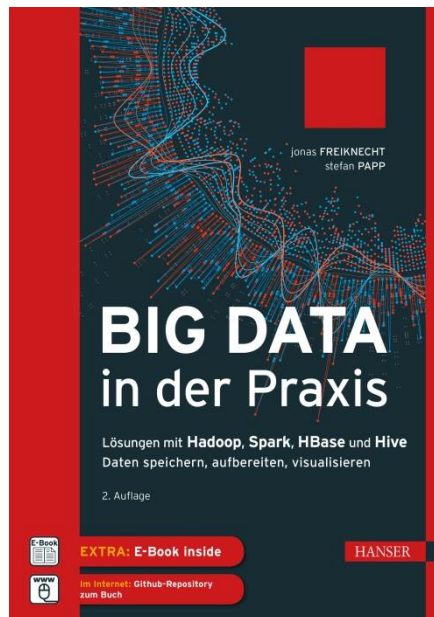


HANSER



Leseprobe

zu

Big Data in der Praxis

Jonas Freiknecht
Stefan Papp

ISBN (Buch): 978-3-446-45396-8

ISBN (E-Book): 978-3-446-45601-3

Weitere Informationen und Bestellungen unter

<http://www.hanser-fachbuch.de/>

sowie im Buchhandel

© Carl Hanser Verlag, München

Inhalt

1	Einleitung	1
2	Big Data	7
2.1	Historische Entstehung	9
2.2	Big Data – ein passender Begriff?	10
2.2.1	Die drei V	11
2.2.2	Weitere Vs	14
2.2.3	Der Verarbeitungsaufwand ist big	14
2.2.4	Sicht der Industrie auf Big Data	15
2.3	Eingliederung in BI und Data Mining	16
3	Hadoop	21
3.1	Hadoop kurz vorgestellt	21
3.2	HDFS – das Hadoop Distributed File System	23
3.3	Hadoop 2.x und YARN	28
3.4	Hadoop als Single-Node-Cluster aufsetzen	30
3.4.1	Falls etwas nicht funktioniert	44
3.5	Map Reduce	46
3.6	Aufsetzen einer Entwicklungsumgebung	49
3.7	Implementierung eines Map-Reduce-Jobs	56
3.8	Ausführen eines Jobs über Kommandozeile	68
3.9	Verarbeitung im Cluster	72
3.10	Aufsetzen eines Hadoop-Clusters	74
3.11	Starten eines Jobs via Hadoop-API	86
3.12	Verkettung von Map-Reduce-Jobs	99
3.13	Verarbeitung anderer Dateitypen	115
3.14	YARN-Anwendungen	130
3.14.1	Logging und Log-Aggregation in YARN	131
3.14.2	Eine einfache YARN-Anwendung	134
3.15	Vor- und Nachteile der verteilten Verarbeitung	159

3.16 Die Hadoop Java-API	160
3.16.1 Ein einfacher HDFS-Explorer	161
3.16.2 Cluster-Monitor	173
3.16.3 Überwachen der Anwendungen im Cluster	175
3.17 Gegenüberstellung zur traditionellen Verarbeitung	177
3.18 Big Data aufbereiten	178
3.18.1 Optimieren der Algorithmen zur Datenauswertung	178
3.18.2 Ausdünnung und Gruppierung	180
3.19 Ausblick auf Apache Spark	182
3.20 Markt der Big-Data-Lösungen	184
4 Das Hadoop-Ecosystem	187
4.1 Ambari	188
4.2 Sqoop	189
4.3 Flume	189
4.4 HBase	190
4.5 Hive	191
4.6 Pig	191
4.7 ZooKeeper	191
4.8 Oozie	192
4.9 Mahout	193
4.10 Data Analytics und das Reporting	193
5 NoSQL und HBase	195
5.1 Historische Entstehung	195
5.2 Das CAP-Theorem	196
5.3 ACID und BASE	197
5.4 Typen von Datenbanken	198
5.5 Umstieg von SQL und Dateisystemen auf NoSQL oder HDFS	201
5.5.1 Methoden der Datenmigration	201
5.6 HBase	203
5.6.1 Das Datenmodell von HBase	203
5.6.2 Aufbau von HBase	206
5.6.3 Installation als Stand-alone	207
5.6.4 Arbeiten mit der HBase Shell	209
5.6.5 Verteilte Installation auf dem HDFS	211
5.6.6 Laden von Daten	214
5.6.7 HBase Java-API	226
5.6.8 Der Umstieg von einem RDBMS auf HBase	249
6 Data Warehousing mit Hive	253
6.1 Installation von Hive	254

6.2	Architektur von Hive	256
6.3	Das Command Line Interface (CLI)	257
6.4	HiveQL als Abfragesprache	259
6.4.1	Anlegen von Datenbanken	259
6.4.2	Primitive Datentypen	260
6.4.3	Komplexe Datentypen	260
6.4.4	Anlegen von Tabellen	261
6.4.5	Partitionierung von Tabellen	262
6.4.6	Externe und interne Tabellen	262
6.4.7	Löschen und Leeren von Tabellen	263
6.4.8	Importieren von Daten	264
6.4.9	Zählen von Zeilen via count	265
6.4.10	Das SELECT-Statement	265
6.4.11	Beschränken von SELECT über DISTINCT	269
6.4.12	SELECT auf partitionierte Tabellen	269
6.4.13	SELECT sortieren mit SORT BY und ORDER BY	270
6.4.14	Partitionieren von Daten durch Bucketing	271
6.4.15	Gruppieren von Daten mittels GROUP BY	272
6.4.16	Subqueries – verschachtelte Abfragen	273
6.4.17	Ergebnismengen vereinigen mit UNION ALL	273
6.4.18	Mathematische Funktionen	274
6.4.19	String-Funktionen	276
6.4.20	Aggregatfunktionen	276
6.4.21	User-Defined Functions	277
6.4.22	HAVING	285
6.4.23	Datenstruktur im HDFS	286
6.4.24	Verändern von Tabellen	286
6.4.25	Erstellen von Views	289
6.4.26	Löschen einer View	289
6.4.27	Verändern einer View	289
6.4.28	Tabellen zusammenführen mit JOINS	290
6.5	Hive Security	292
6.5.1	Implementieren eines Authentication-Providers	298
6.5.2	Authentication-Provider für HiveServer2	303
6.5.3	Verwenden von PAM zur Benutzerauthentifizierung	303
6.6	Hive und JDBC	304
6.7	Datenimport mit Sqoop	322
6.8	Datenexport mit Sqoop	324
6.9	Hive und Impala	325
6.10	Unterschied zu Pig	326
6.11	Zusammenfassung	327

7	Big-Data-Visualisierung	329
7.1	Theorie der Datenvisualisierung	329
7.2	Diagrammauswahl gemäß Datenstruktur	335
7.3	Visualisieren von Big Data erfordert ein Umdenken	336
7.3.1	Aufmerksamkeit lenken	337
7.3.2	Kontextsensitive Diagramme	339
7.3.3	3D-Diagramme	341
7.3.4	Ansätze, um Big-Data zu visualisieren	342
7.4	Neue Diagrammarten	344
7.5	Werkzeuge zur Datenvisualisierung	348
7.6	Entwicklung einer einfachen Visualisierungskomponente	352
8	Auf dem Weg zu neuem Wissen – Aufbereiten, Anreichern und Empfehlen	365
8.1	Eine Big-Data-Table als zentrale Datenstruktur	368
8.2	Anreichern von Daten	370
8.2.1	Anlegen einer Wissensdatenbank	371
8.2.2	Passende Zuordnung von Daten	372
8.3	Diagrammmempfehlungen über Datentypanalyse	376
8.3.1	Diagrammmempfehlungen in der BDTTable	378
8.4	Textanalyse – Verarbeitung unstrukturierter Daten	384
8.4.1	Erkennung von Sprachen	385
8.4.2	Natural Language Processing	386
8.4.3	Mustererkennung mit Apache UIMA	394
9	Infrastruktur	415
9.1	Hardware	416
9.2	Betriebssystem	417
9.2.1	Paketmanager	417
9.2.2	Git	418
9.2.3	VIM	419
9.2.4	Terminalumgebung	419
9.3	Virtualisierung	420
9.4	Container	420
9.4.1	Docker-Crashkurs	421
9.4.2	Infrastructure as Code	424
9.5	Distributionen	424
9.6	Reproduzierbarkeit	425
9.7	Zusammenfassung	425
10	Programmiersprachen	427
10.1	Merkmale	428
10.1.1	Funktionale Paradigmen	428

10.2	Big-Data-Programmiersprachen	429
10.2.1	Java	429
10.2.2	Scala	430
10.2.3	Python	433
10.2.4	R	436
10.2.5	Weitere Programmiersprachen	437
10.3	Zusammenfassung	438
11	Polyglot Persistence	439
11.1	Praxis	440
11.1.1	Redis	440
11.1.2	MongoDB	443
11.1.3	Neo4j	443
11.1.4	S3	444
11.1.5	Apache Kudu	447
11.2	Zusammenfassung	447
12	Apache Kafka	449
12.1	Der Kern	450
12.2	Erste Schritte	450
12.3	Dockerfile	454
12.4	Clients	454
12.5	Python Chat Client	454
12.6	Zusammenfassung	456
13	Data Processing Engines	457
13.1	Von Map Reduce zu GPPEs	457
13.1.1	Herausforderungen	458
13.1.2	Verfahren zur Verbesserung	459
13.1.3	Von Batch und Streaming zu Lambda	461
13.1.4	Frameworks in a Nutshell	462
13.2	Apache Spark	462
13.2.1	Datasets	462
13.2.2	Von RDDs zu Data Frames	463
13.2.3	Hands On Apache Spark	463
13.2.4	Client-Programme schreiben	465
13.2.5	Das Spark-Ecosystem	470
13.3	Zusammenfassung	474
14	Streaming	475
14.1	Kernparadigmen	475
14.2	Spark Streaming	478
14.2.1	Beispiel	479

14.3	Apache Flink	480
14.4	Zusammenfassung	483
15	Data Governance	485
15.1	Begriffsdschungel	486
15.2	Governance-Pfeiler	487
15.2.1	Transparenz	487
15.2.2	Verantwortung	488
15.2.3	Standardisierung	489
15.3	Fokusthemen von Data Governance	489
15.3.1	Policies	489
15.3.2	Quality	490
15.3.3	Compliance	490
15.3.4	Business Intelligence	490
15.4	Datenschutz	491
15.4.1	Werkzeuge	492
15.5	Sicherheit im Hadoop-Ecosystem	497
15.6	Metadatenmanagement	498
15.6.1	Open-Source-Werkzeuge	499
15.6.2	Kommerzielle Datenkataloge	500
15.7	Organisatorische Themen	500
15.7.1	Privacy by Design	501
15.7.2	k-Anonymity	501
15.7.3	Standards	503
15.8	Zusammenfassung	503
16	Zusammenfassung und Ausblick	505
16.1	Zur zweiten Auflage 2018	505
16.2	Zur ersten Auflage 2014	507
17	Häufige Fehler	511
18	Anleitungen	517
18.1	Installation und Verwendung von Sqoop2	517
18.2	Hadoop für Windows 7 kompilieren	523
19	Literaturverzeichnis	527
Index	531

■ 2.3 Eingliederung in BI und Data Mining

Um die Begriffe BI und Data Mining in Relation zu Big Data setzen zu können, gilt es, diese im Vorfeld zu definieren. Kemper, Mehanna & Unger bezeichnen BI als Filter, der Daten in strukturierte Information umwandle (Kemper, et al., 2010). Gartner hingegen konstatiert etwas ausführlicher, dass BI ein Überbegriff für Anwendungen, Infrastruktur, Werkzeuge und Best Practices sei, die den Zugriff auf und die Analyse von Informationen ermöglichen, um Entscheidungsfindung und Performance zu erhöhen (Gartner). Hält man sich nun strikt an die Definitionen, besteht der Unterschied zwischen BI und Big Data darin, dass BI sich auf bereits vorliegende Informationen bezieht, die dazu noch strukturiert sind und sich auf einen eindeutigen Kontext beziehen.

Das Ziel von BI und der Big-Data-Explorationen ist jedoch dasselbe, nämlich aus vorhandenen Daten neue Erkenntnisse zu gewinnen, die der Entscheidungsfindung bei vorher definierten Fragestellungen dienen. Der Trend bei Big Data geht dabei auch oft in die Richtung zu lernen, was man noch nicht weiß.

BI ist jedoch mittlerweile mehr als diese einfache Begriffsdefinition. Es hat sich in den letzten Jahren zu einem festen Prozess samt einem Set aus technischen Werkzeugen entwickelt, um das Berichtswesen in Unternehmen zu automatisieren. Dazu gehören die Datenaufbereitung, die Datenspeicherung in DWHs sowie deren Darstellung aus verschiedenen Perspektiven.

Welche Techniken, Methoden und Arbeitsschritte werden aber nun angewandt, um Informationen aus vorliegenden Daten zu extrahieren? Die Antwort darauf gibt der sogenannte KDD-Prozess (*Knowledge Discovery in Databases*).

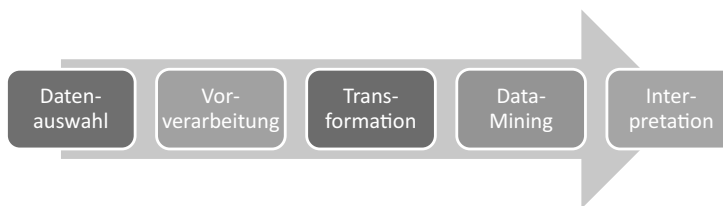


Bild 2.5 Der KDD-Prozess nach (Kononenko, et al., 2007)

Der (iterative und interaktive) KDD-Prozess hat das Ziel, gültige, neue, nützliche und verständliche Muster in vorhandenen Daten zu erkennen (Fayyad, et al., 1996). Wirft man nun einen Blick auf den vierten Schritt des in Bild 2.5 illustrierten Ablaufs, so ist zu erkennen, dass Data Mining einen Teil des KDD-Prozesses darstellt. Dieser nimmt gesäuberte, korrigierte und transformierte Daten entgegen, extrahiert daraus Muster und Zusammenhänge und gibt diese zur Interpretation frei. Quellen müssen, anders als der Begriff KDD vermuten lässt, nicht zwingend Datenbanken sein, sondern können auch als simple Datensätze gesehen werden, z.B. als Flat Files, CSV, XML oder Dateisystemstrukturen. Wichtig ist, dass diese bereits im Vorfeld aufbereitet wurden. Zu dieser Aufbereitung (*Preprocessing*) gehören:

- Formatanpassungen (z. B. Datums- und Zahlenformate)
- Korrigieren von Ausreißern (Messfehler, Verarbeitungsfehler, bewusste Falschangabe)
- Auffüllen dünn besetzter Daten



HINWEIS: In späteren Kapiteln wird der KDD-Prozess im Detail durchgegangen.

Stellt man nun die drei Begriffserklärungen BI, Data Mining und Big Data einander gegenüber, so erkennt man schnell einige Gemeinsamkeiten sowie Unterschiede.

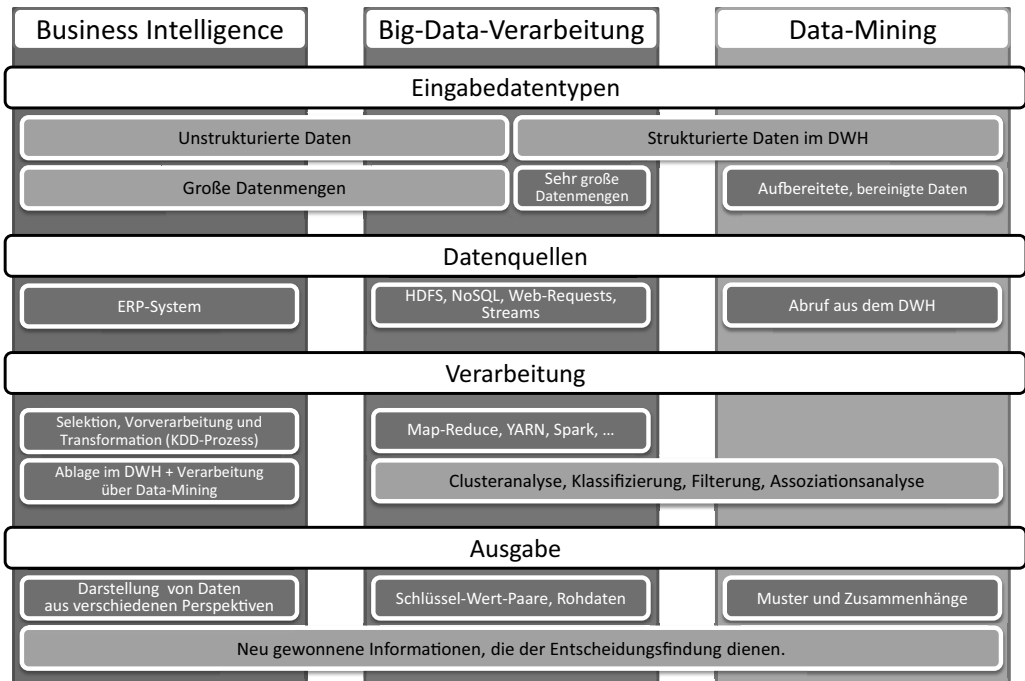


Bild 2.6 Definitionsvergleich von BI, Big Data und Data-Mining

Vor Big Data wurden größere Analysen fast ausschließlich in Datenbanken ausgeführt. Es gab dazu auch eine Königsklasse, das MPP (Massive Parallel Processing), das eine parallele Verarbeitung zuließ. Im Kern folgte aber jedes Data Warehouse den Prinzipien der relationalen Datenbanken.

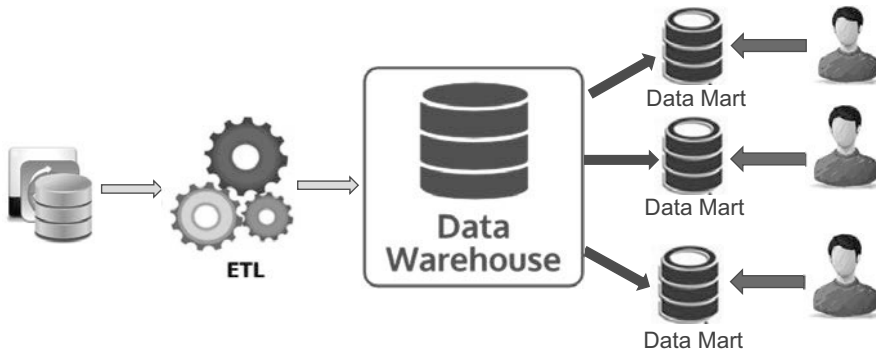


Bild 2.7 Das Data Warehouse

Das obige Beispiel illustriert den Prozess in den DWHs. Daten werden aus ihren Quellen entnommen, in sogenannten ETL-Tools aufbereitet und als relationale, strukturierte Daten im DWH abgelegt. Vom DWH werden Extrakte in Data Marts abgelegt, auf die dann Analysten zugreifen, um Ergebnisse abzuleiten.

In der ersten Evolutionsstufe von Hadoop empfahlen manche DWH-Systemhersteller, Hadoop als billigen Speicherplatz zu nehmen. Hadoop würde dann die Rolle von ETL übernehmen, die Daten aufbereiten und ins DWH laden. Die Auswertung erfolgte dann im DWH. Für Spezialfälle boten die Systemhersteller auch Möglichkeiten an, sogenannte Kaltdaten aus Kostengründen auf Hadoop zu belassen und sie bei Bedarf ins DWH zu laden.

Der Vorteil vom DWH wurde so angepriesen, dass bekannte analytische Verarbeitungsmethoden (Stichwort *Online Analytical Processing Cube*) und eine reichhaltige Funktionsbibliothek zur Verfügung stehen. Auch steckte jahrelanges Know-how in Komponenten wie der SQL Engine des DWHs. SQL auf Hadoop war zumindest am Anfang extrem langsam und hatte bei Joins eine schlechte Performance.

Auch die Rolle von sogenannten Self-Service BI Tools (wie Tableau) und vor allem Excel darf nicht unterschätzt werden. Soll also in einer Grafik der Gewinn eines Unternehmens inklusive aller Tochterfirmen angezeigt werden, müssen im Reporting-Werkzeug lediglich vom Benutzer die bereits ermittelten Gewinne der Tochterfirmen dem Gesamtgewinn hinzuzugediert werden. Um das zu bewerkstelligen, gibt es Konnektoren zu verschiedenen Datenlieferanten.

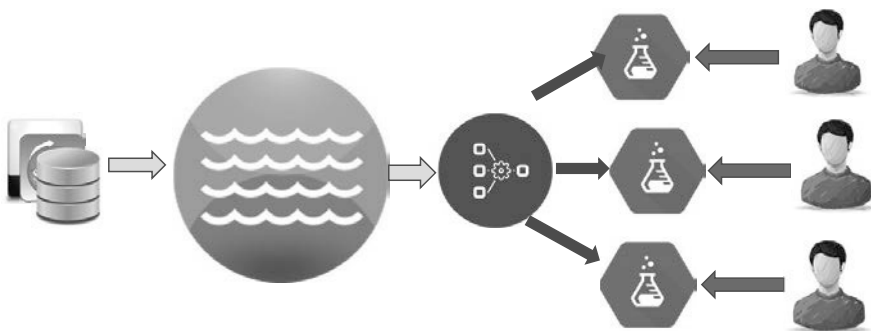


Bild 2.8 Der Data Lake

Mit Hadoop hat sich vieles geändert. Daten werden sofort auf den Hadoop-Cluster geladen. Wir sprechen hier auch vom Data Lake. Man spricht von ETL (Extract Transform Load), wenn die Daten vor der Beladung vom DWH noch transformiert wurden, bevor sie abgelegt werden konnten,

Im Falle des Data Lakes spricht man aber von ELT (Extract Load Transform). Daten werden aus den Quellsystemen gezogen, auf den Data Lake geladen und schließlich in ein Zielsystem transformiert.

Bei der Aufbereitung von Daten werden die unstrukturierten Daten oft in ein strukturiertes Schema überführt. Daten, die in Apache Hive-Tabellen abgelegt werden, können auch als Datenlieferanten von Self-Service BI Tools wie Tableau dienen.

Mit der Zeit wurde Hadoop zu einer Datenplattform, die auf einem verteilten Dateisystem aufbaut und zahlreiche Möglichkeiten bietet, die Daten aufzubereiten. Diese Daten können in Strukturen gebracht und gleichzeitig analysiert werden.

Die Reise war damit aber noch nicht zu Ende. Das komplette Open-Source-Ecosystem von Big Data bietet mittlerweile zahlreiche Werkzeuge an, die die Verarbeitung von Daten noch weiter vereinfachen und neue Methoden wie die Echtzeitverarbeitung mit sich bringen.

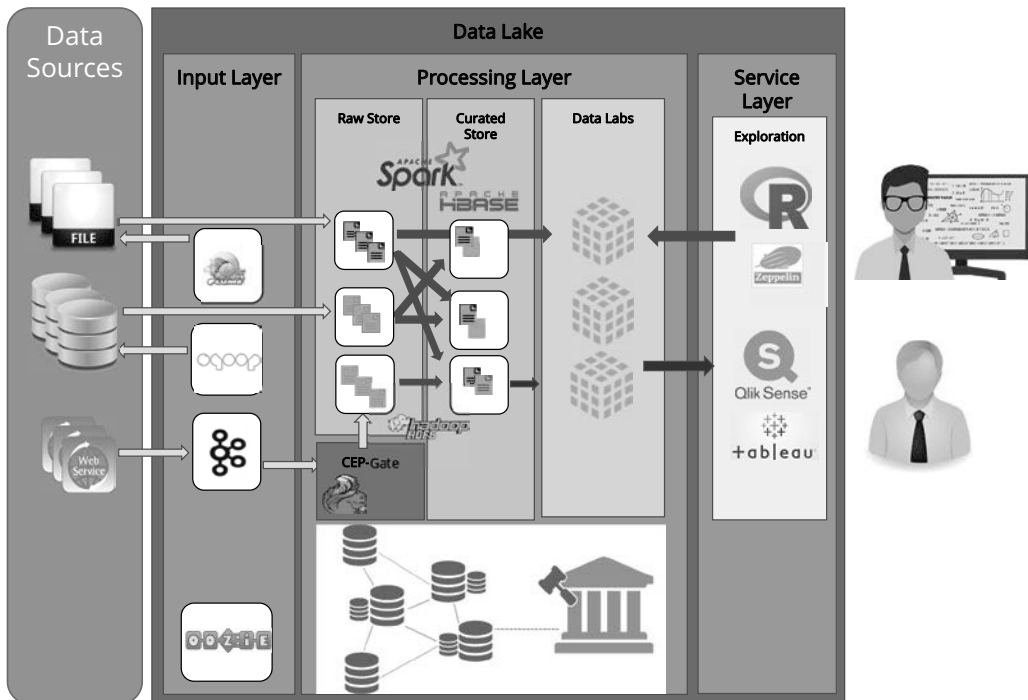


Bild 2.9 Big-Data-Referenzarchitektur

Bild 2.9 zeigt eine komplette Referenzarchitektur mit Big-Data-Komponenten. Es landen unterschiedliche Daten auf einer Datenplattform (in diesem Beispiel Hadoop). Sind die Daten auf Hadoop geladen, werden sie für verschiedene Szenarien unterschiedlich weiterverarbeitet und dann in Tabellen abgelegt, auf die dann analytische Werkzeuge zugreifen können.

Architekturen wie die Lambda- und Kappa-Architektur zeigen, dass man auch Architekturmuster bauen kann, die auf Echtzeitverarbeitung aufbauen.

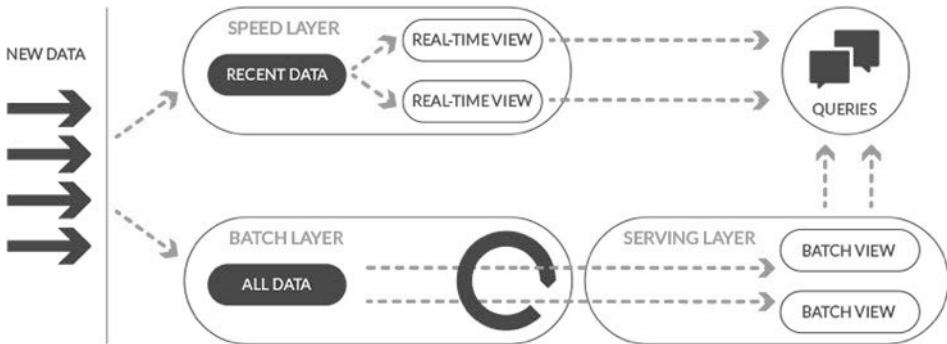


Bild 2.10 Big-Data-Architekturen

Zusammengefasst kann man sagen, dass in in Lambda-Architekturen Daten beim Laden in zwei Bereiche geteilt werden. Im Speed Layer werden die Daten im Regelfall in Memory gehalten. Im Batch Layer werden alle Daten archiviert. Verschiedene Prozesse verarbeiten die Daten kontinuierlich und bereiten die Ergebnisse im Serving Layer für einen schnellen Zugriff auf. Details können im Buch von Nathan Marz nachgelesen werden (Marz, 2015).

Ein weiteres zentrales Thema, das im Umfeld von Big Data anzutreffen ist, ist Data Governance. Dieser Begriff umfasst die Verwaltung von Daten, konkret geht es um Katalogisierung, Klassifikation und weitere Prozesse wie z.B. Retention, um Daten nach einem bestimmten Ablauf auch wieder zu löschen. In Data Governance spielen auch Themen wie Sicherheit und Datenqualität hinein.

Ein weiteres Thema sind natürlich alle Methoden, um Daten zu analysieren. Nicht umsonst hat der Beruf des Data Scientist einen hohen Stellenwert bekommen. Auch gibt es mittlerweile zahlreiche Kurse zu Deep Learning und Machine Learning.

Als dieses Buch in der ersten Version geschrieben wurde, wurden fast ausschließlich Hadoop-Technologien beschrieben. In einem Interview erklärte Ted Dunning, Chief Architect von MapR, dass Open-Source-Technologien deswegen so erfolgreich sind, weil sie über saubere Schnittstellen verfügen. Einzelne Komponenten wären somit leichter austauschbar.

Die Konsequenz daraus ist, dass viele Open-Source-Big-Data-Systemlandschaften wie ein Baukasten aufgebaut sind, wo einzelne Komponenten austauschbar sind. Das trifft auch auf Hadoop zu. MapR hat beispielsweise ein eigenes Dateisystem geschrieben, das auf der HDFS-API aufbaut und mehr Funktionalität aufweist als HDFS. Neue Komponenten wie Apache Spark nutzen nur mehr Schnittstellen und können Daten auf HDFS schreiben, aber auch auf viele andere Plattformen wie Cassandra oder Amazon S3. Die zentrale Botschaft ist, dass Hadoop in diesem Buch zwar nach wie vor einen hohen Stellenwert hat, aber dieses Buch kein reines Hadoop-Buch ist, sondern viele Technologien vorstellt, die man mit Hadoop verwenden kann, die aber im eigentlichen Sinne von Hadoop unabhängig sind.

- *Geografische Daten* werden oft als Karte, z. B. in Bild 7.2 als *Choroplethenkarte*, dargestellt, um bestimmte Ausprägungen eines Merkmals auf ein bestimmtes Areal abzubilden oder die Aufmerksamkeit des Betrachters durch Marker auf bestimmte Koordinaten, Städte, Flüsse oder Strecken zu lenken.

Kurniawan behält sich vor, dass es noch andere Datentypen gibt, die in keines dieser Schemata passen. Hier wäre beispielsweise der Datensatz anzuführen, der einer *Tag Cloud*, dem Diagramm der *Web-2.0-Ära*, zugrunde liegt. Dabei handelt es sich um eine Liste nicht zueinander in Beziehung stehender Daten, die letztendlich in einer zusammengesetzten, ineinander verwinkelten Textabbildung zusammengeführt werden.

■ 7.3 Visualisieren von Big Data erfordert ein Umdenken

Was wir bisher bezüglich der Datenvisualisierung besprochen haben, ist nicht neu. Die Datentypen, die dargestellt werden, waren auch schon vor Auftreten der Big-Data-Thematik bekannt, und die Tatsache, dass Diagramme unserem Verstand schneller zugänglich sind als bloße Daten, ist auch keine bahnbrechende Erkenntnis. Wo also liegen die Besonderheiten und die Herausforderung bei der Darstellung von Big Data? Klar ist, dass der Datenumfang erst einmal eingeschränkt werden muss. Stellen Sie sich vor, dass Sie Millionenstädte auf einer Landkarte anzeigen wollen und zu jeder Stadt im Sichtfeld Informationen anbieten möchten, z. B. Einwohnerzahl, Gründungsjahr oder das Durchschnittseinkommen. Insbesondere sollen des Weiteren die Städte hervorgehoben werden, die über drei Millionen Einwohner aufweisen. Aus diesem einfachen Szenario ergeben sich im Grunde zwei grob formulierte Anforderungen an die Big-Data-Visualisierung. Diese lauten:

- *Wie können wir viele Daten auf einmal darstellen?*
- *Wie können wir in großen Datenmengen die Aufmerksamkeit auf einen Datensatz lenken?*

Bereits in Abschnitt 3.18.2 wurde erwähnt, dass das Ausdünnen der Daten eine elementare Rolle bei der Aufbereitung von Big Data spielen kann. Dieser Schritt kommt bei der Datenvisualisierung besonders zum Tragen, bedenkt man, dass Diagramme häufig auf Client-Seite via JavaScript oder Flash gerendert werden und die Daten dazu von unserem Server auf den Rechner des Betrachters transportiert werden. Hier müssen wir aus Zeit- und Performancegründen die Datenmenge auf ein akzeptables Minimum reduzieren, um die Ladezeiten kurz und die *Usability* der Anwendung hoch zu halten.

Eine zweite Herausforderung, die sich aus der ersten Frage ergibt, ist die des Platzes, der uns für die Darstellung der Daten zu Verfügung steht. Wie können wir diesen effizient nutzen, sodass der Betrachter trotz der erhöhten Anzahl der Datensätze trotzdem die Aussage des Diagramms erkennt, es thematisch einordnen kann und das Diagramm Interesse weckt und nicht schon im Vorfeld zu komplex wirkt? Und wie heben wir bestimmte Areale hervor, die für den Betrachter besonders interessant sein könnten? Lassen Sie uns die folgenden Abschnitte als möglichen Lösungsansatz für die angesprochenen Punkte betrachten.

7.3.1 Aufmerksamkeit lenken

Jenifer Tidwell (Tidwell, 2011) stellt die Behauptung auf, dass visuelle Werkzeuge bereits Informationen liefern, ohne dass der Betrachter ihnen überhaupt aktiv Beachtung schenkt. Ist diese Behauptung auch noch zu vertreten, wenn Big Data dargestellt wird? Schließlich müssen hier ja viel mehr Daten auf demselben Platz untergebracht werden, und da liegt die Vermutung nahe, dass diese intuitive Informationsaufnahme vielleicht nicht mehr ganz so selbstverständlich erfolgt. Wie also lassen sich wichtige Informationen in Diagrammen auch bei großem Umfang weiterhin hervorheben, sodass diese auf einen Blick ersichtlich sind? Und wie wird vermieden, dass das Hindeuten auf diese Merkmale nicht abermals zu viel Platz auf der Abbildungsfläche in Anspruch nimmt? Zwar ließen sich leicht große Pfeile auf interessante Punkte eines Diagramms zeichnen und beschriften, jedoch ginge dadurch ein wesentlicher Teil der Grafik verloren, der zur Darstellung weiterer Datensätze genutzt werden könnte. Tidwell führt für diesen Fall folgende Möglichkeiten (Bild 7.5) an, um einzelne Bereiche hervorzuheben. Dabei verzichtet sie weitestgehend auf Beschriftungen oder weitere Elemente auf der Grafik, sondern modifiziert die jeweiligen dargestellten Daten nur leicht in Form, Position und Farbe.



Bild 7.5 Werkzeuge, um Aufmerksamkeit gezielt zu lenken (Tidwell, 2011)

Zu Beginn werden drei Möglichkeiten angeführt, die die Farbe des hervorzuhebenden Datums ändern, genauer: Farbton, Helligkeit und Sättigung. Gewiss könnte man die drei zu einem einzelnen Punkt zusammenfassen, jedoch ergeben sie einzeln genommen verschiedene Möglichkeiten der Darstellung, die bei einer einfachen Änderung des RGB-Wertes (Rot, Grün, Blau) verwehrt blieben. Abermals soll die Choroplethenkarte aus Bild 7.2 als Beispiel dienen, auf der verschiedene Niveaus durch die Helligkeit der Farbe dargestellt werden. Anders als bei einer eigens erdachten Farbskala (z. B. einer *Heat Map*, die Werte von Grün über Gelb nach Rot darstellt), kann so ein klarer Verlauf gezeigt werden, der beispielsweise von 0 bis 100 reicht und keine sichtbaren Schwellenwerte aufweist. Bei der als Beispiel angeführten *Heat Map* ruft jeder Farbwechsel beim Betrachter eine sofortige Assoziation mit einer Zustandsänderung hervor. Die Farbe Grün wird sehr positiv wahrgenommen, Gelb gilt als kritisch und Rot als negativ. Eine Karte, die mit einer einzigen Farbe arbeitet und lediglich deren Helligkeit verändert, führt nicht zu derartigen Interpretationen.

Bezüglich des Farbschemas eines Diagramms gilt es, eine gewisse Konsistenz zu wahren und Farben zu wählen, die dem Betrachter gegenüber angenehm erscheinen. Solche Schemata lassen sich leicht über eigens dafür erstellte Web-Dienste² erstellen. Neben der Farbgebung spielt ebenso die Texturierung eine Rolle, die durch ein Muster Datensätze hervorheben kann.

² <http://colorshemesdesigner.com/>

Die form- oder positionsändernden Merkmale wie Positionierung, Orientierung, Größe oder Form sind mit Vorsicht zu genießen, da diese voraussetzen, dass alle anderen Daten geordnet und gleich verteilt zueinander ausgerichtet sind und über ein einheitliches Erscheinungsbild verfügen. Weniger aussagekräftig wäre diese Darstellung etwa auf einer Landkarte, da die Grenzen dem Betrachter zwar geläufig sein können, die Umrisse aber so stark variieren, dass bei geringen Änderungen der Form eines bestimmten Areals nicht wahrgenommen werden kann, ob sich dieses nun verändert hat oder nicht.



Bild 7.6 Geburten werden über Formveränderungen auf einer Karte der USA dargestellt (Allen).

So zeigt Bild 7.6 eine Karte, die die Geburtenrate in den USA visualisieren soll. Auch mit Kenntnissen der Topografie von Nordamerika ist es schwer, Aussagen darüber zu treffen, in welchen Staaten die Rate nun besonders hoch ist. Kalifornien, New York, Florida und Texas scheinen etwas ausgeprägter zu sein als die anderen, jedoch ist eine klare Aussage, ohne Beachtung der Farbhelligkeit, nicht zu treffen. Nichtsdestotrotz hat Shawn Allen mit dieser Karte eine fantastische Visualisierungsmethode geschaffen, die ihre ganze Aussagekraft erst entfaltet, wenn der Benutzer aktiv mit dem Diagramm interagiert. Dabei kann die Karte in einer Animation in ihren Normalzustand transformiert werden, wodurch die Ausprägung des Merkmals *Geburtenrate* erst sichtbar wird. Besonders eindrucksvoll ist weiterhin die Möglichkeit, zwischen verschiedenen Merkmalen zu wechseln (Bild 7.7), z. B. mit der Migrationsrate oder der Bevölkerungsdichte. Eine Karte visualisiert hier also durch Nutzung von interaktiver Veränderung von Form, Farbe und Größe 16 Merkmale auf engstem Raum. Das Stichwort Interaktivität soll nun direkt in den nächsten Abschnitt überleiten.

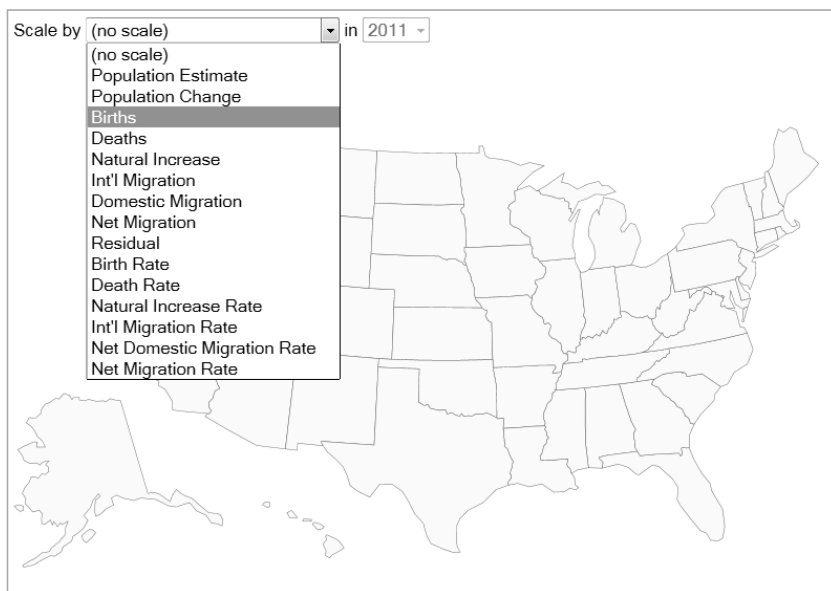


Bild 7.7 Nordamerika ohne farbliche oder strukturelle Veränderung (Allen)

So etwa wird Alaska unten links in Bild 7.6 durch eine geringe Geburtenrate stark komprimiert. Ebenso ergeht es Montana und North Dakota im Norden. Die Karte, die Allen hier entworfen hat, zeigt ihre ganze Aussagekraft erst dadurch, dass der Benutzer verschiedene Kennzahlen aus einer Drop-down-Box auswählt und sich die Abbildung dann sofort entsprechend verformt und einfärbt.

„A good visualization is not just a static picture or a three-dimensional (3D) virtual environment that we can walk through and inspect like a museum full of statues. A good visualization is something that allows us to drill down and find more data about anything that seems important.“ – Colin Ware (2012, S. 345)

Wenn nicht alle Kennzahlen auf eine Grafik passen und die Möglichkeit der Interaktion gegeben ist, dann kann dem Benutzer also gestattet werden, die Kennzahlen und somit die Grafik zu manipulieren. Dieses Feature bleibt Print- und anderen statischen Medien verwehrt, bringt jedoch im Bereich der digitalen Visualisierung viele Vorteile mit sich, wie z. B. die exemplarisch gezeigte Änderung des der Visualisierung zugrunde liegenden Datensatzes.

7.3.2 Kontextsensitive Diagramme

Die grafische Darstellung großer Datenmengen benötigt Platz. Die Betrachtung in *Microsoft Excel* oder *PowerPoint* erstellter Diagramme zeigt, dass ein beachtlicher Teil für Beschriftungen, Legenden und Indizes verwendet wird. Häufig lassen sich diese jedoch vermeiden, ohne die Aussagekraft eines Diagramms zu reduzieren. Besonders durch den Einsatz von kontextsensitiven Diagrammen, die durch ihr bloßes Aussehen bereits eine Thematik kom-

munizieren, kann ein signifikanter Teil der Beschriftungen eingespart werden. Diese These soll nun durch ein passendes Szenario untermauert werden.

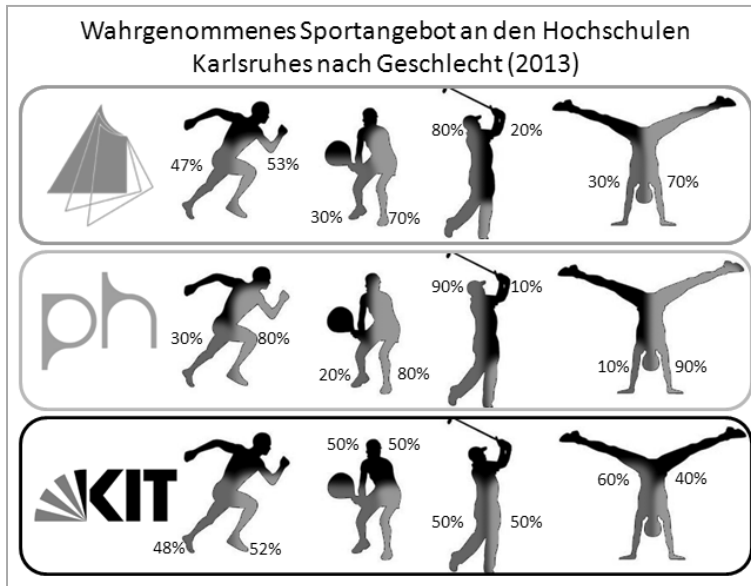


Bild 7.8 Eine Infografik mit bewusst reduzierter Beschriftung zeigt das Sportangebot der Hochschulen in Karlsruhe.

Bild 7.8 zeigt eine auf erdachten Daten erstellte Statistik über die wahrgenommenen Sportangebote an den verschiedenen Hochschulen in Karlsruhe. Dabei wird auf die namentliche/textuelle Nennung der Sportarten verzichtet. Diese wird stattdessen in Form von eingefärbten Silhouetten dargestellt. Ebenso werden die Namen der Hochschulen nicht ausgeschrieben, sondern durch die entsprechenden Logos ersetzt. Die Platzersparnis, die durch den Einsatz der Grafiken erreicht wird, hält sich hier sicher in Grenzen. Jedoch weiß der Betrachter sofort, dass die Daten im Diagramm thematisch dem Bereich Sport und Hochschulen zuzuordnen sind. Zu guter Letzt weckt eine grafische Repräsentation von Themen und Kontexten Aufmerksamkeit, und sie rufen weiterhin persönliche Assoziationen mit den eigenen Erfahrungen und Gefühlen des Betrachters hervor. Hat dieser etwa an der Hochschule Karlsruhe studiert, so entwickelt er direkt einen Bezug zu den präsentierten Werten. Die farbliche Kennzeichnung der Geschlechter könnte klassisch in den Farben Blau und Pink geschehen, sodass es keiner weiteren Erklärung der Farbgebung bedarf. Die prozentualen Angaben über die einzelnen Sportarten sind nicht unbedingt notwendig, um die Aussage der Grafik zu erschließen. So ist zum Beispiel bei bloßer Betrachtung der Silhouetten sofort zu sehen, dass Tennis weitestgehend von Frauen und Golf überwiegend von Männern gespielt wird. Wenn der Platz es jedoch zulässt und bestimmte Ausprägungen eines Merkmals teilweise schwer zu unterscheiden sind (z. B. Golf am KIT mit 50 % zu 50 %), dann kann eine Beschriftung einen vorteilhaften Effekt haben.

7.3.3 3D-Diagramme

Seit dem Erscheinen von *HTML5* und *WebGL (Web Graphics Library)* ist es in aktuellen Browsern möglich, aufwendige 3D-Grafiken zu rendern. So entstand beispielsweise die Bibliothek *PhiloGL* unter der Schirmherrschaft der Firma *Sencha*, die sich zwar nicht ausschließlich auf Datenvisualisierung beschränkt, aber dennoch einige interessante Ansätze zeigt, welche Art von Daten sich unter Zuhilfenahme einer dritten Dimension darstellen lassen. Wie so oft ist auch *Google* ganz vorne mit dabei, wenn es um moderne Web-Technologien geht, und stellt mit der für den hauseigenen Browser *Chrome* aufgesetzten Plattform *Chrome Experiments*³ einige interessante Ansätze dar, die zeigen, welche Daten wie in 3D im Browser dargestellt werden können. Eines dieser Beispiele wurde als Inspiration verwendet, um später im praktischen Teil geologische Daten auf einer Weltkarte zu visualisieren.

Der Vorteil einer 3D-Darstellung liegt auf der Hand: Dem Benutzer wird nicht nur eine weitere Dimension geboten, in der er Daten ablegen und betrachten kann, vielmehr ist er außerdem in der Lage, viel intuitiver durch die drei Dimensionen zu navigieren. Natürliche Vorwärts- und Rückwärtsbewegungen ersetzen dabei beispielsweise einen Drilldown, um tiefer in Daten hineinzuschauen. Und, nicht zu vergessen, 3D wirkt immer innovativ und fortschrittlich! Gehen Sie mal über die *CeBIT*: Bei genauerem Hinsehen werden Sie von dreidimensionalen Darstellungen förmlich erschlagen.



Bild 7.9 3D-Sonographie eines Säuglings (© Valentina R. – Fotolia.com)

Eine praktische Anwendung findet die 3D-Darstellung bereits im medizinischen Sektor in den klassischen bildgebenden Verfahren wie etwa der *CT* (Computertomographie) oder der *MRT* (Magnetresonanztomographie). Dabei wird häufig aus einem Satz zweidimensionaler Bilder ein dreidimensionales Volumen rekonstruiert, das dann als 3D-Modell aus dem gewünschten Winkel betrachtet werden kann. Mitunter wird dem Verfahren der diagnostische Vorteil noch nicht zugesprochen, dient jedoch der Wissensvermittlung an Personen, denen die Daten nicht geläufig sind (Arzt ↔ Patient). So ist es z. B. für werdende Eltern viel interessanter, den Freunden und Verwandten eine Aufnahme ähnlich Bild 7.9 zu zeigen statt ein einfaches Ultraschallbild.

³ <http://www.chromeexperiments.com/>

7.3.4 Ansätze, um Big-Data zu visualisieren

Aus den letzten drei Abschnitten ergeben sich nun einige Lösungsansätze, durch die wir auch große Datenmengen ansehnlich darstellen können. Diese sind neben den traditionellen Punkten, die Tidwell nennt, vor allem diese:

- *Interaktivität* – Erlauben Sie den Betrachtern, mit den Daten zu spielen, die Sichten zu verändern und verschiedene Kennzahlen und deren Beziehungen hervorzuheben. So gelingt es Ihnen nicht nur, mehrere Informationen zu kommunizieren, sondern Sie schaffen auch auf spielerischem Wege Motivation, das Diagramm weiter zu verändern und zu erforschen.
- *Dimensionen nutzen* – Es existiert ein Mythos, der besagt, dass ein Mensch lediglich 10% der Leistung seines Gehirns benutzt. Auch wenn die Wissenschaft diese Aussage heutzutage widerlegt hat, hat sie doch in unserem Falle einen wahren Kern: Wir werden, zumindest beim Betrachten von Diagrammen, nicht besonders gefordert. Häufig offenbart sich uns der Sinn eines Balkendiagramms auf den ersten Blick, so betrachten wir etwa Umsätze eines bestimmten Zeitraums oder die Auslastung pro Mitarbeiter einer Abteilung und sehen sofort, welcher der umsatzstärkste Monat war oder welcher Mitarbeiter der *Mitarbeiter des Jahres* wird.

Seien Sie bei der Auswahl Ihrer Diagramme mutig und verzichten Sie auf Balken- und Tortendiagramme. Wählen Sie stattdessen eines der im folgenden Abschnitt vorgestellten Charts und fordern Sie Ihre Mitarbeiter dadurch heraus, die Daten auch aus einer anderen, möglicherweise viel sprechenderen Sicht zu verstehen. Dieses *Fordern* kann dadurch geschehen, dass Sie neben der einfachen Zweidimensionalität und einer Farbgebung auf Basis von Grün, Gelb und Rot etwas mutiger werden und Bewegungen, zeitliche Veränderungen oder Dreidimensionalität in Ihre Visualisierung aufnehmen (auch hierzu folgen in Abschnitt 7.4 einige Beispiele). Geben Sie den Betrachtern mehr Zeit, um die Diagramme zu verstehen, und falls Sie die Diagramme nicht selbst erklären (etwa in einem Dashboard), ergänzen Sie ein Hilfesymbol, das die Verwendung des Diagramms erklärt. Sie werden sehen, dass Sie Ihr Publikum mit schicken, innovativen Diagrammen viel mehr begeistern als mit althergebrachten Grafiken. Verzichten Sie jedoch auf den Einsatz von Geräuschen, es sei denn, das Hauptaugenmerk Ihrer Datenrepräsentation liegt auf dem auditiven Kanal. Erstens haben Sie nicht immer die Möglichkeit, Geräusche überall abzuspielen, zweitens ist das Gehör der Menschen nicht immer gleich gut ausgebildet, um einzelne Nuancen der Daten zu erkennen, und drittens wirken Töne häufig störend bzw. lenken den Betrachter ab.

⁴ <http://datavisualization.ch/showcases/sound-mapping-in-new-york-city/>



HINWEIS: Allerdings gibt es auch Beispiele, in denen eine Audiospur durchaus unterstützen kann. So stellt Mark Edward Campos beispielsweise das Leben in New Yorks Straße in einem 24-Stundenzyklus dar und zeigt die Öffnungszeiten verschiedener Einrichtungen an, etwa einer Bar, eines Cafés oder eines Fitnessstudios⁴. Er nutzt Bewegungen, um die Tageszeit in Stunden darzustellen, und eine Karte von New York im Hintergrund, um den ungefähren Standort des aktuellen Schauplatzes anzugeben. Campos setzt hier einen Audiokanal ein, um das Ambiente des jeweiligen Ortes widerzugeben. So hört man etwa im Park Vogelgezwitscher oder im Café sich unterhaltende Personen.

- *Kontextsensitivität* – Kontextsensitivität wurde bereits angesprochen, soll hier aber noch mal explizit als Möglichkeit genannt werden, Big Data verständlich zu visualisieren. Nutzen Sie in Ihrem Diagramm Grafiken, die darauf hindeuten, in welchen Themenbereichen sich die Daten bewegen. So können Sie etwa Logos von Automarken in einem Diagramm als Marker verwenden und Beziehungen durch Straßen darstellen. Somit wecken Sie Interesse und müssen Datensätze nicht explizit beschriften, wodurch Sie wiederum Platz sparen.

Behalten Sie diese drei Punkte einmal im Hinterkopf, wenn wir in den folgenden Abschnitten ein Visualisierungsframework auswählen, und überprüfen Sie diese Diagrammtypen auf das Vorhandensein von interaktiven Elementen, kontextsensitiven Darstellungen und die Verwendung verschiedener Dimensionen in Form von Bewegung, Zeit und Schichten. Sie werden sehen, dass diese häufig Verwendung finden.



TIPP: Die Ansätze, zu denen ich hier rate, sind natürlich nicht universell gültig und in gewisser Hinsicht auch von Erfahrungen und Geschmäckern einzelner Personen oder Unternehmen abhängig. Nicht jedem gefallen bunte, sich bewegende Diagramme, manche wünschen sich etwas mehr Konservativität und Gewöhnlichkeit. Hier eine kleine Anekdote von einer Präsentation, die ich auf einem Workshop 2014 zum Thema *Big-Data-Visualisierung* gehalten habe. Mein Vorredner referierte sehr kompetent und kurzweilig über *Visual Business Analytics*, riet jedoch bei animierten, *glitzernden* und interaktiven Diagrammen zur Vorsicht. Ich zeigte hingegen direkt im Anschluss vor allem bunte, ausgefallene, komplexe und interaktive Diagramme. In der ersten Pause kamen die Zuschauer auf die Redner zu und unterhielten sich über das Gesehene. Ich kam schnell mit ein paar Studenten und jungen Entwicklern ins Gespräch, mein Vorredner hingegen mit Leuten aus der Forschung, von Banken und Professoren. Damit will ich sagen, dass es kein *Richtig* oder *Falsch* gibt, sondern dass es immer ganz darauf ankommt, welches Publikum man mit seinen Diagrammen anzusprechen gedenkt.

Relationale Datenbanksysteme galten bis in die Milleniumsahre als unangefochtener Standard, um Daten abzulegen und auszuwerten. Wurde früher in Schulen und Universitäten das Fach Datenbanken unterrichtet, ging es fast ausschließlich um Themen wie Normalisierung, relationale Algebra und das ACID-Modell. Standards abseits von relationalen Datenbanksystemen wie objektorientierte Datenbanksysteme galten als Nischenprodukte.

Martin Fowler propagierte 2011 den Gedanken der Polyglot Persistence (Fowler, 2011). Dieser Begriff besagt, dass für spezifischere Anforderungen auch spezifischere Datenbanksysteme benötigt werden. Er folgerte daraus, dass es eine große Bandbreite an Technologien für unterschiedliche Probleme geben wird.

Über die drei Vs, die in den ersten Kapiteln thematisiert wurden, können Anforderungen skizziert werden, die mit relationalen Systemen nicht oder nur teilweise gelöst werden können. Polystrukturierte Daten passen im Regelfall nicht in zweidimensionale Schemata. Ausnahmen sind relationale Datenbanksysteme, die spezielle Spaltentypen anbieten. Im Regelfall erfordert es eine Kompromisslösung, wenn JSON- oder XML-Dokumente in einer Spalte abgelegt werden sollen. Eine dieser Kompromisslösungen wäre, die Daten als BLOB-Objekt zu speichern und erst beim Auslesen zu interpretieren. Das kann für manche Anwendungsfälle zu langsam sein. Zudem braucht man auch eine API, die das Verarbeiten und Interpretieren der Daten übernimmt.

Eine andere Lösung wäre es, polystrukturierte oder speziell codierte Daten mittels ETL-Werkzeuge aufzubereiten und in eine relationale Datenbank zu laden. Doch hier gibt es zwei Probleme:

- *Performanz:* Bei komplexen Formaten und großen Datenmengen kann das ETL-Werkzeug zum Flaschenhals werden. In einem kontinuierlichen Datenstrom entsteht das Risiko, dass mehr Daten angeliefert als verarbeitet werden können.
- *Duplizierung:* Es kann Applikationen in der IT-Landschaft geben, die die Daten im Ursprungsformat verarbeiten. So müssten die Daten wieder deserialisiert werden, was Zeit kostet und aufwendig sein kann. Eine Alternative wäre, die Daten zweifach zu speichern, was wiederum zu Synchronisationsproblemen führen kann.

Immer mehr Datenbanksysteme und Datenplattformen, die oft auf verteilten Dateisystemen aufbauen, kommen auf den Markt. Antreiber für diese Entwicklung sind neue Features, die für Applikationen benötigt werden, aber auch neue Infrastrukturmöglichkeiten wie etwa die Cloud. Polyglot Persistence ist ein Trend, der seit der ersten Auflage des Buches an

Bedeutung gewinnt: Hierbei stellt sich weniger die Frage, ob Hadoop oder ein relationales Datenbanksystem verwendet werden soll. Auch ob man sich auf unstrukturierte oder strukturierte Daten konzentriert, ist vielleicht nicht mehr die zentralste Frage. Die Kernfrage ist, welches System am besten auf die Anforderungsmatrix passt, die von Anwendungsfällen abgeleitet werden können.

Folgende Trends sind bei den Datenplattformen zu sehen:

- **Dateisysteme:** HDFS ist vielleicht das bekannteste verteilte Dateisystem. Das liegt an der hohen Verbreitung von Apache Hadoop. Es gibt jedoch in der Kategorie DFS (Distributed File System) auch zahlreiche Alternativen. Vertreter davon sind Systeme wie z. B. HopFS (The Morning Paper, 2017). Im Hadoop-Ecosystem ist MapR-XD (vormals MapR-FS) eine interessante Alternative zu HDFS, da sich MapR-FS durch seine Architektur auch besser für kleine Dateien eignet (Drushal, 2017).
- **Relationale Datenbanken:** Auch wenn die Marktführer am Markt wie Oracle und Teradata (Teradata, 2018) ihre Systeme mittlerweile auch in der Cloud anbieten, gibt es zahlreiche Anbieter, die ihre Systeme als native Cloudlösung anbieten und somit neue Dynamiken in den Markt bringen. CockroachDB und Snowflake sind beides Datenbanken, die immer wieder in Verbindung mit Cloud-Infrastrukturlösungen genannt werden. Natürlich darf hier auch Redshift von Amazon nicht unerwähnt bleiben.
- **NoSQL:** Die in Kapitel 5 bereits vorgestellten NoSQL-Systeme werden immer wieder durch neue Systeme erweitert. Auch hier gilt, dass Cloud-Lösungen neuen Schwung bringen können, da Marktführer wie MongoDB, Redis und andere Systeme nicht Cloud-nativ sind.
- **Cloud-spezifische Massenspeicher:** Cloud-native Lösungen wie Amazon S3 werden bei Firmen immer populärer, um Massendaten zu speichern. Die Idee ist, einen Objektstore zu verwenden, bei dem das Speichern von Daten mit geringen Kosten und hoher Verfügbarkeit verbunden ist. Datenverarbeitungsengines wie Spark oder SQL-Engines wie Impala sind auch in der Lage, Daten direkt von S3 zu laden.

■ 11.1 Praxis

Im Folgenden werden verschiedene Systeme anhand von Beispielen vorgestellt, damit sie diese besser evaluieren können.

11.1.1 Redis

Key Value Stores wurden in Kapitel 5 vorgestellt. Ein Vertreter dieser Kategorie ist Redis. Der Hersteller von Redis gibt fünf Hauptanwendungsgebiete an (Redis, 2017):

- User Session Data Management
- Echtzeitanalysen wie Zähler und Ranglisten

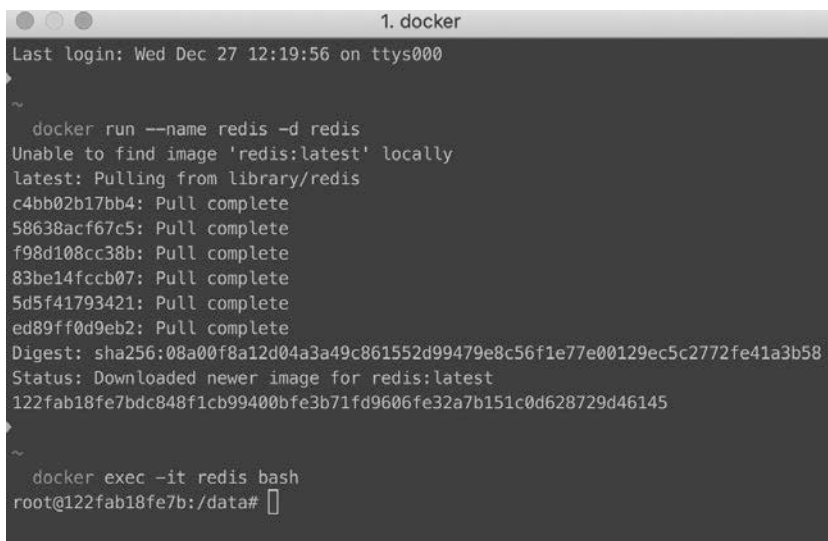
- Empfehlungen wie Kauf- oder Artikelempfehlungen aufgrund von Profilcharakteristika.
- Message Queues für Workflows und andere Jobs
- Caching von statischen und interaktiven Daten

Sie werden sich nun vielleicht fragen, was die Verbindung zu Big Data ist. Redis eignet sich perfekt dazu, Ergebnisse aus Big-Data-Analysen zwischenspeichern. Key Value Stores wie Redis sind somit für den Einsatz in großen Big-Data-Landschaften gut geeignet.

Die folgenden Zeilen erstellen einen Docker-Container mit Redis. Als Erstes wird ein Docker-Container erstellt und gestartet. Das Docker-Image stammt vom Hersteller selbst.

```
docker run --name redis -d redis
docker exec -it redis bash
```

Danach sollten Sie mit einem Docker-Container verbunden sein, in dem Redis läuft.



```
1. docker
Last login: Wed Dec 27 12:19:56 on ttys000

~
docker run --name redis -d redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
c4bb02b17bb4: Pull complete
58638acf67c5: Pull complete
f98d108cc38b: Pull complete
83be14fccb07: Pull complete
5d5f41793421: Pull complete
ed89ff0d9eb2: Pull complete
Digest: sha256:08a00f8a12d04a3a49c861552d99479e8c56f1e77e00129ec5c2772fe41a3b58
Status: Downloaded newer image for redis:latest
122fab18fe7bdc848f1cb99400bfe3b71fd9606fe32a7b151c0d628729d46145

~
docker exec -it redis bash
root@122fab18fe7b:/data#
```

Bild 11.1 Der Docker-Container von Redis wird gestartet.

Die folgenden drei Befehle starten einen Server. Der zweite Befehl ist ein Ping, und mit dem letzten Befehl wird der Client gestartet. Diese Befehle führen Sie bitte in der aktiven Konsole aus.

```
redis-server
redis-cli ping
redis-cli
```

Nach dem Ausführen dieser Befehle sollte die Ausgabe dem folgenden Screenshot ähneln.

```

1. docker
docker exec -it redis bash
root@122fab18fe7b:/data# clear
root@122fab18fe7b:/data# redis-server
20:C 29 Dec 09:26:21.861 # 000000000000 Redis is starting 000000000000
20:C 29 Dec 09:26:21.861 # Redis version=4.0.6, bits=64, commit=00000000, modified=0, pid=20, just started
20:C 29 Dec 09:26:21.861 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
20:M 29 Dec 09:26:21.862 # Creating Server TCP listening socket *:6379: bind: Address already in use
root@122fab18fe7b:/data# redis-cli ping
PONG
root@122fab18fe7b:/data# redis-cli
127.0.0.1:6379>

```

Bild 11.2 Die Redis-Kommandozeile

Nachdem der Client gestartet ist, setzen Sie einen einfachen Wert:

```

set mykey somevalue
get mykey

```

Mit diesen beiden Befehlen decken Sie bereits den Kernanwendungsfall eines Key Value Store ab: Sie setzen einen Wert und rufen ihn wieder ab.

Ein beliebiger Anwendungsfall von Key Value Stores sind Zähler und Ranglisten.

```

set counter 100
incr counter

```

Wer Redis gerne auch einmal mit Python ansprechen will, kann folgende Befehle ausführen:

```

apt-get update
apt-get install -y python python-pip
sudo pip install redis

```

Wer dann den Python-Interpreter startet, kann die Python-REPL nutzen, um folgende Befehle auszuführen:

```

>>> import redis
>>> r = redis.StrictRedis(host='localhost', port=6379, db=0)
>>> r.set('foo', 'bar')
>>> r.get('foo')

```

Diese kleine Einführung sollte ausreichen, um weiter mit Redis zu arbeiten. Im Internet finden Sie zahlreiche Anleitungen, wie Sie mit der Programmiersprache Ihrer Wahl Daten in Redis verarbeiten können.

11.1.2 MongoDB

MongoDB ist eine freie, plattformübergreifende Dokumentendatenbank (MongoDB, 2018)

Folgende Zeilen sind der schnellste Weg zum Testsystem:

```
docker run --name mongo -d mongo
docker exec -it mongo bash
```

Der Mongo-DB-Client wird gestartet, und einzelne Befehle werden ausgeführt:

```
mongo
db.literature.insert([
  {
    title: 'Hadoop - Definitive Guide',
    author: 'Tom White',
    publisher: 'OReilly Media',
    url: 'http://shop.oreilly.com/product/0636920033448.do',
    tags: ['hadoop', 'hive', 'yarn'],
    releasedate: 'April 2015',
    Pages: 235
  },
  {
    title: 'PolyglotPersistence',
    description: "An article on polyglot persistence",
    author: 'Martin Fowler',
    url: 'https://martinfowler.com/bliki/PolyglotPersistence.html',
    tags: ['rdbms', 'mongodb', 'neo4j'],
    releasedate: '16. November 2011'
  }
])
db.literature.find().pretty()
```

Obiges Beispiel veranschaulicht exzellent den klassischen Anwendungsfall eines Literaturverzeichnisses. Literaturverzeichnisse können unterschiedliche Felder haben und sind somit als Schema schwer fassbar. Ist dieses Schema in JSON in MongoDB festgehalten, ist es auch leicht möglich, die Daten zu extrahieren und als JSON-Dokument an Applikationen zu übergeben.

11.1.3 Neo4j

Neo4j ist eine Graphendatenbank, die sich für die Darstellung von Beziehungen eignet. Sie gilt als Standard für alle Formen von Graphen, wie z. B. die Darstellung von Wegverbindungen. Neo4j kann über folgenden Befehl gestartet werden:

```
docker run \
  --publish=7474:7474 --publish=7687:7687 \
  --volume=$HOME/neo4j/data:/data \
  neo4j
```

Nun kann im Browser auf <http://localhost:7474> gewechselt werden, wo man sich mit dem User *neo4j* und dem Passwort *neo4j* in der WebUI einloggen kann. Dort findet man auch ein einfaches Tutorial zu neo4j.

11.1.4 S3

Sie kennen wahrscheinlich die berühmte Gretchen-Frage aus Goethes *Faust*. Abgeleitet davon kann man auch vielen IT-Leitern die berühmte Frage stellen: „Nun sag, wie hast du’s mit der Cloud?“

Die Antworten könnten nicht unterschiedlicher sein. Viel hängt auch davon ab, in welchen Branchen die Unternehmen tätig sind. Auch Kultur spielt eine Rolle. Im angloamerikanischen Raum scheint man Cloud-Technologien gegenüber offener zu sein als im zentraleuropäischen Bereich.

Einige Unternehmen halten am Grundsatz fest: „Wir geben doch unsere Daten keinem amerikanischen Unternehmen, wo wir gar nicht wissen, was da passiert!“ Andere beginnen, sich nach und nach zu öffnen. Lassen Sie uns deswegen einen Blick auf Amazon S3 werfen. Neben Amazon S3 gibt es natürlich auch andere Cloud-Anbieter, die eine Art Massenspeicher in der Cloud anbieten, der wie ein Filesystem genutzt werden kann.

Die Daten, die auf S3 gespeichert werden, haben eine garantierte Verfügbarkeit von 99,99999999%. Auf <https://aws.amazon.com/s3/pricing/> können die Preise abgerufen werden. Mit Stand beim Schreiben dieses Buchs kostete ein Gigabyte 0,0245 US\$. Ein Terabyte für 24,50 Euro ist für eine Firma kein großes Problem. Würden eine Firma ein Terabyte an Daten selbst hosten, müsste sie Infrastruktur einkaufen und IT-Mitarbeiter einstellen, die den Betrieb gewährleisten. Selbst bei billigster Hardware und geringsten Gehaltskosten ist offensichtlich, dass internen IT-Abteilungen ein ähnliches Schicksal drohen könnte wie dem Tante-Emma-Laden um die Ecke.

Für viele Kunden stellt sich also durchaus die Frage, warum nicht Daten auf Massenspeicher wie Amazon S3 hochladen und das Filesystem wie HDFS nutzen. SQL-Engines wie Impala können auch S3-Daten abfragen, und auch Apache Spark verfügt über Schnittstellen zu S3.

Berücksichtigen sollte man bei der Entscheidung für S3 allerdings Sicherheits- und Performanceaspekte. Ein Kompromiss, den viele Unternehmen dann in Erwägung ziehen, ist ein Hybridbetrieb. Amazon S3 wird manchmal als sogenanntes Cold Storage genutzt, um Daten zu archivieren. Auch die Durchführung von sogenannten POCs (Proof of Concepts) in der Cloud kann für viele Unternehmen reizvoll sein.

Wer Daten auf Amazon Web Services (AWS) ablegen möchte, muss sich einen Account anlegen. Dabei kommt wieder das leidige Thema auf, dass man bei Cloud-Anbietern seine Kreditkartendaten hinterlegen muss. Ich verstehe vollkommen, dass das nicht jeder will. Falls dieser Schritt – Ihre Finanzdaten einzugeben – für Sie in Frage kommt, können Sie, nachdem Sie Ihren Account angelegt haben, ein paar der folgenden Schritte selbstständig nachspielen. Die folgenden Schritte sollten selbsterklärend sein.

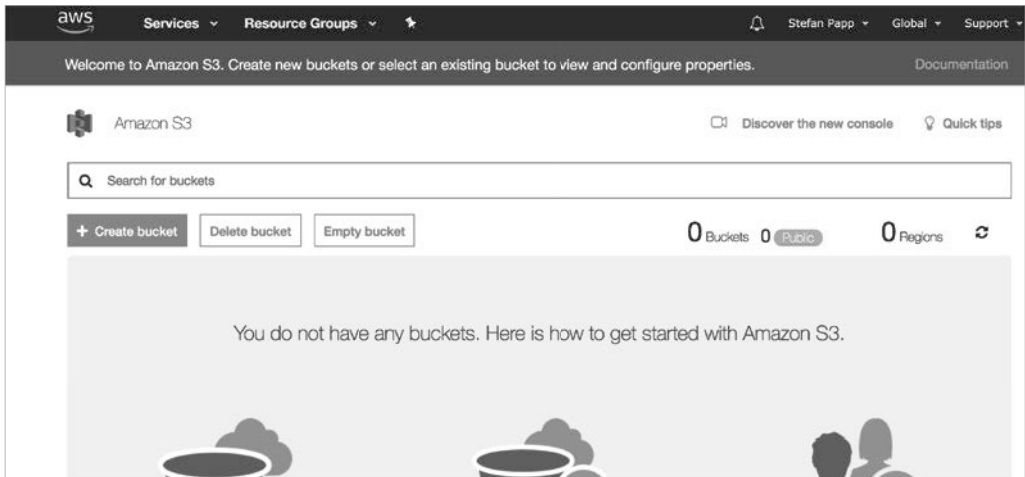


Bild 11.3 AWS-Benutzeroberfläche

Bild 11.3 zeigt die Amazon S3-Oberfläche. Sie können sogenannte *Buckets* anlegen (siehe Bild 11.4), also Container, in denen man ähnlich wie in Verzeichnissen Daten ablegen kann.

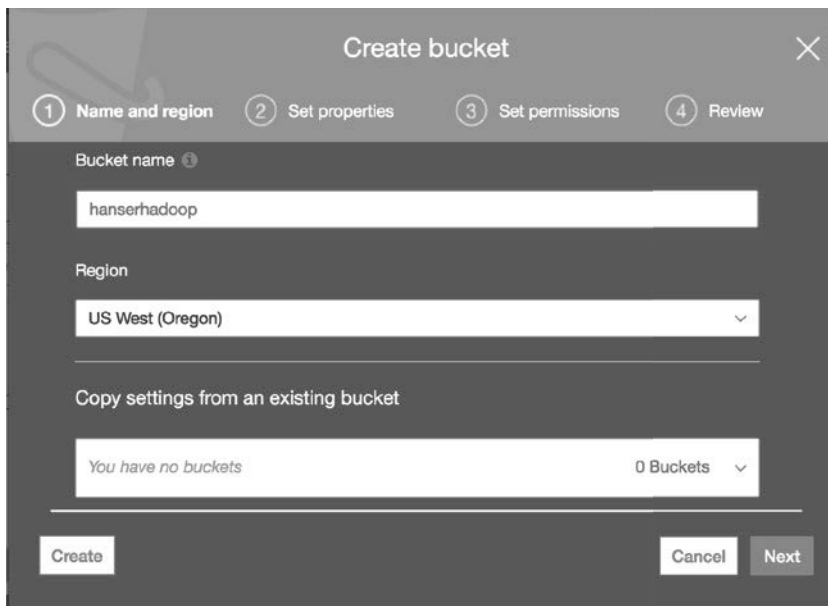


Bild 11.4 Erstellen eines Buckets

Im Schnelldurchlauf kann man einige Parameter durchgehen, die ebenfalls selbsterklärend sind, um die Schritte in der AWS-UI abzuschließen. Man kann festlegen, wer auf die Buckets Zugriff bekommt, und auch eine Versionierung ist konfigurierbar. Wesentlich ist, einen eindeutigen Namen festzulegen und bei der Namensfindung kreativ zu sein, da bei vielen weltweiten Nutzern natürlich auch viele eindeutige Namen bereits vergeben sein können.

Um dann vom lokalen Dateisystem auf die Cloud zugreifen zu können, müssen Sie auch noch unter https://console.aws.amazon.com/iam/home?#/security_credential Ihre Security Credentials konfigurieren.

Wer von der Kommandozeile Daten in die Cloud hochladen will, muss eine API installieren, die man über die Konsole auch aufrufen kann. Die AWS CLI ist in Python geschrieben und kann auch über den Python-Installer installiert werden:

```
pip install awscli
```

Danach gilt es, die CLI zu konfigurieren. Wesentlich dabei ist, die davor generierten Schlüssel zu setzen. Dies geschieht über

```
aws configure
```

Ich habe im Vorfeld, wie man im Screenshot sehen kann, einen Bucket namens *hanserhadoop* angelegt. Diesen kann ich nun abfragen:

```
aws s3 ls s3://hanserhadoop
```

Um zu zeigen, dass alles funktioniert, legen wir eine leere Datei an und laden sie auf AWS hoch. Danach führen wir den *ls*-Befehl erneut aus:

```
touch test.txt
aws s3 cp test.txt s3://hanserhadoop
aws s3 ls s3://hanserhadoop
```

The screenshot shows a terminal window titled "1. zsh" with three tabs labeled "zsh", "%1", "zsh", "%2", and "zsh", "%3". The terminal output is as follows:

```
Last login: Fri Jan 26 18:03:34 on ttys003

~
touch test.txt

~
aws s3 cp test.txt s3://hanserhadoop
upload: ./test.txt to s3://hanserhadoop/test.txt

~
aws s3 ls s3://hanserhadoop
2018-01-26 18:24:13          0 test.txt

~
[]
```

Bild 11.5 AWS-Befehle auf der Kommandozeile ausführen

Natürlich sieht man die hochgeladene Datei dann auch in der Web-Oberfläche.

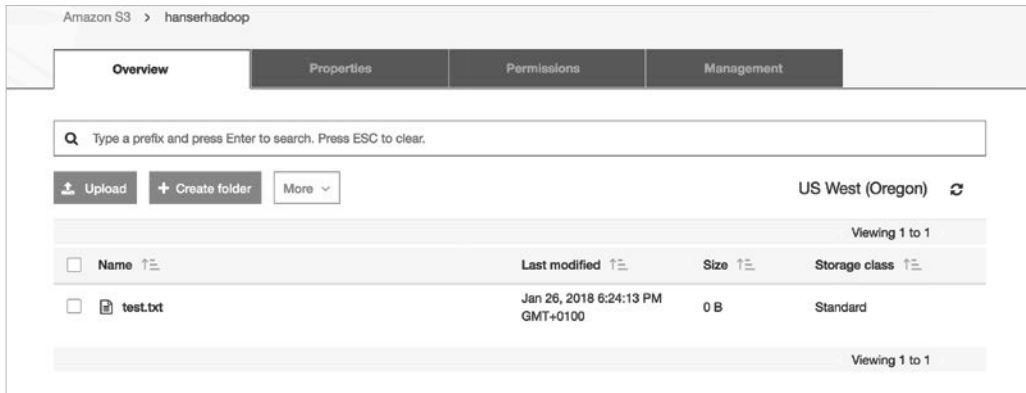


Bild 11.6 AWS Web-Oberfläche

Die Daten in der Cloud lassen sich also von den eigenen Systemen aus abfragen. Die logische Konsequenz ist, dass wir alle Aktionen, die wir auf einem verteilten Dateisystem wie HDFS ausgeführt haben, auch auf S3 ausführen können.

11.1.5 Apache Kudu

In diesem Buch wurde als spaltenorientierte Datenbank HBase bereits hinreichend dargestellt. Es macht jedoch Sinn, eine Alternative zumindest zu erwähnen.

Apache Kudu ist ein Projekt, das ein Hybrid zwischen einer Datenbank und dem Filesystem sein will.

Leider gab es zum Zeitpunkt des Verfassens des Buchs kein funktionierendes Docker-Image. Deswegen kann ein Quickstart nur unter <https://kudu.apache.org/docs/quickstart.html> abgerufen werden.

■ 11.2 Zusammenfassung

In diesem Kapitel wurde gezeigt, dass es für jeden Anwendungsfall ein eigenes System gibt, um Daten abzulegen und zu analysieren. Die Ecosysteme ändern sich rasant und Sie haben einen Vertreter aus jeder Kategorie kennegelernt.

Wichtig ist, die Technologie immer den spezifischen Anforderungen entsprechend auszuwählen. „Glaubenskriege“ darüber welche Technologie die beste sei, sind hier meist nicht besonders hilfreich.

Index

Symbole

\$HOME 35
\${JAVA_HOME} (Maven) 142
.bashrc 36
/etc/hostname 80, 89
/etc/hosts 75

A

Access Control Lists 495
Accountability 486
ACID 197, 439
AE 402
Aktienkurse 370
Alation 493
Alpine 417
Alternativen zu Hadoop 184
Amazon 185
Amazon S3 440
Ambari 85, 188
AMRMClient 152
AMRMClientAsync 152
Anreichern 365, 370, 375
Apache Atlas 499
Apache Commons 161, 311
Apache Flink 462, 480
Apache Knox 493
Apache Kudu 447
Apache Oozie 114
Apache POI 119
Apache Sentry 298, 492
ApplicationMaster 29, 79, 136, 149
ApplicationsManager 79
ApplicationSubmissionContext 147
Atkinson, Rowan 14

Aufbereitung 16
Ausdünnung 336
Authentifizierung 491
Autorisierung 491
AWS 444
Azkaban 114

B

BASE 197
BashReduce 184
Batch Layer 461
Batchverarbeitung 461
Beispieldaten 6
BigTable 195
Breadcrumbs 168
Brewer's Theorem 196
Bubble-Charts 349
Bucketing 286
Business Intelligence 489f.

C

Calendar-Chart 347
CapacityScheduler 147
CAP-Theorem 196
CAS 412
Cascading 114, 457
Cascading Style Sheet 167
Cassandra 185, 197, 204
CentOS 417
Channel (Flume) 190
Chord-Chart 346
Chord-Diagramm 349
Choroplethenkarte 331

Chrome Experiments 341
 Churn-Modell 1
 CLI 257
 Cloudera 424
 CockroachDB 440
 Column-Family 203, 223, 226, 229 f.
 Combine-Phase 47f.
 Commodity-Hardware 21, 46, 195
 commons-fileupload 171
 Compliance 489
 Configuration 140, 145, 163, 227
 Configured 102
 Container 415
 ContainerLaunchContext 146
 ContainerRequests 152
 ControlledJobs 114
 copyFromLocal 26, 82
 copyToLocal 26
 core-default.xml 146
 core-site.xml 38, 146
 Cost Based Optimizers 460
 CouchBase 185
 CouchDB 197
 cp 26
 Customer Experience Solutions 1
 Cutting, Doug 10

D

DAG 114, 256
 Data Catalog 487
 Data Classification 487
 Data Driven Documents 351
 Data Governance 485
 Data in Motion 496
 Data in Rest 496
 Data Lake 18, 23
 Data Management 486
 Data Monetisation Use Cases 1
 Data Node 24, 38, 74
 Data Ownership 486
 Data Processing Engines 457
 Data Retention 486
 Data Scientist 48, 368
 Datasets 462
 Data Swamp 485
 Data Visualizer 333
 Data Warehouse 253, 327, 365
 Datenkompression 41
 Datenlokalität 22

Datenmigration 201
 Datenschutzgrundverordnung 485
 Denormalisierung 249
 Deployment Descriptor 87
 Derby 199
 DevOps 415
 dfs.datanode.data.dir 39
 dfs.datanode.name.dir 39
 dfs.permissions 39
 dfs.replication 39
 Diagrammempfehlungen 378
 Directed Acyclic Graph 114, 460
 Disco 184
 Docker 421
 Dockerfile 454
 Document Object Model 358
 Document-Store 199
 Dokumentenorientierte Datenbank 199
 DOM 358
 Domäne (GlassFish) 52
 Driver 62, 101
 DSGVO 485

E

Echtheit von Daten 14
 Eclipse 49
 Entwicklungsumgebung 49
 Ereignisdaten 13
 ETL 439
 ext4 24

F

Fehler 44
 Fehlererkennung 179
 Fehler in Daten 13
 Fehlertoleranz 179
 FileSplit 123
 FileStatus 151
 FileSystem 160
 final 40
 Flare-Chart 344
 Flume 189
 Formale Sprachen 386
 Formatieren 41
 Fourth Extended Filesystem 24
 Fragen 6
 Fremdschlüssel (Sqoop) 225

fs.file.impl 141
 fs.hdfs.impl 141
 Full Profile (JavaEE) 53
 Fully distributed 31
 Funktionale Programmierung 429

G

GDPR 485
 GenericOptionsParser 64
 getApplications 176
 getNodeRepos 175
 Git 418
 Github 375
 Glassfish 50
 Goals (Maven) 59
 Google File System 24
 google-gson 363
 GPPE 457
 Gradle 429
 Graphen-Datenbanken 199, 443
 Graph Engines 473
 GraphFrames 471

H

H2O 471
 Hadoop 21
 Hadoop Distributed File System 22f.
 Hadoop-Ecosystem 187
 Hadoop Process Definition Language 192
 HADOOP_USER_NAME 94
 Hashing 497
 HBase 190, 200, 478

- Autosharding 207
- BinaryComparator 239
- BinaryPrefixComparator 239
- BitComparator 239
- Bulk Loading 214
- Bytes.toBytes 230
- Cell 232
- ColumnCountGetFilter 238
- ColumnPaginationFilter 237
- ColumnPrefixFilter 237
- ColumnRangeFilter 237
- Comparator 239
- CompareFilter 238
- count 210
- create 209

- delete 210
- Delete 236
- deleteall 210
- DependentColumnFilter 238
- disable 210
- Distributed-Mode 211
- drop 210
- enable 211
- EQUAL 238
- Family 232
- FamilyFilter 237
- Filter 237
- FilterBase 238
- FilterList 239
- FirstKeyOnlyFilter 238
- get 209
- Get 232
- GREATER 239
- GREATER_OR_EQUAL 239
- Hadoop-JARs 212
- Hot Spotting 205
- InclusiveStopFilter 238
- KeyOnlyFilter 238
- LESS 238
- LESS_OR_EQUAL 238
- list 211
- MultipleColumnPrefixFilter 237
- MUST_PASS_ALL 240
- MUST_PASS_ONE 240
- NO_OP 239
- NOT_EQUAL 239
- NullComparator 239
- PageFilter 237
- Paging 233, 242
- PrefixFilter 238
- Pseudo-Distributed-Mode 211
- put 209
- Put 234
- Qualifier 232
- QualifierFilter 237
- RegexStringComparator 239
- Region 205
- Region-Server 206
- Row 232
- RowFilter 237
- Row-key 204
- scan 210
- Schema 205
- setReversed 233, 242
- Shell 209
- SingleColumnValueExcludeFilter 238

- SingleColumnValueFilter 237
- Stand-alone 207
- SubstringComparator 239
- TimestampFilter 237
- Value 232
- ValueFilter 238
- Vergleichsoperatoren 238
- Web-Interface 213
- HBaseAdmin 227
- HBaseConfiguration 227
- hbase-default.xml 227
- hbase-env.sh 208
- HBase Java-API 226
- hbase-site.xml 207, 212, 227
- HCatalog 218, 254
- HColumnDescriptor 230
- HDFS 23, 440
- hdfs-site.xml 38, 78
- Heat Map 337
- Herunterladen (HDFS) 161
- HFiles 214
- History Server 74
- Hive 191
 - ADD COLUMNS 288
 - Aggregatfunktionen 276
 - ALTER 286
 - Architektur 256
 - Arithmetische Operatoren 269
 - Authentication-Provider 297, 320
 - Auto-Increment 277f.
 - Autorisierung und Authentifizierung 297
 - Benutzer 293
 - Benutzerverwaltung 321
 - Bucketing 271
 - Case-Sensitivity 259
 - CHANGE 288
 - CLI 322
 - CLUSTER BY 271
 - Command Line Interface 257
 - COMMENT 259, 287
 - Compiler 256
 - COUNT() 265
 - CREATE DATABASE 259
 - CREATE EXTERNAL TABLE 263
 - CREATE ROLE 294
 - CROSS JOIN 291
 - DESCRIBE FUNCTION 275
 - Directed Acyclic Graph 256
 - DISTINCT 269
 - DISTRIBUTE BY 271
 - Driver 256
 - DROP 289
 - DROP ROLE 294
 - Execution Engine 256
 - EXPLAIN 266
 - Externe Tabellen 262
 - FULL OUTER JOIN 291
 - GRANT 296
 - Grantor 295
 - GRANT ... TO ROLE 295
 - GROUP BY 272
 - Gruppen 293
 - HAVING 285
 - HCAT_HOME 323
 - hive-default.xml.template 255
 - HiveQL 253, 259
 - hiveserver 257
 - hiveServer2 257, 307
 - hive-site.xml 255, 293, 302, 304
 - Hive-Web-Interface 257
 - IF NOT EXISTS 261
 - Import 322
 - IMPORT 264
 - Interactive Shell Mode 258
 - JDBC 304
 - JOIN 290
 - JPam 303
 - KERBEROS 303
 - Komplexe Datentypen 260
 - LDAP 303
 - LEFT OUTER JOIN 291
 - LEFT SEMI JOIN 291
 - Lightweight Directory Access Protocol 303
 - LIMIT 266
 - LOAD DATA 264
 - LOCATION 262
 - Logging 284
 - Logische Operatoren 267
 - Mathematische Funktionen 274
 - Metadaten 317
 - Metastore 256
 - ORDER BY 270
 - OVERWRITE 265
 - Paging 305, 307, 314, 318, 320
 - PAM 303
 - Partitionierte Tabellen 262
 - Partition Pruning 269
 - PasswdAuthenticationProvider 299
 - Pluggable Authentication Module 303
 - Primitive Datentypen 260
 - Privilegien 293

- REPLACE 288
- ResultSet 309
- ResultSetMetaData 319
- REVOKE 296
- RIGHT OUTER JOIN 291
- Rolle 293
- Security 292
- SELECT 265
- SELECT .. AS 266
- SELECT ... WHERE 266
- setCatalog 309
- SET TBLPROPERTIES 289
- SHOW DATABASES 259
- SHOW FUNCTIONS 275
- SHOW GRANT 295
- SHOW ROLES 296
- SORT BY 270
- Stinger-Initiative 254
- String-Funktionen 276
- Subquery 273
- UDFType 279
- UNION 273
- USE 259
- User-Defined Functions 277
- View 289

Hive Metastore 498
HiveQL 114, 191
Hochladen (HDFS) 161
Hortonworks 424
HPDL 192
HTableDescriptor 230
HTML5 341, 350
HttpFS 27
HttpServlet 171
Hue 257
hung_task_timeout_secs 224
HWI 257

I

ifconfig 44
Impala 254, 325
ImportTsv 215, 226
include file (JSP) 362
Infografik 330, 349
Infrastructure as Code 424
In-Memory 182
In-Memory-Datenbanken 199
InputFormat 116f.
InputFormat-Klassen 66

InputSplit 116f., 123
InputStreamReader 354
IPv6 35
isSplittable 121

J

JAAS 300
Java 429
Java Authentication and Authorization Service 300
Java Database Connectivity 257
Java Persistence API 363
Java Runtime Environment 33
JavaScript-Validator 355
JDBC 257
JDK 50
jdk.tools 227
JobControl 114
JobTracker 29
Join 251
JPA 363
Jps 43
JRE 33
JSON 354
JSON.parse 360
Jupyter 436, 470
JVM 427

K

Kafka 449
Kali-Linux 417
k-Anonymity 501
KDD-Prozess 16
Kerberos 494
Key-Value-Datenbank 198
KeyValueInputFormat 66
KeyValueTextInputClass 66
Klassifikator 388, 391
Klassifizierung 387
Knowledge Discovery in Databases 16
Kompression 42
Kontextsensitive Diagramme 339

L

Lambda 461, 477
 Lambda-Expressions 183
 Language Detection 387
 Latenzzeit 253
 Lineare Regression 472
 LingPipe 387
 LoadIncrementalHFiles 215
 LocalResource 146
 LocalResourceVisibility 146
 Loci-Methode 330
 Log-Aggregation 132
 Logging (Hadoop) 131, 136
 ls 26

M

Machine Learning 193, 393
 Mahout 193
 Mapper 61, 102, 105
 Map-Phase 47f.
 MapR 424
 mapred 64
 mapred-site.xml 39, 42
 Map Reduce 22, 46
 mapreduce.task.timeout 224
 Maven 55, 429
 maven-assembly-plugin 59, 137
 MD5 205
 Mehrdeutigkeiten 387
 META-INFO 141
 Misco 184
 mkdir 26
 Mobius 437
 MongoDB 185, 443
 MPP 476
 mv 26
 MySQL 199
 MySQL-Server 218

N

Name Node 24, 38, 74
 Nashorn 352
 Natural Language Processing 385
 Neo4j 185, 199, 443
 netcat 480
 NLP 385

Node Manager 74, 152
 Normalform 249
 Not only SQL 195
 NullOutputFormat 119

O

ODBC 257
 Oozie 192
 Open Data 375
 Open Database Connectivity 257
 OpenNLP 179, 366, 387
 – Abhängigkeiten 388
 – DoccatModel 390
 – DocumentCategorizerME 391
 – getAllResults 391
 – getBestCategory 391
 OpenSSH-Client 76
 OpenSSH-Server 34
 OpenStreetMap 344
 Oracle 440
 ORC 498
 OutputCollector 118
 OutputFormat 116, 119

P

PARQUET 498
 Partitioner 118
 Partitions 118
 PDFBox 119
 PDFInputFormat 120
 Perimetersicherheit 491
 Perspektive (Eclipse) 50
 PhantomJS 352
 Pig 191, 326, 457
 Pig Latin 114, 191, 326
 PII 485
 Plattformsicherheit 491
 Policy 489
 Polyglot Persistence 439, 449
 Polystrukturiert 13
 Primärschlüssel 250
 Privacy by Design 501
 Programmiersprachen 427
 Project Facet 87, 107, 161, 227, 306
 Projekt importieren (Eclipse) 56
 Pseudo distributed 31
 Python 433

Q

Quality 489
Queue 147

R

R 436
Random 358
Random Read/Write 253
RDBMS 201f.
RDDs 463
Recommendation-Engine 377
RecordReader 116f.
RecordWriter 116, 125, 127
Redirect 248
Redis 440
Redshift 440
Reduce-Phase 47f.
Reducer 62, 103, 106
Region-Server 213
Regular Expression 239, 268
Relational Database Management System 201
Relationale Datenbank 198, 439
Relationen herstellen 365
Repliken 25
Reporter 118
Resilient Distributed Datasets 183
ResourceManager 29, 74, 256
ResourceTracker 79
REST-API 85
rm 26
RStudio 437

S

S3 444
Sandbox 31
SAP-Hana 197
SAXParserException 44
Scalable Vector Graphics 350
Scala (Programmiersprache) 182, 430
Scalding 457
Scale-out 200
Scale-up 200
Scheduler 79
Schemafreiheit 195
Scoop 217
Scrollen (Ubuntu) 70

Secure Shell 34
Sekundärschlüssel 250
Selbstfahrende Autos 506
Sensordaten 370
Sentiment-Analysis 385, 387, 393
SequenceFileInputFormat 66
SequenceFileOutputFormat 119
Service Layer 461
setInputFormatClass 66
setJarByClass 98
setMapperClass 66
setOutputFormatClass 66
setOutputKeyClass 66
setOutputValueClass 66
setReducerClass 66
SFTP 305
Shared-Nothing-Architektur 458
Shark 183
Shuffling 118
SingleColumnValueFilter 236f., 239
Single Point of Failure 28, 74
Sink (Flume) 190
Snowflake 440
Social Media 13
Source (Flume) 190
Spaltenorientierte Datenbank 198
Spark 182, 199
Spark-Ecosystem 470
Spark Graph 473
SparkML 470, 472
SparkSQL 470f.
Spark Streaming 471
Speed Layer 461
Split-Phase 47
Splits 116
Sprachen 385
Sprachenerkennung 386
Sqoop 189, 205, 226, 234, 248, 254, 322
Sqoop2 217
sqoop-env.sh 218
sqoop-env-template.sh 217
SSH 34
SSH File Transfer Protocol 305
Stack Traces 70
Standalone 31
Standardisierung 487
start-all.sh 43
StoreFiles 216
Storm 182
Streaming 461, 475
StringBuilder 246

sudo 33
 Supervised Learning 472
 SVG 350
 systemPath (Maven) 142

T

Tag Cloud 336, 344
 Talend Open Studio 115
 TensorFlow 471
 Teradata 440
 Testdaten 100, 356
 Textanalyse 365, 384
 TextInputFormat 66
 Text-Mining 384
 TextOutputFormat 119
 Thrift 214, 257
 ToolRunner 64
 touchz 27
 Trainingsdaten 179, 387ff.
 Transaktionsdaten 12
 Transparent Encryption 496
 Transparenz 487
 Tree Map 345
 Trifacta 493
 TSL-Verschlüsselung 496

U

Ubuntu Server 31, 417
 UDF 277
 UIMA 366, 385, 394

- Abhängigkeiten 396
- Analysis-Engine 402
- AnnotationIndex 402
- CAS 412
- Eclipse-Plug-in 395
- getDocumentText 402
- JCas 399
- JCasGen 398
- Primitive 403
- Testkonfiguration 407
- Type System Definition 397

 Unstructured Information Management Architecture 394
 Upload-Servlet 169
 URL-Codierung 166, 168
 URL-Decodierung 166, 168

V

Validator 373, 385
 Variety 12
 Velocity 12
 Verantwortung 487
 Verarbeitung (Big Data) 177
 View (Eclipse) 50
 Vim 419
 Visual Analytics 334
 Visualisierung 329

- 3D-Diagramme 341
- Assoziation 340
- Audio 342
- Aufmerksamkeit lenken 337
- Circos 350
- Computertomographie 341
- D3.js 350ff., 361
- Datameer 349
- Datenstrukturen 335
- Datumswerte 347
- Diagrammarten 344
- Dimensionen 342
- Frameworks 350
- Geografische Daten 336
- Hierarchische Daten 335
- infogr.am 350
- InfoViz 350
- Interaktion 338
- Interaktivität 342
- JGraph 351
- Klassische Werkzeuge 348
- Kontextsensitivität 343
- Lineare Daten 335
- Magnetresonanztomographie 341
- Matlab 350
- m-n-Relation 344
- Netzstruktur 335
- Precog 349
- Processing 350
- ReportGrid 349
- RGB 337
- R-Project 350
- Tabellarische Daten 335
- Visualisierungsempfehlungen 365

 VMware Player 31
 Volume 11
 Vorverarbeitung 195
 VVV 11

W

waitForCompletion 66, 98, 112
Waterline Data 493
WebGL 341
Web Graphics Library 341
Web-Interface 44, 70, 132, 156
Web Profile (JavaEE) 53
Wikipedia 9
Windows-Binaries (Hadoop) 92
WinSCP 56, 68
winutils.exe 511
Word Cloud 344

X

XAMPP 219
XML-Validator 397

Y

YARN 22, 28
YarnClient 144, 160, 174

YarnConfiguration 145
YARN-Kompatibilität 39
yarn.nodemanager.aux-services 40
yarn.nodemanager.aux-services.mapreduce.
 shuffle.class 40
yarn.nodemanager.delete.debug-delay-
 sec 40, 157
yarn.nodemanager.vmem-pmem-ratio 40
yarn-site.xml 40, 79
Yet Another Resource Negotiator 28

Z

Zielgruppe 4
Zookeeper 191, 207, 228, 451
 – Quorum 228, 241