Paper submitted for STC 2013

# Performance Testing of Big Data Applications

**Author:**

**Mustafa Batterywala: Performance Architect | Impetus Technologies**
mbatterywala@impetus.co.in

**Shirish Bhale: Director of Engineering | Impetus Technologies**
sbhale@impetus.co.in

Impetus Technologies | 5300 Stevens Creek Boulevard,
Suite 450, San Jose,
CA. 95129, USA

# Table of Contents




Impetus Technologies | 5300 Stevens Creek Boulevard,
Suite 450, San Jose,
CA. 95129, USA

# Abstract

One of the latest buzz words in the IT industry is Big Data. Although still in its infancy with enterprises attempting to gain maturity in big data engineering, the potential it promises is huge with a forecast of $1–1.5 billion in industry revenue by 2015.

With the coming of Big data technologies like Hadoop, NoSQL, Messaging Queues etc. organization have got the tools to dive deep into the large amounts of data and come up with analytics and intelligence that can help them drive business growth and take right decisions in time. But, testing Big data is one of the biggest challenges that the organizations face because of the lack of knowledge on what to test and how to test. They have been facing challenges in defining appropriate test strategies, tools, working with NoSQL, setting up optimal test environments and so on.

In this paper we will talk about that how we can performance test these systems by addressing the above challenges. We present the important areas in a big data application that are primary candidates for performance testing like data ingestion capacity and throughput, data processing capacity, NO SQL systems, simulation of expected usage, map reduce jobs and so on. We look at available tools that are capable of solving the above challenges and can be used for such performance testing needs. We will also share details on how different technologies like Cassandra, MongoDB, Hadoop etc. be monitored and stress tested using specific tools set.

Impetus Technologies | 5300 Stevens Creek Boulevard,
Suite 450, San Jose,
CA. 95129, USA

# Introduction to Big Data

Big data is a buzzword, or catch-phrase, used to describe a massive volume of both structured and unstructured data that is so large that it's difficult to process using traditional database and software techniques.

An example of big data might be petabytes (1,024 terabytes) or Exabyte's(1,024 petabytes) of data consisting of billions to trillions of records of millions of people -- all from different sources (e.g. Web, sales, customer contact center, social media, mobile data and so on). The data is typically loosely structured data that is often incomplete and inaccessible.

- Every day, we create 2.5 quintillion bytes of data — so much that 90% of the data in the world today has been created in the last two years alone.
- Gartner defines Big Data as high volume, velocity and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.
- According to IBM, 80% of data captured today is unstructured, from sensors used to gather climate information, posts to social media sites, digital pictures and videos, purchase transaction records, and cell phone GPS signals, to name a few. All of this unstructured data is Big Data.

Big data analytics is the process of examining large amounts of data of a variety of types (big data) to uncover hidden patterns, unknown correlations and other useful information. Such information can provide competitive advantages over rival organizations and result in business benefits, such as more effective marketing and increased revenue.

Big data analytics can be done with the software tools commonly used as part of advanced analytics disciplines such as predictive analytics and data mining. But the unstructured data sources used for big data analytics may not fit in traditional data warehouses. Furthermore, traditional data warehouses may not be able to handle the processing demands posed by big data. As a result, a new class of big data technology has emerged and is being used in many big data analytics environments. The technologies associated with big data analytics include NoSQL databases, Hadoop and Map Reduce. These technologies form the core of an open source software framework that supports the processing of large data sets across clustered systems.

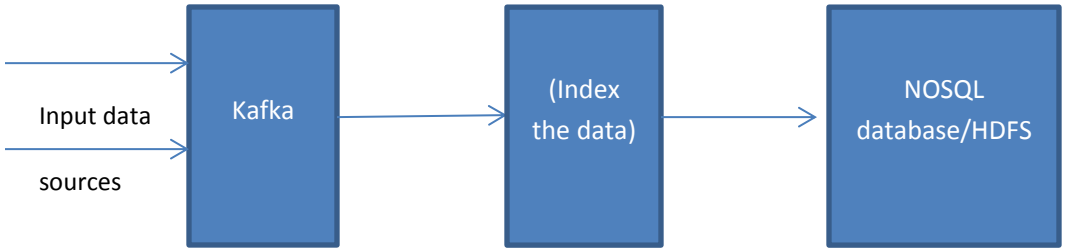
Typically, a big data application makes use of the following set of technologies:

1) Map Reduce frameworks: Map Reduce is a software framework that allows developers to write programs that process massive amounts of unstructured data in parallel across a distributed cluster of processors or stand-alone computers. E.g Apache Hadoop, Cloudera etc.
2) NoSQL databases: SQL has been the main stay of database professionals for so many years. Without much thought to any alternatives, we continued using table-oriented relational database for all of

our data storage and retrieval needs. Those times, however, are changing. No SQL database models questions the relational model by abandoning many deeply held beliefs about the "proper" structure for a database. In exchange for removing some of the design constraints, NoSQL databases can achieve enhanced performance and flexibility.

3) Message Queue: A publish-subscribe messaging system to input data into a system from multiple sources. E.g. Kafka, RabbitMQ
4) Search components like Elastic search and etc.

# Performance Test Focus Areas




The above diagram is a very high level representation of a big data analytics application. As a first step, multiple input streams are used to input data in the system via Kafka queues. The input data goes through the queue and it is moved to either a NoSQL data store or HDFS. Depending on the data store we can write NoSQL queries or map reduce programs to extract the data and create reports for enabling business decisions.

Unlike the traditional web applications that are performance tested from the end user perspective, these systems present an altogether different performance testing areas that we need to focus:

1) Data ingestion and throughout: One of the critical performance areas is that how fast system can consume data from different data sources. Primarily this is done through message queues and hence we need to make sure that the queues perform optimally and have a maximum throughput. This type of testing involves identifying the maximum message that the queue can process in a given amount of time with optimum resource utilization. We also need to identify that how quickly data can be inserted into the underlying data store. For e.g. what is the insertion rate into a Mongo and Cassandra database?
2) Data processing: Data processing refers to the speed with which the queries and/or map reduce jobs are executed to generate the resultant data set of which the analytics can be run for generating business reports and analysis. It involves testing the data processing in isolation when the underlying data store is populated with the data sets. This is achieved by running the focus performance tests depending on the technology. For e.g. it could be running the PIG/Hive scripts or queries against the NoSQL or running Map reduce jobs on the underlying HDFS.

3) Sub component performance: From the figure above, it is clear that these systems are made of multiple sub components and each component would have multiple instances. This makes the entire deployment very large and introduces the possibility of performance bottlenecks at multiple places. We need to test each and every component in isolation for e.g. how quickly message are consumed and indexed, query performance, map reduce jobs, search etc.

# Performance Testing Challenges

Performance testing Big Data is one of the challenges faced by the organizations because of lack of knowledge on what to test, how to test and how much data to test. Organizations have been facing challenges in defining the strategies for validating the performance of individual sub components, creating an appropriate test environment, working with NoSQL and other systems. These challenges are responsible for poor quality in production, delayed implementation and increase in cost. Let's look at some of these challenges in a bit more details:

1) Diverse set of technologies: As seen above, each sub component involved in a big data application belongs to a different technology. While we need to test each in isolation, no single tool can support each of the technologies. This is unlike the web applications where though the technology may differ but underlying communication is through Http. But, in this case the communication mechanism vary from component to components
2) Unavailability of specific tools: No single tool can cater to each of the technology. For e.g. database testing tools for NoSQL might not be a fit for message queues. Similarly, custom tools and utilities will be require to test map reduce jobs.
3) Test scripting: There are no record and playback mechanisms for such systems. A high degree of scripting is required to design test cases and scenarios.
4) Test environment: It might not always be feasible to create a performance testing environment that can simulate production usages because of the cost and scale. We need to have a scaled down version sufficient to predict realistic performance with all the components.
5) Monitoring solutions: Since each component has a different way of exposing performance metrics limited solutions exists that can monitor the entire environment for performance anomalies and detect issues.
6) Diagnostic solutions: Custom solutions need to develop to further drill down the performance bottleneck areas.

# Performance Testing Approach

Any big data project involves in processing huge volumes of structured and unstructured data and is processed across multiple, nodes to complete the job in less amount of time. At times because of poor design and architecture performance is degraded. Some of the areas where performance issues can occur are imbalance in input slits, redundant shuffle and sorts, moving most of the aggregation computations to reduce process and so on. Performance testing is conducted by setting up huge volume of data in an environment

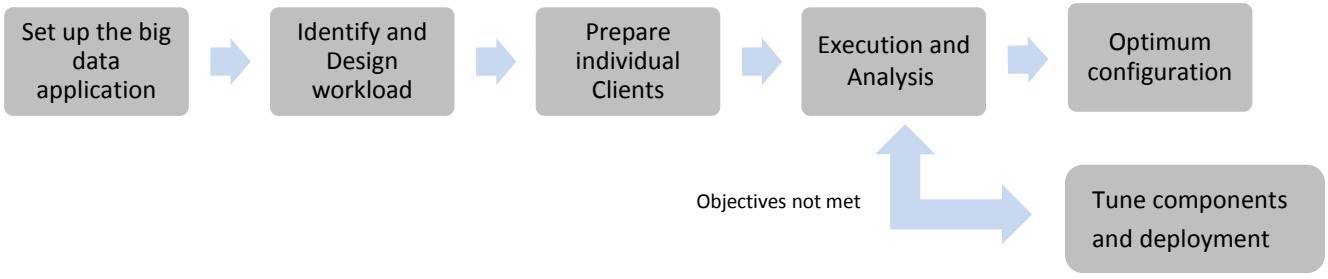
close to production. Utilities like Nagios, Zabbix, Hadoop monitoring etc. can be used to capture performance metrics and identify the bottlenecks. Performance metrics like memory, throughput, job completion time etc. are critical for monitoring and analysis.

## Performance characteristics of NoSQL

Relational databases have always relied upon some very hard-and-fast, structured rules that govern the conduct of database transactions. These are encoded in the ACID model and require that the database always preserve the atomicity, consistency, isolation and durability of database transactions. NoSQL turns the ACID model upside down and instead of requiring rigid adherence to the ACID principles, the model offers three loose guidelines: basic availability, soft state and eventual consistency. Traditional relational database solutions are not suitable for a majority of data sets like for e.g. the data is too unstructured and/or too voluminous for a traditional RDBMS to handle. Such Big Data systems cannot be analyzed with SQL or similar technologies. In fact, database schema does not allow complex unstructured formats to be defined and managed in these data warehouses.

NoSQL solutions are very different from your usual RDBMS, but they are still bound by the usual constraints: CPU, I/O and most importantly how it is used! Although Cassandra is lighting fast and mostly I/O bound it's still Java and you have the usual problems – e.g. GC needs to be watched. Cassandra provides a lot of monitoring metrics but seeing the flow end-to-end really helps to understand whether the time is spent on the client, network or server and makes the runtime dynamics of Cassandra much clearer. This makes performance testing of such systems extremely important. Next, we look at some of the available solutions and how these can be used.

An end to end procedural approach involved in load testing a big data application is represented diagrammatically below:




The process starts with the setting up of the Big data cluster which is to be tested for performance. Depending on the typical usage of the components, we need to identify and create corresponding workloads. For e.g. a typical workload could be that 90% of the time inserts are performed while the remaining are read operations. As a next step, custom scripts are created. Depending on the tool, this can be done from the UI or

through some APIs. Further, tests are executed to simulate realistic usage and results are identified. Based on the results, we can tune the cluster and re-execute the tests till the maximum performance is achieved.

To measure the scalability, the same tests can be repeated by increasing the number of clusters. The throughout must scale linearly with the number of clusters.

# Available Performance Testing Solutions

1) YCSB: YCSB is a cloud service testing client that performs reads, writes and updates according to specified workloads. Running from the command line it can create an arbitrary number of threads that will query the system under test. It will measure throughput in operations per second and record the latency in performing these operations. YCSB can run in parallel from multiple hosts.
   On each test client instance, from inside the ~/YCSB directory, run:
   ./bin/ycsb load cassandra-10 -P workloads/workloada -P cassandra.props -threads 50 -s > loaddata-cassandra.results
   This command tells YCSB to run the load component of workload A (which inserts data that subsequent tests rely on) using the cassandra-1.* client (load cassandra-10 -P workloads/workloada), it also tells YCSB to load our configuration file (-P cassandra.props), use 50 client threads (-threads 50) and output status updates to stderr and test results to the loaddata-cassandra.results file (-s > loaddata-cassandra.results).
   At the end of each test, YCSB will output a summary of the test. This will be found in the piped output results files created in the YCSB directory (workloada-cassandra.results, etc.).
   Evaluate the results against your application throughput and latency requirements. Remember to sum the average throughput of each YCSB client to get the total average throughput. YCSB also reports the 95th and 99th percentile latencies for each test.
2) SandStorm: SandStorm is an automated performance testing tool that supports big data performance testing. It provides a scripting interface to load and stress test any big data stack. It also provides a Recorder component to quickly create test scripts for the end-to-end workflows and test the performance of entire application. Additionally, it also provides resource monitoring of the big data cluster to identify the performance issues during the test execution.

3) JMeter: JMeter provides few plugins to apply load to Cassandra. This plugin acts as a client of Cassandra and can send requests over Thrift. The plugin is fully configurable. After creating the thread group you can confirm that the Cassandra JMeter plugin has been loaded correctly. Select the Thread Group, right click and a pull down menu will appear. Select Add, then Sampler. The 7 Cassandra Samplers should be included in the list.
4) Independent Custom Utilities: Cassandra stress test etc.

# Critical Performance Areas

As a general rule, it's important to note that simply adding nodes to a cluster will not improve performance on its own. You need to replicate the data appropriately, and then send traffic to all the nodes from your clients. If you aren't distributing client requests, the new nodes could just stand by somewhat idle.
The following is the list of important areas that should be looked at and monitored to achieve optimum performance from the Big datacluster.

1) Data Storage: How data is stored across different nodes. What is the replication factor?
2) Commit logs: You can set the value for how large the commit log is allowed to grow before it stops appending new writes to a file and creates a new one. This is similar to setting log rotation on Log4J.
3) Concurrency: There are two settings related to how many threads can perform read and write operations: concurrent_reads and concurrent_writes.
4) Caching: There are several settings related to caching, both within Cassandra and at the operating system level. Caches can use considerable memory, and it's a good idea to tune them carefully once you understand your usage patterns.
   There are two primary caches built into Cassandra: a row cache and a key cache. The row cache caches complete rows (all of their columns), so it is a superset of the key cache. If you are using a row cache for a given column family, you will not need to use a key cache on it as well.
5) Timeouts: Values for connection time out, query timeout etc.
6) JVM parameters: GC collection algorithms, heap size etc.
7) Map reduce performance: Sorts, merge etc.
8) Message queue: Message rate, size etc.

# Conclusion

We believe that organizations have to choose the solutions that best suit their needs, to solve their performance testing challenges. Nevertheless, it is desirable to get all the possible options like testing and monitoring available under the same hood, as it will help in reducing the complications that arise when dealing with multiple alternatives to achieve a common goal. When it comes to Big Data, we can use any of the above tools to run performance and stress test directly on the database to identify and resolve any potential bottlenecks.

# About the Author (s)

**Mustafa Batterywala**

Mustufa is an experienced, delivery-focused technology specialist with close to 10 years development experience. He is currently working as a Performance Architect at Impetus Technologies. He brings diverse technology solution experience in design, development, testing and deployment. His expertise includes performance engineering for software products, bottleneck identification and diagnostics, profiling and tuning app and database servers. The recent focus has been on cloud computing, big data and mobile performance.

He has participated at various conferences like GTAC2010, Star West 2011, 2012, Cloud Connect 2012, QAI 2012, Better Software Conference 2013, World Conference on Next Generation Testing 2013, CMG 2013 and many more. A regular speaker at webinars, Mustufa is also an active contributor to performance engineering forums.

**Shirish Bhale**

Shirish Bhale heads the Performance Engineering practice at Impetus Technologies. He has been involved in various initiatives and R&D in this domain and has played a pivotal role in design and development of SandStorm, Impetus' enterprise performance testing tool.

For the past sixteen years, Shirish has been actively involved in performance engineering and testing space, seeing the software industry move from client server to an ASP to a SaaS and now to a cloud model. He is also in charge of account management, project deliveries, and enduring customer relationships.

Shirish is a certified SCRUM Master.

Shirish is a regular speaker and contributor at technology conferences (including SQE), forums, workshops and webinars. He has also delivered trainings on Performance Engineering to mid-level engineers.