Developing Scalable Smart Grid Infrastructure to Enable Secure Transmission System Control

EP/K006487/1

UK PI: Prof Gareth Taylor (BU)

China PI: Prof Yong-Hua Song (THU)

Consortium UK Members: Brunel University (BU), Alstom Grid Ltd (AGL), Intel Corporation (IC), National Grid (NG)

Consortium China Members: Tsinghua University (THU), Sichuan University (SU), CEPRI

Document Title	Working paper critically evaluating suitability of approaches investigated in Task 1.1 and 1.2
Document Identifier	
Version	Version 1.0
Work package number	WP2
Task Number(s)	T2.3
Distribution	Public
Reporting consortium member	BU, THU, SU and CEPRI
Internal reviewer & review date	Prof G A Taylor (date)

Document Information

EPSRC Project Number	EP/K006487/1	**China Funding Council**	NSFC
Full Title	Working paper critically evaluating suitability of approaches investigated in Task 1.1 and 1.2		
Project URL	http://www.brunel.ac.uk/sed/ece/research/bips/epsrc-smart-grids		
Document URL	N/A		

Report Number	T2.3	**Title**	Experimental feedback on proposed algorithms and sensors positioning (delivery: M35)
			N/A
Work Package Number	WP2	**Title**	Development and deployment of scalable secure high performance computing tools and infrastructure

Date of Delivery	**Work Plan**	M6	**Actual**	M12
Status	Version 1.0			
Nature	Internal Report			
Dissemination Level	Consortium			
Author(s) (Partners)	G. Taylor (BU), M. Li (BU) M. Khan (BU), Z. Huang (BU)			
Lead Author	**Name**	Prof. G A Taylor	**E-mail**	Gareth.Taylor@brunel.ac.uk
	Partner	BU	**Phone**	+44 (0) 1895 266610
Abstract	Big data analytics such as Hadoop MapReduce is highly applicable to off-line scalable data analytics. The high processing overhead associated with input and output files limits the application of Hadoop with regard to on-line analysis of power system data. Cluster-based Stream Computing technology with Hadoop can be deployed for real-time analysis of high volume of power system data. This report first critically evaluates the performance of Hadoop and then introduces the stream computing platforms for real-time analysis of massive power system data.			
Keywords	Hadoop MapReduce, streams data analytics, power system			

Contents

Abbreviations

API	Application Programming Interface
DAG	Direct Acyclic Graph
DRPC	Distributed Remote Procedure Call
HPC	High Performance Computing
PMU	Phasor Measurement Unit

1. Introduction

Due to the advent of smart grid, advancement in metering infrastructure and rapid deployments of Phasor Measurement Units (PMUs), the power systems have become more computational intensive and generate extremely large amounts of data, often called big data. The concept of big data typically has three characteristics including volume, velocity and variety. The extremely large volume of data is generated in power system due to the advancement in metering devices. The high velocity of data is producing in power grid due to rapid deployments of PMUs which collects data at a typical rate of 50 samples per second. The variety is referred to datasets from different sources which are not necessarily part of the conventional power systems [1]. The timely and effective processing of big data can perform a vital role in improving the power system security, operation and protection. However, the timely and effective processing of large datasets with traditional database technologies are difficult. A solution of processing big data can be found in the field of High Performance Computing (HPC) through parallel computation, such as Hadoop MapReduce.

Hadoop MapReduce has become a major computing platform for big data analytics. It offers a reliable, fault-tolerant, scalable and resilient framework for storing and processing massive dataset in distributed manner. Hadoop system is highly applicable for offline big data analytics due to high throughput. However, a high latency and processing overhead associated with input and output files limit the applications of Hadoop system to offline analytics and make it inefficient for real-time/near real-time analysis on big data.

Real-time systems process data on short time windows i.e. they perform analytics/event predictions on data generated in the last few minutes. In order to achieve better analytics capabilities, the real-time systems often leverage offline processing systems such as Hadoop MapReduce. In this report we provide an introduction to open-source big data analytic systems that can be employed with Hadoop system to process massive amounts of data in real-time/near real-time, in a fault-tolerant and distributed manner.

The remainder of this report is organised as follows. Section 2 critically evaluates Hadoop MapReduce system. Section 3 introduces stream computing platforms that can be employed with Hadoop to process large amounts of data in real-time. Finally, Section 4 concludes the report.

2. Critical Evaluation of Hadoop MapReduce

In this project, we have mainly investigated Hadoop MapReduce system for massive PMUs data storage and analysis. Therefore, in this section we only present the critical performance evaluation of Hadoop system i.e. how Hadoop is vital for offline processing of massive PMUs data and why its performance is limited with regard to on-line analysis of power system data streams.

Hadoop MapReduce has emerged as one of the major distributed computing platform for managing massive amounts of structured and unstructured data in an efficient manner. It improves the efficiency through data distribution and parallel processing. Sensor devices data, social network data, text documents data and other forms of untraditional data types can be efficiently stored and processed in offline mode in Hadoop distributed environment because it has a high throughput and it employs parallel computing techniques. As a result, Hadoop system is vital for storing and processing massive amounts of power system data. We have deployed Hadoop system in order to store and analyse massive PMU data for event detection [2].

Hadoop system is particularly useful for handling two aspects of big data i.e. volume and variety due to high throughput and support of unstructured data [3]. However, the high latency response time and processing overhead associated with input and output files limit the performance of Hadoop with regard to on-line analysis. The power system community needs sophisticated big data analytics platforms that can calculate results incrementally and provide the obtained results immediately at any time. In the next section, we propose a number of stream data processing technologies that are not only optimised for high throughput but also optimised for low-latency applications. These technologies can be deployed with Hadoop to efficiently process massive power system data in near real-time.

3. Stream Computing Platforms for Real Time Big Data Analytics

In this section, we introduce sophisticated cluster-based stream computing platforms that are widely used for processing large amounts of data in real time or near real time. These platforms can be employed on top of Hadoop system. These platforms are open-source and freely available.

3.1 Apache Spark

Apache spark [4] is an open-source, fast and scalable data analytics platform used for big data analytics. It incorporates primitives for in-memory computing and therefore it can be used for real-time large-scale data processing. Spark can be deployed as standalone cluster mode [5], on EC2 (Elastic Cloud Computing) [6], on top of Hadoop YARN or on Apache Mesos platform [7]. It can access data from different sources including HDFS, Cassandra, HBase and S3. Spark platform supports different programming languages and parallel applications can be written quickly using Scala, Java, Python or R language. The Scala programming language is native language of the Spark implementation. Apache spark stack contains a number of libraries such as Spark SQL and DataFrames, MLib, GraphX and Spark Streaming. Spark SQL allows the application developers to query structure data inside the spark program. MLib is scalable machine learning library and contains common machine learning algorithms such as classification, regression, clustering, collaborative filtering, dimensionality reduction as well as lower-level optimization primitives and high-level pipeline. GraphX library is built on the top of Spark framework which uses to automatically parallelise graph iterative computations across multiple nodes. Spark Streaming is a language-integrated Application Programming Interface (API) that enables application developers to easily build scalable and fault-tolerant streaming data processing applications. It can run on Spark standalone mode or on EC2 and uses zookeeper and HDFS for high availability.

Both Spark and Hadoop are based on cluster computing architecture, however, Spark represents a new cluster computing platform with some useful differences. First, Spark is designed for a certain workloads such as a computation that repeatedly process datasets across parallel operations (e.g. machine learning algorithms). Second, Spark introduces the concept of in-memory data processing where data can be cached in memory. This primitive is not only optimises the iterative computations and reduces the latency of access to datasets but also enable the Spark that can be used for real time data analysis [8] [9].

3.2 Apache Storm

Apache Storm is a free and open-source distributed real-time computation framework for processing large volume of continuous streams data. Storm is used for real-time analytics, online machine learning, unbounded data processing, Distributed Remote Procedure Call (DRPC) and ETL (Extract/Transform/Load). It is scalable and fault-tolerant computation system and it can be easily configured and operated. The Apache Storm is extremely fast and it has the ability to process over a million of records per second per node of a computer cluster. The enterprises can harness the high speed of Storm and combine it with other data access applications in Hadoop to efficiently process a full range of workloads from real-time to interactive to batch processing. The five characteristics including fast, scalable, fault-tolerant, reliable and easy to configure and operate makes Storm platform suitable for the real-time analytics [10] [11].

Storm framework is based on a topology in the form of a Directed Acyclic Graph (DAG) with spouts and bolts. The spouts define source of information in computation (e.g. a Twitter API) and the bolts represent manipulations i.e. processes input streams and produces output streams. The manipulation operations include: ran functions, filter, join data, aggregate or communicate with database system. At a high level, the topology structure of Storm is similar to MapReduce. However, the main difference is that Storm framework processes the data in the online mode as opposed to MapReduce framework, which processes the data in the offline mode. Moreover, a Storm topology (similar to job in MapReduce) is running forever until terminate it while a MapReduce job must eventually terminate [12]. A Storm cluster consists of three nodes i.e. Nimbus node, ZooKeeper node and Supervisor node. The Nimbus node is a master node and performs a number of functions, such as uploads computations for execution, distributes code across the cluster, launches workers across the cluster, monitors computation and reallocates worker as required. The ZooKeeper node coordinates the cluster nodes. The Supervisor node communicates with Nimbus node through ZooKeeper, start and stop the worker nodes according to message received from Nimbus [13].

3.3 Apache Samza

Apache Samza [14] is an open-source, scalable, fault-tolerant and distributed framework for processing large amounts of continuous streams in near real-time. It continuously processes data streams and produces results as streams arrive. The Samza system has been developed on top of Apache Kafka [15], [16] and Hadoop YARN and has built-in supports for them. Apache Kafka is open-source and a low-latency distributed messaging system that provides a unified, high-throughput, low-latency platform for processing real-time input streams. It is used for durability i.e. it provides guarantee that all messages are analysed in the order they were received and that no messages are lost. Apache YARN provides fault-tolerance, scalability, security and processor isolation functionality to Samza framework. Whenever, a worker node of Samza cluster is failed the YARN transparently migrates the running tasks to another worker. YARN controls the cluster resources allocations among different users and also controls the resources utilization of individual worker node.

A Samza application is composed of input streams and jobs. An input stream consists of continuous sequences of messages of a similar type or category and it is provided by Apache Kafka. In order to deal with large amounts of stream, the Samza framework divides input streams into multiple partitions. Within each partition, the sequence of message is in order and each message position is identified by its offset. A job is user written code that consumes and processes input streams. In order to achieve a scalable throughput and job parallelism, the Samza job is broken down into smaller executable tasks. The tasks are executed independently parallel across Samza cluster nodes and process data from one or more partitions [17]. The architecture of Samza is similar to Hadoop and is composed of three layers:

1. **Streaming Layer**: In the architecture, the Apache Kafka is working as streaming layer. It is responsible for providing input and partitioned streams that are replicated and durable. In Hadoop this layer is HDFS.
2. **Execution Layer**: YARN is working as an execution layer. It is responsible for resources allocations, scheduling and executing tasks, coordinating tasks across the machine and assigning tasks to worker nodes.
3. **Processing Layer**: Samza API is working as processing layer. The Samza API is responsible for processing the input streams and applying transformations.

4. Conclusion

In this report we have critically evaluated the performance of Hadoop system. Hadoop MapReduce is a suitable platform for offline data processing, however, the high latency response times and processing overhead associated with input and output files limits the performance of Hadoop with regard to on-line analysis. Moreover, in this report we have suggested a number of sophisticated and reliable stream data analytics technologies that can be used with Hadoop for real-time analysis of massive power system data.

References

[1] M. Kezunovic, L. Xie, and S. Grijalva, "The role of big data in improving power system operation and protection," in *Bulk Power System Dynamics and Control - IX Optimization, Security and Control of the Emerging Power Grid (IREP), 2013 IREP Symposium*, 2013, pp. 1–9.

[2] M. Khan, P. M. Ashton, M. Li, G. A. Taylor, I. Pisica, and J. Liu, "Parallel Detrended Fluctuation Analysis for Fast Event Detection on Massive PMU Data," *Smart Grid, IEEE Trans.*, vol. 6, no. 1, pp. 360–368, Jan. 2015.

[3] D. Bhattacharya and M. Mitra, "Analytics on Big Fast Data Using Real Time Stream Data Processing Architecture," *EMC*. [Online]. Available: https://education.emc.com/content/_common/docs/ks_articles/2013KS_Bhattacharya_Mitra-Analytics_on_Big_FAST_Data.pdf. [Accessed: 26-Jan-2016].

[4] "Apache Spark." [Online]. Available: http://spark.apache.org/. [Accessed: 03-Dec-2015].

[5] "Apache Spark Standalone," *Apache Spark*. [Online]. Available: http://spark.apache.org/docs/latest/spark-standalone.html. [Accessed: 17-Jan-2016].

[6] "Running Spark on EC2," *Apache Spark*. [Online]. Available: http://spark.apache.org/docs/latest/ec2-scripts.html. [Accessed: 17-Jan-2016].

[7] "Program against your datacenter like it's a single pool of resources," *Apache Mesos*. [Online]. Available: http://mesos.apache.org/. [Accessed: 17-Jan-2016].

[8] “Spark: alternative for fast data analytics,” *IBM Developer Works*. [Online]. Available: http://www.ibm.com/developerworks/library/os-spark/. [Accessed: 03-Dec-2015].

[9] “Apache Spark speeds up big data decision-making,” *Computer Weekely*. [Online]. Available: http://www.computerweekly.com/feature/Apache-Spark-speeds-up-big-data-decision-making. [Accessed: 03-Dec-2015].

[10] “Apache Storm.” [Online]. Available: http://storm.apache.org/. [Accessed: 03-Dec-2015].

[11] “Apache Storm in Hadoop.” [Online]. Available: http://hortonworks.com/hadoop/storm/. [Accessed: 03-Dec-2015].

[12] “Storm (Event Processor),” *wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Storm_%28event_processor%29. [Accessed: 19-Jan-2016].

[13] “Processing Real-time events with Apache Storm.” [Online]. Available: http://hortonworks.com/hadoop-tutorial/ingesting-processing-real-time-events-apache-storm/. [Accessed: 03-Dec-2015].

[14] “Apache Samza.” [Online]. Available: http://samza.apache.org/. [Accessed: 20-Jan-2016].

[15] “Apache Kafka.” [Online]. Available: http://kafka.apache.org/. [Accessed: 03-Dec-2015].

[16] “Transporting Real-Time Events Stream with Apache Kafka.” [Online]. Available: http://hortonworks.com/hadoop-tutorial/simulating-transporting-realtime-events-stream-apache-kafka/. [Accessed: 02-Dec-2015].

[17] “Apache Samza, LinkedIn’s Framework for Stream Processing.” [Online]. Available: http://thenewstack.io/apache-samza-linkedins-framework-for-stream-processing/. [Accessed: 21-Jan-2016].