entwickler.press

MongoDB

Sag ja zu NoSQL

Marc Boeker

Marc Boeker

MongoDB

"If you're looking at this book, you're probably interested in a scalable, high-performance non-relational database solution like MongoDB. For quick and practical coverage of many aspects of installing, deploying, configuring, and developing MongoDB and MongoDB client applications, this book is for you."

Eliot Horowitz, CTO & Co-Founder, 10gen

entwickler.press

Marc Boeker
MongoDB
ISBN: 978-3-86802-244-5

Ein Imprint der Software & Support Media GmbH

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Ihr Kontakt zum Verlag und Lektorat:
Software & Support Media GmbH
entwickler.press
Geleitsstr. 14
60599 Frankfurt am Main
Tel.: +49 (0)69 630089-0
Fax: +49 (0)69 930089-89
lektorat@entwickler-press.de
http://www.entwickler-press.de

Lektorat: Sebastian Burkart
Korrektorat: Ella Klassen
Satz: Pobporn Fischer
Belichtung, Druck & Bindung: M.P. Media-Print Informationstechnologie GmbH, Paderborn

Inhaltsverzeichnis

F wie Vorwort

Zuerst einmal vielen Dank, dass du dich für dieses Buch, das du gerade in deinen Händen hältst, entschieden hast. Wie, es ist (noch) gar nicht deins? Kein Problem, dann schon mal vielen Dank für dein Interesse.

Viele Bücher sagen immer: Erstellen Sie zu erst ein Pflichtenheft, führen Sie daraus das Lastenheft ab und entwickeln Sie ein Entity Relationship Model und ein UML-Diagramm. Das Projekt setzen Sie dann nach dem Wasserfallmodell um, blah blubb. Immer der gleiche Käse. Mein Ziel ist es, dass du mit dieser übergewichtigen Step-by-Step-Anleitung am Ende eigene Anwendungen entwickeln kannst, die MongoDB richtig einsetzen. Wie, du kennst die Basics alle schon? Super, dann lass uns zusammen mit diesem Buch Replica Sets konfigurieren, ein Sharding Cluster aufsetzen und geobasierte Apps bauen.

Für wen ist diese Lektüre denn überhaupt gedacht? Meiner Meinung nach für alle, die lesen können und an einer chronischen SQL-Frustration leiden. Auf den folgenden Seiten versuche ich mit MongoDB eine, aus meiner Sicht, geniale Alternative zu herkömmlichen RDBMS (Relationalen Datenbankdingsbums) vorzustellen. Ja ja, der Name MongoDB ist etwas unglücklich gewählt, aber was soll's? Hauptsache, der Chef/Kunde kann sich nach dem Meeting zumindest noch an den Namen der Datenbank erinnern. Und wenn MongoDB nebenbei noch das Herzinfarktrisiko bei der Entwicklung von Webapps senkt, umso besser.

Wenn das Buch schon MongoDB heißt, dann darf man wahrlich keine sonderlich große Objektivität und Fairness gegenüber anderen Datenbanksystemen erwarten. Immerhin haben uns diese in der letzten Zeit schon oft genug in den Wahnsinn getrieben. Da das alles hier aus der Sicht eines zufriedenen und teilweise MongoDB-süchtigen Anwenders geschrieben ist, bitte ich vorsorglich schon einmal um Nachsicht.

Natürlich möchte ich gerne noch den obligatorischen Dank an bestimmte Personen aussprechen, die (in)direkt an diesem Buch mitgewirkt haben. Da wären Bea und Heiner, denen ich mit zwölf Jahren eine monatliche Internetrechnung von 490 DM präsentiert habe und die mich trotzdem haben weitersurfen lassen, in der Hoffnung, dass der Junge mal etwas Anständiges wird. Vielen Dank an euch. Hat auch, glaube ich, geklappt. Denn heute schreibt er nämlich MongoDB-Bücher. Dann gibt es noch Michael, Freund und Gründerkollege bei ONchestra.com, der immer beide Augen zugedrückt hat, wenn sich meine Augenringe um einen Platz an der Sonne geschlagen haben, nachdem eine literarische Nachtschicht angesagt war. Ebenfalls einen großen Dank an Sebastian von entwickler.press, der sehr diplomatisch meine überphantasievollen Buchtitel kommentiert und die nötige Ordnung in mein Geschreibsel gebracht hat. And a huge thanks to Richard Kreuter

and his colleagues from 10gen for doing a gorgeous review job. Ja und dann gibt es noch meine mehr als bessere Hälfte Diana, die immer die richtige Dosis an Verständnis, Motivation, Unterstützung und Humor parat hatte. Daher gehe ich auch vom Erlös des Buchs mit ihr etwas Leckeres essen. Ob es ein gutes Restaurant oder einfach Burger King wird, entscheidet letztendlich ihr.

1 Wer? Wie? Was? – MongoDB kurz und knackig

1.1 Was ist MongoDB?

In der letzten Zeit scheint NoSQL bei vielen Webentwicklern sehr gefragt zu sein, was ich persönlich auch sehr gut nachvollziehen kann. Denn die Annahme, dass Daten immer schön strukturiert vorliegen, ist oft absoluter Humbug. Daher haben sich einige schlaue Köpfe Gedanken gemacht, wie man das Problem der unterschiedlichen Datenstrukturen eleganter und vor allem auch für den Entwickler deutlich einfacher lösen kann. Vielen Dank liebe schlaue Köpfe!

Angefangen hat alles mit Memcached, einem sehr einfachen System, das gerne eingesetzt wird, um betagten und langsamen MySQL-/PostgreSQL-Datenbanken unter die Arme zu greifen. Dabei handelt es sich um einen kleinen unscheinbaren Key/Value Store, der, wie sein Name schon verrät, alles im RAM zwischenspeichert. Das kann aber bei sehr großen Datenmengen ziemlich teuer werden.

Die Einfachheit von Key/Value Stores führt aber auch zu einem Komfortverlust. Denn wenn man es gewohnt ist, dass einem die SQL-Datenbank alles abnimmt und man sich um nichts mehr kümmern muss, rostet das Hirn ein. Jetzt ist es zwar so, dass ein Key/Value Store vielleicht unter dem Key „blah“ den Wert „blubb“ ablegt, aber wirklich komfortabel arbeiten kann man damit, im Vergleich zu einer relationalen Datenbank, (noch) nicht.

Und genau an diesem Punkt kommt nun MongoDB ins Spiel!

In Abb. 1.1 kann man sehr schön erkennen, wo genau sich MongoDB selbst sieht. Deutlich komfortabler als Key/Value Stores, aber trotzdem noch in der Lage zu skalieren. Und genau das ist auch das Erfolgsgeheimnis dahinter. Überfrachtet man das Datenbanksystem nicht mit unzähligen Funktionen, hat man nachher auch die Möglichkeit, einfach und schnell zu skalieren. Aber was ist MongoDB jetzt genau?

> **KOMPAKT:** MongoDB ist eine dokumentenorientierte Datenbank, die genial einfach skaliert, hoch performant ist und deine Art des Entwickelns verändern wird. Hoffentlich.

Über eine integrierte Query-Funktion kann man strukturiert Datensätze abfragen, ändern oder löschen. Dies ist auch der große Vorteil von MongoDB gegenüber anderen dokumentenbasierten Systemen wie CouchDB. Denn dort muss man sich nämlich mit Map und Reduce auseinandersetzen, um an seine Daten zu kommen. Um es vorwegzunehmen,

dieses Buch hat nicht vor, einen Glaubenskrieg zwischen MongoDB und CouchDB Fans anzuzetteln, aber ein bisschen Werbung wird man doch machen dürfen?

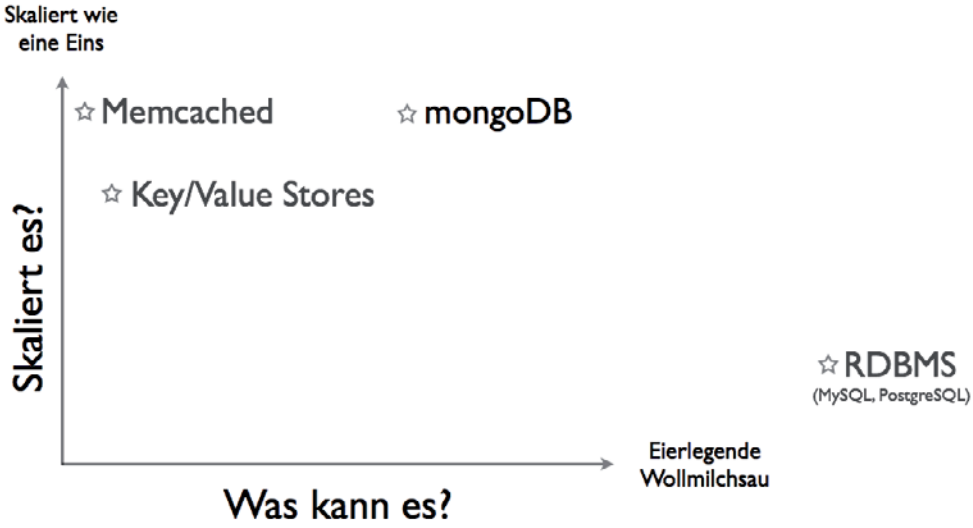

Abbildung 1.1: Wo ist MongoDB einzuordnen?

Grob gesagt kann man mit ein bisschen Lust am Experimentieren fast jede Anwendung, die auf einem normalen RDBMS aufsetzt, auch auf MongoDB portieren. Allerdings muss man dazu die eigene Denkweise und einige Erfahrungen, die man bereits im SQL-Bereich gesammelt hat, über Bord werfen. Am Anfang habe ich den Fehler gemacht, eine Webapp 1 : 1 von MySQL auf MongoDB umzustellen. Nicht besonders schlau, denn das ging mächtig in die Hose, da ich noch MySQL gedacht aber MongoDB entwickelt habe. Sinnvoll ist es, nicht gleich das Fünf-Mannjahre-Projekt umzustellen, sondern mit einer kleinen Sandkastenanwendung zu starten. Dadurch sieht man, wie man mit MongoDB klarkommt und sammelt vorweg hilfreiche Erfahrungen.

Ist erst einmal alles umgestellt, kann man sich eigentlich fast schon zurücklehnen, denn Skalierung und Replikation sind in MongoDB bereits integriert und können mit ein paar kleinen und einfachen Handgriffen konfiguriert werden. Wie genau das funktioniert, verrate ich im hinteren Teil des Buchs.

Alle Eigenschaften noch einmal im Schnelldurchlauf:

KOMPAKT:

– Dokumentenorientiert auf Basis von BSON (Binary JSON)
– Integrierte Query Language für einfache und komplexe Abfragen
– Replikation und Sharding bereits an Bord
– Unterstützung für eine Vielzahl von Indizes
– Open Source (GNU AGPL v3.0)
– Kommerzieller Support verfügbar

1.2 Wer steckt hinter MongoDB?

Wenn ich meine Anwendung jetzt umstelle, wer garantiert mir, dass es MongoDB morgen noch gibt? Diese und weitere Fragen möchte ich nun etwas genauer beantworten.

Das Unternehmen

Hinter MongoDB steckt das Unternehmen 10gen[1] aus den USA. 10gen wurde 2007 von Dwight Merriman und Eliot Horowitz gegründet. Dwight war davor CTO bei DoubleClick und Eliot war Mitgründer von ShopWiki.com (das ebenfalls MongoDB einsetzt – oh Wunder, oh Wunder). Man sieht also, dass beide wissen, wovon sie sprechen. Dabei sind es nicht nur Dwight und Eliot, die MongoDB entwickeln, sondern 10gen besteht aus einem Team von vielen begeisterten Entwicklern, die neben ihrem Entwicklungsjob auch noch einen unglaublich guten Support liefern. Mehr dazu aber in Kapitel 8.

Open Source und Licensing

Die Codebasis des MongoDB Servers wurde von Anfang an als Open-Source-Lösung unter der GNU AGPL v3.0[2] zur Verfügung gestellt. Client Libraries, mit denen man verschiedene Programmiersprachen an MongoDB andocken kann, sind unter der Apache License v2.0[3] veröffentlicht. Diese verpflichtet mich nur, Änderungen öffentlich zu machen, die ich an der Datenbank selbst vorgenommen habe. Den Sourcecode meiner Applikation muss ich nicht der Öffentlichkeit zugänglich machen, auch wenn diese sich vielleicht darüber freuen könnte.

Datenschutz

Da 10gen MongoDB entwickelt, aber nicht selbst als gehosteten Service anbietet, liegen die Daten genau da, wo ich sie haben möchte. Nämlich bei mir als Betreiber. Somit sollten

1 Web: *10gen.com*

2 GNU AGPL v3.0 im Web: *bit.ly/agplv3* | *www.gnu.org/licenses/agpl-3.0.html*

3 Apache License v2.0 im Web: *bit.ly/a_license_2* | *www.apache.org/licenses/LICENSE-2.0.html*

auch paranoide IT-Datenschutzbeauftragte nachts in Ruhe schlafen können. Die beliebte Funktion von closed-source-Software, nämlich das „nach-hause-telefonieren“, gibt es in MongoDB nicht. Und wenn es sie geben würde, hätten ein paar externe fleißige Entwickler sie längst aus dem Sourcecode entfernt. Denn dieser ist öffentlich. Fazit: Aufatmen und wohlfühlen.

Kommerzieller Support

Wer unternehmenskritische Anwendungen auf Basis von MongoDB betreiben möchte, hat die Möglichkeit, kommerziellen Support von 10gen zu beziehen. Auch hierzu mehr in Kapitel 8 „MongoDB SOS“. Aber eines vorweg: Man muss nicht immer alles selbst machen. Lieber mal etwas machen lassen und in der gewonnenen Zeit eine Party feiern.

1.3 Wer setzt MongoDB ein?

Die Liste der weltweiten MongoDB-Installationen in einer Produktivumgebung ist ziemlich lang. Um es ein wenig abzukürzen, habe ich in Tabelle 1.1 ein paar bekannte Namen herausgepickt, die man kennen könnte. Die gesamte Liste findest du hier[4].

Name des Unternehmens	Einsatzzweck/verwendete Funktionen
Foursquare	Geolocation, Capped Collection
Bit.ly	Userhistory speichern und auswerten
Etsy	Speichern, abfragen, auswerten von favorisierten Produkten
New York Times	Formulargenerator für Bilderupload
GitHub	Internes Reporting-System
Justin.tv	Usertracking und Analytics
ShopWiki	Speichern riesiger Mengen von Produktdaten
Diaspora	Der Facebook-Killer?

Tabelle 1.1: MongoDB im Produktionseinsatz

Wie man sieht, setzen auch größere Unternehmen MongoDB ein. Zwar ist die Datenbank, im Vergleich zu Urgesteinen wie MySQL oder Oracle, noch relativ jung, aber ich kann sie für den Einsatz in Produktionsumgebungen besten Gewissens empfehlen. Denn eine große Entwicklercommunity hat MongoDB bisher schon intensiv auf Herz und Nieren geprüft.

Besonders empfehlen kann ich aber auch die Präsentationen der einzelnen Unternehmen, die MongoDB einsetzen. Diese sind unter Fußnote 4 aufgelistet. Dort gibt es einen Hau-

4 Wer nutzt MongoDB: *bit.ly/mongodb_users* | *www.mongodb.org/display/DOCS/Production+Deployments*

fen Praxisbeispiele zum Einsatz von MongoDB in der Produktivumgebung. Braucht man noch Argumente, um seinen Chef von MongoDB zu überzeugen? Kein Problem. Auch hierfür sind in den jeweiligen Präsentationen genügend Anregungen vorhanden. Darüber hinaus bietet es sich an, direkt mit den Entwicklern des jeweiligen Unternehmes/Startups in Kontakt zu treten, falls man noch spezielle Fragen zu MongoDB unter Last hat. Die E-Mail-Adresse findet sich meist auf der ersten oder letzten Folie der jeweiligen Präsentation.

1.4 Für wen ist MongoDB geeignet

1.4.1 Aus technischer Sicht

Der wohl wichtigste Punkt beim Evaluieren einer neuen Datenbank ist mit Sicherheit die technische Sicht auf das System. In diesem Kapitel möchte ich dir ein paar Punkte vorstellen, mit denen du entscheiden kannst, ob MongoDB die richtige Datenbank für dich ist.

Wo möchte ich MongoDB einsetzen?

Generell kann man sagen, dass MongoDB zwar mit fast jedem Typ von Applikation, egal ob Web, Desktop oder Mobile, zurechtkommt, aber ob dies wirklich Sinn macht, steht auf einem anderen Blatt. Der häufigste Einsatzbereich sind sicherlich Web– und Mobile-Apps. Denn die Libraries, mit denen man auf die Datenbank zugreifen kann, sind primär für die populärsten Skriptsprachen wie Ruby, Python und PHP sowie viele weitere verfügbar. Also das perfekte Fressen für einige dahergelaufene IT-Consultants, die gerade Software as a Service und Cloud Computing als die goldene Zukunft anpreisen. Ziemlich klasse ist, dass MongoDB die Daten in einem JSON-ähnlichen Format ablegt und auch über ein solches die Daten zur Verfügung stellt. Damit ist es quasi für den Einsatz in Web-Apps geboren worden, oder?

Meine Datenstrukturen ändern sich oft?

Bei MySQL, PostgreSQL und anderen RDBMS ist man gewohnt, jede Änderung an der Struktur mühsam über Migrationsskripte durchführen zu müssen. Oft werden nämlich die Daten mit viel Aufwand aus dem alten Schema in das Neue überführt. Eine wirklich ätzende Arbeit. Dieser unnötige Mehraufwand entfällt mit MongoDB. Denn man ist an kein festes Schema gebunden. Super, Freiheit für alle! Wenn ich kurzerhand ein neues Feld (zu vergleichen mit einer Tabellenspalte) hinzufügen möchte, kann ich dies einfach in meinem Programmcode vornehmen. Und sofort werden neue Dokumente mit dem zusätzlichen Feld angelegt. Datensätze, die schon in der DB existieren, müssen dabei nicht aufwändig konvertiert werden (gilt natürlich nicht, wenn ich ein Feld umbenenne). Natürlich hat dies auch einen Nachteil. Man ist gezwungen, Ordnung im nicht vorhandenen Schemawald zu halten. Lege ich sämtliche Arten von Informationen wild durcheinander

in einer Collection (Infos dazu in Kapitel 3.1) ab, entsteht ein heilloses Chaos. Daher wird hier an die Sorgfältigkeit des Entwicklers appelliert.

MEINUNG: Deshalb immer dran denken: Es zählen nicht nur die inneren Werte. Pimp your code and make it beautiful!

Verschachtelte Daten oder Referenzenchaos?

Wer kennt es nicht? Mit der Zeit wird jedes Datenbankschema zu einer Schlacht aus Fremdschlüsseln und Abhängigkeiten. Jede Kleinstinformation wird in eine extra Tabelle ausgelagert, um durch den Normalisierungs-TÜV zu kommen. Zwar geht der Trend, bei größeren Anwendungen mit viel Last, hin zu einer gewissen Denormalisierung, aber hier ist es extrem schwer, die Daten konsistent zu halten. Durch den Einsatz von Embedded Documents (Kapitel 3.1) kann man viel Zeit und Nerven sparen, indem man gewisse Zusatzinformationen direkt in das Hauptdokument einbettet. Dies spart zusätzlich noch wertvolle Bytes. Also das Paradies für alle schwäbischen Entwickler.

Ist Datensicherheit und Skalierbarkeit für mich wichtig?

Lass mich raten, dir schwebt gerade ein „ja“ vor? Denn für wen ist es nicht wichtig, dass seine Daten sicher und konsistent sind? Aber es gibt auch Anwendungsfälle, in denen eine Replikation von Daten nicht unbedingt notwendig oder absolut unsinnig ist. Beispielsweise wenn man an das Logging von nicht kritischen Aktionen denkt. Diese Log-Einträge müssen nicht unbedingt auf drei Hochleistungsserver mit NSA-Sicherheitsrichtlinien verteilt werden. Wer aber auf großes Wachstum und integrierte Datensicherheit setzt, sollte sich unbedingt die Begriffe „Replica Sets“ und „Sharding“ (Kapitel 6.2 und 6.3) genauer anschauen. Denn genau für solche Herausforderungen ist MongoDB konzipiert worden.

MEINUNG: Skalierst du noch oder nutzt du schon MongoDB?

Bist du Datenjäger oder -sammler?

Die Kosten pro GB Festplattenplatz fallen von Quartal zu Quartal. Früher war man über jedes KB freien Speicher froh. Heute speichert man aber gerne die ein oder andere unnötige Extrainformation mit ab. Man hat’s ja. Problem an der Sache ist nicht das Speichern der Daten, sondern die spätere Auswertung. Denn riesige Datenmengen fragt man nicht einfach mal eben so ab, ohne vorher einen Index gesetzt zu haben. Indizes kosten aber unnötige Zusatzressourcen. Man spricht bei Indizes auch von einem seltenen und sehr wertvollen Rohstoff. Daher greift man bei solchen Problemstellungen gerne auf das Map-and-Reduce-Verfahren (Kapitel 4) zurück. MongoDB bietet nämlich die Möglichkeit, beliebig viele Server zusammenzuschalten, um diese dann gleichmäßig zu befüllen (Stich-

wort: Sharding). Zum Abfragen dieser Daten bieten die Client Libraries von MongoDB ein Map-and-Reduce-Interface an. Damit kann eine Suchanfrage meist parallel auf mehreren Servern abgearbeitet werden und man erhält am Ende ein wunderschönes Ergebnis zu seiner Anfrage. Getreu dem Motto: Lass viele kleine Computer für dich arbeiten und fasse am Ende nur noch alles zu einem übersichtlichen Ergebnis zusammen.

Binärdateien in der Datenbank ablegen?

Ja ich weiß, für Dateien gibt es das Dateisystem. Heißt ja nicht umsonst so. Aber wenn ein Fileserver vom Platz her nicht mehr ausreicht, habe ich doch ein Problem, oder? Wie kann ich denn eine Datei über mehrere Server und Rechenzentren replizieren, ohne das Rad neu erfinden zu müssen? Ganz einfach. Nämlich mit der GridFS Implementation (Kapitel 4.3 steht dazu Rede und Antwort) in MongoDB. Damit kannst du ein einfaches, serverübergreifendes Dateisystem aufsetzen, dass perfekt für das Ausliefern und Archivieren von Dateien geeignet ist.

Gerade wenn dir Amazon S3 zu teuer ist, solltest du dir GridFS genauer anschauen. Denn mit einfacher Standardhardware lassen sich schon beeindruckende Benchmark-Ergebnisse (z. B. Messung der Requests pro Sekunde) erzielen.

Geodaten speichern und auswerten?

Geolocation Services sind doch der letzte Schrei? Jeder baut eine Anwendung, mit der man die aktuelle Position der eigene Toilette speichern und twittern kann. Doch was mache ich später mit diesen Informationen? Wie kann ich z. B. die Top-10-Toiletten in meiner direkten Umgebung finden? Mit dem Geospatial Indexing in MongoDB können Daten als Geodaten klassifiziert und für Geo-basierte Abfragen (z. B. Umkreissuche) genutzt werden.

Auswertung

Kann die Mehrzahl der Fragen mit „ja“ beantwortet werden, hat MongoDB gewonnen. Ansonsten einfach nochmal von vorne beginnen und die Fragen anders beantworten. Hier aber nochmal alle Fragen in der Übersicht.

- Soll die Datenbank für eine Web-/Mobile-App sein?
- Ändere ich häufig meine Datenstruktur/Schema?
- Nutze ich viele Referenzen/Fremdschlüssel?
- Benötige ich Datensicherheit und Out-of-the-box-Skalierbarkeit?
- Speichere ich Unmengen an Daten und möchte diese Auswerten?
- Sollen meine Daten/Dateien über mehrere Server verteilt werden?
- Ist das Speichern und Abfragen von Geodaten interessant für mich?

1.4.2 Aus ökonomischer Sicht

Natürlich spielt der finanzielle Aspekt bei der Auswahl der Datenbank eine große Rolle – Finanzkrise und so. Aber auf der anderen Seite musste ich mich auch schon oft mit Aussagen wie „was nichts koscht isch au nichts" herumschlagen. Eines kann ich aber vorweg nehmen. Diese Aussage trifft bei MongoDB nicht zu. Es gibt ein paar wichtige Fragen, die man sich aber trotzdem vor dem Einsatz von MongoDB stellen sollte.

Ich hatte nämlich vor kurzer Zeit in einem Webcast Dwight Merriman, CEO von 10gen gefragt, wie sie eigentlich Geld verdienen wollen, wenn doch MongoDB Open Source und damit kostenlos verfügbar ist. Zwar dachte ich mir, dass die Antwort „Support und Trainings" heißen würde, aber war mir unsicher, da doch in der MongoDB Google Group alle Fragen vom Entwicklerteam kostenlos und kompetent beantwortet werden. Seine Aussage schien mir jedoch plausibel, denn er meinte, dass viele größere Unternehmen einen zuverlässigen Partner bräuchten, der sich um einen störungsfreien Betrieb kümmert. Diese „großen Fische" haben nämlich keine Lust, lange herumzufrickeln, um die perfekte Konfiguration herauszufinden, sie wollen sich auf ihre Hauptaufgabe konzentrieren. Daher ist ein kompetenter Partner, der sich um das Drumherum kümmert, extrem wichtig. Ebenfalls sind Trainings ein wichtiger Punkt, damit 10gen seinen Kühlschrank füllen kann.

Quizfrage: Brauche ich Trainings oder kostenpflichtigen Support?

Auch wenn ich der Meinung bin, dass man mit ein wenig herumexperimentieren MongoDB auf eigene Faust gut erforschen kann, sind doch viele Unternehmen an professionellen Trainings interessiert. Der Vorteil besteht darin, dass man sich einen Haufen Zeit sparen und genau die Fragestellungen behandeln kann, die für das aktuelle Projekt interessant sind. Soll die Anwendung reibungslos laufen und eventuell skaliert werden, so bietet es sich an, das aktuelle Szenario zusammen mit den Chefdenkern von MongoDB durchzusprechen. Meiner Meinung nach ist dies gut investiertes Geld – und sichert die MongoDB-Weiterentwicklung. Schließlich spart man sich auch einen großen Batzen an Lizenzkosten. Um es vorwegzunehmen, ich verdiene leider keine Provision mit dieser Schleichwerbung.

Wie teuer ist die Hardware, auf der MongoDB sich wohl fühlt?

Diese Frage kann ich nur teilweise beantworten, da jeder Anwendungsfall unterschiedliche Hardware voraussetzt. Jedoch sagte Eliot Horowitz in einer Präsentation einmal, dass MongoDB für das horizontale und nicht das vertikale Skalieren (nicht Gewerbe) konzipiert sei. Damit meinte er, dass, wenn man mehr Leistung und Speicherplatz braucht, man nicht einen größeren und damit teureren Server hinstellen soll (vertikale Skalierung). Sondern MongoDB verfolgt das Google-Prinzip: Viele kleine und kostengünstige Server sinnvoll miteinander vernetzt, sind effizienter als ein teurer großer Server. Macht Sinn, oder? Dies nennt man auch das Prinzip des horizontalen Skalierens. Der Grund, wieso viele zu einem größeren und teureren Server neigen, um mehr Durchsatz zu erzielen, liegt

in der Tatsache, dass man gewohnt ist, pro Server Lizenzgebühren abzudrücken. Dies entfällt glücklicherweise bei MongoDB.

Als Faustregel gilt aber, dass eine schnelle Festplatte mit niedriger I/O-Zugriffszeit (optimalerweise SAS HDDs) und so viel RAM wie das Budget hergibt, die Performance deutlich erhöhen. Multicore-CPUs können durch das Starten mehrere Datenbankinstanzen auf einem Server optimal ausgenutzt werden.

Die Suche nach dem perfekten Entwickler/Admin

Dieser Kostenblock ist meiner Meinung nach nicht zu unterschätzen. Denn MongoDB ist eine relativ neue Technologie, bei der es hier und da noch an der nötigen Erfahrung fehlt.

Zwar kann man mit den Größen Zeit und Begeisterung Mitarbeiter für MongoDB heiß machen und entsprechend schulen, aber optimal ist es, wenn man eine treibende Kraft an Bord hat, die MongoDB schon in der Webwildnis erfolgreich eingesetzt hat. Man sollte nicht vergessen, dass ein Datenbankserver, den man lokal betreibt, nicht mit einem Sharding Cluster in der Produktion vergleichbar ist. Daher lieber die Kaffeekasse plündern und ein bisschen mehr für einen guten Entwickler ausgeben!

Fazit

Auch Open Source kostet. Die Kosten sind zwar meist geringer als die für kommerzielle Software, aber man sollte sich vorher genau überlegen, für was man Geld ausgeben möchte. Hier nochmal eine kurze Zusammenfassung:

KOMPAKT:

- Kosten für kommerzielle Trainings/Support ins Budget mit einplanen.
- Lieber auf viele günstige, als auf wenige teure Server setzen.
- Einsparungen durch Open Source in fähige Entwickler/Schulungen investieren.
- Kommerzieller Support tut nicht weh und macht dann Sinn, wenn man nicht weiterkommen oder die Anwendung optimieren möchte.

1.4.3 Aus „Just For Fun“-Sicht

Dies ist meine persönliche Lieblingssicht. Denn so kann man am einfachsten mit MongoDB starten. Viele kleine Projekte haben mir über die Zeit hinweg geholfen, die Funktionsweise von MongoDB zu verstehen. Auch gibt es einige Open-Source-Projekte, die auf MongoDB basieren und von denen man viel lernen kann. Dazu genügt es, einfach mal GitHub zu durchforsten[5].

5 GitHub MongoDB Projekte: *bit.ly/github_mongodb* | *github.com/search?&q=mongodb&type=Repositories*

Im Folgenden habe ich ein paar Fragen zusammengestellt, mit denen du das bevorstehende Projekt genauer einschätzen kannst. Unter jeder Frage sind ein paar Beispielideen für kleine Projekte, mit denen du die jeweilige MongoDB-Funktion erlernen kannst. Hört sich komisch an, ist es aber nicht. Allerdings macht es erst Sinn, damit anzufangen, wenn die Grundlagen sitzen.

Ich erfasse große Datenmengen und soll diese auswerten?

Dies könnte ein klassisches Tracking-/Log-Analyseprojekt sein. Man speichert z. B. jede Pageview ab und erstellt später via Map and Reduce eine Auswertung, wer wann was aufgerufen hat und wie lange er auf der jeweiligen Seite geblieben ist. Quasi Google Analytics für Arme. Dazu bietet es sich an, ein kleines Skript zu schreiben, das die Visits mit einem Timestamp und den HTTP-Request Informationen in die DB schreibt. Später kann man dann z. B. nach Uhrzeit gefiltert abfragen, welcher User welche Seite aufgerufen hat und wie lange er geblieben ist. Die Grundlagen zum Erzeugen von Datensätzen in MongoDB mithilfe einer Skriptsprache kannst du in Kapitel 5 nachlesen. Das Abfragen via Map and Reduce ist in Kapitel 4 genauer erklärt.

Ich habe strukturierte Daten, die ich schnell durchsuchen möchte?

Hier ist es besonders wichtig, die genaue Funktionsweise von Indizes kennen zu lernen. Dazu mehr in Kapitel 4.1. Mein erstes Projekt in diesem Bereich habe ich mithilfe der Amazon Product Advertising API[6] gebastelt. Es ging darum, ca. 600 000 Produktdaten über die API einzulesen und später nach Kriterien wie Kategorie, Preis und Userbewertungen zu selektieren. Wenn die Daten einigermaßen strukturiert vorliegen und Indizes sinnvoll gesetzt sind, kannst du über die komfortable Query Language von MongoDB schnell auf die benötigten Informationen zugreifen. In Kapitel 4 sind sämtliche Query-Operatoren vorgestellt.

Ich möchte Skalierung via Sharding genauer unter die Lupe nehmen?

Zuerst ist es sinnvoll, Sharding und ggf. eine Replikation zu konfigurieren. Dies kannst du anhand von Kapitel 6.2 und 6.3 vornehmen. Keine Sorge, der erste Versuch ist immer etwas zäh. Aber danach läuft es wie geschmiert. Nun ist das System soweit vorkonfiguriert, aber es fehlen Daten zum Testen. Hier würde ich mir keine große Mühe geben und ein kleines Skript basteln, das Zufallszahlen und -wörter generiert und diese in eine Datenbank schreibt. Es ist jedoch zu beachten, dass die Datenmenge mehrere hundert MB erreichen sollte, damit das Sharding auch in Kraft tritt. Wieso, wird in Kapitel 6.3 genauer erklärt. Hat die Datenbank nun eine gewisse Größe erreicht, schaltet man einfach einen Datenbankserver im Cluster ab. Nun sollte MongoDB ein automatisches Failover durchführen und ohne einen Fehler weiterarbeiten. Auch sollte die Größe einer Datenbank in

6 Amazon Advertising API: *bit.ly/product_api* | *affiliate-program.amazon.com/gp/advertising/api/detail/main.html*

einem einzelnen Replica Set (Replikationsverbund) weniger sein, als die Gesamtgröße der Collection über alle Replica-Sets hinweg. Ist dies gegeben, scheint das Sharding zu funktionieren, denn dann verteilt MongoDB die Daten brav über alle Shards hinweg.

Ich möchte mit Geodaten arbeiten?

Hier bietet es sich an, im einem Team mit Google Maps zu spielen. Man nehme die Postleitzahlen der amerikanischen Städte[7] und lade sie in MongoDB. Nun setzt man einen Geospatial-Index auf das Feld, in dem die Koordinaten hinterlegt sind und kann mit dem Abfragen beginnen. Zum Beispiel zeige mir alle Städte im Umkreis von 200 km rund um New York auf einer Google Map an. Ein faszinierendes Pixelkino.

Natürlich versteht man im Moment nur Hüttenkäse, aber am Ende des Buchs kann man sich noch einmal mit diesem Miniprojekt befassen und dann sollte alles klarer sein als im Moment. Matthias Stearn, Entwickler bei 10gen, hat einen Spickzettel[8] für das obige Projekt geschrieben. Diesen und seine Videos zu der Idee findest du ebenfalls im Web[9].

Ideen und Anregungen

KOMPAKT:

- Viele Daten, die keine Liveabfrage benötigen, können mit Map and Reduce abgefragt werden.
- Strukturierte Daten, bei denen schnelle Abfragezeiten entscheidend sind, können anhand von sauber gesetzten Indizes und der MongoDB Query Language ausgelesen werden. Dazu einfach die Amazon Product Advertising API anzapfen.
- Skalierung und Replikation können mit einem Skript, das viele Zufallsdaten generiert, getestet werden. Deaktiviert man einen Datenbankserver, sollte Failover eingreifen.
- Geodaten werden mithilfe von Geospatial-Indizes verarbeitet und können zur Übersicht z. B. in einer Google Map dargestellt werden.

1.5 MySQL vs. MongoDB – Let the fight begin

Um es gleich vorweg zu nehmen. Jede Datenbank hat ihre Berechtigung. Die eine mehr, die andere weniger. Im Folgenden möchte ich kurz ein paar Unterschiede zwischen MySQL und MongoDB aufzeigen. Keine Gewähr auf Vollständigkeit/Richtigkeit und Objektivität.

7 Die Postleitzahlen amerikanischer Städte als JSON: *bit.ly/zips_json* | *media.mongodb.org/zips.json*

8 Der Spickzettel im Web: *bit.ly/geo_mongodb* | *github.com/RedBeard0531/Mongo_Presentations/blob/master/20100521-mongony/geospatial_script.js*

9 Präsentation zum Projekt:
Teil 1: *bit.ly/geo_video_a* | *lacantine.ubicast.eu/videos/21-06-2010-164138-partie-2/*
Teil 2: *bit.ly/geo_video_b* | *lacantine.ubicast.eu/videos/21-06-2010-173743-partie-1/*

Verbindung und Authentifizierung

Beide Datenbanksysteme kommunizieren über eine TCP-/IP-Socket-Verbindung. Zwar kann man MongoDB mit etwas Aufwand und Friemelei als embedded-Datenbank betreiben, aber davon rate ich momentan noch ab. Viel zu viel Aufwand. Außerdem gibt es dafür (noch) SQLite.

MySQL und MongoDB bieten jeweils ein eigenes Authentifizierungssystem an. Bei MySQL ist dieses ausgereifter und granularer aufgebaut. So kann man dort den Zugriff auf Tabellenebene festlegen, wohingegen MongoDB nur auf Datenbankebene authentifiziert. Daher sollte man den Zugriff auf MongoDB mit iptables oder einer anderen Firewall-Lösung einschränken. Dies bietet zusätzliche Sicherheit.

Schema und Referenzen

Dass MongoDB kein festes Schema besitzt, hatte ich schon angesprochen. Dies spart Zeit in der Entwicklungsphase, da man nicht ständig Daten migrieren und keine Workarounds für Datenstrukturen schaffen muss, die nicht auf Anhieb in ein RDBMS passen. Man neigt aber auch zu Unordentlichkeit, wenn kein festes Schema definiert werden muss. Bei MySQL ist die Sache etwas restriktiver. Es wird nur das gespeichert, was auch vorher definiert wurde.

Referenzen zwischen Tabellen, die über Fremdschlüssel realisiert werden, sorgen in MySQL für eine konsistente Datenbasis. Einmal festgelegt, meckert MySQL wenn ein Datensatz die Integrität verletzt. MongoDB bietet zwar mit DBRef auch einen Referenzdatentyp an, jedoch kommt dieser in der Praxis kaum zum Einsatz. Daher sollte man darauf achten, dass Referenzen stets als ObjectID in MongoDB (mehr dazu in Kapitel 3.1) gespeichert werden.

Sonderfall: Stored Procedures und Trigger

Mit Stored Procedures kann SQL direkt im MySQL Server ausgeführt werden. Allerdings sind Stored Procedures alles andere als wartungsfreundlich und einfach zu entwickeln. MongoDB bietet einen ähnlichen Ansatz. JavaScript-Code kann serverseitig gespeichert und nach Belieben serverseitig ausgeführt werden. Somit kann man eine Helper-JavaScript-Funktion schreiben und diese bei Bedarf auf dem Server ausführen lassen. Dies hat den Vorteil, dass die ganzen Dokumente nicht erst zum Client übertragen werden müssen, um dort vom Programm ausgewertet zu werden. Sondern die Dokumente werden serverseitig beackert.

MySQL's Trigger helfen einem, bestimmte Aktionen auszuführen, wenn gewisse Bedingungen eingetreten sind. Ein Beispiel wäre, dass bei jedem neuen Datensatz, der angelegt wird, eine Cleanup-Funktion hinterher irgendetwas aufräumt. Diese Funktionalität bietet leider nur MySQL an. Bei MongoDB müsste man im Client gezielt eine Funktion aufrufen, wenn die gewünschte Bedingung eingetreten ist. Also nach dem Erzeugen eines neuen Dokuments würde dann automatisch die Cleanup-Funktion aufgerufen. Nachteil MongoDB.

Funktion	MySQL	MongoDB
Open Source	Ja, GNU GPL v2.0 + LE	Ja, GNU AGPL v3.0
Kommunikationsprotokoll	TCP/IP	TCP/IP
Standard Port	3306	27017 28018 (HTTP Interface)
Datenstruktur	Spaltenorientiert	Dokumentenorientiert
Untereinheit einer Datenbank	Tabelle	Collection
Primärschlüssel	Primary Key	ObjectID
Basic-/Unique-/Compound Indexes	Ja/Ja/Ja	Ja/Ja/Ja
Transaktionen	Ja	Nein, jedoch atomare Updates
Stored Procedures	Ja	Jain, JavaScript kann in Server ausgeführt werden
Trigger	Ja	Nein
Cursor	Ja	Ja
Views	Ja	Evt. über Map and Reduce
Volltextsuche	Ja	Nein, nur Regex möglich
Sharding	Ja, mit externen Hilfsmitteln	Ja, Out of the box
Replikation	Master/Slave (Standard)	Replica Sets (ein Master, n-Slaves)
Austauschbare Storage Engines	Ja, z. B. InnoDB, MyISAM	Nein
Export/Import/Backup	*mysqldump*	*mongodump, mongoexport*
Commandline Client	*mysql*	*mongo*
Sourcecode/vorkompiliert	Ja/Ja (Mac, Windows, Linux)	Ja/Ja (Mac, Windows, Linux)
Webfrontend	PHPMyAdmin	Fang-of-Mongo[1] und weitere
Speichern von Binärdaten	Ja, BLOB	Ja, GridFS oder als Byte-String
Ruby, Python, PHP, Java Client Libraries	Ja	Ja
Zahlreiche Funktionen in Query Language, z. B. STRLEN()	Ja[2]	Nein, höchstens in einer serverseitigen JS-Funktion

1 Fang-of-Mongo Webfrontend: *bit.ly/fang_of_mongo* | *github.com/Fiedzia/Fang-of-Mongo*

2 MySQLFunktionsübersicht: *bit.ly/mysql_functions* | *dev.mysql.com/doc/refman/5.0/en/string-functions.htm*

Funktion	MySQL	MongoDB
Authentifikation	Ja	Ja
Map and Reduce	Nein	Ja
Gutes Gefühl beim Entwickeln	Sehr selten	Immer

Tabelle 1.2: Vergleich zwischen MySQL und MongoDB

Fazit

Wie in Tabelle 1.2 dargestellt, gewinnt MySQL eindeutig, wenn es um zusätzliche Funktionen geht. Auf diese verzichtet MongoDB relativ oft und bietet somit die Möglichkeit der einfachen Replikation und des Out of the box Shardings. Daher sollte man sich vorher Gedanken machen, auf was die zu entwickelnde Applikation abzielt.

Für Webapps, die auf einfach strukturierten Daten aufsetzen und bei denen sich Last und Datenmenge in Grenzen halten, sollte man sich MongoDB genauer anschauen.

Möchte man dagegen eher eine skalierbare Webapp entwickeln, deren Schema ohne großen Aufwand von Zeit zu Zeit angepasst werden kann, empfiehlt es sich auf MongoDB zu setzen. Ich wäre doch unglaubwürdig, wenn ich dir hier zu einer MySQL-Datenbank raten würde. Ein Mercedes-Händler verkauft auch nicht freiwillig einen Lada.