2016 IEEE International Congress on Big Data

Quantitative Analysis of Scalable NoSQL Databases

Surya Narayanan Swaminathan, Ramez Elmasri
University of Texas at Arlington, CSE
Arlington, TX 76019
surya.swaminathan@mavs.uta.edu, elmasri@uta.edu

***Abstract*—NoSQL databases are rapidly becoming the customary data platform for big data applications. These databases are emerging as a gateway for alternative approaches outside traditional relational databases and are characterized by efficient horizontal scalability, schema-less approach to data modeling, high performance data access, and limited querying capabilities. The lack of transactional semantics among NoSQL databases has made the choice of a particular consistency model dependent on the application. Therefore, it is essential to examine methodically, and in detail, the performance of various databases under diverse workload conditions. Three of the most commonly used NoSQL databases: MongoDB, Cassandra and HBase are evaluated using the Yahoo Cloud Service Benchmark, a popular benchmark tool. The horizontal scalability of the three systems under different workload conditions and varying dataset sizes is captured. A benchmark suite which summarizes the results of the evaluation is presented.**

***Keywords*-NoSQL; YCSB; MongoDB; Cassandra; HBase; Database Benchmark;**

I. Introduction

The exponential increase in the number of Internet users and social media platforms and the consequent increase in velocity and size of the data generated have showcased the shortcomings of traditional database management. This necessitated the emergence of highly distributed and scalable database systems. The difficulty in managing very large amounts of data in a distributed database, a rigid relational schema, the lack of conformance of data to a structured format propelled the developer community to create a new kind of data store alternative to the traditional relational stores.

NoSQL databases have emerged as a solution to the aforementioned drawbacks and have become the preferred storage option for big data applications. Currently, there are more than 225 recognized NoSQL databases [1]. Except for non-relational characteristics, NoSQL databases are quite diverse, and so some may not have good performance results under a given workload condition.

The basic operations in a database can be formulated from one or more of the following: Create, Read, Update, and Delete (commonly referred as CRUD). Datastores can be tailored to handle varied workloads of CRUD operations to satisfy the requirements of specific applications. Therefore, it is necessary to identify, among the available databases, the optimal NoSQL database for a given application workload.

The performance of an application in a globally distributed environment is influenced by (but not limited to) the ability to scale horizontally, the total turnaround time for each request and the risk mitigation strategy to handle partial failure. Hence, it is a challenge to determine the optimal number of nodes for which the application will have the best performance. A scalability study that benchmarks the performance of different databases would assist in planning the hardware/software requirements for the application.

Few previous works [2][3][4] on benchmarking NoSQL databases and comparative study reports mainly focused on overall execution time of different NoSQL databases on a standalone system. In some published works [5], scalability of a specific NoSQL database was analyzed with fixed operation count and fixed record size. Prominent work on the performance analysis [6] of NoSQL databases concentrated on latency vs. throughput trade-off in a high performance computing environment. In contrast, we perform a comprehensive performance analysis of different NoSQL databases on their ability to scale horizontally with varying record sizes, dataset sizes and varying operation workloads.

In this paper, we aim to provide guidance that would map the application requirements to a suitable NoSQL database. We also find the number of nodes in a cluster for which a NoSQL database produces the best performance. We consider three most popular NoSQL databases MongoDB(document based) [7], HBase(column based) [8], Cassandra(hybrid) [9] and the performance of each database on its ability to scale with different dataset sizes, operation counts and CRUD operations is analyzed. Yahoo! Cloud Service Benchmark (YCSB) [6], a NoSQL benchmarking tool, was used for our experimentation.

The organization of this paper is as follows. In Section 2, the experimental setup is described. In Section 3, the benchmark testing results for all three databases will be compared and analyzed and a summary of the benchmark experiments is presented. Section 4 provides the conclusion.

II. Experimental Setup

For experimentation, a standard, flexible and customizable benchmarking platform was required. Yahoo! Cloud Service Benchmark [6] (YCSB), is the defacto choice for benchmarking NoSQL databases. The benchmark tool facilitates synthetic data generation, which is a set of random characters indexed by a primary identifier. It has a workload executor to handle multiple client threads and execute the defined CRUD operations. To achieve the target of identifying a suitable database for a predefined set of database operations, we customized some workloads in accordance with current trends in big data applications in addition to some standard workloads. Table I shows the workloads defined for this benchmark testing.

Table I
WORKLOADS DEFINITION

Workloads	Applications
50% Read - 50% Write	Session stores
100% Read	Hadoop
100% Blind Write	Logging
100% Read-Modify-Write	User profile updates
100% Scan	Search operations

In order to analyze the behavior of NoSQL databases on varying record sizes, we generated records with 10 fields and records with 20 fields. Additionally, we generated records with size of each field set to 100 bytes and 200 bytes. The different dataset sizes considered for this benchmark testing are presented in Table II.

Table II
DATASETS

Record count	Record size (for each record)	Total size (of the dataset)
1 Million	1 KB	1 GB
1 Million	4 KB	4 GB
10 Million	1 KB	10 GB
10 Million	4 KB	40 GB

In creating test cases close to real world applications, we used zipfian strategy to distribute the data. This distribution strategy places the extremely popular records at the top of the distribution and least popular records at the tail. Similarly, for scanning a record, zipfian distribution was used for HBase and MongoDB. But, with Cassandra, we encountered time-outs for scan operations with zipfian distribution. Hence, we set the scan distribution strategy to uniform for Cassandra. The number of operations executed (Read, Write, Scan, etc.) on the database was increased proportionally to analyze the variations in throughput performance (operations per second) with respect to operation count. We calculated the mean of the throughput to avoid noisy results.

In order to analyze the pattern in performance of the database and to find the optimal size of the cluster required for a given dataset size, the experiments were executed on six scenarios: 2, 3, 5, 6, 12 and 13 nodes. The experiments were performed on a cluster made of 14 servers connected through 1 Gigabit Ethernet Switch where each server is running the CentOS. A single server consists of Xeon CPU, 4 GB RAM, and a 1.4 TB hard disk. One of the node acts as master, which consists of a 2.7 TB hard disk. Each database was configured on a horizontal cluster. Following are some of the configurations of these databases: MongoDB (3.0.6), config_servers: 1, sharding: Range partitioning; Cassandra (2.0.16), sharding: Murmur3Partitioner, key_cache: 100 MB, row_cache: None; HBase (1.1.0), sharding: auto, Hadoop (2.2.0). For this benchmark testing, executing a specific workload condition on a database by fixing the dataset size, cluster size and operation count defines a test case. The scope of the experiments can be viewed in Figure 1. A total of 4 (Data sizes) * 4 (Operation counts) * 5 (Workloads) * 6 (Cluster sizes) = 480 experiments were conducted for each database.

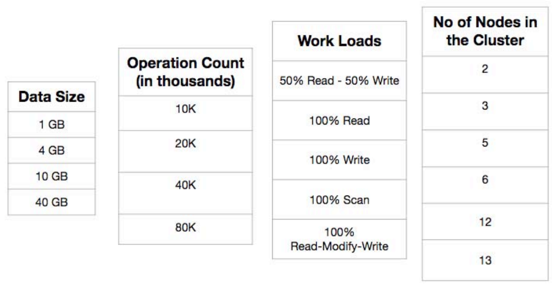


Figure 1. Comprehensive view of the scope of the experiments

III. EVALUATION

This section evaluates the results of the benchmark testing. The benchmarking results for the three databases: MongoDB, HBase and Cassandra will be compared. For a defined workload condition and dataset size, the throughput performance of the three databases are plotted against increasing cluster size. We present our analysis on the performance of each system and the reasoning contributing to the behavior of the systems.

A. 50% Read - 50% Write

The results of execution of 50% read - 50% write workload is shown in Figure 2. From the results, we observed that Cassandra had a better throughput performance overall. However, for a small database with 1 GB of data and medium database with 4 GB of data, HBase scaled better with approximately 20% better throughput performance than Cassandra when cluster size is increased to more than 12 nodes. The over distribution of data and the overhead associated with the communication protocol (gossip) may be the reason for the decrease in performance of Cassandra for a small dataset sizes. In case of HBase, the autonomous functioning of slave nodes in a master-slave architecture influences the scalability.

B. 100% Read

Given the circumstance where the database executes only read operations, our experiments indicate that MongoDB

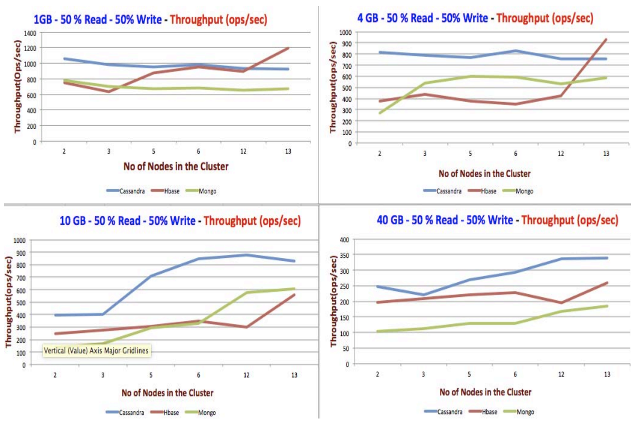


Figure 2. MongoDB vs. Cassandra vs. HBase: 50% Read - 50% Write.

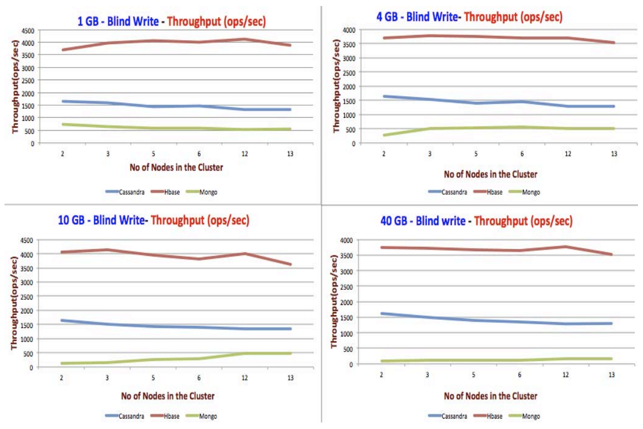


Figure 4. MongoDB vs. Cassandra vs. HBase: 100% Blind Write.

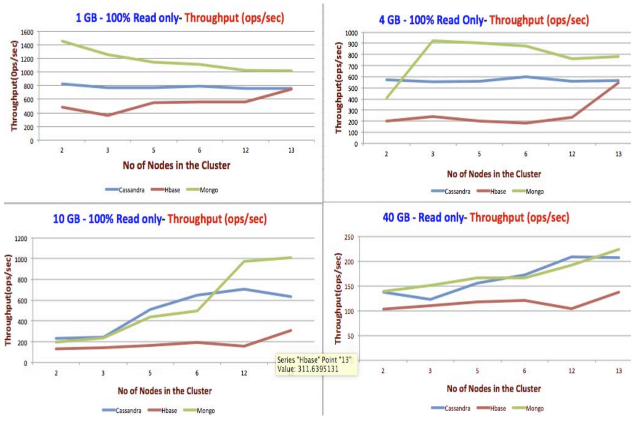


Figure 3. MongoDB vs. Cassandra vs. HBase: 100% Read.

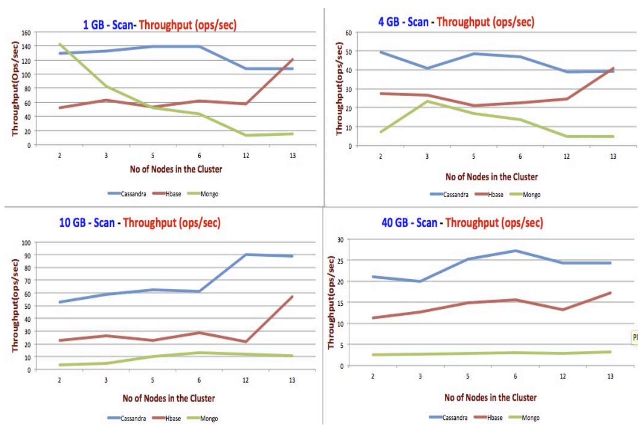


Figure 5. MongoDB vs. Cassandra vs. HBase: 100% Scan.

was the obvious choice (see Figure 3). MongoDB stores the data as BSON (Binary JSON) documents [7] and this could be the reason for the better throughput performance for read only operations. In MongoDB, the data is horizontally partitioned into shards, known as chunks. For this experimentation, the size of each chunk was limited to 64 MB. Data is assigned to a chunk until it reaches the limit and the chunk is flushed to a node in the cluster. It was observed that MongoDB achieves best throughput performance when the chunk distribution per node was approximately between 10 and 13 chunks.

C. 100% Blind Write

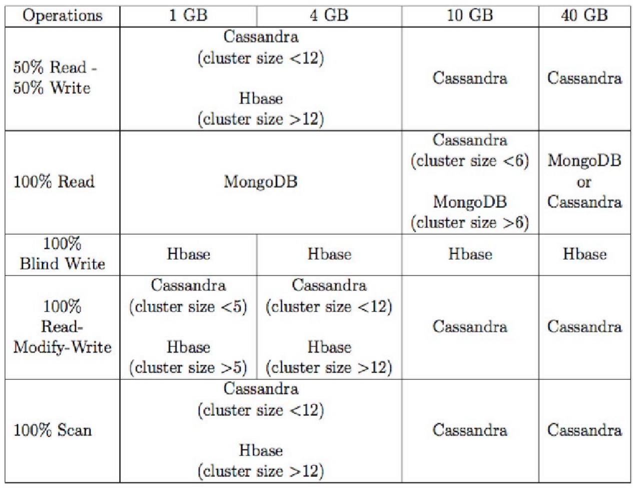
In our experiments, HBase had the best performance for this workload, irrespective of the size of the database and the size of the cluster, with 265% better throughput performance than Cassandra (see Figure 4). Though HBase and Cassandra are modeled after Google's Bigtable [10], the difference in the performance is due to their different approach in handling the write requests. An observation with HBase unique to this workload was the increase in the throughput performance with the increase in the number of operations. Approximately 300% increase in throughput was observed with the increase in the number of operations. These results indicate that HBase is designed to handle large amounts of write requests more efficiently.

D. 100% Read-Modify-Write

The performance behavior exhibited in this workload was similar to 50% read- 50% write workload. Cassandra outperformed the other two databases for large sets of data (10 GB and 40 GB). For a small set of data (1 GB), HBase achieved better throughput.

E. 100% Scan

In contrast to all the other workloads a scan operation is executed over a range of records, similar to range query in SQL. In this benchmark testing the range is set to 100 records, which means that, for each scan operation a specific range of 100 records are read. Therefore, the performance of



Operations	1 GB	4 GB	10 GB	40 GB
50% Read - 50% Write	Cassandra (cluster size <12) Hbase (cluster size >12)		Cassandra	Cassandra
100% Read	MongoDB		Cassandra (cluster size <6) MongoDB (cluster size >6)	MongoDB or Cassandra
100% Blind Write	Hbase	Hbase	Hbase	Hbase
100% Read-Modify-Write	Cassandra (cluster size <5) Hbase (cluster size >5)	Cassandra (cluster size <12) Hbase (cluster size >12)	Cassandra	Cassandra
100% Scan	Cassandra (cluster size <12) Hbase (cluster size >12)		Cassandra	Cassandra

Figure 6. Summary

a database to execute scan operations is directly influenced by the partitioning technique implemented for that database. Figure 5 shows the performance of the three databases for 100% scan workload. Cassandra had the best throughput performance for large data sets (10 GB and 40 GB), approximately twice as efficient as HBase. In case of small (1 GB) and medium (4 GB) datasets, sparse distribution of data caused a decrease in the performance of Cassandra with increase in the cluster size. Conversely, performance of HBase increased with the increase in the size of the cluster. The results of 100% scan workload indicate that MongoDB is not well suited to handle range queries.

F. *Summary*

The results discussed in this section are summarized as a benchmark suite in Figure 6. This benchmark would assist in choosing the suitable NoSQL database for the defined workload conditions and dataset sizes. Additionally, the information regarding the size of the cluster for which the database exhibits the best performance would serve as guidance while planning the hardware/software resources for an application.

IV. Conclusion

Different NoSQL databases have various design features, which are advantageous for specific types of operations. It is valuable to examine the behavior of different NoSQL databases in order to make an informed decision on the choice of database suitable for an application. In this work, three of the most popular open source NoSQL databases, MongoDB, Cassandra and HBase, were analyzed in detail. A comprehensive analysis on the performance compares the three NoSQL databases on their ability to scale under different workload conditions and with different dataset sizes. Finally, a benchmark report, which provides a recommendation on the suitable database for a specific type of application requirements, was presented.

A more thorough analysis on the behavior of each NoSQL system based on the benchmark testing results can be found in [11].

Acknowledgment

The author would like to thank Mr. Upa Gupta and Mr. Mohammadhani Fouladgar [1] for their support.

References

[1] S. Edlich. (2015) NoSQL. [Online]. Available: http://nosql-database.org

[2] Veronika Abramova; Jorge Bernardino and Pedro Furtado, "EXPERIMENTAL EVALUATION OF NOSQL DATABASES," *International Journal of Database Management Systems (IJDMS) Vol.6, No.3, June 2014*, vol. 6, no. 100, pp. 1–16, 2005.

[3] V. Abramova, J. Bernardino, and P. Furtado, "Which NoSQL Database? A Performance Overview," *Open Journal of Databases (OJDB)*, vol. 1, no. 2, pp. 17–24, 2014.

[4] V. Abramova and J. Bernardino, "Nosql databases: Mongodb vs cassandra," in *Proceedings of the International C* Conference on Computer Science and Software Engineering*, ser. C3S2E '13. New York, NY, USA: ACM, 2013, pp. 14–22. [Online]. Available: http://doi.acm.org/10.1145/2494444.2494447

[5] Veronika Abramova; Jorge Bernardino and Pedro Furtado, "Testing cloud benchmark scalability with cassandra," in *Services (SERVICES), 2014 IEEE World Congress on*, June 2014, pp. 434–441.

[6] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," *Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10*, pp. 143–154, 2010.

[7] MongoDB. (2009) MongoDB. [Online]. Available: https://www.mongodb.com/

[8] Apache, "HBase," 2013. [Online]. Available: http://hbase.apache.org/

[9] Cassandra, "The Apache Cassandra Project," 2010. [Online]. Available: http://cassandra.apache.org/

[10] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*, 2006, pp. 205–218.

[11] S. N. Swaminathan, "Quantitative analysis of scalable nosql databases," p. 100, 2015, copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2016-03-04.

[1]Both Phd candidates at University of Texas at Arlington.