Smart Grids Data Management: A Case for Cassandra

Gil Pinheiro, Eugénia Vinagre, Isabel Praça, Zita Vale, Carlos Ramos

GECAD – Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, Institute of Engineering, Polytechnic of Porto (ISEP/IPP), Portugal

{1090574, empvm, icp, zav, csr}@sep.ipp.pt

Abstract. The objective of this paper is to present a SMACK based platform for microgrids data storage and management. The platform is being used in a real microgrid, with an infrastructure that monitors and controls 3 buildings within the GECAD - ISEP/IPP campus, while, at the same time, receives and manages data sources coming from different types of buildings from associated partners, to whom intelligent services are being provided. Microgrid data comes in different formats, different rates and with an increasing volume, as the microgrid itself covers more customers and areas. Based on the atual available computational resources, a Big Data platform based on the SMACK stack was implemented and is presented. The Cassandra component of the stack has evolved. AC version 2 is still supported until the version 4 release, and is often still used in production environments. However, a new stable version, version 3, introduces major optimizations in the storage that bring disk space savings. The main focus of this work is on the Data Storage and the formalization of the data mapping in Cassandra version 3, which is contextualized by means of a short example with data coming from the monitoring infrastructure of the microgrid.

Keywords: Big Data Storage; Smart Grids; Cassandra

1 Introduction

Technological developments led to a huge spread of monitoring equipment that now provide an enormous quantity of data, based on which intelligent services may become available, turning into dynamic the traditionally centralized management of certain areas. In power systems, technological developments and the roll out of meters turn the Smart Grid (SG) as a new reality. Indeed, digital data sources range from sensors, that measure electric parameters (current, voltage, phase shift and frequency), to meters that monitor in real time consumption data and distributed generation sources, to environmental sensors (temperature, humidity, etc.), all of them being relevant to shift from a static structure to a more intelligent and flexible way to manage the electrical energy resources. The monitoring of the grid status results in a huge amount of collected data to deal and sharing with various parties [1].
The volume, velocity, and variety of the data make traditional data storage systems inappropriate to obtain the relevant value from the data analysis in a very short time.

Big Data platforms are now the most promising way for the storing and analysis of high volumes of data. Apache technologies are being used in several domains. In this paper, we define a SMACK based architecture and implement a platform to support a real microgrid infrastructure existent at GECAD research group. Particular insights are given to the data storage process and the main focus of our contribution is given to the data mapping in relation to the partition size in Cassandra's most atual version.

The paper is structured into 4 sections, with section 2 addressing Big Data (BD) in SG context. Session 3 presents a platform based on SMACK, having Apache Cassandra (AC) as distributed storage for GECAD microgrid, with particular insights on the formalization of the data mapping process. Finally, the conclusions are presented in section 4.

2 Big Data and Smart Grids

To improve decision making, a system must be in place capable of collecting, managing, and processing information. In BD, the sheer volume of information requires new approaches when designing a solution that extracts knowledge within a reasonable period. This phenomenon, referred to as BD, is characterized by 5 Vs (i.e. Volume, Velocity, Variety, Veracity, Value) [2]. Each of these Vs represents real challenges (e.g. how to collect and transport a large volume of information; how to store this information, how to analyze and extract knowledge, how to ensure its security and privacy, how to process it in real time, etc.). The management of information with these characteristics raised great interest in the scientific and business community. Hadoop and Spark, are the most referenced frameworks.
The Apache Hadoop Framework was the first mainstream BD solution. It is based in Batch Processing, distributed file system HDFS (Hadoop Distributed File System), a programming model MapReduce and YARN (Yet Another Resource Negotiator) [3]. Apache Spark is a set of tools and high level APIs for large scale distributed processing of data in-memory [4]. Currently, Spark is considered as the most active open source project in BD. Its speed advantages, allied with an out of the box integration of data manipulation using SQL like syntax, support for several storage systems, and ability to distribute machine-learning computation, have contributed to its success.
In the new ecosystem of SG, all the players (i.e., power generation, transmission, distribution, customers, service providers, operations and markets) support their operations using a varied range of equipment that generate a large flow data. The last report issued by the European Union [5] refers numerous projects focusing the implementation of smart metering (SM). According to the same source, around 72 % EU customers are expected to be equipped with SM by 2020. The success of these projects launches an alert for the extensive amount of data generated in real time, that need adequate storage and analysis means to provide the development of dynamic services to better manage grid resources. Also, in the literature there are numerous references that characterize the type of data circulating on SG, at very high rates, as unstructured or, at most, semi-structured. Extrapolating this reality to the universe of

the existent equipment and the foreseen roll outs by 2020 is easy to understand the challenge is now on the data management. However, all the data being generated can only be transformed into value if properly analyzed in order to generate key knowledge in decision-making and the development of intelligent services able to dynamically manage the grid towards increasing sustainability, efficiency and safety. The value will be so much greater as the increasing ability to feed the ecosystem with data collected outside their own domain (e.g. atmospheric data, events, consumer behavior, etc.), correlate and analyses them, not only to decide and predict, but also to discover something even imaginable.

There are numerous operations in the SG area that require data analysis (e.g., operability; cybersecurity and privacy; self-healing fault; demand response; competitive energy markets; auto configuration; resource optimization; real-time decisions; forecasting; monitoring; etc.). Big data analytics is one of the biggest challenges in the BD domain. Traditional methodologies and algorithms are not prepared to run with large datasets (i.e. petabytes or more). Over the years, great efforts have been made on this issue and there are numerous references in the literature, of works and tools, for data analysis (based on batch and / or streaming) [4].

3 A Big Data Platform

Microgrids, sometimes referred as SG building blocks, are small areas with distributed energy resources that can operate in isolated mode. In this section we describe the BD cluster, with a distributed architecture, implemented in a real microgrid infrastructure existent in GECAD research centre [6].

3.1 GECAD microgrid

GECAD infrastructure includes 3 individual and independent buildings within the campus of the Institute of Engineering from the Polytechnic of Porto (ISEP/IPP), with photovoltaic (PV) and wind power generation; GECAD microgrid laboratory, which provides real-time simulation capabilities, a weather station from ISEP and consumption data from different types of buildings, some being monitored in real time and some others received through unstructured files. Fig. 1 illustrates the main data inputs for the platform.

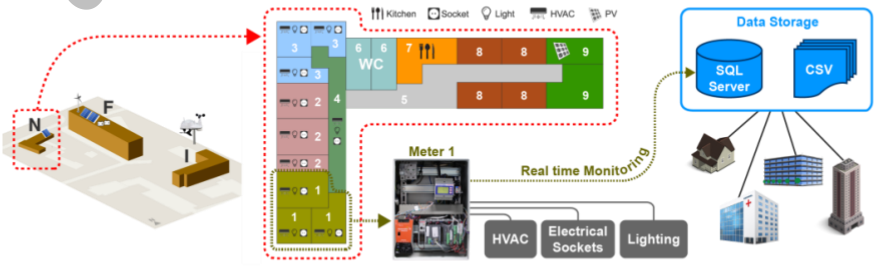


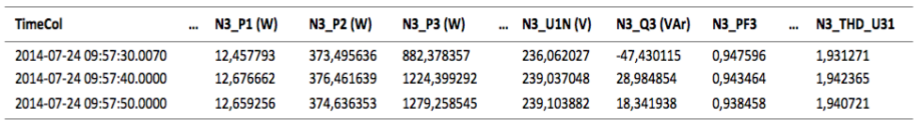
Fig. 1. Available data sources in GECAD's microgrid.

The 3 buildings are: Building F, with a PV system and a wind power system, where production is acquired every 10 seconds; a Building I, with consumption data acquisition every 10 to 15 seconds; and Building N, with its own PV system, injected directly into the building grid, consumption and generation is being acquired every 10 seconds. Details about GECAD microgrid can be found at [7].

Inside the buildings, three-phase energy analyzers measure data of three load groups grouped by rooms: Heating, Ventilation and Air Conditioning group (HVAC); Lighting group; and Electrical Sockets Group. The actual system stores data in time intervals ranging from 10 to 40 seconds (depending on the building) in a single SQL Server database.

To better illustrate the consumption data that is being acquired since 2014, with a total of 35 measures every 10 seconds, from grid frequency to current and voltage total harmonic distortion, power factors, apparent, active and reactive power, imported active energy, etc. Table 1 presents an excerpt of the information captured from the energy analyzers. The idea is to collect data directly from the energy analyzers, abandon the single server setup and store it in a distributed storage system.

Table. 1. Consumption data measurements.



TimeCol	...	N3_P1 (W)	N3_P2 (W)	N3_P3 (W)	...	N3_U1N (V)	N3_Q3 (VAr)	N3_PF3	...	N3_THD_U31
2014-07-24 09:57:30.0070		12,457793	373,495636	882,378357		236,062027	-47,430115	0,947596		1,931271
2014-07-24 09:57:40.0000		12,676662	376,461639	1224,399292		239,037048	28,984854	0,943464		1,942365
2014-07-24 09:57:50.0000		12,659256	374,636353	1279,258545		239,103882	18,341938	0,938458		1,940721

3.2 Distributed Architecture

A BD distributed architecture was designed and implemented in GECAd real mirogrid. As illustrated in Figure 2, the design is based on the SMACK stack, a combination of the Lambda and Kappa architecture models, that uses primarily open-source technologies (Spark, Mesos, Akka, Cassandra and Kafka) in an orchestrated pipeline that tackles both batch and streaming analysis for real-time scenarios under a single development language (Scala or Java) [8].

The cluster's hardware resources and software are managed by Apache Mesos. Apache Kafka serves as a message queue, which also provides APIs for streaming data ingestion, from the SQL database and directly from the energy analyzers. AC serves as the distributed persistent storage database and enables application development, while Apache Spark provides a richer query language over the stored data and the creation of forecasting machine learning models and streaming analysis. In total, there are two nodes for storing data in AC and for Spark analysis. The master nodes' responsibility includes managing the slave resources, scheduling Spark applications and has a Kafka node for data ingestion and message queuing.

The system makes use of Kafka's Connect API to pull raw data from the energy analyzers through the Modbus communication protocol into Kafka, where it is stored temporarily. Finally, another connector moves data from Kafka maps it into Cassandra, where it is persisted indefinitely and available for random access reads.

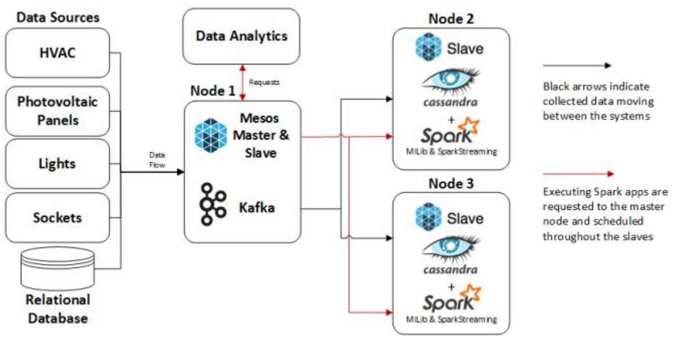


Fig. 2. Proposed BD architecture for GECAD's microgrid scenario.

This architecture is implemented with the computational resources actually available at GECAD. This serves as a first insight into the large-scale implementation of the architecture, with more advantageous computational resources and able to support several microgrids, e.g., a SG.

3.3 Data Storage in Cassandra

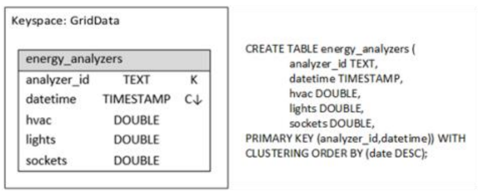
The actual SQL database presents itself as a serious storage system for massive data in the context of SGs. AC has been proposed to store historic data in a cloud-based architecture for SG [9] and considered to fulfill a similar role in a distributed data analytics platform for Wide-Area Synchrophasor Measurement Systems [10]. The system's success can be mainly attributed to its masterless architecture, linear scalability, multiple data center deployments and continuous availability.

However, due to scalability concerns, AC's Query Language (CQL) limits the possibilities of retrieving the data. When using this database, the queries are not an afterthought, but defined before the data model itself. In [11] the authors detail the concepts of the Cassandra data model and the preferred modeling methodology, including the concepts of: keyspaces, column families, partitions and clustering columns.

3.4 Data Mapping

Fig. 3 showcases a possible physical data model of a column family, illustrated with a small sub set of the registry, related to the active power of the 3 load groups from the energy analyzer N3, and its respective CQL statement. In this example, only the fields related to the load groups 'N3_P1', 'N3_P2' and 'N3_P3', from in Table. 1, are being used. A unique identifier is used as a partition key and the time of measurement is stored as a clustering column "datetime" in descending order.

In CQL, all selection statements must include the key. Retrieving all data at once can result in a timeout. Equality and lesser/greater searches on non-primary key columns are not possible, and as such, filtering records for any the load groups below or above a value is only possible, for instance, with SparkSQL.



```
CREATE TABLE energy_analyzers (
        analyzer_id TEXT,
        datetime TIMESTAMP,
        hvac DOUBLE,
        lights DOUBLE,
        sockets DOUBLE,
PRIMARY KEY (analyzer_id,datetime)) WITH
CLUSTERING ORDER BY (date DESC);
```

Fig. 3. Column Family for active power measurements and the respective CQL statement.

A visual representation of two energy analyzers partitions is seen in Fig. 4. The first energy analyzer has two rows separated in time while second only has a single record. From the image, we can see that the clustering value for the time of the measurement is repeated for each field in a row, and the field names are repeated across the whole partition. Both issues are addressed in AC version 3.

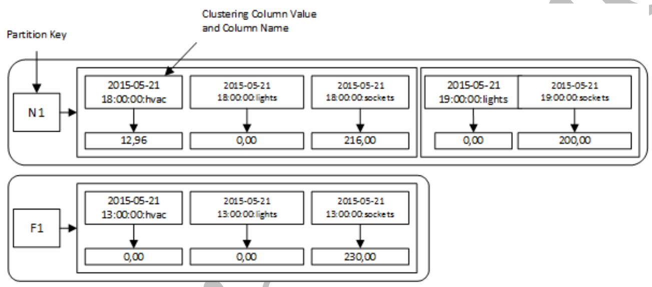

Partition Key
Clustering Column Value and Column Name
N1
2015-05-21 18:00:00:hvac
2015-05-21 18:00:00:lights
2015-05-21 18:00:00:sockets
12,96
0,00
216,00
2015-05-21 19:00:00:lights
2015-05-21 19:00:00:sockets
0,00
200,00
F1
2015-05-21 13:00:00:hvac
2015-05-21 13:00:00:lights
2015-05-21 13:00:00:sockets
0,00
0,00
230,00

Fig. 4. Visual representation of how AC version 2 stores data on disk.

Since a single partition is stored in one cluster node, as the records grow, the likelihood of creating hot nodes in the cluster increases. Therefore, it is important to split partitions to improve cluster health and performance.

3.5 Partition Size

Cassandra has limitations on how wide a partition can grow in terms of the number of cells and its size, before suffering from performance concerns. The maximum number of cells is 2 billion and the recommended partition size is approximately hundreds of megabytes [12]. Because the data sources are infinite streams, the previously suggested partition key must be changed. In this case, the time bucketing method of splitting partitions is preferred. Time bucketing involves adding a new date or time field to the partition key, effectively splitting the partition into smaller groups in time. For instance, monthly partitions can be achieved using a concatenated string of the month and year. However, a single partition per query will, at most, return the data for a specific month and year. Retrieving more data is accomplished by issuing parallel statements.

Datastax's Academy Course DS220 presents two formulas for roughly estimating the partition size for a column family for AC version 2 [13]. Equation 1 states that the number of values (cells) N_v is equal to the product of the estimated number of rows

N_r with the number of regular columns (in parenthesis), plus the number of static columns N_s. Regular columns are a result of the subtraction of the number of primary key columns N_{pk} and static columns N_s to the total N_c.

$$N_v = N_r \times \left(N_c - N_{pk} - N_s\right) + N_s \ . \tag{1}$$

Given N_v, equation 2 is used to calculate the partition size P_s in bytes. The equation can be broken into parts: first, we add the summations of the sizes of primary key columns C_k and static columnsC_s; second, multiply N_r with the addition of the size of each regular column C_r and the total size of all clustering columns C_c (Cassandra repeats the clustering values for each cell); third, we multiply the value of 8 (size of a hidden timestamp hidden in each cell)) to previously obtained N_v.

$$P_s = \sum_i sizeOf\left(C_{k_i}\right) + \sum_j sizeOf\left(C_{k_j}\right) + Nr \times \sum_k \left(sizeOf\left(C_{r_k}\right) + \sum_l sizeOf\left(C_{c_l}\right) \right) + 8 \times N_v \ . \tag{2}$$

If the data collection occurs at every second, at the end of a week, the value for N_r is 604 800 and $N_v = 604\,800 \times (5 - 2)$ totaling 1 814 400. For the partition size, we first assume the size in bytes of each of the fields: ‘analyzer_id’ 10 bytes; ‘datetime’ 8 bytes; ‘hvac’ ‘lights’ and ‘socket’s all 8 bytes each. Thus, $P_s = 10 + 0 + 604\,800 \times \left((8 + 8) + (8 + 8) + (8 + 8)\right) + 8 \times 1\,814\,400$, totaling approximately 42 MB.

Table 2 displays the results for various time resolutions and partition sizes at the end of the day, week and month. With the time resolution of one second, a monthly partition can be used cautiously, while weekly partitions keep the size below 100 MBs. With the time resolution of one second, a monthly partition can be used cautiously, while weekly partitions keep the size below 100 MB.

Table 2. Partition Size for one energy analyzer with different time resolutions for AC v2.

Time Resolution	End of	N_r	Nv	P_s (MB) Approximately
Each second	Day	86400	259 200	6
	Week	604 800	1 814 400	42
	Month	2 592 000	7 776 000	178
10 seconds	Day	288	864	0,6
	Week	2016	6048	4,1
	Month	8640	25 920	18

The storage engine for version 3.0 changes radically, providing storage savings [14]. The major differences include: the timestamps for conflict resolution are delta-encoded and can be written only once per row when all the cells have the same timestamp; the field names are stored at a row level; and the clustering column values are no longer repeated for each cell. Other important changes include better serialization that is discussed in detail in [15].

As of the latest version of AC (i.e. version 3.0), there are no widely accepted formulas for estimating partition size. We propose to change equation 2 to:

$$P_s = \sum_i sizeOf(C_{k_i}) + \sum_j sizeOf\left(C_{s_j}\right) + Nr \times \left(\sum_k sizeOf(C_{r_k}) + \sum_l sizeOf(C_{c_l}) + 8\right). \quad (3)$$

Equation 3 moves the clustering columns and the hidden timestamp to a row level. It is important to note that both equations do not consider the field names serialization of the different storage engines. This is important when using columns with dozens of regular columns, as the effects in size will be more significant. The updated results can be seen in Table 3. There is a substantial decrease in partition size using the new storage engine, and the monthly partition can be recommended.

Table 3. Partition size for one energy analyser using the adapted formula.

Time Resolution	End of	N_r	Nv	P_s (MB) Approximately
Each second	Day	86400	259 200	3,2
	Week	604 800	1 814 400	23
	Month	2 592 000	7 776 000	98

It is important to remember that partition bucketing will have consequences on the querying strategy. The best approach is to consider the widest partition strategy relative to the application requirements while trying to maintain the size to 100 MB or below.

4 Conclusions

In this paper a BD platform for SG has been proposed. It is based on Smart Stack architecture, implemented in a distributed way and already being fed by several data sources from GECAD microgrid infrastructure and external sources from partners. The implementation was done with the available computational resources at GECAD, but this cluster provides a first insight about the scale up of the platform for SG management. An important contribution is the data mapping in AC version 3.0 and how the partition size should be done to take the best profit of the data for real time data analysis. It's critical to have the ability to compare multiple data stores intelligently and objectively so that sound architectural decisions can be made. So, in the next step we will validate the platform proposed with YCSB (i.e. an open standard for comparative performance evaluation of NoSQL data stores). Further expanding the cluster with additional computational resources is something previewed in the short run.

Acknowledgements. This work has received funding from EU Horizon 2020 R&D programme under Marie Sklodowska-Curie grant agreement No 641794 (project DREAM-GO), EUREKA - ITEA2 Project FUSE-IT (nr.13023), Project GREEDI

(ANI|P2020 17822), FEDER Funds through COMPETE program and National Funds FCT under the UID/EEA/00760/2013 and SFRH/BD/103089/2014.

References

1. Skopik, F., Ma, Z.: Attack Vectors to Metering Data in Smart Grids under Security Constraints. In: 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops. pp. 134–139. IEEE (2012).
2. Chandarana, P., Vijayalakshmi, M.: Big Data analytics frameworks. In: 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA). pp. 430–434. IEEE (2014).
3. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop Distributed File System. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). pp. 1–10. IEEE (2010).
4. Zaharia, M., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S.: Apache Spark: A Unified Engine for Big Data Processing. Commun. ACM. 59, 56–65 (2016).
5. Felix Covrig, C., Ardelean, M., Vasiljevska, J., Mengolini, A., Fulli, G., Amoiralis, E., Sánchez Jiménez, M., Filiou, C., European Commission. Joint Research Centre. Institute for Energy and Transport.: Smart grid projects outlook 2014. Publications Office (2014).
6. Vinagre, E., Gomes, L., Vale, Z.: Electrical energy consumption forecast using external facility data. Proc. - 2015 IEEE Symp. Ser. Comput. Intell. SSCI 2015. 659–664 (2016).
7. Gomes, L., Lefrancois, M., Faria, P., Vale, Z.: Publishing real-time microgrid consumption data on the web of Linked Data. In: 2016 Clemson University Power Systems Conference (PSC). pp. 1–8. IEEE (2016).
8. Estrada, R., Ruiz, I.: Big Data SMACK. Apress, Berkeley, CA (2016).
9. Mayilvaganan, M., Sabitha, M.: A cloud-based architecture for Big-Data analytics in smart grid: A proposal. In: 2013 IEEE International Conference on Computational Intelligence and Computing Research. pp. 1–4. IEEE (2013).
10. Zhou, D., Guo, J., Zhang, Y., Chai, J., Liu, H., Liu, Y., Huang, C., Gui, X., Liu, Y.: Distributed Data Analytics Platform for Wide-Area Synchrophasor Measurement Systems. IEEE Trans. Smart Grid. 1–9 (2016).
11. Chebotko, A., Kashlev, A., Lu, S.: A Big Data Modeling Methodology for Apache Cassandra. In: 2015 IEEE International Congress on Big Data. pp. 238–245. IEEE (2015).
12. Datastax: Estimating partition size, https://docs.datastax.com/en/landing_page/doc/landing_page/planning/planningPartitionSize.html. Accessed 23 Nov 2016
13. Datastax Academy: Physical: Partition Size, https://academy.datastax.com/courses/ds220-data-modeling/physical-partition-size. Accessed 20 Nov 2016
14. Lebresne, S.: Putting some structure in the storage engine, http://www.datastax.com/2015/12/storage-engine-30. Accessed 4 Dec 2016
15. Morton, A.: Introduction To The Apache Cassandra 3.x Storage Engine, http://thelastpickle.com/blog/2016/03/04/introductiont-to-the-apache-cassandra-3-storage-engine.html. Accessed 4 Dec 2016