erwähnen ist allerdings auch der hohe Anteil an Open Source Lösungen, die vor allem für kleine und mittelständische Unternehmen interessante Potentiale aufweisen. Open Source Lösungen wie HSQLDB und SQLite finden daher auch, wie bereits erwähnt, in bekannten Anwendungen, wie zum Beispiel "OpenOffice", "Firefox" und "Skype", Einsatz.[114]

Der Hintergrund dieser Marktübersicht war es, einen Überblick über weitere Produkte am In-Memory Markt zu geben. Nicht Ziel der Arbeit an dieser Stelle war es, eine ebenso ausführliche Betrachtung zu treffen, wie diese bereits für SAP HANA durchgeführt wurde. Das liegt vordergründig daran, dass es in der Arbeit um die Erläuterung und Umsetzung von IMDB Merkmalen ging. Des Weiteren ist, wie bereits erwähnt, zu den anderen Produkten nicht ausreichend Literatur für eine komplette Ausführung verfügbar. Abschließend bleibt zu erwähnen, dass eine Komplettbetrachtung mehrerer Produkte nicht in den zeitlichen Rahmen dieser Untersuchung gepasst hätte.

[114] Vgl. Ott, J./Stirnimann, R. (2012)

5 Vergleich IMDBS vs. RDBMS (A. Croll, T. Dietz, P. Koch, T. Soetebeer)

Die wesentlichen Merkmale von In-Memory Datenbanken wurden bereits erläutert und das Potenzial für die Verbesserung der Performance dieser Datenbanken aufgezeigt. Während die In-Memory Datenbanktechnologie jedoch im Hinblick auf ihren technischen Reifegrad immer noch eher am Anfang steht, sind relationale Datenbanken schon deutlich länger etabliert und konnten sich als zuverlässige Alternative bezüglich der Backup- und Recovery-Verfahren am Markt beweisen. In diesem Punkt wirft die beschriebene Hauptspeichervariante teilweise noch diverse Fragen auf. Um die Chancen und Risiken von In-Memory Datenbanken zu verdeutlichen, soll im Folgenden noch einmal ein kurzer Vergleich zur relationalen Datenbanktechnik durchgeführt werden, der vor allem die Unterschiede hinsichtlich der beschriebenen Merkmale sowie die damit verbundenen Auswirkungen genauer betrachtet.

Zunächst besteht ein entscheidender Unterschied in dem Speicherort der Daten. Da bei relationalen Datenbanken die Daten grundsätzlich immer zunächst von der Festplatte gelesen und in den Arbeitsspeicher geladen werden müssen, entsteht hier ein Flaschenhals hinsichtlich der Bearbeitungsgeschwindigkeit. Bei In-Memory befinden sich die Daten bereits im Arbeitsspeicher, wodurch die Antwortzeiten deutlich verkürzt werden können. Abbildung 18 zeigt in diesem Zusammenhang, dass relationale Datenbanken erst bei Bedarf zur Laufzeit die gewünschten „Data Pages“ von der Festplatte in den Buffer Pool laden. IMDBs hingegen laden bereits beim Start der Datenbank sämtliche Daten in den Speicher. Der „Query Optimizer“ kann dann direkt auf alle Daten zugreifen, ohne kostspielige I/O-Operationen (Input/Output) zu erzeugen.

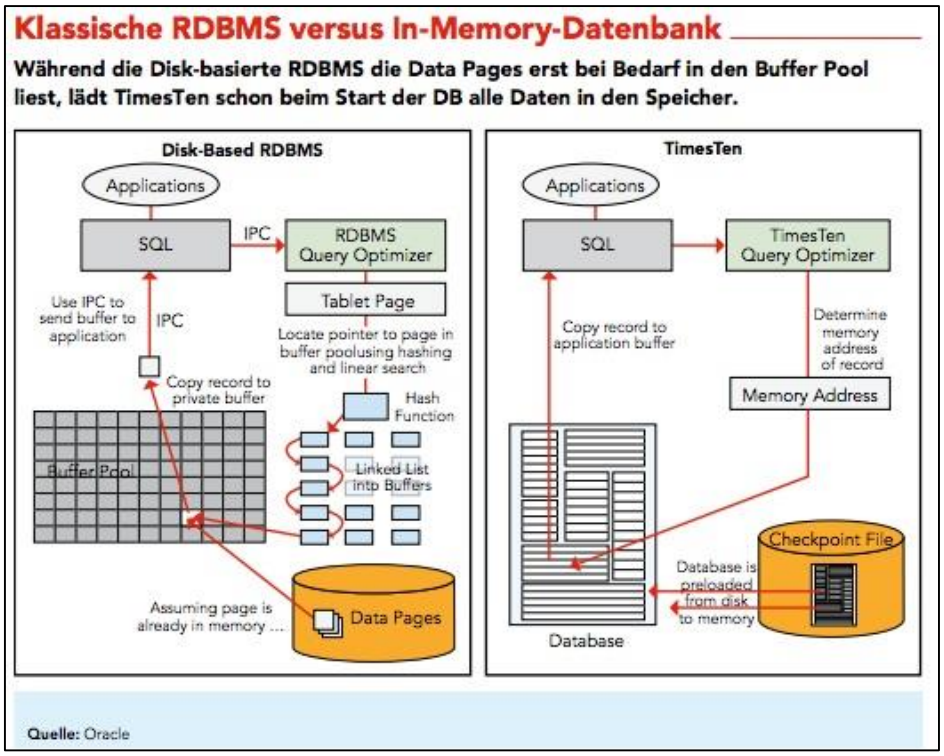


Abbildung 19: Klassische RDBMS versus In-Memory-Datenbank[115]

An dieser Stelle zeigt sich eine der Schwächen von IMDBs: Der als Speichermedium verwendete Hauptspeicher ist im Gegensatz zu Festplatten heute noch deutlich teurer – zwar sind die Preise bereits enorm gesunken, diese haben aber noch nicht das Niveau von Festplatten erreicht.

Eine deutliche Differenz zeigt sich zudem bei der Art der Speicherung von Datensätzen. Während eine Mehrzahl der relationalen Datenbanken die Daten ausschließlich zeilenorientiert speichern und folglich bei Auswertungen über viele Datensätze, aber nur wenige Attribute deutlich an Performance verlieren, können In-Memory Datenbanken eine Kombination aus zeilen- und spaltenorientierter Datenhaltung verwenden und somit nicht nur je nach Art der Abfrage und Struktur der Daten die Vorteile der beiden Varianten bestmöglich ausnutzen, sondern auch gegenüber relationalen Datenbanken eine entscheidende Beschleunigung der Zugriffsgeschwindigkeit in gewissen Anwendungsbereichen aufweisen.

Aufgrund der Tatsache, dass in relationalen Datenbanken, anders als bei In-Memory, die Zeilendaten nicht immer denselben Datentyp haben, sind Kompressionsverfahren nur begrenzt einsetzbar. Diese versprechen nur dann einen entsprechenden Performancevorteil, wenn eine Ähnlichkeit benachbarter Daten gegeben ist. Durch die spaltenorientierte Speicherung im Falle von In-Memory Datenbanken ist diese Voraussetzung erfüllt und als

[115] Enthalten in: WinfWiki (2012)

Folge lässt sich eine deutliche Speicherplatzersparnis erreichen. Damit grenzt sich auch hier die In-Memory Datenbanktechnologie von relationalen Datenbanken ab, welche zur Optimierung der Performance lediglich auf die Indexierung der Zeilendaten zurückgreifen.

Ein weiterer Unterschied besteht im Umgang mit der Parallelisierung der Datenbank. Wie bereits beschrieben, ergibt sich durch die Parallelisierung und die damit verbundene Partionisierung bei In-Memory Datenbanken ein enormes Optimierungspotenzial. Daten können nicht nur über mehrere Kerne, sondern über eine Vielzahl an verschiedenen Maschinen parallel prozessiert werden. Darüber hinaus besteht die Möglichkeit, Daten auf mehreren Servern sowie Applikationen zu verteilen. Bei relationalen Datenbanken ist zwar ebenfalls eine Parallelisierung auf mehreren Kernen möglich, wie Abbildung 19 jedoch zeigt, kann der Durchsatz nur selten linear mit der Prozessorenanzahl gesteigert werden, so dass ab einem bestimmten Punkt eine weitere Erhöhung der Prozessorenanzahl eher negative Auswirkungen auf die Antwortzeit hat und die Effizienz der Parallelisierung entsprechend eingeschränkt wird.

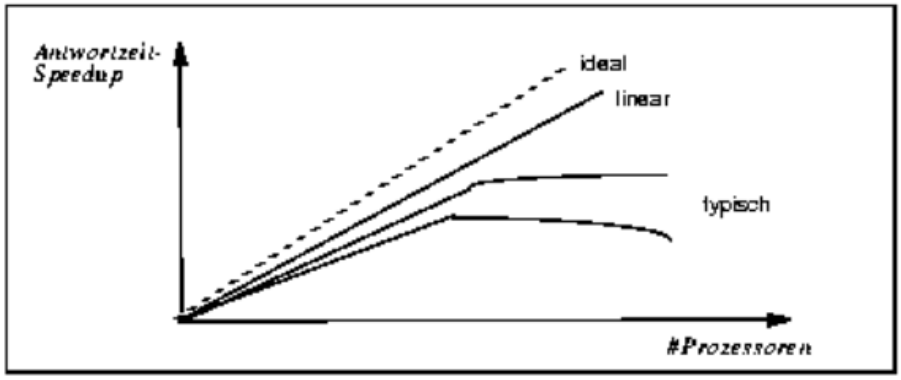


Abbildung 20: Idealer, linearer und typischer Antwortzeit-Speedup[116]

Dennoch stellt die Persistenz eine wichtige Herausforderung für sämtliche Datenbanktechniken dar. Bei relationalen Datenbanken ist dies eher unproblematisch, da Daten direkt auf der Festplatte gespeichert und verarbeitet werden und entsprechend auch einen Zusammenbruch des Systems unbeschadet überstehen. Da es sich beim Hauptspeicher jedoch um flüchtigen Speicher handelt und folglich die dauerhafte Speicherung als wichtiger Bestandteil des ACID Prinzips nicht ohne weiteres erfüllt ist, müssen bei In-Memory Datenbanken alternative Lösungen zur Erreichung von Persistenz bedacht werden. Diese behelfen sich mit dem Ablegen von Log-Einträgen und Änderungen auf einem nichtflüchtigen Speicher, wodurch im Falle eines Zusammenbruchs wie in Abbildung 20 die Modifizierungen

[116] Enthalten in: Rahm, E. (1996)

nachvollzogen werden können und der Stand des letzten Snapshots wiederhergestellt werden kann.

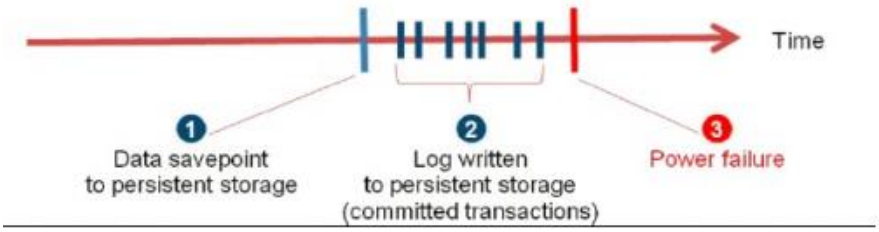


Abbildung 21: SAP HANA Persistent Storage[117]

Abschließend sollen noch einmal die Einsatzmöglichkeiten der beiden Datenbanktechnologien anhand verschiedener Einsatzszenarien aufgezeigt werden. Nicht nur in Form von Geschäftssoftware ist In-Memory bereits weit verbreitet – auch beispielsweise in der Medizin entstehen immer neue Geschäftsfelder. Basierend auf dieser Technologie lassen sich mit Hilfe des SAP Oncolyzers in kürzester Zeit enorme Datenmengen analysieren. Diese haben häufig einen eher statischen Charakter und sind nicht zwingend auf dem neuesten Stand. Die durchgeführten Abfragen sind primär analyseorientiert und werden in unregelmäßigen Abständen von einer kleinen Anzahl an Benutzern durchgeführt, um strategische Entscheidungen zu unterstützen.[118] Im Fall von SAP Oncolyzers werden Daten aus einer Vielzahl von Krankenhausinformationssystemen für den Arzt in der translationalen Tumorforschung kombiniert, welche schnell zu einer Menge im Terabyte-Bereich anwachsen.[119] Über die Methoden der In-Memory-Technologie können diese in Echtzeit analysiert und für die IT-gestützte Behandlung des Krebspatienten sowie für andere OLAP-Analysen genutzt werden.[120]

Dagegen spielen relationale Datenbanken auch heute noch ihre Stärke dort aus, wo vorrangig Auswertungen über einzelne Datensätze gemacht werden. Es handelt sich grundsätzlich um aktuelle Daten mit dynamischem Charakter. Relationale Datenbanken sind im Gegensatz zu In-Memory eher transaktionsorientiert, wobei sie auf regelmäßiger Basis von einer hohen Anzahl an Benutzern zur Unterstützung von tagtäglichen Entscheidungen genutzt werden. Während sich In-Memory-Datenbanken bei lesenden Abfragen als vorteilhaft erweisen, spielen relationale Datenbanken bei schreibenden Vorgängen ihre Stärken aus.[121] Beispielhaft

[117] Enthalten in: HANA Tutorials (2011)
[118] Vgl. Datawarehouse4u (2010)
[119] Vgl. Hasso-Plattner-Institut (2014)
[120] Vgl. New Trends in IT (2013)
[121] Vgl. Datawarehouse4u (2010)

sind in diesem Zusammenhang Enterprise-Resource-Planning-Systeme (ERP) zu nennen, in denen lediglich Stammdaten verwaltet werden. Besonders in geschäftskritischen Bereichen wie Banken und Versicherungen liegt der Fokus auf der Datensicherheit und der Verhinderung von Datenverlusten, wo die traditionellen Datenbanken derzeit schlichtweg noch zuverlässiger und vertrauenswürdiger aus Sicht vieler Kunden sind.

6 Alternativen zu In-Memory-DB und ein Ausblick in die Zukunft (A. Croll, T. Dietz, P. Koch, T. Soetebeer)

6.1 Alternativen zur In-Memory-Technologie

Nach der theoretischen Einführung in die Thematik der In-Memory-Datenbanken und der praktischen Anwendungen anhand konkreter Beispieldatenbanken sollen nun alternative marktrelevante Lösungen zur In-Memory-Technologie aufgezeigt und bewertet werden, welche das gleiche Ziel, also eine Verbesserung der Zugriffsgeschwindigkeit, mit Hilfe einer abgewandelten Umsetzung verfolgen.

Vor allem der Einsatz von Solid State Drives (SSD) kann hier als möglicher Ansatz betrachtet werden, da die Beschaffungskosten für SSDs kürzlich immer weiter sinken. Bei SSDs handelt es sich um einen Kompromiss zwischen der Datenspeicherung auf klassischen Festplatten und der Nutzung von In-Memory-Technologie: Hier geht eine persistente Speicherung auf Festplatten mit einer deutlich verbesserten Performance gegenüber herkömmlichen Festplatten einher.[122] SSDs können überdies mit großen Universal Serial Bus (USB)-Sticks verglichen werden. Die Daten werden auf Flash-Speichern abgelegt und es werden keine mechanischen Bauteile wie z.B. Disks oder Lese- und Schreibköpfe mehr benötigt.[123] Dies bringt jedoch den Nachteil mit sich, dass jeder Speicherblock nicht unendlich häufig beschrieben werden kann, so dass diese Technologie für Anwendungsfälle mit einer hohen Anzahl an Datenbankänderungen nur bedingt sinnvoll ist.[124]

[122] Vgl. Funk, M./Marinkov, B./Paar, M. (2012)
[123] Vgl. OSBI Blog (2011)
[124] Vgl. Funk, M./Marinkov, B./Paar, M. (2012)

Die folgende Tabelle liefert einen schnellen Überblick über den Vergleich von In-Memory-Datenbanken zu SSDs:

	SSD	**In-Memory**
Implementierung	Einfach	Schwierig
Kosten	Niedrig	Hoch
Verlässlichkeit	Sehr gut	Riskant
Verfügbarkeit	Sehr gut	Gut
Skalierbarkeit	Sehr gut	Gut
Mehrbenutzbarkeit	Viele Server	Ein Server

Tabelle 1: In-Memory-Datenbanken vs. Solid State Disks[125]

Anhand dieser Tabelle wird deutlich, dass der Einsatz von Solid State Disks entscheidende Vorteile im Vergleich zur In-Memory-Technologie mit sich bringt. Während für die Implementierung von SSDs keinerlei Anpassungen hinsichtlich des Codes gemacht werden muss, erfordert die Installation von In-Memory eine umfangreiche Modifikation der betroffenen Anwendung, welche nicht nur einen hohen Kosten- und Zeitaufwand bedingt, sondern auch das Risiko deutlich steigert.[126] Neben den Anschaffungskosten spielen bei In-Memory-Datenbanken vor allem die Kosten für die Speicherung auf Servern und die Erweiterung der Speicherkapazität mit Hilfe von zusätzlichen Prozessoren eine entscheidende Rolle. Im Umgang mit der Verlässlichkeit enthalten SSDs verschiedene Features wie Error Checking and Correction (ECC), welche der Fehleridentifikation und –korrektur dienen, indem vor der Datenspeicherung oder –übertragung den Daten eine zusätzliche Redundanz durch zusätzliche Bits beigefügt werden, über die auf der Zielseite die Fehler und ihre Positionen einfacher bestimmt werden können.[127] Zudem ermöglicht die Nutzung von SSDs, die Speicher- und Serverkomponenten einer Anwendung voneinander zu entkoppeln, so dass der Speicher nicht von Serverproblemen beeinträchtigt wird.[128] Darüber hinaus ist auch die Skalierbarkeit

[125] Vgl. Hutsell, W./Bowen, J. (o.J.)
[126] Vgl. Hutsell, W./Bowen, J. (o.J.)
[127] Vgl. Stamp, N. (2013)
[128] Vgl. Hutsell, W./Bowen, J. (o.J.)

bei SSDs gegeben, da zusätzlicher Speicher problemlos zu dem bestehenden System hinzugefügt werden kann. Während eine In-Memory-Datenbank nur von einem Server genutzt werden kann, können SSDs zwischen mehreren Servern zeitgleich geteilt werden.

Abbildung 21 verdeutlicht noch einmal zusammenfassend, inwiefern SSDs einen Mittelweg im Vergleich zu klassischen Datenbanken und der In-Memory-Technik anhand der Kriterien Größe, Preis und Performance darstellt:

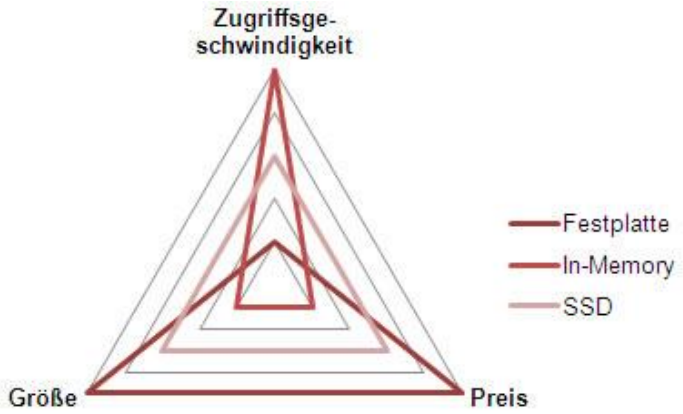


Abbildung 22: Vergleich Festplatte, In-Memory und SSD[129]

Neben den Solid State Drives kommt auch die Caché-Datenbank als Alternative zur In-Memory-Technik in Frage. Diese von der Firma InterSystems entwickelte postrelationale Lösung zeichnet sich vor allem dadurch aus, dass die Performance nicht nur der von In-Memory-Datenbanken entspricht, sondern zudem auch weitere Anforderungen an das Verhalten einer Datenbank erfüllt sind: Während in In-Memory-Datenbanken die Daten in flüchtigen Arbeitsspeichern abgelegt werden, speichert die Caché-Datenbank die historischen und transaktionalen Daten mit Hilfe eines Permanentspeichers persistent, so dass diese im Falle eines Absturz des Systems nicht verloren gehen. Des Weiteren bietet das integrierte Datenbank-Mirroring im Falle eines Ausfalls den automatischen Serverwechsel zwischen zwei Caché-Systemen.[130] Caché greift dabei auf eine spezielle Form der Datenverwaltung zurück: Die sogenannte Unified Data Architecture wie in Abbildung 22 ermöglicht es, über viele verschiedene Front-Ends auf die Datenbestände zuzugreifen, und diese für unterschiedliche Auswertungen zur Verfügung zu stellen.[131] Als Folge ist der Entwickler beim Datenzugriff völlig unabhängig vom Tool und dem Betriebssystem.

[129] Enthalten in: OSBI Blog (2011)
[130] Vgl. heiseDeveloper (2010)
[131] Vgl. Röhrig, B. (2001)

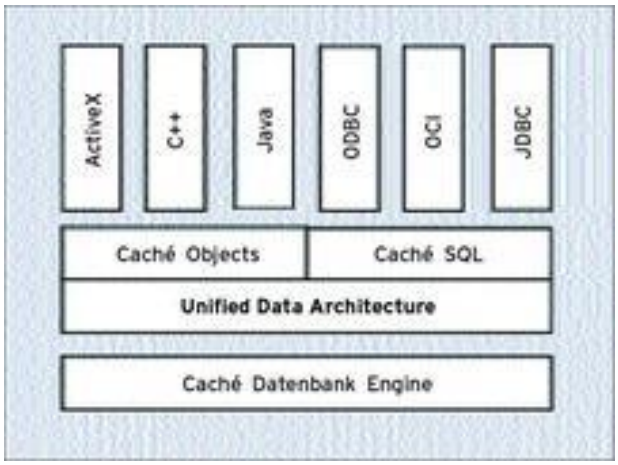


Abbildung 23: Unified Data Architecture[132]

Zudem nutzt Caché im Gegensatz zu herkömmlichen Datenbank anstelle von Spalten und Zeilen multidimensionale Datenfelder. Daten werden in der vom Architekten bestimmten Form gespeichert. Dieselben Strukturen, die der In-Memory-Cache nutzt, befinden sich auch auf der Festplatte. Folglich kann Caché sehr schnell auf Festplattendaten zugreifen.[133] Die verwendete Architektur bringt weiter den Vorteil mit sich, dass auf die Daten in Form multidimensionaler Arrays simultan per SQL oder als Objekt zugegriffen werden kann – und das ohne das ansonsten benötigte „Mapping“ in relationale Tabellen.[134] Ebenso ist die Größe des Datensatzes in Caché nicht mehr von der Menge der verfügbaren RAMs abhängig – liegen Informationen also nicht im lokalen Cache vor, werden sie entweder aus einem Remote-Cache oder nahtlos von der Festplatte bezogen. Somit können Petabytes an Daten problemlos gehandhabt werden.[135] Letztlich bietet diese Lösung noch eine hohe Skalierbarkeit durch den Einsatz vom InterSystems Enterprise Caché Protocol (ECP), dass Computer in einem verteilten System die Datenbanken auf den einzelnen Computern gleichzeitig nutzen können.

Abschließend stellen auch hybride Datenbanktypen als Abwandlung von der In-Memory-Datenbank eine mögliche Alternative dar. Konkret ermöglichen diese, die Daten sowohl im Speicher, als auch auf der Festplatte zu lagern. Teilweise ist es ebenfalls möglich, reine Datenbanken im hybriden Modus zu betreiben. Auch hier werden die verschiedenen Vorteile von relationalen und In-Memory-Datenbanken miteinander vereint: Die hohe Performance der In-Memory-Technik wird weiterhin erreicht, während die Persistenz traditioneller Datenbank ebenfalls gesichert ist.

[132] Vgl. Röhring, B. (2001)
[133] Vgl. Kaaret, D. (2012)
[134] Vgl. ebenda
[135] Vgl. ebenda

Bezogen auf die zukünftigen Entwicklungen hängt es bei der Entscheidung für eine der vorgestellten Lösung vordergründig davon ab, in welchem Geschäftsumfeld die jeweilige Technologie eingesetzt werden soll – während In-Memory zwar einen schnellen Datenzugriff bietet, ist der Einsatz in geschäftskritischen Bereichen wie z.B. Banken und Versicherungen problematischer. Caché bietet sich vor allem bei Complex Event Processing (CEP)-Umgebungen an, während sich SSD eher bei Anwendungsfällen mit einer niedrigen Anzahl an Datenbankveränderungen als sinnvoll erweist.

6.2 Zukunftsausblick / Entwicklung von In-Memory-DB

Wie bereits erwähnt, ebnen die immer weiter sinkenden Preise für Hardwarekomponenten, insbesondere Arbeitsspeicher, den Weg für die Anwendung von In-Memory Datenbanksystemen in verschiedenen Bereichen. Auch kleine und mittelständische Unternehmen erhalten zunehmend die Möglichkeit in In-Memory zu investieren. Es liegt nahe, dass auch aus diesem Grund Gartner 2010 in seinem Technologie "Hype-Cycle" (grafische Darstellung zur Reife und Verbreitung von Technologien) In-Memory Datenbanksysteme auf ihrem Höhepunkt sieht.[136] Dies weist ebenso deutlich auf eine immer weiter wachsende Marktpräsenz von IMDBS hin und begründet die Frage, welche Innovationen überhaupt noch möglich sind.

Unter der Prämisse immer weiter wachsender Multicore-Lösungen für Hauptspeicher wächst die Idee von "Echtzeit-Computing", welche vor allem Business Intelligence zu neuen Potentialen verhelfen kann.

Bei der Nutzenmaximierung der Vorteile von IMDBS und der Entwicklung von Zukunftsszenarien müssen zwangsläufig auch die direkten Einflussfaktoren auf die Datenbank und deren Entwicklung berücksichtigt werden. Dies führt dazu, dass die Bewertung der technologischen Zukunft von In-Memory Datenbanken entscheidend von ihren Integrationsmöglichkeiten mit zukunftsweisenden Anwendungen zusammen hängt.

So ist es vorstellbar, dass agile Prozesse der Anwendungssoftware als solches zu einem Paradigmenwechsel verhelfen. Die Entwicklung geht hierbei in Richtung Mobilität, Flexibilität und Benutzerzentriertheit. IMDBS sind geeignete Systeme im Rahmen der Datenverwaltung, um auf diese Veränderungen zu reagieren, sie benötigen hierzu allerdings erweiterte Möglichkeiten zu Transaktionsdurchführung.[137] In-Memory Entwicklung bedeutet folglich eine

[136] Vgl. Thoo, E.; u.a. (2010)
[137] Vgl. Loos, Prof. Dr., P. (o.J.)

Symbiose aus Entscheidungsunterstützung, im Sinne von Business Analytics, und Transaktionen in Verbindung mit modernen Anwendungssystemen zu schaffen.

Um die Kapazitäten von In-Memory Datenbanken zu erweitern, sollen Systeme in Zukunft nur noch die benötigten Spalten in ihrem vollen Umfang in ihrem Speicher halten, während Daten die zum Beispiel aus Revisionszwecken benötigt werden stark komprimiert werden.[138]

Ein weiterer Schritt ist dann die Verlagerung von Berechnungen aus der Anwendung in das Datenbanksystem, dies wäre somit der erste Schritt hin zu einer vollständigen Integration von Business Intelligence in In-Memory Datenbanken, anstatt diese Funktion, wie bisher, als Zusatz zu betrachten.[139]

Die Entwicklungsszenarien von IMDBS sind vielfältig und es muss deutlich gesagt werden, dass technologische Trends wie Big Data und Business Analytics deutliche Gründe für die Weiterentwicklung dieser Datenbanktechnologie sind. Die Zukunft von IMDBS ist folglich stark abhängig von der weiteren Entwicklung dieser Trends.

[138] Vgl. Bube, L. u.a. (2010)
[139] Vgl. ebenda

7 Fazit
(A. Croll, T. Dietz, P. Koch, T. Soetebeer)

Die innerhalb der Ausarbeitung durchgeführten Untersuchungen haben aufgezeigt, dass IMDBs einen veränderten Ansatz gegenüber traditionellen Datenbankmanagementsystemen darstellen. Dies spiegelt sich vor allem in den ausgewählten Merkmalen für IMDBs wider. Mit Hilfe der Betrachtung dieser Merkmale anhand eines konkreten Beispiels in Abgrenzung zu einer reinen Produktbeschreibung von SAP HANA hat die Untersuchung folgende Erkenntnisse hervorgebracht.

Die Alleinstellungsmerkmale von In-Memory Datenbanken setzen sich aus zwei Perspektiven zusammen. Die Speicherung der Daten unterscheidet sich einerseits auf Hardware-Ebene und andererseits in der strukturellen Umsetzung der Datenverarbeitung.

Die Hardware liefert in diesem Zusammenhang die Grundlage für die Umsetzung von SAP HANA und anderen In-Memory Datenbank Managementsystemen. Vor allem die Verwendung des Hauptspeichers als primäres Speichermedium sowie die Anwendung einer Multicore-, Multinode-Architektur vermeidet den bestehenden Flaschenhals beim Zugriff auf die Festplatte.

Hinsichtlich der Organisation und Speicherung von Daten kombiniert SAP HANA Spalten- und Zeilenorientierung, um je nach Struktur der Abfrage die Vorteile beider Ansätze ausnutzen zu können. Die Spaltenorientierung stellt dabei die Grundlage für den Einsatz verschiedener Kompressionsverfahren dar. Neben der Kompression tragen auch die Parallelisierung und die Indexierung der Datensätze entscheidend zur Verbesserung der Performance in IMDBs bei. Dies spiegelt sich in der reduzierten Dauer von Datenbankabfragen wider.

Dennoch stellt die Anforderung der Persistenz weiterhin eine Herausforderung im Umgang mit In-Memory Datenbanken dar, weil es sich beim verwendeten Hauptspeicher um ein flüchtiges Speichermedium handelt. Folglich sind diverse Maßnahmen zur Sicherstellung der Persistenz in IMDBs notwendig.

Rückblickend hat sich das gewählte Vorgehen als praktikabel erwiesen, da anhand von SAP HANA die Relevanz der ausgewählten Merkmale sowie die Grenzen von In-Memory Datenbanken aufgezeigt werden konnten. SAP HANA hat sich als sinnvolle Beispieldatenbank erwiesen, da es sich bei diesem System um eines der führenden Produkte auf dem Markt handelt und eine Vielzahl verschiedener Anwendungsszenarien abdeckt.

Die Beschaffung neutraler Fachliteratur stellte eine Herausforderung dar. Der Großteil der am Markt verfügbaren Quellen ist stark auf SAP HANA fokussiert, wodurch eine neutrale Betrachtung erschwert wurde.

Abschließend betrachtet zeigt sich die Notwendigkeit, In-Memory Datenbanksysteme auf verschiedene Anwendungsfälle im Geschäftsumfeld auszurichten, wobei hier vor allem OLAP und OLTP zu berücksichtigen sind. Des Weiteren geht aus der Ausarbeitung hervor, dass In-Memory-Datenbanken in bestimmten Anwendungsszenarien schneller arbeiten als herkömmliche Systeme.

Allerdings muss auch beachtet werden, dass eine Umstellung auf eine In-Memory-Lösung auch für erhebliche Kosten für zusätzliche Hardware, Softwarelizenzen sowie einen erhöhten Aufwand für Wartung und Backup/Recovery-Mechanismen sorgt.

Anwender sollten das Hauptaugenmerk darauf legen, eine möglichst hohe Ausfallsicherheit zu gewährleisten, da sich eine dauerhafte Persistenz bekanntermaßen als größtes technisches Problem für IMDBs darstellt. Hierbei darf der Geschwindigkeitsvorteil durch IMDBs nicht auf Kosten der Ausfallsicherheit der Datenbestände hingenommen werden. Wie sich gezeigt hat, existieren hier unterschiedliche Lösungen, um diese Persistenz zu gewährleisten bzw. um Datenverlust bspw. durch Stromausfälle zu vermeiden.

Abschließend gilt es also die Frage zu stellen, ob eine Migration wirtschaftlich sinnvoll ist. In Zeiten sinkender Preise für Hardwarekomponenten ist Arbeitsspeicher noch immer teuer im Vergleich zu herkömmlichen Speichermedien, aber durch den technologischen Fortschritt können auch hier weitere Einsparungen erwartet werden. Dies ist vor allem für die Serverarchitektur von Bedeutung, welche die Grundlage für eine IMDB bildet.

Eine Umstellung oder Migration muss daher momentan vor allem unter folgenden beiden Gesichtspunkten beurteilt werden: Einerseits ist zu beurteilen, ob genügend Know-How für die Umstellung vorhanden ist und andererseits muss abgewogen werden, ob die finanziellen Mittel verfügbar sind bzw. entsprechend eingesetzt werden können.

Zusammenfassend stellen In-Memory Datenbanken nicht nur eine Alternative zu herkömmlichen Technologien dar, sondern entwickeln sich auch zu einer langfristigen Lösung für moderne Geschäftsanwendungen, vor allem im Kontext der Verarbeitung wachsenden Datenmengen im Rahmen von Big Data.

Quellenverzeichnis

Literaturverzeichnis

Association for Computing Machinery (Hrsg.) (2008): Proceedings of the 34th VLDB Conference, Auckland, New Zeeland: Jagadish, H. V.

Bauer, A. (2013): Data Warehouse Systems – Architektur, Entwicklung, Anwendung, Hrsg. Günzel, H., 4. Auflage, Heidelberg: dpunkt.verlag GmbH

Choinyambuu, S. (2010): In Memory Database Performance evaluation based on query time, Seminar Database Systems, Rapperswil: HSR Hochschule für Technik Rapperswil

Duden (1993): Duden Informatik: Ein Sachlexikon für Studium und Praxis, Hrsg. Dudenverlag, 2. Auflage, Dudenverlag: o.Ort

Gruenes, J., Ilacqua, C., Oehler, K. (2012): IBM Cognos TM1 The Official Guide, 1. Auflage, o.Ort: McGraw – Hill Osborne Media

Kempler, A., Eickler, A. (2009): Datenbanksysteme, München: Oldenbourg Verlag

Plattner, H. (2013): A Course in In-Memory Data Management – The Inner Mechanics of In-Memory Databases, Hrsg. Plattner, H., Heidelberg Berlin New York Dortecht London: Springer Verlag

Plattner, H. (2013a): Lehrbuch In-Memory Data Management – Grundlagen der In-Memory Technologie, Hrsg. Plattner, H., Wiesbaden: Springer Gabler

Quinn, E. (2013): When In-Memory Matters, Basics for Database Buyers and Service Providers, o.O.: HSF Research

Verzeichnis von Journalen, Tagungsberichten und wissenschaftliche Ausarbeitungen

Boncz, P., Kersten, M. L., Manegold, S. (o.J.): Database Architecture Opimized for new Bottleneck: Memory Access, CWI and Data Distilleries B.V., Amsterdam: o.V.

Loos, Prof. Dr., P. (o.J.): In-Memory-Datenmanagement in betrieblichen Anwendungssystemen, in: Wirtschaftsinformatik, 06/2011

Ott, J.; Stirnimann, R. (2012): In-Memory Datenbanken im Vergleich, in: Computerwoche, 03/2012

SAP HANA (2012): SAP HANA Database for Next-Generation Business Applications and Real-Time Analytics – Explore and Analyze Vast Quantities of Data from Virtually Any Source at the Speed of Thought, o.Ort, o.V.

Verzeichnis der Internet- und Intranet-Quellen

Bendelac, C. (2014): SAP HANA – High Availability Whitepaper http://www.saphana.com/docs/DOC-2775, Abruf: 22.01.2014

Bernard, M. (2011): SAP HANA features, http://scn.sap.com/thread/2043200, Abruf: 23.01.2014

Bube, L. u.a. (2010): In-Memory-Datenbanken – Traum und Wirklichkeit, http://www.crn.de/software/artikel-84147-4.html, Abruf: 14.12.2013

Cao, J., Gupta, U. (2013): In-Memory Database Platform for Big Data – Help you to tame Big Data, http://de.slideshare.net/SAPTechnology/inmemory-database-platform-for-big-data, Abruf: 09.01.2014

Chan, C.-Y.; Ioannidis, Y. (Hrsg.) (1998): Bitmap Index Design and Evaluation, ACM SIGMOD Conference, http://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CDkQFjAA&url=http%3A%2F%2Fwww.comp.nus.edu.sg%2F~chancy%2Fsigmod98.pdf&ei=fRPoUqDnI-iJ7AaA2YD4Cw&usg=AFQjCNGVQk8hyTdudIR3cDfubyw8DM61ZA&sig2=SuKPMP2Mdw2zFLodoZI_qw&bvm=bv.60157871,d.ZGU&cad=rja, Aufruf: 26.01.2014

Cole, B. (2007): Hybrid embedded database merges on-disk and in-memory data management, http://www.embedded.com/electronics-news/4135733/Hybrid-embedded-database-merges-on-disk-and-in-memory-data-management, Aufruf: 23.01.2014

Datawarehouse4u (Hrsg.) (2013): OLTP vs. OLAP, http://datawarehouse4u.info/OLTP-vs-OLAP.html, Abruf: 25.01.2013

Datenbanken Online Lexikon (2013): SAP-HANA, http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/SAP-HANA, Abruf: 13.01.2013

Dontcheff, J. (2013): HANA and her sisters, http://juliandontcheff.wordpress.com/2013/11/30/hana-and-her-sisters/, Abruf: 23.01.2014

eliteinformatiker (Hrsg.) (2011): Zeilen- und spaltenorientierte Datenbanken, http://eliteinformatiker.de/2011/08/07/zeilenorientierte-und-spaltenorientierte-datenbanken/, Abruf: 03.01.2014

Funk, M. / Marinkov, B./Paar, M. (2012): Trends in der IT: In-Memory-Technologie, http://trends-in-der-it.de/?Fachartikel/In-Memory-_Technologie, Abruf: 14.12.2013

Föll H. (o.J.): Entropie und Information, Uni Kiel, http://www.tf.uni-kiel.de/matwis/amat/mw1_ge/kap_5/advanced/t5_3_2.html, Abruf: 25.01.2014

Hasso Plattner Institut (2013): Dictionary Encoding – Video Podcast, http://www.saphana.com/docs/DOC-3157, Abruf: 09.01.2014

HANA Tutorials (Hrsg.) (2011): How In-Memory Data & Persistent Storage works in HANA?, http://www.freehanatutorials.com/2012/11/how-in-memory-data-persistent-storage.html, Abruf: 21.01.2014

Hasso Plattner Institut (Hrsg.) (o.J.): In-Memory Data Management for Enterprise Applications, https://epic.hpi.uni-potsdam.de/Home/InMemoryDataManagementInnovations#attributeindex, Aufruf: 15.01.2014

heiseDeveloper (Hrsg.) (2010): Caché-Datenbank auf Java-Umgebungen mit Complex Event Processing abgestimmt, http://www.heise.de/developer/meldung/Cache-Datenbank-auf-Java-Umgebungen-mit-Complex-Event-Processing-abgestimmt-1097363.html, Abruf: 14.12.2013

Herden, O. (2012): Spaltenbasierte Datenbanken – Ein Konzept zur Handhabung großer Datenmengen, http://www.gil-net.de/Publikationen/25_135.pdf, Abruf: 04.01.2014

HTWK Leipzig (Hrsg.) (o.J.): Merkmale eines DBMS, http://www.imn.htwk-leipzig.de/~kudrass/TB-Datenbanken/probe.pdf, Abruf: 19.12.2014

Hutsell, W. / Bowen, J. (o.J.): In Memory Databases vs. Solid State Disks, www.remote-dba.net/t_in_memory_cohesion_ssd.htm, Abruf: 14.12.2013

IBM Germany (Hrsg.) (o.J.): IBM solidDB, http://www-03.ibm.com/software/products/de/ibmsoli/, Aufruf: 24.01.2014

Kareet, D. (2012): InterSystems-Caché als Alternative zu In-Memory-Datenbanken, http://www.intersystems.de/cache/whitepapers/pdf/Whitepaper_InMemory.pdf, Abruf: 14.12.2013

Kreitz, C. (1995): Grundlagen der Datenbanken, http://www.cs.uni-potsdam.de/ti/kreitz/Teaching/Datenbanken/Folien.pdf, Abruf: 19.12.2014

Krueger, J. u.a. (o.J.): Hauptspeicherdatenbanken für Unternehmensanwendungen, http://ares.epic.hpi.uni-potsdam.de/apps/static/papers/Version_11.10.2010.pdf, Abruf: 04.01.2014

Kurzlechner, W. (2013): SAP hängt Oracle weiter ab, CIO Magazin, http://www.saphana.com/community/about-hana, Abruf: 27.01.2014

Löhl, J. u.a. (2005): Historische Entwicklung von Datenbanken http://www.chrisix.net/fhhn/ss05/1_hist_entwicklung_handout.pdf, Abruf: 19.12.2014

McObject (2014): In-Memory Database Systems – Questions and Answers http://www.mcobject.com/in_memory_database, Abruf 05.01.2014

Narayanan, D.; Hodson, O. (2012): Whole-system Persistence with Non-volatile Memories, http://research.microsoft.com/apps/pubs/default.aspx?id=160853, Aufruf: 28.01.2014

OSBI Blog (Hrsg.) (2011): Datenspeicherung auf Festplatte, SSD oder In-Memory, http://www.osbi-blog.de/2011/05/datenspeicherung-festplatte-ssd-in-memory/, Abruf: 14.12.2013

Philipps Universität Marburg (Hrsg.) (o.J.): Eigenschaften von Datenbanksystemen: http://gisbsc.gis-ma.org/GISBScL4/de/html/GISBSc_VL4_V_lo2.html, Abruf: 19.12.2014

Plattner, H. (o.J.a): Hash-Join Algorithmen, Hasso Plattner Institut, http://www.hpi.uni-potsdam.de/fileadmin/hpi/FG_Naumann/folien/WS0809/advanced_topics_in_db/Hash-Join_Algorithmen.pdf, 15.01.2014

Plattner, H. (o.J.b): Parallel Aggregation, Hasso Plattner Institut, http://www.saphana.com/servlet/JiveServlet/download/3175-2-9389/parallel_aggregation.pdf, Aufruf: 27.12.2013

Plattner, H. (o.J.c): Parallel Data Processing, Hasso Plattner Institut, http://www.saphana.com/servlet/JiveServlet/download/3168-2-9382/parallel_data_processing.pdf, 15.01.2013

Plattner, H. (o.J.d): Parallel Join, Hasso Plattner Institut, http://www.saphana.com/servlet/JiveServlet/download/3174-2-9388/parallel_join.pdf, Aufruf: 20.12.2013

Plattner, H. (o.J.e): Parallel Select, Hasso Plattner Institut, http://www.saphana.com/servlet/JiveServlet/download/3172-3-9386/parallel_select.pdf, Aufruf: 17.01.2014

Puff, Michael (2008): Gegenüberstellung von Datenbankmodellen, http://www.michael-puff.de/Archiv/Ausbildung_FIAE/Referate/Datenbankmodelle.pdf, Abruf: 19.12.2014

Rahm, E. (1996): Parallele Datenbanksysteme, http://dbs.uni-leipzig.de/abstr/pdbs.html, Abruf: 20.01.2013

Röhrig, B. (2001): Postrelationale Datenbank Cache, http://www.linux-magazin.de/Ausgaben/2001/11/Objektive-Vielfalt, Abruf: 14.12.2013

SAP (2014): What is SAP HANA?, http://www.saphana.com/community/about-hana, Abruf: 27.01.2014

SAP AG (Hrsg.) (2013): SAP HANA Developer Guide, http://help.sap.com/hana/SAP_HANA_Developer_Guide_en.pdf, 28.01.2014

SAP HANA (2013): HANA Row Store and Column Store, SAP HANA Forum, http://www.saphana.com/thread/2505, Abruf: 10.01.2014

Sieprath, S. (2005): Skript zur Vorlesung Datenbanken http://www.rz.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaabhrdm, Abruf: 18.01.2014

Somers, K. (2012): SAP HANA – Core Architecture http://en.community.dell.com/techcenter/b/techcenter/archive/2012/09/28/sap-hana-core-architecture.aspx, Abruf: 23.01.2014

Stamp, N. (2013): Was bedeutet ECC-Fehlerkorrektur?, http://www.computerwoche.de/a/alles-was-sie-ueber-ssds-wissen-muessen,2367089,9, Abruf: 14.12.2013

Strehlitz, M. (2011): Datenbanken: Kampf zwischen Zeilen und Spalten, http://www.zdnet.de/41558616/datenbanken-kampf-zwischen-zeilen-und-spalten/, Abruf: 03.01.2014

Süddeutsche Zeitung (Hrsg.) (2011): 100 Jahre IBM - Digitaler Code in Lochkarten, http://www.sueddeutsche.de/digital/jahre-ibm-digitaler-code-in-lochkarten-1.1108949-4, Abruf: 19.12.2014

Tagesanzeiger (Hrsg.) (2011): Erfinder der Festplatte wird 100 Jahre alt, http://www.tagesanzeiger.ch/digital/computer/Erfinder-der-Festplatte-wird-100-Jahre-alt/story/10427793, Abruf: 19.12.2014

Thoo, E.; u.a. (2010): Hype Cycle for Data Management, www.gartner.com/doc/1406713, Aufruf: 20.01.2014

WinfWiki (Hrsg.) (2012): Beschreibung und Einsatz von In-Memory-Computing, http://winfwiki.wi-fom.de/index.php/Beschreibung_und_Einsatz_von_In-Memory-Computing, Abruf: 25.01.2014

Witter, C. (2009): Klassifizierung von Ansätzen für column oriented DBMS http://wwwiti.cs.uni-magdeburg.de/iti_db/publikationen/ps/auto/thesisWittwer.pdf, Abruf: 04.01.2014

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

NoSQL-Datenbanksysteme/-Dienste aus der Cloud

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“
Kompetenzzentrum Open Source (KOS)

Vorgelegt von

Manuel Meyer
David Pohlmann

am 31.01.2014

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
Kurs WWI2011I

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Grundlegend kann man heutzutage zwei verschiedene Arten von Datenbanken unterscheiden: die SQL- und die NoSQL-Systeme.

SQL, oder früher auch SEQUEL, als Kurzform für „Structured Query Language“[1] genannt, „ist eine Skriptsprache, die gleichermaßen dem Datenbankadministrator, dem Datenbankentwickler und dem Datenbankbenutzer einen Zugriff auf die Datenstruktur und die Datenbank ermöglicht.“[2] Sie wurde in den 70er Jahren des letzten Jahrhunderts von IBM-Forschern entwickelt. Lange Zeit galt SQL als der de-facto Standard der Datenbanksprachen.

Zur selben Zeit startete eine andere Bewegung bezüglich Datenbanksystemen, die der sogenannten NoSQL-Systeme. Die erste Key/Hash Datenbank wurde 1979 von Ken Thompson entwickelt.[3] Weitere Ansätze folgten, bauten jedoch häufig noch auf einer SQL-Grundlage auf. Grundlegender Unterschied war allerdings, dass diese Datenbanksysteme keine SQL-API mehr anboten.[4]

Ein großer Vorteil der heutigen NoSQL-Datenbanken ist die Art der Datenspeicherung, die es ermöglicht, Datenbanken sehr stark und vor allem ohne großen Aufwand zu skalieren. Dies begründet auch den Zuwachs an Attraktivität im Zeitalter des Web 2.0.[5] Durch die ständig steigende Menge der Daten und die zunehmende Vernetzung dieser gilt es, sich stets neuen Herausforderungen der Datenverwaltung und -bearbeitung zu stellen und diese zu lösen. Hierin finden NoSQL-Datenbanken ihr Hauptanwendungsgebiet.[6]

Im Allgemeinen kann man die sogenannten NoSQL-Datenbanken in vier Gruppen unterteilen: Neben den Key/Value, Document und Wide-Column Stores gibt es zudem die Graphendatenbank.

---

[1] Vgl. Taylor, A. (2004), S. 45

[2] Däßler, R. (2001), S.95

[3] Vgl. Edlich, S u.a. (2010), S. 1

[4] Vgl. ebenda, S. 1

[5] Vgl. ebenda, S. 1

[6] Vgl. Andrikopoulus, V. u.a. (2012), S.500

In Unternehmen kommen die NoSQL-Datenbanksysteme immer häufiger zum Einsatz. Bei der XING AG dient eine NoSQL-Datenbank zum Beispiel zur „Bereitstellung des Activity-Streams“[7], der persönlichen Startseite eines XING-Profils.

Marc Schwering, Solution Architect der MongoDB, Inc., nennt OTTO.de, idealo, Bosch, die Scout24-Gruppe, SAP sowie die Bayrische Staatskanzlei als deutsche Unternehmen, die auf NoSQL-Datenbanken zurückgreifen. Internationale Unternehmen seien zudem CERN, Goldman Sachs, the Guardian oder auch eBay.[8]

Durch die steigende Attraktivität der NoSQL-Datenbanken, aufgrund ihrer Vorteile gegenüber herkömmlichen SQL-Datenbanken und die sehr vielseitige Einsetzbarkeit, gibt es heute viele Anbieter von NoSQL-Systemen „as a Service“. Innerhalb von wenigen Minuten kann ein entsprechender Speicherplatz in der Cloud beschafft werden, um eine solche Datenbankanwendung anzulegen.

Diese Arbeit soll einige der wichtigsten Database-as-a-Service-Anbieter (DBaaS) näher erläutern und einen Vergleich verschiedener Provider ermöglichen. Dabei werden zum einen allgemeine Cloud-Anbieter, wie zum Beispiel der Cloud-Service Amazon AWS, vorgestellt und anschließend spezielle NoSQL-DBaaS-Anbieter beleuchtet. Zudem werden Kriterien festgelegt anhand welcher Unternehmen die Möglichkeit haben sollen, den passenden Provider für entsprechende Geschäftstätigkeiten zu ermitteln.

[7] Abel, F. (2013)

[8] Schwering, M. (2013)

2 Arten von NoSQL-Datenbanken

2.1 Key/Value Stores

Die Key/Value Datenbank funktioniert nach dem Prinzip des assoziativen Datenfeldes. Dabei wird mit Hilfe eines meist nicht nummerischen Schlüssels auf den abgespeicherten Wert zugegriffen[9]. Einfach ausgedrückt handelt es sich also um eine Tabelle mit zwei Spalten, eine, die den Key speichert, die andere, die den eigentlichen Wert beinhaltet. Dieses Verfahren nennt sich Hashwertberechnung, bei dem die Hashtabelle als Indexstruktur dient[10]. Die Wertezuordnung in diesem assoziativen Array kann auch als Funktion mit endlichem Wertebereich angesehen werden.[11]

Die zugrundeliegende Indexstruktur bietet einen entscheidenden Vorteil gegenüber relationalen Datenbanken. Durch die eben angesprochene Hashwertberechnung werden die Zielobjekte bereits nach einem Schritt enorm eingeschränkt, sodass eine Suchabfrage wesentlich schneller durchgeführt werden kann. Dies ist der Tatsache zu verdanken, dass nicht alle Werte ausgelesen werden müssen um den Zielwert zu erhalten, sondern lediglich die Werte mit dem passenden Schlüssel.

Die eigentliche Datenlogik liegt bei den Key/Value Stores nicht in der Datenbank selbst, sondern ist in der Anwendung zum Auslesen der Daten hinterlegt. Die in der Anwendung enthaltenen Zugriffsmuster können dabei stark optimiert und an den Nutzer angepasst werden. Man spricht hier auch von den „Client defined Semantics“. Ein großer Vorteil hierbei ist die hohe Schreibperformance der Datenbank.

Durch die einfache Datenspeicherung und die zusätzliche Anwenderschicht ist die Key/Value Datenbank sehr gut skalierbar[12]. Desweiteren ist die Datenbank sehr einfach über mehrere Server zu verteilen, da kein komplexes Datenmodell zugrunde liegt, sondern lediglich die Schlüssel und die zugehörigen Werte gespeichert sind. Diese Schlüssel sind daher ohne größere Probleme auf verschiedene Server verteilbar (spread keys). Daraus ergibt sich die Möglichkeit, mehrere Abfragen gleichzeitig machen zu können. Dies verkürzt die Zugriffszeit auf die Daten.

---

[9] Vgl. Heise (2013)

[10] Vgl. Wikipedia (2013a)

[11] Vgl. Wikipedia (2013b)

[12] Vgl. Ha, T. (2011)

Die Hashtabelle bringt allerdings auch Nachteile mit sich, die beachtet werden sollten. Die Suche läuft bei den Key/Value Stores stets über die Hashwertberechnung ab, in dem über die Schlüssel der Zielobjekte die gesuchten Werte ermittelt werden. Dies zieht mögliche Probleme bei steigender Datenmenge mit sich. Auf der einen Seite besteht bei der Hashtabelle die Gefahr der sogenannten Entartung bei zunehmender Datensatzanzahl. Durch Kollision kann es dabei zu Fehlern bei Suchabfragen kommen.[13] Auf der anderen Seite kann die Suchperformance nur bedingt beeinflusst werden, da sie eben nach demselben Prinzip abläuft. Jedoch kann man durch das Erstellen einer zweiten Indexstruktur dieses Problem beheben. Dieser Schritt ist ab einem gewissen Datenvolumen anzudenken.

Haupteinsatzgebiet der Key/Values Stores ist vor allem die Verwaltung von Benutzerprofilen, Benutzersessions oder zum Beispiel Warenkörben. Ein bekanntes Beispiel hierfür ist der Amazon Shopping Warenkorb, der auf der Key/Value Datenbank Amazon DynamoDB basiert.[14] Ein weiteres Beispiel für einen Key/Value Store ist Redis. Der Anbieter für die DynamoDB ist Amazon AWS. Ein Beispielanbieter für Redis ist Garantia.

2.2 Document Stores

Document Stores verwenden Dokumente als Speichergrundlage. Dabei kann es sich um einzelne Werte oder auch ganze Code-Abschnitte handeln, der Aufbau ist also beliebig[15]. Es handelt sich also nicht um Dokumente im eigentlichen Sinn, sondern um „strukturierte Datensammlungen wie JSON, YAML oder RDF-Dokumente“[16] (Siehe Abbildung 1).

[13] Goossaert, E. (2013)

[14] Vgl. Rahien, A. (2010)

[15] Vgl. MongoDB, Inc. (2014)

[16] Edlich, S u.a. (2010), S. 8

Category and Subcategory Documents

Key (Category ID)	Document
1	CategoryName: Bikes Subcategories: SubcategoryID: 1 SubcategoryName: Mountain Bikes SubcategoryID: 2 SubcategoryName: Road Bikes ...
2	CategoryName: Components Subcategories: SubcategoryID: 10 SubcategoryName: Handlebars SubcategoryID: 11 SubcategoryName: Bottom Brackets ...
3	CategoryName: Clothing Subcategories: SubcategoryID: 30 SubcategoryName: Bib Shorts SubcategoryID: 31 SubcategoryName: Caps ...

Abb. 1: Beispiel eines Dokuments [17]

Jedes der abgespeicherten Dokumente ist über einen eindeutigen Schlüssel abrufbar. Diese Suchabfragen über die Schlüssel können durch eine Indexierung der Schlüssel verbessert werden. Außerdem besteht bei den Document Stores zudem die Möglichkeit, Abfragen des Dokumenteninhalts zu machen, wodurch die Suche weiter modifiziert und personalisiert werden kann.

Vorteil dieser NoSQL-Datenbankart ist auf der einen Seite die einfache Speicherung unstrukturierter Daten. Dies ist möglich, da die „Dokumente“ selbst einzelne Einheiten darstellen, die nicht von anderen abgelegten Datensammlungen abhängig sind. Durch die Unabhängigkeit der Daten besteht außerdem die Möglichkeit, die Daten einfach zu verteilen und somit auch auf unterschiedlichen Servern zu speichern. Hinzukommt, dass wie bei den Key/Value Stores auch bei den Document Stores zunächst kein Informationsschema benötigt wird, da die Logik ebenfalls in der Anwendung selbst hinterlegt ist.

Hauptanwendungsgebiete sind hauptsächlich Webseiten mit einer Vielzahl an Nutzern und Aufrufen. Darunter fallen zum Beispiel Blogs. Desweiteren sind Hauptanwendungsgebiete Big-Data-Anwendungen sowie stark frequentierte Applikationen.

[17] Microsoft (2014d)

Zwei Beispiele für Document Stores sind MongoDB und CouchDB. Anbieter für MongoDB sind Object Rocket, mongolab, MongoHQ und MongoSoup. IrisCouch und cloudant.com sind Anbieter für CouchDB.

2.3 Wide-Column Stores

Die spaltenorientierten Datenbanken stehen im Gegensatz zu den zeilenorientierten Speicheransätzen. Bei herkömmlichen, zeilenorientierten Datenbanken werden bei Abfragen alle vorhandenen Attribute eines Tupels ausgelesen[18]. Durch die spaltenorientierte Speicherung der Wide-Column Stores können bei Abfragen jedoch Tupel so ausgelesen werden, dass jeweils nur das entscheidende Attribut betrachtet wird. Somit sind Abfragen wesentlich schneller als bei zeilenorientierten Datenbanken[19]. Das erklärt auch den Hauptanwendungsbereich in OLAP-Anwendungen[20].

Ein Beispiel für einen Datensatz mit den Attributen ID, Name und Alter in einer zeilenorientierten beziehungsweise spaltenorientierten Datenbank sähe wie folgt aus:
Zeilenorientiert: 1, Tom, 42; 2, Mary, 18; 3, Paul, 36
Spaltenorientiert: 1, 2, 3; Tom, Mary, Paul; 42, 18, 36.[21]

Ein großer Nachteil der Wide-Column Stores ist allerdings der vergleichsweise große Aufwand beim Schreiben von Tupeln. Durch die anders organisierte physische Datenorganisation „ist das Schreiben und Lesen von Objektstrukturen, d.h. von zusammengehörigen Spaltendaten „aufwendiger, da man viel springen und suchen muss“.[22]

Desweiteren gibt es die sogenannten „partiell spaltenorientiert[en]“ Datenbanken.[23] Diese Art Datenbank speichert Daten in „Datensätzen mit potentiell sehr vielen dynamischen Spalten ab“[24]. Google hat mit seiner Datenbank Google Big Table die Grundlage hierfür geschaffen. Die Architektur zeichnet sich durch „mehrdimensionale Tabellen“[25] aus und kann als ein zweidimensionaler Key/Value Store angesehen werden.[26] Google selbst bezeichnet diese

[18] Vgl. Bösswetter, D. (2010)
[19] Vgl. Steemann, J. (2012)
[20] Vgl. Harizopoulos, S. (2009)
[21] Vgl. Edlich, S u.a. (2010), S. 53
[22] Ebenda, S. 53
[23] Ebenda, S. 53
[24] Solid IT (o.J.)
[25] Vgl. Edlich, S u.a. (2010), S.54
[26] Solid IT (o.J.)

mehrdimensionale Struktur des Datenmodells als „sparse, dirstributed, persistent multidimensional sorted map“[27] Die Daten werden zunächst in übergeordnete „Column Families“ strukturiert, die in sich jedoch nochmal anders organisiert sind. Auf einen bestimmten Wert in einer „Column Family“ wird mittels des Zeilennamens, des Spaltenschlüssels sowie eines sogenannten „Timestamps“ zugegriffen.[28] In der untenstehenden Abbildung besitzt der Reihenname den Wert „98725“ und der Spaltenschlüssel ist hier durch die „Column Names“ dargestellt, die in mehreren Ebenen existieren dürfen. Der Timestamp fehlt in dieser Abbildung. Er wird jedem Wert automatisch zum Einfügezeitpunkt zugewiesen.[29]

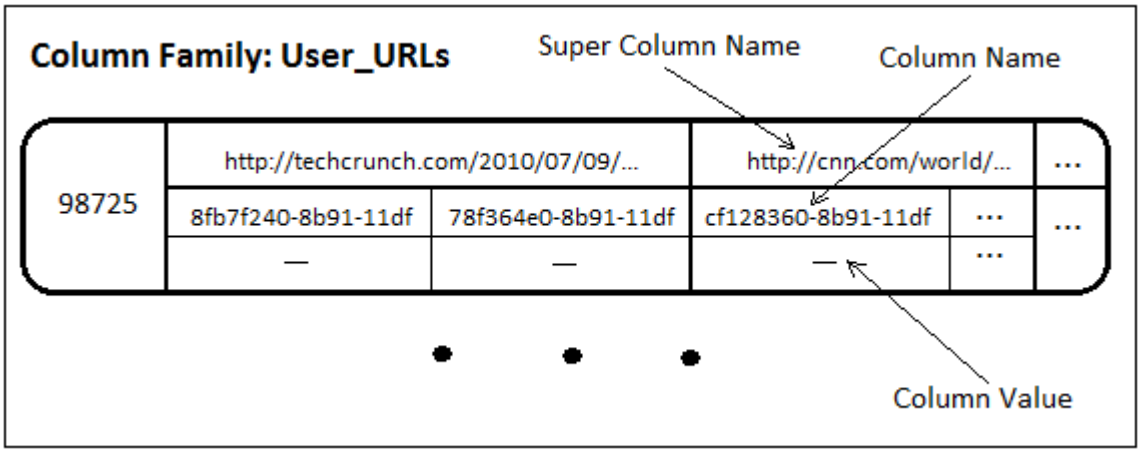


Abb. 2: Column Family[30]

Diese Art der Datenbank ist sehr stark skalierbar. Es wird zwar davon ausgegangen, dass die Anzahl der „Column Families“ recht gering ist. Google geht hier von unter 100 „Column Families“ aus, jedoch kann die Anzahl der Reihenschlüssel enorm hoch sein.[31] Diese Datenbankart ist daher typischer Weise für große Datenmengen geeignet.[32]

Bekannteste Beispiele für partionell spaltenorientierte Datenbanken sind Amazon SimpleDB, Google BigTable, Cassandra oder auch Hadoopanwendungen wie HBase und Cloudera.[33] Ein Anbieter für Cassandra ist zum Beispiel instaclustr.

---

[27] Chang, F. u. a. (2006), S.1

[28] Ebenda, S.1

[29] Ebenda, S.2

[30] Grinev, M. (2010)

[31] Chang, F. u. a. (2006), S.2

[32] Vgl. Yang, B./Qian, W./Zhou, A. (2008), S.211

[33] Vgl. Edlich, S u.a. (2010), S.54

2.4 Graphendatenbank

Die Graphendatenbanken verwenden weder einen tabellen- noch einen dokumentenorientierten Speicheransatz. Die Datenstruktur bei dieser NoSQL-Variante besteht vielmehr aus einem Graphen.[34] Dieser wird in Form einer Netzstruktur, bestehend aus Knoten und Kanten, dargestellt. Dabei können sowohl Knoten als auch Kanten Attribute besitzen, wobei Kanten in der Regel Beziehungen zwischen Knoten, den Objekten, abbilden.[35]

Durch die grafisch gut realisierbare Datenstruktur ist die Datenbankmodellierung für den Anwender recht einfach und verständlich. Ein weiterer Vorteil ist die Agilität: Soll die Datenbank um weitere Prozessabbildungen ergänzt werden, können diese implementiert werden, ohne den bestehenden Graph ändern zu müssen, da bestehende Beziehungen von der Erweiterung meist nicht betroffen sind.

Graphendatenbanken sind auch im Bezug auf die Leistung sehr vorteilhaft. Die Dauer von Abfragen ist beispielsweise proportional zur Ergebnismenge.[36] Das heißt, man hat eine gleichbleibende Abfragegeschwindigkeit, unabhängig vom Datenvolumen. Diese teilweise Verarbeitung der totalen Datenmenge begünstigt das Caching sowie die Skalierbarkeit. Ein großer Vorteil stellt die graphenorientierte Datenbank für die Verwendung von Adhoc-Abfragen dar.[37] Ab dem Einfügezeitpunkt der Daten besteht die Beziehung zwischen den einzelnen Datenfragmenten. Somit gibt es kein Problem mit Join-Abfragen, da die Verbindungen bereits bestehen.[38]

Auch in dieser NoSQL-Datenbankvariante ist wenig Datenbanklogik in der Datenbank selbst vorhanden. Dies macht Datenbanken leicht migrierbar. Die Verteilung der Datenbank auf verschiedene Server erfolgt durch Partionierung. Dabei muss der Graph in sinnvolle Teilgraphen zerlegt werden.[39]

Hauptanwendungsgebiete der Graphendatenbank sind zum Beispiel Netzwerkanalyseanwendungen wie Rechenzentrumsverwaltung oder Supply-Chain-Management sowie soziale Netzwerke aufgrund der gegebenen Vernetzung der Datengrundlage. Desweiteren fallen

---

[34] Vgl. Schönung, S. (o. J.)

[35] Vgl. Rauch, H. (2011)

[36] Vgl. Neubauer, P. (2010)

[37] Vgl. Hunger, M./Neubauer, P. (2013)

[38] Vgl. Eifrem, E. (2013)

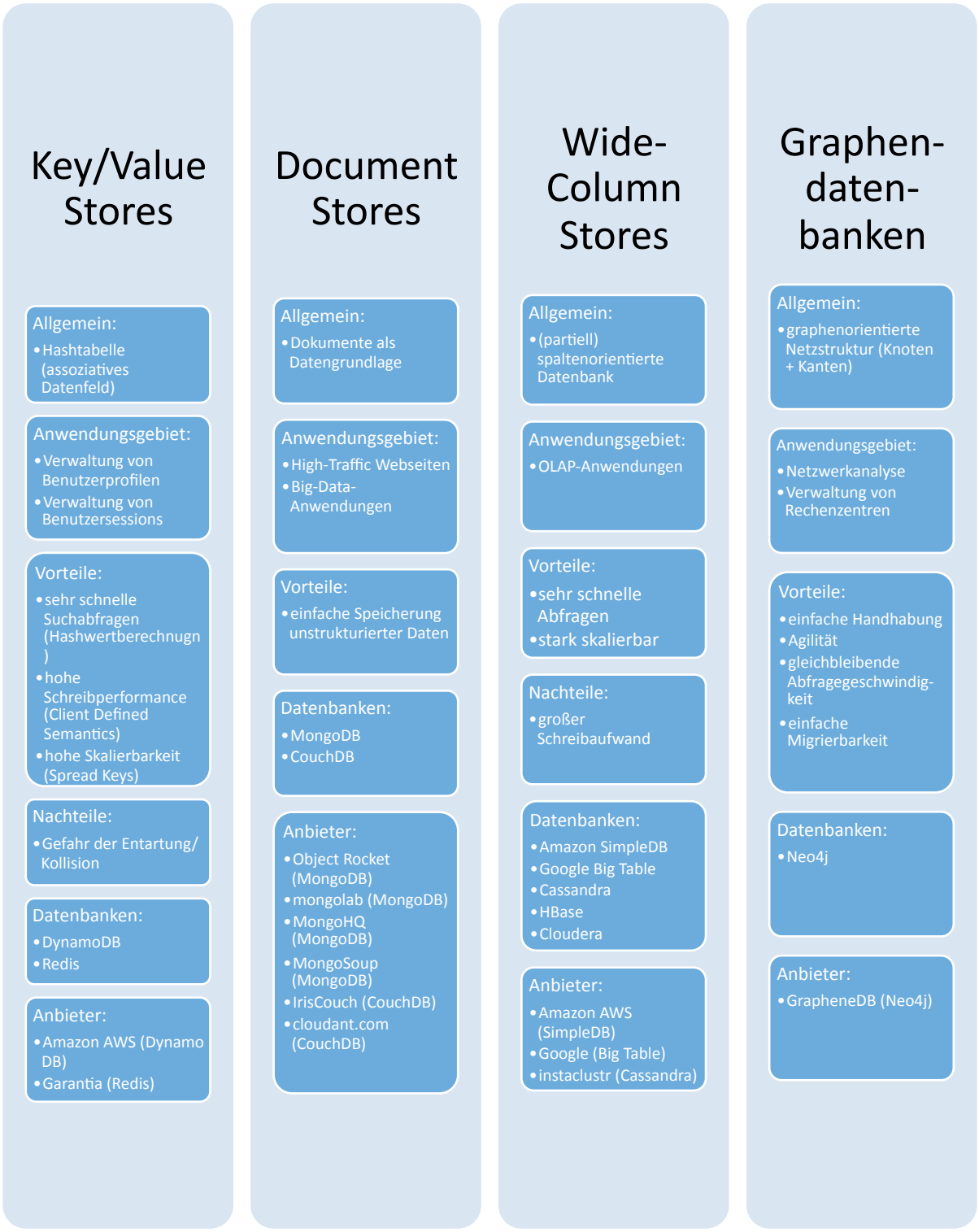
[39] Vgl. FH Köln (2013)

Anwendungen des Informationsmanagement wie Risiko- bzw. Portfoliomanagement in den Anwendungsbereich von Graphendatenbanken. Auch bei Verkehrsleitsystemen, Fahr- und Flugplanoptimierungen kommt diese Datenbankart oft zum Einsatz.[40]

Ein Beispiel für eine Graphendatenbank ist Neo4j, die unter anderem von GrapheneDB angeboten wird.

[40] Vgl. Eifrem, E. (2013)

2.5 Übersicht



Tab. 1: Übersicht NoSQL-Datenbanken

3 NoSQL Database-as-a-Service Provider im Vergleich

Im Rahmen der Recherche für dieses Projekt stellte sich heraus, dass sich im Allgemeinen zwei Arten von Cloud-Anbietern im DBaaS-Segment unterscheiden lassen: Zum einen gibt es allgemeine Cloud-Service-Provider wie Amazon AWS, die generell Rechenleistung bzw. Speicher „on demand" zur Verfügung stellen. Zum anderen gibt es speziell auf DBaaS ausgerichtete Anbieter, die ein oder mehrere bestimmte (NoSQL)-Datenbankdienste bereitstellen.

Die Gruppe der spezialisierten Provider lässt sich weiter dahingehen unterscheiden, ob es sich um einen komplett unabhängigen und selbstständigen Service handelt oder lediglich eine zusätzliche Service-Schicht gebotet wird, welche die Infrastruktur allgemeiner und großer Cloud-Service-Provider wie Amazon AWS oder Rackspace nutzt, um das implementieren und bereitstellen einer NoSQL-Datenbank zu erleichtern. Zudem haben die allgemeinen Cloud-Service-Provider teilweise auch dedizierte, auf NoSQL abgestimmte Services im Programm. Hauptanwendungsgebiet ist hier allerdings das manuelle und selbstständige Implementieren und Einrichten von NoSQL-Datenbanken auf generischen Server-Systemen.

Sowohl die allgemeinen Cloud-Service-Provider im DBaaS-Segment, als auch die spezialisierten Provider sollen im Folgenden näher beleuchtet werden.

3.1 Allgemeine Cloud-Service-Provider im DBaaS-Segment

Zu den wichtigsten Vertretern der allgemeinen Cloud-Service-Provider im DBaaS-Segment gehören Amazon AWS, Windows Azure, Rackspace und Joyent. Desweiteren soll als Vertreter der in Deutschland angesiedelten Cloud-Anbieter Uberspace ebenfalls betrachtet werden. Dem ist hinzuzufügen, dass Amazon AWS vermutlich die Rolle des wichtigsten Cloud-Service-Providers in diesem Bereich einnimmt.[41]

3.1.1 Amazon AWS

3.1.1.1 Allgemeines

Amazon Web Services (AWS) ist das Cloud- und Webserviceangebot des Internetkonzerns Amazon.com. Das Portfolio kann in Bezug auf NoSQL-Datenbanken in drei Rubriken unterteilt werden: Neben dem „Fully Managed Non-Relational Service" und den „Amazon Solution Providers" gibt es die Möglichkeit, den eigenen nicht-relationalen Datenspeicher auf den von Amazon zur Verfügung gestellten Cloud-Servern EC2 oder EBS zu betreiben.

[41] Roth, A. (2013)

3.1.1.2 Technik

Mit dem „Fully Managed Non-Relational Service“ stellt Amazon, als einziger der genannten Anbieter zwei spezielle, eigens entwickelte NoSQL-DBaaS-Lösungen, SimpleDB und DynamoDB bereit. Bei SimpleDB handelt es sich um einen Wide-Column Store für recht kleine Anwendungen bis 10 GB. Amazon DynamoDB hingegen ist ein Datenbankservice, der für beliebig große Datenmengen konzipiert wurde. Er ist vollständig verwaltet und unbegrenzt skalierbar. Ein weiterer Vorteil ist die hohe Verfügbarkeit. Diese wird durch Solid State-Laufwerde (SSD) und die Replikation in drei verschiedenen Zonen gewährleistet.[42]

Mit Hilfe der „Amazon Solution Providers“ zu denen unter anderem 10gen und Couchbase zählen, hat man die Möglichkeit Support und Trainings zu beziehen. Dieser Service bezieht sich vor allem auf die Kunden, die ihre Anwendungen auf den Servern EC2 und EBS laufen haben.

Bei EC2 oder auch „Elastic Compute Cloud“ handelt es sich um einen Service, „der Anwendern virtuelle Server zur Verfügung stellt“.[43] Diese Instanzen können sowohl mit Linux als auch mit Windows betrieben werden. Dabei kann man mit Hilfe des AMI (Amazon Machine Image) Instanzen starten und somit die Anzahl der genutzten Instanzen stündlich neu definieren.[44] Durch die sogenannte „Elastic IP Address“[45] werden die die Server durch IP Adressen zuordenbar und somit zu virtuellen Maschinen. Bei Amazon EBS handelt es sich um einen blockorientierten Speicherdienst.[46] Dies sind „Storage Volumes“ die dem Entwickler die Möglichkeit bieten, eigene Dateisysteme einzurichten, die bzw. diese direkt mit einer Datenbank zu verbinden. Zudem kann EBS direkt an die EC2-Instanzen angebunden werden.[47]

Die dritte Möglichkeit die AWS für NoSQL-Datenbankanwendungen zur Verfügung stellt ist das Bereitstellen von Cloud-Speicher auf den EC2 und EBS Servern. Es handelt sich dabei um einen allgemeinen Cloud-Server. Die Auswahlmöglichkeit besteht hier zwischen einem Linux oder Windows OS. Es gibt dabei eine Vielzahl auswählbarer Standorte. Auch mehrere Standorte gleichzeitig sind möglich. Mit diesem Service kann man die eigene NoSQL Datenbank auf dem Server betreiben und besitzt zu jeder Zeit volle Kontrolle über die Anwendung. Zudem entfällt mit diesem Service die Bereitstellung und Einrichtung benötigter Hardware.

---

[42] Vgl. Amazon Web Services, Inc. (2013a)

[43] Hochstätter, C. (2011)

[44] Vgl. Amazon Web Services, Inc. (2013d)

[45] Ihlenfeld, J. (2008a)

[46] Vgl. Ihlenfeld, J. (2008b)

[47] Vgl. ebenda

3.1.1.3 Kosten/Preismodell

Sowohl für die Amazon DynamoDB als auch für Amazon EC2 wird keine Mindestgebühr veranschlagt, der Einstieg ist also kostenlos.

Zusätzlich zur kostenfreien Nutzung der Amazon DynamoDB gibt es 100 MB kostenlosen Speicherplatz. Wird dieses Speichervolumen überschritten, kann man zusätzliche GB für knapp 0,20 Euro pro Monate dazukaufen. Die Kosten für die Datenbank berechnen sich ab einer gewissen Nutzung ausgehend vom Schreib- und Lesedurchsatz.[48]

Das EC2-Einstiegsangebot bietet ein Jahr kostenlose Nutzung. Diese beinhaltet 750 Stunden monatliche Nutzung einer EC2-Instanz sowie unter anderem 30GB Standard-Volume-Speicher. Nach Ablauf werden die Kosten nach Stunden und Speicher berechnet.[49]

3.1.2 Windows Azure

3.1.1.1 Allgemeines

Windows Azure ist das Cloud-Computing-Angebot von Microsoft. Es handelt sich dabei ebenfalls um eine bedarfsgesteuerte und skalierbare Cloud-Infrastruktur. Auch Microsoft bietet zwei unterschiedliche Möglichkeiten Windows Azure für den Einsatz von NoSQL-Datenbankanwendungen zu nutzen. Neben dem HDInsight-Dienst stellt Windows Azure auch Virtual Machines zur Verfügung.

3.1.1.2 Technik

Bei dem Windows Azure HDInsight-Dienst handelt es sich nicht, wie bei der Amazon DynamoDB, um eine eigene NoSQL-DBaaS-Lösung. HDInsight ist lediglich ein auf Hadoop basierendes Business Intelligence System im Programm. Es basiert vollständig auf der Apache Hadoop Lösung in der Cloud. Je nach Nachfragebedarf können die hierbei verwendeten Cluster innerhalb weniger Minuten erweitert oder reduziert werden.

Ebenso wie bei Amazon AWS hat man bei Windows Azure mit Hilfe der virtuellen Computer (Virtual Machines)[50] die Möglichkeit zwischen einem Linux oder Windows System zu wählen, wodurch eine beliebige NoSQL-Datenbank implementiert und konfiguriert werden kann.

3.1.1.3 Kosten/Preismodell

Im Gegensatz zum Amazon AWS Angebot gibt es keinen kostenlosen Einstieg. Der Preis für den Windows Azure HDInsight-Dienst wird nach den verwendeten Haupt- und Serverknoten

---

[48] Vgl. Amazon Web Services, Inc. (2013b)

[49] Vgl. Amazon Web Services, Inc. (2013c)

[50] Microsoft (2014c)

berechnet. Die Abrechnung kann nutzungsbasiert geschehen. Optional kann auch der etwas günstigere Sechs- bis Zwölfmonatstarif gewählt werden. Der Preis für einen Hauptknoten beläuft sich auf ca. 350 Euro pro Monat, der eines Serverknotens auf ca. 180 Euro monatlich.[51] Je nach Intensitätsgrad der Nutzung gibt es auch Rabatte.

Die virtuellen Computer werden von Microsoft nach Minuten abgerechnet. Die monatlichen Kosten für eine kleine Instanz (A1) belaufen sich auf ungefähr 50 Euro im Monat, die einer extra großen (A4) hingegen auf knapp 400 Euro.[52]

3.1.3 Rackspace

3.1.1.1 Allgemeines

Bei Rackspace handelt es sich um einen dritten großen Anbieter im Bereich Cloud-Service-Provider im DBaaS-Segment. Im Gegensatz zu Amazon AWS und Windows Azure bietet Rackspace lediglich das Hosting von Servern mit Linux oder Windowssystem, bietet also keine eigene NoSQL-Datenbanklösung oder ähnliches an. Dafür gibt es bei Rackspace zwei verschiedene Cloud-Angebote.

3.1.1.2 Technik

Auf der einen Seite gibt es bei Rackspace die sogenannte Hybrid Cloud, auf der anderen Seite die Managed Cloud. Auch hier hat man wieder die Möglichkeit jegliche NoSQL-Datenbanken zu implementieren und zu konfigurieren.

Die Hybrid Cloud ermöglicht die Bestimmung der Zusammensetzung der Cloud aus Public Cloud, Private Cloud und dedizierten Server um die Cloud-Infrastruktur optimal auf die Bedürfnisse abstimmen zu können.[53] Zusätzlich kann man mit dem „Fanatical Support“ einen Rund-um-die-Uhr-Support buchen.

Die Managed Cloud hingegen übernimmt Aufgaben wie Planung, Überwachung, Support und Optimierung der Cloud-Umgebung. Somit ist man nicht für die Instandhaltung der Infrastruktur verantwortlich.

3.1.1.3 Kosten/Preismodell

Bei Rackspace berechnet sich der Preis anhand des erwünschten Arbeitsspeichers. Dabei wird in Größen von 15GB bis 120 GB unterschieden. Der Kapazität des SSD-Datenträgers

[51] Vgl. Microsoft (2014a)

[52] Vgl. Microsoft (2014b)

[53] Vgl. Rackspace (o. J.a)

reicht von 150GB bis hin zu 1200GB SSD-Speicherplatz. Die Kosten für die angebotenen Pakete werden nach Stundennutzung abgerechnet und belaufen sich bei 730 Stunden monatlicher Nutzung dementsprechend auf 440 Euro bis knapp 3500 Euro.[54]

3.1.4 Joyent

3.1.1.1 Allgemeines

Joyent ist ein weiterer großer Cloud-Service-Provider. Joyent bietet wie die vorherigen drei Anbieter mit der JoyentCloud fertige Server-Instanzen, was eine schnelle und unkomplizierte Bereitstellung ermöglicht. Zudem bietet Joyent auch NoSQL-Datenbankspezifische Leistungen.

3.1.1.2 Technik

Joyent stellt, wie bereits erwähnt, ebenfalls Cloud-Infrastruktur bereit. Die Systeme sind SmartOS, Linux und Windows. Die Services sind dabei in „Joyent Compute Service“, „Joyent Manta Storage Service“ und „Joyent Private Cloud“ gegliedert und unterscheiden sich im Umfang der Leistung.[55]

Bei dem „Joyent Compute Service“ handelt es sich um eine Cloud-Infrastruktur die vor allem belastbare und zuverlässige Anwendungen unterstützen soll. Dabei besteht die Auswahlmöglichkeit aus fünf verschiedenen Instanzen: „Standard“, „High CPU“, „High Memory“, „High Storage“ und „High I/O“. Somit gibt es sowohl für Instanzen für Anwendung mit hohem Durchsatz die den Fokus auf den Arbeitsspeicher legen oder aber auch Instanzen mit SSD-Speicher für Datenbankanwendungen mit hoher Auslastung.[56]

Der „Joyent Manta Storage Service“ ist ein objektbasierter Speicher mit einem integrierten Rechenservice. Der Service speichert die Objekte mittels einer HTTPS REST API in mehreren Rechenzentren. Es handelt sich insgesamt um eine stark parallelisierte Rechenumgebung, die es ermöglicht extrem kurze bis keine Wartezeiten bei Datenmanipulation zu erreichen.[57]

[54] Vgl. Rackspace (o. J.b)

[55] Vgl. Joyent, Inc. (2014a)

[56] Vgl. Joyent, Inc. (2014b)

[57] Vgl. Joyent, Inc. (2014c)

Die „Private Cloud“ von Joyent oder aber auch „Joyent SmartDataCenter™“ genannt, ist das Angebot eines privaten Cloud-Services. Hierbei handelt es sich um eine „customer self-service“ Lösung.[58]

Zusätzlich zur Cloud-Infrastruktur bietet Joyent für die gängigsten NoSQL-Datenbanken „One-Click-Installationen“ inklusive den Server-Instanzen. Dabei handelt es sich um dynamisch skalierbare virtuelle Maschinen, die für Datenbanken wie MongoDB oder Riak schon optimiert wurden.[59]

3.1.1.3 Kosten/Preismodell

Die Kosten eines „Images“ bewegen sich zwischen 0,20 Euro und 4,10 Euro pro Stunde für eine Standardinstanz. Die Preise unterscheiden sich je nach Anzahl der Gibibytes der Instanz. Eine „High Storage“-Instanz kostet zwischen 0,80 Euro und 3,50 Euro.

Joyent bietet eine kostenlose Testversion im Wert von 130 USD. Dieses Angebot beinhaltet zudem den „Manta Storage Service“ mit 10 GB.[60] Allgemein wird der Preis „Manta Storage Service“ von Joyent pro Monat abgerechnet.

3.1.5 Uberspace

3.1.1.1 Allgemeines

Uberspace gehört sicherlich nicht zu den Keyplayern der Cloud-Service-Provider. Vielmehr soll Uberspace hier als ein deutscher Vertreter der Cloud-Anbieter erwähnt werden, der zudem durch sein herausragendes Preismodell auffällt.

3.1.1.2 Technik

Genau wie auch Joyent bietet Uberspace „One-Click-Installationen“ unter anderem für MongoDB und CouchDB mit zugehöriger Infrastruktur. Zudem stellt Uberspace auch Instanzen für den RAM-basierten Key/Value Stores Redis zu Verfügung.[61]

Desweiteren ist das Rechenzentrum erwähnenswert. Die gesamte Server-Infrastruktur befindet sich in Deutschland. Zudem wird darauf geachtet, dass das Rechenzentrum vorwiegend mit Strom aus erneuerbaren Energien betrieben wird.

---

[58] Vgl. Joyent, Inc. (2014d)

[59] Vgl. Borenstein, P. (2013)

[60] Vgl. Joyent, Inc. (2014e)

[61] Vgl. Uberspace (2014)

3.1.1.3 Kosten/Preismodell

Uberspace unterscheidet sich vor allem durch das Preismodell zu den bisherigen DBaaS-Providern. Erstens ist es möglich eine kostenlose Testzeit in Anspruch zu nehmen. Wesentlich interessanter ist jedoch, dass der Kunde selbst über den Preis bestimmen kann. Ab einem Mindestbetrag von 1 Euro pro Monat entscheidet der Nutzer selbst, wie viel ihm der Service, welche Rechenleistung und welcher Speicher, wert ist.[62]

3.1.5 Übersicht

Amazon AWS	Windows Azure	Rackspace	Joyent	Uberspace
Technik: • Fully Managed Non-Relational Service (SimpleDB, DynamoDB) • Amazon Solutions Providers • EC2 Cloud-Server in Kombination mit EBS	**Technik:** • HDInsight-Dienst - Hadoop basiert • Virtual Machines	**Technik:** • Managed Cloud • Hybrid Cloud	**Technik:** • Compute Service • Manta Storage Service • Private Cloud	**Technik:** • One-Click-Installationen • Redis-Instanzen (Key/Value Store)
Kosten: • DynamoDB: • Kostenloser Einstieg bis 100MB • 0,20€/GB/ Monat • EC2: • Kostenloser EInstieg (1 Jahr bis zu 750h/ monatl.) • Anschließend Abrechnung nach Stunden und Speicher	**Kosten:** • HDInsight-Dienst • Hauptknoten 350€/Monat • Serverknoten 180€/Monat • Virtueller Computer • A1-Instanz 50€/ Monat • A4-Instanz 400€/Monat	**Kosten:** • 15GB Performance/ 150GB SSD 440€ • 120GB Performance/ 1200GB SSD 3500€	**Kosten:** • Joyent Image: • Standard: 0,2-4,1€/h • High Performance: 0,8-3,5€/h • kostenlose Testversion im Wert von $130	**Kosten:** • Frei wählbar • mind. 1€/Monat

Tab. 2: Übersicht allgemeine Cloud-Provider

[62] Vgl. Uberspace (2014)

3.2 Spezielle DBaaS-Anbieter

3.2.1 Object Rocket

3.2.1.1 Allgemein

Der erste Vertreter für MongoDB Hosting ist Object Rocket. Der aus Austin, Texas stammende Anbieter hat sich auf der Hosting des populären Document Stores spezialisiert und gilt als einer der ersten am Markt.[63] Obwohl Object Rocket seit Februar 2013 zu Rackspace gehört[64], lässt sich alternativ auch Amazon AWS als Cloud-Infrastruktur auswählen.
Zwar leistet Object Rocket im Gegensatz zu vielen Konkurrenten seinen Support selbst und bezieht ihn nicht direkt von MongoDB Inc., trotzdem wird hier 24x7x365 Verfügbarkeit garantiert.[65]

3.2.1.2 Technik

Technologisch sind bei Object Rocket vor allem folgende Punkte wichtig: Neben der speziell auf MongoDB abgestimmten Server-Architektur und Ausstattung sowie Konfiguration des Betriebssystems sorgen vor allem die standardmäßige Ausrüstung der Maschinen mit Solid State Disks für hohe Performance.[66]

Geographisch kann für das Hosting zwischen insgesamt 3 Regionen gewählt werden. Zusätzlich sorgt das sogenannte "sharding" (von "Scherben/Bruchstücke") für noch bessere Skalierbarkeit.[67] Hierbei wird ein gesamter Server Cluster in einzelne „Shards“ aufgeteilt. Jeder dieser Server-Instanz wird dann ein Shard-Key zugewiesen, mit Hilfe dessen später einfach weitere Instanzen hinzugeschaltet werden können.[68]

Bezüglich Security bietet Object Rocket automatische Backups sowie - je nach Bedarf - eine Replizierung über mehrere Regionen hinweg. Wichtig hierbei: Entsprechende Replizierungsdaten werden verschlüsselt übertragen.[69]

Softwareseitig werden von Object Rocket zwei wesentliche APIs angeboten: Zum einen kann die MongoDB Instanz so verwaltet und administriert, eigene Tools und Apps geschrieben und bestimmte Vorgänge automatisiert werden. Auf der anderen Seite existiert eine Monito-

[63] Vgl. Object Rocket (2012a)
[64] Vgl. Object Rocket (2012b)
[65] Vgl. Object Rocket (2012a)
[66] Vgl. ebenda
[67] Vgl. ebenda
[68] Vgl. MongoDB (2013a)
[69] Vgl. Object Rocket (2012a)

ring API, über die aktuelle Nutzer-Statistiken, Auslastung, Fehlerberichte und ähnliches abgerufen werden können.[70]

Natürlich steht zusätzlich aber auch ein von Object Rocket eigens zu diesem Zweck entwickeltes Web Interface zur Verfügung, was zur Verwaltung und zum Monitoring verschiedener Instanzen verwendet werden kann.[71]

3.2.1.4 Kosten/Preismodell

Das Preismodell von Object Rocket lässt vor allem auf extreme Skalierbarkeit schließen und unterteilt sich wie folgt[72], dabei wird grundsätzlich in drei simple Kategorien unterschieden: Small, Medium und Large. Der Small-Tarif richtet sich mit einem Preis von 19USD/GB/Monat und einer maximalen Größe von 5GB vorrangig an Privatanwender, Startups und sehr kleine Unternehmen. Zusätzlicher Unterschied sind eine mangelnde SSL-Verfügbarkeit, kein optionales Sharding und der ledigliche 8x5 Support mit einem Tag Antwortzeit. Wichtigster Punkt ist allerdings die Begrenzung auf maximal eine Datenbank. Eine weitere Besonderheit des Small-Plans: Es kann ein unverbindlicher, 30-tägiger Test erfolgen.

Der Medium-Plan skaliert von 149USD/Monat für 5GB bis 1599USD/Monat für 100GB. Hier sind sämtliche Small-Beschränkungen aufgehoben. Das heißt, es gibt neben weiteren Features 24x7 Support, eine unbegrenzte Anzahl von Datenbanken, SSL sowie die Möglichkeit zum Sharding. Dieser Tarif ist vor allem für datenintensive Startups sowie für Klein- bis Mittelständler geeignet.[73]

Der letzte und größte Tarif ist sehr individuell und lässt sich nahezu frei ausgestalten. Möglich sind hier Datenmengen von 100GB bis 10PB. Neben allen Features der anderen Tarife sind Beratungs- und Migrationsdienstleistungen möglich. Außerdem kann der zuvor verhandelte Preis (keine Standardpreise) auf Rechnung bezahlt werden.[74]

3.2.2 mongolab

3.2.2.1 Allgemein

Wie die meisten seiner Konkurrenten stammt auch mongolab aus den USA, konkret aus San Francisco, Kalifornien.[75] Ähnlich wie bei Object Rocket handelt es sich auch hier um einen

[70] Vgl. Object Rocket (2012a)

[71] Vgl. ebenda

[72] Vgl. Object Rocket (2012c)

[73] Vgl. ebenda

[74] Vgl. ebenda

[75] Vgl. mongolab (2014a)

etablierten Anbieter, der seine eigene Schicht über bereits bestehende und zuverlässige Cloud-Services legt, um dediziertes MongoDB Hosting anzubieten.

Zur Auswahl stehen hier neben den obligatorischen Amazon Web Services auch Rackspace, Joyent, Google oder Windows Azure.[76]

Ein wichtiger Unterschied zu Object Rocket ist der offizielle MongoDB Inc. Support mit einer vereinbarten Reaktionszeit von einer Stunde 24x7.[77]

3.2.2.2 Technik

Über die tatsächliche Technik und Ausstattung der Server erfährt man bei mongolab relativ wenig. Lediglich die optionale Möglichkeit zu SSD Festplatten wird deutlich.[78] Die wichtigsten Features sind sicherlich die Regionen-Redundanz (allerdings nur bei Wahl von AWS), APIs und Monitoring. Als API bzw. externer Zugriff bietet mongolab grundsätzlich zwei Möglichkeiten: Zum einen können die offiziellen MongoDB-Treiber für die jeweilige Programmiersprache verwendet werden, was auch die von mongolab empfohlene Vorgehensweise ist. Es existieren beispielsweise Interfaces für C#, Java, PHP und weitere. Andererseits gibt es eine von mongolab entwickelte REST API, über die die Datenbank-Instanz ebenso verwaltet und abgerufen werden kann.[79]

Interessant ist hier auch das Monitoring-System, was eine grafische Auswertung ermöglicht sowie die damit einhergehenden Alarmfunktionen für Leistungsspitzen oder Ausfälle. Obligatorische Backups und ein Web Interface zur Administration sind ebenfalls vorhanden.[80]

3.2.2.3 Kosten/Preismodell

Grundsätzliche Kategorisierung im Preismodell ist Shared bzw. Dedicated Plan. Bei den Shared Tarifen teilt man sich Leistung und Rechenkapazität eines Servers mit verschiedenen anderen Nutzern. Außerdem kann man nicht frei zwischen den verschiedenen Cloud-Partnern wählen. Die Dedicated Pläne bescheren einen eigenen Server sowie die Möglichkeit, den passenden Cloud-Anbieter (wie Rackspace oder AWS) selbst zu wählen. Vorteilhaft hier ist vor allem, dass die Datenbank ggf. im selben Rechenzentrum wie die Applikation liegen kann, um Latenzen zu verringern und Performance zu steigern.[81]

---

[76] Vgl. mongolab (2014b)

[77] Vgl. mongolab (2014c)

[78] Vgl. mongolab (2014d)

[79] Vgl. mongolab (2014b)

[80] Vgl. ebenda

[81] Vgl. mongolab (2014d)

Bei den Shared Tarifen existiert ein kostenloser Plan mit 0,5GB Speicher. Außerdem werden zwei 2GB-Tarife angeboten, von denen einer mit 15USD/Monat und der andere mit 89USD/Monat und einer zweiten virtuellen Maschine aufwartet. Die Pläne sind außerdem optional für 5USD/GB/Monat erweiterbar.[82]

Im Dedicated Bereich reichen die Pläne - je nach Cloud-Anbieter - von 200USD/Monat für 40GB bis über 5.000USD/Monat für Pläne mit über 1TB Speicher. Außerdem kann mit AWS Backend ein individueller Tarif geschnürrt werden.[83]

3.2.3 MongoHQ

3.2.3.1 Allgemein

MongoHQ stammt ebenso wie Konkurrent mongolab aus Kalifornien, diesmal aus San Mateo.[84] Je nach ausgewähltem Tarif basiert der Service auf AWS, IBMs SoftLayer oder Heroku. Für bestimmte Pläne ist auch hier ein offizieller MongoDB Inc. Support 24x7 verfügbar, ansonsten werden entsprechende Anfragen selbst beantwortet.[85]

3.2.3.2 Technik

Auch MongoHQ hält sich im Vergleich zu einigen Mitbewerbern relativ bedeckt bezüglich verwendeter Hardware und anderen technischen Aspekten. Dennoch kann auch hier wieder die zwischen herkömmlichem und schnellerem SSD Speicher gewählt werden.[86]

Softwareseitig gibt es allerdings ein umfangreiches Featureset[87]: Neben den ohnehin obligatorischen Backups stehen verschiedene Tools zum Monitoring und zur Verwaltung der Datenbanken zur Verfügung: Skalierbarkeit wird gewährleistet, indem RAM, Durchsatz und Weiteres per "Klick" erhöht werden können, ohne die Datenbank Instanz abschalten oder herunterfahren zu müssen.[88]

---

[82] Vgl. ebenda

[83] Vgl. ebenda

[84] Vgl. MongoHQ (o. J.a)

[85] Vgl. MongoHQ (o. J.b)

[86] Vgl. ebenda

[87] Vgl. ebenda

[88] Vgl. MongoHQ (o. J.c)

Außerdem existiert neben dem herkömmlichen Monitoring (inkl. Alarmfunktionen) ein sogenanntes Performance Toolset, um Flaschenhälse zu identifizieren und die Leistung, Stabilität und Geschwindigkeit der MongoDB Datenbank zu steigern und zu optimieren.[89]

Auch ein ausgeklügeltes Database Life Cycle Management erleichtert die Arbeit und beinhaltet Funktionalitäten wie Teamkollaboration, verschiedene Datenbankzustände wie beta, production oder post-production und Ähnliches.[90]

Das Web Interface zur Administration und Verwaltung der Daten ist sehr stark an MongoDBs Kommandozeile angelehnt, wodurch vor allem erfahrene MongoDB-Anwender hier vermutlich schneller arbeiten können als mit einem klassischen "Klick-Web-Interface". Für Neueinsteiger könnte dies jedoch zunächst eine Hürde sein.[91]

3.2.3.3 Kosten/Preismodell

Das Preismodell von MongoHQ basiert auf drei Modellen[92]: Shared Server mit klassischem Speicher, Shared Server mit SSD Speicher und dediziert auf AWS Instanzen.

Die erste Variante beinhaltet einen kostenfreien Tarif mit 512MB Speicher sowie zwei weitere mit bis zu 5GB für 49USD/Monat.[93] Im SSD Bereich kann zwischen 4GB für 100USD/Monat und 25GB für 500USD/Monat gewählt werden. Ansonsten unterscheiden sich die bisherigen Pläne nicht.[94]

Das dedizierte AWS Hosting richtet sich an große und datenintensive Unternehmen und bietet Tarife von 70GB für 250USD/Monat bis 600GB für 6545USD/Monat. Außerdem erhält der Nutzer hier den bereits erwähnten MongoDB Inc. Support.[95]

Im Vergleich zu den anderen MongoDB-Providern fällt eine relativ restriktive Tarifstruktur auf, vor allem im Einsteiger-Segment.

---

[89] Vgl. ebenda

[90] Vgl. ebenda

[91] Vgl. ebenda

[92] Vgl. MongoHQ (o. J.b)

[93] Vgl. ebenda

[94] Vgl. ebenda

[95] Vgl. ebenda

3.2.4 mongosoup

3.2.4.1 Allgemein

Der wahrscheinliche Ausreißer im Bereich der MongoDB-Hoster ist mongosoup. Der einzige nicht aus den USA stammende Anbieter kommt aus Deutschland, genauer aus München und bietet deshalb neben ähnlicher technischer und preislicher Ausgestaltung vor allem den Mehrwert deutscher bzw. europäischer Datenschutzrichtlinien.[96]

Dennoch kann auch hier neben einem tatsächlichen (shared) Hosting im deutschen Rechenzentrum und AWS Hosting in Irland (shared & dediziert) gewählt werden.[97]

Weitere Besonderheit ist die Inanspruchnahme eines Inhouse-Service, um mongosoup Technologie und Betreuung im eigenen Rechenzentrum einzusetzen, was vor allem für größere Unternehmen interessant sein dürfte. MongoDB Inc. Hosting ist diesmal bei allen Tarifen inkludiert.[98]

3.2.4.2 Technik

Leider erfahren wir auch hier wieder sehr wenig über die technischen Gegebenheiten von mongosoup. Lediglich das Vorhandensein automatischer Backups und die Möglichkeit zum Sharding, also die Skalierbarkeit über mehrere Server hinweg, werden als relevante technische Features genannt.[99]

3.2.4.3 Kosten/Preismodell

Wie bereits erwähnt unterscheiden sich die Tarife in deutsches shared Hosting, shared AWS Hosting in Irland und dedizierte Server.[100]

Die vier deutschen Tarife decken 2GB Speicher für 15Euro/Monat bis 10GB für 70Euro/Monat ab. Die Preise für das shared AWS Hosting sind identisch.

Im Bereich der dedizierten Server reicht das Portfolio von 10GB für 200Euro/Monat bis 100GB für 1600Euro/Monat. Außerdem wird hier ein zusätzlicher 8x5 Support von mongosoup geboten. Neben dem Speicher sind im Bereich der dedizierten Server auch Arbeits-

[96] Vgl. mongosoup (o. J.a)

[97] Vgl. mongosoup (o. J.b)

[98] Vgl. mongosoup (o. J.c)

[99] Vgl. ebenda

[100] Vgl. mongosoup (o. J.b)

speicher und Prozessor entsprechend angepasst und reichen von 1,7GB RAM und 1 CPU Kern bis hin zu 15GB RAM mit 8 CPU Kernen.[101]

Durch die vergleichsweise teuren Preise ist mongosoup vor allem für diejenigen Unternehmen interessant, die Wert auf ein Hosting in deutschen oder zumindest in europäischen Rechenzentren liegen. Dies sind in erster Linie Unternehmen mit datenschutzrelevanten bzw. sehr sensitiven Daten.

3.2.5 IrisCouch

3.2.5.1 Allgemein

Erster Vertreter aus dem Bereich des CouchDB Hostings ist der Anbieter IrisCouch aus Los Angeles, Kalifornien.[102] Obwohl mit Cloudant ein weiterer Anbieter unter diese Kategorie gefasst wurde, ist IrisCouch der einzige eigentliche CouchDB-Provider, wie sich später noch zeigen wird.

Es ist nicht ersichtlich, ob IrisCouch eigene Rechenzentren bzw. Server betreibt oder die Infrastruktur eines renommierten Anbieters wie AWS oder Rackspace verwendet. Der Fokus liegt hier ganz klar auf der Funktionalität und Ausstattung der Software bzw. des Service.

3.2.5.2 Technik

Wie bereits erwähnt, liegt der Fokus hier auf der Funktionalität und Ausstattung der Software[103]: Der Nutzer erhält eine oder mehrere CouchDB-Instanzen, die er über ein Web Interface verwalten kann. Es existiert eine REST API, welche die Synchronisation mit Servern und Clients (z.B. Apps) erleichtern soll.

Wichtigstes Alleinstellungsmerkmal von IrisCouch ist jedoch die Unterstützung des GeoCouch-Plugins.[104] Damit ist es möglich, ortsbezogene Daten - in aller Regel Koordinaten - speziell abzuspeichern und vor allem abzufragen. Dies ist besonders dann interessant, wenn geografische Apps Suchen wie "in der Nähe" oder "im Umkreis von" durchführen.

3.2.5.3 Kosten/Preismodell

Auch das Geschäfts- bzw. Preismodell von IrisCouch unterscheidet sich von den MongoDB Vertretern und ist extrem nutzenabhängig. Grundsätzlich gilt[105]:

---

[101] Vgl. mongosoup (o. J.b)

[102] Vgl. IrisCouch (2010a)

[103] Vgl. ebenda

[104] Vgl ebenda

[105] Vgl. IrisCouch (2010c)

- 1USD/GB/Monat
- 1 Cent für 500 Lesezugriffe
- 1 Cent für 100 Schreibzugriffe
- 2 Cent für obige Zugriffe mit SSL

Wichtig hierbei ist, dass die Nutzung unter einem Gesamtbetrag von 5USD kostenfrei ist, was besonders für kleinere Unternehmen oder bei unregelmäßigen Spitzen interessant sein dürfte.

3.2.6 Cloudant

3.2.6.1 Allgemein

Einer der umfangreichsten Anbieter von NoSQL-DBaaS-Hosting ist Cloudant aus Cambridge, Massachusetts. Der von drei MIT Physikern gegründete Dienst basiert zwar auf der Apache CouchDB, verwendet jedoch eine eigene Implementierung und Erweiterung dessen.[106]

Als Cloud-Backend stehen mit AWS, IBM SoftLayer, Rackspace, Joyent sowie Windows Azure eine ganze Reihe Hosting-Partner zur Verfügung. Außerdem kollaboriert Cloudant mit einigen Platform-as-a-Service Anbietern wie Heroku.[107]

Der Support wird von Cloudant selbst geleistet und ist 24x7 verfügbar und kann gegen Aufpreis zum "Gold Support" aufgewertet werden, der unter anderem den Vorteil einer 1-stündigen Reaktionszeit bietet.[108]

3.2.6.2 Technik

Je nach gewähltem Cloud-Anbieter (AWS, Rackspace, etc.) stehen unterschiedliche Hardware-Konfigurationen zur Auswahl. Der Vorteil von Cloudant liegt aber nicht in der verwendeten Hardware sondern vor allem in Funktionalität, Ausstattung und Leistung des Services bzw. des angeboteten Software-Layers.

Neben einem Webinterface für die Administration sowie einem umfangreichen Monitoring-System, stehen viele spezielle Funktionen und Prozesse zur Vereinfachung und Perfomance-Steigerung der Datenbank zur Verfügung[109]: Sync, MapReduce, Search und Geospatial sind die wichtigsten Vertreter.

---

[106] Vgl. Cloudant (2013a)

[107] Vgl. Cloudant (2013b)

[108] Vgl. Cloudant (2013c)

[109] Vgl. Cloudant (2013d)

Cloudants Sync System ermöglicht - ähnlich wie IrisCouch - eine einfache Synchronisierung und Einbeziehung vieler, vor allem mobiler Endgeräte, indem jeweils eine lokale Kopie der Datenbank erzeugt und entsprechend synchronisiert wird.[110]

MapReduce und Search dienen vor allem der Performance Steigerung von Abfragen, indem bestimmte Views mit weniger Attributen[111] für wiederholte Suchanfragen sowie Indizes[112] intelligent erstellt und verwaltet werden.

Genau wie der CouchDB-Mitbewerber IrisCouch bietet auch Cloudant eine integrierte Möglichkeit zur Speicherung, Auswertung und Verarbeitung von Geodaten.[113]

Ein weiterer, aufgrund der Komplexität und des Funktionsumfangs definitiv sinnvoller Punkt, sind die angeboteten Trainings und Beratungsdienstleistungen.[114]

3.2.6.3 Kosten/Preismodell

Auch im Preismodell ähnelt Cloudant sehr seinem Konkurrenten IrisCouch[115]: Grundsätzlich sind die ersten 30 Nutzungstage kostenlos, anschließend wird erst - wie bei IrisCouch - ab einer Ausschöpfung von 5USD/Monat eine Zahlung fällig.

Die nutzungsabhängige Zusammensetzung der Kosten ergibt sich bei Cloudant folgendermaßen[116]:

- 1USD/GB/Monat
- 15 Cent für 500 Lesezugriffe
- 15 Cent für 100 Schreibzugriffe

Die Preise sind also minimal teurer als bei IrisCouch, allerdings wird keine Unterscheidung zwischen SSL und nicht SSL vorgenommen. Preise für einen dedizierten Cloudant-Server sind nur auf Anfrage erhältlich.

---

[110] Vgl. Cloudant (2013e)

[111] Vgl. Cloudant (2013f)

[112] Vgl. Cloudant (2013g)

[113][113] Vgl. Cloudant (2013c)

[114] Vgl. Cloudant (2013h)

[115] Vgl. Cloudant (2013i)

[116] Vgl. ebenda

3.2.7 instaclustr

3.2.7.1 Allgemein

Neben den Vertretern der dokumenten-basierten NoSQL Hosting-Anbietern für MongoDB und CouchDB ist der auf Apaches Cassandra basierende Dienst instaclustr der erste Anbieter eines Key-Value Hostings. instaclustr ist ein eigenständiges Unternehmen und hat seinen Hauptsitz in Canberra, Australien.[117]

Momentan wird als Infrastruktur nur AWS oder Rackspace angeboten, jedoch ist laut Anbieter-Information die Unterstützung für Windows Azure, Joyent, Google sowie eigenes Hosting in Arbeit.[118]

Es wird ein Basis Support gewährleistet, konkrete Angaben gibt es dazu aber nicht.

3.2.7.2 Technik

Technisch können je nach Cloud-Infrastruktur herkömmliche oder SSD Speicher gewählt werden. Ansonsten ist über die technischen Gegebenheiten relativ wenig bekannt, aufgrund von renommierten Anbietern wie AWS oder Rackspace allerdings auch von eher untergeordneter Rolle.

Ansonsten besteht der Mehrwert von instaclustr vor allem in der Installation, Konfiguration und dem Leistungstuning der Cassandra-Instanz auf dem jeweilig gewählten, von instaclustr administrierten, Server.[119]

Weiterhin werden ein automatische Monitoring sowie regelmäßige Backups angeboten.
Ein spezielles Webinterface oder eine API ist allerdings nicht verfügbar, was die den Mehrwert von instaclustr für viele Nutzer vermutlich erheblich einschränkt.[120]

3.2.7.3 Kosten/Preismodell

Die Preise orientieren sich sehr stark an den entsprechenden Instanzen der Cloud-Anbieter und richten sich primär an größere Kunden.

Während auf AWS sechs Pläne von 400GB für 199USD/Monat mit herkömmlichem Speicher bis zu 2TB SSD Speicher für 2999USD/Monat möglich sind, sind bei Rackspace 4 Pläne von 160GB für 299USD/Monat bis 1,2TB für 1799USD/Monat möglich, allerdings beides ohne

[117] Vgl. instaclustr (2013a)

[118] Vgl. instaclustr (2013b)

[119] Vgl. instaclustr (2013c)

[120] Vgl. ebenda

SSD.[121] Entsprechend analog verlaufen die Werte für Arbeitsspeicher und Anzahl der CPU Kerne.[122]

3.2.8 Garantia redis

3.2.8.1 Allgemein

Die Redis Cloud von Garantia ist die einzige relevante Lösung für das Hosting der In-Memory-Datenbank redis.

Das aus Santa Clara, Kalifornien stammende (Big) Data Unternehmen Garantia[123] arbeitet hier mit den Cloud-Providern AWS, IBM SoftLayer und Windows Azure zusammen.[124]
Support ist unter anderem durch eine kostenlose 24x7 Hotline gegeben.

3.2.8.2 Technik

Garantia hat ein, gerade im Bereich der In-Memory-Datenbanken wichtiges, eigenes System zum automatischen sharding entwickelt, um eine hohe Skalierbarkeit durch den Zusammenschluss von Servern zu gewährleisten. Dadurch erreicht Garantia mit eigenen Angaben eine unerschöpfliche Arbeitsspeicher Kapazität.[125]

Durch die Verteilung der Daten auf mehrere Server wird außerdem eine einfache und schnelle Wiederherstellung im Falle eines Crashs ermöglicht. Hervorzuheben ist, dass die bei redis relativ aufwändige Installation, Konfiguration und Skalierung vollständig von Garantia übernommen wird.[126]

3.2.8.3 Kosten/Preismodell

Bedingt durch die Tatsache, dass es sich bei redis um eine In-Memory-Datenbank handelt, sind die Kosten pro GB nur schwer mit anderen Systemen zu vergleichen.[127]

Dennoch gibt es sowohl bei AWS als auch bei Azure mit 25MB ein kostenloses Basis-Paket. Der größte Standard-Tarif ist mit 5GB Arbeitsspeicher ausgestattet und kostet auf AWS 375USD/Monat und auf Windows Azure 520USD/Monat.[128] Da sich die SoftLayer Variante

---

[121] Vgl. instaclustr (2013b)
[122] Vgl. ebenda
[123] Vgl. Garantia (2013a)
[124] Vgl. Garantia (2013b)
[125] Vgl. Garantia (2013c)
[126] Vgl. Garantia (2013d)
[127] Vgl. Garantia (2013b)
[128] Vgl. Garantia (2013b)

offenbar noch im Beta-Status befindet, kann der Nutzer hier momentan eine kostenlose Instanz mit einer Größe von 1GB erhalten.[129]

3.2.9 GrapheneDB

3.2.9.1 Allgemein

Der noch relativ junge Service GrapheneDB spezialisiert sich auf das Hosting der Graph-Datenbank Neo4j und ist ein Produkt der aus Las Palmas, Spanien stammenden Entwickler-Gruppe aentos.[130]
Auch hier findet wie bei den meisten anderen NoSQL-Hostern die AWS Cloud Verwendung.[131]

GrapheneDB befindet sich noch im Beta-Stadium, kann aber grundsätzlich schon produktiv eingesetzt werden. Deshalb steht auch bereits ein Basis Support zur Verfügung.[132]

3.2.9.2 Technik

GrapheneDB bietet neben den bereits erwähnten Aspekten einer Graph-Datenbank allgemein bzw. von Neo4j im speziellen einige zusätzliche Vorteile[133]: Die Einrichtung und Konfiguration ist vollständig automatisiert. Außerdem kann die Administration der Datenbank und der Daten über ein eigenes Webinterface erfolgen oder die API von Neo4j verwendet werden.

Durch die Unterstützung von Plugins die von der Community bereitgestellt werden, lassen sich bestimmte Funktionalitäten einfach hinzufügen bzw. erweitern. Datenbanken müssen bei Wachstum nicht migriert werden, da Skalierung in punkto Speicher und Leistung von Anfang an möglich ist.[134] Tägliche Backups und ständiges Monitoring sind ebenfalls gewährleistet.

3.2.9.3 Kosten/Preismodell

Zum Testen des Service bietet GrapheneDB einen Sandbox Account, bei dem man - einmal registriert - 512MB Speicher und 256MB Arbeitsspeicher kostenlos erhält.[135]

---

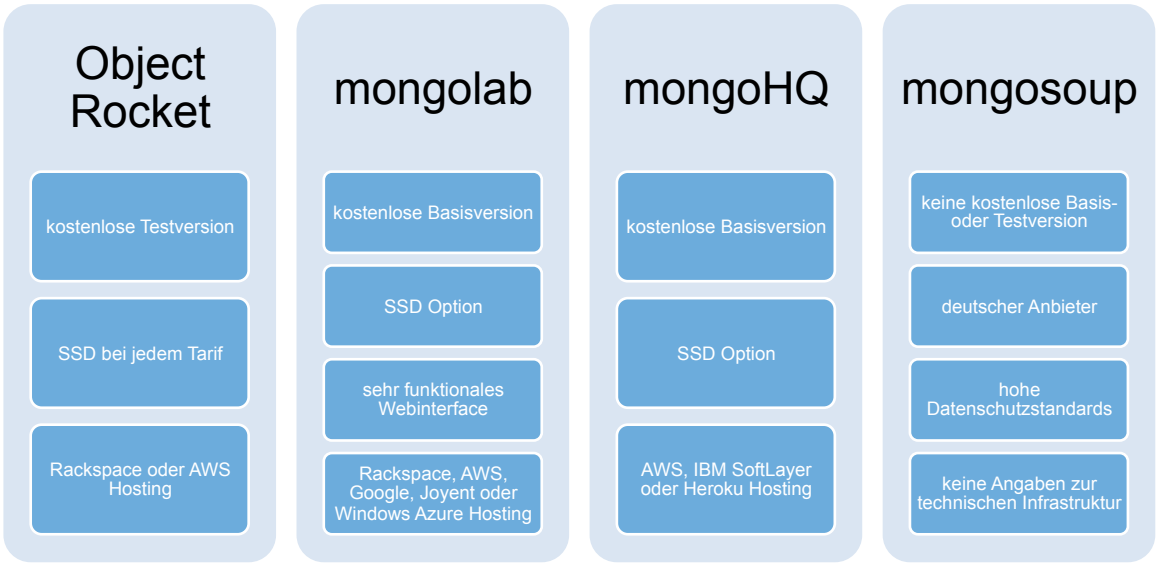
[129] Vgl. ebenda
[130] Vgl. Aentos (o. J.a)
[131] Vgl. Aentos (o. J.b)
[132] Vgl. Aentos (o. J.a)
[133] Vgl. Aentos (o. J.a)
[134] Vgl. ebenda
[135] Vgl. Aentos (o. J.b)

Darüber hinaus gibt es vier vorkonfigurierte AWS-Instanzen von 20GB Speicher mit 1,7GB RAM bis 160GB Speicher mit 15GB RAM. Die entsprechende AWS-Instanz kann in einer beliebigen Region gehostet werden, jedoch stehe für die vier genannten Pläne keine Preise zur Verfügung, diese werden nur auf Anfrage herausgegeben.[136]

3.2.10 Übersicht

3.2.10.1 MongoDB Anbieter

Die folgende Übersicht zeigt die vier wesentlichen MongoDB-Clou- Anbieter, auf die auch im Rahmen dieser Arbeit eingegangen wurde. Da technologisch und leistungsmäßig alle Anbieter ein ähnliches Niveau haben, sind vor allem der Preis oder persönliche Präferenzen entscheidend bei der Auswahl. Mongosoup ist insofern eine Ausnahme, da es sich um einen deutschen Anbieter und so neben AWS auch Hosting im deutschen Rechenzentrum möglich ist.

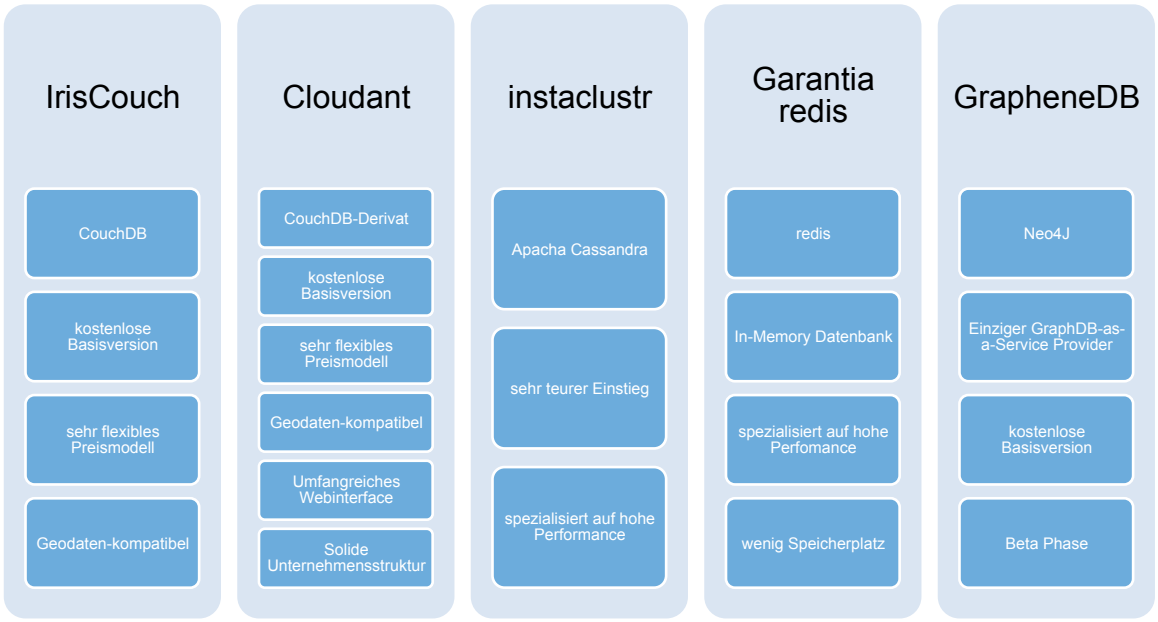


Tab. 3: MongoDB-Provider Übersicht

3.2.10.2 Sonstige Anbieter

Nachfolgend sind die neben den verschiedenen MongoDB-Anbietern anderen NoSQL-Cloud- Anbieter aufgelistet. Neben den zwei CouchDB-Providern IrisCouch und Cloudant ist mit instaclustr ein Cassandra und mit GrapheneDB ein Neo4j-Hoster aufgeführt. Instaclustr hat hier aufgrund des sehr speziellen Angebots vermutlich die geringste Relevanz.

[136] Vgl. ebenda



Tab. 4: Sonstige NoSQL-Provider Übersicht

4 Kriterien

Die vorrangegangene Evaluierung und Beschreibung der allgemeinen und speziellen DBaaS-Provider soll nun in einen Kriterienkatalog überführt werden, um die Vergleichbarkeit zwischen den einzelnen Systemen und Anbietern herzustellen.

Der Kriterienkatalog untergliedert sich in die vier Kategorien Allgemein, Datenstruktur, Technologie und Wirtschaftlichkeit. Im allgemeinen Teil soll auf den Anbieter und dessen Leistungen an sich sowie besondere Features eingegangen werden. Mit Hilfe der Kategorie Datenstruktur soll der bestgeeignetste NoSQL-Typ (z.B. Document Store) bzw. das bestgeeignetste Datenbanksystem bestimmt werden. Dazu wird vorrangig auf die Struktur und Art der später zu verwendenden Daten eingegangen.

Im Bereich der Technologie werden die Aspekte Performance, Speicherplatz sowie Funktionalität und Verwaltung eingegangen. Zudem soll der benötigte Konfigurierbarkeitsgrad näher bestimmt werden.

Der wirtschaftliche Teil deckt ergänzend die Aspekte Kosten/Nutzen-Verhältnis sowie unterschiedliche Preis- und Geschäftsmodelle ab.

Folgende fünf Fragestellungen je Kategorie haben sich basierend auf den vorangestellten Nachforschungen herauskristallisiert:

Allgemein

Wie wichtig ist Ihnen eine solide Unternehmensstruktur des Anbieters?
Wie wichtig ist Ihnen Datenschutz?
Wie wichtig ist Ihnen Support?
Wie wichtig ist Ihnen ein umfangreiches Webinterface und Monitoring?
Wie wichtig ist Ihnen eine umfangreiche API?

Datenstruktur

Wie wichtig ist Ihnen die Möglichkeit zur Verwaltung stark vernetzter und komplexer Datensätze wie Netzwerk-Infrastruktur, Social Media?
Wie wichtig ist Ihnen die Möglichkeit zur Verwaltung von Geodaten?
Wie wichtig ist Ihnen der Einsatz von OLAP-Anwendungen?
Wie wichtig ist Ihnen die Möglichkeit, viele Datensätze und viel Traffic bewältigen zu können?
Wie wichtig ist Ihnen die Verwaltung von Benutzerprofilen bzw. Sessions?

Technologie

Wie wichtig ist Ihnen eine einfache und schnelle Skalierung?

Wie wichtig ist Ihnen Speicherplatz?

Wie wichtig ist Ihnen Performance?

Wie wichtig ist Ihnen ein ready-to-run System?

Wie wichtig ist Ihnen Freiheit bei Installation, Konfiguration und Optimierung?

Wirtschaftlich

Wie wichtig ist Ihnen ein kostengünstiger Anbieter?

Wie wichtig ist Ihnen ein flexibles Preismodell?

Wie wichtig ist Ihnen eine flexible Vertragslaufzeit?

Wie wichtig ist Ihnen eine kostenlose Basisversion?

Wie wichtig ist Ihnen eine kostenlose Testversion?

Ziel ist es, mit Hilfe dieser Fragen eine Möglichkeit der Vorauswahl für Unternehmen bei der Auswahl des individuell passenden Anbieters zu bieten.

5 Hands-On

Dieses Kapitel soll anhand einiger zuvor beschriebenen Cloud-Provider zeigen, wie eine entsprechende NoSQL-Datenbank eingerichtet werden kann. Dabei wird der Fokus auf das Anlegen einer Datenbank sowie das Einfügen von Daten gelegt.

5.1 Dynamo DB – Amazon AWS

Die Key/Value-Datenbank DynamoDB von Amazon wird über die Webseite von Amazon AWS zugänglich gemacht. Der Link zur DynamoDB befindet sich direkt auf der „AWS Management Console“. Der Service ist so gestaltet, dass man nach dem Anlegen eines Amazon AWS Benutzerkontos die Datenbank mit Hilfe weniger Schritte erstellen kann. Zudem ist der Einstieg für DynamoDB wie bereits erläutert bis 100MB kostenlos.

Über den Link zur DynamoDB gelangt man auf die Startseite der Dynamo DB. Bevor man beginnt die Datenbank anzulegen hat man zunächst die Möglichkeit den Standort des Servers zu bestimmen, auf dem die Anwendung laufen soll. In Europa wird als Standard Irland angeboten. Es gibt jedoch auch die Möglichkeit Server an verschiedenen Standorten in den USA, Asien oder Südamerika zu wählen.

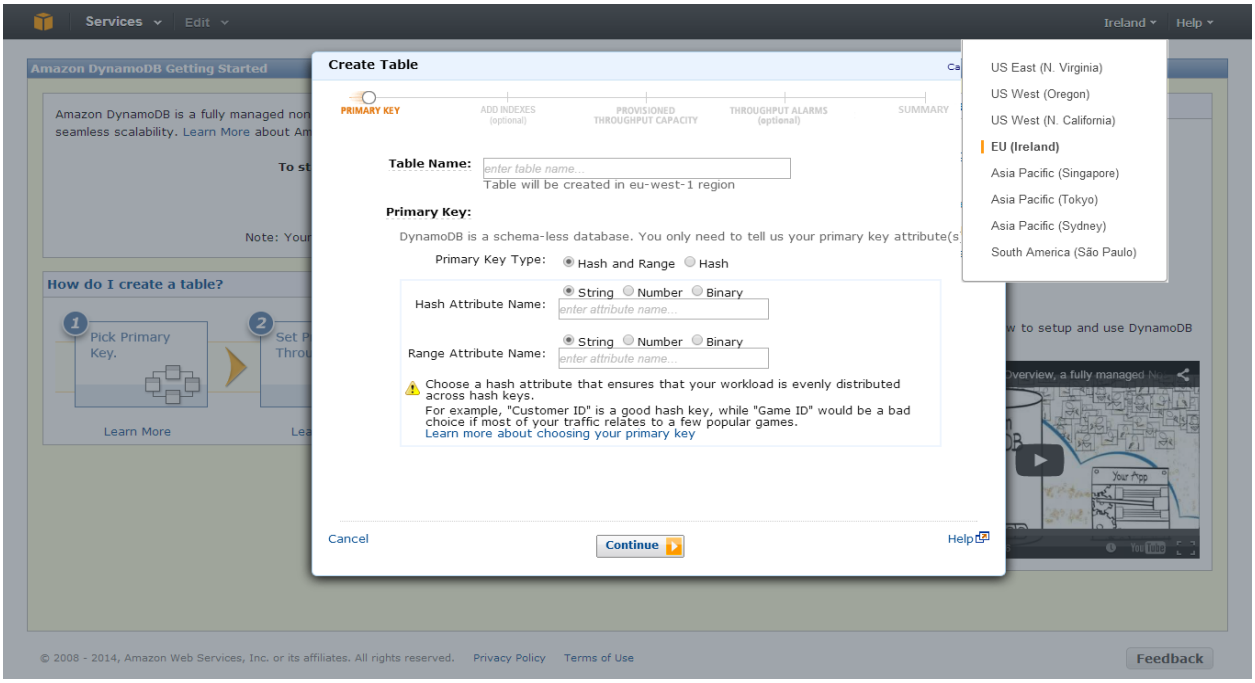


Abb. 3: „Create Table“-Maske[137]

[137] Entnommen aus: Amazon Web Services, Inc. (o.J.)

Ist die Standortwahl erfolgt kann mit dem Anlegen der ersten Tabelle begonnen werden. Durch die Auswahl des „Create Table“-Links öffnet sich die „Create Table“-Maske (siehe Abb. 3). Hier gilt es die Daten der ersten Tabelle anzugeben. Dazu gehören neben dem Tabellenname auch noch die Definition spezifischer Einstellung bezüglich des Primärschlüssels. Der sogenannte „Hash Key“ sollte dabei so gewählt werden, dass die Auslastung gleichmäßig verteilt wird. Dafür sollte ein Wert für ein Objekt gewählt werden welches recht häufig vorkommt. Er sollte also nicht für einen Sammelbegriff festgelegt werden. Nachdem diese zwei Punkte bearbeitet wurden, können noch spezifische Einstellungen zur Indexierung und zum „Throughput“-Management getroffen werden (siehe Abb. 4) Nach dem Bearbeiten dieser Schritte wird die Tabelle bereits erstellt. Der Erstellvorgang kann dabei wenige Minuten dauern. Der gesamte Prozess ist auch im Anhang 1 zu sehen.

Abb. 4: Schritte zur Datenbankerstellung[138]

Das Anlegen von einzelnen Datensätzen ist von Amazon ebenfalls sehr einfach gestaltet. Über den Link „New Item“ im „Table Explorer“ gelangt man Einfügemaske, dem „Put Item View“ (siehe Abb. 5). An dieser Stelle gilt es zunächst den Wert für das Primärattribut anzugeben, als dem „Key“ der Key/Value-Datenbank. Desweiteren können hier die zusätzlichen Attribute und Werte festgelegt werden. Dabei kann die Anzahl und die Arte der Attributstypen pro „Item“ unterschiedlich sein. Lediglich der Primärschlüssel für die Objekte sind vom gleichen Typ.

[138] Entnommen aus: Amazon Web Services, Inc. (o.J.)

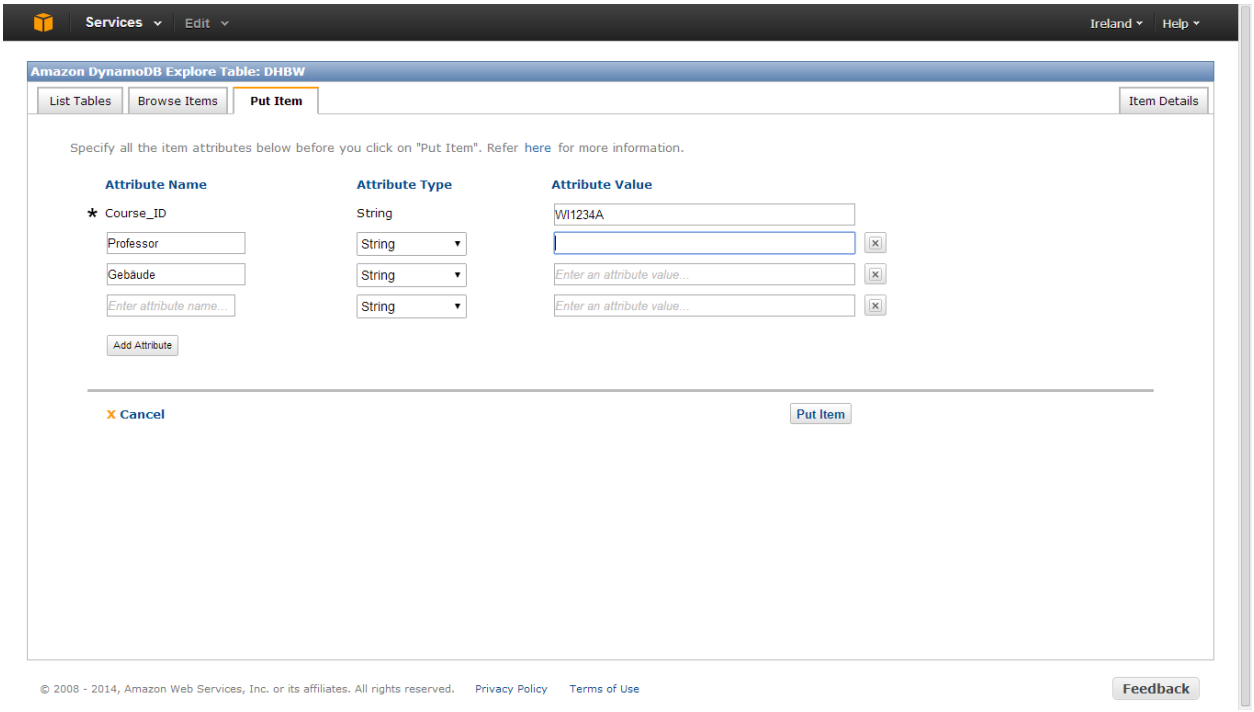


Abb. 5: Objekt anlegen[139]

Sind diese Daten vollständig eingetragen, wird das Objekt über den Befehl „Put Item“ erstellt. Der gesamte Prozess ist auch nochmal im Anhang 2 zu sehen.

[139] Entnommen aus: Amazon Web Services, Inc. (o.J.)

5.2 mongolab

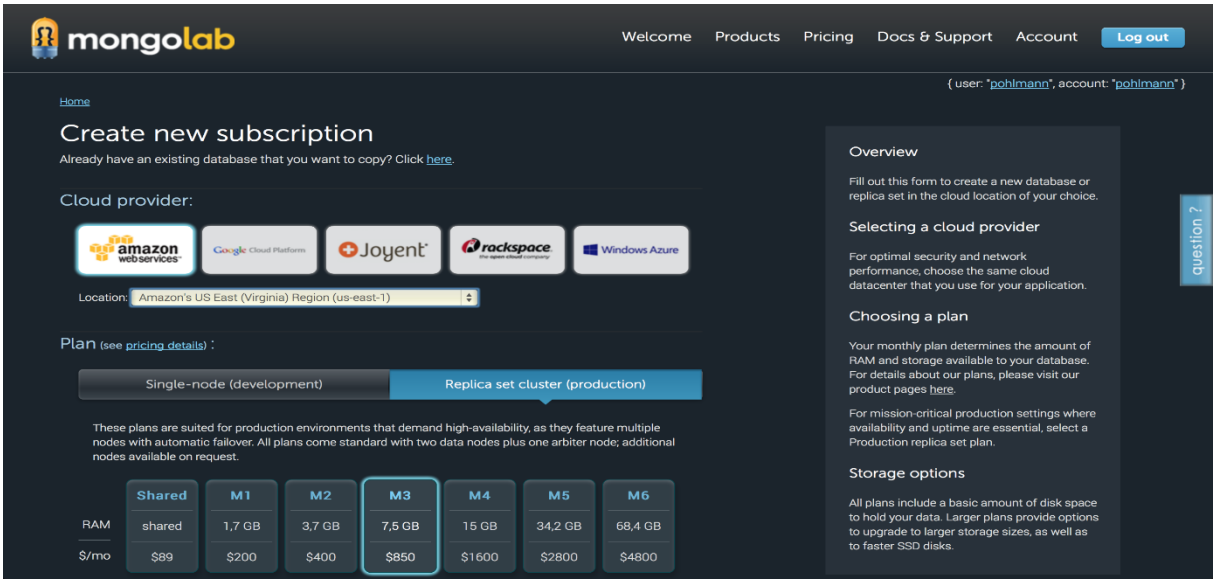


Abb. 6: mongolab Webinterface "Create new"[140]

Das Webinterface des Anbieters mongolab ist benutzerfreundlich und intuitiv aufgebaut. Das Anlegen einer neuen Datenbank ist selbsterklärend und kann in folgende Schritte gegliedert werden: Nach dem man die „Create new“ Form erreicht hat, wählt man zunächst den gewünschten Cloud Provider und die entsprechende Region. Anschließend kann zunächst zwischen den bereits erwähnten Optionen Single-node bzw. Replica set cluster und dann jeweils zwischen verschiedenen Leistungen gewählt werden. Es folgen die Auswahl der MongoDB-Version sowie verschiedene Speichergrößen zu unterschiedlichen Preisen. Abschließend kann ein Datenbankname eingetragen und die gewählte Konfiguration mit „Create new MongoDB deployment“ bestätigt werden. Die Datenbank ist nun innerhalb weniger Sekunden verfügbar. Außerdem verfügt das Webinterface über die Möglichkeit, Datenbanken zu im- bzw. exportieren oder zu klonen, Backups durchzuführen, Statistiken auszuwerten und ähnliches. Eine weitere Funktion bei mongolab ist es, einzelne MongoDB-Dokumente direkt wie mit Hilfe eines Texteditors zu bearbeiten. Der vollständige Prozess ist außerdem in Anhang 3 zu finden.

[140] Entnommen aus: mongolab (2014e)

5.3 Webinterface: GrapheneDB

Abb. 7: GrapheneDB / Neo4j Webinterface[141]

Das eigentliche Webinterface bei GrapheneDB beinhaltet lediglich das Anlegen neuer Datenbanken, Backups sowie Monitoring. Für die Verwaltung bzw. Administration wird das Standard Neo4j Webinterface verwendet, wie auf dem Screenshot zu sehen.

5.4 Webinterface: Cloudant

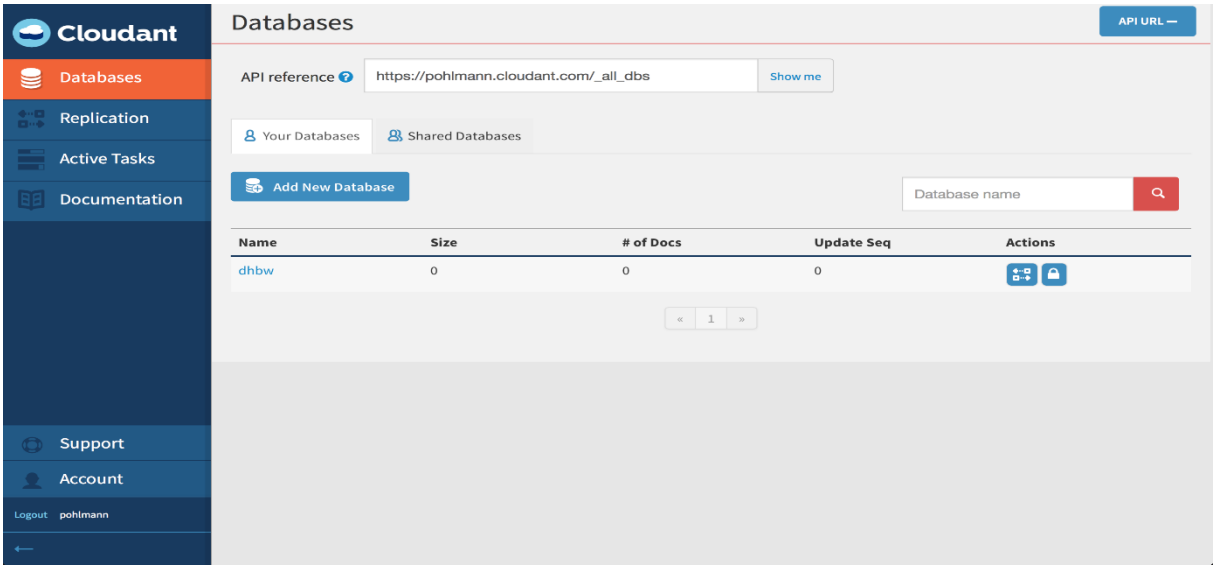


Abb. 8: Cloudant Webinterface[142]

Cloudant verwendet ein sehr einfach gehaltenes Webinterface. Datenbanken können sehr unkompliziert angelegt und verwaltet werden. Ein besonderer Vorteil ist stets die angezeigte API URL der jeweiligen Datenbank bzw. Ansicht.

---

[141] Entnommen aus: GrapheneDB (o.J.)

[142] Entnommen aus: Cloudant (o.J.)

6 Fazit

Der Markt der NoSQL-DBaaS-Provider ist noch sehr neu und deshalb auch noch nicht wirklich eindeutig aufgeteilt. Auf der einen Seite sind die großen Cloud Service Provider wie Amazon Web Services, Rackspace oder Windows Azure, die natürlich - wie jeder normale andere Server auch - genutzt werden können, um eine NoSQL Datenbank aufzusetzen. Auf der anderen Seite gibt es die vielen kleinen Anbieter, die sich auf ein spezielles NoSQL Datenbanksystem spezialisiert und in aller Regel ein eigenes System zur Verwaltung entwickelt haben. Doch auch hier bildet die technische Grundlage in den meisten Fällen ein Backend eines etablierten Anbieters wie AWS.

Für wen sich welches System besser eignet, ist nur relativ schwer pauschal zu beantworten. Es gibt sehr viele Faktoren, die diese Entscheidung beeinflussen können. Dazu gehören - wie im vorherigen Kapitel bereits genannt - neben den allgemeinen auch wirtschaftliche und technologische Aspekte. Für die Wahl eines entsprechenden NoSQL Datenbanksystems oder -typs, also Document Store, Key-Value Store, usw., ist außerdem vor allem die Art und Struktur der Daten relevant.

Eine grundlegende Einteilung lässt sich wie folgt vornehmen: Will man sehr flexibel sein, ein System nicht nur für einen Einsatzzweck nutzen und die NoSQL Datenbank eventuell wechseln, eignet sich am ehesten ein allgemeiner Cloud Anbieter. Amazon hat hier eine solide Marktposition und ist preislich und infrastrukturell sowohl für Privatanwender als auch für Großkonzerne geeignet. Nachteil dieser Variante des NoSQL-Hostings ist die Tatsache, dass jegliche Installation eines NoSQL-Datenbanksystems selbst erfolgen muss. Dasselbe gilt für Konfiguration, Monitoring und Wartung. Aus diesem Grund macht diese Variante vor allem für technisch versierte Personen Sinn.

Die zweite, immer noch relativ allgemeingültige Möglichkeit, ist die Verwendung eines MongoDB-as-a-Service-Providers. Die im Rahmen dieser Ausarbeitung vorgestellten Anbieter sind die Bekanntesten und Größten dieser Art. Wie bereits erwähnt, sind Document Stores sehr vielseitig einsetzbar. Sowohl der Bereich Social Media, Log-Daten Auswertung als auch gewöhnliche Datenbank-Aufgaben sind gut umsetzbar.

MongoDB ist das populärste System im Bereich Document Stores, was nicht zuletzt durch namhafte deutsche (wie Bosch, die scout24 Gruppe, SAP, usw.) wie auch internationale

Kunden (Goldman Sachs, eBay, John Deere, usw.) bestätigt wird.[143] Außerdem stehen zahlreiche APIs zur Verfügung, um eine bestehende Anwendung einfach an MongoDB anzubinden. Sowohl mongolab als auch MongoHQ bieten beispielsweise zudem sowohl eine kostenlose, zeitlich begrenzte Testversion als auch eine komplett freie Basisversion an.

Ein weiterer Fall, der kurz beleuchtet werden soll ist die Verwendung und Verarbeitung von Geodaten: Dieser relativ spezielle Fall gewinnt gerade im Zeitalter mobiler Apps immer mehr von Bedeutung und stellt ein ganz neues Anwendungsgebiet für NoSQL Datenbanken dar. Allerdings eignen sich dafür weniger die generischen Anbieter. Grundsätzlich bieten Graph Datenbanken die beste Art der Verarbeitung von Geodaten. Unter den beschriebenen Providern verwendet nur GrapehenDB ein derartiges System, in diesem Fall Neo4j. Aber auch die CouchDB-Vertreter IrisCouch und Cloudant warten mit spezieller Funktionalität - extra zum Bewältigen und Prozessieren komplexer Geodaten - auf.

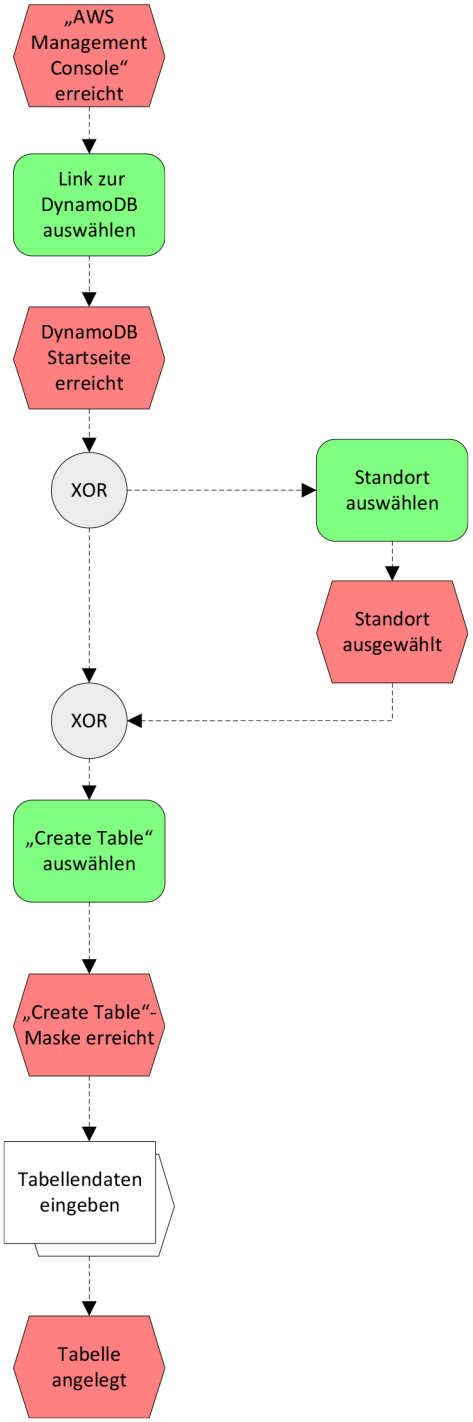
Wie bereits erwähnt, ist es sehr schwer, pauschal und generell den besten Anbieter zu bestimmen. Das liegt zum einen an den bereits genannten Punkten, zum anderen auch an der Tatsache, dass die meisten anbieterbezogenen Informationen lediglich auf eigenen Informationen der jeweiligen Provider beruhen, da in dem noch relativ jungen Feld nahezu keine Meinungen Dritter verfügbar sind. Das heißt, es könnte sich im negativsten Fall um unwahre Marketinginformationen und keine sachlichen Fakten handeln. Tatsächlich haben sich die gesammelten Informationen aber mit den stichprobenartig durchgeführten Tests der einzelnen Anbieter gedeckt und diese Ausarbeitung kann grundsätzlich sehr wohl als hilfreicher Guide bei der Auswahl eines NoSQL-DBaaS-Providers nützlich sein.

[143] Schwering, M. (2013)

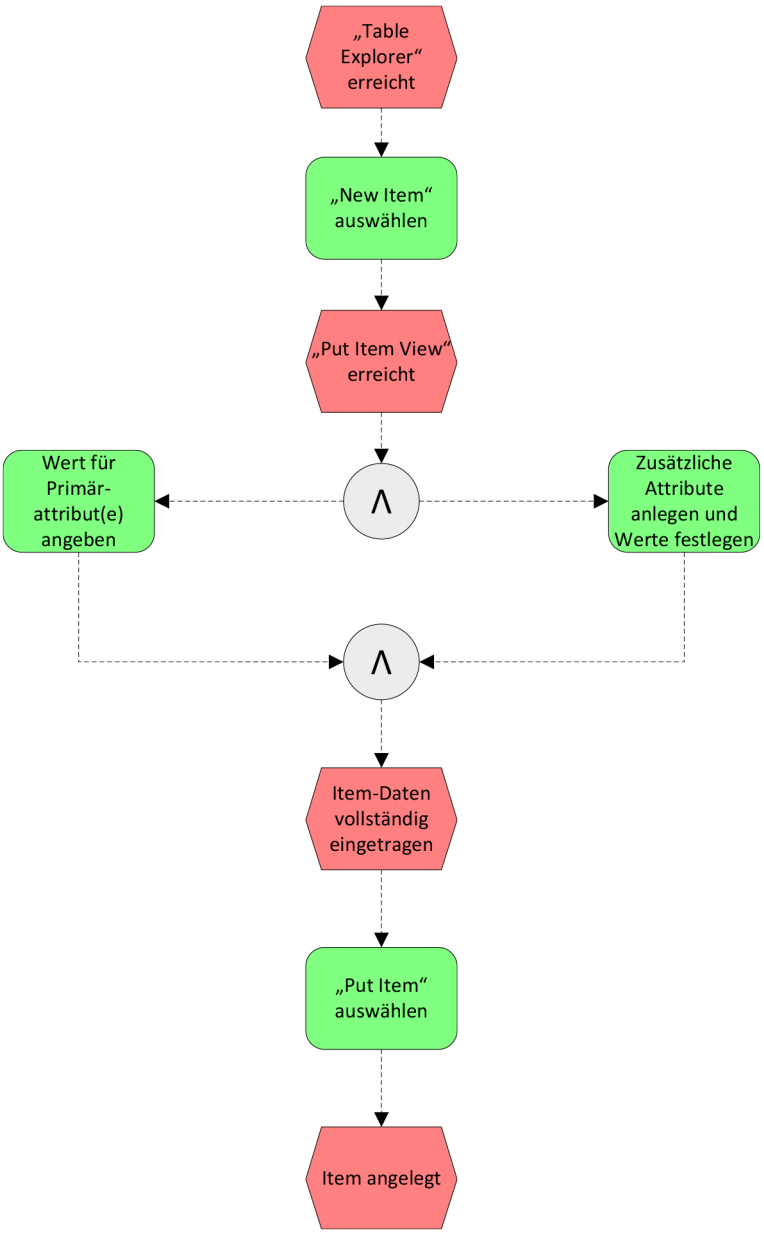
Anhang

Anhangverzeichnis

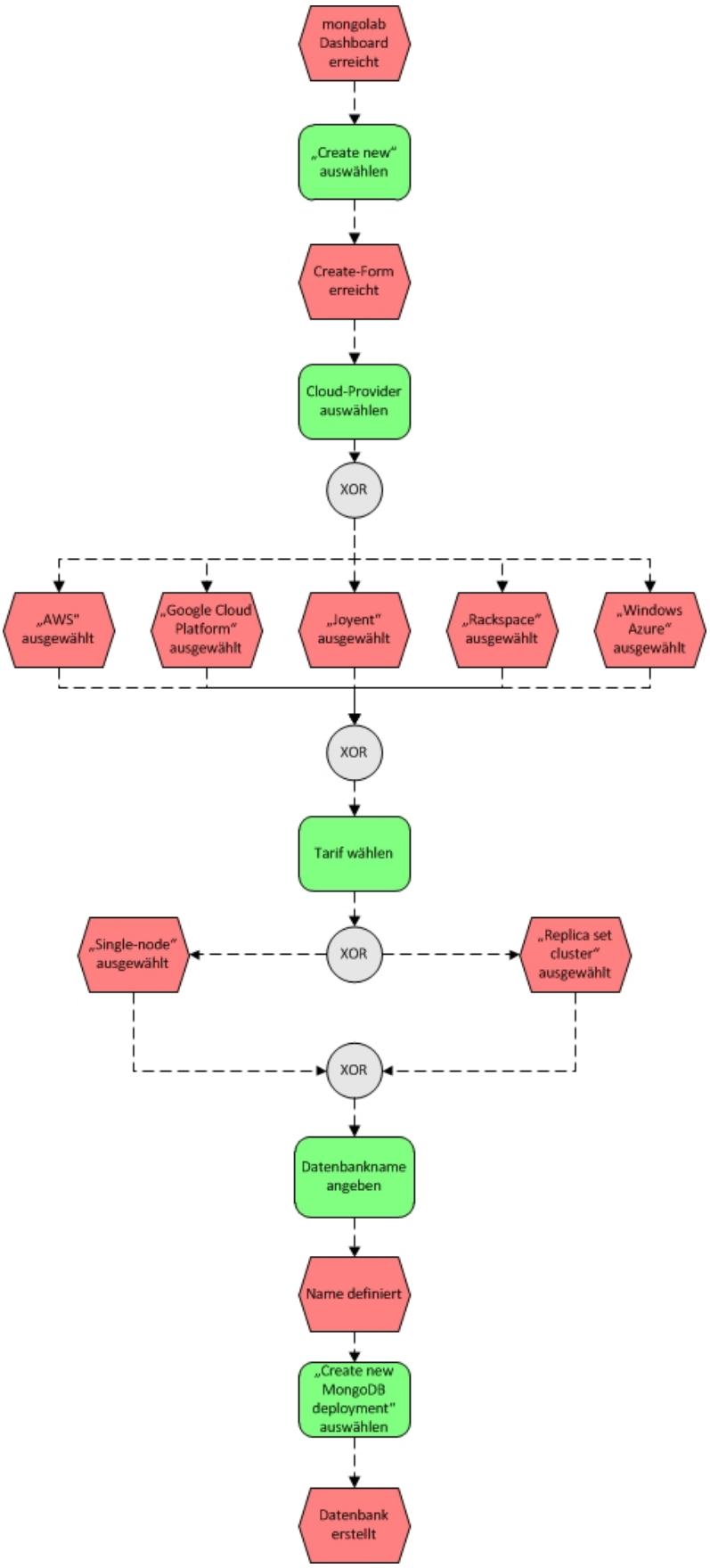
Anhang 1: EPK-Diagramm AWS DynamoDB anlegen



Anhang 2: EPK-Diagramm AWS DynamoDB Objekt anlegen



Anhang 3: EPK-Diagramm mongolab Datenbank anlegen



7 Quellenverzeichnis

7.1 Literaturverzeichnis

Andrikopoulus, V. u.a. (2012): How to adapt applications for the Cloud environment, in: Computing, 2013, Springer-Verlag Wien

Chang, F. u. a. (2006): Bigtable: A Distributed Storage System for Structured Data, in: OSDI, 2006

Däßler, R. (2001): Das Einsteigerseminar MySQL, 1. Aufl., Bonn: verlag moderne industrie Buch AG & Co. KG, Landsberg

Edlich, S u.a. (2010): NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken, München: Carl Hanser Verlag

Roth, A. (2013): Re:invent – Amazon setzt ein Zeichen, in: Computerwoche, 2013, Nr. 47

Taylor, A. (2004): SQL für Dummies, 3. Aufl., Bonn: MITP-Verlag

Yang, B./Qian, W./Zhou, A. (2008): Using Wide Table to manage web data: a survey, in: Front. Comput. Sci. China, 2008, Higher Education Press and Springer-Verlag

7.2 Internet-Quellen

Aentos (o. J.a): GrapheneDB Homepage, http://www.graphenedb.com, Abruf: 18.01.2014

Aentos (o. J.b): GrapheneDB Pricing, http://www.graphenedb.com/plans.html, Abruf: 18.01.2014

Amazon Web Services, Inc. (2013a): Amazon DynamoDB, http://aws.amazon.com/de/dynamodb/, Abruf: 18.01.2014

Amazon Web Services, Inc. (2013b): Preise, http://aws.amazon.com/de/dynamodb/pricing/, Abruf: 18.01.2014

Amazon Web Services, Inc. (2013c): Amazon EC2 – Preise, http://aws.amazon.com/de/ec2/pricing/, Abruf: 18.01.2014

Amazon Web Services, Inc. (2013d): Amazon EC2-Instances, http://aws.amazon.com/de/ec2/instance-types/, Abruf: 22.01.2014

Amazon Web Services, Inc. (o.J.) AWS Management Console, https://console.aws.amazon.com/dynamodb, Abruf: 27.01.2014

Borenstein, P. (2013): MongoDB Instance, http://wiki.joyent.com/wiki/display/jpc2/MongoDB+Instance, Abruf: 18.01.2014

Bösswetter, D. (2010): Spaltenorientierte Datenbank, http://www.gi.de/service/informatiklexikon/detailansicht/article/spaltenorientierte-datenbanken.html, Abruf: 18.01.2014

Cloudant (2013a): Cloudant About us, https://cloudant.com/about-us/, Abruf: 15.01.2014

Cloudant (2013b): Cloudant Partner, https://cloudant.com/about-us/partners/, Abruf: 15.01.2014

Cloudant (2013c): Cloudant Service, https://cloudant.com/product/service/, Abruf: 15.01.2014

Cloudant (2013d): Cloudant Features, https://cloudant.com/product/cloudant-features/, Abruf: 15.01.2014

Cloudant (2013e): Cloudant Features - Sync, https://cloudant.com/product/cloudant-features/sync/, Abruf: 15.01.2014

Cloudant (2013f): Cloudant Features - MapReduce, https://cloudant.com/product/cloudant-features/mapreduce/, Abruf: 15.01.2014

Cloudant (2013g): Cloudant Features - Search, https://cloudant.com/product/cloudant-features/cloudant-search/, Abruf: 15.01.2014

Cloudant (2013h): Cloudant Training, https://cloudant.com/product/training-services/, Abruf: 15.01.2014

Cloudant (2013i): Cloudant Pricing, https://cloudant.com/product/pricing/, Abruf: 15.01.2014

Cloudant (o.J.): Cloudant, https://www.cloudant.com, Abruf: 27.01.2014

Eifrem, E. (2013): Graphendatenbanken – Start-up-ready, http://entwickler.de/artikel/Graphendatenbanken-Start-up-ready, Abruf: 18.01.2014

FH Köln (2013): Graphendatenbank, http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/Graphendatenbank, Abruf: 18.01.2014

Garantia (2013a): Garantia About, http://garantiadata.com/about-us, Abruf: 18.01.2014

Garantia (2013b): Garantia Pricing, http://garantiadata.com/Pricing, Abruf: 18.01.2014

Garantia (2013c): Garantia Scalability, http://redis-cloud.com/redis/tour/redis-infinite-scalability, Abruf: 18.01.2014

Garantia (2013d): Garantia Setup, http://redis-cloud.com/redis/tour/redis-simple-setup-and-operation, Abruf: 18.01.2014

Goossaert, E. (2013): Implementing a Key-Value Store – Part 5: Hash table implementations, http://codecapsule.com/2013/05/13/implementing-a-key-value-store-part-5-hash-table-implementations/, Abruf: 27.01.2014

GrapheneDB (o. J.): GrapheneDB Dashboard, https://beta.graphenedb.com, Abruf: 27.01.2014

Grinev, M. (2010): A Quick Introduction to the Cassandra Data Model, http://maxgrinev.com/2010/07/09/a-quick-introduction-to-the-cassandra-data-model/, Abruf: 27.01.2014

Ha, T. (2011): Charakteristika und Vergleich von SQL- und NoSQL-Datenbanken,, http://dbs.uni-leipzig.de/file/seminar_1112_tran_ausarbeitung.pdf, Abruf: 18.01.2014

Harizopoulos, S. (2009): Column-Oriented Database Systems, http://www.cs.yale.edu/homes/dna/talks/Column_Store_Tutorial_VLDB09.pdf, Abruf: 18.01.2014

Heise (2013): Key-Value-Datenbank, http://www.heise.de/open/artikel/NoSQL-im-Ueberblick-1012483.html?artikelseite=2, Abruf: 18.01.2014

Hochstätter, C. (2011): Amazon AWS: So nutzt man den Cloud-Dienst EC2, http://www.zdnet.de/41557555/amazon-aws-so-nutzt-man-den-cloud-dienst-ec2/, Abruf: 22.01.2014

Hunger, M./Neubauer, P. (2013): Die vernetzte Welt, http://www.heise.de/developer/artikel/Abfragesprachen-fuer-Graphendatenbanken-1985666.html, Abruf: 18.01.2014

Ihlenfeld, J. (2008a): Amazon wird zum Webhoster, http://www.golem.de/0803/58663.html, Abruf: 22.01.2014

Ihlenfeld, J. (2008b): Amazon EBS - bis zu 20 TByte Storage-Volumes, http://www.golem.de/0808/61890.html, Abruf: 22.01.2014

instaclustr (2013a): instaclustr About, https://www.instaclustr.com/colophon, Abruf: 15.01.2014

instaclustr (2013b): instaclustr Provider, https://www.instaclustr.com/?provider=AWS&data-centre=US_EAST1#pricing, Abruf: 15.01.2014

instaclustr (2013c): instaclustr Homepage, https://www.instaclustr.com, Abruf: 15.01.2014

IrisCouch (2010a): IrisCouch Contact, http://www.iriscouch.com/contact, Abruf: 11.01.2014

IrisCouch (2010b): IrisCouch Homepage, http://www.iriscouch.com, Abruf: 11.01.2014

IrisCouch (2010c): IrisCouch Service, http://www.iriscouch.com/service, Abruf: 11.01.2014

Joyent, Inc. (2014a): Joyent Products, http://www.joyent.com/products, Abruf: 18.01.2014

Joyent, Inc. (2014b): The Joyent Compute Service, http://www.joyent.com/products/compute-service, Abruf: 25.01.2014

Joyent, Inc. (2014c): Manta Storage Service – Big Compute for Big Data, http://www.joyent.com/content/06-developers/01-resources/45-joyent-manta-data-sheet/20130920-datasheet-manta.pdf, Abruf: 25.01.2014

Joyent, Inc. (2014d): Joyent SmartDataCenter™, http://www.joyent.com/products/private-cloud, Abruf: 25.01.2014

Joyent, Inc. (2014e): Joyent Image Pricing, http://www.joyent.com/products/compute-service/image-pricing, Abruf: 25.01.2014

Microsoft (2014a): HDInsight – Preisdetails, http://www.windowsazure.com/de-de/pricing/details/hdinsight/, Abruf: 18.01.2014

Microsoft (2014b): Virtuelle Computer – Preisdetails, http://www.windowsazure.com/de-de/pricing/details/virtual-machines/, Abruf: 18.01.2014

Microsoft (2014c): Virtuelle Computer http://www.windowsazure.com/de-de/services/virtual-machines/, Abruf: 18.01.2014

Microsoft (2014d): Implementing a Document Database, http://msdn.microsoft.com/en-us/library/dn313284.aspx, Abruf: 27.01.2014

MongoDB, Inc. (2013): MongoDB Introduction Sharding (2013), http://docs.mongodb.org/manual/core/sharding-introduction/, Abruf: 03.01.2014

MongoDB, Inc. (2014): Document Database, http://www.mongodb.com/document-databases, Abruf: 18.01.2014

MongoHQ (o. J.a): MongoHQ About, http://www.mongohq.com/about, Abruf: 08.01.2014

MongoHQ (o. J.b): MongoHQ Pricing, http://www.mongohq.com/pricing, Abruf: 08.01.2014

MongoHQ (o. J.c): MongoHQ Features, http://www.mongohq.com/features, Abruf: 08.01.2014

mongolab (2014a): mongolab Company, https://mongolab.com/company/, Abruf: 03.01.2014

mongolab (2014b): mongolab Products, https://mongolab.com/products/, Abruf: 03.01.2014

mongolab (2014c): mongolab Homepage, https://mongolab.com/welcome/, Abruf: 03.01.2014

mongolab (2014d): mongolab Pricing, https://mongolab.com/products/pricing/, Abruf: 03.01.2014

mongolab (2014e): mongolab "Create new", https://mongolab.com/create, Abruf: 27.01.2014

mongosoup (o. J.a): mongosoup Homepage, http://www.mongosoup.de/index.html, Abruf: 08.01.2014

mongosoup (o. J.b): mongosoup Pricing, http://www.mongosoup.de/index.html#pricing, Abruf: 08.01.2014

mongosoup (o. J.c): mongosoup About, http://www.mongosoup.de/index.html#about, Abruf: 08.01.2014

Neubauer, P. (2010): Neo4j – die High-Performance-Graphendatenbank, http://jaxenter.de/artikel/Neo4j-High-Performance-Graphendatenbank-167918, Abruf: 18.01.2014

Object Rocket (2012a): Object Rocket Details, http://www.objectrocket.com/details, Abruf: 03.01.2014

Object Rocket (2012b): Object Rocket About, http://www.objectrocket.com/about, Abruf: 03.01.2014

Object Rocket (2013c): Object Rocket Pricing, http://www.objectrocket.com/pricing, Abruf: 03.01.2014

Rackspace (o. J.a): Hybrid Hosting, http://www.rackspace.com/de/hybrid-hosting, Abruf: 18.01.2014

Rackspace (o. J.b): Next Generation Cloud Servers Pricing, http://www.rackspace.com/es/de/cloud/servers/next-gen-pricing, Abruf: 27.01.2014

Rahien, A. (2010): That NoSQL Thing – Key/Value stores, http://ayende.com/blog/4449/that-no-sql-thing-key-value-stores, Abruf: 18.01.2014

Rauch, H. (2011): Graphdatenbanken,

http://de.slideshare.net/HenningRauch/vortrag-graphendatenbanken-uni-stuttgart,
Abruf: 18.01.2014

Schönung, S. (o. J.): Graphendatenbanken,
http://baun-vorlesun-gen.appspot.com/SEM12/Dokumente/CLCP_SEM_SS2012_Graphendatenbanken_Ausarbeitung.pdf,
Abruf: 18.01.2014

Solid IT (o.J.) Wide Column Stores,
http://db-engines.com/de/article/Wide+Column+Stores,
Abruf: 27.01.2014

Steemann, J. (2012): Column-oriented databases,
http://de.slideshare.net/arangodb/introduction-to-column-oriented-databases,
Abruf: 18.01.2014

Uberspace (2014) Uberspace Homepage,
https://uberspace.de/,
Abruf: 27.01.2014

Wikipedia (2013a): Verteilte Hashtabelle,
http://de.wikipedia.org/wiki/Verteilte_Hashtabelle,
Abruf: 18.01.2014

Wikipedia (2013b): Assoziatives Datenfeld,
http://de.wikipedia.org/wiki/Assoziatives_Datenfeld,
Abruf: 18.01.2014

7.3 Gesprächsverzeichnis

Abel, F. (2013) Data Scientist XING AG Hamburg, Deutschland, Schriftverkehr am 25. November 2013

Schwering, M. (2013) Solutions Architect MongoDB, Inc. London (UK), Schriftverkehr am 27. Dezember 2013