UNIVERSITY OF TRENTO - Italy

Department of Information Engineering and Computer Science

Bachelor's Degree in Computer Science

Final Dissertation

Evaluating MongoDb Performace:

How and where NoSQL databases are getting over Relational Databases

Supervisor
Alberto Montresor

Student
Michele Romani

Academic year 2015/2016

Acknowledgements

In my experience, everything we do in our lives has some kind of contribution from the people that live around us at home, at school, at work, everywhere. Unless you are a hermit, of course. And there are different kinds of support people can give you, depending on the relation you have with them. On a more affective side, I'd like to thank my family for always believing in me and in what I do, and supporting me in my choices. My girlfriend and my friends, for their support and the great time I always spend with them and that helps me recovering from the overwhelming amount of commitments of every day life. And my course mates, for all the time we studied together sharing knowledge and useful suggestions. On the professional side, first I'd like to express my deepest gratitude to my Supervisor and Professor Alberto Montresor for the passion he transmitted me while teaching Algorithms on the second year, for involving me in interesting projects and for his supervision in this last work of my bachelor degree. I'd also like to thank my Erasmus Professor Erki Eassar for his skill in making me appreciate the course of Databases and for sharing me useful suggestions and materials even months after being his student in Tallinn University of Technology. At last, I'd like to acknowledge my internship tutors in Tai, Andrea Carpineti and David Votino, for their technical and motivational suggestions about the project from which this dissertation has origin, and my colleague Bruno Graziano for his precious help in configuring and maintaining the system of virtual machines that hosted my clients and my Mongo nodes for the main tests. Have a good reading.

Contents

Summary

The IT world is evolving faster and faster every year, with new breaking technologies coming to our lives, even changing our way to communicate and live in our society. With the advent of social networks, cloud storage and computing, a new definition for the amount of data they involve has been coined: Big Data[1]. The challenge of Big Data involves both developing better retrieving solutions using advanced data mining techniques and functional storage solutions. Several companies are switching their old systems and technologies to more scalable and reliable solutions to optimize their costs in terms of time and money, improving their profits. The company in which I am actually working entrusted me to develop a software in order to evaluate MongoDB [2], a new non-relational database technology in anticipation of a new contract from a customer that needs to support an application with several hundred thousand of users and millions of records. The challenge is to obtain acceptable results from Mongo in stressing conditions like a production software: retrieving data in less than 2 seconds, preventing loss of data and most importantly, preventing a system crash of the database. I entirely developed a Java software based on the Spring framework, following my project leader and my tutor directives, capable of launching specific benchmark tests aimed at stress-testing and maybe even crashing a virtual machine running a MongoDB instance. For my architecture used the technique of Microservices [3], that consists in building a modular application, with each module dedicated to a specific service. It is an advanced development technique that is getting more and more successful, also thanks to famous use cases such as Netflix, with the strength of easy reusability and maintainability of the software. My choice of this technique is due to a possible future experimentation of other storage technologies, even relational, as the modularity of the applications allow to quickly develop and connect a new module with drivers for other Database Management Systems. The choice of MongoDB was made by both our manager and our customer because of its ease of configuration and its availability as an open-source software. This research aims to explain many reasons why NoSQL technologies are taking over the well-known relational databases in new enterprise applications, focusing on selected use cases. In particular, I have been committed to develop a software that could perform a stress-test on MongoDB to verify if it could stand the customer requirements. The team involved 3 persons:

- me as Software Developer.
- an internal System Engineer that helped me configuring MongoDB instances on different nodes (depending on the test requirements) and configuring the virtual machines that have been used through the evaluation.
- an internal Software Engineer, my stage tutor, that helped me define the architecture of the application and choose the right frameworks for both backend and frontend. He also contributed in defining test cases and testing the functionalities of the software.

To clarify, the research does not demonstrate that NoSQL technologies are a better choice than Relational DBMS in any case, nor that the relational databases will get outdated and out of use. In fact, both of them have strenghts and weaknesses based on the situation in which they are used. The future of databases will likely involve the parallel use of different technologies or maybe a "fusion" like, for example, NewSQL databases that are currently under experimental development. But even tough relational databases are not going to disappear soon, we will explain why NoSQL are really taking over them in the highly demanding requests of the new market of Big Data challenging applications in terms

[1] https://datascience.berkeley.edu/what-is-big-data/
[2] https://www.mongodb.com/what-is-mongodb
[3] http://microservices.io/patterns/microservices.html

of high scalability, usability and performance. This will likely lead to a relegation of Relational DBMS to specific roles in a system or to specific use cases in which they still have better reliability or even better performance than NoSQL regardless their higher cost of configuration and maintainability and their restrictions as explained by the *CAP theorem* [4]. MongoDb Performcance was developed entirely in Java on the backend side, while the frontend side was developed in Html 5, Css and Javascript. It depends on several frameworks and libraries, among which the most relevant are:

- *Java Spring* [5] - The future of Java Development, based on REST calls and Annotations.
- *AngularJS* [6] - An essential web framework to build single page applications with dynamic loading of contents, used in combination with *Twitter Bootstrap* [7].
- *MetricsGraphics.js*[8] - A versatile Javascript framework based on D3, used to plot data.

The code of the project can be visualized on GitHub [9] only after authorization as its property rights are owned by the company.

[4] Also named Brewer's theorem, will be explained in chapter 2
[5] https://spring.io
[6] https://angularjs.org
[7] http://getbootstrap.com
[8] http://www.metricsgraphicsjs.org
[9] https://github.com/BRomans/mongodb-performance-app

1 Introduction

In the first two parts of this chapter a brief overview will explain the most known databases technologies while in the last part NoSQL databases will be introduced through the descrption of the most famous implementations from which many others derive.

1.1 Discovery of NoSQL technologies

Commonly students have their first encounters with database technologies during their studies in high school or bachelor degree and, to better understand all the fundamentals concepts, they are taught the basic principles of relational databases. It i's the most common choice of every school teaching the very foundation of Databases to make students understand the the meaning of *CRUD operations, relations, consistency, redundancy* [1], etc... and how to correctly set up the entities of their systems following proven patterns and constraints. Detaching from well-known developing habits is not always so simple, but it is necessary to understand why big companies such as Facebook decided to invest money in developing their own database solution, Cassandra, instead of using an existing relational database. It is important to know that there are many different ways to build a database, some are better than others in certain use cases. Nowadays, an huge amount of data are spread around the world everyday through the Web and it needs to be stored and retriewed quickly to save companies' money and give the users a perfect feeling of resposivity. But let's start from the beginning to get an overview of a technology we rely on every day, even without being aware of its presence in every single application we use.

[1] [2]

1.2 Databases

A Database is an organized collection of data even though we often use the term to refer to the entire database system. The Database Management System, or DBMS, is the name of the entire system that handles data, transactions, relations and eventually problems. The term DBMS has been replaced by RDBMS in the common language, that stands for Relational Database Management Systems, since for decades the relational model has been a standard for data storage.

1.2.1 Relational Databases

Probably the most popular and for many years also most used model, a relational database is composed by tables representing entities (users, customers, courses...) where each column represents a field or attribute and each row represents a record. Tables can have relations each other with the use of foreign keys, and each table has a primary key that is unique on each record. It's fundamental for a good design of a relational database that its schema is in *Normal Form*, following three main steps:

- *First Normal Form* :
- *Second Normal Form* :
- *Third Normal Form* :

[1] CRUD

Apart from Normal Form, to ensure that a relational database guarantees the reliability of its transactions it must ensure ACID[2] properties:

- *Atomicity* :
- *Consistency* :
- *Isolation* :
- *Duraibility* :

The most famous relational databases follow SQL syntax that stands for Structured Query Language and is a standard since 1986. Among this category, the most famous and used are MySQL, PostgreSQL, Microsoft SQL Service, Oracle Database.

1.2.2 Navigational Databases

The first generation of databases used pointers from one record to another to "navigate" the database and that's why they were called Navigational databases. The fundamental problem of this kind of database was that the user needed to know the physical structure of the database to query data from it. The only way to add an extra field was achieved only by rebuilding the storage scheme. In addition, the absence of a standardisation among vendors made those databases disappear quickly in favour of more functional choices

1.2.3 Object-oriented Databases

In the long story of the database evolution, object-oriented databases helped developing the communication between databases and programming languages, but they failed due to their bounds to a specific programming language. They offered advanced features like inheritance and polymorphism and could support a large number of data types. What is left of their inheritance in the aftermath is the implementation of drivers and bindings between databases and programming language. NoSQL technologies are a perfect example of the evolution of the idea of object-oriented databases.

1.2.4 NoSQL Databases

The last generation of databases is called NoSQL because of its detachment from the classic relational model in terms of schema and their use of query languages different than SQL. They aim to great performance and scalability, to support the increasing need of those applications that daily transfers huge amounts of data. Since the category is itself generic, and new different implementations are release every year, they can be broadly divided in those sub-categories:

- *Graph databases* : as foundation of this kind of databases there is the graph theory and the concept of nodes and edges. Each node corresponds to an entity and edges correspond to relations between them. They use an index-free adjacency that grants each element a pointer to its adjacent element and does not require the full indexing of the database.
- *Key-value stores* : thanks to simple concepte of a key assigned to each value, similarly to hastables, the model of those databases usually grants higher performance. It is even possible, depending on the database implementation, that a key could be bound to an entire collection of values.
- *Document stores* : this is the family which MongoDB belongs and they work around the concept "Document". Documents are records and they are stored into tables called collections. Usually collections don't require the same number of fields, so there could be different versions of the same document inside a collection. A great advantage of this model is the ease of data access and manipulation.

The core aspect of theri implementation is that they have no predefined schema, in addition most of them does not require their records to have the same number of fields if not enforced, also called

[2]Acid properties

Dynamic Schema [3]. They support to replication of the primary server on many other servers, like MongoDB's ReplicaSet, and this provides reliability in case of failover of one of the nodes granting no data loss in production applications. Servers execute same transactions and keep their data synchronized to eliminate any errors, and they usually write a backup copy or a snapshot of the data after any operation. As is it possible to imagine, this multi-node architecture cannot fully guarantee the respect of ACID properties and might sometimes present synchronization issues with the possible result of a secondary node becoming primary with partially outdated data. The absence of constraints on the schema allows query to be executed faster withput the need of expensive *join* operations on the collections, but when it comes to execute complex queries NoSQL databases performance fall if they are not well configured. It is then important to provide a good configuration of the architecture in order make the most of the strengths of those databases.

1.2.5 The CAP Theorem

1.3 NoSQL: a brief panoramic over the actual situation

NoSQL databases area literally spreading around, with many companies and insitutes implementing their own custom version so it would be impossible to describe them all. Most of them are new implementations of the pioneers who brought this innovating technology to the market less than 10 years ago, that are briefly described in the following part.

1.3.1 MongoDB

MongoDB is a document-oriented DBMS that uses a JSON-style documents called BSON, making data integration from certain kind of applications easier and faster. Originally developed as a component of a bigger software, it then became open source in 2009 under the supervision of MongoDB Inc. company. It offers support for many programming languages such as Java, C++, Python and many others and it's being used as backed from a large number of web sites and services like eBay, Foursquare, NYTimes among the others. As db-engines.com reports, it is the most popular NoSQL database now.

1.3.2 Google Big Table

Big Table [4] is the proprietary database system of Google, developed back in 2004 and build on Google File System [5]. It shares the characteristics both of row-oriented databases and column-oriented databases. Google decided to develop its own database with the purpose of scalability and better control over performance: in fact it's designed to support a data-load of petabytes over thousands of machines. Big Table supports many Google applications such as Reader, Maps, Books, Earth, Gmail and even YouTube. Google announced a new version called Google Cloud Bigtable[6], actually in beta, that will be distributed as public version of Big Table.

1.3.3 Apache Cassandra

Another open-source project is Apache Cassandra [7], developed at Facebook in 2007 to improve the research of the internal mail system and then entered in the Incubator project from Apache in 2008 where it begun its growth as DBMS. Like Big Tables it offers a key-value storage structure with eventual consistency. Each keys correspond to a value and all the values are grouped in families of columns. Families are defined when the database is created and Cassandra adopts an hybrid approach between DBMS oriented to columns and memorization oriented to rows. Other famous sites than Facebook that uses Cassandra are Twitter and Digg, and many benchmark tests, in terms of performance and scalability confirms Cassandra as the best NoSQL database in the current scenario.

[3] Dynamic Schema
[4] Big Table
[5] GFS
[6] cloud big table
[7] apache cassandra

1.3.4 Amazon DynamoDB

Amazon DynamoDB [8] is the proprietary database system of Amazon available for developers since 2012, build on the model of Dynamo but with a different implementation and offered as a part of the Amazon Web Services [9] portfolio. The particularity is that DynamoDB allows developer to purchase a service based on the desired throughput rather than the storage, that will be increased by the administrators of the system if needed. Many programming languages have a DynamoDB binding, including Java, Node.js, Python, Perl and C#. Most of the Amazon services uses DynamoDB as storage system.

2 The Choice

In this chapter we explain the motivations that determined MongoDB as choice for the evaluation and consequently the commission among other NoSQL possibilites. Following there is a deep description of many Mongo core features mostly taken from the official documentation, that could be a good introduction for interested users.

[8] amazon dynamp
[9] amazon web services

Bibliography

[1] Ict business. http://www.ictbusiness.it/. ultimo accesso 15/06/2015.

[2] Donoho D. L. Compressed sensing. *IEEE Trans. Inf. Theory*, 52(4):1289–1306, 2006.

Allegato A Titolo primo allegato

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sed nunc orci. Aliquam nec nisl vitae sapien pulvinar dictum quis non urna. Suspendisse at dui a erat aliquam vestibulum. Quisque ultrices pellentesque pellentesque. Pellentesque egestas quam sed blandit tempus. Sed congue nec risus posuere euismod. Maecenas ut lacus id mauris sagittis egestas a eu dui. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque at ultrices tellus. Ut eu purus eget sem iaculis ultricies sed non lorem. Curabitur gravida dui eget ex vestibulum venenatis. Phasellus gravida tellus velit, non eleifend justo lobortis eget.

A.1 Titolo

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sed nunc orci. Aliquam nec nisl vitae sapien pulvinar dictum quis non urna. Suspendisse at dui a erat aliquam vestibulum. Quisque ultrices pellentesque pellentesque. Pellentesque egestas quam sed blandit tempus. Sed congue nec risus posuere euismod. Maecenas ut lacus id mauris sagittis egestas a eu dui. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque at ultrices tellus. Ut eu purus eget sem iaculis ultricies sed non lorem. Curabitur gravida dui eget ex vestibulum venenatis. Phasellus gravida tellus velit, non eleifend justo lobortis eget.

A.1.1 Sottotitolo

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sed nunc orci. Aliquam nec nisl vitae sapien pulvinar dictum quis non urna. Suspendisse at dui a erat aliquam vestibulum. Quisque ultrices pellentesque pellentesque. Pellentesque egestas quam sed blandit tempus. Sed congue nec risus posuere euismod. Maecenas ut lacus id mauris sagittis egestas a eu dui. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque at ultrices tellus. Ut eu purus eget sem iaculis ultricies sed non lorem. Curabitur gravida dui eget ex vestibulum venenatis. Phasellus gravida tellus velit, non eleifend justo lobortis eget.

Allegato B Titolo secondo allegato

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sed nunc orci. Aliquam nec nisl vitae sapien pulvinar dictum quis non urna. Suspendisse at dui a erat aliquam vestibulum. Quisque ultrices pellentesque pellentesque. Pellentesque egestas quam sed blandit tempus. Sed congue nec risus posuere euismod. Maecenas ut lacus id mauris sagittis egestas a eu dui. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque at ultrices tellus. Ut eu purus eget sem iaculis ultricies sed non lorem. Curabitur gravida dui eget ex vestibulum venenatis. Phasellus gravida tellus velit, non eleifend justo lobortis eget.

B.1 Titolo

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sed nunc orci. Aliquam nec nisl vitae sapien pulvinar dictum quis non urna. Suspendisse at dui a erat aliquam vestibulum. Quisque ultrices pellentesque pellentesque. Pellentesque egestas quam sed blandit tempus. Sed congue nec risus posuere euismod. Maecenas ut lacus id mauris sagittis egestas a eu dui. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque at ultrices tellus. Ut eu purus eget sem iaculis ultricies sed non lorem. Curabitur gravida dui eget ex vestibulum venenatis. Phasellus gravida tellus velit, non eleifend justo lobortis eget.

B.1.1 Sottotitolo

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec sed nunc orci. Aliquam nec nisl vitae sapien pulvinar dictum quis non urna. Suspendisse at dui a erat aliquam vestibulum. Quisque ultrices pellentesque pellentesque. Pellentesque egestas quam sed blandit tempus. Sed congue nec risus posuere euismod. Maecenas ut lacus id mauris sagittis egestas a eu dui. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque at ultrices tellus. Ut eu purus eget sem iaculis ultricies sed non lorem. Curabitur gravida dui eget ex vestibulum venenatis. Phasellus gravida tellus velit, non eleifend justo lobortis eget.