International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-2 Number-4 Issue-6 December-2012

Improving Network Scalability Using NoSql Database

Ankita Bhatewara[1], Kalyani Waghmare[2]

Pune Institute of Computer Technology,Pune, India
B.E.Computer[1], M.E. Computer[2]
bhatewara.ankita@gmail.com, kalyani mehunkar@yahoo.com.

Abstract

The traditional database is designed for the structured data and the complex query. In the environment of the cloud, the scale of data is very large, the data is non-structured, the request of the data is dynamic, these characteristics raise new challenges for the data storage and administration, in this context, the NoSQL database comes into picture. This paper discusses about some non-structured databases. It also discusses advantages and disadvantages of Cassandra and how Cassandra is used to improve the scalability of the network compared to RDBMS.

Keywords

Non-structure, NOSQL , Network Scalability.

1. Introduction

For years, database administrators have relied on scale up by buying bigger servers as database load increases rather than scale out distributing the database across multiple hosts as load increases. However, as transaction rates and availability requirements increase, and as databases move into the cloud or onto virtualized environments, the economic advantages of scaling out on commodity hardware become irresistible. RDBMS might not scale out easily on commodity clusters, but the new breed of NoSQL databases are designed to expand transparently to take advantage of new nodes, and
They are usually designed with low-cost commodity hardware in mind.

Cloud Data Management is a new data management concept with the development of cloud computing, it must be able to efficiently manage of large data sets in the cloud, and quickly locate specific data in mas-sive data sets, which makes the Cloud Data management with the following common characteristics: (1) high concurrent read and write performance, (2) efficiently store and access huge mounts of data, and (3) high scalability and high availability requirements of the database. In the face of these demands, the traditional relational data management system (RDBMS) has encountered an insurmountable obstacle. Therefore, NoSQL database systems rose alongside major internet companies, such as Google, Amazon, Twitter, and Facebook which had significantly different challenges in dealing with data that the RDBMS solutions could not cope with. These companies realized that performance and real time nature was more important than consistency, which traditional relational databases were spending a high amount of pro-cessing time to achieve. As such, NoSQL databases are often highly optimized for retrieve and append operations and often offer little functionality beyond record storage. The reduced run time flexibility compared to RDBMS systems is compensated by significant gains in scalability and performance. Often, NoSQL databases are categorized according to the way they store the data and fall under categories such as key-value stores(e.g. Dynamo[1]), BigTable implementations[2] and document store databases(e.g. MongoDB[3]). However, due to the immature technology of cloud data mangement, there are still many issues need to be addressed in actual production environment.

2. Approach: Replacing RDBMS with NoSql Database

To explain the difference in the performance of non-structured and structured database, we show an simple example of a network management system. We use RDBMS as structured and Cassandra as non-structured database.

The system consists of management software and number of security appliances mounted on it. Users are connected to management software through security appliance. The goal is to allow large number of users logins to make the network scalable.
The system is represented as:

S={ M, A, U }
Where,
M is set of Management Software.
A is set of Security Appliance.
U is set of user.
A= {a_1, a_2, a_3…}
U= {u_1, u_2, u_3 ...}
Function *f* defined for M and A as,
f (M → A)
Mapping from Management Software to Security Appliance is one to many.
Function g defined for C and W as,
g (A → U)
Mapping from security appliance to user is one to many.

Login Workflow:

This section explains the login process in detail. Whenever any new user login into the network it does it via security appliance. Security appliance sends an event to management software to add that user into its database and accordingly the database is updated with the new user entry. Then the management software takes care of reflecting the changes in the graphical User Interface. This process is shown in fig.1.

As soon as any new user login to the network some events are generated, this events are called configuration change events. After the generation of this event, database needs to be updated. As the number of users login into the network increases, number of writes to the database also increases. After a particular number of users the login process becomes slow and many a times the events get dropped. When any configuration change event gets dropped the login is not successful. Hence in order to support large number of users or to make the network scalable the writes to database should be faster. Here comes the non-structured database into picture, one of which is Cassandra, it allows over a million of writes per second.

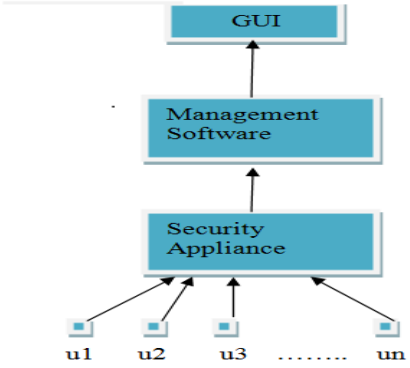

Fig.1 Login Workflow

3 .Database Replacement

Relational databases consist of tables, whereas Cassandra contains Column Families or SuperColumnFamilies [4]. Column Families contains row keys where each row key contains one or more columns and each column is a name/value pair. In relational tables if we don't have value for a particular column, we use NULL where as in Cassandra that particular name/value pair can be omitted. Hence in Cassandra's column family different row keys may have different number of columns. The architecture described in the section above uses RDBMS (Postgresql) as database. For this system, when any login event occurs some of the tables need to be updated. The architecture is designed in such a fashion that on any login event three or four tables are updated. The current system which uses Cassandra as a database is designed such that all the columns from those tables which gets updated on occurrence of login event are placed in a single column family. To compare the performance of both the database we carried out a 100k users login tests

on the databases. We found that the total time required for 100k users logins it took 180 minutes for relational database where as with Cassandra it was completed in just 70 minutes. Also with relational database some events were dropped but with Cassandra all 100k users logins were successful. The results of the test are shown below. It is observed that using Cassandra, 100k logins were completed in almost half time as it took for postgres database. From the graphs shown in Fig.2 and Fig.3 it can be verified that the time required for relational database to complete 100k login is approximately double than time required for Cassandra. Graphs show that using Cassandra time for most of the logins is between 0 to 100 milliseconds whereas for postgres it is between 0 to 400 milliseconds.

In the proposed section we must include what the author's proposed along with the advantages and disadvantages in your own language.

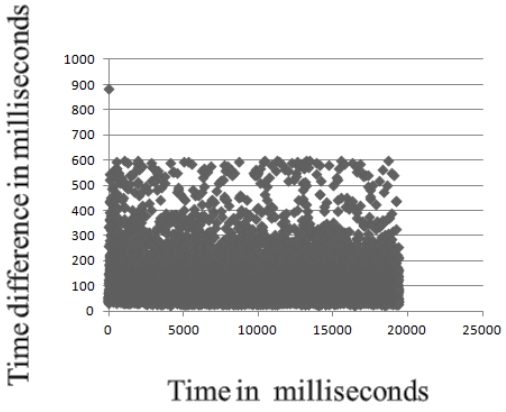

Fig.2 Performance using Postgres

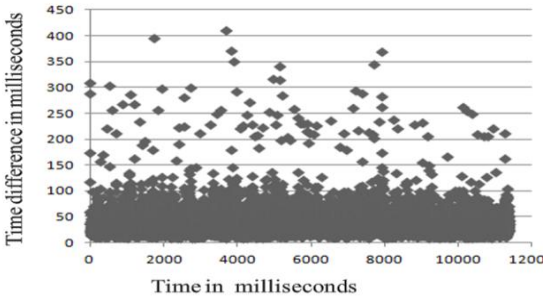

Fig.3 Performance using Cassandra

4. Conclusion

In this paper, we discussed about structured and non-structured databases. We presented an example of network management system and showed how a non-structured database i.e. Cassandra which has elastic scalability feature, improves the performance of the system. We also showed the test results in the graph which verifies our proposal that Cassandra performs better than relational database. Hence we conclude that using Cassandra we can scale the network without changing any hardware or buying bigger servers. Thus network scalability is improved with low-cost commodity hardware.

References

[1] Yimeng Liu, Yizhi Wang,Yi Jin, Research on The Improvement of MongoDB Auto-Sharding in Cloud Environment", The 7th International Conference on Computer Science and Education.

[2] Fay Chang, Jeffery Dean, Sanjay Ghemawat, et al. Bigtable: A Distributed Storage System for Structured Data. 7th Symposium on Operating System Design and Implementation. Seattle, WA, USA: 2006.

[3] 10gen. MongoDB. http://www.mongodb.org, 2011-

07-15.

[4] Cassandra: The Definitive Guide, Eben Hewitt,

O'REILLY.

Ankita Bhatewara has received B.E. (Computer Science) degree in 2010 from Pune University. She is currently appearing for Master of Computer Engineering in Pune Institute of Computer Technology, Pune. Her research area is Data mining.
.