Proceedings of The National Conference
On Undergraduate Research (NCUR) 2016
University of North Carolina Asheville
Asheville, North Carolina
April 7-9, 2016

Using MapReduce to Detect Anomalies in the Real-Time Smart Grid

Colin Hwang, Dylan Chamberlen, Michael Parros, John Russell
Department of Electrical Engineering and Computer Science
United States Military Academy
West Point, New York 10997 USA

Faculty Advisors: CPT Jeremy Spruce, Dr. Aaron St. Leger, and Dr. Suzanne J. Matthews

Abstract

Modern power grids quickly distribute electricity across large geographic areas with a high level of reliability. However, they are not invulnerable to widespread failures. In occasions of hardware failure, or fault (such as a transmission line tripping) the system can operate in a sub-optimal state and result in a loss of electric power to some customers. These events require grid operators to locate the point of failure in order to resolve the problem, a process which can take minutes or several days in large cascading blackouts. In recent years, engineers have explored ways to automate the rerouting process so that the grid can regulate itself. One such proposed system is the Real-Time Smart Grid, which seeks to monitor the vitals of a power grid in real-time. In this paper, the anomaly detection software components of the Real-Time Smart Grid is described. The solution proposed here incorporates the Phoenix++ MapReduce framework to process the large amount of data constantly produced by the grid in parallel. The algorithm enables the Real-Time Smart Grid to detect anomalies rapidly, provide data to automated controllers, and notify grid administrators of the location of any points of failure. This can enable grid operators to analyze and mitigate potential issues and concerns in a matter of seconds.

Keywords: smart grid, MapReduce, anomaly detection

1. Introduction

Many devices used today depend on a constant flow of electricity. A blackout poses a significant risk to national security. Yet, much of the core infrastructure (e.g. transmission and distribution systems) of the United States power grid has not been updated in over half a century. In 2003, a blackout affected areas of the Northeastern and Midwestern United States, eliminating power to 50 million citizens. The outage was caused by the loss of three transmission lines in Ohio over the course of an hour [1]. A joint task force was convened to investigate the cause of this blackout and provide recommendations to prevent such a large event from reoccurring. Four main causes and forty-six recommendations for correction were presented in the task force's final report [2]. The main causes concerned insufficient system analysis and operation. While capable systems for monitoring and operation of the power grid are already implemented, large blackouts continue to occur and their impacts underscore the need for improvement.

One innovation that has been introduced to improve reliability and resiliency of the power grid under extreme circumstances (e.g. large weather events), is the development and deployment of the smart grid [3] technology. In order for a power grid to successfully operate, it must have wide area situational awareness. One way in which smart grids allow for more efficient and sustainable systems is through sophisticated and thorough gathering of data from the grid, creating a Wide Area Monitoring and Control (WAMC) system. WAMC systems enable the monitoring of grid performance and current operating state, and facilitates advanced wide area control when required. The WAMC technology increases grid operators' situational awareness and control capability.

In this paper, a wide area measurement system that enables real-time detection of anomalies in the smart grid is presented. The system developed can process large amounts of data very quickly to provide analysis in real-time. For the purposes of this work, real-time is defined in the order of seconds. At the core of the design are two MapReduce [4, 5] algorithms that are capable of sifting through large amounts of power data for operator-defined anomalies. Anomalies are communicated to grid managers through a graphical user interface (GUI). Finally, a database stores historical data and outputs information to grid operators. With this rapid access to information, grid managers have the power to make adjustments to the power grid in real-time order to avoid major problems.

The proposed method was tested on a smart grid testbed that provides 1000:1 scale emulation of a seven bus power grid. The program is run against "slices" of data output by the grid, accumulated over specified intervals of time. Results illustrate timing results against time slices of 15, 30, 45, and 60 minutes of data. While the slices of data are not indicative of real-time analysis, the quantity of data in these time slices reflects data sizes from a much larger system. This is more representative for an application on a real power grid. For example, sampling data from seven buses over 60 minutes provides as much data as sampling data from 32,400 nodes for one second. Preliminary results indicate that on a 60-minute time slice, simple-constraint anomalies can be detected in 6.73 seconds, and temporal-constraint anomalies in 13.20 seconds.

The presented results suggest that the proposed algorithm is capable of detecting anomalies on the order of seconds for large sets of power system measurement data. This initial work is simply analyzing raw data for anomalies. These anomalies could be the result of bad measurement data or an issue with the power grid operation. At this point, the method does not differentiate between the two. Future work will implement further analysis into this algorithm.

The rest of this paper is organized as follows. Section 2 of this paper outlines the design of the project, and explains how the MapReduce algorithm fits within the system at large. Section 3 discusses experimental design and results. The paper is conclude in Section 4.

2. Methodology

Figure 1 depicts an overview of the system design. Prior work [6] at USMA has developed a smart grid testbed for developing and studying cyber-physical and WAMC systems. The power system is emulated with real hardware and instrumented with Phasor Measurement Units (PMUs). Syncrophasor data is transmitted from the testbed to a Windows machine, which currently houses the GUI and database, which is connected to the OpenPDC [7] system. This data is transferred from the database to a shared directory on the Linux server in pre-determined time slices, and stored as a comma separated value (CSV) file. The Linux server runs the anomaly detection algorithm, and logs any errors. If errors are found, they are transmitted directly to the GUI.

In real systems, time slices will need to be necessarily small; the larger the time slices the longer the response time. However, in this work on a small scale emulation, the goal in picking time slices was to generate larger datasets in order to stress-test the proposed framework. A 60-minute time slice in the scaled emulation provides the same amount of data as a one second time slice from 32,400 PMUs in a real system. Therefore, the results on these larger intervals in scaled-emulation allow for predicting performance on a real system, where the same amount of data is captured in smaller time intervals.

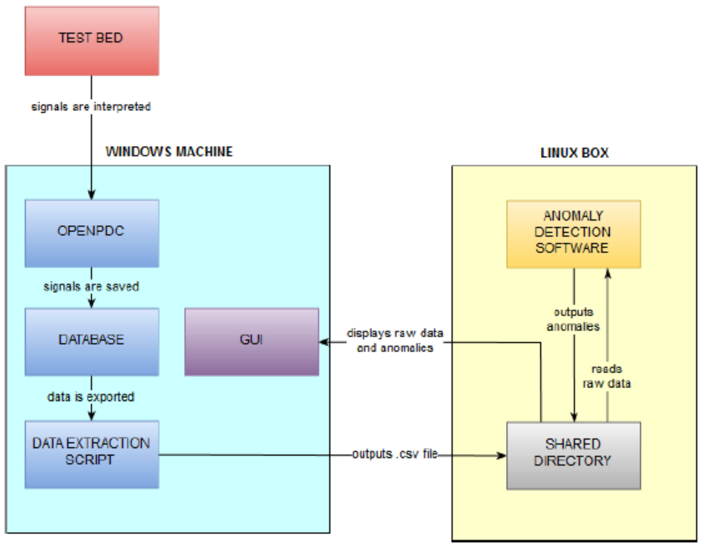

Figure 1. Overview of the system design

2.1. MapReduce

In order to analyze data at a real-time pace, the anomaly detection algorithm utilizes MapReduce [4]. The MapReduce paradigm consists of two primary phases: map and reduce. During the map phase, each instance of the map function (mapper) takes a chunk of data as input and converts it into a series of *(key,value)* tuples. These tuples go through a combiner which sorts them into *(key,list(values))* tuples, where all the values with a corresponding key are grouped together. During the reduce phase, each instance of the reduce function (reducer) takes a set of *(key, list(value))* tuples and performs a reduction operation on the list portion of each tuple. The final output is a set of *(key,value)* tuples, where the corresponding value of each key is the result of the reduction operation.

Phoenix++ [5], a shared-memory implementation of the MapReduce framework, was used for implementing the anomaly detection approach. Prior work [8,9] has shown that Phoenix++ is very effective in achieving speedup on multicore architectures. The focus was on achieving speedup on a small cluster of multicore machines for this project, as they are a simpler and low-cost alternative for WAMC implementation. Other implementations of MapReduce such as Hadoop [10] require larger clusters and petascale data in order to fully leverage the power of the system. Phoenix++ enables the features of MapReduce (fault tolerance, ease of use) in a more cost-effective way. However, the algorithms discussed in this paper can be implemented in a framework such as Hadoop, if the need arises.

2.2. Anomaly Detection Algorithms

In order to rapidly process syncrophasor data outputted from the testbed, two novel MapReduce algorithms were developed which were implemented using the Phoenix++ framework. The two algorithms tackle two categories of anomalies: constraint-based anomalies and temporal-based anomalies. Constraint-based anomalies are measurements that err from a pre-defined window of allowed variance. For example, a voltage measurement can be measured as being too low or too high. In contrast, temporal-based anomalies are only erroneous when viewed in the context of a time-sequence of measured values. For example, frequency differential anomalies are detected by examining successive measurements from a single PMU. If the frequency over time suddenly increases or decreases, a reading is anomalous.

The two algorithms discussed are meant to be run in parallel on two different machines on the network. The shared directory is accessible on the network, enabling each machine to access the data necessary for simultaneous analysis. On each machine, each parallel algorithm leverages all available cores to detect anomalies. Thus, there are two layers of parallelism: node level and core level. This allows the two algorithm to run efficiently and cheaply in tandem.

Figure 2 depicts an overview of the constraint-based anomaly detection MapReduce algorithm. The input to the algorithm is a CSV file containing a signal ID, timestamp data, and measurement data. Each unique measurement appears on its own row of the CSV file. The signal ID uniquely identifies a substation and the type of measurement. Observe the first line of the "input data" in Figure 2. The mock ID "A-Volt" indicates the measurement is a voltage measurement originating from substation A. The next comma separated value in that line is a timestamp indicating when the measurement was taken (shown in military time). In this example, the timestamp indicates the voltage was measured at 1200 hours. The last comma separated value denotes the actual measurement, 26.1 kilovolts in this example.

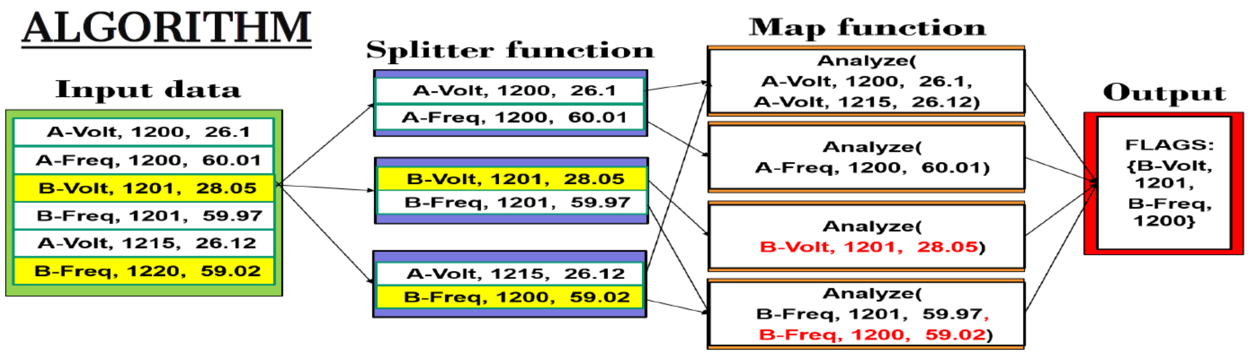

Figure 2. Constraint-Based Anomaly detection algorithm

To detect constraint-based anomalies, the input data first passes through a splitter function that separates the data into chunks. The mapper then processes each chunk for anomalies. The signal ID is used to determine the type of reading (i.e., voltage), and the corresponding measurement is checked to verify that it is within an allowable range. In Figure 2, the voltage from substation B has a measured value of 28.05 (third line of CSV file), which is outside the allowed variation. The map function thus marks it as an anomaly and emits it to the reduce function. In contrast, the measurement in the first line of the CSV file is not anomalous. The map phase confirms this, and the measurement is silently ignored. Note that only anomalous measurements are emitted to the reduce phase. These anomalies, grouped by signal ID, are then outputted to the GUI. Note that each mapper runs independently and simultaneously on multiple cores.

The temporal-based anomaly detection algorithm is similar to the constraint-based detection algorithm. It begins again with the same CSV input file. The splitter function splits the data into equal sized chunks. The mapper, however, outputs *(key,value)* pairs where the key is the signal ID, and the value is the tuple *(timestamp, measurement)*. The combiner sorts the data into *(signal ID, list(timestamp, measurement))* tuples. In the reduce phase, the list of *(timestamp, measurement)* tuples associated with each signal ID is sorted according to timestamp, allowing the data to be analyzed sequentially. While the principle benefit of this approach is that it enables the detection of temporal anomalies, the temporal-constraint algorithm is more inefficient due to the extra sorting step required.

3. Experimental Design and Results

To test the proposed approach, a 1000:1 scale emulation of a distribution power grid with real hardware, including transmission lines, buses, protection relays, and PMUs was used. Figure 3 depicts this testbed. The data source is eight PMUs connected to the power grid. A PMU manages power and records incoming data from the power grid. Data flows from the PMUs to a Phasor Data Concentrator (PDC) which processes, time aligns, and archives the PMU data to a database, which also keeps track of long term data for future use of analyzing trends. A connection string from the program OpenPDC connects to Microsoft SQL Server 2008. The database was created from the OpenPDC default and stores all of the measurement values from the PMUs. SQL commands were used to export the requisite syncrophasor data from the database to the desired CSV files, which represent time intervals of size *t*. Lastly, the GUI enables the grid manager to identify grid anomalies and make adjustments to the grid.

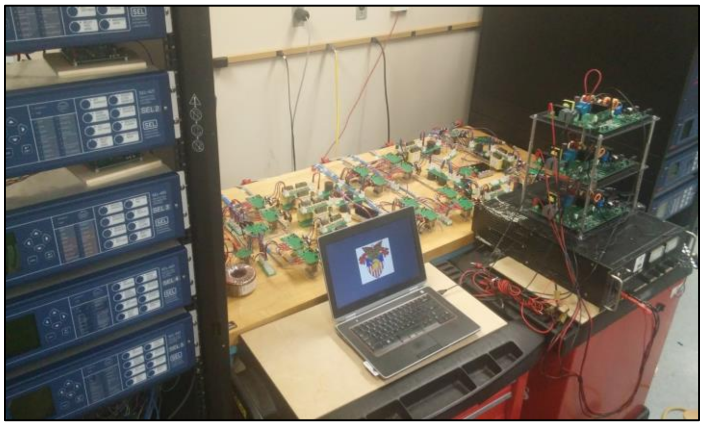

Figure 3. The physical testbed and OpenPDC server

The primary objective is to process syncrophasor data as efficiently as possible. Parallel computation was leveraged to provide timely information to operators, and potentially tie processed data to wide area control applications. A controllable load allows for replication of any load profile, and enables grid managers to make adjustments to the grid as anomalies arise. A solar micro-inverter provides a secondary source of power that converts solar energy into usable current for the grid. The GUI used HTML for the basic design of the interface. PHP is used to receive data from the database and the anomaly detection server to display data to the user. The GUI updates itself every second, displaying new values from its data source. In this section, the anomaly detection criteria is defined for the testbed and the performance of the anomaly detection algorithms is presented.

3.1. Anomaly Detection Criteria

Table 1 lists the criteria for the 1000:1 power grid emulation in real world quantities. Please note that actual emulated values are orders of magnitude smaller for voltages and currents. However, measurement equipment is scaled such that the raw data is relative to real-world values. The parameters can be varied based on application. To use this approach in a different framework, it suffices to update the constraints information.

For any power system, operators want to track under voltage bus or overvoltage bus situations. It is also desirable to detect if subsequent samples differ too rapidly, as this can be indicative of an underlying issue. If the grid channels too much current, there is risk in damaging the components of the grid, as there are limitations on lines and transformers in regards to current carrying capability. Frequency must be tightly monitored around the value of 60 Hz, the standard of the United States power grid. Furthermore, the difference between subsequent frequency samples should remain relatively low in order to ensure a steady frequency state. For the purposes of this proof of concept, the voltage and frequency differentials are analyzed at subsequent samples (sample rate is 1/60 seconds). However, in application the acceptable bound for these differentials will be heavily dependent on the system being monitored. The parameters in this anomaly detection algorithm can be set to appropriate values based on the application.

Table 1. Criteria for the Power Grid Emulation

Parameter	Acceptable Bounds
Voltage	25-27.6 kV
Voltage Differential	0 – 0.0263 kV
Current	0 – 2000 A
Frequency	59.95-60.05 Hz
Frequency Differential	0 – 0.02 Hz

3.2. Performance of Anomaly Detection Algorithm

Performance of the anomaly detection algorithms was benchmarked on a 12-core machine running Red Hat Enterprise Linux. Each core is an Intel Xeon X5690 with a clock rate of 3.47 GHz. The system has 141 GB of main memory, and 1 TB of hard disk. Time slices of data read from the OpenPDC server in 15 minute, 30 minute, 45 minute, and 60 minute increments were used. The number of cores was varied from 1 to 8 (in powers of 2), and computed the average run-time and speedup.

Table 2 summarizes the experimental results of the constraint-based anomaly detection algorithm. A 15 minute time slice takes 4.06 seconds to process on a single core, while a 60 minute time slice takes approximately 17.55 seconds. Increasing the number of cores does result in an improvement in run-time, albeit a non-linear one. For the 15-minute increment slice, the run-time was reduced to 1.65 seconds on 8 cores, a speedup of 2.46. For the 60-minute slice, a run-time of 6.73 seconds was achieved on 8 cores, which corresponds to a speedup up 2.61. This data processing speed would be indicative of analyzing data from a one second time interval from 32,400 nodes on a power grid in 6.73 seconds for eight cores.

Table 2. Average run-time results for constraint-based detection algorithm.

Time Slice	1 Cores	2 Cores	4 Cores	8 Cores
15 minutes	4.06	2.71	2.04	1.65
30 minutes	8.43	5.61	4.10	5.09
45 minutes	12.84	8.34	6.33	5.09
60 minutes	17.55	11.53	8.41	6.73

Table 3 depicts the performance analysis of the temporal-based anomaly detection algorithm. The serial version of this algorithm took approximately the same as the constraint-based approach. However, the parallel implementation was not as efficient as the constraint-based approach. This is perhaps to be expected, given the additional sorting step required in the reduce phase of the temporal-based algorithm. On the 15-minute time slice, approximately 36% improvement was observed when running the approach on eight cores. On the 60-minute time slice, a 33% improvement.

Table 3. Average run-time results for 2nd algorithm variant

Time Slice	1 Cores	2 Cores	4 Cores	8 Cores
15 minutes	4.34	3.86	3.25	3.19
30 minutes	8.60	7.24	6.76	6.53
45 minutes	13.09	11.10	10.26	10.05
60 minutes	17.53	14.67	13.78	13.20

The parallel approach presented here could benefit from some optimization. First, there are likely inefficiencies in the implementation. This initial implementation is just a proof of concept; the main goal was to show that the MapReduce paradigm could be used to parallelize the analysis of syncrophasor data. In this vein, the project is a success. The multicore run-times demonstrated here are indicative that large sets of PMU data from a large scale power grid (tens of thousands of buses) can be sorted and processed on the order of seconds, fast enough to allow for real-time, or near real-time, detection of anomalies. Second, it is possible that the data collected from the PMUs were insufficient to leverage the full power of the Linux system. It is suspected that with larger datasets, better speedups will be achieved. This, along with further optimization of the algorithm, will be the focus of future work.

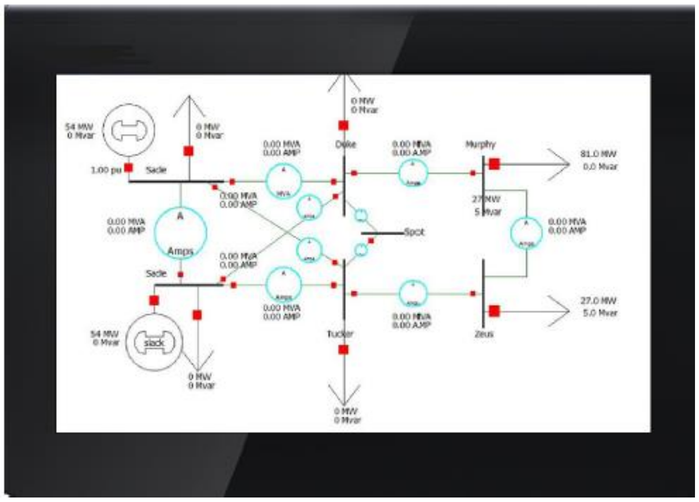

Figure 8. A mock-up of the GUI.

4. Conclusion and Future Work

From the presented results, it can be concluded that the implementation of the proposed anomaly detection algorithm in a real smart-grid environment is feasible with today's computing capabilities. While the testbed is a 1000:1 scale emulation, the scripts and programs created to analyze data is scalable and capable of handling larger sums of data with more grueling requirements. From building this system, the benefits of working with MapReduce and other parallel computing frameworks are apparent for the implementation of any large scale data analysis.

Two anomaly detection algorithms were presented. The first enables the real-time detection of constraint-based anomalies. The second enables temporal-based anomalies to be detected in parallel. While the implementations of these algorithms could stand improvement, the results lend credence to the claim that parallel computing can enable the real-time detection of power grid anomalies. Future work will concentrate on code optimization, running larger datasets for analysis, and adding more thorough analysis into the algorithm (e.g. state estimation, voltage stability assessment, etc.).

Future improvement to the GUI include options that will allow grid administrators to toggle between manually adjusting the grid and automatically having the grid regulate itself. This requires us to address how a TCP connection can be made from the PMUs through the GUI itself, in order to independently control opening and closing the breakers. Furthermore, displaying historical data and alerting the user of potentially problematic trends over a period of time will assist grid administrators in assessing the health of the grid. In the future, grid operators should be able to view grid information on a portable device such as a smartphone or tablet. A mockup is shown in Figure 8. Furthermore, work needs to be completed in order to enable to GUI to properly display anomaly detection warnings.

While the program is optimized for speed, in reality there are many other factors to consider as well. Within recent years, several actors – both independent and government sponsored – have targeted rival countries' power grids in attempts to both disrupt quality-of-life and obtain military advantages. As such, any software tied to a power grid needs to feature robust intrusion detection mechanisms and must avoid unsecure protocols. Improving the security of the proposed approach represents another avenue of future work.

5. Acknowledgements

The opinions expressed in this work are of those of the authors and do not necessarily reflect the U.S. Military Academy, U.S. Army, or U.S Department of Defense.

The authors are incredibly grateful for the assistance of the Computer Support Group and Electronic Support Group at the United States Military Academy for helping troubleshoot the servers. We would also like to acknowledge Michael Parros, who was a student contributor and co-author of this work. Shortly after graduating from the United States

Military Academy, Michael tragically passed away. We will forever be greatful for his contributions and, more importantly, for the honor of having had Michael as a collegue and friend.

6. References

1. Tollefson, Jeff. “US electrical grid on the edge of failure”. Nature News, August 23, 2013. *Available on-line: www.nature.com/news/us-electrical-grid-on-the-edge-of-failure-1.13598.*
2. U.S.-Canada Power System Outage Task Force. “Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations”. April 2004.
3. U.S. Department of Energy. “The Smart Grid: An Introduction”. Vol. 1. Washington D.C. 2009.
4. Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.
5. Talbot, Justin, Richard M. Yoo, and Christos Kozyrakis. "Phoenix++: modular MapReduce for shared-memory systems." *Proceedings of the second international workshop on MapReduce and its applications*. ACM, 2011.
6. St. Leger, Aaron, et. al. “Smart Grid Testbed for Wide-Area Monitoring and Control Systems”. *Proceedings of 2016 PES Transmission and Distributed Conference and Exposition.* Dallas, TX, May 2-5, 2016.
7. Alliance, Grid Protection. "OpenPDC." *Available on-line: http://openpdc.codeplex.com* (2011).
8. Tyson, Bryce, et al. "Using MapReduce to Compare Large Collections of Phylogenetic Trees." *Proceedings of the 2014 ACM Southeast Regional Conference*. ACM, 2014.
9. St. Amour Leo, et al. "PAVE: Writeprint Creation with MapReduce" *Defense Technical Information Center (DTIC) technical report*. Army Research Labs (Accession number:AD1005367), 2016.
10. White, Tom. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.