Leveraging Single Board Computers for Anomaly Detection in the Smart Grid

Suzanne J. Matthews, Aaron St. Leger
Department of Electrical Engineering & Computer Science
United States Military Academy, West Point, NY 10996
Email: [suzanne.matthews, aaron.stleger]@usma.edu

***Abstract*—Smart Grid Technology is becoming an integral part of ensuring reliable and resilient operation of the power grid. The high sample rate and time synchronization of Phasor Measurement Units (PMUs) can provide enhanced situational awareness and more detailed information on power system dynamics as compared to traditional SCADA systems. A smart grid system must be able to detect alarm events (such as sudden voltage fluctuations or drops in current) in close to real-time. However, the communication network and bandwidth requirements to transfer large amounts of PMU data for real-time analysis is problematic. In this paper, we propose the use of a decentralized architecture for rapidly analyzing PMU data using single board computers to provide energy efficient monitoring locally in the power grid. This approach reduces communication requirements and enables real-time analysis. We present a novel anomaly detection scheme and test our approach on a real dataset of 1.4 million measurements derived from 8 PMUs from a 1000:1 scale emulation of a working power grid. Our results show that a single Raspberry Pi is sufficient to analyze data from multiple PMUs at a rate suitable for real-time analysis.**

Keywords: Smart Grid, Single Board Computer, Raspberry Pi, Anomaly Detection

I. Introduction

Smart grid technology is enabling improved efficiency, resilience and reliability in power grids. Phasor Measurement Units (PMUs) are a key enabling technology. Specifically, PMUs provide direct measurement of voltage and current phasors (magnitude and phase angle of an alternating current waveform), are timestamped via global positioning system clocks, and can provide measurement rates up to 60 Hz. PMU integration as compared to traditional transducers provides more data (measurements), higher measurement accuracy and measurement correlation through time synchronization.

Many offline and online applications using PMU data are currently being investigated [1]. Online applications require acquisition and analysis of data in real-time or near real-time to provided actionable information to power system operators. Examples include Wide Area Monitoring Systems (WAMS) deployed in Texas [2] and India [3]. These projects focused on integrating PMUs and communication networks to provide PMU data to existing SCADA systems and develop new applications based on PMU data.

One online application of great interest is detecting anomalous power grid behavior. For example, detection and notification of power and frequency oscillations is identified as one of the most beneficial uses of PMU data by the North American Electric Reliability Corporation (NERC) [2]. The additional data can also enhance real-time situational awareness of power system operators. However, as more PMUs are deployed it becomes challenging to transmit and analyze the data, and provide actionable information in real-time as discussed in [4]. Most WAMS are designed to aggregate all data and analysis at a centralized server. This requires a robust communications network and will add latency in anomaly detection as all data must be transferred prior to analysis.

Compressed sampling can lower the bandwidth requirements for WAMS [5]. An alternate solution is to complete the bulk of anomaly detection as close to individual substations as possible. In this type of architecture, compute nodes are networked directly to individual (or collections of) PMUs and perform anomaly detection; alarm events are then propagated through the network to the centralized controller. The advantage of this type of distributed architecture is reduced latency. To enable such an architecture, the compute nodes must be small, power-efficient, inexpensive, and able to detect anomalies as data packets arrive. For a PMUs reporting at 60 HZ, a packet must be inspected for anomalies in under 16.67 ms to be real-time.

In this paper, we propose the use of single board computers (SBCs) for localized anomaly detection. The Raspberry Pi is a power-efficient, compact and inexpensive device at $35.00. It consumes approximately 5 watts of power and is roughly the size of a credit card. We compare the performance of the Raspberry Pi to a commercial off the shelf (COTS) system on a real dataset consisting of 1.4 million measurements from 8 PMUs, derived from a 1000:1 scale emulation of a working power grid, operating at 60 Hz. Our results show that a Raspberry Pi can detect anomalies in 70.5 μs and 159.6 μs on 1-phase and 3-phase data derived from a single PMU. Furthermore, the Pi is capable of detecting anomalies from 8 PMUs in 1.32 ms. Our results indicate that the processing power of the Pi is sufficient for real-time anomaly detection, even up to 50 PMUs. This strongly suggests the utility of the Raspberry Pi and similar SBCs for distributed anomaly detection in the power grid.

The rest of the paper is organized as follows. In Section II we give background and related work. We discuss an overview of the proposed architecture in Section III. The anomaly detection process is discussed in Section IV. Our results are shown in Section VI. Finally, we conclude in Section VII.

978-1-5386-1104-3/17/$31.00 ©2017 IEEE

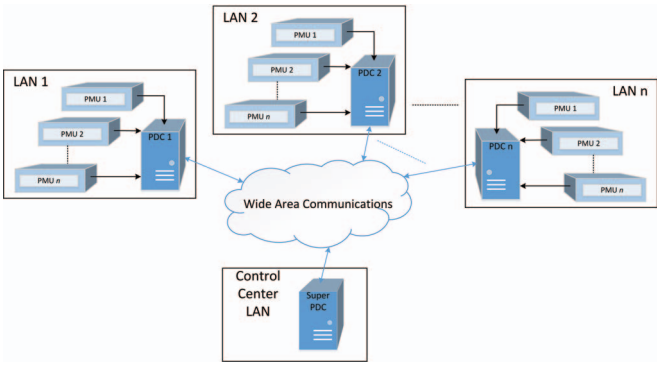

Fig. 1. Overview of a PMU based wide area measurement system

II. Background

PMU-based WAMS provide an opportunity for enhanced situational awareness for power system operators as compared to traditional Supervisory Control and Data Acquisition (SCADA) systems. Information provided to conventional SCADA systems (such as measurements of voltage and current magnitudes, system frequency, and power) are asynchronous in nature and updated on the order of seconds to minutes. This data is then used to estimated power system topology, power system states, and provide useful information to system operators. The rate of data acquisition and asynchronous nature of the data limits the speed, accuracy and anomaly detection capability of the subsequent analyses.

One advantage of PMU-based WAMS is the capability of sub-second sampling rates. Specifically, the IEEE C37.118 Standard [6] supports data rates up to 60 samples per second for a 60 Hz power system frequency. Higher sampling rates are encouraged but not required to be compliant with the standard. The higher fidelity of sampling allows for system operators to observe, monitor, and take remedial action on power system oscillations that were previously unobservable with traditional SCADA tools. For example, the WAMS pilot project in India [3] had immediate improvements to visualization of power system dynamics and understanding of system operation, enhanced real-time situational awareness, visualization of oscillations, and allow for more thorough offline analysis and modeling of the power system. An overview of a PMU based WAMS, derived from the structure of WAMS projects presented in [2] and [3], is shown in Figure 1.

PMUs are installed throughout the power grid to provide synchrophasor measurements and connected to local Phasor Data Concentrators (PDC) via a Local Area Network (LAN). The purpose of the PDC is to aggregate and time align data from multiple PMUs, archive data in local storage, and serve this data to the control center. Multiple PDCs are required throughout the system based on the location and number of installed PMUs. A wide area communication network is required to transmit data from all the PDCs to the super PDC in the control center, which aggregates and serves all PMU data to the system operator and WAMS applications. The wide area communication network is a critical, and challenging, aspect to implementing a PMU-based WAMs. The PMU sampling rate used in [3] was 25 samples per second and it was reported that "for such a high rate of reporting, in general, a dedicated fiber channel is required for reliability of real-time data" [3]. The wide area communication requirements will continue to grow for centralized WAMs systems as PMU integration continues.

An alternative approach is to process raw PMU data in a decentralized fashion within the power system and report usable information (e.g. a reported anomaly or alarm) in real-time to the system operator. The primary advantages of this approach are a significant reduction in data transfer requirements for real-time situational awareness and that the reportable data can be directly integrated into existing SCADA tools. A disadvantage to the distributed architecture is that analysis is limited to a subset of system PMU data. However, such distributed analysis can augment existing WAMs systems and provide more timely anomaly detection to system operators. If PMU data is processed and anomalies identified in real-time within the grid, anomalies can then be reported directly to WAMS applications within the latency time of the wide area communications systems. The centralized architecture transfers significantly more data from PMUs to the centralized PDC prior to analysis.

A distributed data analytics approach to WAMS has recently been presented in [7] which proposes a dedicated data server for real-time distributed PMU data analytics. The approach of this work is similar but investigates different algorithms for anomaly detection and the suitability of a low-cost single board computer for processing PMU data in a distributed fashion.

A *single board computer* (or SBC) is a device in which the entirety of the computer is printed on a single circuit board. Unlike microcontrollers, a SBC is a fully functioning computer, with an operating system, random access memory, and flash storage. These features enable SBCs to be easily reprogrammable and capable of more computationally difficult tasks than microcontrollers. While FPGAs are another attractive option, SBCS are easier to reprogram and relatively inexpensive. For example, the Raspberry Pi (arguably the most popular SBC on the market) costs only $35.00 and features a 1.2 GHz processor and 1 GB of RAM [8].

While SBCs typically have similar System on a Chip (SoC) processors to those found in smartphones, they are designed for hardware integration, typically including Ethernet, General Purpose Input Output (GPIO), and a host of other ports not typically found on mobile devices. For example, the Raspberry Pi SBC features 4 USB ports, a HDMI port, and display and camera standard interfaces in addition to integrated wireless and bluetooth.

Research on wireless sensor networks have shown that at-node data processing is critical for lowering their latency and power consumption [9], as the energy required to transfer the data often exceeds the amount needed to process the data at the node. In the context of a power system, we posit that SBCs can act as so-called "gateway" nodes, performing at PMU

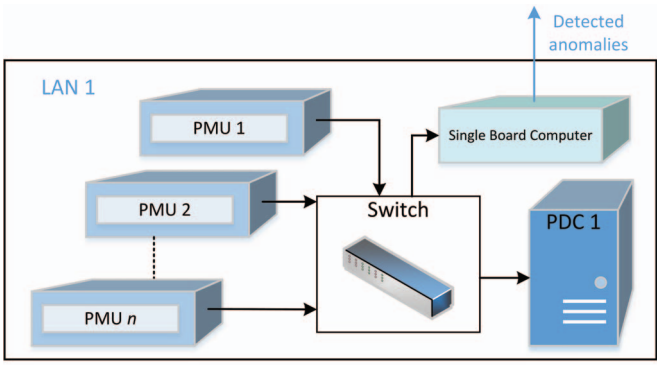

Fig. 2. Proposed decentralized SBC based PMU anomaly detection architecture

or close to PMU anomaly detection. Other researchers have shown the feasibility of using the Raspberry Pi for localized information processing in sensor networks; Bell [10] has a nice introduction.

Utilizing single board computers for smart grid anomaly detection supports the notion of energy proportional computing. Barroso and Hölzle observe that Google servers operate at peak energy efficiency at peak utilization, concluding that system architects should develop machines that consume energy in proportion to the work performed [11]. Da Costa later showed that combining servers with power-efficient single board computers in a data center results in a more power efficient system [12]. We hypothesize that if single board computers can perform anomaly detection at requisite speeds, they would do so closer at peak utilization, resulting in improving the energy efficiency of anomaly detection in the power grid.

III. System Architecture Overview

Recently, Matthews and St. Leger described a novel MapReduce algorithm [13] for real-time anomaly detection for Wide Area Monitoring Systems of an architecture shown in Figure 1. Their algorithm was capable of analyzing 45 minutes worth of data (18 million measurements) derived from 7 PMUs for anomalies in under three seconds on a multicore commercial of the shelf (COTS) system. In contrast, this work investigates the use of power efficient SBCs, such as the Raspberry Pi, for anomaly detection close to the PMU. Specifically, the contributions of this work are as follows:

- A description of an inexpensive distributed architecture capable of real-time anomaly detection in the smart grid.
- The use of single board computers such as Raspberry Pi for the compute nodes in the aforementioned architecture.
- Achieving real-time anomaly detection on real power grid data using single board computers.

Figure 2 gives an overview of the proposed decentralized architecture with a SBC. The SBC, such as the Raspberry Pi, is connected via Ethernet to the LAN. This provides direct connection to the PMU data stream in a similar fashion as the PDC. SBCs would be embedded throughout the power system. A single SBC per LAN houses local PMUs and a PDC. The concept is that each SBC streams PMU data from the LAN and analyzes the data for anomalies in real-time. Each SBC in the network has copy of the code, and a "constraints" file containing nominal ranges for synchrophasor measurement data.

Each PMU in the LAN transmits a data frame, encapsulated within a single TCP/IP packet, at a rate commensurate to the reporting rate. For example, a PMU set to report 60 measurements per second (60 Hz reporting rate) will send a packet of data 60 times per second. This translates to a packet every 16.67 ms. As per the synchrophasor standard, a data frame contains a time stamp, phasor measurements, frequency measurement, rate of change of frequency measurement, and any analog/digital measurements defined for a given PMU.

For the purposes of the anomaly detection in this work, a subset of reportable data extracted from a packet is analyzed for anomalies. Specifically, for a PMU reporting 1-phase, 5 measurements per frame are analyzed; those reporting 3-phase have 11 measurements analyzed. The SBC receives packets, extracts the pertinent data frame, then analyzes the frame for both constraint and temporal anomalies. Triggered anomalies are then transmitted through the network, eventually reaching an operator via a user interface.

There are several advantages to the strategy. First, SBCs are inexpensive and power-efficient, making integrating them into existing smart grids economical. Second, having data processed close to the PMU reduces overall latency of the system, as the bulk of data will be processed by the Raspberry Pi, rather than being transferred over the network. Lastly, the described strategy also reduces the total number of measurements that need to processed at any given time, lowering the computational requirements.

Data collection and testing is derived from a Smart Grid Test-Bed developed at the United States Military Academy [14]. The power system emulated in the test-bed is based on a seven bus three-phase distribution grid. It contains nine transmission lines with a line voltage of 46 kV. The test bed is a 1000-scale emulation, with a system line voltage of 46 V and a maximum line current of 2 Amperes. It contains eight real Schweitzer Engineering Lab (SEL) Relays and IEEE C37.118 compliant PMUs, time synchronized to GPS satellite clocks.

IV. Anomaly Detection

As previously mentioned, each PMU sends a packet of data to a single SBC at a desired reporting rate. Each packet contains a predefined (based on PMU configuration) data frame. This data frame consists of a time stamp, phasor measurements, frequency measurement and a rate of change of frequency measurement. Each packet also contains a unique

Hash	Timestamp	Value
37E9A476-99AC-4AE0-B65B-8321A6C226B0	01-Aug-2016 14:54:03.966	27011.1687859055
96CD18EA-B2FE-450F-A97A-1C2954CC6F95	01-Aug-2016 14:54:03.966	26.8312162492571
EFEDF24E-D0CE-4D9C-AE7E-4BACE825BACE	01-Aug-2016 14:54:03.983	-5.33860431521961
37E9A476-99AC-4AE0-B65B-8321A6C226B0	01-Aug-2016 14:54:03.983	27010.7424920033
96CD18EA-B2FE-450F-A97A-1C2954CC6F95	01-Aug-2016 14:54:03.983	26.8312162492571
EFEDF24E-D0CE-4D9C-AE7E-4BACE825BACE	01-Aug-2016 14:54:04.000	-5.32893611411344

Fig. 3. A snapshot of the measurements collected from the PMUs.

ID code which identifies which PMU is sending the information. The SBC extracts data from the packet to compose a time series of measurements. Bandwidth requirements per PMU will depend on reporting rate, communciation protocol overhead, and configuration of the PMU. Section C.1 in [6] provides guidance on estimating communication bandwidth requirements for PMUs.

For example, a PMU reporting 2 phasors at 60 Hz in integer format requires 40.32 Kbps of bandwidth for data and an additional 36.48 Kbps for communication protocol overhead (44 bytes per packet for TCP and 2 bits per byte for asynchronous communication). Summarizing, a single PMU with this configuration requires a total of 76.8 Kbps of bandwidth. For this work scaled data representing up to 50 PMUs was tested for a single SBC. This would require streaming and extracting data at a rate of 3.85 Mbps in this configuration. Given the expected performance of the LAN and a 100 Mbps Ethernet card on the SBC, it is assumed that the extraction of data from 50 PMUs can occur at line speed.

Given that the SBC can extract data from a small number of PMUs at line speed, the measurement data is provided to the SBC in comma separated value (CSV) format for the context of this paper (see Figure 3). In other words, for testing purposes the data provided to the SBC has already been extracted from packets. Each measurement occupies a single line, and consists of a signal ID, time-stamp, and measurement values. The signal ID is a 36 character hash that is unique to a particular PMU and measurement type (i.e., voltage, current, etc.) The time-stamp has centisecond precision. The measurement value has double floating point precision.

Our goal in conducting this research was to ascertain the viability of using single board computers such as the Raspberry Pi to perform anomaly detection on synchrophasor data. We have not yet integrated our Pis with our testbed; that is future work. For our purposes we simulate the stream by creating subsets of our collected measurement data and loading that data into memory. Data is read from the stream at a user-specified reporting rate which correlates to the PMU reporting rate. For our 60 Hz reporting rate, either 5 or 11 measurements are read from the stream per PMU at a given time, emulating the transmission of packet data frames from 1-phase and 3-phase power systems. To achieve real-time performance, our algorithm must be able to detect all anomalies in the frames in under 16.67 ms.

A. Measuring Anomalies

In our work, we concentrate on detecting two different classes of anomalies: constraint and temporal. A *constraint anomaly* is defined as a measured value that is outside a predetermined acceptable range which was heuristically determined. Nominal fluctuations in measured quantities such as voltage, current, and angle differential are consider normal in power systems. Detecting measurements that fall outside an allowable domain of nominal fluctuation is possible at line speed; a program simply needs to check each measurement against its range of allowed variation as it is read from a stream.

Temporal anomalies require more data and further processing in order to detect. A *temporal anomaly* is defined as a rapid oscillation, or change, in a measurement value within a given time frame. Thus, while any individual measurement value does not fall outside the range of allowed variation, the magnitude of the change within a given period of time is significant enough to warrant further investigation. Laboratory experimentation, data analysis, and heuristics were used to define Fano factor limits for temporal anomalies in this work. To detect rapid oscillations or temporal changes, we compute the Fano factor of the measurements in window w. The Fano factor (F) is defined in Equation 1:

$$F = \frac{\sigma_W^2}{\mu_W} \tag{1}$$

where σ_W^2 is the variance calculated over the window, and μ_W is the mean value over that same window. The Fano factor has previously been shown [13] to be a good indicator of distribution of measurement data in a given window, and we use it here to detect anomalies with PMU data. This approach has shown to be robust in detecting oscillations and step changes in PMU data.

B. Overview of Anomaly Detection Approach

There are two principle inputs to our program. The first is an operator-specified constraint file that contains the set of allowable variations for each measurement and precomputed Fano factors for each signal ID being tracked. For practical applications, the acceptable ranges for constraint anomalies and the acceptable Fano factor values for temporal anomalies must be developed for a specific power system. Offline analysis of PMU data and historical knowledge can be used to determine appropriate settings. The second input is the file used to simulate the data stream.

Prior to running the anomaly detection component, the input data file is loaded into memory to simulate the stream. Next, the information in the constraint file is hashed by signal ID. Each signal ID's hash record contains the ranges of the allowed variation and Fano factor.

The anomaly detection procedure is a two step process. During the first step, a window of $p \times m$ measurements is read serially from the input stream, where p is the number of PMUs and m corresponds to the number of measurements in the data frame from each PMU to be analyzed. For clarity, we refer to each of the read measurements as an *event*. As each event is read, it is decomposed into its signal ID, time-stamp, and value components. The signal ID's corresponding hash record is accessed from hashtable C, and the minimum

and maximum defining the extent of allowed variation is extracted. The event's value component is then compared to this minimum and maximum; if it is out of range, the event triggers a constraint anomaly.

Regardless of whether or not a constraint anomaly is triggered, the event is inserted into a queue corresponding to its signal ID. The queue corresponding to a particular signal ID is also determined by hashing the signal ID. Each queue is implemented as a circular buffer, and is the same length as the reporting rate (r). For a reporting rate of 60 Hz, each queue holds $r = 60$ measurements. An index variable determines the next location that the event's information should be placed. With every insert, the index is incremented. The modulo operator ensures that the elements in the queue are in sequential order. There are S total queues, where S corresponds to the number of unique signal IDs.

In the second step, the program scans over all S queues, checking for temporal anomalies in each. For each queue, the associated signal ID is rehashed to access the corresponding hash record in C, in order to obtain the allowable Fano factor limit (f). The program then computes the Fano factor of the measurements in the queue. If the computed Fano factor exceeds f, a temporal anomaly is triggered.

C. Analysis of Anomaly Detection Scheme

Let S be the total number of unique signal IDs that is generated by our p PMUs, m be the number of measurements received from each PMU with each packet, and r be the reporting rate. The building of the hashtable C is a one-time operation that requires $O(S)$ time. The run time of Step 1 is dependent on the number of events read from the stream, or $p \times m$. Constant time is required to read an event from the stream, decompose it into its component parts, and perform a hash lookup in C. Since a hashtable is used to manage the S queues, the lookup to the event's corresponding queue is also constant time. Lastly, since each queue is implemented as a circular buffer, insertion of event data into a particular queue also requires constant time. Thus, the theoretical runtime of Step 1 is linear with the number of events read from the stream, or $O(p \times m)$.

The time required for Step 2 is heavily dependent on S and r. For each signal ID, we iterate over the r measurements in the queue twice to compute the Fano factor. The rest of the components of the Fano factor computation require constant time. Therefore, the total time for Step 2 is $S \times 2r$. Thus the total time required for our approach to process w events over S unique signal IDs derived from p PMUs is $O((p \times m) + (S \times r))$.

V. Experimental Methodology

Our anomaly detection code is written in the C language. We test our anomaly detection approach on a commercial off the shelf (COTS) system and a Raspberry Pi 3 SBC. Our COTS is identical to the machine referenced in [13], with a quad-core Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and 32 GB of RAM. The Raspberry Pi has a quad-core ARM Cortex A53 CPU, with a clock rate of 1.2GHz and 1 GB of RAM. We chose to focus on the Raspberry Pi due to its popularity and relative inexpensiveness to other commercially available SBCs. A single Raspberry Pi costs approximately $35.00. In contrast, the COTS system costs approximately $900.00.

TABLE I
Average Time (in microseconds) needed to detect a single constraint and temporal anomaly.

	1-phase		3-phase	
System	constraint	temporal	constraint	temporal
COTS	0.14	0.43	0.16	0.44
Pi	1.87	5.09	2.12	5.36

TABLE II
Average Time (in microseconds) needed to detect anomalies in a packet.

	1-phase			3-phase		
System	step 1	step 2	total	step 1	step 2	total
COTS	5.42	2.85	8.27	5.63	6.38	12.01
Pi	35.7	34.8	70.5	80.6	79.0	159.6

Using a KillAWatt [15], we determine the Raspberry Pi consumes approximately 4.5 Watts of power when running our application. Compare this to the COTS system, which consumes roughly 50 Watts of power. Despite the COTS having significantly more computing capability, we hypothesized that the more power-efficient Pi would still be able to perform anomaly detection at a rate corresponding to real-time.

We test our approach on a real dataset of 1.4 million synchrophasor measurements collected over 15 minutes from the Smart Grid Test Bed at the United States Military Academy. 3-phase data was derived from the 1-phase dataset by tripling the set of signal IDs for the voltage, current magnitude, and phase angle differential for each PMU. Thus, there are 5 unique signal IDs (hashes) for each PMU for the 1-phase dataset, and 11 unique signal IDs for each PMU for the 3-phase dataset.

VI. Results

We benchmark our approach using three rounds of experimentation. In the first round of experiments, we compare the raw performance of the Raspberry Pi to the COTS system for detecting constraint and temporal anomalies using our approach. In the second round of experiments, we measure performance when a single Pi is fed data from multiple PMUs. Lastly, we simulate between $10 \ldots 50$ PMUs and explore how well a single Pi can analyze data.

A. Single PMU Experiments

In our first set of experiments, we determine the average amount of time needed to detect constraint and temporal anomalies on the COTS system and on a Raspberry Pi. In this scenario, we assume that either device is collecting data from a single PMU with a reporting rate of 60 Hz. The current PMU reporting rates for a 60 Hz power system are 10 Hz, 12 Hz, 15 Hz, 20 Hz, 30 Hz and 60 Hz according to the IEEE standard [6]. In our experiments a reporting rate of 60 Hz was used. Thus, to achieve real-time performance, we must be

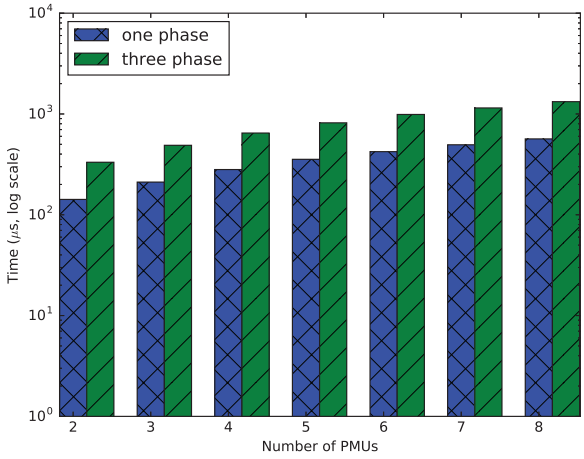

Fig. 4. Test Bed Anomaly Detection Time (μs).

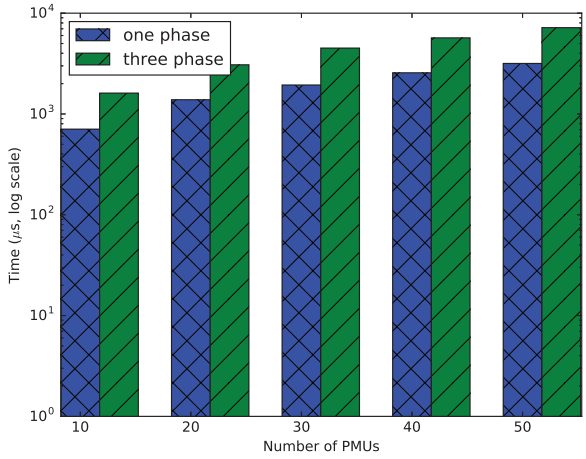

Fig. 5. Scaled PMUs Anomaly Detection Time (μs).

able to analyze data frame for anomalies in under 16.67 ms. More time would be permitted for the other standard reporting rates. We first consider the average amount of time required to detect a single constraint and temporal anomaly. Table I shows our results. Each element in the cell is the average run time (in μs) required to detect a particular type of anomaly for both 1-phase and 3-phase data. Predictably, the COTS system is an order of magnitude faster than the Raspberry Pi. For example, while it takes on average 0.14 μs for the workstation to detect constraint anomalies, the Raspberry Pi requires 1.87 μs. Temporal anomalies take the longest to detect. For the 3-phase dataset, the COTS system and the Raspberry Pi require 0.44 and 5.36 μs on average to detect temporal anomalies.

Next, we consider the average total time required to detect constraint and temporal anomalies in a single data frame extracted from a packet transmitted from the PMU. Constraint anomalies can be detected at line speed, and are detected in step 1 of our described anomaly detection approach. Temporal anomaly detection is a two step process, whose total time is equal to the sum of the amount of time needed to add measurements to the queue (step 1) and examine the queues for temporal anomalies (step 2). Table II show the total time required to inspect a single packet of data for anomalies. Again, the COTS system is an order of magnitude faster than the Raspberry Pi.

While the Raspberry Pi is slower than the COTS system, our results show that the Raspberry Pi is able to detect anomalies in a packet of information in approximately 70.5 μs on our single phase dataset, and 159.6 μs for 3-phase dataset. This matches the criteria for real-time performance, as a new packet of data only arrives every 16.67 ms. These results suggest the viability of using Pis for synchrophasor anomaly detection.

B. Multiple PMU Experiments

One argument against our proposed architecture is perhaps expense. While associating Raspberry Pis rather than COTS systems per PMU is clearly more economical, the use of Pis in larger grids with thousands of PMUs can quickly become expensive. Since anomaly detection on one PMU's worth of data is so fast, we decide to explore if a single Pi can analyze data from multiple PMUs in our testbed.

To ascertain the Pi's ability to handle data from p PMUs, we interleave their measurement data. Thus, during step 1 of our algorithm we read the equivalent of p packets of data, reflecting the amount sent to the Pi every 16.67 ms from p PMUS. The rest of our approach does not change. For brevity, we show only the total time required to detect anomalies for p PMUs. In these set of experiments, we achieve real-time performance if p packets are analyzed in under 16.67 ms.

Figure 4 show the total anomaly detection time for our 1-phase and 3-phase datasets as we add additional PMUs worth of data. The x-axis represents the total number of combined PMUs worth of data processed by a Pi. The y-axis (in log scale) represents the amount of time (in μs) the Pi needs to detect all temporal anomalies on a particular number of cores.

The average time required for anomaly detection on our one-phase dataset varies between 142 μs and 566 μs. The larger 3-phase dataset requires more time. At 6 PMUs, our detection times start exceeding 1 millisecond. The average time to process 7 and 8 PMUs worth of data is 1.14 and 1.32 ms respectively. This is still well under the 16.67 ms maximum required for real time analysis. Our results show that a single Raspberry Pi is sufficient for detecting anomalies from the 8 PMUs in our testbed.

Lastly, we scale the number of PMUs in our 1-phase and 3-phase dataset to range between $10 \ldots 50$ PMUs in increments of 10. We accomplish this by splitting each PMU's set of measurements in the 1-phase and 3-phase datasets into smaller subsets, and assigning those to new signal IDs. Thus, in the resulting scaled datasets, the total number of measurements remains constant. However, the total number of PMUs (and unique signal IDs) across those sets of measurements are increased.

Figure 5 show the results of these scaled experiments. For these larger numbers of PMUs, our detection times vary from 0.71 and 7.18 ms. Once again, the 3-phase dataset requires more time for anomaly detection. For example, for 50 pmus, we require on average 3.17 ms vs 7.18 ms to detect anomalies on our 1-phase vs 3-phase datasets respectively. Even with 50 pmus, we are still faster than the 16.67 ms threshold required for real-time analysis. These results suggest that the processing power of a single Raspberry Pi is sufficient to analyze the data from 50 PMUs in real-time.

Our experiments suggest that single board computers like the Raspberry Pi offer an inexpensive way to perform real-time anomaly detection in power grids. Even when scaling our datasets to 50 PMUs, we are still fast enough to achieve real-time performance.

VII. Conclusions

In this paper, we propose the use of inexpensive single board computers (or SBCs) to perform anomaly detection in the smart grid. Using SBCs to analyze synchrophasor measurements close to PMUs minimizes the amount of data flowing in the network, reducing energy consumption. Furthermore, using SBCs support the notion of energy proportional computing.

To test our hypothesis that SBCs could sufficiently perform anomaly detection, we simulated a data stream consisting of 1.4 million real synchrophasor measurement from a $1000:1$ scale emulated power grid. We compare the rate of anomaly detection using a commercial of the shelf (COTS) system and a Raspberry Pi.

While the COTS system was faster, the Raspberry Pi was still able detect anomalies at a rate needed for maintaining real-time. Additionally, we show that we can analyze data from multiple PMUs and still maintain real time performance. Our results strongly support the use inexpensive SBCs for localized, distributed anomaly detection in the smart grid. We believe our results will be of great interest to grid engineers looking for novel techniques for reducing the power consumption and latency of their anomaly detection workflows.

In the future, we plan to fully integrate Raspberry Pis into the USMA test bed to further investigate our proposed decentralized architecture, and explore the use of libraries and languages such as FastFlow [16] and Go [17]. Since the performance of our anomaly detection procedure falls well within the time requirements for real-time, we also plan to explore more complex forms of detection and analysis, such as the use of Discrete Fourier Transform (DFT). We anticipate that with more complex analyses, we can begin to leverage threading to utilize the multiple cores of the Raspberry Pi. In our current implementation, while parallelism does offer some improvements, experimentation has shown that the extremely fast run times made the overall benefits negligible.

Lastly, we note that as SBC technology improves, so will the processing capabilities, power efficiency and form factor. For example, the recently announced Raspberry Pi Zero W [18] features a 1 Ghz processor, 512 MB of RAM, and costs $\$10.00$. As the ecosystem of SBCs continues to evolve, we plan to explore the utility of other SBCs for smart grid analysis, including the use for anomaly detection and other applications such as smart meters.

Acknowledgments

Funding for the SBC hardware and algorithmic development was provided by the Office of Naval Research. Funding for the initial algorithm development was provided by U.S. Army Armament Research, Development and Engineering Center (ARDEC). The opinions in this work are solely of the authors and do not necessarily reflect those of the U.S. Military Academy, the U.S. Army, or the Department of Defense.

References

[1] NERC, "Real-time application of synchrophasors for improving realiability," *Tech. Rep. 1*, 2010.

[2] 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.

[3] S. K. Soonee, V. K. Agrawal, P. K. Agarwal, S. R. Narasimhan, and M. S. Thomas, "The view from the wide side: Wide-area monitoring systems in india," *IEEE Power and Energy Magazine*, vol. 13, no. 5, pp. 49–59, Sept 2015.

[4] R. Gore and S. P. Valsan, "Big data challenges in smart grid iot (wams) deployment," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2016, pp. 1–6.

[5] M. N. Aravind, L. S. Anju, and R. Sunitha, "Application of compressed sampling to overcome big data issues in synchrophasors," in *2016 IEEE 6th International Conference on Power Systems (ICPS)*, March 2016, pp. 1–5.

[6] "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.

[7] 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, Sept 2016.

[8] Raspberry Pi Foundation. (2016) Raspberry pi 3 model b. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[9] M. Winkler, K.-D. Tuchs, K. Hughes, and G. Barclay, "Theoretical and practical aspects of military wireless sensor networks," *Journal of Telecommunications and Information Technology*, pp. 37–45, 2008.

[10] C. Bell, *Beginning sensor networks with Arduino and Raspberry Pi*. Apress, 2014.

[11] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, 2007.

[12] G. Da Costa, "Heterogeneity: The key to achieve power-proportional computing," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013, pp. 656–662.

[13] S. J. Matthews and A. St. Leger, "Leveraging mapreduce and synchrophasors for real-time anomaly detection in the smart grid," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[14] 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.

[15] P3 International. (2005) Kill a watt - electricity usage monitor [p3]. [Online]. Available: http://www.p3international.com/products/p4400.html

[16] M. Aldinucci, M. Danelutto, P. Kilpatrick, and M. Torquati, "Fastflow: high-level and efficient streaming on multi-core," *Programming multi-core and many-core computing systems, parallel and distributed computing*, 2014.

[17] Google. (2012) The Go programming language. [Online]. Available: https://golang.org/project/

[18] Raspberry Pi Foundation. (2017) Raspberry pi zero w. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-zero-w/