White Paper


Big Data

# MongoDB vs Hadoop

## *Big Solutions for Big Problems*

*Written by: Deep Mistry, Open Software Integrators*

In recent years there has been an explosion of data. IDC predicts that the digital universe will grow to 2.7 zettabytes in 2012, up 48% from 2011. By 2015, this is expected to grow to 8 zettabytes of data. 1 Zettabyte = 1,000,000,000,000,000,000,000 bytes. More than half of this is the unstructured data from social networks, mobile devices, web applications and other similar sources. Traditional RDBMS systems are data storage solutions that were designed decades ago. At that time, most of the data was structured and had a different flavor than today. We need solutions that address Big Data problems and were designed on the basis of recent changes in the nature of data.[1] [2]

There are many solutions that address the Big Data problem. Some of these solutions are tweaks and/or hacks around standard RDBMS solutions. Some are independent data processing frameworks, data storage solutions or both. There is a website which lists many of them.[3]

According to Jaspersoft, there are two leading technologies in this new field. First is Apache Hadoop, a framework that allows for the distributed processing of large data sets using a simple programming model. The other is MongoDB, an open source NoSQL scalable, distributed database. Both of these solutions have their strengths, weaknesses and unique characteristics.[4] [5] [6]

MongoDB and Hadoop are fundamentally different systems. MongoDB is a database while Hadoop is a data processing and analysis framework. MongoDB focuses on storage and efficient retrieval of data while Hadoop focuses on data processing using MapReduce. In spite of this basic difference, both technologies have similar functionality. MongoDB has its own MapReduce framework and Hadoop has HBase. HBase is a scalable database similar to MongoDB.[7] [8]

The main flaw in Hadoop is that it has a single point of failure, namely the "NameNode". If the NameNode goes down, the entire system becomes unavailable. There are a few workarounds for this which involve manually restoring the NameNode.[9]

Regardless, the single point of failure exists. MongoDB has no such single point of failure. If at any point in time, one of the primaries, config-servers, or nodes goes down, there is a replicated resource which can take over the responsibility of the system automatically.[10] [11]

MongoDB supports rich queries like traditional RDBMS systems and is written in a standard JavaScript shell. Hadoop has two different components for writing MapReduce (MR) code, Pig and Hive. Pig is a scripting language (similar to python, perl) that generates MR code, while Hive is a more SQL-like language. Hive is mainly used to structure the data and provides a rich set of queries. Data has to be in JSON or CSV format to be imported into MongoDB. Hadoop, on the other hand can accept data in almost any format. Hadoop structures data using Hive, but can handle unstructured data easily using Pig. With the help of Apache Sqoop, Pig can even translate between RDBMS and Hadoop.[12] [13] [14]

## What about transactions?

Database transactions: A transaction is a logical unit of work that usually consists of one or more database operations.[15]

MongoDB has no transactions. Initially that fact may be a concern for people from a RDBMS background. Transactions are somewhat obsolete in the age of the web. When dealing with distributed systems, long-running database operations and concurrent data contention the concept of a "database transaction" may require a different strategy. Jim Gray - The Transactional Concept : Virtues and Limitations.[16]
MongoDB does have something that is "transaction-like" in that a database write can happen in a single blocking synchronous "fsync". MongoDB supports atomic operations to some extent. So long as the schema is structured correctly, you can have a reliable write for a single entry.

---

1. http://www.smartercomputingblog.com/2012/03/21/finally-more-data-about-big-data/ 2. http://en.wikipedia.org/wiki/Big_data
3. http://nosql-database.org/ 4. http://nosql.mypopescu.com/post/20001178842/nosql-databases-adoption-in-numbers
5. http://hadoop.apache.org/ 6. http://www.mongodb.org/ 7. http://hadoop.apache.org/common/docs/current/mapred_tutorial.html#Overview
8. http://hbase.apache.org/ 9. http://wiki.apache.org/hadoop/NameNode 10. http://www.mongodb.org/display/DOCS/Replica+Sets
11. http://www.mongodb.org/display/DOCS/Configuring+Sharding#ConfiguringSharding-ConfigServers
12. http://pig.apache.org/ 13.http://hive.apache.org/ 14. http://sqoop.apache.org/ 15. http://en.wikipedia.org/wiki/Database_transaction
16. http://en.wikipedia.org/wiki/Database_transaction

MongoDB (written in C++) manages memory more cost-efficiently than Hadoop's HBase (written in Java). While Java garbage collection (GC) will, in theory, be as CPU/time efficient as unmanaged memory, it requires 5-10x as much memory to do so. In practice, there is a large performance cost for GC on these types of large scale distributed systems. Both systems also take a different approach to space utilization. MongoDB pre-allocates space for storage, improving performance, but wasting space. Hadoop optimizes space usage, but ends up with lower write performance by comparison with MongoDB.

Hadoop is not a single product, but rather a software family. Its common components consist of the following:

- Pig, a scripting language used to quickly write MapReduce code to handle unstructured sources
- Hive, used to facilitate structure for the data
- HCatalog, used to provide inter-operatability between these internal systems [15]
- HBase, which is essentially a database built on top of Hadoop
- HDFS, the actual file system for hadoop. [16]

MongoDB is a standalone product with supported binaries. The learning curve for MongoDB is generally lower than that of Hadoop.

Recently, there has been a lot of talk about security with NoSQL databases. Both MongoDB and Hadoop have basic security. MongoDB has simple authentication and MD5 hash and Hadoop offers fairly rudimentary security in its various frameworks. This is not a flaw. NoSQL databases like MongoDB were never designed to handle security, they were designed to efficiently handle Big Data which they effectively do. It is simple to implement security in your application instead of expecting it from your data solution. Here is more on this.[17]

For data processing and data analysis there is almost no technology, Open Source or otherwise, that beats Hadoop. Hadoop was designed specifically to address this issue and thereby it contains all the components necessary to rapidly process terabytes to petabytes of information. Writing MapReduce code in Hadoop is elementary. Pig is easy to learn and makes it uncomplicated to write user-defined functions. MongoDB has its own MapReduce framework, which, though subpar to Hadoop, does the job well. When it boils down to sheer numbers, Hadoop is ahead. Yahoo has a 4000 node Hadoop cluster

*Continued on page 3...*

## What is MapReduce? Why do we want it?

MapReduce is a framework for processing highly distributable problems across huge datasets.

It divides the basic problem into a set of smaller manageable tasks and assigns them to a large number of computers (nodes). An ideal MapReduce task is too large for any one node to process, but can be accomplished by multiple nodes efficiently.

MapReduce is named for the two steps at the heart of the framework.

- **Map step**: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. Each worker node processes its smaller problem, and passes the result back to its master node. There can be multiple levels of workers.
- **Reduce step**: The master node collects the results from all of the sub-problems, combines the results into groups based on the key and then assigns them to worker nodes called reducers. Each reducer processes those values and sends the result back to the master node.

MapReduce can be a huge help in analyzing and processing large chunks of data:

- buying pattern analysis, customer usage and interest patterns in e-commerce
- processing the large amount of data generated in the fields of science and medicine
- processing and analyzing security data, credit scores and other large data-sets in the financial industry

These and many other uses make MapReduce, an indispensable tool in the software industry.

15. http://incubator.apache.org/hcatalog/
16. http://hadoop.apache.org/hdfs/
17. http://www.darkreading.com/blog/232600288/a-response-to-nosql-security-concerns.html
18. http://developer.yahoo.com/blogs/hadoop/posts/2008/09/scaling_hadoop_to_4000_nodes_a/

and someone is aiming to test a 10000 nodes soon enough. MongoDB is typically used in clusters with around 30-50 shards and a 100 shard cluster under testing. Typically, MongoDB is used with systems less than approximately 5 TB of data. Hadoop, on the other hand, has been used for systems larger than 100 TB, including systems containing petabytes of data.[18]

There are several use cases of both systems.

**Hadoop**

- **Log Processing (best use case)** - Log files are usually very large and there are typically lots of them. This creates huge amounts of data. A single machine may not be able to efficiently process them. Hadoop is the best answer to this problem; splitting the log into smaller workable chunks and assigning them to workers results in very fast processing.
- **ETL** -- For unstructured data streaming in real time, say from a web-application, Hadoop is a good choice to structure the data and then store it. Additionally, Hadoop provides ways to pre-process data prior to structuring it.
- **Analytics** - Various organizations are using Hadoop's superior processing capabilities to analyze huge amounts of data. The most famous usage is Facebook's 21PB, 2000 node cluster, which Facebook uses for several tasks.
- **Genomics** - Scientists working in this field constantly need to process DNA sequences. These are very long and complicated strands of information which require large amounts of storage and processing. Hadoop provides a simple cheap solution to their processing problem.
- **Machine education** - Apache Mahout is built on top of Hadoop and essentially works along with it to facilitate targeted trade in e-commerce.

**MongoDB**

- **Archiving** - Craigslist uses MongoDB as an efficient storage for archives. They still use SQL for active data while they archive most of their data using MongoDB.
- **Flexibility in data handling -** With data in a non-standard format (like photos) it is easy to leverage the unstructured document oriented storage of MongoDB, in addition to its scaling capacities.
- **E-Commerce** - OpenSky had a very complicated e-commerce model. It was very difficult to design the correct schema for it and even then tables had to be altered many times. MongoDB allows them to alter the schema as and when required and simultaneously scale.
- **Online website data-storage** - MongoDB is very good at real-time inserts, updates, and queries. Scalability and replication are required for very large websites.
- **Mobile systems** - MongoDB is a great choice for mobile systems due to its geo-spatial index.

## Why NoSQL?

Relational databases have been used for decades. NoSQL databases are radically different and less mature. Relational databases can scale and handle Big Data. The difference is that relational databases require more effort to scale.

Facebook uses MySQL-based database storage. Facebook with 800 million users and their ever growing numbers, is the best example of how relational databases can scale.[17]

The question arises, why do we need one of these new NoSQL databases? Is it to scale? Clearly not. Relational databases have been a part of large-scale systems for years. The issue is the amount of work to make them scale. For example, Twitter began as a relational database accessed by a Rails application. Amazon began as a relational database accessed by a C++ application. Facebook also began as a relational database accessed with PHP. While these giants have been successful at scaling, their success has not come easily. They had to make many tweaks and changes and even implement a few software and hardware tricks to achieve it.

Consider the simple act of "sharding" and what would be required to shard a set of customer records on a traditional RDBMS. What about the related data? How do you handle an unusual number of orders on the node for customers with names beginning with the letter S? You can shard an RDBMS, but a lot of man hours will go into what a NoSQL database can handle automatically. *(Continues...)*

Sometimes it is helpful to use MongoDB and Hadoop together in the same system. For instance, with a system that is reasonably large that requires rich queries with indexes and effective retrieval, we can leverage the best qualities of both systems.

17. http://gigaom.com/cloud/facebook-shares-some-secrets-on-making-mysql-scale/

**Hadoop + MongoDB**

- **The "Catching Osama" problem:**
  Problem
  - A few billion pages, with geospatial-tags
  - Tremendous storage and processing required
  - MongoDB fails to effectively process data due to a single-thread bottleneck per node.
  - Hadoop does not have indexes thereby data retrieval takes longer than usual

  Solution
  - Store all images in MongoDB with geo-spatial indexes
  - Retrieve using the Hadoop framework and assign pro cessing to multiple mappers and reducers

- **Demographic data analysis:**
  Whenever there is a need for processing demographic data spread over a large geographical area, the geospatial in-dexes of MongoDB are unmatched and MongoDB becomes an ideal storage. Hadoop is essential when processing hun dreds of terabytes of information.

- Given an unstructured data source and the desire to store it in MongoDB, Hadoop can be used to first struc ture the data. This facilitates easier storage in MongoDB. Then Hadoop can be used repeatedly for processing it later in large chunks.

  Both MongoDB and Hadoop have their place in modern systems. MongoDB is rapidly becoming a replacement for highly scalable operational systems and websites. Hadoop offers exceptional features for applying MapReduce to Big Data and large analytical systems. Currently, both are dis-placing traditional RDBMS systems.

*This white paper was written by Deep Mistry, Open Software Inte-grators. Open Software Integrators, LLC is a professional services company that provides consulting, training, support and course development.*

## Why NoSQL? *Continued*

What of ETL? Processing terabytes of data is possible using parallel queries, star schemas, disk layouts and other techniques. Using these traditional means, you can eek out as much CPU as possible and then push the data into separate databases for additional pro-cessing. However, it will still be hard to achieve, through this manual structur-ing, what you can achieve with Hadoop or a similar solution. It can be done, and has been many times before, but it will not be as easy or as quick as one of these modern automated solutions.

What about reliability? Oracle RAC is tried and true. However, it is expensive and difficult to work with on larger systems. It requires abandoning many of the features of the Oracle RDBMS. Microsoft SQL Server also supports clustering with a similar approach and similar limitations as Oracle. MongoDB and some (but not all) of the other NoSQL solutions support out of the box reliability.

For a startup or a growing company, scalable databases, which require investing a lot of effort and time into making the storage solution scale, is not an option. This is where NoSQL data-bases are the clear winner. For a larger company, with IT budget cuts and increasing pressure to make the best use of staff and out of the box solutions, NoSQL and Big Data solutions can no longer be ignored.

## Contact

www.osintegrators.com
345 W. Main St. Suite 201
Durham, NC 27701
(919) 321-0119
info@osintegrators.com