Benchmark: NoSQL Databases MongoDB vs Riak vs ZHT

Abhilash Bhurse, Suyog Kharage, Ashish Wagh

abhurse@hawk.iit.edu, skharage@hawk.iit.edu, awagh@hawk.iit.edu

ABSTRACT

With most scientific applications becoming data-centric, the effective management and analysis of large scale data has become a matter of prime importance. NoSQL systems have come up as possible solutions in this regard. MongoDB and Riak are two popular NoSQL systems, each offering a variety of different features. ZHT has emerged as new system that promises to be the foundational block for future distributed systems. This calls for a comparative study of ZHT with other popular systems such as MongoDB and Riak to establish the better system or mechanism in terms of performance, scalability and latency. Through this project, we have studied and presented comparative evaluation for MongoDB and Riak. Our evaluations show that MongoDB is the better database for insert operation and performs almost 10x faster than Riak. For update operation, Riak performs better than MongoDB with its performance being approximately 2x faster. For search operation, Riak shows a consistent performance, but performance of MongoDB is seen to be increasing linearly with the increase in number of records. Both databases show a constant and nearly same performance for delete operation.

1. INTRODUCTION

Today's scientific applications, which are generating data on a very large scale, are getting more and more data-centric rather than compute-centric, whereas the storage infrastructure is not developing at a good enough rate to cope with the storage and archival of such huge amount of data. This makes the management and analysis of data of such huge scale a matter of prime importance. Traditional Relational databases prove inefficient for dealing with data of huge scale. This is where NoSQL databases come into picture. A NoSQL (originally referring to "non-SQL" or "non-relational") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. These databases are widely used in Big-Data applications. NoSQL might also support SQL-like query languages as well. The data structures used by NoSQL databases (e.g. key-value, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL.

Design philosophy of MongoDB is focused on combining the critical capabilities of relational databases with the innovations of NoSQL technologies. NoSQL offers Expressive query language & secondary Indexes, Strong consistency and Enterprise Management and Integrations. Users should be able to access and manipulate their data in sophisticated ways to support both operational and analytical applications. For providing efficient access to data, Indexes play a critical role and supported natively by the database rather than maintained in application code. Applications should be able to immediately read what has been written to the database. As databases are just one piece of application infrastructure, organizations need a database that can be provisioned, secured, monitored, upgraded, and integrated with their existing technology infrastructure, processes, and staff, including operations teams, DBAs, and data engineers [10].

MongoDB offers Flexible Data Model, Scalability and Performance and Always-On Global Deployments. To address the requirements for the data we see dominating modern applications, MongoDB databases emerged. Sharding or partitioning allows the database to be scaled out across commodity hardware deployed on-premises or in the cloud, which enables higher growth and throughput and lower latency than relational databases. It also provides consistent, high quality experience for users all over the world using Always-On Global Deployments which enables replication to automatically synchronize data across servers and data centers. Flexible document data model allows data representation as simple key-value pairs and table structures and objects with nested arrays and nested documents. Documents can be queried in many ways using expressive query language. Also, application owners can deploy storage engines for different workload and various operational requirements using flexible storage architecture. MongoDB query model provides Querying and Visualizing Data, indexing and query optimization. Features like auto sharding, consistency, availability, end-to-end compression and many security features are offered by MongoDB data management. With scalability and flexibility that you want with querying and indexing, MongoDB stores data in flexible, JSON-like documents i.e. data structure can be changed over time. User can access and analyze the data using Ad hoc queries, real time aggregation and indexing. Instead of using tables and rows as in relational databases, MongoDB is built on an architecture of collections and documents.

Riak is another popular NoSQL key-value store database which is highly available. One of the biggest differences between Riak and relational systems is availability. Riak is designed to be deployed to, and runs best on, multiple servers. In the presence of hardware and network failures, it continues to function normally. Most relational databases offer a master/slave architecture for availability, where master server is available for data updates. If the master fails, the slave takes over. Riak is explicitly designed to expect server and network failure., Any server can respond to read or write requests because Riak is a master less system. After failure of one server, others will continue to service client requests. Once this server becomes available again, the cluster will feed it any updates that it missed. Riak's system allows for reads and writes when multiple servers are offline or otherwise unreachable, data may not always be consistent across the environment. All updates will propagate to all servers making data eventually consistent using mechanisms like read repair and Active Anti-Entropy. Riak provides features such as fault tolerant availability, multi-datacenter replication and tunable consistency. With Riak, we can query for PUT, GET, POST and DELETE functions. We can store key-values in both memory and disk. Key-value Design of Riak provides powerful data model to store unstructured data of large amount [11].

A new system –ZHT, has emerged recently, which is a zero-hop distributed hash table, which has been tuned for the requirements of high-end computing systems. ZHT aims to be a building block for future distributed systems, such as parallel and distributed file systems, distributed job management systems, and parallel programming systems. The goals of ZHT are delivering high availability, good fault tolerance, high throughput, and low latencies, at extreme scales of millions of nodes. ZHT has some important properties, such as being light-weight, dynamically allowing nodes join and leave, fault tolerant through replication, persistent, scalable, and supporting unconventional operations such as append (providing lock-free concurrent key/value modifications) in addition to insert/lookup/remove.

2. PROPOSED SOLUTION

Through this project we aim to carry out an empirical performance evaluation of MongoDB and Riak and present the results based on metrics such as latency, performance and scalability. We also aim to study and compare similar features of these systems and propose the better system for specific real-world applications such as **Hospital Record Maintenance System**. We would mainly evaluate these systems based on the basic operations supported by each system, such Create, Read and Delete in MongoDB and PUT, GET, DELETE in Riak and compare the results. Datasets would be roughly in the range of 10K to 150K records. We would test the scalability of each of these systems up to 8 to 16 nodes depending on availability on the Chameleon Cloud Platform. We intend to perform strong scaling experiments by keeping the workload fixed and increasing the number of nodes. The evaluation will be illustrated through the use of various comparative graphs that compare the performance and scalability of MongoDB and Riak.

There have been a variety of NoSQL systems that have emerged in recent years that employ different techniques for storage and analysis of largely unstructured data and each one claims to be better at some or the other aspect. MongoDB is Document Store system that supports field, range queries and regular expression searches, and uses Grid File System for File Storage. Riak is a key value store system that provides support for basic PUT, GET, POST and DELETE functions. A new system ZHT has emerged recently, which is a zero-hop distributed key value store system, and promises to be a building block for future distributed systems. This calls for a comparative study of these popular systems to establish the better system or mechanism in terms of performance, scalability and latency. We take a real life scientific application- a **Hospital Record Maintenance System,** and will try to evaluate the performance of different NoSQL databases for basic operations that such a system would require to do on a daily basis. As the medical records of more and more people get inserted into the system each day, migration of such systems, which mostly run on top of a relational databases currently, into a NoSQL based database is imminent. Keeping this scenario in mind, establishing a NoSQL database which would be more suitable for operations such as insert, update, search and delete, would prove to be of immense help for quicker selection of a NoSQL database to migrate to.

The main motivation behind analyzing the time required for inserting, searching, updating and deleting records for both the systems is to establish a system that is better suited for a migration of Relational Database to a NoSQL Database. With the scale of data growing larger each day, many applications, such as a Hospital Record Maintenance System, which currently runs on a relational database, might need to migrate to a NoSQL database in the future to better manage the huge scale of data and store the data for archival purposes. Such archived data, stored in a NoSQL database with many different features, would prove to be of great help for other big data applications that could come up with a variety of valuable information, patterns and trends related to the health issues of people over the years. This might prove to be very useful for a variety of different purposes, for example, establishing a suitable cure for certain deceases.

Implementation:

The code operates in the following sequence:

1. Establish connection to database.

2. Create an instance of database client.

3. Read data from input excel file.

4. Use the instance created to send data to the database.

5. Perform insert, search, update and delete operations.

6. Calculate and Display the time required for each of the operations.

The code is written in Java Programming Language. The code for MongoDB is of 177 lines, whereas for Riak it is 187 lines. MongoDB implementation has dependencies on Poi JAR (and its own dependencies) for reading the input records from an excel file and Mongo Java Driver for connecting to the Mongo Database. Riak implementation has dependencies on Poi JAR (and its own dependencies) for reading the input records from an excel file and Riak Client and Riak Java Client for connecting to the Riak Database.

The relevant source code can be accessed through the following link:

https://drive.google.com/open?id=1khxSDQI28RN6RbAbzSlUnF2xkBYzYylX

3. EVALUATION

In this section, we present the comparative evaluation of MongoDB and Riak's performance, in terms of basic operations supported by each of them such as Insert, Search, Update and Delete. Firstly, we describe the configuration of test beds and benchmark setup. Secondly, we presented a comprehensive performance evaluation. We were able to install ZHT but could not get the ZHT servers up and running. One possible reason may be that the Operating system version was not supported, since ZHT is not under active development since the last two years. Hence the comprehensive evaluation of only two databases – MongoDB and Riak is presented in this section. We intend to explore the features and functions of ZHT as part of future work of this project.

3.1. Testbed, Metrics, and Datasets

We used Chameleon Cloud – OpenStack KVM platform to run the experiments. The experiments were performed on 1,2,4 and 8 virtual CPUs set up on small, medium, large and extra-large instances of Chameleon Cloud- OpenStack KVM platform respectively. Following are the relevant specifications of each instance type:

Instance Type	**VCPUs**	**Disk Space**	**RAM**
m1.small	1	20 GB	2048 MB
m1.medium	2	40 GB	4096 MB
m1.large	4	80 GB	8192 MB
m1.xlarge	8	160 GB	16384 MB

Table.1 Chameleon Cloud OpenStack KVM instance details

On each instance an Ubuntu Linux (version 16.04) operating system was installed, followed by installing Java Development Kit (JDK) version 8 on top of the operating system. This was followed by installation of MongoDB and Riak databases on each instance.

The metrics measured and reported are:

Performance :

Insert Performance: The time taken by the system to insert all the records input into the database, measured in milliseconds (ms).

Search Performance: The time taken by the system to search a single record among all the records previously inserted into the database, measured in milliseconds (ms).

Update Performance: The time taken by the system to update a single record at the end of the records present in the database, measured in milliseconds (ms).

Delete Performance: The time taken by the system to delete a single record among all the records previously inserted into the database, measured in milliseconds (ms).

Scalability : The scalability of both the systems up to 8 nodes.

The datasets were in the range of 10K to 130K, starting with 10K and followed by 30K, 50K, 80K,100K and 130K. Each file contained sample medical data with the key elements being the Patient Id and the Patient Name.

3.2. INSERT EVALUATIONS

For the dataset of 10K records, the time taken by MongoDB to insert the records was found to be much less than that taken by Riak. MongoDB was found to be almost 10x faster than Riak on 1,2,4 and 8 nodes.

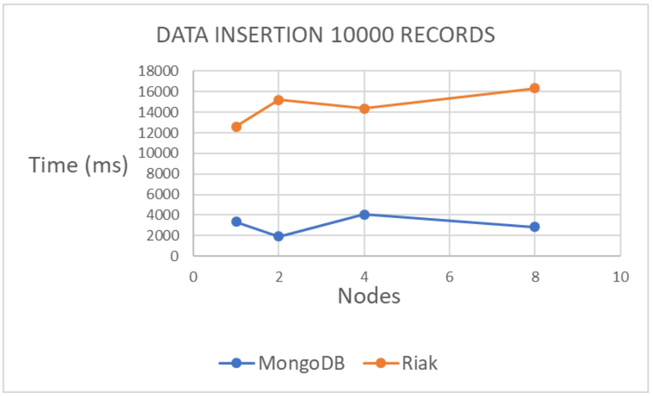

Fig.1 Data Insertion – Constant-Number of Records(10K), Variable- Number of Nodes

For higher datasets of up to 130K records, the trend was found to be the same. As the number of records increase, the time taken to insert the records increases for both the systems, but MongoDB is always at least 10x faster than Riak throughout.

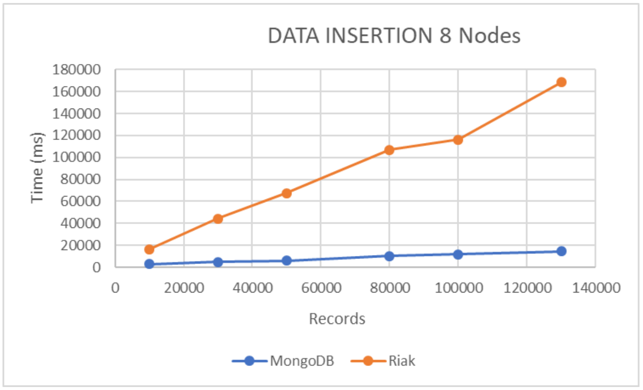

Fig.2 Data Insertion – Constant Number of Nodes(8), Variable- Number of Records

The higher time taken by Riak for inserting records into the database can be attributed to the fact that Riak creates a Key-Value pair for each record before inserting the record. For each record an object is created first, and each object is stored in a key. Each key is stored in bucket, which has a particular bucket type. Additionally, content type and value for each object has to be set. Thus, it spends extra time on additional functions before the actual insertion and eventual storage of a record can be performed.

Whereas for MongoDB, the records (called documents in MongoDB terms) are inserted into a collection. The insert() method is used to insert records into the a collection. If the collection is not present, MongoDB creates a collection first and then inserts the records. Thus, it takes comparatively less time for MongoDB to insert the records. Thus, this trend is synonymous with the expected trend.

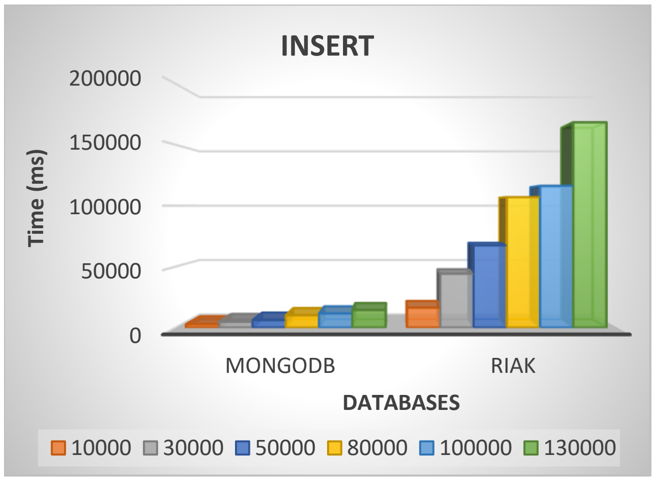

Fig.3 Data Insertion – Constant Number of Nodes(8), Variable- Number of Records

3.3. SEARCH EVALUATIONS

For Search operation, the time taken by both the systems to search a single record is much less than that for inserting records. The values range from 8 ms to 145 ms.

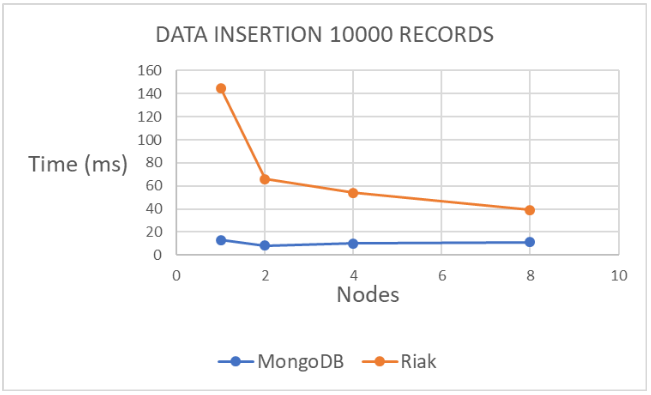

Fig.4 Record Search – Constant-Number of Records(10K), Variable- Number of Nodes

For datasets of 10K records, MongoDB shows better performance than Riak, as the number of nodes increases from 1 to 8. But as the size of datasets increases to 130K records, Riak starts showing a constant performance, that is, searching time becomes almost constant for Riak as the number of records increase.

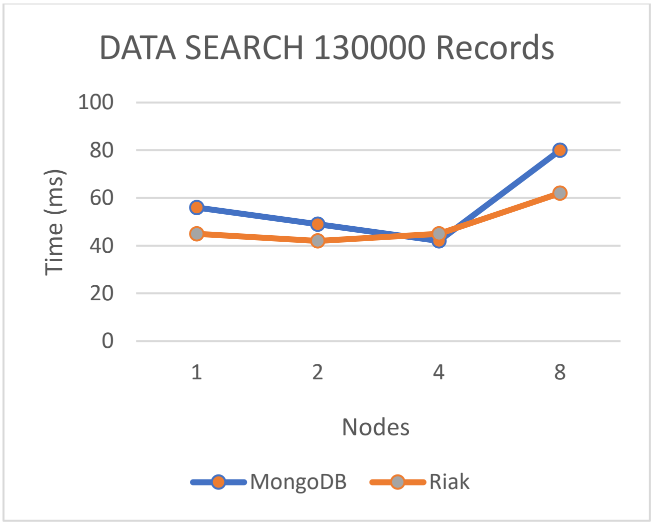

Fig.5 Single Record Search – Constant-Number of Records(130K), Variable- Number of Nodes

For example, for 8 nodes, as we keep increasing the number of records from 10K to 130K, an interesting trend is observed. The time taken by MongoDB is found to be increasing as the number of records on which search has to be performed increases. Whereas for Riak, the time is almost constant. This indicates the time required to search one record in Riak would take the same time irrespective of the size of the records once the records reach a higher range.

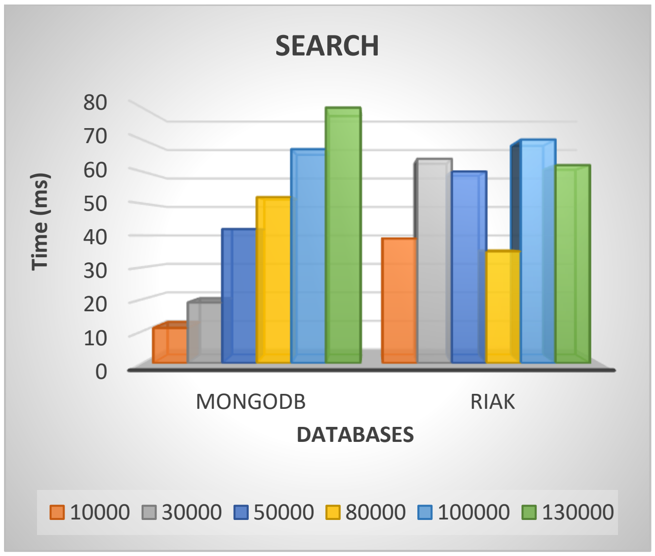

Fig.6 Single Record Search – Constant-Number of Nodes(8), Variable- Number of Records

This behavior indicates that searching operation in Riak is independent of the record size. On the other hand, the search time for MongoDB exhibits a dependency on the records size. To check the validating of this behavior we decided to perform the search operation for multiple records rather than a single record. We modified the code to run in a loop for searching 500 records.

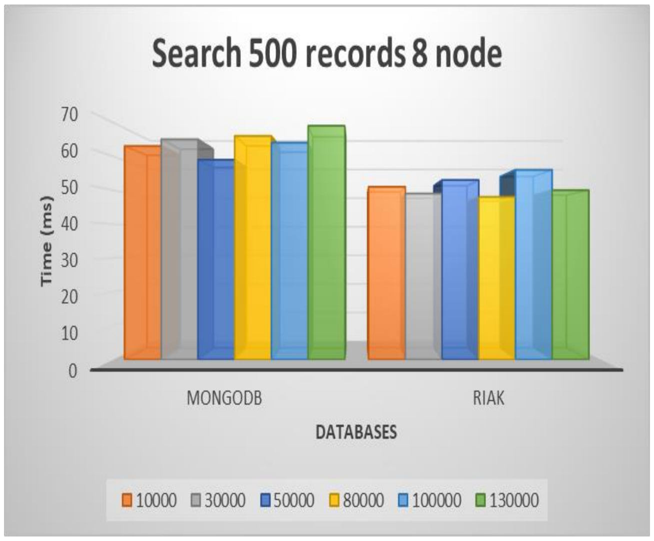

Fig.7 Multiple Record Search – Constant-Number of Nodes(8), Variable- Number of Records

As can be seen from Fig.7, even though the time taken by Riak for search is still less than MongoDB, the time for searching multiple records is almost the same for MongoDB for variable record size. This contradicts the trend observed for searching a single record, indicating some time is spent on other operation before actual search operation is performed in MongoDB for a record search, which also likely explains the extra bit of time it takes more as compared to Riak.

3.4. UPDATE EVALUATIONS

For update operation, Riak performs consistently better than MongoDB for different datasets and on different nodes. Riak is approximately 2x faster than MongoDB for update operations.

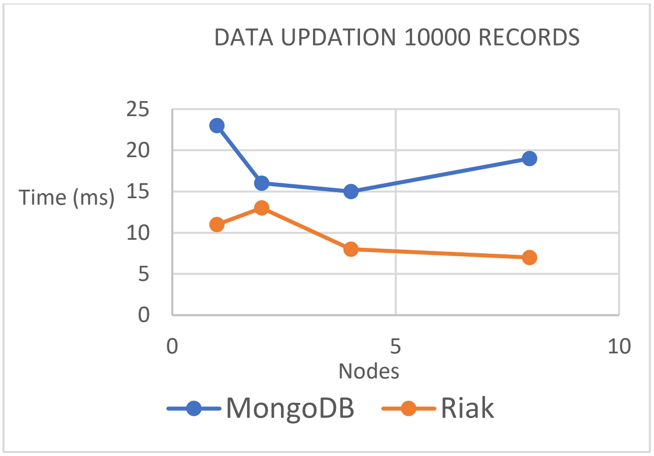

Fig.8 Single Record Update – Constant-Number of Records(10K), Variable- Number of Nodes

Both Riak and MongoDB show a consistent performance for different datasets and for different nodes.

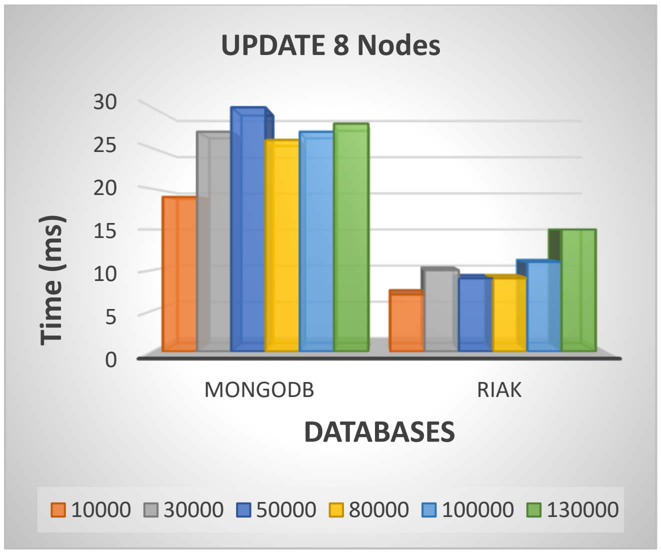

Fig.9 Single Record Update – Constant-Number of Nodes(8), Variable- Number of Records

From Fig.8 and Fig.9 it is evident that Riak performs better than MongoDB for update operations.

3.5. DELETE EVALUATIONS

Delete operations provide less insightful information concerning the performance of MongoDB and Riak. Both systems take approximately the same amount of time for deleting a record from the database.

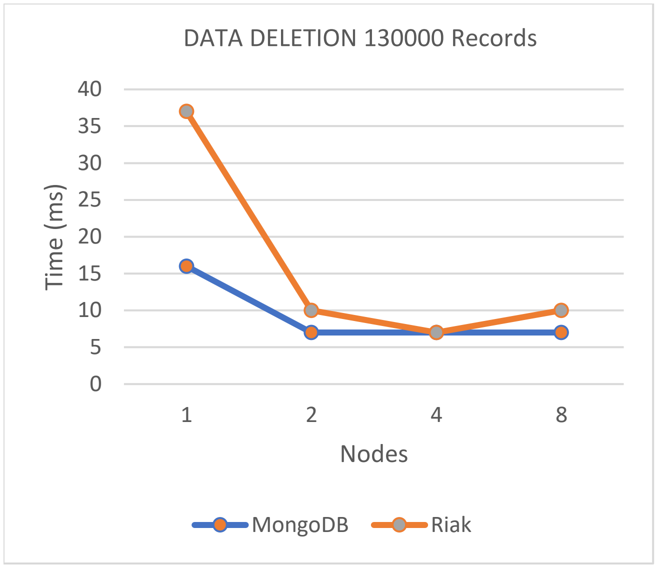

Fig.10 Single Record Delete – Constant-Number of Records(130K), Variable- Number of Nodes

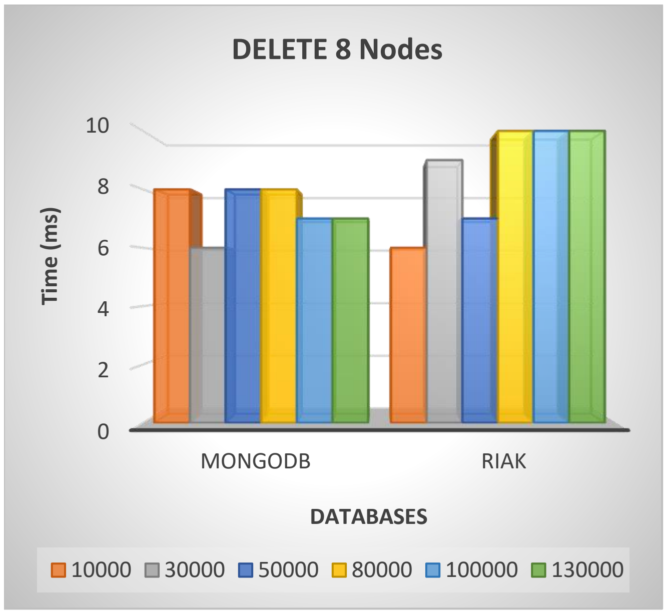

Fig.11 Single Record Delete – Constant-Number of Nodes(8), Variable- Number of Records

Fig.10 shows that on a single node, delete operation is faster for MongoDB as compared to Riak, but as the number of nodes increase, the time for deletion becomes almost constant and almost the same for both the databases.

Fig.11 shows the time taken to delete one record for Datasets ranging from 10K to 130K on 8 nodes. As evident, the time remains almost constant for both databases.

3.6. SCALABILITY

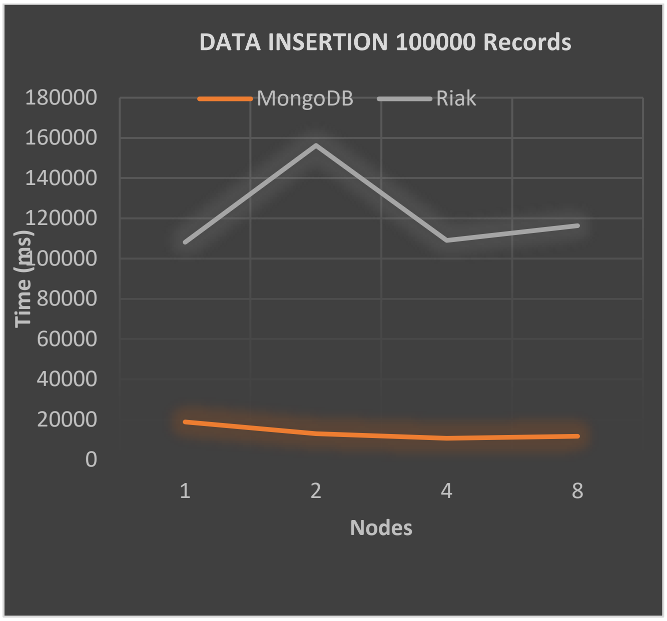

Fig.12 Data Insert – Constant – Number of Records(100K),Variable- Number of Nodes

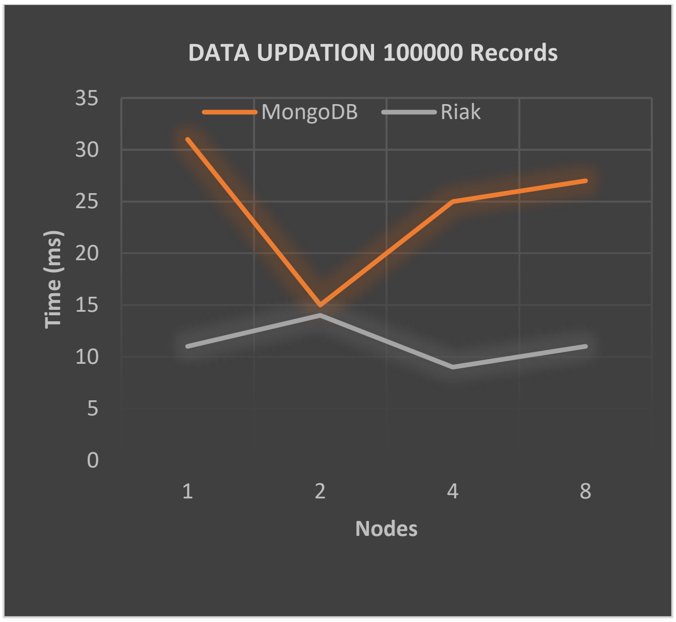

Fig.13 Data Update – Constant-Number of Records(100K),Variable- Number of Nodes

Fig.12 shows that MongoDB scales consistently well for insert operation as we move from a single node to 8 nodes. Whereas Riak shows inconsistent performance.

On the other hand Riak scales consistently well for update operation as we move from a single node to 8 nodes. Whereas MongoDB shows an inconsistent performance.(refer Fig.13)

The above presented evaluations show that MongoDB is the better database for insert operation and performs almost 10x faster than Riak. For update operation, Riak performs better than MongoDB with its performance being approximately 2x faster. For search operation, Riak shows a consistent performance, but performance of MongoDB is seen to be increasing linearly with the increase in number of records. Both databases show a constant and nearly same performance for delete operation.

4. RELATED WORK

A lot of papers have been published and many research works are still going on related to comparison of NoSQL databases. Paper related to comparison between MongoDB and MySQL compares the advantages of MongoDB over the MySQL databases. This paper compares MongoDB with relational database - MySQL, where operations like insert, select, update and delete are performed. MongoDB provides lower execution times than MySQL in all four basic operations [2].

MongoDB can be used for bigger applications like forums that may be having ten thousand of users. But there is no evaluation whether MongoDB is better or worse as compared to other NoSQL databases like Riak.

There are some papers published which compare the same category NoSQL databases such as Hbase and Cassandra, both of which are key-value store category. In write intensive environment, to read in a given time, Cassandra executes more operations than Hbase and to write in a given time, Hbase executes more operations than Cassandra. This is the same case in read intensive environment [1][6]. Paper by Bartholomew gives a tutorial introduction to the history of and differences between SQL and NoSQL databases [7]. There is another paper which provides a comprehensive treatise on NoSQL databases with comparing scalability, consistency, support for data model, queries and management tools [8]. In terms of write performance, NoSQL databases like MongoDB, Cassandra, Hypertable and Couchbase perform better than SQL Express. The read performance of SQL Express is better than some, but not all, of the NoSQL databases [9]. NoSQL databases have high availability and are high-performance databases without compromising the ACID properties of databases. NoSQL databases are used for heavy read/write operations which are not supported by RDBMS databases. Also, there are various taxonomies for the NoSQL database classification. But there is less information available for differences between other category of databases like Document store (MongoDB) and key-value store (Riak).

ZHT is a zero-hop distributed key-value store system, which has been modified for the requirements of HPC [3]. With ZHT we can have additional newer operations like append, compare and swap along with the traditional operations-insert, lookup and remove present in RDBMS. ZHT has also been compared with other distributed hash tables and key/value stores and found it offers superior performance for the features and portability it supports. [4] Some real-time systems like FusionFS and Istore have adopted ZHT.

5. CONCLUSION

From the evaluations we conclude that MongoDB is much better in performance than Riak for insert operation. It performs 10x faster than Riak. So for an application that would require migration of existing relational database into NoSQL database, MongoDB would prove to be a better system to migrate to, since it would take less time for inserting the records to MongoDB.

For searching a single record, Riak performs better than MongoDB. Riak’s performance is consistent over multiple datasets and multiple nodes. However, for searching multiple records, MongoDB also shows a consistent trend. This leads us to conclude that MongoDB spends additional time on certain other operations before performing the actual search operation. Keeping in mind that any big scientific application would require to search more than a single record at times, selection of any of the two databases would yield a similar performance. Riak’s performance for update operation is approximately 2x better than that of MongoDB. Hence if the application requires frequent update operations, Riak would prove to be a better system than MongoDB. For delete operation both the databases perform similarly. Hence selection of any database would yield a similar performance.

At the beginning of this project we set out with a goal to come up with a specific evaluation illustrating the differences between the popular NoSQL systems in terms of the basic operations supported by each. By the end of this project we were able to come up with and present the evaluations that we had set out to achieve. Hence we conclude this project to be a success.

FUTURE WORK

If we get a chance to work further on this project, we would like to explore and test other parameters related to these systems. We would also try to install ZHT and get the ZHT servers up and running, so that we could create an application on top of ZHT and compare the results with the ones which we have presented in this report. Since both MongoDB snd Riak support clustering, we would also try evaluating the operational performance of the distributed database i.e. clustering of the database instances so that we could be able to analyze the scalability of the databases on the distributed environment more precisely.

6. REFERENCES

[1] Tudorica B.G., 2011, A comparison between several NoSQL databases with comments and notes,10.1109/RoEduNet.2011.5993686

[2] Gyrödi C., Gyrödi R., Pecherle G., Olah A., 2015, A Comparative Study: MongoDB vs. MySQL, 978-1-4799-7650-8/15

[3] Li T., Zhou X, Wang K., Zhao D., Sadooghi I., Zhang Z., Raicu I.,2015, A Convergence of Key-Value Storage Systems from Clouds to Supercomputers, 10.1002/cpe

[4] Li T., Zhou X, Brandstatter K., Zhao D., Wang K., Rajendran ., Zhang Z., Raicu I., 2013, A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table, 10.1109/IPDPS.2013.110

[5] Nayak A., Poriya A., Poojary D., 2013, Type of NOSQL Databases and its Comparison with Relational Databases, 2249-0868

[6] Cooper, Brian F., “Yahoo! Cloud Serving Benchmark”, http://research.yahoo.com/files/ycsb-v4.pdf, (unpublished)

[7] D. Bartholomew, “SQL vs. NoSQL,” *Linux Journal*, no. 195, July 2010

[8] S. Tiwari, *Professional NoSQL*. Wiley/Wrox, August 2011

[9] Yishan Li and Sathiamoorthy Manoharan, 2013, A performance comparison of SQL and NoSQL, 10.1109/PACRIM.2013.6625441

[10] MongoDB Architecture Guide - https://webassets.mongodb.com/_com_assets/collateral/MongoDB_Architecture_Guide.pdf?_ga=2.228649438.314929589.1512174707-946253448.1511970696

[11] Relational to Riak – High Availability - http://basho.com/posts/technical/relational-to-riak-high-availability-part-1/

APPENDIX:

Abhilash : Study of ZHT, Creating instances on Chameleon Cloud OpenStack KVM, Installation of JDK version 8 on instances, Installation of ZHT, MongoDB and Riak on Chameleon Cloud instances, Writing code for update operation on MongoDB, Recording and noting timing statistics of all the operation, Creating Graphs, Preparing PowerPoint Presentation, Preparing Technical Report.

Suyog : Study of Riak, Creating instances on Chameleon Cloud OpenStack KVM, Installation of JDK version 8 on instances, Installation of ZHT, MongoDB and Riak on Chameleon Cloud instances, Writing code for update operation on Riak, Recording and noting timing statistics of all the operation, Creating Graphs, Preparing PowerPoint Presentation, Preparing Technical Report.

Ashish : Study of MongoDB, Creating instances on Chameleon Cloud OpenStack KVM, Installation of JDK version 8 on instances, Installation of ZHT, MongoDB and Riak on Chameleon Cloud instances, Writing code for insert, search and delete operations on MongoDB and Riak, Recording and noting timing statistics of all the operations, Creating Graphs, Preparing PowerPoint Presentation, Preparing Technical Report.

Graphs:

We came up with a variety of different graphs that throw a bit more light on the specific differences between MongoDB and Riak, but could not include all the graphs in this report. These additional graphs can be accessed through the following link :

https://drive.google.com/open?id=1wrCOZCFgSuAmG0DqegGjGP-raeyzMBjK