ISSN 2277-3061

Highly Scalable Network Management Solution Using Cassandra

Ankita Bhatewara, Kalyani Waghmare
Pune Institute of Information Technology, University of Pune

Abstract

With the current emphasis on Big Data, NOSQL databases have surged in popularity. These databases are claimed to perform better than SQL databases. 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 shows how Cassandra is used to improve the scalability of the network compared to RDBMS.

Indexing terms/Keywords

Non-structure; NOSQL; Network Scalability

Journal: INTERNATION JOURNAL OF COMPUTERS AND TECHNOLOGY
Vol. 13, No. 10
editorijctonline@gmail.com
www.ijctonline.com, www.cirworld.com

1. Introduction

Traditional database systems for storage have been based on the relational model. These are widely known as SQL databases named after the language they were queried by [1]. In the last few years, however, non-relational databases have dramatically risen in popularity. These databases are commonly known as NOSQL databases, clearly marking them different from the traditional SQL databases. Most of these are based on storing simple key-value pairs on the premise that simplicity leads to speed[1]. For years, database administrators have relied on scale up by buying bigger servers as database load in- creases - 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're usually designed with low-cost commodity hardware in mind.

2. Academic Discipline and Sub-Disciplines

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 massive 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 amounts 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 Face book 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 processing 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 [2]), Big Table implementations [3] and document store databases (e.g. Mongo DB [4]). However, due to the immature technology of cloud data management, there are still many issues need to be addressed in actual production environment.

3. Subject Classification

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. To explain the difference in the performance of nonstructured and structured database, we show a simple example of a network management system. We use RDBMS as structured and Cassandra as non-structured database. Relational databases consist of tables, whereas Cassandra contains Colum- n Families or Super Column Families [5]. 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 up- dated. 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 de- signed such that all the columns from those tables which get updated on occurrence of login event are placed in a single column family.

3.1 Mathematical Model

The system is represented as: $S = \{M, A, U, C_i, C_o\}$

Where,

M is set of Management Software.

A is set of Security Appliance.

U is set of user.

C_i = Count of user logged in

C_o = Count of user stored in database and logged in successfully.

$A = \{a_1, a_2, a_3 \ldots\}$

$U = \{u_1, u_2, u_3 \ldots\}$

Success Condition:

$C_i = C_o$

Failure Condition:

$C_i = C_o$

Function *f defined* for M and A as,

$f(M \rightarrow A)$

Mapping from Management Software to Security appliance is one to many.

Function *g* defined for C and W as,

$g(A \rightarrow U)$

Mapping from security appliance to user is one to many. This mapping is shown in Fig.1

```
   Functions
CCESubscriber(CCE)
{
   if CCE Queue is full then:
      drop;
   else
      add CCE to CCEQueue;
   end if
}
CCEProcess(CCE)
{
   CCE: = Fetch_ from CCEQueue
         if_Is_ login_ CCE then:
      dbCall_ Cassandra();
   else
      dbCall_ Postgres();
   end if
   }
```

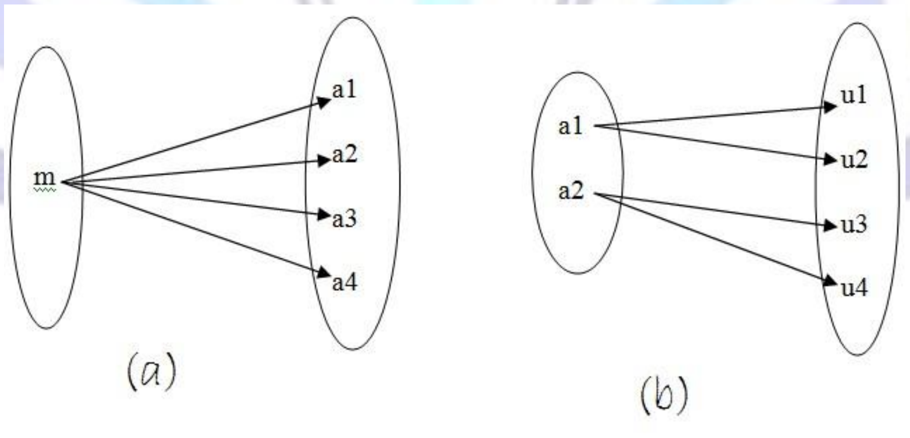


Figure1: (a) Mapping between Management Software and Security Appliance.

(b)Mapping between Security Appliance and User

3.2 Data Flow architecture

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 up- dated. 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 de- signed such that all the columns from those tables which gets updated on occurrence of login event are placed in a single column family.

Fig.2 shows the login process in detail. Whenever any new user login into the net- work 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.

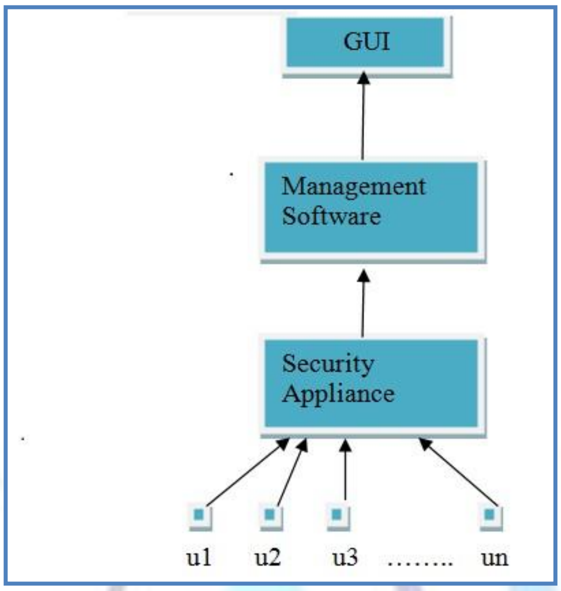
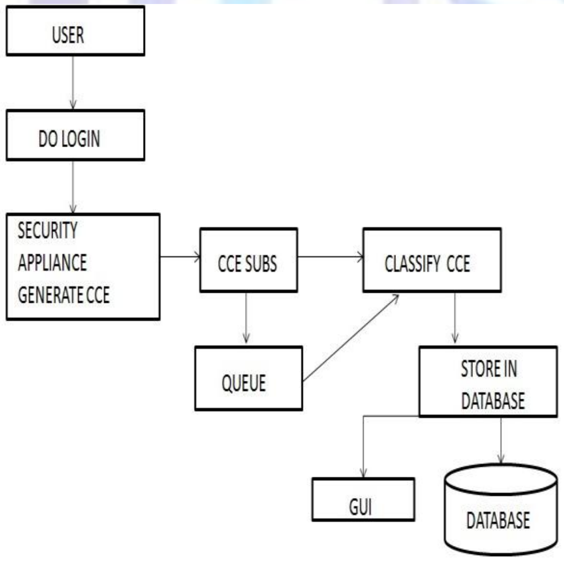


Figure 2: System Architecture

3.3 Multiplexer Logic

The overall system consists of a single server node i.e. management software and multiple client nodes i.e. security appliance. For each user login the server maintains a unique thread. When any new user logs in, a new thread is assigned to it thus it's a multithreaded system.



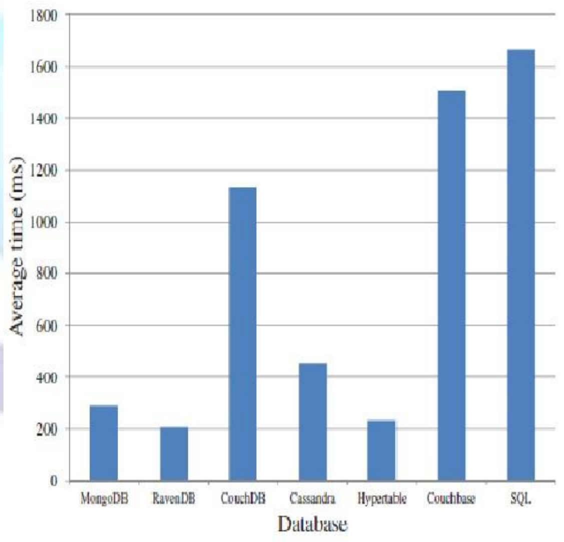


Figure 3: Login Workflow **Figure 4: Time for instantiating database (ms)**

Parallelism is achieved as each user's functionality is carried out parallel by unique thread. Fig.3 shows the login workflow of the system.

4. Results and Discussion

The first experiment measures the time taken to instantiate a database bucket [1]. See Figure 4 [1] which summarizes the results of this experiment. Note that the times are averaged over five runs. The absolute time values are not significant; what are significant are the time values relative to one another.

We observe that Raven DB, Hyertable, MongoDB offer the fastest creation of database buckets. CouchDB, Couchbase, and SQL Express are among the slowest to create buckets. To compare the performance of Cassandra and SQl database we carried out a 100k user's 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 droped 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.5 and fig.6 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.

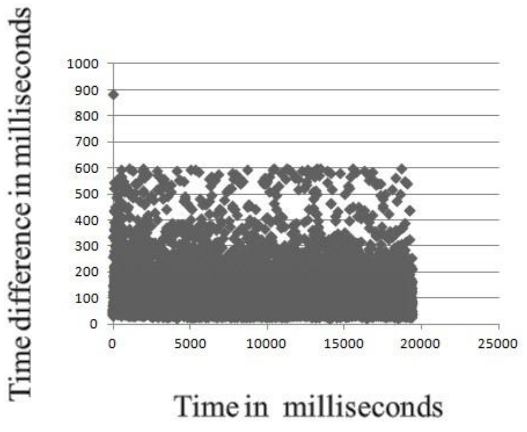


Figure 5: Performance using Postgres

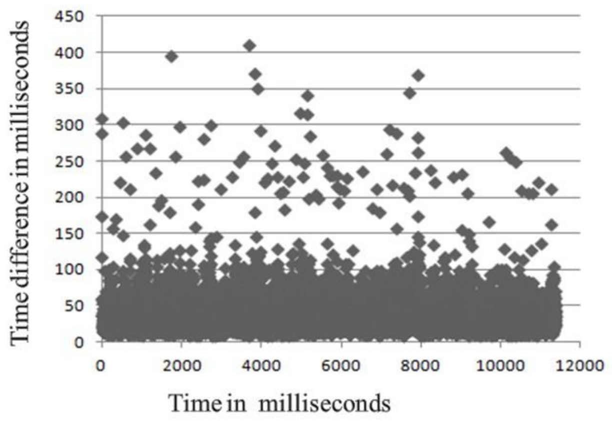


Figure 6: Performance using Cassandra

5. 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] Yishan Li and Sathiamoorthy Manoharan, "ŞA performance comparison of SQL and NOSQL databases", communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference.

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

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

[4] 10gen. MongoDB. http://www.mongodb.org, 2011-07-15.

[5] Cassandra: The Definitive Guide, Eben Hewitt, O'REILLY.