Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 19 (2015) 858 – 865

Procedia Technology

8th International Conference Interdisciplinarity in Engineering, INTER-ENG 2014, 9-10 October 2014, Tirgu-Mures, Romania

A Brief Survey on Smart Grid data analysis in the Cloud

Adela Bereş*, Béla Genge, Istvan Kiss

"Petru Maior" University of Tg. Mureş, N.Iorga St., No. 1, Tg.Mureş, 540088, Romania

Abstract

Cloud Computing has become more and more of a choice for storing and processing data due to the performance, scalability, availability and interoperability that it offers. The adoption of generic off-the-shelf Information and Communication Technologies (ICT) for the Smart Grid has come with many advantages, but also raised issues regarding the security, availability and reliability of the Smart Grid. The data generated by the Smart Grid's systems has the properties of a time series and its analysis needs to be secure and in real-time. As shown in this paper, cloud computing has become a real candidate for Smart Grid data analysis. We present the particularities of the data generated in the Smart Grid and the operations involved in its analysis. As well we compare existing solutions for stream data processing against this basic set of operations.

Peer-review under responsibility of "Petru Maior" University of Tirgu Mures, Faculty of Engineering

Keywords: Smart Grid; cloud computing; time series analysis; stream data

1. Introduction

According to the National Institute of Standards and Technology (NIST) "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. On-demand self-service, broad network access, resource pooling, rapid elasticity and measured service are the characteristics that Cloud Computing is offering.

* Corresponding author. Tel.: +40-0265-241680
E-mail address: adela.beres@stud.upm.ro, bela.genge@ing.upm.ro, mailto:istvan.kiss@stud.upm.ro

Peer-review under responsibility of "Petru Maior" University of Tirgu Mures, Faculty of Engineering
doi:10.1016/j.protcy.2015.02.123

These make it a powerful candidate for storing and processing Smart Grid data. Also its high performance and availability satisfy the demand for real-time analysis on the data.

Smart Grid, characterized as the next generation power grid, is more and more present in the day-to-day life mainly in the electricity field. The adoption of generic off-the-shelf ICT came with the benefits of two-way communication, control and reliability. But the data generated by the Smart Grid needs to be securely stored and processed and its analysis done real-time. A particularity of the Smart Grid data is that it has the properties of a time series so the operations performed on it are very specific.

Briefly, in this paper we present the particularities of the properties of the Smart Grid data and of the operations performed on it and we compare existing solutions for stream data processing.

The rest of the paper is organized as follows. Section 2 describes the properties of the Smart Grid data. Section 3 depicts the operations that can be performed on the Smart Grid data. Section 4 discusses the existing solutions for data stream analysis. Section 5 presents related work done by other researches. Conclusions are drawn in Section 6.

2. Smart Grid data properties

Smart Grid is the next generation power grid. The integration of ICT within the Smart Grid has made it more reliable, secure and enabled two-way communication. The data flow in the Smart Grid starts from the power plants and physical devices like smart meters, intelligent electrical devices or monitors. This data is then streamed to the control systems which monitor in real-time the status of the grid and also store it so it can be later processed and used for forecasting or historical investigations. The flow of the data is described in Fig. 1:

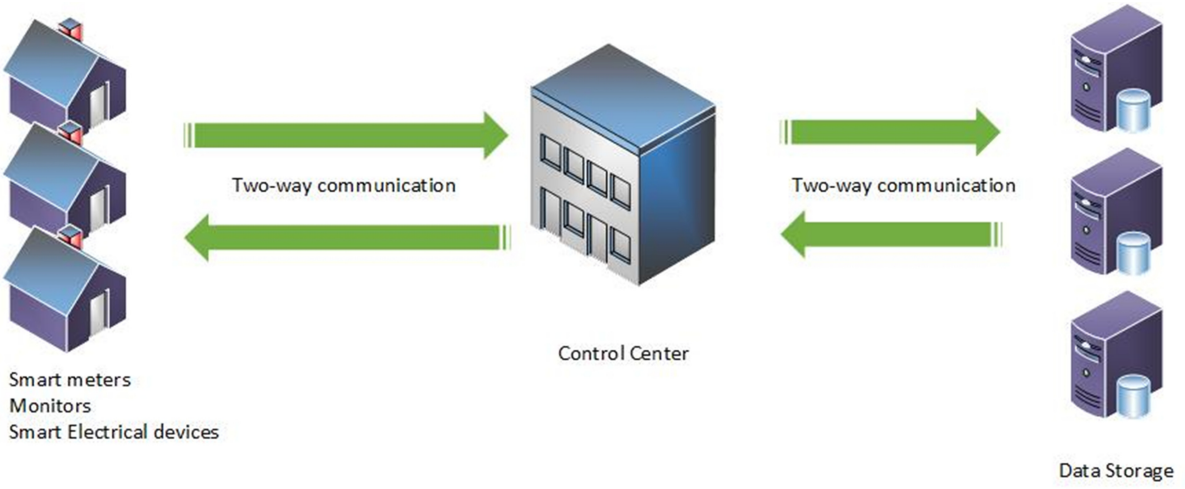

Fig. 1. Smart Grid data flow

The physical devices generate the data with a timestamp and then stream it through the communication infrastructure. This data can be defined as a time series or a data stream. A data stream is assumed to be an infinite sequence of time-stamped records. Each record consists of key-value pairs, where the keys are the attributes of the reading, and the values are the corresponding data of the reading [2]. Taking this into account the properties of the Smart Grid data are:

- Heterogeneous – diversity is due to device types, interfaces, capabilities.
- Time-stamped – the date and time when the data is generated is recorded in the value being streamed.
- High data generation rate – in Smart Grid, stream elements are generated at a rapid rate. The elements thus have to be processed in a timely manner in order to keep up with the stream rate. Usually a single scan of such data stream is necessary [3].

- Unboundedness – data streams are potentially unbounded and can thus generate an infinite amount of data. That means it is not possible to store stream entirely [3].
- Evolving nature –the characteristics of the data stream as well as its elements evolve over time. This property is referred to as temporal locality and adds an inherent temporal component to the data stream mining process. Stream elements should thus be analyzed in a time-aware manner to accommodate the changes in stream characteristics [3]. Also special events can cause structural breaks in the data – these events can be caused by attacks or malware.
- Unordered – even though the data is time-stamped it's not always streamed in the order it was generated.

3. Operations on Smart Grid data

After researching the existing solutions for stream data analysis we defined a basic set of operations that a Cloud-based system for processing Smart Grid data should implement:

- Storage – it should provide a storage place for the data. Taking into account the rate at which the data is streamed, a Cloud-based system could provide the necessary space, but archiving or deletion of old/unnecessary data should be considered.
- Indexing – such a system will deal with a large volume of stream data so an efficient indexing algorithm should be implemented to easily find the items needed.
- Aggregation – queries on data may be very complex. Data should be processed in a batch-oriented way. Aggregate computation is very useful in such cases.
- Clustering – to reduce the amount of memory and time needed for processing the Smart Grid data, a Cloud-based system should implement a clustering algorithm.
- Sampling – this operation is important for efficiently returning a sample of the stream data in the case of historical forensics or in case of an attack or detection of tampering with the data.
- Searching – mining is a common operation performed on data streams. This should be efficient as well as secure in the case of Smart Grid data which contains sensitive information.
- Auditing – any operation performed on the data should be logged not only for historic purposes but also to detect and prevent harmful attacks or unauthorized access.

4. Stream data analysis solutions

In this section we present several existing data stream analysis solutions.

4.1. ElasticStream

A. Ishii and T. Suzumura from the Tokyo Institute of Technology have implemented a prototype system using Amazon EC2 and IBM System S stream computing which transfers data stream processing to a Cloud environment in response to the changes of the data rate in the input data stream [4].

ElasticStream, the prototype system, has three components: one that receives the data, the second one that splits the data over multiple nodes and the third one that processes the data in parallel. The economic cost is minimized by formulating a trade-off between latency and economic cost in a Service Level Agreement (SLA).

ElasticStream was tested on two types of applications: "Data-Parallel Application" that distributes a data stream to multiple nodes and computes in parallel, and "Task-Parallel Application" that distributes a computation process to multiple nodes [4]. Based on the performance evaluation done by the authors ElasticStream kept the latency low and also reduced the economic costs with 80% compared with a local system implementation.

4.2. SenQ

SenQ is a multi-layer embedded query system for interactive wireless sensor networks (IWSN) which enables user-driven and peer-to-peer in-network query issues [5]. It was developed at the University of Virginia.

SenQ has a modular architecture with layers that can be optional, separated or even placed on the physical devices. These layers are [5]:

- Sensor Sampling and Processing – this uses three types of sensors: EventSensor, SplitPhaseSensor and PollableSensor which read and convert the data. This layer also performs spatial aggregation on the collected data.
- Query Processing and Network Messaging – this layer processes the queries issued on the data. It also includes a Discovery component that allows locating nearby devices or other processing modules.
- Query Management and Data Storage – this layer provides services for connectivity and data analysis.
- SenQL – provides a declarative query language which uses a constraint subset of SQL-99.

SenQ's efficiency and performance was evaluated in a testbed for assisted-living. SenQ reduced the network load and energy consumption by using temporal and spatial aggregation.

4.3. Aurora

Aurora is a system for data stream management developed at Brandeis University, Brown University and M.I.T [6].

Aurora is meant to replace the traditional Database Management Systems (DBMS) which couldn't handle the processing and storing of data generated by monitoring applications or sensors. The input data streams are processed using the popular boxes and arrows paradigm found in most process flow and workflow systems. Aurora can also maintain historical storage, primarily in order to support ad hoc queries [6].

Aurora also has its own query language, SQuAl, which supports operators like select, split, union, aggregate, resample. The queries are dynamically optimized based on run-time statistics.

Aurora's architecture includes a storage manager, scheduler, Quality of Service (QoS) monitor and a load shedder. QoS attributes include response time, tuple drops and values produced. The authors focused on run-time data storage and processing issues, discussing storage organization, real-time scheduling, introspection, and load shedding [6].

4.4. Splunk

L. Bitincka et al. propose Splunk, a semi-structured time series database that can be used to index, search and analyse massive heterogeneous datasets [7]. Their work is based on the observations collected during their experience with a wide variety of datasets. The authors came to the conclusion that time is the essence and it best correlates heterogeneous data which is hard to fit into a traditional Sql database [7].

The authors use a MapReduce divide and conquer algorithm with indexed data stores and use different ways to optimize the data format like de-normalization, timestamp or keyword indexes. Also a specific search language was developed based on the Unix concept of pipes and commands which speeds up the search through the data.

Splunk was successfully used in real world environments: an IT powerhouse for a major industry and an IT organization at a University.

4.5. Stream

Stream is a general purpose tool that includes modelling and simulating data streams as well as an extensible framework for implementing, interfacing and experimenting with algorithms for various data stream mining tasks like clustering. Stream framework has two main components: Data Stream Data (DSD) and Data Stream Task (DST) [8].

Data Stream Data manages or creates streams with static structure or concept drift. Data can also be read from a file. The Data Stream Task processes the data and performs clustering tasks.

This paper is a very technical paper, describing the classes, methods and algorithms implemented in the Stream framework with examples written in R.

4.6. Indexing data streams solution

K. Q. Pu et al. index structure uses bitmap based techniques to efficiently sketch the structures to allow space-efficient lossless archiving of the data stream. It also allows very fast query processing on the archived data stream. Furthermore, it also adapts to structural evolutions of the stream to ensure good indexing and querying performance both in space and time [2].

The storing of the data is done both as a row in a relational database and as a key-value record in a flat file. It also groups segments of data into sections so it gets stored more homogeneous. Performance tuning is done based on attribute expiration and extra bit allocation.

The authors also describe a structural clustering algorithm to group the incoming data stream into multiple sub-streams each of which is more structurally homogeneous. By grouping similarly structured data records into the same section, the number of sections is kept small without sacrificing efficiency [2].

4.7. Multilevel secure data stream processing

In their paper, R. Adaikkalavan et al. propose an architecture that meets the goals of Multilevel Secure Data Stream Management Systems (MLS DSMS). An MLS DSMS is associated with a security structure that is a partial order, having a set of security levels with dominance relation between levels. The users of the system are cleared to different security levels and can see and query only the data corresponding to that level [9].

The proposed architecture is based on the general DSMS architecture consisting of an Input Processor, a Continuous Query (CQ) Instantiator, a Scheduler, a Run-Time Optimized, a Query Processor and a CQ Output Manager. From the existing solutions for MLS systems the authors chose to replicate the Query Processor for each security level. Even though MLS can be applied at attribute level, tuple level and stream level, in this paper the security level of the attributes is not considered.

This architecture is not implemented, but the authors plan to create a prototype, as well as for kernelized and trusted MLS architectures and perform a comparative study.

4.8. Iris

Another work, this time related to secure cloud storage and processing of data was done by E. Stefanov et al. who created Iris, a scalable cloud file system with efficient integrity checks [10]. It's focused on the freshness of data and high availability even in the case of cloud failures.

Iris lets an enterprise tenant maintain a large file system in the cloud [10]. The system has 2 components: a distributed portal and the cloud storage. The distributed portal has a portal service through which the clients issue file operations and a storage interface to communicate with the cloud. The portal also caches error correcting information. The entire file system is stored in the cloud together with the copy of the error correcting information.

Iris is based on an authenticated file system design to provide the data integrity and freshness. This approach is very specific to file systems using a balanced Merkle-tree data structure that authenticates both file-system data and meta-data blocks [10]. Auditing is done using Proof of Retrievability (PoR) protocol.

5. Related surveys

In his paper, T. W. Wlodarczyk provides a survey on the main frameworks that exist for storing and processing time series data in the Cloud [11]. These frameworks are: Chukwa, OpenTSDB, TempoDB and Squwk. Because of the early stage of the development of the solutions in this field the author does a qualitative comparison of the existing platforms. Chukwa, OpenTSDB, TempoDB and Squwk are also architectural different so a performance comparison wouldn't have been relevant.

This paper also presents a summary of the earlier alternative storage methods like: location-aware peer-to-peer storage systems and storage in sensor networks. The problem with these solutions was with the immediate transport of all data to sync which results in network congestion and increase power consumption [11].

Chukwa and OpenTSDB are open source platforms, while TempoDB and Squwk are hosted but free. These platforms use distributed file system, key-value store or NoSql to store the data. OpenTSDB has the simpler architecture and is more generic being the most popular choice for time-series storage. TempoDB instead offers better basic functionality.

T. W. Wlodarczyk also does a comparison of other works which are not yet developed into full platforms. Some of these propose the use of Cassandra instead of MongoDB for data storage. Others propose a key-value pair based solution to store network sensor data combined with ontology for continuously changing data objects [11].

Security-wise, B. Genge et al. compare and evaluate existing cloud platforms for secure Smart Grid against a basic set of security requirements [12].

The authors define three main areas in a cloud-based Smart Grid platform: cloud, communication and Smart Grid. From the studied platforms only two provide security solutions in all three areas. The compared platforms are mostly in an experimental stage, Cryptonite being implemented in a university grid.

6. Discussions

In this section we compare the studied solutions for data stream processing based on the basic set of operations defined earlier. As shown in Table 1 none of the solutions cover all of the basic operations we defined for the analysis of Smart Grid data.

Table 1. Overview of existing stream data analysis solutions

Data Stream Processing Solution	Storage	Indexing	Aggregation	Clustering	Sampling	Searching	Auditing
ElasticStream						✓	
SenQ	✓		✓		✓	✓	
Aurora	✓		✓		✓	✓	
Splunk	✓	✓	✓	✓		✓	
Stream				✓			
Indexing data streams solution	✓	✓		✓	✓	✓	
MLS DSMS			✓			✓	
Iris	✓					✓	✓

Smart Grid data has the properties of a data stream. Information is constantly generated and transmitted to the control centers. It is highly important that this data is stored so it can be queried later. Also real-time processing should be possible. The data stream can be tampered, or somebody could try to get unauthorized access to the data. The Smart Grid generates sensitive data, which needs to be kept confidential and private. Quickly detecting intruders or tampered data should be a priority.

ElasticStream and Iris both use cloud environments. ElasticStream uses the virtual machines (VMs) from an Infrastructure as a Service (IaaS) cloud to economically process the stream data. Iris stores the file system in a cloud environment and performs all the operations needed there. All the other solutions concentrate on data stream analysis, not choosing a specific environment. Taken into account the nature of the solutions it should not be a problem to migrate them in the cloud.

Except from Stream all solutions support searching on the stream data which is very important for Smart Grid data processing. Storage is also important for Smart Grid data. Not all solutions provide storage, but as mentioned earlier cloud is the perfect candidate for storing this kind of data.

Private searching could also be considered. R. Ostrovsky et al. propose a solution for private searching on streaming data which could be applied in a cloud-based system for Smart Grid data analysis. The results outlined in their paper can be viewed in a variety of ways: as a generalization of the notion of Private Information Retrieval (to

more general queries and to a streaming environment); as positive results on privacy-preserving data mining; and as a delegation of hidden program computation to other machines [13].

From the security point of view only MLS DSMS and Iris provide solutions. MLS DSMS proposes a security model based on levels and role-based access control (RBAC). Iris instead proposes a more comprehensive solution based on a custom auditing framework. It's a reactive not proactive system, but providing audit is already a plus. As a drawback thought it's customized for file data and not stream data which requires a different approach.

Testing a cloud-based solution for Smart Grid data analysis taking into account also the security requirements discussed in [12] should be done. For this purpose SCYAMIX can be used, a middleware aimed at facilitating cyber-physical security experimentation with Sensei/IoT* standard proposal and physical processes for Smart Grid [14].

7. Conclusion

The data flows in the Smart Grid have the properties of a data stream and should be processed as such. There are a couple of platforms that already do stream data analysis, some already integrated with cloud platforms. The majority of these solutions are already implemented and tested on real life testbeds and applications.

We defined the properties of the Smart Grid data and a basic set of operations that a cloud-based analysis system of this data should perform. From these operations storage and searching are highly important and the majority of the studied platforms implement them.

Security is also a key feature of such a system. Unfortunately the studied solutions do not take into account this aspect, except from one which implements an audit system for the data.

We intend in the future to continue the work on a cloud-based Smart Grid data analysis framework that will perform the operations that we defined and also include security requirements.

Acknowledgements

B. Genge's work on this research was supported by a Marie Curie FP7 Integration Grant within the 7th European Union Framework Programme.

References

[1] P. Mell, T. Gance, "The NIST Definition of Cloud Computing", *NIST Special Publication 800-145*, 2011.

[2] K. Q. Pu, Z. Ying, "Efficient Indexing of Heterogeneous Data Streams with Automatic Performance Configurations", *19th International Conference on Scientific and Statistical Database Management ,* 2007.

[3] M. Dipti, T. Patel, "K-means based data stream clustering algorithm extended with no. of cluster estimation method", *International Journal of Advance Engineering and Research Development (IJAERD) Volume 1,Issue 6,* June 2014.

[4] A. Ishii, T. Suzumura, "Elastic Stream Computing with Clouds", *2011 IEEE International Conference on Cloud Computing (CLOUD),* pp. 195-202, 2011.

[5] A. D. Wood, L. Selavo, J. A. Stankovic, "SenQ: An Embedded Quary System for Streaming Data in Heterogeneous Interactive Wireless Sensor Networks", *Proceedings of the 4th International Conference on Distributed Computing in Sensor Systems,* pp. 531-543, 2008.

[6] D. J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik, "Aurora: a new model and architecture for data stream management", *The VLDB Journal — The International Journal on Very Large Data Bases, Volume 12 Issue 2,* pp. 120-139, 2003.

[7] L. Bitincka, A. Ganapathi, S. Sorkin, S. Zhang, "Optimizing data analysis with a semi-structured time series database", *SLAML'10 Proceedings of the 2010 workshop on Managing systems via log analysis and machine learning techniques*, 2010.

[8] J. Forrest, "Stream: A Framework for Data Stream Modeling in R*", B.S. Thesis, Southern Methodist University*, 2011

[9] R. Adaikkalavan, I. Ray, X. Xie, "Multilevel secure data stream processing", *DBSec'11 Proceedings of the 25th annual IFIP WG 11.3 conference on Data and applications security and privacy,* pp.122-137, 2011.

[10] E. Stefanov, M. van Dijk, A. Juels, A. Oprea, "Iris: a scalable cloud file system with efficient integrity checks", *ACSAC '12 Proceedings of the 28th Annual Computer Security Applications Conference,* pp. 229-238, 2012.

[11] T.W. Wlodarczyk, "Overview of Time Series Storage and Processing in a Cloud Environment", *2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom),* pp. 625-628, 2012.

[12] B. Genge, A. Beres, P. Haller, "A Survey on Cloud-based Software Platforms to Implement Secure Smart Grids", *49th Universities' Power Engineering Conference - UPEC2014,* 2014.
[13] R. Ostrovsky, W. E. Skeith III, "Private Searching On Streaming Data", *Journal of Cryptology Volume 20 Issue 4*, pp. 397-430, 2014.
[14] B. Genge, P. Haller, A. Gligor, A.Beres, "An Approach for Cyber Security Experimentation Supporting Sensei/IoT for Smart Grid", *2nd International Symposium on Digital Forensics and Security*, 2014.