International Journal of Computer Applications (0975 – 8887)
Volume 127 – No.12, October 2015

Performance Analysis of NoSQL Databases

Prateek Nepaliya
Department of Computer Science and Engineering
JEC, Jaipur

Prateek Gupta
Department of Computer Science and Engineering
JEC, Jaipur

ABSTRACT

From couple of years working on big data is a challenge. Currently there are lot of document oriented tools, we have to choose best tool from them, and so they are need to be compared. We compared CouchDB, and RavenDB to know which one of them is good. NoSQL derived from relational database, information can be retrieved fast and is portable database. RDBMS is not flexible enough to handle the variety of data. Couch dB is also considered in web world. RavenDB is a transactional, open-source Document Database written in .NET, and offering a flexible data model designed to address requirements coming from real-world systems. Using RDBMSs for Big Data is prohibitively expensive. Each one of these database has advantages and limitation hence, If a query is executed enough times, those indices are promoted to be permanent (auto-generated) indices. Even a simple query requires significant programming expertise, and commonly used BI tools do not provide connectivity to NoSQL. Databases are analyzed on the basis of their time complexities and space and finally in the end we found out of which one has real ability to get used in different situation. And also which is more efficient.

General Terms

Databases vary in complexities, especially document oriented. On retrospection we found that CouchDB and RavenDB must be compared on the basis on their performance which can be improved as efficacies.
Comparison between databases has been done in very profound way i.e. comparing various sizes of JSON files/documents and on their time complexities. The snapshot of various sizes of files and their time on both the document databases is provided.

Keywords

NOSQL, CouchDB, RavenDB, RDMS, Map reduce, Document Databases.

1. INTRODUCTION

NoSQL (Not Only SQL) is one of another type of data storage other than databases that is used to store huge amount of data storage like data in Social networking sites (Which is increasing every day). NoSQL derived from relational database, information can be retrieved fast and is portable database. Non-relational database does not store data in a non-normalized way. These databases are open source that means; anyone can look into its code and update it according to his needs. NoSQL databases use the concept of data replication. NoSQL systems are open source projects, and although there are usually one or more firms offering support for each NoSQL database, these companies often are small start-ups without the global reach, support resources, or credibility of an Oracle, Microsoft, or IBM.

The design goals for NoSQL may be to provide a zero-admin solution, but the current reality falls well short of that goal. NoSQL today requires a lot of skill to install and a lot of effort to maintain. NoSQL databases offer few facilities for ad-hoc query and analysis. Even a simple query requires significant programming expertise, and commonly used BI tools do not provide connectivity to NoSQL.

1.1 CAP theorem

In a distributed system, managing consistency(C), availability (A) and partition toleration (P) is important, Eric Brewer put forth the CAP theorem which states that in any distributed system we can choose only two of consistency, availability or partition tolerance (Hence also known as brewers theorem, First come into play in 1998). Many NoSQL databases try to provide options where the developer has choices where they can tune the database as per their needs.

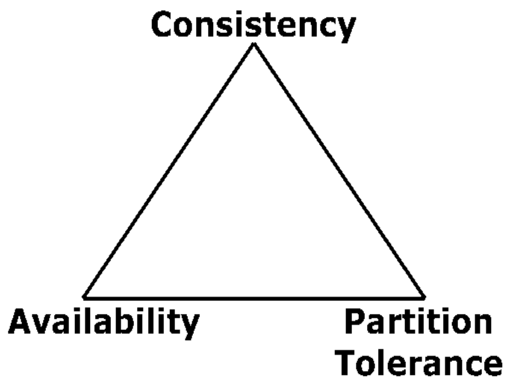


Figure1: CAP theorem

Here are some broad reasons to consider the use of NoSQL databases:

- Enhancing the effectiveness of programmer and making sure that they use database which is very prefect for their application need.
- To improve data access performance via some combination of handling larger data volumes, reducing latency, and improving throughput.

1.2 SHORTCOMINGS OF RDBMS

The volume of data and its handling is increasing day by day, so for RDBMS, it is one hell of problem in handling it and work upon it, if it is handling it includes dollars in great amount too. It is not flexible enough to handle the variety of data, which is like semi-structural data, specifically the modern type of data i.e. social media analysis financial stats etc. Modern companies and application require less response Time, flexibility which is as we see, RDBMS is facing the problem.

Big Data also demands collection of an extremely wide variety of data types, but RDBMSs have inflexible schemas.

The problem is that Big Data primarily comprises semi-structured data, such as social media sentiment analysis and text mining data, while RDBMSs are more suitable for structured data, such as weblog, sensor and financial data.

In addition, Big Data is accumulated at a very high velocity. Since RDBMSs are designed for steady data retention, rather than for rapid growth, using RDBMSs for Big Data is prohibitively expensive.

Finally, many modern day applications don't require the strong-but-expensive guarantees offered by RDBMSs. While a growing number of web applications can tolerate weak consistency, they also require low and predictable response times, high availability, effective scalability, flexible schemas and geographically distributed datacenters.

1.3 CouchDB

Couch dB is JSON document oriented, and was written in Erlang process and messaging. One main advantage of Couch is it allows parallel computing and maintenance. Couch dB is also considered in web world i.e. it is web oriented too. The fascinating part of couch dB is, it replicates two databases and it will detect the changes. It uses JavaScript as its query language, and which fully uses map -reduce and hypertext as its application interface. Its adaption to well different sizes of computing devices is very good.

Figure 2: CouchDB

CouchDB is a document oriented database which is **nothing new** [although focusing on JSON instead of XML makes it buzzword compliant] and is definitely not a replacement/evolution of relational databases. Document oriented database work well for semi-structured data where each item is mostly independent and is often processed or retrieved in isolation. This describes a large category of Web applications which are primarily about documents which may link to each other but aren't processed or requested often based on those links (e.g. blog posts, email inboxes, RSS feeds, etc.). However, there are also lots of Web applications that are about managing heavily structured, highly interrelated data (e.g. sites that heavily utilize tagging or social networking) where the document-centric model doesn't quite fit.

CouchDB has some really amazing features and As per us some of them are:

1. It stands up better to synchronous use by multiple users because it has utterly no read locks. This is possible because CouchDB never updates documents in place. Changes are always appended to the end of the database file. Consequently, writes that occur while views are being queried won't ever interfere with those queries.
2. Non-Relational database means no table/key model: CouchDB databases are non-relational, hence, very different from SQL databases. They can be easily managed and are flexible and have several data models.
3. The open source nature of CouchDB databases means development of large application is comparatively more economical.

1.4 RavenDB

RavenDB is a transactional, open-source Document Database written in .NET, and offering a flexible data model designed to address requirements coming from real-world systems. RavenDB allows you to build high-performance, low-latency applications quickly and efficiently.

Figure 3: RavenDB

The following point I'd like to share:

- There is a very complete C# client API.
- It is very easy to set-up a new project on RavenDB. You can start with an embedded database and easily transition into a server-hosted mode.
- No additional mark-up is required in your documents (i.e. no attributes are needed to be able to locate documents).
- Auto-indexing. RavenDB will automatically create indices as queries are executed. If a query is executed enough times, those indices are promoted to be permanent (auto-generated) indices. In addition to the automatic performance tweak this provides, it also grants you some insight into which indices you should consider adding yourself.
- Map/reduce queries can be written in C# as part of index definitions. With some of the other NoSQL databases, you still need to drop to JavaScript to run map/reduce operations.

2. EXPERIMENTAL ENVIRONMENT

2.1 For Couch

- **Operating System:** Windows 8.1 pro, 64 bit operating system.
- Web based CouchDB client.
- Without Admin.

2.2 For Raven

- **Operating System:** Windows 8.1 pro, 64 bit operating system.
- Web based client.
- .NET framework 4.0.

3. RESULT AND ANALYSIS

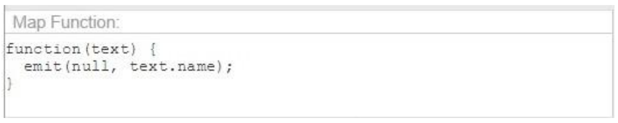
Databases are taken to analyze the efficiency and which of both the document oriented databases are more efficient. Sample JSON document of random data plus a map reduce code to emit whole document. This particular map reduce commands emits the doc with duration of the processing, which can be as simple as finding the name of the person with initials "P" or the whole document for reading on further comparison the duration is graphed.

3.1 COUCHDB

Table 1: Time taken in JSON file by CouchDB

S.NO	FILE TYPE	FILE	TIME TAKEN
1	JSON	20k lines	120ms
2	JSON	50k Lines	150ms
3	JSON	75k Lines	188ms

The Map function for the sample emitting of name of the document is:



```
Map Function:
function(text) {
  emit(null, text.name);
}
```

Figure 4: Map-reduce

The revealed data graph for CouchDB is:

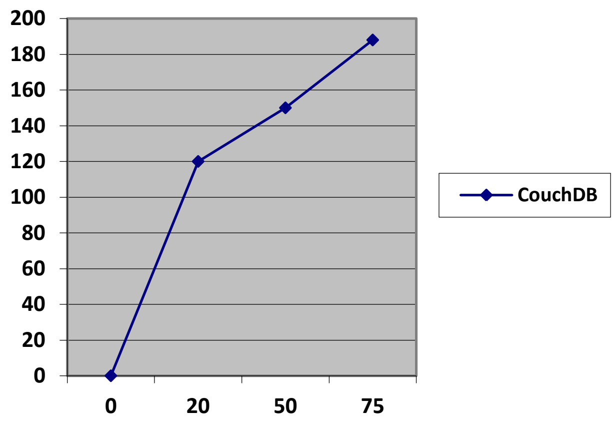


Figure 5: Graphical Analysis of document and time taken

3.2 Ravendb

Table 2: Time taken in JSON file by RavenDB

S.NO	FILE TYPE	FILE SIZE	TIME TAKEN
1	JSON	20k lines	132ms
2	JSON	50k Lines	208ms
3	JSON	75k Lines	280ms

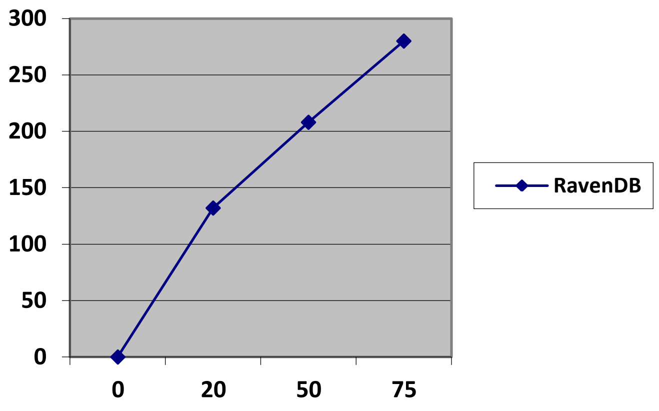


Figure 6: Graphical Analysis of document and time taken

3.3 Analysis

This is very clear that using CouchDB is more effective than using RavenDB. The amounts of time taken for processing a easy map reduce command i.e. a simple map reduce commend to emit the names in file.

We found that using the same command we can easily chose a single database for a particular type of application; however the other database (RavenDB) can also be used for particular type of application development. RavenDB is used for dynamic applications such as transactional model.

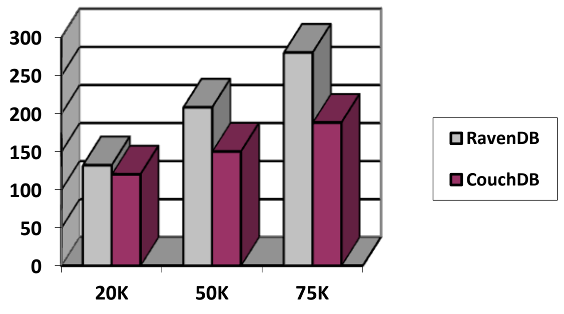


Figure 7: Comparison between Couch and Raven

3.4 Limitations of CouchDB

- It doesn't support transactions.
- Not perform well in 3rd normal form relational data
- Temporary views in CouchDB on large datasets are really slow which has really bad impression on map reduce.
- Not designed for frequent update.
- No inter-walking between documents.

3.5 Limitations of RavenDB

- No indexing.
- Lucene indexing/querying is weird, compared to a "normal" RDBMS (all Indexes are string based; tokenized, stop words are stripped.)

4. CONCLUSION

What we found on comparing both the databases is RavenDB performs well on a single line query, though it also performs comparatively well on multi line queries on some thousands of lines of database. On the other hand Couch has its slight downfall in time taken to run a query of the same 20k, 50k, 75k lines of JSON file.

As the number of records in database increases, the difference between the execution time taken by CouchDB for the computation of different database operations is better in comparison to RavenDB.

For the data retrieval operation, the performance of CouchDB is about 25 percent better in comparison with RavenDB.

For any update in document the performance increases in CouchDB in comparison with RavenDB. For update and deletion of data CouchDB is almost double the percent of RavenDB.

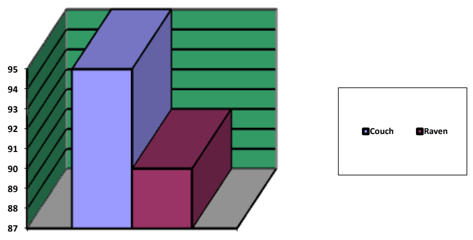


Figure 8: Overall Performance (%)

5. FUTURE SCOPE

In future we can compare more two or three document based databases for the different types of documents such as xml, JSON and csv. Comparative study is always helpful in choosing one of the databases. Also limitation and advantages are known by comparative study. More lines of documents can be compared on the same databases.

6. REFERENCES

[1] “Rabl, Tilmann; Sadoghi, Mohammad; Jacobsen, Hans Arno; Villamor, Sergio Gomez-; Mulero -, Victor Muntes; Mankovskii, Serge (2012-08-27)."Solving Big DataStax (2013-01-15).

[2] Dheeraj Bhardwaj, "Parallel Computing- A Key to Performance", Department of Computer Science & Engineering, Indian Institute of Technology Delhi, August 2011.

[3] “About data consistency". Retrieved 2013-07-25. Ellis, Jonathan (2012-02-15).

[4] D. an Mey,” Two Open MP programming patterns", Proceedings of the Fifth European Workshop on Open MP - EWOMP'03, September

[5] R. Angles and C. Gutierrez,” Survey of graph data-base models”, ACM Computer. Surv. 40(1):39, 2008. [13]Neo4j. The neo database (2006), Avail-able http://dist.neo4j.org/neo-technology-introduction.pdf

[6] M. I. Jordan (Ed). (1998).”Learning in Graphical Models". MIT Press. Hypergraph DB website, Availablehttp://www.kobrix.com/hgdb.jsp

[7] “Introduction to No-SQL”. http://www.cs.tut.fi/~tjm/seminars/nosql2012/NoSQL-Intro.pdf

[8] Gobioff, Sanjay Ghemawat H.; Leung, Shun-Tak: The Google File System. In: SIGOPS Oper. Syst. Rev. 37 (2003), No. 5, p. 29–43. – http://labs.google.com/papers/gfs-sosp2003.pdf

[9] Cooper, Brian F. ; Ramakrishnan, Raghu ; Srivastava, Utkarsh ; Silberstein, Adam ; Bohannon, Philip ; Jacobsen, Hans A. ; Puz, Nick ; Weaver, Daniel ; Yerneni, Ramana: PNUTS: Yahoo!’s hosted data serving platform. In: Proc. VLDB Endow. 1 (2008), August, No. 2, p. 1277–1288. – Also available online. http://research.yahoo.com/files/pnuts.pdf

[10] Cattell, Rick: High Performance Scalable Data Stores. February 2010. – Article of 2010- 02-22. http://cattell.net/datastores/Datastores.pdf

7. AUTHOR’S PROFILE

Prateek Nepaliya is computer science student pursuing Bachelor of technology in Jaipur Engineering College, Rajasthan Technical University. His research interest involves big data, NO-SQL databases, Data science and Statistics.

Prateek Gupta is computer science student pursuing Bachelor of technology in Jaipur Engineering College, Rajasthan Technical University. His research interest involves big data, study of NO-SQL databases, Map-reduce functionality and Data mining.