ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 4, pp: (140-148), Month: October - December 2015, Available at: www.researchpublish.com

# Benefits of HBase Over Relational Databases in Context of Scalability, Performance and Agile Development

Mandar Jadhav[1], Prof. K.K Joshi[2]

[1]MTech Student, [2]Professor

[1,2]Department of Computer Engineering and IT, Veermata Jijabai Technological Institute, Mumbai, India

***Abstract*: The explosive growth of internet in recent years led to tremendous increase in volume of data. Today this data is accessed more frequently and data processing needs have increased. Relational databases were not designed to deal with scaling and agility challenges, nor were they built to harness the benefits of high processing power and cheap storage available in recent times. Today in era of cloud computing many companies and organizations are shifting towards an alternative for legacy relational infrastructure used since early 1970s. Apache's HBase, a NoSQL database is the perfect alternative for relational databases. The purpose of this paper is to identify the benefits of HBase and the motivating factors that drive organizations to choose HBase over relational databases. The paper provides a thorough technical background about traditional SQL databases and their limitations before proceeding to compare them with HBase. Finally it shows performance evaluation of HBase and various issues being addressed that are difficult to deal with if we use the relational model.**

***Keywords:* HBase; NoSQL; Hadoop; Relational Database; Column oriented Database; Cloud storage; RDBMS; Scalability.**

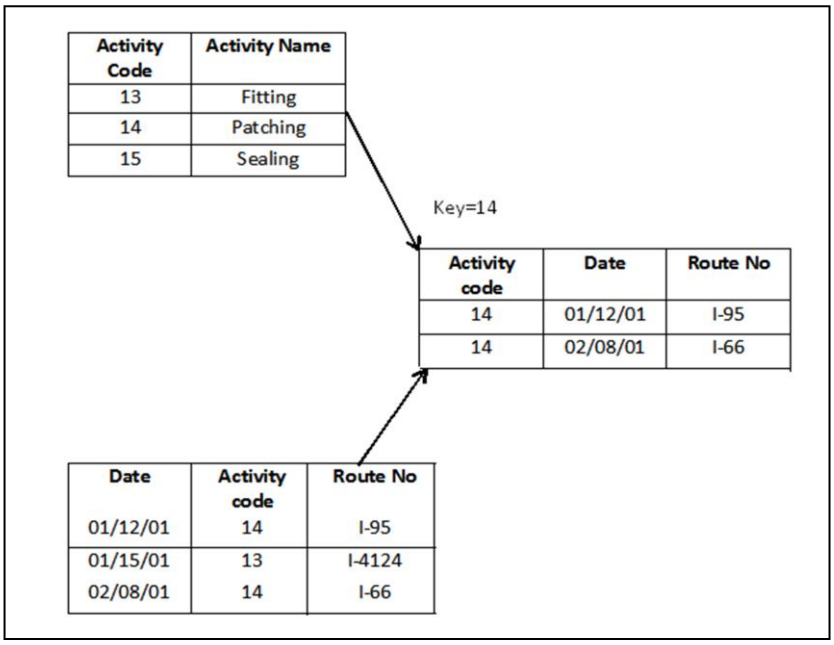
## I. INTRODUCTION

In 1970's, for storing structured data in well-organized tabular format relational database technology came into existence. The Structured Query Language (SQL) model was developed to handle that data. Since then, databases are considered to be an important entity of any organization and are being used globally. However, there are various limitations of relational databases as there is a considerable growth in volume of stored and analyzed data. Due to increasing size of data there is loss of data querying efficiency, thus the management and storage of larger databases is also challenging. New features of managing data were provided by NoSQL databases which helped in overcoming the limitations of currently used RDBMS. NoSQL databases exhibit flexibility and are scalable horizontally in comparison to relational databases. The main advantage of non-relational database is the absence of any rigid data structure that, in relational databases, has to be defined before storing the data. This approach prevents us to use SQL for querying stored records but data can be easily stored and retrieved regardless of structure and content. NoSQL databases exhibit easier and better horizontal scalability. Those databases does not need manual distribution of information or any sort of administrator management to take new nodes and clusters transparently. Hundreds of low-cost servers can satisfy user requests due to horizontal scalability, thus reducing company's cost. Using a NoSQL system is more profitable than building high-end capacity mainframes. The cost per request or volume of stored data is comparatively less than relational model. The data is automatically managed, distributed and there is automatic fault recovery in NoSQL databases. Therefore, non-relational databases like HBase are gaining momentum as future technologies that replace or complement the usage of traditional relational databases. HBase is developed for running on computers in a cluster, not for single system. Commodity hardware can be used to build such clusters. Horizontal scaling is possible with HBase, thus you add more computers to the cluster. Every node in the cluster has some amount of storage, cache, and a bit of computation. This allows Hbase to be incredibly flexible. As concept of uniqueness does not hold true for any node, you can replace any failed machine with another working system.

## II. RELATIONAL DATBASE OVERVIEW

### *A. RDBMS:*

The name Relational Database System was coined based on the name of implemented model: The Relational Model, which offers an adapt way of keeping and using the structured data. A database which follows 12 rules of Codd is called RDBMS. Typical RDBMSs are row-oriented, static-schema databases based on ACID properties and they consist of a sophisticated SQL engine for querying. It is modified implementation of the flat model and the network model, by adding the feature of relations. The benefits of using relations are storing data as constrained collections in structured way which is referred as group-keeping of the data, assigning attribute values to attributes for relating all input (e.g. a Customer's ID number). A well-defined and clearly set schemas are basic requirements for RDBMS. Schemas represent the amount of information belonging to each record and its type. They are like tables with columns, and entries are represented by rows.



| Activity Code | Activity Name |
|---|---|
| 13 | Fitting |
| 14 | Patching |
| 15 | Sealing |

| Activity code | Date | Route No |
|---|---|---|
| 14 | 01/12/01 | I-95 |
| 14 | 02/08/01 | I-66 |

| Date | Activity code | Route No |
|---|---|---|
| 01/12/01 | 14 | I-95 |
| 01/15/01 | 13 | I-4124 |
| 02/08/01 | 14 | I-66 |

**Figure 1. A relational model**

### *B. Key Characteristics:*

- **Keys Usage:** There is a unique "key" that identifies each tuple of data in any table. It is known as the primary key. The rows of different tables can be linked by adding key values of one table to rows of a other table.

- **Avoiding Data Redundancy:** Each data item, a customer name for example, is stored in one location. Thus there is no need of maintaining the same data in different locations.

- **Constraints:** You can specify the type of data that is allowed to be stored in columns of database. Fields containing text, numeric data, dates, etc can be specified according to one’s needs.

- **Integrity of Data:** Reliability and integrity of data can be increased by setting constraints, properties and by linking of tables.

- **Permissions:** In RDBMS rights can be assigned to users according to their roles. Operations like selection, insertion, deletion, creation, altering, etc. can be allowed or disabled for a user.

- **Structured Query Language (SQL) :** This programming language is designed for management of data stored in a relational database management system (RDBMS). A long-established standard adopted by ISO & ANSI is used by SQL whereas there is no clear standard adopted by Non-SQL databases.

### *C. Key Properties:*

The properties that guarantee reliable processing of database transactions are known as ACID properties.

- **Atomicity:** This property ensures that either all or none of the transaction tasks are performed. An all or nothing rule is followed before modifications to database.

- **Consistency:** This property states that the database should remain in a consistent state prior to the start and after the transaction is over irrespective of whether its successful or not. A distributed database has some form of weak consistency or can be strongly consistent. Weak consistency is analogous to eventual consistency, in which the database reaches a consistent state eventually.

- **Isolation:** This property states that intermediate state data cannot be accessed or seen by other operations during a transaction. Thus, every transaction in the system is unaware of other concurrently executing transactions.

- **Durability:** This property states that after user is notified about the success, the transaction will persist, and it won't be undone which means it will survive even if there is failure of system. A transaction is committed only after it is written in the log.

### *D. Limitations:*

- **Scaling:** Relational databases are scalable vertically but not horizontally. When you start to focus on scale and performance rather than correctness, you end up short-cutting and optimizing for your domain-specific use cases everywhere possible. Once you start implementing your own solutions to your data problems, the overhead and complexity of an RDBMS gets in your way. The abstraction from the storage layer and ACID requirements are an enormous barrier for scale.

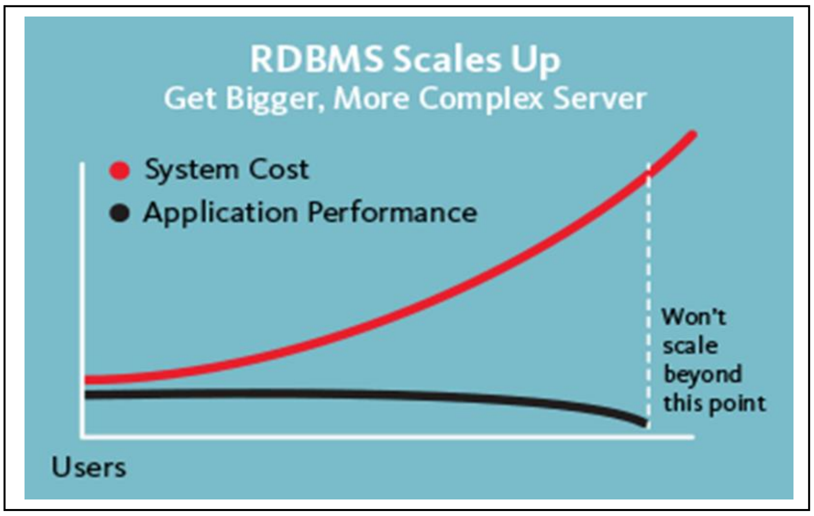


**Figure 2. Scaling in RDBMS**

- **Cost:** The proprietary software are quite expensive. The cost per request and storage cost increases as volume of data rises.

- **Data Handling:** Relational databases are not suitable for massive volumes of it semi-structured, unstructured and polymorphic data.

## III. HBASE

HBase is an open-source column-oriented, non-relational, distributed database. It is an implementation of Google's Big Table architecture for storage. It is a random access platform for storing and retrieving data, which means that you can write and read data according to your need. The semi-structured and structured data present in HBase can be loaded with tweets, log files and a product catalog along with reviews by customers. Unstructured data can also be stored unless it's too large. A dynamic, flexible model of data without any constraints on kind of data is provided. Big data is responsible for its emergence.

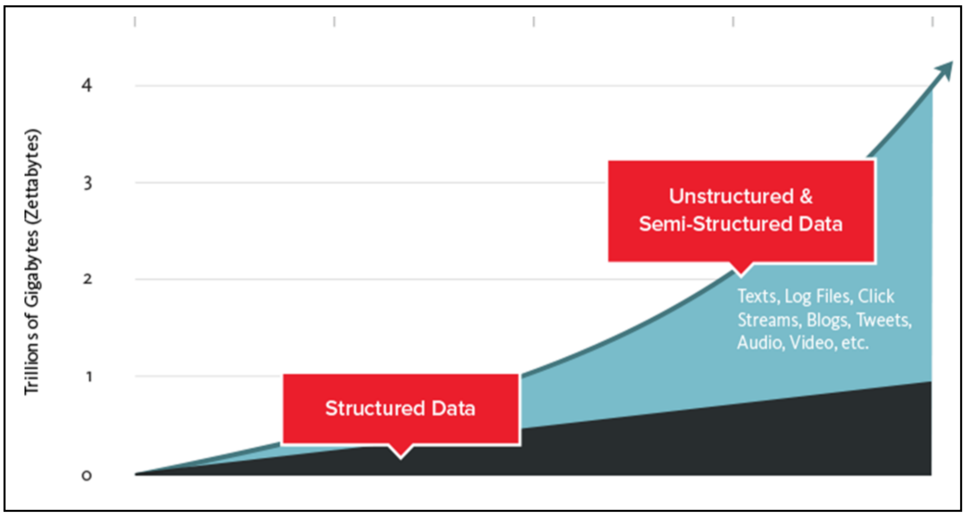


**Fig 3. Rising volume of da**

Today, HBase is a top-level Apache project with its well-formed developer and user communities. It is a core component of infrastructure and is being used worldwide in companies like Twitter, Facebook, Stumble Upon, Salesforce, Adobe, and Trend Micro.

***A. Features:***

- Scale-out Architecture - Multiple servers can be added for increasing capacity.
- Full Consistency – It guarantees protection against failure of nodes or simultaneous write operations on same record.
- High Availability – Data can be accessed anytime due to presence of many master nodes.
- Replication – Data is replicated at multiple nodes which is helpful in disaster recovery.
- Filters - When scanning tables, Filter instances refine the results returned to the client.
- Counters - support for read-and-update atomic operations.
- Full-text, Faceted Search - Give non-technical users and your applications a familiar yet powerful, interactive search experience.
- **Distributed query support:** "Sharding" a relational database can reduce or eliminate the ability to perform complex data queries. NoSQL database systems retain their full query expressive power even if the databases are distributed across many servers.
- **Integrated caching:** Data is cached in memory of system for increasing throughput and reducing latency.

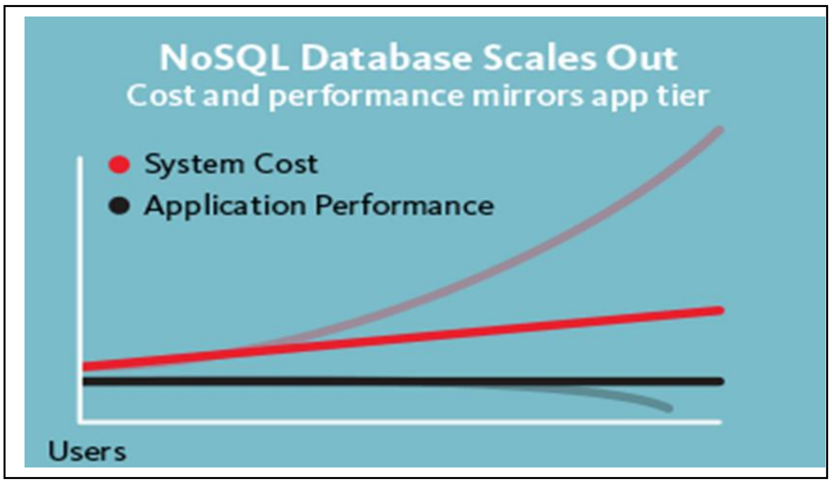


**Figure 4. Scaling in NoSQL database : HBase**

### *B. Factors influencing use of HBase:*

#### *1. Technical:*

There is need for scaling and performing beyond the capabilities of currently used systems in organizations.

#### *2. Software cost*:

Organizations need suitable, viable alternative software systems as replacement for expensive proprietary software.

#### *3. Agility or speed of development*:

Agile development methods are being embraced by organizations for rapidly adapting to market demands.

These drivers are applicable to transactional and analytical applications. Companies are shifting workloads to Hadoop for their offline, analytical workloads, and they are building online, operational applications with a new class of data management technologies.

Development teams have a strong say in the technology selection process. This community tends to find that the relational data model is not well aligned with the needs of their applications.

- Developers are working with new data types structured, semi-structured, unstructured and polymorphic data and massive volumes of it.
- Long gone is the twelve-to-eighteen month waterfall development cycle. Now small teams work in agile sprints, iterating quickly and pushing code every week or two, even every day.
- Object-oriented programming is the norm, and developers need a data model that aligns with this approach, that is easy to use and that provides flexibility.
- Organizations are now turning to scale-out architecture using commodity servers and cloud computing instead of large monolithic architectures.

## IV. WORKING PRINCIPLE OF HBASE

### *A. CAP Theorem:*

The CAP theorem (Brewer's theorem) was introduced by Eric Brewer in 2000. The theorem states that there can be at the most two out of following three characteristics in a distributed database system:

- Consistency: At given point of time every node belonging to any cluster will see the same data.
- Availability: Even after failure of node the database will not be rendered inoperative.
- Partition tolerance: Nodes will continue functioning even after loss of communication with other nodes.

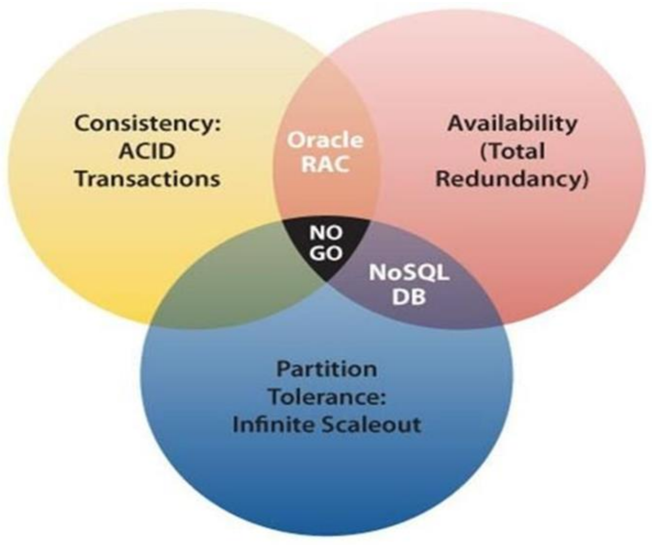

Consistency: ACID Transactions
Oracle RAC
Availability (Total Redundancy)
NO GO
NoSQL DB
Partition Tolerance: Infinite Scaleout

**Figure 5. CAP System**

Example: There is replication of data between two connected nodes. Applying CAP theorem to this scenario:

$N_1$ -------- $N_2$

Consistency: If data set in $N_1$ is changed then the change should be replicated in $N_2$.

Availability: User should be able to query/update the data from any of the two nodes $N_1$ or $N_{2.}$

Partition Tolerance**:** User should be able to query/update the data even after failure of link between $N_1$ and $N_{2.}$

A better way of understanding the CAP theorem is to apply it on network partitioning scenario.

$N_1$ ---- //---- $N_2$

After network partition no communication is possible between $N_1$ and $N_{2.}$ Assuming that we can discover partition; Now, if user changes the data by talking to $N_1$, then there are no means by which this change can be propagated to $N_2$. If we wish to achieve availability then we must allow changes in both $N_1$ and $N_2$. This creates a problem called a "split brain problem" in which few updates are reflected in $N_1$ and others in $N_2$ leading to an inconsistent system.

If we wish consistency then all changes on $N_1$ and $N_2$ should be blocked thus leading to an unavailable system.

### *B. Eventual Consistency:*

The ACID properties proved to be a bottleneck for high availability in partitioned databases because they enforced strong consistency. Hence, for improving speed and availability, a weaker form of consistency called Eventual consistency (normally asynchronous transactions) is used. Availability is also reduced due to 2-phase commit transaction. Eventual consistency is also called BASE (Basically Available, Soft state, Eventual consistency) and it is the opposite of ACID (Atomicity, Consistency, Isolation and Durability). According to ACID consistency is needed after every operation but, BASE optimistically accepts that the consistency will be in a flux state. One can say that eventual consistency is a simple acknowledgement informing about an unbounded delay in propagation of a change made on one system to all the existing copies that may produce stale data.

### *C. NRW(Read-Your-Writes) Notation:*

Node, Read, Write (NRW) helps in analyzing trade off and tuning of database consistency, read/write performance.

$N \rightarrow$ number of nodes for keeping copies of records.

$W \rightarrow$ number of nodes for acknowledging successfully committed write operations.

$R \rightarrow$ number of nodes for sending common value of data unit to be accepted as read by the system.

H Base uses the following rule:

This above rule leads to following inferences:

$N>W>1$

$W<N$
High write availability

$R<N$
High read availability

$W+R>N$
Strong consistency

$W+R<=N$
Eventual consistency

## V. PERFORMANCE EVALUATION

We will evaluate HBASE in terms of eventual consistency. The expected consistency of any eventually consistent data store can be predicted using Probabilistically Bounded Staleness model. Below is the performance evaluation for dynamo-style quorums for accuracy of 234 iterations / point.

Case 1 : W+R>N

N=7, W= 3, R=5

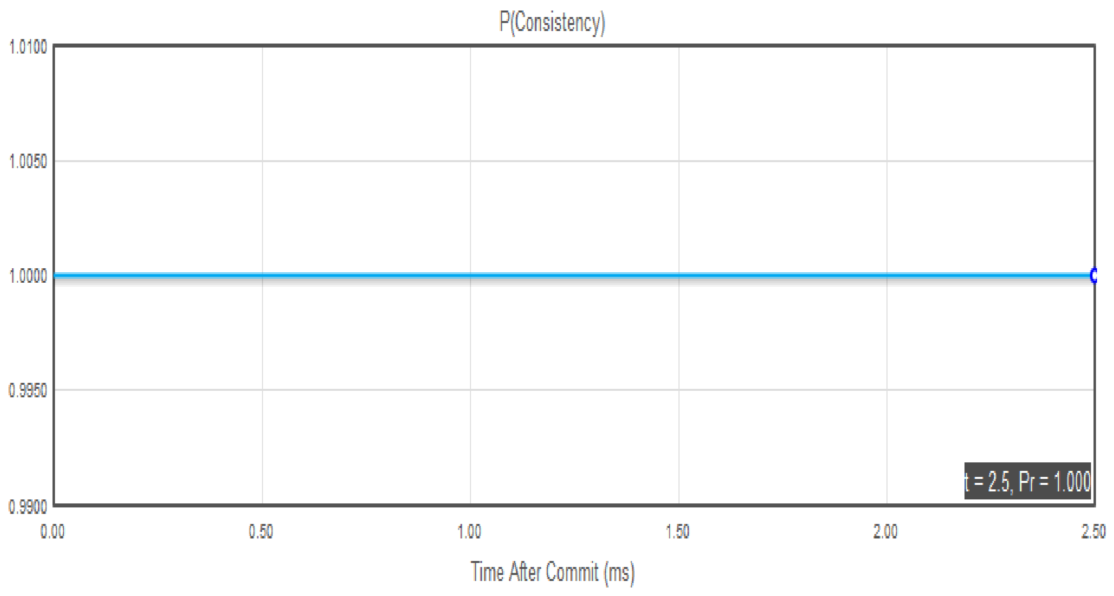


**Figure 6.**

Case 2 : W+R<N

N=7, W=3, R=3

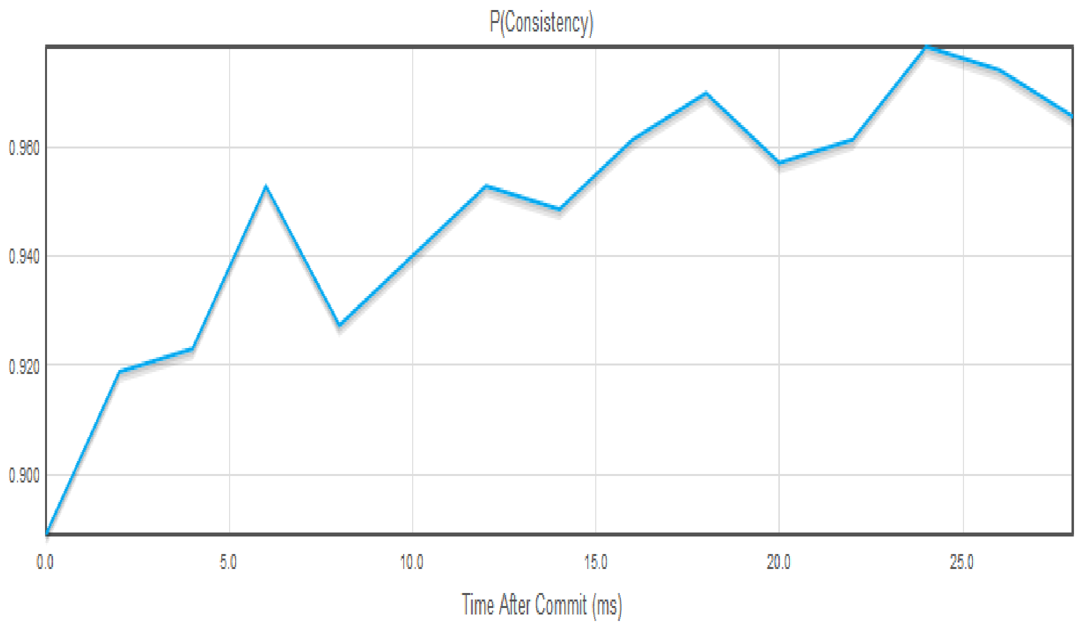


**Figure 7**

Case 3 : W+R=N

N=6, W=3, R=3

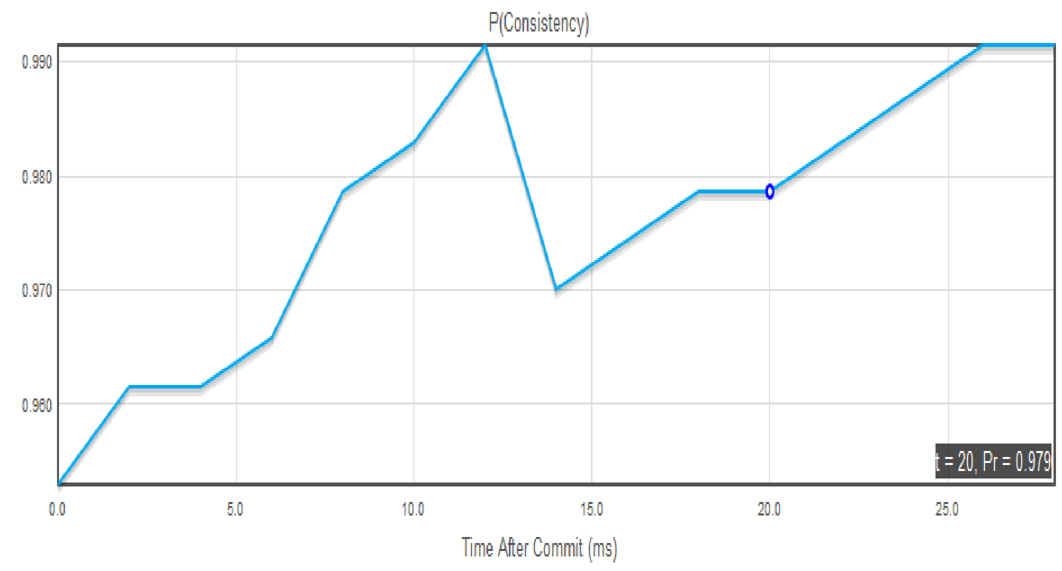


**Figure 8**

## VI. CONCLUSION

As compared to relational databases, HBASE is more suitable for handling big data as it offers massive processing power at low cost. It can handle large amount of structured, semi-structured and unstructured data. Support for frequent code pushes, agile development and fast iterations add up to its phenomenal features. Agile development is possible because schemas in HBASE are dynamic. Though HBASE will not replace SQL databases in near future, but it is compliant with any relational databases due to its numerous advantages in terms of scalability, flexibility, performance, object oriented programming support and agile development.

## REFERENCES

[1] Apache Hadoop.http://hadoop.apache.org/.

[2] APACHE HBASE TRAINING - Hadoop University http://www.hadoopuniversity.in/apache-hbase-training/

[3] Relational database characteristics – Tekstenuitleg http://en.tekstenuitleg.net/articles/software/database-design-tutorial/database-characteristics.html

[4] What is NoSQL Database & Why NoSQL | Couchbase http://www.couchbase.com/nosql-resources/what-is-no-sql

[5] HBase Performance Testing at hstack. http://hstack.org/hbase-performance-testing/

[6] Brewerâ€™s CAP Theorem Explained: BASE Versus ACID. http://ivoroshilin.com/2012/12/13/brewers-cap-theorem-explained-base-versus-acid/

[7] HBase – Cloudera. http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/hbase.html

[8] HBase Performance Testing | New IT Farmer. http://blog.newitfarmer.com/big_data/big-data_store/hbase/12905 /repost-hbase-performance-testing

[9] Can someone provide an intuitive proof/explanation of CAP. http://www.quora.com/Can-someone-provide-an-intuitive-proof-explanation-of-CAP-theorem

[10] Big Data Offerings | RapidValue Solutions. http://www.rapidvaluesolutions.com/big-data-offerings/

[11] Understanding SQL And NoSQL Databases And Different Database. https://www.digitalocean.com/community/ tutorials/understanding-sql-and-nosql-databases-and-different-database-models

[12] Digitalocean VPS: SQL And NoSQL Database http://digitaloceanvps.blogspot.com/2014/04/sql-and-nosql-databases-and-different.html

[13] BASE: An Acid Alternative - ACM Queue. http://queue.acm.org/detail.cfm?id=1394128

[14] Apache Hadoop Â« Xu Fei's Blog. https://autofei.wordpress.com/2010/07/11/apache-hadoop/

[15] Lightwolf Technologies. http://lightwolftech.com/index.php?page=backgrounders

[16] HBase Introduction – hadoopmaterial. http://www.hadoopmaterial.com/2013/11/hbase-introduction.html

[17] Vikas Kumar's Blog: RDBMS to NoSQL. http://vikaskumar9.blogspot.com/2011/08/rdbms-to-nosql.html

[18] Consistency models in nonrelational dbs. http://www.toadworld.com/products/toad-for-cloud-databases/w/wiki/320.consistency-models-in-nonrelational-dbs.aspx

[19] The Professionals Point: NoSQL vs RDBMS. http://theprofessionalspoint.blogspot.com/2014/01/nosql-vs-rdbms-why-and-why-not-to-use.html

[20] Hbase in Action – Scribd. https://www.scribd.com/doc/124675906/Hbase-in-Action

[21] Problems of RDBMS | TechLedger. https://techledger.wordpress.com/2011/07/15/problems-of-rdbms/

[22] Transactions â€" Doctrine 1.2.4 documentation. http://doctrine.readthedocs.org/en/latest/en/manual/transactions.html

[23] What is the difference between DBMS , RDBMS and SQL? http://itknowledgeexchange.techtarget.com/itanswers/what-is-the-difference-between-dbms-rdbms-and-sql/

[24] Why NoSQL? | Verteez. http://www.verteez.com/why-nosql/

[25] Accumulo - The Platform for Big Data. http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/accumulo.html

[26] SQL Interview Questions and Answers ~ Software Testing. http://www.gcreddy.com/2014/07/sql-interview-questions-and-answers.html

[27] SQL - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/SQL

[28] PBS : Probabilistically Bounded Staleness. http://pbs.cs.berkeley.edu/#demo.