Leveraging MapReduce and Synchrophasors for Real-Time Anomaly Detection in the Smart Grid

Suzanne J. Matthews, *Member, IEEE,* and Aaron St. Leger, *Senior Member, IEEE,*

Abstract—The rapid detection of anomalous behavior in SCADA systems such as the U.S. power grid is critical for system resiliency and operator response in cases of power fluctuations due to hazardous weather conditions or other events. Phasor measurement units are time synchronized devices that provide accurate synchrophasor measurements in power grids. The rapid deployment of PMUs enable improved real-time situational awareness to grid operators through wide area measurement systems. However, the quantity and rate of measurements obtained from PMUs is significantly higher than traditional devices, and continues to grow as more are deployed. Efficient algorithms for processing large-scale PMU data and notifying operators of anomalies is critical for real-time system monitoring. In this paper, we propose a novel, two-step anomaly detection approach that processes raw PMU data using the MapReduce paradigm. We implement our approach on a multicore system to process a dataset derived from real PMUs containing $4,500$ PMUs (~ 18 million measurements). Our experimental results indicate the proposed approach detects constraint and temporal anomalies in under three seconds on 8 cores. Our work demonstrates the applicability of MapReduce for designing anomaly detection algorithms for the smart grid, and motivates the creation of novel MapReduce approaches for other SCADA applications.

Index Terms—MapReduce, Wide Area Monitoring Systems, Anomaly Detection, ICS/SCADA, multicore, Phasor Measurement Units

1 Introduction

SMART grid technology is being introduced to improve reliability, resiliency, and efficiency of the power grid. This can be very beneficial under extreme circumstances (i.e. large weather events) in enabling operators or automated controllers to respond quickly to events. Many online and offline smart grid applications are currently being researched and developed [1].

Offline examples include baselining power system performance, event analysis, and model development. Online examples include Wide Area Monitoring Systems (WAMS), Wide Area Control Systems (WACS), State Estimation (SE), and dynamic line ratings. Some online applications are performed in real-time (WAMS) while others are not, or cannot be, performed in real-time (SE). Real-time analysis must be faster than power grid operator control actions. We define real-time as being less than five seconds, which falls into the definition of real-time in existing literature [2]. Rapid anomaly detection, through PMU data analysis, in WAMS and notification to power grid operators are most beneficial in real-time. While some of these applications currently exist in power grids, the advent of smart grid technology and wider deployment of PMUs allow for enhancement of existing applications and the development of new applications as discussed in [3], [4], [5].

Modern situational awareness tools can be improved through the use of real-time synchrophasor data [1]. However, the integration of large numbers of PMUs into a smart grid causes a glut of synchrophasor data that must be processed quickly to ensure real-time response to anomalies. In this paper, we present novel MapReduce algorithms that leverages the data fidelity and accuracy of PMU data to provide robust, two-step, anomaly detection. The first step processes the data for constraint violations. The second step performs temporal analysis to detect anomalous dynamic behavior within the grid.

We test our approaches on a dataset of over 18 million measurements derived from real PMUs on a $1000:1$ scale emulation power grid. Our algorithms are capable of detecting all anomalies in the dataset in under three seconds. Alarms are output, indicating each detected anomaly along with information on what was flagged as anomalous (e.g. frequency temporal anomaly at a specific PMU). Further analysis can then investigate the anomaly (e.g. low frequency, rate of change of frequency, frequency oscillation and damping rate, etc.). The primary contribution of this work is a novel, computationally efficient, anomaly detection methodology integrated with the MapReduce paradigm that is suitable for real-time applications with thousands of PMUs. The performance of our work shows significant improvement compared to prior work and can directly benefit system operators by providing a solution to the expected PMU big data issue in WAMS. We believe our work can help developers create scalable algorithms for detecting anomalies for other SCADA systems as well.

The rest of the paper is organized as follows. Section 2 provides background information on WAMS and MapReduce. Section 3 outlines our approach for anomaly detection. Data collection and results are in Sections 4 and 5 respectively, and we conclude in Section 6.

• *S.J. Matthews and A. St. Leger are with the Department of Electrical Engineering and Computer Science, United States Military Academy, West Point, NY, 10996.*
E-mail: [suzanne.matthews, aaron.stleger]@usma.edu

2 Background

This section provides background information on Wide Area Monitoring Systems (WAMS) augmented with PMUs and related work in this area. We also discuss the MapReduce paradigm, including its advantages, common implementations, and the Phoenix++ MapReduce framework that was used to implement the algorithms described in this work.

2.1 Wide Area Monitoring with PMUs

Analyzing telemetry data to detect abnormal conditions (i.e. anomalies or alarm events) is a key tool for providing situational awareness to power system operators. Traditional SCADA systems include tools for performing this analysis. One drawback to traditional methods and sensors (which do not incorporate PMUs) is that measurements are acquired on the order of seconds and are not time-synchronized. This limits how rapidly situational awareness tools can perform analysis and notify system operators of an alarm event. Additionally, the data rate limits the ability to detect temporal anomalies that occur between data samples (e.g. low frequency oscillations as discussed in [6]).

PMUs are time synchronized devices that provide synchrophasor measurements for use in power grids. This technology provides more accurate and frequent measurements (up to 60 per second) as compared to traditional transducers integrated into SCADA systems. The rapid deployment of PMUs improves real-time situational awareness to power grid operators. According to [4], there were fewer than 500 PMUs installed on North America's transmission grid in 2009; none were being used for real-time situational awareness. Today, nearly $2,000$ PMUs provide real-time grid data to control room operators and engineering applications [4]. The number of deployed PMUs is expected to continue to increase at a rapid rate.

While the availability of PMU data provides opportunity for improving grid operation, the amount of raw data provided is increasing at a rapid rate as PMUs are deployed. As a result, fast and efficient algorithms for processing PMU data and notifying operators of anomalous conditions is integral for real-time system monitoring tools with large amounts of PMUs.

There is much research into integrating synchrophasor data with situational awareness tools, and some tools are available for use today. For example, the Real Time Dynamics Monitoring System (RTDMS) has the capability of integrating PMU data [7] for constraint violations and model based analysis. According to RTDMS literature at the time of this writing, the largest implementation of PMUs is through a Californio ISO project with 85 PMUs. A commercially available package e-terravision [8] has similar capability for PMU integration. However, neither of these tools have reported being applied to an implementation containing a large number of PMUs (hundreds to thousands within a single operating area), or provided specific details on their anomaly detection approaches.

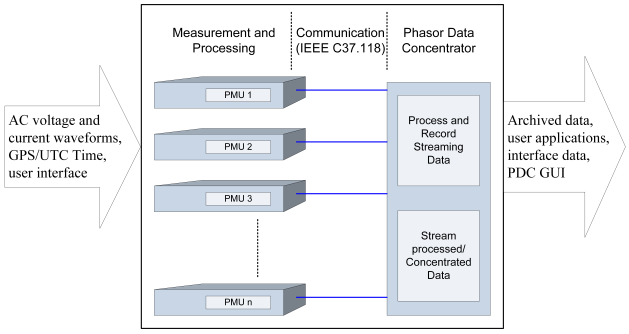


Fig. 1. Wide Area Monitoring System Concept.

The concept of a WAMS is shown in Figure 1. A number of PMUs are installed into a system for monitoring the power grid. Data from these PMUs are send to a Phasor Data Concentrator (PDC) which aggregates and time aligns the data, stores data, and provides an output data stream. This data could go directly to applications, or to other data aggregators (e.g. Super PDCs). A realistic scenario for a large deployment of PMUs is a large number of PDCs networked and working in tandem to provide a PMU data stream to WAM applications. PMUs can provide data at rates up to 60 Hz and provides excellent fidelity and accuracy for anomaly detection.

2.2 Related Work

There has been some, albeit limited, research explored into parallel computing on multicore architectures as applied to the smart grid. Peng *et. al.* use OpenMP [9] to parallelize power oscillation monitoring using small, simulated data sets (less than 1000 samples) [10]. Giuntoli *et. al.* use OpenMP to assess power grid reliability with Monte Carlo Methods on a simulated power grid implemented on a multicore system [11]. Jin *et. al.* design a multicore approach for contingency analysis [12], which is a model-based analysis tool that would be applied after anomaly detection and state estimation are performed.

To the best of our knowledge, we are the first researchers to propose MapReduce algorithms for real-time anomaly detection on the smart grid. Other researchers have used Hadoop MapReduce for analyzing synchrophasor data, but not in a real-time context. For example, researchers at TVA [13] used Hadoop MapReduce [14] for historical analysis of 25 terabytes of synchrophasor data over four years. For real-time analysis, Hadoop is a suboptimal choice owing to the amount of data required for analysis to overcome the overhead of writing to the distributed file system. For example, Edwards *et. al.* [15] note that their serial approach is faster than their parallel Hadoop approach, owing to the small size of the data. Bach *et. al.* [16] performed analysis on Electronic Data Recorder (EDR) data, but note that their Hadoop application is only superior to their serial

application after the data size grows past 6.2 GB. While Khan *et. al.* [17] use Hadoop to parallelize the analysis of sycrophasor data, the fastest their approach ran on their 8-node cluster was approximately 8 minutes [17] on 16 million simulated measurements. In contrast, our approach takes under 3 seconds to process 18 million real measurements on a single multicore server.

Interrante and Aggour [18] present work similar to ours and implement a small signal ocillation detection (SSOD) algorithm and demonstrate how readings from up to 500 PMUs sampling at 60 Hz can be analyzed by a single multicore machine. Our work supports the findings in their paper, and demonstrates how measurements from $4,500$ PMUs can be analyzed by a larger single multicore machine using MapReduce. Zhou *et. al.* [2] discuss real-time anomaly detection (called "triggers" in their work) within the FNET/GridEye framework, to detect anamolies for further processing. However, their results, using Apache Spark framework, focus on the post processing of anomalies and do not describe the performance of their trigger algorithms using a large number of PMUs. Based on discussion in [2], it appears that the number of PMUs used is 200. In comparison, the methodology presented in this work could be leveraged to provide triggering for thousands of PMUs in real-time within the FNET/GridEye framework, or other similar WAMs.

2.3 Our Contributions

To be precise, our unique contributions to this problem space include:

- Investigation of anomaly detection with large scale deployment of real PMUs and 18 million real measurements (big data problem).
- Development of a novel, computationally efficient, model-free anomaly detection scheme for synchrophasor data which is robust in the presence of expected PMU measurement error.
- Description of a MapReduce algorithm and application for real-time anomaly detection in smart grids.

Model-free methods which analyze raw (or minimally processed) PMU data are more suitable for real-time applications as compared to tools that require non-linear power system models (e.g. state estimation [19], fault detection and identification [20]). Some examples of model-free methods include data mining voltage magnitude measurements using a Density-Based Spatial Clustering of Applications with Noise [21], classifications of disturbances and cyber attacks using time synchronized data [22], and applying temporal analysis of PMU data to detect indicators of instability [23].

Most model free methods are based on statistical analysis of PMU data to detect, and/or identify anomalous behavior. An issue of these data mining approaches is managing and processing large amounts of data quickly enough for real-time applications. The approach in this work is a model-free technique that can very rapidly identity constraint based and temporal anomalies from raw PMU data. Our approach can identify the presence of an anomaly, and type of anomaly (e.g. voltage or frequency anomaly, and constraint or temporal anomaly), at a particular location based on PMU location.

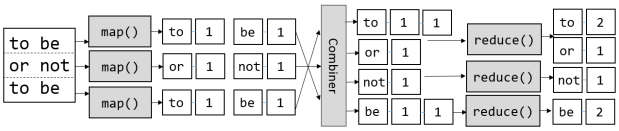


Fig. 2. MapReduce Canonical WordCount Example.

Any anomaly that can be observed by PMU measurement data can be detected by our approach. Examples include rapid changes in system voltages or currents due to an unexpected switching event (physical anomaly or a cyber-attack taking a control action), frequency oscillations (generator trip or load shedding), and gross measurement error/loss of sensor or data from a sensor (physical anomaly with measurement hardware or cyber anomaly with data). The output of this tool can provide subsets of pertinent data to other tools (e.g. model-based tools) for further analysis of the anomaly. The primary limitation of this technique is the requirement of PMU data. If the PMU data stream is disrupted, then anomalies associated with that data cannot be detected.

2.4 MapReduce

We use the MapReduce [24] paradigm to guide the development of our parallel algorithm. MapReduce is designed for efficient processing of large datasets over distributed architectures [24], [14]. MapReduce's attraction lies in its simplicity; to utilize MapReduce, programmers are required to implement only two functions: `map()` and `reduce()`. The underlying scheduling framework automates the parallelization of these tasks on the underlying architecture.

Traditional libraries for parallelism (such as MPI [25] and OpenMP [9]) require developers to explicitly manage synchronization and communication constructs in their code. Ensuring that a parallel program has no errors can require more code than the basic algorithm itself; the extra complexity can lengthen development and deployment times. Programming GPUs is also notoriously difficult, requiring quite a bit of effort to optimize. For applications that are meant to be extensible by non-computational experts, a framework that is relatively easy to program and optimize is essential. Leveraging MapReduce can enable SCADA programmers to develop scalable alarm detection approaches efficiently and in less time than the alternatives.

To illustrate MapReduce, we discuss the canonical application WordCount. In WordCount, the goal is to capture the set of unique words in a collection of documents and their associated frequencies. Figure 2 illustrates how WordCount is parallelized using MapReduce. In this

example, we assume that there are three instances of the `map()` and `reduce()` functions respectively. For simplicity, we use the opening to Hamlet's famous soliloquy as input.

During the Map phase, each instance of the `map()` function (mapper) takes as input a block of text (or a set of files) and produces an intermediate set of $(key, value)$ pairs. In the case of WordCount, the output $(key, value)$ pairs contain a single word and an associated count of 1. For example, mapper 1 takes the block of text "to be" as input, and outputs the $(key, value)$ pairs of (to,1) and (be,1). Notice that all mappers execute independently of each other; this allows them to be executed in parallel.

The $(key, value)$ pairs form the input to the combiner, which aggregates the intermediates to form $(key, list(value))$ pairs. For example, while the intermediate (to,1) was separately emitted by mappers 1 and 3 in our example, they are aggregated by the combiner into the $(key, list(value))$ pair (to,[1,1]). We note that the Map phase and Combiner phase can occur concurrently; however, both phases must complete before the beginning of the Reduce phase.

In the Reduce phase, each instance of the `reduce()` function (reducer) takes its assigned $(key, list(value))$ pairs and performs a reduction operation on each, producing a final $(key, value)$ pair. In the case of WordCount, the reduction operation is addition. For example, reducer 3 sums the values in the $(key, list(value))$ input (be,[1,1]) to form the final $(key, value)$ pair (be, 2). In contrast, reducer 2 is assigned the $(key, list(value))$ pair of (not,[1]); it simply emits (not,1). Note that all reducers run independently and in parallel. At the conclusion of the Reduce phase, we have the set of unique words in our input and their associated counts.

2.5 Phoenix++

Google and other companies use MapReduce to process terascale or petascale data over large distributed clusters. With smaller datasets however (e.g. 1 to 2 GB), large-scale implementations of MapReduce such as Hadoop may be counterproductive, due to the overhead of writing intermediates to the distributed file system. However, this does not mean that MapReduce is inapplicable to smaller datasets; the strength of MapReduce lies in its abstraction, enabling developers to deploy parallel solutions with relative ease. In these cases, it is appropriate to take a closer look at MapReduce implementations for other architectures, such as GPU and multicore.

In the context of our work, we use Phoenix++ [26], an open-source multicore implementation of the MapReduce framework. Phoenix++ was developed by researchers at Stanford University. Critically, intermediates are stored in shared memory rather than being written to disk. The authors demonstrated good speedups with several applications [26], including the aforementioned WordCount example.

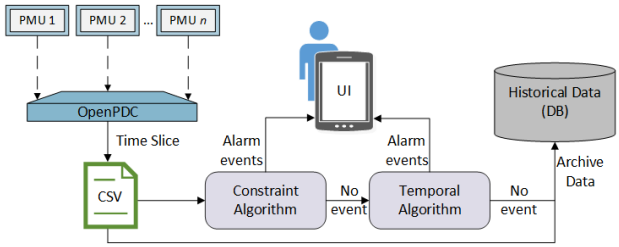


Fig. 3. Anomaly Detection Workflow.

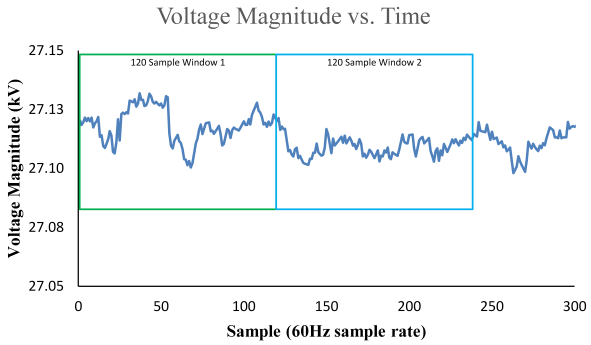


Fig. 4. Example Time Slice of Data Highlighting Two Windows for Voltage Magnitude Measurements.

3 Anomaly Detection

Figure 3 gives an overview of our Anomaly Detection approach. From the PDC, we extract "time slices" of data containing a predefined numbers of windows of measurements to monitor for alerts. For example, we show a sample time slice of five minutes in Figure 4. Highlighted in the figure are two windows of 120 measurements (2 minutes) each. The data within the time slice is organized in a comma separated value (CSV) file. The CSV file serves as input to our constraint detection algorithm (Section 3.4.1). If alarm events are discovered, they are immediately communicated to the grid operator via a user interface (UI) or can be passed on to other near real-time applications as discussed in [2]. If the data passes the checks in constraint detection algorithm, it gets inputted into the temporal detection algorithm (Section 3.4.2). Any alarm events are once again communicated to the grid operator or other applications for further analysis. Regardless of alarm events, data is placed in a database (DB) for archival storage. In the subsections below, we describe the measurements used to detect anomalies, the methodology of anomaly detection, and our anomaly detection algorithms, in greater detail.

3.1 Power System Measurements

The key measurement device used in this work are PMUs. For the purposes of anomaly detection, voltage and current phasors, and frequency measurements are analyzed. Measurement error is inherent in this process.

TABLE 1
Observed Measurement Errors For SEL421 PMU.

Measurement	Typical Error	Full Range Error
Voltage Magnitude	0.091%	0.125%
Current Magnitude	0.356%	0.397%
Frequency	0.70 mHz	5.0 mHz

A measurement, $x_{(n)}$, containing components of the true value being measured, x, and the measurement error, $v_{(n)}$, is expressed in Equation 1:

$$x_{(n)} = x + v_{(n)} \tag{1}$$

Expected deviation from the true value can be quantified via a percent error specification for a measurement device. Measurement error can be modeled with a random variable (normally distributed random variables were used in this work). For smart grid applications using PMU data, measurement error is induced by the transducers, which step down voltage and currents to levels suitable for the PMU, and the PMU itself. The frequency measurement provides a scalar quantity corresponding to the measured frequency at the PMU location. Phasor measurements are complex quantities. A phasor measurement (Equation 2) is in polar coordinates where a phasor measurement of phasor $\mathbf{X}$ is comprised of two quantities: A magnitude, X, and a phase angle θ:

$$\mathbf{X} = |\mathbf{X}| \angle \theta = X \angle \theta \tag{2}$$

IEEE synchrophasor standards C37.118.1-2011 [27] and C37.119.2-2011 [28] were developed to facilitate the development and interoperability of PMUs. These standards defined phasor measurements, required accuracy of the measurements, and the communication protocol for transmitting measurement data, among other items. The accuracy standard for the PMU is established in terms of Total Vector Error (TVE), which takes into account magnitude and phase deviations between the measured phasor and the actual phasor. It is quantified by Equation 3 [29], and must be less than 1% for all of the conditions specified in the C37.118 standard:

$$TVE_{(n)} = \frac{|\mathbf{X}_{MEAS(n)} - \mathbf{X}_{TRUE}|}{|\mathbf{X}_{TRUE}|} \times 100\%, \tag{3}$$

where $\mathbf{X}_{MEAS(n)}$ is the phasor measurement and $\mathbf{X}_{TRUE}$ is the true phasor value. The presence of measurement error must be accounted for in anomaly detection as the true values cannot be known explicitly. While the TVE is specified to be less than 1% by the IEEE standard, in application the error of commercial PMUs can be much less. Pacific Northwest National Laboratory conducted a performance evaluation for SEL 421 PMUs and determined the data depicted in Table 1 [30].

For this work, the full range error from PMUs was assumed (worst case). We also assumed that class 0.5 transducers (maximum error of 0.5%) are used. Lastly, it was assumed that instrumentation errors from transducers would not affect frequency measurement. As a result the expected measurement error is 0.625% for voltage magnitude measurements, 0.897% for current magnitude measurements, and an absolute error of 5.0 mHz for frequency measurements. These error figures and the PMU measurement model were used to create numerous scenarios to develop, test, and benchmark performance of the proposed temporal anomaly detection scheme. In application, PMUs will stream measurement data at a specified rate. More specifically, the synchrophasor standard specifies measurement rates of 10, 12, 15, 20, 30, or 60 Hz for 60 Hz electrical systems. For this work a measurement rate of 60 Hz was used.

3.2 Constraint Anomaly Detection

The constraint anomaly detection scheme in this work monitors PMU data at specified time slices and checks voltage magnitude, current magnitude, and frequency to see if it is within a predetermined acceptable range. This approach works well as a rapid analysis of power system state. Under normal operating conditions, node voltages will be within a well-defined range, frequency will be very close to nominal, and current magnitudes should be below equipment ratings and around an estimated value based on system operation. For example, if a line is in service and transmitting a scheduled amount of power, the measurement should be indicative of this. It is important to note that bad or missing data will also set off the anomaly detector in our approach. Further analysis, e.g. state estimation, may be required to make the determination if the data is reflective of the power system state, or a bad measurement.

3.3 Temporal Anomaly Detection

We define a temporal anomaly as a rapid shift (or oscillation) in the measured value within a given temporal time frame. The temporal anomaly detection scheme developed in this work tracks temporal variation of voltage magnitude, current magnitude, and frequency and determines if deviation from nominal (or expected) operation occurs. Specific items of interest in power systems include change in frequency and oscillations in frequency. Frequency anomalies are indicative of generator or line tripping, load shedding, and potential stability issues. Additionally, rapid changes of voltage magnitudes or current magnitudes are indicative of problems (e.g. voltage stability) or some action being taken within the power grid (e.g. a line tripping will show a step change from a nonzero current magnitude to zero, and a capacitor bank or tap changing transformer operating will show a step change in voltage magnitude). This detection algorithm can also help identify bad data or confirm that commanded control actions have occurred by monitoring the measurements for an expected response without the need for state estimation. In other

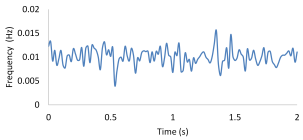
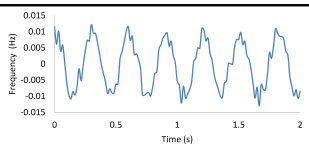
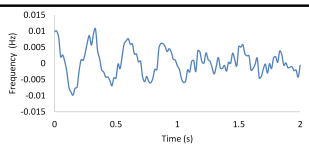
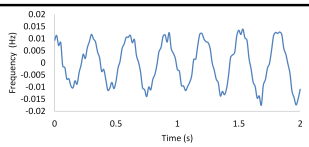
Case	Frequency Deviation from Mean vs. Time	Fano Factor (100 trials)
Fixed Frequency (baseline)		$\|F\| < 0.0004$
3.3 Hz Oscillation (10 mHz magnitude)		$\|F\| < 8.900$ $\|F\| > 0.210$
3.3 Hz Decaying Oscillation (10 mHz magnitude)		$\|F\| < 4.500$ $\|F\| > 0.040$
3.3 Hz Growing Oscillation (10 mHz magnitude)		$\|F\| < 21.40$ $\|F\| > 0.160$

Fig. 5. Temporal Anomaly Detection Examples for Frequency Measurements.

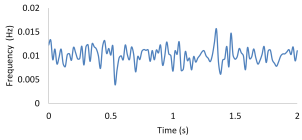
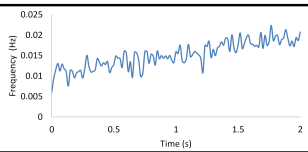
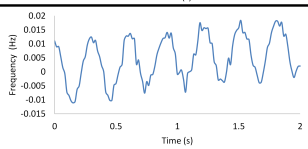
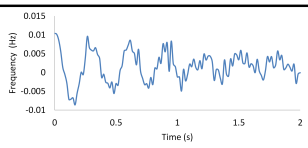
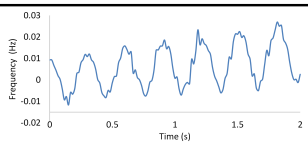
Case	Frequency Deviation from Mean vs. Time	Fano Factor
Fixed Frequency (baseline)		$\|F\| < 0.0004$
Ramp Frequency (5 mHz/s)		$\|F\| < 0.0009$ $\|F\| > 0.0006$
3.3 Hz Oscillation (10 mHz magnitude)		$\|F\| < 0.014$ $\|F\| > 0.011$
3.3 Hz Decaying Oscillation (10 mHz magnitude)		$\|F\| < 0.012$ $\|F\| > 0.007$
3.3 Hz Growing Oscillation (10 mHz magnitude)		$\|F\| < 0.017$ $\|F\| > 0.013$

Fig. 6. Temporal Anomaly Detection Examples for Ramp Frequency Measurements.

words, the focus here is on detecting anomalies in real-time as opposed to near real-time as in SE. As previously mentioned, PMU data is continuously streaming at a specified rate. The concept of temporal anomaly detection used here is to analyze finite sets of this data across a predefined time window. More specifically, the Fano factor is calculated for each set of data (Equation 4):

$$F = \frac{\sigma_W^2}{\mu_W} \tag{4}$$

where σ_W^2 is the variance calculated from data in the window, μ_W is the mean value from the data in the window. The Fano factor is a good indicator of dispersion across a window of data which is used here to indicate an anomaly in the power grid based on PMU data.

The general approach was to develop a baseline of steady-state system operation by calculating the Fano Factor with measurement error. Significant deviation from this baseline is then indicative of an anomaly. A number of cases, representative of aforementioned concerns in power systems, were studied for frequency, voltage magnitude, and current magnitude to validate this approach of anomaly detection. For frequency, the deviation from the mean frequency was used with the magnitude of the Fano factor as it can be negative with these data sets. A number of cases with resulting Fano factors are shown in Figure 5.

The first plot shows a data set of 120 measurements spanning two seconds. The data was generated by adding normally distributed measurement noise, corresponding to PMU measurement model and defined measurement errors, onto a nominal frequency. One hundred trials (windows of data) were developed and analyzed. The Fano Factor magnitude was never above 0.0004 for the baseline. In order to properly detect anomalies, the F-value of an anomalous event must be discernible from this value. One stability concern in power systems is the presence of low frequency oscillations and, if present, the damping of these oscillations. One case in Texas was reported on in [5]. The oscillations observed in that work was used as an example here. Three oscillation cases were studied: no damping (fixed magnitude of oscillation) positive damping (decaying oscillation), and negative damping (growing oscillation). 100 trials were performed for each case with a 3.3 Hz frequency oscillation with a 10 MHz initial amplitude. The Fano factor range observed across 100 trials are shown for each case. The oscillations can clearly be detected for all three cases as the minimum values are three orders of magnitude larger than that of the base case. Additionally, the growing oscillation (which would be of great concern in regards to system stability) exhibited a consistently larger Fano factor than the decaying oscillation. As such, the magnitude of the Fano factor can indicate the severity of the event. A similar set of test cases was performed with a ramp of frequency change in addition to the oscillations. Slight changes of system frequency are expected as generators participating in automatic generation and control respond to control signals. As a result, it is important to discern between an oscillation

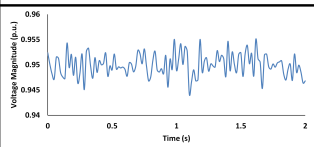
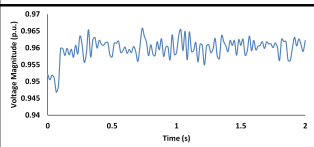
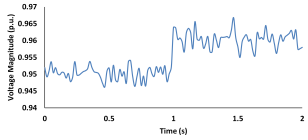
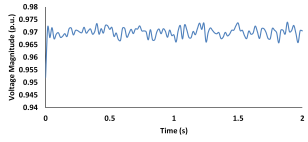
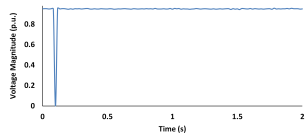
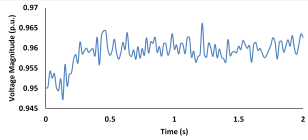
Case	Voltage Magnitude (p.u.) vs. Time	Fano Factor (100 trials)
Fixed Magnitude (baseline)		$F < 5.95\text{x}10^{-6}$
Step Change at 0.1s (0.01 p.u. magnitude)		$F > 7.23\text{x}10^{-6}$ $F < 1.23\text{x}10^{-5}$
Step Change at 1s (0.01 p.u. magnitude)		$F > 2.45\text{x}10^{-5}$ $F < 3.42\text{x}10^{-5}$
Step Change at 1/60s (0.02 p.u. magnitude)		$F > 6.11\text{x}10^{-6}$ $F < 9.64\text{x}10^{-6}$
Impulse Change		$F < 0.080$ $F > 0.078$
Ramp Change (0.01 p.u. magnitude)		$F < 1.73\text{x}10^{-5}$ $F > 9.81\text{x}10^{-6}$

Fig. 7. Temporal Anomaly Detection Examples for Voltage Magnitude Measurements.

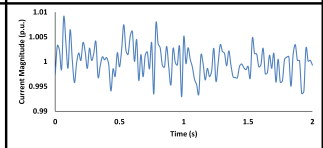
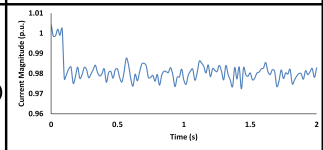
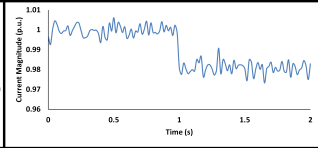
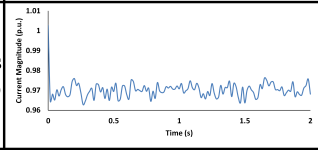
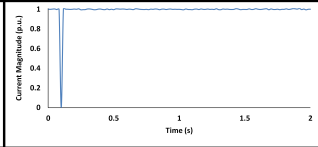
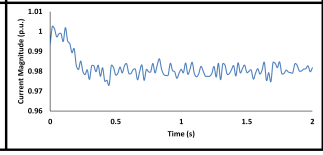
Case	Current Magnitude (p.u.) vs. Time	Fano Factor (100 trials)
Fixed Magnitude (baseline)		$F < 1.19\text{x}10^{-5}$
Step Change at 0.1s (0.02 p.u. magnitude)		$F > 2.22\text{x}10^{-5}$ $F < 3.48\text{x}10^{-5}$
Step Change at 1s (0.02 p.u. magnitude)		$F > 1.00\text{x}10^{-4}$ $F < 1.27\text{x}10^{-4}$
Step Change at 1/60s (0.03 p.u. magnitude)		$F > 1.21\text{x}10^{-5}$ $F < 2.19\text{x}10^{-5}$
Impulse Change		$F < 0.085$ $F > 0.082$
Ramp Change (0.02 p.u. magnitude)		$F < 4.32\text{x}10^{-5}$ $F > 3.05\text{x}10^{-5}$

Fig. 8. Temporal Anomaly Detection Examples for Current Magnitude Measurements.

and expected frequency changes. As shown in Figure 6, all oscillation cases exhibit a Fano factor at least an order of magnitude above the ramp frequency. Additionally, the ramp can be detected from a constant frequency (baseline case). In summary, our proposed approach can clearly delineate between a constant frequency, slow ramp change of frequency, and larger frequency events (oscillations and rapidly changing frequency) with realistic PMU data. The window width will have some effect on the Fano factor values. Experimentation with window widths has shown that the anomalies of interest can be clearly detected assuming a change is observed within the window. However, the actual Fano factor values will change. In application, care must be taken to obtain an appropriate baseline for a given window width.

Test cases for voltage magnitude and current magnitude are shown in Figures 7 and 8 respectively. Three scenarios were studied and compared to a baseline: step changes at various times in the window, an impulse change with one data point far from nominal (this is indicative of a bad or missing measurement), and a ramp change. The focus of these studies was to see how minute of a change could be detected from a baseline. In summary, the impulse is very easy to detect and changes of 0.02 per unit for voltage magnitude and 0.03 for current magnitude can be detected using the Fano factor approach. These marginal changes are far lower than what would be concerning for power grid operators. Additionally, this shows that anomalies can begin to be detected with a single measurement point and location of the anomaly in the window is not imperative to detecting an anomaly. However, the position of the anomaly in the window will have an effect of the Fano factor magnitude. As a result, the Fano factor will not be a reliable indicator of the magnitude of the change/anomaly, but a robust indicator that an anomaly has occurred and with what measurement. Raw data from the window can be then further analyzed to determine the nature of the anomaly.

Note that the actual Fano factor limits will be dependent on the the power system, acceptable limits/operating conditions of the power system, and the window width. Representative examples shown here confirm the capability and applicability of using this approach to detecting anomalies in power systems using PMU measurement data. Application with real PMU data is shown in the results section.

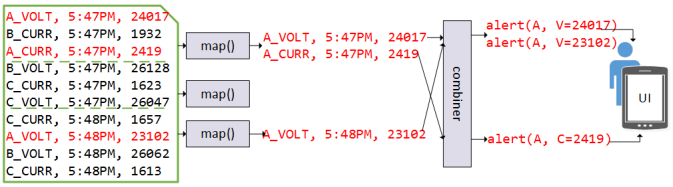


Fig. 9. Constraint Anomaly Detection Algorithm.

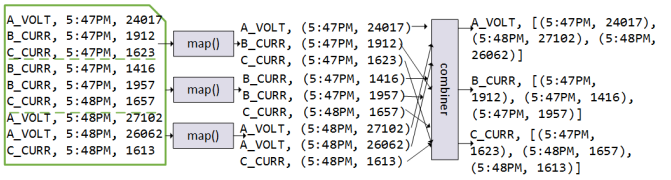


Fig. 10. Map Phase of Temporal Anomaly Detection Algorithm.

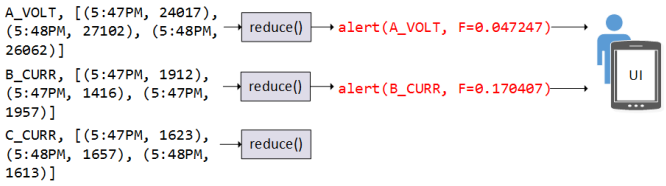


Fig. 11. Reduce Phase of Temporal Anomaly Detection Algorithm.

3.4 Anomaly Detection Algorithms

The input to both algorithms is a CSV file consisting of a time slice of data outputted from the PDC, along with an operator-designated file of constraints. For our $1000:1$ scale-emulation of a system with a nominal 26.3 kV phase voltage, a voltage measurement triggers an alarm event in our constraint algorithm if it is above 27.6 kV or under 25 kV. Current measurements must be below the rated value of 2000 Amperes. Prior to execution of the MapReduce portion of the code, these operator-designated constraints are loaded into a hash table that resides in shared memory.

Each line of the CSV file is composed of of a signal ID, a timestamp, and a raw measurement. The signal ID is in reality a unique hash that represents a particular PMU and type of measurement. For the figures in the paper, we replace the hash with a string (e.g. `A_VOLT`, for the voltage measurement from substation A). Note that the actual collected data and the operator-provided constraint file contain actual hash values.

3.4.1 Constraint Anomaly Detection Algorithm

Figure 9 gives an overview of our MapReduce algorithm for detecting constraint anomalies. Each mapper is assigned a chunk of the input, where chunk boundaries are along lines in the file. For example, in Figure 9, mapper 1 is assigned the first three lines of the file, mapper 2 is assigned the next three lines, and mapper 3 is assigned the last four lines in our example CSV file. Each mapper then processes it assigned input line by line, looking up line's signal ID in the hash table, and checking to see if the associated value falls within the allowable window set by the operator. If it does, the line is silently ignored.

In the case of mapper 2 in Figure 9, none of its values are consider anomalous. Therefore, the mapper emits nothing. In contrast, mapper 1 and mapper 3 have a current and two voltage readings that violates the set of constraints. These are emitted as (signalID, measurement) $(key, value)$ pairs to the combiner. The combiner sorts and aggregates all the anomalies. These are then outputted to the user interface for review by the grid operator. Unlike our next algorithm, there is no reduce phase.

3.4.2 Temporal Anomaly Detection Algorithm

Our MapReduce algorithm for detecting temporal anomalies is more complicated. Note this algorithm only runs if no constraint anomalies are found. Figures 10 and 11 illustrate our MapReduce algorithm for detecting temporal anomalies. In this example, none of the listed values fall outside the allowable range for our system. However, there are some fluctuations that may be cause for concern.

During the Map Phase, each mapper takes its assigned block of the memory-mapped file and emits $(key, value)$ pairs where the key is a signal ID, and the value is tuple consisting of the timestamp and measurement. In our previous algorithm, we only emit $(key, value)$ pairs in case of anomalies. In our temporal approach however, every single measurement (despite not displaying anomalous characteristics by itself) could be part of a larger fluctuation pattern, and therefore must be emitted to the combiner.

The combiner sorts the intermediates into $(key, list(value))$ pairs, aggregating the intermediates by common signal ID. For example, while mappers 1 and 2 in Figure 10 independently process current measurements from Station B, these are all combined together into a single $(key, list(value))$ pair by the combiner. We note that the values in the $list(value)$ portion of each pair outputted by the combiner are sorted by time stamp; this is crucial for our next phase.

Unlike our constraint algorithm, our temporal anomaly detection algorithm has a Reduce phase. During the Reduce phase, each reducer takes the set of $(key, list(value))$ pairs assigned to it and computes the associated Fano Factor (F). While in Figure 11 we calculate the Fano Factor over all the measurements, in reality, we slide a window of 1 second increments over the entire array of values, calculating the Fano Factor for each window. If the values in any window are anomalous, they are emitted to the grid operator. Voltage and Current magnitude measurements are normalized prior to calculating the Fano Factor.

For example, in Figure 11, the calculated Fano Factor for `A_VOLT` is $F = 0.047247$, which is above our allow-

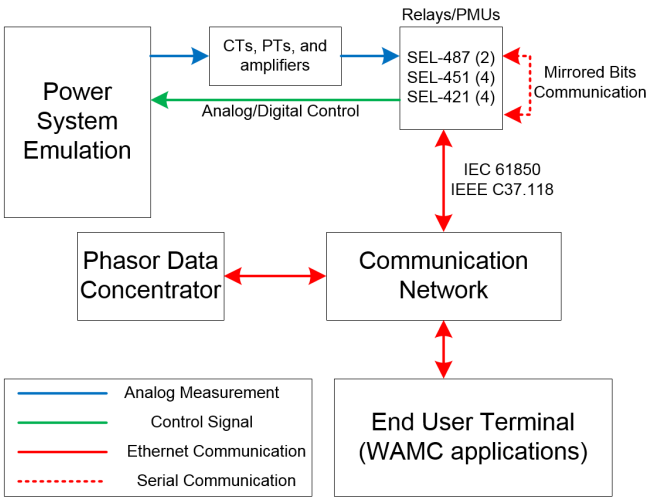


Fig. 12. Block Diagram of Smart Grid Test-Bed.

able threshold of 0.0001 for voltage. Thus, an alarm event is triggered which is communicated to the grid operator. Likewise, `B_CURR` has a Fano Factor of $F = 0.170407$, which is above the allowable F threshold for current (0.15). In contrast, `C_CURR` has a Fano Factor of $F = 0.014142$, which is below our maximum threshold. As a result, no alarm event is triggered for this signal ID.

4 Data Collection

PMU data was required for the development and testing of the proposed approach to anomaly detection. A Smart Grid Test-Bed has been developed at the United States Military Academy for education and research. More specifically, the system was designed for research related to Wide Area Monitoring and Control Systems [31]. Real PMU data was obtained from this test-bed and used for this work.

4.1 Smart Grid Test-Bed

A block diagram of the Smart Grid Test-Bed is shown in Figure 12 [31] and a picture of the test-bed in Figure 13 [31]. The power system modeled, and emulated, in the test-bed is based on a seven bus 46 kV (line voltage) three-phase distribution grid containing nine transmission lines. The system model was developed based on a real world distribution grid in the eastern United States. The power system is emulated on a small scale by scaling voltage and current levels by a factor of 1000. In other words, the line voltage in the system is 46 V and the maximum line currents are 2 Amperes in the test-bed. The test-bed also contains measurement equipment (PMUs and voltage/current transducers), relays for protection and control, and a PDC for automated data acquisition, archiving, and serving data to the end user terminal.

A communication network consisting of physical and simulated components which interconnects the PMUs, PDC, and end user terminal was also developed [32]. Eight of the Schweitzer Engineering Lab (SEL) Relays installed in the test-bed include IEEE C37.118 compliant PMUs which are time synchronized to GPS satellite clocks. This hardware was chosen to provide primary relay protection in addition to PMU measurements. The PMUs feature level 1 compliance to the standard and have two PMU channels (six PTs and six CTs) per PMU. However, only one channel is currently being used on each relay as this provides system observability within the testbed (it should be noted that the anomaly detection in this work does not rely on system observability). These relays are of the same manufacturer, and four of the same model, that were tested to produce the error data shown in in Table 1. The end state of the proposed anomaly detection system is to operate in real-time within this test-bed and integrate with WAMS applications. PMU data was obtained from this test-bed to develop, test, and benchmark this work.

Fig. 13. Smart Grid Test-Bed.

4.2 PMU Data

Each of the eight PMUs were configured to report the following data at a 60 Hz rate: frequency, rate of change of frequency, A phase voltage phasor, A phase current phasor, a time stamp for the data (seconds of century and factional seconds), and time quality indicator. To obtain data for this work, the test-bed was run for various periods of time, under various conditions with and without anomalies, and PMU data was collected and archived in the PDC database.

The objective was to obtain large datasets of real PMU data. For example, the system was run for ~ 1.5 hours and a 1.3 GB dataset with over 18 million measurements was obtained for testing and validating the proposed anomaly detection scheme. The size of this data was representative of a large scale power grid with many PMUs. However, the number of unique signal IDs is not representative of a large scale system as there are only eight PMUs. The raw PMU data was processed to develop a realistic data set for a large scale power grid with many PMUs. More specifically, each PMU in the dataset was segmented into 125 sample blocks (~ 2

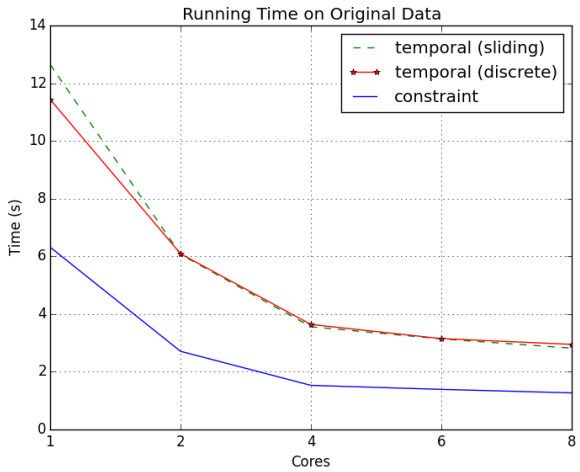


Fig. 14. Running Time of Constraint and Temporal Algorithms on Original Data Set consisting of 8 PMUs.

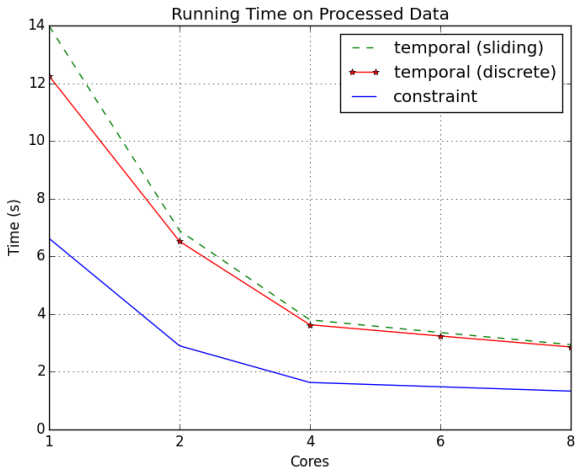


Fig. 15. Running Time of Constraint and Temporal Algorithms on Processed Data Set consisting of $4,500$ PMUs.

minutes of data) and unique signal IDs were synthesized for each block of data. The result was a 1.3 GB dataset with over 18 million measurements where each PMU's data spanned over ~ 2 minutes with signal IDs from $4,500$ PMUs. Note that the PMU measurement data was not altered. The anomaly detection algorithm was conducted with both datasets (original and processed data) to observe the performance for a larger time set of data with fewer PMUs as compared to a small time set of data with a large number of PMUs. The latter being representative of a real-time anomaly detection application in a large power grid with $4,500$ PMUs.

5 Results & Discussion

Our anomaly detection source code is written in C++ and utilizes the Phoenix++ MapReduce library. It is compiled with GCC 4.4.7 and the `-O3` optimization flags. Experiments are run on a Red Hat Enterprise Linux with quad-core Intel i7-3770 at 3.40 GHz, and 32 GB of total RAM. Due to hyper-threading, there are 8 virtual cores. We use the aforementioned dataset collected from our 8 real PMUs from our scale-emulation power grid and the processed dataset with $4,500$ PMUs. We vary the number of cores 1 to 8 in increments of 2. For each dataset and number of cores, the run-time numbers discussed are derived from an average over 10 independent runs.

There are two version of the temporal algorithm. In the first, we use a sliding window of one second (60 measurements) over all the measurements. In the second, we use discrete windows of 1 second each. The running times for both approaches are roughly equivalent, with the sliding window approach being slightly slower than the discrete window approach. For brevity, we highlight only the numbers for sliding window approach in the subsection below. The time required to perform temporal checks is longer than what is required for constraint checks, owing to the presence of a reduce phase.

We also discuss the speedup of our approach as we increase the number of cores. Speedup is calculated as $S_n = \frac{\text{Time(1 core)}}{\text{Time}(n \text{ cores})}$. Ideally, running the algorithm on n cores would yield a speedup close to n.

5.1 Execution Time on PMU Datasets

We measure the time it takes for our algorithm to detect all anomalies in the input dataset, both on the raw dataset derived from our 8-PMU scale emulation power grid, and the processed dataset consisting of $4,500$ PMUs. Both datasets contain 18 million measurements. Our results are summarized in Figures 14 and 15. The x-axis is the number of cores being utilized, while the y-axis indicates the average run time in seconds.

Figure 14 shows the measured execution time of the 8-PMU dataset. On a single core, we detect anomalies in 6.32 seconds on average. On 2 cores, that number decreases to 2.71 seconds, a speedup of 2.33. On 4 cores, it takes 1.53 seconds, a speedup of 4.13. Running our constraint detection algorithm on 8 cores takes 1.27 seconds, yielding our highest speedup of 4.98.

On a single core, it takes our sliding window temporal check approximately 12.65 seconds to complete, vs. 11.44 seconds for our discrete approach. On 2 cores, it takes 6.07 seconds, a speedup of 2.08. Increasing the cores to 4 reduces execution time to 3.56 seconds, yielding a speedup of 3.55. It takes 2.82 seconds on 8 cores, yielding our maximum speedup of 4.49.

Figure 15 shows the measured execution time of the dataset consisting of $4,500$ PMUs. Our constraint algorithm is slightly slower on this dataset, requiring 6.62 seconds on 1 core. The algorithm requires 1.33 seconds on 8 cores, yielding a speedup of 4.98. We see similar performance on our temporal algorithm. The temporal check takes 13.98 seconds on 1 core for the sliding window approach. On 8 cores, this time reduces to 2.94 seconds, for a final speedup of 4.76.

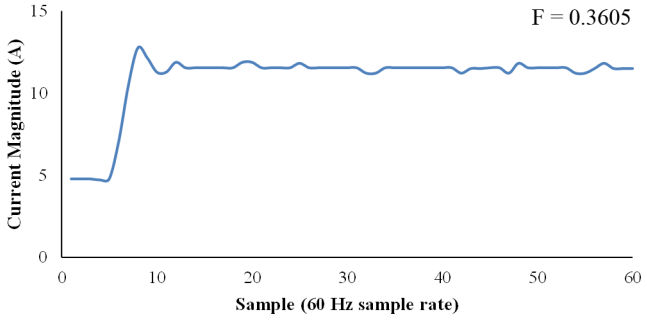


Fig. 16. Example of Detected Current Magnitude Temporal Anomaly.

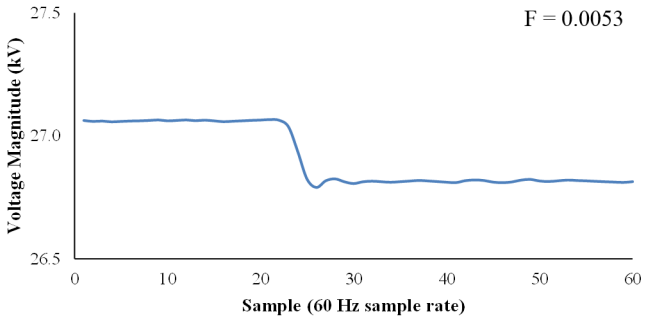


Fig. 17. Example of Detected Voltage Magnitude Temporal Anomaly.

To validate this approach, anomalous incidents were created in the testbed for both the constraint and temporal detection algorithms. The created anomalies were correctly identified with both detection algorithms. For example, a rapid load change was created and correctly detected by the temporal anomaly detection algorithm. The current magnitude data and Fano factor are shown in Figure 16. Anomalous behavior in the local power distribution grid, which is the primary source of power to the testbed, were also observed and flagged by our detection algorithms. For example, a rapid change in voltage magnitude is shown in Figure 17.

5.2 Analysis of Execution Time

Our algorithms can process the dataset and report all detected anomalies in under three seconds on 8 cores. As soon as an anomaly is detected it is outputted to the user; however, the time it takes to report the first anomaly is dependent on the anomaly's location in the file. In this section, we present worst-case asymptotic run time. Let P denote the number of PMUs, and assume that each emits M measurements in a given time slice. Thus, the input dataset consists of $P \times M$ total measurements. On a system with c cores, the asymptotic run-time for our parallel constraint algorithm is therefore $O(\frac{P \times M}{c})$.

Our temporal approach requires longer time. First, $O(\frac{P \times M}{c})$ time is needed in the map phase to process the input into corresponding $(key, value)$ pairs. The combiner (which hashes and sort the pairs), requires $O(M \log M)$ time. Each reducer then receives M measurements corresponding to a single PMU. Assuming a window size of W, there are $\frac{M}{W}$ windows. The Fano factor for each window takes $O(\frac{M}{W})$ to calculate. So, for the discrete windowing approach, the reduce phase takes $O((\frac{M}{W})^2)$ time, while the sliding window takes $O(\frac{M^2}{W})$ time. We note that as the window size gets smaller, the discrete windowing time approaches the sliding window time. Thus, the total asymptotic run-time for the temporal approach is $O(\frac{P \times M}{c} + \frac{P \times \frac{M^2}{W}}{c})$.

6 Conclusions

In this paper we presented a novel, model-free, approach to processing large sets of PMU data to rapidly detect anomalies and improve situational awareness in the smart grid. Our contribution is significant in two important ways. First, we present two novel MapReduce algorithms for detecting constraint and temporal anomalies in real-time. Second, we test our approach on real data collected from a scale emulation of a power grid.

We implement our MapReduce algorithms using Phoenix++, a shared-memory implementation of the MapReduce framework. We test our algorithms on a real dataset of over 18 million PMU measurements, and a processed dataset representative of a scenario with $4,500$ PMUs. In both datasets, our algorithms are able to detect anomalies in under three seconds on 8 cores, making the proposed approach applicable for real-time detection in smart grid applications. For comparison, China has future plans to deploy $2,000$ PMUs in their Smart Grid infrastructure [18]. This transfers the bottleneck of detecting anomalies from the data processing to the latency of collecting data from the PMUs.

The implications of our work are significant for protecting the smart grid. First, our use of a multicore implementation of MapReduce shows that we can leverage this paradigm for real-time anomaly detection on smaller power subsystems; petascale or terascale data is not needed to attain real-time performance with the MapReduce paradigm. Multicore systems are less costly to maintain than clusters, and have lower power and cooling requirements. For large subsystems that produce upwards of terascale data, our algorithms can be implemented in Hadoop, enabling scalability. Future work includes implementing our algorithms in the smart grid test-bed WAMS at the USMA and investigating the performance of our approach on other low-power architectures such as GPUs and microclusters.

Acknowledgments

This work was partially funded by Grant 10800174 from the U.S. Army Research Labs Faculty and Cadet Collaborative Research Program and the Network Science Center. The opinions in the work are solely of the authors, and do not necessarily reflect those of the U.S. Army, U.S. Army Research Labs, the U.S. Military Academy, or the Department of Defense.

REFERENCES

[1] NERC, "Real-time application of synchrophasors for improving reliability," NERC, Princeton, NJ, Tech. Rep. 1, 2010.
[2] D. Zhou, J. Guo, Y. Zhang, J. Chai, H. Liu, Y. Liu, C. Huang, X. Gui, and Y. Liu, "Distributed data analytics platform for wide-area synchrophasor measurement systems," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2397–2405, 2016.
[3] V. Madani, J. Giri, D. Kosterev, D. Novosel, and D. Brancaccio, "Challenging changing landscapes: Implementing synchrophasor technology in grid operations in the wecc region," *IEEE Power and Energy Magazine*, vol. 13, no. 5, pp. 18–28, Sept 2015.
[4] P. Overholt, D. Ortiz, and A. Silverstein, "Synchrophasor technology and the doe: Exciting opportunities lie ahead in development and deployment," *IEEE Power and Energy Magazine*, vol. 13, no. 5, pp. 14–17, Sept 2015.
[5] K. M. Koellner, S. Burks, B. Blevins, S. N. Nuthalapati, S. Rajagopalan, and M. L. Holloway, "Synchrophasors across texas: The deployment of phasor measurement technology in the ercot region," *IEEE Power and Energy Magazine*, vol. 13, no. 5, pp. 36–40, Sept 2015.
[6] D. Yang, W. Tang, C. Rehatanz, and K. Gorner, "A systematic method and its near-real time application to analyze the low frequency oscillation based on wams," in *Power Engineering Conference (AUPEC), 2013 Australasian Universities*, Sept 2013, pp. 1–8.
[7] M. Parashar and J. Mo, "Real time dynamics monitoring system (rtdms): Phasor applications for the control room," in *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*, Jan 2009, pp. 1–11.
[8] General Electric, "e-terravision: Fast and reliable decision making in the control centers," 2015. [Online]. Available: http://www.gegridsolutions.com/alstomenergy/Grid/global/Resources/Documents/NMS/NMS/e-terravision_mark_trans%20100615.pdf
[9] L. Dagum and R. Menon, "Openmp: an industry standard api for shared-memory programming," *IEEE computational science and engineering*, vol. 5, no. 1, pp. 46–55, 1998.
[10] J. C. H. Peng, A. Meads, and N. K. C. Nair, "Parallel computing for smart power oscillation monitoring using synchrophasor measurements," in *TENCON 2010 - 2010 IEEE Region 10 Conference*, Nov 2010, pp. 657–662.
[11] M. Giuntoli, P. Pelacchi, and D. Poli, "Parallel computing of sequential montecarlo techniques for reliable operation of smart grids," in *EUROCON 2015 - International Conference on Computer as a Tool (EUROCON), IEEE*, Sept 2015, pp. 1–6.
[12] S. Jin, Z. Huang, Y. Chen, D. Chavarría-Miranda, J. Feo, and P. C. Wong, "A novel application of parallel betweenness centrality to power grid contingency analysis," in *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*. IEEE, 2010, pp. 1–7.
[13] P. Trachian, "Machine learning and windowed subsecond event detection on pmu data via hadoop and the openpdc," in *IEEE PES General Meeting*. IEEE, 2010, pp. 1–5.
[14] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.
[15] M. Edwards, A. Rambani, Y. Zhu, and M. Musavi, "Design of hadoop-based framework for analytics of large synchrophasor datasets," *Procedia Computer Science*, vol. 12, pp. 254–258, 2012.
[16] F. Bach, H. K. Çakmak, H. Maass, and U. Kuehnapfel, "Power grid time series data analysis with pig on a hadoop cluster compared to multi core systems," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2013, pp. 208–212.
[17] 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," *IEEE Transactions on Smart Grid*, vol. 6, no. 1, pp. 360–368, 2015.
[18] J. Interrante and K. S. Aggour, "Applying cluster computing to enable a large-scale smart grid stability monitoring application," in *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*. IEEE, 2012, pp. 328–335.
[19] J. Zhao, G. Zhang, K. Das, G. N. Korres, N. M. Manousakis, A. K. Sinha, and Z. He, "Power system real-time monitoring by using pmu-based robust state estimation method," *IEEE Transactions on Smart Grid*, vol. 7, no. 1, pp. 300–309, Jan 2016.
[20] H. Jiang, X. Dai, D. W. Gao, J. J. Zhang, Y. Zhang, and E. Muljadi, "Spatial-temporal synchrophasor data characterization and analytics in smart grid fault detection, identification, and impact causal analysis," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2525–2536, Sept 2016.
[21] A. Mukherjee, R. Vallakati, V. Lachenaud, and P. Ranganathan, "Using phasor data for visualization and data mining in smart-grid applications," in *2015 IEEE First International Conference on DC Microgrids (ICDCM)*, June 2015, pp. 13–18.
[22] S. Pan, T. Morris, and U. Adhikari, "Classification of disturbances and cyber-attacks in power systems using heterogeneous time-synchronized data," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 650–662, June 2015.
[23] E. Cotilla-Sanchez, P. D. H. Hines, and C. M. Danforth, "Predicting critical transitions from time series synchrophasor data," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1832–1840, Dec 2012.
[24] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[25] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the mpi message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.
[26] J. Talbot, R. M. Yoo, and C. Kozyrakis, "Phoenix++: modular mapreduce for shared-memory systems," in *Proceedings of the second international workshop on MapReduce and its applications*. ACM, 2011, pp. 9–16.
[27] "IEEE standard for synchrophasor measurements for power systems," *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)*, pp. 1–61, Dec 2011.
[28] "IEEE standard for synchrophasor data transfer for power systems," *IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005)*, pp. 1–53, Dec 2011.
[29] K. E. Martin, D. Hamai, M. G. Adamiak, S. Anderson, M. Begovic, G. Benmouyal, G. Brunello, J. Burger, J. Y. Cai, B. Dickerson, V. Gharpure, B. Kennedy, D. Karlsson, A. G. Phadke, J. Salj, V. Skendzic, J. Sperr, Y. Song, C. Huntley, B. Kasztenny, and E. Price, "Exploring the IEEE standard C37.118 x2013;2005 synchrophasors for power systems," *IEEE Transactions on Power Delivery*, vol. 23, no. 4, pp. 1805–1811, Oct 2008.
[30] Z. Huang, T. Faris, K. Martin, J. Hauer, C. Bonebrake, and J. Shaw, "Laboratory performance evaluation report of sel 421 phasor measurement unit," Pacific Northwest National Laboratory, Richland, WA., Tech. Rep. PNNL-16852, 2007.
[31] A. St. Leger, J. Spruce, T. Banwell, and M. Collins, "Smart grid testbed for wide-area monitoring and control systems," in *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T D)*, May 2016, pp. 1–5.
[32] C. Holt, A. Kong, A. St. Leger, and D. Bennett, "Communications network emulation for smart grid test-bed," in *2016 IEEE Power and Energy Society General Meeting*. IEEE, 2016, pp. 1–5.

Suzanne J. Matthews is an Assistant Professor of Computer Science in the Department of Electrical Engineering & Computer Science at the United States Military Academy, West Point. She received her Ph.D. degree in Computer Science from Texas A&M University, and her M.S. and B.S. degrees in Computer Science from Rensselaer Polytechnic Institute.

Aaron St. Leger is an Associate Professor of Electrical Engineering in the Department of Electrical Engineering & Computer Science at the United States Military Academy, West Point. He received his B.S., M.S., and Ph.D degrees in Electrical Engineering from Drexel University, Philadelphia, PA, in 2003, 2005 and 2008, respectively.