The high rate of data samples reported by devices that support PMU functionality forces the use of non-traditional methods in order to attempt realtime anomaly detection. Two methods discussed are offline machine learning and a realtime sliding window procedure. In using machine learning techniques it is possible to assert a classifier algorithm, which to a certain degree of accuracy can flag incoming data for further operation when applied in realtime. The open source project Hadoop provides the storage architecture for large datasets (petabyte scale) as well as the MapReduce computational framework for distributed computing to produce these classifiers. Additionally, a sliding window of realtime data can be used to present a longer data sample window than the device report rate allowing for a heuristic hysteresis approach. The open source openPDC promotes the implementation of the classifier and sliding window in a realtime environment operating on new measurements thirty times a second.

openPDC, synchrophasor, Hadoop, PMU

I. INTRODUCTION

OVER the last 20 years the advent of IP-based high speed communication lines connecting the substation to the control center has enabled the utilization of a string of technologies related to reliability and monitoring. Synchrophasor technology has consequently enjoyed a proliferation due to the removal of bandwidth limitations and the steady installation of PMUs and/or integrated support in multifunction devices such as relays. The openPDC was developed to take advantage of this new ecosystem and allows for realtime event detection based on this synchrophasor data. In contrast the Hadoop Map Reduce model incorporates a batch-based massively parallel approach to consuming petabyte scale datasets of historic synchrophasor data. Instance learning is examined for use in Hadoop and windowing techniques are incorporated into the openPDC for its event detection.

II. openPDC AND REAL-TIME EVENT DETECTION

A. openPDC Background

The openPDC is an evolution of the original SuperPDC developed by TVA for the Eastern Interconnect Phasor Project. Designed to be vendor agnostic and parse all the major communication protocols in use (IEEE C37.118 [1], IEEE 1344 [2], BPA PDCStream, FNet, Macodyne, SEL Fast Message, etc.) the software project began in 2004 and reached 120 simultaneous PMU streams from 17 companies by 2008. As the costs and maintenance had steadily risen over the course of the project NERC presented TVA with a 5 year contract for continued maintenance of the then-current system as well as development of a Generation 2 system. This Generation 2 system is known as the openPDC.

B. openPDC Architecture

The openPDC consists of three layers, each delegated to a specific task. The first layer is the Input Layer, the second the Action Layer, and the third the Output Layer (see Fig. 1). Each layer is an extension of a modular framework whereby plugins can be inserted to discretely perform a specific function.

Fig. 1. openPDC architecture.

In the first of these layers, the Input Layer, a TVA-provided Phasor Protocol module allows the parsing of the differing communication protocols, stripping off their proprietary format thus reducing them to the atomic unit of measurements. These measurements are then passed along to the second layer, the Action Layer, whereby additional user defined modules can perform specific tasks upon the measurements. One of the most important tasks completed is that of concentration. This module sorts the measurements by their...
associated GPS timestamps as provided at the time of measurement by the PMU device located at the substation. Once a group of measurements with similar time stamps is identified it is assembled into a “bucket” correspondingly tagged with the time slice identified. Commonly PMUs report synchrophasor measurements at a rate of once every two cycles or thirty samples a second. In this case thirty of these “buckets” will be presented to the Output Layer (or another Action Adapter) for publishing as a stream encoded by any of the supported input formats or as a binary stream.

Each of these layers is available to the user or developer who can extend the functionality transparently. If a data stream is using a protocol not currently supported, a module can be written to parse measurements from such a stream and then be plugged in to the Input Layer as an Input Adapter. The subsequent layers of Action and Output will operate in the same manner as before with no alteration. Additionally, an Action Adapter can be added to the Action Layer to provide desired functionality such as MW/MVAR calculation, flat line detection, or any other function desired. It is expected that as these action adapters grow in complexity the resource allocation will require the use of a tiered architecture. In this tiered architecture those adapters which require high CPU cycles will be split into a separate hardware unit as would those requiring I/O intensive operations. It is essential in the maintaining of a realtime approach that parallelization be used in as many avenues as available in order to prevent timing dependencies or congestion from reducing the efficacy of the system.

In the following section we will discuss the OMS Action Adapter developed by TVA in conjunction with Washington State University.

C. openPDC Event Detection

The OMS (Oscillation Monitoring System) is a custom Action Adapter whose algorithms were developed by WSU from 2006-2007. [3] In 2008 TVA developed the current system based on the WSU prototype. Currently it has been plugged into the openPDC framework to provide realtime modal and dampening analysis on synchrophasor data. A rolling window of 7200 samples (four minutes of data at thirty samples a second) is used as the operating data. Each sample of data is correlated to its neighbors, and then an FFT is performed to move the data to the frequency domain. Single value decomposition (SVD) then determines the modes of the data and produces data points for visualization as seen in the top graph of Fig. 2. [4]

These modes provide a key step in understand grid dynamics as the low frequency modes (0.2 - 0.45) indicate inter-area power flow swings while the higher frequency modes represent local area fluctuations.

Experienced operators are able to recognize via inculcation the common modes of their system and concentrate only on those which seem aberrant. In the TVA local area the most common modes are from 0.2-0.4 and thus anything else can be an item of study. An interesting case which arose from monitoring the local modes was the indication that an installed power stability system (PSS) unit had reached its operating limit and was ceasing to perform its duty under certain conditions. In Fig. 3 the MW output of a large fossil plant is graphed against a binary interpretation of high and low modal characteristics. As can be seen, the local area mode changed from low to high only when the production of the plant exceeded 2250 MW. This led to an investigation and replacement of the PSS unit and improved the area’s reliability.

Fig. 2. OMS Display showing current system modes.

In addition to modal analysis, the realtime OMS system provides a dampening visualization to indicate the distance to danger in respect to undamped oscillations. This is calculated using Prony analysis on a specific subset of the data. As the oscillation dampening becomes smaller, increasingly dangerous characteristics begin to surface. This system (pictured in Fig. 4 below) allows for realtime monitoring of these trends.

Fig. 3. Megawatt output correlated to modal analysis.

This data is calculated from the peak generated by the modal analysis done by SVD. Once these global peaks are discovered the two surrounding valleys are mathematically
calculated and the data from valley to valley is converted back to the time domain via an inverted FFT function. Prony analysis is then performed on this data to produce the damping data points.

While these two displays have been found to be extremely useful, one limitation is the small window of four minutes in which the data is available. The following section addresses a different approach to event detection from a “Big Data” perspective utilizing the distributed Map Reduce approach of the Hadoop framework.

III. HADOOP AND HISTORICAL EVENT DETECTION

As the expected growth rate of synchrophasor adoption and installation may exceed 10x the current number (anticipated 10 terabytes a month by 2012) the problem of how to store and analyze large datasets came to the forefront of engineering design. TVA has chosen the Hadoop project to handle these concerns due to its scalability, redundancy, and cost-effectiveness.

A. Hadoop Background

Hadoop is a distributed computing and storage framework inspired by the Google File System (GFS) [5] and Google’s MapReduce [6]. Provided by the Apache Foundation as an open source project, it seeks to provide a highly reliable storage mechanism called the Hadoop Distributed File System (HDFS) combined with massively parallel computing possibilities, all of which run on commodity hardware. The project enjoys a thriving ecosystem of support from volunteer contributors and large commercial vendors [7] as well as an aggressive update cycle continually bringing improvements to the codebase.

B. Hadoop Architecture

The Hadoop Architecture is made up of two primary components, the Name Node and many Data Nodes. When data is moved to the Hadoop Cluster the cluster will make a user-definable number of copies of that data and store the copies on different Data Nodes. The Name Node retains an index of all locations where each copy of data is stored. The failure of a harddrive or, more severely, the server hosting a data node, will not cause the loss of data. If the Hadoop cluster concludes that a Data Node has failed it will determine via the index on the Name Node what copies of data have been lost and will automatically produce more copies to maintain the user defined number of replication. Additionally, a constantly running process will check for file corruption and will sound the need for a new copy should a file be found to have been damaged. This replication of data also allows for parallel data reads, greatly decreasing the time needed for pulling large data sets over a traditional sequential read from one device at a time (see Fig. 5).

As well as the data reliability and validity abilities of Hadoop, the project also provides the MapReduce framework for creating and running massively parallel batch processing jobs. The Hadoop cluster installed at TVA makes use of 180 processor cores simultaneously to vastly increase the speed and scope of data mining and corresponding event detection algorithms.

C. Hadoop and Event Detection

In contrast to the realtime event detection done by the openPDC, the Hadoop cluster is utilized as an analysis tool for extremely large historical datasets. The current TVA repository of phasor data spans 4 years and nearly 25 terabytes. Along with a different technology, a different approach is required to effectively cull the significant information from the large amount of ambient data. Instance-based learning can be used to “train” Hadoop to recognize a specific pattern, ignoring this noise and flagging interesting data sets for future in-depth study or indexing for fast retrieval later.

In instance-based learning a certain number of training examples are provided and a distance function is used to calculate which member of a training set is most highly correlated with, or closer to, the data being analyzed. There exist a large number of distance functions each with tradeoffs in performance, speed, and accuracy. A combination of SAX [7] (a more efficient version of Discrete Fourier Transforms) and the Euclidian distance function (1) can provide a normalized implementation of the simple 1 Nearest Neighbor algorithm to determine how closely relevant archived phasor data matches training examples for undamped oscillation, sudden load shed, islanding, etc.

\[
D(Q,C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \]  

where \(Q=q_1\ldots \ q_n\) and \(C=c_1\ldots \ c_n\) (1)

Although instance-based learning is simple to implement and is quite effective, often times it can be slow. Fortunately, techniques exist to optimize the execution time. The most clear way to locate which member of a training set is in fact closest to a desired test instance is to calculate the distances from every member of that training set and then use the smallest. In this case, the time taken to make a single prediction is a linear function of the number of training instances. Thus, the time taken to calculate is the product of the number of test and training sets. A more efficient method is to use a kind of metric tree known as a ball tree. [9]

In order to use a ball tree the nearest neighbor to the desired
point is located starting from the top of the tree. Once the leaf is found the closest point to the leaf which is still inside the ball is determined [9]. This provides the largest distance the search should then encompass. Then, if the distance from the desired point to the sibling’s center is greater than the desired point’s radius in addition to the upper bound, it cannot contain a closer point and the search is complete (Fig. 6). Using this method combined with the reduced dimensionality of the SAX method allows for a very quick search when comparing a large amount of data to a large number of possible training sets.

![Ball tree representation. The closest point to a white ball within the area of a grey ball represents the upper bound of the distance.](image)

This approach has currently produced a one-to-one correspondence of classifier to sample test data where concerned with undamped oscillations. Simple standard deviation calculations have also proven to be extremely effective in determining oscillation events while interesting issues of time dilation or warping with respect to the periodicity of those oscillations present notable challenges for the future. Tuning and optimization plays a very large role in the length of time needed to achieve these numbers and a large amount of energy will have to be expended in that realm. Tests run concerning standard deviation and Euclidean distance on sample hardware less powerful than a single Data Node revealed an encouraging speed of calculation (see Table I), rivaling realtime metrics.

<table>
<thead>
<tr>
<th>Std Dev (SAX)</th>
<th>Euclidean (SAX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.00% - 0.0110 seconds</td>
<td>100.00% - 0.0062 seconds</td>
</tr>
<tr>
<td>100.00% - 0.0047 seconds</td>
<td>100.00% - 0.0016 seconds</td>
</tr>
<tr>
<td>100.00% - 0.0078 seconds</td>
<td>100.00% - 0.0047 seconds</td>
</tr>
<tr>
<td>100.00% - 0.0093 seconds</td>
<td>100.00% - 0.0048 seconds</td>
</tr>
<tr>
<td>100.00% - 0.0095 seconds</td>
<td>100.00% - 0.0046 seconds</td>
</tr>
<tr>
<td>100.00% - 0.0110 seconds</td>
<td>100.00% - 0.0045 seconds</td>
</tr>
<tr>
<td>100.00% - 0.0062 seconds</td>
<td>100.00% - 0.0033 seconds</td>
</tr>
<tr>
<td>100.00% - 0.0093 seconds</td>
<td>100.00% - 0.0045 seconds</td>
</tr>
</tbody>
</table>

After scanning through the 25 terabytes of available phasor data it is TVA’s intention to develop a searchable index of the found events, categorized via classifier. It has been shown that even in large datasets sharding of an index can produce extremely fast results for dynamic user interaction [10].

Only through Hadoop is it possible to find the intersection of an extremely large dataset in the same location as a massively parallel execution engine. In this manner, Hadoop actually moves the processing to the data instead of the classical paradigm of moving the data to the processing.

IV. CONCLUSION

Two approaches are expected to be commonly used in the analytical use of phasor data: realtime and batch. The openPDC provides a modular structure to incorporate any efficient algorithm which can complete within an increment of the report rate of the data providing device. As this is a function of resources this may in the future require the use of a tiered architecture for intensive algorithms. The current example of the OMS shows how a realtime detection process can help to diagnose issues such as PSS failure.

The second approach to phasor data is to have a much larger dataset provide extended context to inform the outcome of a desired analytical process. The size of the dataset forces the use of less common methods such as functional programming for the MapReduce language in Hadoop as well as a large amount of data mining expertise.

With these two approaches it is believed that significant knowledge can be obtained about heretofore unknown characteristics of the power grid and at a time when stability and reliability are of the greatest importance.

V. ACKNOWLEDGMENT

The authors gratefully acknowledge the contributions of Josh Patterson, Ryan Zuo, Galen Riley and Andy Hill for their hard work on the research, design, and implementation of these systems.

VI. REFERENCES

Paul Trachian was born in Atlanta, Georgia in 1982. He has majors in Electrical Engineering and Computer Engineering from the University of Tennessee at Chattanooga.

He has been employed at TVA for nine years and has been working in his current capacity as an operations engineer dealing with synchrophasor technology for last four years. Current projects include managing long term storage solutions such as Hadoop for phasor data as well as openPDC development and integration. Research into optimizations for analysis of synchrophasor data and improved communication techniques are planned for the future.