DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

KOS.content

01 | 2013

Ergebnisse der Untersuchungen des
Kompetenzzentrum Open Source der DHBW-Stuttgart

Frühjahr 2013
band.2

INHALT BAND.2

Inhalt __

Das Kompetenzzentrum Open Source (KOS)

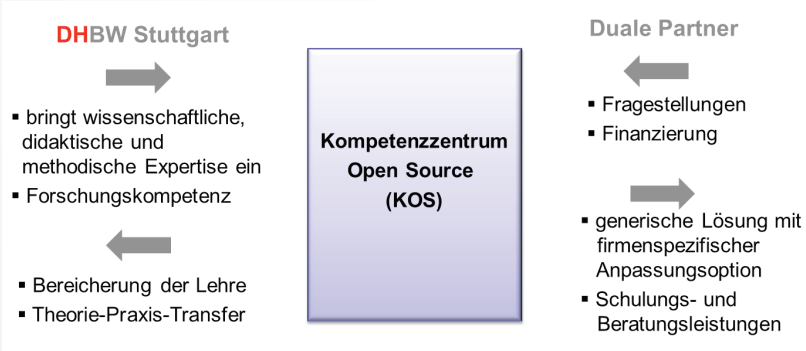
Ziel des Projektes

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

Open-Source Suchmaschinen

Marktanalyse und Bewertung

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“
Kompetenzzentrum Open Source (KOS)

Vorgelegt von

Larissa Königs, Simon Pieper,
Melanie Thum

am 25.01.2013

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
<WWI2010I>

Inhaltsverzeichnis

Abkürzungsverzeichnis

DHBW	Duale Hochschule Baden-Württemberg
VERS	Hier anonymisierte Versicherung
BW	Baden-Württemberg
URL	Uniform Resource Locator
LDAP	Lightweight Directory Access Protocol
JSON	JavaScript Object Notation
WWF	World Wide Fund for Nature
GNU GPL	„GNU's Not Unix" General Public License
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
PC	Personal Computer
P2P	Peer to Peer
XML	Extensible Markup Language
HTML	Hypertext Markup Language
MPL	Mozilla Public License
TREC	Text REtrieval Conference
HTTP	Hypertext Transfer Protocol
UTF	Unicode Transformation Format
JSP	Java Servlet Pages
JRE	Java Runtime Environment
JDK	Java Development Kit
RAM	Random Access Memory
API	Application Programming Interface
REST	Representational State Transfer
CSV	Comma-separated Values
NoSQL	Not Only Structured Query Language

Abbildungsverzeichnis

1 Einleitung

Die anonymisierte Versicherung (VERS) ist Europas größter XXXversicherer und besteht, in der Form wie es sie heute gibt, seit XXX. Ca. XXX Menschen in Deutschland sind hier versichert.[1]

Das Projekt, das auf den folgenden Seiten behandelt wird, wurde von der VERS angestoßen. In diesem Projekt geht es um eine Marktanalyse von Open-Source Suchmaschinen. Hintergrund des Ganzen ist die Suche im Intranet der VERS, das aus einem Wikisystem besteht. Diese Wikis sind zwar alle mit einer integrierten Suchfunktion ausgestattet, allerdings ist mit dieser nur die Suche innerhalb eines Wikis möglich. Ziel des Projektes ist es, Suchmaschinen daraufhin zu untersuchen, ob sie sich für eine übergeordnete Suche über das gesamte Wiki-System eignen. Bei der Auswahl der zu untersuchenden Suchmaschinen wird der Fokus auf eine möglichst große Bandbreite an Konzepten gelegt, da eine ähnliche Arbeit, die ebenfalls im Rahmen dieses Projekts erstellt wurde, sich mit weniger Suchmaschinen, dafür aber eher mit den technischen Anforderungen und der Umsetzung beschäftigt.

Das Projekt gliedert sich in drei Hauptbestandteile. Zum einen soll in Zusammenarbeit mit der VERS ein Kriterienkatalog erstellt werden, nach dem verschiedene Suchmaschinen im nächsten Schritt untersucht werden sollen. Abschließend soll eine Empfehlung ausgesprochen werden, welche Open-Source Suchmaschine sich für die Ansprüche der VERS am besten eignet.

[1] o.V (o.J), anonymisiert

2 Theorie zu Suchmaschinen

2.1 Was sind Wikis

Wikis werden heutzutage sehr vielfältig eingesetzt. Die folgende Definition gibt einen guten Überlick, was Wikis sind und wie sie allgemein funktionieren:

„Wikis sind web- basierte Computerprogramme, die es dem Nutzer erlauben, schnell, einfach und in Gemeinschaft mit anderen Informationsangebote mit dem Web- Browser zu erstellen und die Inhalte sofort am Bildschirm ändern zu können. Auch die Ergebnisse dieser Zusammenarbeit werden Wikis genannt.“[2]

Der wohl bekannteste Vertreter der Wikis ist Wikipedia. Hierbei können Online-User nach Begriffen suchen und sich über diese informieren. Es besteht aber auch die Möglichkeit, Beiträge online zu ändern oder eben auch neue Beiträge hinzuzufügen. Wikipedia stellt in diesem Fall ein Wiki dar, welches online für jeden zugänglich ist. Es gibt allerdings auch sogenannte Unternehmenswikis, die lediglich unternehmensintern zugänglich sind und das Wissensmanagement deutlich erleichtern. Hierbei kann es Wikis geben die sich inhaltlich auf die verschiedenen Abteilungen beziehen, ebenso ist es aber auch möglich Wikis über Projekte zu führen, in denen Mitarbeiter an dem Projekt wichtige Informationen festhalten und für andere zugänglich machen können. Bei der VERS wird ein Unternehmenswiki eingesetzt um intern Informationen austauschen zu können.

2.2 Wofür werden Suchmaschinen eingesetzt?[3]

Suchmaschinen werden vielfältig eingesetzt. Sie dienen dem schnellen Auffinden von verschiedenen Inhalten. Gesucht werden können z.B. Dokumente, Links, Inhalte, Dateien, Bilder, Personen, oder ganz allgemein gesprochen: Informationen. Für den normalen Verbraucher ist wohl die Online Suchmaschine, wie z.B. Google oder Bing, am bekanntesten. Sie gehören zu der Kategorie der Websuchmaschinenan und sind, wenn sie öffentlich sind, für jeden über das Internet frei zugänglich. Ebenfalls Internet-basiert sind sogenannte vertikale Suchmaschinen. Diese durchsuchen das World Wide Web, allerdings nur zu bestimmten Themen, wie z.B. Sport, Medizin oder Technik. Beide bereits genannten Suchmaschinenarten sind meist frei zugänglich für jeden über das Internet. Eine Internetsuchmaschine im Gegensatz dazu ist z.B. nur für die Mitarbeiter einer Firma nutzbar und durchsucht auch nicht das World Wide Web sondern die Inhalte, die im Intranet des Unternehmens verfügbar sind. Eine Erweiterung zu der Intranetsuchmaschine ist die Enterprise Search Suchmaschine.

[2] Schmidt, B. (2006), http://kola.opus.hbz-nrw.de/volltexte/2006/58/pdf/sa_schmidt.pdf (Stand: 28.12.2012)
[3] Vgl. o.V. (2013), http://de.wikipedia.org/wiki/Suchmaschine#Datenquelle (Stand: 14.01.2013)

Hierbei wird nicht nur das firmeninterne Intranet, sondern Datenquellen, die in dem Unternehmen vorhanden sind, durchsucht. Desktop-Suchmaschinen werden für die Suche von Daten auf einem einzelnen Computer verwedent.

2.3 Verteilung von Suchmaschinen

Die Verteilung der Suchmaschinen, mit denen die Benutzer ihre Suchbegriffe im Web suchen, muss individuell betrachtet werden. Im Folgenden werden Daten aus den USA mit denen aus Deutschland verglichen.

In der Verteilung der meistgenutzten Suchmaschinen 2011 in den USA hält Google den ersten Platz mit einem Marktanteil von 64,42 %. Alle Suchen, die durch Bing durchgeführt wurden, ergeben in Summe 30 % und somit belegt das Produkt von Microsoft Rang 2. Auf Rang 3 ist Yahoo mit 15,69 %. Rein statistisch betrachtet ist der Marktanteil von Google in diesem Segment unglaublich hoch und lässt auf eine große Marktdominanz und –kontrolle schließen.

Abbildung 1: Marktanteile der Suchmaschinen in den USA 2007 bis 2011[4]

Im Vergleich zu dem bereits großen Marktanteil von Google in den USA sind die Zahlen für Deutschland umso interessanter und erstaunlicher. Auch hier belegt Google mit deutlichem Abstand Platz 1. Dessen Marktanteil liegt in Deutschland aber sogar bei über 80%.

[4] Vgl. o.V.(2013), http://de.statista.com/statistik/daten/studie/163595/umfrage/marktanteile-der-suchmaschinen-in-den-usa/ (Stand: 10.01.2013)

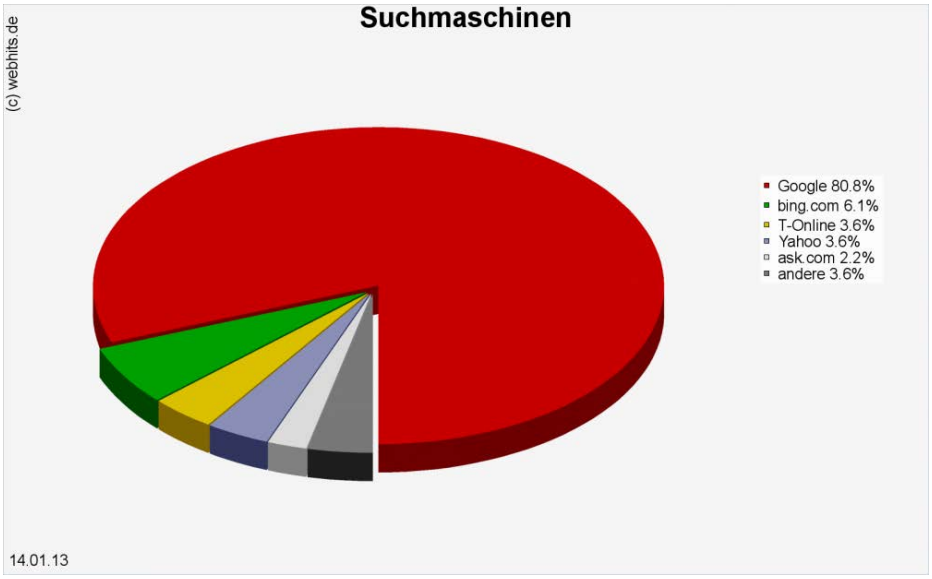


Abbildung 2: Marktanteil nach Nutzung in Deutschland[5]

Wie man in beiden Statistiken feststellen kann, sind keine Open Source Suchmaschinen unter den Ersten zufinden. Eine Theorie zu diesem Ergebnis könnte sein, dass Open Source Lösungen über weniger finanzielle Mittel verfügen, um zum Beispiel in Marketing zu investieren, um die Suchmaschine bekannter zu machen.

[5] Vgl. Meister, A./van Nerven, A. (o.J.), http://winfwiki.wi-fom.de/index.php/Implikationen_des_Social_Webs_auf_Suchmaschinentechnologien (Stand:07.01.2013)

2.4 Gegenüberstellung verschiedener Lizenzvarianten

Variante	Definition	Vorteile	Nachteile	Vertreter der jeweiligen Variante
Opensource	• Freie und kostenlose Weitergabe • Quellcode verfügbar • Abgeleitete Software darf, unter gleicher Lizenz weiter verbreitet werden	• Kostenlos • Fehler und Sicherheitslücken können durch eine Vielzahl von Programmierern behoben werden • Bei großer Akzeptanz schnelle Weiterentwicklung • Anpassungen leicht möglich	• Langsame Weiterentwicklung durch fehlende Einnahmen/ fehlende Akzeptanz am Markt möglich • Professioneller Support muss eingekauft werden	Nutch
Kommerziell	„Kommerzielle Software wird meist durch Unternehmen entwickelt, um einen Profit zu erwirtschaften.“[6]	• Professionelle Software • Sicherheit, in Zukunft weiterhin zu bestehen • Customizing ist durch den Provider leicht sicher zustellen • Professioneller Support	• Kostenpflichtig • Support ebenfalls kostenpflichtig • Meistens „Closed Source“	Google, Bing, Yahoo
Freeware	„…im allgemeinen proprietäre Software von Herstellern, die ihr Produkt kostenlos verbreiten. Das Kopier- und Weiterverbreitungsverbot	• Kostenlos • Ähnlich gute Qualität wie bei kommerzieller Software • Hohe Sicherheit	• Quellcode ist nicht verfügbar	Copernic (Freeware Suchmaschine für Windows-PCs)

[6] Bußkamp, D. (o.J.) http://www.dbus.de/eip/kapitel01e.html (Stand:20.12.2012)

	ist somit nicht mehr gültig, wohl aber alle anderen einschränkenden Lizenzbedingungen. Die Veränderung ist nicht erlaubt und auch nicht möglich, da der Quellcode nicht zur Verfügung steht. Freeware ist somit keine freie Software.“[7]			
Grüne Suchmaschinen	Eine „grüne“ Suchmaschine bezeichnet eine Suchmaschine, die durch unterschiedlichste Aspekte von der Verwendung von Öko-Strom, über Spenden bis zum Kauf von CO2 Zertifikaten einen nachhaltigen Gedanken verfolgt und die Umwelt schont. Spenden werden entweder über das Betriebsergebnis/Werbeeinnahmen oder pro Suchanfrage ausgezahlt. Derzeit wird noch mit Suchergebnissen großer Vertreter wie Google und Bing gearbeitet.	• Für Suchanfragen wird Ökostrom verwendet • Teilweise wird pro Suchanfrage ein bestimmert Betrag gespendet	• Bei den meisten grünen Suchmaschinen laufen die Suchanfragen weiterhin über die großen Suchmaschinen wie z.B. Google	Ecosia

[7] Bußkamp, D. (o.J.) http://www.dbus.de/eip/kapitel01e.html (Stand:20.12.2012)

Eine Vielzahl von Software Produkten sind auf dem Markt verfügbar. Diese lassen sich aber in grundlegenden Punkten unterscheiden. Zum einen gibt es Open Source Produkte, die kostenlos erhältlich sind. Diese Variante ist meist sehr gut entwickelt, da der Quellcode frei zugänglich ist und von jedem optimiert werden kann. Ebenso besteht aber auch die Möglichkeit, dass die Weiterentwicklung aufgrund geringen Finanzierungsmöglichkeiten schleppend verläuft. Der fehlende Support ist hierbei ebenfalls ein Nachteil. Im Gegensatz dazu gibt es kommerzielle Software und Freeware. Kommerzielle Software ist zwar kostenpflichtig allerdings ist professioneller Support vorhanden. Freeware ist wie Open Source Produkte zwar kostenlos, der Quellcode ist allerdings nicht frei verfügbar.

2.5 Funktionsweise

Suchmaschinen setzen sich im Allgemeinen aus drei Bestandteilen zusammen:

1. Web Robot System

   Das Web Robot System, oder auch häufig Crawler genannt, sucht „aktiv im Internet neue Websites und Daten“[8]. Dies ist die Basis für die Datensammlung auf die später die Suchmaschine zugreift und aus der die Ergebnisse geliefert werden. Der Webcrawler gelangt mit Hilfe von Hyperlinks zu den verschiedenen Webseiten und die entsprechenden URLs werden in einer Liste gespeichert.[9] Dies ist dann die bereits erwähnte Datensammlung. Die in der Datensammlung gespeicherten Webseiten werden von dem Crawler aktiven in regelmäßigen Abständen auf Änderungen untersucht.[10]

2. Information Retrival System

   Das Information Retrival System hat die Aufgabe, die Daten, die von dem Crawler gefunden wurden, so aufzubereiten, dass sie später möglichst schnell und effizient durchsucht werden können. Dafür wird der unstrukturierte Datenbestand zunächst „analysiert und aufbereitet“.[11] Um später eine Indexierung durchführen zukönnen, wird nach passenden Schlüsselbegriffen in den Webseiten gesucht und diese dann festgelegt. Danach wird eine einheitliche Datenstruktur erstellt, die es dem Query Processor später ermöglicht, schnell die passenden Ergebnisse zu finden und diese nach Relevanz zu sortieren.

[8] Gaulke, M. (o.J), http://www.suchmaschinenkompetenz.de/Suchdienste-Suchmaschinenkompetenz.htm (Stand: 22.12.2012)
[9] o.V. (2012), http://de.wikipedia.org/wiki/Webcrawler (Stand: 22.12.2012)
[10] o.V. (o.J.), http://www.drisol.com/informationen/seo-lexikon/crawler/ (Stand: 22.12.2012)
[11] Vgl. Gaulke, M. (o.J) http://www.suchmaschinenkompetenz.de/Funktionsweise-Suchmaschine-Suchmaschinenkompetenz.htm (Stand: 22.12.2012)

3. Query Processor

Der Query Processor wird aktiv, sobald eine Suchanfrage gestellt wird. Er durchsucht den vom Inromation Retrival System aufbereiteten Datenbestand auf Einträge, die zu den eigegebenen Suchbegriffen passen. Hierbei erfolgt das Durchsuchen mit Hilfe von Indexen, die anhand der Schlüsselbegriffe angelegt worden sind. Die gefundenen Ergebnisse werden dann vom Query Processor nach Relevanz sortiert und dem User aufgelistet.

In Abbildung 3 ist zu sehen, wie die drei verschiedenen Bestandteile zusammenarbeiten. Ein User veröffentlicht eine Website, der Crawler erfasst diese Seite und nimmt sie in den Datenbestand mit auf. Dieser Datenbestand wird dann durch das IR-System sortiert und eine Indexierung vorgenommen. Der Query Processor durchsucht bei einer Anfrage durch einen User nach Eingabe eines Suchbegriffs den Datenbestand und sortiert die Ergebnisse. Die Suchergebnisse werden dann dem User zurückgegeben.

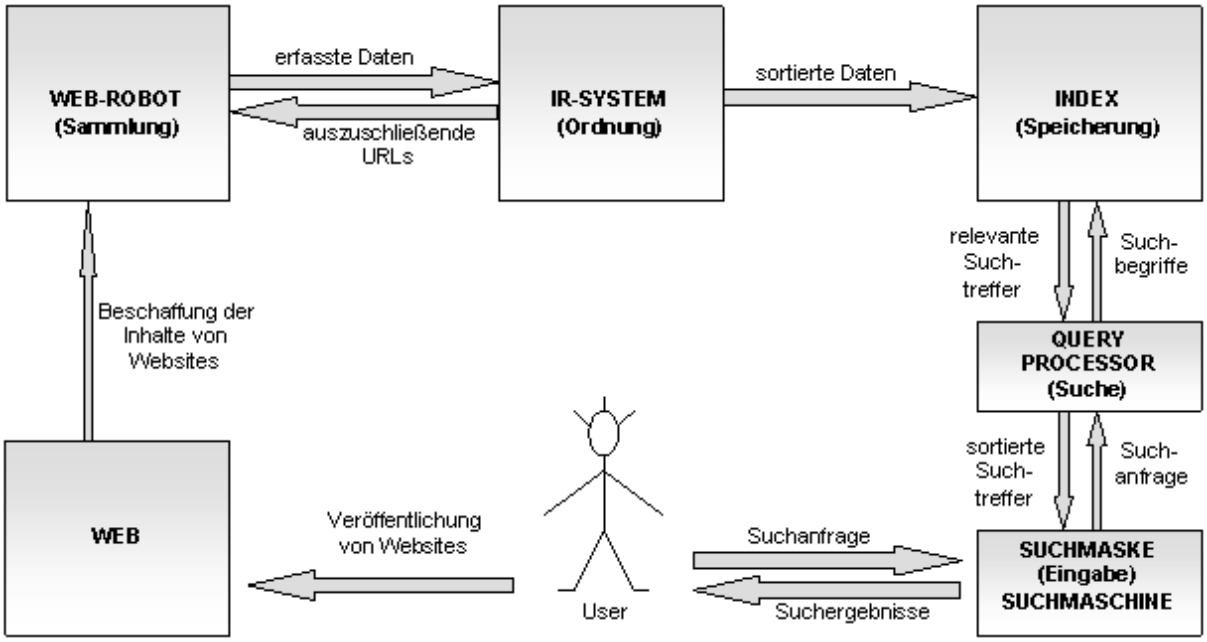


Abbildung 3: Funktionsweise von Suchmaschinen[12]

Für das Durchsuchen von Datenbeständen gibt es verschiedene Suchalgorithmen. Im Folgenden werden zwei Algorithmen vorgestellt.

[12] Vgl. Gaulke, M. (o.J) http://www.suchmaschinenkompetenz.de/Funktionsweise-Suchmaschine-Suchmaschinenkompetenz.htm (Stand: 22.12.2012)

Lineare Suchalgorithmen

Die lineare Suche ist die einfachste Möglichkeit, einen Datenbestand zu druchsuchen. Dieser Datenbestand kann bei dieser Suche unstrukturiert sein, aber je weniger Daten in dem Bestand enthalten sind, desto schneller kann der Algorithmus ein Ergebnis liefern. Dieser Algorithmus läuft ein Element in der Liste nach dem anderen, bis er das gesuchte Element findet. Im besten Fall steht das gesuchte Element an erster Stelle. In diesem Fall wäre die Suche sehr schnell erfolgreich. Die Anzahl der Durchläufe des Algorithmus‘ ist bei dieser Suche gleich der Anzahl der Elemente, die in dem datenbestand vorhanden sind.

Binäre Suchalgorithmen[13]

Die Voraussetzung für die binäre Suche ist eine schon bestehende Struktur des Datenbestands. Hierbei wir in dem gesamten Datenbestand das Element ermittelt, das genau in der Mitte ist. Hier wird nun geschaut, ob das mittlere Element größer oder kleiner als das Gesuchte ist. Abhängig davon werden entweder die Elemente durchsucht, die „rechts“ bzw. „links“ von dem mittleren Element stehen. Die andere Hälfte der Daten muss dann nicht mehr bearbeitet werden. Die Komplexität dieses Algorithmus‘ ist in diesem Fall $\log_2 n$.

[13] o.V. (o.J.), http://de.wikipedia.org/wiki/Bin%C3%A4re_Suche#Algorithmus (Stand:24.01.2013)

3 Kriterienkatalog

Um die Marktanalyse genau an die Wünsche und Anforderungen der VERS anzupassen, wurde ein Fragebogen erstellt, der von Mitarbeitern der VERS, die an diesem Projekt beteiligt sind, ausgefüllt wurde. Desweiteren wurde ein Interview durchgeführt, bei dem noch einmal spezifische Fragen geklärt wurden.

Zunächst einmal wurde festgehalten, dass das Internet von der Suche komplett ausgeschlossen ist und sich die Suche lediglich auf das interne Netz und die vorhandenden Wikis bezieht. Verwendet wird ein natives Wikisystem mit einer Fileablage, bestehend aus einem Dokuwiki, mit 14 Instanzen. Das bedeutet, dass keine zusätzliche Datenbank im Hintergrund arbeitet und mit dem Wikisystem verbunden ist. Die relevante Datenquelle ist also das Wiki-System. Bisher geschieht die Suche nur über die Suchfunktion, die in dem Wiki-System integriert ist. Hierbei werden aber immer nur einzelne Wikis durchsucht. Ziel des Projekts ist es, eine Suchmaschine einzubinden, die das gesamte System durchsucht und eine zentrale Suchmaske verwendet. In dieser übergeordneten Suche sollen alle relevanten Daten und Dokumente eingebunden werden.

Die lokale Suche für einzelne Wikis funktioniert bisher gut und kann somit bei Bedarf neben der zukünftig übergeordneten Suche beibehalten werden. Allerdings sollte analysiert werden, ob diese Suche nicht überflüssig wird, da die übergeordnete Suchmaschine die einzelnen Wikis ebenfalls durchsuchen kann. Was allerdings als verbesserungswürdig eingestuft wurde, sind die gruppenbezogenen Zugriffsrechte. Diese sind zwar schon vorhanden, müssen aber teilweise genauer definiert werden. So ist der Zugriff von der Gruppenzugehörigkeit abhängig und den „Attributwerten im Verzeichnisdienst“[14]. Durch LDAP wird ein „bestimmtes Zugriffsrecht auf ein Wiki-System“[15] erteilt. User, die nicht angemeldet sind haben keine Berechtigung auf Dokumente zuzugreifen. Anhand der Gruppenzuordnung ist definiert, ob ein schreibender Zugriff auf ein Wiki zulässig ist oder nicht. Ansonsten dürfen Dokumente nur gelesen werden.

Die übergeordnete Suchmaschien soll Open Source basiert sein, wobei es hier keine Einschränkung gibt, welche Open Source Suchmaschinen untersucht und später eingesetzt werden soll. Hier wünscht der Kunde eine unabhängige Analyse einiger Lösungen. Bei den Prgrammiertechniken die von den Lösungen vorausgesetzt werden gibt es allerdings eine Einschränkung. Das hängt damit zusammen, das die Implementierung der neuen Such-Lösung von der internen IT-Abteilung durchgeführt werden soll. In der VERS sind Kenntnisse über Java und XML vorhanden, nicht jedoch über JSON. Dies ist ein wichtiges Kriterium, da

[14] Vgl. Fragebogen (s.Anhang)
[15] Vgl. Fragebogen (s.Anhang)

der Programmieraufwand bei der Integration einer Suchmachine ein großer Bestandteil ist und unbedingt in die Analyse der vohandenden Suchmaschinen mit aufgenommen werden muss.

Weitere Anforderungen sind unter anderem nicht funktionale Kriterien wie zum Beispiel Skalierbarkeit, Zuverlässigkeit, Umfang der Community oder auch wie sich die Suchmaschine in Zukunft entwickeln wird und ob sie auch in einigen Jahren noch besteht. Skalierbarkeit ist ein wichtiger Aspekt, da sich die Anforderungen eines Unternehmens mit der Zeit verändern können. Hier sollte die Suchmaschine jederzeit beliebig erweiterbar sein um zu vermeiden, dass ein neues Produkt, das die benötigte Funktionalität beinhaltet. Die Kriterien Umfang der Community und das Weiterbestehen einer Suchmaschine sind in sofern von Bedeutung, als dass Communities einen sehr hilfreichen Support darbieten. Das Weiterbestehen sollte auch mit in die Kriterien einfliessen, da es wenig sinn macht, ein Produkt einzuführen, dass in wenigen Jahren nicht mehr besteht.

4 Marktanalyse

Die folgende Marktanalyse betrachtet acht Suchmaschinen aus der großen Vielzahl an verfügbaren Suchmaschinen. Vier davon wurden näher betrachtet, weil sie in der Aufgabenstellung des Projekts enthalten sind. Es handelt sich dabei um Lucene, ElasticSearch, Solr und Hsearch. Die anderen vier Maschinen wurden ergänzend analysiert, um auch verschiedene Konzepte darzustellen und miteinzubeziehen. Ecosia wurde hier als Vertreter der „grünen“ Suchmaschinen ausgewählt. Ecosia wurde als Vertreter der „grünen“ Suchmaschinen ausgewählt, weil sie sich durch einen hohe Reifegrad, seriöses Auftreten und viel Transparenz auszeichnet. Die Suchergebnisse werden durch Bing bereitgestellt und sind dadurch von gewohnt hoher Qualität, als auch sehr schnell verfügbar. Als weitere Suchmaschine haben wir uns für Yacy entschieden. Hauptkriterium für die engere Auswahl liegt bei Yacy in der völlig differenten Gesamtkonzeption. Entgegen aller großen Vertreter auf dem Markt, wagt es Yacy ein Peer-to-Peer Konzept umzusetzen. Nach intensiverer Betrachtung erweist sich das Konzept als performant und konkurrenzfähig. Weiterhin untersuchten wir die Suchmaschine Swish-e detaillierter. Wir entschieden uns für Swish-e, da das Prgramm im Web als sehr schnell in der Indexierung und Suche bewertet wird und von der Universität Berkeley in den USA erweitert wurde. Eine Reihe verschiedener Webseiten benutzt seit Jahren diese Suchmaschine, unter anderem eine Vielzahl von Universitäten und das Guggenheim Museum. Schließlich ist die Suchmaschine Terrier in die nähere Betrachtung gekommen. Die Wahl fiel auf diese Suchmaschine, weil einige out-of-the box Funktionalitäten geboten werden sollen und der Innovationsreport 2005 schrieb, es erfülle alle Voraussetzungen für eine europäische Antwort auf Google. Dies galt es,näher zu betrachten. Anhand der Marktanalyse werden dann drei Favoriten, die den Anforderungen des Unternehmens entsprechen, empfohlen.

4.1 „Grüne“ Suchmaschinen

Das Thema Green IT ist bereits seit Jahren fester Bestandteil in den Diskussionen um den Stromverbrauch von Rechenzentren wie Google alleine durch Suchanfragen. Mittlerweile ist es möglich, gewisse Aussagen zu tätigen, wie hoch der Stromverbrauch für eine einzelne Suchanfrage ist. Fraglich ist, wie glaubhaft solche Statements sind. Seit mehreren Jahren gibt es auf dem Markt verschiedene Anbieter von „grünen“ Suchmaschinen. Sie verpflichten sich dem Umweltschutz und nachhaltigem Handeln. Die Ansätze reichen von einer Spende pro Suchanfrage bis hin zu nachhaltigem Wirtschaften eines Unternehmens, das sich verpflichtet hat, 80% aller Einnahmen an ein Regenwaldprojekt zu spenden. Die wichtigsten Vertreter in Deutschland sind: ecosia, ecosearch, forestle, goodsearch, greenseng, hornvogel, searchgreen, znout und umlu. Sie verfügen alle über eine eigene Suchmaske, arbeiten im Hintergrund aber nur mit den drei Anbietern Google, Bing und Yahoo. Damit sind diese

kein Open Source und, wenn überhaupt in einem Unternehmen einsetzbar, kostenpflichtig. Um die Ansätze solcher Suchmaschinen und die Gedanken dahinter zu verstehen, haben wir uns nichtsdestotrotz entschieden, eine dieser Suchmaschinen in unsere nähere Betrachtung einzubeziehen.

Definition: Eine „grüne“ Suchmaschine bezeichnet eine Suchmaschine, die durch unterschiedlichste Aspekte von der Verwendung von Öko-Strom, über Spenden bis zum Kauf von CO_2 Zertifikaten einen nachhaltigen Gedanken verfolgt und die Umwelt schont. Spenden werden entweder über das Betriebsergebnis/Werbeeinnahmen oder pro Suchanfrage ausgezahlt. Derzeit wird noch mit Suchergebnissen großer Vertreter wie Google und Bing gearbeitet.

4.2 Ecosia

Ecosia.org ist die Suchmaschine der Ecosia GmbH, die sich selbst als Social Business beschreibt, das sich für ökologische Nachhaltigkeit einsetzt. 80% der Einnahmen des Unternehmens fließen nachweislich an ein Regenwaldschutzprojekt des WWF. Die Nachweise dazu liefert Ecosia auf der Homepage durch Geschäftsberichte und Spendenbelege. Die Suchmaschine wurde im Dezember 2009 von Christian Kroll und freien Mitarbeitern entwickelt. Die Suchergebnisse werden von Bing bereitgestellt, die Suchanzeigen von Yahoo.

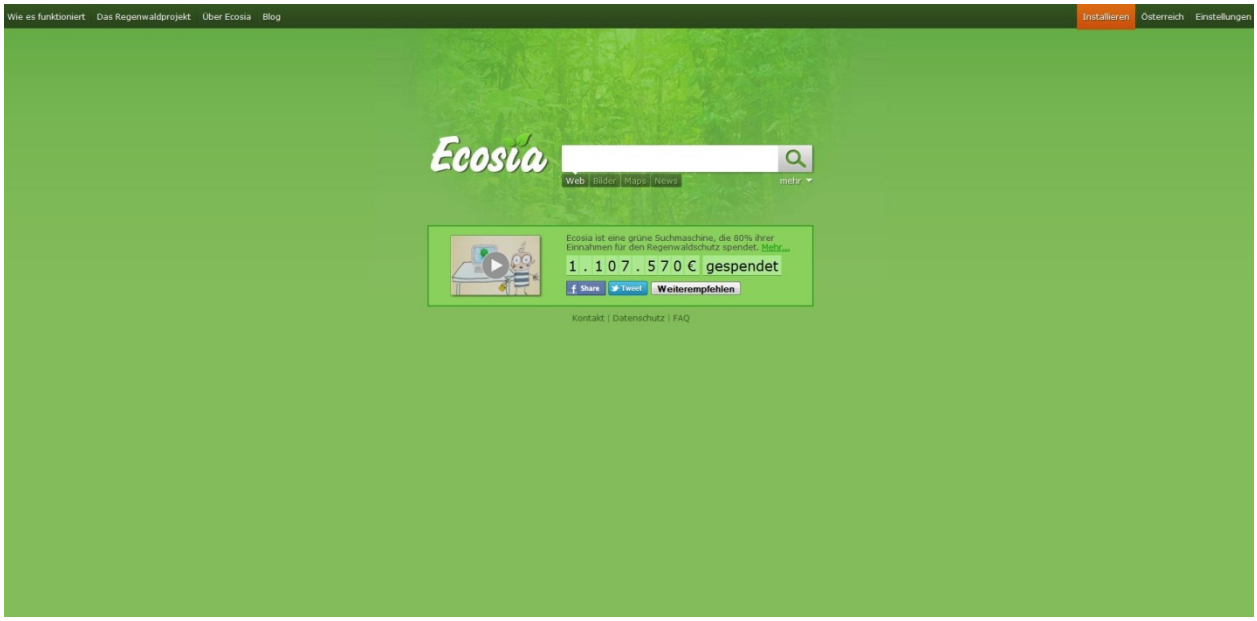


Abbildung 4: Startseite der Suchmaske von Ecosia[16]

[16] Enthalten in: http://ecosia.org (2013)

4.2.1 Vorteile

- Ecosia setzt ein Zeichen für die Umwelt
- Liefert einen Beweis für die erfolgreiche Umsetzung von Green IT bei Suchmaschinen
- Nachweislicher Umweltschutz einer renommierten, internationalen Organisation
- Die eigenen Server werden mit Ökostrom betrieben
- Für jede Suche investiert Ecosia in Gold-Standard-Projekte, die den CO_2 Ausstoß ausgleichen
- Qualitativ hochwertige Suchergebnisse durch Bing und Suchanzeigen von Yahoo. So kann der User von gewohnt guten Suchergebnissen ausgehen.
- Weil mit Bing und Yahoo gearbeitet wird, sind steigende Suchanfragen kein technisches Problem
- Trotz Spenden in Höhe von 80% der Einnahmen kann das Unternehmen bestehen. Ecosia bezeichnet sich nicht als Non-Profit-Organisation.
- Das Unternehmen agiert sehr transparent
- Einfache, aber sehr gute Suchmaske
- ECOLinks sorgen dafür, dass Ecosia Provision erhält, die an das WWF Projekt abgegeben wird. Die Nachweise liefern die Spendenbelege
- Ecosia bietet zusätzlich eine Plattform an, die Blogs und Ideen bereitstellt[17]
- Personenbezogene Nutzerdaten werden nach 48 Stunden gelöscht, so Ecosia
- Suchanfragen sollen nicht analysiert und Benutzerdaten nicht weiterverkauft werden[18]

4.2.2 Nachteile

- Die Suchanfragen laufen weiterhin auf den großen Servern von Microsoft, die nicht über Ökostrom laufen.
- Es wird kein Strom gespart, sondern ein Teil der Einnahmen gespendet.

[17] Vgl. ecosia.org (2013)
[18] Vgl. de.wikipedia.org (2013)

- Eine konkrete Regenwaldfläche pro Spendenbeitrag kann nicht angegeben werden
- Die CO_2 Emissionszertifikate werden nur gekauft, um den „unsauberen“ Strom von Microsoft auszugleichen
- Laut WWF Interview wird das Geld nicht direkt für den Schutz/Kauf einer bestimmten Regenwald-Fläche verwendet, sondern als Gehalt für Ranger und deren Ausrüstung[19]
- Nicht jede Suche, sondern nur jeder Klick auf einen ECOLink fördert das Projekt[20]

4.2.3 Funktionsweise

Technisch passiert bei Ecosia selbst nicht viel. Der Client greift über eine Suchmaske in seinem Browser auf die Ecosia Server zu. Diese leiten die Suchanfragen direkt an bing.com weiter, welches ebenfalls die Ergebnisse liefert. Die Suchanzeigen werden in der Suchmaske von Yahoo bereitgestellt. Die Ecosia Server laufen mit Ökostrom von Greenpeace Energy. Das Unternehmen erhielt einige Auszeichnungen. Das WWF Projekt, welches von Ecosia unterstützt wird, ist ein Schutzprojekt im Tumucumaque-Nationalpark im Amazonas.

4.2.4 Fazit

Auch wenn Ecosia keine Open Source Suchmaschine ist, ist es dennoch sehr interessant einen Blick auf dieses Konzept zu werfen. Das Konzept beweist, dass es möglich ist, Ansätze der Green IT in den Bereich der Suchmaschinen erfolgreich zu integrieren. Je nach Resonanz der User ist in Betracht zu ziehen, eine eigenständige, grüne Suchmaschine zu implementieren. Diese sollte dann unabhängig von den größeren Betreibern sein und eine ausgereifte Lösung bereitstellen, die der Kritik standhält. Eine grüne Suchmaschine, die mehr „Schein als Sein“ ist, würde das Vorhaben und die „ECO-Gemeinschaft“ stark nach hinten werfen. Trotz der Kritik an Ecosia können wir das Bestreben nur begrüßen, da versucht wird, ein Zeichen zu setzen und jegliche Bemühungen, die Umwelt zu schützen und die Nachhaltigkeit voranzubringen, positiv sind. Wir bewerten die Lösung als ausgereift und, bedingt durch die Unterstützung von Microsoft im Hintergrund, mit den anderen Suchmaschinen am Markt konkurrenzfähig. Eine Integration von Ecosia in das Intranet der VERS BW kommt hier nicht in Frage, da Ecosia wie bereits angesprochen nicht Open Source und damit kostenpflichtig ist (ebenfalls keine Freeware).

[19] Vgl. abenteuerdenken.blogspot.de (2010)
[20] Vgl. zeitblick.wordpress.com (2009)

Anforderungen des Unternehmens: nicht erfüllt, da kein Open Source	
Besonderheiten:	Best Practice Beispiel für den Einsatz von Green IT
	Ansatz der Motivation, „grüner“ zu denken
	Hochwertige und schnelle Suchen durch Bing

4.3 Yacy

Im Rahmen der Marktanalyse wurde entschieden, acht Suchmaschinen näher zu betrachten. Normale Suchmaschinen, die technisch gesehen nach dem „Google-Prinzip“ funktionieren, gibt es sehr viele. Im Laufe der Recherchen sind wir auf eine Suchmaschine gestoßen, die nach einem anderen Prinzip arbeitet, dem Peer-to-Peer-Prinzip. Umso interessanter ist es, zu analysieren, wie dieses Verfahren umgesetzt wurde und ob es konkurrenzfähig ist. Im Folgenden betrachten wir die Suchmaschine Yacy.

Yacy ist eine Suchmaschine, die nicht wie Google und Co. über zentrale Server läuft, die die Indexierung durchführen und die Suchmaske bereitstellen. Yacy arbeitet dezentral nach einem Peer-to-Peer Prinzip, in welchem die Nutzer selbst die Server bereitstellen und die Indexierung durchführen. Entwickelt wurde diese Suchmaschine von dem Frankfurter Informatiker Michael Christen im Jahr 2003. Die aktuelle Version ist 1.2 (Nov. 2012).

Sie ist unter der GNU GPL Lizenz veröffentlicht, kostenfrei zugänglich und auf Deutsch verfügbar. „Derzeit sind etwa 1.4 Milliarden Dokumente im Index, es kennt über 600 Peer-Betreiber pro Monat und rund 130.000 Suchanfragen pro Tag.“[21] Der Name kommt aus dem Englischen von *Yet another Cyberspace.*

[21] yacy.net/de/ (2013)

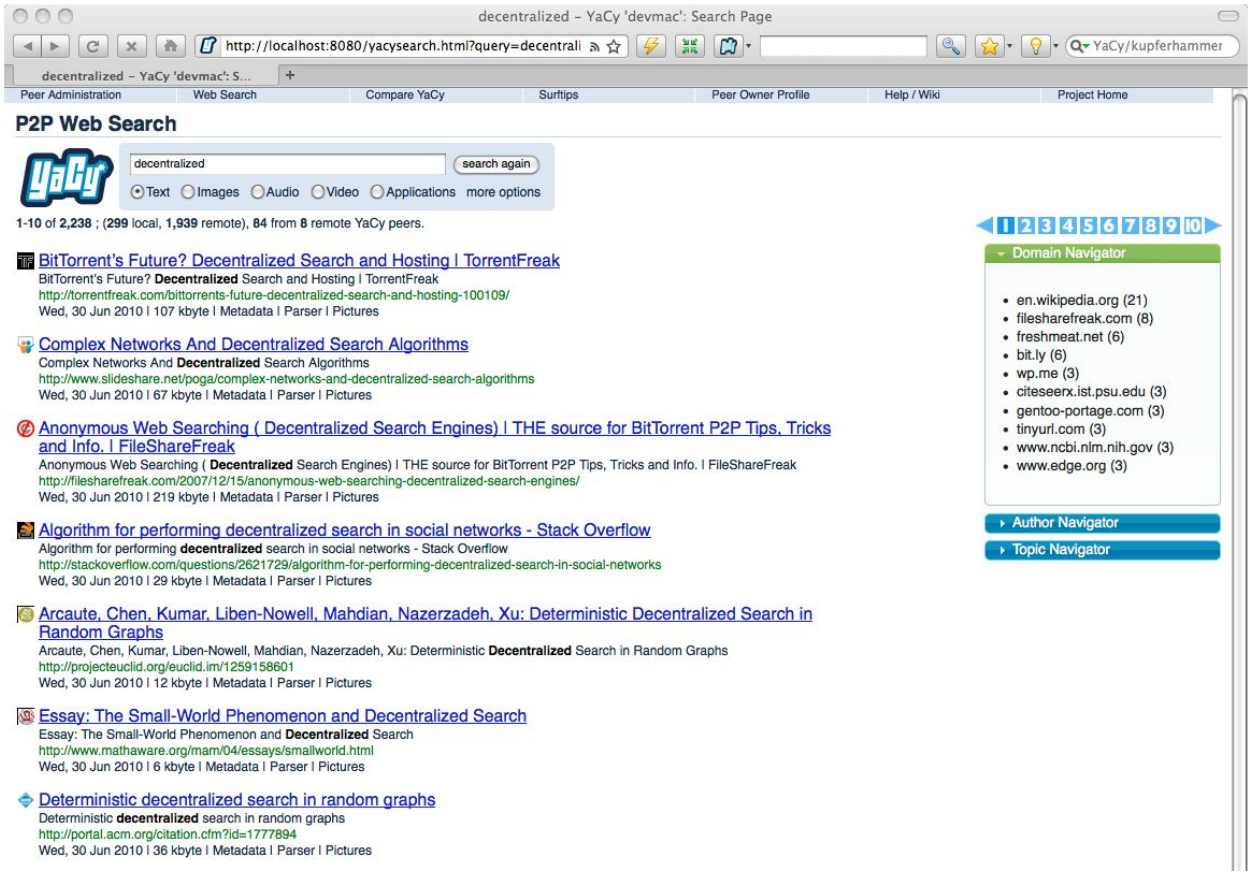


Abbildung 5: Yacy Suchmaske bei Ausgabe eines Suchbegriffs[22]

4.3.1 Vorteile

- Unabhängigkeit – kein Konzern verwaltet die Suchmaschine. Dadurch gibt es keine sponsored Links und keine Werbung.
- Unabhängigkeit der Internetnutzer, da kein Konzern zensiert oder ein bezahltes Ranking der Suchergebnisse durchführt.
- Rasantes Wachstum möglich: Einmal bei ein paar hundert Nutzern bekannt, kann sich die Suchmaschine schnell verbreiten. Mit der Verbreitung steigt die Qualität der Suchergebnisse und die Schnelligkeit der Suche.
- Ein Komplettausfall der Suchmaschine ist durch die dezentrale Anordnung nicht möglich. Dazu müsste jeder Nutzer offline sein, was unwahrscheinlich ist.
- Selbst Seiten aus dem Deep Web oder nicht öffentlichen Seiten werden zugänglich, da die Indexierung über den Proxy des Nutzers stattfindet. Crawler von öffentlichen Suchmaschinen finden diese Daten aus verschiedenen Gründen nicht.

[22] Enthalten in: yacy.de (2013)

- Yacy kann als private Suchmaschine im Internet benutzt werden, da jeder Nutzer seine eigene Indexierung erstellt. Damit würde jeder Nutzer über seine eigene, private Suchmaschine verfügen. Damit ist eine Verwendung in einem Unternehmens-Intranet ebenfalls möglich.

- Das Aufsetzen eines eigenen Webcrawler ist möglich, der an einer vorgegebenen Seite startet und alle Links nach Hyperlinkstruktur durchsucht und indexiert.

4.3.2 Nachteile

- Inhalte können nicht zensiert werden und somit stehen alle Inhalte für jeden zur Verfügung.

- Nur Nutzer, die online sind, liefern ein „Inhaltsverzeichnis". Sind wenige Anwender online, kann nur eine begrenzte Anzahl an Suchtreffern angezeigt werden.

- Fallen große Peers aus, werden ebenfalls nur wenige Ergebnisse gefunden.

- Da eine Suche über andere Peers läuft, ist die Geschwindigkeit im Vergleich zu kommerziellen Suchmaschinen wie Google niedrig.

- „Das YaCy-Protokoll funktioniert über einzelne HTTP-Requests, wodurch es eine höhere Latenz aufweist als UDP oder TCP mit dauerhaften Verbindungen."[23]

- Die Verteilung von Spam ist theoretisch sehr einfach, indem der Spammer eigene Peers bereitstellt, die falsche Ergebnisse ausgeben. „Falsche Suchergebnisse werden aber dadurch nicht möglich, da ein Peer durch Nachladen der Ergebnisseiten vor der Anzeige die Treffer verifiziert."[24]

4.3.3 Funktionsweise

Das Herzstück der Suchmaschine ist anders als bei anderen Suchmaschinen nicht eine zentrale Seite, sondern ein Computerprogramm, das auf fast allen Betriebssystemen läuft. Die Suche verläuft über eine lokale Webseite, die vom installierten Programm ausgeliefert wird. In der Browserleiste erscheint „localhost", damit greift der User auf seinen eigenen PC zu. Die Anzeige der Ergebnisse erfolgt hier wie gewohnt als HTML-Seite. Der Client sieht eine simple Suchmaske, in welcher nun der Suchbegriff eingegeben werden kann.

[23] de.wikipedia.org (2013)
[24] de.wikipedia.org (2013)

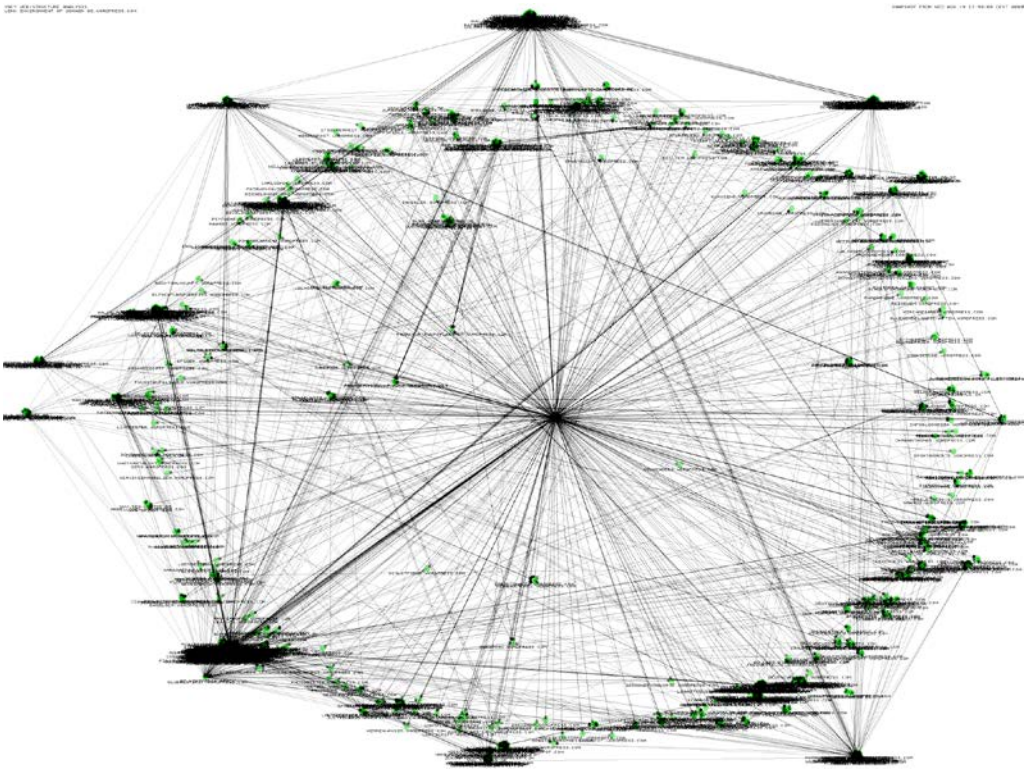


Abbildung 6: Link Topologie der Yacy Nutzer/Peers[25]

Die obige Abbildung zeigt die Link Topologie der einzelnen Peers. Je mehr Nutzer der Suchmachine hinzukommen und je mehr Seiten insgesamt indiziert wurden, desto größer die obige Struktur und desto höher die Leistungsfähigkeit der Suchmaschine im Ganzen.

Parallel zum P2P-System kann ein Proxyserver verwendet werden, der automatisch die besuchten Seiten indiziert. „Dies findet nicht bei Seiten statt, denen via GET oder POST weitere Daten übergeben werden oder die Cookies oder HTTP-Authentifizierung verwenden (z. B. Seiten in einem Login-Bereich). Somit ist sichergestellt, dass auch wirklich nur öffentlich zugängliche Daten indiziert werden.“ [26] Zusätzlich ist es möglich, private oder öffentliche Lesezeichen zu verwalten.

4.3.4 Fazit

Yacy ist ein Beispiel für eine sehr unkonventionelle Suchmaschine, die zwar von der Suchmaske den „Großen“ am Markt entspricht, im Kern aber ein völlig anderes Prinzip verfolgt. Diese Lösung wird generell als ausgereift bewertet, da alle notwendigen Funktionen bereitgestellt werden. Das Anpassen an unternehmensinterne Strukturen ist ohne Probleme umsetzbar, da Yacy über die Open Source Lizenz verfügbar ist. Der Literatur ist zu entnehmen, dass derzeit etwa 1.4 Mrd. Dokumente im Web indiziert wurden. Damit ist eine Implementie-

[25] deruwe.de (2009)
[26] de.wikipedia.org (2013)

rung in einem Intranet an diesem Punkt bedenkenlos. Das Anbinden von vielen Clients macht die Suchmaschine performanter. Je mehr Clients angeschlossen sind, desto mehr Ergebnisse liefert die Suche. Das Peer-to-Peer-Prinzip ermöglicht im Web die Erfassung und Indexierung des Deep Web, da die Indexierung über den Proxy des Nutzers stattfindet und einen klaren Vorteil gegenüber dem bekannten Prinzip ausmacht. Je nach Anzahl der Clients, die sich in einem Intranet bewegen, macht eine Verwendung von Yacy in einem Unternehmen mehr oder weniger Sinn. Je mehr Clients aktiv sind, desto eher lohnt sich die Betrachtung. Dies muss individuell geprüft werden und ist abhängig von der Anzahl der Mitarbeiter, die im täglichen Betrieb online sind.

Anforderungen der VERS:	Bedingt erfüllt, Perfromance Test notwendig
Besonderheiten:	Peer-to-Peer-Konzept erfolgreich umgesetzt
	Fast ausfallsicher, da der online-User gleichzeitig den Server bildet

4.4 Swish-e

Swish-e steht für "Simple Web Indexing System for Humans - Enhanced"[27]. Swish bedeutet auch „sausen“, was unter zwei Gesichtspunkten zutreffend ist: Die Suche liefert die Suchergebnisse verhältnismäßig schnell und die Indizierung soll ebenfalls rasch stattfinden können. Die Suchmaschine wurde 1995 von Kevin Hughes programmiert und 1997 von der Universität Berkeley weiterentwickelt und auf die Version Swish-e 2 gebracht. Seitdem soll sie kontinuierlich weiterentwickelt werden. Sie stellt eine eigene Suchmaschine dar und bietet den Vorteil, dass sämtliche Dateien indiziert werden können. Startet man das Programm in der Grundeinstellung ohne manuelle Konfiguration, verarbeitet das Programm nur TXT-, HTML- und XML-Dateien.[28] [29]

[27] swish-e.org (2013)
[28] Vgl. linux-magazin.de (2004)
[29] Vgl. pcwelt.de (2010)

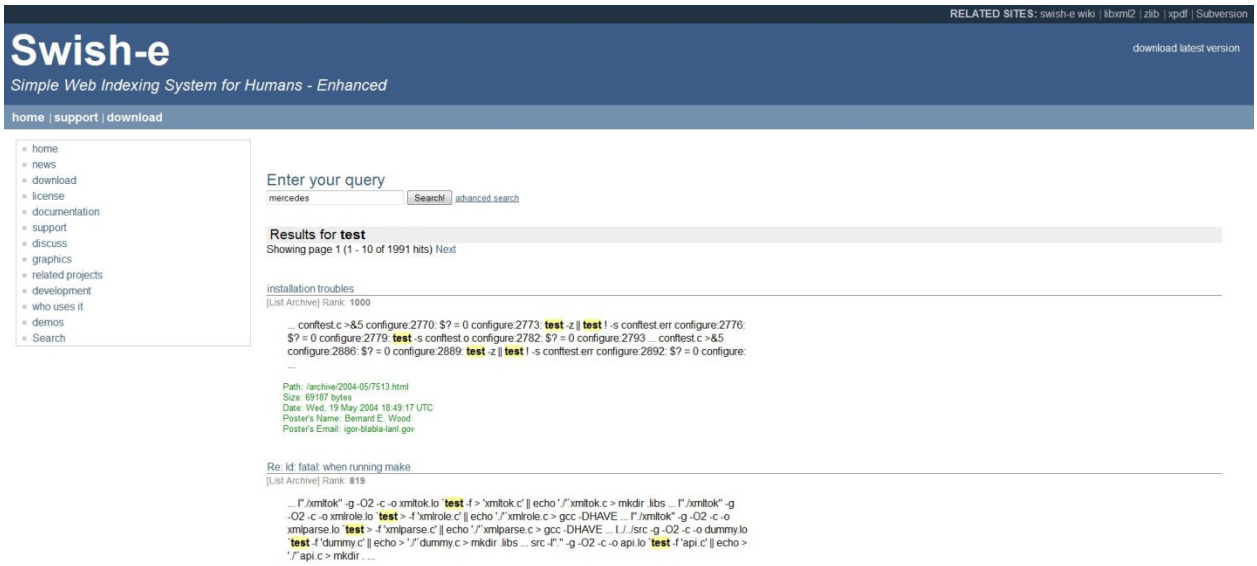


Abbildung 7: Suchmaske der Suchmaschine Swish-e[30]

4.4.1 Vorteile

- Schnelles Indizieren einer großen Anzahl von Dokumenten in verschiedenen Formaten (TXT, HTML und XML in der Grundkonfiguration).
- Benutzen von Filtern, um andere Formate wie PDF, mp3 oder PostScript zu indizieren
- Enthält einen Webspider, um Dokumente zu indizieren.
- Dokumente von einer relationalen Datenbank können gelesen und indiziert werden.
- Dokumenten Eigenschaften können im Index gespeichert und bei der Suche zurückgegeben werden (META Elemente).
- Dokumenten Zusammenfassungen können bei jeder Suche zurückgegeben werden.
- Erlaubt die „unscharfe“ Suchmethode (Fuzzy Search). Bei dieser Methode werden Rechtschreibfehler beim Eintippen in der Suchmaske erkannt und das eigentliche Wort trotzdem gesucht. Zusätzlich kann über eine optionale Synonym-Wortliste automatisch nach Synonymen gesucht werden.
- Suche nach Phrasen und Wildcards ist möglich.
- Die Suche kann limitiert werden, bspw. nur auf HTML Tags im Dokument
- Suchen können auf Teile einer Webseite begrenzt werden.

[30] Enthalten in: swish-e.org (2013)

- Die Ergebnisse können nach Relevanz oder nach einer beliebigen Zahl sortiert werden
- Errors in XML und HTML können angezeigt werden.
- Eine indizierte Datei kann auf jeder Plattform laufem, die Swish-e unterstützt.
- Es gibt eine Swish-e library, die in eigene Anwendungen portiert werden kann. Ein Perl Modul ist verfügbar, welches eine Standard API für den Zugriff bereitstellt.
- Es beinhaltet ein Beispielsuche-Skript, was den Suchbegriff enthält und die Phrasen hervorhebt.
- Sehr schnell bei der Indizierung von Texten, HTMLs und XMLs.[31]
- "Regular Expressions" existieren, um „Dokumente noch tiefgreifender zu filtern, als es über die Endung möglich wäre.“[32]
- Es besteht Support im Web.

4.4.2 Nachteile

- Ein Index sollte nach jedem Update des Swish-e Systems neu generiert werden, so der Entwickler
- Arbeitsspeicher und CPU leiden stark unter der schnellen Suche

4.4.3 Funktionsweise

Da in der Grundeinstellung nur von TXT-, HTML- und XML-Dateien ausgegangen wird, startet die Indizierung, indem jede Datei einem dieser Formate zugeordnet wird und einen Default-Index „verpasst“. So arbeitet sich das Programm vom Startverzeichnis Schritt für Schritt durch die anderen Verzeichnisse.

Im Folgenden wird die Konfiguration für das Listing 1 für HTML-Dateien erklärt. Die Darstellung erfolgt in Abbildung 1.

Da es weit mehr Dateitypen als die oben genannten drei gibt, muss mithilfe von Konfigurationsdateien festgelegt werden, welches Verzeichnis bearbeitet wird (»IndexDir«), wie die

[31] Vgl. swish-e.org (2013), Aussagen der Community
[32] winfwiki.wi-fom.de (2010)

indizierte Datei heißen soll (»IndexFile«) und auf welche Dateien sich die Indizierung beschränken soll (»IndexOnly«).[33]

Eingeleitet wird die Indizierung durch den Befehl »swish-e -c localweb.conf«. Fügt man dem Startbefehl noch den Zusatz »-v3« bei, wird angezeigt, was das Programm gerade genau indiziert.

Ruft man nun Swish-e mit anderen Parametern auf, kann in einem der eben erstellten Indexe gesucht werden. Ein Beispiel wäre der Befehl »swish-e -f localweb.index -w "Hallo"«. Dieser sucht nun in dem erstellten Index »localweb.index« nach dem Wort »Hallo«. Sollen komplexere Suchen durchgeführt werden, muss ich mit »*« als Wildcard und den logischen Operatoren »and«, »or« und »not« arbeiten. [34]

Durch Hinzufügen von Filtern können auch andere Formate verarbeitet werden. „Diese Filter verwandeln den neuen Dateityp in eines der drei bekannten Formate. Das funktioniert zum Beispiel mit Jpeg und Open Office, aber auch mit Mailboxen und Datenbanken. Meist ist dafür nicht viel mehr erforderlich, als den Namen des Konvertierungsprogramms in die Konfigurationsdatei einzutragen, nur selten muss man ein kleines Shellskript verfassen.“[35]

Listing 1: Mini-Konfiguration für HTML-Dateien

```
01 # localweb.conf
02 IndexDir   /srv/www/htdocs
03 IndexOnly  .html
04 IndexFile  ./localweb.index
```

Abbildung 8: Listing 1 Konfiguration[36]

4.4.4 Fazit

Swish-e bietet eine Reihe interessanter und nützlicher Funktionen. Es leistet mehr als eine einfache Indexsuche und ermöglicht das Suchen nach Dateiformaten über HTML hinaus. Experten meinen, der Umfang der Features ist dem von Nutch sehr ähnlich. Sehr angenehm ist, dass sogar explizit Datenbanken durchsucht werden können und die Suche stets schnell abläuft. Sehr userfreundlich ist nicht nur die einfache Bedienung, sondern auch die Möglich-

[33] Vgl. linux-magazin.de (2004)
[34] Vgl. linux-magazin.de (2004)
[35] linux-magazin.de (2004)
[36] Enthalten in: linux-magazin.de (2004)

keit der konfigurierbaren Filter-Funktion. Auf der anderen Seite ist die graphische Oberfläche nicht sehr userfreundlich. Experten meinen, man solle hier eher ein eigenes Interface davorsetzen. Der Einsatz für unternehmensinterne Seiten ist damit möglich und mithilfe von Swish-e auch zu empfehlen. Durch die vielen Funktionen und Konfigurationsmöglichkeiten sollte es einem internen Einsatz nur entgegenkommen. Dem gegenüber stehen relativ hohe Investitionen in Hardware, da das System vergleichsweise „Hardware-hungrig“ ist. Was genau die Hardware beansprucht, ob zum Beispiel der Crawler oder der Indexer, ist bisher nicht bekannt. Ein Einsatz von Swish-e stellt für das Unternehmen eine gute Alternative dar und sollte umsetzbar sein.

Anforderungen der VERS:	Erfüllt
Besonderheiten:	Schnelles Indexieren und Suchen
	Benutzen einer optionalen Filterfunktion für versch.
	Dateiformate
	Bestehender Web-Support

4.5 Terrier

Die Terrier Open Source Software ist seit November 2004[37] unter der Mozilla Public Lizenz (MPL) veröffentlicht und stellt eine modulare Plattform zur schnellen Entwicklung von Web, Intranet und Desktop Suchen dar. Sie wurde im Jahr 2000 entwickelt von der Universität Glasgow in Schottland und ermöglicht das Indexieren, Suchen und Bewerten von TREC Komponenten. Terrier ist in Java geschrieben. Der Innovationsreport schrieb 2005, es erfülle alle Voraussetzungen für eine europäische Antwort auf Google. Im Folgenden wird genauer auf die Funktionen und Risiken eingegangen.

[37] Vgl. University of Glasgow (2011)

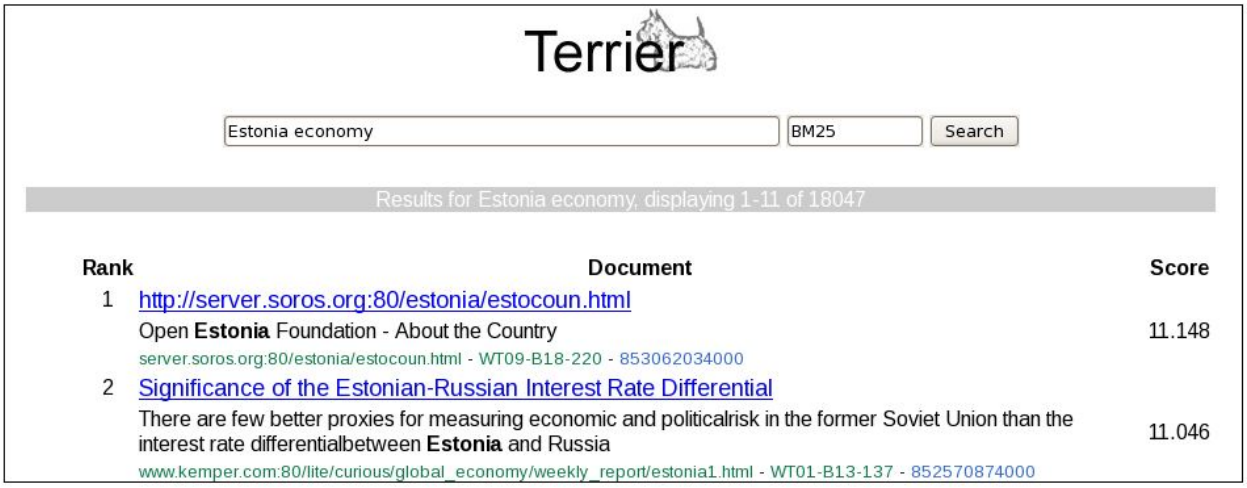


Abbildung 9: Web-Suchmaske bzw. customized Version der Suchmaschine Terrier[38]

4.5.1 Vorteile

- Folgende Out-of-the Box Vorteile:
    - Indexieren von Dokumentensammlungen, die mit Tags versehen sind (Beispiel Text REtrieval Conference (TREC) Testsammlung)
    - Indexieren verschiedener Formate möglich (HTML, PDF, Word, Excel und PowerPoint Dateien)
    - Unterstützen für verteiltes Indexieren in den Hadoop MapReduce Einstellungen
- Indexieren der Feldinformationen wie TITEL oder H1 HTML Tag
- Indexieren von Positionsinformationen eines Wortes oder Blocks
- Unterstützt verschiedene Kodierungen (UTF), um mehrsprachige Wiederherstellung zu ermöglichen. Damit werden auch die Umlaute im Deutschen darstellbar sein.
- Unterstützt das Ändern von benutzten Tokens
- Alternativ schnelleres „single-pass“ Indexieren möglich
- Desktop, Kommandozeilen und Web basierte Suchinterfaces sind verfügbar.
- Kann in Applikationen integriert werden wie der bereits existierenden Desktop-Suche

[38] Enthalten in: terrier.org (2011)

- Erweiterter Such-Sprachen Support wie Synonyme und +/- Operatoren (siehe http://terrier.org/docs/v3.5/querylanguage.html)
- Unterstützt das flexible Prozessieren von Ausdrücken durch eine „pipeline of components“[39] [40]

4.5.2 Nachteile

Selbst nach intensiveren Recherchen zu Terrier, konnten im Rahmen der Suchen über die Google Suche keine Erläuterungen zu den Nachteilen gefunden werden. Dies beweist allerdings im Gegenzug, dass die Community noch sehr klein ist und Terrier zu den kleineren, unbekannten Vertretern der Open Source Suchmachinen zählt. Dementsprechend gering wird der Support ausfallen. Dieser Punkt spricht ganz klar gegen einen Einsatz in einem Unternehmen.

4.5.3 Funktionsweise

Vorabinformationen: Für das Web-basierte Interface benötigt man etwas höhere Ressourcen als für Terrier selbst. Da mit JSPs gearbeitet wird, wird anstelle der JRE das komplette JDK benötigt. Des Weiteren gilt für Benutzer, die bereits mit Terrier arbeiten, dass normale Indexe nicht mit dem web-basierten Interface benutzt werden können, da im Standard Terrier Dokumente, Ausschnitte und Metadaten nicht mit gespeichert werden. Deshalb muss die Sammlung über Scratch indiziert werden.

Viele Punkte der Funktionsweise sind bereits in den Vorteilen enthalten. Gewisse Details sind nur schwer zu verstehen und erfordern technisches Hintergrundwissen, über diverse Programmiersprachen und Betriebssysteme heraus. Mit der obigen Darstellung werden alle wichtigen Punkte aufgelistet und im Rahmen unseres Verständnisses dargestellt. Detaillierte Informationen sind auf der Webseite von Terrier unter
http://terrier.org/docs/current/terrier_http.html zu finden.

4.5.4 Fazit

Terrier bietet eine Reihe zusätzlicher Features, die gerade im Bereich Intranet Suche von Vorteil sind. Ein vordefiniertes Web Interface ist vorhanden und kann verwendet werden. Die hierfür erforderlichen Voraussetzungen sind ohne Probleme zu erbringen. Terrier ist sehr flexibel und effizient aufgebaut und ermöglicht Veränderungen für Indizieren größerer Dokumentensammlungen. Laut dem Innovations-Report von 2005 hat diese Software auf der Internationalen Text Retrieval Konferenz 2004 alle anderen, teilnehmenden Technologien an

[39] terrier.org (2011)
[40] Vgl. mscerts.programming4.us

Leistung übertroffen. Die Konferenz wurde unter der Obhut des National Institute of Standards and Technology durchgeführt. Außerdem haben verschiedene staatliche Institutionen bekanntgegeben, dass sie ein Interesse an dem Einsatz von Terrier in ihrem Intranet haben. Die englische Computing Society und das italienische Ministerium für Kommunikation benutzten zu diesem Zeitpunkt diese Software für ihre internen Intranet-Suchen.[41] Leider gibt es derzeit keinen aktuelleren Innovationsreport zu diesem Thema. Dieser wäre sehr interessant und würde Feedback zu dem Einsatz und der Open Source Version von Terrier liefern. Dennoch kann anhand dieser Aussagen und der Evaluierung der Funktionen dieser Software zusammengefasst werden, dass Terrier eine ernst zu nehmende Alternative darstellt und durch seine Flexibilität und Anpassbar-Fähigkeit überzeugt. Ein Einsatz in dem Unternehmen ist nach jetzigem Kenntnisstand möglich und zu empfehlen. Abschließend wird die Bewertung von Terrier mit einem Zitat der Universität von Glasgow beendet, welches Aufschluss über die Charakteristika von Terrier gibt: „Terrier implements state-of-the-art indexing and retrieval functionalities, and provides an ideal platform for the rapid development and evaluation of large-scale retrieval applications. “[42]

Anforderungen der VERS:	Erfüllt
Besonderheiten:	Viele out-of-the-box Vorteile
	Ein vordefiniertes Web Interface ist vorhanden.

4.6 Lucene

Lucene ist ein in der Programmiersprache Java geschriebenes, performantes und plattformübergreifendes Framework, das eine Suchmaschinenbibliothek bereitstellt. Diese stellt Klassen und Funktionen zur Verfügung, aus denen eigene Volltextsuchmaschinen erstellt werden können.[43] Bekannt ist Lucene vor allem aufgrund der Realisierung der Suchfunktion in Wikipedia.[44]

4.6.1 Vorteile[45]

- Geringer RAM-Bedarf
- Leistungsstarke Indizierung

[41] Vgl.innovations-report.de (2005)
[42] terrier.org (2011)
[43] Vgl. Heidenreich, A. (o. J.), http://www.lucene.de/ (Stand: 27.12.2012)
[44] Vgl. o. V. (2012), http://de.wikipedia.org/wiki/Lucene (Stand: 22.12.2012)
[45] Vgl. o. V. (2012), http://lucene.apache.org/core/ (Stand: 18.12.2012)

- Skalierbar, das heißt, es können ohne Probleme mehr Dokumente, Wikiinstanzen, etc. hinzugefügt werden.
- Effiziente Suchalgorithmen, unter anderem aufgrund der Möglichkeit boolsche Operatoren und Wildcards bei der Suche einzusetzen.
- Einfache API, die es dem Entwickler erlaubt, die Suchmaschine weiterzuentwickeln.
- Wahlmöglichkeit zwischen selber entwickelter Suchsyntax und den umfangreichen Standardsuchoptionen.
- Lucene unterstützt nützliche Suchfunktionen wie beispielsweise „Fuzzy"-Suche, Distanzsuche oder auch Verstärkungsfaktoren.[46]
    - „Fuzzy"-Suche bedeutet, dass bei einer Suchanfrage auch nach synonymen Begriffen gesucht wird oder bei Tippfehlern des Nutzers trotzdem nach dem korrekten Begriff gesucht wird.
    - Distanzsuche ermöglicht das Suchen von mehreren Begriffen, die in einem Text nicht direkt hintereinander stehen.
    - Verstärkungsfaktoren können eingesetzt werden, um eine Gewichtung der Begriffe zu Erreichen.
- Datenbankübergreifende Suchanfragen sind durchführbar.
- Plattformunabhängigkeit; die Suchmaschine läuft nicht nur auf einem Betriebssystem oder Gerät, sondern mit entsprechender Konfiguration auf allen Geräten und Systemen.
- Flexible Konfigurierbarkeit

4.6.2 Nachteile

- Javaprogrammierkenntnisse sind nötig
- Nicht deutschsprachig

4.6.3 Funktionsweise

Vor der eigentlichen Suche wird ein invertierter Index erstellt. Dabei wird eine Liste erstellt, die die Informationen enthält, welche Wörter/Wortbestandteile oder auch Token in welchem

[46] Vgl. o. V. (o. J.), http://dev.innovationgate.de/doc-wga41/html/default/dr.qr.lucene.de.html (Stand: 26.12.2012)

Dokument vorkommen, wie oft und wo. Für das Extrahieren der Inhalte aus HTML, XML oder anderen Dateiformaten für das Indexdokument gibt es eine große Auswahl an Bibliotheken. In der untenstehenden Abbildung werden exemplarisch die Parser Tika und POI verwendet. Bei der Suche wird die Indexliste dann mithilfe des IndexSearchers nach den relevanten Query Strings durchsucht. Die relevanten Query Strings verweisen dann auf die Dokumente, in denen sie vorkommen.

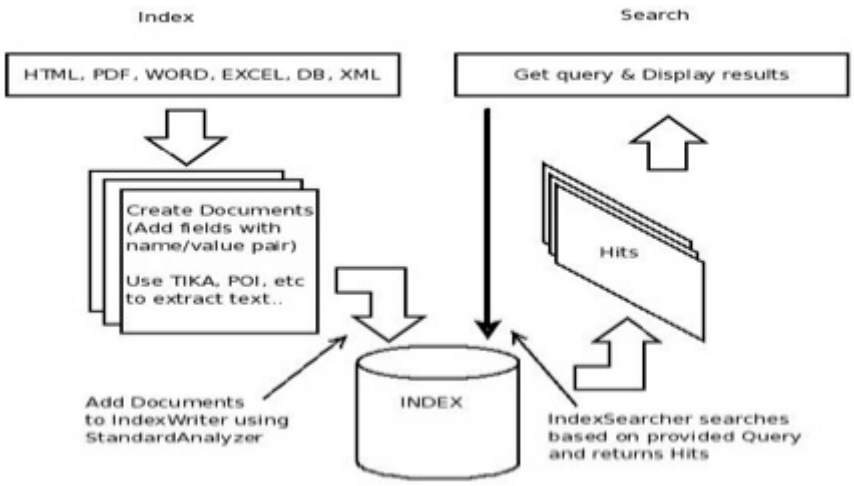


Abbildung 10: Indizierung und Suche in Lucene[47]

4.6.4 Fazit

Lucene bietet eine Vielzahl an Klassen und Funktionen für erfolgreiche benutzer- bzw. projektspezifische Eigenentwicklungen. Unterstützt wird der Entwickler dabei durch die große und aktive Community.[48]

Allerdings existieren bereits gute auf Lucene basierende Weiterentwicklungen, von denen zwei im Folgenden näher erläutert werden. Für diese Suchmaschinen sind wesentlich weniger Javaprogrammierkenntnisse erforderlich als für eigene Weiterentwicklungen. Lucene erfüllt sowohl die von der VERS BW genannten Kriterien, als auch die Kriterien, die aus den Überlegungen der Autoren stammen. Deshalb und aufgrund der großen Bekanntheit und aktiven Community eignet sich Lucene für die Implementierung bei der VERS BW und ist der zweite, der insgesamt drei Favoriten.

[47] Vgl. Sitaula, S. (2012), http://blog.cloudera.com/blog/2012/01/hadoop-for-archiving-email-part-2/ (Stand: 20.01.2013)
[48] Vgl. o. V. (o. J.), http://lucene.apache.org/core/developer.html (Stand: 19.12.2012)

Anforderungen der VERS:	Erfüllt
Besonderheit:	Große und aktive Community
	Basis für eigene Suchmaschinen

4.7 Elastic Search

Elasticsearch ist eine verteilte (Verteilt = Indexe können in Shards aufgeteilt werden), dokumentenorientierte und RESTful Suchmaschine, die auf Apache Lucene aufgesetzt ist und unter den Bedingungen von Apache License 2.0 veröffentlicht wurde.

Sie ist prädestiniert für den Cloud-Einsatz[49] und wird aufgrund derselben Backend-Suchmaschine Apache Lucene häufig mit Solr verglichen.[50]

ElasticSearch wurde in Java entwickelt und benötigt als Laufzeitumgebung Java der Version 6 oder höher. ElasticSearch ist auf allen Betriebssystemen, die javafähig sind einsetzbar, ist schemafrei und verwendet das Datenformat JSON.[51]

Die Suchmaschine ist in den drei folgenden Varianten verfügbar:

- Fertige Distribution
- Quellcodearchiv
- Quellcode

4.7.1 Vorteile[52]

- Gute horizontale Skalierbarkeit (= Durch zusätzlichen Knoten wird Daten- und Rechenlast automatisch verteilt) ohne großen Konfigurationsaufwand und zugleich einfache Bedienung

[49] Vgl. Stäuble, M. (2010), http://it-republik.de/jaxenter/artikel/ElasticSearch---Verbindung-von-Lucene-und-Cloud-3134.html (Stand:19.12.2012)
[50] Vgl. Karich P. (2012), http://www.heise.de/ix/artikel/Immer-flexibel-1440270.html (Stand: 25.12.2012)
[51] Vgl. o. V. (2013), http://en.wikipedia.org/wiki/ElasticSearch (Stand: 25.12.2012)

[52] Vgl. Lieweries, M. (2012), http://maiks-chaos.blogspot.de/2012/01/elasticsearch-die-moderne-art-zu-suchen.html (Stand: 26.12.2012)

- Man kann für jeden Index eigene Einstellungen konfigurieren. Dabei werden vorherige Einstellungen überschrieben. Zu diesen Einstellungen gehört beispielsweise, ob man Memory oder File-based Storage möchte.

- Eingefügte JSON-Dokumente werden automatisch indiziert und Datentypen automatisch erkannt und entsprechend behandelt. Dennoch erlaubt ElasticSearch die vollständige Kontrolle darüber, wie ein JSON-Dokument auf per Typ und per Indexlevel in die Suchmaschine gemappt wird. [53]

- Elasticsearch ist mandatenfähig (= Mehrere Benutzer/Kunden, jeder kann nur seine eigenenen/für ihn autorisierten Daten sehen und konfigurieren).

- Ein ausgeglügelter Support, der benutzerdefinierte Skripte und andere Features ermöglicht ist vorhanden, kann aber ausgeblendet werden, um einfache Abfragen zu ermöglichen. [54]

- Hohe Performance/Schnelligkeit aufgrund einfacher vertikaler Skalierung.

- Hohe Ausfallsicherheit, da durch die Bildung von Shards die Daten auf mehrere Server verteilt werden können und durch Replikas Redundanzen vorhanden sind.

- ElasticSearch bietet nützliche Zusatzfunktionen. Beispielsweise die Funktion „Fuzzy-Logik“, mit der es möglich ist, dass die Maschine Tippfehler, Falschschreibung oder Synonyme erkennt oder „Boosting“-Funktionalitäten, die es ermöglichen, dass bestimmte Dokumente in der Ergebnisliste weiter oben erscheinen.

- Da ElasticSearch dokumentenorientiert ist und das Datenformat JSON verwendet muss ein Attribut eines Dokuments nicht Typ eines primitiven Datentyps sein sondern kann selbst auch ein Dokument oder eine Liste von Werten oder weiteren Dokumenten sein. Man spricht auch von dynamischer Datenstrukturierung.

- Zusätzlich zu REST steht auch eine Programmierschnittstelle für Java und Groovy zur Verfügung.

- Weniger Javakenntnisse als für Lucene notwendig, da es als „Out-of-the-box“-Lösung verwendet werden kann.

[53] Vgl. o. V. (2013), http://www.elasticsearch.com/products/elasticsearch/ (Stand: 02.01.2013)
[54] Vgl. o. V. (o. J.), http://www.elasticsearch.org/ (Stand: 02.01.2013)

4.7.2 Nachteile[55]

- Hohe horizontale Skalierbarkeit ist nur gegeben, wenn man bereits bei der Indexkonfiguration berücksichtigt, genügend Shards und Replizierung zu ermöglichen. Ansonsten bringt zusätzliche Rechenleistung keinen Mehrwert.

- Tutorials, Guides und andere Lehr- und Informationsmaterialien sind vorwiegend nicht in deutscher Sprache verfügbar.

- In der Vergangenheit gab es nur einen Hauptentwickler. Allerdings wurde dieses Problem bereits dadurch abgeschwächt, dass mittlerweile durch die Firma ElasticSearch mehrere Entwickler daran arbeiten.

4.7.3 Funktionsweise und Aufbau

Wenn Dokumente der Suchmaschine übergeben werden, erstellt Apache Lucene, auf das ElasticSearch aufsetzt, einen invertierten Index. Dies wird mit JSON über HTTP durchgeführt. Dabei verweisen Token(= Felder mit Typ String/Wörter) auf diejenigen Dokumente, in denen sie vorkommen und speichern gegebenenfalls noch Metadaten wie z.B. Positionen ab. Beim Indexieren wird immer die Dokument-ID, der Typ und der Index angegeben, nicht existierende Komponenten werden dabei automatisch von ElasticSearch erzeugt.

Die eigentliche Suche kann durch einen Lucene-basierenden Query String oder durch eine JSON-basierende Query DSL erfolgen.

4.7.4 Fazit

Die umfangreichen Tutorials und Guides auf der Startseite von ElasticSearch ermöglichen eine schnelle Installation und einen einfachen Einstieg. An der Weiterentwicklung und dem kontinuierlichen Ausbau der Dokumentation wird aktiv gearbeitet.

Als Weiterentwicklung von Apache-Lucene bietet ElasticSearch umfangreiche Funktionalitäten, ist über Module erweiterbar und dennoch im Vergleich zu anderen Suchmaschinen einfach zu bedienen. Durch die Mandatenfähigkeit ist eine Nutzung für eine unternehmensinterne Suche mit unterschiedlichen Berechtigungsleveln möglich und der Schutz der Daten vor unberechtigtem Zugriff gewährleistet. ElasticSearch ist generell eine empfehlenswerte Suchmaschine, allerdings ist für eine benutzerspezifische Konfiguration Lucene empfehlenswerter und Solr, die ElasticSearch sehr ähnlich ist bringt aufgrund der großen Verbreitung mehr Erfahrungswerte und eine größere Community mit. Daher scheidet ElasticSearch als Favorit für die VERS BW aus.

[55] Vgl. o. V. (2012), http://stackoverflow.com/questions/10213009/solr-vs-elasticsearch (Stand: 26.12.2012)

Anforderungen der VERS:	Erfüllt
Besonderheit:	„Out-of-the-box“-Lösung

4.8 Solr

Solr basiert ebenso wie ElasticSearch auf Apache Lucene. Es handelt sich hierbei um eine voll ausgewachsene in Java geschriebene Volltextsuchmaschine, die für hohe Last und Verfügbarkeit optimiert ist. Es stellt verschiedene Funktionalitäten wie z.B. eine verteilte Indizierung, Replikation, lastausgleichende Abragen, automatisierte Ausfallsicherung und Wiederherstellung sowie eine zentralisierte Konfiguration bereit.[56] Solr besitzt außerdem Funktonalitäten die es in Lucene nicht gibt. Dazu gehören beispielsweise Warm-up, Caching und Clustering. Solr bietet eine Reihe von Musterbenutzeroberflächen von Solritas. Der folgende Screenshot zeigt eine Version davon.

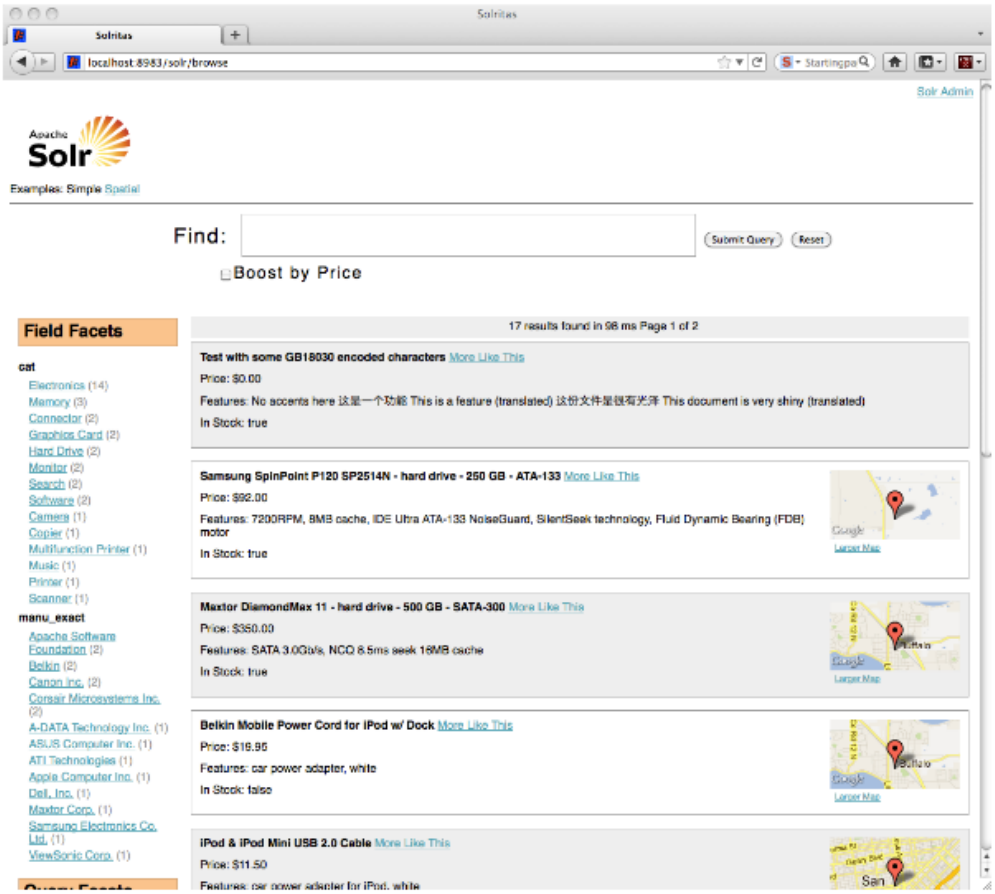


Abbildung 11: Screenshot einer Suchanfrage[57]

[56] Vgl. o. V. (2012), http://lucene.apache.org/solr/ (Stand: 27.12.2012)
[57] Vgl. o. V. (2012), http://lucidworks.lucidimagination.com/display/solr/Overview+of+Searching+in+Solr (Stand: 19.12.2012)

4.8.1 Vorteile[58]

- Besitzt umfangreiche Pluginarchitektur, falls eine höhere Kundenanpassung benötigt wird
- Hervorhebung der Treffer, auch Highlighting genannt
- Facettierte Suche, ermöglicht eine Erweiterung der Suche dahingehend, dass man nur in bestimmten Genres sucht, um die Suche einzugrenzen und so zu beschleunigen. Dazu muss man aber auch schon bei der Indizierung die Genres berücksichtigen.
- Nahezu Echtzeitindizierung
- Dynamische Clusterbildung, bedeutet die Cluster können verändert warden und sind nicht statisch.
- Datenbankintegration möglich
- Hohe Zuverlässigkeit
- Skalierbar
- Fehlertolerant, dank Fuzzy-Search
- Wenig Javakenntnisse sind ausreichend, da es als “out-of-the-box”-Lösung implementierbar ist.

4.8.2 Nachteile

- Es sind weniger Javakenntnisse nötig als für Lucene. Es werden aber dennoch Programmierkenntnisse benötigt.
- Vorwiegend englisches Informationsmaterial verfügbar

4.8.3 Funktionsweise

Solr basiert ebenso wie ElasticSearch auf Lucene. Daher ähnelt die Indizierung und Suche größtenteils der von Lucene. Solr besteht aus einer dokumentenorientierten Architektur und stellt eine Webschnittstelle zum Hinzufügen und Ändern von Dokumenten bereit. Das Hinzufügen von Dokumenten, also die Indizierung, erfolgt über XML, JSON, CSV oder binär über HTTP. Die Abfragen laufen über HTTP und liefern XML, JSON, CSV oder binäre Ergebnisse.

[58] Vgl. o. V. (2012) http://lucene.apache.org/solr/ (Stand: 27.12.2012)

Dabei sind die Dokumente nicht Schema-frei. Jedes Feld, das belegt wird muss vordefiniert werden.

„In Lucene enthalten Felder grundsätzlich Text. In Solr sind Felder hingegen typisiert, ähnlich wie Variablen in Java. Die klassischen Typen wie Date, Float, Double, Integer, Long, Boolean und verschiedenen Arten von Text-Typen sind vordefiniert, eigene können ebenfalls verwendet werden.

Jedes Feld besitzt einen Typ, gegen den der Wert des Feldes beim Hinzufügen geprüft und gegebenenfalls abgelehnt wird. Des Weiteren wird in der Deklaration des Feldes festgelegt, welche Lucene-spezifischen Verarbeitungsschritte beim Indizieren vorgenommen werden sollen, ob eine Zerlegung stattfindet und ob es "suchbar" sein soll. Außerdem können weitere spezialisierte Schritte, so genannte Filter, definiert werden, die bei Indizierung und Suche berücksichtigt werden sollen. Zum Beispiel können so die eingehenden Daten zu Kleinbuchstaben umgewandelt werden oder es lassen sich Wörter ("Stopwords") definieren, die auf keinen Fall in den Index aufgenommen werden sollen.“[59]

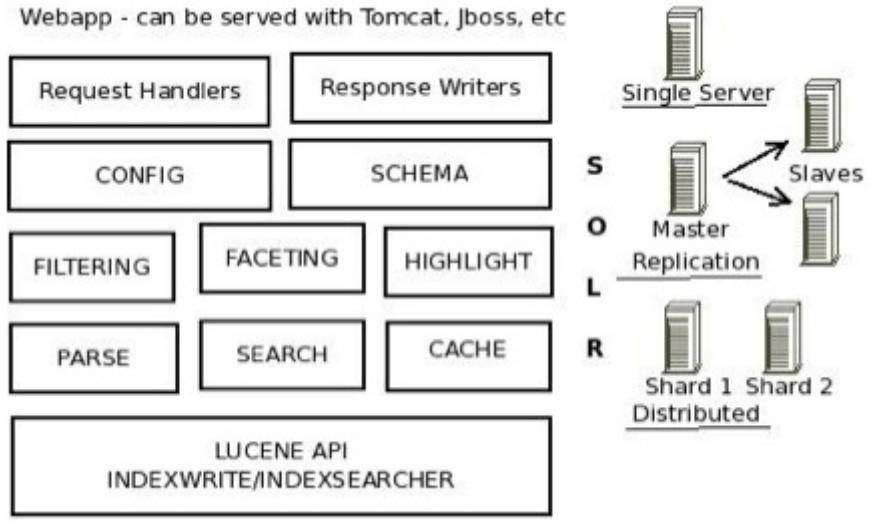


Abbildung 12: Architektur und Komponenten von Solr[60]

Die untenstehende Abbildung zeigt die in Solr enthaltenen Komponenten und die generelle Architektur. Man erkennt, dass Solr die Indizierung von Lucene mit weiteren Funktionen, wie z.B. Highlighting ergänzt. Highlighting bedeutet, dass in den Ergebnissen einer Suche die Wörter, nach denen gesucht wurde, hervorgehoben werden.

[59] Fondermann, B. (2009), http://it-republik.de/jaxenter/artikel/Solr-der-Suchserver-2176.html (Stand: 28.12.2012)

[60] Vgl. Sitaula, S. (2012), http://blog.cloudera.com/blog/2012/01/hadoop-for-archiving-email-part-2/ (Stand: 20.01.2013)

Die request handlers bearbeiten, wie der Name schon sagt, die Suchanfragen und die response writers bearbeiten die Suchergebnisse nach den konfigurierten Anforderungen, wie z.B. Sortierung nach Relevanz.

Bei der Replikation, also bei der mehrfachen Speicherung von Daten an verschiedenen Orten und deren Synchronisation arbeitet Solr nach dem sogenannten Master-Slave-Prinzip. Das bedeutet, es wird eine Master-Kopie erstellt und weitere, davon abhängige Replikate, die sogenannten Slaves.

Zusätzlich ermöglicht Solr die Aufteilung des Indexes in Shards, um zu lange Indexe zu verhindern. Das ist besonders bei einer Vielzahl an indexierten Dokumenten von Vorteil, da mehrere kürzere Indexe schneller durchsucht werden können als ein Großer.

4.8.4 Fazit

Solr ist einfach zu installieren und setzt ein installiertes Java sowie einige Programmierkenntnisse voraus. Es besitzt alle Vorteile und Funktionalitäten von Apache Lucene und ergänzt diese. Solr ist aufgrund machtvoller externer Konfiguration auch an Applikationen anbindbar, die nicht auf Java basieren und somit vielseitig einsetzbar. Durch die Optimierung in den Bereichen Verfügbarkeit und Auslastung eignet sich Solr für eine Intranetsuche, da die Verfügbarkeit auch bei hoher Auslastung durch die Mitarbeiter zu Stoßzeiten gewährleistet ist. Solr erfüllt alle Kriterien des Anforderungskataloges und ist unter den untersuchten Suchmaschinen die am meisten verbreitete. Da sie ein Apacheprodukt ist, ist ebenso wie bei Lucene und ElasticSearch das zukünftige Bestehen und die zukünftige Weiterentwicklung der Suchmaschine gewährleistet. Aus all diesen Gründen ist Solr der ausgewählte dritte Favorit für die VERS.

Anforderungen der VERS:	Erfüllt
Besonderheit:	„Out-of-the-box“-Lösung
	Optmimierte Verfügbarkeit und Auslastung
	Große und aktive Community

4.9 Hsearch

Hsearch ist eine NoSQL Suchmaschine, die unter der Verwendung von Hadoop entwickelt wurde. Es ist unter der Apache Lizenz verfügbar und in Java geschrieben. Hsearch eignet sich vor allem wenn man Echtzeitsuche bei Big Data benötigt. Es handelt sich um eine ver-

teilte Suchmaschine, die sich sowohl für die Suche nach strukturiereten als auch für die Suche nach unstrukturierten Inhalten eignet.[61]

4.9.1 Vorteile

- Kontinuierliche Indexaktualisierung
- Mithilfe von Multimaschinen ist eine parallele Indizierung möglich, das eignet sich besonders für große Datenmengen.
- Unterstützt XML
- Automatische Replikation, das bedeutet es warden automatisch Sicherungskopien erstellt.
- Automatisches Sharding[62], um die Rechenlast bzw. die Indexe aufzuteilen.
- Verschiedene Darstellungsmöglichkeiten der Suchergebnisse
- Stellt viele Funktionalitäten, wie z.B. „Fuzzy Search“ oder die Gewichtung der Suchergebnisse zur Verfügung

4.9.2 Nachteile

Aufgrund wenig vorhandener Literatur zu dieser Suchmaschine wurden keine Nachteile gefunden. Dies bedeutet nicht, dass es keine gibt, lediglich, dass diese nicht im Internet auffindbar sind.

4.9.3 Funktionsweise

Hsearch basiert auf der HBase Plattform. Der vollständige Index ist in einer HBase Tabelle gespeichert und enthält somit die Hbasefunktionalitäten.

Bei der Indizierung können die Dokumente, die hinzugefügt werden sollen, aus verschiedenen Quelltypen stammen. Beispielsweise aus Dateisystemen, relationalen Datenbanken, Mailservern, Webseiten, XML Dateien. Dabei erlaubt das Framework den externen Systemen die Inhalte in Echtzeit über REST-basierende oder über Java-basierende Programmierschnittstellen hinzuzufügen.

Die Sicherheit wird unter anderem durch eine Zugriffskontrolle auf Dokumentenebene gewährleistet. Die Kontrolle basiert auf UserIDs, Rollen, Teams, Organisationen oder Abteilun-

[61] Vgl. o. V. (2010), http://bizosyshsearch.sourceforge.net/ (Stand: 28.12.2012)
[62] Vgl. o. V. (2010), http://bizosyshsearch.sourceforge.net/ (Stand: 28.12.2012)

gen. Außerdem stellt Hsearch benutzerspezifische Schnittstellen zur Verfügung, über die sich die Nutzer einloggen können.[63]

4.9.4 Fazit

Hsearch wurde entwickelt um in einer verteilten Umgebung genutzt zu werden. Es eignet sich besonders für die Nutzung durch Unternehmen, da es hinsichtlich Vielseitigkeit, Skalierbarkeit und Sicherheit Standardanforderungen von Unternehmen erfüllt. Zusätzlich bietet Hsearch Unternehmen, die sich nicht selber um die IT kümmern möchten, gegen Bezahlung einen Support an.[64] Allerdings ist Hsearch eher für Unternehmen geeignet die es mit unglaublich großen Datenmengen zu tun haben, was bei dem Wikisystem der VERS nicht der Fall ist. Außerdem gibt es wenig Foren und nur eine kleine Community, die den Entwicklern Unterstützung bieten. Es gibt natürlich kostenpflichtigen Support, doch es wäre eine Verschwendung von Ressourcen, da die VERS BW selber über ein IT-Team verfügt. Bei anderen Suchmaschinen ist die VERS BW mithilfe ihres Know-hows daher auf jeden Fall in der Lage, diese selber zu implementieren und zu warten. Bei Hsearch ist diese Sicherheit aufgrund mangelnder Erfahrungsberichte nicht gegeben.

Anforderungen der VERS:	Teilweise erfüllt
Besonderheit:	Eignet sich für Suche in „Big Data“
	Nicht weit verbreitet

[63] Vgl. Bizosys Technologies (2010), http://www.hadoopsearch.net/hsearchdatasheet.pdf (Stand: 29.12.2013)
[64] Vgl. o. V. (2011), http://yourstory.in/2011/06/start-up-profile-hsearch-bizosys-technologies/ (Stand: 29.12.2013)

5 Implementierung und Integration

Das Implementieren einer Suchmaschine in eine eigene Umgebung und die Integration inklusive Anbindung an vorhandene Datenquellen variiert ja nach ausgewählter Suchmaschine sehr stark. Viele Anbieter werben mit einer sogenannten out-of-the box Fähigkeiten für einfaches Anbinden. In diesem Fall soll es für den User relativ schnell und einfach sein, diese Funktionen zum Laufen zu bringen. Out-of-the box bedeutet hier, dass der Anwender die Software downloaden, die Installierung durchführen muss und nach einem Neustart bereits einen Teil der Software ohne weitere Anstrengungen nutzen kann. Meist wird explizit angegeben, welche Funktionen auf diese Weise zur Verfügung stehen. Dieser „Schnellstart“ wird nicht von jedem Hersteller bereitgestellt und bietet meistens nur einen sehr begrenzten Funktionsumfang. Generell müssen bei einer Integration in eigene Umgebungen/Applikationen unter anderem folgende Fragen beachtet und beantwortet werden:

Welche(s) Betriebssystem(e) werden unterstützt?

In welcher Sprache ist die Software geschrieben?

Welche Laufzeitumgebung wird benötigt?

Stehen ausreichend Programmierressourcen zur Verfügung?

Ist eine Implementierung ohne besonderen Aufwand möglich?

Können bestehende Wiki-Systeme eingebunden werden? oder je nach Anforderung:

Können Features wie Internet/Intranet-Suche integriert werden?

Sollen verschlüsselte Dokumente gefunden werden?

Wird die Datenanbindung funktionieren? Welche Datenbanken unterstützt die Software?

Welche Hardware Ressourcen werden benötigt? Investitionen nötig?

Wie können Sicherheitsaspekte wie Nutzergruppen implementiert werden(Rollenverteilung)?

Ist die Software auf den geschätzten traffic ausgelegt?

Ist das bestehende/Software beinhaltende Interface ausreichend?

Wie kommen Mitarbeiter damit zurecht?

Wie viel Zeit muss ein nicht-IT Mitarbeiter für eine Suche aufwenden?

Muss eine eigene GUI programmiert werden?

Wie groß wird der Wartungsaufwand sein (Zeit/Skills/Kosten)?

Ist das System auf die vorhandenen Datenmengen ausgelegt?

Ist das System skalierbar genug?

Welche zusätzlichen Anforderungen sind zu beachten?

Diese Auswahl an Fragen zeigt auf, wie viele Aspekte vor und während einer Integration in ein eigenes Netzwerk/System beachtet werden müssen. Dennoch ist selbst nach rücksichtsvoller Untersuchung nicht auszuschließen, dass verschiedene Probleme bei einer Integration auftreten können. Zusätzlich kann den Versprechungen der Hersteller ihrer out-of-the box Funktionen nicht vertraut werden. Nichtsdestotrotz ist große Sorgfalt bei einer Integration gefragt.

5.1 Beispiel Integration von Swish-e in ein Intranet

Im Folgenden wird beispielhaft erklärt, wie eine Integration von Swish-e in ein eigenes Netz möglich ist und welche Probleme dabei auftreten können. Das Beispiel ist einer Beschreibung von Stefan Macke entnommen.

Die Integration wird für ein Intranet erklärt, welches auf Open Suse 10.1 64Bit Server läuft. Zunächst müssen die Quellen von der Webseite geladen und entpackt werden. Anschließend muss wie beschrieben die Installation durchgeführt werden. Dies geschieht in der Kommandozeile mithilfe folgender Anweisungen: „./configure“, „make“, „make check“ und „make install“. Nach erfolgreicher Installation kann ein erster Test gestartet werden (Befehl „swish-e –v“). Nun erscheint die Meldung: swish-e: error while loading shared libraries: libswish-e.so.2: cannot open shared object file: No such file or directory.[65] Problem hier ist, dass das benötigte Modul nicht in dem library-include Pfad liegt. Mit „strace swish-e“ kann überprüft werden. Gesucht wurde in diesem Beispiel unter /usr/lib64/ und anderen Pfaden. Leider lagen die Module allerdings unter dem Pfad /usr/local/lib. Nun muss der Pfad in die /etc/ld.so.conf eingetragen werden. Anschließend lässt man ldconfig laufen und das Problem sollte behoben sein. Um nun einen ersten Test durchzuführen, wird in der Beschreibung empfohlen das Apache Manual zu indexieren. Folgende Meldung erscheint: „Can't locate LWP/RobotUA.pm in @INC.“[66] Hier fehlt nun das Perl Modul LWP. Das Problem wird behoben, indem man ncftpget installiert und cpan konfigurieren lässt. Eine Indexierung ist nun

[65] blog.stefan-macke.com (2006)
[66] blog.stefan-macke.com (2006)

möglich. In diesem Beispiel verfügt das Intranet über verschiedene Teilbereiche. Eine Suche mit getrennten index Dateien ist möglich mit folgender Einstellung in der Konfigurationsdatei .swishcgi.conf: swish_index => ['/srv/www/swish-e/seite1.index', '/srv/www/swish-e/seite2.index'],

Nun kann endgültig mit einer ersten Websuche gestartet werden. Eine detaillierte Installationsanleitung hat Swish-e bereitgestellt unter http://swish-e.org/docs/install.html.

6 Bewertung

6.1 Beurteilung

Die vorliegende Arbeit wurde durchgeführt, um eine Opensource Suchmaschine zu finden, welche die VERS BW zur internen Suche innerhalb eines vorhandenen Wikisystems implementieren und nutzen kann. Daher wurde auf Basis eines Kriterienkataloges eine Marktanalyse durchgeführt. Der Schwerpunkt lag dabei darin, nicht nur die vorgegebenen Suchmaschinen zu betrachten, sondern eine große Bandbreite an verschiedenen Konzepten zu integrieren. Bei der Analyse wurden Vor- und Nachteile ausgewählter Suchmaschinen aufgelistet, die Funktionsweise betrachtet und der Gesamteindruck sowie die Eignung zur Implementierung bei der VERS BW bewertet. Es handelt sich mit einer Ausnahme bei allen untersuchten Maschinen um Opensource Suchmaschinen, da dies eine Anforderung der Auftraggeber war. Diese Ausnahme wurde in die Analyse integriert, um auch einen Vertreter der „grünen“ Suchmaschinen zu bewerten. Da aktuell keine „grünen“ Opensource Suchmaschinen existieren, haben die Autoren auf Ecosia, eine nicht opensourcebasierende Suchmaschine, zurückgegriffen.

Die naheliegende Idee, eine Nutzwertanalyse durchzuführen, wurde verworfen, da eine objektive Bewertung nicht möglich ist. Das liegt daran, dass jedes Unternehmen unterschiedliche Anforderungen auch anders gewichtet. Des Weiteren sind alle in der Marktanalyse aufgelisteten Suchmaschinen nicht direkt am Kunden getestet worden. Aufgrund dieser unterschiedlichen Präferenzen und der bisher nicht durchgeführten Tests ist es nicht möglich, eine Suchmaschine als die Geeignetste oder Beste zu deklarieren.

Die Marktanalyse ergab außerdem, dass der Funktionalitätenumfang der Suchmaschinen etwa proportional zu den notwendigen Programmierkenntnissen ist. Die untenstehende Abbildung verdeutlicht diesen Sachverhalt. Man benötigt beispielsweise für die Standardversion von Lucene wenig Programmierkenntnise, während man für ein benutzerdefiniertes Projekt auf Basis von Lucene umfangreiche Programmierkenntnisse benötigt.

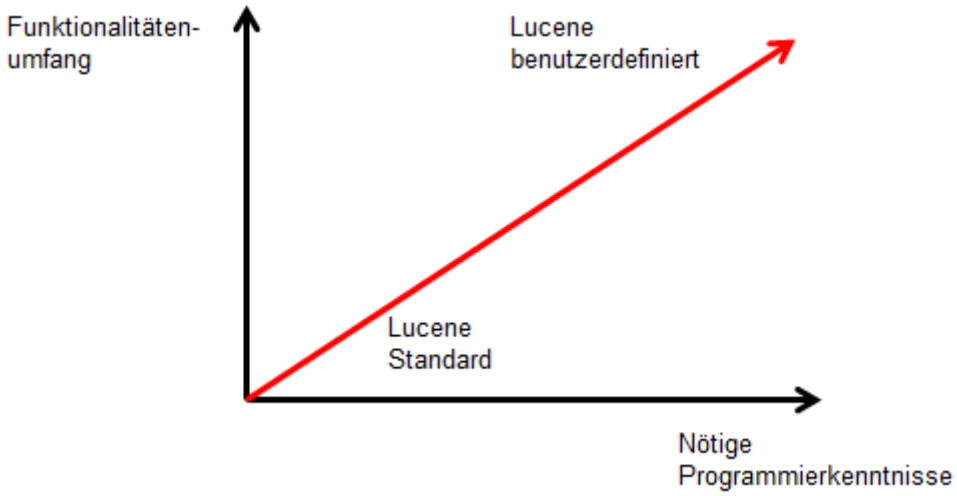


Abbildung 13: Zusammenhang zwischen Funktionalitäten und Programmierkenntnissen[67]

Bei einigen Suchmaschinen wurden mithilfe der Internetrecherche kaum oder keine Nachteile gefunden. Dies bedeutet nicht, dass diese Maschinen keine aufweisen. Die Autoren sind der Auffassung, dass bei einigen Suchmaschinen aus einem geringem Bekanntheits- und Nutzungsgrad eine mangelnde Literatur zu den jeweiligen Nachteilen resultiert. Je weniger Menschen eine Suchmaschine nutzen, desto Weniger beschweren sich darüber oder suchen in Foren nach Hilfe bei Problemen.

Alle vorgestellten Suchmaschinen mit Ausnahme von Ecosia sind in der Anschaffung und in der Administation kostenfrei. Ecosia kann generell nicht erworben werden, da mit Bing gearbeitet wird. Kosten entstehen im Allgemeinen nur im Sinne von Arbeitszeit, die für die Implementierung und Wartung aufgewendet wird. Wenn ein Unternehmen hier Kosten einsparen möchte, wird zu einer Suchmaschine geraten, die wenig Programmieraufwand erfordert, wie z.B. Solr, da es ein fertiges Projekt ist und daher einfach zu implementieren ist.

6.2 Empfehlung und Fazit

Generell ist von den untersuchten Suchmaschinen fast jede geeignet, um für unternehmensinterne Suche verwendet zu werden. Die Ausnahme könnte Yacy bilden, da die Suchmaschine generell als langsam eingestuft wird und vorab individuell getestet werden muss . Die Auswahl, welche Suchmaschine für welches Unternehmen geeignet ist, hängt immer von den Anforderungen, Ressourcen und dem Know-how der jeweiligen Unternehmen ab.

Speziell für die VERS BW ist aus der Analyse ersichtlich, dass bis auf Ecosia und Yacy alle ausgewählten Suchmaschinen geeignet wären. Die engere Wahl der 3 Favoriten wurde auf-

[67] Eigene Darstellung

grund geringfügiger Unterschiede oder Besonderheiten getroffen. Wichtig war vor allem, ob es eine aktive Community gibt, da die Suchmaschine von der VERS BW selbst implementiert werden soll. Auch auf die Sicherheit, ob ein Projekt in Zukunft auch weiterentwickelt und unterstützt wird, wurde großen Wert gelegt. Bei Lucene und Solr ist diese beispielsweise aufgrund der Apachebasis gegeben. Naheliegend ist da natürlich die Frage, wieso dann nicht auch ElasticSearch in die Favoritenliste aufgenommen wird. Die Antwort darauf lautet, weil sich Solr und ElasticSearch so ähnlich sind, dass die Autoren nur eine davon empfehlen wollen, um noch ein anderes Konzept, nämlich Swish-e empfehlen zu können. Solr wurde bevorzugt, da die Verbreitung größer ist. Gründe, warum z.B. Hsearch nicht in die Favoritenliste aufgenommen wurde sind einerseits die geringe Verbreitung und andererseits, dass Hsearch seinen Schwerpunkt auf die Suche in „Big Data“ gesetzt hat und das für die VERS – bis dato kein Kriterium war. Daher stellt die Implementierung einer Suchmaschine, die für extrem große Datenmengen ausgelegt ist, eine unnötige Ressourcenverschwendung dar.

Die VERS muss nun lediglich noch entscheiden, welche der Vor- und Nachteile für sie bedeutender sind und aufgrunddessen eine davon auswählen. Beispielsweise könnte ihre Wahl auf Lucene fallen, weil das Team der VERS die nötigen Programmierkenntnisse aufweist, um die Standardversion auf ihre Bedürfnisse anzupassen und weil Lucene bereits für eine Wikisuche - nämlich Wikipedia - bekannt ist. Ebenso könnte es sein, dass sie sich für ein fertiges Projekt wie Solr oder ElasticSearch entscheiden, da hier die Zusatzfunktionen bereits implementiert sind und sie den hohen Grad an Benutzerspezifikation, den Lucene bietet, überhaupt nicht benötigen.

Da diese Arbeit lediglich acht von einer sehr großen Auswahl Opensource-Suchmaschinen betrachtet, ist nicht auszuschließen, dass es noch Andere gibt, die sich gut oder besser für die Implementierung im Wikisystem der VERS BW eignen. Dennoch stellen die ausgewählten „Top 3“ zufriedenstellende Lösungen dar, da sie alle im Kriterienkatalog gelisteten Anforderungen erfüllen und teilweise auch übertreffen.

Anhang

Fragen an die VERS BW

1. allg. Unternehmensstruktur

a. Gibt es ein Orgchart? (Kriterien, nach denen bestimmt wird, wer auf was Zugriff hat)

→ Zugriff ist abhängig von der Gruppenzugehörigkeit und Attributwerten im Verzeichnisdienst (eDirectory)
→ gesetztes LDAP Attribut bestimmt Zugriffrecht auf ein Wiki-System (bspw. Referat) d.h. Abmeldung
→ Wiki-ACLs setzen die Rechte für Wiki-Seiten um mittels Gruppenzugehörigkeit

- Nicht angemeldete User haben keinerlei Berechtigungen (@ALL)
- Benutzer dürfen lesend auf das Wiki zugreifen (@User). Die Anmeldung wird über das eDirectory-Attribut VERSbwWiki reguliert.
- Über die Gruppenzuordnung im eDirectory wird der schreibende Zugriff auf das Wiki geregelt (@Wiki_IT*_Grp).

2. Detaillierte Erklärung, wie Intranet/Wikis/Suche bisher funktioniert hat

→ Suche in den einzelnen Wiki-Systemen über die interne Wiki-Suche möglich
→ Zentrale Suchmaske für alle Wiki-Systeme nötig (Suche über alle Wikis)
→ Einbindung aller relevanter Daten und Dokumente (pdf…) in die übergeordnete Suche
→ Berücksichtigung der in den Wiki-Systemen gültigen Zugriffsrechte durch die Suchfunktion
(Wie es auch bei der Wiki-internen Suche der Fall ist)

3. Zweck:

a. Wird sie für das Intranet verwendet oder soll eine Website-Suche auf der eigenen Website implementiert werden? Ist das Internet komplett außen vor?

→ Das Internet ist komplett ausgeschlossen
→ Die Suchmaschine wird nur im eigenen Netz verwendet

b. Soll die Open Source Suchmaschine "Out-of-the-Box" lauffähig sein oder möchte man ein eigenes Interface zum Bedienen der Suchmaschine?

→ Egal…

4. Welche Systeme müssen angebunden werden (Datenbanken)?

→ Wiki-Systeme

5. Welche Systeme sind bereits angebunden?

→ Verschiedene Suchlösungen im Einsatz
→ Erste Tests mit Suchlösungen, bei denen die Wiki-Systeme eingebunden sind

6. Ist es richtig, dass das Intranet nur aus Wikis und Websiten besteht?

→ Die relevanten Datenquellen sind Wiki-Systeme

7. Was ist gut/zufriedenstellend am bisherigen System, was ist verbesserungswürdig?

a. konkrete Problemsituation

→ Suche innerhalb der einzelnen Wiki-Systeme für die lokale Suche in Ordnung

Bitte speziell betrachten:

→ gruppenbezogene Berücksichtigung von Zugriffsrechten

→ Suchlösung vom Hersteller bzw. der Community unterstützt → aktives Projekt

b. konkrete Aufgabenstellungen

→ Finden einer Suchlösung auf Open Source Basis, die die Anforderungen erfüllt

8. Warum nur Open Source?

→ Weil es ein KOS Projekt ist

9. Sind bestimmte Open Source Suchmaschinen nicht anwendbar aufgrund der technischen Anforderungen/Einbindung in das System?

→ Bitte unabhängige Evaluierung vornehmen

a. Sollen diese überhaupt detailliert untersucht werden?

10. Weitere Anforderungen an das System (neben Autorisierung und Datenbankanbindung)?

→ Bitte dokuwiki anschauen... keine Datenbankanbindung vorhanden.

11. Wird die Implementierung intern durchgeführt oder von externer IT-Firma?

→ Intern

12. Für wen ist die Dokumentation bestimmt? (Zwecks Art und Weise und Verwenden von Fachwörtern von unserer Seite in der Doku)

→ IT Abteilung

Quellenverzeichnisse

Verzeichnis der Internet-Quellen

o.V (o.J):	Informationen zum Unternehmen XXX, Abruf: 24.01.2013
Gaulke ,M. (o.J):	Funktionsweise von Suchmaschinen, http://www.suchmaschinenkompetenz.de/Funktionsweise-Suchmaschine-Suchmaschinenkompetenz.htm, Abruf:02.01.2013
Schmidt, B. (2006):	Aufbau eines Wiki-Systems also kooperative Informationsplattform der Arbeitsgruppe Softwareergonomie und Information Retrival http://kola.opus.hbz-nrw.de/volltexte/2006/58/pdf/sa_schmidt.pdf , Abruf: 28.12.2012
o.V. (2013):	Suchmaschine, http://de.wikipedia.org/wiki/Suchmaschine#Datenquelle, Abruf: 14.01.2013
o.V.(2013):	Marktanteile von Suchmaschinen in den USA von 2007 bis 2011, http://de.statista.com/statistik/daten/studie/163595/umfrage/marktanteile-der-suchmaschinen-in-den-usa/, Abruf: 10.01.2013
Meister, A./van Nerven, A. (o.J.):	Implikationen des Social Webs auf Suchmaschinentechnologien, http://winfwiki.wi-fom.de/index.php/Implikationen_des_Social_Webs_auf_Suchmaschinentechnologien, Abruf: 07.01.2013
Bußkamp, D. (o.J.):	Open Source Tutorial, http://www.dbus.de/eip/kapitel01e.html, Abruf: 20.12.2012
Gaulke, M. (o.J):	Suchdienste im Internet, http://www.suchmaschinenkompetenz.de/Suchdienste-Suchmaschinenkompetenz.htm, Abruf: 22.12.2012

o.V. (2012): Webcrawler, http://de.wikipedia.org/wiki/Webcrawler, Abruf: 22.12.2012

o.V. (o.J.): Crawler, http://www.drisol.com/informationen/seo-lexikon/crawler/, Abruf: 22.12.2012

Heidenreich, A. (o. J.): Lucene – Eine einfache Suchmaschine für Freitextsuche, http://www.lucene.de/, Abruf: 27.12.2012

o. V. (2012): Lucene, http://de.wikipedia.org/wiki/Lucene, Abruf: 25.12.2012

o. V. (2012): Apache Lucene Core, http://lucene.apache.org/core/, Abruf: 18.12.2012

o. V. (o. J.): Lucene, http://dev.innovationgate.de/doc-wga41/html/default/dr.qr.lucene.de.html, Abruf: 26.12.2012)

Sitaula, S. (2012) Apache Lucene and Apache Solr, http://blog.cloudera.com/blog/2012/01/hadoop-for-archiving-email-part-2/, Abruf: 20.01.2013

o. V. (o. J.): Lucen Dev, http://lucene.apache.org/core/developer.html, Abruf: 19.12.2012

Stäuble, M. (2010): ElasticSearch – Verbindung von Lucene und Cloud, http://it-republik.de/jaxenter/artikel/ElasticSearch---Verbindung-von-Lucene-und-Cloud-3134.html, Abruf: 19.12.2012

Karich, P. (2012): Immer flexibel, http://www.heise.de/ix/artikel/Immer-flexibel-1440270.html, Abruf: 25.12.2012

o. V. (2013): ElasticSearch, http://en.wikipedia.org/wiki/ElasticSearch, Abruf: 25.12.2012

Lieweries, M. (2012): ElasticSearch – Die moderne Art zu Suchen, http://maiks-chaos.blogspot.de/2012/01/elasticsearch-die-moderne-art-zu-suchen.html, Abruf: 26.12.2012

o. V. (2013): Why ElasticSearch ?, http://www.elasticsearch.com/products/elasticsearch/, Abruf: 02.01.2013

o. V. (o. J.):	Elasticsearch, you know, for search, http://www.elasticsearch.org/, Abruf: 02.01.2013
o. V. (2012):	Solr vs. ElasticSearch, http://stackoverflow.com/questions/10213009/solr-vs-elasticsearch, Abruf: 26.12.2012
o. V. (2012):	Apache Solr, http://lucene.apache.org/solr/, Abruf: 27.12.2012)
o. V. (2012):	Overview of Searching in Solr, http://lucidworks.lucidimagination.com/display/solr/Overview+of+Searching+in+ Solr, Abruf: 19.12.2012
Fondermann, B. (2009):	Solr, der Suchserver, http://it-republik.de/jaxenter/artikel/Solr-der-Suchserver-2176.html, Abruf: 28.12.2012
o. V. (2010):	Welcome to Hsearch! , http://bizosyshsearch.sourceforge.net/, Abruf: 28.12.2012
Bizosys Technologies (2010):	HSearch Data Sheet, http://www.hadoopsearch.net/hsearchdatasheet.pdf, Abruf: 29.12.2012
o. V. (2011):	Start-up Profile: HSearch Bizosys Technologies, http://yourstory.in/2011/06/start-up-profile-hsearch-bizosys-technologies/, Abruf: 29.12.2012
o. V. (2013):	FAQ auf der Herstellerseite Ecosia, http://ecosia.org/faq.php#39, Abruf: 09.01.2013
o. V. (2013):	Ecosia, http://de.wikipedia.org/wiki/Ecosia#Kritik, Abruf: 09.01.2013
Paschke, T. (2013):	Blog Eintrag zu Ecosia, http://abenteuerdenken.blogspot.de/2010/06/ecosia-vision-oder-betrug.html, Abruf: 09.01.2013
o. V. (2009):	Blog Eintrag zu Ecosia, http://zeitblick.wordpress.com/2009/12/12/ecosia-org-rettung-oder-schwindel/, Abruf: 09.01.2013
o. V. (2013):	Yacy, homepage, http://yacy.net/de, Abruf: 08.01.2013

o. V. (2013):	Yacy, http://de.wikipedia.org/wiki/YaCy, Abruf: 08.01.2013
o. V. (2013):	Swish-e, http://swish-e.org/, Abruf: 10.01.2013
Strathewerd, R. (2004):	Beitrag zu Swish-e im Linux Magazin, http://www.linux-magazin.de/Ausgaben/2004/04/Finden-im-Sauseschritt Abruf: 10.01.2013
o. V. (2010):	Artikel zu Swish-e, http://www.pcwelt.de/downloads/Swish-E-1217245.html, Abruf: 10.01.2013
Universität Glasgow (2005):	Publikation, Terrier: A High Performance and Scalable Information Retrieval Platform, Department of Computing Science, University of Glasgow Scotland
Universität Glasgow (2011):	Terrier, http://terrier.org/docs/current/terrier_http.html, Abruf: 13.01.2013
Murray, J. (2005):	Innovationsreport, What's the European answer to Google - TERRIER, the search engine with added byte, http://www.innovations-report.de/html/berichte/informationstechnologie/bericht-41606.html, Abruf: 13.01.2013
o. V. (2005):	Terrier 2.1, http://mscerts.programming4.us/de/239656.aspx, Abruf: 13.01.2013
Macke, S. (2006):	Blogeintrag, http://blog.stefan-macke.com/2006/11/30/installation-von-swish-e-unter-suse-101/, Abruf: 21.01.2013

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Einsetzbarkeit von Meta-Suchmaschinen zur Abdeckung bestehender Suchlösungen in verschiedenen Bereichen

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“
Kompetenzzentrum Open Source (KOS)

Vorgelegt von

Wilhelm Evert, Marcel Reifenberger,
Michael Ried, Jan-Philipp Oppermann

am 22.01.2013

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
WWI2010V

Inhaltsverzeichnis

Abkürzungsverzeichnis

BSD	**B**erkeley **S**oftware **D**istribution (Lizenztyp)
CRM	**C**ustomer **R**elationship **M**anagement
DB	**D**aten**b**ank
DMS	**D**okument-**M**anagement-**S**ystem
ERP	**E**nterprise-**R**esource-**P**laning
GB	**G**iga**b**yte
GIF	**G**raphics **I**nterchange **F**ormat
GPL	**G**eneral **P**ublic **L**icense
HTML	**H**yper**t**ext **M**arkup **L**anguage
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
IRC	**I**nternet **R**elay **C**hat
IT	**I**nformations**t**echnologie
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
JSON	**J**ava **S**cript **O**bject **N**otation
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol
LGPL	**L**ibrary **G**eneral **P**ublic **L**icense
MB	**M**ega**b**yte
MPL	**M**ozilla **P**ublic **L**icense
NFS	**N**etwork **F**ile **S**ystem
NoSQL	**N**ot **o**nly **SQL**
RAM	**R**andom-**A**ccess-**M**emory
REST	**Re**presentational **S**tate **T**ransfer
RSS	**R**ich **S**ite **S**ummary
SQL	**S**tructured **Q**uery **L**anguage
WAIS	**W**ide **A**rea **I**nformation **S**erver System
XML	E**x**tensible **M**arkup **L**anguage

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Zu einem entscheidenden Kriterium für die Leistungsfähigkeit eines Unternehmens gehört das schnelle, orts- und zeitunabhängige Auffinden von benötigten Informationen. Zu diesem Zweck werden Wissensdatenbanken verwendet, welche ebendiese Informationen speichern. Eine wesentliche Herausforderung besteht darin, die richtigen Informationen ohne großen Aufwand zu finden. Dies lässt sich wiederum über Suchmaschinen bewerkstelligen.
Bei der Implementierung einer Suchmaschine ist zunächst die Entscheidung zu treffen, ob ein kommerzielles System oder ob eine Open Source Suchmaschine verwendet werden soll. Neben dem Kostenaspekt gehört auch die Art der Weiterentwicklung zu den Vorteilen von Open Source Suchmaschinen. So gibt es bei Open Source Projekten meist eine größere Gemeinde, welche den Entstehungs- und Entwicklungsprozess einer Anwendung begleitet und Rückmeldungen und Verbesserungsvorschläge von Nutzern in kommenden Versionen umsetzt. Zudem entfallen die lizenz- und vertragsbedingten Restriktionen, die oft mit kommerziellen Systemen verbunden sind.[1]
Diese Arbeit verfolgt das Ziel, eine Open Source Lösung für die Verwendung von Suchmaschinen im Unternehmensbereich zu finden. Hierzu trägt die Arbeit mit folgenden Punkten bei:

- Auf Basis von definierten Anforderungen sowie einem Bewertungskonzept werden vier bekannte Vertreter von Open Source Suchmaschinen hinsichtlich ihrer Funktionen und Einsatzmöglichkeiten betrachtet und bewertet. Zusätzlich werden die zugrunde liegenden Open Source Projekte evaluiert.

- Daran anschließend erfolgt eine bewertende Gegenüberstellung aller Suchmaschinen, mit der letztendlich die Auswahl der besten Suchmaschine ermöglicht wird.

Es ist nicht Ziel der Arbeit, kommerzielle Suchmaschinenanbieter zu betrachten oder genaue Berechnungen über Betriebs- oder Managementkosten durchzuführen.

[1] Vgl. Arnold, S. E. (2011), S.26

2 Konzept zur Bewertung von Open Source Suchmaschinen

Die Bewertung der Open Source Suchmaschinen erfolgt auf zwei Ebenen. Zunächst wird das Open Source Projekt bewertet, unter dem die jeweilige Suchmaschine entwickelt wird. Oft lassen sich bereits anhand des Projekts Rückschlüsse auf Qualität und Entwicklungsstand des Produkts ziehen. In einem zweiten Schritt erfolgt die eigentliche Bewertung der Meta-Suchmaschinen auf Basis von zuvor definierten Anforderungen.

2.1 Methoden zur Beurteilung von Open Source Projekten

Eine wichtige Grundlage für die Bewertung bildet das zugrundeliegende Open Source Projekt. Die Qualität und das Potential einer Open Source Software hängen stark vom Erfolg des Projekts ab, welches die Anwendung entwickelt. In diesem Zusammenhang ist der Reifegrad des Open Source Projekts ein entscheidendes Bewertungskriterium. Im Folgenden werden ausgewählte Faktoren beschrieben, nach denen der Reifegrad bestimmt werden kann. Die Faktoren werden dabei drei Kategorien zugeordnet: Produkt, Dokumentation und Support.[2]

1. Produkt
    - Produktalter

    In diesem Punkt wird das Alter des Projekts (in Jahren) bewertet. Ältere Projekte gelten in der Regel als fortgeschrittener und reifer und sind somit höher zu bewerten.

    - Lizenzen

    Open Source Lizenzen können in unterschiedlichen Ausprägungen auftreten, sowohl mit kostenpflichtigen Inhalten als auch komplett kostenfrei. Letzteres ist bei den hier ausgewählten Suchmaschinen zu bevorzugen. Zusätzlich ist bei Lizenzen für Firmenkunden zu berücksichtigen, welche Restriktionen bezüglich der Verwendung des Codes vorliegen. Es sind diejenigen Lizenzen vorzuziehen, bei denen der Code möglichst frei verwendet werden kann. Grundsätzlich kann dabei nachfolgende Unterscheidung getroffen werden (Tab. 1):

[2] Vgl. Golembowska, A. u. a. (2012), S.18

Kombinations-möglichkeit mit proprietärer Software	Keine Einbindung in proprietären Code möglich.	Statisches und dynamisches Linken von Code mit proprietärer Software möglich. Eigenentwicklungen dürfen als proprietäre Software weitergegeben werden.	Keine Vorgaben; der gesamte Code darf auch als proprietäre Software weitergegeben werden
Beispiel-Lizenz	GPL	LGPL, MPL	BSD, Apache

Tab. 1: Arten von Open Source Lizenzen[3]

- Plattform und Kompatibilität

Hier ist zu beurteilen, welche bzw. wie viele unterschiedliche Plattformen (hardware- und betriebssystemseitig) die Anwendung unterstützt. In diesem Zusammenhang spielt insbesondere die Kompatibilität mit verschiedenen Betriebssystemen (z.B. Windows und Unix-Betriebssysteme) eine wichtige Rolle.

- Funktionalität

Dieses Kriterium umfasst den Funktionsumfang der Meta-Suchmaschine. Ein umfassendes Funktionsspektrum spricht für eine gute Unterstützung durch die Entwickler und für die Qualität des Open Source Projekts.

- Modularität

Unter diesem Punkt ist die Möglichkeit der Erweiterbarkeit eines Produkts zu verstehen. Dabei ist vor allem zu beurteilen, ob Add-Ons vorhanden sind. Diese Möglichkeit wird auch als Baukastenprinzip bezeichnet.

- Releaseabstände

Dabei steht die Zeit im Mittelpunkt, seit dem die letzte Softwareversion veröffentlicht wurde. Des Weiteren ist die Anzahl vorhandener Updates, Hotfixes, Patches und Service Packs, sowie deren Aktualität und Häufigkeit zu beachten. Grundsätzlich spricht eine große Anzahl an regelmäßigen Updates für eine aktive Community. Für

[3] Mit Änderungen entnommen aus: Kleijn, A. (2006)

Unternehmen ist es meist sinnvoll, dass die wichtigsten Patches und Hotfixes in angemessenen, nicht zu häufigen Zeitabständen veröffentlicht werden, kritische Probleme jedoch sofort beseitigt werden.[4]

2. Dokumentation

   - Benutzerhandbuch

   Das Vorhandensein eines Benutzerhandbuchs ist ein elementares Kriterium, da es wichtige Basisinformationen zur Software beinhaltet. Dabei ist es wichtig, dass Installation, Konfiguration und die Bedienung der Anwendung beschrieben wird.

   - Sprache der Dokumentation

   Die meisten Open Source Projekte sind englischsprachig und oft sind auch die Referenzen und Anleitungen ausschließlich in englischer Sprache verfügbar. Wünschenswert wäre, insbesondere für einen deutschen Kunden, die Verfügbarkeit von deutschsprachiger Dokumentation.

   - Qualität der Dokumentation

   Die bloße Existenz einer Dokumentation ist wenig hilfreich, wenn diese nicht verständlich aufbereitet ist. Deshalb muss die inhaltliche Qualität, die Strukturierung und die Vollständigkeit der Dokumentation bewertet werden. Ein Anwender muss in der Lage sein, anhand der beschriebenen Vorgehensweisen ohne großen zeitlichen Aufwand die richtige Lösung für seine Problemstellung zu finden. Idealerweise sollten keine zusätzlichen Quellen herangezogen werden müssen.[5]

3. Support

   - Community und Ökosystem

   Aus der Aktivität der Community eines Software Projekts lässt sich oft die Nachhaltigkeit und Reife des Projekts bzw. Produkts ableiten. Zu den Indikatoren zählen hier Foren und Newsgruppen, und speziell Beiträge zu Problemlösungen und Beiträge von Entwicklern. Als quantitatives Merkmal kann die Zahl der Forumseinträge bzw. Themen herangezogen werden. Das Ökosystem eines Open Source Projekts setzt

[4] Vgl. Golembowska, A. u. a. (2012), S.18 ff.

[5] Vgl. ebenda, S.21 f.

sich aus einer Reihe an Stakeholdern zusammen. Darunter fallen Anwender (private und geschäftliche Nutzer), Anbieter (Distributoren, Dienstleister), Meinungsbildner (Analysten, Presse, Experten), unterstützende Stiftungen (Non-Profit-Organisationen) sowie die eben erwähnte Community. Weitere Bestandteile des Ökosystems bilden die vorhandenen Projekte und die angebotenen Produkte bzw. Lösungen.[6] Für das Bewertungskonzept wäre zu bewerten, inwieweit die hier genannten Stakeholder und Bestandteile im jeweiligen Open Source Projekt ausgeprägt sind.

- direkter Ansprechpartner

Ein hilfreicher Aspekt für Neueinsteiger ist das Vorhandensein von Kontaktdaten des Entwickler(team)s. Dabei ist auch zu beurteilen, ob bei einer Anfrage ein Kontakt zustande gekommen ist und wie die Qualität einer möglichen Antwort zu werten ist.

- Vorhandensein von FAQ

Eine sehr schnelle Problemlösung bieten „Frequently Asked Questions“ (FAQs). Ein FAQ ist häufig auf der Projekt-Homepage als erste Anlaufstelle bei Fragen zu finden.[7]

Die Bewertung der Suchmaschinenprojekte erfolgt nach dem folgenden Schema:

- erfüllt ein Projekt ein Kriterium gar nicht, erhält es 0 Punkte
- erfüllt ein Projekt ein Kriterium, erhält es einen Punkt
- erfüllt ein Projekt ein Kriterium im Vergleich zu den anderen bewerteten Suchmaschinenprojekten in höherem Maße (quantitativ oder qualitativ), bekommt es, abhängig vom relativen Erfüllungsgrad, 2, 3 oder 4 Punkte

Als Basis dient dabei nachfolgende Beurteilungsmatrix (Tab 2). Diese beinhaltet alle einzelnen Bewertungskriterien sowie eine Spalte, welche die Gewichtung für die Bewertungspunkte angibt. Die Gewichtungen sind der Quelle „Entwicklung eines Modells zur Bewertung von Open Source Produkten hinsichtlich eines produktiven Einsatzes“ [Golembowska, A. u. a. (2012)] entnommen. In den Spalten der Suchmaschinen steht die erreichte Punktzahl (0 bis 4). Die erreichte Punktzahl wird mit der jeweiligen Gewichtung multipliziert und die dadurch ermittelten Werte werden in den Summenzeilen der zugehörigen Kategorie aufsummiert.

[6] Vgl. Zacher, M. (2007)

[7] Vgl. Golembowska, A. u. a. (2012), S.22 f.

Schließlich werden die Werte aus den Summenzeilen der Kategorien mit ihrer Gewichtung multipliziert und in der untersten Summenzeile summiert.

Kriterium	Gewichtung	Suchmaschinen			
		Lucene	Solr	Elastic-search	HSearch
Produktalter	20%				
Lizenzen	15%				
Plattform u. Kompatibilität	15%				
Funktionalität	20%				
Modularität	15%				
Releaseabstände	15%				
SUMME Produkt	50%				
Benutzerhandbuch vorhanden	75%				
Sprache der Dokumentation	5%				
Qualität der Dokumentation	20%				
SUMME Dokumentation	30%				
Community und Ökosystem	50%				
direkter Ansprechpartner	20%				
Vorhandensein von FAQ	30%				
SUMME Support	20%				
SUMME	100%				

Tab. 2: Matrix zur Bestimmung des Reifegrads von Open Source Projekten

Der hierdurch identifizierte Reifegrad soll als Bewertungskriterium nicht isoliert betrachtet werden, sondern im Zusammenhang mit einem zweiten Aspekt (Marktakzeptanz) analysiert werden. Zur Darstellung des Zusammenhangs zwischen zwei Faktoren eignet sich die Vier-Felder-Matrix der Boston Consulting Group.

Bei dieser Matrix handelt es sich um ein Modell der Produkt-Portfolio-Analyse, welches den Zusammenhang zwischen zwei Faktoren herstellt. Als externer Faktor wird das Marktwachstum (senkrechte Achse) und als interner Faktor der relative Marktanteil (waagerechte Achse) festgelegt. Das Marktwachstum repräsentiert die derzeitige Lage eines Produkts im Produktlebenszyklus. Je niedriger dieses ist desto fortgeschrittener ist der Lebenszyklus. Je höher

der Marktanteil eines Produkts ist umso niedriger sind dessen Stückkosten.[8] In Abb. 1 wird dieser Zusammenhang dargestellt.

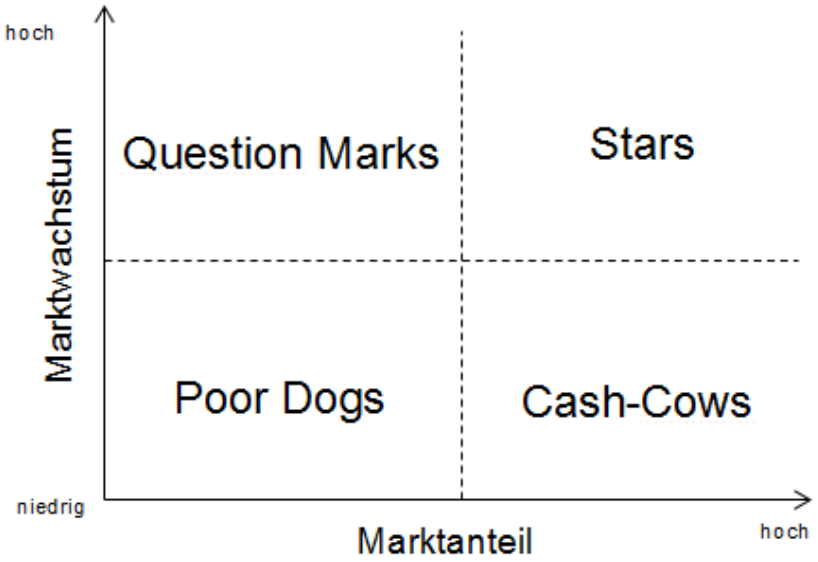


Abb. 1: Vier-Felder-Matrix der Boston Consulting Group[9]

Innerhalb der Matrix lassen sich vier Felder erkennen, zu denen die einzelnen Produkte eines Portfolios zugeordnet werden.

- Question Marks

Produkte in diesem Bereich stehen am Anfang eines Produktlebenszyklus und weisen einen niedrigen Marktanteil auf. Die weitere Entwicklung von diesen Produkten ist zu diesem Zeitpunkt noch ungewiss und ein Unternehmen muss sich entscheiden, ob eine Investition in das Produkt lohnenswert ist.

- Stars

In diesem Feld stehen Geschäftsfelder mit hohem Marktwachstum und hohem Marktanteil. Es handelt sich somit um eine Idealsituation, in der der Marktanteil durch Reinvestition der erzielten Überschüsse gesichert werden muss.

[8] Vgl. Wöhe, G. / Döring, U. (2010), S. 89

[9] Mit Änderungen entnommen aus: ebenda, S. 90

- Cash-Cows

Das Wachstum geht in diesem Bereich zurück, der Marktanteil bleibt aber hoch. Das Produkt befindet sich im Reifeprozess. Es sollte versucht werden, soviel Gewinn wie möglich mit dem Produkt abzuschöpfen, bevor der Marktanteil zu gering wird.

- Poor Dogs

Bei diesen Produkten sind sowohl Marktanteil als auch Marktwachstum gering. Folglich befinden sich diese Geschäftsfelder am Ende des Produktlebenszyklus und erwirtschaften nur noch geringe Deckungsbeiträge. Aufgrund dieser Konstellation ist eine Eliminierung meist sinnvoll.[10]

Bezogen auf die Bewertung von Open Source Projekten, soll als externer Faktor der zuvor definierte Reifegrad und als interner Faktor die Marktakzeptanz herangezogen werden. Ausschlaggebend für die Marktakzeptanz sind dabei die folgenden Faktoren:

- Anzahl der Stakeholder

Zu den Stakeholdern gehören Personen und Unternehmen, die das Projekt unterstützen. Als Unterstützer zählen Einzelpersonen bzw. Unternehmen, die das jeweilige System, das unter dem Projekt entwickelt wird, verwenden oder als Sponsoren für das Projekt fungieren.

- Anzahl der Downloads

Als Indikator für die Marktakzeptanz dient vor allem die Anzahl der Downloads für das jeweilige Produkt. Damit lässt sich eine Aussage über die Verbreitung der Software treffen. Informationen über Downloadzahlen können entweder direkt über die Entwicklerseite oder über externe Downloadseiten (z.B. chip.de) festgestellt werden.

- globale oder deutschlandweite Popularität (Traffic Rank)

Neben der Downloadanzahl ist die Besucherzahl der Entwickler-Homepage ein zu berücksichtigender Faktor bei der Beurteilung der Marktakzeptanz. Die Messung von Besucherzahlen ist über die Seite alexa.com möglich. Dabei ist auch eine Unterscheidung nach deutschen und weltweiten Besuchern möglich.[11]

[10] Vgl. Wöhe, G. / Döring, U. (2010), S. 89 f.

[11] Vgl. Golembowska, A. u. a. (2012), S.23 f.

Die drei Kriterien für die Bestimmung der Marktakzeptanz sind in Tab. 3 mit ihrer jeweiligen Gewichtung aufgelistet. Diese Tabelle dient der späteren Bewertung der Projekte, wobei das Bewertungsschema identisch zu dem der Reifegradbestimmung ist.

Kriterium	Gewichtung	Suchmaschinen			
		Lucene	Solr	Elastic-search	HSearch
Anzahl der Stakeholder	50%				
Anzahl der Downloads	25%				
Traffic Rank	25%				
SUMME	100%				

Tab. 3: Matrix zur Bewertung der Marktakzeptanz von Open Source Projekten

Stellt man den Zusammenhang zwischen Reifegrad und Marktakzeptanz grafisch dar, sieht die Vier-Felder-Matrix wie folgt aus (Abb. 2).

Abb. 2: Vier-Felder-Matrix für die Bewertung von Open Source Projekten

Als „Question Marks“ können hier diejenigen Projekte aufgezählt werden, die zwar als reif gelten, jedoch noch keine große Verbreitung genießen. Aufgrund der geringen Akzeptanz ist die Zukunft dieser Projekte ungewiss. Besonders erfolgreiche und fortgeschrittene Projekte fallen in den Bereich „Stars“. Das Gegenstück dazu bilden Projekte, die noch unausgereift sind und kaum Anwender gefunden haben. Diese gelten als „Poor Dogs“ und sind für den Einsatz im Unternehmen zu meiden. Das Feld der „Cash-Cows“ belegen solche Projekte, die trotz ihres geringen Reifegrads breite Zustimmung erfahren und vielfach von Unternehmen

eingesetzt werden. Diese Kategorie ist dann vorstellbar, wenn ein Projekt Produkte entwickelt, für die es bisher keine vergleichbaren Alternativprodukte anderer Anbieter gibt und diese deshalb schnell an Popularität gewinnen. Ein treffenderer Name für dieses Feld wäre daher „Shooting Stars“.
Dieses Modell der Bewertung hat vor allem folgende Vorteile:

- Möglichkeit eines einfachen Vergleichs mehrerer Alternativen durch Darstellung des relativen Reifegrads bzw. der relativen Marktakzeptanz
- gute Anschaulichkeit und Möglichkeit eines schnellen Überblicks durch geringe Komplexität

Dem gegenüber sind diese Nachteile zu erwähnen:

- Beschränkung auf nur zwei betrachtete Faktoren (Reifegrad und Marktakzeptanz)
- die Matrix zeigt nur eine Momentaufnahme, stellt keine Prognose dar

2.2 Kriterienkatalog zur Bewertung von Meta-Suchmaschinen

Die zweite Ebene der Bewertung ist die Evaluierung der Meta-Suchmaschinen und speziell derer Funktionen. Hierzu wird ein Kriterienkatalog mit Anforderungen an Enterprise Suchmaschinen erstellt. Die hier aufgezählten Bewertungskriterien beziehen sich ausschließlich auf die Suchmaschinen und deren Funktionalität. Projekt- und lizenzspezifische Faktoren und Anforderungen werden im Bewertungskonzept aus Kapitel 2.1 abgedeckt. Im Folgenden werden alle Kriterien erläutert.

- Codierung und Sprachunterstützung

Das Suchsystem sollte idealerweise in deutscher Sprache installiert, konfiguriert und bedient werden können. Bezüglich der Codierung muss die Suchmaschine zwingend deutsche Sonderzeichen unterstützen (z.B. UTF-8).

- Crawler

Bei einem Crawler handelt es sich um einen Programmbaustein, mit dem eine Suchmaschine auf zu durchsuchende Dokumente zugreifen kann.[12] Die meisten Suchsysteme bieten diese Crawler „out-of-the-box“. In einigen Fällen werden jedoch nur Programmbibliotheken bereitgestellt, welche vom Anwender selbst durch die gewünschten Komponenten erweitert

[12] Vgl. Ziesche, P. / Arinir, D. (2010), S.112

werden müssen. Folglich besitzen diese Systeme nicht zwingend einen Crawler. Das Vorhandensein von Suchmaschinen-Crawlern wird aufgrund des geringeren Implementierungs- und Verwaltungsaufwands bevorzugt. Ferner ist hier zu beurteilen, inwieweit der Crawler (über Module / Add-Ons) angepasst und modifiziert werden, um zum Beispiel zusätzliche Datenquellen durchsuchen zu können.

- Dateiunterstützung

Es können die gängigen Datei-Formate, wie .doc, .xls, .pdf, .ini, .cfg, .xml, .vbs /.vba, durchsucht werden. Da Dokumente in diesem Format im Unternehmensumfeld am häufigsten vorkommen, muss eine ideale Suchlösung ebendiese Formate beherrschen.

- Cloud Unterstützung

Die Suche kann nicht nur lokal am PC erfolgen, sondern auch an entfernten Systemen. In diesem Zusammenhang wäre es von Vorteil, wenn beispielsweise soziale Netzwerke, wie Facebook, Xing, Twitter und LinkedIn, durchsucht werden können. Hintergrund für diese Anforderung ist die stetige Zunahme an Informationen in Web 2.0 Anwendungen.

- Volltextsuche

Ziel des Suchverfahrens ist es, nicht nur ausschließlich Titel zu durchsuchen, sondern den gesamten Inhalt im jeweiligen Dokument. Damit sind Suchmaschinen zu bevorzugen, die eine Volltextsuche unterstützen. Eine Volltextsuche erlaubt es, durch Speicherung des aufbereiteten Textes und durch entsprechende Algorithmen, jedes Dokument zu finden, das mindestens ein Wort der Suchanfrage enthält.[13] Hierbei handelt es sich um ein Pflichtkriterium.

- Synonyme und Korrektur

Die Suche bezieht sich nicht nur auf die eingegebenen Parameter, sondern auch auf alle in einem logischen Zusammenhang stehenden Variationen. Des Weiteren soll es eine Korrektur in Form einer Begriffsklärung („Meinten Sie…“) geben. Folglich ist eine gewisse Unschärfe bei der Suche wünschenswert. Diese wäre sowohl aus syntaktischer als auch aus semantischer Sicht hilfreich, wobei letzteres aufgrund dessen Komplexität voraussichtlich nur mit einem höheren Aufwand realisierbar ist. Beide Funktionen tragen entscheidend zu besseren Suchergebnissen und zur Ergonomie bei der Benutzung des Systems bei.

[13] Vgl. Konrad, R. (2009), S.4

- Cached Results und Hintergrundinformationen

Jedes Ergebnis wird mit einem Link versehen, welcher direkt zur "Fundstelle" führt. Bei jeder Suche wird außerdem angezeigt, wie lange die Suche dauert, wie oft das Suchwort gesucht und gefunden wurde, oder ob sich das Suchwort im Titel oder Text befand. Nützlich sind auch Vorschaufenster für gefundene Ergebnisse. All diese Funktionen tragen ebenfalls zur Ergonomie bei.

- Ausgabe der Suchergebnisse

Die gefundenen Suchergebnisse müssen dem Anwender in geeigneter Form ausgegeben werden. Die hierbei bevorzugte Methode ist die Ausgabe über ein Webinterface. Das wohl bekannteste Webinterface für eine Suche bietet Google (Abb. 3). Dort werden die Suchergebnisse in Form von Links dargestellt. Zusätzlich findet sich eine Vorschau in Textform sowie als graphische Darstellung der gefundenen Seite. Auch lassen sich konkrete Kategorien (und damit Dateiformate) durchsuchen.

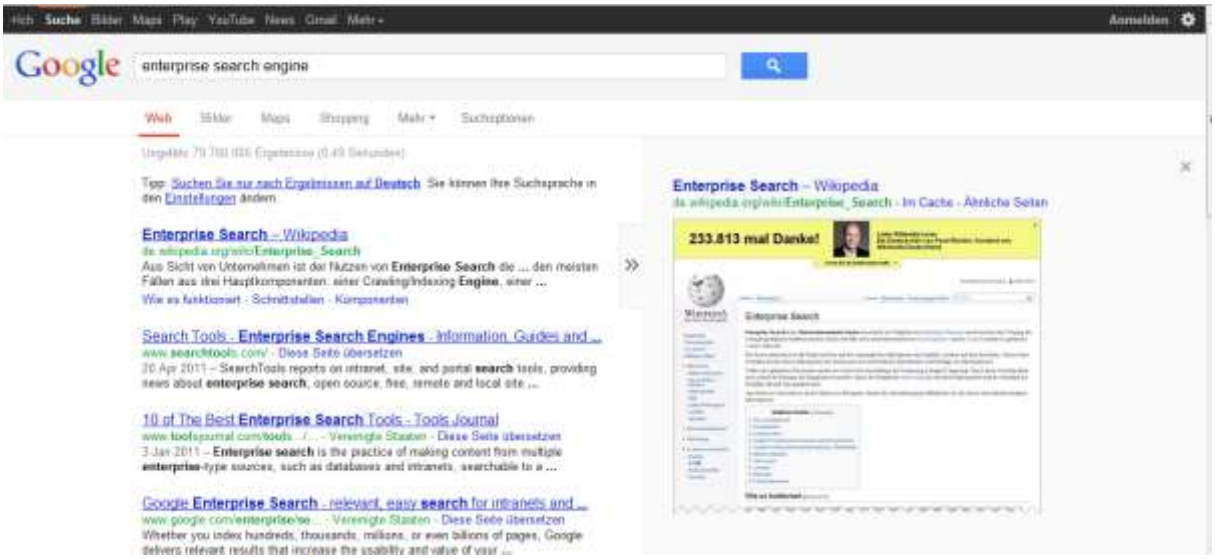


Abb. 3: Webinterface von Google

- Skalierbarkeit

Die physikalische Hardware kann über mehrere Standorte verteilt sein. Daher muss eine Anbindung mehrerer Hardware-Ressourcen unterschiedlichster Art gewährleistet sein. Beispiele für Hardware-Ressourcen sind in diesem Fall Server, Clients, Festplatten und (Netz-) Laufwerke. Damit kann gewährleistet werden, dass eine hohe Anzahl an Dokumenten über viele verschiedene verteilte Geräte erreicht wird. Dies wiederum trägt zu einem zentralen sowie orts- und zeitunabhängigen Wissensmanagement bei.

- LDAP-Anbindung / Umsetzung bestehender Zugriffsrechte

Die Übernahme des implementierten Berechtigungskonzepts stellt eines der zu berücksichtigenden Hauptkriterien dar. In diesem Zusammenhang ist die Anbindung der Suchmaschine an LDAP-Dienste (z.B. eDirectory) ein wichtiger Punkt. Ferner muss ein eventuell vorhandenes Berechtigungskonzept im Bereich der Wiki-Systeme ohne Probleme für die Suchmaschine adaptiert werden können. Gleichzeitig muss es möglich sein, dass mehrere Benutzer über das gleiche Suchinterface mit unterschiedlichen Zugriffsrechten auf die verschiedenen Datenquellen Zugang haben.

- Anbindung an Datenquellen

Der wichtigste Aspekt ist die Möglichkeit, die im Unternehmen vorhandenen Datenquellen anzubinden. Hierzu gehören beispielsweise Wiki-Systeme, Notes Datenbanken und webbasierte Dokumente im Intranet. Wünschenswert wären außerdem die Anbindung und die Möglichkeit des Durchsuchens von vorhandenen Netzlaufwerken.

- Wartungs- und Betreuungsmöglichkeiten

Die für die Verwaltung der Anwendung zuständigen Bereiche haben die Möglichkeit, notwendige Konfigurationsmaßnahmen vorzunehmen. Hierzu sollten zum Beispiel Skripte vorhanden sein, mit denen der Aufwand manueller Arbeiten minimiert wird. Als weitere Hilfestellung dienen Monitoring- und Reporting-Werkzeuge zur Anzeige von Suchstatistiken, Laufzeiten und Ähnlichem.

- Berechtigungsverwaltung

Unabhängig von der Möglichkeit der LDAP-Anbindung und der Übernahme bestehender Zugriffsrechte kann eine lokale Verwaltung von Berechtigungen von Vorteil sein. Hier sind Szenarien denkbar, bei denen zum Beispiel zu Testzwecken verschiedene Berechtigungen an einzelne Benutzer vergeben werden. Dies kann dadurch realisiert sein, dass für einzelne Benutzer (z.B. aus Active Directory bei gegebener LDAP-Anbindung) oder Benutzergruppen unterschiedliche Zugriffsrechte für bestimmte Datenquellen festgelegt werden können.

- Programmiersprache

Das Projekt ist in einer der gängigen Programmiersprachen geschrieben (C++, Java, etc.). Dies ermöglicht bei Bedarf eine einfachere Anpassung des Quellcodes (falls Funktionalitäten hinzugefügt oder modifiziert werden sollen).

- Performance

Die sehr gute Performance (Antwortzeit) ist ein entscheidender Aspekt bei der Auswahl eines Suchsystems. Diese Laufzeit muss auch in Zeiten hoher Last (z.B. wenn besonders viele Benutzer gleichzeitig Suchanfragen durchführen) gute Werte aufweisen. Zudem soll die Indizierung neuer oder bereits vorhandener Datenmengen nur einen minimalen Zeitaufwand verursachen. Gemessen wird die Performance beispielsweise anhand der Anzahl an Indizierungen pro Sekunde. Daraus lässt sich dann die Antwortzeit ableiten. Als Richtwert für diese Arbeit werden 300 Millisekunden als Indikator für eine gute Antwortzeit definiert. Gleichzeitig sollten 1000 Millisekunden nicht überschritten werden. Die Antwortzeiten beziehen sich dabei auf Zeiten, bei denen eine größere Anzahl an Anwendern (bis 1.000) gleichzeitig Suchanfragen durchführt.

- intuitive Bedienung

Weiterhin ist die intuitive Bedienung mit einer schnellen und unkomplizierten Einarbeitung als wichtige Anforderung aufzuführen. Hilfreich wäre in diesem Zusammenhang eine Unterstützung bei der Einarbeitung durch die Anwendung sowie darüber hinaus verfügbare Online-Hilfe. Beispiele hierfür sind Tutorials und Quick Start Guides. Die Einfachheit des Systems zusammen mit der allzeit verfügbaren Hilfe sorgen dafür, dass Schulungen für die Suchmaschine überflüssig sind.

- einfache Installation und Konfiguration

Wie auch schon bei der Bedienung, ist auch bei der Installation und Konfiguration des Suchservers zu berücksichtigen, dass beide Prozesse unkompliziert und ohne den Bedarf einer Schulung ablaufen. Hinsichtlich der Installation sollten die Benutzereingaben minimal sein. Dies kann beispielsweise dadurch erreicht werden, indem optimale Einstellungen standardmäßig vorbelegt werden und der Anwender den Installationsprozess mit wenigen Klicks durchlaufen kann. Hier sollte die Dauer der Installation 20 Minuten nicht übersteigen.

- Kompatibilität

Hierzu gehört die Kompatibilität der verwendeten Suchmaschine mit dem im Unternehmen verwendeten Betriebssystem (insbesondere Windows und Unix-Systeme) und der vorhandenen Hardware.

Jedes der aufgeführten Kriterien wird entsprechend den gestellten Anforderungen nach Relevanz gewichtet. Die Bewertung der Suchmaschinen erfolgt nach dem gleichen Schema wie auch schon in Kapitel 2.1:

- erfüllt eine Suchmaschine ein Kriterium gar nicht, erhält es 0 Punkte
- erfüllt eine Suchmaschine ein Kriterium, erhält es einen Punkt
- erfüllt eine Suchmaschine ein Kriterium im Vergleich zu den anderen bewerteten Suchmaschinen in höherem Maße (quantitativ oder qualitativ), bekommt es, abhängig vom relativen Erfüllungsgrad, 2, 3 oder 4 Punkte
- handelt es sich um ein Pflichtkriterium, wird das Feld mit JA belegt, falls die Anforderung erfüllt wird, und mit NEIN, falls sie nicht erfüllt wird

Die Beurteilung der Suchmaschinenqualität wird damit anhand nachfolgender Matrix durchgeführt (Tab. 4). Alle Kriterien werden in sieben Kategorien zusammengefasst: grundsätzliche Funktionen, Suchalgorithmus, Ergebnisdarstellung, Skalierbarkeit, Möglichkeiten der Anbindung, Administration und nichtfunktionale Anforderungen.

Kriterium	Gewichtung	Bewertung			
		Lucene	Solr	Elastic-search	HSearch
grundsätzliche Funktionen					
Codierung u. Sprachunterstützung	PFLICHT				
Crawler	8%				
Dateiunterstützung	8%				
Cloud Unterstützung	1%				
Suchalgorithmus					
Volltextsuche	PFLICHT				
Synonyme u. Korrektur	6%				
Ergebnisdarstellung					
Cached Results u. Hintergrundinformationen	3%				
Ausgabe der Suchergebnisse	9%				
Skalierbarkeit					
Skalierbarkeit	10%				
Möglichkeiten der Anbindung					
Umsetzung bestehender Zugriffsrechte	PFLICHT				
Anbindung an Datenquellen	20%				
Administration					
Wartungs- und Betreuungsmöglichkeiten	5%				
Berechtigungsverwaltung	2%				
Programmiersprache	1%				
nichtfunktionale Anforderungen					
Performance	6%				
intuitive Bedienung	7%				
einfache Installation / Konfiguration	8%				
Kompatibilität	6%				
SUMME	100%	0	0	0	0

Tab. 4: Matrix zur Bewertung von Meta-Suchmaschinen

3 Analyse von Open Source Suchmaschinen

Die Informationsvielfalt im Unternehmen unterliegt einem stetigen Wachstum. Es werden täglich neue Daten ergänzt. Ohne den genaueren Inhalt dieser Dateien zu definieren, findet man hauptsächlich Bilder, Videos oder Textdateien. Meist sind diese mit abstrusen Synonymen betitelt und machen es dem „Sucher“ schwer, das Richtige zu ermitteln. Akribisch historisierte Daten sollen jedoch nicht zur Verzweiflung führen, vielmehr sollten Sie den Nutzer dabei unterstützen, sich gezielter über einen Sachverhalt zu informieren. Im Zuge des Knowledge Managements (Wissensdatenbanken) können Neueinsteiger, gleichermaßen aber auch langjährige Mitarbeiter, ihr Wissen teilen und sich gegenseitig bereichern. Oftmals werden solche Wissensdatenbanken – um die Übersicht zu wahren – abteilungsbezogen angelegt. Daraus resultiert ein sehr nützliches, jedoch gleichermaßen unübersichtliches Datenwirrwarr. Würde man alle Datenquellen nun auf einen Quelldatenträger zusammenführen, würde dies einen erheblichen Arbeitsaufwand zu Folge haben. Dieser Arbeitsaufwand kann vermieden werden, indem man sich einer Suchmaschine bedient, welche alle separierten Datenquellen durchläuft, und dann ein gewünschtes Endergebnis liefert.
Der Einsatz von Meta-Suchmaschinen im Unternehmensbereich bringt insbesondere die nachfolgenden Vorzüge mit sich:

- Mitarbeiterwissen effizienter nutzen

Wissen ist ein wichtiges Gut in einem Unternehmen und es wird täglich neues Wissen generiert. Der Wert dieses Wissen steigt, wenn es geteilt wird. Dies kann mit Hilfe von Wikis bewerkstelligt werden, welche wiederum eine Suchmaschine benötigen.

- neue Mitarbeiter schneller einarbeiten

Der Eintritt neuer Mitarbeiter ist oft mit einer langen Einarbeitungszeit verbunden. In diesem Zusammenhang wäre eine zentrale Wissensdatenbank von großem Nutzen, da nicht zuerst die Ressourcen gesucht werden müssen, die entsprechendes Wissen besitzen.

- Unterstützung von Mitarbeitern im Außendienst

In einigen Branchen ist ein großer Teil der Mitarbeiter im Außendienst oder hat einen mobilen Arbeitsplatz. Bei diesen ist die schnelle und ortsunabhängige Informationsbeschaffung von großer Bedeutung. Auch in diesem Fall bieten Suchmaschinen Aushilfe.

- schnellere Akquisition von Unternehmen

Die Übernahme eines Unternehmens ist mit einer Übernahme von dessen Daten und Informationen verbunden. In diesem Fall erleichtern Wissensdatenbanken und Suchverfahren innerhalb des übernommenen Unternehmens die Akquisitionsprozesse.[14]

Im weiteren Verlauf der Ausarbeitung sollen vier unterschiedliche Systeme vorgestellt werden und dabei auf Hintergrundinformationen und grundsätzliche Funktionsweise sowie Funktionalitäten eingegangen werden. Die Auswahl beschränkt sich auf vier bekannte Implementierungen von Meta-Suchmaschinen. Zum einen wird das von der Apache Software Foundation forcierte Projekt „Lucene" betrachtet werden. Dieses hat mittlerweile einen großen Bekanntheitsgrad und konnte einen weiten Kundenkreis für sich gewinnen. Zudem werden „Solr" und „Elasticsearch" als zwei unterschiedliche Formen der Erweiterung von Lucene betrachtet. Mit „HSearch" ist zusätzlich ein Vertreter dabei, der mit dem NoSQL-Konzept eine andere Richtung einschlägt und auf dem populären Framework „Hadoop" basiert.

3.1 Lucene

3.1.1 Hintergrundinformationen

Lucene ist ein in Java entwickeltes Open-Source-Projekt von Doug Cutting, welches mittlerweile bei der Apache Software Foundation angesiedelt und unter Apache 2.0 lizensiert ist. Dieses Open-Source Projekt ermöglicht die Implementierung einer plattformunabhängigen Suchmaschine, mit der eine Volltextsuche möglich ist.[15]
Wie bei Volltextsuchen üblich, bildet auch bei Lucene ein invertierter Index die Basis des Algorithmus. Bei diesem Index verweisen einzelne Wörter auf Dokumente, in denen die Wörter vorkommen. Zudem besteht die Möglichkeiten, weitere Informationen, wie die Position des Wortes innerhalb des Dokuments, zu speichern. Bei einer Suche kann dann mit Hilfe von Bitoperationen festgestellt werden, welche Dokumente zwei oder mehr Suchwörter gleichzeitig beinhalten. Diese werden dann als relevante Suchergebnisse an vorderster Stelle angezeigt.[16]
Angenommen, es wird eine Suchanfrage nach den Wörtern „Enterprise Search Engine" gestartet. Zunächst wird das bereits angelegte Wortverzeichnis durchlaufen, um so die beinhaltenden Dokumente zu finden. Die Suchvorgänge nach den einzelnen Wörtern werden dann mit AND-Operationen verkettet, sodass diejenigen Dokumente, die die meisten Suchbegriffe

[14] Vgl. White, M. (2012), S.9 ff.

[15] Vgl. The Apache Software Foundation (2012a)

[16] Vgl. Karich, P. (2012), S.68

enthalten, aufgespürt werden können.[17] Dieses Beispiel ist in Abb. 4 visualisiert. Das Suchwort „Engine“ ist in den Dokumenten 1, 3 und 4 enthalten während sich „Enterprise“ in den Dokumenten 1, 3, 5 und 7 findet und „Search“ in den Dokumenten 3, 5 und 6 vorzufinden ist. Damit enthält Dokument 3 die meisten Treffer und ist als bestes Ergebnis aufzuführen. An zweiter Stelle folgen Dokumente 1 und 5, und schließlich die Dateien 4, 6 und 7. Dokument Nr. 2 enthält keine Treffer und wird deshalb nicht aufgeführt.

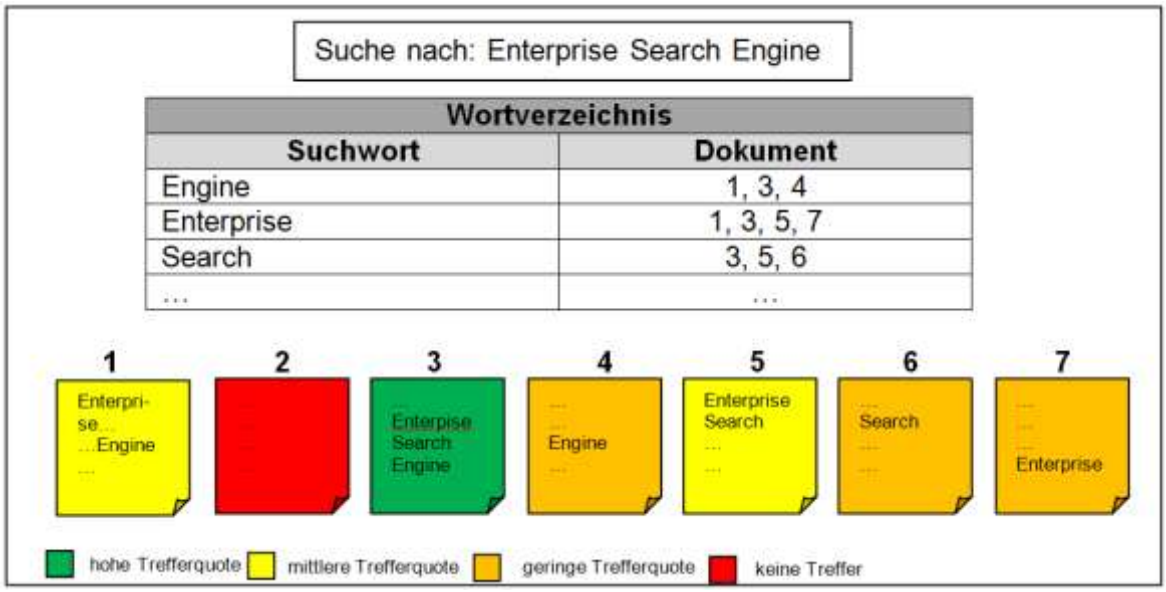


Abb. 4: Invertierter Index: Beispiel für eine Suche

Mit Lucene können 95 GB in einer Stunde durchsucht werden. Die RAM-Anforderungen sind dabei mit 1 MB im Heap sehr gering (siehe Anhang 2).
Die wichtigsten Hintergrundinformationen zu Lucene sind in Tab. 5 aufgeführt.

aktuelle Version (Jan. 2013)	4.0
Entwickler	Doug Cutting
Programmiersprache	Java
Lizenz	Apache 2.0 (kostenlos)
Benutzerkreis	u.a. Wikipedia, Twitter, FedEx, Hewlett-Packard
Besonderheiten	- Volltextsuche - Wildcard-Abfrage
Entwickler-Homepage	http://lucene.apache.org/
Informationsseite / Wiki	http://wiki.apache.org/lucene-java/FrontPage
Community	http://lucene.472066.n3.nabble.com/
Kontaktmöglichkeiten	http://lucene.apache.org/core/discussion.html

Tab. 5: Hintergrundinformationen zu Lucene

[17] Vgl. Karich, P. (2012), S.68

3.1.2 Betrachtung der Funktionsweise

Bei einer Suchmaschine stellt der Index das zentrale Element dar. Wie bereits erwähnt, handelt es sich bei Lucene um die besondere Form des invertierten Index. Das Suchwerkzeug zerlegt dabei ein Dokument in sog. Postings. Diese bestehen aus dem Wort w und dem Dokument d, in welchem dieses Wort vorkommt. Bei d handelt es sich nicht um das tatsächliche Dokument, sondern um einen Verweis auf dieses Dokument. Ein Posting enthält dabei nicht nur die Dokumenten-ID und das Wort, sondern oft auch noch die Häufigkeit des Auftretens eines Worts sowie die Positionen in einem Dokument. Bei normaler Indexierung wird jedes Dokument nacheinander nach dem gesuchtem Wort durchsucht. Dies hat natürlich eine geringe Performance zur Folge. Bei der invertierten Indexierung wird dies umgedreht. Es wird zuerst nach den Wörtern gesucht und dann auf die dazugehörigen Dokumente verwiesen.[18]

Dem invertierten Index sind zwei Prozesse vorgelagert, zum einem das Stopword-Removal und zum anderen das Stemming. Das Stopword-Removal basiert auf einer Negativliste. Dort sind jene Worte enthalten, die in einem Text so häufig vorkommen, dass sie für die Suche keinerlei Aussagekraft besitzen. Dies sind z.B. Worte, wie „der“, „ist“ und „hat“. Solche Wörter werden herausgefiltert und nicht für den Index herangezogen. Das Stemming reduziert Wörter auf ihren Wortstamm. Beispielweise werden „gefunden“ und „findet“ zu „find“ reduziert. Durch diese beiden Prozesse wird die zu indizierende Menge in der Regel um etwa 70% reduziert. Jedoch sind diese beiden Verfahren sprachenabhängig (siehe hierzu Abb. 5).[19]

[18] Vgl. Ziegler, C. (2006), S.120 f.

[19] Vgl. ebenda, S.121 f.

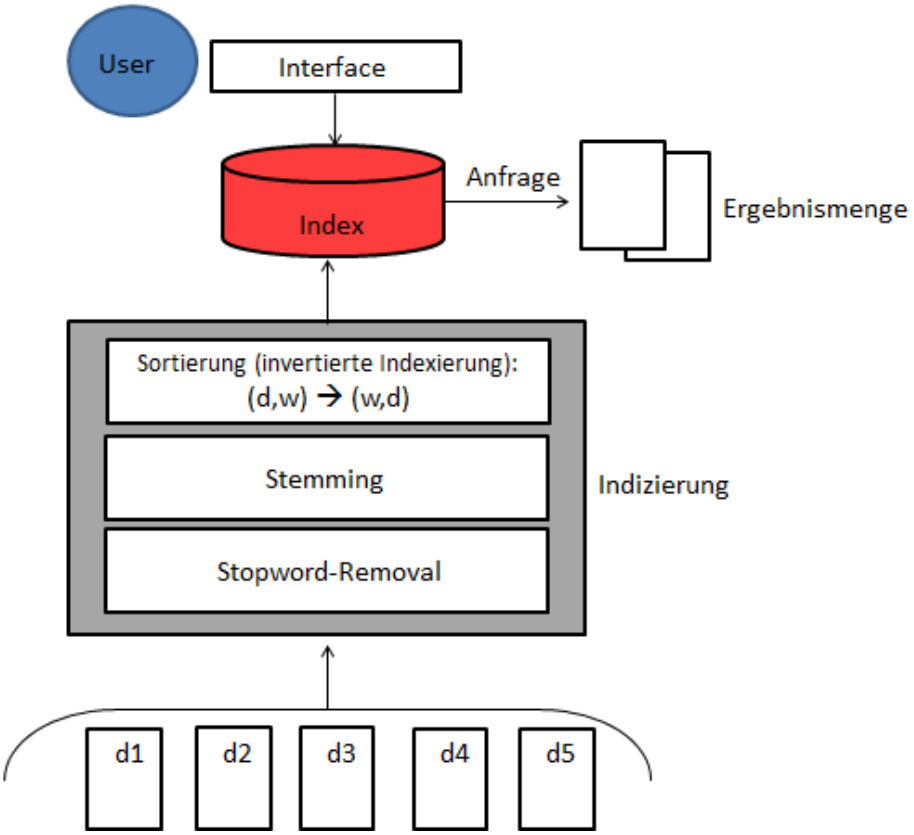


Abb. 5: Funktionsweise von Lucene[20]

Eine weitere Funktion von Lucene ist die Datumsbereichssuche. Mit dieser Funktion ist es möglich, Dokumente zu suchen, die innerhalb eines bestimmten Zeitraums erstellt oder zum letzten Mal abgespeichert wurden.[21]

Lucene besitzt auch eine Vielzahl von Operatoren, wie z.B. die Möglichkeit, Wildcards einzusetzen. Damit kann mit einem Suchbefehl nach mehreren ähnlichen Wörtern gesucht werden. Weitere wichtige Operatoren sind in der folgenden Tabelle (Tab. 6) dargestellt.

Typ	Beispiel	Erklärung
Keyword	Fußball	Sucht nach allen Dokumenten, die Fußball enthalten
Phrase Query	“Snow Crash“	Wählt alle Dokumente aus, in denen Snow Crash vorkommt (beide Wörter direkt hintereinander)
Konkatenation	Snow AND Crash	Snow und Crash müssen im Dokument vorkommen, aber nicht unbedingt direkt hintereinander
Disjunktion	Snow OR Crash	Nur einer der beiden Terme muss im Dokument enthalten sein

[20] Mit Änderungen entnommen aus: Ziegler, C. (2006), S.121

[21] Vgl. The Apache Software Foundation (2012a)

Feld-Query	Title “Snow Crash“	Sucht nach Dokumenten, die ein Feld namens Title enthalten, in dem Snow Crash vorkommt
Wildcard	Sch?tten / Min*g	Wählt alle Dokumente aus, die Schatten, Schotten beziehungsweise Ming, Mining oder Minderung enthalten
Proximity Search	“Snow Crash“~5	Wählt Dokumente aus, die Snow und Crash enthalten; beide Terme dürfen höchstens durch fünf Wörter voneinander getrennt sein
Term Boosting	Snow^3 Crash	Das Auftreten des Terms Snow wird dreimal so stark gewichtet wie das von Crash

Tab. 6: Wichtige Operatoren von Lucene[22]

Bei Lucene handelt es sich also lediglich um eine Java-Programmbibliothek, welche als Grundlage verwendet werden kann, eine eigene Suchmaschine zu implementieren. Lucene basiert auf einem invertierten Index. Dadurch werden Suchen deutlich schneller durchgeführt als bei einer normalen Indexierung. Eine weitere Besonderheit von Lucene sind die beiden vorgelagerten Prozesse, das Stopword-Removal und das Stemming. Die Performance der Suchmaschine wird dadurch nochmals deutlich gesteigert. Es ist jedoch zu beachten, dass Lucene nur eine sehr begrenzte Anzahl an Features beinhaltet. Auf dem Markt ist aber eine Vielzahl an Erweiterungen vorhanden. Mit diesen kann eine große Anzahl an Funktionen in Lucene-Projekte eingebaut werden.

3.2 Solr

3.2.1 Hintergrundinformationen

Solr ist im Jahre 2004 als eigenständige Suchplattform für CNET Networks-Intranet entstanden und hat sich im Jahr 2006 zu einer Open-Source-Suchplattform entwickelt. Um den aktuellen Leistungsstand des Programms zu erreichen, wurde das Solr-Projekt im Jahre 2010 mit dem Lucene-Projekt zusammengefasst.[23] Diese Verschmelzung hatte zur Folge, dass eine Volltextsuchmaschine, gleichwohl aber auch ein Suchframework entstanden ist.
Solr ist in Java geschrieben, basiert auf einer REST-Architektur und verwendet JSON als Datenaustauschformat.[24] REST steht für Representational State Transfer und es handelt sich dabei um ein Designmuster zur Gestaltung dynamischer Web-Inhalte. REST fasst unter anderem Standards wie HTTP, XML, HTML, JPEG und GIF zusammen. Der Begriff „Re-

[22] Mit Änderungen entnommen aus: Ziegler, C. (2006), S.123

[23] Vgl. Klose, M. (2012)

[24] Vgl. The Apache Software Foundation (2012b)

presentation“ steht in diesem Kontext für die Bereitstellung einer Ressource (zugrundeliegende Daten und Metadaten). Die Ressource wird in einer Datei, einem Web- bzw. Applikationsserver oder einem Datenbankserver bereitgestellt (Abb. 6). „State“ beschreibt in diesem Zusammenhang den aktuellen Zustand einer Anwendung und „Transformation“ steht für den Übergang des Zustands einer Anwendung durch Aufruf einer anderen Ressource.[25]

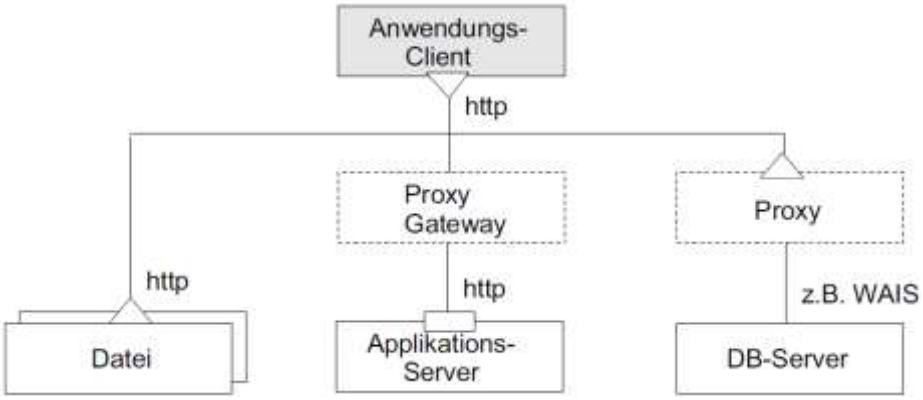


Abb. 6: Schema der REST-Architektur[26]

Bei JSON handelt es sich um ein programmiersprachenunabhängiges Textformat, das unter anderem Konventionen der Programmiersprachen C, C#, C++, Java und Perl beinhaltet.[27]
Seit Apache 4.0 hat Solr einen großen Sprung gemacht, denn erstmals wurde durch die Neueinführung der Solr Cloud die Konfiguration und Verwaltung verteilter Systeme zunehmend vereinfacht. In vorherigen Versionen führte der Ausfall einer Komponente (meist eines Servers) zum Ausfall des Gesamtsystems (Single Point of Failure). Sollte nun ein Server ausfallen, wird dieser durch einen anderen Server in der Cloud ersetzt.[28]
Aufgrund all dieser fortschrittlichen Entwicklungspunkte hat sich Apache Solr zu einer weit verbreiteten Java Suchmaschine entwickelt und kann einen breiten Nutzerstamm aufweisen. Zu den bekanntesten Nutzern zählen Ebay, Panasonic, AOL und Instagram.[29]
Die wichtigsten Basisinformationen zu Solr finden sich in Tab. 7.

[25] Vgl. Bauer, G. (2009), S.154 f.

[26] Enthalten in Bauer, G. (2009), S.155

[27] Vgl. o. V. (o. J.a)

[28] Vgl. Klose, M. (2012)

[29] Vgl. Worral, P. (2012)

aktuelle Version (Jan. 2013)	4.0
Entwickler	Apache Software Foundation
Programmiersprache	Java
Lizenz	Apache 2.0 (kostenlos)
Benutzerkreis	u.a. Ebay, Panasonic, AOL, Instagram
Besonderheiten	- Volltextsuche - REST-Architektur - JSON-Datenaustauschformat - Solr Cloud
Entwickler-Homepage	http://lucene.apache.org/solr/
Informationsseite / Wiki	http://wiki.apache.org/solr/
Community	http://lucene.472066.n3.nabble.com/Solr-f472067.html
Kontaktmöglichkeiten	http://lucene.apache.org/solr/discussion.html

Tab. 7: Hintergrundinformationen zu Solr

3.2.2 Betrachtung der Funktionsweise

Wie oben beschrieben, ist Solr eine Volltext-Suchmaschine. Diese kann sowohl nach Stichworten als auch nach Kenngrößen selektieren. Die vorhandene Web-Schnittstelle liefert zwei grundlegende Funktionen. Mittels HTTP-Request erfolgt eine Indizierung der Daten. Sollen Suchanfragen gestartet werden, bedient man sich der HTTP-Methode „GET“.[30]
Durch Clustering (als Cluster bezeichnet man das Zusammenfassen von Objekten mit ähnlichen Eigenschaften; eine bestimmte Anzahl an Datensätzen in einem Cluster bezeichnet man als Clustering) und mehrdimensionale Suche können dank der Datenbankintegration ERP-Systeme, CRM-Systeme, Dokument-Management-Systeme sowie E-Mails und andere webbasierte Dokumente durchsucht werden.[31]
Besonders interessant sind nachfolgende Funktionen von Solr:

- Echtzeitabfrage (Realtime-Get)

Dokumente lassen sich direkt nach der Indizierung aus der Solr Cloud abrufen. Dadurch entfällt eine explizite Suche.

[30] Vgl. The Apache Software Foundation (2012b)

[31] Vgl. Maas, G. (2012)

- Suche nahe an Echtzeit (NearRealTime)

Mit der Commit-Strategie „softCommit“ können neu hinzugefügte Dokumente ohne Latenzzeit gefunden werden. Die bedeutet, dass ein Dokument direkt nach der Indexierung für Suchvorgänge zur Verfügung steht.[32]

- Update-Sicherheit

Alle Änderungen werden durch einen „Transaction Log“ protokolliert. Somit gehen Dokumente, bei denen noch keine Transaktion erfolgt ist, nicht verloren.

- Automatisches Failover bei schreibenden oder lesenden Zugriffen.[33]

Solr übernimmt die Funktion eines Server-Wrappers, der über eine XML-over-HTTP-Schnittstelle die Funktionen der Java-Bibliotheken von Lucene bereitstellt. Hierbei übernimmt Solr die Rolle eines eigenständigen Suchservers. Diese Eigenschaften von Solr sind insbesondere in heterogenen Systemumgebungen, bei denen Anwender von unterschiedlichen Programmiersprachen aus auf eine Suchmaschine zugreifen, von Vorteil, da Solr plattformübergreifend eingesetzt werden kann. Mit der zugrundeliegenden Lucene-Bibliothek bietet Solr einige Zusatzfunktionen, welche mit dem Werkzeug mitgeliefert werden, sodass hierfür der Programmieraufwand entfällt. Bei diesen Zusatzfunktionen handelt es sich beispielsweise um folgende:

- typisierte Dokumentenfelder (nicht nur Strings)
- Vereinfachte Indexierung durch das Mapping von Dokumenten zur Datenquelle (Datenbanktabellen, XML- oder HTML-Dateien)
- Facettensuche
- Rechtschreibkorrektur
- Autovervollständigung / Suchvorschläge
- konfigurierbare Caches für Suchergebnisse
- Replikationsmechanismen
- Unterstützung von Sharing
- eigene Gewichtungskriterien bei der Erstellung von Rankings[34]

Die interessanteste Funktion von Solr verbirgt sich hinter dem Solr Facetting (zu Deutsch Facettensuche). Mit der Facettensuche kann ein bestehendes Suchergebnis nach weiteren

[32] Vgl. Miller, M. u. a. (2012)

[33] Vgl. Klose, M. (2012)

[34] Vgl. Kammerer T. / Mandl, P. / Baumgärtner, R. (2011), S.133 f.

Kriterien eingeschränkt werden. Damit lässt sich die eigene Suche durch Kombination von Filtern beliebig einschränken.[35] Dies kann man sich wie bei einer Suche in einem Online-Shop vorstellen: Neben dem eigentlichen Suchbegriff (z.B. T-Shirt) lassen sich zusätzlich Kategorien (z.B. Herrenbekleidung, Größe L) einstellen, sodass man präzisere Ergebnisse erhält. Diese Suche kann schneller und zielstrebiger zum gewünschten Ergebnis führen sowie die Qualität der selektierten Dateien steigern. Ebenfalls von sehr großem Vorteil sind die Möglichkeiten der syntaktischen und semantischen Vervollständigungs- und Korrekturmechanismen, da hierdurch die Suche nach den richtigen Ergebnissen deutlich vereinfacht und beschleunigt wird. Aus Sicht der Ergonomie sind diese Zusatzfunktionen besonders positiv zu bewerten.

Schließlich lassen sich noch die Suchresultate in unterschiedlichen Formaten, wie XML oder JSON zurückgeben. Die Integration in einen Webserver bzw. Servlet-Container (z.B. Apache Tomcat) ist ebenfalls möglich[36]

Zusammenfassend lässt sich festhalten, dass Solr lediglich als eine Art „Hülle“ fungiert, welche Lucene als Grundgerüst für dessen Funktionalitäten verwendet. Damit kann Solr als Suchmaschine ohne Lucene als Suchbibliothek nicht existieren. Der wesentliche Vorteil gegenüber Lucene liegt vor allem darin, dass Solr ohne großen Aufwand als vollwertige Suchmaschine eingesetzt werden kann und eine breite Palette an Zusatzfunktionen „out-of-the-box“ liefert. Möchte man das volle Potential ausnutzen, sollte wie in Abb. 7 vorgegangen werden. Mit Solr können sämtliche Dokumente in Business-Intelligence-Systemen durchsucht werden.

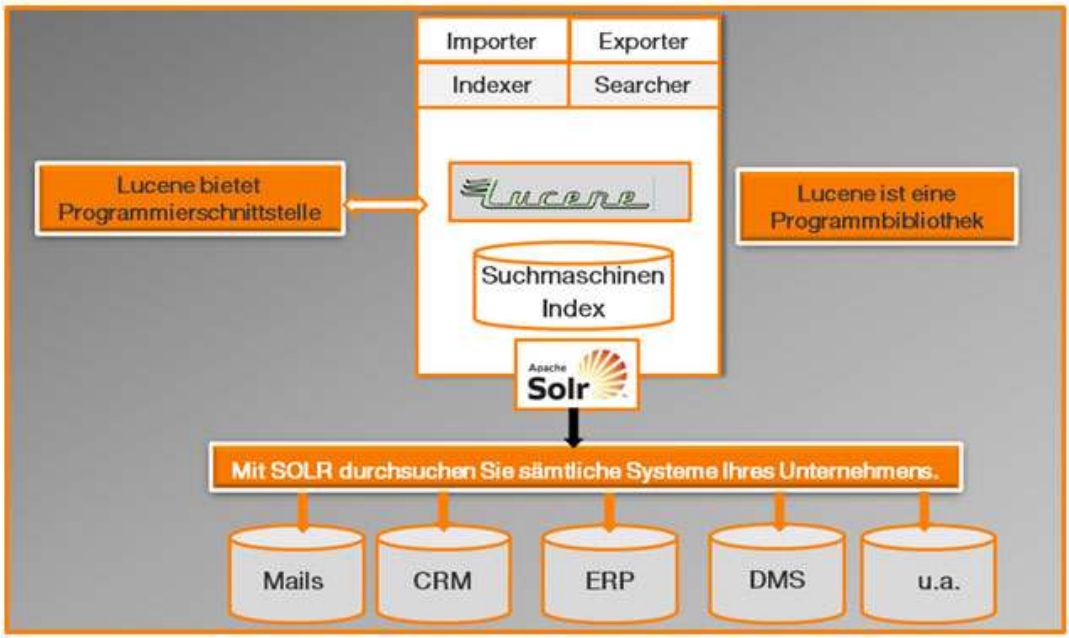


Abb. 7: Zusammenspiel von Solr und Lucene[37]

[35] Vgl. Maas, G. (2012)

[36] Vgl. ebenda

3.3 Elasticsearch

3.3.1 Hintergrundinformationen

Elasticsearch ist ein auf Lucene basierender, durch Apache 2.0 lizensierter und von Shay Banon entwickelter Suchserver.[38] Diesem liegt, wie auch bei Solr, eine REST-Architektur zugrunde. Ferner handelt es sich bei Elasticsearch um eine verteilte Suchmaschine.[39] Bei einer verteilten Suchmaschine wird die Suchanfrage an mehrere einzelne Computer weitergeleitet, welche jeweils eine eigene Suchmaschine besitzen. Anschließend werden die Ergebnisse zusammengeführt.[40]

Elasticsearch ist in Java programmiert, verwendet JSON als Datenaustauschformat sowie dokumentorientierte Datenbanken. Zu den größten Nutzern von Elasticsearch gehören Mozilla, StumbleUpon und Sony Computer Entertainment. Laut Entwickler-Homepage sind derzeit zwölf kommerzielle, bekannte Nutzer aufgezählt.[41]

In Tab. 8 werden die Hintergrundinformationen zu Elasticsearch zusammengefasst.

aktuelle Version (Jan. 2013)	0.20.2
Entwickler	Shay Banon / Elasticsearch (Firma)
Programmiersprache	Java
Lizenz	Apache 2.0 (kostenlos)
Benutzerkreis	u.a. Mozilla, StumbleUpon, Sony Computer Entertainment
Besonderheiten	- REST-Architektur - JSON-Datenaustauschformat - verteilte Suchmaschine
Entwickler-Homepage	http://www.elasticsearch.org/ (Entwicklerseite) http://www.elasticsearch.com/ (Unternehmensseite)
Informationsseite / Wiki	http://www.elasticsearch.org/guide/
Community	https://groups.google.com/forum/#!forum/elasticsearch
Kontaktmöglichkeiten	http://www.elasticsearch.org/community/

Tab. 8: Hintergrundinformationen zu Elasticsearch

---

[37] Enthalten in: Maas, G. (2012)

[38] Vgl. Elasticsearch (o. J.a)

[39] Vgl. ebenda

[40] Vgl. Köhler, M./ Arndt, H./ Fetzer, T (2008), S.3

[41] Vgl. Elasticsearch (o. J.a)

3.3.2 Betrachtung der Funktionsweise

Im ersten Schritt werden die vorhandenen Dokumente an die Suchmaschine übergeben. Dieser Vorgang wird als Feeding oder Indexing bezeichnet. Dabei wird, wie schon bei Lucene, ein invertierter Index erstellt (ähnlich einem Wortverzeichnis in einem Buch).
Bei der Indexierung eines Dokuments sind in Elasticsearch eine Identifikationsnummer (ID), der Typ des Dokuments und dessen Index anzugeben. Falls einer dieser Bestandteile fehlt, wird er von der Suchmaschine automatisch angelegt. Daneben lässt sich auch die Zeit konfigurieren, nach der ein indexiertes Dokument auffindbar sein soll. Bei Bedarf lässt sich auch ein sofortiges Auffinden erzwingen. Alle Felder werden, unabhängig vom Typ, in einem optionalen, nichtindexierten Feld „_source“ abgelegt. Bei diesem können über „GET-Abfragen“ die abgelegten Felder ohne Verzögerung durch einen Zugriff auf die ID abgerufen werden. Folglich kann _source als NoSQL-Datenspeicher verwendet werden. Dies ermöglicht eine übersichtliche Datenhaltung ohne die Notwendigkeit von externen Datenbanklösungen sowie ein besonders schnelles Auffinden von Dokumenten. Strings werden standardmäßig automatisch indexiert und im Feld „_all“ gespeichert. Dieses ermöglicht wiederum eine Anfrage an alle Felder gleichzeitig. Die automatische Indexierung lässt sich bedarfsweise deaktivieren.[42]

Da es sich hier um eine verteilte Suchmaschine handelt, bietet Elasticsearch die Möglichkeit, Indizes in sog. „Shards“ aufzuteilen. Mit Hilfe von Shards können Datenmengen auf mehreren Rechnern verwaltet werden. Dadurch wird das Problem des beschränkten Haupt- oder Festplattenspeicherplatzes eines einzelnen Rechners umgangen. Die Anzahl der Shards lässt sich nach der erstmaligen Festlegung nicht nachträglich ändern. Die Indizes selbst lassen sich ebenfalls replizieren. Bei einer Indexreplikation werden gleichzeitig die vorhandenen Shards vervielfältigt. Dies bietet einige Vorzüge, wie z.B. die Verteilung der Anfragelast auf mehreren Rechnern oder unterschiedliche Einstellungsmöglichkeiten für verschiedene Indizes / Shards (z.B. ein Index für den Hauptspeicher und eine Index für den Festplattenspeicher).[43]
Laufende Instanzen von Elasticsearch werden auf verteilten Systemen als Nodes bezeichnet. Diese speichern die lokalen Indizes und Shards. Einzelne Nodes können zu Clustern zusammengefasst werden. Cluster ermöglichen mehrere voneinander unabhängige Instanzen von Elasticsearch, z.B. für Entwicklungs-, Test- und Produktionsumgebung oder für unterschiedliche Benutzer. Bei der Clusterverwaltung wird zwischen Master und Slaves unterschieden. Ersterer übernimmt die Clusterverwaltung und koordiniert Veränderungen, wie zum Beispiel Änderungen bei der Anzahl der Nodes oder Indizes und einer damit verbundenen Verschiebung von Shards. Nichtverwaltende Instanzen heißen Slaves. Bei einem Ausfall

[42] Vgl. Karich, P. (2012), S.68

[43] Vgl. Elasticsearch (o. J.a)

eines Masters-Nodes springt einer der Slave-Nodes ein. Die Rollenverteilung kann jederzeit manuell verändert werden.[44]

Im Folgenden wird ein Beispiel für die Indexierung, Replikation und Lastverteilung grafisch dargestellt:[45]

Schritt 1 (Abb. 8):

- Client und Node starten
- auf Node: Erstellung eines Index mit 2 Shards (S1 und S2) und einer Replikation (für die Replikation wird ein zweiter Node benötigt)

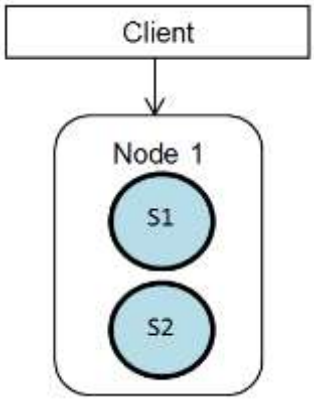


Abb. 8: Elasticsearch: Indexierung und Indexreplikation (Schritt 1)

Schritt 2 (Abb. 9):

- zweiter Node wird gestartet
- auf zweitem Node findet die Replikation der Shards (B1 und B2) automatisch statt

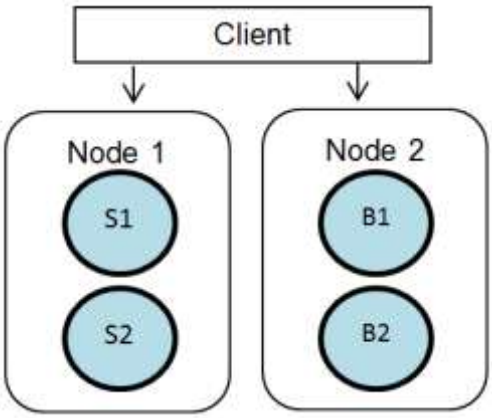


Abb. 9: Elasticsearch: Indexierung und Indexreplikation (Schritt 2)

[44] Vgl. Karich, P. (2012), S.70

[45] Vgl. Elasticsearch (o. J.a)

Schritt 3 (Abb. 10):

- dritter Node wird gestartet
- Shards verteilen sich gleichmäßig auf den Nodes
- hier: Node 2 und Node 3 teilen sich die Shards

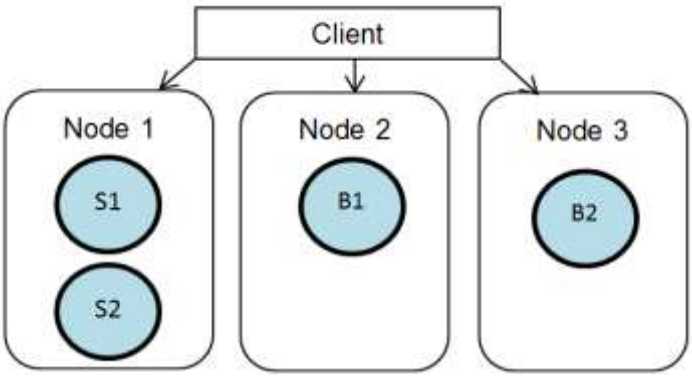


Abb. 10: Elasticsearch: Indexierung und Indexreplikation (Schritt 3)

Schritt 4 (Abb. 11):

- vierter Node wird gestartet
- Lastverteilung analog zu Schritt 3

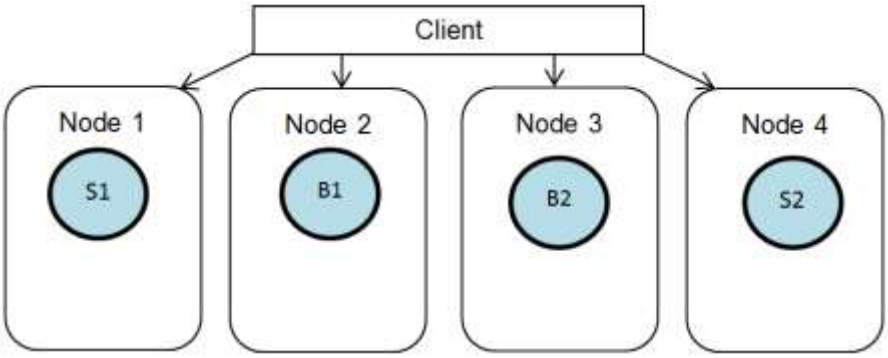


Abb. 11: Elasticsearch: Indexierung und Indexreplikation (Schritt 4)

Wie auch schon bei Solr, werden Suchanfragen über HTTP-GET abgewickelt. Daneben ist auch die Facettensuche, deren Funktionsweise in Kapitel 3.2.2 beschrieben wurde, eine in Elasticsearch unterstützte Funktion. Eine weitere Funktion, die sich Solr und Elasticsearch teilen, ist die Cloud-Anbindung. Ebenfalls erwähnenswert ist das sog. Gateway. Mit dessen Hilfe kann im Falle eines Ausfalls des gesamten Clusterverbunds der letzte Zustand wiederhergestellt werden. Dieser kann entweder aus dem lokalen Node-Speicher eingespielt oder aus einem Shared-Storage (z.B. NFS oder Amazon Simple Storage Service) repliziert werden.[46]

[46] Vgl. Elasticsearch (o. J.a)

Aus der Analyse von Elasticsearch geht hervor, dass einige nützliche Funktionen aus dem Lucene-Projekt auch für Elasticsearch übernommen wurden. Auch finden sich einige Ähnlichkeiten mit Solr (z.B. JSON und REST-Architektur). Elasticsearch kann sich insbesondere mit Funktionen, wie zum Beispiel der Indexreplikation zur Lastverteilung, Gateway zur verbesserten Ausfallsicherheit oder der Bereitstellung eines NoSQL-Datenspeichers von den anderen Suchmaschinenprojekten abheben. Jedoch signalisiert die aktuelle Version 0.20.2, dass das Projekt noch nicht ganz ausgereift und sich noch ein einer recht frühen Entwicklungsphase befindet.

3.4 HSearch

3.4.1 Hintergrundinformationen

HSearch ist eine auf Hadoop basierende, unter Apache 2.0 lizensierte und von der Firma Bizosys entwickelte Echtzeitsuchmaschine. Es handelt sich auch in diesem Fall um eine verteilte Suchmaschine mit einer REST-Architektur. Unterstützt werden strukturierte (z.B. relationale Datenbanken)[47], semi-strukturierte (z.B. Message Stores und Verzeichnisdienste)[48] und unstrukturierte Datenquellen (z.B. Emails, Daten aus sozialen Netzwerken, Bilder und Videos)[49]. Damit können Produkte aus dem Bereich der Business Intelligence (beispielsweise ERP- oder CRM-Systeme), unternehmensinterne Anwendungen (z.B. Content Management Systeme oder E-Mail-Anwendungen) und schemafreie Datenbestände aus Social-Media-Anwendungen durchsucht werden. Die aktuellste Version 0.94 vom 16. Februar 2012 ist mit neueren Windows- und Unix-Betriebssystemen kompatibel.[50]
Die Suchmaschine verfolgt das Ziel, 100 Milliarden Datensätze zu indexieren. Der vollständige Index wird dabei in einer HBase-Datenbank gespeichert.[51] Bei HBase handelt es sich um die NoSQL-Datenbank von Hadoop. HBase unterstützt sehr große Datenmengen, ist verteilt und frei skalierbar. Hadoop ist die von der Apache Software Foundation entwickelte Software Bibliothek für NoSQL-Datenbanken. Dabei handelt es sich um ein Java-Framework für verteilte und skalierbare Anwendungen.[52]
In Tab. 9 sind die Basisinformationen zu HSearch dargestellt.

---

47 Vgl. Schwichtenberg, H. (2001), S. 306

48 Vgl. ebenda

49 Vgl. Schonschek, O. (o. J.)

50 Vgl. Bizosys Technologies Pvt Ltd. (2011)

51 Vgl. Bizosys Technologies Pvt Ltd. (2010a)

52 Vgl. The Apache Software Foundation (2012c)

aktuelle Version (Jan. 2013)	0.94 (vom 16. Februar 2012)
Entwickler	Bizosys Technologies Pvt Ltd.
Programmiersprache	Java
Lizenz	Apache 2.0 (kostenlos)
Benutzerkreis	keine Informationen vorhanden
Besonderheiten	- Volltextsuche - REST-Architektur - verteilte Suchmaschine - unterstützt Vielzahl an Datenformaten - NoSQL Datenbank HBase (Hadoop Framework)
Entwickler-Homepage	http://bizosyshsearch.sourceforge.net/
Informationsseite/ Wiki	http://www.hadoopsearch.net/
Community	http://www.bizosys.com/forum.html
Kontaktmöglichkeiten	http://sourceforge.net/projects/bizosyshsearch/support

Tab. 9: Hintergrundinformationen zu HSearch

3.4.2 Betrachtung der Funktionsweise

Mit HSearch lassen sich eine Reihe unterschiedlicher Quellen durchsuchen, darunter relationale Datenbanken, Mail Server, Dateisysteme, Webseiten, RSS Feeds und XML-Dateien. Die gefundenen Ergebnisse lassen sich dabei beliebig darstellen, z.B. als Listen mit Filter- und Sortierfunktion oder in dynamischen Portlets. Daneben gibt es Möglichkeiten der Vorschau für Suchergebnisse und Such- und Navigationshilfen (z.B. Brotkrümelnavigation und Facetttensuche) sowie eine Cloud Unterstützung.[53]

Aus Sicherheitssicht werden eine LDAP-Anbindung sowie die Integration anderer Sicherheitsschnittstellen, wie z.B. Kerberos oder X.509 unterstützt. Zudem erlaubt HSearch eine Authentifizierung über Web-Proxys und die Verschlüsselung sensitiver Indizes. Hardwareseitig ist eine Skalierung mit mehreren Festplatten gewährleistet. Auch ist eine parallele Indizierung und Suche in Clusterumgebungen möglich. Innerhalb von Clusterverbänden kann eine Replikation konfiguriert werden, was zu einer verbesserten Ausfallsicherheit beiträgt. So können bei einem Maschinenausfall die dann fehlenden Daten über eine andere Maschine wiederhergestellt werden.[54]

Bei der Indizierung wird derzeit nur die englische Sprache unterstützt. Hierbei sind Synonyme und Stoppwörter bzw. -sätze möglich. Mit letzteren sind solche Wörter und Sätze ge-

[53] Vgl. Bizosys Technologies Pvt Ltd. (2010b)

[54] Vgl. ebenda

meint, die bei einer Volltextindizierung aufgrund ihres sehr häufigen Vorkommens nicht erfasst werden (z.B. Artikel, Konjunktionen oder Präpositionen). Darüber hinaus können 267 Dateiendungen indiziert werden. Zur Reduzierung des Speicherbedarfs können Indizes komprimiert werden. Weitere Funktionalitäten sind die Begriffsklärung („Meinten Sie…“) sowie die unscharfe Suche. Die Gewichtung der Suchergebnisse kann über Ranking-Methoden konfiguriert werden. Dabei lässt sich die Relevanz der Ergebnisse anhand verschiedener Faktoren festgelegt werden, wie zum Beispiel Aktualität, Speicherort, Autor oder Art der Quelle. Die Suchmaschine ist in kleinen Teilen auch selbstlernend. Sie kann, je nach dem Ranking des Benutzers, die Suchinhalte hervorheben und anzeigen. Diese Gewichtung ist dynamisch und kann wie die anderen Punkte auch jederzeit angepasst oder parallel genutzt werden.[55]

Hinsichtlich der Administration bietet HSearch einen reichen Funktionsumfang: Zum einen lässt sich ein Benutzerzugriffskonzept festlegen, bei dem UserIDs, Rollen, Teams und Abteilungen verwaltet werden können. Zum anderen lassen sich beispielsweise Ranking-Regeln, Sprachen und Datenquellen einrichten. Einen weiteren Bestandteil der Administration bilden Reporting- und Monitoring-Werkzeuge zur Überwachung von Indizierung und Suchvorgängen. Im Reporting lassen sich die Top-100-Abfragen darstellen, ergebnislose Suchläufe sammeln und Wörterbuch- sowie Synonymsammlungen erstellen.[56] Weitere Funktionen von HSearch können Anhang 1 entnommen werden.

Abschließend kann festgehalten werden, dass HSearch eine auf Hadoop basierende Suchmaschine ist, die viele weitreichende Funktionen für den Anwender bietet. Aktuell steht HSearch noch bei der Version 0.94 und scheint sich in einer erweiterten Betaphase zu befinden. Positiv aufgefallen sind vor allem die weitreichenden Möglichkeiten, unterschiedlichste Dokumente zu suchen, und die zahlreichen Administrationswerkzeuge. Fortschritte müssen gerade noch im Bereich Dokumentation gemacht werden, da Informationsquellen zu HSearch nur spärlich vorhanden sind.

[55] Vgl. Bizosys Technologies Pvt Ltd. (2010b)

[56] Vgl. ebenda

4 Einzelbewertung der Suchmaschinen

Nachfolgend wird die Bewertung der Suchmaschinen nach dem Bewertungskonzept aus Kapitel 2 durchgeführt. Dabei wird jede einzelne Suchmaschine in zwei Schritten bewertet: Zunächst wird das zugrundeliegende Open Source Projekt betrachtet, anschließend das Suchsystem selbst.

4.1 Bewertung von Lucene

In diesem Kapitel wird das Projekt Lucene der Apache Software Foundation anhand der gegebenen Kriterien bewertet. In Tab. 10 erfolgt zunächst die reine Projektbewertung.

	Lucene
Produkt	
Produktalter	• erster Release 1997 • seit 2004 läuft das Projekt bei der Apache Software Foundation[57]
Lizenzen	Apache 2.0 (freie, kostenlose Lizenz)[58]
Plattform u. Kompatibilität	Java-Entwicklung, damit plattformunabhängig; unterstützte Betriebssysteme: • Windows 95 – Windows 7 • Linux • Mac OS, Mac OS X[59]
Funktionalität (exemplarisch)	• Volltextsuche • Keyword-Suche • Bereichssuche • Wildcard-Suche • Phrasensuche[60]
Modularität	für Lucene sind einige Add-Ons vorhanden, z.B.: • Lucene.Net (Lucene in C#) • Lucy (Lucene in C) • Droids (Framework für Bots/Crawler)

[57] Vgl. Walker-Morgan, D. (2011)

[58] Vgl. The Apache Software Foundation (2012d)

[59] Vgl. o. V. (2013)

[60] Vgl. The Apache Software Foundation (2012a)

	• LIMO (Lucene Index Monitor) • Tika (Unterstützung unterschiedlicher Dokument-typen) • Nutch (Java-Framework für Suchmaschinen) • Bobo (Bibliothek zur Facettensuche)[61]
Releaseabstände	• Lucene 3.4: September 2011 • Lucene 3.5 : November 2011 • Lucene 3.6: April 2012 • Lucene 3.6.1: Juli 2012 • Lucene 4.0: Oktober 2012 • Lucene 3.6.2: Dezember 2012[62] im Durchschnitt wird alle 2,5 Monate ein Update für Lucene veröffentlicht
Dokumentation	
Benutzerhandbuch vorhanden	• Hauptdokumentation im Lucene-Wiki • Release-Dokumentaion[63] • Tutorials inklusive Videos vorhanden[64]
Sprache der Dokumentation	ausschließlich englisch
Qualität der Dokumentation	• ausführliche und umfangreiche Dokumentation inklusive beispielhaftem Quellcode im Wiki • Tutorials mit knappen Erklärungen und Quellcode • keine Downloadmöglichkeiten als PDF-Datei o.Ä.
Support	
Community und Ökosystem	Stand: Januar 2013 • es gibt ein eigenes Lucene Forum • 3.464 User sind dort angemeldet • 98.619 Themen wurden in diesem Forum erstellt • auf die meisten Fragen wird relativ schnell ge-antwortet[65]

[61] Vgl. Rowe, S. u. a. (2011)

[62] Vgl. The Apache Software Foundation (2012e)

[63] Vgl. The Apache Software Foundation (2012f)

[64] Vgl. The Apache Software Foundation (2012g)

[65] Vgl. o. V. (o. J.b)

direkter Ansprechpartner	• Mailing-List mit E-Mail Adressen von Entwicklern vorhanden • Kontakt über IRC möglich[66] • Kontaktversuch per E-Mail war jedoch erfolglos
Vorhandensein von FAQ	FAQ auf Entwickler-Homepage vorhanden[67]
	Marktakzeptanz
Anzahl der Stakeholder	Unternehmen, die Lucene verwenden: • AOL • Apple • Disney • IBM • Twitter • Wikipedia Insgesamt 271[68] Sponsoren der Apache Software Foundation u.a.: • Google • Microsoft • HP • AMD • Citrix • IBM und ca. 40 andere Unternehmen und Einzelpersonen[69]
Anzahl der Downloads	keine Informationen vorhanden
Traffic Rank	Stand: Januar 2013 Traffic Rank von apache.org (gesamte Apache-Stiftung): • Weltweit: 1.353 • Deutschland: 1.852[70] Traffic Rank von http://lucene.apache.org/ nicht fest-stellbar.

Tab. 10: Bewertung des Lucene-Projekts

[66] Vgl. The Apache Software Foundation (2012h)

[67] Vgl. Rowe, S. u. a. (2011)

[68] Vgl. ebenda

[69] Vgl. The Apache Software Foundation (2012m)

[70] Vgl. Alexa Internet, Inc. (2013a)

Als nächstes wird der Fokus auf die Suchmaschinen-Bibliothek selbst gelegt (Tab. 11).

	Lucene
Kriterium	**Beurteilung**
grundsätzliche Funktionen	
Codierung u. Sprachunterstützung	Lucene unterstützt die deutschen Sonderzeichen (UTF-8)[71]
Crawler	• ein Crawler ist in Lucene nicht vorhanden • evtl. durch Add-Ons erweiterbar[72]
Dateiunterstützung	über das Toolkit „Apache Tika“ lassen sich diverse Dateien indexieren: - Office Document Format - PDF - HTML, XML - OpenDocument Format - Rich Text Format - Audioformate (MP3, MIDI) - Videoformate (Flash) - Bildformate (JPEG) JAVA-Klassen-Dateien[73]
Cloud Unterstützung	keine Cloud-Unterstützung (nur z.B. über Solr möglich)[74]
Suchalgorithmus	
Volltextsuche	Volltextsuche unterstützt[75]
Synonyme u. Korrektur	• Stemming: Wort wird auf seinen Wortstamm reduziert (Beispiel: „Gefunden“ und „findet“ werden zu „find“ reduziert); bei einer Suche werden alle Ergebnisse zu diesem Wortstamm angezeigt.[76]

[71] Vgl. The Apache Software Foundation (2006)

[72] Vgl. Rowe, S. u. a. (2011)

[73] Vgl. ebenda

[74] Vgl. Schmidt, J. (2012)

[75] Vgl. The Apache Software Foundation (2012a)

[76] Vgl. Schneider, T. (2004)

	• Rechtschreibkorrektur enthalten[77]
Ergebnisdarstellung	
Cached Results u. Hintergrundinformationen	• auf einzelne Ergebnisse kann verlinkt werden (Cached Results)[78] • Möglichkeit zur Anzeige von Hintergrundinformationen nicht vorhanden
Ausgabe der Suchergebnisse	individuelle Ausgabe über Webinterface möglich[79]
Skalierbarkeit	
Skalierbarkeit	Mit Lucene direkt nicht zu realisieren. In Verbindung mit Solr und dem SolrCloud-Projekt können z.B. Indexierungsaufgaben auf mehrere Server verteilt werden.[80]
Möglichkeiten der Anbindung	
LDAP-Anbindung / Umsetzung bestehender Zugriffsrechte	Anbindung an LDAP ist nicht möglich (nur über Solr etc. realisierbar)[81]
Anbindung an Datenquellen	Lucene ist nur eine Programmbibliothek und kann deswegen nicht direkt an Datenquellen angebunden werden.[82]
Administration	
Wartungs- und Betreuungsmöglichkeiten	eine Monitoring-Erweiterung kann eingebaut werden (Add-On: Apache LIMO).[83]
Berechtigungsverwaltung	• Indexdateien standardmäßig für alle lesbar und beschreibbar • die dafür zuständige Methode kann jedoch überschrieben werden • Berechtigungsverwaltung für einzelne Benutzer und Gruppen möglich[84]

[77] Vgl. Hülsbömer, S. (2011)

[78] Vgl. Rowe, S. u. a. (2011)

[79] Vgl. Kumar, B. (2006)

[80] Vgl. Schmidt, J. (2012)

[81] Vgl. Lightwerk (o. J.)

[82] Vgl. The Apache Software Foundation (2012i)

[83] Vgl. o. V. (o. J c)

[84] Vgl. Zend Technologies Ltd. (2012)

Programmiersprache	• in Java geschrieben[85] • Implementierungen auch in C#[86] und C++[87]
nichtfunktionale Anforderungen	
Performance	• Indexierung 95 GB / Stunde bei moderner Hardware[88] • zum Teil sogar noch deutlich schneller (siehe Anhang 2)
intuitive Bedienung	• zur allgemeinen Bedienung: Weboberflächen verfügbar • zur Erweiterbarkeit: einfache Programmierung in Java • Tutorials sind auf der Homepage vorhanden[89]
einfache Installation / Konfiguration	Lucene ist einfach zu installieren. Es muss lediglich die Programmbibliothek in Java eingebunden werden.[90]
Kompatibilität	• plattformunabhängig • mit Windows 95 bis Windows 7 kompatibel • mit Mac OS / Mac OS X kompatibel • läuft unter Java 6 oder höher[91]

Tab. 11: Bewertung von Lucene

[85] Vgl. The Apache Software Foundation (2012a)

[86] Vgl. The Apache Software Foundation (2012j)

[87] Vgl. Neumann, A. (2012)

[88] Vgl. The Apache Software Foundation (2012a)

[89] Vgl. ebenda

[90] Vgl. The Apache Software Foundation (2012i)

[91] Vgl. o. V. (2013)

4.2 Bewertung von Solr

In diesem Kapitel wird das Projekt Solr der Apache Software Foundation anhand der gegebenen Kriterien bewertet. In Tab. 12 erfolgt zunächst die reine Projektbewertung.

	Solr
Produkt	
Produktalter	erster Release 2004 von CNET[92]
Lizenzen	Apache 2.0 (freie, kostenlose Lizenz)[93]
Plattform u. Kompatibilität	Java-Entwicklung, damit plattformunabhängig; • Windows-Systeme • Linux • Mac OS, Mac OS X[94]
Funktionalität (exemplarisch)	• voll funktionsfähiger Suchserver "out of the box" • Volltextsuche • HTML-Administrationsinterface • Hit-Highlighting • Cloud-Unterstützung • Echtzeitabfrage (Realtime-Get) • zahlreiche funktionale Erweiterungen gegenüber Lucene (z.B. linguistische Textanalyse, Facetting, Caching)[95]
Modularität	erweiterbar durch Vielzahl an Plugins, z.B.: • ValueSourceParser • Highlighting • SolrFragmenter • SolrFormatter • Similarity • CacheRegenerator • Analyzer

[92] Vgl. Klose, M.(2012)

[93] Vgl. The Apache Software Foundation (2012b)

[94] Vgl. Innovent Solutions (o. J.)

[95] Vgl. Comundus (2011)

	• Tokenizer and TokenFilter • FieldType • Internals • SolrCache • UpdateHandler[96]
Releaseabstände	• alle 6 Monate[97]
Dokumentation	
Benutzerhandbuch vorhanden	• Hauptdokumentation im Solr-Wiki[98] • Release-Dokumentation • Tutorials inklusive Videos vorhanden[99]
Sprache der Dokumentation	ausschließlich englisch
Qualität der Dokumentation	• ausführliche und umfangreiche Dokumentation inklusive beispielhaftem Quellcode im Wiki • Tutorials mit knappen Erklärungen und Quellcode • keine Downloadmöglichkeiten als PDF-Datei o.Ä.
Support	
Community und Ökosystem	Stand: Januar 2013 • Forum mit 3464 Nutzern • 28428 Themen auf 813 Seiten • ca. 27.000 Views • E-Mail • IRC • Facebook • Twitter[100]
direkter Ansprechpartner	• Mailing-List mit E-Mail Adressen von Entwicklern vorhanden • Kontakt über IRC möglich[101]
Vorhandensein von FAQ	FAQ in Wiki implementiert[102]

[96] Vgl. Miller, M u. a. (2012)

[97] Vgl. The Apache Software Foundation (2012n)

[98] Vgl. Miller, M u. a. (2012)

[99] Vgl. The Apache Software Foundation (2012l)

[100] Vgl. o. V. (o. J.d)

[101] Vgl. The Apache Software Foundation (2012k)

[102] Vgl. Miller, M u. a. (2012)

Marktakzeptanz	
Anzahl der Stakeholder	bekannte Nutzer: • Ebay • AOL • Disney • Panasonic • Instagram • AT & T • NASA • SourceForge • buy.com • Apple • MTV • CISCO und mehr als 250 weitere[103] Sponsoren der Apache Software Foundation u.a.: • Google • Microsoft • HP • AMD • Citrix • IBM und ca. 40 andere Unternehmen und Einzelpersonen[104]
Anzahl der Downloads	keine Informationen vorhanden
Traffic Rank	Stand: Januar 2013 Traffic Rank von apache.org (gesamte Apache-Stiftung): • Weltweit: 1.353 • Deutschland: 1.852[105] Traffic Rank von http://lucene.apache.org/solr/ nicht fest-stellbar.

Tab. 12: Bewertung des Solr-Projekts

[103] Vgl. Worral, P. (2012)

[104] Vgl. The Apache Software Foundation (2012m)

[105] Vgl. Alexa Internet, Inc. (2013a)

Als nächstes wird der Fokus auf Solr selbst als Suchmaschine gelegt (Tab. 13).

	Solr
Kriterium	**Beurteilung**
grundsätzliche Funktionen	
Codierung u. Sprachunterstützung	Solr unterstützt die deutschen Sonderzeichen (UTF-8)[106]
Crawler	Webcrawler über Apache Nutch realisiert[107]
Dateiunterstützung	über das Toolkit „Apache Tika“ lassen sich diverse Dateien indexieren: • Office Document Format • PDF • HTML, XML • OpenDocument Format • Rich Text Format • Audioformate (MP3, MIDI) • Videoformate (Flash) • Bildformate (JPEG) • JAVA-Klassen-Dateien[108]
Cloud Unterstützung	Cloud-Unterstützung out-of-the-box[109]
Suchalgorithmus	
Volltextsuche	Volltextsuche unterstützt[110]
Synonyme u. Korrektur	• Auto-Vervollständigung • Solr Facetting • Rechtschreibvorschläge • Synonymsuche • Hervorhebung

[106] Vgl. Miller, M u. a. (2012)

[107] Vgl. Nioche, J. (2011)

[108] Vgl. o. V. (o. J.e)

[109] Vgl. Miller, M u. a. (2012)

[110] Vgl. The Apache Software Foundation (2012b)

	• Stopwörter • Gewichtung[111]
Ergebnisdarstellung	
Cached Results u. Hintergrundinformationen	Anzeigen von Hintergrundinformationen ist möglich, zum Beispiel: Aufrufanzahl, letzte Bearbeitung o.Ä.
Ausgabe der Suchergebnisse	Ausgabe z.B. in Webinterface möglich.[112]
Skalierbarkeit	
Skalierbarkeit	• SolrCloud ermöglicht die Integration einer großen Server-Infrastruktur • Replikationsmöglichkeiten und Load-Balancer-Systeme
Möglichkeiten der Anbindung	
LDAP-Anbindung / Umsetzung bestehender Zugriffsrechte	Authentifizierung über LDAP ist möglich[113]
Anbindung an Datenquellen	Sämtliche Systeme des Unternehmens, wie z.B.: • E-Mails • ERP • DMS • CRM • Wikis • Exchange Server • File Server • Intranet-Dokumente[114] • Lotus Notes DB über Solr NotesDB Connector[115]
Administration	
Wartungs- und Betreuungsm.	administrative Steuerung und Solr Monitoring bieten gezielte Wartungs- und Betreuungsoptionen[116]

[111] Vgl. dkd Internet Service GmbH (2013)

[112] Vgl. o. V. (o. J.f)

[113] Vgl. Miller, M u. a. (2012)

[114] Vgl. Comundus (2011)

[115] Vgl. Search Technologies Corp. (2013)

Berechtigungsverwaltung	• Verwaltung über ein Benutzer- oder Gruppenkonzept • z.B. Integration des Berechtigungskonzepts aus Active Directory möglich
Programmiersprache	Java, aber auch Portierungen u.a. nach C++, Python, Perl
nichtfunktionale Anforderungen	
Performance	• Echtzeitsuche • 50 Indexe pro Sekunde • daraus folgt eine Antwortzeit von unter 50 ms[117]
intuitive Bedienung	• zur allgemeinen Bedienung: Weboberflächen verfügbar • zur Erweiterbarkeit: einfache Programmierung in Java • Tutorials sind auf der Homepage vorhanden
einfache Installation / Konfiguration	Anleitung / Tutorials für Installation und Konfiguration sind auf der Homepage (mit Codeausschnitten) sowie in Wiki verfügbar[118]
Kompatibilität	• plattformunabhängig • mit Windows-Systemen kompatibel[119] • mit Mac OS / Mac OS X kompatibel[120]

Tab. 13: Bewertung von Solr

[116] Vgl. Comundus (2011)

[117] Vgl. Maas, G (2012)

[118] Vgl. The Apache Software Foundation (2012l)

[119] Vgl. Miller, M u. a. (2012)

[120] Vgl. Rockstar template (2013)

4.3 Bewertung von Elasticsearch

In diesem Kapitel wird das Projekt Elasticsearch, welches von der gleichnamigen Firma entwickelt wird, anhand der gegebenen Kriterien bewertet. In Tab. 14 erfolgt zunächst die reine Projektbewertung.

	Elasticsearch
Produkt	
Produktalter	erster Release im Februar 2010[121]
Lizenzen	Apache 2.0 (freie, kostenlose Lizenz)[122]
Plattform u. Kompatibilität	Java-Entwicklung, damit plattformunabhängig; unterstützt Unix-Betriebssysteme (z.B. Debian) sowie Windows-Systeme[123]
Funktionalität (exemplarisch)	• voll funktionsfähiger Suchserver "out of the box" • Volltextsuche • Cloud-Unterstützung • Clustering • Indexreplikation • Gateway • Eigenschaften: schemafrei, dokumentorientiert, verteilt[124]
Modularität	• zahleiche Module als Code auf der offiziellen Homepage verfügbar, darunter aktuell über 40 Plugins (Januar 2013) • Plugins ermöglichen die Integration verschiedener Systeme und Analysewerkzeuge[125] • es existieren zusätzlich eine Reihe an Clients (z.B. für PHP, .NET, Ruby, Java, Python) und Front Ends (v.a. Web-Front-Ends) • der Code für alle Module, Clients und Front Ends

[121] Vgl. Elasticsearch (o. J.b)

[122] Vgl. Elasticsearch (o. J.a)

[123] Vgl. Elasticsearch (o. J.c)

[124] Vgl. Elasticsearch (o. J.a)

[125] Vgl. Elasticsearch (o. J.d)

	ist auf github.com veröffentlicht[126]
Releaseabstände	• 49 Releases innerhalb von 2 Jahren • z.T. mehrere Releases innerhalb eines Monats[127]
Dokumentation	
Benutzerhandbuch vorhanden	Tutorials auf der Homepage verfügbar: • Installationsanleitung • Konfigurationsanleitung • Anleitung zur Integration unterschiedlicher Systeme (z.B. Couch DB, JavaScript) • offizielle Videos auf Youtube[128]
Sprache der Dokumentation	ausschließlich englisch
Qualität der Dokumentation	• Anleitungen nur als Tutorials auf der Homepage verfügbar • keine Downloadmöglichkeiten als PDF-Datei o.Ä. • knappe Dokumentation mit Codedarstellung
Support	
Community und Ökosystem	Stand: Januar 2013 • Benutzerforum vorhanden • 7.549 Forumsthemen, viele aktuelle Themen • 74.326 Aufrufe im Forum • Benutzer können Fragen stellen, erfahrene Benutzer antworten[129] • Kontakt über Google Groups mit Benutzern und Entwicklern möglich • zudem Kontakt und gegenseitiger Austausch über Facebook und Twitter • französische und chinesische Community verfügbar • Kontaktaufnahme über IRC möglich[130]
direkter Ansprechpartner	• direkter Kontakt mit Benutzern via E-Mail möglich

[126] Vgl. Elasticsearch (o. J.e)

[127] Vgl. Elasticsearch (o. J.b)

[128] Vgl. Elasticsearch (o. J.f)

[129] Vgl. Banon, S. (o. J.)

[130] Vgl. Elasticsearch (o. J.g)

	• direkter Kontakt mit Entwicklern (E-Mail, Twitter, Forum, Blog, Facebook, IRC) möglich[131] • Fragen an Entwickler wurden adäquat beantwortet
Vorhandensein von FAQ	• kein „klassisches“ FAQ • zahlreiche Hilfeseiten auf der offiziellen Homepage und Videos • Quellcode auf github.com[132]
Marktakzeptanz	
Anzahl der Stakeholder	bekannte, kommerzielle Nutzer (laut Homepage): • Mozilla Foundation • SoundCloud • StumbleUpon • Sony Computer Entertainment • Infochimps • Desk.com • Ataxo Social Insider • Bazzarvoice • Klout • Sonian Inc. • IGN • InSTEDD (zwölf Nutzer aufgezählt)[133] bekannte Sponsoren (laut Homepage): • speedchilli IT solutions • JetBrains[134]
Anzahl der Downloads	keine Informationen vorhanden
Traffic Rank	Stand: Januar 2013 elasticsearch.org: • globaler Rang: 52.177 • Rang in Deutschland: 55.799[135]

Tab. 14: Bewertung des Elasticsearch-Projekts

[131] Vgl. Elasticsearch (o. J.g)

[132] Vgl. GitHub Inc. (2013)

[133] Vgl. Elasticsearch (o. J.h)

[134] Vgl. Elasticsearch (o. J.g)

[135] Vgl. Alexa Internet, Inc. (2013b)

Als nächstes wird der Fokus auf Elasticsearch selbst als Suchmaschine gelegt (Tab. 15).

	Elasticsearch
Kriterium	**Beurteilung**
grundsätzliche Funktionen	
Codierung u. Sprachunterstützung	unterstützt Ein- und Ausgaben in Unicode[136]
Crawler	besitzt Crawler out-of-the-box[137]
Dateiunterstützung	über das Toolkit „Apache Tika“ lassen sich diverse Dateien indexieren: • Office Document Format • PDF • HTML, XML • OpenDocument Format • Rich Text Format • Audioformate (MP3, • MIDI) • Videoformate (Flash) • Bildformate (JPEG) • JAVA-Klassen-Dateien[138]
Cloud Unterstützung	Cloud-Unterstützung out-of-the-box[139]
Suchalgorithmus	
Volltextsuche	Volltextsuche unterstützt
Synonyme u. Korrektur	große Anzahl an Modulen u.a. für: • Synonyme • Stoppwörter • Groß- / Kleinschreibung • zusammengesetzte Wörter Listen mit Synonymen u.Ä. müssen selbst gepflegt werden[140]

[136] Vgl. o. V. (2012a)

[137] Vgl. Elasticsearch (o. J.a)

[138] Vgl. Elasticsearch (o. J.i)

[139] Vgl. Elasticsearch (o. J.a)

[140] Vgl. Elasticsearch (o. J.j)

Ergebnisdarstellung	
Cached Results u. Hintergrundinformationen	große Anzahl an Analyse-Tools liefert Hintergrundinformationen für unterschiedliche Bereiche, darunter auch Cached Results[141]
Ausgabe der Suchergebnisse	unterschiedliche Front Ends implementierbar, darunter Webinterfaces[142]
Skalierbarkeit	
Skalierbarkeit	• verteilte Suchmaschine • hohe Skalierbarkeit durch Indexreplikation möglich • dadurch Anfragelastverteilung und Ausfallsicherheit
Möglichkeiten der Anbindung	
LDAP-Anbindung / Umsetzung bestehender Zugriffsrechte	keine eingebaute Funktionalitäten zur LDAP-Authentifizierung
Anbindung an Datenquellen	Keine Anbindungsmöglichkeiten „out-of-the-box“, aber Module für die Anbindung an • File Server • Notes Datenbanken • webbasierte Dokumente derzeit KEINE Anbindung an Wikis möglich[143]
Administration	
Wartungs- und Betreuungsmöglichkeiten	zahlreiche Monitoring- und Reporting-Werkzeuge als Erweiterungen verfügbar[144]
Berechtigungsverwaltung	nicht implementiert
Programmiersprache	• programmiert in JAVA • JSON / REST API
nichtfunktionale Anforderungen	
Performance	• Möglichkeiten der Echtzeitsuche und –indizierung • keine Informationen über allgemeine Laufzeiten

[141] Vgl. Elasticsearch (o. J.j)

[142] Vgl. Elasticsearch (o. J.k)

[143] Vgl. Pilato, D. (2012)

[144] Vgl. Elasticsearch (o. J.d)

intuitive Bedienung	• zur allgemeinen Bedienung: Weboberflächen verfügbar • zur Erweiterbarkeit: JSON / REST[145]
einfache Installation / Konfiguration	Anleitung / Tutorials für Installation und Konfiguration sind auf der Homepage (mit Codeausschnitten) verfügbar[146]
Kompatibilität	Kompatibilität mit Unix- und Windows-Betriebssystemen[147]

Tab. 15: Bewertung von Elasticsearch

4.4 Bewertung von HSearch

In diesem Kapitel wird das Projekt HSearch der Firma Bizosys anhand der gegebenen Kriterien bewertet. In Tab. 16 erfolgt zunächst die reine Projektbewertung.

	HSearch
Produkt	
Produktalter	erster Release seit 2010 verfügbar.[148]
Lizenzen	Apache 2.0 (freie, kostenlose Lizenz)[149]
Plattform u. Kompatibilität	Java-Entwicklung, damit plattformunabhängig; unterstützt Unix-Betriebssysteme sowie Windows-Systeme[150]
Funktionalität (exemplarisch)	• voll funktionsfähiger Suchserver "out of the box" • Volltextsuche • unscharfe Suche • Unterstützung strukturierter, semi-strukturierter und unstrukturierter Datenquellen • Berechtigungsverwaltung • Monitoring-Tools • verteilte Suchmaschine

[145] Vgl. Elasticsearch (o. J.a)

[146] Vgl. Elasticsearch (o. J.c)

[147] Vgl. Elasticsearch (o. J.a)

[148] Vgl. Bizosys Technologies Pvt Ltd. (2010a)

[149] Vgl. ebenda

[150] Vgl. ebenda

	• Ranking-Funktionen[151]
Modularität	Erweiterbarkeit: • benutzerdefinierte Indexierung und Sequenzierung • benutzerdefinierter Anschluss an Legacy-Systeme • benutzerdefinierte Verarbeitungsschritte[152]
Releaseabstände	• Version 0.89 (07.12.2010) • Version 0.90 (19.01.2011) • Version 0.94 (16.02.2012) Releases ca. im 1-Jahresrhythmus[153]
Dokumentation	
Benutzerhandbuch vorhanden	• Installationsdokument vorhanden[154] • Link zur Dokumentation führt ins Leere[155]
Sprache der Dokumentation	ausschließlich englisch
Qualität der Dokumentation	• Installationshandbuch in PDF-Format • Installationshandbuch besteht nur aus Code-Beispielen • Link zur Dokumentation führt ins Leere
Support	
Community und Ökosystem	Stand: Januar 2013 • zwei Projektforen vorhanden, jedoch letzter Eintrag von Juli 2011 • vier Projekt-Tracker für Bugs, Support Requests, Patches und Feature Requests • Letzter Bug-Eintrag ist vom 26.11.2011 • Patches wurden bisher nicht eingestellt[156]
direkter Ansprechpartner	• Kontakt zu Sponsor BIZOSYS Technologies via E-Mail oder soziale Netzwerke • keine Angaben über Kontaktdaten zu Entwicklern

[151] Vgl. Bizosys Technologies Pvt Ltd. (2010b)

[152] Vgl. ebenda

[153] Vgl. Bizosys Technologies Pvt Ltd. (2011)

[154] Vgl. ebenda

[155] Vgl. Bizosys Technologies Pvt Ltd. (2010a)

[156] Vgl. SourceForge (2013)

Vorhandensein von FAQ	kein FAQ vorhanden
Marktakzeptanz	
Anzahl der Stakeholder	• HSearch hat das Unternehmen BIZOSYS Technologies als Sponsor • keine Informationen über Nutzer von HSearch verfügbar • HSearch basiert auf Hadoop • bekannte Nutzer von Hadoop sind z.B. Facebook, AOL, IBM und Yahoo[157]
Anzahl der Downloads	keine Informationen vorhanden
Traffic Rank	Stand: Januar 2013 http://www.hadoopsearch.net/: • globaler Rang: 17.500.584 • keine regionalen Daten feststellbar[158] Traffic Rank für http://bizosyshsearch.sourceforge.net/ nicht feststellbar.

Tab. 16: Bewertung des HSearch-Projekts

Als nächstes wird der Fokus auf HSearch selbst als Suchmaschine gelegt (Tab. 17).

	HSearch
Kriterium	**Beurteilung**
grundsätzliche Funktionen	
Codierung u. Sprachunterstützung	HSearch unterstützt die deutschen Sonderzeichen (UTF-8)[159]
Crawler	besitzt Crawler out-of-the-box[160]
Dateiunterstützung	unterstützt alle gängigen Dateiformate[161]
Cloud Unterstützung	• Anwendung ist webfähig, soziale Netzwerke können angebunden werden[162]

[157] Vgl. o. V. (2012b)

[158] Vgl. Alexa Internet, Inc. (2013c)

[159] Vgl. Bizosys Technologies Pvt Ltd. (2010b)

[160] Vgl. ebenda

[161] Vgl. ebenda

	• Cloud-Unterstützung nicht genauer spezifiziert
Suchalgorithmus	
Volltextsuche	Volltextsuche unterstützt[163]
Synonyme u. Korrektur	Begriffsklärung vorhanden, unscharfe Suche eingebaut[164]
Ergebnisdarstellung	
Cached Results u. Hintergrundinformationen	• Vorschauansichten können in mehreren Dateiformaten angezeigt werden • Werkzeuge zu Hintergrundinformationen (z.B. Ranking, gefundene und nicht gefundene Suchergebnisse)[165]
Ausgabe der Suchergebnisse	Inhalte lassen sich vielfältig ausgeben: • Listenansicht • tabellarisch • als dynamische Portlets • Ausgabe in XML[166]
Skalierbarkeit	
Skalierbarkeit	• verteilte Suchmaschine • gute Skalierbarkeit u.a. durch automatische Replikation und Indexzerlegung möglich[167]
Möglichkeiten der Anbindung	
LDAP-Anbindung / Umsetzung bestehender Zugriffsrechte	Authentifizierung über LDAP ist möglich[168]
Anbindung an Datenquellen	• relationale und dokumentorientierte Datenbanken • Dateisysteme • Mail-Server • RSS-Feeds • Webseiten[169]

[162] Vgl. Bizosys Technologies Pvt Ltd. (2010a)

[163] Vgl. Bizosys Technologies Pvt Ltd. (2010b)

[164] Vgl. ebenda

[165] Vgl. ebenda

[166] Vgl. ebenda

[167] Vgl. Bizosys Technologies Pvt Ltd. (2010a)

[168] Vgl. Bizosys Technologies Pvt Ltd. (2010b)

Administration	
Wartungs- und Betreuungs-möglichkeiten	• Reportingwerkzeuge • Monitoring-Tools • Bugtracker • Konfigurationsmöglichkeiten (Datenquellen, Indexierung, Spracheinstellungen, Ausgabe, Zugriffssteuerung)[170]
Berechtigungsverwaltung	Berechtigungsverwaltung über UserID, Rolle oder Team/Gruppe realisiert[171]
Programmiersprache	• programmiert in JAVA • JSON / REST API
nichtfunktionale Anforderungen	
Performance	• Möglichkeiten zur Echtzeitsuche • Tests in den USA und Indien lieferten gute Performance-Ergebnisse (siehe http://bizosyshsearch.sourceforge.net/)
intuitive Bedienung	• Link zur Dokumentation führt ins Leere • zur allgemeinen Bedienung: Ausgabe über verschiedene Formate möglich • zur Erweiterbarkeit: JSON / REST
einfache Installation / Konfiguration	Installationsdokumentation vorhanden, jedoch nur knapp beschrieben / hauptsächlich nur Codedarstellung
Kompatibilität	Kompatibilität mit Unix- und Windows-Betriebssystemen

Tab. 17: Bewertung von HSearch

[169] Vgl. Bizosys Technologies Pvt Ltd. (2010a)

[170] Vgl. Bizosys Technologies Pvt Ltd. (2010b)

[171] Vgl. ebenda

5 Gegenüberstellung und Gesamtbewertung der Open Source Suchmaschinen

Nachdem die Meta-Suchmaschinen und deren Projekte einzeln betrachtet und bewertet wurden, erfolgt in diesem Kapitel eine bewertende Gegenüberstellung. Der erste Schritt ist die Bewertung der Open-Source-Projekte. Das Bewertungsverfahren wurde in Kapitel 2.1 beschrieben. In Tab. 18 findet sich die ausgefüllte Bewertungsmatrix für den Reifegrad aus Kapitel 2.1.

Kriterium	Gewichtung	Suchmaschinen			
		Lucene	Solr	Elastic-search	HSearch
Produktalter	**20%**	4	3	2	1
Lizenzen	**15%**	1	1	1	1
Plattform u. Kompatibilität	**15%**	2	2	1	1
Funktionalität	**20%**	1	3	2	2
Modularität	**15%**	4	3	3	1
Releaseabstände	**15%**	3	2	4	1
SUMME Produkt	**50%**	2,5	2,4	2,15	1,2
Benutzerhandbuch vorhanden	**75%**	3	3	2	1
Sprache der Dokumentation	**5%**	0	0	0	0
Qualität der Dokumentation	**20%**	2	2	1	0
SUMME Dokumentation	**30%**	2,65	2,65	1,7	0,75
Community und Ökosystem	**50%**	4	3	2	1
direkter Ansprechpartner	**20%**	3	3	4	1
Vorhandensein von FAQ	**30%**	4	4	2	0
SUMME Support	**20%**	3,8	3,3	2,4	0,7
SUMME	**100%**	**2,81**	**2,66**	**2,07**	**0,97**

Tab. 18: Reifegradermittlung der Suchmaschinenprojekte

Das Projekt von Lucene erreicht mit 2,81 Punkten den höchsten Wert. Dicht dahinter steht, wohl auch aufgrund der Ähnlichkeit der Projekte, Solr mit 2,66 Punkten. Einen noch zufriedenstellenden Wert erzielt Elasticsearch (2,07) während Hsearch mit 0,97 Punkten mit Abstand auf dem letzten Platz landet. Insgesamt konnten bei diesem Bewertungsverfahren 4,00

Punkte erreicht werden. Die Projekte Apache Lucene und Apache Solr schneiden dabei in allen drei Kategorien ähnlich gut ab. Insbesondere in der Kategorie Support können die Projekte mit 3,8 (Lucene) und 3,3 (Solr) Punkten überzeugen. Elasticsearch kommt in allen drei Kategorien nicht an Lucene und Solr heran und schwächelt insbesondere in der Kategorie Dokumentation mit 1,7 Punkten. Noch geringere Werte erzielt HSearch und scheint v.a. in den Bereichen Dokumentation und Support mit weniger als 1,00 Punkten noch sehr unausgereift zu sein.

Es lässt sich zudem feststellen, dass alle drei Suchmaschinenprojekte unter Apache 2.0 lizensiert sind, damit vollkommen kostenlos sind und ohne Einschränkungen verwendet werden können. Ebenfalls fällt auf, dass bei keinem der Projekte eine deutschsprachige Dokumentation vorhanden ist.

Den zweiten Teil der Projektbewertung bildet die Analyse der Marktakzeptanz, welche aus Tab. 19 hervorgeht.

Kriterium	Gewichtung	Suchmaschinen			
		Lucene	Solr	Elastic-search	HSearch
Anzahl der Stakeholder	**50%**	4	4	2	1
Anzahl der Downloads	**25%**	0	0	0	0
Traffic Rank	**25%**	3	3	2	1
SUMME	**100%**	**2,75**	**2,75**	**1,50**	**0,75**

Tab. 19: Ermittlung der Marktakzeptanz der Suchmaschinenprojekte

Für keine der Suchmaschinen konnten Informationen über Downloadzahlen gefunden werden. Daher muss sich der Vergleich auf die Kriterien „Anzahl der Stakeholder“ und „Traffic Rank“ beschränken. Betrachtet man die Ergebnisse, findet sich auch in dieser Kategorie ein ähnliches Bild wie schon beim Reifegrad. Lucene und Solr erreichen mit 2,75 von 4,00 möglichen Punkten die gleichen Ergebnisse. Dahinter landet Elasticsearch mit 1,50 Punkten, wobei es wieder bei keinem der beiden Kriterien an Lucene oder Solr herankommt. Auch bei der Marktakzeptanz kann HSearch nicht überzeugen und landet mit 0,75 Punkten weit hinter den beiden erstplatzierten Projekten.

Der Zusammenhang zwischen Reifegrad und Marktakzeptanz der Suchmaschinenprojekte wird nun in der Vier-Felder-Matrix von Abb. 12 visualisiert. Hier sind die einzelnen Projekte gemäß ihrer erreichten Punktzahlen relativ zueinander eingeordnet.

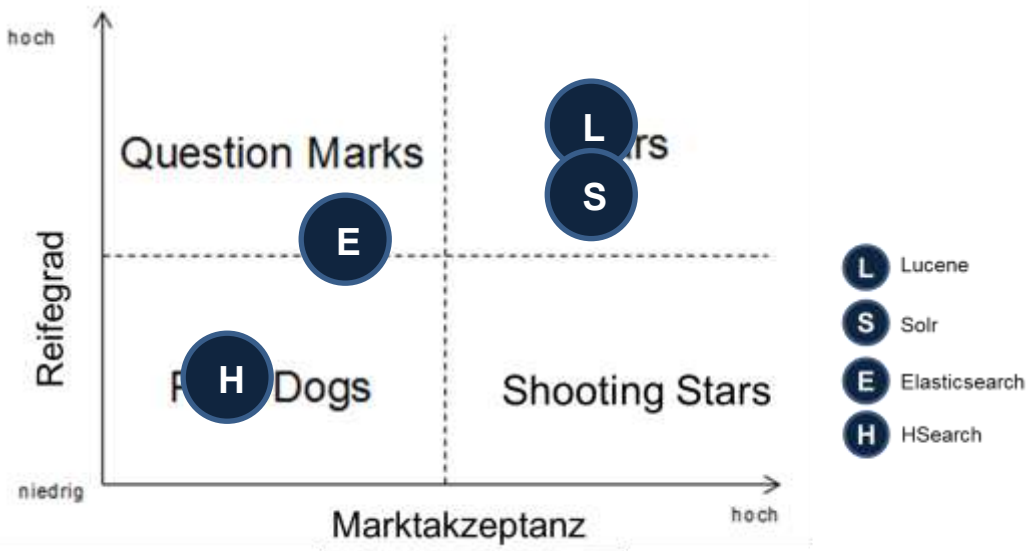


Abb. 12: Vier-Felder-Matrix für die Bewertung der Open Source Projekte mit Einordnung

Wie schon aufgrund seines Alters und hohen Bekanntheitsgrads zu erwarten war, erzielt das Projekt Apache Lucene die besten Werte sowohl beim Reifegrad als auch bei der Marktakzeptanz. Damit ist dieses klar im Feld der „Stars“ einzuordnen. Ganz ähnlich schneidet Apache Solr ab und findet sich, aufgrund eines etwas geringeren Reifegrads, knapp unterhalb von Lucene und damit ebenfalls im Bereich der Stars. Diese beiden Open Source Projekte scheinen recht ausgereift zu sein und finden auch bei einer größeren Anzahl an Anwendern und Unterstützern anklang. Rein aus der Sicht der Projektbewertung sind diese also zu bevorzugen.

Mit einer vergleichsweise deutlich geringeren Markakzeptanz sowie einem leicht niedrigerem Reifegrad kann Elasticsearch auf der Grenze zwischen „Poor Dogs“ und „Question Marks“ eingeordnet werden. Die Tendenz geht jedoch eher Richtung „Question Marks“. Das Projekt Elasticsearch ist noch sehr jung, besitzt aber dennoch schon jetzt eine gewisse Reife, was Produkt und Support angeht. Elasticsearch hat damit durchaus Potenzial, in Zukunft weiterhin zu wachsen und noch mehr Anwender und Entwickler für sich zu gewinnen. Dennoch ist eine genaue Prognose zu diesem frühen Zeitpunkt nicht möglich, sodass Elasticsearch zu Recht als unsicheres aber vielversprechendes Projekt als „Question Mark“ eingestuft wird.

Schließlich bleibt noch HSearch von Bizosys, das bei keinem der Kriterien gute Werte erreichen konnte und deshalb dem Feld der „Poor Dogs“ zugeordnet wird. Zwar besitzt es mit Hadoop ein sehr großes und populäres Framework als Basis, konnte aber als Suchmaschinenprojekt weder einen adäquaten Nutzerkreis finden noch eine ausreichende Reife vorweisen. Hinzu kommt, dass Informationen zu diesem Projekt nur spärlich verfügbar sind. Als Projekt ist HSearch damit für den Einsatz im Unternehmensumfeld kritisch zu betrachten.

Als nächstes werden die Meta-Suchmaschinen selbst mit ihrem Funktionsumfang und ihren nichtfunktionalen Eigenschaften gegenübergestellt und bewertet (Tab 20).

Kriterium	Gewichtung	Bewertung			
		Lucene	Solr	Elastic-search	HSearch
grundsätzliche Funktionen					
Codierung u. Sprachunterstützung	PFLICHT	JA	JA	JA	JA
Crawler	8%	0	1	1	1
Dateiunterstützung	8%	1	1	1	1
Cloud Unterstützung	1%	0	1	1	0
Suchalgorithmus					
Volltextsuche	PFLICHT	JA	JA	JA	JA
Synonyme u. Korrektur	6%	1	4	2	3
Ergebnisdarstellung					
Cached Results u. Hintergrundinformationen	3%	1	4	3	2
Ausgabe der Suchergebnisse	9%	1	1	1	1
Skalierbarkeit					
Skalierbarkeit	10%	0	2	3	2
Möglichkeiten der Anbindung					
Umsetzung bestehender Zugriffsrechte	PFLICHT	NEIN	JA	NEIN	JA
Anbindung an Datenquellen	20%	0	4	2	2
Administration					
Wartungs- und Betreuungsmöglichkeiten	5%	1	2	1	2
Berechtigungsverwaltung	2%	1	3	0	3
Programmiersprache	1%	1	4	2	2
nichtfunktionale Anforderungen					
Performance	6%	2	1	1	1
intuitive Bedienung	7%	1	1	1	1
einfache Installation / Konfiguration	8%	2	1	1	1
Kompatibilität	6%	4	3	2	2
SUMME	**100%**	**0,93**	**2,21**	**1,57**	**1,60**

Tab. 20: Bewertende Gegenüberstellung der Meta-Suchmaschinen

Betrachtet man zunächst die wichtigen Pflichtanforderungen, fällt auf, dass zwei Suchmaschinen diese nicht komplett erfüllen können. Weder Lucene noch Elasticsearch bieten Funktionalitäten, mit denen eine LDAP-Anbindung möglich ist. Damit können bestehende Berechtigungskonzepte nicht übernommen werden, was ein K.O.-Kriterium darstellt, weswegen diese beiden Suchmaschinen nicht in die engere Auswahl gezogen werden können.
Unabhängig von den Pflichtkriterien erreicht die Suchmaschine Solr mit 2,21 Punkten deutlich den ersten Platz. Dahinter finden sich mit geringem Abstand zueinander die Suchmaschinen HSearch (1,60) und Elasticsearch (1,57). Mit einem ebenso deutlichen Abstand landet die Programmbibliothek Lucene auf dem letzten Platz (0,93 Punkte).
Solr punktet v.a. aufgrund der Tatsache, dass viele Funktionen ohne die Notwendigkeit einer Erweiterung mit dem System mitgeliefert werden (z.B. Synonyme und unscharfe Suche, Werkzeuge zur Anzeige von Hintergrundinformationen bzw. Cached Results). Zusätzlich zeichnet sich die Suchmaschine durch eine breite Kompatibilität, Skalierbarkeit, zahlreiche Anbindungsmöglichkeiten und die Möglichkeit der lokalen Berechtigungsverwaltung aus. Zudem kann Solr in einer großen Anzahl gängiger Programmiersprachen implementiert werden.
Ähnlich wie Solr bietet auch HSearch eine recht große Auswahl an Funktionalitäten, ohne dass diese nachgerüstet werden müssen. Hier sind vor allem die Tools zu nennen, mit denen eine Überwachung bzw. ein Reporting möglich ist, sowie die unscharfe Suche. Weitere Vorzüge sind die gute Skalierbarkeit, die Unterstützung von vielen Datenquellen und die eingebaute Benutzer- und Zugriffsverwaltung. Abzüge gab es hauptsächlich für die fehlende Anbindungsmöglichkeit für Wiki-Systeme sowie die fehlende Cloud-Unterstützung.
Die Stärken von Elasticsearch liegen in dessen guten Skalierbarkeit und der hervorragenden Modularität. Letztere ermöglicht es, dass Funktionen, die standardmäßig nicht im Funktionsumfang enthalten sind, ohne weiteres über Add-Ons nachgerüstet werden können. Betrachtet man diese von der Community entwickelten Add-Ons, findet man eine breite Palette an Werkzeugen, z.B. Tools zur Darstellung von Hintergrundinformationen zur Suche, Tools zur Darstellung von semantisch und syntaktisch ähnlich geschriebenen Suchwörtern oder Programme zur Performanceüberwachung. Die Schwächen liegen bei den Anbindungsmöglichkeiten von Elasticsearch. So können die Wiki-Systeme gar nicht und die übrigen Datenquellen nur über die Integration von Modulen angebunden werden. Als größte Schwäche ist die fehlende Möglichkeit der LDAP-Anbindung und damit der Integration von Berechtigungsrichtlinien und deren Verwaltung zu nennen. Aus diesem Grund kommt Elasticsearch als Suchmaschine in dem hier beschriebenen Fall nicht in Frage.
Lucene ist als Programmbibliothek besonders einfach zu implementieren und mit sehr vielen Systemen kompatibel. Das wichtigste Argument für Lucene ist dessen Modularität. Apache umfasst viele Projekte, die auf Lucene aufbauen bzw. in Lucene integriert werden können

(siehe hierzu auch Kapitel 4.1). Solr und Elasticsearch sind Beispiele für Projekte, die Lucene als Basis verwenden. Daher sollte Lucene nicht als alleinstehende Suchmaschine betrachtet werden, sondern vielmehr als Bibliothek, auf der je nach Bedarf aufgebaut werden kann. Damit werden auch die Nachteile von Lucene deutlich. Komponenten, wie z.B. Crawler, Werkzeuge zur Analyse, Tools zur Informationsanzeige oder die Cloud-Unterstützung werden nicht unterstützt oder sind nur über Erweiterungen verfügbar. Ein Zugriff auf die betrachteten Datenquellen ist ebenso wenig möglich wie die Anbindung an Verzeichnisdienste, wie z.B. eDirectory. Folglich ist Lucene für den in dieser Arbeit betrachteten Fall nicht zu empfehlen.

Von den in dieser Arbeit vorgestellten Meta-Suchmaschinen wird einzig Solr den gestellten Anforderungen in einem akzeptablen Ausmaß gerecht. Solr erfüllt nicht nur die Pflichtkriterien sondern auch viele der zusätzlichen Bewertungspunkte und kann alle geforderten Datenquellen (Wiki-Systeme, Dateien auf Netzlaufwerken, Notes-Datenbanken, webbasierte Dokumente) indexieren. Auch das Open Source Projekt, unter dem Solr entwickelt wird, erscheint aufgrund eines akzeptablen Reifegrads und einer soliden Marktakzeptanz durchaus vielversprechend. Die rege Aktivität der Community, das gut entwickelte Ökosystem und Apache als unterstützende Stiftung sind weitere Argumente, die für die Zukunft des Solr-Projekts sprechen. Von den vier analysierten Enterprise Suchmaschinen ist Solr als Suchmaschinenlösung bei den in dieser Arbeit gegebenen Anforderungen zu empfehlen.

6 Zusammenfassung und Ausblick

Das Projekt hat das Ziel verfolgt, die Einsetzbarkeit von Meta-Suchmaschinen zur Abdeckung verschiedener Bereiche zu evaluieren. Das Augenmerk lag dabei ausschließlich auf Open Source Projekten. Als Basis zur Evaluierung diente ein Kriterienkatalog mit Anforderungen, die für ein Unternehmen gelten können.

Das oberste Ziel war es, eine Volltext-Suchmaschinenlösung zu finden, mit der möglichst viele Datenquellen angebunden werden können. Zudem musste ein vorhandenes Berechtigungskonzept auch innerhalb des Suchmaschinenkonzepts wiederverwendet werden können sowie eine Unterstützung deutscher Sonderzeichen gewährleistet sein.

Diese Anforderungen, zusammen mit anderen Bewertungskriterien, bilden die Grundlage eines Bewertungskonzepts, mit dem speziell die Funktionalitäten und Eigenschaften der Suchmaschinen bewertet wurden. Zusätzlich wurde nach dem Vorbild der Vier-Felder-Matrix der Boston Consulting Group ein Konzept zur Bewertung und Gegenüberstellung von Reifegrad und Marktakzeptanz definiert.

Im Zentrum des Projekts standen die vier Open Source Projekte Lucene, Solr, Elasticsearch und HSearch. Aus der reinen Projektbewertung ging hervor, dass die beiden von der Apache Stiftung geförderten Projekte Lucene und Solr eine akzeptable Marktakzeptanz und einen guten Reifegrad besitzen. Dagegen erschienen die Projekte Elasticsearch und insbesondere HSearch noch unausgereift und bieten noch eine Menge Verbesserungspotential. Die Bewertung der Produkte selbst ergab, dass Lucene und Elasticsearch nicht alle Pflichtkriterien erfüllen und damit nicht für den Einsatz in diesem Fallbeispiel geeignet sind. HSearch erwies sich dabei als unausgereiftes Produkt, bei dem v.a. aufgrund mangelnder Unterstützung durch die Community wichtige Anforderungen nicht in ausreichendem Maße erfüllt wurden. Solr konnte sich bei nahezu allen Kriterien durchsetzen und geht damit als beste Suchmaschine aus dem Projekt hervor.

Der Markt bietet eine große Auswahl an Open Source Suchmaschinen. Damit ist nicht auszuschließen, dass andere Open Source Lösungen existieren, die die Anforderungen besser erfüllen als die vier in diesem Projekt betrachteten Systeme. Außerdem kann sich die Gewichtung der einzelnen Bewertungskriterien aus Unternehmenssicht im Laufe der Zeit ändern und es können neue Anforderungen hinzukommen. Ebenso befinden sich die hier analysierten Projekte teilweise noch in einem frühen Entwicklungsprozess, sodass der weitere Entwicklungsverlauf weiterhin beobachtet werden sollte. Nichtsdestotrotz zeigt diese Arbeit, dass Solr die gegebenen Anforderungen am besten erfüllt.

Anhang

Anhangverzeichnis

Anhang 1: HSearch Datenblatt[172]

Capability	Feature	Solution
User Experience	Automatic transformation of search results and Multiple display styles	Search results displayed as • List view with snippet • Sort able grid view with filters • Dynamic Portlets • Result rendering in configurable XML templates
	Special searches	• Form based search on specific field and content types
	Discovery aids/ Guided navigation	• Natural language clusters • Thesaurus based bread crumb trails • Meta data based facetted navigation • Tag clouds
	Content spotlighting	• Pushed content by editorial team
	Previews	• Tabular data extraction from HTML pages • Document cover page extraction such as MS Word, MS PowerPoint, Adobe PDF, etc • Multiple page document browsing
	Automatic transformation of data	• Interactive, rich views using UI widgets, such as tables, charts, maps, snapshots, calendar, etc.
Integration	Content Crawl	Bizosys Search Framework indexes content from various source types including: • Relational databases • File systems • Mail servers • RSS feeds • Web pages • XML files • Tab separated files • Custom crawlers *(not in built)*
	Content Push to Bizosys Search Framework	Bizosys Search Framework allows external systems to push content in real time using - • REST based APIs • Java based APIs
	Actions	Bizosys Search Framework auto integrates search results to external systems while applying user ACLs via • Web services • REST

[172] Enthalten in: Bizosys Technologies Pvt Ltd. (2010b)

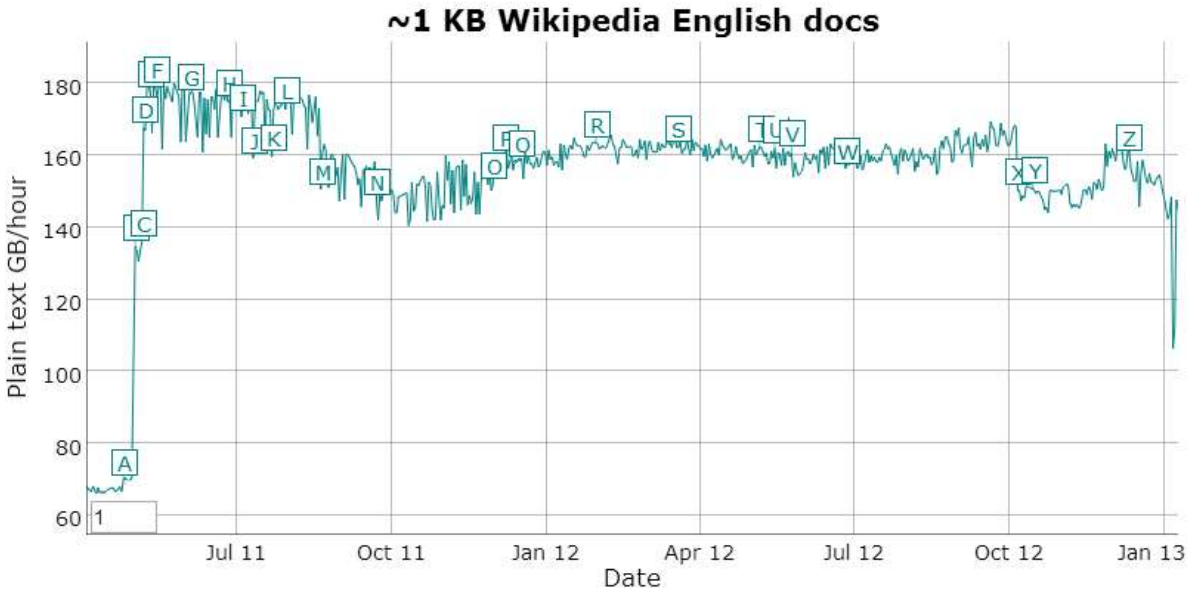
Security	Access Control	Document level Access Control based on: • User ID • Role • Team/Group • Organization Department
	Enterprise signon	• Bizosys Search Framework allows users authenticated from enterprise LDAP • User access via custom security interfaces
	Secure crawling	• Basic LDAP, NTLM based authentication • PKI authentication with X.509 • Kerberos • Windows Integrated Authentication • Web proxy authentication
	Index Encryption	• Configurable encryption of sensitive content
	Anti-Virus	• Configurable encryption
Extensibility	Plug-in Architecture	• Custom indexing step and sequencing • Custom connector to legacy systems • Custom result processing step
Scalability	Single machine	• configurable leveraging of multiple hard disks
	Cluster environment	• Parallel indexing • Parallel searching • On demand machine acquisition and release
Fail-safe	Machine failure	• configurable replication level to address machine failure (results in only that machine being removed and not the entire rack/row in the cluster)
Indexing	Language (English)	• Stemming, normalization, stop words & phrases, synonyms
	File Extensions	• 267 file extensions including HTML, PDF, MS Office, and many more
	Index compression	• Compressed index to save disk space

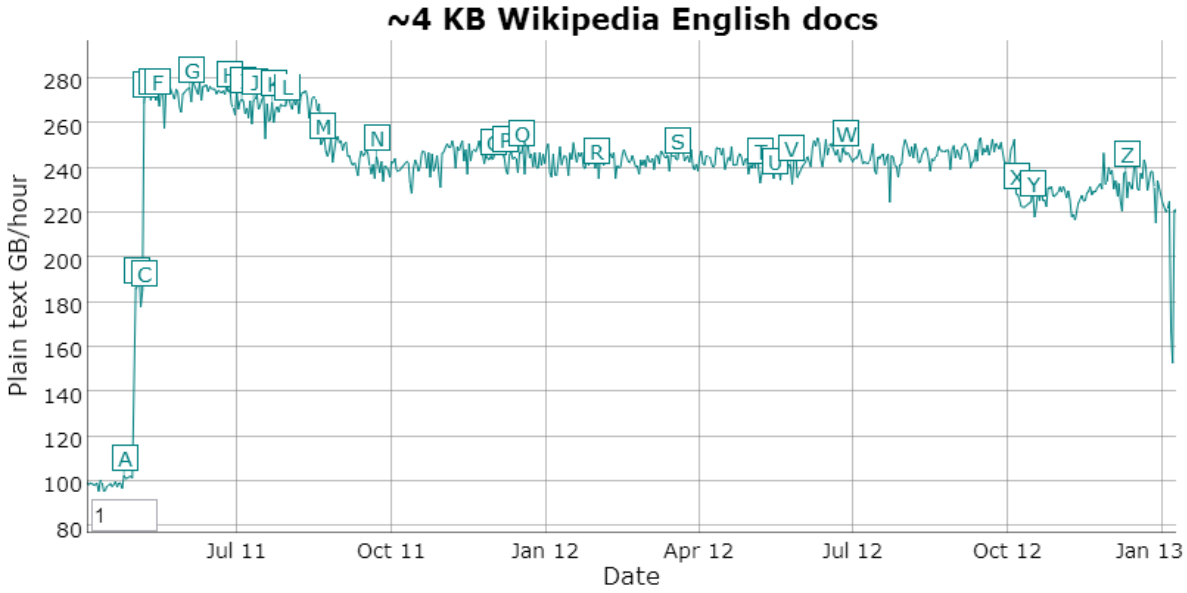
	Real time updating	• Content and ACLs updated in real time
	Document boosting	• Configurable document boosting to aid relevancy
Ranking	Weightage based objective computation	Over nine dimensions used to compute relevancy objectively including: • Latest content • Location proximity • Machine proximity • Source importance • Author importance • Term importance • Phrase location
	Weightage based subjective computation	• Self learning - User click based ranking to map search term to document
	Multiple ranking policies	• Dynamic weightage assignment to different sources or content types. For e.g. ranking in news can provide higher weightage based on content creation time. Similarly, in case of researching a topic, the source or author parameter could be assigned higher weightage relative to other parameters
Disambiguation	'Did you mean'	• WWW based extraction of meaning to find alternative terms • Fuzzy search
Knowledge extraction	Thesaurus	• Automatic extraction of hierarchical and thematic relationships across multiple web sites to create a unified thesaurus like representation
	Tacit knowledge	• History of users selection of documents tagged to the search terms represents user intent and memories; reflected in the Index as tacit knowledge
	Business entity extraction	• Extraction of business entities from unstructured documents and dynamically tagging them to structured records/data to exploit a host of related services based on user permissions

Administration	Configuration	• Configure SoA - web services, REST • Ranking policies • Template management • Access setup • Language setup • Data source setup
	Execution	• Index scheduling • Index monitoring
	Fine tuning & reporting	• Top 100 queries by role • Missed searches • Slow responses • Dictionary and thesaurus tuning

Anhang 2: Lucene Benchmarking[173]

Jede Nacht wird dieses Benchmarking automatisch ausgeführt. Dabei wird immer auf einen gespeicherten Testindex zugegriffen. Dabei handelt es sich um einen XML-Export der englischen Wikipedia-Internetseite, der am 15.01.2011 erstellt wurde. Die Ergebnisse sind in den folgenden Abbildungen dargestellt.





Verwendetes System:

- Betriebssystem: Linux
- Java Version: 1.7
- CPU: 2 Intel Xeon X 5680 mit 4,0 GHz (insgesamt 24 Kerne)

[173] Vgl. McCandless, M. (2013)

Quellenverzeichnisse

Literaturverzeichnis

Arnold, S. E. (2011):	Redefining Search, Open Source Search: Revolution or Evolution?, in: Information Today, Januar 2011, S.26
Bauer, G. (2009):	Architekturen für Web-Anwendungen, Eine praxisbezogene Konstruktions-Systematik, Wiesbaden: Vieweg+Teubner
Golembowska, A. u. a. (2012):	Entwicklung eines Modells zur Bewertung von Open Source Produkten hinsichtlich eines produktiven Einsatzes, Seminararbeit, Stuttgart: o. Verl.
Kammerer T. / Mandl, P. / Baumgärtner, R. (2011):	Pfadsucher, Suchfunktionen auf der Basis von Lucene, in: iX 12/2011, S.131-134
Karich, P. (2012):	Immer flexibel, Apache Solr bekommt Konkurrenz: ElasticSearch, in: iX 3/2012, S.67-71
Köhler, M./ Arndt, H./ Fetzer, T (2008):	Recht des Internet, 6.Auflage, Heidelberg / München / Landsberg / Berlin: Verlagsgruppe Hüthig-Jehle-Rehm
Konrad, R. (2009):	Volltextsuche im Kontext relationaler Datenbanken am Beispiel einer systeminternen DBMS- Komponente von MySQL, München: GRIN Verlag GmbH
Schwichtenberg, H. (2001):	COM-Komponenten-Handbuch, Systemprogrammierung und Scripting mit COM-Komponenten, München: Addison-Wesley
White, M. (2012):	Enterprise Search, Sebastopol (USA): O'Reilly Media, Inc.
Wöhe, G. / Döring, U. (2010):	Einführung in die Allgemeine Betriebswirtschaftslehre, 24. Auflage, München: Vahlen
Ziegler, C. (2006):	Suche nach Suche, Apaches Lucene: eigene Suche und Indizierung, in: iX 6/2006, S.120-123
Ziesche, P. / Arinir, D. (2010):	Java: Nebenläufige & verteilte Programmierung: Konzepte, UML-Modellierung, Realisierung und Testen in Java, 2.Auflage, Witten: W3L GmbH

Verzeichnis der Internetquellen

o. V. (o. J.a):	Introducing JSON, http://www.json.org/, Abruf: 05.12.2012
o. V. (o. J.b):	Lucene, http://lucene.472066.n3.nabble.com/, Abruf 18.01.2013
o. V. (o. J.c):	Lucene Index Monitor, http://limo.sourceforge.net/, Abruf 18.01.2013
o. V. (o. J.d):	Solr, http://lucene.472066.n3.nabble.com/Solr-f472067.html, Abruf 18.01.2013
o. V. (o. J.e):	Indexing files like doc, pdf – Solr and Tika integration, http://solr.pl/en/2011/04/04/indexing-files-like-doc-pdf-solr-and-tika-integration/, Abruf 18.01.2013
o. V. (o. J.f):	Anleitung: Installation Apache Solr für TYPO3, http://jweiland.net/typo3-hosting/anleitung/solr-fuer-typo3-einrichten.html, Abruf 18.01.2013
o. V. (2012a):	Unicode Support, http://elasticsearch-users.115913.n3.nabble.com/Unicode-Support-Newbie-Looking-for-Clarification-td4024086.html, Abruf 18.01.2013
o. V. (2012b):	Hadoop Wiki, http://wiki.apache.org/hadoop/PoweredBy, Abruf 18.01.2013
o. V. (2013):	Lucene 4.0, http://www.heise.de/download/lucene-1146503.html, Abruf 18.01.2013
Alexa Internet, Inc. (2013a):	Search Results for http://apache.org/, http://www.alexa.com/search?q=http%3A%2F%2Flucene.apache.org%2Fcore%2F&r=home_home&p=bigtop, Abruf 18.01.2013
Alexa Internet, Inc. (2013b):	Search Results for http://www.elasticsearch.org, http://www.alexa.com/search?q=http%3A%2F%2Fwww.elasticsearch.org&r=site_screener&p=bigtop, Abruf 18.01.2013

Alexa Internet, Inc. (2013c): Search Results for http://www.hadoopsearch.net/, http://www.alexa.com/search?q=http%3A%2F%2Fwww.hadoopsearch.net%2F&r=home_home&p=bigtop, Abruf 18.01.2013

Banon, S. (o. J.): ElasticSearch Users, http://elasticsearch-users.115913.n3.nabble.com/, Abruf 18.01.2013

Bizosys Technologies Pvt Ltd. (2010a): Welcome to HSearch!, http://bizosyshsearch.sourceforge.net, Abruf: 18.12.2012

Bizosys Technologies Pvt Ltd. (2010b): HSearch Data Sheet, http://www.hadoopsearch.net/hsearchdatasheet.pdf, Abruf: 05.01.2013

Bizosys Technologies Pvt Ltd. (2011): HSearch, http://www.hadoopsearch.net, Abruf: 18.12.2012

Comundus (2011): Enterprise Search - SOLR, http://www.comundus.com/export/sites/default/downloads/opencms/Flyer_SOLR_Liferay.pdf, Abruf 18.01.2013

dkd Internet Service GmbH (2013): Solr für TYPO3 – Features, http://www.typo3-solr.com/de/solr-fuer-typo3/feature-liste/#c655, Abruf 18.01.2013

Elasticsearch (o. J.a): elasticsearch., http://www.elasticsearch.org/, Abruf: 02.12.2012

Elasticsearch (o. J.b): blog, http://www.elasticsearch.org/blog/, Abruf 18.01.2013

Elasticsearch (o. J.c): 0.20.2, http://www.elasticsearch.org/download/2012/12/27/0.20.2.html, Abruf 18.01.2013

Elasticsearch (o. J.d): Plugins, http://www.elasticsearch.org/guide/reference/modules/plugins.html, Abruf 18.01.2013

Elasticsearch (o. J.e): Modules, http://www.elasticsearch.org/guide/reference/modules/, Abruf 18.01.2013

Elasticsearch (o. J.f): Guide, http://www.elasticsearch.org/guide/, Abruf 18.01.2013

Elasticsearch (o. J.g): community, http://www.elasticsearch.org/community/, Abruf 18.01.2013

Elasticsearch (o. J.h): users, http://www.elasticsearch.org/users/, Abruf 18.01.2013

Elasticsearch (o. J.i): Attachment Type, http://www.elasticsearch.org/guide/reference/mapping/attachment-type.html, Abruf 18.01.2013

Elasticsearch (o. J.j): Analysis, http://www.elasticsearch.org/guide/reference/index-modules/analysis/, Abruf 18.01.2013

Elasticsearch (o. J.k): Clients & Integrations, http://www.elasticsearch.org/guide/appendix/clients.html, Abruf 18.01.2013

GitHub Inc. (2013): elasticsearch, https://github.com/elasticsearch, Abruf 18.01.2013

Hülsbömer, S. (2011): Die besten freien Business-IT-Tools, http://www.computerwoche.de/a/die-besten-freien-business-it-tools,2495289,2, Abruf 18.01.2013

Innovent Solutions (o. J.): Lucene & Solr Resources, http://www.innoventsolutions.com/lucene-solr-resources.html, Abruf 18.01.2013

Kleijn, A. (2006): Open-Source-Lizenzen, http://www.heise.de/open/artikel/Open-Source-Lizenzen-221957.html, Abruf: 10.01.2013

Klose, M. (2012): Wolkennah, Die Neuerungen von Apache Solr 4.0, http://www.heise.de/developer/artikel/Die-Neuerungen-von-Apache-Solr-4-0-1650785.html, Abruf: 12.12.2012

Kumar, B. (2006): Beef up Web search applications with Lucene, http://www.ibm.com/developerworks/library/wa-lucene2/, Abruf 18.01.2013

Lightwerk (o. J.): Enterprise Search mit Apache SOLR, http://www.lightwerk.de/produkte/solr-amp-lucene.html, Abruf 18.01.2013

Maas, G (2012): Die Enterprise Search SOLR, Erfolgreich Suchen mit einer Open Source Lösung, http://www.contentmanager.de/magazin/die_enterprise_search_solr.html, Abruf: 13.12.2012

McCandless, M. (2013): Indexing Throughout, http://people.apache.org/~mikemccand/lucenebench/indexing.html, Abruf: 15.01.2013

Miller, M. u. a. (2012): Apache Solr, http://wiki.apache.org/solr/, Abruf: 10.01.2013

Neumann, A. (2012): Lucene.net endlich ein Apache-Top-Level-Projekt, http://www.heise.de/developer/meldung/Lucene-net-endlich-ein-Apache-Top-Level-Projekt-1669454.html, Abruf 18.01.2013

Nioche, J. (2011): Web Scale Crawling with Apache Nutch, http://de.slideshare.net/hustwj/web-scale-crawling-with-apache-nutch, Abruf 18.01.2013

Rockstar template (2013): Install and Configure SOLR in Mac OS X, http://rockstartemplate.com/tutorial/install-configure-solr-mac/, Abruf 18.01.2013

Rowe, S. u. a. (2011): Lucene-java Wiki, LuceneFAQ, http://wiki.apache.org/lucene-java/LuceneFAQ, Abruf 18.01.2013

Schmidt, J. (2012): Suchen leicht gemacht: Apache veröffentlicht Lucene 4.0 und Solr 4.0, http://www.heise.de/developer/meldung/Suchen-leichtgemacht-Apache-veroeffentlicht-Lucene-4-0-und-Solr-4-0-1728500.html, Abruf 18.01.2013

Schneider, T. (2004): Anwenderhandbuch Regain, http://docs.huihoo.com/regain/Anwenderhandbuch_Regain_v1.0.pdf, Abruf 18.01.2013

Schonschek, O (o. J.): Big Data, Information geht auf Kollisionskurs, http://www.mittelstandswiki.de/wissen/Big_Data, Abruf: 16.01.2013

Search Technologies Corp. (2013): A Lotus Notes Data Connector for Solr, http://www.searchtechnologies.com/lotus-notes-connectors-solr.html, Abruf 18.01.2013

SourceForge (2013): HSearch From Bizosys, http://sourceforge.net/projects/bizosyshsearch/support, Abruf 18.01.2013

The Apache Software Foundation (2006): Apache Lucene - Index File Formats, http://lucene.apache.org/core/3_6_1/fileformats.html, Abruf 18.01.2013

The Apache Software Foundation (2012a): Apache Lucene Core, http://lucene.apache.org/core, Abruf: 06.12.2012

The Apache Software Foundation (2012b): Apache Solr, http://lucene.apache.org/solr/, Abruf: 12.12.2012

The Apache Software Foundation (2012c): What Is Apache Hadoop?, http://hadoop.apache.org/, Abruf: 18.12.2012

The Apache Software Foundation (2012d): Apache License, http://www.apache.org/licenses/LICENSE-2.0, Abruf 18.01.2013

The Apache Software Foundation (2012e): Lucene Core News, http://lucene.apache.org/core/corenews.html, Abruf 18.01.2013

The Apache Software Foundation (2012f): Lucene Release Docs, http://lucene.apache.org/core/documentation.html, Abruf 18.01.2013

The Apache Software Foundation (2012g): Lucene Tutorials, http://lucene.apache.org/core/quickstart.html, Abruf: 18.01.2013

The Apache Software Foundation (2012h): Lucene Mailing Lists and IRC, http://lucene.apache.org/core/discussion.html, Abruf 18.01.2013

The Apache Software Foundation (2012i): Lucene 4.0.0 demo API, http://lucene.apache.org/core/4_0_0/demo/overview-summary.html#overview_description, Abruf 18.01.2013

The Apache Software Foundation (2012j): Apache Lucy, http://lucy.apache.org/, Abruf 18.01.2013

The Apache Software Foundation (2012k): SolrTM Mailing Lists and IRC, http://lucene.apache.org/solr/discussion.html, Abruf 18.01.2013

The Apache Software Foundation (2012l): Solr Tutorial, http://lucene.apache.org/solr/tutorial.html, Abruf 18.01.2013

The Apache Software Foundation (2012m): Thanks, http://www.apache.org/foundation/thanks.html, Abruf: 18.01.2013

The Apache Software Foundation (2012n): Solr News, http://lucene.apache.org/solr/solrnews.html, Abruf: 18.01.2013

Walker-Morgan, D. (2011): Ten years of the Lucene search engine at Apache, http://www.h-online.com/open/news/item/Ten-years-of-the-Lucene-search-engine-at-Apache-1350761.html, Abruf 18.01.2013

Worral, P. (2012): Public Websites using Solr, http://wiki.apache.org/solr/PublicServers, Abruf: 12.12.2012

Zacher, M. (2007): Wie Open Source die IT verändert, http://www.computerwoche.de/a/wie-open-source-die-it-veraendert,1847565, Abruf: 11.01.2013

Zend Technologies Ltd. (2012): Indexerstellung, http://framework.zend.com/manual/1.12/de/zend.search.lucene.index-creation.html#zend.search.lucene.index-creation.permissions, Abruf 18.01.2013

Gesprächsverzeichnis

Pilato, D. (2012):	Entwickler Elasticsearch, E-Mail am 08.01.2013

Evaluation of Asynchronous Server Technologies

by Can Paul Bineytioglu, Joe André Boden, Rocco Schulz,

Max Vökler, Robert Wawrzyniak

provided on January 25, 2013

School of: Business

Program: International Business Information Management

Course: WWI2010I

Corporate State University
Baden-Wuerttemberg - Stuttgart

Contents

List of Abbreviations

AJAX ..... Asynchronous JavaScript and XML

CRUD ..... Create Read Update Delete

CSS ..... Cascading Style Sheets

HTML ..... Hyper Text Markup Language

HTTP ..... Hyper Text Transfer Protocol

HTTPS ..... Hyper Text Transfer Protocol Secure

I/O ..... Input / Output

IETF ..... Internet Engineering Task Force

IPC ..... Inter Process Communication

J2EE ..... Java 2 Enterprise Edition

JSON ..... JavaScript Object Notation

JVM ..... Java Virtual Machine

NPM ..... Node.js Package Manager

POC ..... Proof Of Concept

RFC ..... Request for Comments

TCO ..... Total Cost of Ownership

TTM ..... Time to Market

UI ..... User Interface

URL ..... Uniform Resource Locator

List of Tables

List of Figures

List of Listings

1 Introduction

1.1 Objectives

Permanent change is a basic concept and global phenomenon in the IT industry. Most recent trends include cloud computing, big data, social media, and mobile computing. The tremendous impact of the latter one is revealed by a Cisco study, which states that in 2011 the mobile data traffic doubled the fourth year in a row.[1] That enormous traffic and the nature of social media applications raised problems in the common synchronous server technologies. Since 1996 the Microsoft Internet Explorer supports asynchronous content loading. A popular asynchronous technology is AJAX (asynchronous JavaScript and XML). It got even more attention when Google introduced a well-respected deployment of standards-compliant browser applications leveraging AJAX - Gmail (2004) and Google Maps (2005).[2] Since then asynchronous technologies encountered a steep carrier and are on the way to change the way of programming. The main goal of this paper is to identify specific use cases for asynchronous technologies on the server-side and to make clear why this technology has evolved, showing when and when not it is worth considering. Special emphasis is put on applications in the business environment. The authors' intention is also to build a working prototype in order to present the advantages and disadvantages of this rising technology.

The two-sided approach – explaining the change in web development and an application in a business environment – harnesses two main programming languages, which are ubiquitous in each community: Java for the enterprise world and JavaScript for web developers. This paper elaborates the concepts behind the young frameworks Node.js, that briefly explained brings JavaScript to the server, and Vert.x, a Java implementation of asynchronous technologies, and analyzes their technical strengths and weaknesses. Furthermore non-functional attributes will be evaluated.

1.2 Methodology and Structure

The whole thesis consists of seven chapters. The first two parts explain the authors' motivation and introduce the basic concepts of web technologies and the difference of synchronous and asynchronous aproaches, whose use cases are explained in chapter three. Two prototypes of an insurance purchase webpage are implemented in Node.js and Vert.x in chapter four. Having set up an exemplary Proof of Concept (PoC), there are additional concepts explained that are important in modern web development (e.g. AJAX and WebSockets). This implementation was furthermore taken into consideration when non-functional attributes like maintainability,

[1] Cf. w. a. (2012a)

[2] Cf. Swartz, A. (2005)

integration and performance were evaluated in chapter five. Chapter six provides an outlook on the future of Node.js and Vert.x before chapter seven concludes the main research findings.

1.3 Problem Description

In traditional web application development data is transmitted synchronously, i.e. upon a GET/-POST request, so that the result can be displayed only after transmission and processing are finished.

While maintaining simplicity and predictability this can cause serious latency when uploading large pieces of data, most commonly complex forms for registration. Naturally rich content such as images and videos cause even more delays.

As demand around collaborative access and media richness evolved, this became a serious bottleneck, essentially preventing these types of applications. On the client-side (browser-side) developers were able to work around the issue of synchronous transmission using the XmlHttpRequest object which allows to request resources programmatically (using JavaScript) while deferring handling of the response to a callback (see Fig. 1[3]) thus enabling much more responsive software.

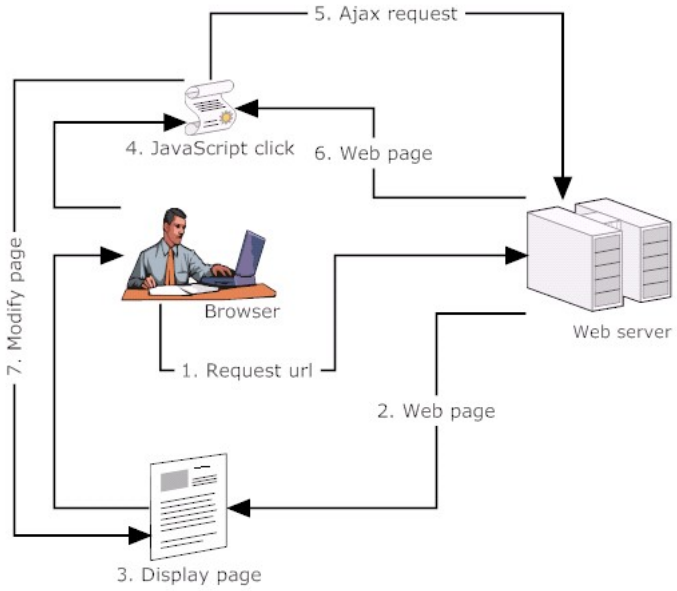


Fig. 1: AJAX diagram

Although this addressed the issue on the client-side, server-side request were still handled very much in a synchronous fashion. The Apache web server e.g. forks a new process for each incoming request[4]. As popular applications have to cope with unprecedented amounts of concurrent users in conjunction with massive request counts, this obviously causes performance issues. Reasons for these issues are that blocking I/O streams also cause the related thread to

[3] Matejka, J. (2012)

[4] Cf. The Apache Software Foundation (2013)

idle and consequently block this thread, meaning it cannot be used for other requests. The next chapter expands upon I/O bound and CPU bound applications as well as single-/multithreaded technologies, and will explain further, why asynchronous technologies can help mitigate problems that especially applications with a lot of I/O streams face.

2 Setting the Context

2.1 Potential Web Development Issues

This section is used to briefly outline some theories that are relevant when dealing with synchronous and asynchronous approaches.

2.1.1 I/O and CPU Bound Applications

I/O bound applications I/O bound means that a system requests data faster than the peripheral devices (e.g. hard disk) can transfer it, causing a process/thread to be put asleep.[5] The program could be sped up immediately by faster I/O (e.g. using a Solid State Drive instead of a normal hard disk).

CPU bound applications CPU bound or compute-bound refers to operations that cause high CPU workload, like compiling code, spell checking, grammar checking, transcoding audio or video data.[6] They are usually executed in several threads in order to not block one thread and have all users waiting for this thread. The program could be sped up immediately by better CPU performance.

2.1.2 Single-/Multithreading and Event-driven Development

In multithreaded applications several threads can run simultaneously within one process. Several threads can access shared memory concurrently, which can cause inconsistent states. This can be avoided by synchronizing threads - e.g. with locks. This means that programmers need to take into account every possible execution order to effectively avoid program defects such as data races and deadlocks.[7] This can be time consuming and potentially results in error-prone code.

Event-driven programming uses an event loop, which is a single thread that is running inside the main process. The loop constantly checks for new events. When an event is detected, the loop invokes the corresponding callback function. The callback is processed in the same thread, which means that there is at most one callback running at a time. The event loop continues when the callback has completed. As a result the developer does not need to take care of concurrency issues during development. But the developer's task is to write light event handlers that can be processed quickly as every callback is an interruption of the event processing in the

[5] Cf. Caldera International Inc. (2003), p. 10
[6] Cf. Richter, J. (2010), p. 718
[7] Cf. Breshears, C. (2009), p. 10

event loop.[8] Memory or processor intense callbacks can lead to growing queues of unserved events, which eventually results in a slow application or service[9].

2.2 Concept of Synchronous Processing

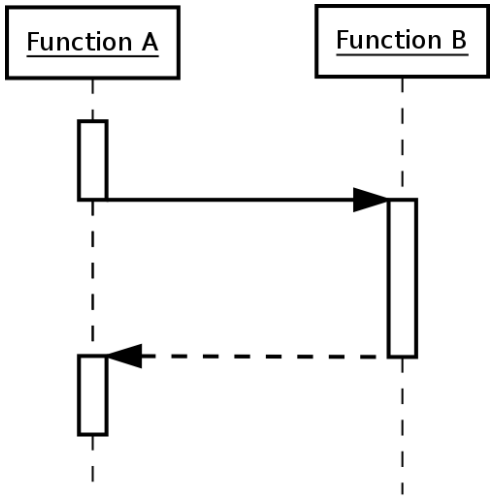


Fig. 2: Synchronous, blocking function call

In synchronous processing a running thread needs to wait for the completion of the I/O operation before it can continue. The thread is in an idle state while it is waiting, which allows another process or thread to occupy the CPU in the meanwhile.

2.3 Concept of Asynchronous Processing

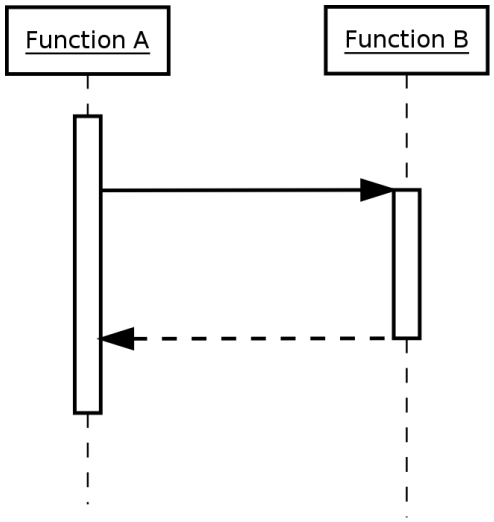


Fig. 3: Asynchronous, non-blocking function call

[8] Cf. Hughes-Croucher, T. (2010)
[9] Cf. Teixeira, P. (2013), p. 48

An asynchronous programming style uses a different concept. The flow of an application is determined by events, which is why this style is also called event-driven programming.[10] Asynchronous processing means that the executing thread is not blocked. There is no return value - instead an event handler is provided as a second argument. This function is also referred to as a callback function. It is called as soon as the read operation has completed and is processed in the same thread, which means that there is at most one callback running at a time and the developer does not face concurrency issues. The developer's task is to write lightweight event handlers that can be processed quickly as every callback is an interruption of the event processing in the event loop.[11] Memory or processor intense callbacks can lead to growing queues of unserved events, which eventually results in a slow application or service.[12]

2.4 Technical Comparison of Asynchronous and Synchronous Processing

A typical synchronous call is provided in Lst. 1. The content of a file is read and displayed afterwards. The program is blocked until the read operation has finished.

```
1 reader = new FileReader();
2 content = reader.readAsText("input.txt");
3 printContent(content);
```

Lst. 1: Pseudocode: Synchronously reading and displaying a file's content

As opposed to the synchronous call, Lst. 2 demonstrates how the same example can be realized in an asynchronous fashion using the event-driven methodology: After the file was read, the callback is triggered in order to display its content.

```
1 read_completed = function(content) {
2   printContent(content);
3 }
4
5 reader = new FileReader();
6 reader.readAsText("input.txt", read_completed);
```

Lst. 2: Pseudocode: Asynchronously reading and displaying a file's content

2.5 Existing Asynchronous Frameworks

The principle of asynchronous processing has existed on the client-side for a long time and has also evolved on the server-side. This section will give a brief market overview, which is followed by the introduction of two upcoming frameworks for server-side asynchronous development: Node.js and Vert.x.

[10] Cf. Teixeira, P. (2013), p. 16
[11] Cf. Hughes-Croucher, T./Wilson, M. (2012), p. 10 et sqq.
[12] Cf. Teixeira, P. (2013), p. 48

2.5.1 Market Overview

Table 1 shows that there is real competition from established communities (namely the Ruby and Python ones). An important aspect in determining the potential success of a new technology is measuring the interest of software developers in the newcomer. Google-developed Google Trends visualizes the 'search interest', i.e. the amount of queries of a particular subject and related ones[13].

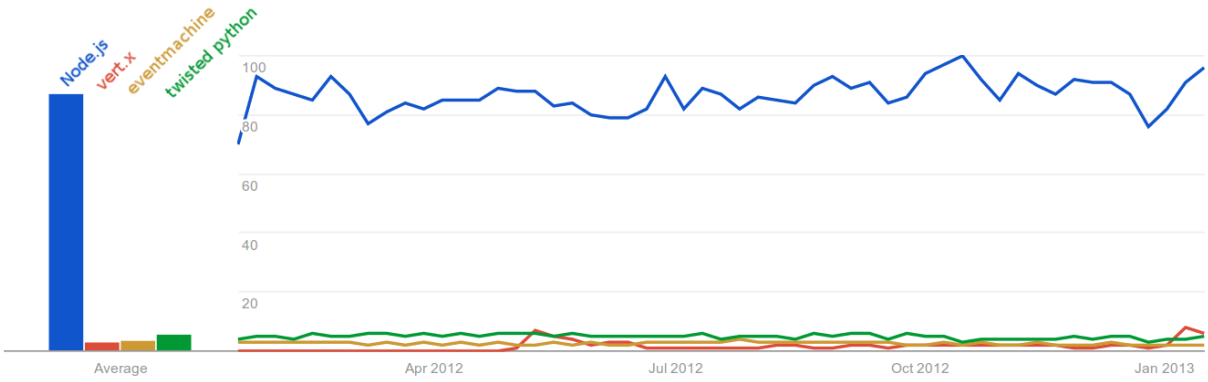


Fig. 4: Google search volume of Node.js, twisted, eventmachine and Vert.x compared

Looking at the graphs in Fig. 4, it can be concluded that interest in Node.js is very high at the moment in absolute terms as well as in relative terms when comparing it to peers at the same point in there lifecycle. Interest in Vert.x is comparably low, considering that it's idea is very similar to Node.js. The gap in interest could be related to many reasons, two of which will be discussed briefly thus concluding this section.

Theory (1) would be that by the time Vert.x became complete enough to develop productively (at least as productive as was possible with Node.js at the same point), Node.js already established itself as the dominant technology especially if mindshare is concerned, which is supported by the data from Google. Theory (2) is based around technical considerations. Because of the dogmatic nature of asynchronicity in the Node.js framework, every package available on npm is built with asynchronous calling and non-blocking I/O in mind. Vert.x on the other hand is in a curious position here. Vert.x itself is architecturally similar to Node.js but its ecosystem of established Java libraries is not developed with these principles in mind. This poses the danger of executing synchronous/blocking code, negating all of Vert.x' advantages. Obviously this can limit interest from the outset.

Another important metric to assess the potential impact of a framework is the technical background and the availability of skilled developers. Redmonk investigated in three consecutive years the popularity of programming languages by analyzing the usage on GitHub[14] and Stack-

[13] Cf. Google Inc (2013)
[13] Cf. ibid.
[14] See `http://github.com`

Overflow[15]. As of September 2012, JavaScript was ranked first before Java, PHP, Python and Ruby[16] – all of which have shown a strong performance over the past three years. This is an affirmative aspect with regard to the potential target audience of developers of Node.js and Vert.x, which will be introduced in the next section.

Name	**Language/s**	**Description**
Twisted	Python	"Twisted is an event-driven networking engine written in Python and licensed under the open source MIT license"[17]. Twisted is a mature framework with a large number of supported networking protocols. Its development is backed by an active community.[18]
EventMachine	Ruby	"EventMachine is a library for Ruby, C++, and Java programs. It provides event-driven I/O using the Reactor pattern."[19]
Node.js	JavaScript	Node.js is based on Chrome's JavaScript runtime *V8*. The platform is fully event-driven and offers core functionalities for network programming. Its functionality can be extended with a large number of modules using an integrated package management system. Node.js started development in 2009 and the community is growing since that.[20]
Vert.x	JavaScript, Java, Python, Groovy, Ruby, Coffeescript	A JVM based platform for network applications which is inspired by Node.js. Vert.x comes with its own event bus system that allows distributing applications among multiple network nodes. Support for the languages Scala and Clojure is scheduled for future releases.[21]

Table 1: Existing asynchronous programming frameworks

[15] See `http://stackoverflow.com`
[16] Cf. O'Grady, S. (2012)
[17] Cf. w. a. (2012c)
[18] Cf. Fettig, A. (2005), p. 12
[19] Klishin, M. (2012)
[20] Cf. O'Dell, J. (2011b)
[21] Cf. w. a. (2012d)

Table1 only considers four major frameworks. Other libraries/frameworks include, but are not limited to:

- Perl's AnyEvent[22]
- Ruby's Rev[23] and Cramp[24] which is built on EventMachine[25]
- Python's tornado[26], asyncore module[27] and sitting on that Medusa[28]
- Python's gevent[29] library
- .NET's Rx[30] library
- JVM's NIO[31]
- Linux' epoll[32], C's libev[33] and libeio[34]
- Erlang[35]

2.5.2 Node.js

Node.js is a framework based on Google's V8 JavaScript Engine, libUV as platform layer[36], and a core library written in JavaScript. Node.js brings JavaScript to the server-side and therefore makes it possible to write a web application on both, the server-side and the client-side in one language – JavaScript.

Compared to other technologies that try to apply the fundamentals of asynchronous programming, Node.js' libraries are programmed completely asynchronous. Its architecture consists of modules to simplify the creation of complex applications. Modules are encapsulated pieces of code like libraries in C or units in Pascal. Every module contains of set of functionalities that share one common purpose. The visibility of functions and variables is limited to the module they belong to. Using a module requires to call the function `require()` and its allocation to a

[22] http://software.schmorp.de/pkg/AnyEvent.html
[23] http://rev.rubyforge.org/rdoc/
[24] http://cramp.in/
[25] http://rubyeventmachine.com/
[26] http://www.tornadoweb.org/
[27] http://effbot.org/librarybook/asyncore.htm
[28] http://packages.debian.org/sid/python-medusa
[29] http://www.gevent.org/
[30] http://msdn.microsoft.com/en-us/data/gg577609
[31] http://docs.oracle.com/javase/1.5.0/docs/guide/nio/
[32] http://linux.die.net/man/4/epoll
[33] http://software.schmorp.de/pkg/libev.html
[34] http://software.schmorp.de/pkg/libeio.html
[35] http://www.erlang.org/
[36] See https://github.com/joyent/libuv

variable. Every function in Node.js is asynchronous, which means that the current thread continues running and does not wait for completion (see Fig. 3). Exemplarily, reading a file works in a way that when the read operation has finished, a callback function is executed to continue working with the file's content.

Node.js is single-threaded, which is why only one thing can happen at the same time. Node.js's asynchronous programming style is implemented through events that determine the flow of execution. The events are processed within an event loop (see section 2.1.2). Consequently, the code execution is thread-safe and there is no issue regarding concurrency that affects the shared memory state.[37]

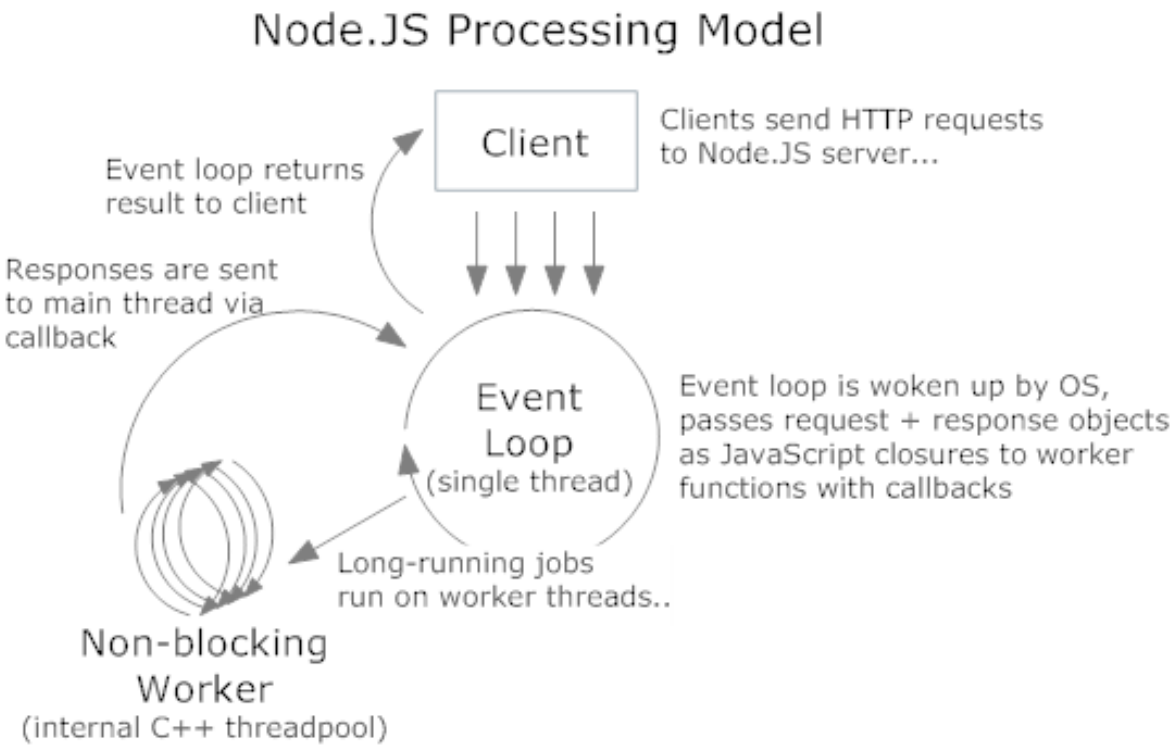


Fig. 5: Processing Node.js[38]

Node.js's large performance benefits are caused by the use of Google's new JavaScript engine V8, which naturally ships with the web browser Chrome and is used for Node.js too. Due to the fact that V8 compiles JavaScript into real machine code before executing it, Node.js has an advantage over traditional techniques such as executing bytecode or interpreting it.[39]
Node.js by default provides core modules that are packed in its binary, to help access files on the file system, create HTTP and TCP/UDP servers, and to perform other useful functions.[40]

[37] Cf. Teixeira, P. (2013), p.16
[38] Stannard, A. (2011)
[39] See `https://developers.google.com/v8/intro`
[40] See `http://nodejs.org/api/modules.html#modules_core_modules`

The latest version of the Node.js environment can be installed from a GitHub repository.[41] Even though the core modules are sophisticated enough to write complex applications, Node.js is not complex itself. The node package manager, however, provides 21,104 packaged modules as of January 18, 2013[42], which enable the developer to leverage a variety of pre-defined complex functionalities. Since Node.js version 0.6.3 the package manager npm is deployed and installed automatically within the environment.[43] Npm is run on the command line and manages dependencies for applications.

Node.js received standing ovations from the web developer community, which lead to spectacular participation in the project. Primarily, Node.js is supported by a San Francisco based company named Joyent, which offers cloud services such as Infrastructure as a Service and Platform as a Service to large companies, focussing on open-source projects such as Ruby on Rails and Node.js.[44]

Node.js's documentation and manual are extensive and cover all functionalities of the non-extended edition. Node.js is licensed under the MIT License[45], meaning that it permits reuse within proprietary software provided all copies of the licensed software include a copy of the MIT License terms.

2.5.3 Vert.x

Vert.x is a programming framework that runs on the Java Virtual Machine (JVM). It is hence possible to scale over available cores without manually forking multiple processes.

The application API is exposed in multiple programming lanugages (see table 1).

In Vert.x the smallest available deployment unit is a verticle, which runs inside an instance. Each instance runs single-threaded. Multiple verticles can be run inside one instance as depicted in Fig. 6.

When multiple instances are run on one machine, Vert.x automatically distributes incoming requests among all running instances in a round-robin fashion, so that each Vert.x verticle instance remains single-threaded. Technically seen, there is more than one event loop, but Vert.x employs a mechanism that ensures that all events that are related (e.g. because they triggered each other) are run in the same context, i.e. the same thread. This allows real concurrency while providing the benefits of single-threaded programming style.

Vert.x also includes a distributed event bus, which enables verticles to communicate with each other, either within the same instance or across different instances. The event bus al-

[41] See `http://nodejs.org/dist/latest/`
[42] Cf. Joyent Inc (w. y.[b])
[43] piscisaureus (2011), Cf.
[44] Cf. Thornsby, J. (2010)
[45] See `http://opensource.org/licenses/MIT`

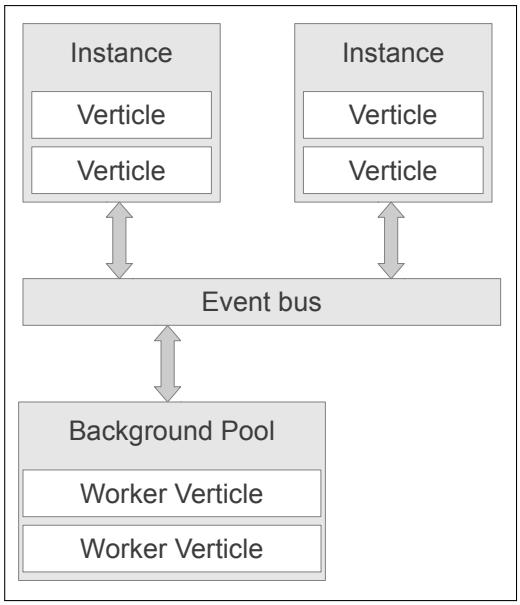


Fig. 6: Abstracted deployment units of Vert.x

lows direct communication with in-browser JavaScript as well. Vert.x uses the SockJS library for this purpose which is a wrapper around normal websockets with a fallback to alternative technologies, such as long-polling[46]. This is shown schematically in Fig. 7.

Vert.x allows to run I/O heavy tasks in separate worker threads that reside in a so-called background pool as these tasks would otherwise block the event loop as described in section 2.1.2.

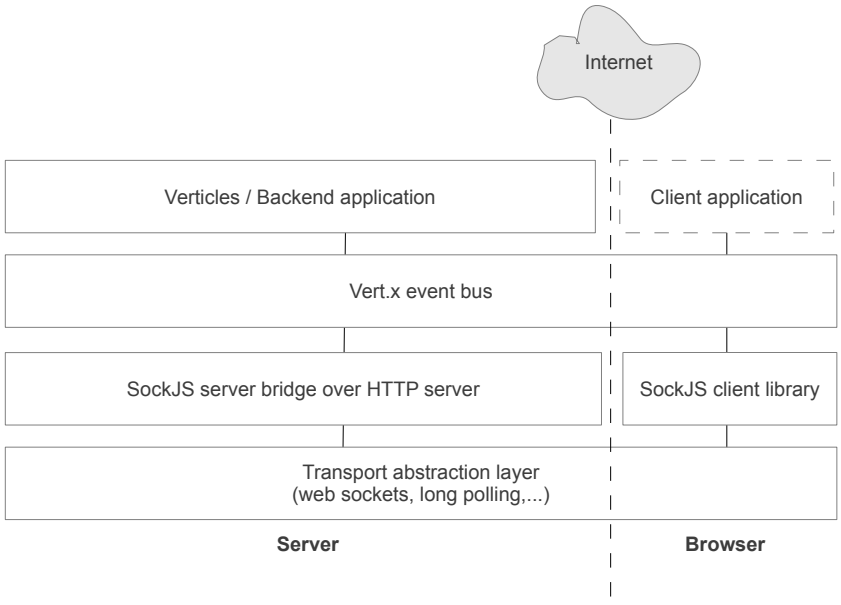


Fig. 7: Vert.x server bridge - schematic view

The core functionality of Vert.x covers basic networking tasks and protocols, web servers, network clients, access to the file system, shared maps and the event bus. The core libraries

[46] Cf. Majkowski, M. et al. (2013)

of Vert.x can be embedded in any JVM program for reuse in larger projects.
As opposed to Node.js, the core functionality and API can be considered quite stable as changes need to be done in all supported languages.[47]
The core can be extended with additional features that are provided by optional modules which can be obtained over a public git-based module repository.[48]. The repository currently contains 16 distinct modules in different versions.[49] Compared to Node.js this number is very low and reflects the relatively low interest in the project in comparison to Node.js (see Fig. 4). However one should bear in mind that many libraries available for the languages supported by Vert.x can be used within this framework as well.

An extensive online documentation is available for all supported languages. Additionally, code examples for most features are available for all supported languages in a public repository[50].
Vert.x is open source and licensed under the Apache Software License 2.0[51], so that commecial redistribution in closed source projects should not be an issue. At present the project is owned by VMware, but it is likely that it will be transferred to an Open Source foundation (see section 5.1.2).

[47] Cf. w. a. (2012d)
[48] Cf. w. a. (2012e)
[49] Cf. Fox, T./Yates, T., et al. (2012)
[50] Cf. Fox, T. (2013a)
[51] See `http://www.apache.org/licenses/LICENSE-2.0.html`

3 Areas of Application

3.1 Use Cases

The non-blocking nature of asynchronous calls is important in all types of applications that need to handle a large number of requests. Its event-driven model also makes it a good choice for most types of real-time processing.
A fairly obvious fit is given for I/O bound applications, as the single-threaded asynchronous processing does not wait for I/O streams to finish, but instead uses callbacks to come back to the processing part as soon as the I/O has finished. The single-threaded nature of event loops allows to make efficient use of the thread as long as callback functions don't take up too much time. This makes asynchronous server applications a good fit for most types of networked applications that either tend to keep many inactive connections or need to handle many concurrent short-lived connections.

A few possible use cases are described in the list below. Furthermore case studies of real implementations are provided in appendices 4, 5, 6 and 7.

Web traffic monitoring By getting notified of every single request that arrives on an online service, an asynchronous application could keep track of the current number of requests and the distribution of requests among all available resources. These simple statistics could then be pushed to a Web interface frequently to allow real-time monitoring of a service in the browser.

Realization: All services that should be monitored could implement a middleware that intercepts each request and fires a call to the asynchronous server. However that would add a lot of overhead for the services that should be monitored. Alternatively all clients that access these services could be forced to send a second request to the asynchronous server.

The asynchronous server instance could then handle each request by adding relevant information to an in-memory database. In the case of HTTP requests these information could simply be taken from the various HTTP header fields.[52]

A second server instance could be set up to handle the online dashboard. In order to allow instantly updating the dashboard all clients should be connected to this instance via a WebSocket (described in section 4.2.3). Depending on the amount of visitors this could result in a large number of open connections. However due to the event loop based processing the server instance will always just use one thread for all connections as opposed to a classical web server, which would keep one thread for each connection. Thus the servers footprint is much smaller.

[52] Cf. Fielding, R. et al. (1999)

Known implementation: Hummingbird is a real-time web traffic visualization tool that uses Node.js for all processing. When loading a website the browser sends a request for each file that is referenced in the HTML file. Hence each page that should be tracked by Hummingbird integrates a reference to a tracking pixel, which is served by the Hummingbird application.[53]. The dashboard is updated 20 times a second over a WebSocket connection.

Streaming Applications that involve a lot of streaming of data, such as file uploading or video broadcasting, represent another appropriate use case for asynchronous technologies. Especially Node.js uses an abstraction called streams. Such a stream represents a flow of data and can appear as a readable stream or writable stream.[54] Why Node.js is beneficial in terms of streaming data becomes clear when considering the slow client problem and the way Node.js and its streams address it.

Realization: The slow client problem refers to the scenario where a process reads data, processes it and sends it to another consumer.[55] If the producer (readable stream) sends data faster than the consumer (writable stream) can process it, the data has to be buffered. For instance, consider uploading a local file: The readable stream is relatively fast as the file is stored locally. Assuming the network connection to the client is slow, the writable stream will consequently be slow as well. As a result, the data has to be buffered. Because data will be buffered in memory for each write command, this will result in memory problems after a certain amount of requests.[56] This problem can be avoided by pausing the producer until the consumer catches up. At this point, Node.js comes into play. Readable streams in Node.js can be paused and resumed, which addresses exactly the described problem.[57] Listing 3 illustrates how this is realized.[58]

The data flow control process can be automated by using the `stream.pipe()` method.
Known implementation: Transloadit[59] is a company offering a solution for file-uploading and encoding. They have implemented their service in Node.js and take advantage of the functionality of Node.js's streams. Also Yahoo uses Node.js for file uploads.[60]

Lightweight network servers Asynchronous technologies are a good fit for HTTP servers and are efficient when it comes to handling many concurrent connections in a single process. A web server application should handle the cookie header parsing, maintainance of

[53] Cf. Nutt, M./Wong, B. (w. y.)
[54] Cf. Teixeira, P. (2013), p. 75
[55] Cf. ibid., p. 80
[56] Cf. ibid., p. 81
[57] Cf. ibid., p. 81
[58] Cf. ibid., p. 81
[59] `http://transloadid.com`
[60] Cf. O'Dell, J. (2012)

```
require('http').createServer(function(req, res) {
  var rs = fs.createReadStream('/path/to/big/file');

  rs.on('data', function(data) {
    if (!res.write(data)) {
      rs.pause();
    }
  });

  res.on('drain', function() {
    rs.resume();
  });

  rs.on('end', function() {
    res.end();
  });
}).listen(8080);
```

Lst. 3: Controlling streams in Node.js

sessions, serving static files, parsing the request body etc.[61] Most of these tasks are very simple and don't require much computations. The Proof of Concept also shows that other protocols, like WebSockets, can take a particular advantage from asynchronous technologies. In general any network server that only needs to handle a small set of tasks could be written easily using one of the mentioned frameworks.

Realization: The realization can be seen exemplarily in listings 6 et sqq., as this is part of the Proof of Concept.

Known implementation: *Connect*[62] is a well-known Node.js-based middleware that provides web server functionality and is considered an industry standard.[63] For Vert.x, there is a web server project called *mod-web-server*[64], which efficiently serves files and can also be configured as en event bus bridge to client-side JavaScript.

Proxies There are two indicators that militate in favor of an asynchronous server: Proxies usually handle a large number of requests and secondly they mainly do I/O related work but no big computations. A common task might be to filter a set of URLs, which can be easily done with an array in Java and JavaScript.

Realization: The requests are caught intercepted by the proxy server, which then requests the actual website and passesit back to the requesting client. Other requests can be handled while the resources are being loaded. There are examples of Node.js showing a proxy written on 20 lines of code[65]. These implementations could be extended to

[61] Cf. Teixeira, P. (2013), 197 et seq,
[62] Cf. O'Dell, J. (2012)
[63] Cf. Roden, G. (2012), p. 145
[64] Cf. w. a. (2012f)
[65] Cf. Krumins, P. (2010)

provide more sophisticated features.

Known implementation: There is a Vert.x implementation called *vertx-proxy*[66] as well as a Node.js proxy called *node-http-proxy*[67], both of which support HTTPS.

Push-notification and messaging systems Asynchronous technologies are suitable for web applications and networking tools that have to handle many concurrent connections as they are event-driven.[68] Such applications process a high number of requests fast, which in most cases means in real-time. At the same time, the tasks are not CPU-intensive and do not block the event loop.

Realization: Such web applications or services can include chat platforms, sport bets, e-mailing and instant messaging applications.[69] As opposed to synchronous processing, the event-loop keeps working while notifications are delivered to clients and potential call-backs are processed as soon as messages were received. As everything occurs in a single thread, an overhead of switching threads and concurrency issues can be avoided.

Known implementation: *AIRCODE* has implemented such a real-time notification application entirely in Node.js (see Appendix 5). Moreover, financial applications that need to show immediate reaction to stock changes may take advantage of asynchronous processing to implement real-time notification systems.

3.2 Don't Use Cases

As with all technologies one has to carefully evaluate whether or not it suits the requirements of a project. Besides those use cases listed in the previous section there are also a few usage scenarios where one should not use asynchronous frameworks like Node.js or Vert.x. In general, the choice always depends on the very specific needs for every single use case. Critics argue that asynchronous frameworks are not necessarily the best choice only because they represent a new stream.[70,71,72]

Since one key feature of asynchronous technologies is non-blocking I/O, use cases that hardly encompass interaction with external resources do not set a precedent and might even be penalized. This penelization would be caused by the fact that the strength of non-blocking I/O could not be leveraged on the one hand and on the other hand one thread would be busy through computations and cause long response times.[73]

[66] Cf. Fletcher, R. (2012)
[67] Cf. w. a. (2012b)
[68] Cf. Teixeira, P. (2013), p. 17
[69] Cf. Geisendoerfer, F. (2011)
[70] Cf. Semerau, R. (2011)
[71] Cf. Arranz, J. M. (2011)
[72] Cf. Behren, R. von/Condit, J./Brewer, E. (2003)
[73] Cf. Roden, G. (2012), p. 14

Expanding upon the non-blocking I/O strenghts, it becomes obvious that asynchronous frameworks are not I/O bound. Consequently, they are not designed to deal with CPU bound use cases.[74]

Datawarehousing with analytics Doing analytical and complex computations on large data sets at runtime usually requires a lot of computing power. It can be expected that each request will require quite some time to be processed. This computation time cannot be shortened much when run on a single thread but could potentially be sped up significantly when run on multiple cores in parallel.

CRUD / HTML applications This refers to classical websites that basically only serve as an interface for an object relational model. At present Node.js and Vert.x do not provide additional benefits to scalability for these types of web applications.[75] Unless the page has to deal with a large number of concurrent requests it should be considered to use more powerfull frameworks like *Ruby On Rails*, *Grails* or *Django*. These are currently better suited for quickly developing such an application. Providing a site that is suited for millions of requests per day does not automatically increase the number of users.

XML-based environments This applies for Node.js in particular. Node.js makes strong use of JSON as a data format and does not support XML. Node.js should hence not be used in an environment that communicates in data formats like XML.

This leads to the simple conclusion that migrating from legacy applications to asynchronous server applications can neither be justified by the benefits that asynchronous technologies can offer, nor by the hype around these frameworks alone. The true indicator should be whether or not it is the best suited technology for the application area in the given enerprise context.

[74] Cf. Nguyen, D. (2012), p. 15
[75] Cf. Roden, G. (2012), p. 15

4 Exemplary Implementations

A simple web form application has been implemented in both Node.js and Vert.x to further analyze non-functional requirements and collect practical experience with these frameworks. These implementations are thoroughly described in the following sections.

4.1 Software Description

The exemplary implementation consists of a web service that can be used to calculate the expected fee for an insurance. The idea behind this application is that two clients can collaborate during the form entry process before submitting all values for the fee calculation to the server. However, this application should only serve as a demonstration of the used asynchronous frameworks and is hence very simplified.
Functionalities that are included are usual POST requests as well as real-time communication via a WebSocket (described in section 4.2).

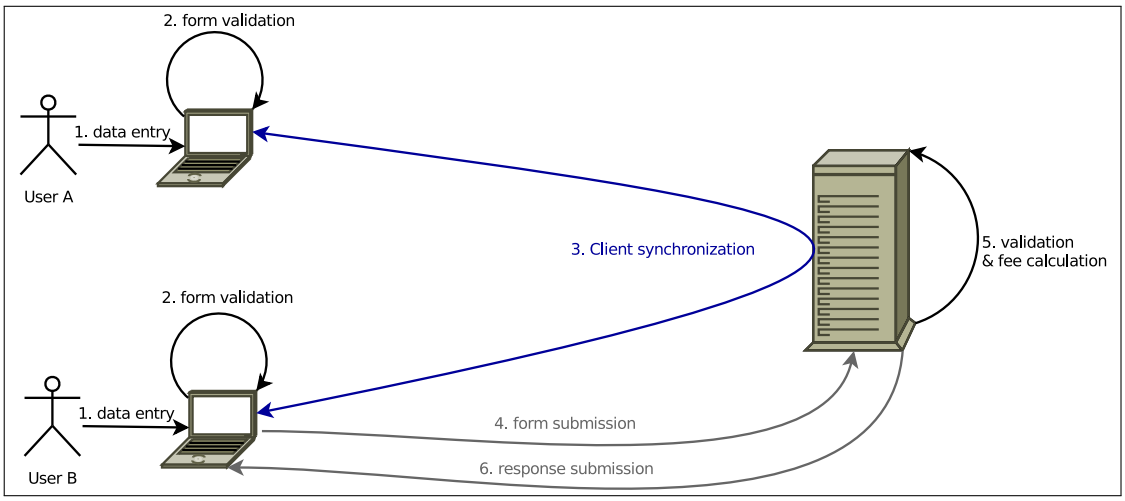


Fig. 8: Usage workflow of the exemplary application

The basic usage workflow is shown in Fig. 8. Upon page load both users can enter a form key in their browsers. All users with the same key will be collaborating with each other and receive all input changes from the other users - this is comparable to a simple chat. The form consists of two parts - the first part being a section for personal details and the second part for parameters that will be used to determine the fee for the insurance.
The users then fill in all necessary form fields (step 1 in Fig. 8) and instantly get feedback on the vailidy of the entered values (step 2 in Fig. 8). The evaluation is done locally on the clients' machines. While typing the data into the form fields, those inputs are instantly sent to the server using WebSockets. The server will distribute these inputs to all clients that have opened the same form, which is identified via the key that was set initially (step 3 in Fig. 8).
Once all fields that are required for the fee calculation are filled in correctly the client automati-

cally submits the values via AJAX to the server over a secure HTTPS connection (step 4 in Fig. 8). On the server these values are then validated again to avoid processing of manipulated requests (step 5 in Fig. 8). The result of the fee calculation is then returned to the client, which closes the HTTP connection (step 6 in Fig. 8).
This AJAX call is triggered when the three fields of the second part (being *career*, *amount insured* and *ownership stake*) are valid.
Using two different technologies (WebSockets and AJAX) for this workflow is certainly not the best approach from a design point of view but is conceptually valuable to show several ways of communication.

4.2 Software Design

This simple usage scenario leads to a few requirements and design decisions. The user interface has to consist of HTML, CSS and JavaScript files as it will be displayed in a web browser. In general, these files are not initially available on the clients machine and need to be delivered via HTTP/HTTPS by the server. These file requests are usually done via GET.

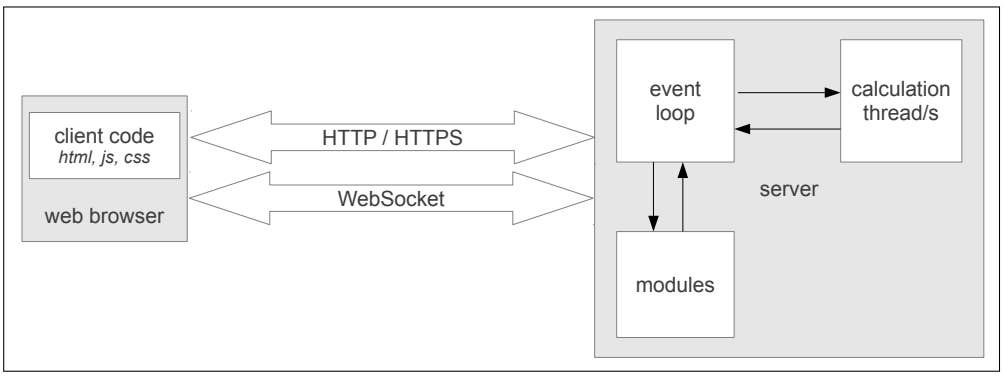


Fig. 9: High level architecture of the application

In addition to the static files, the server will also need to process requests that are sent via POST in order to receive the form data for the fee calculation.
The body size of these requests is fairly small as the submitted values consist of plain text only. Therefore it is safe to assume that it will not be necessary to buffer each part of the requests body but start calculating once the data submission has been completed.
In reality these calculations are much more complex than in this simple scenario, so that they have to be done in a separate thread outside of the event loop. In the implementation these long running calculations are simulated artificially for demonstration purposes.
Additional features in the back-end can be integrated via existing modules from the repositories of Vert.x or Node.js.

4.2.1 HTTPS

To create a secure application, HTTPS is required. To demonstrate an HTTP and HTTPS server likewise, the application includes two web servers, both of which serve the same functionality. It is necessary to generate a pair consisting of a private and a public key, because the handshake prior to an HTTPS session follows a specific procedure[76]: (1) Client contacts the server using a secured URL. (2) The server responds sending the digital certificate to the browser. (3) The client certifies that the certificate is valid (e.g. by trusted authorities). (4) Once the certificate is validated, the one-time session key generated by the client will be used to encrypt all communication with the server. (6) The client encrypts the session key with the public key of the server. Only the server can decipher the data using the private key.
The generation of private and public keys can be done by using the following commands, leveraging OpenSSL, an Open Source toolkit for implementing secure protocols[77]:

```
1 $ openssl genrsa -out privatekey.pem 1024
2 $ openssl req -new -key privatekey.pem -out certrequest.csr
3 $ openssl x509 -req -in certrequest.csr -signkey privatekey.pem -out certificate.pem
```

Lst. 4: Generating a new pair of public/private keys

4.2.2 AJAX - Asynchronous JavaScript and XML

AJAX applies the previously explained concept of asynchronous communication to the client-side, meaning that the user who sends a request from a website does not have to wait for the answer, but can continue using that website in the meantime without any obvious loading activity that deters him/her from consuming content.[78] In terms of sending data to the server, it usually goes back to the *XMLHttpRequest* object in JavaScript that is called by most browsers to send data to the server (further developments may apply). Receiving information in response to a request is realized using callback functions (JavaScript), whose principle was introduced too.[79] The Proof of Concept primarily uses jQuery as library for AJAX, which "simplifies [...] event handling, animating, and Ajax interactions for rapid web development"[80]. This in particular ensures better compatibility across several platforms, as it covers their specific requirements. The following code snippet in listing 5 only demonstrates briefly an exemplary call, because client-side asynchronous calls are beyond the scope of this paper.

[76] Cf. Nemati, H. R./Yang, L. (2011)
[77] See `http://www.openssl.org/`
[78] Cf. Riordan, R. M. (2008), p. 46
[79] Cf. ibid., p. 21
[80] Cf. w. a. (w. y.)

```
$.ajax({
    url: '/testurl',
    type: 'POST',
    contentType: "application/json; charset=utf-8",
    data: {"testkey" : "testvalue"},
    success: function(response) {
        // success actions, the response object contains the information sent from
            the server
        alert('Everything worked out!');
    }
});
```

Lst. 5: Exemplary jQuery AJAX call

4.2.3 WebSockets

Moreover, a key technology to allow for real-time communication that is leveraged in the Proof of Concept is the WebSocket Protocol. The WebSocket Protocol is standardized by the Internet Engineering Task Force (IETF) in Request for Comments (RFC) 6455[81]. "The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code."[82] The protocol consists of two parts – a handshake and the data transfer, where the handshake is compatible with the HTTP-based server-side software and intermediaries. The data transfer can start as soon as server and client both have sent their handshakes and the actual handshake was successful. Each side can then send data independently from each other in a two-way communication channel.[83]

4.3 Software implementation

4.3.1 Node.js

The Node.js implementation starts with four modules that are required to run the application, as shown in Lst. 6:

```
var express = require('express');
var http = require('http');
var https = require('https');
var fs = require('fs');
```

Lst. 6: Including modules

The HTTP and HTTPS modules are necessary to set up the web servers accordingly. In addition, the “fs” (FileSystem[84]) module is necessary to read the encryption keys for HTTPS. The web framework *Express* offers complementary functionality for a web server, like routing,

[81] Cf. Fette, I./Melnikov, A. (2011)
[82] . ibid.
[83] Cf. ibid.
[84] See `http://nodejs.org/api/fs.html`

and will therefore be used for convenience purposes in this PoC, as recommended by Roden, G. (2012).

Express can be installed using the command-line command shown in Lst. 7:

```
$ npm install express
```

Lst. 7: Installing Express via command-line

In order to enable HTTPS by leveraging the Node.js class `https.Server`, the `privatekey.pem` and `certificate.pem` files need to be read and their content can be written into the `httpsOptions` array by using the *FileSystem* module and its function `readFileSync()`. The files in Lst. 8 are located in the same location as the application file.

```
var httpsOptions = {
    key: fs.readFileSync(__dirname + '/privatekey.pem'),
    cert: fs.readFileSync(__dirname + '/certificate.pem')
};
```

Lst. 8: Reading encryption keys

To run the HTTP/HTTPS servers, the code in Lst. 9 is used, which first initializes Express (`express()`) and then creates the server using the express variable. Afterwards, the ports on which the servers are listening are defined.

```
var app = express();
var server = http.createServer(app);
var secureServer = https.createServer(httpsOptions, app);
server.listen(8888);
secureServer.listen(4431);
```

Lst. 9: Starting servers in Node.js

The HTTP server can be accessed using `http://localhost:8888/` as opposed to the HTTPS server that is available at `https://localhost:4431/`. Afterwards, the *socket.io* module for WebSockets comes into play. First of all, it needs to be installed using the command in Lst. 10.

```
$ npm install socket.io
```

Lst. 10: Command to install socket.io via Node.js' package manager

Afterwards, it is included and initialized by passing the secureServer variable (for the HTTPS server), as shown in Lst. 11.

```
var io = require('socket.io').listen(secureServer);
```

Lst. 11: Initialization of the socket.io module in Node.js

The first Express functionality used is that static files can be served with Node.js. Using the statement in Lst. 12, the directory "/" is made the root directory for a `http[s]://localhost:[port]/`.

```
app.use(express.static(__dirname + '/UI'));
```

Lst. 12: Serving static assets with Express

Another Express feature is the *bodyParser*, which is used as middleware to parse request bodies, for this Proof of Concept especially JSON. It is initialized as shown in Lst. 13.

```
1  app.configure(function(){
2      app.use(express.bodyParser());
3  });
```

Lst. 13: Using the bodyParser

Moreover, Express is used to catch requests to specific URLs. As the AJAX call coming from the user interface is set up as POST-request, the following code in Lst. 14 shows how to catch this (AJAX-driven) POST-request in Node.js on http[s]://localhost:[port]/insurances:

```
1  app.post('/insurances', function (req, res) {
2    // iterating through the array that contains JSON-objects
3    // write the value of the name attribute of each JSON-object into the console
4    for(var i = 0; i < req.body.length; i++) {
5            console.log(req.body[i].name);
6    }
7
8    res.contentType("text/plain");
9    res.send(200, OK)
10 }
```

Lst. 14: Iteration through an array consisting of JSON data

As the AJAX POST-request contains an array filled with JSON-objects, this example demonstrates how to access each item. Within the PoC, this for-loop is used to apply regular expressions in order to validate each form field. `req.body[i].name` is an example on how the Express *bodyParser* is leveraged.

So far, this code provides a web server and is able to respond to requests via HTTP and HTTPS. This would be fine to do server-side validation and saving data in a database, which is a fair use case but would only demonstrate the strength of Node.js if real-time server-side validation (with low CPU workload) was a key criterion. However, this Proof of Concept goes further and demonstrates the strength of Node.js when it comes to real-time communication among several clients by distributing form inputs in real-time to all clients that have opened one specific form at the same time. The technology behind is called WebSockets, as explained in section 4.2.3. The code in Lst. 15 is necessary on the server-side to enable Node.js in combination with *socket.io* to handle WebSocket connections:

The first line refers to the "connection" event, which is triggered by default when a client connects to a server. The parameter of the callback function is the socket that is used for the communication with the client.[85] Line four shows the "on" function of the socket, which refers to the socket object *createForm*. This one is used to write the room's name into the console (line five), write the room's name into the socket session of a client (line six) and finally join the room (line seven). A room is a functionality of *socket.io* allowing "closed" areas.[86] Starting at line

[85] Cf. Roden, G. (2012), p. 197

[86] See https://github.com/LearnBoost/socket.io/wiki/Rooms

```
1  io.sockets.on('connection', function (socket) {
2
3      // create new form
4      socket.on('createForm', function (data) {
5          console.log('room:' + data);
6          socket.room = data;
7          socket.join(data);
8      });
9
10     socket.on('liveform', function (data) {
11         console.log(data);
12         io.sockets.in(socket.room).emit('liveform', data);
13     });
14
15 });
```

Lst. 15: Using WebSockets to distribute form inputs in real-time

10, the actual "live form" functionality begins. Whereas the *createForm* function was used to establish sort of "closed areas" for the individual forms, the *liveform* socket actually distributes data to the clients that have joined one form (i.e. room). Line 12 demonstrates a broadcast of data to all clients that have joined a room.

As the client-side here is specific to socket.io, this part will be explained as well to provide a comprehensive understanding of how WebSockets are used. First of all, the JavaScript library *socket.io.js* needs to be imported as shown in Lst. 16.

```
1  <script src="/socket.io/socket.io.js"></script>
```

Lst. 16: Including the client-side socket.io JavaScript

Afterwards, a connection is established to *https://localhost:4431/* using the code in Lst. 17:

```
1  var socket = io.connect('https://localhost:4431/');
```

Lst. 17: Establishing socket.io connection from the client-side

Having established a connection, the code in Lst. 18 refers to the variable *formId*, which was previously filled with the query string "formId" or left null in case there was none provided. In case it was left null, the user is asked (line two) for an ID. In Case it is there, the ID will be sent (line four) and as the code above shows, the ID is used as a room's name.

```
1  if (formId == null) {
2    socket.emit('createForm', prompt('What is your ID?'));
3  } else {
4    socket.emit('createForm', formId);
5  }
```

Lst. 18: Joining a specific area – "room"

Up to this stage, the process of establishing a connection that enables interaction within a specific area was explained. The code in Lst. 19 demonstrates how to send data to the server using WebSockets. As the form that is used in the Proof of Concept has INPUT fields as well as SELECT fields, it is necessary to distinguish between these both: Whereas the value of INPUT

```
1  function sendToServer(fieldId) {
2      var JSONdata = {};
3      // write JSON, depending on INPUT or SELECT field
4      if (document.getElementById(fieldId).tagName == "SELECT") {
5          JSONdata[fieldId] = document.getElementById(fieldId).selectedIndex;
6      } else {
7          JSONdata[fieldId] = document.getElementById(fieldId).value;
8      }
9      socket.emit("liveform", JSONdata);
10 }
11
12 // trigger request for each key written in input/select fields
13 $('input[type="text"]').on('keyup', function () {
14     sendToServer($(this).attr("id"));
15 });
16 $("select").on("change", function () {
17     sendToServer($(this).attr("id"));
18 });
```

Lst. 19: Sending data to the server

fields can be read via *[element].value* (line seven), the selected value of a SELECT field must be read by [element].selectedIndex (line five). However, within the JSON, both can be saved the same way. Line nine shows how to send data to the server using the *liveform* socket and the JSON data, which includes one field's value. This sendToServer(fieldId) function is called whenever the user changes the form, meaning with every letter the user types or with every SELECT field the user changes (lines 12 to 18)

On the other hand, it is for sure necessary to receive data on the client-side and populate the form fields appropriately. The code in Lst. 20 demonstrates how this is realized. Line one shows how to "listen" on a socket, in this case *liveform*. The for-each loop starting at line three is just a simplified way to read the JSON (which only has one data item), because the name of the key used in that data item is unknown. Again, the distinction between INPUT and SELECT fields needs to be taken into account as explained above, which is why the IF on line five reflects that issue. Lines seven and 10 write the data received in the JSON into the form fields. In case of the INPUT fields "SummeInput", "BehaltInput" and "BerufInput", there is the possibility that the calculation of the insurance rate needs to be triggered using function f_req(), which is why lines 12 to 14 check for these INPUT fields. Finally, line 17 triggers the validation of the INPUT fields using the jQuery plugin h5Validate, which is primarily important to create the same visual effects on every client's form.

To use the application, the Node.js servers can be started by using the command-line. After navigating into the project's folder using cd, the only command needed is shown in Lst. 21:

```
1 $ node [filename].js
```

Lst. 21: Executing Node.js code

```
1  socket.on("liveform", function (data) {
2      // read (unknown) field from JSON
3      for (key in data) {
4          // distinction between select and input fields
5          if (document.getElementById(key).tagName == "SELECT") {
6              // write content into field
7              document.getElementById(key).selectedIndex = data[key];
8          } else {
9              // write content into field
10             document.getElementById(key).value = data[key];
11             // also trigger the price calculation when these fields are filled
12             if (key == "SummeInput" || key == "BehaltInput" || key == "BerufInput")
                   {
13             f_req();
14             }
15         }
16         // triqgger validation to have smiliar visual appearances (valid/invalid)
               for the fields
17         $('#' + key).h5Validate("isValid");
18     }
19 });
```

Lst. 20: Receiving data using WebSockets

4.3.2 Vert.x

The Vert.x application is entirely written in Java, but it is also possible to write each verticle in a different language if desired. This could be done to maximize code reuse between front-end and back-end.

The business logic of the back-end consists of three verticles and additional utility classes. One verticle serves as a web server that accepts HTTP and HTTPS requests. GET requests are interpreted as file requests and are answered directly from within the verticle. POST requests are interpreted as form submissions. Whenever a POST requests arrives, the verticle transforms the request body to a JSON object and submits it via Vert.x' event bus.

The second verticle is a worker verticle which performs the CPU intense calculation tasks. This verticle listens on the event bus for incoming messages. Whenever such a message arrives it transforms the received JSON object into a form instance for validation and fee calculation. The results are then sent back via the event bus when this operation has been completed, so that the event loop can return the result to the client in its next iteration.

The third verticle serves as a starter script which programmatically starts the previously mentioned verticles and terminates afterwards as it does not define any callbacks itself (see Lst. 22).

The actual deployment of the other verticles happen in lines 13 and 14. An important detail is the second parameter in the `deployWorkerVerticle` method. Vert.x allows deploying multiple instances of a verticle programmatically which eases scaling an application to the number of available processors of a machine. In this case the main loop is single-threaded and for each remaining processor there is one worker verticle. Vert.x automatically distributes requests among these worker verticles (as described in section 2.5.3).

```
1  package main;
2  import org.vertx.java.core.SimpleHandler;
3  import org.vertx.java.core.json.JsonObject;
4  import org.vertx.java.deploy.Verticle;
5
6  public class AppStarter extends Verticle {
7    private static final String HTTP_CONF = "http_conf";
8    private static final int NUM_PROCESSORS = Runtime.getRuntime().availableProcessors
         ();
9
10   @Override
11   public void start() throws Exception {
12     JsonObject httpConfig = appConfig.getObject(HTTP_CONF);
13     container.deployVerticle("main.Server", httpConfig);
14     container.deployWorkerVerticle("form.Calculator", NUM_PROCESSORS - 1);
15   }
16 }
```

Lst. 22: Verticle that starts two other verticles

A global configuration file is read in line 12. This configuration file is formatted as JSON string and can include settings for modules or verticles. The path to the configuration file has to be provided to the `vertx` command upon server start to allow access to it from within the verticles. The command that has to be invoked for the Proof of Concept is shown in Lst. 23.

```
1  $ vertx run main.AppStarter -conf app_conf.json -cp bin
```

Lst. 23: Command for starting the Vert.x application

Vert.x allows either running verticles by providing the compiled Java class or by providing the source file, which is then automatically compiled by Vert.x. However, in this setup we experienced some issues with the Vert.x `run` command when we tried to use it with the source files. The issues observed were partially related to Vert.x bug 444[87]. Due to incomplete dependency resolution, not all classes were compiled so that Vert.x could not find all required classes in the path at runtime. These startup issues can easily be avoided by using compiled classes instead of the source files when multiple verticles need to be started programmatically. The path to the compiled classes has to be provided with the `cp` switch as shown in Lst. 23.

[87] see `https://github.com/vert-x/Vert.x/issues/444`

```
1  /**
2   * Triggers the fee calculation based on the given form data.
3   */
4  private Handler<HttpServerRequest> calcHandler = new Handler<HttpServerRequest>() {
5    public void handle(final HttpServerRequest req) {
6      // read body of request
7      req.bodyHandler(new Handler<Buffer>() {
8        @Override
9        public void handle(Buffer buff) {
10         JsonObject form = new JsonObject(buff.toString());
11         // trigger calculation in worker verticle
12         eBus.send("form.calculate", form,
13         new Handler<Message<JsonObject>>() {
14           public void handle(Message<JsonObject> message) {
15             req.response.headers().put("Content-Type", "application/json; charset=
                   utf-8");
16             req.response.end(message.body.encode());
17           }
18         });
19       }
20     });
21   }
22 };
```

Lst. 24: Vert.x form handler

```
1  public class Calculator extends Verticle {
2    /**
3     * Instantiates an insuranceForm and calculates the fee based on the forms data.
4     * The result is sent back via the event bus.
5     */
6    private Handler<Message<JsonObject>> calculationHandler = new Handler<Message<
         JsonObject>>() {
7      public void handle(Message<JsonObject> message) {
8              InsuranceForm form = new InsuranceForm(message.body);
9              JsonObject fee = new JsonObject();
10             fee.putNumber("price", form.getInsuranceFee());
11             message.reply(fee);
12         }
13     };
14
15   @Override
16   public void start() throws Exception {
17     final EventBus eBus = vertx.eventBus();
18         eBus.registerHandler("form.calculate", calculationHandler);
19   }
20 }
```

Lst. 25: Vert.x worker verticle that receives the message sent by the form handler

The implementation of the communication between the web server and the worker verticle was rather easy by using the event bus. The event bus has a very simple adressing mechanism where the address consists of a normal string. Listing 24 shows the form handler, which is defined in the web server verticle and gets called whenever a form submission arrives. The handler registers another handler that is called as soon as the entire request body has arrived (line 7). The data that arrives as a buffer is than turned into a JSON Object (line 10) and submitted to the event bus (line 12).

As soon as a reply is returned over the event bus it is forwarded to the client line 16). Lst. 25 shows the worker verticle that receives the message, does the calculation and then returns the result via the message bus.

The live communication via the WebSocket Protocol is handled in a similar way. As mentioned in section 2.5.3, Vert.x allows access to its event bus from within the client's browser. This is accomplished by using a so-called *bridge* that forwards client messages to the server event bus and vice versa. As this poses a potential security vulnerability it is recommended to set up the bridge with a filtering instance that only allows certain addresses or message contents for inbound or outbound communication. Lst. 26 contains the configuration that has been used for the real-time communication between clients and server in the Vert.x implementation. Inbound messages are allowed on the address "form.data" and outbound messages are allowed on all addresses that start with "form.data.client." and end with a number. The variable number is used as the form key for grouping clients, as described in section 4.1. Note that this is still an insecure solution as attackers could register theirselves with a large number of form keys to receive all data that is being sent. Possible ways to address this issue are either encrypting submitted data or by allowing connections for authorized clients only.

```
1  HttpServer server = vertx.createHttpServer()
2  JsonObject bridgeConfig = new JsonObject().putString("prefix", "/eventbus");
3
4  JsonArray inboundPermitted = new JsonArray();
5  JsonArray outboundPermitted = new JsonArray();
6
7  // Let through messages for data synchronization
8  JsonObject addressData = new JsonObject().putString("address", "form.data");
9  inboundPermitted.add(addressData);
10
11 // Allow sending messages to all addresses that match this pattern
12 new JsonObject().putString("address_re", "form.data.client\\.\\d+")
13 outboundPermitted.add();
14
15 vertx.createSockJSServer(server).bridge(bridgeConfig, inboundPermitted,
       outboundPermitted);
```

Lst. 26: Vert.x event bus bridge configuration

5 Evaluation of Non-functional Attributes

5.1 Maintainability

Maintainability is one of the most important issues in the enterprise as developers typically switch between projects (or participate only in a temporary manner). Thus the time needed to become familiar with a framework and its language directly impact the development cost either because of the time needed related to the learning curve or because of resources invested in maintenance and bug-fixing.

	Node.js	**Vert.x**
API stability	Changes frequently[88]	Changes rarely[89]
Support of dated versions	No official policy known	No official policy known
Testing	Has good test coverage. Various 3rd party test frameworks applicable.[90]	Has good test coverage and comes with an own testsuite.[91] Established test libraries like JUnit can be used.
Skill transferability	Coding style is similar to established JavaScript guidelines.[92]	Depends on the used language. Java coding style in Vert.x e.g. differs from J2EE practices while JavaScript style is similar to established JavaScript frameworks.
Official documentation	API documentation, manual[93]	API documentation, manual[94], basic tutorial, code examples[95]
Additional documentation	Large number of books and blog articles	Few blog articles, users group
License	MIT license	Apache Software License 2.0

Table 2: Maintainability comparison of Node.js and Vert.x

[88] Cf. Joyent Inc (2012)
[89] Cf. w. a. (2012d)
[90] Cf. Joyent Inc (2013)
[91] Cf. Fox, T. (2013b)
[92] Cf. McMahon, C. (2012)
[93] Cf. Joyent Inc (2012a)
[94] Cf. w. a. (2012d)

Table 2 gives a quick overview over the differences between Node.js and Vert.x in this context. For both,Vert.x and Node.js there is no official policy how security issues or defects in dated versions are handled, so that upgrading to the latest stable versions is recommended for both frameworks. API changes need to be checked with every upgrade. Further framework-specific details are provided in the following sub-sections.

5.1.1 Node.js maintainability

Node.js allows for leveraging the JavaScript skills already in the market which significantly lowers the Time to Market (TTM) due to its similarity to established libraries and conventions in web application development. JavaScript's lack of blocking I/O libraries allowed the project founder to establish a non-blocking standard for I/O operations.[96] Unfortunately TTM is not everything because equal attention needs to be given to Total Cost of Ownership (TCO), i.e. the total expenditure required including costs incurred post-deployment[97]. Here Node.js presents a challenge in its reliance on callback leading to hard-to-follow and hard-to-fix code driving up TCO significantly because patches become more and more difficult to write with increasing $\Delta t_{Launch<->Present}$, especially if the code is not to be maintained by the original developers.

Furthermore, Node.js is still at an unstable stage of code maturity (at time of writing, the current version is $0.9.7$[98]) causing infrequent breaking changes to its APIs, which usually is a no-go in the enterprise. The API documentation distinguishes between five stability states (deprecated, experimental, unstable, stable, API frozen, locked), whereas the latter ones are rarely found in the documented features.[99]. This is reinforced by a community mindset of 'Upgrade Node, Update Code' based on a preference for new features. The unintended benefit of this is a lot of available material for Node.js programmers in blogs and on StackOverflow[100]. This leads to the conclusion that the current pace of progress does not facilitate enterprise adoption, although it is not a showstopper.

5.1.2 Vert.x maintainability

Vert.x offers a similar picture to Node.js, however some of Node.s's shortcomings are more pronounced on the Vert.x side of things. Whereas Node.js can recruit from established web developers, is Vert.x suffering from a curious conundrum. It is based on the JVM, which has enormous amounts of talent (especially in the enterprise) but because of its diametrical dif-

[95] Cf. Fox, T. (2013a)
[96] Cf. Hughes-Croucher, T./Wilson, M. (2012), p.10 et sqq.
[97] Cf. Holtsnider, B./Jaffe, B. D. (2010), pp. 203-207
[98] Cf. w. a. (2013a)
[99] Cf. Joyent Inc (2012a)
[100] See `www.stackoverflow.com`

ferences to the conventional libraries it does not seem to be able to recruit interest from this pool. Also its support for multiple programming languages (basically every JVM language) should lower the barrier for entry even more, as for example Groovy already features Vert.x-aligned language constructs. Unfortunately this means that new Vert.x programmers need to be trained, which increases TTM significantly.
Another important issue is the unstable state of the Vert.x codebase. Due to the rather small community, issues are often detected after new versions are considered final. At the time of writing there are currently 90 open issues on the project site. Section 2.5.1 already mentioned the ease of calling synchronous code with Vert.x – although this can be worked around with by placing the code in new verticles, this still requires a lot of vigilance.

TCO is difficult to estimate for Vert.x due to a lack of success/failure stories, it can be said though that because Vert.x requires Java hosting infrastructure there might be an associated licensing cost for application servers.

At present there is a discussion in the Google users group [101] considering the future of the Vert.x project which has been owned by VMWare. The founder of the project and former employeee of VMWare had to transfer ownership of the GitHub project, domain, Gooole group and blog to VMWare after he left the company in December 2012[102]. This turn of events has put uncertainty in the community and might have increased the entry barriers for new developers. The outcome of this discussion has been that Vert.x should be submitted to the Eclipse Foundation.[103].

5.2 Integration

Vert.x and Node.js are supposed to be used as fully event-driven standalone applications, which can be extended with event-driven modules. However, when introducing a Node.js or Vert.x application into the current application landscape, it might be desired to integrate or communicate with existing systems that are not fully event-driven. Furthermore, integration of these technologies might not be limited to mere IPC but instead could establish a new middleware-layer between current and new applications.

5.2.1 Node.js Integration

Node.js is not designed to handle CPU-intensive tasks efficiently. However, there is a way a Node.js process can perform such tasks without impairing the application performance. Node.js uses so-called child processes for this (provided by the *child process* module). That module

[101] See `https://groups.google.com/forum/#!forum/vertx`
[102] Cf. Fox, T./Daryl, T., et al. (2013)
[103] Cf. Fox, T./Teo, D., et al. (2013)

```
1  var spawn = require('child_process').spawn,
2      ls    = spawn('ls', ['-lh', '/usr']);
3
4  ls.stdout.on('data', function (data) {
5    console.log('stdout: ' + data);
6  });
7
8  ls.stderr.on('data', function (data) {
9    console.log('stderr: ' + data);
10 });
11
12 ls.on('exit', function (code) {
13   console.log('child process exited with code ' + code);
14 });
```

Lst. 27: Example of running *ls -lh /usr* in Node.js, capturing stdout, stderr, and the exit code

is basically a wrapper around the unix tool *popen* and provides access to a child process' communication streams (stdin, stdout, stderr).[104]

There are two cases child processes are used for:

First, CPU-intensive tasks can be performed outside Node.js by assigning them to a different process (which is then called a child process) in order not to block the event loop. The output data from the child process is then sent back to the parent process.[105] In this case, the child process is used to outsource a task that requires high computation work and would otherwise block the event loop. This approach, however, requires routines running within the child process to be written in JavaScript as well, so that this is not usable to properly connect an existing application with the Node.js application.

A second way of using the child process module is to actually run external commands, scripts, files and other utilities that cannot directly be executed inside Node.[106]

This characteristic lets external processes get well integrated with Node.js. A basic example of the child process module is shown in Lst. 27[107]. The child process instance that is created in lines 1 and 2 is an event emitter that allows registering callbacks for certain events (see lines 4,8 and 12).

Invoking processes on a different machine across the network still requires a remote API or a distributed message bus system.

Moreover, Kue is an example of a message queue developed in Node.js, which could serve as connector to the enterprise service bus. Appendix 2 expands upon this module.

[104] Cf. Joyent Inc (w. y.[a])
[105] Cf. Teixeira, P. (2013), p. 63
[106] Cf. ibid., p. 63
[107] Taken from Joyent Inc (w. y.[a])

5.2.2 Vert.x Integration

In Vert.x there are multiple ways to communicate with external programs. Depending on the language used there are already multiple libraries that can be used to invoke child processes. However, these libraries usually only expose synchronous APIs, so that these calls need to be done inside a worker verticle that does not block the event loop.
Listing 28 shows a minimal worker verticle written in Java that invokes *ls -lh /usr* via the *Apache Commons Exec* library[108] in a blocking way. This worker verticle runs the command whenever it receives a message via the event bus. The command to run could also be provided via the JSON object that is passed into the verticle via the event bus for higher flexibility. A watchdog (instantiated in line 27) is used to terminate the process after six seconds. Otherwise a runaway process could block the worker thread so that it cannot be used anymore[109].

This approach might not be very satisfying, as the output is not captured and execution errors aren't handled properly. Luckily the *Apache Commons Exec* library can be used in an event -driven style as well. To achieve this one first needs to write a class that handles the output streams. An exemplary implementation of such a class is provided in Lst. 29.

An instance of this class can then be provided to the executor as shown in listing 30, where the instance gets passed into a PumpStreamHandler (line 21)

```
1  CommandLine cmdLine = new CommandLine("ls");
2  cmdLine.addArgument("-lh");
3  cmdLine.addArgument("/usr");
4
5  DefaultExecuteResultHandler resultHandler = new DefaultExecuteResultHandler(){
6    public void onProcessComplete(int exitValue){
7      logger.info("Process terminated normally");
8    }
9
10   public void onProcessFailed(ExecuteException e){
11     logger.info("Process failed");
12   }
13 };
14
15 StdoutStream stdout = new StdoutStream();
16
17 ExecuteWatchdog watchdog = new ExecuteWatchdog(60*1000); //timeout after 6 seconds
18 DefaultExecutor executor = new DefaultExecutor();
19 executor.setStreamHandler(new PumpStreamHandler(stdout, null, null));
20 executor.setExitValue(0);
21 executor.setWatchdog(watchdog);
22 try {
23   executor.execute(cmdLine, resultHandler);
24 } catch (IOException e1) {
25   e1.printStackTrace();
26 }
```

Lst. 30: Asynchronous process execution in Java

[108] See `http://commons.apache.org/exec/`
[109] Cf. Nemeth, E./Snyder, G./Hein, T. R. (2007), p. 67

```
1  import java.io.IOException;
2  import org.apache.commons.exec.CommandLine;
3  import org.apache.commons.exec.DefaultExecutor;
4  import org.apache.commons.exec.ExecuteWatchdog;
5  import org.vertx.java.core.Handler;
6  import org.vertx.java.core.eventbus.EventBus;
7  import org.vertx.java.core.eventbus.Message;
8  import org.vertx.java.core.json.JsonObject;
9  import org.vertx.java.core.logging.Logger;
10 import org.vertx.java.deploy.Verticle;
11
12 public class CommandExecutor extends Verticle {
13   Logger logger = container.getLogger();
14
15   @Override
16   public void start(){
17     final EventBus eBus = vertx.eventBus();
18
19         eBus.registerHandler("cmd.execute", new Handler<Message<JsonObject>>() {
20
21       public void handle(Message<JsonObject> message) {
22         CommandLine cmdLine = new CommandLine("ls");
23         cmdLine.addArgument("-lh");
24         cmdLine.addArgument("/usr");
25         //timeout after 6 seconds
26         ExecuteWatchdog watchdog = new ExecuteWatchdog(60*1000);
27         DefaultExecutor executor = new DefaultExecutor();
28         executor.setWatchdog(watchdog);
29         try {
30           executor.execute(cmdLine);
31         } catch (IOException e) {
32           e.printStackTrace();
33         }
34             }
35         });
36   }
37 }
```

Lst. 28: Example of running *ls -lh /usr* in Vert.x from within a worker verticle

```
1  import java.util.LinkedList;
2  import java.util.List;
3  import org.apache.commons.exec.LogOutputStream;
4
5  public class StdoutStream extends LogOutputStream {
6    @Override
7    protected void processLine(String line, int level) {
8      System.out.println(line);
9    }
10 }
```

Lst. 29: Subclass of LogOutputStream as a handler for the stdout stream

There are some more noteworthy details in Lst. 30. The `DefaultExecuteResultHandler`, which is defined in lines 5 to 13, can be used to provide callbacks for successfull termination or failure of the external process. The call to `executor.execute` (line 23) automatically becomes asynchronous when a `StreamHandler` is set for the executor (line 19)[110].

When writing verticles in Java it is also possible to make use of Javas Remote Method Invocation (RMI) mechanism to communicate with services that are on different machines in the network. The most natural way for inter-application communication is by either using the integrated message bus or by using a separate bus system. Vert.x can be embedded in any JVM based program by including the according libraries in the path, so that these applications can use the bus system as well.[111]

5.3 Scalability and Performance

Both Vert.x and Node.js offer great performance and scalability under certain circumstances that were identified in section 3. In order to allow a comparison for a very simple use case, a micro-benchmark was run to collect information on request handling of Node.js, Vert.x and the two popular web servers Apache2 and Nginx.

Experiment setup Each server was run isolated on a virtual machine that was assigned 4GB of memory and 4 of 8 available processors with a limited processor rate of 2.5GHz (to make sure that the host system has enough free capacity). A static file with 200kb has been created using `dd`, which then had to be delivered by the web servers.

Requests were then sent from another virtual machine on the same host system to the running server using the *Apache HTTP server benchmarking tool*[112]. The benchmarking tool was configured to send 10,000 requests with a concurrency level of 50 (50 requests in parallel).
Vert.x and Node.js were run single-threaded. Apache2[113] and Nginx[114] were run in their default configuration. All servers were run without caching.

Outcome The results of this benchmark confirmed the strengths of asynchronous servers described in section 3. Plotted results are shown in figures 10 and 11.
Apache2 had the worst respone times, that were increasing rapidly with the number of requests sent. Nginx was able to deliver an almost constant response time for up to 5000 requests before response times started to increase. Interesting is the fact, that Nginx's response times where slightly below those of Node.js and Vert.x for up to 3000 requests. Both Node.js and Vert.x

[110] Cf. The Apache Software Foundation (2010)
[111] Cf. w. a. (2012d)
[112] See `http://httpd.apache.org/docs/2.2/programs/ab.html`
[113] See `http://httpd.apache.org/`
[114] See `http://nginx.org/en/`

were able to deliver the file in almost constant time for up to 9500 requests. Node.js performed slightly better than Vert.x. A partial plot is shown in figure 11 for better comparability between Node.js and Vert.x.

Although not being a representation for a real usage scenario, this benchmark gives a good representation of the scalability attributes of asynchronous servers. The bad performance of Apache2 is due to the process forking for each new request. Nginx is a lightweight HTTP server that by default only uses as many threads as there are processors provided by the host, which is why it performs so well. More detailed benchmarking results are shown in Appendix 8.

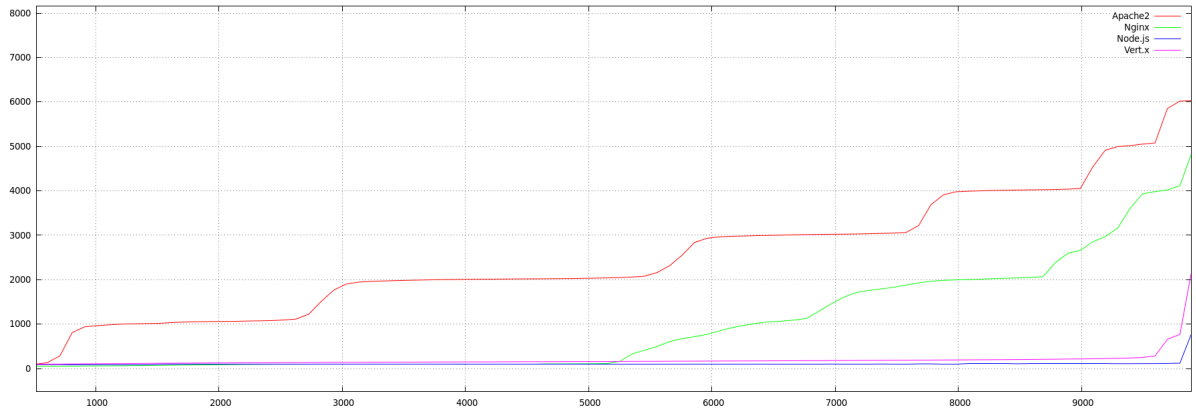


Fig. 10: Plot of the web server benchmark results between Apache2, Nginx, Node.js, Vert.x (x-axis: requests, y-axis: time in ms)

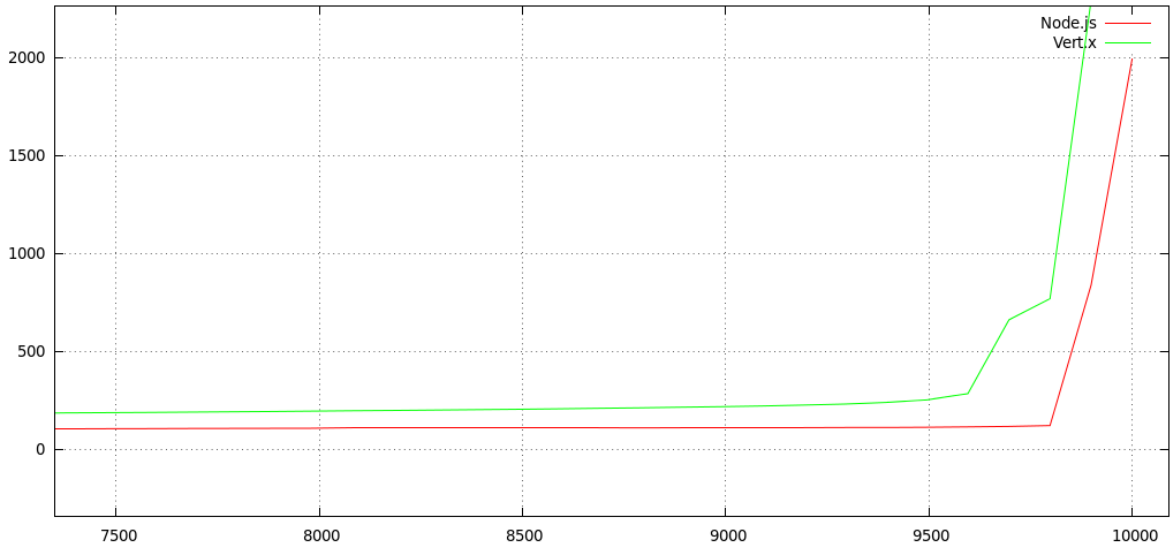


Fig. 11: Plot of the web server benchmark results between Node.js and Vert.x (x-axis: requests, y-axis: time in ms)

Scaling on a single machine Support for single machine scaling using multiple threads exists natively in Vert.x and can be accomplished easily on a per-instance basis like shown in

Lst. 22. Each instance can be deployed multiple times. This allows configurations like having one instance for the main event loop and 4 instances for the worker verticles, so that computation intense tasks can be handled by a thread pool of 4 threads. Vert.x provides thread-safe data-structures that developers can use to share data between instances. However these data-structures are limited to immutable data types[115]

Node.js is designed to run single-threaded and can only scale to all available cores by using the Cluster module[116], which is still in an experimental state. This module allows to take advantage of a multi-core system by launching a cluster of Node.js processes to handle the load. Basically it enables the developer to run several worker processes (depending on the number of cores) and create an HTTP server for each of them, whereby all of them share one port.

Scaling on distributed machines Vert.x uses Hazelcast's[117] clustering functionality internally to scale on multiple machines.

```
1 $ vertx run server.js -cluster -cluster-port 22000
```

Lst. 31: Starting a Vert.x application in cluster mode

Listing 31 shows how a Vert.x cluster can be set up on a network with the default configuration. The cluster-port argument specifies on which port distributed instances will communicate with each other and defaults to 25500 when omitted. Each instance that is started in cluster mode will have access to a distributed event bus.[118] Implementation details on the cluster internals can be found in the Hazelcast documentation on the project's website.

Up to now, for Node.js there is no known way of distributing workloads among different servers, however that wish was addressed.[119]

General ways of distributing workload among distributed machines One possible procedure is to deploy an application multiple times and to use an additional (external) load balancer that distributes incoming requests among all running deployments. This has some implications: Due to the isolation in separate processes, application instances cannot share any data or communicate directly with each other. Furthermore, data might be stored in a physically separated database like CouchDB[120] or in a key-value store like Redis[121]. This database could then po-

[115] Cf. w. a. (2012d)
[116] Cf. Joyent Inc (2012b)
[117] See `http://www.hazelcast.com/`
[118] Cf. w. a. (2012d)
[119] Cf. McCalla, J. (2011)
[120] See `http://couchdb.apache.org/`
[121] See `http://redis.io/`

tentially become a bottleneck when multiple application processes require access, depending on its ability to provide parallel processing.

Speeding up single requests Node.js and Vert.x do not offer any possibility to speed up a single request. These requests will aways run at the same speed. What these frameworks do is to maximize the number of possible requests while keeping the processing speed steady. This is why it scales so well (see section 5.3). The only additional delay between request and response is the time that a request waits in the event loop until it gets processed. In many cases however it is desirable to minimize the computation time itself for a single request. Once this is achieved it becomes a goal to keep that speed for a larger amount of requests.

6 Outlook

This section is going to cover future prospects of the two technologies that were subject to investigation within this paper.

Node.js Node.js, as became evident throughout the paper, is popular and enjoys healthy development. It is backed by a variety of modules that enable developers to implement complex functionality. The packaged modules available at `npmjs.org` indicate great interest, as the number of downloads (about 14.6 million last month[122]) suggests. In addition, the GitHub Commit History[123] shows constant contributions.

Vert.x Recently there has been some uncertainty concerning the future development of Vert.x, as recent news articles indicate.[124] This is mainly related to the reaction of VMware when Vert.x foudner Tim Fox left the company. Moreover, there is currently no developer roadmap available that allows to infer the future development. Its commit history on GitHub[125] is another sign of a decreasing popularity, which, however has always been on a lower level than on GitHub but might currently be related to the unclear future. It is likely that the development activity for the Vert.x project starts increasing once it has completed the transition to the Eclipse foundation.

Evolving the Proof of Concept The Proof of Concept introduced within this paper is an initial implementation that was set up to show the potential usage of asynchronous server technologies while remaining on a moderate level of complexity. However, the use case that was shown – real-time insurance form infilling and instant price calculation based on inserted values – provides great potential for companies to evolve their business model. The technology enables salesmen to interact more easily with clients and transforms geographically spread conclusion of contracts to the next stage, because the anonymity of web-based contracts can be mitigated by real-time communication that is not only limited to filling a form, but instead can be extended to live web-conferences with video/audio stream next to the form through the use of asynchronous server technologies.

[122] Cf. Joyent Inc (w. y.[b])
[123] Cf. w. a. (2013b)
[124] Cf. Asay, M. (2013)
[125] Cf. w. a. (2013b)

7 Conclusion

The objective of this paper was to evaluate server-side asynchronous technologies with special focus on enterprise-readiness. This paper addressed this goal by outlining basic technologies in web development and pointing out the different approaches of synchronous and asynchronous communication.

The initial market overview has shown that there is a variety of frameworks for established programming languages available, which, however, are not always in a satisfying state of development. The scope of investigations within this paper was limited to two frameworks, which were evaluated in detail: Node.js and Vert.x.

Node.js While Node.js is clearly dominating the market at present, it also comes with its fair share of issues, especially if reliability and maintainability are concerned. This is not to say that Node.js is unstable and/or unreliable, sections 2.5.2 and 5.1 proved that, but rather that its community philosophy of *"Update Node, Upgrade code."* is diametrically different to the principles of enterprise software deployment.

Vert.x Vert.x certainly has a much stronger position for entering the enterprise market (due to its JVM underpinnings and ease of distributed programming, see section 2.5.3), however its small community and uncertain state of affairs do weaken its chances. Moving to the Eclipse Foundation does alter this once the project has moved as this provides a very strong foundation (pun intended) for the future.

Asynchronous server technologies require careful evaluation of one's needs and wants and in particular implementation goals in order to only use the most capable solution. It is key to bring the actual use cases and inappropriate use cases to mind. Exceptionally good use cases are those with a high number of requests, real-time communciation and (very important) low CPU workload in order to leverage non-blocking I/O streams. Inappropriate use cases for asynchronous technologies are CPU intensive applications (e.g. data warehousing with analytics) or simple CRUD/HTML applications. Moreover, as non-blocking I/O is a key feature, use cases that hardly encompass interaction with external resources do not set a precedent and might even be penalized. It is important to point out that it is not worth using asynchronous technologies for the sake of leveraging single secondary features (like the similarity of client- and server-side programming language in case of Node.js) when the overall application is not a good fit. Computing intensive tasks that are handled in a single thread or a very limited number of threads jeopardize the availability of an application and there should be good reasons to work around such issues instead of using a synchronous approach. It can be a valid approach to implement single functionalities like real-time communication (chats, push-notifications) and

use a synchronous technology for other parts of an application. Depending on the use case and environment, using Node.js or Vert.x both could potentially be beneficial. However, in an enterprise context, there are caveats for both frameworks due to their early stages in development.

In conclusion, it is safe to assume that this kind of technology will evolve and become mature very soon. The Appendices 4 to 7 describe actual implementations that leverage the described strengths. Provided the use cases are wisely chosen, companies can take away great benefits, since (a) there is a lot of programming skill that can easily applied and (b) this new approaches effectively address some of today's IT problems.

Appendices

List of Appendices

Appendix 1: Middleware as use case for Node.js

This appendix is to briefly highlight theoretical as well as practical perspectives worth noting regarding the usage of Node.js and Vert.x as middleware, of which a definition by Geihs, K. can be found in the blockquote.

> "The term middleware refers to the software layer between the operating system—including the basic communication protocols—and the distributed applications that interact via the network. This software infrastructure facilitates the interaction among distributed software modules."[126]

The JVM and its associated libraries building the foundation for Vert.x are a classic example of the concept defined above although most of the currently available JVM-middleware is synchronously implemented. This offers opportunities to leverage the underlying concepts of both frameworks regarding business logic implementation for middleware development. Realization of this potential manifests itself not only in the number of packages available on npm (21,104 as of January 18, 2013)[127] but also in new projects built on top of Node.js.

One issue mentioned when dealing with middleware is the format of messages that are exchanged among distributed systems.[128] As Node.js brings a typical client-side scripting language to the server-side, the developer can take advantage of that fact by using the serialization format JavaScript Object Notation (JSON) by using "application/json" as the response content type.[129] JSON is a common exchange format,e.g. CouchDB is a database storing documents in JSON and could particularly benefit from Node.js as middleware.

Furthermore, sec. 3.1 outlined the potential of Vert.x and Node.js as proxy servers, which is another hint at the potential of these technologies in terms of middleware. Appendices 2, 6 and 3 will showcase what is possible when turning Node.js into a middleware-layer (which fundamentally is also true for Vert.x).

[126] Geihs, K. (2001)
[127] Cf. Joyent Inc (w. y.[b])
[128] Tanenbaum, A. S./Van Steen, M. (2007), p. 149
[129] Cf. Crockford, D. (2006)

Appendix 2: Kue — A Message Queue (Job Queue) based on Node.js

A message queue is "an area where you can add messages representing a task"[130]. According to Microsoft Message Queue (MSMQ) it helps solve two basic problems:[131]

- Unavailability of communications with servers or clients, and
- Communications between disparate systems and components.

Another use case for message queues is storing less-important information in a message queue while your DBMS is busy serving customers during the day.[132] The basic concept behind message queues is as follows: When a message arrives from a client, the producer places it on a queue. It has then done its job and waits for the next input data. The consumer, which can be a process or application that may also be on a different server or even a different operating system[133], can then retrieve messages from the queue for processing (basically, the FIFO principle is applied).[134] A common scenario for message queues is when only a little amount of requests create high loads on the server, e.g. video platforms such as Youtube or Vimeo.[135] In such cases, it is useful to externalize CPU-intensive tasks in order not to impair responsiveness of the application.[136] Asynchronous server technologies such as Node.js can leverage message queues very well. As a feature of the event-driven model, an event handler is triggered when a new message is available. Thus, the queue does not need to be constantly polled for messages.[137]

Kue is an example of a message queue (also called job queue) that is entirely developed in Node.js. It uses redis as a queue server. With Kue, jobs can be prioritized, delayed, monitored graphically and in real-time, repeated if they failed and more. A key feature is the user interface to view and manage jobs in different states (queued, active, failed and completed). Kue can be installed easily according to Listing 32.

```
1 $ npm install kue
```

Lst. 32: Installing Kue via command-line

[130] McGlennon, G. (2007)
[131] Cf. ibid.
[132] Cf. Thomson, C. (2002), p. 449
[133] Cf. McGlennon, G. (2007)
[134] Cf. Thomson, C. (2002), p. 450
[135] Cf. Roden, G. (2012), p. 247
[136] Cf. ibid., p. 247
[137] Cf. Knight, M. (2011)

Appendix 3: Meteor and Opa — Next Generation Web Application Frameworks

Meteor and Opa are two frameworks built on top of Node.js and MongoDB created around a radically new development workflow. Both eliminate the distinction between client-side and server-side code, instead the developer writes code in a very similar way to conventional desktop programming.[138] The difference is that the framework provides functionality to that enables the code to alter its behavior depending on where it runs.[139] An example of this methodology can be found in appendix 3.

Meteor leverages Node.js enabling it to make use of its JavaScript-based concept and the developer to take advantage of the variety of packages available for Node.js. Special attention should be paid to the fact that Meteor server code runs in one thread per request as opposed to the usual asynchronous callback methodology that applies to Node.js itself normally.[140] Through heavy integration of MongoDB (or Meteor's data layer in general) it is also unnecessary to write and handle AJAX calls manually as all data transmission (e.g. synchronization, publish, subscribe) is abstracted away.[141] The Meteor-provided example app 'leaderboard', whose code is provided in listings 33[142] and 34[143] (CSS omitted), highlights this concept very well.

At its current stage it is an ideal prototyping environment for it allows ideas to be brought into usable form very quickly, even including hosting in the cloud. These capabilities are especially useful when creating single-page applications (i.e. JS UI with only one HTML page served). Currently this means that more complex software is much more difficult to build using Meteor especially if it requires serving multiple pages (e.g. public material, user area and administrative area). The POC implemented as part of this paper was originally planned to be built using Meteor, this however had to be shelved as undocumented errors arose. These quickly became unfixable as the error messages provided proved no help in debugging and conventional JS debuggers also were unable to provide assistance. Trying to shoehorn Meteor into something it was not built for, this behavior is not unexpected.

Opa is conceptually similar although its technical underpinnings as well as the developer-used framework is different which is illustrated in fig. 12.[144]

[138] Cf. Schmidt, G. et al. (w. y.[a])
[139] Cf. ibid.
[140] Cf. ibid.
[141] Cf. ibid.
[142] Cf. Schmidt, G. et al. (w. y.[b])
[143] Cf. ibid.
[144] Opa Team (w. y.)

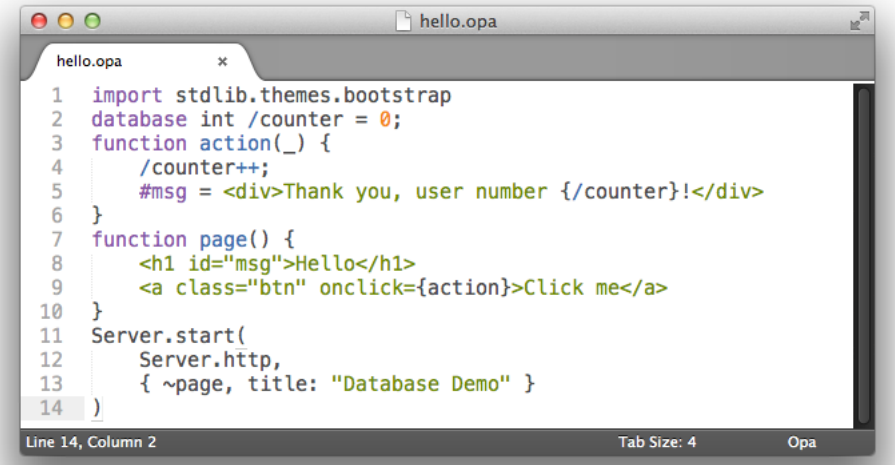


Fig. 12: Opa example code

```
<head>
  <title>Leaderboard</title>
</head>

<body>
  <div id="outer">
    {{> leaderboard}}
  </div>
</body>

<template name="leaderboard">
  <div class="leaderboard">
    {{#each players}}
      {{> player}}
    {{/each}}
  </div>

  {{#if selected_name}}
  <div class="details">
    <div class="name">{{selected_name}}</div>
    <input type="button" class="inc" value="Give 5 points" />
  </div>
  {{/if}}

  {{#unless selected_name}}
  <div class="none">Click a player to select</div>
  {{/unless}}
</template>

<template name="player">
  <div class="player {{selected}}">
    <span class="name">{{name}}</span>
    <span class="score">{{score}}</span>
  </div>
</template>
```

Lst. 33: HTML of the Meteor example 'Leaderboard'

```
// Set up a collection to contain player information. On the server,
// it is backed by a MongoDB collection named "players".

Players = new Meteor.Collection("players");

if (Meteor.isClient) {
  Template.leaderboard.players = function () {
    return Players.find({}, {sort: {score: -1, name: 1}});
  };

  Template.leaderboard.selected_name = function () {
    var player = Players.findOne(Session.get("selected_player"));
    return player && player.name;
  };

  Template.player.selected = function () {
    return Session.equals("selected_player", this._id) ? "selected" : '';
  };

  Template.leaderboard.events({
    'click input.inc': function () {
      Players.update(Session.get("selected_player"), {$inc: {score: 5}});
    }
  });

  Template.player.events({
    'click': function () {
      Session.set("selected_player", this._id);
    }
  });
}

// On server startup, create some players if the database is empty.
if (Meteor.isServer) {
  Meteor.startup(function () {
    if (Players.find().count() === 0) {
      var names = ["Ada Lovelace",
                   "Grace Hopper",
                   "Marie Curie",
                   "Carl Friedrich Gauss",
                   "Nikola Tesla",
                   "Claude Shannon"];
      for (var i = 0; i < names.length; i++)
        Players.insert({name: names[i], score: Math.floor(Math.random()*10)*5});
    }
  });
}
```

Lst. 34: JS of the Meteor example 'Leaderboard'

Appendix 4: Case Study – LinkedIn

In 2012, LinkedIn, the career-oriented social network, has changed their back-end infrastructure of the mobile application, which was built on Ruby on Rails until then.[145] LinkedIn moved from Ruby on Rails to Node.js for performance and scalability reasons and has made immense performance improvements.[146] LinkedIn had scalability problems although less than 10LinkedIn therefore wanted to implement an event-driven model and tested and evaluated three possible candidates: Rails/Event Machine, Python/Twisted, and Node.js. Node.js was identified as the best solution for following benefits[147]:

- Better performance (running up to 20x faster in certain scenarios) and lower memory overhead than other tested options,
- Programmers could make use of their JavaScript skills.
- Frontend JavaScript and backend mobile teams could be merged into a single unit.
- Servers were cut from 30 to 3, leaving room to handle 10x current levels of resource utilization.

[145] Cf. Avram, A. (2012)
[146] Cf. O'Dell, J. (2011a)
[147] Cf. Avram, A. (2012)

Appendix 5: Case Study – AIRCODE

The following information are taken from the website of AIRCODE.[148] AIRCODE has implemented an application providing real-time news and visitor's geolocation to the visitors of their website. The application is implemented in Node.js and aims to reveal the ability of Node.js of event-driven notification in real-time, i.e. in this specific case to deliver web pages with real-time news, based on live occurring events in six countries of the world. This is accomplished by providing the user with news from six countries at the same time – news being published in the moment the user receives them. The idea is to show that different news can be delivered at the same time, which makes use of Node.js's strength of the underlying event-driven model. Node.js fetches the news feeds and distributes them to every current visitor in real-time. When new visitors connect to the web site, all other visitors will be notified with the others' position. For this, Node.js uses the broadcast function from the socket.io module.

[148] See aircode.co

Appendix 6: Case Study – Etherpad

On 22 August 2011 the Etherpad Lite, that is implemented with Node.js, was released.[149] Etherpad is an Open Source online editor providing collaborative editing of documents in real-time.[150] The original Etherpad had the following main problems:[151] Resource utilization: – the JVM that Etherpad was based on, consumed 1-2 GB RAM during normal operation time. Besides memory leaks, infinite loops, constantly high server loads and server outages were other severe problems. Peter Martischka, an Etherpad developer, migrated the entire application from Rhino to Node.js. It was a successful act. Node.js solved many problems in terms of performance and resource utilization.

[149] Cf. Martischka, P. (2011)

[150] See `http://www.etherpad.org`

[151] Cf. Martischka, P. (2011)

Appendix 7: Case Study – Clock, a UK web agency

The following information are taken from an online article[152] on VentureBeat[153] and are based on the statements by Paul Serby, Chief Technology Officer of Clock. Clock is a web agency in the United Kingdom that builds the websites for their clients in Node.js. They chose Node.js for several reasons: First, JavaScript as the underlying programming language of Node.js is very beneficial. Most developers at Clock have JavaScript expertise what makes it relatively easy to introduce Node.js and therefore saves time and costs. In addition to that, Clock uses MongoDB. Thus, code can be used on the browser-side, server-side and for the database as MongoDB uses JavaScript for querying data.[154] Paul says that "our browser-side JavaScript code improved in quality and structure". Another aspect the developers at Clock appreciate is the highly active community which is useful when solving programming issues. Also, the developers have had "large productivity gains in HTML and CSS using Jade and Stylus[155]". Furthermore, the amount of traffic could be increased with Node.js compared to PHP with Apache which was used before. Clock says that using Node.js and MongoDB they could scale up much better than with PHP and PostgreSQL. Moreover, Clock developers appreciate the many hosting options that come with Node.js (e.g. No.de, Joyent's SmartMachine, Heroku and Nodejitsu). Joyent's SmartMachine is supposed to be used for upcoming projects.
In conclusion, Clock states that at the beginning the projects took some extra time as developers had to get used to Node.js and the underlying concepts. However, Node.js fitted their use case very well (which was to develop high-scaling web applications) and after a short time, it made their development "extremely effective".

[152] Cf. Serby, P. (2012)
[153] http://venturebeat.com/
[154] http://www.mongodb.org/
[155] http://clock.co.uk/tech-blogs/a-simple-website-in-nodejs-with-express-jade-and-stylus

Appendix 8: Benchmarking results

```
#------------------------------------------------------------------------------
#Node.js
#------------------------------------------------------------------------------
ab -n10000 -c50 -g 200kb.node.dat http://192.168.178.39:8000/file.200kb
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.178.39 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:
Server Hostname:        192.168.178.39
Server Port:            8000

Document Path:          /file.200kb
Document Length:        204800 bytes

Concurrency Level:      50
Time taken for tests:   23.428 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      2050780000 bytes
HTML transferred:       2048000000 bytes
Requests per second:    426.83 [#/sec] (mean)
Time per request:       117.142 [ms] (mean)
Time per request:       2.343 [ms] (mean, across all concurrent requests)
Transfer rate:          85482.42 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   5.0      0     204
Processing:    74  117 169.0     98    1992
Waiting:        6   42  22.6     41    1436
Total:         74  117 170.2     98    1992

Percentage of the requests served within a certain time (ms)
  50%     98
  66%    102
  75%    105
  80%    107
  90%    110
  95%    112
  98%    121
  99%   1490
 100%   1992 (longest request)
```

```
#------------------------------------------------------------------------------
# Vert.x
#------------------------------------------------------------------------------

ab -n10000 -c50 -g 200kb.vertx.dat http://192.168.178.39:8000/file.200kb
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.178.39 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:
Server Hostname:        192.168.178.39
Server Port:            8000

Document Path:          /file.200kb
Document Length:        204800 bytes

Concurrency Level:      50
Time taken for tests:   42.570 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      2048430000 bytes
HTML transferred:       2048000000 bytes
Requests per second:    234.90 [#/sec] (mean)
Time per request:       212.852 [ms] (mean)
Time per request:       4.257 [ms] (mean, across all concurrent requests)
Transfer rate:          46990.80 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   4.5      0     225
Processing:    14  212 389.7    159    4176
Waiting:        3  159  83.9    154    1970
Total:         14  212 390.2    159    4176

Percentage of the requests served within a certain time (ms)
  50%    159
  66%    176
  75%    187
  80%    195
  90%    218
  95%    252
  98%    766
  99%   2425
 100%   4176 (longest request)
```

```
#------------------------------------------------------------------------------
# Nginx
#------------------------------------------------------------------------------

ab -n10000 -c50 -g 200kb.nginx.dat http://192.168.178.39:8080/~rocco/file.200kb
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.178.39 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        nginx/1.2.1
Server Hostname:        192.168.178.39
Server Port:            8080

Document Path:          /~rocco/file.200kb
Document Length:        204800 bytes

Concurrency Level:      50
Time taken for tests:   194.599 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      2050280000 bytes
HTML transferred:       2048000000 bytes
Requests per second:    51.39 [#/sec] (mean)
Time per request:       972.996 [ms] (mean)
Time per request:       19.460 [ms] (mean, across all concurrent requests)
Transfer rate:          10288.97 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   15 102.2      0    1006
Processing:     1  958 1231.5    108    6404
Waiting:        0  173 381.8     71    2007
Total:          2  973 1235.4    109    6404

Percentage of the requests served within a certain time (ms)
  50%    109
  66%   1072
  75%   1838
  80%   1998
  90%   2665
  95%   3957
  98%   4120
  99%   4875
 100%   6404 (longest request)
```

```
#------------------------------------------------------------------------------
# Apache2
#------------------------------------------------------------------------------

ab -n10000 -c50 -g 200kb.apache.dat http://192.168.178.39:80/~rocco/file.200kb
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.178.39 (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:        Apache/2.2.22
Server Hostname:        192.168.178.39
Server Port:            80

Document Path:          /~rocco/file.200kb
Document Length:        204800 bytes

Concurrency Level:      50
Time taken for tests:   488.405 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      2050340000 bytes
HTML transferred:       2048000000 bytes
Requests per second:    20.47 [#/sec] (mean)
Time per request:       2442.025 [ms] (mean)
Time per request:       48.840 [ms] (mean, across all concurrent requests)
Transfer rate:          4099.64 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   83 262.4      4    1008
Processing:    18 2359 1455.8   2022    9009
Waiting:        2  398 629.5     10    4683
Total:         21 2442 1448.3   2033    9014

Percentage of the requests served within a certain time (ms)
  50%   2033
  66%   3007
  75%   3052
  80%   3985
  90%   4053
  95%   5052
  98%   6018
  99%   6034
 100%   9014 (longest request)
```

Lst. 35: Benchmarking results as shown by the Apache benchmarking tool for web servers

Lists of References

List of Literature

Breshears, C. (2009)	The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications, Sebastopol: O'Reilly Media
Fettig, A. (2005)	Twisted Network Programming Essentials, Developing With Python's Event-driven Framework, Sebastopol: O'Reilly Media
Holtsnider, B./Jaffe, B. D. (2010)	IT Manager's Handbook: The Business Edition, Burlington: Elsevier
Hughes-Croucher, T./Wilson, M. (2012)	Node: Up and Running, Scalable Service-Side Code with JavaScript, Sebastopol: O'Reilly Media
Nemati, H. R./Yang, L. (2011)	Applied Cryptography for Cyber Security and Defense: Information Encryption and Cyphering, Hershey: IGI Global
Nemeth, E./Snyder, G./Hein, T. R. (2007)	Linux Administration Handbook, 2nd Edn., New York: Pearson Education
Nguyen, D. (2012)	Jump Start Node.js, Collingwood: SitePoint
Richter, J. (2010)	CLR via C#, 3rd, Redmond: Microsoft Press
Riordan, R. M. (2008)	Head First Ajax, Sebastopol: O'Reilly Media
Roden, G. (2012)	Node.js & Co., Skalierbare, hochperformante und echtzeitfaehige Webanwendungen professionell in JavaScript entwickeln, Heidelberg: dpunkt.verlag
Tanenbaum, A. S./Van Steen, M. (2007)	Distributed Systems, Principles and Paradigms, 2nd Edn., Upper Saddle River: Pearson Education
Teixeira, P. (2013)	Professional Node.js: Building Javascript Based Scalable Software, Indianapolis: John Wiley & Sons
Thomson, C. (2002)	Database Programming with C-Sharp, New York: Springer-Verlag

List of Internet and Intranet Resources

Arranz, J. M. (2011) — The “joy” of asynchronous programming, http://www.theserverside.com/discussions/thread.tss?thread_id=61693, retrieval: 01/21/2013

Asay, M. (2013) — VMware’s dealings with Vert.x founder should serve as a warning, http://www.theregister.co.uk/2013/01/14/opensource_ownership/, retrieval: 01/22/2013

Avram, A. (2012) — Ruby on Rails vs. Node.js at LinkedIn, http://www.infoq.com/news/2012/10/Ruby-on-Rails-Node-js-LinkedIn, retrieval: 01/19/2013

Behren, R. von/Condit, J./Brewer, E. (2003) — Why Events Are A Bad Idea (for high-concurrency servers), http://static.usenix.org/events/hotos03/tech/full_papers/vonbehren/vonbehren_html/index.html, retrieval: 01/21/2013

Caldera International Inc. (2003) — Identifying disk I/O-bound systems, http://osr507doc.sco.com/en/PERFORM/ident_IO_bound.html, retrieval: 01/22/2013

Crockford, D. (2006) — The application/json Media Type for JavaScript Object Notation (JSON), http://tools.ietf.org/html/rfc4627, retrieval: 01/18/2013

Fette, I./Melnikov, A. (2011) — The WebSocket Protocol, http://tools.ietf.org/html/rfc6455, retrieval: 01/21/2013

Fielding, R. et al. (1999) — RFC2616: HTTP header field definitions, http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14, retrieval: 01/18/2013

Fletcher, R. (2012) — vertx-proxy, https://github.com/robfletcher/vertx-proxy, retrieval: 01/21/2013

Fox, T. (2013a) — Vert.x code examples, https://github.com/vert-x/vert.x/tree/master/vertx-examples/src/main, retrieval: 01/10/2013

Fox, T. (2013b) — Vert.x repository, https://github.com/vert-x/vert.x, retrieval: 01/10/2013

Fox, T./Daryl, T., et al. (2013) Vert.x google group: An important announcement to the Vert.x community, https://groups.google.com/forum/#!msg/vertx/gnpGSxX7PzI/uRNaMtJaIJUJ, retrieval: 01/10/2013

Fox, T./Teo, D., et al. (2013) Discussion: The future of the Vert.x project, https://groups.google.com/forum/?fromgroups=#!topic/vertx/WIuY5M6RluM[1-25-false], retrieval: 01/19/2013

Fox, T./Yates, T., et al. (2012) Vert.x Module Repository on Github, https://github.com/vert-x/vertx-mods, retrieval: 01/11/2013

Geihs, K. (2001) Middleware challenges ahead, in: Computer Volume 34.6, pp. 24–31

Geisendoerfer, F. (2011) Felix's Node.js Convincing the boss guide, http://nodeguide.com/convincing_the_boss.html, retrieval: 01/20/2013

Google Inc (2013) Web Search Interest: node.js, vert.x, http://www.google.com/trends/explore?hl=en-US#q=node.js,%20vert.x&cmpt=q, retrieval: 01/18/2013

Hughes-Croucher, T. (2010) Understanding event loops, http://developer.yahoo.com/blogs/ydn/posts/2010/10/understanding-the-event-loops-and-writing-great-code-for-node-js-part-1/, retrieval: 01/01/2013

Joyent Inc (2012) API changes between v0.6 and v0.8, https://github.com/joyent/node/wiki/API-changes-between-v0.6-and-v0.8, retrieval: 01/21/2013

Joyent Inc (2012a) Node.js v0.8.15 Manual & Documentation, http://nodejs.org/api/, retrieval: 12/11/2012

Joyent Inc (2012b) Node.js v0.8.18 Manual & Documentation, http://nodejs.org/api/cluster.html#cluster_how_it_works, retrieval: 01/21/2013

Joyent Inc (2013) Node.js modules: Testing / Spec Frameworks, https://github.com/joyent/node/wiki/modules#wiki-testing, retrieval: 01/19/2013

Joyent Inc (w. y.[a]) — Child Process Node.js v0.8.17, http://nodejs.org/api/child_process.html, retrieval: 01/17/2013

Joyent Inc (w. y.[b]) — Node Packaged Modules, https://npmjs.org/, retrieval: 01/18/2013

Klishin, M. (2012) — EventMachine Readme, https://github.com/eventmachine/eventmachine/blob/b79d508532e210084a7fcda31f330a8a81bf78aa/README.md, retrieval: 01/12/2013

Knight, M. (2011) — Message Queues in Node.js, http://www.matt-knight.co.uk/2011/message-queues-in-node-js/, retrieval: 01/19/2013

Krumins, P. (2010) — A HTTP Proxy Server in 20 Lines of node.js Code, http://www.catonmat.net/http-proxy-in-nodejs/, retrieval: 01/21/2013

Majkowski, M. et al. (2013) — SockJS client, https://github.com/sockjs/sockjs-client, retrieval: 01/22/2013

Martischka, P. (2011) — Wie Etherpad Lite entstand, http://pitapoison.de/wie-etherpad-lite-entstand, retrieval: 01/19/2013

Matejka, J. (2012) — w. t., http://jirimatejka.cz/ajax-diagram, retrieval: 01/12/2013

McCalla, J. (2011) — Node.js – Cluster for multiple servers? https://groups.google.com/forum/?fromgroups=#!topic/nodejs/V4Rx09ADuGM, retrieval: 01/21/2013

McGlennon, G. (2007) — Message Queues, http://www.dotnetprofessional.com/blog/file.axd?file=CHAPTER+4+-+Message+Queues.pdf, retrieval: 01/15/2013

McMahon, C. (2012) — Node.js: style and structure, http://caolanmcmahon.com/posts/nodejs_style_and_structure/, retrieval: 01/19/2013

Nutt, M./Wong, B. (w. y.) — Hummingbird Project Website, http://hummingbirdstats.com/, retrieval: 01/16/2013

O'Dell, J. (2011a) Exclusive: How LinkedIn used Node.js and HTML5 to build a better, faster app, http://venturebeat.com/2011/08/16/linkedin-node/, retrieval: 01/19/2013

O'Dell, J. (2011b) Why Everyone Is Talking About Node, http://mashable.com/2011/03/10/node-js/, retrieval: 01/09/2013

O'Dell, J. (2012) Node at scale: What Google, Mozilla, & Yahoo are doing with Node.js, http://venturebeat.com/2012/01/24/node-at-google-mozilla-yahoo/, retrieval: 01/18/2013

O'Grady, S. (2012) The RedMonk Programming Language Rankings: September 2012, http://redmonk.com/sogrady/2012/09/12/language-rankings-9-12/, retrieval: 01/18/2013

Opa Team (w. y.) Opa example, http://opalang.org/, retrieval: 01/22/2013

piscisaureus (2011) npm is now part of Node.js, http://blog.nodejs.org/2011/11/25/node-v0-6-3/, retrieval: 01/22/2013

Schmidt, G. et al. (w. y.[a]) Documentation – Meteor, http://docs.meteor.com/, retrieval: 01/18/2013

Schmidt, G. et al. (w. y.[b]) Leaderboard example, http://meteor.com/examples/leaderboard, retrieval: 01/15/2013

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/, retrieval: 01/21/2013

Serby, P. (2012) Case study: How & why to build a consumer app with Node.js, http://venturebeat.com/2012/01/07/building-consumer-apps-with-node/, retrieval: 01/20/2013

Stannard, A. (2011) Node.js processing, http://www.aaronstannard.com/image.axd?picture=nodejs%20for%20dotnet_thumb.png, retrieval: 01/22/2013

Swartz, A. (2005) A Brief History of Ajax, http://www.aaronsw.com/weblog/ajaxhistory, retrieval: 01/22/2013

The Apache Software Foundation (2010) — DefaultExecutor (Commons Exec API), http://commons.apache.org/exec/apidocs/org/apache/commons/exec/DefaultExecutor.html, retrieval: 01/18/2013

The Apache Software Foundation (2013) — Multi-Processing Modules (MPMs) - Apache HTTP Server, http://httpd.apache.org/docs/2.2/mpm.html, retrieval: 01/20/2013

Thornsby, J. (2010) — Node.js Moves to Joyent, http://jaxenter.com/node-js-moves-to-joyent-32530.html, retrieval: 01/05/2013

w. a. (2012a) — Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011–2016, http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf, retrieval: 01/22/2013

w. a. (2012b) — nodejitsu / node-http-proxy, https://github.com/nodejitsu/node-http-proxy, retrieval: 01/21/2013

w. a. (2012c) — Twisted Project Page, http://twistedmatrix.com/trac/, retrieval: 01/01/2013

w. a. (2012d) — Vert.x Main Manual, http://vertx.io/manual.html, retrieval: 01/10/2013

w. a. (2012e) — vert.x Module Manual, http://vertx.io/mods_manual.html, retrieval: 01/10/2013

w. a. (2012f) — vert-x / mod-web-server, https://github.com/vert-x/mod-web-server, retrieval: 01/22/2013

w. a. (2013a) — Node v0.9.7 (Unstable), http://blog.nodejs.org/2013/01/18/node-v0-9-7-unstable/, retrieval: 01/19/2013

w. a. (2013b) — vert-x/vert.x Commit History, https://github.com/vert-x/vert.x/graphs/commit-activity, retrieval: 01/22/2013

w. a. (w. y.) — jQuery: The Write Less, Do More, JavaScript Library, http://jquery.com/, retrieval: 01/21/2013

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

NoSQL-Datenbanken

Hadoop und seine Komponenten

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“
Kompetenzzentrum Open Source (KOS)

Vorgelegt von

Sabine Benndorf,
Sabine Dahms,
Martin Gasch,
Kai Stängle

am 22.01.2013

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
2010 V

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACID	**A**tomacity, **C**onsistency, **I**solation, **D**urability
Amazon S3	**Amazon S**imple **S**torage **S**ervice
API	**A**pplication **P**rogramming **I**nterface
ASF	**A**pache **S**oftware **F**oundation
AWS	**A**mazon **W**eb **S**ervices
BASE	**b**asically **a**vailable, **s**oft-state, **e**ventually consistent
BI	**B**usiness **I**ntelligence
CSV	**C**omma-**S**eperated **V**alues
DB	**D**aten**b**ank
DBS	**D**aten**b**ank**s**ystem
DDL	**D**ata **D**efinition **L**anguage
DML	**D**ata **M**anipulation **L**anguage
DWH	**D**ata **W**are **H**ouse
ERP-System	**E**nterprise-**R**esource-**P**lanning-**System**
ETL	**E**xtract, **T**ransform, **L**oad
HDFS	**H**adoop **D**istributed **F**ile **S**ystem
HiveQL	**Hive Q**uery **L**anguage
FS Shell	**F**ile **S**ystem **Shell**
IDC	**I**nternational **D**ata **C**orporation
IP	**I**nternet **P**rotokoll
IT	**I**nformation **T**echnology
JAR	**J**ava **Ar**chive
JDBC	**J**ava **D**ata**b**ase **C**onnectivity
JDK	**J**ava **V**irtual **M**achine
JRE	**J**ava **R**untime **E**nvironment
JVM	**J**ava **V**irtual **M**achine
NoSQL	**N**ot **o**nly **SQL**
ODBC	**O**pen **D**ata**b**ase **C**onnectivity
PDF	**P**ortable **D**ocument **F**ormat
PPA	**P**ersonal **P**ackage **A**rchive
RDBMS	**r**elationales **D**aten**b**ank**m**anagement**s**ystem
REST	**Re**presentational **S**tate **T**ransfer
RPC	**R**emote **P**rocedure **C**all
RRS	**R**educed **R**edundancy **S**torage
SQL	**S**tructured **Q**uery **L**anguage

SSH	**S**ecure **Sh**ell
TB	**T**era**b**yte
TCP	**T**ransmission **C**ontrol **P**rotocoll
UDF	**U**ser **D**efined **F**unctions
WebDAV	**Web**-based **D**istributed **A**uthoring and **V**ersioning
XML	E**x**tensible **M**arkup **L**anguage

Abbildungsverzeichnis

1 Einleitung

Das Volumen unstrukturierter Daten wird immer größer. Gerade bei den Big-Players im Web 2.0 wie Facebook, Yahoo und Google fallen immer mehr Daten an. In den letzten zwei Jahren wurden durch diese Firmen ca. 90% des gesamten Datenvolumens im Internet erzeugt. Mit den traditionellen relationalen Datenbanksystemen, wie MySQL, DB2 oder Oracle, sind diese enormen Datenmengen, welche sich im Terabyte-Bereich bewegen, nicht mehr effizient zu verarbeiten. Um diese „Big Data“ in den Griff zu bekommen, bieten mittlerweile große und kleine Anbieter viele verschiedene Werkzeuge an, welche allesamt auf sogenannten „Not only SQL“ (NoSQL) Datenbanken aufbauen. Zentraler Bestandteil der NoSQL-Systeme ist häufig die Technologie Hadoop.[1]

Hadoop ist ein Software-Framework, das in Java programmiert wurde. Es erlaubt dem Anwender, rechenintensive Prozesse mit großem Datenvolumen auf Server-Clustern abarbeiten zu können und somit die Effizienz des Datendurchsatzes enorm zu steigern. Hadoop besteht aus zwei Teilen: dem Hadoop Distributed File System (HDFS) und dem MapReduce-Algorithmus. HDFS verteilt die Daten auf die einzelnen Systeme. Verarbeitet werden die Daten auf den einzelnen Rechnern mit Hilfe des MapReduce-Algorithmus. Weitere Werkzeuge bauen verschiedenste Funktionen um den Kern der Hadoop-Technologie herum. Diese ermöglichen beispielsweise eine benutzerfreundliche Bedienung oder eine standardisierte Verteilung der Daten auf die einzelnen Systeme im jeweiligen Rechner-Netz.

Das MapReduce-Programmiermodell wurde bereits Ende 2004 von Google veröffentlicht. Kurze Zeit später wurde daraufhin das Open-Source Projekt Hadoop von Apache gestartet, welches den MapReduce-Algorithmus mit HDFS kombiniert. 2008 erhielt Hadoop den Status eines Top-Level-Projekts der Apache Software Foundation (ASF). Die erste offizielle Release-Version 1.0.0 wurde Ende 2011 veröffentlicht. Heute verwenden Unternehmen wie Ebay, Facebook und Google die Technologie Hadoop.[2][3]

In dieser Ausarbeitung sollen die Grundkomponenten von Hadoop erläutert werden. Dabei stehen einleitend Erläuterungen zum Hintergrund und der Problemsituation. Anschließend wird der Begriff Big Data näher beschrieben. Der vierte Themenblock beschreibt anschaulich Hadoop und dessen Basiskomponenten. Anhand von Beispielen werden das HDFS und das MapReduce-Programmiermodell erklärt. Wie Hadoop mit den Basiskomponenten umgeht

[1] Vgl. Bayer, M. (2012)
[2] Vgl. The Apache Software Foundation (2012h)
[3] Vgl. White, T. (2012), S. 19 f

und wie die Funktionalitäten aufgebaut sind, wird aufbauend auf den Erkenntnissen des vorherigen Kapitels im nächsten Unterpunkt behandelt. In der darauf folgenden Rubrik werden die zusätzlichen Komponenten zu Hadoop erläutert, die das Arbeiten mit der Technologie erleichtern. Im sechsten Kapitel wird mit den Ergebnissen des vierten Kapitels das Aufsetzen eines Prototyps von Hadoop dokumentiert. Dabei wird sowohl die Cloud-Lösung, als auch die Implementierung auf einem Rechner behandelt. Abschließend werden in einem Fazit die Kernaussagen der Ausarbeitung zusammengefasst und ein Zukunftsausblick über Hadoop und dessen Komponenten gegeben. Im Fazit werden ebenfalls mögliche zusätzliche Vertiefungspunkte im Hadoop-Umfeld angesprochen.

2 Hintergrund und Problemsituation

Wenn die Diskussion, um eine neu zu erstellende Datenbank aufkommt, ist meist von Anfang an klar, dass ein relationales Datenbanksystem verwendet wird. Allerdings gibt es auch Anwendungsbereiche, bei denen eine relationale Datenbank nicht die beste Lösung darstellt. Vor allem wenn es sich bei den zu erfassenden Daten um eine sehr große Menge semistrukturierter Daten handelt. Hierbei ist eine Verarbeitung mit NoSQL sinnvoller. Im Folgenden werden einige Vorteile von NoSQL im Vergleich zu relationalen Datenbanken erläutert.

Vor allem die für NoSQL wichtigen Suchanfragen werden bei relationalen Datenbanken viel langsamer bearbeitet als bei MapReduce. Hadoop ist zudem für nicht-normalisierte, semistrukturierte Daten geeignet, während relationale Datenbanken nur normalisierte, strukturierte Daten verarbeitet. Außerdem ist MapReduce für einen einmaligen Schreibprozess und viele Leseprozesse ausgelegt, während herkömmliche Systeme sowohl für viele Lese- als auch viele Schreibprozesse konzipiert wurden. Ein weiterer Vorteil von NoSQL ist die Schemafreiheit. Wenn bei relationalen Datenbanken Datendefinitionen verändert werden müssen, hat dies meistens eine Überarbeitung des gesamten Schemas zur Folge, wohingegen dies bei NoSQL kein Problem ist, da diese Systeme mit „Versionierung der Daten und Konvertierungen im Rahmen von Hintergrundprozessen arbeitet" [4]. Relationale Datenbanken sind zudem streng am Konsistenzmodell ACID ausgelegt, wonach Daten atomar, konsistent, isoliert und dauerhaft sein müssen. Im Gegensatz dazu müssen NoSQL-Datenbanken nicht streng konsistent, sondern vor allem hoch verfügbar sein. Somit fällt es unter ein anderes Schema, dem BASE. BASE besteht aus drei Grundsätzen: „basically available, soft-state, eventually consistent"[5]. Dies bedeutet, dass BASE zwar eine hohe Verfügbarkeit und Partitionstoleranz

[4] Pürner, H. (2011), S. 29
[5] Ebenda, S. 29

voraussetzt, aber eine eher weiche Konsistenz hat, was das System zwar schneller, allerdings unsicherer macht.[6] Eberhard Wolff findet in einem Artikel der Zeitschrift Computerwoche folgende Zusammenfassung für NoSQL: "Die Zukunft gehört nicht der Ablösung von relationalen Datenbanken, sondern ihrer Ergänzung". [7]

3 Definition Big Data

Auch bei vielen Firmen fällt immer öfter der Begriff „Big Data“. Dies ist ein recht junges Themengebiet der IT, das noch keinen Einzug in die breite Wirtschaft gefunden hat. Erst wenige Unternehmen schöpfen das Potenzial von Big Data aus. Doch worum handelt es sich hierbei genau? Eine exakte, scharfe Definition fällt auch Experten schwer. Big Data allein mit der Anhäufung von großen Datenmengen zu definieren, würde der Thematik bei weitem nicht gerecht werden. Auch mit dem Zusatz, dass sich diese großen Datenmengen schnell verändern, wird das Ausmaß nicht abgedeckt. Bei dem komplexen Thema bedarf es einer ebenso komplexen Definition, wie beispielsweise die der International Data Corporation Inc. (IDC). Diese teilt Big Data in vier Faktoren ein:

- Volume
- Variety
- Velocity
- Value

Zu Deutsch: Menge, Variabilität, Geschwindigkeit und Wert. Diese vier Vs werden im Folgenden näher erläutert.[8]

Volumen

Das Volumen an strukturierten Daten, die sich klassischen Datenbankfeldern zuordnen lassen, umfasst derzeit lediglich etwa 20%, mit abnehmender Tendenz. Der Rest des Datenvolumens besteht aus sogenannten semi- oder polystrukturierten Daten. Von unstrukturierten Daten kann hier nicht gesprochen werden, da selbst Bilder, Texte oder Voicemails eine gewisse Struktur ausweisen. Um einen Eindruck davon zu bekommen, wie groß das Datenvolumen von Big Data ist, stellte die IDC eine Statistik mit dem weltweiten Datenvolumen auf. Beispielsweise betrag das Volumen im Jahre 2011 bereits 1,8 Zetta-Byte, wobei ein Zetta-Byte über eine Milliarde Tera-Byte umfasst. Zudem findet ca. alle zwei Jahre eine Verdopp-

[6] Vgl. Pürner, H. (2011), S. 28 ff
[7] Wolff, E. (2012), S. 17
[8] Vgl. Spies, R. (2012), S.141 f

lung des weltweit genutzten Volumens an Daten statt. Wenn dieser Trend weiter anhält, hätte dies zur Folge, dass im Jahr 2017 mehr als 14 Zetta-Byte Daten genutzt werden. Dieser enorme Anstieg hätte starke Auswirkungen auf unvorbereitete IT-Abteilungen. Da Daten viel schneller produziert als ausgewertet werden können, wäre das zentrale Problem hierbei nicht etwa die Speicherung der vielen Daten, sondern vielmehr ein sinnvoller Umgang mit den Informationen, die in den Daten enthalten sind. [9]

Variety

Eine weitere Komponente von Big Data ist die Vielfalt bzw. Vielschichtigkeit der Daten. Vor allem Texte, Bilder, Sounds oder elektronische Korrespondenz, wie E-Mails, Chats oder sonstiger Nachrichten müssen in einer passenden Datenbank gespeichert werden. Diese sind, wie bereits erwähnt, meist semi- bzw. polystrukturiert. Daher werden bei Big Data die Daten, die zuvor meinst in Content-Management-Systemen ("Redaktionssystem, mit dessen Hilfe der Inhalt z.B. von Websites verwaltet wird" [10]) verwaltet wurden, mit den strukturierten Daten aus bestehenden klassischen Unternehmensanwendungen, wie beispielsweise einem Enterprise-Resource-Planning-System (ERP-System), integriert und zusammen verwaltet. Unter einem ERP-System versteht man im Allgemeinen eine Anwendungssoftware, die alle im Unternehmen ablaufenden Geschäftsprozesse unterstützt.[11] Zudem müssen alle Daten unterschiedlicher Daten-Typen mit einer einheitlichen Sicht erfasst und für Anwender verfügbar gemacht werden können. [12]

Velocity

Ein weiterer Punkt bei Big Data ist die Geschwindigkeit der Verarbeitung der Daten. In vielen Unternehmen fließen zusätzliche Daten aus Marketing- oder Service-Bereichen mit in die Datenverarbeitung ein. Hierbei spielt vor allem die Integration der gesammelten Daten aus Social Media Plattformen eine wichtige Rolle. Zum einen kann Social Media als ein ordinäres Kommunikationsmittel zwischen dem Unternehmen und Kunden bzw. Interessenten gesehen werden. Zum anderen können hierbei allerdings auch wertvolle Erkenntnisse über die Affinitäten und Bedürfnisse der Kunden ermittelt werden. Dies bedeutet, dass riesige Datenströme durchsucht werden müssen, um alle für das Unternehmen relevante Daten abzuleiten. Zudem können hierbei aus einer Vielzahl von Daten vorgegebene oder neue Strukturen erkannt werden. Alle vom System als unerheblich angesehenen Daten dürfen nicht erfasst werden, um den Datenstrom möglichst gering zu halten.

[9] Vgl. Spies, R. (2012), S. 143 f
[10] Vgl. Lackers, R. u.a. (2012)
[11] Vgl. Vahrenkamp, R., Siepernamm, C. (2012)
[12] Vgl. Spies, R. (2012), S. 144 f

Eine weitere Möglichkeit ist das Einscannen von Briefen und das digitale Erfassen von E-Mails, Voicemails und Callcenter-Dialogen, um diese mit Hilfe von Business Intelligence (BI) Technologien zu analysieren. Somit können beispielsweise wiederkehrende Muster bei Kundenanfragen erkannt und mögliche Problemfelder im Vorhinein verhindert werden. Außerdem können so eingehende Dokumente ohne manuelle Aufgabenverteilung automatisch, also per Dunkelverarbeitung, an den richtigen Mitarbeiter weitergeleitet werden. Somit hat Big Data zudem Einfluss auf Geschäftsprozess-Management-Systeme eines Unternehmens. Ein weiterer Beweis dafür, dass Big Data weit mehr ist als die übliche Datenanalyse mit Data Warehouse und BI. Ohne weitere Beispiele aufzuführen, prophezeit die IDC, dass es bald keinen IT-Bereich mehr geben wird, der nicht von Big Data beeinflusst ist. [13]

Value

Die letzte Komponente von Big Data zeigt, wie sich daraus ein zusätzlicher Wert für den Unternehmenserfolg ableiten lässt. Das bisherige BI, was hier besser als „Business Analytics“ bezeichnet werden sollte, und dessen Technologien müssen nun auf ein großen Maß an neuen Datentypen angewendet werden. Im Business Analytics gibt es im Gegensatz zum klassischen BI nicht nur Abfragen, Berichte und Analysen, sondern auch beispielsweise verwandte Felder wie das Performance Management, also eine integrierte Unternehmenssteuerungsarchitektur, und ähnliche Analytics-Umgebungen, wie geografische Analysen. Das Data Mining ist zwischen diesen beiden Bereichen einzuordnen.

Es ergeben sich für Unternehmen somit zwei große Chancen mit Big Data. Zum einen werden unterschiedliche Datentypen, Quellen und Zielrichtungen miteinander verknüpft, wodurch das Darstellen der technischen Aspekte von Big Data ermöglicht wird. Zum anderen wird verdeutlicht, dass die IT-Abteilungen wertvolle Informationen für das gesamte Unternehmen bereitstellt, die aus einer Vielzahl von Daten mit unterschiedlichen Datentypen und Quellen gefiltert werden müssen. Ein wichtiges Kriterium hierbei ist, dass die Fachabteilungen sehr gut mit den IT-Abteilungen zusammenarbeiten müssen. Außerdem ist ein gut organisiertes IT Business Alignment unabdingbar. [14] Unter IT Business Alignment versteht man die Forderung nach einem auf das Business abgestimmten Einsatz der IT.[15]

[13] Vgl. Spies, R. (2012), S. 145 f
[14] Vgl. ebenda, S. 146 f
[15] Vgl. Winter, R., Landert, K. (2006), S. 309

4 Hadoop & Technologien

Wie aus den letzten Ausarbeitungspunkten hervorgeht, sind relationale Datenbanksysteme nicht für die Verarbeitung von Big Data ausgelegt. Es gibt mittlerweile einige Systeme, um diese Datenart zu verarbeiten. Die zentrale Technik der NoSQL-Systeme ist oft Hadoop. Hadoop basiert auf einem speziellen Dateisystem, dem HDFS, und einem darauf arbeitenden Programmiermodell, dem MapReduce-Verfahren. Hadoop wird in den nächsten Ausarbeitungspunkten ausführlicher erklärt. Zu Beginn steht hier die Vorstellung der Basiskomponenten HDFS und MapReduce. Anschließend wird die Architektur von Hadoop und dessen Funktionsweise erklärt.

4.1 HDFS

Dem HDFS liegt eine Master/Slave Architektur zugrunde, die in einem Cluster realisiert ist. Ein Cluster besteht aus mehreren Racks. Ein Rack besteht dabei aus 30 bis 40 Nodes, also einzelnen Rechnern, die alle am selben Netzwerk-Switch angeschlossen sind. Ein Cluster ist somit die Verbindung vieler Rechner zu einer Recheneinheit. Bei einem Cluster im Allgemeinen ist es nötig, die Server (Nodes) in einem physikalischen Rack zu organisieren. Hadoop ist aber in der Lage diese Informationen auszuwerten, um Daten ausfallsicher zu verteilen und die Netzwerkauslastung zu optimieren. Bei Hadoop besteht jeder HDFS Cluster aus einem NameNode (Master) und weiteren DataNodes (Slaves). Der NameNode verwaltet das Dateisystem und regelt den Datenzugriff durch die Clients. Die zugehörigen DataNodes verwalten die gespeicherten Datenblöcke.

NameNode und DataNodes

Intern werden die gespeicherten Dateien üblicherweise in 64 MB große Blöcke aufgeteilt und auf den DataNodes verteilt abgelegt. Der NameNode führt Operationen wie, z.B. öffnen, schließen und umbenennen von Dateien und Verzeichnissen im Dateisystem, aus. Außerdem entscheidet der NameNode über die Zuordnung der Datenblöcke auf den DataNodes. Die DataNodes sind für die Lese- und Schreibanfragen der Clients verantwortlich und führen auf Anweisung des NameNodes die Erstellung, Löschung und Replikation von Blöcken aus. Der NameNode speichert die Metadaten eines Clusters, welche Informationen darüber enthalten, welche Dateiblöcke auf welchem DataNode liegen und wie oft diese weiter repliziert wurden. Zudem ist der dieser so ausgelegt, dass Daten aus den Big Data-Blöcken niemals über den NameNode laufen.[16] Die beschriebene Architektur von HDFS ist zudem in Abb. 1 visualisiert.

[16] Vgl. Borthakur, D. (2009), S. 4 f

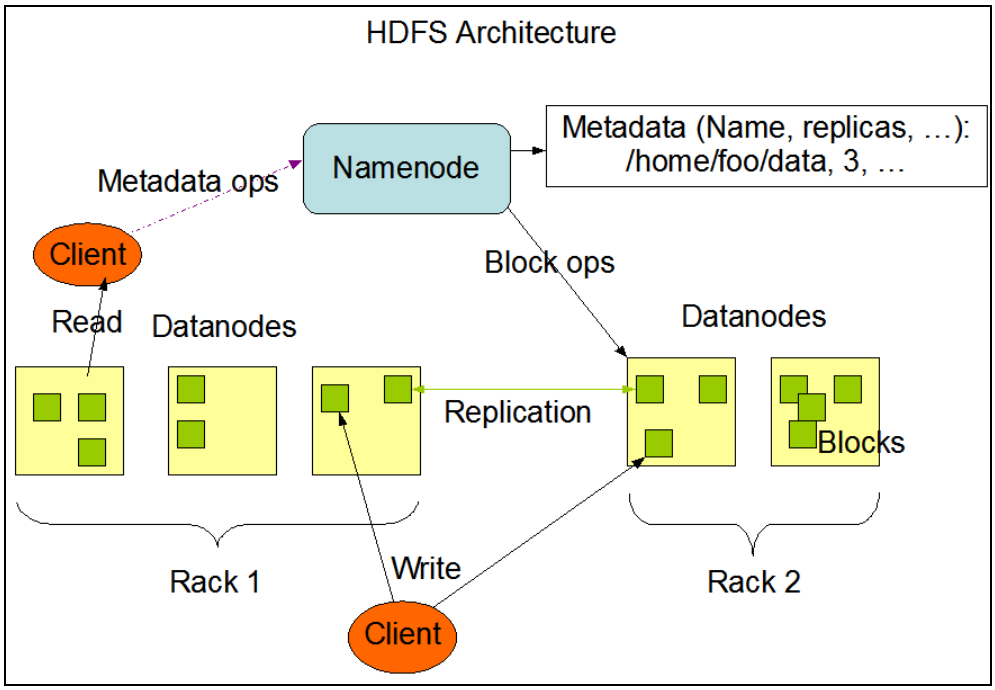


Abb. 1: HDFS Architektur[17]

Aufbau des Dateisystems

HDFS unterstützt eine traditionelle und hierarchische Dateiorganisation. Dadurch können Verzeichnisse erstellt und Dateien in diesen gespeichert werden. HDFS beinhaltet bis jetzt keine Limitierung des Speicherplatzes je Benutzer. Dennoch schließt die HDFS Architektur die Implementierung dieser Eigenschaften nicht aus. Der NameNode erhält das Dateisystem aufrecht. Jegliche Änderungen am Dateisystem oder seiner Eigenschaften werden vom NameNode sofort registriert.[18]

Replikation der Daten

HDFS wurde speziell dafür entwickelt, sehr große Dateien verteilt in einem Cluster zu speichern. Jede Datei wird in einer Sequenz aus Blöcken gespeichert, die alle dieselbe Größe haben. Diese Blöcke sind komplett gefüllt, bis auf den letzten Block, der im Regelfall weniger Daten enthält, aber die gleiche Blockgröße hat. Die Blöcke der Dateien werden aus Gründen der Fehlertoleranz repliziert. Die Blockgröße und der Replikationsfaktor sind dabei konfigurierbar. Der NameNode trifft alle Entscheidungen bezüglich der Replikation von Blöcken. Er empfängt periodisch einen Impuls von den DataNodes, der signalisiert, dass alles in Ordnung ist. Dieser wird auch „Heartbeat“ genannt. Zusätzlich erhält er einen Blockreport von jedem DataNode im Cluster, der eine Liste aller auf dem entsprechenden DataNode gespeicherten Blöcke enthält. [19]

[17] enthalten in: Borthakur, D. (2009), S. 5
[18] Vgl. Borthakur, D. (2009), S. 5 f
[19] Ebenda, S. 6

Speichern der Metadaten im Dateisystem

Der NameNode verwendet ein Transaktionslog namens EditLog zur Speicherung aller Änderungen, die die Metadaten des HDFS Dateisystems betreffen. Der EditLog wird auf dem lokalen Dateisystem des NameNode gespeichert. Das Speichern der Zusammengehörigkeit der einzelnen Blöcke zu der Ursprungsdatei und die Eigenschaften des HDFS Dateisystems sind in einer Datei namens FsImage, ebenfalls auf dem lokalen Dateisystem des NameNode, gespeichert. Der NameNode behält ein Image der Datei FsImage im Arbeitsspeicher. Beim Start des NameNode werden die Dateien FsImage und EditLog von der Festplatte gelesen. Danach werden alle Transaktionen aus dem EditLog ausgelesen und in das sich im Hauptspeicher befindende FsImage geschrieben. Daraufhin wird das aktualisierte FsImage zurück auf die lokale Festplatte des NameNode geschrieben und die alte Version des EditLog entfernt.

Der DataNode speichert jeden Datenblock in einer separaten Datei auf seinem lokalen Dateisystem ab, ohne dabei Kenntnisse über die HDFS Dateien zu haben. Beim Start des DataNode wird ein Scan des lokalen Dateisystems durchgeführt, eine Liste aller HDFS Datenblöcke erstellt und den daraus entstandenen Blockreport an den entsprechenden NameNode gesendet. [20]

Kommunikationsprotokolle

Alle HDFS Kommunikationsprotokolle bauen auf TCP/IP auf. Ein Client baut eine Verbindung über einen konfigurierbaren TCP Port zum NameNode auf. Daraufhin kann er über das Client Protokoll mit dem NameNode kommunizieren. Der DataNode kommuniziert über das DataNode Protokoll mit dem NameNode. Eine Art Remote Procedure Call (RPC) umhüllt das Client Protokoll und das DataNode Protokoll. Somit startet ein NameNode niemals einen RPC Aufruf, er erhält lediglich RPC Anfragen, die durch einen DataNode oder Client ausgelöst wurden.[21]

Zugriffsmöglichkeiten

Auf das HDFS kann auf unterschiedliche Art und Weise zugegriffen werden. Standardmäßig stellt HDFS ein Application Programming Interface (API) für andere Anwendungen und einen Wrapper für die Programmiersprache C bereit. Zusätzlich kann über einen Browser und dem Web-based Distributed Authoring and Versioning (WebDAV) Protokoll im Dateisystem einer HDFS Instanz navigiert werden. [22]

[20] Vgl. Borthakur, D. (2009), S. 9
[21] Ebenda, S. 9
[22] Ebenda, S. 12

FS Shell

HDFS erlaubt die Organisation von Benutzerdaten in Form von Dateien und Verzeichnissen. Zur Benutzer-Interaktion mit den Daten innerhalb des Dateisystems bietet HDFS ein Kommandozeilen-Interface namens FS Shell (File System Shell). Die FS Shell wird genutzt um einen Überblick über die Daten zu erhalten und um sie zu manipulieren. [23]

Browser Interface

Eine typische HDFS Installation verfügt über einen Webserver, um das Dateisystems über einen konfigurierbaren TCP-Port erreichen zu können. Dies erlaubt dem Benutzer die Navigation durch das Dateisystem und zeigt den Inhalt seiner Dateien im Browser an. [24]

4.2 MapReduce-Programmiermodell

Das MapReduce-Programmiermodell ist der Basisalgorithmus des Hadoop-Systems. Erst hierdurch wird die effiziente Verarbeitung sehr großer Datenmengen möglich. MapReduce wurde bereits 2004 von Google veröffentlicht.

Durch das Auslagern von Daten und Anwendungen auf mehrere Rechner können viele Funktionen parallel ausgeführt und anschließend zusammen evaluiert werden. Um die Effizienz zu gewährleisten, muss sichergestellt werden, dass die Daten den jeweiligen Verarbeitungsprozessen zur Verfügung stehen. Einfach gesagt: „Die Anwendungen müssen zu den Daten kommen, nicht die Daten zu den Anwendungen.“[25] Somit werden also die Anwendungen und die Daten auf die verschiedenen Knoten im Cluster-System, ergo bei Hadoop auf die einzelnen DataNodes verteilt. Die Zerteilung der gesamten Daten in Blöcke und die Verteilung derer wird vom HDFS übernommen. Das für die Auswertung der Daten zugrundeliegende Programmiermodell muss wegen der vielen einzelnen Datenblöcke leicht parallelisierbar sein. MapReduce erfüllt die Anforderungen höchst effizient. Bei dem Programmiermodell geht es um Berechnungen zu den Daten, nicht um deren Manipulation. MapReduce besteht im Wesentlichen aus zwei einzelnen Phasen. Die Phase, in der die Map-Funktion (Mapper) ausgeführt wird, und die, in der die Reduce-Funktion ausgeführt wird. Durch die Aufteilung in die beiden Phasen kann die Verarbeitung einfach parallelisiert werden.

Es wird ein Mapper pro definiertem Datei-Abschnitt initialisiert. Die Map-Funktion baut in dem jeweils ausgewählten Datenfeld Key-Value-Paare. Dabei wird für die ausgewählten Schlüsseldaten jeweils der dazugehörige Wert gespeichert. Beispielsweise ein beliebiges Wort als

[23] Vgl. Wartala, R. (2012), S. 74 ff
[24] Vgl. Borthakur, D. (2009), S. 13
[25] Wartala, R. (2012), S. 17

Key und dessen Anzahl in einem bestimmten Text als Value. Um dies zu realisieren, liest die Map-Funktion die Input-Daten sequenziell. Input-Parameter des Mappers sind eine Liste mit den zu suchenden Key-Value-Paaren und eine Funktion für deren Verarbeitung. Beispielsweise die Addition der einzelnen Values jedes Keys oder das Aufspüren des größten Wertes jedes Paares. Nützlich kann auch eine Operation zu absoluten Häufigkeiten sein, also beispielsweise welches Wort in der Email-Korrespondenz eines Mitarbeiters wie oft vorkommt. Als Output produziert der Mapper wiederrum eine Liste mit den jeweiligen Key-Value-Paaren mit der angewiesenen Verarbeitung. Daraufhin wird eine Unterfunktion der Map-Funktion ausgeführt, die Combine-Funktion. Diese fasst mehrfach auftretende Key-Value-Paare zusammen und speichert das Ergebnis. Diese lokal gespeicherten Daten für jeweils eine Datei werden dann an die Reduce-Funktion übergeben. Die vielen einzelnen Ergebnislisten der Mapper werden mit Hilfe der Reduce-Funktion wieder zu einer Gesamtliste zusammengeführt. Der Map-Algorithmus stellt dann auf einem einzelnen DataNode genau die Daten zur Verfügung, die vom NameNode angefragt wurden. Als Abschluss der Reduce-Funktion werden die ausgewerteten Daten wieder zurück in das Dateisystem geschrieben, bei Hadoop also in das HDSF.[26]

Die Parallelität des MapReduce-Programmiermodells kann sowohl durch die Aufteilung der Daten in verschiedene Eingabelisten optimiert werden, als auch durch den Einsatz des Master-Slave-Ansatzes mit NameNode und DataNodes. Effizient wird das System dadurch, dass mehrere Map- und Reduce-Prozesse mit nur einem bestimmten Teil der gesamten Daten auf bestimmten Rechnern arbeiten können.

In Abb. 2 ist der Aufbau von MapReduce anhand eines Beispiels visualisiert. Dabei handelt es sich um einen schematischen Ausschnitt des eigentlichen Verarbeitungsprozesses. In der Praxis handelt es sich um eine große Vielzahl solcher Prozesse mit Input-Daten im Tera- und Peta-Byte Bereich. Im Schritt ① werden die großen Input-Daten vom Framework in viele kleine Dateien gesplittet. Diese werden in diesem Schritt simultan vom HSDF auf einzelne DataNodes verteilt. In dem veranschaulichten Beispiel handelt es um eine Text-Datei mit dem Inhalt „Hello World Bye World Hello Hadoop Goodbye Hadoop“. Diese Datei wird in kleine Blöcke geteilt und auf die einzelnen DataNodes verteilt. Die Datei wird in zwei Blöcke geteilt, die jeweils die gleiche Anzahl von Wörtern beinhalten. Anschließend beginnt die Verarbeitung mittels MapReduce-Programmiermodell. Auf dem jeweiligen DataNode wird in Schritt ② die Map-Funktion ausgeführt. Jeweils ein Mapper wird auf eine Datei angesetzt. Den Mappern werden dabei die auszuwertenden Key-Value-Paare und die Operationsanweisung derer mitgegeben. In dem Beispiel wird der Mapper darauf angesetzt die absoluten

[26] Vgl. The Apache Software Foundation (2012i)

Häufigkeiten der einzelnen Wörter in den Dateien zu finden. Die Mapper in Schritt ② verarbeiten die Dateien sequenziell und versehen lediglich die Key-Felder, also die einzelnen Wörter, mit dem aktuellen Value, also der Anzahl. Hier ist somit jedes Wort einzeln aufgeführt mit der Häufigkeit 1. Schritt ③ zählt immer noch zur zweiten Phase. Der Combiner führt die ihm angewiesene Operation aus, in diesem Fall die Addition der Häufigkeiten für jedes Wort. Denkbar wäre hier aber auch beispielsweise nur die Ausgabe des Wortes mit der größten Häufigkeit. In diesem Beispiel werden jetzt die vom Mapper gelieferten Ergebnisse jeweils in den einzelnen Dateien zusammengerechnet. Gleiche Key-Value-Paare werden also zusammenaddiert. Das Ergebnis der Combiner sind zusammengefasste Ausgabelisten der gemappten Daten. Da in der ersten Datei das Wort „World“ zweimal vorkommt, wird es vom Combiner zusammengefasst und als Key-Value-Paar „World, 2“ ausgegeben. Simultan kombiniert der Combiner der anderen Datei deren gemappte Daten. In Phase zwei wird Schritt ④, und somit der letzte Schritt der Verarbeitung vorgenommen, das Zusammenführen der einzelnen Dateien. Diese Funktion wird vom Reducer übernommen. Er fasst die beiden Dateien wieder zu einer zusammen. Wie auch bei Schritt ③ werden hier die Ergebnisse gegebenenfalls nochmal zusammengefasst. Am Ende steht die Ausgabe-Datei mit allen Key-Value-Paaren der Inputdatei, also alle Wörter aus der Inputdatei mit den jeweiligen absoluten Häufigkeiten: „Bye, 1; Goodbye, 1; Hadoop, 2; Hello, 2; World, 2“.[27]

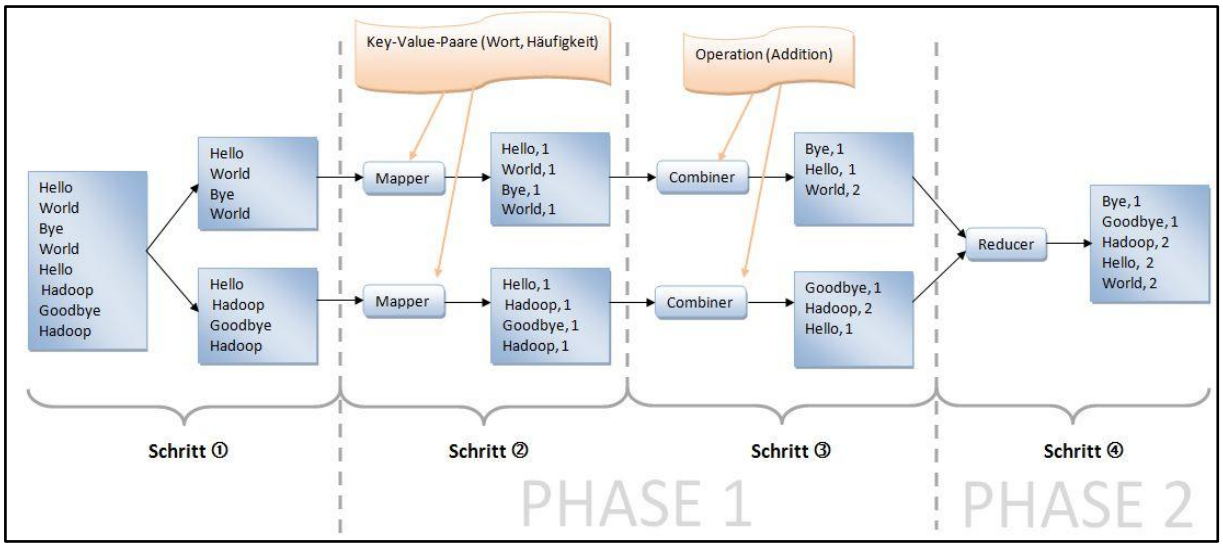


Abb. 2: Schema MapReduce

Beim praktischen Einsatz wird dieser Prozess mit einem enormen Datenvolumen der Input-Datei auf einer Vielzahl von Rechnern bearbeitet. Dies geschieht durch die redundante Verteilung der Daten parallel. Damit wird höchste Effizienz und Sicherheit ermöglicht. Das Potenzial kann erst durch den Großeinsatz der Technologie mit vielen Rechnerknoten und großen Datenmengen ausgeschöpft werden.

[27] Vgl. The Apache Software Foundation (2012i)

4.3 Funktionsweise von Hadoop

Den einzelnen Nodes in einem Hadoop-Cluster sind unterschiedliche Aufgaben zugewiesen. Die HDFS-Nodes werden unter anderem in die bereits angesprochenen NameNodes und DataNodes unterteilt. Wie aus dem Ausarbeitungspunkt 4.1 hervorgeht, gibt es in einem Cluster genau einen NameNode und viele DataNodes.

Hadoop realisiert bei der Einbindung des MapReduce-Verfahrens ebenfalls die Aufteilung der Aufgaben an verschiedene Nodes. Dabei ist zwischen JobTracker- und TaskTracker-Aufgaben zu unterscheiden, welche auf verschiedenen Nodes ausgeführt werden. In dem Hadoop-Cluster existiert ein JobTracker-Knoten. Dieser nimmt die MapReduce-Aufträge des Clients an, koordiniert die Map-Aufgabe zu den jeweiligen TaskTrackern und überwacht deren Ausführung. Die Map-Verarbeitung wird von den TaskTrackern auf den DataNodes ausgeführt. Jeder DataNode ist dabei typischerweise mit einem TaskTracker ausgestattet, der die Map-Anfragen für den gewünschten Datenblock durchführt. Die Ergebnis-Tabellen der einzelnen Map-Prozesse werden anschließend in der Reduce-Phase zu einer Ergebnis-Tabelle zusammenführt. Kommt es bei der Ausführung zu einem Fehler, steuert der JobTracker einen anderen TaskTracker an, um die Verarbeitung erneut auszuführen. Es gibt mehrere TaskTracker, um die Verarbeitung der MapReduce-Prozesse parallel auszuführen. Die TaskTracker erzeugen anschließend jeweils Java Virtual Machines (JVM), um die Map- beziehungsweise Reduce-Funktionen auszuführen.

Der TaskTracker sollte nach Möglichkeit immer auf dem DataNode mit den zu verwendenden Daten ausgeführt werden. Ist dies nicht möglich, sollte ein TaskTracker im gleichen Rack ausgewählt werden, um Performanceeinbußen zu vermeiden. [28]

[28] Vgl. McDonald, K. (2011a)

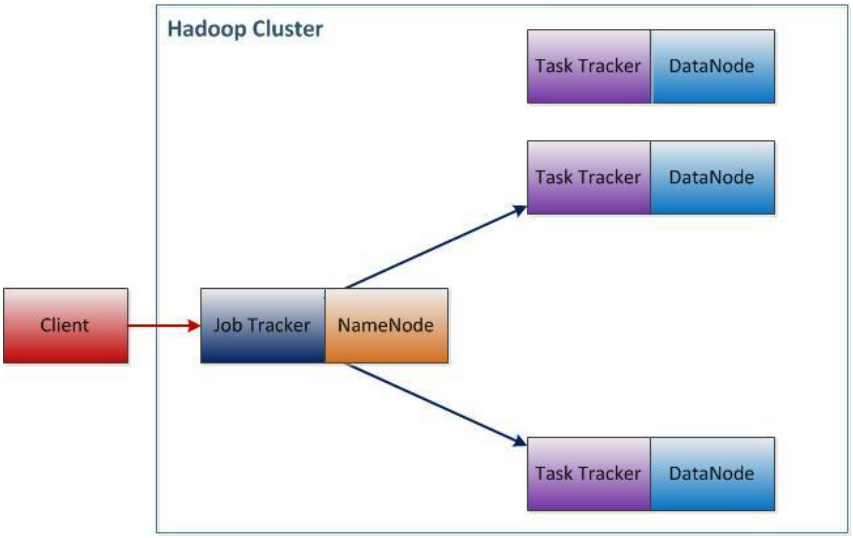


Abb. 3: Kommunikation der Nodes in Hadoop[29]

Typischerweise sind die JobTracker auf der gleichen (virtuellen) Maschine realisiert, wie auch der NameNode. Die TaskTracker sind stets auf den DataNodes implementiert. In Abb. 3 ist die Kommunikation zwischen den Knoten visualisiert. Die Anfrage kommt stets vom Client. In diesem Szenario kommuniziert der Client mit dem JobTracker, er könnte aber auch in einem anderen Anwendungsbeispiel direkt mit dem NameNode oder einem der DataNodes kommunizieren. Nachdem der Client die Anfrage an den JobTracker geschickt hat, bedient dieser die jeweiligen TaskTracker der entsprechenden DataNodes mit den MapReduce-Aufgaben. Die TaskTracker übergeben dabei den Heartbeat an den NameNode.

Wenn ein Client eine Datei anfordert, bekommt dieser vom NameNode gesagt, auf welchem DataNode diese zu finden ist. Der Client liest die Datei darauffolgend direkt vom DataNode. Der DataNode übernimmt das Management der einzelnen Datenblöcke und bedient damit die Anfrage vom Client.[30]

[29] Mit Änderungen entnommen aus: McDonald, K. (2011a)
[30] Vgl. McDonald, K. (2011a)

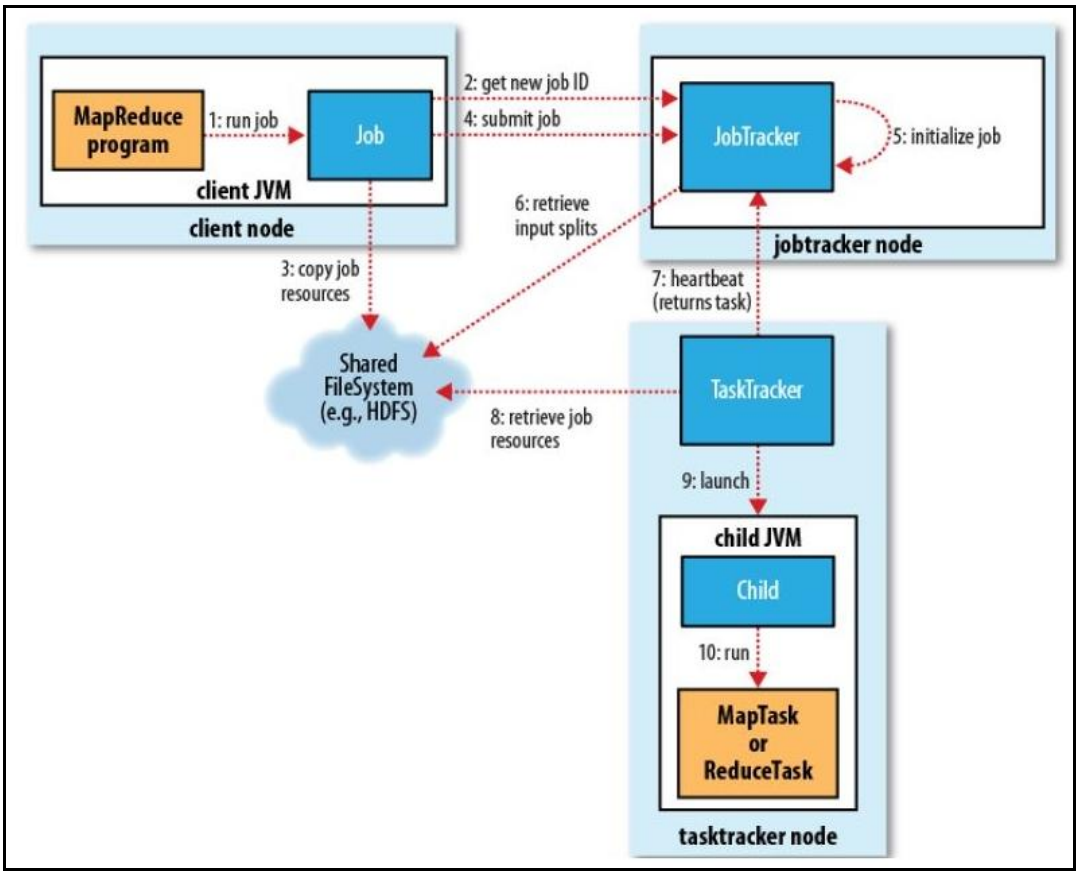


Abb. 4: Prozess-Abfolge einer MapReduce Anfrage [31]

Eine Visualisierung der Prozess-Abfolge einer MapReduce Anfrage in Hadoop findet sich in Abb. 4. Im ClientNode ist der Rechner, von dem aus eine Abfrage ausgeführt wird. In diesem Knoten wird eine sog. Client JVM gestartet. Der Input besteht aus einer MapReduce Anfrage. Die JVM startet anschließend den JobClient. Daraufhin deklariert der Client eine JobID und übergibt diese anschließend an den JobTracker Node. Der Client Node kopiert im dritten Schritt die Daten der Anfrage in das HDFS. Nachdem die beiden Schritte abgeschlossen sind, wird die Abfrage an den JobTracker geschickt. Im fünften Schritt initialisiert dieser die Anfrage lokal. Der JobTracker holt sich vom HDFS (NameNode) die Informationen, auf welchem DataNode sich die jeweiligen Datenblöcke befinden. Daraufhin werden die jeweiligen TaskTracker auf den DataNodes gestartet. Ab diesem Zeitpunkt übertragen die TaskTracker den Heartbeat zyklisch an den JobTracker. Anschließend liest der TaskTracker die Informationen, welche Datenblöcke zu verarbeiten sind, vom HDFS (NameNode). Als nächsten Schritt initialisiert der TaskTracker Node eine weitere lokale JVM, um die Anfrage auszuführen. Im letzen Schritt wird diese Anfrage von der JVM im TaskTracker Node ausgeführt.[32]

[31] Enthalten in: White, T. (2012), S. 191
[32] White, T. (2012), S. 190 ff

Hadoop ist nicht für alle Verarbeitungsarten geeignet. Es eignet sich nicht für Datentransaktionen, da die Daten redundant gespeichert sind und die Verarbeitung nicht auf einem bestimmten Knoten, sondern auf einem zufällig ausgewählten Knoten stattfindet. Des Weiteren ist Hadoop nicht effizient, wenn die Verarbeitung nicht parallelisiert werden auch. Viele kleine Dateien sollten ebenfalls nicht mit Hadoop verarbeitet werden, da diese nicht der Blockgröße des HDFS entsprechen. Sowohl die Speicherung als auch die Verarbeitung beziehungsweise die Kalkulation kleiner Dateien sollte mit einem anderen System gelöst werden. Hadoop ist für die Verarbeitung von Big Data optimiert.[33]

Hadoop wird über die FS Shell gesteuert. Hier werden alle nötigen Befehle eingegeben. Über die Shell werden die Daten in das Hadoop-System eingespielt und manipuliert. Um Auswertungen vorzunehmen, wird die Java Archive-Datei (JAR-Datei) von der FS Shell aus aufgerufen, die den MapReduce-Algorithmus enthält. Um eine MapReduce-Anfrage für Hadoop zu programmieren, sind weitreichende Java-Kentnisse erforderlich.

Diese Ausarbeitung widtmet sich der Theorie zu Hadoop, dessen Basissysteme und den zustätzlichen Komponenten. Detailliertere Einblicke und die praktische Umsetzung der Anwendungen werden auf Grund des formellen Rahmens dieser Arbeit nicht beachtet. Somit wird die Realisierung von aufwendigen MapReduce-Algorithmen auf Java-Basis in dieser Ausarbeitung nicht weiter spezifiziert. Bei den zusätzlichen Komponenten in Kapitel 5 wird aber auf Systeme eingegangen, mit denen Abfragen einfacher erstellt werden können. In weiterführenden Studien wäre ein detailierterer Blick auf die Programmierung von MapReduce-Algorithmen ein durchaus interessanter Aspekt.

Fehlertoleranz:
Um die Effektivität von Hadoop zu gewährleisten, ist es sehr fehlertolerant. Bei der Ausführung von Hadoop gibt es drei Arten von Fehlern. Die erste ist ein Fehler des Tasks, also wenn die Verarbeitung an sich fehlschlägt. Dieser könnte beispielsweise durch einen Programmierfehler in dem MapReduce-Verfahren ausgelöst werden. Die JVM meldet dann dem TaskTracker, dass die Verarbeitung fehlgeschlagen ist. Dieser markiert daraufhin den Versuch als fehlgeschlagen und gibt den Node für eine andere Ausführung frei. Wenn der Task stehenbleibt, wird dies ebenfalls bemerkt. Dies könnte beispielsweise durch einen Hardwarefehler hervorgerufen werden. Der Task wird daraufhin gelöscht und der JobTracker führt den Task auf einem anderen Node aus. Wenn der Fehler weiterhin auftritt, wird die komplette Ausführung abgebrochen. Die zweite Fehlerart bezieht sich auf den TaskTracker. Der JobTracker erwartet in regelmäßigen Abständen einen „Heartbeat“ vom TaskTracker. Wenn

[33] Vgl. Chaudhri, A. (2011)

dieser nicht mehr ankommt, bemerkt der JobTracker den Fehler. Daraufhin wird der TaskTracker aus dem System gelöscht. Die dritte und letzte Fehlerart ist ein Fehler des JobTrackers. Da es nur einen JobTracker in jedem Hadoop-Cluster gibt, ist bei einem solchen Fehler dieser nicht mehr zu beheben und die gesamte Ausführung fehlgeschlagen. [34]

4.4 Anwendungsbereiche

Wie eingangs bereits erwähnt, benutzen hauptsächlich große Firmen, wie Facebook, Twitter oder Yahoo Hadoop und dessen Erweiterungen. Allerdings gibt es viele weitere Anwendungsbeispiele für Hadoop, die im Folgenden beispielhaft aufgelistet werden.

Der Großrechner IBM Watson, der 2011 das Fernseh-Quizz Jeopardy! gewann, arbeitete vor allem beim Einlesen aller relevanten Daten mit Hadoop und MapReduce. Auch große Unternehmen benutzen das Datenbankmodell. Der Telefonanbieter China Mobile nutzt Hadoop Cluster für Data Mining, um alle Telefongespräche abzurechnen. Hierbei wird eine Datenmenge von ca. fünf bis acht Terabyte pro Tag verarbeitet. Die Nutzung von Hadoop ermöglicht es China Mobile, bis zu 10-mal mehr Daten zu verarbeiten als mit herkömmlichen Systemen. Zudem erfolgt eine erhebliche Kosteneinsparung, da nunmehr nur etwas 1/5. der Kosten anfallen. Als weiteres Beispiel ist die New York Times zu nennen. Die Zeitungsgesellschaft plant, alle Artikel zwischen den Jahren 1851 und 1922 online zu stellen. Hierfür mussten 11 Millionen Bilddateien in PDF-Dokumente umgewandelt werden, was ein Datenvolumen von ca. 1,5 Terabyte ausmachte. Für die Umsetzung dieses Projekts benötigte ein einzelner Mitarbeiter mit Hilfe eines Amazon EC2 Hadoop Clusters nur 24 Stunden.[35]

Zudem können folgende praxisnahe Anwendungsbereiche für Versicherungsunternehmen genannt werden. Die Suche innerhalb eines Unternehmens-Wikis kann durch die Realisierung über ein Hadoop-System vereinfacht. Diese semi-strukturierten Daten fallen genau in Definition von Big Data und sind somit für eine effiziente Verarbeitung mittels Hadoop geeignet. Zudem ist vorstellbar, dass versicherungsspezifische Auswertungen über ein Hadoop-System gestartet werden. Beispielsweise könnten Unfallmeldungen eingelesen und verarbeitet werden, um Statistiken zu erstellen. Mit diesen Statistiken kann beispielsweise ausgewertet werden, wie viele Versicherungsfälle welcher Art in einer bestimmten Region aufgetreten sind, um so Gefahrenklassen zu erstellen.

[34] Vgl. McDonald, K. (2011b)
[35] Vgl. Chaudhri, A. (2011)

5 Zusätzliche Komponenten / Ergänzende Systeme

Hadoop kann durch weitere Komponenten ergänzt werden. Diese erfüllen die unterschiedlichsten Funktionen, darunter die Übertragung, Abfrage, Darstellung und Aggregation von Daten sowie deren Auswertung im Data Warehouse-Prinzip (DWH). Im Folgenden werden einige dieser ergänzenden Systeme vorgestellt und ihre Funktionsweise sowie ihr Anwendungsbereich näher beschrieben.

5.1 Spaltenorientierte Datenbank HBase

Spaltenorientierte Datenbanken [36] [37]

In einem relationalen Datenbanksystem (DBS) sind die Daten als Datensätze in Tabellen mit Reihen und Spalten dargestellt und werden zeilenweise abgefragt. Relationale DBS eignen sich hiermit gut für Abfragen, in die viele Attribute mit einbezogen, jedoch nur wenige Ergebniszeilen erwartet werden.

In einer spaltenorientierten Datenbank (DB) hingegen werden die Attribute jeweils in eigenen kleinen Tabellen gespeichert. Somit können Abfragen, die wenige oder nur ein Attribut mit einbeziehen aber viele Datensatzzeilen zurückliefern sollen, effizienter durchgeführt werden.

Hier ein Beispiel einer Datentabelle, wie sie in einem relationalen DBS und in einer spaltenorientierten DB aussehen kann:

EmpId	**Lastname**	**Firstname**	**Salary**
1	Schmidt	Martin	45000
2	Hahn	Maria	51000
3	Johann	Kathrin	42000

Das DBS muss technisch bedingt die Daten aus den Tabellen in Reihen von Strings auf die Festplatte schreiben. Diese Strings sehen bei einem relationalen DBS, welches Datensatzweise speichert, wie folgt aus:

1,Schmidt,Martin,45000;
2,Hahn,Maria,51000;
3,Johann,Kathrin,42000;

[36] Vgl. Wartala, R. (2012), S. 139 f
[37] Vgl. IBM Corporation (2012)

In der spaltenorientierten DB hingegen werden die Daten spaltenweise in die Strings geschrieben und sehen somit wie folgt aus:

1,2,3;
Schmidt,Hahn,Johann;
Martin,Maria,Kathrin;
45000,51000,42000;

Diese differenzierte Art der Persistierung (Speicherung) ermöglicht es einem DWH, wie z.B. Hive, auf welches später noch eingegangen wird, die Daten einfacher zu aggregieren und auch strukturiert abzufragen.

HBase

Apache HBase ist eine verteilte, skalierbare, spaltenorientierte Datenbank für große Datenmengen, welche ganz speziell für die Hadoop Technologie verwendet wird. [38] Es setzt auf der Hadoop Technologie HDFS auf und ist ein Open Source Produkt. Im Gegensatz zu relationalen DB bietet HBase dem Anwender keine Abfragesprache, wie z.B: SQL, denn HBase ist nicht im entferntesten eine relationale DB. HBase-Anwendungen sind, genau wie MapReduce-Algorithmen, in Java geschrieben. [39]

Technisch gesehen ist HBase mehr ein Datenspeicher als eine Datenbank, denn es fehlt HBase an vielen Funktionen, die ein relationales Datenbankmanagementsystem (RDBMS) im Normalfall zur Verfügung stellt, wie z.B. Sekundärschlüssel, Trigger (Auslöser), komplexere Abfragesprachen etc. [40]

HBase ist dazu ausgelegt, mit Hilfe des HDFS große Datenmengen zu bewältigen und wird von einigen bekannten Firmen, wie z.B. Facebook und Adobe, als hochverfügbarer Datenspeicher oder zur Verarbeitung von Nachrichten genutzt. [41]

Hadoop/HDFS vs. HBase

HDFS ist ein verteiltes Dateisystem, das sich sehr gut eignet, um große Dateien zu speichern. Jedoch bietet es keine Möglichkeit, schnelle, individuelle Abfragen aus dem Dateibestand zu generieren. HBase hingegen setzt auf HDFS auf und bietet einem genau diese Möglichkeit der schnellen, individuellen Datenabfrage auf großen Tabellen. [42]

[38] Vgl. The Apache Software Foundation (2012c)
[39] Vgl. IBM Corporation (2012)
[40] Vgl. The Apache Software Foundation (2012c)
[41] Vgl. Wartala, R. (2012), S. 141
[42] Vgl. The Apache Software Foundation (2012c)

Wofür nutzt man HBase?

Um HBase effektiv nutzen zu können, benötigt man eine sehr große Menge an Daten (viele Millionen oder gar Billionen von Datensätzen), die verwaltet werden sollen. Zudem sollte man sich darüber im Klaren sein, dass man auf einige Funktionen, die einem ein RDBMS bietet, verzichten muss. Eine Anwendung, die momentan auf ein RDBMS zugreift, kann nicht einfach so umgeschrieben werden, dass sie in Zukunft auf HBase zugreift. Hierzu bedarf es eines kompletten Redesigns der Anwendung inklusive der Schnittstellen. Zu guter Letzt benötigt man für die Nutzung von HBase ausreichend Hardware. Hier sind einige DataNodes plus NameNode nötig, um HBase (wie auch HDFS) überhaupt lauffähig zu bekommen. Nur im Ausnahmefall, zum Beispiel zu Test- oder Konfigurationszwecken, kann HBase auf einem einzigen Rechner installiert werden.[43]

Architektur

HBase ist nach dem bekannten Client-Server-Modell aufgebaut. Dabei werden in HBase alle Daten in Tabellenform, ähnlich einer überdimensionalen Excel-Tabelle, gespeichert. Werden Daten in die Tabelle eingefügt, wird der zelleneigene Zeitstempel aktualisiert und somit der Zellinhalt versioniert. Über Primärschlüssel werden die einzelnen Zeilen der Tabelle eindeutig referenziert. Durch das Bilden von Spaltenfamilien (Column Families) werden in HBase die Spalten einer Tabelle gruppiert.[44]

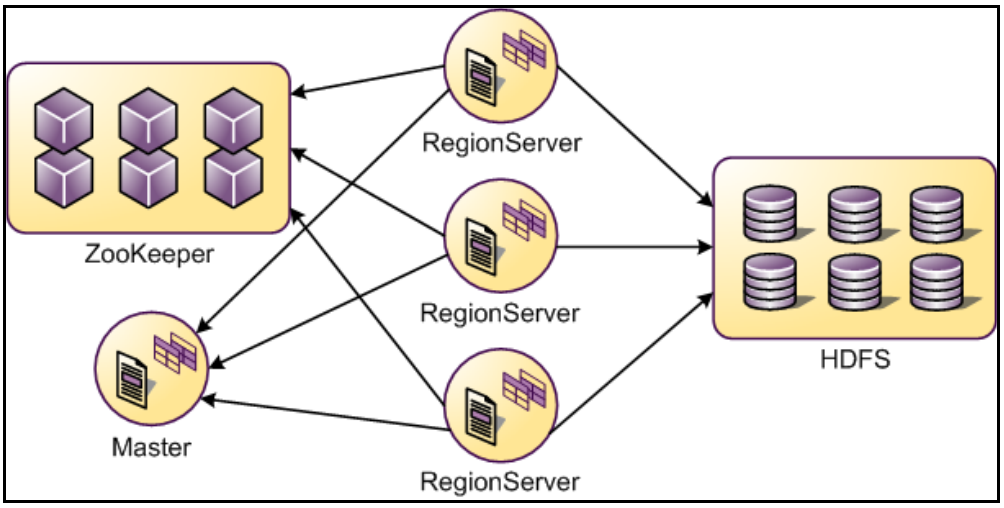


Abb. 5: Architektur HBase[45]

Ähnlich der Architektur von HDFS hat HBase einen Master Node, der die Cluster verwaltet und in jedem Cluster mehrere RegionServer, die Teile der Tabellen speichern und die Arbeit

[43] Vgl. The Apache Software Foundation (2012c)
[44] Vgl. Wartala, R. (2012), S. 142
[45] Enthalten in: Figuière, M. (2009)

auf den Daten ausführen (Vgl. Abb. 5). Genau wie HDFS ist die Schwachstelle von HBase der Verlust des MasterNode.[46] [47]

Über den Zeilenschlüssel jeder Zeile einer Tabelle können diese in Regionen aufgeteilt werden. Eine Region beinhaltet mehrere Zeilen einer Tabelle. Die verschiedenen Regionen einer Tabelle können auf verschiedene Region Server verteilt gespeichert werden. Die Verteilung auf die einzelnen Server überwacht der MasterServer. Besteht ein Cluster in HBase aus einem Master- und mehreren RegionServern, so werden diese auch Master- und SlaveNodes genannt. Status und Konfigurationen von Master- und SlaveNode werden in diesem Beispiel mit Hilfe von ZooKeeper gespeichert.[48]

Installation und Anwendung

Einzige Voraussetzung für die Installation einer Standalone-Lösung von HBase ist ein Rechnersystem mit der Laufzeitumgebung Java Runtime Environment (JRE) Version 6. Die Installation auf einem Rechner-Cluster ist jedoch weitaus aufwändiger. Hierfür sind mehrere komplexe Installations- und Konfigurationsprozedere sowie die Anbindung an das HDFS und ZooKeeper notwendig.[49]

Um mit den HBase Servern zu interagieren, gibt es die HBase Shell. Hierüber können fünf verschiedene Arten von Kommandos eingegeben werden, die den folgenden Zwecken dienen: [50]

- Data Definition (Beschreibung der Daten und Datenstrukturen)
- Data Manipulation (Lesen und Schreiben von Daten)
- Replikation von HBase
- Steuerung der internen Werkzeuge
- Sonstige Statusabfragen

Neben dieser Shell gibt es noch weitere Schnittstellen, über die HBase mit anderen Systemen kommunizieren kann. Standardmäßig bringt HBase drei Schnittstellen mit, über welche andere Anwendungen Zugriff auf die Daten und Funktionalitäten erhalten können. Diese sind:[51]

- Java-API
- Representational State Transfer (REST)

[46] Vgl. IBM Corporation (2012)
[47] Vgl. The Apache Software Foundation (2012c)
[48] Vgl. Wartala, R. (2012), S. 144
[49] Vgl. Wartala, R. (2012), S. 144 ff
[50] Vgl. Wartala, R. (2012), S. 147
[51] Vgl. Wartala, R. (2012), S. 150 ff

- Thrift (Services, die mit verschiedenen Programmiersprachen arbeiten können)

In der Zukunft soll auch noch der Zugriff auf HBase über Apache Avro, ein Daten-Serialisierungs-Framework, ermöglicht werden.[52] [53]

5.2 Hadoop Streaming

Üblicherweise werden MapReduce Anwendungen mit Hilfe der Hadoop-API in Java programmiert. Da es jedoch seitens der Entwickler programmspezifische Vorgaben gibt, in welcher Sprache sie programmieren müssen, bietet Hadoop eine Streaming-Schnittstelle an. Mit dieser können Daten über die Standard-Ein- und Ausgabe „gestreamt" werden. Hierfür benötigt man das Hadoop-Java Archive (JAR) hadoop-streaming-0.20.2.jar. Übergibt man diesem JAR das Mapper- und Reducer-Skript sowie Ein- und -Ausgabedaten (Comma separated values (CSV) Dateien) als Parameter, so werden diese von der Streaming-Anwendung in die Hadoop-Umgebung geladen. Diese Technologie ermöglicht nicht nur die Programmierung von Mappern und Reducern in anderen Sprachen sondern zusätzlich, MapReduce-Anwendungen in lokalen Hadoop-Umgebungen zu testen.[54]

Zur Verdeutlichung hier eine detailliertere Beschreibung dieses Hadoop-Streaming-Dienstes:[55]

```
$HADOOP_HOME/bin/hadoop jar hadoop-streaming.jar \
    -input myInputDirs \
    -output myOutputDir \
    -mapper cat \
    -reducer wc
```

Im oben stehenden Beispiel sind Reducer und Mapper ausführbare Dateien, welche die Eingabeparameter Zeile für Zeile aus der Inputdatei einlesen und die (resultierenden) Ausgabeparameter in die Ausgabedatei schreiben. Der Streaming-Dienst erzeugt einen MapReduce Job, übergibt diesen an das passende Cluster und überwacht den Ablauf des Jobs bis dieser beendet ist.

[52] Vgl. Wartala, R. (2012), S. 155
[53] Vgl. ebenda, S. 156
[54] Vgl. ebenda, S. 95 ff
[55] Vgl. The Apache Software Foundation (2011)

Ist ein JAR für einen **Mapper** bestimmt, führt jede Map-Anwendung die Datei als einen separaten Prozess aus, sobald der Mapper initialisiert wird. Sobald der Map-Prozess läuft, wandelt er seine Eingabeparameter in Zeilen und schreibt diese in die Eingabedatei des Prozesses. Währenddessen sammelt der Mapper die zeilenweisen Ausgabeparameter aus der Ausgabedatei des Prozesses und wandelt jede Zeile in Key-Value-Paare um, die wiederum als Ausgabeparameter des Mappers gesammelt werden. Standardmäßig bildet der erste Teil einer Zeile bis hin zum ersten Tabstopp den Key (Schlüssel) und der Rest der Zeile (ohne Tabstopp) den Value (Wert) des Key-Value-Paares. Existiert kein Tabstopp in der Zeile, so wird die gesamte Zeile als Key betrachtet und der Value als null betrachtet. Diese Interpretation kann jedoch angepasst werden.

Ist ein JAR für einen **Reducer** bestimmt, führt jede Reduce-Anwendung die Datei als einen separaten Prozess aus, sobald der Reducer initialisiert wird. Sobald der Reduce-Prozess läuft, wandelt er seine Key-Value-Paare in Zeilen um und schreibt diese in die Eingabedatei des Prozesses. Währenddessen sammelt der Reducer die zeilenweisen Ausgabeparameter aus der Ausgabedatei des Prozesses und wandelt jede Zeile in Key-Value-Paare um, die wiederum als Ausgabeparameter des Reducers gesammelt werden. Wie bereits zuvor für die Map-Anwendung beschrieben, bildet der erste Teil einer Zeile bis hin zum ersten Tabstopp den Key und der Rest der Zeile (ohne Tabstopp) den Value des Key-Value-Paares. Wie auch beim JAR für den Mapper wird die gesamte Zeile als Key und der Value als null betrachtet sofern kein Tabstopp in der Zeile existiert.

Dieses Vorgehen bildet die Basis für das Kommunikationsprotokoll zwischen dem MapReduce Framework und der Streaming-Anwendung.

5.3 Datenfluss-Sprachen

Basierend auf Hadoop gibt es drei verschiedene Datenfluss-Sprachen:

- Hive
- Pig
- CloudBase

Alle drei unterscheiden sich in den Ansätzen, die sie realisieren. Hive stellt eine SQL-ähnliche Abfragesprache zur Verfügung, Pig implementiert eine Art Skriptsprache mit Variablen und Funktionsaufrufen und CloudBase funktioniert ähnlich wie Hive, nur dass sie mehr SQL zu implementieren versucht und dafür auf Metadaten-Verarbeitung verzichtet. Daher wird in dieser Ausarbeitung nicht näher auf CloudBase eingegangen. [56]

[56] Vgl. Wartala, R. (2012), S. 100 f

5.3.1 Hive

Ab 2008 versuchte sich Facebook mit Hilfe des Projektes Hive an der Überführung von SQL-ähnlichen Abfragen in MapReduce-Jobs. Dies sollte dazu dienen, mittels einfacher HiveQL-Abfragen auch Java-fremden Mitarbeitern die Datenanalyse zu ermöglichen. Das Projekt wurde ab 2009 von der Apache Foundation fortgeführt und damit der Allgemeinheit zugänglich gemacht.[57]

Hive kann als eine Art DWH-System für Hadoop verstanden werden. Es setzt auf den Basistechnologien von Hadoop auf und beinhaltet einen Mechanismus, der die gespeicherten Daten zusammenfasst, aggregiert und ihnen eine Struktur gibt. Kurz gesagt, es stellt Hadoop eine DWH-Infrastruktur zur Verfügung. [58] Es ist eine Art Client innerhalb der Hadoop-Umgebung und nutzt das HDFS als Basis für seine Analysen. Daten aus verschiedensten Quellen können mit Hilfe von Hive aggregiert werden. Ebenfalls realisierbar sind die Funktionen ETL von Daten (Extract, Transform, Load). [59]

Da Hive auf Hadoop, welches ein Batch-verarbeitendes Programm ist, aufsetzt, benötigt es sehr lange Antwortzeiten bei Abfragen. [60] Es ist nicht dazu gedacht, schnelle Echtzeitabfragen auf die Daten auszuführen. Durch den MapReduce-Prozess, welcher für jede Abfrage zunächst ausgeführt werden muss, beläuft sich die Antwortzeit bei Abfragen auf mehrere Minuten bis Stunden. Im Bereich der Big Data ist dies jedoch vertretbar, da hier sehr große Datenmengen verarbeitet werden, die ein Relationales DBS gar nicht leisten könnte. [61]

Bei Hive besonders hervorzuheben sind seine Skalierbarkeit, Erweiterbarkeit und Fehlertoleranz.[62]

[57] Vgl. Wartala, R. (2012), S. 101
[58] Vgl. The Apache Software Foundation (2012a)
[59] Vgl. Wartala, R. (2012), S. 101
[60] Vgl. The Apache Software Foundation (2012a)
[61] Vgl. Wartala, R. (2012), S. 101
[62] Vgl. The Apache Software Foundation (2012b)

Architektur

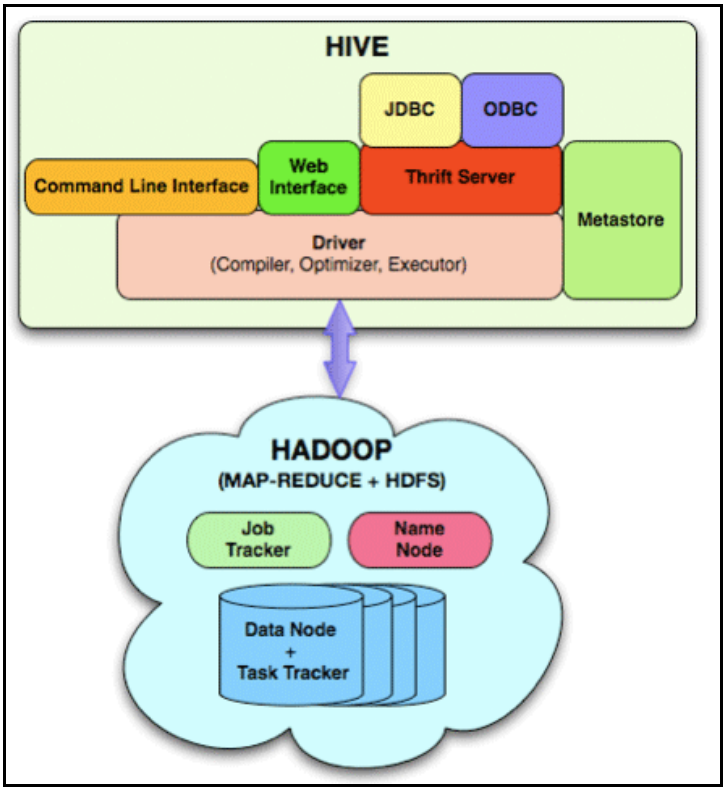


Abb. 6: Hive-Architektur und Kopplung mit Hadoop[63]

Abb. 6 stellt die verschiedenen Komponenten von Hive dar. Hierzu gehört der **Metastore**, in welchem die Datenbank, Tabellen und Views als Objekte verwaltet werden. Standardmäßig wird hier die Datenbank Derby (Apache Foundation) als Megastore genutzt. Die Tabellenverwaltung geschieht partitioniert, d.h. die Daten werden logisch nach gesetzten Kriterien gruppiert und auf verschiedene Tabellen verteilt (z.B. alle Daten eines Jahres, Artikels oder Kunden etc.). [64]

Hive lässt sich am besten und schnellsten über die **Kommandozeile** (Command Line Interface) ausführen. In einer Shell müssen die Hive-Befehle eingegeben werden. Die Trennung der Befehle erfolgt, wie auch in SQL, mit Semikolon.[65] Alternativ lässt sich Hive auch webbasiert nutzen. Hier erfolgt der Zugriff auf die Daten mittels des **Hive Web Interface**. [66]

Der Query Language Compiler (innerhalb der Treiber-Komponente) übersetzt eingegebene oder über eine standardisierte Datenbankschnittstelle, wie z.B. Java Database Connectivity

[63] Enthalten in: Cubrid (2012)
[64] Vgl. Wartala, R. (2012), S. 104
[65] Vgl. ebenda, S. 102
[66] Vgl. ebenda, S. 105

(JDBC) bzw. Open Database Connectivity (ODBC), übermittelte Befehle, optimiert sie und setzt sie in Mapper-/Reducer-Jobs um, die dann an das Hadoop-System übergeben werden.[67] Über die Kommandozeile lässt sich Hive auch als **Thrift-Dienst** starten. Ein Thrift-Dienst dient der Erstellung von Services, die mit verschiedenen Programmiersprachen arbeiten können. Dadurch können Thrift-Clients auf Hive zugreifen, die in anderen Sprachen als Java programmiert wurden. Eine weitere Funktion in diesem Modus ist der Zugriff auf Hive über **JDBC- oder ODBC-Treiber**. [68]

Datenabfrage, -definition und -manipulation

Die Daten können mittels einer SQL-ähnlichen Abfragesprache (HiveQL) ad hoc abgefragt und analysiert werden. HiveQL ermöglicht den Entwicklern die Verwendung einer Syntax, die der von SQL gleicht. Alternativ können MapReduce Entwickler damit ihre eigenen „Mapper“ und „Reducer“ dort benutzen, wo es umständlich bzw. ineffizient ist, diese Logik in HiveQL auszudrücken. [69]

Vergleicht man Hive mit relationalen DBS, so verfügt es nur über einen Bruchteil an Befehlen, die man wie folgt unterteilen kann: [70]

- DDL – Datenbeschreibungssprache zur Erzeugung, Modifikation und Löschen von Tabellen-Schemata und Views; z.B. CREATE TABLE; ALTER TABLE
- DML – Datenmanipulationssprache zum Laden und Schreiben von Daten in Hive und aus Hive heraus
- HiveQL – Datenabfragesprache zum Selektieren, Gruppieren, Verknüpfen und Transformieren von Daten innerhalb von Hive

Genau wie mit SQL kann man mit Hive Datenbankabfragen mittels der SELECT-Anweisung ausführen, Daten aus verschiedenen Tabellen sortieren oder gruppieren und das Ergebnis dieser Abfragen wiederum in einer neuen Tabelle speichern. Zudem ist es mit Hive möglich, mehrere Tabellen auf einmal mit Daten aus einer Quelldatei / Quelltabelle zu aktualisieren. Ein weiterer Vorteil von Hive ist die Möglichkeit des Zugriffs auf Daten, welche in HDFS oder auch HBase gespeichert sind sowie die Abfrageausführung über MapReduce.[71] [72]

[67] Vgl. Wartala, R. (2012), S. 105
[68] Vgl. ebenda, S. 105
[69] Vgl. The Apache Software Foundation (2012a)
[70] Vgl. Wartala, R. (2012), S. 102
[71] Vgl. ebenda, S. 109
[72] Vgl. The Apache Software Foundation (2012b)

Funktionen

Hive implementiert zudem eine Vielzahl an Funktionen, welche über die Abfragekommandos aufgerufen werden können. Hierzu zählen unter Anderem mathematische Funktionen, Funktionen zur Verarbeitung diverser Angaben (Datum, Zeit) sowie User Defined Functions (UDF), welche vom Anwender selbst erstellt und in Hive eingebunden werden können.[73] [74]

Datenaustausch

Eine der wichtigsten Schnittstellen für den Datenaustausch mit Hive ist die JDBC. Mit dieser Datenbankschnittstelle kann die Oberflächen-Anwendung SQuirreL genutzt werden, um Daten mit Hilfe von Hive abzufragen. Mit SQuirreL ist auch ein Export der Ergebnisdaten aus Hive in Excel-, CSV- oder XML-Dateien möglich.[75]

5.3.2 Pig

Pig ist, ähnlich wie Hive, eine Plattform, um große Datenmengen zu analysieren und abzufragen. Die verwendete Sprache, Pig Latin, ist eine einfache Abfragesprache, mit der man Abfragen, Sortierungen sowie Gruppierungen vornehmen kann. Außerdem können, ebenso wie mit Hive, einfache Funktionen und auch UDF aufgerufen werden.[76] [77]

Pig eignet sich auch für Entwickler und Nicht-Entwickler, die nicht viel Erfahrung mit MapReduce haben, da Pig-Latin-Skripte automatisch in Map-Reduce-Jobs übersetzt und ausgeführt werden.[78]

Auch Pig benötigt, ebenso wie Hive, für die Installation nur eine JRE Version 6 und kann entweder lokal oder aber auf einem Hadoop-Cluster laufen.[79]

Pig Latin

Die Sprache, um Abfragen in Pig zu realisieren ist Pig Latin. Sie besitzt folgende Eigenschaften:[80]

- Einfachheit (auch parallele Ausführung von komplexen Abfragen auf großen Datenmengen einfach nachvollziehbar)

[73] Vgl. Wartala, R. (2012), S. 110
[74] Vgl. The Apache Software Foundation (2012b)
[75] Vgl. Wartala, R. (2012), S. 111 ff
[76] Vgl. The Apache Software Foundation (2012e)
[77] Vgl. Wartala, R. (2012), S. 114
[78] Vgl. ebenda, S. 114
[79] Vgl. ebenda, S. 115
[80] Vgl. The Apache Software Foundation (2012d)

- Optimierungsmöglichkeit (System bestimmt die Reihenfolge der Abfrageausführung selbst und optimiert dadurch automatisch den Abfrageablauf)
- Erweiterbarkeit (User kann eigene Funktionen implementieren und ausführen; Pig lässt sich damit individualisieren)

Im Folgenden werden die verschiedenen Typen zur Datenrepräsentation, welche in Pig Latin verwendet werden, beschrieben:[81]

Atom - einzelner Wert (Zahl, Zeichenkette)

Tuple - komplexer Datentyp (Reihe unterschiedlicher Atome)

Bag – ungeordnete Menge an Tupeln

Map – Menge von Schlüssel-Wertpaaren

Die Syntax von Pig Latin ist recht einfach, da sich lediglich Wertzuweisungen, nicht aber Schleifen und Kontrollstrukturen realisieren lassen. Zudem ist es nicht case-sensitive. Als Schnittstelle für die Arbeit mit Pig dient die Grunt-Shell, welche über die üblichen Funktionalitäten, wie aus anderen Shells bekannt, verfügt. Besonderheiten sind jedoch die Historisierung der eingegebenen Kommandos auf der Shell-Oberfläche sowie der Auto-Complete-Mechanismus.[82]

Datenoperatoren & Entwicklung

Pig Latin ermöglicht das Ausführen einiger relationaler Operationen, wie z.B. FOREACH, FILTER, GROUP, ORDER, JOIN etc. Diese ähneln denen der Structured Query Language (SQL) sehr.[83] Die Skripte für Pig Latin können sowohl über die Grunt-Shell als auch über andere Entwicklungsumgebungen, wie z.B. Eclipse oder TextMate editiert werden.[84]

5.4 Datenserialisierung mit Sqoop

Mit Sqoop werden Daten aus einem RDBMS gelesen und in ein Hadoop System übertragen sowie umgekehrt Daten aus Hadoop an ein RDBMS übergeben. Sqoop ist als Java-

[81] Vgl. Wartala, R. (2012), S. 115
[82] Vgl. ebenda, S. 115 f
[83] Vgl. ebenda, S. 125 ff
[84] Vgl. ebenda, S. 131

Anwendung realisiert und nutzt MapReduce Methoden zur Parallelisierung und zur Fehlervermeidung. Hierdurch ist es möglich sämtliche RDBMS, für die es einen JDBC-Treiber gibt, anzubinden. Zusätzlich bringt Sqoop nativen Support für MySQL ab Version 5 sowie PostgreSQL ab Version 8.3 mit.[85]

SQL-to-HDFS

Sqoop wird über die Kommandozeile gesteuert. Bei der Ausführung eines Import-Befehls erzeugt Sqoop eine Java-Klassendatei mit dem Namen der zu importierenden SQL-Tabelle. In dieser Datei ist die Schnittstelle zum Hadoop Dateisystem mit den notwendigen Getter- und Setter-Methoden sowie den Transformationen definiert. Jede Zeile der Tabelle wird im HDFS als kommaseparierte Datei eingefügt. Man kann die Anzahl der parallelen Verarbeitungen festlegen, indem man die Anzahl der verwendeten Mapper über die Kommandozeile mit dem Parameter *num-mappers* angibt.[86] Mit den Argumenten *where* , *columns* und *query* kann die Menge der zu importierenden Zeilen eingeschränkt werden.[87]

SQL-to-Hive

Mit Sqoop können Tabellen aus einem RDBMS auch direkt in das Hive Data Warehouse System von Hadoop überführt werden. Sqoop mappt hierbei automatisch das Tabellenschema der SQL Datenbank auf das entsprechende Schema im Data Ware House.[88]

SQL-to-HBase

Die dritte Fähigkeit von Sqoop besteht darin, die SQL-Tabellen direkt in das Hadoop Datenbanksystem HBase zu übertragen. HBase benötigt einen Primärschlüssel in seinen Tabellen. Sqoop verwendet automatisch den Primärschlüssel aus der SQL Tabelle, sofern dieser vorhanden ist. Die Spalte mit dem Primärschlüssel kann jedoch auch beim Import vom Entwickler festgelegt werden.[89]

Hadoop-to-SQL

Um die mittels Hadoop verarbeiteten Daten wieder im Quellsystem verwenden zu können bietet Sqoop eine Exportfunktion an. Mit dieser können die Daten wieder in das RDBMS übertragen werden. Hierbei setzt Sqoop das Vorhandensein der Zieltabelle im RDBMS voraus.[90]

---

[85] Vgl. Wartala, R. (2012), S. 180
[86] Vgl. The Apache Software Foundation (2012f)
[87] Vgl. ebenda, S. 180 ff
[88] Vgl. ebenda, S. 183
[89] Vgl. ebenda, S. 183
[90] Vgl. ebenda, S. 184

5.5 ZooKeeper

ZooKeeper wurde konzipiert als ein Dienst, der die Verwaltung von Konfigurationen verteilter Systeme übernimmt, dabei möglichst einfach ist und selbst als verteiltes System auf mehreren Servern läuft. Hierdurch soll die Ausfallsicherheit des Dienstes sichergestellt werden. Um eine möglichst hohe Performance zu erreichen, hält ZooKeeper seine Daten im Speicher (in-Memory) und gewährleistet so eine geringe Zugriffszeit und einen hohen Datendurchsatz. ZooKeeper verwaltet die Dienste, die er überwacht, in einer Dateisystem-ähnlichen Baumstruktur. Die Blätter des Baums werden als *ZNodes* bezeichnet. Ein *ZNode* kann, anders als bei einem normalen Dateisystem, sowohl ein Ordner sein als auch Daten beinhalten. Die Datenmenge ist pro *ZNode* auf 1MB begrenzt. Da es sich aber nur um Statusinformationen der überwachten Dienste handelt, bewegt sich die Datenmenge einer *ZNode* normalerweise im Byte bis Kilobyte Bereich.[91]

Der Zugriff auf eine *ZNode* erfolgt über die UNIX Standardnotation für Pfade (/Node1/ChildNode1). Es gibt zwei Arten von *ZNodes*. Die regulären *ZNodes* werden vom Client erzeugt und zerstört, die kurzlebigen (*emphemeral*) *ZNodes* werden vom Client erzeugt, der Server entscheidet jedoch selbst, wann diese wieder zerstört werden.[92]

Ein weiteres Konzept im ZooKeeper sind die *Watches*: Ein Client kann auf eine beliebige *ZNode* eine Überwachung (*Watch*) registrieren und wird daraufhin durch den ZooKeeper Dienst über eine Änderung in dieser *ZNode* hingewiesen. Dies wird verwendet, um u.a. den gleichzeitigen Start von Berechnungen auf mehreren Systemen sicher zu stellen.[93]

ZooKeeper als verteiltes System

Um die Ausfallsicherheit zu gewährleisten, läuft der ZooKeeper Dienst auf mehreren Servern. Je nach Konfiguration können einer oder mehrere Server ausfallen, ohne dass dadurch die Funktion des Cluster beeinträchtigt wird. Hierzu replizieren sich die Server untereinander. Ein Cluster wird bei ZooKeeper als *Ensemble* bezeichnet. Ein Server ist der *Leader,* die anderen Server sind *Follower*. Schreibzugriffe der Clients werden an den *Leader* weitergeleitet, dieser sendet diese als Vorschläge an die *Follower.* Die wiederum akzeptieren die Änderung über ein internes Nachrichtensystem und validieren sie damit. Mit diesem Nachrichtensystem wird außerdem die Rolle des *Leaders* zwischen den *Followern* neu zugewiesen, wenn der *Leader* ausfällt. Um diese Synchronizität innerhalb des *Ensembles* zu erreichen, verwendet ZooKeeper Versionsnummern für jedes Update, das ein Client sendet. Damit wird sichergestellt, dass die Updates der verbundenen Clients in der richtigen Reihenfolge ausgeführt

[91] Vgl. Wartala, R. (2012), S. 216 f
[92] Vgl. ebenda, S. 216 f
[93] Vgl. ebenda, S. 218

werden. Außerdem gilt das Transaktionsprinzip: Updates sind atomar, entweder werden sie ganz ausgeführt oder sie werden komplett verworfen. [3]

Jeder Server hält die Informationen des gesamten *Ensembles* im Speicher bereit. Ein Abbild der Konfiguration wird im persistenten Speicher aller Server abgelegt. Grundsätzlich verbindet sich ein Client nur mit einem ZooKeeper Server. Er besitzt jedoch eine Liste aller Server, um sich beim Ausfall seines zugewiesenen Servers automatisch mit einem anderen zu verbinden. Aus diesem Grund kann jeder Server alle Informationen zur Verfügung stellen.[94]

6 Prototyping

6.1 Simulation auf PC (Pseudo-Distributed Operation)

Die einfachste und grundlegende Installation eines Hadoop Systems ist die Single-Node-Instanz. Dabei befinden sich NameNode und DataNode auf einem Rechner. Diese Art der Installation ist keineswegs nur als Spielwiese für Profis gedacht, sondern für alle Interessierten, die das Prinzip und die Funktionsweise von Hadoop kennen lernen und verstehen möchten. Das in Anhang 1 befindliche Installationsbeispiel beschreibt die Installation und Konfiguration einer Single-Instanz auf Basis von Ubuntu Desktop 12.10 in der 32-Bit-Version. Zusätzlich werden die Voraussetzungen geschaffen, um diesen Single-Node später z.B. an Amazons Cloud-Computing-Plattform anbinden zu können. Um mehr Verständnis für die Arbeitsweise und die Architektur von Hadoop zu bekommen, wird auf die Hadoop-Version 1.0.4 der Apache Foundation zurückgegriffen.

6.2 Cloud am Beispiel von Amazon Web Services mit Amazon S3

Bei Amazon Simple Storage Service (S3) handelt es sich um Web-Speicher im Cloud-Format. Flexible Skalierbarkeit der Daten im Netz ist der größte Pluspunkt dieser Speichertechnik. Mittels einer Webservice-Schnittstelle können jederzeit von jedem beliebigen Ort aus Daten abgerufen und gespeichert werden. Auch Amazon selbst nutzt S3 zum Ausführen seines globalen Website-Netzwerks.[95]

[94] Vgl. The Apache Software Foundation (2012g)
[95] Vgl. Amazon Web Services, Inc. (2012a)

Amazon S3 verfügt über eine Vielzahl an **Funktionen**, unter Anderem:[96]

- schreiben, lesen, löschen von Dateien bis 5 Terabyte (TB)
- unbegrenzte Anzahl speicherbarer Objekte
- Speicherung der Objekte in Buckets, die individuell abgerufen werden können
- verschiedene Regions-Server stehen zur Speicherung zur Verfügung
- Mechanismen zur Authentifizierung und Rechteverwaltung
- sicherer Up-/Download
- u.v.m.

Die Sichere **Speicherung** von Daten sowie diverse Kontrollmechanismen erhöhen die Datensicherheit innerhalb der S3 Cloud. Der Anwender ist somit gegen Anwenderfehler, physische und logische Ausfälle sowie gegen Datenverlust auf Grund von z.B. Infrastrukturausfällen geschützt.[97]

Standard-Speicher[98]

Die Speicherinfrastruktur von Amazon S3 ist äußerst zuverlässig und daher besonders gut geeignet für kritische Daten und Anwendungen. Daten werden redundant auf mehreren Systemen und in jedem System auf mehreren Geräten gespeichert. Mit Hilfe eines speziellen Prüfsummenverfahrens kann S3 feststellen, ob die gespeicherten oder empfangenen Datenpakete fehlerhaft bzw. beschädigt sind. Dieses Prüfverfahren läuft permanent und führt so zu einer systematischen Überprüfung des Datenverkehrs und im Fehlerfall zu einer Behebung der Probleme. Auch eine Versionierung von Objekten in den Buckets ist in S3 vorgesehen. Diese ermöglicht die Wiederherstellung von z.B. unbeabsichtigt gelöschten oder veränderten Objekten. Jedoch führt diese Versionierung auch zu höheren Speicherkosten. Die Vorteile des Amazon Standard-Speicher sind eine hohe Zuverlässigkeit und Verfügbarkeit sowie das Verhindern gleichzeitiger Datenverluste in zwei Systemen.

Reduced Redundancy Storage (RRS) [99]

Im Gegensatz zum Standard-Speicher ist Reduced Redundancy Storage (RRS) eine neue Methode von Amazon S3 zum Speichern von nicht-kritischen, reproduzierbaren Daten mit geringerer Redundanz als SRS. Dies hilft den Kunden, die für die Speicherung anfallenden Kosten zu senken. Die Vorteile von RRS sind eine hohe Zuverlässigkeit und Verfügbarkeit sowie das Verhindern von Datenverlusten in einer einzigen Anlage.

[96] Vgl. Amazon Web Services, Inc. (2012a)
[97] Vgl. ebenda
[98] Vgl. ebenda
[99] Vgl. ebenda

Amazon Glacier[100]

Amazon Glacier hingegen ist ein sehr kosteneffektiver Speicher zur Archivierung von Daten. Ideal eignet er sich für Daten, auf die man selten zugreifen muss und auf die ein Zugriff auch längere Zeit dauern darf. Ein Beispiel hierfür sind Daten, die laut Gesetz noch aufbewahrt werden müssen, jedoch nicht mehr verarbeitet werden. Die Vorteile von Amazon Glacier sind eine hohe Zuverlässigkeit und Verfügbarkeit sowie das Verhindern von Datenverlusten in zwei Systemen.

Datenverwaltung[101]

Ein weiteres Feature von Amazon S3 sind die mannigfaltigen Möglichkeiten zur Verwaltung der gespeicherten Daten. Neben der Verwaltung des Lebenszyklus der Daten (automatisches Löschen / Archivieren) können auch die Kosten, die mit der Nutzung von S3 anfallen, kontrolliert werden, indem sie z.B. auf die Kostenstellen verteilt werden, in denen sie tatsächlich anfallen. Berichte zu diesen Verwaltungs- und Kontrollmechanismen runden das Angebot von Amazon S3 ab.

Preise[102]

Bei S3 wird nur das gezahlt, was auch wirklich genutzt wird. Je nach Speichermethode fallen unterschiedliche Gebühren pro Terabyte und Zeiteinheit (z.B. Monat) für die Speicherung der Daten an. Die Anfrage sowie Übertragung der Daten werden differenziert (nach Anzahl Abfragen bzw. nach Terabyte pro Monat) abgerechnet.

Verwendung[103]

Amazon S3 bietet eine userfreundliche Anwendung in wenigen Schritten:

- Erstellen Bucket
- Wahl der Region
- Objekte ins Bucket hochladen
- Einrichten von Zugriffskontrollen

Das Ganze kann über die Amazon Web Services (AWS) Management Console abgewickelt werden.

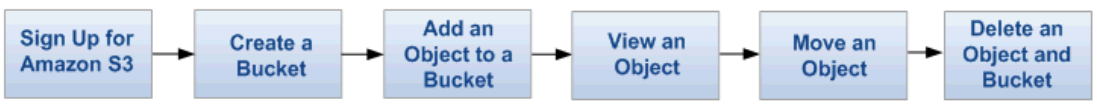


Abb. 7: Anleitung Amazon S3 unter Verwendung der AWS Management Console[104]

[100] Vgl. Amazon Web Services, Inc. (2012a)
[101] Vgl. ebenda
[102] Vgl. ebenda
[103] Vgl. Amazon Web Services, Inc. (2012b)
[104] Enthalten in: Amazon Web Services, Inc. (2012b)

7 Fazit

Alles in allem wurden in dieser Arbeit die notwendigen Grundlagen erläutert, um Hadoop und seine Funktionsweise zu verstehen. Es wurde kurz auf Big Data und NoSQL eingegangen. Anschließend wurde das Dateisystem von Hadoop, das HDFS, und der Basisalgorithmus MapReduce erläutert. In den letzten beiden Unterpunkten des Grundlagen-Kapitels wurde sowohl auf die Funktionsweise von Hadoop und dessen Fehlertoleranz, als auch auf diverse Anwendungsbereiche eingegangen. Zudem wurde ein Prototyp als Einzelplatzlösung aufgesetzt, um die grundlegenden Funktionen von Hadoop zu testen. Eine ausführliche Dokumentation über die Implementierung ist im Anhang 1 nachzulesen. Es ist denkbar in weiteren studentischen Arbeiten die Grundlagen von Hadoop genauer zu beleuchten. Insbesondere die Programmierung der MapReduce-Algorithmen konnte auf Grund des engen Zeitrahmens in dieser Ausarbeitung nicht näher beleuchtet werden.

Ergänzende Systeme, welche auf Hadoop aufsetzen und zusätzliche Funktionalitäten bieten, wurden in dieser Ausarbeitung nur oberflächlich betrachtet und grob beschrieben. Sie bieten dem Nutzer jedoch die Möglichkeit, Hadoop anwenderfreundlich und facettenreich einzusetzen. Dieses Erweiterungspotenzial macht es lohnenswert, diese ergänzenden Systeme in einer weiteren Studie genauer zu betrachten und Vor- und Nachteile, sowie die Anwendungsmöglichkeiten dieser Komponenten hervorzuheben.

Allgemein sind die ergänzenden Systeme einfacher zu bedienen als ein reines Hadoop, was somit ein erheblicher Vorteil für Firmen ist, da sich Mitarbeiter nicht speziell mit der komplizierten Programmierung von MapReduce-Algorithmen auseinandersetzen müssen, sondern nur mit der Handhabung eines ergänzenden Systems.

Big Data und dessen Verarbeitung ist ein vergleichsweise neues Themengebiet der IT. In Zukunft wird die Menge von Big Data, und damit dessen Bedeutung in der breiten Wirtschaft, weiter wachsen. Heute nutzen nur die wenigsten Unternehmen das Potenzial dieser Informationsquelle. Hadoop bietet, vor allem in Kombination mit den zusätzlichen Komponenten, hervorragende Möglichkeiten das Potenzial von Big Data auszuschöpfen. Es wird somit in absehbarer Zukunft verstärkt Einzug in die Wirtschaft finden.

Anhang

Anhangverzeichnis

Anhang 1: Installationsbeispiel einer Single-Node-Instanz

Die folgende Anleitung beschreibt ergänzend zum Kapitel 6.1 die schrittweise Installation und Konfiguration einer Hadoop Single-Node-Instanz.

Vorbereitungen

Bevor das eigentliche Hadoop Framework installiert und konfiguriert werden kann, müssen einige Grundlagen geschaffen werden. Als essentielle Grundlage dient das Linux Betriebssystem Ubuntu. Die Installation und Konfiguration von Ubuntu Desktop 12.10 in der 32Bit-Version wird hier als Voraussetzung gesehen und auch nicht näher behandelt. Im nächsten Schritt ist eine geeignete Java Virtual Machine (JVM) zu installieren. In diesem Fallbeispiel kommt Oracles Java Development Kit (JDK) in der Version 7 zum Einsatz. Ubuntu selbst besitzt aus lizenzrechtlichen Gründen kein Installationspaket von Oracles JDK 7. Daher wird sich hier einem vorgefertigten Personal Package Archive (PPA) bedient. Das Repository des PPA muss Ubuntu über die Paketliste bekannt gemacht werden:

```
$ sudo apt-get-repository ppa:webupd8team/java
```

Daraufhin sollte die Paketliste neu eingelesen werden, um sie auf dem aktuellsten Stand zu halten:

```
sudo apt-get update
```

Danach kann das Paket installiert werden:

```
sudo apt-get install oracle-java7-installer
```

Nach erfolgreicher Installation des Java Paketes befinden sich die zugehörigen Dateien im Verzeichnis /usr/lib/jvm/java-7-oracle. Bei einer reinen Single-Node-Instanz könnte nun mit der Installation des Hadoop Frameworks begonnen werden. Da hier jedoch bereits die Vorbereitungen zur Anbindung an ein Cluster getroffen werden sollen, muss zuerst noch die Möglichkeit zur Kommunikation der einzelnen Knoten geschaffen werden. Hierzu ist ein Benutzer *hadoop* und eine zugehörige Gruppe *hadoop* anzulegen:

```
$ sudo addgroup hadoop
$ sudo adduser –ingroup hadoop hadoop
```

Für die sichere Netzwerkkommunikation der Hadoop Knoten ist ein Secure Shell (SSH) Zugang einzurichten. Die Installation des SSH Daemon sieht folgendermaßen aus:

```
$ sudo apt-get install ssh
```

Nach der Installation ist unter dem Benutzer *hadoop* ein passwortloser SSH Schlüssel anzulegen:

```
$ ssh-keygen –t rsa –P “”
```

Da sich bei diesem Beispiel DataNode und NameNode auf einer Maschine befinden ist der öffentliche Schlüssel in den Schlüsselbund der autorisierten Schlüssel auf dieser Maschine einzufügen:

```
$ cat /home/hadoop/.ssh//id_rsa.pub >>
/home/hadoop/.ssh/authorized_keys
```

Mit diesem Schritt wären die Vorbereitungen abgeschlossen und das Hadoop Framework kann installiert werden.

Hadoop Framework

Zunächst gilt es das Hadoop Framework von der Apache Foundation herunterzuladen. Dabei ist darauf zu achten, dass die aktuell als stable markierte Version verwendet wird. In diesem Beispiel handelt es sich um die Version 1.0.4, die wie folgt mit dem Kommando wget auf die Festplatte gespeichert wird:

```
$ wget
http://mirror.softaculous.com/apache/hadoop/core/stable/hadoop-
1.0.4.tar.gz
```

Entpackt und installiert wird das heruntergeladene Archiv mit:

```
$ tar xvzf hadoop-1.0.4.tar.gz
$ sudo mv hadoop-1.0.4 /usr/local/hadoop
```

Die Rechte zur Bearbeitung und Ausführung der Dateien werden für den Benutzer *hadoop* folgendermaßen gesetzt:

```
$ sudo chown -R hadoop:hadoop /usr/local/hadoop
$ sudo chmod a+w /usr/local/hadoop
```

Nach der Installation ist das System zu konfigurieren. Alle Konfigurationsdateien des Hadoop-Systems befinden sich im Verzeichnis /usr/local/hadoop/conf. Die Datei hadoop-env.sh enthält die Konfiguration für die Laufzeitumgebung. Hier werden alle Java spezifischen Einstellungen vorgenommen. Besonders wichtig ist es hier die Umgebungsvariable JAVA_HOME zu setzen und *hadoop* somit mitzuteilen, welche JVM es verwenden soll. Hierfür muss der Eintrag von:

```
# export JAVA_HOME /usr/lib/j2sdk1.5-sun
```

zu:

```
export JAVA_HOME /usr/lib/jvm/java-7-oracle/jre
```

geändert werden. Weiter geht es mit der Konfiguration des Verzeichnisses und der Adresse des HDFS. Hierzu muss zuerst als Benutzer root ein lokales Verzeichnis angelegt und mit entsprechenden Rechten versehen werden:

```
$ sudo mkdir /usr/local/hadoop/hdfs
$ sudo chown hadoop:hadoop /usr/local/hadoop/hdfs
```

Danach sind in der Datei core-site.xml folgende Einträge zu machen:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!—Put site-specific property overrides in this file. -->

<configuration>
<property>
      <name>hadoop.tmp.dir</name>
      <value>/usr/local/hdfs/hadoop-hadoop</value>
      <description>Temporaeres Verzeichnis fuer das
HDFS</description>
</property>

<property>
      <name>fs.default.name</name>
      <value>hdfs://localhost:54310</value>
      <description>URI des HDFS NameNode</description>
</property>
</configuration>
```

Als nächstes muss in der Datei hdfs-site.xml der Replikationsfaktor wie folgt gesetzt werden:

```
<?xml version=“1.0“?>
<?xml-stylesheet type=“text/xsl“ href=“configuration.xsl“?>

<!—Put site-specific property overrides in this file. -->

<configuration>
<property>
      <name>dfs.replication</name>
      <value>1</value>
      <description>Replikationsfaktor</description>
</property>
</configuration>
```

Zum Schluss müssen noch die Adresse und die Portnummer der JobTracker Instanz in die Datei mapred-site.xml eingetragen werden:

```
<?xml version=“1.0“?>
<?xml-stylesheet type=“text/xsl“ href=“configuration.xsl“?>
```

<!—Put site-specific property overrides in this file. -->

```
<configuration>
<property>
        <name>mapred.jop.tracker</name>
        <value>localhost:54311</value>
        <description>Hostname des JobTrackers</description>
</property>
</configuration>
```

Bevor mit hadoop gearbeitet werden kann, ist das HDFS zu formatieren und wird somit unter /usr/local/hdfs/hadoop-hadoop/ angelegt:

```
$ cd /usr/local/hadoop
$ bin/hadoop namenode -format
```

Damit sind die Installation und Konfiguration des Single-Node abgeschlossen und die Daemon Prozesse können gestartet werden:

```
$ bin/start-all.sh
```

Das Kommando `jps` zeigt alle laufenden Daemon Prozesse an. In unserem Beispiel sind das:

- JobTracker
- TaskTracker
- DataNode
- NameNode
- SecondaryNameNode

Zur Überprüfung der korrekten Arbeitsweise des Single-Node oder zur Fehlerbeseitigung, bietet sich der Blick in die Dateien unter /usr/local/hadoop/logs an.

Quellenverzeichnisse

Literaturverzeichnis

Bayer, M. (2012) Hadoop – der kleine Elefant für die große Daten, in: Computerwoche, 2012, 11/12, S. 22-24

Spies, R. (2012) CIO-Handbuch 2012/2013, Best Practices für die neuen Herausforderungen des IT-Managements, Hrsg.: Lang, M., Düsseldorf: Symposion Publishing GmbH

Wartala, R. (2012) Hadoop: Zuverlässige, verteilte und skalierbare Big-Data-Anwendungen, München: Open Source Press

White, T. (2012) Hadoop, The Definitive Guide, 3. Auflage, Sebastopol: O'Reilly

Aufsätze in Zeitschriften

Pürmer, H. (2011) NoSQL, die neue (alte) Datenbank-Generation?, in: Computerwoche, 2011, 38/11, S. 28-31

Winter, R., Landert, K. (2006) IT/Business Alignment als Managementherausforderung, WI-Editional zum Schwerpunktthema, in: WIRTSCHAFTSINFORMATIK, 2006, 48 (2006) 5, S. 309

Wolff, E. (2012 NoSQL, die neue Datenbankgeneration?, in: Computerwoche, 2012, 37/12, S. 16-17

Verzeichnis der Internet- und Intranet-Quellen

Amazon Web Services, Inc. (2012a) Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/de/s3/, Abruf: 03.01.2013

Amazon Web Services, Inc. (2012b) Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/de/s3/, Abruf: 03.01.2013

Borthakur, D. (2009) HDFS Architecture Guide, http://hadoop.apache.org/docs/r0.20.0/hdfs_design.pdf, Abruf: 07.01.2013

Chaudhri, A. (2011) Big Data University – What is Hadoop?, http://bigdatauniversity.com/courses/mod/url/view.php?id=7463, Abruf: 09.01.2013

Cubrid (2012)	Hive Architecture, http://www.cubrid.org/files/attach/images/220547/047/369/hive_architecture.png, Abruf: 19.12.2012
Figuière, M. (2009)	Xebia France, Devoxx – Jour 1 – NoSQL avec HBase, http://blog.xebia.fr/wp-content/uploads/2009/11/hbase-shema.png, Abruf: 01.01.2013
IBM Corporation (2012)	IBM Software, Information Management, InfoSphere Platform, Big Data Analytics - What is HBase?, http://www-01.ibm.com/software/data/infosphere/hadoop/hbase/, Abruf: 01.01.2013
Lackers, R. u.a. (2012)	Content Management System (CMS), http://wirtschaftslexikon.gabler.de/Archiv/75915/content-management-system-cms-v8.html, Abruf: 04.01.2013
McDonald, K. (2011a)	Big Data University – Hadoop architecture, http://bigdatauniversity.com/courses/mod/url/view.php?id=7470, Abruf: 09.01.2013
McDonald, K. (2011b)	Big Data University – Hadoop MapReduce, http://bigdatauniversity.com/courses/mod/url/view.php?id=7519, Abruf: 09.01.2013
The Apache Software Foundation (2011)	Hadoop MapReduce 0.22 Documentation - Hadoop Streaming - How Streaming works, http://hadoop.apache.org/docs/mapreduce/r0.22.0/streaming.html, Abruf: 17.12.2012
The Apache Software Foundation (2012a)	Hadoop Hive – What is Hive?, http://hive.apache.org/docs/r0.8.1/, Abruf: 16.12.2012
The Apache Software Foundation (2012b)	Apache Hive, https://cwiki.apache.org/confluence/display/Hive/Home, Abruf: 19.12.2012
The Apache Software Foundation (2012c)	The Apache HBase ™ Reference Guide: Architecture, Overview, http://hbase.apache.org/book/architecture.html#arch.overview, Abruf: 17.12.2012
The Apache Software Foundation (2012d)	Apache Pig – Welcome to Apache Pig, http://pig.apache.org/ Abruf: 17.12.2012

The Apache Software Foundation (2012e)	Apache Pig – Welcome to Apache Pig, http://pig.apache.org/ Abruf: 17.12.2012
The Apache Software Foundation (2012f)	Apache Sqoop - Sqoop Documentation - Sqoop User Guide (v1.4.2), http://sqoop.apache.org/docs/1.4.2/SqoopUserGuide.html, Abruf: 01.01.2013
The Apache Software Foundation (2012g)	Apache ZooKeeper - ZooKeeper Documentation – ZooKeeper: A Distributed Coordination Service for Distributed Applications http://zookeeper.apache.org/doc/trunk/zookeeperOver.html, Abruf: 01.01.2013
The Apache Software Foundation (2012h)	Hadoop Wiki – PoweredBy, http://wiki.apache.org/hadoop/PoweredBy, Abruf: 09.01.2013
The Apache Software Foundation (2012i)	Hadoop - MapReduce Tutorial, http://hadoop.apache.org/docs/mapreduce/r0.22.0/mapred_tutorial.html, Abruf 09.01.2013
Varenkamp, R., Siepermann, C. (2012)	Enterprise Resource Planning-System, http://wirtschaftslexikon.gabler.de/Archiv/17984/enterprise-resource-planning-system-v7.html, Abruf: 04.01.2013

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

NoSQL Datenbanken

Typisierung

Schriftliche Ausarbeitung
im Rahmen der Lehrveranstaltung „Projekt des 5. Semesters“
Kompetenzzentrum Open Source (KOS)

Vorgelegt von

Teresa Bogolowski, Tana Brunner,
Sophie Lingelbach, Christin Wattler

am 25.01.2013

Fakultät Wirtschaft
Studiengang Wirtschaftsinformatik
Kurs WWI2010V

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACID	**A**tomicity, **C**onsistency, **I**solation and **D**urability
API	**A**pplication **P**rogramming **I**nterface
BASE	**B**asically **A**vailable, **S**oft State, **E**ventually consistent
BSON	**B**inary J**SON**
CAP	**C**onsistency, **A**vailability, **P**artition Tolerance
CF	**C**olumn **F**amily
CLI	**C**ommand **L**ine **I**nterface
CQL	**C**assandra **Q**uery **L**anguage
DB	**D**aten**b**ank/ Database
DBMS	**D**aten**b**ank**m**anagement**s**ystem
ID	**Id**entifier
JSON	**J**ava **S**cript **O**bject **N**otation
JVM	**J**ava **V**irtual **M**achine
RDBMS	**R**elationales **D**aten**b**ank**m**anagement**s**ystem
SQL	**S**tructured **Q**uery **L**anguage
UUID	**U**niversally **U**nique **I**dentifier

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Datenbanken erleben einen ständigen Wandel. Lange Zeit herrschte der Glaube, dass alle Datenbankfragen mit dem relationalen Datenbankmodell gelöst werden können. Zur selben Zeit als relationale Datenbanken ihren Höhepunkt erlebten, begann jedoch auch die Entwicklung der NoSQL Datenbanken. So entstanden beispielsweise mit Lotus Notes und Berkley DB bereits in den 80er Jahren erste Vorreiter der NoSQL-Bewegung, die noch bis heute bekannt sind.[1] Einen richtigen Schub erleben NoSQL-Datenbanken erst seit der Entwicklung des Web 2.0 mit der Entstehung von Big Data. Big Data ist der Überbegriff für rasant wachsendes Datenvolumen. Einer Studie zufolge ist die weltweite Datenmenge in den letzten fünf Jahren um ein fünffaches gewachsen und so soll 2011 auch das weltweit produzierte Datenvolumen auf 1,8 Zettabyte (1,8 Billionen Gigabyte) ansteigen. Weitere Entwicklungen in diesem Zusammenhang sind steigende Nutzerzahlen von Datenanalysen, steigende Anforderungen an die Performance von Datenauswertungen und eine wachsende Zahl an Datenquellen. Zudem müssen auch semi- beziehungsweise quasistrukturierte Daten verwaltet werden. Hierfür sind relationale Datenbanken nicht geeignet. Eine Lösung zur Verwaltung dieser Daten bieten jedoch NoSQL-Datenbanken. Eine optimale Datenverwaltung könnte aus einer Kombination von relationalen und NoSQL-Datenbanken bestehen.[2] Inhalt der folgenden Arbeit sind die verschiedenen Eigenschaften, Möglichkeiten und Formen von NoSQL-Datenbanken.

Im zweiten Kapitel werden die allgemeinen Grundlagen und die verschiedenen Arten von NoSQL-Datenbanken mit ihren möglichen Einsatzbereichen sowie den prägnantesten Vor- und Nachteilen erläutert.

Im nächsten Kapitel wird zu jeder Art ein Beispiel aufgeführt und im Hinblick auf die Funktionsweise sowie besondere Eigenschaften näher betrachtet. Diese Datenbanken dienen als Grundlage für die Prototypen, die im vierten Kapitel folgen.

Abschließend soll im letzten Kapitel ein Fazit gezogen werden, indem die Erkenntnisse dieser Arbeit noch einmal übersichtlich zusammengefasst werden.

[1] Vgl. Edlich, S. u. a. (2010), S. 1
[2] Vgl. Computerwoche (2012)

2 Fachliche Grundlagen

2.1 NoSQL

Während bislang traditionelle relationale Datenbanken die Lösung aller Probleme versprochen haben, rücken NoSQL-Datenbanken immer mehr in den Fokus. Dabei bedeutet NoSQL nicht die grundsätzliche Ablehnung von SQL, sondern vielmehr nicht die ausschließliche Anwendung von SQL, wortwörtlich wiedergegeben bedeutet das Kürzel „Not Only SQL“.[3]

Diese neue Generation von Datenbanksystemen erfüllt häufig einige der folgenden Aspekte:

- nichtrelationales Datenbankmodell
- Ausrichtung auf eine verteilte und horizontale Skalierbarkeit
- System ist Open Source
- Schemafreiheit oder schwächere Schemarestriktionen
- Unterstützung einer einfachen Datenreplikation durch verteilte Architektur
- Bereitstellung einer einfachen Programmierschnittstelle
- Verwendung des Konsistenzmodells BASE[4]

Die Entwicklung bestimmter Alternativen zu relationalen Datenbanken findet ihre Begründung in den ständig steigenden Anforderungen an die Datenbanken. Ein ausschlaggebender Faktor in diesem Kontext ist das permanent wachsende Datenvolumen, welches in erster Linie durch eine nicht unerhebliche Anzahl an neuen Datenquellen verursacht wird. Hervorgerufen durch die Interaktivität des Internets oder die zunehmende Digitalisierung von Geschäftsprozessen benötigen relationale Datenbanken leistungsfähigere Server, um die erhöhte Datenverwaltung oder Performanceanforderungen abdecken zu können. Aber auch diese Möglichkeit ist erschöpflich, da die Serverleistung limitiert ist und mit der Leistung der Preis überproportional wächst. Diese Faktoren summiert mit der Zunahme an weniger strukturierten Daten führte zu der Entwicklung alternativer Datenbanken, mit denen diese Probleme behoben werden sollen.[5]

[3] Vgl. Walker-Morgan D. (2010)
[4] Vgl. Edlich, S. u. a. (2010), S. 2
[5] Vgl. Wolff, E. (2012), S. 16

Die im Laufe der Zeit entstandenen NoSQL-Datenbanken lassen sich aufgrund des Datenbankmodells folgendermaßen unterteilen:

- dokumentenorientierte Datenbanksysteme
- Graphendatenbanken
- Key/Value-Datenbanksysteme
- spaltenorientierte Datenbanksysteme[6]

ACID-Prinzip

Wie bereits erwähnt, arbeiten diese Systeme gegensätzlich zu den relationalen Datenbanksystemen nicht nach dem ACID Prinzip, sondern nach dem BASE Prinzip. Das ACID Prinzip umfasst die folgenden Eigenschaften einer Datenbank:

- **A**tomicity *(Atomarität)*
- **C**onsistency *(Konsistenz)*
- **I**solation *(Isoliertheit)*
- **D**urability *(Dauerhaftigkeit)*

Atomicity bedeutet, dass Transaktionen entweder ganz oder gar nicht ausgeführt werden. Consistency besagt, dass eine Transaktion in einem konsistenten System ebenfalls wieder in einen konsistenten Datenzustand mündet. Isolation drückt aus, dass Transaktionen voneinander unabhängig ablaufen und sich so nicht beeinflussen können. Durabilitiy spiegelt wider, dass nach einem erfolgreichen Transaktionsabschluss die Daten garantiert langfristig gespeichert sind.[7]

BASE-Prinzip

Das BASE Konzept umfasst im Gegensatz die anschließenden Merkmale:

- **B**asically **A**vailable *(grundsätzlich verfügbar)*
- **S**oft State *(weicher Zustand)*
- **E**ventually consistent *(eventuelle Konsistenz)*

Basically Available bedeutet, dass eine Verfügbarkeit des Systems grundsätzlich gegeben ist es aber toleriert wird, dass einzelne Teile ausfallen können. Soft State und Eventually consistent besagen, dass ein System am Ende über konsistente Daten verfügt es allerding im Laufe einer Transaktion zu inkonsistenten Zuständen kommen kann, welche gegebenenfalls an eine Anwendung ausgeliefert werden.[8]

[6] Vgl. Pürner, H. A. (2011), S. 29 f.
[7] Vgl. Wolff, E. (2012), S. 17
[8] Vgl. Wolff, E. (2012), S. 17

CAP-Theorem

Das CAP-Theorem beschreibt ein magisches Dreieck der Datenbanktheorie. Im Anhang 1 ist dieses zusammen mit den gängigsten NoSQL-Systemen dargestellt. Dieses Dreieck besteht aus drei verschiedenen Hauptfaktoren nach denen eine Datenbank aufgebaut ist:

- Consistency *(Datenkonsistenz)* beschreibt die Eigenschaft der bestehenden Daten sowie ggf. deren Replikationen auf dem gleichen Stand zu sein. Das heißt alle Clients sehen beim Aufruf die gleichen Daten.
- Availability *(Verfügbarkeit)* betrifft die Antwortzeiten des Datenbanksystems dem Client gegenüber.
- Partitioning Tolerance *(Partitionstoleranz)* Verteilung der Daten über viele Datenknoten und das Verhalten bei Ausfall eines Knotens bezüglich der Sicherung der betroffenen Daten (Replikationen o.Ä.).

Die folgende Abbildung zeigt das Zusammenspiel dieser Faktoren noch einmal übersichtlich.

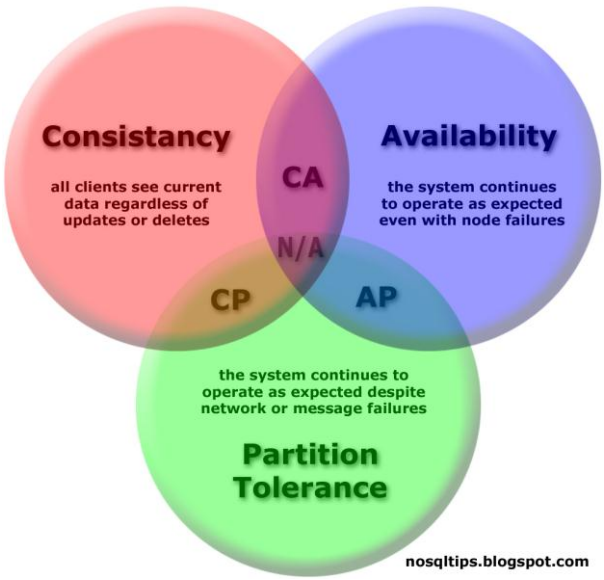


Abb. 1: Cap Theorem[9]

Ein Datenbanksystem kann zeitgleich nur zwei dieser Anforderungen befriedigen. Die Eignung für ein spezielles Einsatzgebiet einer Datenbank begründet sich folglich in ihrer Ausrichtung. Dabei werden beispielsweise traditionell CA-Systeme durch RDBMS realisiert und sind vor allem bei gültigen Operationen nach dem Transaktionsprinzip gefragt. Bei NoSQL-Datenbanken andererseits treten zumeist die Aspekte „Verfügbarkeit“ und „Partitionstoleranz“ an vorderste Stelle um Massen von Daten zu bewältigen, bei denen eine absolute Konsistenz nicht erforderlich ist.

[9] Entnommen aus: Kwangsub (2012)

2.2 Dokumentenorientierte Datenbanksysteme

Dokumentenorientierte Datenbank werden oft auch als dokumentenbasierte Datenbanken bezeichnet und speichern im Gegensatz zu relationalen Datenbanken die Daten nicht in Form von Tabellen, sondern als Dokumente ab.[10] „Ein Dokument ist in diesem Zusammenhang zu verstehen als eine strukturierte Zusammenstellung bestimmter Daten.“[11] Innerhalb eines Dokuments ist die Definition von beliebigen Feldern möglich. Diese Felder werden als Schlüssel bezeichnet. Zusätzlich ist jedem Schlüssel jeweils ein Wert zugeordnet. Auch dieser Wert kann beliebig gewählt werden, es kann sich beispielsweise um eine Zahl, ein Wort, ein Text oder ein Dokument handeln. Außerdem ist es nicht notwendig die Länge des Wertes vorab zu definieren, denn auch diese kann nach Belieben gestaltet werden.[12] Die Abbildung 2 veranschaulicht den möglichen Aufbau eines Dokuments im Sinne der dokumentenbasierten Datenbank.

Abb. 2: Beispiel für ein Dokument[13]

„In dokumentenbasierten Datenbanken existieren keine Relationen zwischen einzelnen Dokumenten, jedes Dokument ist eine in sich geschlossene Einheit mit individuellem Schema und individuellen Inhalten.“[14] Bei dieser Art der Datenspeicherung wird auf eine Vorgabe für die spezifische Strukturierung der zu sichernden Daten verzichtet. Für die Datenbank ist kein umspannendes Datenbankschema vorhanden, es besteht die Möglichkeit in jedem separaten Dokument eine andere Struktur zu verwenden.[15]

[10] Vgl. Datenbanken Online Lexikon (2012)
[11] Datenbanken Online Lexikon (2012)
[12] Vgl. Schnelle, J. (2010)
[13] Enthalten in: Datenbanken Online Lexikon (2012)
[14] Datenbanken Online Lexikon (2012)
[15] Vgl. Datenbanken Online Lexikon (2012)

„Durch die kaum vorhandenen Einschränkungen beim Aufbau der Dokumente ergibt sich eine umfassende Gestaltungsfreiheit [...]. Jedoch folgt aus dieser Freiheit auch ein gewisser Aufwand, da die Struktur der Dokumente selbst festgelegt und kontrolliert werden muss [...].“[16] Die Tabelle 1 fasst die prägnantesten Vorteile und Nachteile der dokumentenorientierten Datenbanken übersichtlich zusammen.

Vorteile	Nachteile
• umfassende Gestaltungsfreiheit • Abbildung beliebig komplizierter Datenstrukturen • gute Skalierbarkeit	• Aufwand durch selbständige Festlegung und Kontrolle der Dokumentenstruktur

Tab. 1: Vor- und Nachteile einer dokumentenorientierten Datenbank[17][18]

Die Intention bei der Entwicklung von dokumentenorientierten Datenbanken war die Speicherung zusammengehöriger Daten an einer Stelle. So bilden diese Datenbanken einen Gegenpol zu den relationalen Datenbanken, bei welchen die Daten getrennt voneinander in verschieden Tabellen abgelegt werden und eine zusammenhängende Information erst über die Abfrage mehrerer Tabellen generiert werden kann.[19] Durch die Möglichkeit größere Mengen Text mit unbestimmter Länge zu speichern eignen sich dokumentenorientierte Datenbanken sehr gut für Content Management Systeme, Blogs oder Wikis.[20]

Die derzeit bekanntesten dokumentenorientierten Open-Source-Datenbanken sind CouchDB und MongoDB.[21] Im weiteren Verlauf dieser Arbeit wird speziell auf MongoDB hinsichtlich der speziellen Eigenschaften, des Datenmodells sowie der möglichen Einsatzgebiete noch näher eingegangen.

[16] Datenbanken Online Lexikon (2012)
[17] Vgl. Datenbanken Online Lexikon (2012)
[18] Vgl. Wolff, E. (2012), S. 16
[19] Vgl. Datenbanken Online Lexikon (2012)
[20] Vgl. Schnelle, J. (2010)
[21] Vgl. Schnelle, J. (2010)

2.3 Graphorientierte Datenbanksysteme

Graphorientierte Datenbanksysteme legen ihre Daten in Form von Graphen ab. Diese bestehen aus Knoten, die über Kanten miteinander verbunden sind. Die Datenstruktur ist somit von Beginn an durch die definierten Kanten verbunden („gejoint“).[22]

Im Gegensatz dazu sind relationale Datenbanken darauf spezialisiert, Daten in Tabellen abzuspeichern. Um Daten aus unterschiedlichen Tabellen zusammenzufügen, werden Joins verwendet. Ein Join kombiniert zwei Tabellen miteinander, wenn sich die Spalten einer Tabelle auf die Spalten einer anderen Tabelle beziehen.

Bei Graphendatenbanken fallen diese zeitintensiven Join-Operationen durch die bereits verknüpfte Datenstruktur weg. Darüber hinaus verlangen Graphendatenbanken auch keine strenge Struktur bei der Datenablage.[23] Sie können dadurch semistrukturierte Daten ablegen.[24]

Die bekanntesten Anwendungsbereiche von Graphendatenbanken sind soziale Netzwerke, Fahr- und Flugplanoptimierungen sowie Empfehlungssysteme.[25] Mit Hilfe von Graphen kann die zugrundeliegende Struktur abgebildet werden. Die Kanten eines Graphen repräsentieren die Verbindung oder Beziehung zwischen den Knoten. Sie können gerichtet oder ungerichtet sein. Eine gerichtete Kante ist eine einseitige Beziehung von einem Knoten zu einem anderen Knoten. Von einer ungerichteten Kante wird gesprochen, wenn es sich um eine beidseitige Beziehung zwischen zwei Knoten handelt. Analog dazu wird auch von gerichteten und ungerichteten Graphen gesprochen.[26] Neben der Richtung einer Kante kann auch eine Gewichtung vorgenommen werde. Dazu wird der Kante ein numerischer Wert zugeordnet.

Anhand eines sozialen Netzwerkes werden die oben genannten Begriffe verdeutlicht: die Benutzer stellen die Knoten des Graphen dar, die Beziehungen zwischen den Benutzern, z.B. „kennt“, werden durch Kanten abgebildet.

[22] Vgl. Sakr, S./Pardede, E. (2011), S. 22
[23] Vgl. Edlich, S. u. a. (2010), S. 182 f.
[24] Vgl. Edlich, S. u. a. (2010), S. 182 f.
[25] Vgl. Edlich, S. u. a. (2010), S. 170
[26] Vgl. Edlich, S. u. a. (2010), S. 172

In Abbildung 3 sind alle Kanten vom gleichen Typ („kennt“). Ein solcher Graph wird als „single-relational Graph“ bezeichnet. Sie werden häufig verwendet, stoßen jedoch schnell an ihre Grenzen, da sowohl die Kanten als auch die Knoten immer vom gleichen Typ sind.[27] Komplexe Datenstrukturen können nicht dargestellt werden.

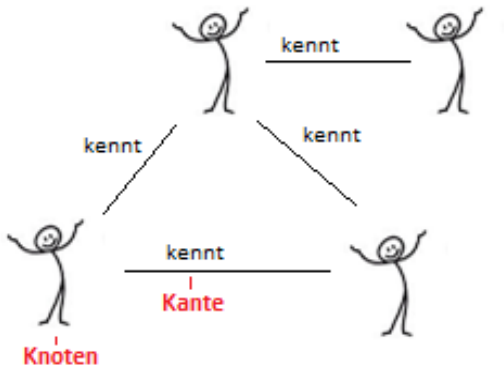


Abb. 3: single-relational Graph[28]

Mit den „multi-relational-Graphen“ lassen sich diese Grenzen überwinden. Ihre Kanten können unterschiedliche Bedeutungen haben und die Knoten von verschiedenen Typen sein. Ein Graph könnte aus den Beziehungen zwischen Benutzern bestehen und zusätzlich die Webseiten einbeziehen, die der jeweilige Benutzer häufig besucht. Die Benutzer und Webseiten sind hierbei die Knoten, die Bezeichnungen „kennt“ und „besucht“ sind die Kanten. Wird ein solcher Graph durch key-value Paare erweitert, ergibt sich ein Property-Graph, wie er in Abbildung 4 zu sehen ist. [29]

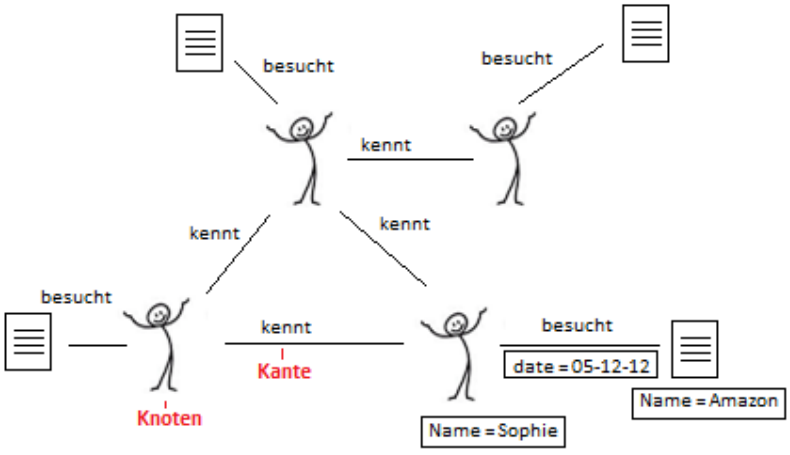


Abb. 4: Property-Graph[30]

[27] Vgl. Rodriguez, M. A. (2010)
[28] Mit Änderungen entnommen aus: Rodriguez, M. A. (2010)
[29] Vgl. Rodriguez, M. A. (2010)
[30] Mit Änderungen entnommen aus: Rodriguez, M. A. (2010)

Fast jede Datenbank kann einen Graphen implizit abbilden zum Beispiel in Form einer Matrix (Tabelle) oder in der Extensible Markup Language (XML). Eine Graphendatenbank unterscheidet sich von anderen Datenbanken dadurch, dass sie einen Graphen explizit darstellen kann. Außerdem benötigt sie keinen Index, um die benachbarten Knoten zu erreichen.[31] Wesentlich ist zudem, dass die Kosten, um von einem Knoten zu seinem Nachbarn zu gelangen, konstant bleiben, auch wenn der Graph wächst. Im Gegensatz dazu steigen die Kosten in einer Nicht-Graphendatenbank, wenn der Index wächst, da mehrere Abfragen nötig werden.[32] Die Tabelle 2 veranschaulicht die Vor- und Nachteile einer Graphendatenbank.

Vorteile	Nachteile
• Ersetzen von teuren JOINS durch Graph-Traversals • konstante Kosten um von einem Knoten zu seinem Nachbarn zu wandern • explizite Darstellung von Graphen • Speicherung semistrukturierter Daten • Performance ist unabhängig von der Größe des Graphen[33]	• keine einheitliche Abfragesprache • Vernachlässigung der Konsistenz

Tab. 2: Vor- und Nachteile einer Graphendatenbank

Das Traversieren ist eine der wichtigsten Methoden in einer Graphendatenbank. Mit ihr kann der Graph von einem Startknoten aus durchlaufen und dabei Fragen wie „Welche Webseiten besuchen meine Freunde?“ oder „Wer ist mit wem befreundet?“ beantwortet werden. Es gibt verschiedene Techniken, um einen Graphen zu durchsuchen, wie beispielsweise die Breiten- und Tiefensuche, die algorithmische Traversierung und die randomisierte Traversierungsmethode.

[31] Vgl. Rodriguez, M. A. (2010)
[32] Vgl. Rodriguez, M. A. (2010)
[33] Vgl. sones GmbH (o. J.)

2.4 Key/Value Datenbanksysteme

Key/Value-Datenbanken oder auch Key/Value-Stores genannt, gibt es schon seit den 70er Jahren. Einen richtigen Schub erleben sie jedoch erst seit der Entwicklung des Web 2.0 und dem Cloud-Computing.[34]

Das Prinzip der Key/Value-Stores ist sehr einfach gehalten. So bestehen einzelne Datensätze nur aus einem Key und einem zugehörigen Value. Die Schlüssel müssen immer eindeutig sein und können in unterschiedliche Namespaces oder Datenbanken aufgeteilt werden. Die Aufteilung läuft meist über den Hash-Wert des Schlüssels.[35] Der Key besteht meistens aus einer Zeichenkette, wobei es sich hierbei nicht zwingend um einfache Textstrings handeln muss, sondern auch ein bestimmtes Schema verwendet werden kann. Der Value eines Datensatzes kann je nach Datenbank nicht nur aus einem String bestehen, sondern beispielsweise auch aus einer Liste oder einem Set.[36] Der Zugriff auf einen Datensatz ist nur über den Schlüssel möglich. Es ist dadurch nicht möglich den Schlüssel zu einem bestimmten Wert zu suchen.[37] Zudem gibt es bei dieser Art von Datenbanken keine Möglichkeit einzelne Datensätze miteinander zu verknüpfen. Jegliche Form von Join muss direkt im Anwendungsprogramm vorgenommen werden. Zusammengefasst lässt sich sagen, dass eine Key/Value-Datenbank lediglich eine Zusammenstellung mehrerer zweispaltiger Tabellen ist, wobei jeweils in der ersten Spalte der Schlüssel und in der zweiten der Wert gespeichert wird.[38]

Die Gruppe der Key/Value-Datenbanken lässt sich in On-Disk-Datenbanken und In-Memory-Datenbanken unterteilen. On-Disk-Datenbanken, auch Hauptspeicherdatenbanken genannt, speichern die Daten direkt auf der Festplatte und können somit gut als Datenspeicher genutzt werden.

Bei der In-Memory Variante werden die Daten im Cache gehalten ohne im Hintergrund eine Datenbank auf der Festplatte zu haben. Sie können somit gut als verteilte Cache-Speichersysteme genutzt werden.[39]

[34] Vgl. Edlich, S. u. a. (2010), S. 131
[35] Vgl. Edlich, S. u. a. (2010), S. 7
[36] Vgl. freiesMagazin (2010)
[37] Vgl. Pürner, H. A. (2011)
[38] Vgl. freiesMagazin (2010)
[39] Vgl. Pürner, H. A. (2011)

Zu den allgemeinen Vertretern gehören beispielsweise Redis, Riak, Voldemort und Dynamo. Bei Redis handelt es sich um eine In-Memory-Datenbank. In dieser Arbeit wird speziell auf Voldemort eingegangen, um anhand dieses Systems die Funktionsweise und die Eigenschaften der Key/Value-Datenbanken zu verdeutlichen. Später wird hiervon auch ein Prototyp entwickelt.

Geeignete Einsatzmöglichkeiten der Key/Value-Stores bieten grundsätzlich alle Anwendungen, in denen die Daten gut in Schlüssel-Wert-Paare strukturiert werden können. Ein Beispiel hierfür sind Microblogging-Dienste wie beispielsweise Twitter. So können hier beispielsweise einzelne Blogeinträge so abgespeichert werden, dass der Schlüssel aus einer fortlaufenden Nummer, dem Nutzernamen und einem Zeitstempel zusammengesetzt wird und der tatsächliche Eintrag dann als zugehöriger Value abgespeichert wird.[40] Eine mögliche Tabelle mit verschiedenen Beispieleinträgen ist die folgende Tabelle 3.

Key	Value
233/mrX/2012-08-09 08:33:14	Hallo Welt!
454/mrsPQ/2012-08-15 09:10:34	Schönes Leben, schöne Welt.
355/MaxM/2012-09-01 18:45:53	Ich bin ein Test-Eintrag.

Tab. 3: Darstellung Key-Value Prinzip

Ein weiteres Beispiel für die Verwendung von Key/Value-Datenbanken ist die Sessionverwaltung eines Websystems. Über die Session-ID kann beispielsweise der aktuelle Zustand der Session hinterlegt werden, um diesen bei den einzelnen Zugriffen abzufragen und zu überschreiben.[41] Vorstellbar sind Key/Value-Datenbanken aber auch für die Artikelverwaltung in Online-Stores oder die Userverwaltung in Sozialen Netzwerken.

[40] Vgl. freies Magazin (2010)
[41] Vgl. Heise Zeitschriften Verlag (2011)

Ein Vorteil der Key/Value Datenbanken ist die gute Skalierbarkeit. Diese ergibt sich daraus, dass die Datensätze nicht miteinander verknüpft sind, denn bei join-intensiven Daten und Anwendungen ist die Skalierung deutlich komplizierter.[42] Ein weiterer Vorteil besteht in der hohen Performance. Teilweise sind Schreib-Operationen sogar schneller als Leseoperationen, was bei dokumentenorientierten oder relationalen Datenbanken beispielsweise nie vorkommt. Grund dafür ist, dass keine Verwaltungstabellen oder Indexe im Hintergrund aktualisiert werden müssen, sondern im besten Fall nur geprüft werden muss ob der gewünschte Schlüssel schon vorhanden ist und der Datensatz dann am Ende der Datenbank eingetragen werden kann.[43] Die einfache Struktur der Daten ist gut was Performance und Skalierung angeht, jedoch ist diese Art der Datenbanken für Anwendungen mit komplexen Datenstrukturen nicht sinnvoll, da diese kaum geeignet abgelegt werden können.[44] Ein Nachteil der Systeme ist auch, dass nur über den Schlüssel auf einen Datensatz zugegriffen werden kann und es somit nur begrenzte Abfragemöglichkeiten gibt. Meist muss sich der Nutzer hier auf die Mächtigkeit der API verlassen und hat nicht die Möglichkeit eigene komplexe Abfragen zu schreiben.[45] Die verschiedenen Vorteile und Nachteile sind in nachfolgender Tabelle 4 noch einmal zusammengefasst.

Vorteile	Nachteile
• gute Skalierbarkeit • hohe Performance	• begrenzte Abfragemöglichkeiten • Abbildung komplexer Datenstrukturen nur schwer möglich

Tab. 4: Vor- und Nachteile einer Key/Value-Datenbank

[42] Vgl. Edlich, S. u. a. (2010), S. 131
[43] Vgl. freiesMagazin (2010)
[44] Vgl. Computerwoche (2011)
[45] Vgl. Edlich, S. u. a. (2010), S. 7

2.5 Spaltenorientierte Datenbanksysteme

Das Prinzip, welches hinter spaltenorientierten Datenbanksystemen beziehungsweise „Wide Column Stores“ steht wurde bereits in den 80er Jahren entwickelt. Im Gegensatz zu relationalen Datenbanksystemen werden die Daten nicht in Zeilen, sondern in Spalten angeordnet. Auf diese Art und Weise wird für jedes Attribut eine eigene Tabelle (bzw. Spalte) angelegt und hintereinander gespeichert. Hierbei besteht eine Spalte zumeist aus drei Elementen, dem Namen, den eigentlichen Daten und einem Timestamp zur Versionierung. Die physikalische Speicherung erfolgt ebenfalls anhand der Spalten. Bei der Speicherung von Daten zu Tieren mit den Attributen ID, Art und Name sähen dies wie folgt aus:
Zeilenorientiert: 001, Hund, Paul, 002, Katze, Mietz, 003, Pferd, Amadeus
Spaltenorientiert: 001, 002, 003, Hund, Katze, Pferd, Paul, Mietz, Amadeus[46]

Die Spalten an sich werden in „Keyspaces“ abgelegt. Sie dienen als oberste „Ordnungsebene“ und als Namensraum. In Analogie zu relationalen Systemen können innerhalb des Keyspaces einzelne Spalten mit „verwandten“ Inhalten zusammengefasst werden. Dies hat im weitesten Sinne eine gewisse Ähnlichkeit mit einer denormalisierten Tabelle im relationalen System. Diese sogenannten „Column Families“ (siehe Abb. 5) besitzen im Gegensatz zu Tabellen jedoch keine festgelegte logische Struktur und können bis zu mehrere Millionen Spalten fassen.[47] Wichtig bei der Modellierung von Column-Families ist das Vorgehen. Anders als bei relationalen Modellen wird nicht nach festen Regeln eine Struktur abgeleitet und diese folglich abgefragt. Die Struktur der Column-Families ergibt sich letztendlich aus den Ergebnissen und Werten, die man abzufragen erwünscht. [48] Das heißt nicht das Schema der Datenbank, sondern letztendlich ihre Funktionalität im Sinne der Erkenntnisse aus der Datenauswertung treten hier in den Vordergrund. So ist es auch nicht verwunderlich, dass Spalten je nach Abfrage mehreren Column-Families zugeordnet werden sollen und daher gegeben falls auch redundant vorkommen. Wenn auch in der Regel eine Versionierung der Daten mittels Timestamp stattfindet, so werden strenge Prinzipien von Redundanzfreiheit und Datenintegrität in Teilen untergraben.[49]

[46] Vgl. Edlich, S. u. a. (2010), S. 53
[47] Vgl. Datenbanken Online Lexikon (2011a)
[48] Vgl. Datenbanken Online Lexikon (2011a)
[49] Vgl. Datenbanken Online Lexikon (2011a)

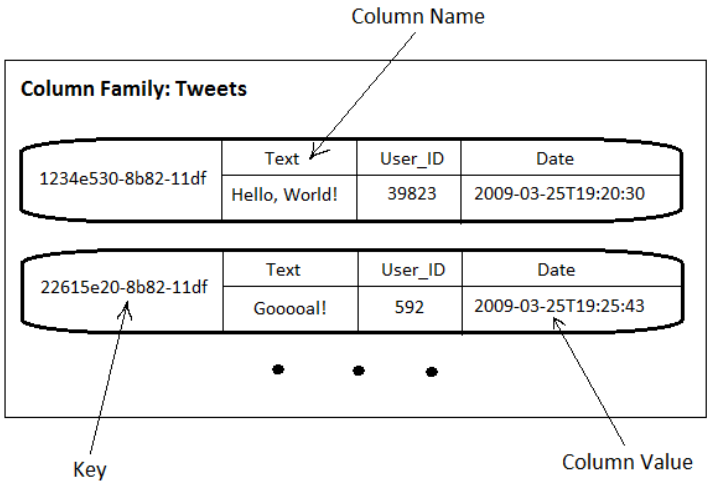


Abb. 5: Column Families[50]

Eine weitere Form der Speicherung kann in sogenannten „Supercolumns“ (siehe Abb. 6) erfolgen, das sind Spalten, welche mehrere der ursprünglichen Spalten als Grundlage nehmen beziehungsweise zu einer Spalte zusammenfassen. Diese Datenorganisation bietet vor allem Vorteile bei Operationen, die nur auf bestimmte Spalten abzielen, beispielsweise bei Aggregationen. Weitere Vorteile bestehen bei der Analyse der Daten und der Datenkompression. Nachteile andererseits können sich bei Such- oder Schreibzugriffen auf Daten oder Lese- und Schreiboperationen auf Objektstrukturen ergeben.[51]

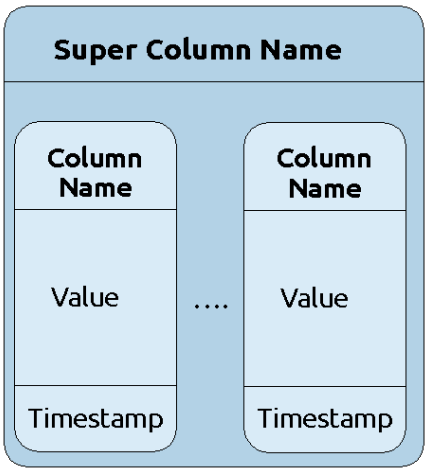


Abb. 6: Super Columns[52]

Weiterhin ist generell der Bereich der entscheidungsunterstützenden Datenauswertung („OLAP“ - Online Analytical Processing) und nicht der Transaktionsbereich (OLTP - Online Transaction Processing) betroffen. Für Letzteres sind im Allgemeinen Systeme mit höchster Konzentration auf Datenintegrität von Nöten, welche in der Form durch Wide Column Stores nicht gewährleistet werden. Reduktion auf die (für gewöhnlich wenig) abgefragten Spalten einer Tabelle, so müssen nicht alle Zeilen mit den nicht abgefragten Spalten in den Speicher geladen werden.

[50] Entnommen aus: Grunev, M. (2010)
[51] Vgl. Edlich, S. u. a. (2010), S. 53
[52] Entnommen aus: Datenbanken Online Lexikon (2011b)

3 Vertreter

3.1 MongoDB

Bei MongoDB handelt es sich um eine dokumentenorientierte Open-Source-Datenbank hinter der das Unternehmen 10gen steht. „Die Lösung ist in der Programmiersprache C++ implementiert und für die Betriebssysteme Windows, Mac OS X und Linux erhältlich. Sowohl 32-Bit- als auch 64-Bit-Systeme werden unterstützt.“[53]

Aufbau und Datenmodell

Bei MongoDB wird im Gegensatz zu relationalen Datenbanken nicht von Tabellen gesprochen, sondern von Kollektionen. Dabei kann jede Kollektion Dokumente beinhalten, welche im BSON-Format gespeichert und ausgegeben werden und dem Javaskript-Objekten sehr ähneln. Ein vergleichendes Beispiel des Aufbaus einer relationalen Datenbank zu einer MongoDB-Datenbank enthält der Anhang 2. BSON bedeutet binäres JSON und lässt sich durch Effizienz und Platzersparnis charakterisieren.[54] In diesem Zusammenhang ist zu beachten, dass Dokumente im BSON-Format auf eine Größe von 4 MB begrenzt sind. Allerdings bietet MongoDB einen speziellen Mechanismus mit der Bezeichnung GridFS, welcher die Speicherung größerer Dokumente ermöglicht. Zu diesem Zweck werden Dateien gesplittet und anschließend auf mehrere Dokumente verteilt.[55] Wie bereits erwähnt werden in MongoDB Daten in Form von Dokumenten abgespeichert. Dabei besteht ein Dokument aus einem sortierten Satz von Eigenschaften, welche sich wiederum aus einem Namen und einem Wert zusammen setzen.[56] Die Dokumente können eine beliebige Anzahl an Feldern besitzen und zusätzlich kann eine verschachtelte Array-Struktur realisiert werden. Außerdem ist es möglich Dokumente innerhalb eines Dokuments zu speichern. Es ist nicht zwingend erforderlich, dass die Dokumente einer Kollektion dieselbe Struktur besitzen, allerdings wirkt sich eine grundlegend gleiche Struktur positiv auf eine Effizienzsteigerung bei der Indizierung aus.[57] Ein Schema wird mit dem Einfügen des Dokuments zur Laufzeit erzeugt.[58] Für die Datenmanipulation und Datenabfrage bietet MongoDB objektspezifische Methoden an, die für alle gängigen Programmiersprachen vorhanden sind, und verzichtet somit auf eine Abfragesprache.[59] Eine Übersicht der gängigsten Operationen enthält der Anhang 3.

[53] Wyllie, D. (2011), S. 24
[54] Vgl. Alvermann, M. (2011), S. 16
[55] Vgl. Edlich, S. u. a. (2010), S. 117
[56] Vgl. Alvermann, M. (2011), S. 16
[57] Vgl. Edlich, S. u. a. (2010), S. 117
[58] Vgl. Edlich, S. u. a. (2010), S. 117
[59] Vgl. Wyllie, D. (2011), S. 25

Replikation und Skalierung

Gegen einen Severausfall kann MongoDB mit Hilfe von einem manuellen und einem automatischen Modus zur Replikation abgesichert werden. Das Prinzip hinter diesen Modi ist das Master-Slave-Prinzip und es beinhaltet einen Master, welcher alle Abfragen der Clients beantwortet, sowie einem Slave, der bei einem Ausfall den Ersatz des Mastes absichert und über eine Kopie von dessen Daten verfügt. Beim manuellen Modus wird die Rollenverteilung von Hand festgelegt und beim automatischen Modus übernehmen die beteiligten Server die Zuweisung eigenständig.[60] Eine weiterführende Funktion von MongoDB ist die Unterstützung eines automatischen horizontalen Skalierens der Datenbank, was auch oft unter dem Begriff Sharding erwähnt wird. Der Ausbau eines einzelnen Servers zu einem Cluster kann ohne große Umstände umgesetzt werden.[61]

Einsatzgebiete

„Da die Datenbank von Anfang an für den Einsatz in Webapplikationen vorgesehen war, lassen sich mit MongoDB Aufgaben wie Dokumentenmanagement, das Verwalten von Anwender- und Sitzungsdaten, Logging, Echt-Zeit-Analysen und generell Aufgaben mit einem hohen Datenaufkommen einfach und bequem lösen.“[62]

Bewertung

Die Entwicklung dieser Datenbank stellt eine Bündelung von langjährigen technischen Erfahrungen dar.[63] „MongoDB hat das Ziel, die Lücke zwischen klassischen relationalen Datenbanken und den Key/Value-Stores zu schließen.“[64] Die Bezeichnung lässt sich von dem englischen Begriff „humongous“ ableiten, was so viel wie „gigantisch“ oder „riesig“ bedeutet.[65] Anhand des Namens der Datenbank kann bereits auf eine ihrer wichtigsten Eigenschaften geschlossen werden: „die performante Verarbeitung großer Datenmengen“[66].[67] Neben vielfältigen Anbindungsmöglichkeiten kann MongoDB auch mit einer starken und aktiven Community punkten.[68]

Für weiterführende Informationen empfiehl sich das Buch von Marc Boeker mit dem Titel „Mongo DB - Sag ja zu NoSQL“.

[60] Vgl. Edlich, S. u. a. (2010), S. 127 f.
[61] Vgl. Edlich, S. u. a. (2010), S. 125
[62] Alvermann, M. (2011), S. 18
[63] Vgl. Edlich, S. u. a. (2010), S. 115 f.
[64] Edlich, S. u. a. (2010), S. 115
[65] Vgl. Wyllie, D. (2011), S. 24
[66] Alvermann, M. (2011), S. 16
[67] Vgl. Alvermann, M. (2011), S. 16
[68] Vgl. Edlich, S. u. a. (2010), S. 128

3.2 Neo4j

Neo4j Community, kurz Neo4j, ist eine Open-Source Graphendatenbank der Firma Neo Technology, die seit 2003 im Produktionseinsatz ist. Die Entwicklung erfolgt in der plattformunabhängigen Sprache Java. Neben der Open-Source Variante gibt es zwei Erweiterungen: Zum einen die „Advanced Version", die unter anderem durch ein Monitoring ergänzt ist, und zum anderen eine „Enterprise Version". Letztere bietet zusätzlich zum Monitoring die Möglichkeit eines High Availability Clusterings. Der Preis der „Advanced Version" liegt pro Instanz bei 6.000 EUR jährlich. Die „Enterprise Version" kostet 24.000 EUR jährlich.[69] Die Wahl einer der kommerziellen Versionen oder des Open-Source Produktes ist vom Einsatzgebiet der Graphendatenbank abhängig. Für die Entwicklung von Online-Anwendungen im Geschäftsbetrieb sind die kommerziellen Produkte zu bevorzugen, für Evaluations- und Testzwecke der Datenbank ist die „Community Version" ausreichend.

Aufbau und Datenmodell

Das Neo4j-Datenmodell speichert alle Datenstrukturen in Form eines Netzwerkes, das drei zentrale Elemente besitzt:

- Knoten
- Kanten mit Richtung und Typ
- Eigenschaften in Form von Schlüssel/Wert-Paaren[70]

Beim Anlegen einer neuen Neo4j-Datenbank wird automatisch ein Referenzknoten erzeugt. Er dient als Einstiegspunkt in die jeweilige Datenbank. Sowohl Knoten als auch Kanten können mit Eigenschaften versehen sein. Die Eigenschaften können für jeden einzelnen Knoten und jede einzelne Kante unterschiedlich sein. Dadurch ist es möglich, unsortierte anwendergenerierte Daten einfach abzulegen. Mit einer eindeutigen ID werden Knoten und Kanten identifiziert.

Replikation und Skalierung

Die Replikation der Daten erfolgt über das Master-Slave-Modell. Ein Server aus dem Cluster wird zum Write-Slave, alle anderen Server werden zum Read-Slave gewählt. Für den Fall, dass ein Read-Slave ausfällt, wird eine neue Instanz mit dem aktuellen Stand der Write-Master Datenbank erstellt. Sollte der Write-Master ausfallen, wird er durch einen Read-Slave ersetzt, der dann den neuen Write-Master darstellt.[71]

[69] Vgl. Neo Technology Inc. (2012)
[70] Vgl. Edlich, S. u. a. (2010), S. 185
[71] Vgl. Edlich, S. u. a. (2010), S. 193

Eine horizontale Skalierung von Neo4j ist nur dann notwendig, wenn mehr als 12 Milliarden Datenelemente angelegt oder die Schreibgeschwindigkeit erhöht werden soll. Die horizontale Partitionierung umfasst sowohl das Sharding, als auch die Partitionierung. Neo4j unterscheidet drei Möglichkeiten der Datenpartitionierung. Durch das Insert-Sharding werden alle Knoten eines Typs auf eine spezielle Partition abgelegt. Das Runtime-Sharding gibt vor, dass häufig zusammen abgefragte Knoten auf der gleichen Graphenpartition liegen. Die partielle Replikation ist eine Optimierung der Partitionierung. Daten von beispielsweise Partitionen 1 werden repliziert und auf Partition 2 übertragen, damit sie sich näher sind.

Bewertung

Im Unterschied zu anderen NoSQL-Datenbanken ist Neo4j ACID-konform, sofern es als eigenständiger Server läuft. Als Embedded System erfüllt Neo4j das BASE Prinzip. Das Durchsuchen des Graphen nach gewissen Kriterien erfolgt mittels eines Index. Neo4j nutzt dazu ein externes Indexsystem, mit welchem es beispielsweise Volltextsuchen bewältigen kann. Neo4j wird vor allem für die Verwaltung großer Datenmengen (10 Milliarden Knoten, Kanten und Eigenschaften) verwendet und ist in der Lage, bis zu 1 Millionen Beziehungen pro Sekunde zu traversieren. Wesentliche Einsatzgebiete der Datenbank sind Soziale Daten, Geographische Daten, Empfehlungssysteme und Content-Management-Systeme. Die Vorteile liegen in ihrer Robustheit, in einer großen Community, durch die die Anbindungen an einige Programmierschnittstellen wie C#, Ruby, Scala oder Python gewährleistet werden und im High Availability Clustering. Von Nachteil sind die Java- und die Sharding-Begrenzung. Die Java-Begrenzung führt dazu, dass die Performanz stark von der Java Virtual Machine auf dem Operativsystem abhängt. Die Sharding-Begrenzung beschränkt die automatisierbare Partitionierung eines Graphen, wenn alle Leistungsmerkmale des unpartitionierten Graphen beibehalten werden sollen.[72]

[72] Vgl. Edlich, S. u. a. (2010), S. 194

3.3 Voldemort

Dieses Key/Value-Datenbanksystem wurde von der Web-Firma LinkedIn entwickelt und gehört auch zu dieser. Die grundlegenden Konzepte des Systems wurden von Amazons Dynamo übernommen.[73]

Aufbau und Datenmodell

Ein Voldemort-Cluster kann mehrere Tabellen, sogenannte Stores, enthalten. Vereinfacht heißt dies, dass hinter dem System eine verteilte Hash-Tabelle steckt.[74] Für Key und Value der einzelnen Tabellen können verschiedene Datentypen verwendet werden, jedoch muss dieser für jeden Store fest eingestellt werden. Grundsätzlich ist der Datentyp frei wählbar, jedoch bietet Voldemort im Moment nur eine Implementierung der Serialisierung für folgende Typen:[75]

- json (JavaScript Object Notation) - ist ein Datenaustauschformat welches schon auf der Struktur von Name/Wert-Paaren aufbaut
- String
- java Object
- protobuf (Protocol buffers) - ist ein Datenformat von Google zur Serialisierung
- thrift - ist ein weiteres Datenformat zur Serialisierung
- avro-generic / avro-specific / avro-reflective - ist ein Datenserialisierungssystem
- identity (pure Bytes)[76]

Replikation und Skalierung

„Voldemort ist für eine unkomplizierte horizontale Skalierung ausgelegt.“[77] Ein Cluster ist immer in verschiedene Knoten eingeteilt, welche alle genau die gleichen Funktionen übernehmen. Der Aufbau eines Clusters kann somit mit einem Peer-to-Peer-Netzwerk verglichen werden, da es keinen verwaltenden beziehungsweise steuernden Knoten gibt. Die einzelnen Datensätze werden nach dem Consistent-Hashing Prinzip, das von Amazon übernommen wurde, auf die einzelnen Knoten verteilt.[78]

[73] Vgl. Edlich, S. u. a. (2010), S. 262
[74] Vgl. Edlich, S. u. a. (2010), S. 263
[75] Vgl. Project Voldemort (o. J.)
[76] Vgl. Project Voldemort (o. J.)
[77] Edlich, S. u. a. (2010), S. 263
[78] Vgl. Edlich, S. u. a. (2010), S. 263

Zur Veranschaulichung des Prinzips dient die untenstehende Abbildung 7. Die einzelnen Knoten sind anschaulich gesprochen in einem Ring angeordnet, wobei jeder Knoten entsprechend des Hash-Werts seines Namens, seiner IP-Adresse oder anderen Merkmalen eingefügt wurde. Der jeweilige Datensatz wird so jeweils dem Knoten mit dem im Urzeigersinn nächstgelegenen Hashwert zugeordnet.[79]

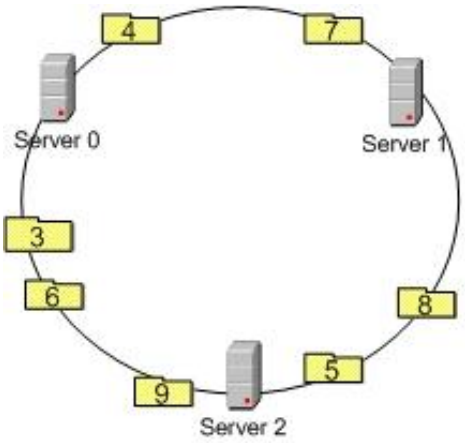


Abb. 7: Consistent-Hashing Verfahren[80]

Eine weitere Eigenschaft von Voldemort ist ein eingebauter Replikationsmechanismus. Dabei kann für die Stores jeweils festgelegt werden, wie oft die Daten repliziert werden sollen. Die einzelnen Replika werden dann automatisch auf die verschiedenen Knoten verteilt. Somit ist auch eine gute Ausfallsicherheit gegeben.

Architektur

Ein interessanter Aspekt, der Voldemort auch von anderen Systemen unterscheidet, ist die mehrschichtige Architektur. Dabei implementiert jede Schicht ein Interface für PUT, GET und DELETE. Zudem erfüllt jede Schicht eine eigene Funktionalität wie beispielsweise das Routing der Daten auf die einzelnen Knoten oder den Versionsabgleich. Die einzelnen Schichten sind sehr flexibel und können unterschiedlich implementiert werden. Es können beispielsweise auch neue Zwischenschichten entwickelt werden.[81] Der Aufbau der logischen Architektur ist in folgendem Schaubild der Abbildung 8 dargestellt.

[79] Vgl. Edlich, S. u. a. (2010), S. 38
[80] Enthalten in: Datenbanken Online Lexikon (2011c)
[81] Vgl. Project Voldemort (o. J.)

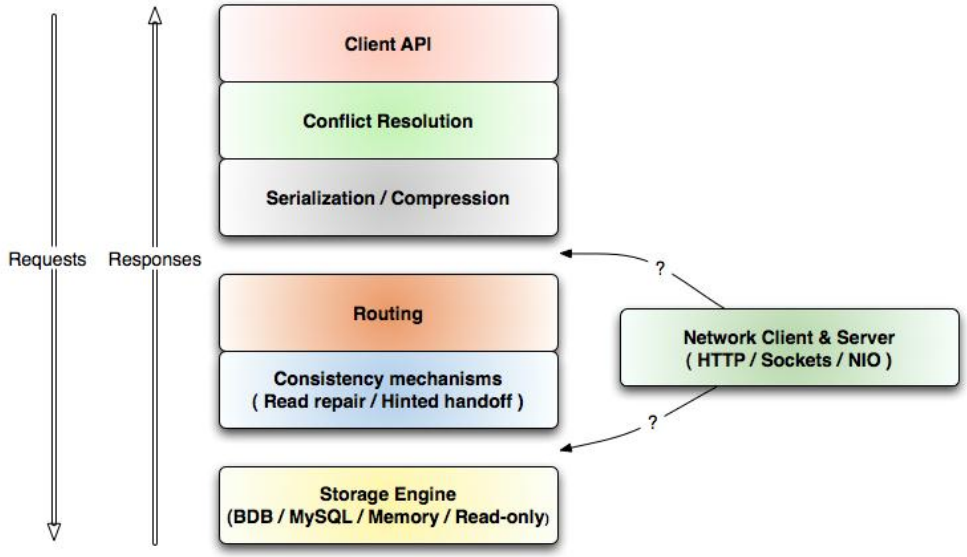


Abb. 8: Logische Architektur von Voldemort[82]

Ob die Netzwerkschicht über oder unter der Routingschicht liegt ist davon abhängig, ob die Daten serverseitig oder clientseitig geroutet werden.

Bewertung

Da ein Hauptaugenmerk bei dem System auf der Performance liegt und auch möglichst viele Schreiboperationen parallel ablaufen sollen, wird weniger Wert auf die ständige Konsistenz der Datensätze und ihrer Replikas gelegt. Die Konsistenz wird immer nur bei einem Lesezugriff über die Versionierung der Werte wieder hergestellt. Die Versionierung funktioniert so, dass beim Abspeichern der Daten jeweils der Knoten, von dem aus geschrieben wird, und die verwendete Version vermerkt werden. So kann später beim Lesen der konsistente Wert berechnet werden.[83] Die Abfragemöglichkeiten bei Voldemort sind noch recht begrenzt, so ist im Moment nur das Einfügen, Lesen und Löschen verschiedener Key/Value-Paare möglich.[84]

Weitere Informationen zu Voldemort sind hauptsächlich auf der Seite des Projekts unter http://www.project-voldemort.com zu finden.

[82] Enthalten in: Project Voldemort (o. J.)
[83] Vgl. Project Voldemort (o. J.)
[84] Vgl. Edlich, S. u. a. (2010), S. 262

3.4 Cassandra

Cassandra ist eine Entwicklung von Facebook und wurde zur Speicherung und Abfrage der Nachrichten der User untereinander verwendet. Fachlich lehnt es sich an Amazons Dynamo und Googles BigTable an, sticht jedoch durch seinen hybriden Charakter hervor. Cassandra nutzt einerseits Key/Value-Eigenschaften und setzt andererseits auf eine flexible Schemaunterstützung. Cassandra will hierbei eine an SQL-Datenbank erinnernde Schemasicherheit bei größtmöglicher Flexibilität und Skalierbarkeit bieten. Letzteres, basiert hierbei auf einer verteilten Architektur („Cassandra-Ring“) mit einer starken horizontalen Skalierbarkeit, die ihrerseits durch die Aufnahme vieler Knoten in ein Cluster realisiert wird. Damit einher geht eine hohe Verfügbarkeit bei eingeschränkter, das heißt zu einem bestimmten Zeitpunkt erreichter, Datenkonsistenz („eventual consistency“).[85]

Aufbau und Datenmodell

Die erste zu definierende Datenstruktur ist - wie im Punkt 2.2 schon erwähnt - der „Keyspace“ (oder auch „Table“), welcher zumeist einer bestimmten Anwendung zugeordnet ist. Zu jedem „Keyspace“ können „Column Families“ deklariert werden, was jedoch vor Anwendungsstart passieren muss und somit den nicht-dynamischen Teil der Schemaverwaltung unter Cassandra darstellt. Die Column Families beschreiben hierbei in etwa die Pendants zu den wichtigsten Tabellen innerhalb eines SQL-DBMS. Die Keyspaces und die Column Families werden in der Datei storage-conf.xml eingetragen und gespeichert. Diese Datei wird beim Start von Cassandra eingelesen. Die eigentlichen Datenstrukturen und Inhalte, d.h. die Columns, Super-Columns sowie die Daten werden erst zur Laufzeit definiert (sofern weder Import noch Migration vorliegen). Hierbei wird jeder Eintrag über einen Key identifiziert, welcher durch eine von Cassandra automatisch erzeugten Index realisiert wird. Wenn in einer CF viele zueinander gehörende Daten gespeichert werden sollen, ist es besonders günstig die entsprechenden Daten in Super Columns zu strukturieren. Eine Super Column besitzt einen Namen und eine Liste von Columns, so werden also mehrere Columns in einer Column zusammengefasst und zu einer Einheit aggregiert. Welche Unterspalten die Super-Column enthält muss jedoch erst zur Laufzeit deklariert werden.[86]

[85] Vgl. Edlich, S. u. a. (2010), S. 69 ff.
[86] Vgl. Edlich, S. u. a. (2010), S. 71 ff.

Neben der Modellierung der CFs benötigt Cassandra eine weitere Information. Bei dem Einfügen werden die Columns sortiert. Dies geschieht im Allgemein anhand ihres Namens sortiert. Bei der Deklaration muss angegeben werden, wie die Einträge pro Column Family bei der Sortierung miteinander verglichen werden sollen (zum Beispiel über einen Zeitstempel des Eintrags, den Byte-Wert oder den UTF8-Wert des Namens).

Beim Beladen der CFs durch Anwendungen sind einige Faktoren zu beachten. Allgemein liegt hier der klassische Schemagedanke zu Grunde, welcher besagt, dass Schlüssel statisch und Werte variabel gehandhabt werden. Abweichend davon werden die Spaltennamen zur Laufzeit frei vergeben. Weiterhin besteht keine Bindung an die vorhandenen Spalten, das heißt es müssen einerseits nicht alle Columns der Column-Family belegt werden und andererseits können flexibel bei der Eintragung Spalten dazu kommen, wenn beispielsweise die Anwendung beschließt die Email des Users zusätzlich zu speichern.[87]

Abfragen in Cassandra

Weiterhin besteht eine Abweichung darin, dass die Schlüssel auch variabel eingesetzt werden können. Sie können demnach auch mit echten Daten gefüllt werden. Auf diese Art und Weise lassen sich bei großen Datenaufkommen performante Bereichsabfragen je Column Family formulieren. Eine andere weit-verbreitete Variante besteht in der Nutzung eines Zeitstempels als Schlüssel. Monoton steigende Zeitstempel wie zeitbasierte UUIDs werden in der Regel in den Bibliotheken der jeweiligen Programmiersprache angeboten. Wichtig hierbei ist nur, Cassandra vorher mitzuteilen, wie die Sortierung der Schlüssel der Column Family stattfinden soll[88]. Die Abfrage selbst kann über CQL oder Konsolenkommandos erfolgen.

Replikation und Skalierung

Replikation und Skalierung sind bei Cassandra sehr eng aneinander gekoppelt. Der Cassandra-Ring wird durch eine Consistent-Hashing Strategie verwaltet und erlaubt dadurch kein hin-und herspringen zwischen den Knoten, bei welchem eine Instanz gefragt wird welcher Knoten verantwortlich ist und demenstprechend weitergereicht wird Bei Cassandra wird der Ring über sogenannte Tokens gesteuert, die einen 2^127 großen Adressraum verteilen. Jeder Knoten der den Ring erweitern soll, benötigt lediglich die Adresse eines anderen Knotens (seed node).

[87] Vgl. Edlich, S. u. a. (2010), S. 71 ff.
[88] Vgl. Edlich, S. u. a. (2010), S. 79

Das Bootstrap-Verfahren sorgt dann dafür, dass eine optimale Verteilung des Wissens mit der Bekanntmachung welcher Knoten wofür verantwortlich ist, stattfindet. Bei der Erweiterung des Rings muss sich das System nun zuerst einpendeln, was am Anfang für höhere Latenzzeiten sorgen kann. Weiterhin können auch Knoten aus dem Ring entnommen werden und somit eine Rück-Skalierung ermöglichen.[89]

Bei der Datenkonsistenz kann zwischen diversen Consistency-Levels für Lese- (r) und Schreibbefehle (w) - oder individuell pro Operation - gewählt werden. Hierbei wird durch die verschiedenen Optionen festgelegt, auf welche Art und wie viele Knoten reagieren müssen bevor der Client eine Antwort erhält. Eine entsprechende Übersicht befindet sich im Anhang 4. Dies ermöglicht ein Abwägen zwischen Performance und Konsistenz. Bei jedem Read-Befehl vergleicht Cassandra die Versionen der Daten. Veraltete Datenknoten. Durch die „nodetool repair"-Operationen können Knoten dazu angehalten werden, derartige „Aktualisierungsreparaturen" selbstständig auszuführen. Ausgefallene Knoten können durch neue Instanzen und das Bootstrap-Verfahren ersetzt werden.

Alle Operationen werden initial im Cache durchgeführt, daher ist es sinnvoll Cassandra mit einem hohen RAM-Zugriff auszustatten. Schreiboperationen werden geloggt und im Hauptspeicher abgelegt, bevor sie später durch ein „commit-log" ausgeführt und per „flush"-Operation versendet werden. Anstatt von Locks bedient sich Cassandra hierbei einer optimistischen Replikation, welche üblicherweise auf drei Knoten eingestellt ist.[90]

Einsatzgebiet

Cassandra bietet sich vor allem für skalierende Webanwendungen an, die viele (mehrere hundert) Nodes nutzt, keine hochkomplexen Abfragen benötigt, aber bei denen das System stabil antworten und skalieren muss.[91]

[89] Vgl. Edlich, S. u. a. (2010), S. 80 f.
[90] Vgl. Edlich, S. u. a. (2010), S. 81
[91] Vgl. Edlich, S. u. a. (2010), S. 71

Bewertung

Die wesentlichen Vor- und Nachteile von Cassandra enthält die Tabelle 5.

Vorteile	Nachteile
• sehr einfache Skalierbarkeit • Datenmodell ist relativ flexibel • Einsatz von Indizes • Bereichsabfragen über die Sortierung der Daten möglich • leichte Konfiguration von Replikationen über verschiedene Knoten und kein „Single Point of Failure“ • Konsistenz, Dauerhaftigkeit und Latenzzeit sind recht gut konfigurierbar • grafische Oberflächen verfügbar • SQL-ähnliche Abfragesprache CQL	• begrenzte Abfragemöglichkeiten • Schemaänderungen der Datei storage-con.xml sind im laufenden Betrieb nur schwer möglich • Verschiedene Speicherformate in bisherigen Versionen erschweren Updates • synchronisierte Uhren für Konflikthandling nötig • neue Abfragen benötigen unter Umständen eine Anpassung des Schemas

Tab. 5: Vor- und Nachteile von Cassandra[92]

3.5 Gegenüberstellung der Vertreter

Grundsätzlich lassen sich die einzelnen Datenbankvertreter nur schwer miteinander vergleichen, da Sie für völlig unterschiedliche Nutzungsmöglichkeiten gedacht sind. Je nach Typ gibt es andere Anforderungen an das System.

Um jedoch einen gewissen Gesamtüberblick zu schaffen und die Eigenschaften der Datenbanken kurz zusammen zu fassen, wurden die verschiedenen Vertreter in der folgenden Tabelle gegenübergestellt. Sie werden so anhand ganz allgemeiner Kriterien aus dem Bereich der Datenbanken verglichen.

[92] Vgl. Edlich, S. u. a. (2010), S. 82 f.

Kriterien	Cassandra	MongoDB	Voldemort	Neo4j
Kategorie	spaltenorientierte DB	dokumentenorientierte DB	Key/Value DB	graphenorientierte DB
Anwendungsbereiche	• Blogging Dienste (z. B. Wordpress, Twitter), • Inbox-Verwaltung • große Datenmengen mit „gleichbleibenden“ Abfragen und flexiblen Schreiboperationen	• Dokumentenmanagement • Verwaltung von Anwender- und Sitzungsdaten • Aufgaben mit einem hohen Datenaufkommen	• Microblogging-Dienste • Sessionverwaltung • Artikelverwaltung • Userverwaltung	• Soziale Netzwerke • Routenplanung • Empfehlungssysteme
API	Thrift (Support für viele Sprachen u. A. Java, Python, Ruby, PHP, C++, C#)	C, C++, Java, PHP, Ruby, Perl, Phython und viele mehr (über Treiber)	Java, Python, Ruby, C++	Java, REST, JRuby, Ruby, Python, Jython, Scala, Clojure, C#
Abfragemöglichkeiten (Anzahl der Möglichkeiten für eine Datenabfrage)	• Abfrage über Key oder indexierte Spalten mit CQL- oder CLI-Befehlen	• Abfrage über ObjectId • Abfrage über Werte	• nur über Key • Abfragemöglichkeiten PUT, GET, DELETE	• Als Embedded System volle Power von Java. • Als standalone Server Cypher oder Gremlin als Abfragesprachen
Betriebssysteme	• Windows • Mac OS X • Linux	• Windows • Mac OS X • Linux	• Linux • Mac OS X • Windows	• Windows • Mac OS X • Linux
CAP Typ	AP	CP	AP	AP
Datenmodell	Column Families mit Spalten	Dokumente mit Schlüssel-Wert-Paaren mit dem Zusatz, dass die Datenstruktur im Wert interpretiert wird	Schlüssel-Wert-Paare	Property-Graph-Modell
Datennavigation (wie sind Daten mit einander verknüpft)	alle relevanten Werte als Spalte in Column Family enthalten	Dokument bündelt alle Informationen an einer Stelle	Navigation nicht möglich	Navigation über Kanten (Beziehungen)
Datenschema	größtenteils schemafrei (ausgenommen Keyspaces und Column Families)	schemafrei	Schema pro Store festzulegen	schemafrei

Kriterien	Cassandra	MongoDB	Voldemort	Neo4j
Lizenz	Apache License, Version 2, + ggf. DataStax/ DataStax Enterprise (für Produktion und Forschung, letzteres kostenfrei)	GNU AGPL v3.0	Apache License	GPL, AGPL und kommerziell
Performanz	++	++	+++	++
Replikation	über Hadoop HDFS, Replikationsanzahl bzw. -strategie im Ring konfigurierbar	Master-Slave-Modell	ja (wie oft wird pro Store festgelegt)	Master-Slave-Modell
Skalierbarkeit (Leistungssteigerung bei mehreren Knoten)	horizontale Skalierung, einfaches Hinzufügen und Abnehmen von Knoten (Bootstrapping)	automatisches Sharding	horizontale Skalierung	domänenspezifisch, horizontales Sharding
Support	Online-Community - Support (DataStax Community), Kommerzieller Support durch Drittanbieter	kommerzieller Support über 10gen	nicht verfügbar	Online-Community-Support (Forum)
Webadresse	http://apache.cassandra.org http://www.datastax.com/download	http://www.mongodb.org	www.project-voldemort.com	http://neo4j.org

Tab. 6: Gegenüberstellung der NoSQL Datenbankvertreter

4 Prototypen

4.1 MongoDB

Für die Installation von MongoDB stellt das Unternehmens 10gen unterschiedliche Downloadvarianten für die Betriebssysteme Windows, Mac OS X oder Linux bereit. Für die Seminararbeit ist die Wahl auf das Betriebssystem Windows gefallen. Eine übersichtliche Installationsanleitung für dieses Betriebssystem befindet sich auf der Internetseite http://www.mongodb.org/display/DOCSDE/Quickstart+Windows.

Die Installation von MongoDB auf einem 32 Bit-Windows-Betriebssystem kann durch die folgende Schritte realisiert werden:

- Download der Windows 32-Bit Variante des Production Release (Recommended)
- Datei in einen selbsterstellten Ordner im Verzeichnis (C:) entpacken
- Erstellung eines Dateiverzeichnisses unter C:\data\db mit Hilfe des Windows Explorers
- Start des Datenbankservers durch Auswahl der ausführbaren Datei im Verzeichnis C:\selbsterstellter Ordner\mongodb-win32-i386-1.8.1\bin\mongod.exe
- Start von MongoDB durch Auswahl der ausführbaren Datei im Verzeichnis C:\selbsterstellter Ordner\mongodb-win32-i386-1.8.1\bin\mongo.exe
- Verbindung zwischen Datenbankserver und MongoDB wird automatisch hergestellt und die Testdatenbank wird geöffnet
- Operationen können nun direkt über die Konsole ausgeführt werden[93]

Mittlerweile gibt es auch einige graphische Oberflächen für die Datenbank, welche die Datensichtung und Datenbearbeitung erleichtern sollen. Hierzu zählen unter anderem folgende Anwendungen:

- UMongo als Anwendung für Windows, Mac OS X oder Linux
- MongoVUE als Anwendung für Windows
- MongoHub als Anwendung für Mac OS X[94]

[93] Vgl. 10gen Inc. (2012a)
[94] Vgl. 10gen Inc. (2012b)

4.2 Neo4j

Neo4j ist unter den Betriebssystemen Linux, Mac OS/X sowie Windows lauffähig[95]. Es kann als eigenständiger Server oder auf mehreren verschiedenen Maschinen verteilt laufen. Daneben besteht die Möglichkeit Neo4j in die Applikation einzubinden und als Embedded System laufen zu lassen.[96]

Um Neo4j zu installieren, kann die aktuellste Version unter dem folgenden Link http://www.neo4j.org/download heruntergeladen werden. Abhängig vom Betriebssystem ist die entsprechende Version zu wählen.

Im Folgenden wird zum einen die Einbindung von Neo4j in eine Java-Applikation beschrieben und die Benutzung von Neo4j als eigenständiger Server.

Um Neo4j in Java einzubinden, müssen die library jars in den Build Path des Projektes übernommen werden. Wird Eclipse als Entwicklungsumgebung verwendet, gelangt man über einen Rechtsklick auf das entsprechende Projekt zu *Build Path -> Configuration Build Path*. Unter *Add External JARs* werden alle jar-Dateien aus dem Ordner Neo4j *lib*/directory ausgewählt und nach dem Klick auf den Button „Ok“ in das Projekt eingebunden.[97]

Nun kann Neo4j als Embedded System genutzt werden. Im ersten Schritt wird eine Instanz der Graphendatenbank angelegt. Alle Operationen wie *hinzufügen, ändern* oder *löschen* von Knoten und Beziehungen müssen in einer Transaktion gekapselt werden. Dies erfordert das Design Konzept von Neo4j. Ein Beispielcode, der eine Datenbank anlegt, sie mit Werten befüllt und die Daten am Ende löscht, befindet sich im Anhang 5.

Um Neo4j als Server auszuführen, muss jdk6 installiert werden. Anschließend kann der Server gestartet werden. Dazu wird die Konsole als Administrator ausgeführt und der Befehl *bin\Neo4j.bat start* eingegeben. Unter *localhost:7474* steht eine Weboberfläche zur Verfügung die unter anderem die Möglichkeit bietet den Graphen visuell darzustellen.

Neo technology stellt außerdem vordefinierte Datasets zur Verfügung. Diese können in die Datenbank eingespeist werden und bieten somit die Möglichkeit, die Datenbank mit einer Vielzahl von Daten auszutesten. Heruntergeladen werden die Testdaten unter dem Link http://www.neo4j.org/develop/example_data. Zudem stellt Neo Technology einen LiveTry zur Verfügung, der es ermöglicht, die Datenbank - ohne sie herunterzuladen - auszuprobieren.

[95] Vgl. solid IT (2012)
[96] Vgl. Neo4j (2012a)
[97] Vgl. Neo4j (2012b)

4.3 Voldemort

Grundsätzlich ist Voldemort betriebssystemunabhängig. Der Prototyp wird jedoch unter Linux beziehungsweise Ubuntu realisiert, da eine Installation unter Windows die Herausforderung mit sich bringt verschiedene Shell Skripte auszuführen und sich deutlich aufwändiger gestaltet. Somit bezieht sich die folgende Anleitung auch auf Linux.

Die jeweils aktuellste Version der Datenbank kann von folgender Seite heruntergeladen werden https://github.com/voldemort/voldemort/downloads. Voraussetzung für die Installation ist eine Installation von Java6 und Apache Ant. Bevor die Datenbank gestartet werden kann, müssen die einzelnen Skripte und Konfigurationsdateien an die jeweiligen Gegebenheiten angepasst werden. Eine Anleitung für die Installation und Konfiguration der Datenbank findet sich im Anhang 6 sowie unter dem Link http://cloudbuzz.wordpress.com/2009/10/13/ voldemort-installation-to-first-run/.

Nachdem die Datenbank gestartet ist, kann jeweils auf die verschiedenen Stores zugegriffen werden. Die Definition der Stores wird in der Konfigurationsdatei stores.xml hinterlegt. Nur über diese können neue Stores angelegt und beispielsweise die Datentypen für key und value der jeweiligen Stores hinterlegt werden.

Um mit der Datenbank zu arbeiten gibt es zum einen die Möglichkeit die Befehle über die Konsole einzugeben und mit dieser zu arbeiten und zum Anderen direkt aus einem Programm in den Sprachen Java, Ruby, Python oder C++ auf die Datenbank zuzugreifen. Eine grafische Oberfläche ist zum aktuellen Zeitpunkt nicht verfügbar.

Für den Prototypen selbst wurden, um im Bereich der Sozialen Netzwerke zu bleiben, beispielhaft folgende Datensätze in ein Store eines Test-Cluster der Datenbank eingefügt:

Max	Max / Mustermann / 01.01.1988 / Berlin
Leo	Leo / Luchs / 09.09.1990 / Köln
Marta	Marta / Maier / 25.04.1987 / Stuttgart

Tab. 7: Beispieldatensätze eines Voldemort-Stores

Die Datensätze bilden die Speicherung von Profilinformationen zu verschiedenen Nutzern ab. Als Key wird jeweils der Username verwendet. Die Attribute des Userprofils werden in Form von Strings miteinander verknüpft und bilden den Value. Wird die Datenbank aus einem Anwendungsprogramm heraus genutzt, können die jeweiligen Profilinformationen auch in Form eines Objekts abgelegt werden. Da für den Prototyp aus zeitlichen Gründen nur die Verwendung der Konsole möglich ist, können jedoch lediglich Strings und keine Objekte abgelegt werden.

Eine Abfrage der Profilinformationen von Max sieht so aus: GET „Max“. Mit folgendem Befehl können die Profilinformationen von Leo abgelegt werden: PUT „Leo“ „Leo / Luchs / 09.09.1990 / Köln“. Um einen Datensatz zu ändern muss ein weiterer Datensatz mit dem gleichen Key und verändertem Value abgelegt werden. Voldemort überschreibt den alten Satz dann automatisch.

4.4 Cassandra

Generell wird Cassandra für alle gängigen Plattformen angeboten. Bei der Installation gibt es zwei Möglichkeiten. Die erste Variante ist, dass man sich von der Apache-Webseite die Cassandra Software herunter lädt. Nachdem entpacken der Zip-Datei müssen Umgebungsvariablen für das Java sowie für das Cassandra-Verzeichnis und die Pfade für die Daten und das Commitlog angelegt werden. Danach können Server und Client gestartet werden. Eine genaue Beschreibung der Installation und Konfiguration befindet sich im Anhang 7.

Die zweite Möglichkeit besteht darin, ein Cassandra-Gesamtpaket von der DataStax Community zu verwenden. Dieses Paket enthält eine CLI Utility, eine CQL Shell zur Verwendung von CQL-Befehlen und ein grafisches Browser-Interface, das DataStax Operation Center. Das Operation Center enthält unter anderem Komponenten zur Visualisierung der Datenbankauslastung, der Datenbankoperationen, der Knotenverteilung sowie die Anzeige der Daten selbst. Dafür wird die DataStax Software heruntergeladen und entpackt. Danach muss lediglich der Installationsroutine gefolgt und eine Zustimmung zur Lizenzvereinbarung erteilt werden. Das Anlegen aller Pfade und Variablen automatisch.

Im Datastax Ops Center können für die Keyspaces und Column Families mit den jeweiligen Vergleichs- und Sortierungswerten über ein Formular angegeben werden. Die Befüllung der Datenbank mit Daten geschieht über die CQL Shell oder die CLI Utility. Das Hinzufügen weiterer Knoten über das Ops Center in die Datenbank erfordert allerdings eine Registrierung bei Datastax.

Für Forschungszwecke bleibt die Lizenz kostenfrei, für Produktionszwecke jedoch nicht.[98] Bei INSERT-Operationen ist nicht von Belange, welche Spalten tatsächlich oder in welcher Reihenfolge sie gefüllt werden. Die Zuordnung erfolgt anhand des Namens, der jedem Datenwert anhaftet. Neben den vorgestellten Ansätzen ist es ebenso möglich Cassandra über andere Programmiersprachen und -schnittstellen zu benutzen, wie zum Beispiel PHP, Python oder Java (Hector Client).[99]

Der Prototyp von Cassandra soll nun eine Datenbank darstellen, die Nachrichten zwischen Benutzern in einem sozialen Netzwerk verwaltet. Vorangestellt muss der Keyspace ‚Social-Network' sowie die Column Family ‚Messages' erstellt werden. Nachfolgend kann die CQL Shell geöffnet und die Spalten der Datenbank deklariert werden. In diesem Beispiel werden alle Informationen benötigt, die für die Zustellung einer Nachricht benötigt werden. Als Key wird hier ein Zeitstempel verwendet. Dahinter werden, die einzelnen Spalten: User_ID_From, User_Name_From, User_ID_To. User_Name_to, Subject und Message deklariert und befüllt:

Key	Spalten					
Name: Key Value: 2013-01-10 19:20:34 Timestamp:	Name: User_ID_From Value: ‚Sophie. Lingelbach@gmx.de Ts: …	N: … V: ‚Sophie' Ts:…	N: … V: ‚Tana.B @gmx.de Ts: …	N: … V: Tana Ts:…	N: … V: NoSQL Projekt Ts:…	N: V: ‚Hey Tana, …' Ts: …

Tab. 8: Beispiel Cassandra

Nach dem zu Grunde liegenden Modell können die Einträge der Datenbank erst einmal nur nach dem Key abgefragt werden. Um nach einzelnen Spalten zu Filtern, muss ein Index darauf gelegt werden, zum Beispiel kann ein Index auf die Spalte ‚User_Name_From' gelegt werden um danach zu ermitteln wie viele Nachrichten ‚Sophie' geschrieben hat. Die Operationen aus dem Beispiel können aus Anhang 8 entnommen werden.

[98] Vgl. Datastax (2012a)
[99] Vgl. Datastax (2012b)

5 Zusammenfassung

Zusammenfassend lässt sich sagen, dass jeder der vier NoSQL-Datenbanktypen aufgrund seiner Eigenschaften spezielle Einsatzgebiete abdeckt, für die er sich besonders eignet. Somit besteht eine Begrenzung der Anwendungsgebiete dieser Datenbanken und sie können deshalb keinesfalls als vollumfänglicher Ersatz für eine relationale Datenbank angesehen werden.

MongoDB ist als dokumentenorientiertes Datenbanksystem besonders geeignet für einen Einsatz im Bereich des Dokumentenmanagements oder generell für Aufgaben mit einem hohen Datenaufkommen. In Sachen Performance kann dieses System auf jeden Fall punkten. Das entwickelnde Unternehmen bietet kommerziellen Support für sein Produkt an und außerdem wird mittlerweile eine Vielzahl an graphischen Oberflächen für die Anwendung angeboten.

Neo4j als graphenorientiertes Datenbanksystem eignet sich für Soziale Netzwerke, Routenplanung sowie Empfehlungssysteme. Die Anwendung ist einfach zu implementieren und bietet eine große Community. Für erste Testzwecke bietet das entwickelnde Unternehmen auf seiner Webseite einen LiveTry an, der noch keine Installation der Datenbank erfordert. Zusätzlich werden hier auch Testdatensätze zur Verfügung gestellt. Auch Neo4j kann über zusätzliche graphische Oberflächen bedient werden.

Das Key/Value-Datenbanksystem Voldemort bietet Einsatzmöglichkeiten im Bereich der Microblogging-Dienste, Sessionverwaltung oder auch Artikelverwaltung. Der große Vorteil dieser Datenbanktypen liegt in der Performance und Skalierbarkeit, welcher erst in einer großen Umgebung und mit einem hohen Datensatzaufkommen richtig zur Geltung kommt. Voldemort bietet viele Programmierschnittstellen an, aber leider keine grafischen Oberflächen. Auch die Dokumentation ist nicht sehr ausgereift und nur auf Englisch verfügbar. Gegebenfalls bietet es sich unter diesen Umständen an, einen weiteren Vertreter der Key/Value-Datenbanken auf seine Eignung zu überprüfen.

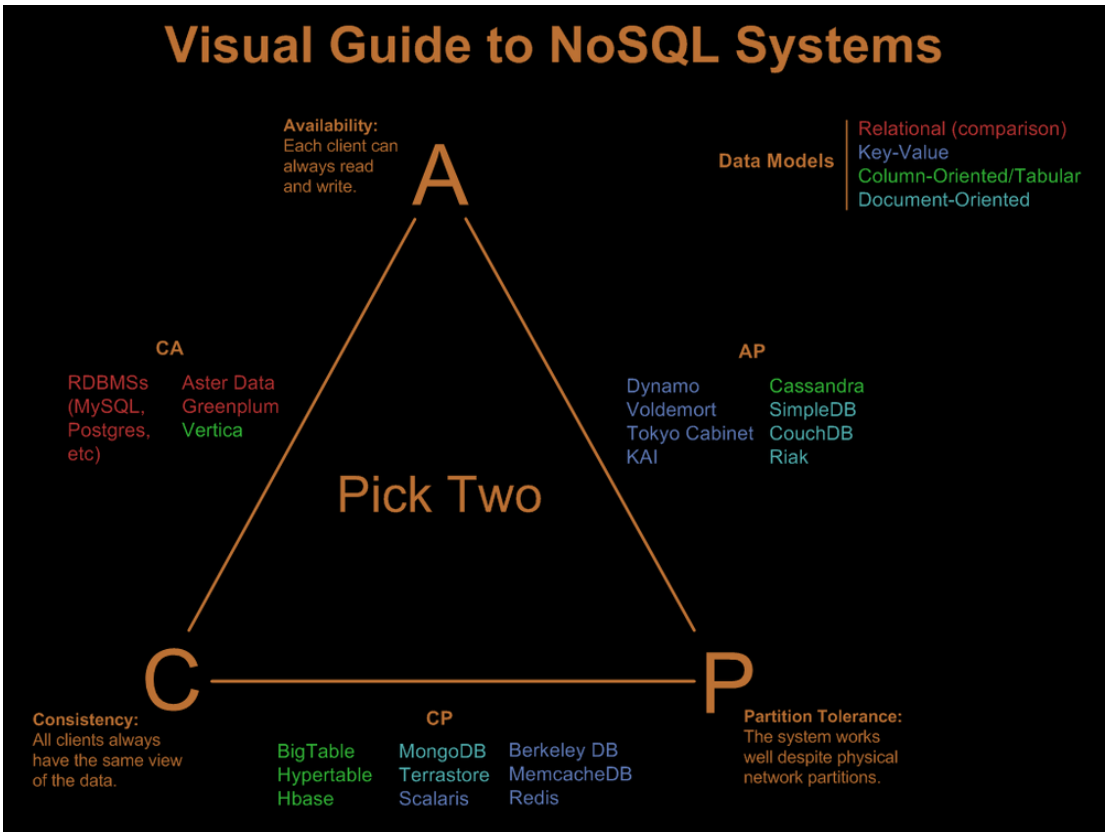
Das spaltenorientierte Datenbanksystem Cassandra bietet sich vor allem für skalierende Webanwendungen an. Der Vorteil liegt hier in einer großen erfahrenen Community sowie zusätzlichen Foren. Komplexe Abfragen wie im relationalen Modell können nicht umgesetzt werden, dafür bietet das Datenbanksystem aber eine höhere Performance und keinen großen Umgewöhnungsaufwand von SQL. Zusätzlich gibt es graphische Oberfläche und viele Konfigurationsmöglichkeiten.

Aufbauend auf dieser Arbeit ist eine umfassende Analyse der vorhanden graphischen Oberflächen für die unterschiedlichen Datenbanksysteme zu empfehlen. Weiterführend können die Performanceeigenschaften sowie die Skalierbarkeit in einem großen Umfeld getestet werden. Dies war im Rahmen dieser Arbeit leider nicht möglich.

Anhang

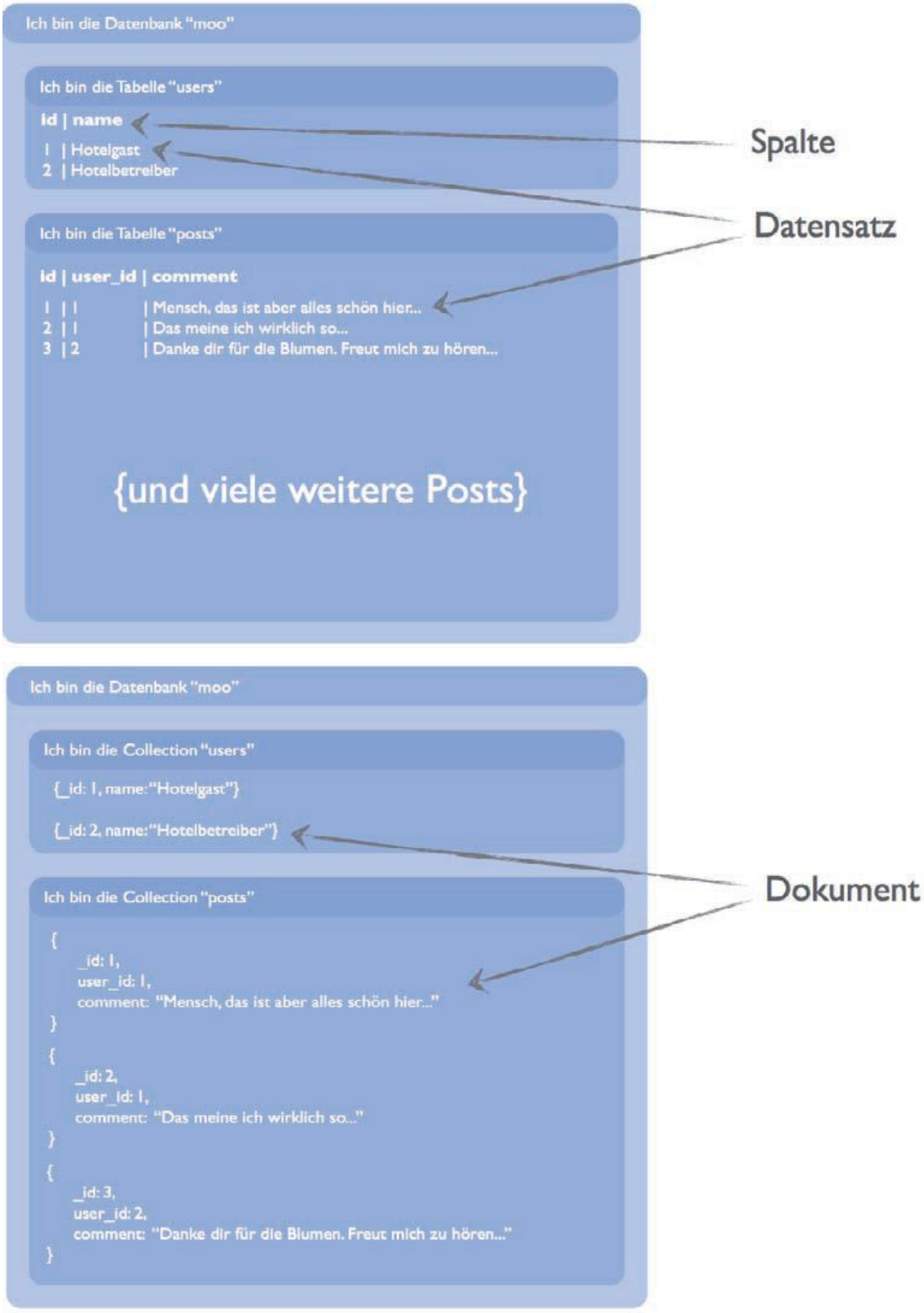
Anhangverzeichnis

Anhang 1: NoSQL Systeme[100]



[100] Entnommen aus: Beany (2011)

Anhang 2: Vergleich einer relationalen Datenbank mit einer MongoDB-Datenbank[101]



[101] Entnommen aus: Boeker, M. (2010), S. 42 f.

Anhang 3: Operationen für MongoDB[102]

Syntax
db.<collection>.<action>(<parameters>);
Einfügeoperation
Anlage eines einfachen Dokuments *db.person.insert({* *name: "fritz",* *age: 20,* *hobbies: ["soccer", "mongodb", "cheeseburger"]* *});*
Leseoperationen
Abfrage der Anzahl an Dokumenten in der Datenbank *db.cities.count();* Abfragen eines Dokuments anhand der ObjectId *db.cities.findOne(ObjectId("4c448e940d897c6443e2f70b"));* Abfrage aller Dokumente mit einem bestimmten Wert *db.cities.find({city: "ATLANTA"})* Abfrage mit mehreren Bedingungen *db.cities.find({key_1: "value_1", key_2: "value_2", key_3: "value_3"})*
Änderungsoperation
Änderung eines Dokuments *db.person.update(* *{name: "hannes"},* *{name: "hans", age: "21"}* *);*
Löschoperation
Löschung von Daten *db.cities.remove({city: 'ATLANTA'});*

Anhang 4: Konsistenzeinstellungen Cassandra[103]

Consistency-Level	Erklärung
ZERO	Für Schreiboperationen wird nichts garantiert
ANY	Schreiboperationen müssen auf mindestens einem Knoten geschrieben worden sein
ONE	Mindestens ein Knoten muss den Commit Log in die RAM-Tabellen geschrieben haben, bevor der Client eine Bestätigung erhält. Beim Lesen wird das erste verfügbare Ergebnis geliefert. Nachfolgende Lesezugriffe erhalten garantiert ein richtiges Ergebnis
QUORUM	Es muss eine bestimmte Anzahl von Servern (ReplikationsFaktor/2 +1) geantwortet haben, bevor Schreibbefehle getätigt werden. Beim Lesen wird der Eintrag mit dem aktuellsten TimeStamp zurückgegeben.
ALL	Alle Replikationsknoten müssen positiv geantwortet haben, bevor die Client-Operation erfolgreich abgeschlossen werden können.

[102] Boeker, M. (2010), S. 52 ff.
[103] Edlich, S. u. a. (2010), S. 77

Anhang 5: Code für die Erstellung einer Graphendatenbank in einer Javaanwendung

```
        import org.neo4j.graphdb.Direction;
        import org.neo4j.graphdb.GraphDatabaseService;
        import org.neo4j.graphdb.Node;
        import org.neo4j.graphdb.Relationship;
        import org.neo4j.graphdb.RelationshipType;
        import org.neo4j.graphdb.Transaction;
        import org.neo4j.graphdb.factory.GraphDatabaseFactory;

public class HelloNeo4J {
        String myString;                                //Ausgabestring
        GraphDatabaseService graphDb;                   //Deklaration der DB
        Node myFirstNode;                               //Deklaration eines Knoten
        Node mySecondNode;                              //Deklaration eines zweiten Knotens
        Relationship myRelationship;                    //Deklaration einer Beziehung

//Beziehung vom Typ Enum
private static enum RelTypes implements RelationshipType {KNOWS}

public static void main( final String[] args ){        //main-Methode
        HelloNeo4J myNeoInstance = new HelloNeo4J();
        myNeoInstance.createDb();                       //creatDB() Methodenaufruf
        myNeoInstance.removeData();                     //removeData() Methodenaufruf
        myNeoInstance.shutDown();                       //shutdown() Methodenaufruf
            }

void createDb(){
        //Initialisierung der Graphendatenbank
        graphDb = new GraphDatabaseFactory().newEmbeddedDatabase("target/neo4j");

        //Schreiboperationen müssen in Transaktionen gekapselt werden
        Transaction tx = graphDb.beginTx();
        try{
           //Initialisierung des Knoten
           myFirstNode = graphDb.createNode();
           //Dem Knoten wird das key-value Paar Name/Sophie und Christin mitgegeben
           myFirstNode.setProperty("name", "Sophie und Christin ");
           mySecondNode = graphDb.createNode();
           mySecondNode.setProperty( "name", "Tana und Teresa" );

//Dem ersten Knoten wird eine Beziehung zum zweiten Knoten des Typ “KNOWS” hinzugefügt
myRelationship = myFirstNode.createRelationshipTo(mySecondNode,RelTypes.KNOWS);
myRelationship.setProperty("relationship-type", "kennen");

//Ausgabe
myString = (myFirstNode.getProperty("name").toString()) + " " + (myRelation-
ship.getProperty("relationship-type").toString())+ " " + (mySecondNode.getProperty("name").toString());
System.out.println(myString);
tx.success();
                }
          finally{
             tx.finish();
                }
           }
void removeData(){
          Transaction tx = graphDb.beginTx();
          try{

//Zuerst muss die Beziehung gelöscht werden, bevor die Knoten gelöscht werden dürfen
myFirstNode.getSingleRelationship( RelTypes.KNOWS, Direction.OUTGOING ).delete();
             System.out.println("Removing nodes...");
             myFirstNode.delete();
             mySecondNode.delete();
              tx.success();
                }
           finally{
              tx.finish();
                }
            }

void shutDown(){
             graphDb.shutdown();
             System.out.println("graphDB shut down.");   }}
```

Anhang 6: Installationsanleitung Voldemort[104]

Nachdem Java6 und Apache Ant installiert sind, muss zunächst der Pfad von Java über folgende Befehle angepasst werden:

```
$ export PATH=/usr/lib/jvm/java-6-sun-1.6.0.12/bin:$PATH
$ export LD_LIBRARY_PATH=/usr/lib/jvm/java-6-sun
1.6.0.12/lib:$LD_LIBRARY_PATH
```

Danach muss aus dem Verzeichnis von Voldemort aus folgender Befehl ausgeführt werden:

```
$ ant
```

Damit Voldemort als nächstes gestartet werden kann, müssen lediglich noch zwei Files von Voldemort angepasst werden. Dies ist zum Einen die Datei voldemort-server.sh. In ihr muss folgender Abschnitt

```
if [ -z $VOLD_OPTS ]; then
  VOLD_OPTS="-Xmx2G -server -Dcom.sun.management.jmxremote"
fi
```

in folgendes geändert werden:

```
if [ -z $VOLD_OPTS ]; then
  VOLD_OPTS="-Xmx256M -server -Dcom.sun.management.jmxremote"
fi
```

Zum Anderen muss nun noch die Properties-Datei server.properties angepasst werden. Folgendes Statement

```
bdb.cache.size=1G
```

muss zu folgendem Statement geändert werden:

```
bdb.cache.size=128M
```

Nun sollte die Datenbank korrekt installiert und konfiguriert sein und kann über folgende Befehle gestartet werden:

```
$ ./bin/voldemort-server.sh config/single_node_cluster 2>&1 | tee
/tmp/voldemort.log &
$ bin/voldemort-shell.sh test tcp://localhost:6666
```

Hiermit wurde direkt das Cluster „test“, welches in der Datei \config\single_node_cluster\config\cluster angelegt wurde, geöffnet. In dieses können nun verschiedene Sätze reingeschrieben oder ausgelesen und gelöscht werden.

[104] Vgl. WorldPress.com (2009)

Anhang 7: Installation von Cassandra[105]

Für die Installation von Cassandra (Apache Cassandra Version 1.2.0) wird JRE6 oder höher vorausgesetzt. Nach dem Entpacken der Zip-Datei müssen folgende Konfigurationsschritte beachtet werden:

- Ändern der Pfade von in der Datei Cassandra.YAML

```
# directories where Cassandra should store data on disk.
data_file_directories:
    - D:\apache-cassandra-1.2.0\data

# commit log
commitlog_directory: D:\apache-cassandra-1.2.0\commitlog
```

Note: Hierbei ist genau auf die Leerzeichen und den Spiegelstrich zu achten

- Setzen der Umgebungsvariablen:

  JAVA_HOME: *Zielpfad zum JRE-Verzeichnis, mindestens jre 1.6*
  - z. B. C:\Program Files\Java\jre6

  CASSANDRA_HOME*: Zielpfad zum Verzeichnis von Cassandra*
  - z. B. C:\Users\Christin\Cassandra\apache-cassandra-1.2.0

- Starten von Cassandra im Command Line Interface:
  - Starten des Servers: Cassandra.bat
  - Starten des Clients: Cassandra-CLI.bat

- Operationen auf der Datenbank über
  - Auswählen einer CF über: use <*Column Family Name*>;
  - CLI-Befehle: [default@<*Column Family Name*]>
  - CQL-Befehle: Cqlsh>

[105] Vgl. Quality Unit (o. J.)

Anhang 8: CQL- und CLI-Befehle aus dem Cassandra Prototypen

CQL:

```
INSERT INTO Messa-
ge(Key,'User_ID_From','User_Name_From','User_ID_To','User_Name_To','
Subject','Message') VALUES ('2013-01-10 19:20:34', 'so-
phie.lingelbach@web.de','Sophie','tana.brunner@sowieso.de', 'Tana',
'Projekt NoSQL', 'Hey Tana, fuer unsere Praesentation brauchen wir
noch ein Beispiel und eine Live-Demo. Das muessen wir uns noch
ueberlegen. LG, Sophie');

INSERT INTO Messa-
ge(Key,'User_ID_From','User_Name_From','User_ID_To','User_Name_To','
Subject','Message') VALUES ('2013-01-10
19:57:16','sophie.lingelbach@web.de','Sophie','tana.brunner@sowieso.
de','Tana', ' Re:Re: Projekt NoSQL', 'Ah, gut das gehen wir dann
Morgen durch. Schoenen Abend! :) ');

CREATE index Sent_From ON Message ('User_Name_From');

SELECT COUNT(*) FROM Message WHERE 'User_Name_From' = 'Sophie';

SELECT 'User_Name_From', 'User_Name_To', 'Subject', 'Message' FROM
Message WHERE 'User_Name_From' = 'Sophie';

UPDATE  Message SET 'User_Name_To' = 'Christin' WHERE
'User_Name_From' =Sophie;

DELETE FROM Message WHERE Key = '2013-01-10 19:57:16';
```

CLI:

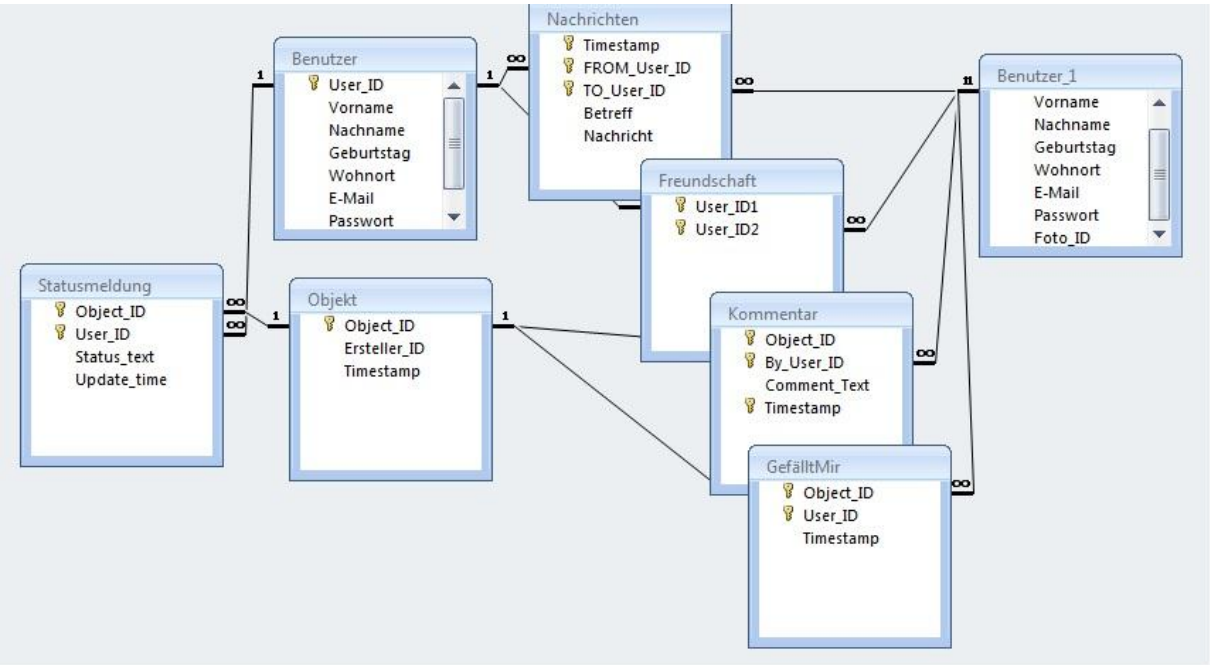
```
Message = {
           2013-01-10 19:57:16:
{From_Mail: {name: "From_Mail", value: "sophie.lingelbach@web.de"
timestamp: 125559999 },
{To_Mail: {name: "From_Mail", value: "tana.brunner@sowieso.de"
timestamp: 125559999 },
{From_Name: {name: "From_Name", value: "Sophie" timestamp: 125559999
},
{To_Name: {name: "From_Mail, value": "Tana" timestamp: 125559999 },
{Subject: {name: "Subject", value: "Projekt NoSQL" timestamp:
125559999 },
{Message: {name: "Message", value: "Re:Re: Projekt NoSQL', 'Ah, gut
das gehen wir dann Morgen durch. Schoenen Abend! :)", timestamp:
125559999 }}

GET Message['2013-01-10 19:20:34'];

DEL Message['2013-01-10 19:40:54'];
```

Anhang 9: RDBMS-Prototyp

Um die Vergleichbarkeit zwischen den Einsatzgebieten der NoSQL-Vertretern und einem traditionellen Datenbanksystem zu erhöhen, wird ebenfalls ein relationales Datenbankmodell abgebildet. Dieser Prototyp beschreibt ein ganzheitliches Modell eines sozialen Netzwerks.



Dieser Prototyp wurde mit MS Access umgesetzt. Das Modell ist in Tabellenstrukturen zeilenorientiert aufgebaut und unterliegt den Prinzipien der Normalisierung.

Bei relationalen Datenbanken müssen alle Datenstrukturen (ggf. Schema, Tabellen, Schlüsseln, Gültigkeitswerten) definiert (CREATE) werden, bevor sie mit Daten beladen werden können. Es werden Daten von 100 generierten Testusern in die Datenbank aufgenommen. Bei der Befüllung der Tabellen mit den Daten von 100 generierten Testusern, ist zu beachten, dass die Inhalte nicht gegen irgendwelche Beschränkungen oder Schlüsseldefinitionen verstoßen dürfen. Hierbei muss an die korrekte Reihenfolge beim Einladen der Datensätze gedacht werden. Das Ändern der Strukturen (ALTER) kann nur mit gewissem Aufwand und nicht zur Laufzeit werden erfolgen. Die Daten mit den Befehlen INSERT-, UPDATE- und DELETE geladen bzw. manipuliert. Zur Abfrage der Daten wird der Befehl SELECT genutzt. Dieser ermöglicht eine große Anzahl an Möglichkeiten Tabellen zu Verknüpfen, Werte abzufragen und zu aggregieren.

Die Daten werden zeilenorientiert abgelegt. Hierbei wird eine Trennung der eigentlichen Daten und der Metadaten vorgenommen.

Quellenverzeichnisse

Literaturverzeichnis

Alvermann, M. (2011): Einführung in MongoDB, in: JavaSPEKTRUM, 2011, Nr. 1

Boeker, M. (2010): MongoDB, Sag ja zu NoSQL, Frankfurt am Main: entwickler.press

Edlich, S. u. a. (2010): NoSQL Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken, München: Carl Hanser Verlag

Pürner, H. A. (2011): NoSQL-Die neue (alte) Datenbank-Generation?, in: Computerwoche, 2011, Nr. 38

Sakr, S./Pardede, E. (2011): Graph Data Management: Techniques and Applications, o. O: Idea Group Reference

Wolff, E. (2012): NoSQL-Die neue Datenbankgeneration?, in: Computerwoche, 2012, Nr. 37

Wyllie, D. (2011): MySQL vs. MongoDB: Datenbanksysteme im Vergleich, in: Computerwoche, 2011, Nr. 43

Verzeichnis der Internet- und Intranetquellen

Beany (2011):	NoSQL Systeme, http://blog.beany.co.kr/archives/275, Abruf : 06.12.2012
Computerwoche (2011):	Datenflut bereitet NoSQL den Weg, http://www.computerwoche.de/a/datenflut-bereitet-nosql-den-weg,1232715, Abruf: 17.12.2012
Computerwoche (2012):	Big Data - die Datenflut steigt, http://www.computerwoche.de/a/big-data-die-datenflut-steigt,2500037,3, Abruf: 27.12.2012
Datastax (2012a):	Getting Started with Apache Cassandra on Windows the Easy Way, http://www.datastax.com/resources/articles/getting-started-with-cassandra-on-windows, Abruf: 04.01.2013
Datastax (2012b):	Cassandra Client APIs, http://www.datastax.com/docs/1.1/dml/about_clients, Abruf: 15.01.2013
Datenbanken Online Lexikon (2011a):	Datenbanken Online-Lexikon - Spaltenorientierte Datenbanken, http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/SpaltenorientierteDatenbank, Abruf: 28.11.2012
Datenbanken Online Lexikon (2011b):	Datenbanken Online-Lexikon - Cassandra, http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/Cassandra, Abruf 06.12.2012

Datenbanken Online Lexikon (2011c): Consistent-Hashing, http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/Consistent-Hashing, Abruf: 17.12.2012

Datenbanken Online Lexikon (2012): Dokumentenorientierte Datenbanken, http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/DokumentenorientierteDatenbank, Abruf: 28.11.2012

Eifrem, E. (2009): Neo4j - The Benefits of Graph Databases, http://www.oscon.com/oscon2009/public/schedule/detail/8364, Abruf: 29.11.2012

freies Magazin (2010): NoSQL - Jenseits der relationalen Datenbank, http://www.freiesmagazin.de/ftp/2010/freiesMagazin-2010-08.pdf, Abruf: 17.12.2012

Grunev, M. (2010): A Quick Introduction t the Cassandra Data Model, http://maxgrinev.com/2010/07/09/a-quick-introduction-to-the-cassandra-data-model/, Abruf: 10.12.2012

Ivarsson, T. (2010): Persistent Graphs in Python with Neo4j, http://de.slideshare.net/thobe/persistent-graphs-in-python-with-neo4j, Abruf: 06.12.2012

Jansen, R. (2010): Neo4j – NoSQL-Datenbank mit graphentheoretischen Grundlagen, http://www.heise.de/developer/artikel/Neo4j-NoSQL-Datenbank-mit-graphentheoretischen-Grundlagen-1152559.html?artikelseite=2, Abruf 17.12.2012

Kwangsub (2012): CAP Theorem, http://kwangsub.blogspot.de/2012/02/cap-theorem.html, Abruf: 06.12.1012

Neo Technology Inc. (2012): Preisliste, http://www.neotechnology.com/price-list/, Abruf: 06.12.2012

Neo4j (2012a): Deployment Szenarien http://docs.neo4j.org/chunked/milestone/deployment-scenarios.html Abruf: 09.01.2013

Neo4j (2012b): Neo4j in Java einbinden http://docs.neo4j.org/chunked/milestone/tutorials-java-embedded-setup.html Abruf: 04.01.2013

Project Voldemort (o. J.): Design, http://www.project-voldemort.com/voldemort/design.html, Abruf: 05.12.2012

Quality Unit (o. J.): Cassandra Installation on Windows 7, http://support.qualityunit.com/249500-Cassandra-installation-on-Windows-7, Abruf 03.01.2013

Rodriguez, M. A. (2010): The Graph Traversal Programming Pattern, http://markorodriguez.com/lectures/, Abruf: 05.12.2012

Schmitt, K. (2008): SPARQL entblättert Datenpakete im Semantic Web, http://www.silicon.de/39161540/sparql-entblaettert-datenpakete-im-semantic-web/, Abruf: 17.12.2012

Schnelle, J. (2010): NoSQL - Jenseits der relationalen Datenbanken, http://www.pro-linux.de/artikel/2/1455/3,einleitung.html, Abruf: 28.11.2012

solid IT (Hrsg.) (2012): Vergleich MySQL und Neo4j http://db-engines.com/de/system/MySQL%3BNeo4j Abruf: 04.01.2013

sones GmbH (o. J.): sones GmbH Unternehmensbroschüre, http://www.sones.de/c/document_library/get_file?uuid=92e25daa-c8be-46c8-99cf-2603aca0cabb&groupId=11476, Abruf: 05.12.2012

10gen Inc. (2012a): MongoDB Quickstart Windows, http://www.mongodb.org/display/DOCSDE/Quickstart+Windows, Abruf: 09.01.2013

10gen Inc. (2012b): Admin Uls, http://www.mongodb.org/display/DOCS/Admin+Uls, Abruf: 17.01.2013

Walker-Morgan D. (2010): NoSQL im Überblick, http://www.heise.de/open/artikel/NoSQL-im-Ueberblick-1012483.html, Abruf: 29.11.2012

WorldPress.com (2009): Voldemort- Installation to First Run, http://cloudbuzz.wordpress.com/2009/10/13/voldemort-installation-to-first-run/, Abruf: 09.01.2013