ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

Comparative Analysis of Various Techniques Used In Managing Big Data

Rakesh Kumar[1], Sanketh S Salian[2], Suraj Nayak[3]

Assistant Professor, AIMIT, St Aloysius College, Mangalore, India[1]

M.Sc (ST) III Semester, AIMIT, St Aloysius College, Mangalore, India[2]

MSc (ST) III Semester, AIMIT, St Aloysius College, Mangalore, India[3]

ABSTRACT
The Digital world is growing very fast and become more complex in terms of volume (terabyte to petabyte), variety (structured and un-structured and hybrid), velocity (high speed in growth) in nature which to the data collection that has grown so large it can't be effectively managed or manipulated using conventional data management tools: e.g., classic relational database management systems (RDBMS).in recent time many applications work based upon Big Data Analytics, Business Intelligence and social networking which has Petabyte datasets and have pushed SQL-like centralized databases to their limits. To handle these problems, traditional RDBMS are provided with an alternative. This led to the development of horizontally scalable, distributed non-relational data stores, called No-SQL databases [7]. The paper shall conclude with suitable NOSQL technology to be used based on the requirement of the organization.

I. INTRODUCTION

The volume of data that is being generated through various social media and other e-commerce transactions has become unmanageable. To manage Big Data in terms of volume, velocity and variety is the biggest challenge of the present day. SQL is a database computer language designed for the retrieval and management of data in relational databases. However, SQL is not capable to handle big data as it follows strict schema and constraints. We have NoSQL databases that is capable of managing big data. It is another type of data storage other than databases (that were used earlier) that is used to store huge amount of data which keeps on increasing day by day. NoSQL is a non-relational database management system and fast information retrieval database and is portable.

II. RELATED WORK

Features of NoSQL [1, 2]:

- ➢ Flexible schema
- ➢ Quicker/cheaper to set up
- ➢ Massive scalability
- ➢ Higher performance and availability
- ➢ No declarative query language (i.e. SQL) $\rightarrow$ more programming
- ➢ Relaxed consistency $\rightarrow$ fewer guarantees

Characteristics of NoSQL [1,2]:

- ➢ Large data volumes
- - Google's "big data"
- ➢ Scalable replication and distribution
- - Potentially thousands of machines
- - Potentially distributed around the world
- ➢ Queries need to return answers quickly
- ➢ Mostly query, few updates

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

- Asynchronous Inserts & Updates
- Schema-less
- ACID transaction properties are not needed – BASE
- CAP Theorem
- Open source development

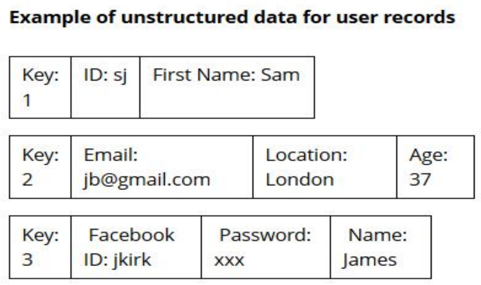
BASE [3]:

- **Basically Available**: Data is always available.
- **Soft State**: Data provided might not be the latest(Not all shards return the same data value)
- **Eventually Consistent**: The system will replicate the data across all the nodes as time passes especially when there is a update process
  - Because of the distributed model, any server can answer any query
  - Servers communicate amongst themselves at their own pace (behind the scenes)
  - The data provided to the user might not be the latest data

Types of NoSQL [1,2]:

- **Key-Value Pair (KVP) Stores[1,2]:**

The data are accessed by strings called keys. There is no specific format for the data. The data may have any format in this type. This provides extremely simple interface. The Data model followed in this type is by (key, value) pair. Record distribution is done based on the keys.
The basic operations are:
Insert(key,value),Fetch(key), Update(key), Delete(key)
The "Value" is stored as a "BLOB" without caring or knowing what it actually contains. It's the responsibility of the application to understand the data. In simple terms, a NoSQL Key-Value store is a single table which contains two columns: one being the (Primary) Key, and the other being the Value which is BLOB object.



Example of unstructured data for user records

Key: 1	ID: sj	First Name: Sam

Key: 2	Email: jb@gmail.com	Location: London	Age: 37

Key: 3	Facebook ID: jkirk	Password: xxx	Name: James

- **Document Stores:**

This is similar to Key-Value Stores, however the value is stored in the form of "Documents". The data model used here is (key, "document") pairs. Format of the document can be JSON, XML, or other semi-structured formats. The **Document Stores** store arbitrary/extensible structures as a "value".
The basic operations are:
Insert(key,document),
Fetch(key),
Update(key) and Delete(key)
And *Fetch()* operation based on document contents.
Documents stored system areCouchDB, MongoDB, SimpleDBetc. Single table can have documents of different format.
An example record from Mongo, using JSON format, might look like:

```
{
"_id" :ObjectId("4fccbf281168a6aa3c215443″),
"first_name" : "Naveen",
"last_name" : "Kumar",
"address" : {
        "street" : "Car street Main Road",
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

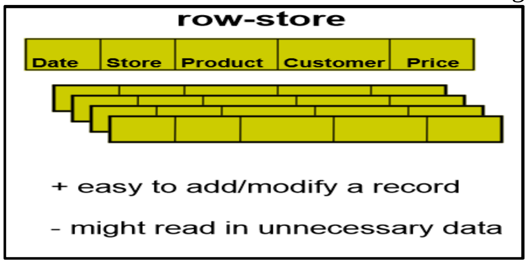
Vol.2, Special Issue 5, October 2014

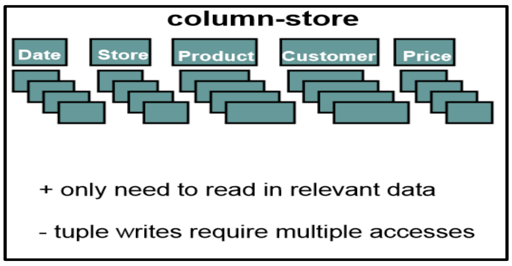
```
        "city" : "Mangalore",
        "state" : "Karnataka"
}
}
```

Though records are called documents, however they are not word processing document, although you can store binary data (using BSON format) in any of the fields in the document. We have the flexibility to modify the structure of the document by adding or removing members from the document either by reading the document into your program, modifying it or re-saving it, or by using various update commands.

- **Column-based Stores**

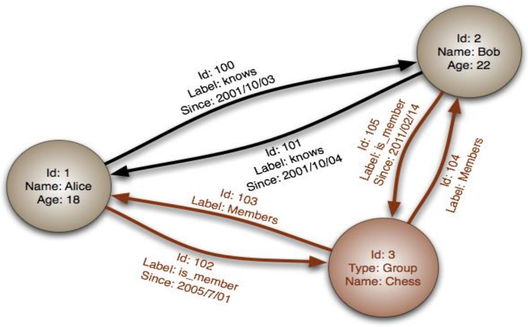
This concept is based on Google's Big Table store. What is a column-based store? Here data is stored in the form of columns rather than in the form of rows. The below figures explains the column-based stores.


row-store
Date
Store
Product
Customer
Price
+ easy to add/modify a record
- might read in unnecessary data


column-store
Date
Store
Product
Customer
Price
+ only need to read in relevant data
- tuple writes require multiple accesses

- **Graph Database Systems**

The concept in the Graph Database Systems in the form of nodes and edges. The nodes may have properties and edges may have labels or roles. To store the information i.e. relationship between the entries, graph theory is used. A graph database is a database that uses graph structures with nodes, edges, and properties to represent and store data [5]. By definition, a graph database is any storage system that provides index-free adjacency [6]. Hence it contains a direct pointer to its adjacent nodes which results in having no index lookups.



ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

Comparison of different NoSQL Databases

Attributes		NoSQL Databases				
Database Model		Document Stored		Column Based		Graph Oriented
Design & Features	**Features**	**MongoDB**	**CouchDB**	**DynamoDB**	**Cassandra**	**Neo4j**
	Data Storage	Volatile Memory File system	Volatile Memory File system	SSD		File System Volatile Memory
	Map Reduce	Yes	Yes	Yes	Yes	No
Integrity	Atomicity	Conditional	Yes	Yes	Yes	Yes
	Consistency	Yes	Yes	Yes	Yes	Yes
	Isolation	Yes	Yes	Yes	No	Yes
	Durability	Yes	Yes	Yes	Yes	Yes
	Transactions	No	No	No	No	Yes
Indexing	Secondary Indexes	Yes	Yes	No	Yes	--
	Graph Support	No	No	No	No	Yes
Distribution	Horizontal Scalable	Yes	Yes	Yes	Yes	No
	Replication	Yes	Yes	Yes	Yes	Yes
	Replication Mode	Master-Slave Replica Replication	Master-Slave Replica Replication	--	Master-Slave Replica Replication	--
System	Operating System	Cross Platform	Ubuntu Red Hat Windows Mac OS X	Cross Platform	Cross Platform	Cross Platform
	Programming Language	C++	Erlang C++ C Python	Java	Java	Java

Database Selection Analysis

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

Cassandra is a column based storage operating under the Apache Software Foundation. Facebook was the company which initially developed it and it was created with the aim to provide availability and also the ability to scale to a very large size as and when the requested. With flexible consistency models, the architecture was particularly helpful to perform write operations.

Couchbase is the company formed by the merger of two databases, CouchDB and Membase. The version 1.8 is purely a distributed key-value store built around the hugely popular memcached cache, and the version 2.0 added extra feature such as secondary index support as well as performance improvements.
MongoDB is different from the other products as it is primarily based on a document database. It provides extensive support for different kinds of secondary indices and a very different approach to scaling and durability.

Aerospike is an open-source, In-memory, NoSQL Database. Aerospike Database is written in C, and it contains three layers:

A) Flash optimized data layer
Data layer is optimized to store data in (SSD) solid state drives or RAM. The database indices are stored in RAM for quick availability.
B) Self-managed distribution layer
This layer is placed across all data centers which helps to ensure consistency. Hence allowing to database to operate correctly even when a server node fails
C) Cluster-aware client layer
It is used to track the configuration of the cluster in the database, and also manages communications in the server node.

Client & Workload Description Data Sets
Loading of data to the database was done using the “load” phase of the Yahoo Cloud Serving Benchmark (YCSB) tool. Replication factor of 2 was used for each database, so each record was stored two times.
Record size: 120 bytes Key size: 23 bytes

Disk-backed Data Set
In this case, Dataset provided is much more than RAM space because the idea is to force a to keep only the data in the dish while using only required data in order to reduce the bottleneck in bandwidth.

In-memory Data Set
In this, most of the data are kept in RAM, which is said to be the preferred setup for MongoDB

Workloads
In this analysis two workloads are being used .A balanced workload of reads and writes and a read-heavy workload.

Workload A - Balanced
Read operations: 50%
Update operations: 50%
Workload B - Read Heavy
Read operations: 95%
Update operations: 5%

III. RESULTS

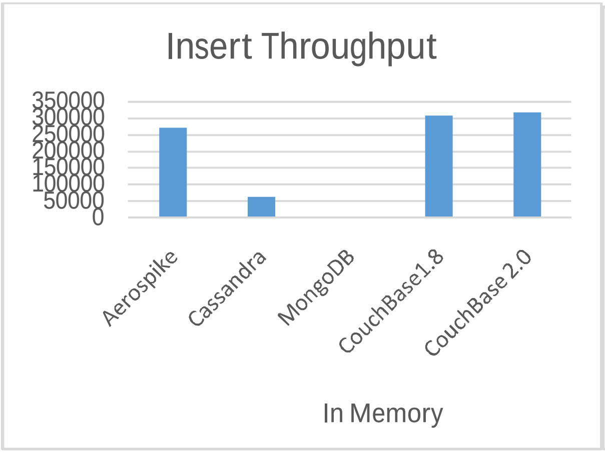
Test 1: Loading Data
The first test was to load the data to each of the database and this was done by using YCSB’s load routine for each dataset. Primary goal of this was to prep the database for the read and update tests.

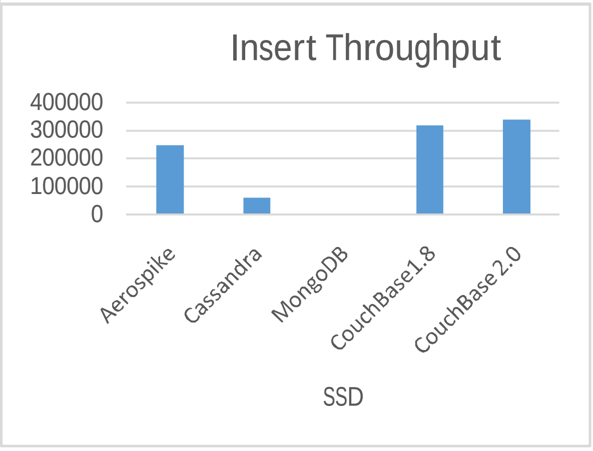
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014



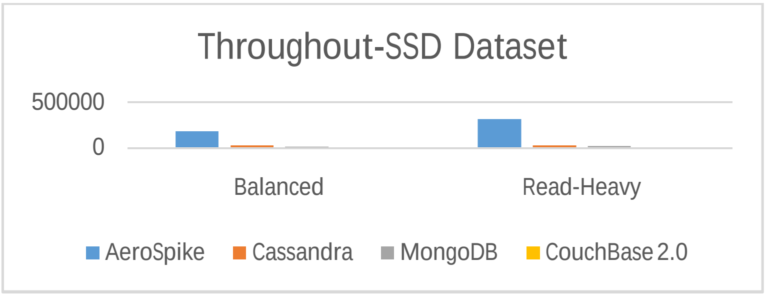


Aerospike	Cassandra	MongoDB	CouchBase1.8	CouchBase 2.0
272000	63000	3000	310000	320000
249000	62000	4000	320000	340000

The asynchronous In-memory numbers show that Aerospike as well as Couchbase performed extremely well i.e. all over 250,000 inserts per second however Couchbase had 10-15% advantage. In SSD, the load was good enough but still not easily comparable to Aerospike because of configuration differences

Test 2: Durability
The throughput test were started with a strong durability model using a dataset which is replicated so that it is much larger than the server's RAM. This test was done to test the use of model for transactional data which requires string durability guarantees.The test was run using YCSB to perform each of the workloads quickly. Here a certain number of client machines are used to maximize usage of database, Cassandra and MongoDB showed only minor performance gains after 4 clients.

However before measuring was done,10 minute warm-up period was done to bring the database into a state that is said to be "not rest" and also ensure any caches were properly used.



Latency results
Test 3: Latency
We then measured latency for various traffic levels for each database. We tracked read and update latencies separately for each workload.

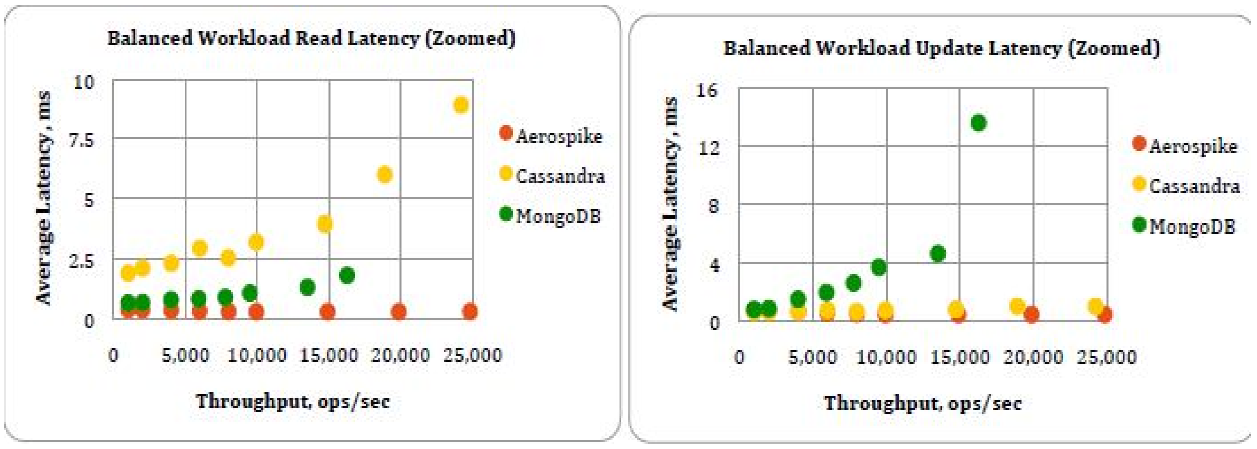
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

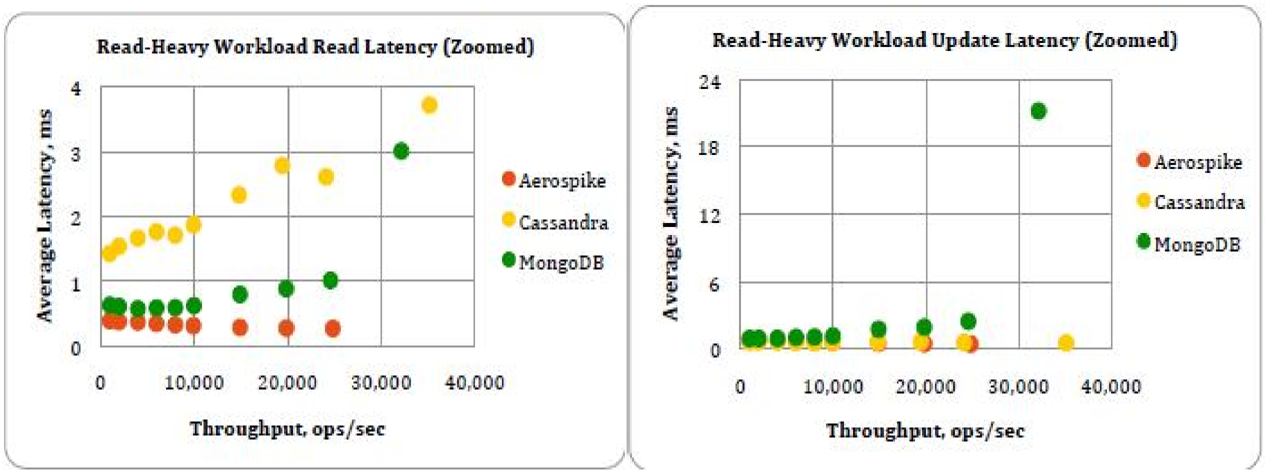
(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

Figures 3a, b: Latency/Throughput Results - Balanced Workload



Figures 4a, b: Latency/Throughput Results - Read-Heavy Workload



In all the cases which contains highest load Aerospike maintained sub-millisecond latencies.cassandra had consistent write latency whereas read latency increased linearly

Test 4: Fast scenario

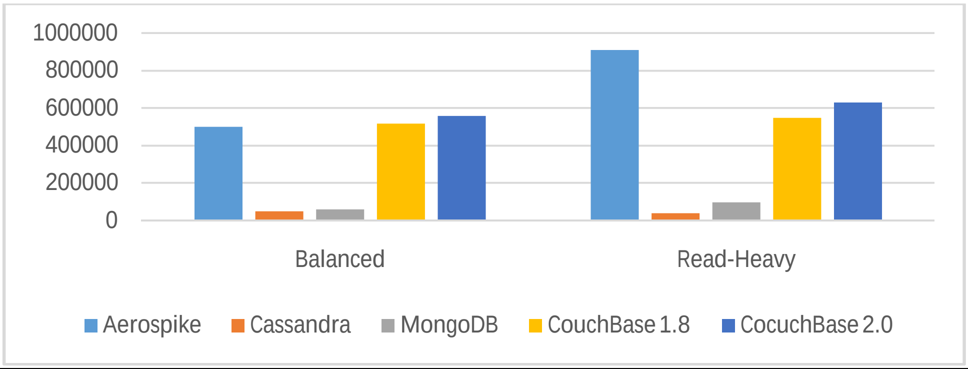
After getting the durable numbers, we cleared the databases and ran the same tests again, this time with a dataset that was able to fit into RAM and with asynchronous replication. The goal of this test was to show the maximal performance that these databases could achieve, when liberated from the requirement of having durable data. All the databases in this test were configured to store the entire working set in memory and persist to disk and replicas as soon as it became available.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

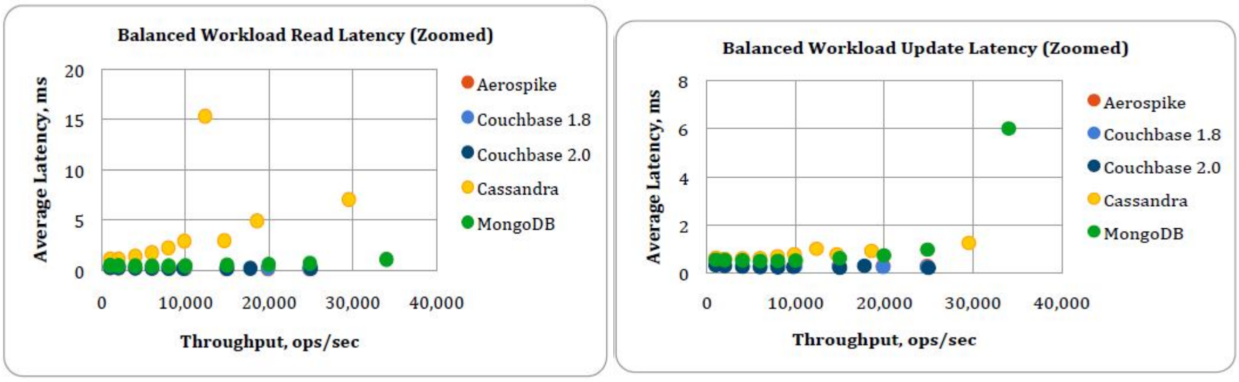
Vol.2, Special Issue 5, October 2014



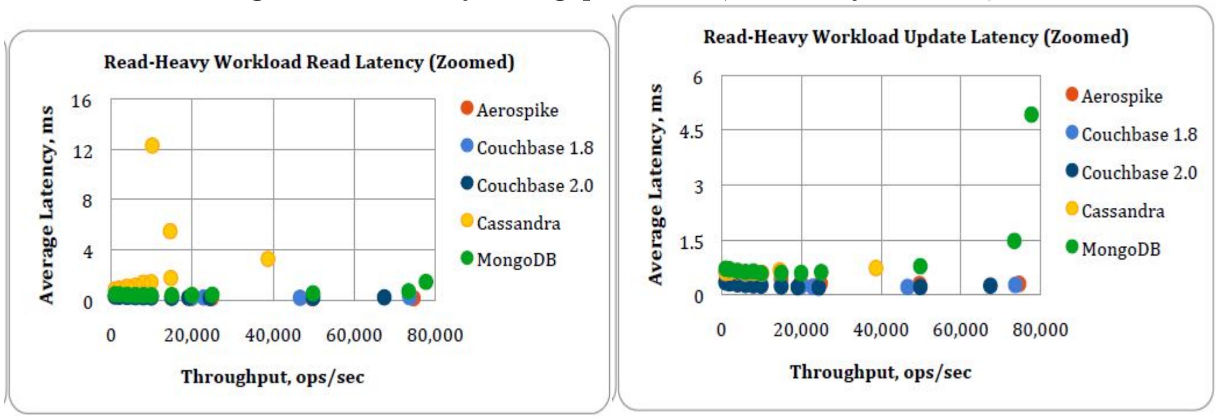
Aerospike	Cassandra	MongoDB	CouchBase 1.8	CocuchBase 2.0
500000	50000	60000	520000	560000
910000	40000	100000	550000	630000

In this scenario, Couchbase was the fastest performer for the balanced workload by about 10%, whereas Aerospike outperformed Couchbase by about 40% on the read-heavy workload. Both of these systems demonstrated truly impressive performance, on the order of half a million of records. MongoDB and Cassandra were an order of magnitude slower -- both were configured with enough RAM cache to contain the full working set.

Figures 6a, b: Latency/Throughput Results (Balanced Workload)



Figures 7a,b: Latency Throughput Results (Read Heavy Workload)



Both Couchbase and Aerospike maintained sub-millisecond latencies up to their highest performance. For MongoDB, latency degraded on writes but stayed consistent on reads

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol.2, Special Issue 5, October 2014

IV. CONCLUSIONS

The best results was the throughput number that Aerospike was able to achieve even while committing the data across multiple data nodes however having a speed of 200,000 op/sec with these strong guarantees has put it far ahead of its nearest competitor and at speeds we would not normally associate with ACID semantics[6].

When the entire data set fit into RAM, durability guarantees are weakened, the results showed both Couchbase and Aerospike in a near to neck in terms of performance. Couchbase slightly outperformed Aerospike for the balanced read-write workload, and Aerospike somewhat more significantly outperformed Couchbase for the read-heavy workload.

One thing to consider that how an organization might scale out their data storage even in the asynchronous model.SSD based test are important since SSDs have higher densities and lower per-gigabyte costs than RAM. As such, it should be possible to scale our much larger data sets on fewer nodes, which is clearly valuable when talking about very large amounts of data.

When scaling is done based on RAM-backed storage,we should consider recovery aspect and also more nodes mean higher rate of node failures so recovery becomes a very important phase. If recovery can be managed effectively then RAM-based approach might be one way to scale. However, if writes to disks and replicas are done asynchronously, recoverability can be problematic.

REFERENCES

[1] Vatika Sharma, Meenu Dave, "SQL and NOSQL Databases", International Journal Of Advanced.
[2] "SQL TUTORIAL", https://tutorialspoint.com
[3] "Ultra-High Performance NoSQL Benchmarking", Analyzing Durability and Performance Tradeoffs By Denis Nelubin, Director of Technology and Ben Engber, CEO, Thumbtack Technology
[4] Database. (n.d.). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Database
[5] Graph database. (n.d.). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Graph_database
[6] Aerospike Benchmark. (n.d.). Retrieved from Aerospike:http://www.aerospike.com/blog/thumbtack-ycsb-benchmark/
[7] A B M Moniruzzaman, S. A. (n.d.). NoSQL Database: New Era of Databases for Big Data Analytics - Classification, Characteristics and Comparison.