Big Data Analytics on PMU Measurements

Mukhtaj Khan[1], Maozhen Li[1], Phillip Ashton[1], Gareth Taylor[1] and Junyong Liu[2]

[1]School of Engineering and Design, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK
[2]School of Electrical Engineering and Information Systems, Sichuan University, 610065, China
Email: Mukhtaj.Khan@brunel.ac.uk

Abstract- Phasor Measurement Units (PMUs) are being rapidly deployed in power grids due to their high sampling rates. PMUs offer a more current and accurate visibility of the power grids than traditional SCADA systems. However, the high sampling rates of PMUs bring in two major challenges that need to be addressed to fully benefit from these PMU measurements. On one hand, any transient events captured in the PMU measurements can negatively impact the performance of steady state analysis. On the other hand, processing the high volumes of PMU data in a timely manner poses another challenge in computation. This paper presents PDFA, a parallel detrended fluctuation analysis approach for fast detection of transient events on massive PMU measurements utilizing a computer cluster. The performance of PDFA is evaluated from the aspects of speedup, scalability and accuracy in comparison with the standalone DFA approach.

Index Terms **-** detrended fluctuation analysis (DFA), *parallel computing, phasor measurement unit* (*PMU), event detection, Amdahl's Law.*

I. INTRODUCTION

Sustainability in power systems is so vital that an enormous effort must be made to avert power system breakdown scenario. The blackout in North East America (August, 14 2003) and previous critical events all over the world are driving the industry to develop more automatic, adaptive and efficient computational tools for power system stability and monitoring analysis. It is becoming highly impossible for traditional supervisory control and data acquisition (SCADA) systems to predict or avert eventualities in a timely manner which may lead to power system catastrophes [1, 2, 9].

One solution to these challenges is the development of the Wide Area Monitoring System (WAMS). WAMS consists of a network of synchronized PMUs [1, 5] which provide a high sampling rate up to 60 samples per second that can be used to enhance the reliability, stability and security of power systems. For this reason PMUs are being rapidly deployed in power systems globally. It is worth noting that the current standard IEEE C37.118 only defines PMU performance under stead-state conditions of power systems, leaving transient performance undefined [7]. Any transient or dynamic events captured by PMU devices can distort the steady-state view around the network and such incidents should be detected and isolated for alternative analysis. As a results, a number of research works have been proposed for detection of PMU events [10, 11, 13, 15, 16, 17].

We have also conducted some research for detection of transient PMU events using detrended fluctuation analysis (DFA) [12]. However, processing the high volumes of PMU data in a timely manner necessitates a high performance and scalable computing infrastructure. For this purpose we have parallelized the work presented in [12] using the MapReduce programming model [3, 4] which has become a de facto standard software technology for big data analytics capitalizing on a cluster of inexpensive commodity computers.

This paper presents the design and implementation of the parallel detrended fluctuation analysis (PDFA) using the MapReduce model. The performance of the PDFA is compared with the sequential DFA in terms of efficiency and accuracy.

The remainder of this paper is organized as follows. Section II presents an overview of the deployment of WAMS in the UK National Grid. Section III briefly introduces the sequential DFA method. Section IV presents the design and implementation of the parallel DFA method using the MapReduce model. Section V evaluates the performance of the PDFA and analyzes its speedup in computation. Section VI concludes the paper and points out some future work.

II. WIDE AREA MONITORING SYSTEM

The WAMS at the UK National Grid is in the early stages of its deployment. PMUs have been installed on the transmission system of England and Wales through upgrades to digital fault recorders and the installation of 4 standalone devices. The majority of the PMUs are configured to report back to a central Phasor Data Concentrator (PDC), installed in the national control centre, for short-term storage and online oscillation analysis. Alarms are sent from this system in real-time to the energy management system to alert the operators when the system is believed to be approaching instability.

The primary role of the system is to monitor any oscillatory behaviour between the generators in Scotland and those of England and Wales. An inter-area mode had been previously identified at around 0.5Hz involving all of the GB system and remains a cause for concern across a major system constraint boundary; two 120km 400kV double circuits that connect the Scottish Network with the North of England, which at present is considered to hinder the transfer of future renewable

This research is supported by the UK EPSRC under grant EP/K006487/1 and National Grid.

generation in Scotland to the main demand centres in England and Wales.

A typical architecture of WAMS is shown in Fig.1 where PMUs collect the data from various sources of power systems. The data sets collected by PMUs are delivered to a local PDC. The data collected by a local PDC is transmitted to a master database called super-PDC. The consolidated data sets collected by super-PDC are fed into analytics applications such as state estimation, stability assessments, data visualization, real-time monitoring and control [6].

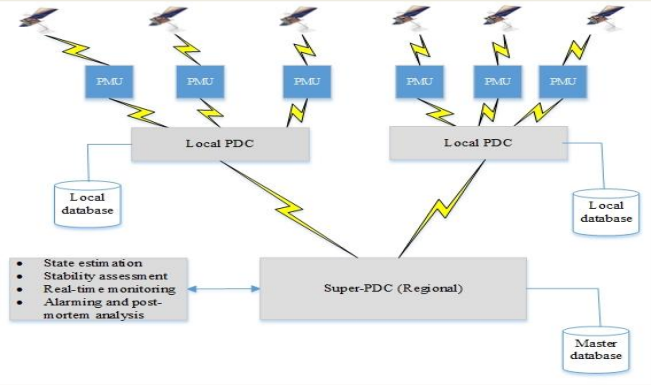

Fig.1. WAMS architecture based on PMUs.

We have installed 2 PMUs at 3-phase 415V AC domestic supply level to measure power system parameters including frequency and voltage phasors. Synchrophasor data measured locally at 50Hz is sent via the Internet from 3 additional UK universities (i.e. Strathclyde, Manchester and Birmingham) to a server in Ljubljana, Slovenia hosted by ELPROS. Fig.2 shows a snapshot of the system. The PMUs are well geographically distributed across the Scottish to England system and give good visibility over the impact of any system events through the Anglo-Scottish connection. The PMUs, connected to voltage only, on the domestic supply, are therefore measuring voltage (magnitude and phase), frequency and rate of change of frequency.

We have installed openPDC software [14] to collect the data from installed PMUs. openPDC is an open system that was designed by the Tennessee Valley Authority (TVA) and administered by Grid Protection Alliance (GPA). The openPDC is used to collect, manage and process real-time synchrophasor measured values from various sources.

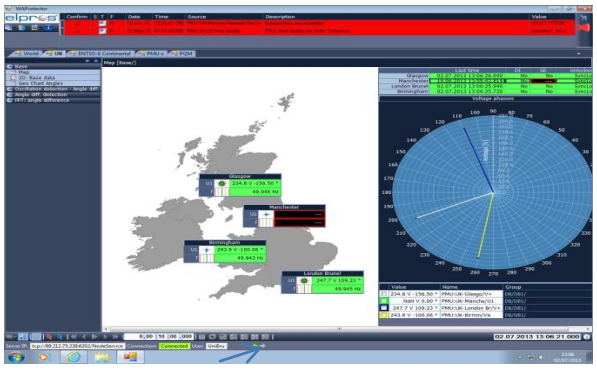

Fig.2. A snapshot of PMU deployment at Brunel.

III. Detrended Fluctuation Analysis

In this section, we give a brief description on the implementation of the sequential DFA method presented in our previous work [12].

The first step is to remove any DC offset from the original signal x using Eq.(1).

$$y(k) = \sum_{i=1}^{k} [x_i - \bar{x}] \qquad (1)$$

where

- k is number of samples.
- x_i is the i^{th} sample of a signal x.
- $\bar{x}$ is the mean value of the overall signal from $i \to k$.
- $y(k)$ is the integrated signal.

In the second step, the integrated signal $y(k)$ is equally divided into blocks with a size of n. The trend of each block $y_n(k)$ is computed using a least square first-order linear approximation.

In the third step, the detrended signal is removed from the integrated signal by subtracting the local trend $y_n(k)$ denoted in Eq. (2).

$$e(k) = y(k) - y_n(k) \qquad (2)$$

In the final step, the root-mean-square fluctuation of (2) is then computed using Eq. (3).

$$F(n) = \sqrt{\frac{1}{n} \sum_{k=1}^{n} [(e(k)]^2} \qquad (3)$$

Here *n* represents the total number of samples in the signal. The value of fluctuation is then used to detect the presence of incident or gross measurement error captured by the PMUs. The greater the value of *F*, the higher the variance in the signal.

IV. The Design of PDFA

In this section we present the design and implementation of the PDFA for event detection on a large amount of PMU data using a MapReduce computer cluster. We first give a brief introduction to the MapReduce programming model, and then present PDFA in detail.

A. MapReduce Programming Model

MapReduce is a parallel and distributed programming model originally developed by Google for processing massive amounts of data in a computer cluster environment [3, 4]. Due to its remarkable features such as fault-tolerance, simplicity and scalability, MapReduce has become a major software technology in support of data intensive applications [19]. MapReduce is a highly scalable model because thousands of commodity computers can be used as an effective platform for parallel and distributing computing. As shown in Fig.3, the MapReduce model divides the computation into *Map* and

Reduce functions where the *Map* function is responsible for splitting input data (files) into small segments while the *Reduce* function collects and combines the results. In the *Map* phase, the job is divided into several *Map* tasks and executed in parallel on cluster nodes. Each *Map* task is produce intermediate result in the form of key/value pairs and store in local storage. In the *Reduce* phase, the *Reduce* function collect the intermediate result and combine the values all together corresponding to single key to produce the final result. The *Map* and *Reduce* functions are executed independently.

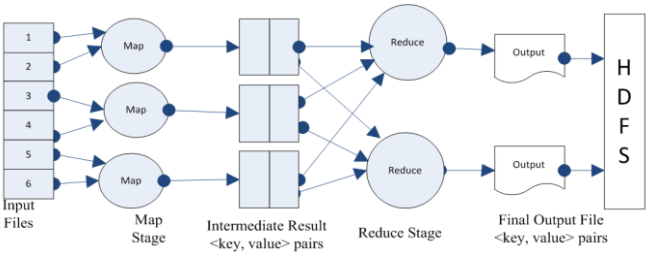

Fig.3. MapReduce model.

B. *MapReduce Implementation with Hadoop*

The MapReduce programming model has been implemented in a number of systems such as Mars [20], Phoenix [21], Dryad [22] and Hadoop [8]. Hadoop is the most popular implementation of MapReduce and has been widely employed by the community due to its open source nature. Hadoop was originally developed by Yahoo to process huge amounts of data (over 300TB) across a cluster of low-cost commodity computers [23]. It is worth noting that Hadoop not only works in computer cluster environments, but also in cloud computing systems such as Amazon EC2 Cloud [18]. Hadoop has its own file system called Hadoop Distributed File System (HDFS) [24]. HDFS is designed to store massive amount of data (terabytes or petabytes) over a large number of computers cluster and provide fast, scalable access to data. HDFS follows a client-server architecture where the name node (NameNode) is the server and the data node (DataNode) is a client. HDFS has only one NameNode and multiple DataNodes. HDFS automatically splits input file into an equal-size blocks (64 MB or 128 MB by default) that are distributed across the DataNodes. Each data block has multiple replicas (3 by default) and is stored on different data nodes. If the cluster network topology has more than one rack then the block replicas will be stored on different rack machines. The purpose of data replication and distributing on different machines is to achieve the reliability and availability of data. The NameNode manages the namespace of the file system and regulates the client access to files. It does not store data itself, but rather maintain metadata file that contain information such as file name, block id, number of replicas, mapping between blocks and DataNodes on which the blocks are stored and location of each block replicas. The DataNodes manages the storage directly attached to each DataNode and executes *Map* and *Reduce* tasks.

The JobTracker runs on the name node (NameNode) and is responsible for dividing user jobs into multiple tasks, scheduling the tasks on the data nodes (DataNode), monitoring the tasks and re-assigning the tasks in case of a failure. The TaskTracker runs on DataNodes receiving the *Map* and *Reduce* tasks from JobTracker and periodically contacts with JobTracker to report the task completion progress and requests for new tasks.

C. *PDFA Implementation*

The parallel detrended fluctuation analysis is implemented using the Hadoop MapReduce framework, through the following steps:

Step 1: A large data set is split into a number of small data blocks such as $B_1, B_2, ...B_n$, where *n* is total number of data blocks.

Step 2: Each data block is assigned to a *Map* task. Map tasks process data blocks in parallel and compute fluctuation values.

Step 3: Through the *Reduce* phase, the fluctuation values are combined and compared with a threshold value to detect any transient events.

PMU data is collected through the openPDC software and buffered in a local hard disk. A software application (data agent) continuously monitors the buffered data. Once the new data file is created in buffered area, the data agent application automatically transfers the file to Hadoop HDFS storage. The HDFS divides the data file into small data blocks, producing three replicas of each data block and distributing over a cluster data nodes. As depicted in Algorithm 1, when a user job is submitted, Hadoop divides the job into multiple tasks. Each *Map* task processes one data block at a time in parallel and computes the fluctuation value (F). The output of each *Map* task is stored in a local disk as an intermediate result (IR). Once the *Map* phase is completed, the *Reduce* phase is initiated to collect the fluctuation values calculated by the *Map* tasks and these values are compared with a threshold value($F>0.2x10^{-3}$) for identification of an event. If at any stage a *Map* task is failed due to software or hardware problems, Hadoop will automatically assign the failed task to another available data node to complete the task.

Algorithm 1: PDFA Implementation.

Input: A PMU data file which will be split into data blocks and distributed in a Hadoop computer cluster
Output: Transient events

Mapper:
// calculates the fluctuation values
// emits the fluctuation values in form values
Map (fluctuation values):
data← $read(datablock)$
window_size ← 50
while(!NOB): // Not-End-Block
PMU buffer ← data[sliding window sample-by-sample]
F ← DFA(PMU buffer, win_size) //call DFA method
Emit(F)
End while
End Map

```
DFA(x, win): // calculates fluctuation
    While(loop through all boxes):
    y(k) ← Σ_{i=1}^{k} [x_i − x̄] // remove DC offset
    End while
    While(loop through all boxes):
      Calculate the trend in each box
    End while
    While(loop through all boxes):
      Calculates the value for the straight line, yn(k) in
      each box
    End while
    While(loop upto Nb):
      e(k) ← y(k) − yn(k)
    End while
//calculates the fluctuation in each sliding window
    F(n) ← sqrt( (1/n) Σ_{k=1}^{n} [(e(k)]^2 )
    return F
End DFA

Reducer:
    // collects the values emitted by mappers
    //emits the event as output
Reduce (values):
    For each val in values
        If (val>threshold)// here val is the fluctuation value
            Emit (event)
        End For
End Reduce
```

V. Evaluation and Experimental Results

We have compared the performance of the PDFA with that of the sequential DFA from the aspects of both efficiency in computation and accuracy. In this section we present the evaluation results.

A. Experiment Setup

The experiments were carried out using a high performance Intel Server machine. The Intel Server has 4 Intel Nehalem-EX processors running at 2.27GHz each with 128GB of physical memory. Each processor has 10 CPU cores with hyper thread technology enabled in each core. As a result, the Intel Server machine has a total of 40 CPU cores and can run up to 80 threads simultaneously. Two disks were configured, with 320GB and 2TB storage capacity respectively for storing a large amount of PMU data. Oracle Virtual Box was installed on the Server for virtualization and 8 Virtual Machines (VMs) were configured into a computer cluster. We installed Ubuntu 12.04 TLS, CDH4.5, Python3.3, numpy and JDK1.6 on the VMs. OpenPDC software was installed on a PC to collect the PMU data.

B. Experimental Results

A number of experiments were carried out to evaluate the efficiency and accuracy of the PDFA method. From Fig.4 we can see that the PDFA outperforms the sequential DFA in computation significantly using 8 VMs. The execution time of the sequential DFA increases with an increasing number of data samples, while the execution time of the PDFA almost keeps consistent.

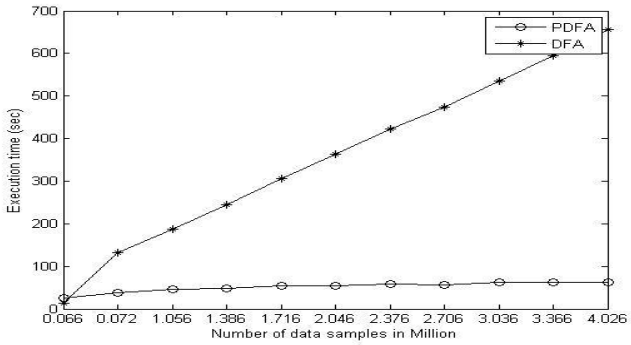

Fig.4.The efficiency of PDFA.

As shown in Fig.5, the accuracy of PDFA is very close to that of the sequential DFA, especially in the cases when the number of data samples is large.

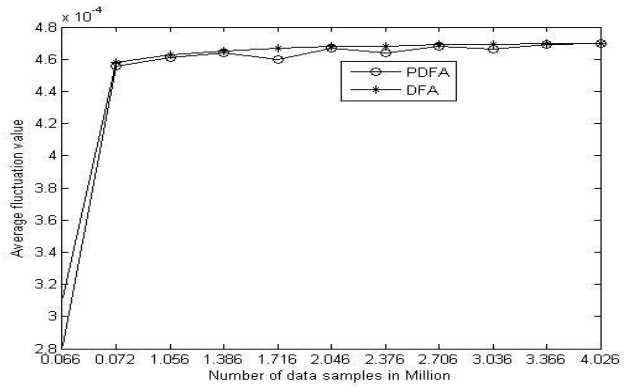

Fig.5. The accuracy of PDFA.

We also evaluated the scalability of the PDFA in terms of a varied number of both VMs and data samples. Fig.6 shows the execution times of the PDFA when processing 3 data sets using a varied number of VMs from 1 to 8. It can be observed that the execution time of the PDFA on each data set decreases with an increasing number of VMs employed. It is worth noting that PDFA performs best in scalability on the largest data set with 32 million data samples.

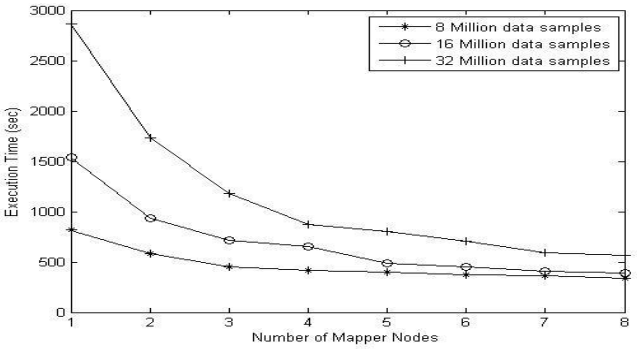

Fig.6. The scalability of PDFA.

Based on the results presented in Fig.8, we plotted Fig.7 which shows the speedup of the PDFA in computation when processing the 3 data sets. Again, the PDFA achieves the best

speedup in computation on the largest data set with 32 million data samples.

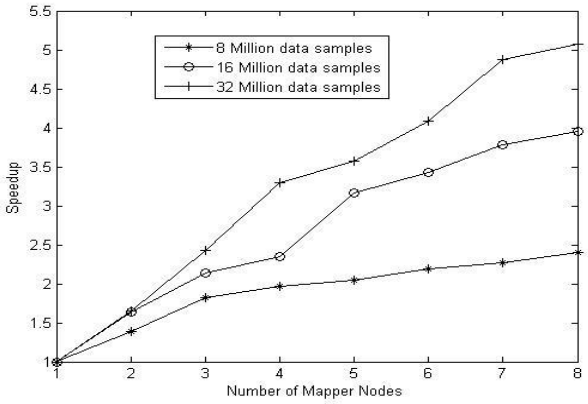

Fig.7. The speedup of PDFA.

VI. CONCLUSION

In this paper, we have presented PDFA, a parallel detrended fluctuation analysis for detection of transient events on a large set of PMU data. PDFA builds on the MapReduce model for data partition and distribution among a cluster of computer nodes. Experimental results have shown the speedup of PDFA in computation while keeping a similar level of accuracy in comparison with the sequential DFA.

The MapReduce Hadoop framework has over 180 configuration parameters. It has been widely recognized that the performance of a Hadoop cluster is largely affected by the varied configurations of these parameters. Following the works presented in [25, 26], we are currently researching the methodologies to optimize the performance of Hadoop based on Starfish [27], a self-tuning system for big data analytics.

REFERENCES

[1]. M. Zima, M. Larsson, P. Korba, C. Rehtanz and G. Andersson , "Design aspect for wide-area monitoring and control system", *Proceedings of the IEEE*, vol. 93, no. 5, pp. 980-996, May 2005.

[2]. P. M. Ashton, G. A. Taylor, M. R. Irving, A. M. Carter, and M. E.Bradley, "Prospective wide area monitoring of the Great Britain transmission systemusing phasor measurement units," in *Proc. IEEE Power Engineering Society General Meeting*, 2012.

[3]. J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large cluster". *Communication of the ACM*, vol. 51, no. 1, pp. 107-133, January 2008.

[4]. R. Lammel, "Google's MapReduce programming model –Revisited", *Science of Computer Programming*, vol.70, no.1, pp. 1- 30, 2008.

[5]. M. Rihan, M. Ahmed and M. S. Beg, "Phasor measurement units in the Indian smart grid", in *Proc.* of *IEEE Conference on Innovative Smart Grid Technologies - India (ISGT India), IEEE PES*, pp. 261-267, 2011.

[6] M. Hurtgen, J. Maun, "Advantages of power system state estimation using phasor measurement units", in *Proc. of 16th Power Systems Computation Conference (PSCC)*, 2008.

[7]. IEEE Power and Energy Society, IEEE Standard C37.118.2-2011, IEEE Standard for Synchrophasor Data Transfer for Power System, December 2011.

[8]. Apache Hadoop, http://hadoop.apache.org [Accessed on 14 August 2013].

[9]. Hauer, J.F.; Bhatt, N.B.; Shah, K.; Kolluri, S., "Performance of "WAMS East" in providing dynamic information for the North East blackout of August 14, 2003," *in Proc. of IEEE Power Engineering Society General Meeting*, pp.1685-1690 Vol.2, 2004.

[10]. J. Z. Tate, "Event detection and visualization based on phasor measurement units for improved situational awareness", PhD Thesis, University of Illinois at Urbana-Champaign, 2008.

[11]. K. Mei, S.M. Rovnyak, C.ong, "Cluster-based dynamics event location using wide-area monitoring phasor measurements", *IEEE Transaction on Power Systems*, vol.23, no. 2, pp. 673-679, 2008.

[12]. P.M. Ashton, G.A. Taylor, M.R. Irving, I.Pisica, A.M. Carter, M.E. Bradely, "A novel application of detrended fluctuation analysis for state estimation using synchrophasor measurements", *IEEE Transaction on Power Systems*, vol.28, no.2, pp.1930-1938, 2013.

[13] J. Interrante, K.S. Aggour, "Applying cluster computing to enable a large-scale smart grid stability monitoring application", in *Proc. of IEEE 14th International Conference on High Performance Computing and Communication*, pp. 328-335, 2012.

[14]. OpenPDC, http://openpdc.codeplex.com [Accessed April 2013].

[15]. P. Trachian, "Machine learning and windowed subsecond event detection on PMU data via Hadoop and the openPDC", in *Proc. of IEEE Power and Energy Society General Meeting*, TVA, USA, pp. 1-5, 2010.

[16]. S.Sohn, A.J. Allen, S.Kulkarni, W. M. Grady and S.Santoso, "Event detection method for PMUs synchnrophasor data", in *Proc. of IEEE Conference Power Electronics and Machines in Wind Application (PEMWA)*, pp. 1-7, 2012.

[17] A.J. Allen, S.Sohn, S.Santoso, and W.M.Grady, "Algorithm for screening PMU data for power system events", in *Proc. of IEEE Int. Conference on Innovative Smart Grid Technologies*, pp.1-6, 2012.

[18]. Amazon Elastic Compute Cloud, Online available: http://aws.amazon.com/ec2

[19]. M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, A. Goldberg "Quincy: fair scheduling for distributed computing Clusters", in *Proc. of the 22nd Symposium on Operating Systems Principles (ACM SIGOPS)*, 2009.

[20]. B.He, W.Fang, Q.Luo, N.K. Govindaraju, T.wang."Mars: MapReduce Framework on Graphics Processors", in *Proc. of ACM 17th Int. Conference on Parallel Architectures and Compilation Techniques*, pp.260-269, 2008.

[21]. C. Ranger, R. Raghuraman, A. Penmetsa, G.Bradski, C.Kozyrakis, "Evaluating MapReduce for multi-core and multiprocessor systems", in *Proc. of the IEEE 13th Int. Symposium on High Performance Computer Architecture*, pp. 13–24, Feb 2007.

[22]. M.Isard, M.Budiu, Y.Yu, A. Birrell, D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks", in Proc. of the European Conference on Computer Systems (EuroSys), pp.59-72, 2007.

[23]. [Yahoo! Launches World's Largest Hadoop Production Application, Available online: http://developer.yahoo.com/blogs/hadoop/posts/2008/02/yahoo-worlds-largest-production-hadoop/, [Accessed on 31 March, 2013].

[24]. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System", *in Proc. of 26th IEEE Symposium on Massive Storage Systems and Technologies (MSST)*, pp.1-10, 2010.

[25]. K. Kambatla, A. Pathak, and H. Pucha, "Towards optimizing Hadoop provisioning in the Cloud", in *Proc. of the Workshop on Hot Topics in Cloud Computing*, held in conjunction with the USENIX Annual Technical Conference, 2009.

[26] S. Babu, "Towards automatic optimization of MapReduce programs", in *Proc. of the 1st ACM Symposium on Cloud Computing (SoCC)*, pp.137-142, 2010.

[27] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: a self-tuning system for big data analytics", in Proc. of 5th Biennial Conference on Innovative Data Systems Research (CIDR), pp.261-272, 2011.