Parallel Detrended Fluctuation Analysis for Fast Event Detection on Massive PMU Data

Mukhtaj Khan, Phillip M. Ashton, *Student Member, IEEE,* Maozhen Li, Gareth A. Taylor, *Senior Member, IEEE,* Ioana Pisica, *Member, IEEE,* and Junyong Liu

***Abstract*—Phasor Measurement Units (PMUs) are being rapidly deployed in power grids due to their high sampling rates and synchronised measurements. The devices high data reporting rates present major computational challenges, in the requirement to process potentially massive volumes of data, in addition to new issues surrounding data storage. Fast algorithms capable of processing massive volumes of data are now required in the field of power systems. This paper presents PDFA, a parallel detrended fluctuation analysis approach for fast event detection on massive volumes of PMU data, taking advantage of a cluster computing platform. The PDFA algorithm is evaluated using data from installed PMUs on the transmission system of Great Britain, from the aspects of speedup, scalability and accuracy. The speedup of the PDFA in computation is initially analysed through Amdahl's Law, a revision to the law is then proposed, suggesting enhancements to its capability to analyse the performance gain in computation when parallelizing data intensive applications in a cluster computing environment.**

***Index Terms*—Amdahl's law, Detrended fluctuation analysis (DFA), event detection, Hadoop, MapReduce, openPDC, parallel computing, PMU, WAMS.**

I. Introduction

SECURITY in power systems is so vital that major efforts must be taken in order to avert potential power system blackout scenarios. The blackout in North East America on the 14^{th} August 2003 and other critical grid events all over the world are driving the industry to develop more automatic, adaptive and efficient computational tools for power system monitoring and stability analysis. It is becoming highly impractical for traditional supervisory control and data acquisition (SCADA) systems to predict or avert eventualities in a timely manner that may lead to power system catastrophes [1]–[3].

One solution to these challenges is presented in the ongoing development of Wide Area Monitoring Systems (WAMS). WAMS comprise a network of synchronized PMUs [1], [4], which provide data at sampling rates typically equivalent to one cycle of the power systems fundamental frequency (50Hz on the GB system). This data, if efficiently managed and processed, can be used to enhance the reliability, stability and security of power systems. For this reason PMUs are being deployed in power systems globally, resulting in rapidly growing volumes of data, posing network operators with new challenges in terms of data storage and timely analysis of the potentially massive datasets.

As a result of the growing complexities in power systems from the increased integration of renewable generation sources and the networks ongoing expansions, it is now vital that data surrounding power system events, such as generation losses, are accurately captured. These events provide the only reliable source of information on the true power system dynamics, providing greater understanding of system inertia, something that is of growing concern on the power system of Great Britain (GB). Timely analysis of these events is critical to understanding the necessary generation response and reserve requirements for a secure network [5]. They also permit the analysis of any trends in the behaviour of the power system under different operating conditions and provide means to validate or improve offline system modelling tools.

In this paper the design and implementation of a parallel detrended fluctuation analysis (PDFA) algorithm, for fast event detection on massive volumes of PMU data, is presented. The approach is implemented in the MapReduce programming model [6], which has become a major software technology in the support of data intensive applications, making use of a cluster of inexpensive commodity computers. The work develops some of the authors' previous studies [7], on the use of detrended fluctuation analysis (DFA) for the detection of power system events on small datasets, more specifically for the detection of instantaneous generation losses, as a requirement for power system inertia estimation [5].

The PDFA is tested and demonstrated in two stages, the first providing details of a laboratory based online setup, using a PMU installed at the domestic supply and the openPDC platform [8] with a localised Data Historian (DH) to collect and store 50Hz resolution data. The second, details the application to the WAMS installed on the transmission system of GB, whereby an offline Data Mining approach is demonstrated. The performance of the PDFA is compared with the original sequential DFA in terms of efficiency and accuracy, using PMU data from the GB WAMS. The speedup of the PDFA in computation is analysed with Amdahls Law, and based on this analysis, a revision to Amdahls Law is then proposed. The revision aims to enhance the capability of analysing

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

Mukhtaj Khan is a PhD student in the School of Engineering and Design, Brunel University, London, UK. (E-mail: mukhtaj.khan@brunel.ac.uk)

Phillip Ashton is with National Grid, Market Operation, Wokingham and Brunel Institute of Power Systems (BIPS), Brunel University, London, UK. (E-mail: phillip.ashton@brunel.ac.uk)

Maozhen Li is with School of Engineering and Design, Brunel University, London, UK. (E-mail:maozhen.li@brunel.ac.uk)

Ioana Pisica and Gareth Taylor are with the Brunel Institute of Power Systems, Brunel University, London, UK. (E-mail: gareth.taylor@brunel.ac.uk)

Junyong Liu is with the School of Electric Engineering and Information Technology, Sichuan University, China. (Email: liujy@scu.edu.cn)

the performance gain in computation when parallelizing data intensive applications in cluster computing environments.

The remainder of this paper is organized as follows. Section II provides an overview of high performance computing and big data analytics. Section III presents the WAMS deployment on the GB system. Section IV details the design and implementation of the PDFA method based on the MapReduce model and Section V evaluates the performance of the PDFA, analysing its speedup in computation. Finally concluding remarks and further work are presented in Section VI.

II. Overview of HPC and Big Data Analytics

With the advent of the smart grid the power system is becoming increasingly complex and computationally intensive. The power systems community faces the challenge of finding suitable methods to solve growing computational issues, for instance, processing massive volumes of PMU data. Such methods can be found in the field of high performance computing (HPC) through parallel processing.

The message passing interface (MPI) is a parallel programming model used to parallelize computation across multiple processors or computers. The MPI model has been used to distribute computation tasks over grid computing nodes [9] and in [10] it was deployed in the HPC environment to parallelize a contingency analysis algorithm. However, the MPI model still requires improvement in areas such as parallel I/O, scalability and topology awareness.

An alternative approach can be found in cluster computing. In [11] a High-Performance Hybrid Computing approach was applied to reduce the execution time of massive contingency analysis algorithms. In this work the algorithm was parallelized using a XMT multithread C/C++ compiler on Gray XMT (multithread HPC computing platform) and conventional cluster computers. In addition, the work in [12] proposed a large scale smart grid stability monitoring application using a conventional cluster of computers to speed up the analysis of PMU measurements. These two separate approaches can increase the speed of program execution by adding more processing nodes however, they rely on centralized management, which can be vulnerable to node failure.

Gao et al. [13] used the parallel computing toolbox within MATLABs Distributed Computer Server (MDCS) to parallelize their contingency analysis algorithm on multiple processors, whilst in [14] a parallel processing method for two monitoring techniques in Prony analysis and an extended complex Kalman filter on multicore systems is explored. Similarly in [15] a genetic algorithm was parallelized. However, these approaches are not resilient and fault-tolerant. The aforementioned approaches can significantly reduce the execution time of large complex computation however, applying these approaches in power system applications is not simply a case of adding more processing units, they require careful design of programs and middleware to make the applications compatible with underlying hardware and software. Furthermore, these approaches (cluster and MPI based) can be scaled by adding more processing nodes. However, they lack the ability to respond to node failures. For example, if any processing node fails as a result of a hardware or software problem, they do not have any remedy to migrate the running tasks to another available node.

Alternatively the work in [14], [16] proposes the cloud computing platform for smart grid data storage and real-time analysis. They parallelize the processing in cloud computing environments to achieve faster computation. To reduce the risk of data accessibility during node failures, data is replicated on multiple machines however, in the instance of node failures no solution is provided to gracefully assign the running computation to another node.

A solution to these issues can be found in the Hadoop MapReduce framework, proposed in a number of areas [17]–[20], offering a reliable, fault-tolerant, scalable and resilient framework for storing and processing massive datasets. In [17] a machine learning technique is applied whilst in [18] simple statistic calculations (maximum, minimum and average) are used to process PMU datasets. However, both of these works leave out the implementation details and provide no evaluation of their methodology or results. The work [19], [20] uses the Hadoop HDFS (Hadoop distributed file system) for storing data and Pig scripting language for simple statistical calculations. The main focus of both works is to compare the performance of the Hadoop distributed processing with the Multi Core system.

III. Wide Area Monitoring GB System

The WAMS running on the GB National Grid is in the early stages of its deployment. Around 40 PMUs have been installed on the transmission system of England and Wales through a series of upgrades to digital fault recorders (DFRs) and the installation of 4 dedicated PMUs, the majority of which are configured to report back to a central Phasor Data Concentrator (PDC) at the national control centre.

The primary role of the system is to monitor for 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; in the two 120km 400kV double circuits that connect the Scottish Network with the North of England. Alarms are sent from this system in real-time to the energy management system (EMS), to alert the network operators when the system is believed to be approaching instability. This constraint is considered to hinder the transfer of future renewable generation in Scotland to the main demand centres in England and Wales.

The PDC is configured to store the 50Hz PMU data at maximum resolution for a rolling one year period, after this time the data is to be archived off at a reduced resolution of 10Hz for upto 10 years. With the amount of PMUs set to increase on the GB system, as additional DFRs are upgraded and new dedicated PMUs are installed [2], this represents a growing challenge in terms of data storage. In addition it is now of vital importance to capture data surrounding system events as they provide the only reliable source of information on the response of the power system, these events need to

be captured at full resolution to assist in inertia estimation methods [5] and continuing validation of the offline network model. Due to the growing volumes of data, importance is therefore placed on timely analysis through fast algorithms and identification of such events.

A. University based WAMS

PMUs have also been deployed at the domestic supply at 4 UK Universities, Brunel, Birmingham, Manchester and Strathclyde. Synchrophasor data, in voltage (magnitude and phase), frequency and rate of change of frequency (RoCoF), is measured locally at 50Hz and sent via the Internet to a server in Ljubljana, Slovenia hosted by ELPROS. This system provides good geographical visibility of the GB transmission system with PMUs well distributed across the network, providing good visibility over the impact of any system events through the Anglo-Scottish connection.

In addition, a laboratory setup exists at Brunel Univerisity where a PMU is configured to communicate data locally to a PDC. The server is running the openPDC software [8], designed by the Tennessee Valley Authority (TVA) and administered by the Grid Protection Alliance (GPA). The openPDC is used to collect, manage and process real-time synchrophasor measured values. This system is an example of a low cost, easy installation alternative to the larger scale WAMS solutions.

IV. The Design of PDFA

A number of research works have been proposed for the detection of system events with PMU data. The work described in [21] details an approach based on finite impulse response (FIR) filtering that is concerned with detecting transient power system events, as a means of determining steady-state information from PMUs to improve situational awareness. Whereas, the work presented in [22] uses a generator clustering approach to determine the source of an event based on detecting the largest initial rotor swing. Other works have dealt with screening volumes of data for significant events, applying algorithms based on Fourier transforms and Yule Walker methods [23], [24].

In contrast the work presented in this paper is focused on determining the exact instant a specific event starts, so that the event can be isolated for additional analysis. Flagging the presence of an event is intended, in the online sense, to act as a trigger for the running of steady-state estimators [7].

The method works by detrending a dataset of PMU frequency measurements on a sample-by-sample sliding window. The window is configured to be 50 samples long, this is to detect for changes over a 1 second period (at 50Hz), looking for a specific loss shape in frequency, following an instantaneous loss in generation. The loss shape typically lasts for 1 second, before primary response services take over and arrest the drop in frequency [5]. A root mean square (RMS) value is then taken of the fluctuation, F for every window, as shown in Equation (1), this value is then compared with a threshold value, predetermined through a number of previous baseline studies, $F = 0.2x10^{-3}$ to detect for the presence of an event.

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

Where n is the size of the window (50 samples), k is the sample number and $e(k)$ is the detrended signal.

Previous works on detrending power system data [25] have focused on removing trends or denoising power system data for the purposes of processing transient oscillations, other work [26] and the original implementation of DFA [27] have focused on the detection of long-range correlations in data series. This is all separate from the work described in this paper. The purpose of detrending the data for this application is to highlight the specific changes in the PMUs measured values as a result of captured transients on the network; the process has the affect of filtering the normal variations in the signal that are predominantly a feature of the high resolution measurements, placing the focus on extreme changes over relatively short time spans.

The PDFA approach is the development of the DFA method to operate efficiently on massive volumes of PMU data, using MapReduce cluster computing.

A. MapReduce Programming Model

MapReduce is a parallel and distributed programming model originally developed by Google for processing massive amounts of data in a cluster computing environment [6], [28]. 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 [29]. MapReduce is a highly scalable model; thousands of commodity computers can be used as an effective platform for parallel and distributed computing.

As shown in Fig.1, the MapReduce model divides computational tasks into Map and Reduce stages. In the Map stage, the computation is divided into several Map tasks to be executed in parallel on cluster computing nodes or virtual machines (VMs). Each Map task (a user-define Map function) processes a block of the input dataset and produces an intermediate result (IR) in the form of key/value pairs, which are then saved in local storage. In the Reduce phase, each Reduce task (a user-define Reduce function) collects the IR and combines the values together corresponding to a single key to produce the final result. It should be noted that the Map and Reduce functions are executed independently.

B. MapReduce Implementation with Hadoop

The MapReduce programming model has been implemented in a number of systems such as Mars [30], Phoenix [31], Dryad [32] and Hadoop [33]. 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

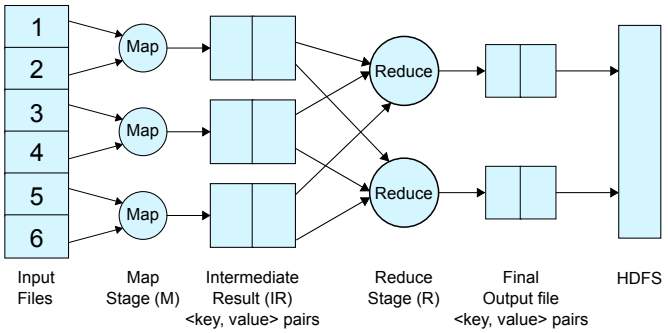

Fig. 1. The MapReduce model.

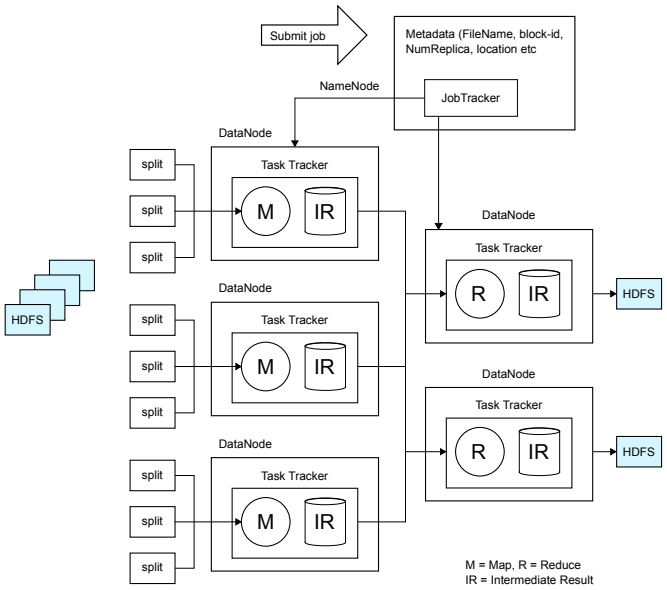

Fig. 2. The Hadoop framework.

computers [34]. It is worth noting that Hadoop not only works in cluster computing environments, but also in cloud computing systems such as the Amazon EC2 Cloud [35].

The architecture of the Hadoop framework, as shown in Fig. 2, comprises its own file system, Hadoop Distributed File System (HDFS) [36]. HDFS is designed to store massive amounts of data (terabytes or petabytes) over a large number of computer clusters and provides fast, scalable access to data. HDFS follows a client-server architecture, where there is a NameNode acting as the server and multiple DataNodes that act as clients. The HDFS has high availability (HA) features by providing the option to configure two NameNodes in the same cluster in the form of active NameNode or passive NameNode (Standby NameNode). This feature is used to reduce the risk of single points of failure. The passive NameNode deals with fast failover in case the active NameNode crashes as a result of software or hardware malfunction [37].

HDFS automatically splits input files into 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), which are 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 distribution on different machines is to maximise reliability and availability of data.

The NameNode manages the namespace of the file system and regulates the clients access to files. It does not store data itself, but rather maintains metadata files that contain information such as file name, block id, number of replicas, mapping between blocks and DataNodes on which the blocks are stored and the location of each block replica. The DataNodes manage the storage directly attached to each DataNode and execute Map and Reduce tasks.

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

Furthermore, the Hadoop MapReduce cluster has over 180 configuration parameters. The system automatically assigns a default value for these parameters if the user does not specify one during a job submission. It has been widely recognized that setting an optimum value for these parameters can have a high impact on the performance of a Hadoop job [38], [39]. Out of all the configuration parameters, precise tuning of the following can have a significant impact on a jobs performance:

- io.sort.mb - Specifies the size of a buffer in memory (MB) used by a map task when sorting a file. The default value is 100MB; however, higher values can improve the performance by reducing the spill to the local disk.
- io.sort.spill.percent - Controls when the system will start the background thread to spill the contents of the memory buffer to local disk. The default value is 0.8 (80%).
- io.sort.factor - Determines the number of spill files to be merged. The default value is 10 and can be up to 30 depending on the RAM of the system.
- mapred.reduce.tasks - This parameter controls the number of Reduce tasks to be set for a job. The default value is 1. However, the user can set more Reduce tasks for a job depending on the structure of the application and requirements of the user.
- dfs.block.size - Controls the size of data block. The default value is 64MB/128MB however, it can be set to a larger size for improved performance.
- dfs.replication - This parameter controls the number of replicas for each data block. The default value is 3. Increasing the number of replicas improves the reliability however at cost of storage space.
- mapred.tasktracker.map.tasks.maximum - This parameter controls the number of Map tasks executed in parallel on a DataNode.
- mapred.tasktracker.reduce.tasks.maximum - This parameter controls the number of Reduce tasks executed in parallel on a DataNode.
- mapred.map.child.java.opts - Specifies the amount of memory for the TaskTracker to use when launching jvm (Java Virtual Machine). The default value is Xmx200m.

C. PDFA Implementation

The original DFA was implemented in MATLAB specifically for the offline application of event detection, focusing on small datasets and the determination of the $t = t0$ moment or exact start time of a specific event.

The PDFA, as described in this paper, is intended for the analysis of massive volumes of PMU data. It was implemented in the Hadoop MapReduce framework using the Python programming language due to its flexibility and open source. The algorithm was implemented as depicted in Fig. 3, through the following 2 staged approaches:

1) Online Implementation: The laboratory based setup at Brunel University comprises a domestic supply connected PMU measuring positive sequence voltage values, frequency and RoCoF. This data is sent through a local area network (LAN) to a openPDC historian. The openPDC software is configured in such a way that when the historian data size reaches 100MB, a new data storage file is created in .d format with a corresponding time-stamp.

A data agent has been created in the Java programming language using a number of Hadoop core libraries and the Java directory watch service package. The application code is encapsulated in the while loop statement to execute continuously, monitoring the historian folder to detect for the presence of new .d files. Once the new file is created in the historian folder, the data agent application automatically moves it to the Hadoop cluster HDFS storage.

It should be noted that the HDFS storage system is not capable of working with the .d file format. At present this file is manually converted to .csv format using the historian playback module within the openPDC software. This process will be automated at a later stage.

2) Offline Implementation - Big Data: Having proven the online data collection side of the system, the following analysis can either be performed as a complement to this process or alternatively it can work in a Data Mining sense where massive datasets are provided directly to the HDFS storage in the .csv file format.

The Hadoop MapReduce supports a number of programming languages such as Java, Python, and C++. Java is the native language of Hadoop and so programs written in Java can be directly executed. Programs written in any other language require application program interfaces (APIs) to execute. For example, programs written in C++ are executed through the Pipes API and programs written in Python will execute through the Streaming API [40].

PDFA has been written in Python in the form of Map and Reduce functions, as Python is open source and unlike Java contains a large amount of the required mathematical functionality. The PDFA is then executed through the Streaming API in the Hadoop MapReduce environment.

When a dataset is moved onto a Hadoop cluster, the HDFS automatically divides it up into blocks B, shown in Fig. 3. The block size is specified in the cluster configuration file (hdfs-site.xml), for instance, if a historian dataset is 16MB and the block size value has been set to 2MB, then the total number of blocks for that dataset will be 8 (16/2 = 8). The total number of Map tasks is equal to the total number of blocks.

When the PDFA program is submitted to the Hadoop framework, the framework automatically divides the PDFA program into a number of Map and Reduce tasks. A block of the PMU dataset is assigned to each Map task and the number of Map tasks executed in parallel to process the dataset depends upon the number of Map slots specified in the cluster configuration file (mapred-site.xml). For the PDFA, one slot was configured on each VM, as a result 8 Map slots were configured in the cluster and so 8 Map tasks were executed in parallel to process the historian dataset. The number of Map slots configured in a VM depends on the processing capacity (physical memory and number of CPU cores) of the VM.

Each Map task processes the assigned data block on a sliding window of 50 samples (as per the DFA algorithm) and calculates the fluctuation value F. The F values are buffered in memory of size 100MB, which can also be set in the configuration file. When the content of the buffer memory reaches a threshold value of 80% (80MB) a background thread is started to spill the contents of the memory buffer to a local disk as an intermediate result (IR). The number of IR files is equal to the number of Reduce tasks.

After completion of the Map phase, the PDFA Reduce tasks are initiated and collect the calculated F vlaues. The number of Reduce tasks is also configurable by the user in the configuration file. The number of Reduce tasks to be executed in parallel depends on the number of Reduce slots configured in the configuration file. For the PDFA, 8 Reduce tasks and 8 Reduce slots were configured, so as to fully utilize all the available Reduce slots. Each Reduce task compares every value of F with the threshold value $F = 0.2x10^{-3}$, any value greater than this threshold is flagged as an event for further analysis.

Most of the conventional cluster-based approaches have issues of reliability and fault-tolerance. The PDFA is implemented in a Hadoop based cluster computing environment, as it offers built-in remarkable features such as high availability, fault-tolerance and scalability. The framework supports multiple replicas of the data blocks and distributes them on different computers/VMs to overcome any fail situations and delays. The cluster can easily be scaled by adding more processing nodes to increase the speedup of computation. During the job execution, if any processing nodes crash due to software or hardware failures, the jobTracker will automatically detect it and assign the running tasks to another available node.

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. The performance was evaluated using 6000 samples of frequency data (2 minutes at 50Hz), provided by National Grid. The data contained a known system event, in the loss of a generator exporting approximately 1000MW. In order to create a Big Data scenario, this dataset was replicated a number of times to provide a relatively large dataset with over 32 million samples.

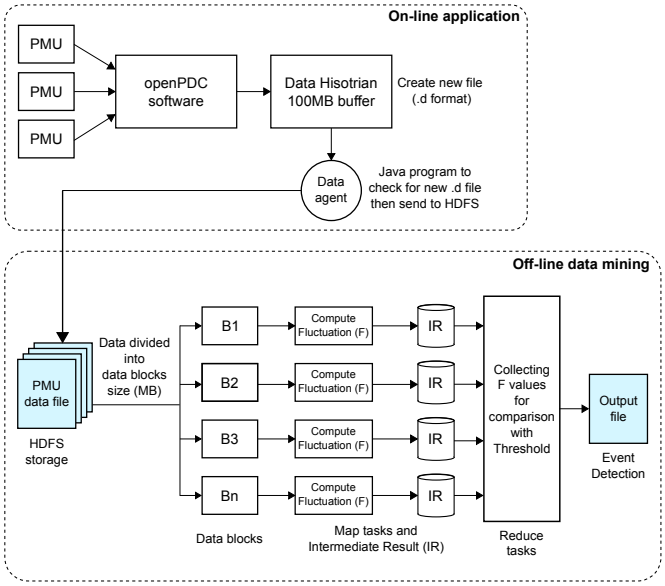

Fig. 3. Architecture of PDFA implementation.

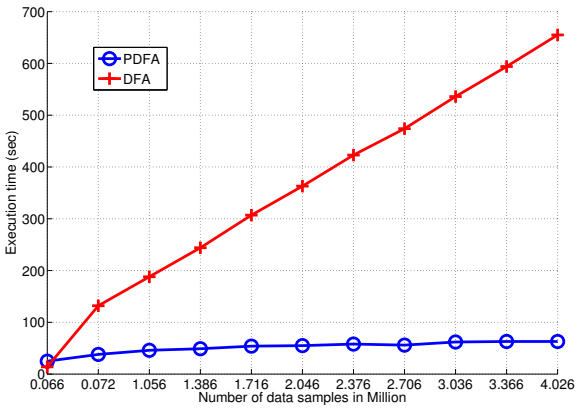

Fig. 4. Analysis of PDFA efficiency.

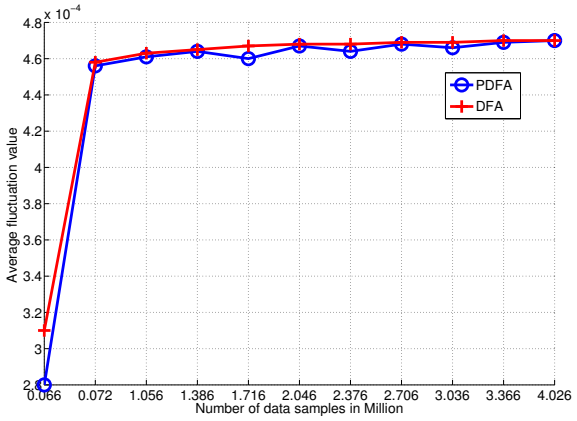

Fig. 5. The relative accuracy of PDFA compared to DFA.

A. Experimental Setup

The experiments were carried out using a high performance Intel Server machine comprising 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. The specific details of the hardware and software implementation are displayed below in Table 1. The analysis of the sequential DFA was carried out on just one of the VMs, whereas the PDFA was run on upto 8 VMs.

TABLE I
EXPERIMENTAL CONFIGURATION OF HADOOP CLUSTER

Hardware	CPU	40 Cores
	Processor	2.27GHz
	Storage	2TB and 320GB
	Connectivity	100Mbps Ethernet LAN
Software	Operating System	Ubuntu 12.04 TLS
	Python	Version 3.3
	JDK	Version 1.6
	Hadoop	CDH 4.5
	Oracle Virtual Box	Version 4.2.8
	openPDC	Version 1.5

B. Results

A number of experiments were carried out to evaluate the efficiency and accuracy of the PDFA method. From Fig. 4 it can been seen 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 remains relatively constant.

The DFA algorithm works on a sliding window, so when comparing the output of the sequential DFA with PDFA it is important to note the possibility of discrepancies in results caused by data partioning due to the way in which the datasets are divided up for parallelization. This does not affect the PDFAs ability to detect events; it just means that the F values could differ slightly from the DFA results. The results of the PDFA are compared with that of the DFA and are displayed in Fig. 5, the relative accuracy of PDFA is very close to that of the sequential DFA, especially in the cases of larger datasets, as the difference converges to zero.

The scalability of the PDFA in terms of a varied number of both VMs and data samples was evaluated. Fig. 6 shows the execution times of the PDFA when processing 3 different sizes of dataset and a varied number of VMs from 1 to 8. The PDFA clearly performs best in scalability on the largest dataset with 32 million data samples. It can be observed that the execution time of the PDFA on each dataset decreases with an increasing number of VMs employed. When processing 8M data samples, 4 VMs generated 2 times speedup, whereas 8 VMs generated 2.5 times of speedup. However, when the number of data samples is increased to 32M, 4 VMs generated 3.3 times of speedup whereas 8 VMs generated 5.4 times of

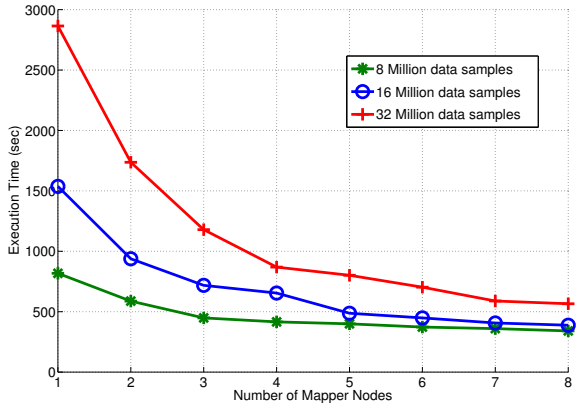

Fig. 6. The Scalability of PDFA, Execution time against number of Mapper nodes (VMs).

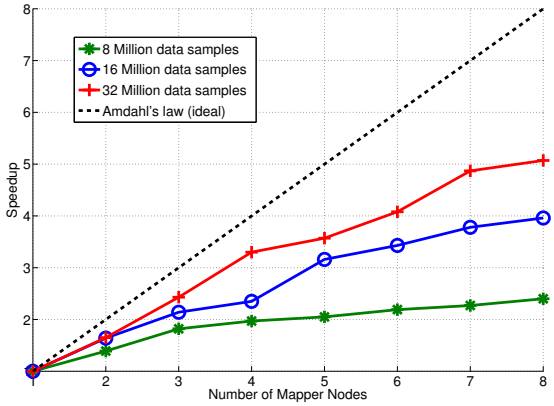

Fig. 7. Speedup analysis of the PDFA algorithm.

speedup. With increasing numbers of data samples, the times of speedup will be increased closer to the number of VMs.

Based on the results presented in Fig. 6, the speedup of the PDFA in terms of computation when processing the 3 different sized datasets was calculated using Equation (2).

$$Speedup = \frac{T_s}{T_N} \tag{2}$$

Where T_s is the execution time of the PDFA on a single VM and T_N represents the execution time of the PDFA on N number of VMs. The results of this calculation are displayed in Fig. 7. Again, the PDFA achieves the best speedup in computation on the largest dataset with 32 million data samples. However, as shown in the figure by the dotted line, the results never achieve that which are to be expected from Amdahl's law [41].

C. Speedup Analysis

When paralleizing a sequential program, the speedup in computation can be calculated using Amdahl's Law [41], defined in Equation (3).

$$Speedup = \frac{1}{(1-P) + \frac{P}{N}} \tag{3}$$

Where P, represents the portion of the sequential programme in percentage that can be parallelized and N represents the number of computers used in the computation.

Theoretically, in the case when a sequential program can be fully parallelized ($P = 1$), as was the case with PDFA, the speedup of the parallelized program should be equal to the number of computers used in the computation N. Therefore, we have:

$$Speedup = \frac{1}{(1-P) + \frac{P}{N}} \leqslant N \tag{4}$$

However, as shown in Fig. 7, the closest speedup to Equation (4) that the PDFA achieved in all the computation scenarios was 3.3 times faster than the sequential DFA when 4 VMs were used in the process. The speedup of the PDFA never achieved N times in a Hadoop cluster with N computers even though the sequential DFA was fully parallelized. This means that Amdahls Law in the form of (4) is not sufficient in calculating the speedup of a parallelized program that is executed in a cluster computing environment. This is because Amdahl's Law in this form does not consider the communication overhead of a user job in cluster computing. For this purpose, a revision to Amdahls Law is proposed in the form of Equation (5), to better reflect the speedup gain when parallelizing a sequential program in cluster computing.

$$Speedup = \frac{1}{(1-P) + \frac{P}{N} + R} < N \tag{5}$$

Where R, represents the ratio of the communication overhead to the computation of a user job, and $R > 0$.

The revised Amdahl's Law (5) better explains the speedup of a parallel program running in cluster computing. The larger a dataset is, the higher overhead in computation will be incurred. As a result, the lower the ratio of communication to computation would be achieved, which leads to a higher speedup in computation. This well explains the speedup of the PDFA in computation when processing the 3 datasets with varied sizes.

To achieve an optimal performance in speedup, the ratio of communication to the communication of a parallel program should be minimized. In the case of Hadoop MapReduce clusters, the size of the segmented data blocks shall be large. On one hand, a large size of data block will generate a small number of tasks that incurs a small overhead in communication. On the other hand, a large size of data block will lead to a high workload in computation. Therefore, a large size of data block will lead to a low communication to computation ratio generating a high speedup.

To evaluate how the size of a data block affects the computational performance of PDFA, the algorithm was run on a dataset of 352MB using 8 VMs with varied sizes of data blocks ranging from 2MB to 32MB. From Fig. 8 it can

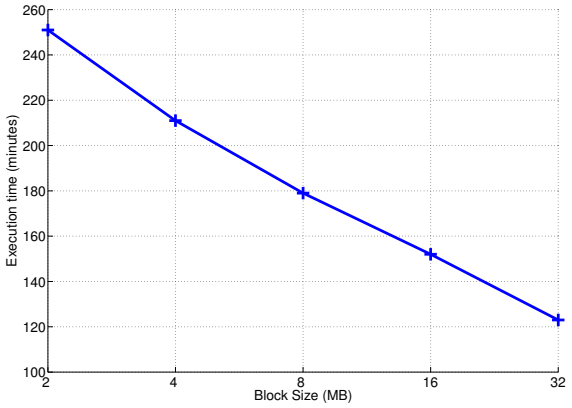

Fig. 8. Computational overhead of PDFA against data block size.

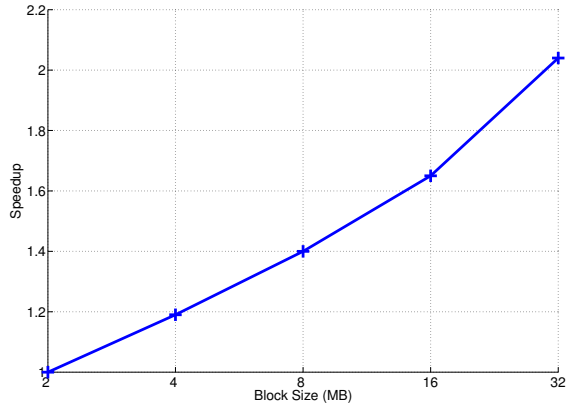

Fig. 9. The speedup of PDFA against data block size.

be observed that the execution time of PDFA decreases with an increasing size of data block. The speedup of PDFA in computation goes up with an increasing size of data block, as shown in Fig. 9. It can be seen that PDFA is 2.04 times faster in computation using 32MB data blocks than when using 2MB data blocks, thus confirming a greater improvement in performance with larger datasets.

VI. Conclusions

In this paper the PDFA approach, for the detection of transient events on massive PMU datasets was presented, in the form of a laboratory based setup for online data collection and analysis, as well as an offline approach in the context of data mining. PDFA builds on the MapReduce model for data partitioning and distribution amongst a cluster of computer nodes. The experimental results have shown the speedup of PDFA in computation whilst maintaining relative accuracy in comparison with the sequential DFA. Based on the analysis in the speedup of computation, an improvement to Amdahl's law is proposed, introducing the ratio of communication to computation to enhance its capability to analyse the performance gain in computation when parallelizing data intensive applications in a cluster computing environment. Further work is proposed to investigate the methodologies to automatically optimize the configuration settings of Hadoop MapReduce parameters. This will further improve the performance of the PDFA algorithm.

References

[1] M. Zima, M. Larson, 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 system using phasor measurement units," in *Proc. IEEE Power Engineering Society General Meeting*, July 2012.

[3] J. F. Hauer, N. B. Bhatt, K. Shah, and S. Kolluri, "Performance of "wams east" in providing dynamic information for the north east blackout of august 14, 2003," in *Proc. IEEE Power Engineering Society General Meeting*, July 2004, pp. 1685–1690.

[4] 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)*, 2011.

[5] P. M. Ashton, G. A. Taylor, A. Carter, and W. Hung, "Application of phasor measurement units to estimate power system intertial frequency response," in *Proc. of IEEE Power Engineering Society General Meeting*, July 2013.

[6] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large cluster," *Communication of the ACM*, vol. 51, no. 1, pp. 107–133, Jan 2008.

[7] P. M. Ashton, G. A. Taylor, M. R. Irving, I. Pisica, A. Carter, and M. E. Bradley, "Novel application of detrended fluctuation analysis for state estimation using synchrophasor measurements," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1930–1938, May 2013.

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

[9] W. Xingzhi, Y. Zhen, and L. Li, "A grid computing based approach for the power system dynamic security assessment," *Journal of Computer and Electrical Engineering, Elsevier*, vol. 36, no. 3, pp. 553–564, May 2010.

[10] G. A. Ezhilarasi and K. S. Swarup, "Parallel contingency analysis in a high performance computing environment," in *Proc. International Conference Power System (ICPS)*, July 2009, pp. 1–6.

[11] I. Gorton, Z. Huang, Y. Chen, B. Kalahar, S. Jin, D. Chavarra-Miranda, and J. Baxter, "A high-performance hybrid computing approach to massive contingency analysis in the power grid," in *Proc. the 2009 Fifth IEEE International Conference on e-Science, IEEE Computer Society*, July 2009, pp. 277–283.

[12] J. Interrante and 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*, 2012.

[13] W. Gao and X. Chen, "Distributed generation placement design and contingency analysis with parallel computing technology," *Journal of Computer and Electrical Engineering, Elsevier*, vol. 4, no. 4, pp. 347–354, April 2009.

[14] J. C. H. Peng, A. Mead, and N. K. C. Nair, "Exploring parallel processing for wide area measurement data applications," in *Power and Energy Society General Meeting, 2011 IEEE*, July 2011, pp. 1–8.

[15] L. Wang and C. Singh, "Multi-deme parallel genetic algorithm in reliability analysis of composite power systems," in *Proc. of the 2009 IEEE Bucharest PowerTech Conference, Bucharest Romania*, July 2009.

[16] K. Maheshwari, M. Lim, L. Wang, K. Birman, and R. Renesse, "Toward a reliable, secure and fault tolerant smart grid state estimation in the cloud," in *Proc. of the IEEE PES Innovative Smart Grid Technologies (ISGT)*, Feb 2013, pp. 1–6.

[17] 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*, 2010.

[18] M. Edwards, A. Rambani, Y. Zhu, and M. T. Musavi, "Design of hadoop-based framework for analytics of large sychnrophasor datasets," *Procedia Computer Science, Complex Adaptive Systems, Elsevier*, vol. 12, pp. 254–258, 2012.

[19] H. Mass, H. K. Camak, F. Bach, and U. G. Kuhnapfel, "One year high rate low voltage recording device, method and results," in *Proc. of the IEEE International Workshop on Applied Measurements for Power Systems (AMPS)*, Sept 2013, pp. 68–72.

[20] F. Bach, H. K. Cakmak, H. Mass, and U. Kuehnapfel, "Power grid time series data analysis with pig on a hadoop cluster compared to multi core system," in *Proc. of IEEE 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Feb 2013, pp. 208–212.

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

[22] K. Mei, S. M. Rovnyak, and C. Ong, "Design aspect for wide-area monitoring and control system," *IEEE Transaction on Power Systems*, vol. 23, no. 2, pp. 673–679, May 2008.

[23] 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)*, 2012.

[24] A. J. Allen, S. Sohn, S. Santoso, and W. M. Grady, "Algorithm for screening pmu data for power system events," in *IEEE Int. Conference on Innovative Smart Grid Technologies*, 2012.

[25] A. R. Messina, V. Vittal, G. T. Heydt, and T. J. Browne, "Nonstationary approaches to trend identification and denoising of measured power system oscillations," *IEEE Transaction on Power Systems*, vol. 24, no. 4, pp. 1798–1807, Nov 2009.

[26] A. Bashan, R. Bartsch, J. W. Kantelhardt, and S. Havlin, "Comparison of detrending methods for fluctuation analysis," *Physica A 387, Elsevier*, pp. 5080–5090, April 2008.

[27] C.-K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger, "Mosaic organization of DNA nucleotides," *Physical Review E*, vol. 49, no. 2, pp. 1685–1689, Feb. 1994.

[28] R. Lammel, "Googles mapreduce programming model revisited," *Science of Computer Programming*, vol. 70, no. 1, pp. 1–30, 2008.

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

[30] B. He, W. Fang, Q. Luo, G. N. K., and T. Wang, "Mars: Mapreduce framework on graphics processors," in *Proc. of ACM 17th Int. Conference on Parallel Architectures and Compilation Techniques*, 2008.

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

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

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

[34] Yahoo!, *Yahoo! Launches Worlds Largest Hadoop Production Application*. http://developer.yahoo.com/blogs/hadoop, [Accessed April 2013].

[35] *Amazon Elastic Computer Cloud*. http://aws.amazon.com/ec2, [Accessed April 2013].

[36] 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)*, 2010.

[37] *An Introduction to HDFS High Availability*. [Available online] http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/latest/CDH4-High-Availability-Guide/cdh4hag_topic_2_1.html, Accessed March 2014.

[38] 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.

[39] S. Babu, "Towards automatic optimization of mapreduce programs," in *Proc. of the 1st ACM Symposium on Cloud Computing (SoCC)*, 2010.

[40] *Hadoop Streaming*. http://hadoop.apache.org/docs/r1.2.1/streaming.html, April 2013.

[41] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc. of AFIPS Spring Joint Computer Conference*, 1967.

Mukhtaj Khan received his MSc in Mobile Computer System from Staffordshire University, UK in 2006. He is currently a PhD student in the School of Engineering and Design at Brunel University, UK. The PhD study is sponsor by Abdul Wali Khan University Mardan, Pakistan. His research interests are focused on high performance computing for big data analysis.

Phillip M. Ashton received his MEng degree from the University of Portsmouth in 2006. He is currently studying an Engineering Doctorate, EngD with Brunel University, UK based in industry with the electricity transmission system operator, National Grid. His research interests are focused around exploiting the use of phasor measurement units for enhanced operation and control of the GB transmission system.

Maozhen Li received the PhD from Institute of Software, Chinese Academy of Sciences in 1997. He was a post-doctoral scholar in the School of Computer Science and Informatics, Cardiff University, UK in 1999-2002. He is currently a Professor in the School of Engineering and Design at Brunel University, UK. His research interests are in the areas of high performance computing (grid and cloud computing) for big data analysis and intelligent systems. He is on the Editorial Boards of Computing and Informatics journal and journal of Cloud Computing: Advances, Systems and Applications. He has over 100 research publications in these areas. He is a Fellow of the British Computer Society.

Gareth. A. Taylor received his BSc degree from the University of London, UK in 1987 and MSc and PhD from the University of Greenwich, UK in 1992 and 1997, respectively. He was the National Grid UK post-doctoral scholar at Brunel University, UK from 2000-2003. He is currently a Professor and Director within the Brunel Institute of Power Systems, Brunel University, UK. His research interests include smart grids, wide area monitoring of power systems and network optimization.

Ioana Piscia received degrees from the University Politehnica of Bucharest and the Academy of Economic Studies from Bucharest, Romania. She is a Research Fellow within the Brunel Institute of Power Systems at Brunel University, London, U.K. She is currently working on the EPSRC funded project ADEPT (Advanced Dynamic Energy Pricing and Tariffs) and her research interests include artificial intelligence techniques, communications and SCADA systems in electrical power engineering.

Junyong Liu is a Professor in the School of Electrical Engineering and Information Technology at Sichuan University, China. His research interest is mainly focused on the areas of intelligent dispatch and operation theory for smart grids, WAMS integration and PMU applications, and power system vulnerability assessment. He has led more than 90 research projects funded by National Natural Science Foundation of China and a large number of electric power grid companies in China. He has over 150 journal and conference papers. He was a Distinguished Visiting Fellow of Brunel University, UK funded by the Royal Academy of Engineering.