A platform for big data analytics on distributed scale-out storage system

Kyar Nyo Aye*

Software Department,
Computer University (Thaton),
The Union of Myanmar
Email: kyarnyoaye@gmail.com
*Corresponding author

Thandar Thein

Hardward Department,
University of Computer Studies (Yangon),
The Union of Myanmar
Email: thandartheinn@gmail.com

Abstract: Big data analytics is the process of examining large amounts of data of a variety of types to uncover hidden patterns, unknown correlations and other useful information. Hadoop-based platform emerges to deal with big data. In Hadoop NameNode is used to store metadata in a single system's memory, which is a performance bottleneck for scale-out. Gluster file system has no performance bottlenecks related to metadata. To achieve massive performance, scalability and fault tolerance for big data analytics, a big data platform is proposed. The proposed big data platform consists of big data storage and big data processing. The Hadoop big data platform and the proposed big data platform are implemented on commodity Linux virtual machines clusters and performance evaluations are conducted. According to the evaluation analysis, the proposed big data platform provides better scalability, fault tolerance, and faster query response time than the Hadoop platform.

Keywords: big data; big data analytics; big data platform; Hadoop MapReduce; Gluster file system; Apache Pig; Apache Hive; Jaql.

Reference to this paper should be made as follows: Aye, K.N. and Thein, T. (2015) 'A platform for big data analytics on distributed scale-out storage system', *Int. J. Big Data Intelligence*, Vol. 2, No. 2, pp.127–141.

Biographical notes: Kyar Nyo Aye is a Tutor of Software Department at Computer University (Thaton). She received her degree of Bachelor of Computer Science (BCSc), the degree of Bachelor of Computer Science (Honours), and the degree of Master of Computer Science in 2004, 2005, and 2009. She received her PhD in 2013. Her research interests include information retrieval, distributed databases, distributed systems, cloud computing, mobile computing, big data analytics and big data technologies.

Thandar Thein received her MSc (Computer Science) and PhD in 1996 and 2004, respectively from University of Computer Studies, Yangon (UCSY), Myanmar. She did her post doctorate research in Korea Aerospace University. She is currently a Professor of UCSY. Her research interests include cloud computing, mobile cloud computing, big data, digital forensic, security engineering, and network security and survivability.

1 Introduction

Today, information is generated continuously around the globe. Almost every growing organisation wants to automate most of its business processes and is using IT to support every conceivable business function. This is resulting into huge amount of data being generated in the form of transactions and interactions. Web has become an important interface for interactions with suppliers and customers generating the huge amount of data in the form of e-mails, etc. Besides this, there is a huge amount of data emitted automatically in the form of logs like network logs and web server logs.

Various telecom service providers get huge amount of data in the form of conversations and call data records. Various social N/W sites have started getting TBs of data every day in the form of tweets, blogs, comments, photos and videos, etc. Facebook generates 4 TBs of compressed data every day. Web Companies like these get huge amount of click stream data generated daily as well. Hospitals have

data about the patients, their diseases and the data generated by various medical devices as well. Sensors used in various machines used for production keep generating so much of event data in seconds. Almost every sector like transport, finance is seeing a tsunami of data.

Now the important question that arises at this point of time is how do we store and process such huge amount of data most of which is Semi structured or Unstructured. There is a high-level categorisation of big data platforms to store and process them in a scalable, fault tolerant and efficient manner (http://bigdataanalytics.blogspot.com). The first category includes massively parallel processing or MPP Data warehouses that are designed to store huge amount of structured data across a cluster of servers and perform parallel computations over it. Most of these solutions follow shared nothing architecture which means that every node will have a dedicated disk, memory and processor. All the nodes are connected via high speed networks. As they are designed to hold structured data so there is a need to extract the structure from the data using an ETL tool and populate these data sources with the structured data.

These MPP data warehouses include:

- MPP databases: these are generally the distributed systems designed to run on a cluster of commodity servers, e.g., AsternCluster, Greenplum, DATAllegro, IBM DB2, Kognitio WX2, Teradata
- appliances: a purpose-built machine with preconfigured MPP hardware and software designed for analytical processing, e.g., Oracle optimised Warehouse, Teradata Machines, Netezza Performance Server and Sun's Data Warehousing Appliance
- columnar databases: they store data in columns instead of rows, allowing greater compression and faster query performance, e.g., Sybase IQ, Vertica, InfoBright Data Warehouse, ParAccel.

Another category includes distributed file systems like Hadoop to store huge unstructured data and perform MapReduce computations on it over a cluster built of commodity hardware. Pavlo et al. (2009) described and compared MapReduce paradigm and parallel DBMSs for large scale data analysis and defined a benchmark consisting of a collection of tasks to be run on an open source version of MR as well as on two parallel DBMSs. Hadoop is a popular open source MapReduce implementation which is being used in companies like Yahoo, Facebook, etc., to store and process extremely large datasets on commodity hardware. However, in Hadoop the NameNode can become a performance bottleneck because it keeps the directory tree of all files in the Hadoop distributed file system. The architecture within Gluster does not depend on metadata in any way. Therefore, Gluster has no performance bottlenecks and no inconsistency risks related to metadata. In addition, using Hadoop was not easy for end users, especially for those users who were not familiar with MapReduce. The MapReduce programming model is very low level and requires developers to write custom programs which are hard to maintain and reuse. Hadoop lacked the expressiveness of popular query languages like SQL and as a result users ended up spending hours to write programs for even simple analysis. In order to analyse this data more productively, the query capabilities of Hadoop need to be improved. So, several application development languages have emerged to make it easier to write MapReduce programs in Hadoop and that run on top of Hadoop. Among them, Hive, Pig, and Jaql are popular.

The aim of this paper is to propose big data platform that is built upon open source and built on Hadoop MapReduce, Gluster File System, Apache Pig, Apache Hive and Jaql. The rest of the paper is organised as follows: in Section 2, we explain big data concepts and technologies such as big data and big data analytics. In Section 3 we introduce our proposed big data platform and performance evaluations are conducted in Section 4. Then vendor products for big data analytics are explained in Section 5 and conclusion is described in Section 6.

2 Big data concepts and technologies

This section provides an overview of big data, big data Analytics, Hadoop and MapReduce framework, Apache Pig, Apache Hive, Jaql, Gluster File System and big data platform. Due to space constraints, some aspects are explained in a highly simplified manner. A detailed description of them can be found in Eaton et al. (2011), Carter (2011) and Russom (2011).

2.1 Big data

Big data are datasets that grow so large that they become awkward to work with using on-hand database management tools. Difficulties include capture, storage, search, sharing, analytics, and visualising. There are three characteristics of Big data: volume, variety, and velocity.

- Volume: volume is the first and most notorious feature. It refers to the amount of data to be handled. The sheer volume of data being stored today is exploding. In the year 2000, 800,000 petabytes (PB) of data were stored in the world. This number is expected to reach 35 zettabytes (ZB) by 2020. Organisations that do not know how to manage massive volumes of data are overwhelmed by it. But the opportunity exists, with the right technology platform, to analyse almost all of the data to gain better insights.

- Variety: variety represents all types of data. With the explosion of sensors, and smart devices, as well as social collaboration technologies, data in an enterprise has become complex, because it includes not only traditional relational data, but also raw, semistructured, and unstructured data. To capitalise on the big data opportunity, enterprises must be able to analyse all types of data, both relational and non-relational: text, sensor data, audio, video, transactional, and more.
- Velocity: a conventional understanding of velocity typically considers how quickly the data is arriving and stored, and its associated rates of retrieval. However, today the term velocity is defined to data in motion: the speed at which the data is flowing. More and more of the data being produced today have a very short shelf-life, so organisations must be able to analyse this data in near real time if they hope to find insights in this data.

There are two types of big data: data at rest (e.g., collection of what has streamed, web logs, e-mails, social media, unstructured documents and structured data from disparate system) and data in motion (e.g., Twitter/Facebook comments, stock market data and sensor data). So dealing effectively with big data requires performing analytics against the volume and variety of data while it is still in motion, not just after it is at rest.

2.2 Big data analytics

Big data analytics is the application of advanced analytic techniques to very big datasets. Advanced analytics is a collection of techniques and tool types, including predictive analytics, data mining, statistical analysis, complex SQL, data visualisation, artificial intelligence, natural language processing, and database methods that support analytics (such as MapReduce, in-database analytics, in-memory database, columnar data stores).

There are three approaches for big data analytics: direct analytics over MPP DW, indirect analytics over Hadoop and direct analytics over Hadoop.

- Direct analytics over MPP DW: the first approach for big data analytics is using a BI tool directly over any of the MPP DW. For any analytical request by the user the BI tool will send SQL queries to these DWs. These DWs will execute the queries in a parallel manner across the cluster and return the data to BI tool for further analytics.
- Indirect analytics over Hadoop: the second approach is indirect analytics over Hadoop which processes, transforms and structures the data inside Hadoop and then exports the structured data into RDBMS. The BI tool will work with the RDBMS to provide the analytics.
- Direct analytics over Hadoop: The last approach is performing analytics directly over Hadoop. In this case all the queries will be executed as MR jobs over big unstructured data placed into Hadoop.

2.3 Hadoop and MapReduce framework

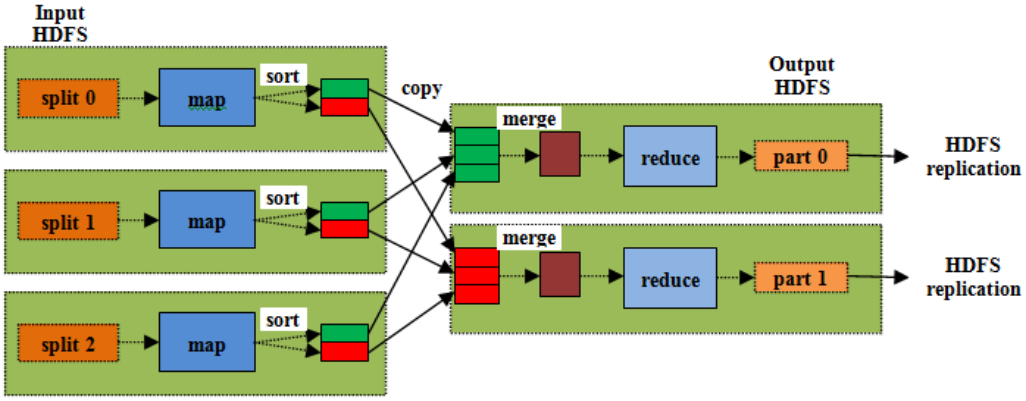
Apache Hadoop is an open source software project that enables the distributed processing of large datasets across clusters of commodity servers. It is designed to scale up from a single server to thousands of machines, with a very high degree of fault tolerance. Rather than relying on high-end hardware, the resiliency of these clusters comes from the software's ability to detect and handle failures at the application layer (Shvachko et al., 2010; Agrawal, 2011; White, 2009; HDFS Architecture Guide).

Hadoop enables a computing solution that is:

- Scalable: new nodes can be added as needed and added without needing to change data formats, how data is loaded, how jobs are written, or the applications on top.
- Cost effective: Hadoop brings massively parallel computing to commodity servers. The result is a sizeable decrease in the cost per terabyte of storage, which in turn makes it affordable to model all data.
- Flexible: Hadoop is schema-less, and can absorb any type of data, structured or not, from any number of sources. Data from multiple sources can be joined and aggregated in arbitrary ways enabling deeper analyses than any one system can provide.
- Fault tolerant: when a node fails, the system redirects work to another location of the data and continues processing.

A MapReduce framework typically divides the input dataset into independent tasks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the jobs are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and reexecuting the failed tasks (Dean and Ghemawat, 2008; http://Hadoop.apache.org/MapReduce). Hadoop is supplemented by an ecosystem of Apache projects, such as Pig and Hive, that extend the value of Hadoop and improves its usability. Figure 1 shows MapReduce data flow with multiple reduce tasks.

Figure 1 MapReduce data flow with multiple reduce tasks (see online version for colours)



2.4 High level query languages

There are three high level query languages for big data analytics: Apache Pig, Apache Hive and Jaql (http://www.macs.hw.ac.uk/~rs46/files/publications/MapReduce-Languages/Comparison_Of_High_Level_Data_Query_Languages.pdf.).

2.4.1 Apache Pig

Apache Pig (Olston et al., 2008) is a platform for analysing large datasets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. Pig is made up of two components: the first is the language itself, which is called PigLatin, and the second is a runtime environment where PigLatin programs are executed. The Pig runtime environment translates the program into a set of map and reduce tasks and runs them. This greatly simplifies the work associated with the analysis of large amounts of data and lets the developer focus on the analysis of the data rather than on the individual map and reduce tasks.

2.4.2 Apache Hive

Apache Hive (Thusoo et al., 2010) is an open source data warehousing solution built on top of Hadoop. Hive supports queries expressed in a SQL-like declarative language – *HiveQL*, which are compiled into MapReduce jobs that are executed using Hadoop. In addition, HiveQL enables users to plug in custom MapReduce scripts into queries. The language includes a type system with support for tables containing primitive types, collections like arrays and maps, and nested compositions of the same. Hive also includes a system catalogue – *Metastore* – that contains schemas and statistics, which are useful in data exploration, query optimisation and query compilation.

2.4.3 Jaql

Jaql (Beyer et al., 2011) is a functional, declarative query language that is designed to process large datasets. For parallelism, Jaql rewrites high-level queries into low-level queries consisting of MapReduce jobs. Jaql is primarily a query language for JavaScript Object Notation (JSON). JSON is the popular data interchange format because it is easy for humans to read, and because of its structure, it is easy for applications to parse or generate. Both Jaql and JSON are record-oriented models, and thus fit together perfectly. JSON is not the only format that Jaql supports, Jaql is extremely flexible and can support many semistructured data sources such as XML, CSV, flat files and more.

2.5 Gluster file system

GlusterFS is a scalable open source clustered file system that offers a global namespace, distributed front end, and scales to hundreds of petabytes without difficulty. It is also a software-only, highly available, scalable, centrally managed storage pool for unstructured data. It is also scale-out file storage software for NAS, object, big data. By leveraging commodity hardware, Gluster also offers extraordinary cost advantages benefits that are unmatched in the industry. There are many advantages of Gluster over any other file systems. These advantages are:

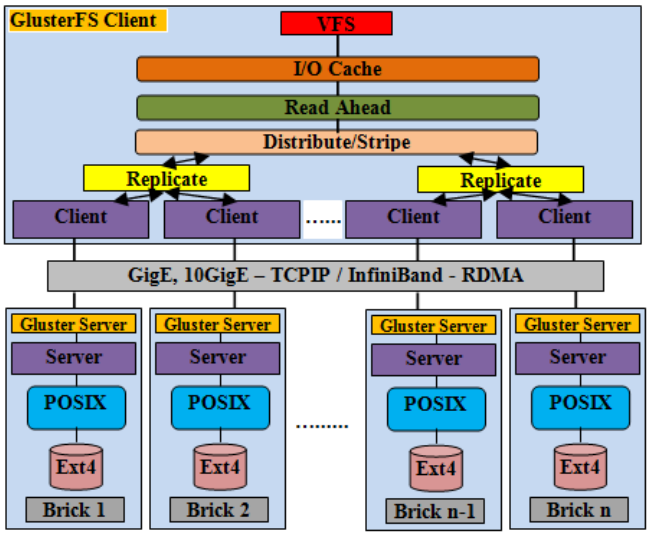
- it is faster for each individual operation because it calculates metadata using algorithms and that approach is faster than retrieving metadata from any storage media
- it is faster for large and growing individual systems because there is never any contention for any single instance of metadata stored at only one location
- it is faster and achieves true linear scaling for distributed deployments because each node is independent in its algorithmic handling of its own metadata, eliminating the need to synchronise metadata
- it is safer in distributed deployments because it eliminates all scenarios of risk which are derived from out-of-synch metadata (Gluster, 2011).

Both performance and capacity can be scaled out linearly in Gluster by employing three fundamental techniques:

- the elimination of metadata
- effective distribution of data to achieve scalability and reliability
- the use of parallelism to maximise performance via a fully distributed architecture

Figure 2 describes the Gluster file system architecture.

Figure 2 Gluster file system architecture (see online version for colours)



2.6 Big data platform

Big data platform cannot just be a platform for processing data; it has to be a platform for analysing that data to extract insight from an immense volume, variety, and velocity of that data. The main components in the big data platform provide:

- *deep analytics*: a fully parallel, extensive and extensible toolbox full of advanced and novel statistical and data mining capabilities
- *high agility*: the ability to create temporary analytics environments in an end-user driven, yet secure and scalable environment to deliver new and novel insights to the operational business
- *massive scalability*: the ability to scale analytics and sandboxes to previously unknown scales while leveraging previously untapped data potential
- *low latency*: the ability to instantly act based on these advanced analytics in the operational, production environments (http://blogs.oracle.com/datawarehousing/entry/big_data_achieve_the_impossible).

3 Proposed big data platform

The proposed big data platform performs large-scale data analysis by using MapReduce framework on unstructured data stored in GlusterFS over distributed scale-out storage system. GlusterFS can provide these features: scalability to petabytes and beyond, affordability (use of commodity hardware), flexibility (deploy in any environment), linearly scalable performance, high availability, and superior storage economics. By combining these advantages of GlusterFS with parallel data processing, schema free processing and simplicity of MapReduce programming model, the proposed big data platform can perform large scale data analysis efficiently and effectively. The proposed big data platform is shown in Figure 3.

The proposed big data platform consists of four layers: application layer, processing layer, interface layer and storage layer.

- Application layer: multiple GlusterFS clients use high level query languages such as Hive, Pig, and Jaql to submit analytical jobs. These jobs are compiled into MapReduce jobs. Pig uses MapReduce to execute all of its data processing. It compiles the Pig Latin scripts that users write into a series of one or more MapReduce jobs that it then executes. In Hive, all commands and queries go to the Driver, which compiles the input, optimise the computation required, and executes the required steps, usually with MapReduce jobs. The Metastore is a separate relational database where Hive persists table schemas and other system metadata. Jaql consists of a scripting language and compiler, as well as a runtime component for Hadoop. The Jaql compiler automatically rewrites Jaql scripts so they can run in parallel on Hadoop.
- Processing layer: the jobtracker coordinates all these MapReduce jobs by scheduling tasks to run on tasktrackers. The tasktrackers run map tasks and reduce tasks.
- Interface layer: the file system function calls flow from MapReduce jobs to the Gluster java library through the FUSE mount. These file system calls are translated into POSIX file system calls.
- Storage layer: the Gluster storage pool is a trusted network of storage servers which consist of one or more bricks. A brick is the GlusterFS basic unit of storage, represented by an export directory.

3.1 Big data analytics on proposed platform

The proposed platform can handle any data type such as call data records, web clickstreams, network logs, and so on. Hadoop MapReduce processes these data that are stored in GlusterFS on commodity servers to extract useful information for users. Users can use high level query languages such as Hive, Pig and Jaql to get analytical results. Figure 4 describes the conceptual architecture of big data analytics on proposed platform.

Figure 3 Proposed big data platform (see online version for colours)

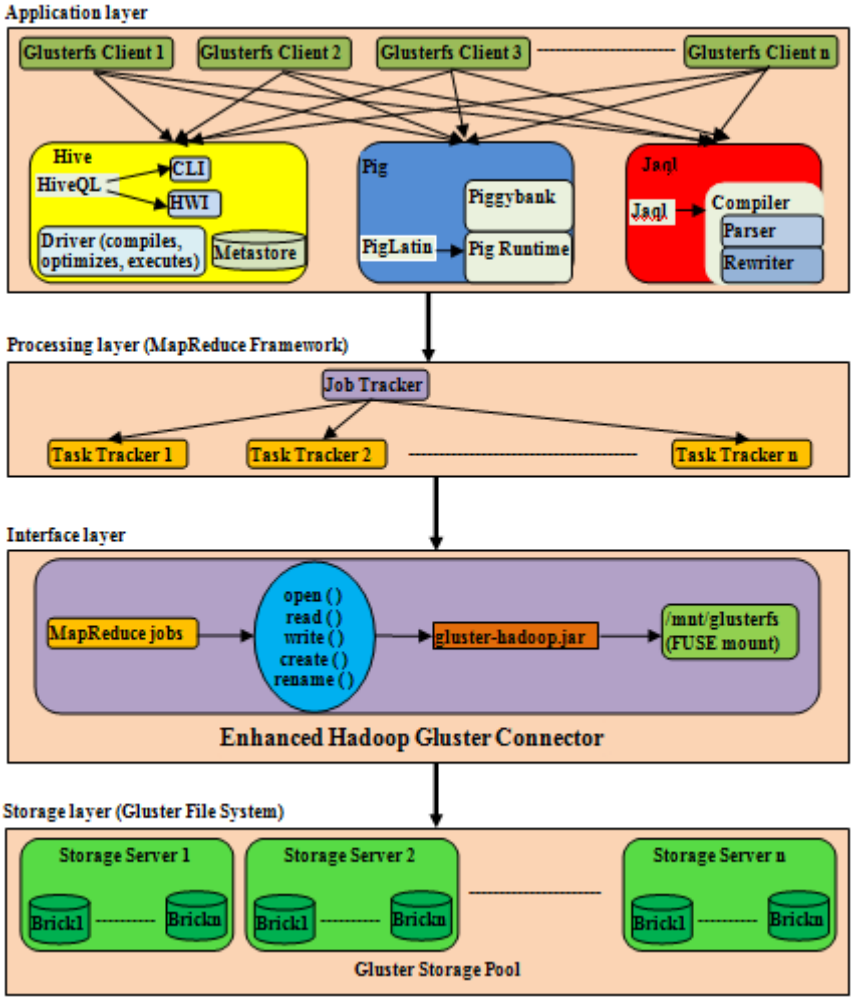
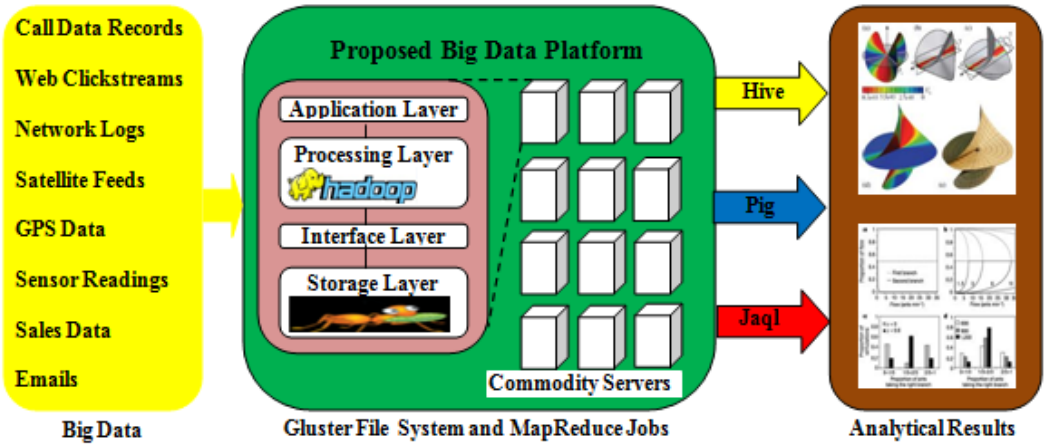


Figure 4 Conceptual architecture of big data analytics on the proposed platform (see online version for colours)



3.2 Gluster file system server volumes

There are seven types of GlusterFS server volumes. These are:

- Distributed volume: it randomly distributes files throughout the bricks in the volume. It can be used in environments where the requirement is to scale storage and the redundancy is either not important or is provided by other hardware or software layers.
- Replicated volume: it creates copies of files across multiple bricks in the volume. It can be used in environments where high-availability and high-reliability are critical.

- Striped volume: it stripes data across bricks in the volume. For best results, it should be used only in high concurrency environments accessing very large files.
- Distributed replicated volume: it distributes files across replicated bricks in the volume. It can be used in environments where the requirement is to scale storage and high-reliability is critical.
- Distributed striped volume: it stripes files across two or more nodes in the cluster. It should be used in environments where the requirement is to scale storage and in high concurrency environments accessing very large files is critical.
- Striped replicated volume: it stripes data across replicated bricks in the cluster. It should be used in highly concurrent environments where there is parallel access of very large files and performance is critical.
- Distributed striped replicated volume: it distributes striped data across replicated bricks in the cluster. It should be used in highly concurrent environments where there is parallel access of very large files and performance is critical.

Enhanced Hadoop gluster connector can support MapReduce workloads on all these volume types. To achieve linear scalability and high performance for big data analytics, striped replicated volume and distributed striped replicated volume are the best storage options.

4 Performance evaluation

We have evaluated the performance of two big data platforms on three commodity Linux clusters – first cluster with two virtual machines (testbed 1), second cluster with three virtual machines (testbed 2), and third cluster with four virtual machines (testbed 3). The VMs are interconnected via a 1 Gigabit ethernet. The host machine runs Windows 7 Ultimate and has Intel Core i7-3.40 GHz processor, 4 GB physical memory, and 950 GB disk. As software components, Hadoop 0.20.2, Gluster 3.4.0, Hive 0.9.0, Pig 0.10.0 and Jaql 0.5.1 are used. Table 1 shows the experimental setup to evaluate the query performance of two big data platforms.

Table 1 Experimental setup for performance evaluations

Testing environments	*Testing parameters*
Commodity Linux VMs clusters	Cluster 1 (2VMs)
	Cluster 2 (3VMs)
	Cluster 3 (4VMs)
Host specification	Intel ® Core™ i7-2600
	CPU at 3.40 GHz
	4 GB RAM
	1 TB hard disk
	1 Gigabit ethernet
Software components	Processing system
	• Hadoop 0.20.2
	Storage system
	• Gluster 3.4.0
	High level query languages
	• Hive 0.9.0
	• Pig 0.10.0
	• Jaql 0.5.1

The parameters of testbed 1, testbed 2, and testbed 3 are shown in Tables 2 to 4 respectively.

Table 2 Specification of testbed 1

MapReduce + HDFS cluster		*MapReduce + GlusterFS cluster*	
VM1	*VM2*	*VM1*	*VM2*
Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz
1,024 MB RAM	512 MB RAM	1,024 MB RAM	512 MB RAM
50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk
NN	DN	Gluster storage pool (VM1 + VM2)	
SNN			
DN			
JT	TT	JT	TT
TT		TT	

Notes: VM1 = Virtual Machine1 NN = NameNode JT = JobTracker
VM2 = Virtual Machine2 DN = DataNode TT = TaskTracker
SNN = Secondary NameNode

Table 3 Specification of testbed 2

MapReduce + HDFS cluster			*MapReduce + GlusterFS cluster*		
VM1	*VM2*	*VM3*	*VM1*	*VM2*	*VM3*
Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz
1,024 MB RAM	512 MB RAM	512 MB RAM	1,024 MB RAM	512 MB RAM	512 MB RAM
50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk
NN	DN	DN		Gluster storage pool (VM1 + VM2)	
SNN					
DN					
JT	TT	TT	JT	TT	TT
TT			TT		

Notes: VM1 = Virtual Machine1 NN = NameNode JT = JobTracker
VM2 = Virtual Machine2 DN = DataNode TT = TaskTracker
VM3 = Virtual Machine3 SNN = Secondary NameNode
VM4 = Virtual Machine4

Table 4 Specification of testbed 3

MapReduce + HDFS cluster				*MapReduce + GlusterFS cluster*			
VM1	*VM2*	*VM3*	*VM4*	*VM1*	*VM2*	*VM3*	*VM4*
Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz	Intel ® Core™ i7-2600 CPU at 3.40 GHz
1,024 MB RAM	512 MB RAM	512 MB RAM	512 MB RAM	1,024 MB RAM	512 MB RAM	512 MB RAM	512 MB RAM
50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk	50 GB hard disk
NN	DN	DN	DN		Gluster storage pool (VM1 + VM2)		
SNN							
DN							
JT	TT	TT	TT	JT	TT	TT	TT
TT				TT			

Notes: VM1 = Virtual Machine1 NN = NameNode JT = JobTracker
VM2 = Virtual Machine2 DN = DataNode TT = TaskTracker
VM3 = Virtual Machine3 SNN = Secondary NameNode
VM4 = Virtual Machine4

US census dataset (http://www2.census.gov/census_2010/04-Summary_File_1) is used to evaluate the performance of two big data platforms. The dataset consists of 331 tables. Population table is used to evaluate the query performance of two big data platforms. It consists of 12,905,514 records for 52 states (50 US states, the District of Columbia, and Puerto Rico). Table 5 describes the data dictionary of population table.

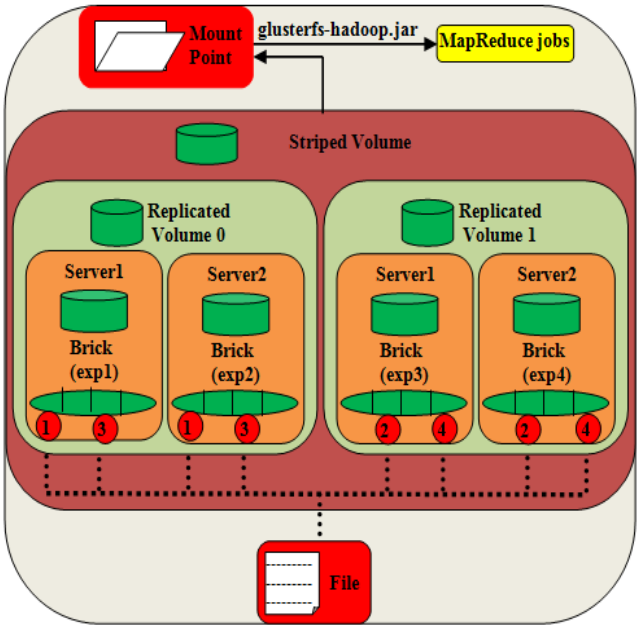
Striped replicated volume is used for storage in testbed 1 and testbed 3 and distributed striped replicated volume is used in testbed 2. To create a striped replicated volume in testbed 1: # gluster volume create population volume stripe 2 replica 2 server1:/exp1 server2:/exp2 server1:/exp3 server2:/exp4.

The striped replicated volume used in testbed 1 is shown in Figure 5.

Table 5 Data dictionary of population table

Field name	*Data dictionary reference name*
ID	Record ID
FILEID	File identification
STUSAB	State/US abbreviation
CHARITER	Characteristic iteration
CIFSN	Characteristic iteration file sequence number
LOGRECNO	Logical record number
P0010001	Population

Figure 5 Striped replicated volume for testbed 1 (see online version for colours)



To create a distributed striped replicated volume in testbed 2: # gluster volume create population volume stripe 2 replica 2 server1:/exp1 server2:/exp2 server3:/exp3 server1:/exp4 server2:/exp5 server3:/exp6 server1:/exp7 server2:/exp8,

The distributed striped replicated volume used in testbed 2 is described in Figure 6.

To create a striped replicated volume in testbed 3: # gluster volume create population volume stripe 2 replica 2 server1:/exp1 server2:/exp2 server3:/exp3 server4:/exp4.

The striped replicated volume used in testbed 3 is shown in Figure 7.

4.1 Sample analytical workloads

Four queries are used as sample analytical workloads for performance evaluation of two big data platforms. Figure 8 shows HiveQL, PigLatin, and Jaql for the first query.

The first section in Figure 8 shows HiveQL of creating a population table, loading the file into the table, and finding the records where population is greater than 30,000. The second section describes a pig program that takes a file composed of population data, selects only those records whose population is greater than 30,000, and displays the result. The last section shows a Jaql sample that finds the records where population is greater than 30,000.

Figure 9 illustrates HiveQL, PigLatin, and Jaql for the second query.

Figure 6 Distributed striped replicated volume for testbed 2 (see online version for colours)

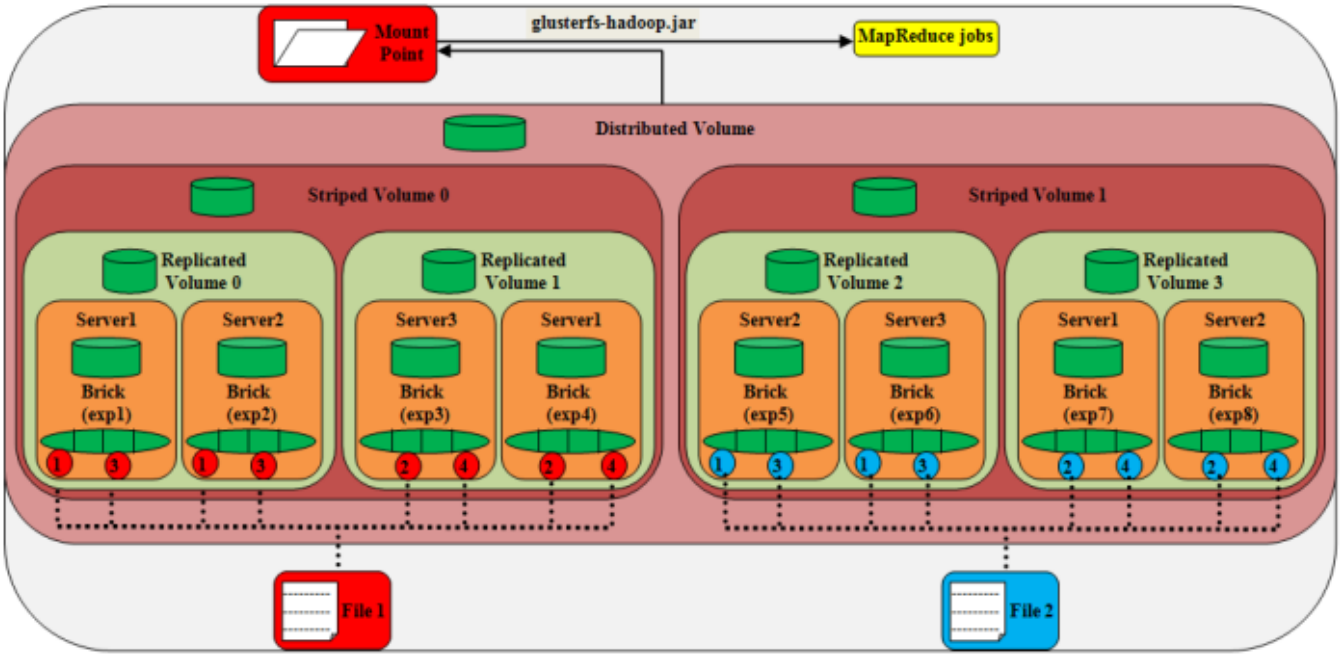


Figure 7 Striped replicated volume for testbed 3 (see online version for colours)

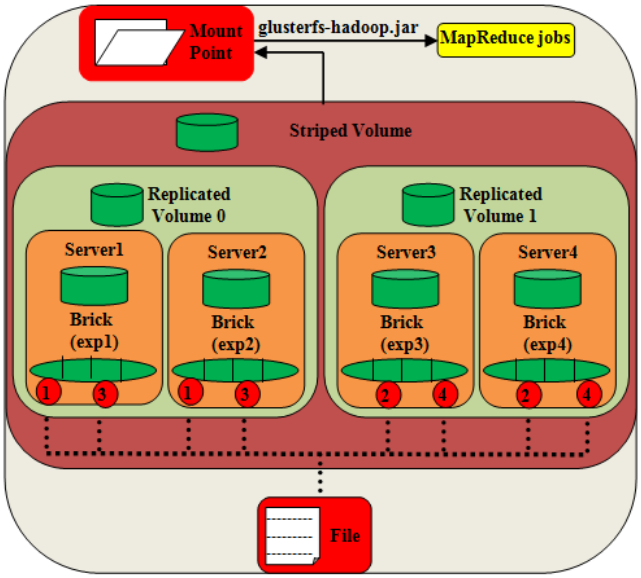


Figure 8 HiveQL, PigLatin, and Jaql for the first query

The HiveQL (Hive Query Language) is **hive**> create table population (ID int, FILEID string, STUSAB string, CHARITER string, CIFSN string, LOGRECNO string, P0010001 int) row format delimited fields terminated by '\,' stored as textfile; **hive**> load data inpath '/user/root/population.csv' overwrite into table population; **hive**> select * from population where P0010001 > 30000;
The PigLatin is **grunt**> population = load '/user/root/population.csv' using PigStorage(',') as (ID: int, FILEID: chararray, STUSAB: chararray, CHARITER: chararray, CIFSN: chararray, LOGRECNO: chararray, P0010001: int); **grunt**> result = filter population by P0010001 > 30000; **grunt**> dump result;
The Jaql is **jaql**> $population= read(del("/user/root/population.csv", { schema: schema { ID: long, FILEID: string, STUSAB: string, CHARITER: string, CIFSN: string, LOGRECNO: string, P0010001: long} })); **jaql**> $population -> filter $.P0010001 > 30000;

Figure 9 HiveQL, PigLatin, and Jaql for the second query

The HiveQL is **hive**> select count(*) from population;
The PigLatin is **grunt**> grouped = group population all; **grunt**>result = foreach grouped generate COUNT (population); **grunt**> dump result;
The Jaql is **Jaql**> $population -> group into count($);

The first and last sections in Figure 9 show queries to find the number of records in population table using Hive and Jaql respectively. The second section illustrates a pig program that groups the population, and displays the number of records in that group.

Figure 10 describes HiveQL, PigLatin, and Jaql for the third query.

Figure 10 HiveQL, PigLatin, and Jaql for the third query

group by STUSAB;
The PigLatin is **grunt**> grouped = group population by STUSAB; **grunt**> result = foreach grouped generate group, SUM(population.P0010001); **grunt**> dump result;
The Jaql is **jaql**>$population -> group by $STUSAB={$.STUSAB} into {$STUSAB, total: sum($[*].P0010001)};

The first section in Figure 10 shows HiveQL of finding the total population for each state. The second section describes a pig program that groups the population by the state, and displays the sum of the number of population for each state. The last section shows a Jaql example that finds the total population for each state.

Figure 11 demonstrates HiveQL, PigLatin, and Jaql for the fourth query.

Figure 11 HiveQL, PigLatin, and Jaql for the fourth query

The HiveQL is **hive**> select STUSAB, count(*) from population group by STUSAB;
The PigLatin is **grunt**> grouped = group population by STUSAB; **grunt**> result = foreach grouped generate group, COUNT(population); **grunt**> dump result;
The Jaql is **jaql**>$population -> group by $STUSAB={$.STUSAB} into {$STUSAB, count: count($)};

The first and last sections in Figure 11 show queries to find the number of records for each state using Hive and Jaql respectively. The second section illustrates a pig program that groups the population by the state, and displays the number of records for each state.

4.2 Experimental results

In this paper, the Hadoop big data platform (MapReduce and HDFS) and the proposed big data platform (MapReduce and GlusterFS) are implemented on three testbeds and performance evaluations are conducted with four queries on different record sizes. Figure 12 shows the query execution time for query 1 on testbed 1 for various states. The data sizes range from 2,571,686 records for ten states to 12,905,514 records for 52 states. Hive provides the fastest query execution time and Pig provides the slowest query execution time on both platforms. There are no significant differences in query execution time between the two platforms.

Figure 12 Query 1's execution time on testbed 1 (see online version for colours)

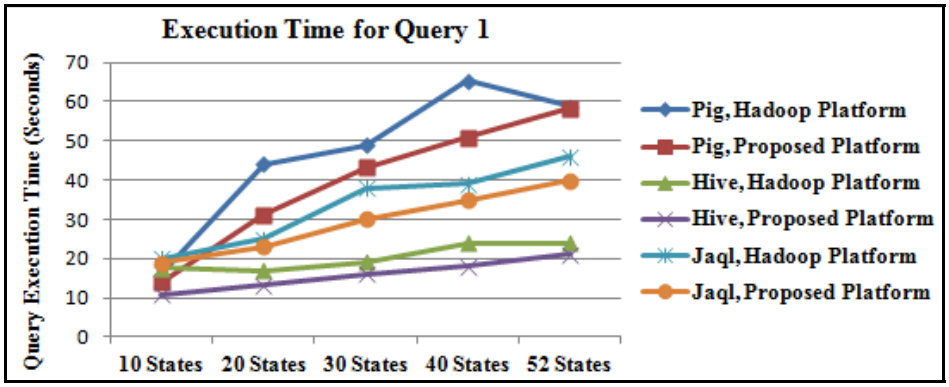


Figure 13 Query 2's execution time on testbed 1 (see online version for colours)

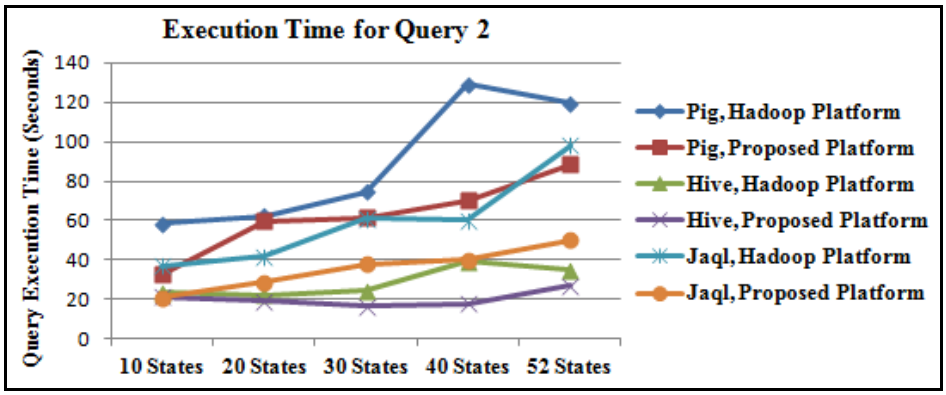


Figure 14 Query 3's execution time on testbed 1 (see online version for colours)

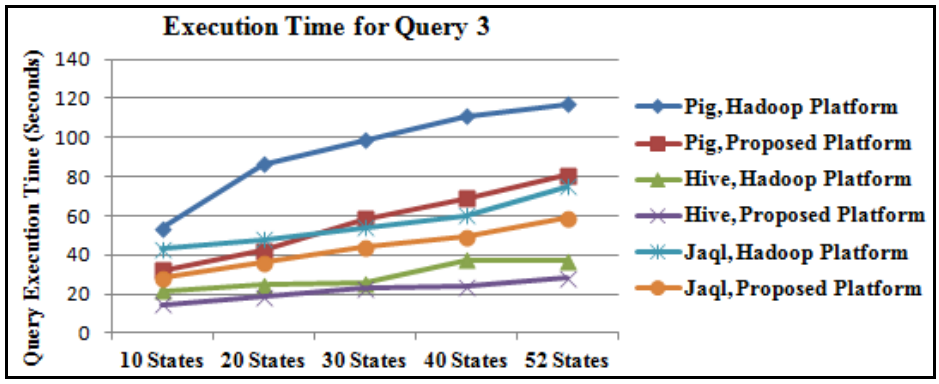


Figure 13 displays the query execution time for query 2 on testbed 1 for various states. Between ten states and 52 states Pig's query execution time and Jaql's query execution time fluctuate on Hadoop platform. The proposed platform provides more stable query execution time than the Hadoop platform.

Figure 14 illustrates the query 3's execution time on testbed 1, measured in seconds over a range from 2,571,686 records for ten states to 12,905,514 records for 52 states. There is a greater difference in Pig's query execution time between the two platforms. Hive's query execution time and Jaql's query execution time have no significant differences between the two platforms.

According to Figures 12 to 15, the proposed platform provides the faster query execution time than the Hadoop platform in three query languages.

The query execution time for query 1 on testbed 2 for various states is plotted in Figure 16. The proposed platform provides slightly faster query execution time in three query languages than the Hadoop platform. Figure 17 displays the query execution time for query 2 on testbed 2 for various states. Although there are significant differences in Pig's query execution time and Jaql's query execution time, Hive's query execution time has slight gap between the two platforms.

Figure 15 Query 4's execution time on testbed 1 (see online version for colours)

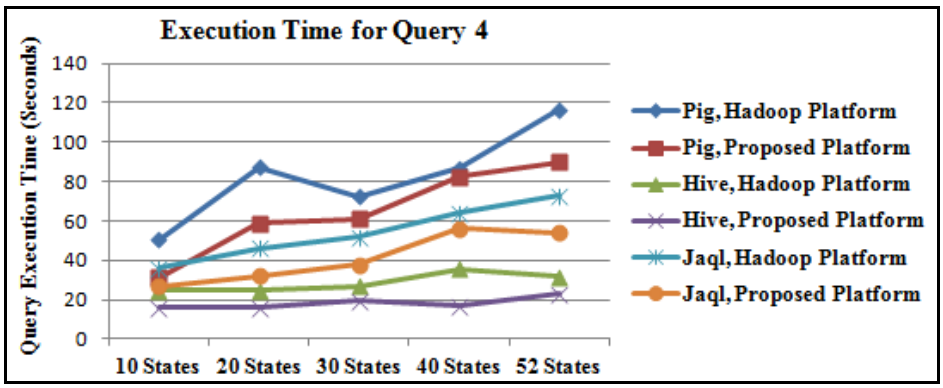


Figure 16 Query 1's execution time on testbed 2 (see online version for colours)

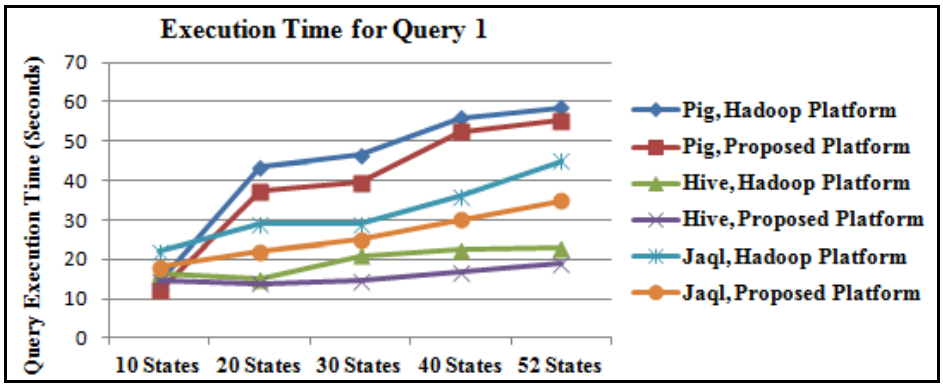


Figure 17 Query 2's execution time on testbed 2 (see online version for colours)

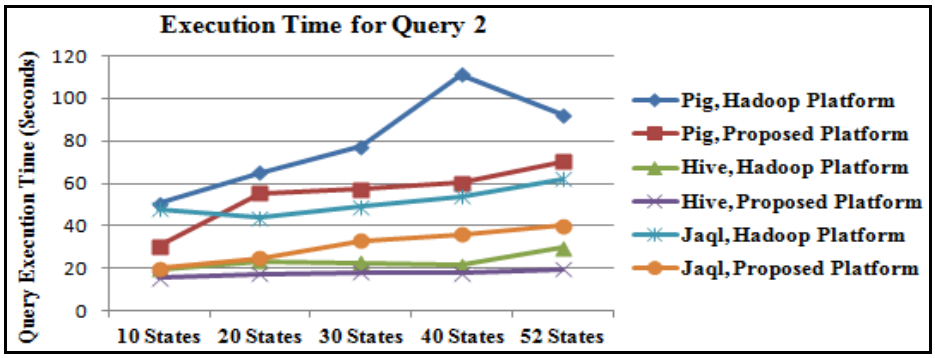


Figure 18 shows the query execution time for query 3 on testbed 2 for various states. Between ten states and 52 states Pig's query execution time fluctuates dramatically, hitting a peak of over 110 seconds on Hadoop platform. The proposed platform provides more stable query execution time than the Hadoop platform.

Figure 19 illustrates the query execution time for query 4 on testbed 2 for various states. According to Figure 19, there is fluctuation in Pig's query execution time on the Hadoop platform and Pig gives significant difference in query execution time between the two platforms.

The query execution time for query 1 on testbed 3 for various states is shown in Figure 20. There is wild fluctuation in Pig's query execution time on the Hadoop platform, but the trend is upward. Hive's query execution time and Jaql's query execution time have slight differences between the two platforms. Figure 21 describes the query execution time for query 2 on testbed 3 for various states. The proposed platform provides faster query execution time in three query languages than the Hadoop platform.

Figure 22 displays the query execution time for query 3 on testbed 3 for various states. The most striking feature is that Pig's query execution time fluctuates on the Hadoop platform from 10 states to 52 states. There are significant differences in query execution time between the two platforms. The query execution time for query 4 on testbed 3 for various states is described in Figure 23. Pig and Jaql have the greater differences in query execution time between the two platforms. The proposed platform provides faster query execution time in three query languages than the Hadoop platform.

Figure 18 Query 3's execution time on testbed 2 (see online version for colours)

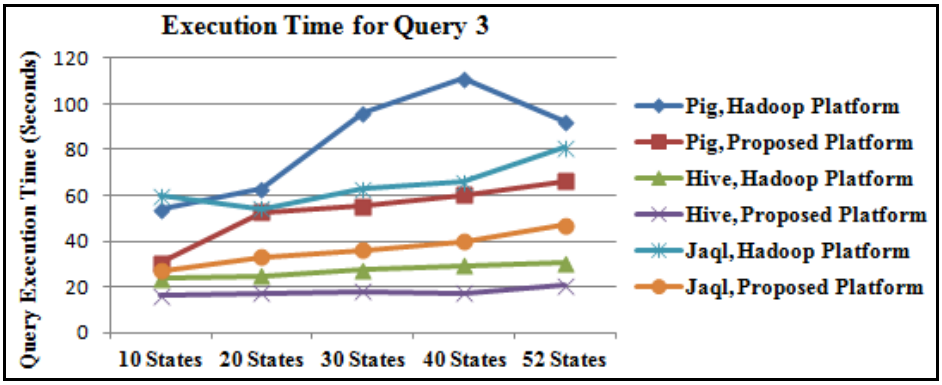


Figure 19 Query 4's execution time on testbed 2 (see online version for colours)

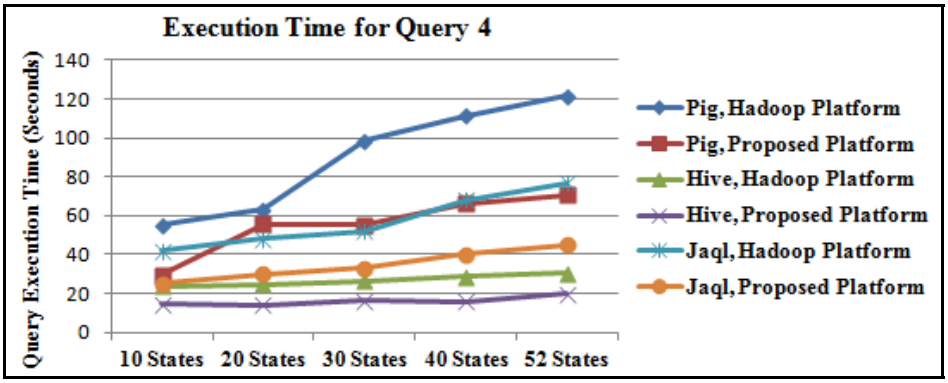


Figure 20 Query 1's execution time on testbed 3 (see online version for colours)

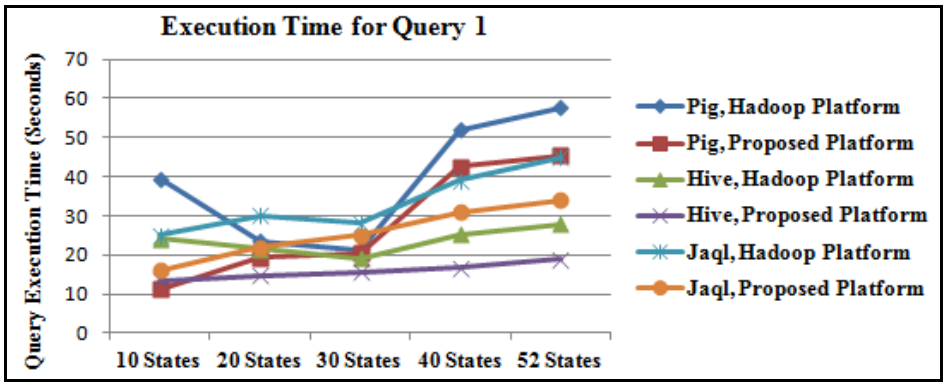


Figure 21 Query 2's execution time on testbed 3 (see online version for colours)

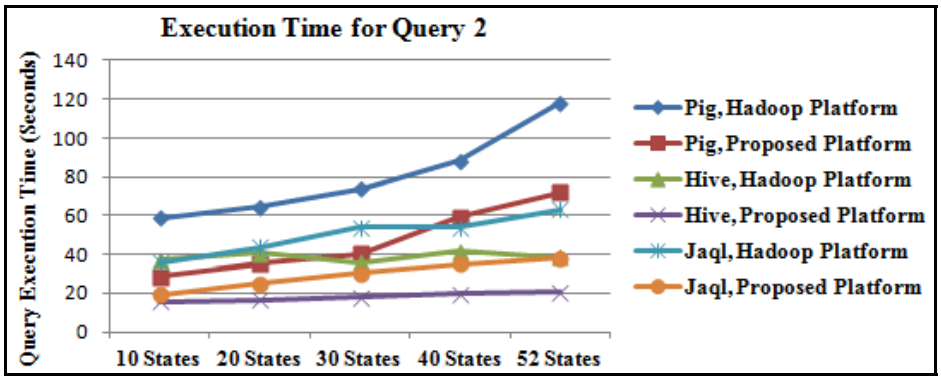


Figure 22 Query 3's execution time on testbed 3 (see online version for colours)

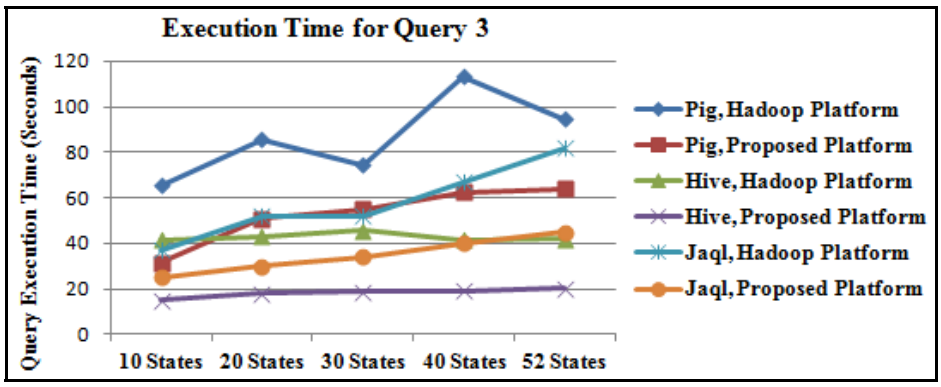


Figure 23 Query 4's execution time on testbed 3 (see online version for colours)

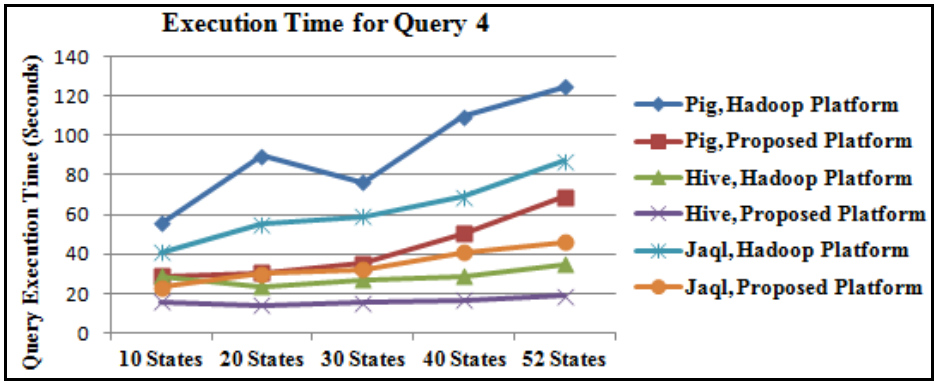


Table 6 Comparison of two big data platforms

	Hadoop platform (MapReduce + Hadoop distributed file system)	*Proposed platform (MapReduce + Gluster file system)*
Diverse job types	√	√
Apache Pig	√	√
Apache Hive	√	√
Jaql	√	√
Better scalability		√
Better performance		√
Fault tolerance	√	√
Faster query language	Hive	Hive

As a result of experiments, we can conclude that Hive provides the fastest query execution time and Pig provides the slowest query execution time on both platforms. However, Pig and Jaql have the greater differences in query execution time between the two platforms. Experimental results prove that three query languages can provide faster query execution time on the proposed platform than the Hadoop platform. Therefore, the proposed big data platform can support large scale data analysis efficiently and effectively. Table 6 describes the comparisons of two big data platforms from various aspects.

5 Vendor products for big data analytics

There are many vendor products to consider for big data analytics. In this paper, we discuss two products – IBM big data platform and Splunk. IBM (Eaton et al., 2011) offers a platform for big data including IBM InfoSphere Biginsights and IBM InfoSphere Streams. IBM InfoSphere Biginsights represents a fast, robust, and easy-to-use platform for analytics on big data at rest. IBM InfoSphere Streams is a powerful analytic computing platform that delivers a platform for analysing data in real time with micro-latency. Splunk (http://www.splunk.com) is a general-purpose search, analysis and reporting engine for time-series text data, typically machine data. It provides an approach to machine data processing on a large scale, based on the MapReduce model. Table 5 describes the comparisons of proposed platform with vendor products.

Table 7 Comparison of proposed platform with vendor products

	IBM big data platform	*Splunk*	*Proposed platform*
Volume, variety, and velocity	Volume and variety (IBM InfoSphere BigInsights)	Volume and variety (splunk hadoop connect)	Volume and variety
	Velocity (IBM InfoSphere streams)	Velocity	
Big data storage	general parallel file system-shared nothing cluster (GPFS-SNC)	Hadoop distributed file system (splunk hadoop connect)	Gluster file system
Processing model	Adaptive MapReduce	Temporal MapReduce	MapReduce
		Spatial MapReduce	
Cloud support	IBM cloud	Splunk storm	×
Query support	Pig, Hive, and Jaql	Splunk search language	Pig, Hive, and Jaql
Scalability	√	√	√
Fault tolerance	√	√	√
Visualisation	BigSheets	Report builder and dashboard editor	×
Enterprise integration	√	√	×
Graphical user interface	BigInsights console	Splunk web UI	×

6 Conclusions

Big data is a growing problem for corporations as a result of sheer data volume along with radical changes in the types of data being stored and analysed, and its characteristics. The main challenges of big data analytics are performance, scalability and fault tolerance. To address these challenges, many vendors have developed big data platforms. In this paper, a big data platform for large scale data analysis by using Hadoop MapReduce framework and GlusterFS over scale-out storage system is proposed. The proposed big data platform solves volume and variety issues of big data and only supports batch processing. Therefore it is necessary to address velocity issue of big data and to support real-time processing. A solution to this can be achieved by adding complex event processing (CEP) techniques to the proposed platform. In addition, the proposed platform does not consider visualisation aspects and this can be solved by using visualisation tools on the proposed platform. Moreover, developing the proposed big data platform requires downloading, configuring, and testing the individual open source projects such as Hadoop, GlusterFS, Pig, Hive and Jaql. The proposed platform should be deployed on Amazon Elastic Compute Cloud (EC2) instances to support cloud computing infrastructure.

References

Agrawal, S. (2011) *The Next Generation of Hadoop Map-Reduce*, Apache Hadoop Summit India.

Beyer, K.S., Ercegovac, V., Gemulla, R., Balmin, A., Eltabakh, M.Y., Kanne, C-C., Özcan, F. and Shekita, E.J. (2011) 'Jaql: a scripting language for large scale semistructured data analysis', *PVLDB*, Vol. 4, No. 12, pp.1272–1283.

Carter, P. (2011) *Big Data Analytics: Future Architectures, Skills and Roadmaps for the CIO*, IDC, September.

Dean, J. and Ghemawat, S. (2008) 'MapReduce: simplified data processing on large clusters', *Communications of the ACM*, January, Vol. 51, No. 1, pp.107–113.

Eaton, C., Deutsch, T., Deroos, D., Lapis, G. and Zikopoulos, P. (2011) *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, McGraw-Hill.

Gluster (2011) *Gluster File System Architecture*, August.

HDFS Architecture Guide [online] http://Hadoop.apache.org/hdfs/docs/current/hdfs design.html.

http://www.macs.hw.ac.uk/~rs46/files/publications/MapReduce-Languages/Comparison_Of_High_Level_Data_Query_Languages.pdf.

Olston, C., Reed, B., Srivastava, U., Kumar, R. and Tomkins, A. (2008) 'Pig Latin: a not-so-foreign language for data processing', in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp.1099–1110, ACM.

Pavlo, A., Paulson, E. and Rasin, A. (2009) 'A Comparison of approaches to large-scale data analysis', in *Proceedings of the 35th SIGMOD International Conference on Management of Data*, ACM.

Russom, P. (2011) *Big Data Analytics*, TDWI Best Practices Report, Fourth Quarter.

Shvachko, K. et al. (2010) 'The Hadoop distributed file system', *Proc. IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST '10*, pp.1–10.

Thusoo, A. et al. (2010) 'Hive – a petabyte scale data warehouse using Hadoop', in *Proceedings of the IEEE 26th International Conference on Data Engineering, ICDE '10*, pp.996–1005.

White, T. (2009) *Hadoop: The Definitive Guide*, O'Reilly and Yahoo! Press.