Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 61 (2015) 113 – 119

Procedia Computer Science

Complex Adaptive Systems, Publication 5
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2015-San Jose, CA

Real-Time Complex Event Processing and Analytics for Smart Grid

Guangyi Liu[a], Wendong Zhu[a], Chris Saunders[a], Feng Gao[a], Yang Yu[b]*

[a]*SGRI North America, 5451 Great America Pkwy #125, Santa Clara, CA 95054, USA*
[b]*Stanford University, Palo Alto, CA, USA*

Abstract

Electric power networks are among the world's most complex human-made systems. The developing smart grid is an inherently complex system which is rapidly evolving in both definition and implementation. Deployment of advanced technologies within the electric utility sector and usage of state-of-the-art computing systems provides companies with innovative capabilities to forecast electricity demand, influence customer usage patterns, create demand response program, optimize unit commitment, and prevent power outages. At the same time, these advances also lead to the generation of unprecedented data volumes, high data communications requirements, and increased system complexity. Utility companies must be capable of high-volume, high-speed data management and advanced analytics which are designed to transform data into actionable insights, if they strive to successfully implement a modern smart grid. As smart grid operations will leverage Advanced Metering Infrastructure (AMI) to drive more real time decision making and operational activities, complex event processing and stream computing are needed for the modern smart grid. This paper explores the challenges and benefits of transitioning to a smart grid, and explores new architectural approaches built on Lambda Architecture and other emerging software standards which may more effectively leverage established forms of complex event processing.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: Smart Grid, Big Data, Lambda Architecture, Complex Event Processing, Real-Time Analytics

1. Introduction

Using traditional electricity and gas meters which could only measure total consumption, utilities usually would collect energy consumption data once a month per customer. Today, Smart Meters and Smart Meter Systems are

* Corresponding author. Tel.: +1-408-727-6261
E-mail address: wendong.zhu@sgrina.com

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2015.09.169

being deployed throughout North America, and utilities are continuing their efforts to improve grid reliability and promote energy efficiency while providing improved services to their customers. The combination of the electronic meters with two-way communications technology for information, monitoring, and control is commonly referred to as Advanced Metering Infrastructure (AMI)[1], with data collection intervals of 15 minutes. With the deployment of AMI, the level of data collection has been increased dramatically, usually in the order of terabytes referred to as "Big Data"[2].

This is just one element of what is currently termed "Big Data", where datasets are terabytes to exabytes in size and come from a variety of sources. The management and analysis of Big Data is beyond the scope of traditional IT tools. From traditional meter readings per month to smart meter readings every 15 minutes, there are 96 million reads per day for every million meters. The result is a 3,000-fold increase in data which can be overwhelming[9].

"Big Data" thus typically refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze[2]. For utilities, Big Data opens up new opportunities, allowing them to do things they never could before. Data gathered from smart meters can provide better understanding of customer segmentation, behavioral patterns, and how pricing influences usage. In addition, it is possible to transform the electrical network and significantly improve the efficiency of electrical generation and scheduling. Big Data can turn the traditional network into smart networks that understand individual energy consumption. This will increase efficiencies, lower electricity price, and reduce the global carbon footprint.

For modern IT systems, the ability for the timely reaction to the occurrence of real-world situations in the system environment has become a fundamental requirement. This applies to many different applications, e.g., in smart energy grids, automatic stock trading, logistics, and production control. For example, in a smart grid scenario, the detection of a divergence between the energy consumption and the energy production can enable the rapid deployment of an intelligent Demand Response system, adapting the energy demand of intelligent appliances to the energy production, which reduces the demand for operating reserve provided in expensive supplemental power plants. In such applications, incoming data streams of low level information arrive from heterogeneous sources at high rates, and need to be processed in real-time in order to detect more complex situations. Those data streams can be bursty, and their rates fluctuate tremendously.

To tackle the problem, the paradigm of Real-Time Complex Event Processing (CEP) has emerged as a favorable approach. Complex Event Processing (CEP) is a new Big Data analytics technique which takes data as unordered series of events coming from different sources. CEP has found wide uses in industries such as financial systems, homeland security, and sensor data processing. In each of these cases, the common element is that data from edge devices must be processed "on the fly", whether it comes in streams or asynchronous bursts. The CEP technology is capable of applying complex queries to multiple data streams simultaneously to detect specified conditions ("events"), thus triggering appropriate actions in real time.

CEP can be used in a variety of utility business functions. These include meter data management, demand response, fault detection, outage management, billing, and remote equipment monitoring, etc. CEP is a flexible tool, and when included in an overall data management strategy and architecture, it can tremendously improve the flexibility needed to implement the data management solutions for Smart Grid.

2. Lambda Architecture

Any system handling data requires some formats of data storage. For decades, the term 'database' has been more or less synonymous with Relational Database Management Systems (RDBMS), which are widely used in utilities. Recently there has been increased adoption of Not Only SQL (NoSQL) databases with the primary motivation being horizontal scaling to handle Big Data.

The database systems, including RDBMS and NoSQL, are not designed to be resilient. There have been various solutions to achieve high availability of database systems, such as backup, failover, etc. Such solutions to achieve high availability or resilience are typically expensive and complex.

According to CAP theorem[3], it is impossible to for a distributed computer system to simultaneously provide Consistency, Availability and Partition Tolerance. In modern IT systems, Partition Tolerance is usually required. Therefore, when designing a data management system, a decision on trade-off between Consistency and Availability

has to be made. Consequently, we can view that RDBMSs favor Consistency over Availability; while NoSQL databases favor Availability over Consistency.

The Lambda Architecture[4], proposed by Nathan Marz, is designed to address the issues mentioned above. It handles massive quantities of data by taking advantage of both batch- and stream-processing methods. It attempts to balance latency, throughput, and fault-tolerance by using batch processing to provide comprehensive and accurate pre-computed views, while simultaneously using real-time stream processing to provide dynamic views. The Lambda Architecture has three major components: the Batch layer, the Serving layer, and the Speed Layer. The Batch layer manages the immutable master dataset, and pre-computes batch views, whereas the Serving layer indexes batch views and loads them as needed, and the Speed layer handles new data and updated real-time views.

A new Big Data processing system for Smart Grids based on Lambda Architecture is proposed.

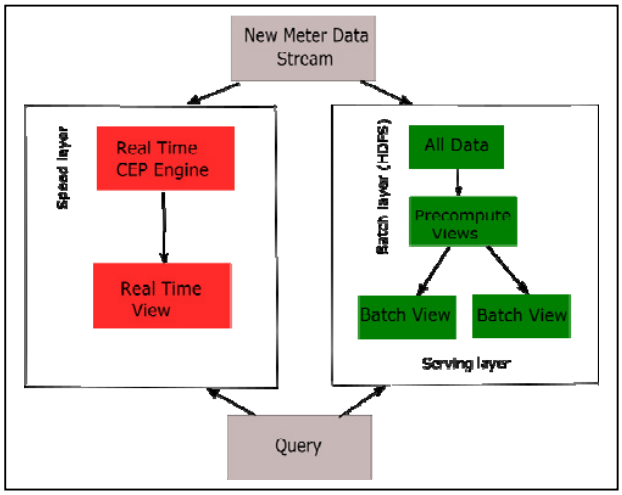

Fig. 1. Lambda Architecture of Smart Grid Big Data processing system.

Fig. 1 illustrates the proposed system. New AMI data is fed to both the master immutable data store in the Batch layer and the Real-Time CEP engine in the Speed layer. The Batch and Serving layers are robust and also human fault tolerant. In the event of machine problems or human error, batch views can be recomputed from scratch. The Batch and Serving layers are also scalable and extensible, and they can be implemented as fully distributed systems. The Speed layer handles fresh real-time data, and the results are combined with those in the Serving layer for querying by users.

The Master data store in the Batch layer can take advantage of any relational databases, or NoSQL databases. Many utility applications, such as billing, load forecast, energy management, etc., fit into the Batch layer. The Speed layer contains the real-time CEP engine. We will discuss the implementation of the Speed layer in the following sections.

3. Complex Event Processing

A power system is inherently complex due to the dynamic nature of power generation equipment, the use of complex technologies, the long distance of electric power transportation, and the instantaneous balance of production and consumption. Electric power networks are among the world's most complex human-made systems. The problems existing in power system fit naturally within the paradigm of Complex Event Processing.

Complex Event Processing (CEP) is the use of technologies to track streams of data from multiple sources, to analyze trends, patterns, and events in real time in order to respond to them as quickly as possible. It can be further

leveraged to monitor diverse and disparate data sources or events, brining organizations enhanced situational knowledge and increased business agility.

CEP allows users to access events that happened in the past and use them in any order. The event can come from various sources and may occur over a long period of time.[5,6] CEP requires sophisticated event interpreters, event pattern definition and matching along with correlation techniques.

With CEP, incoming data is being continuously monitored and acted upon using declarative conditions. Moreover, the data monitoring and processing works at a near-zero latency. Different events may come from different sources, and the CEP system can assemble a complex event to internally model the components as one object.

A CEP system is aimed at solving the velocity problem of big data, while data comes as a stream of predefined events. The sliding window approach used by CEP systems ensures that only a portion of actual data simultaneously passes into the main memory, whereas the old events may be discarded or archived. This way, all the data does not have to fit into system memory, but still the most recent events can be efficiently analyzed.

In a Smart Grid, the data sources may include PMUs (typically 2880 samples per second), SCADA (typically acquiring data every 2 to 5 seconds), AMI (typically acquiring data every 1 to 15 minutes), weather data, and third-party data, etc. Since data is transferred at a steady high-speed rate, the input data may be treated as streams. The data is continuously evaluated by queries. Figure 2 describes the system architecture of CEP.

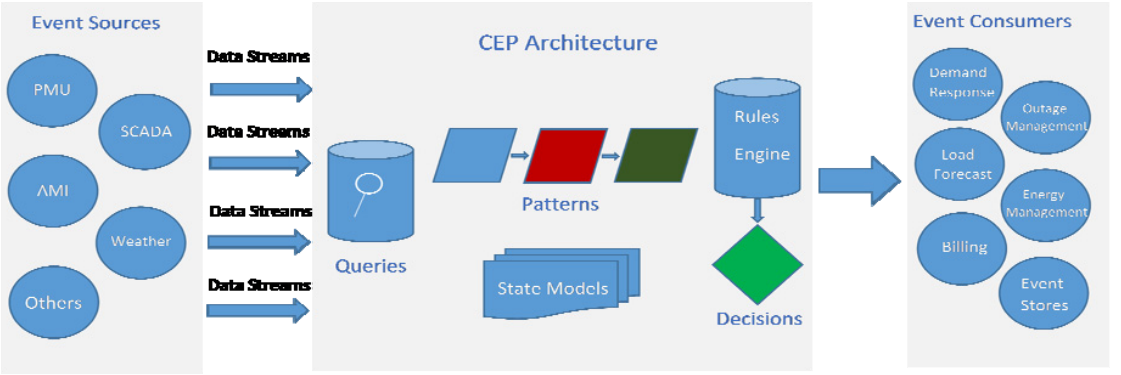

Fig. 2. Complex Event Processing System Architecture

CEP systems are continuously developing and evolving. The state-of-the-art CEP takes advantage of stream computing to handle unstructured data and large numbers of business events per second. CEP pushes data through highly sophisticated analytics processes to deliver real-time analytics results on data-in-motion, to help increase responsiveness when dealing with high levels of input. It enables both descriptive (simple) and predictive (sophisticated) analytics to support real-time decisions. Essentially, CEP enables organizations to capture and analyze as much data as they can handle at any time and in near-real time.

Complex event processing will be required to support AMI and Smart Grid needs as grid operations will leverage AMI infrastructure to drive more real time decision making and operational activities.

4. Spark Streaming-based Real-Time CEP for Smart Grid

In this paper we propose a Spark Streaming-based Real-Time CEP for the Smart Grid. A number of commercial and open source event processing software is available to developers who are building complex event processing applications. Spark Streaming is being evaluated in this paper due to its open source nature and combination of streaming with both batch and interactive queries.

Apache Spark is a fast and general engine for large-scale data processing[7]. It is a unified platform combining Spark SQL, Spark Streaming, MLLib for machine learning and GraphX. Spark now boasts the ability to not only

process streams of data at scale, but to "query" that data at scale using SQL-like syntax. This ability makes Spark a viable alternative to established Complex Event Processing platforms and provides advantages over other open source stream processing systems. Especially with regards to the former, Spark will now allow for the creation of "rules" that can run within stream "windows" of time and make decisions with the ease of SQL queries. This is a remarkably powerful combination.

Spark Streaming is an interesting extension to Spark that adds support for continuous stream processing. All the strengths of Spark's unified programming model apply to Spark Streaming, which is particularly relevant for real-time analytics that combine historical data with newly collected data.

Spark Streaming ingests data from any source, including file systems such as S3 and HDFS. Users can express sophisticated algorithms easily using high-level functions to process the data streams. The core innovation behind Spark Streaming is to treat streaming computations as a series of deterministic micro-batch computations on small time intervals, executed using Spark's distributed data processing framework. Micro-batching unifies the programming model of streaming with that of batch use cases and enables strong fault recovery guarantees while retaining high performance. The processed data can then be stored in any file system (including HDFS), database (including Hbase), or live dashboards.

Our proposed Real-Time CEP engine for Smart Grid is based on Spark Streaming and Spark SQL.

4.1. Implementation

The fundamental abstraction in Spark is Resilient Distributed Datasets (RDDs)[8], which is an immutable and partitioned collection of elements that can be operated on in parallel. Spark supports a rich set of higher-level tools such as Spark Streaming that allows for continuous processing via short interval batches. The basic abstraction in Spark Streaming is Discretized Stream (or a DStream, a continuous series of RDDs) which represents a continuous stream of data.

In our implementation, AMI data from a collection of residential consumers is represented as DStream. The Real-Time CEP engine has a core set of operations: Windowing, Transformation, Aggregation/Grouping, Merging/Union, Filtering(Selection/Projection), Sorting/Ranking, and Join.

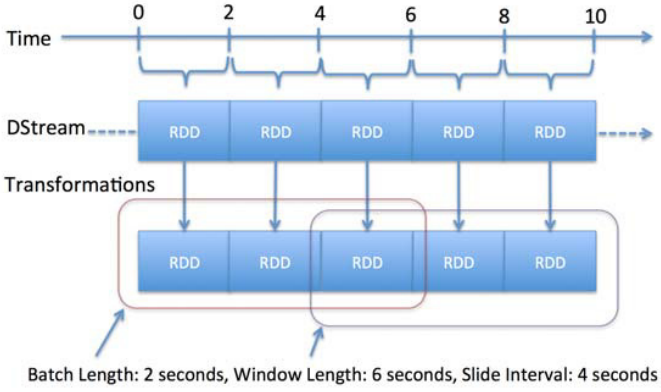

Fig. 3. Sliding Window in CEP

Fig. 3 shows a sliding window of AMI data in our CEP. In the above diagram, the batch length is two seconds, with a window length of three batches (six seconds), and window sliding interval of two batches (four seconds).

To calculate CEP aggregate functions such as Average, Max, Min, etc., we use operations countByWindow, countByValueAndWindow, reduceByWindow, reduceByKeyAndWindow provided by Spark Streaming.

Here is an example of creating a sliding window to aggregate data:

```
val AMIdataStream = data.reduceByWindow((rdd1: Array[String], rdd2: Array[String]) =>
                                        max(rdd1, rdd2), Seconds(6), Seconds(4))
```

The window length is six seconds and sliding interval is four seconds. The method *max()* is user-defined, and the returned result *AMIdataStream* is a new stream by applying aggregate function upon AMI data over a sliding interval. The above example is written in Scala to calculate the maximum value of AMI data during a sliding window of six seconds. Ideally, the window length should be as long as possible, while the sliding interval is close to batch length.

4.2. CEP Rules

Rules are important part of CEP since decisions and actions are made according to rules. Unlike many CEP vendors where rules are specified by vendor-specific languages, rules can be conveniently expressed using SQL in our implementation of a Real-Time CEP engine. This is advantageous since SQL is a standard and a common skillset among software developers.

For instance, to create a rule "For any voltage reading above nominal value by 5%, find the meter ID and trigger an alert", we can specify it as:

*SELECT meter_id, timestamp, voltage FROM meter_data WHERE voltage > 1.05 * NOMINAL_VOLTAGE;*

followed by sending an alert.

5. Experimental Results

Experiments were run on a Linux (CentOS 7) Server with 32 Cores and 128G RAM. We used Gridlab-D, a power distribution system simulation application to generate meter reading data, such as power, current, and voltage readings. Total 1594 customers/meters were used. For each meter and each day, we collected 96 points of data. Three datasets were used in the experiments, as explained in Table 1.

Table 1. Description of Datasets.

	Number of Meters	Collecting Duration (Days)	Number of Records
Dataset 1	1,594	1	153,024
Dataset 2	1,594	38	5,814,912
Dataset 3	1,594	365	55,853,760

Performance was measured on the Real-Time CEP. We tested aggregate functions such as max/min, daily, weekly, monthly average over the whole dataset with incoming data of six-second intervals. Fig. 4 shows the results.

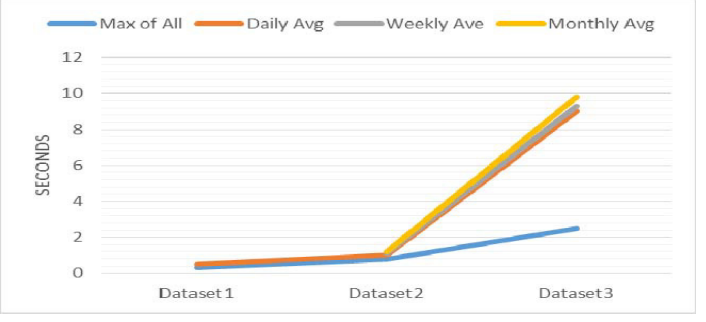

Fig. 4. Performance of aggregate functions

It takes less than 10 seconds to calculate aggregate functions such as average over the dataset of more than 55 million records in our test. Further performance tuning can be done on the system.

6. Conclusions

Today, many utilities are moving to smart meters and smart grids to ensure a reliable energy supply, improve power grid efficiency, and meet the rapidly increasing electricity demands of their customers. With utilities growing in their usage of Big Data, and the desire to respond in real time, utilities will inevitably leverage the CEP paradigm. Complex Event Processing is a key element of the future of work that could be applied broadly by electric utilities. By utilizing state-of-the-art streaming computing technologies, Real-Time CEP will help utilities with energy efficiency programs and grid management. We propose a new Big Data processing system for a Smart Grid, based on the Lambda Architecture, where a Real-Time CEP engine is embedded in the Speed layer. We also explore an example of implementation of Real-Time CEP using Spark Streaming, and demonstrated the efficiency of our implementation.

Acknowledgements

The research work was supported by Science and Technology Research Foundation for smart distribution big data analysis from SGCC.

References

1. Edison Electric Institute (EEI), Smart Meters and Smart Meter Systems: A Metering Industry Perspective. Washington, D.C.; March 2011.
2. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute, June 2011.
3. Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News, Volume 33 Issue 2 (2002), pg. 51-59
4. Marz, Nathan; Warren, James. Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Manning Publications, 2015.
5. K.Chandy. Event-driven applications: Cost, benefuts and design approaches. Gartner application integration and web services; 2006
6. B Michelson. Event-driven architecture overview: Event-Driven SOA is just part of the EDA story. Patricia Seybold Group; February 2006
7. Spark official website: https://spark.apache.org/
8. Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, pages 423–438. ACM, 2013.
9. Nanpeng Yu, Sunil Shah, Raymond Johnson, Robert Sherick, Mingguo Hong and Kenneth Loparo. Big Data Analytics in Power Distribution Systems. To appear in IEEE ISGT 2015.