Electric Power Systems Research 151 (2017) 369–380

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Big data framework for analytics in smart grids

Amr A. Munshi[a,b,*], Yasser A.-R. I. Mohamed[a]

[a] Electrical and Computer Engineering Department, University of Alberta, Edmonton, Alberta, Canada
[b] Computer Engineering Department, Umm Al-Qura University, Makkah, Saudi Arabia

ARTICLE INFO

Article history:
Received 27 March 2017
Received in revised form 15 May 2017
Accepted 9 June 2017

Keywords:
Smart grid
Smart meters
Big data
Data management
Dynamic demand response

ABSTRACT

Smart meters are being deployed replacing conventional meters worldwide and to enable automated collection of energy consumption data. However, the massive amounts of data evolving from smart grid meters used for monitoring and control purposes need to be sufficiently managed to increase the efficiency, reliability and sustainability of the smart grid. Interestingly, the nature of smart grids can be considered as a big data challenge that requires advanced informatics techniques and cyber-infrastructure to deal with huge amounts of data and their analytics. For that, this unprecedented smart grid data require an effective platform that takes the smart grid a step forward in the big data era. This paper presents a framework that can be a start for innovative research and take smart grids a step forward. An implementation of the framework on a secure cloud-based platform is presented. Furthermore, the framework has been applied on two scenarios to visualize the energy, for a single-house and a smart grid that contains over 6000 smart meters. The application of the two scenarios to visualize the grid status and enable dynamic demand response, suggests that the framework is feasible in performing further smart grid data analytics.

1. Introduction

Ever growing volume of data production is the reality we are living in. The recent technological advancements have led to a deluge of data from various domains, such as social networks, scientific sensors, smart cities, and the Internet. The global data volume from 2005 to 2020 is predicted to grow by a factor of 300, from 130 exabytes to 40,000 exabytes, representing a double growth every two years [1]. To cope with the volume, velocity and variety of data produced, the term "big data" was brought up to capture the meaning of this evolving trend of data.

Big data are becoming a new technology focus in science and engineering domains. Big data systems include a set of tools and mechanisms to acquire, store, and process disparate data while leveraging the massively parallel processing power to perform complex transformations and analysis. However, designing and deploying a big data framework system for a specific application is not a trivial or straightforward task [2,3]. This is due to the fact that data comes from multiple, heterogeneous and autonomous sources with complex and evolving relationships, and keeps on growing. Moreover, the rise of big data applications where data collection has grown tremendously is beyond the capability of current commonly used hardware and software platforms to manage, store and process within a tolerable amount of time [2].

Many utilities are transferring to smart meters and smart grids as part of long range planning to improve the reliability of power supply, incorporate distributed generation resources, develop storage solutions, use the power plants efficiently, and enable customers to participate in controlling their energy use. To accomplish this, many utilities are deploying smart meter systems as a first step. This leads to incorporate other challenges. For example, going from a system that reads the meter once a month to a smart meter that can provide meter readings every few minute leads to millions of reads per hour. The result is a massive increase in data that is overwhelming if not managed properly. This generated data if managed efficiently can provide better understanding of customer behavior and assist in defining electric tariffs. For example, time of use pricing encourages customers to operate certain higher voltage appliances at off-peak periods. Consequently, customers save money and less power is generated.

Smart grids are a dynamic field of research and development and are currently observing a phase where research ideas are of interest. Technology changes are starting to permeate through the entire

* Corresponding author at: Electrical and Computer Engineering Department, University of Alberta, Edmonton, Alberta, Canada.
E-mail address: aamunshi@ualberta.ca (A.A. Munshi).

http://dx.doi.org/10.1016/j.epsr.2017.06.006
0378-7796/

smart grid, from generation to transmission and distribution. For example, renewable power resources such as wind and solar power are being included in the generation mix, not just by the power generation utilities, but also by consumers through micro-grids. Also, vehicle-to-grid (V2G) and grid-to-vehicle (G2V) technologies can provide power flow from vehicles to power lines and back. Also, smart meters are being deployed at consumer premises to monitor near real-time energy consumption data and securely communicate it back to the utility over communication networks. These meters can also receive data from the utility with information on dynamic power usage and incentives for reducing load during peak periods. Streaming various data from thousands of smart meters, sampled every few minutes, must be collected and correlated for monitoring, controlling, and research purposes. Also, such data can be processed and shared between utilities and customers to enable dynamic demand response (DDR) for near real-time monitoring and response. Smart grids are characterized by concepts of sustainability, interoperability, and controllability. The contemporary developments toward the future smart grid will require acquiring and analyzing data of integrated devices, such as distributed storage, intelligent loads, and distributed energy resources [4]. However, the massive amounts of data evolving from smart grids needs to be sufficiently managed to increase the efficiency, reliability and sustainability of the grid. This is a big data challenge that requires advanced informatics techniques and cyber-infrastructure to deal with huge data and their analytics. Interestingly, big data reflects the true nature of the smart grids. For that, the unprecedented data volumes require an effective platform that takes the smart grid a step forward in the big data era.

In Refs. [2,5–9] various challenges and issues in adapting big data technology were discussed. From their research, it was concluded that refining a unified framework suitable for every module is not straight forward due to the diversity of applications. For that big data frameworks for smart grids are of research interest. In Refs. [7,9] several main elements of big data and data base technologies that are beneficial within the utility ecosystem are scratched on, but a comprehensive idea on how big data elements can construct a framework to deal with smart grid data has not been presented. The authors of Ref. [8] designed architecture for smart grids based on mathematical foundations and statistical procedures. Ref. [3] analyzed several challenges of big data and suggested that high-performance computing platforms are required to unleash the power of big data. In Ref. [10], statistical learning algorithms for big data were presented. Also, Ref. [11] presented an unsupervised method for early event detection in smart grids with big data. A big data value chain in Ref. [2] was presented; it decomposed big data into four sequential modules, namely data generation, data acquisition, data storage, and data analytics. More in detail, numerous approaches for each phase were highlighted, and a prevalent framework for addressing big data challenges was suggested. Ref. [12] demonstrates a cloud-based DDR platform project being deployed in the University of Southern California (USC) campus micro-grid as a testbed for transforming Los Angeles utility into a smart grid in the future. Works in big data analytics mainly describe possible theoretical frameworks and challenges, and lack practical implementation. Specially, handling real smart grid big data is becoming an area of research interest for that, this paper focuses on the development of a comprehensive big data framework for analytics in smart grids. Furthermore, an implementation and application of the framework is applied to test its feasibility.

The contributions of this paper to the research field are:

- A comprehensive big data framework for smart grids.
- The utilization of open source state-of-the-art prevalent Hadoop platform for addressing smart grids big data challenges.
- The adoption of open source tools to provide an easy and cost-effective development environment for practicing engineering to develop similar tools for their demanding smart grid applications.
- Practical implementation of the framework, including necessary configuration and coding, and application of the framework on real smart grid data.
- The development of a secure cloud-based DDR platform.
- Outlining potential research directions that can be applied using the framework.
- The ability to perform various data analytical applications on top of the framework.

The remainder of this paper is structured as follows. Section 2 briefly discusses the main characteristics of big data. The big data core components used in the framework for smart grids and the features of using the Hadoop platform in the smart grid environment are highlighted in Section 3. Furthermore, the big data framework for smart grids including the lifecycle of smart grid data from data generation to data analytics is introduced. An implementation with relevant source code on a secured cloud platform is presented in Section 4. The application of the framework on two scenarios is presented in Section 5. Finally, the conclusions are drawn in Section 6.

2. Characteristics of big data

Concurrently, there has been much discussion about what big data actually means [2,12,13]. However, the most common definition in literature is the “Vs” definition [2,14–19] which contains several characteristics of big data beginning with the letter “V”. In this paper, we limit ourselves to the “4Vs” definition [19] (volume, variety, velocity and veracity) which are discussed next.

- **Volume**: big data implies enormous volumes of data. Concurrently, that data is generated by machines, networks, and social media. Thus, the volume of data to be analyzed is massive.
- **Variety**: refers to the various sources and formats of data (structured, semi- structured and unstructured). As data comes in the form of photos, videos, logs, sensor devices, etc., this variety of data formats creates challenges for storage, mining and analyzing.
- **Velocity**: the velocity of data is the rate at which data arrives. This also includes the time that it takes to process and understand the acquired data to help in decision making.
- **Veracity**: is a term that refers to the quality or trustworthiness of the data. Tools that help handle big data’s veracity discard noise and abnormality in data and transform the data into trustworthy insights.

3. Framework’s big data core components for smart grids

This section highlights the big data core components used in the framework for smart grids. Also, the features of using Hadoop’s big data platform in smart grids environment are highlighted. Further, the framework that covers the lifecycle of smart grid data from data generation to data analytics is covered.

3.1. Big data core components—state-of-the-art

Fig. 1 illustrates the hierarchical architecture of the core components of the suggested framework for smart grid big data. In the following sub-sections, the state-of-the-art components of the data acquisition, data storing and processing, data querying and, data analytics components are introduced. The data storing and processing components are included in the same subsection as they fall under the same platform (Hadoop).

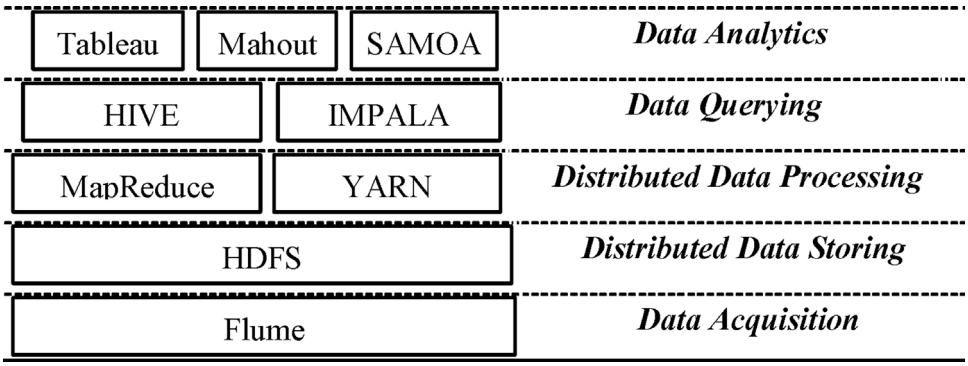


Fig. 1. A hierarchical architecture of the core components for smart grid big data, including the components of data acquisition, data storing, data processing, data querying and data analytics.

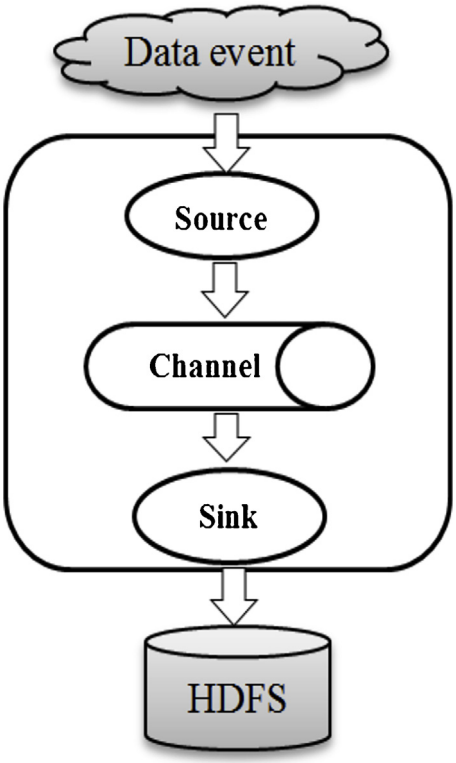


Fig. 2. A basic Flume topology to ingest data into HDFS.

3.1.1. Data acquisition component

Flume is a distributed system developed by Apache that efficiently collects, aggregates, and transfers large amounts of log data from disparate sources to a centralized storage. However, flume can be used to ingest large amounts of streaming data such as social media and sensor data into the Hadoop's Distributed File System (HDFS) which will be introduced in the following subsection.

Fig. 2 depicts a representative Flume topology [20] and the following components make up the Flume data acquisition tool:

- **Event**: a stream of data that is transported by Flume.
- **Source**: the entity through which data enters into Flume. A source can actively poll for data or wait for data to be delivered to it.
- **Sink**: the entity that delivers the data to the destination. A variety of sinks allows data to be streamed to a range of destinations. Here the HDFS sink that writes events to an HDFS storage is used.
- **Channel**: the conduit between the source and the sink. Sources ingest events into the channel, and the sinks drain the channel.

3.1.2. Distributed data storing and processing components

Hadoop [21] is an open-source software platform that supports massive data storage and processing. Instead of relying on expensive, proprietary hardware to store and process data, it enables distributed processing of large amounts of data on clusters of commodity servers. The core of Hadoop consists of two main components: a storage component which is Hadoop's Distributed File System (HDFS) [22] and a processing component called MapReduce [23]. Hadoop is considered to be a major part of any architecture in big data.

Hadoop was inspired by Google's work on its distributed file system, Google's File System (GFS) [24] and the MapReduce programming model. Shortly after Google published papers describing GFS and MapReduce, it became apparent that the GFS and MapReduce modules together had applications beyond the search engines. In 2006, these software components became an Apache project, called Hadoop.

HDFS is a distributed storage file system developed to run on commodity hardware that references the GFS architecture. An HDFS cluster consists of a single NameNode that manages the file system metadata, and numerous DataNodes that store the actual data. A file is split into one or more blocks, and these blocks are stored in a set of DataNodes. Each block has several replications distributed in different DataNodes to avoid missing data.

MapReduce [23] is the processing component of Hadoop. It consists of a single master (JobTracker) and one slave (TaskTracker) per cluster node. The master is responsible for scheduling jobs for the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master. MapReduce and HDFS run on the same set of nodes, which allows tasks to be scheduled on the nodes in which data are already available. MapReduce was sufficient at a few specific kinds of batch processing, which was an obstacle in taking advantage of Hadoop's flexibility with data and storage. In 2013, Hadoop was released with Yarn [25] (Yet Another Resource Negotiator) or MapReduce2. Yarn is a general-purpose resource manager for Hadoop, which enabled applications from other processing frameworks to run on a Hadoop cluster in a distributed manner. Also, it started a paradigm for stream data applications. The fundamental idea of Yarn is to split up the two major responsibilities of the JobTracker/TaskTracker of the MapReduce into separate entities. Yarn basically consists of a global ResourceManager and per-node slave NodeManager for managing applications in a distributed manner. The ResourceManager is the ultimate authority that arbitrates resources among all applications in the cluster of commodity servers. An Application-

Master in Yarn negotiates resources from the ResourceManager and works with the NodeManager(s) to execute and monitor the component tasks. The ResourceManager has a scheduler, which handles allocating resources to the various applications running in the cluster. The scheduler schedules based on the resource requirements of each application. Each ApplicationMaster has responsibility for negotiating appropriate resource containers from the scheduler, tracking their status, and monitoring their progress. Unlike MapReduce, this architecture allows an unlimited number of nodes to be added to a cluster at any time without affecting the functionality of the existing cluster.

The interested reader is referred to Refs. [21–25] for more details on the Hadoop platform.

3.1.3. Data querying components

Hive [26] and Impala [27] are two SQL-like high-level declarative languages that express big data analysis tasks. They facilitate querying and managing big data residing in distributed storage. Hive express big data analysis tasks in MapReduce operations. Whereas, Impala is a real-time interactive SQL query tool on big data [27]. Impala does not have to translate an SQL query into another processing framework, such as MapReduce operations on which Hive depends on. The execution of an Impala query is executed in parallel in each node's memory in the cluster. The intermediate results of the nodes are transmitted and aggregated then returned. As a result, Impala queries can provide near real-time results.

3.1.4. Data analytics components

Mahout is a data mining library implemented on top of Hadoop and provides batch machine learning processing. It contains various core algorithms for scalable performant machine learning applications.

SAMOA [28] (Scalable Advanced Massive Online Analysis) is a distributed streaming machine learning framework that contains a programming abstraction for distributed streaming algorithms for the most common data mining and machine learning tasks.

Tableau is an interactive data visualization tool that enables users to analyze, visualize and share information and dashboards.

3.2. Features of Hadoop's platform in smart grids

Hadoop has attracted substantial attention from both industry and scholars. In fact, Hadoop has long been the mainstay of the big data movement. Hadoop has many advantages, and the following features make it suitable for smart grid big data management and analysis:

3.2.1. Scalability

Hadoop allows hardware infrastructure to be scaled up and down with no need to change data formats [2]. The system can automatically re-distribute data and computation tasks to accommodate hardware changes. For example, if new neighborhoods or generation utilities are added to the grid, additional nodes and storage devices can be added to the existing cluster without affecting the functionality of the existing nodes.

3.2.2. Real-time cost efficient computation

Hadoop's Yarn [25] brings massively parallel computation to commodity servers, leading to a sizeable decrease in cost per terabyte of storage, which makes parallel computation affordable with the growing volume of smart grid data. This presents an opportunity to share data between utilities and customers to enable DDR for near real-time monitoring and decision making. Also, this enables low-latency mining and forecasting tasks on smart grid stream data.

3.2.3. Flexibility

Hadoop is free of schema and able to absorb various types of data from numerous sources. Moreover, different types of data from numerous sources can be aggregated for further analysis. Hence, many challenges of the various types of smart grid data can be addressed.

3.2.4. Fault tolerance

Missing data and computation failures are common in smart grid data. Hadoop can recover the data and computation failures caused by node breakdown or network congestion by storing the data at many nodes and distributing the computation work to other healthy nodes in the cluster.

3.3. Big data framework for smart grids

The framework to deal with smart grid big data is presented in Fig. 3. The framework covers the lifecycle of smart grid data from data generation to data analytics. The following sub-sections discuss the framework stages in details.

3.3.1. Data generation

Streaming data is generated from thousands of smart meters in the smart grid, sampled every few minutes. The generated data may belong to a supplier site (e.g., power plants and wind farms) or a demand site (e.g., residential homes and factories). In addition, environmental events such as weather conditions from weather stations can be beneficial. For example, to predict the amount of power that can be generated from a certain power resource such as, a wind farm. In this framework, data from various sources are considered these include electric vehicles (EV), residential homes, commercial buildings, industrial factories, solar panels, wind turbines and various power plants. Considering data from such sources increases the grid's reliability as technology changes are starting to permeate through the entire smart grid, from generation to transmission to distribution. For example, renewable power resources are being included in the generation mix, not just by the power generation utilities, but also by consumers through rooftop solar panels, residential wind turbines, EVs and other micro-generators that can act as positive supply to the grid.

3.3.2. Data acquisition

The data acquisition for smart grids' data can be decomposed into three sub-tasks namely, data collection, data transmission, and data pre-processing. The data generated from the previous stage are collected proactively by centralized/distributed agents. The collected data is then transmitted to a master node(s) in the Hadoop cluster. It should be noted that local master nodes can be included in the system. Once the raw data are gathered, it is transferred to a data storage infrastructure for subsequent processing. Due to the diverse source of data, the collected data may have different formats and information, accordingly, data pre-processing is required. Data integration techniques aim to combine data from different sources and provide a unified view of the data [2]. In this framework, the data is transferred to comma-separated value (csv) files. The attributes of the data contain information such as, the timestamp, smart meter's ID, generated/consumed power and location. Also, in the pre-processing of data, inaccurate and incomplete data are amended or removed to improve the quality of data.

Flume can fulfil the function of the data acquisition. It can collect, aggregate, and transfer the large amounts of generated data from the various sources to a Hadoop master node. When a Flume source receives data, it stores it into one or more channels. The channel is a passive store that keeps the event until it is consumed by a flume sink. The flume sink removes the event from the channel and puts it into an external repository. In this framework, it is

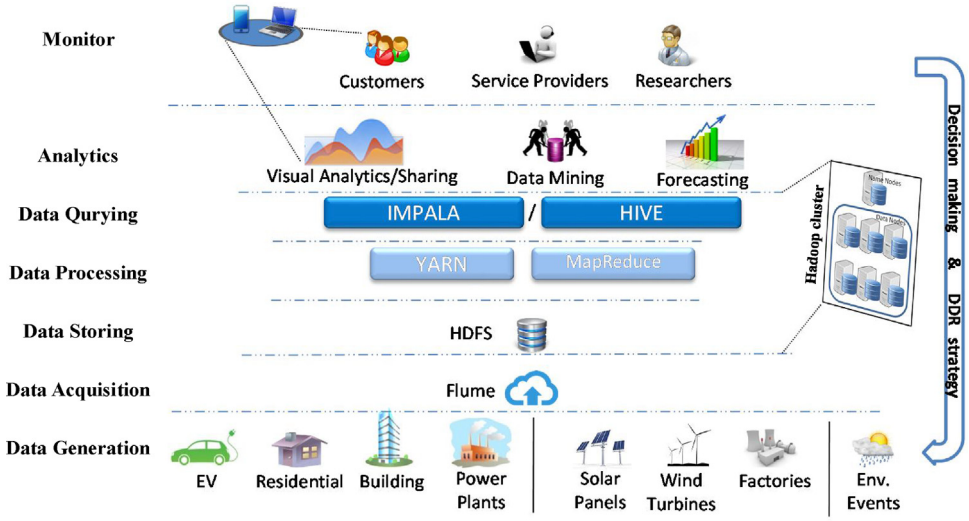


Fig. 3. The framework to deal with smart grid big data for visual analytics. The framework covers the lifecycle of smart grid data from data generation to data analytics and forms a learn and response loop.

desired to sink the data files into an external HDFS repository. The HDFS sink in Flume allows writing data into an HDFS repository in a desired format by allowing plug-in serializers. Serializers convert and restructure the Flume data into a desired format. Consequently, the data are pre-processed, and a unified view of the data can be achieved.

3.3.3. Data storing and processing

After the smart grid data has been acquired, in this stage, Hadoop's HDFS handles storing the data for further processing. As mentioned previously, an HDFS cluster consists of a single NameNode that manages the file system metadata, and collections of DataNodes that store the actual data. The received smart grid data is split into one or more blocks, and these blocks are stored in a set of DataNodes. Hadoop Yarn is the computation core for big data analysis. The HDFS and Yarn run on the same set of nodes, which allows tasks to be processed on the nodes in which smart grid data are already present. Fortunately, open-source Apache Hadoop distributions that include the MapReduce, HDFS and other basic components exist. The most common open-source Hadoop distributions include Cloudera, MapR and Hortonworks.

3.3.4. Data querying

Hive and Impala are used in this framework to read the smart grid data from a HDFS repository and select, analyze or generate data of interest. For example, the consumption of electricity for a certain region or the aggregated power produced from wind farms can be obtained. The data querying stage runs on top of a Hadoop cluster which allows obtaining prompt results. It should be noted that other data querying elements such as 'Apache Pig', which creates MapReduce operations, can be used.

3.3.5. Data analytics

The smart grid data acquired must be shared to improve the efficiency of the smart grid. For example, this data can be utilized by analytics for proposing curtailment, researchers for data mining and correlations, and consumers visualizing and gaining knowledge of their power profiles. The sharing of such data must be balanced against the concerns of data security; however, this issue is beyond the scope of this paper. Secure transmission of such data can be aggregated using methods such as in Refs. [29] and [30]. Many tools exist to perform data analytics such as, Tableau for big data visualization, and Mahout and SAMOA for mining big data.

The data analytics stage has two main objectives, to learn and to response. Sharing the grid's status between utilities and consumers promotes the stability of the smart grid. Also, consumers act as an active part in the stability of the grid. This can be achieved through visualization dashboard portals, which provide a visualization of the smart grid's status that can be accessed via the Internet or mobile apps. Consequently, a DDR strategy by analytics can be suggested to determine customers and buildings to target during a peak load period. Moreover, dynamic power pricing and incentives for reducing load during peak periods can be advertised for.

4. Implementation on a cloud computing platform

This section demonstrates the implementation of the framework on a cloud computing platform. Then a method to establish a secure connection between the cloud cluster nodes is briefly presented. Further, the settings of the components that are used at each stage of the framework are discussed. For simplicity, in the data storing and data processing stages a Hadoop distribution namely, Cloudera distribution Hadoop (CDH) that contains MapReduce/Yarn and HDFS is used instead of setting up each component separately.

4.1. Cloud platform

Cloud computing can be deployed as the infrastructure layer for big data systems to meet certain infrastructure requirements, such as cost-effectiveness, improved accessibility, and scalability. Based on the requirements of the proposed framework, Infrastructure as a Service (IaaS) clouds [31] are appropriate to use to implement the smart grid big data framework. Cloud service providers such as, Amazon AWS and Google can be utilized to build a cluster that will host the framework. In this implementation, a Google cloud platform cluster with six machines is used. Five machines running CentOS Linux operating system will be deployed for the Hadoop platform. The remaining machine will be running Windows operating system to perform the visual analytical tasks. In the Hadoop cluster there will be one master node and four slave nodes. The IP address and host name of the nodes are identified at each node in the/*etc/hosts* file (Fig. 4):

It should be noted that at any time a new node is added to the cluster, it must be defined to all other cluster nodes in the/*etc/hosts* file.

```
10.240.0.2    master.sgf    #IP and node name for master
10.240.0.3    slave1.sgf    #IP and node name for slave1
10.240.0.4    slave2.sgf    #IP and node name for slave2
10.240.0.5    slave3.sgf    #IP and node name for slave3
10.240.0.6    slave4.sgf    #IP and node name for slave4
```

Fig. 4. The IP address and host name of the machines identified at each cluster node in the/*etc/hosts* file.

```
~]$ ssh-keygen
~]$ cd ~/.ssh
~]$ cp id_rsa.pub authorized_keys   #copying node key to list of authorized keys
```

Fig. 5. Commands to setup an SSH connection.

Often smart grid big data analysis is conducted with a vast array of data sources that come from many unvetted sources. For that, it is required to be aware of the security and governance policies that apply to various smart grid data sources. The data that remains will need to be secured and governed. Therefore, a well-defined security strategy is required. It should be notated that security is something that requires frequent update strategies because the state of the art is constantly evolving. Also, data encryption, for example, when sending the smart grid data to a cloud provider can be considered, however, this point is beyond the scope of this paper. In order to establish the smart grid framework in a secure cluster environment, Secure Shell version 2 (SSH) protocol [32] is adopted. SSH is a cryptographic network protocol that allows network systems to operate securely over an unsecured network. The SSH provides a secure encrypted link in a client–server architecture, which connects a client with a server. In addition, SSH provides authentication, encryption and data integrity to secure network communications. The setup of SSH enables different operations on the cluster, such as starting, stopping, and distributing operations to nodes. In the context of our cluster, it provides secured connection between the slaves and master(s) nodes. Also, specifies how a node can connect securely to another node, and then use the resulting secure connection to access the other nodes resources. An advantage of implementing the framework using a known cloud service providers, is that their cloud service compleys with the Cloud Security Alliance (CSA) standards which promote the use of best practices for providing and ensuring security within cloud compting. To implement the SSH in the cloud cluster, the public key authentication method in used. Public key authentication in SSH is considered to be a popular strong authentication method, that it used to authenticate the cluster nodes. To accomplish this, a manually generating public and private key are generated on a node. The public key will be given to all cluster nodes that require authentication. Any data encrypted with that public key, will be decrypted with the corresponding private key. Thus, every cluster node has a file which contains the complete list of the other node keys. While authentication is based on the private key, the key itself is not transferred through the network during authentication. The SSH only verifies if the same node offering the public key also owns the matching private key. This will prevent a node that does not belong to the cluster to connect as an authenticated node (eavesdropping). The commands to setup an SSH connections are presented in Fig. 5.

```
2009-07-14 00:00:00,1001,0.286,80302
2009-07-14 00:00:00,1002,0.089,80302
2009-07-14 00:00:00,1003,0.086,80302
2009-07-14 00:00:00,1004,0.149,80302
2009-07-14 00:00:00,1005,0.086,80302
```

Fig. 6. Sample of the csv file that includes attributes of timestamp, smart meter's ID, generated power and zip code.

Then the public key (*id_rsa.pub*) is copied to each node. Hence, every node in the cluster has the complete list of the other node keys.

The components that are needed to implement the framework (i.e., Flume, Hadoop, Hive, Impala) are setup on the cluster. Fortunately, those components can be found in open source distributions. For simplicity, in this framework Cloudera distribution Hadoop (CDH) is used. During the setup of CDH, the master and salve nodes are identified by their host name or IP address. The master node will run the master "daemons" and it knows where the slaves are located and how many resources they have. A node identified as master runs several services; the most important is the ResourceManager which decides how to assign the resources. Nodes identified as slaves announce themselves to the ResourceManager. Periodically, they send heartbeats to the ResourceManager. Each slave node offers resources to the cluster. The resource capacity is the amount of memory and the number of cores. At run-time, the ResourceManager will decide how to use this capacity.

4.2. Flume

Once the data are available from smart meters, it is sent to a local node. An advantage of using the cloud service is that the data can be sent from any location as long as there is an Internet connection and the security protocol of the cloud allows it. For example, a certain neighborhood's data can be sent to a node using one Internet connection. It should be noted that the network communication and aggregation of smart meter data is beyond the scope of this paper. Papers that discuss these topics can be found in Ref. [29,30]. Nodes that act as flume agents can be master nodes or slave nodes. In this implementation, a flume agent receives data in a csv file format. The attributes of the file can include data such as, the timestamp (Datetimes), smart meter's ID (ID), generated (Gen)/consumed (Cons) power and zip code (Zip) (Fig. 6).

When a flume source receives an event it stores it into one a channel that keeps the file event until it is consumed by a flume sink (see Fig. 2). The sink removes the file event from the channel and puts it into the HDFS sink. The file events are removed from the channel only after they are stored in the HDFS repository for reliability. The flume agent configuration is stored in a local configuration file (*/etc/flume-ng/conf/flume.conf*). This is a text file that follows the Java properties file format. Configurations for one or more agents can be specified in the same configuration file. The configuration file includes properties of each source, sink and channel in an agent and how they are wired together to form data flows.

4.3. Hadoop platform

In this illustration the Cloudera Manager Hadoop distribution is used for simplicity. The reason for using such distribution is to provide an easy development environment for practicing users that are not familiar with CentOS operating system to develop sim-

```
# Initialize agent's source, channel and sink
agent.sources = SGFExampleDir
agent.channels = memoryChannel
agent.sinks = flumeHDFS
# Setting the source to spool directory where the file exists
agent.sources. SGFExampleDir.type = spooldir
agent.sources. SGFExampleDir.spoolDir =
/usr/local/flumeSGF
# Setting the sink to HDFS repository
agent.sinks.flumeHDFS.type = hdfs
agent.sinks.flumeHDFS.hdfs.path =
hdfs://master/user/flume/sgf
agent.sinks.flumeHDFS.hdfs.fileType = DataStream
# Write format can be text or writable
agent.sinks.flumeHDFS.hdfs.writeFormat = Text
# Connect source and sink with channel
agent.sources. SGFExampleDir.channels = memoryChannel
agent.sinks.flumeHDFS.channel = memoryChannel
```

Fig. 7. Flume configuration file *flume.conf* that defines how the source, sink and channel are wired together to form the data flows.

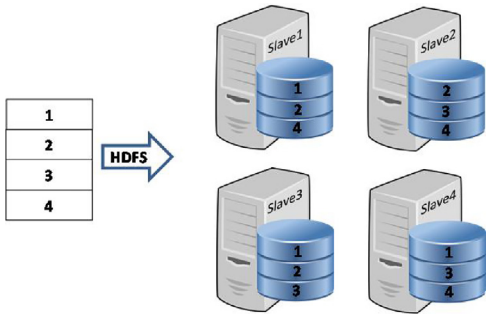


Fig. 8. HDFS distributes file blocks among cluster nodes.

ilar tools for their demanding smart grid applications. Cloudera Manager automates the installation of the essential configuration, debugging, SQL database, CDH agents and other components.

During the installation, the cluster nodes are specified using the IP address and host name (Fig. 4). It should be noted that an existing link between the cluster nodes should exist. This has been completed by establishing the SSH secure encrypted link (Section 3.1). Also, the cluster nodes are assigned master and slave roles, and there can be more than one master node in the cluster. The nodes are assigned roles that run on them. For example, nodes that perform MapReduce tasks are specified and nodes that perform a Hive task are also specified. A cluster node may be specified to perform more than one task. The status and usage of nodes can be monitored through the Cloudera Manager. This can assist decision makers in adding/reducing the number of nodes in the cluster and monitoring the reliability of nodes.

Identifying the master and slave nodes was completed during the setting up of CDH for the cluster. In the previous step, Flume ingested the data into the HDFS repository. HDFS manages storage on the cluster by breaking the incoming files into blocks, and storing each of the blocks redundantly across the slaves. In the common case, HDFS stores three complete copies of each file by copying each block to three different nodes for reliability (Fig. 8). Each block size

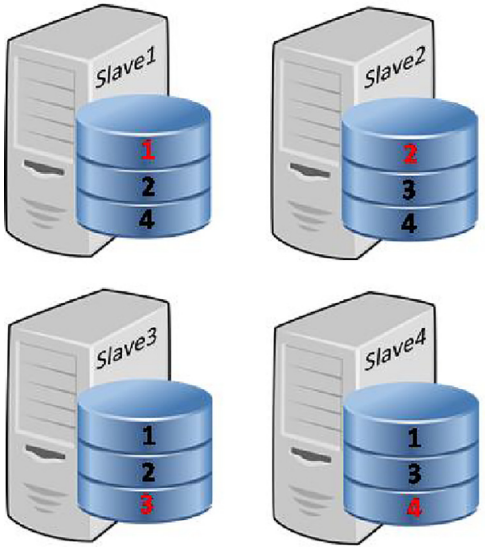


Fig. 9. CDH distributes the work out to the nodes.

```
CREATE EXTERNAL TABLE ConsumptionsTable (
  Datetimes TIMESTAMP,
  ID        BIGINT,
  Cons      FLOAT )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION 'user/flume/sgf';
```

Fig. 10. The Hive SQL-like query to build a table that includes the time stamp, ID and consumption of the smart meters.

is 128 MB by default, and can be changed to meet the application demand. However, decreasing the block size could lead to a huge number of blocks throughout the cluster, which causes the master node to manage an enormous amount of metadata. Determining the optimal size of blocks is beyond the scope of this paper.

The CDH processing component, YARN, takes advantage of this data distribution by distributing the work involved in an analysis out to many different nodes. Each of the nodes runs the analysis on its own block from the file (Fig. 9). The results are collated and digested into a single result after each involved block has been analyzed. The CDH monitors jobs during execution, and will restart work lost due to node failure if necessary.

4.4. Hive

Hive facilitates reading, writing, and managing the data stored in the HDFS repository using an SQL-like interface. In this framework Hive reads the smart grid data file from the HDFS repository, and generates a data table of interest. In this implementation it is desired to build a table that includes only the time stamp, smart meter ID and consumption of the smart meter. The SQL-like query can achieve this (Fig. 10).

This query will read data from 'user/flume/sgf' (the path where Flume sinks the data in Fig. 7), and anytime new data is ingested into this directory, the "ConsumptionsTable" table will be updated automatically.

```
CREATE EXTERNAL TABLE SingleHouse (
datetimes TIMESTAMP , Cons float, pvpower float,
windpower float, Zip STRING )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY','
LOCATION 'user/flume/sgf';
Invalidate metadata;
ORDER BY (datetimes) DESC;
```

Fig. 11. The Impala SQL-like query to build a table that includes the timestamp, consumption, pvpower, windpower and zip code.

4.5. Impala

Similar to Hive, Impala is able to read, write, and manage the data stored in the HDFS repository using SQL-like queries. However, Hive and Impala differ in the way they function and how SQL-like statements are written. The following query is an example that deals with a single house smart meter data, including power generated from solar panel (pvpower) and a residential wind turbine (windpower). The query creates a table that includes the timestamp, consumption, pvpower, windpower and zip code (Fig. 11). The "Invalidate metadata" statement is required after a table is created, to update the metadata. To respond to queries, Impala must have current metadata about the data and tables that clients query directly. Therefore, if the data table used by Impala is modified, the information cached by Impala must be updated. In cases where a delay of data occurs due to power outages, Flume will ensure the data is stored into the HDFS storage and Impala/Hive will update the tables when the data becomes available in "'user/flume/sgf'". Adding the "ORDER BY" statement will reorder the data based on the timestamp in this case.

4.6. Visual analytics

In this implementation the Tableau software is used for the smart grid big data visual analytics. The Tableau presents interactive data visualizations by means of SQL queries. Here it is desired to send SQL queries to Hive/Impala to build a visualization of data of interest. In order to accomplish this, a connection between Tableau and Hive or Impala has to be established. Open database connectivity (ODBC) interface allows applications to access data in database management systems (DBMS) using SQL as a standard for accessing the data. The remaining machine running Windows operating system is used to setup Tableau and install Hive/Impala ODBC drivers [33] by Tableau. In the Tableau software, a Hadoop server is chosen to connect to, and the machine and port are determined. The IP address or host name of one of the machines that run Hive/Impala are entered. For example, to connect to the master node the IP address should be 10.240.0.2 (from Fig. 4). In the port field, the port is the number of the transmission control protocol (TCP) port that the Hive/Impala server uses to listen for client connections. In CDH, the Hive TCP port is 10000 and the Impala TCP port is 21050. Once a connection is established, the desired HDFS data can be reached. The visualizations in Tableau are built by sending SQL-like Hive/Impala queries generated by Tableau. The Hive/Impala perform on top of the Hadoop platform which allows vastly improved speed on big datasets for prompt visual analytics.

5. Practical applications of the framework

The framework described in Section 3.3 can be applied to manage energy for a single-house, neighborhood or the entire grid. In this section, the cloud platform and Hadoop cluster to implement our framework is presented, and then the application of the framework is applied to two scenarios. In the first scenario, the framework is applied on a single-house to manage its power usage, to save power and contribute to the smooth and efficient functioning of the electric grid. In the second scenario, the framework is applied on a recently available smart metering data set that consists of 6436 homes and businesses. The framework was hosted on an IaaS Google cloud platform that consists of six machines. A Hadoop cluster was setup on five machines with one master node and four slave nodes. The master node is a 2.6 GHz, 7.5 GB RAM running 64-bit Linux operating system. All slave nodes were 2.6 GHz, 3.75 GB RAM running 64-bit Linux operating system. The remaining machine was a 2.6 GHz, 3.75 GB RAM running 64-bit Windows operating system to run Tableau to perform the visualization tasks. A secure encrypted link between the cluster nodes was setup using the SSH protocol.

In the first scenario, beside the typical household appliances that consume power, this house includes micro-power generators namely, a residential wind turbine and rooftop photovoltaic (PV) solar panels. Furthermore, an EV is included in the scenario, as EVs are a hot topic in smart grids. The household electric power consumption data was obtained from the UCI data repository [34]. It includes the global active power and three sub-meterings. The first sub-metering corresponds to the kitchen, containing mainly a dishwasher, an oven, and a microwave. The second sub-metering corresponds to the laundry room, containing a washing-machine, a dryer, a refrigerator and a light. The third sub-metering corresponds to a water heater and an air-conditioner. To calculate the power output of the wind turbine and PV solar panels, the wind speed, temperature and irradiation data were obtained from [35] with latitude of 39.74°N and longitude of 105.18°W. Due to the lack of data sets that include the power consumption and micro-power generators' data, it is assumed that the house exists in a location where the data of micro-power generators is available. For that, it is assumed that the house is located with latitude of 39.74°N and longitude of 105.18°W, which is the same coordinates of the data of the wind speed, temperature, and irradiation. The wind turbine considered was a 3 kW residential turbine. The wind turbine's power output was calculated using the weather data (i.e., the wind speed and air density) of the specified location and the wind turbine's data sheet [36] put in the formulas in Appendix A. The number of rooftop PV solar panels [37] was ten, and the method to calculate the power output with respect to the weather data (i.e., ambient temperature and solar irradiation) for the location is described in details in Appendix B. The EV charging (G2V) profile was obtained from a study of EV driver recharging habits in the north east of England [38], and the EV discharging (V2G) habit was obtained from Ref. [39] as it suggested that 10% of the EV's energy can be discharged into the network. The data generation rate in this application is one-minute for all sub-metering. Fig. 12(a) illustrates the consumption and generation of the house with one-minute time resolution using the Tableau visualization tool. While Fig. 12(b) illustrates a dashboard for the power status of the aforementioned house updated every one minute. The consumption of power (household loads and G2V) is in red color whereas the aggregated generation from the PV panels, wind turbine and V2G is in blue (Fig. 12(b)). A pie-chart can be added on the location of the houses to observe the average consumption/generation power (Fig. 12(c)). This can be applied to other houses on the map if data were available.

Due to the lack of lengthy data sets similar to the data used in the first scenario that consider micro-generators, in the second scenario a large smart grid data set is considered. The reason for applying the first scenario was to show that micro-generators, including EV systems, can be included in the framework. In the sec-

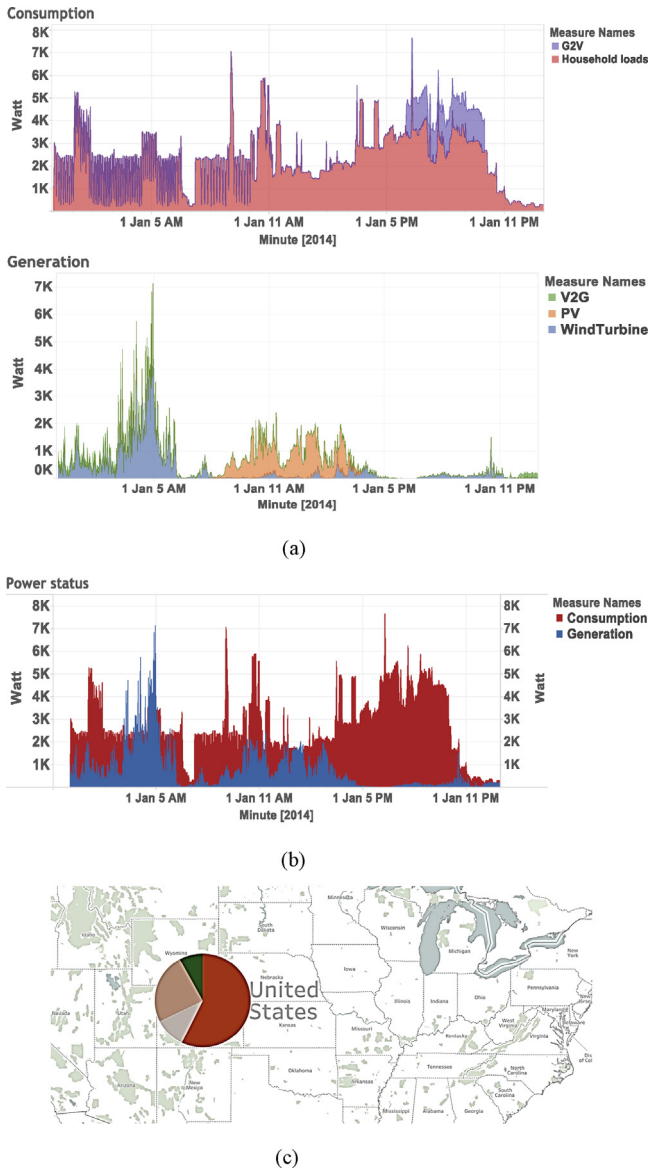


Fig. 12. Dashboards for power status. (a) cumulative consumption and generation with one-minute time resolution. (b) power status of the house. (c) Map with pie-chart for consumption (red) and generation for the house. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ond scenario, it is also desired to test the efficiency of the framework in handling lengthy smart grid data.

In the second scenario, a smart metering electricity behavior data set from the Irish Social Science Data Archive [40] that took place from July, 2009 to December, 2010 for 6436 Irish homes and businesses with a 30-min time resolution is used to test the feasibility of the framework. Each smart meter produced around 25,730 electricity consumption time series readings during the mentioned period. This corresponds to over 165 million electricity consumption readings to be ingested. The data generation rate of each smart meter was 30-min. Thus, 6436 smart meter data observations were ingested every 30-min. Each observation contains the time-stamp, smart meter ID number, and electricity consumption. Utility companies may have access to additional data about their customers, e.g., location and square footage of the home. However, this information is usually not available to third-party applications. The data were individually ingested to a master node in the Hadoop cluster using Flume. Once the data are collected, it is stored into an HDFS

```
Operator          #Hosts   Avg Time    Max Time
----------------------------------------------
06:AGGREGATE           4  233.197ms  360.253ms
05:EXCHANGE            4  201.407us  247.779us
04:AGGREGATE           4  353.504ms  419.331ms
00:UNION               4  991.916us    1.796ms
|--02:SCAN HDFS        4    8.568ms   14.741ms
|--03:SCAN HDFS        4    2.426ms      4.5ms
01:SCAN HDFS           4   11.151ms    12.75ms
```

Fig. 13. Impala query execution time breakdown to produce/update the incoming 6436 smart meter data.

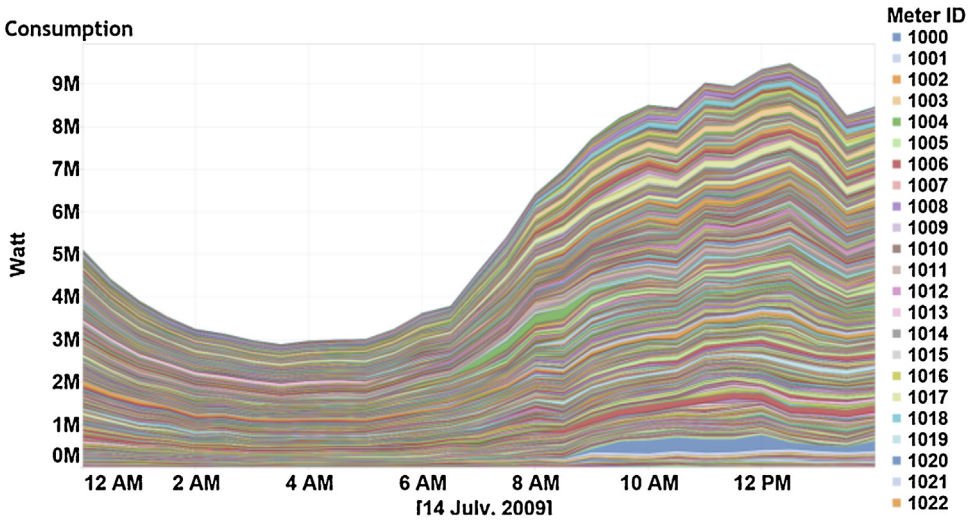
infrastructure. The SQL queries to read and arrange tables from the electricity consumption of the 6436 smart meter readings are run in Hive and Impala. As Hive and Impala differ in the way they function, it was desirable to test both to observe which one can perform faster. In our Hadoop cluster, both Hive and Impala were able to produce/update the table with the newly stored 6436 smart meter data (i.e., the time-stamp, smart meter ID number and electricity consumption from each smart meter every 30-min) in a comparable amount of time. Fig. 13 presents a breakdown for the time execution of the used Impala query to produce/update the smart meter readings. In this application, we focus on developing a dashboard to visualize the status of the smart grid for DDR purposes. To achieve this, Tableau visualization software was used. The Tableau software connects with the Hadoop cluster through Hive or Impala queries to achieve near real-time visualization. Here Impala was able to outperform Hive in updating the visualization of the smart grids status. This suggests that Hive can be suitable for big data batch processing, whereas, Impala can satisfy the requirement of near real-time big data processing [41]. Fig. 14(a) presents a dashboard for the status of the on-hand smart grid with the aggregated consumption of 6436 Irish homes and businesses. Consequently, a DDR strategy by analytics can be suggested during peak load periods. Utilities could proactively identify abnormal conditions and take action to prevent power outages and promote the grid reliability. Also, dynamic power pricing and incentives for reducing load during peak periods can be advertised for. However, this is beyond the scope of this paper. The locations of the smart meters' locations can be visualized on the map similar to Fig. 12(c). Also, a view of the power consumption for a certain region or specific smart meters can be obtained (Fig. 14(b)) for further analysis, for example, identifying neighborhood groups with distinct load profiles.

6. Discussions and conclusions

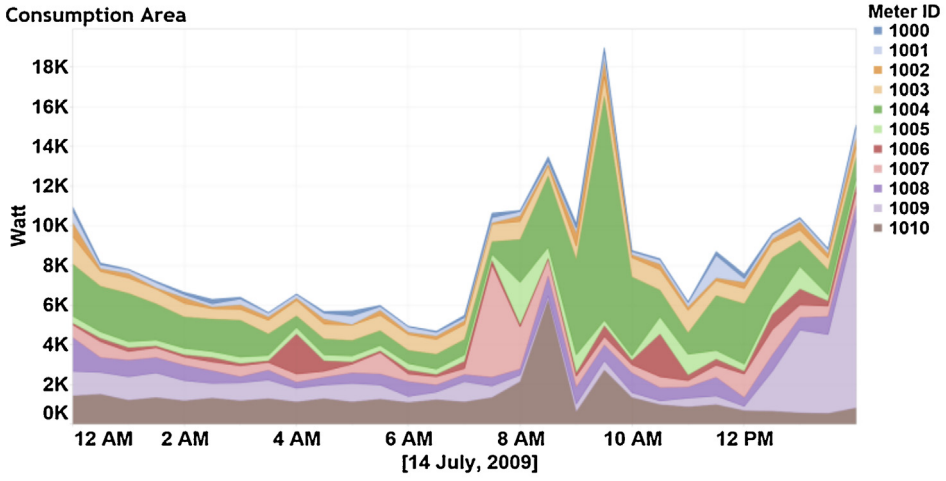
6.1. Discussion

The framework presented in this paper was able to acquire, store, process and query the massive amount of data in near real-time. As Hive and Impala differ in the way they function, both were able to produce/update the newly stored 6436 smart meter data in a comparable amount of time. However, it was observed that Hive can be suitable for big data batch processing, whereas, Impala can satisfy the requirement of near real-time big data processing. The Hive and Impala queries that build the smart meter data tables were presented. It should be noted that the power of the Hadoop platform is also capable of handling and processing online smart grid applications where real-time operation is required. Also, this paper covered a DDR task, by enabling the smart grid users to share and visualize its information. This can present following benefits:

- The integration of renewables by using demand side management to address supply fluctuations is promoted.
- The grid's reliability can be increased by using the customers as a virtual power resource during peak periods (negative demand is equivalent to a positive supply).



(a)



(b)

Fig. 14. Dashboards for power status in the smart grid. (a) Power consumption of 6436 Irish home and businesses updated every 30-min. (b) Power consumption of 11 selected smart meters.

- This can elude the need to build new power plants for standby generation, by lowering the peak periods and advertising for incentives for reducing loads.
- The environmental impacts can be limited as monitoring micro-generators are included, especially V2G power.

Based on the research presented in this paper, some of the applications that can be carried out in the future are summarized in the following:

- Applying data mining tools such SAMOA on near real-time smart grid data streams for analyzing and forecasting.
- Consumer-oriented applications to provide feedback to end-users on reducing electricity consumption and saving money through smartphone push notifications.
- Producer-oriented applications to provide information about consumers, such as their daily habits for the purposes of sending incentives or clustering customers.

6.2. Conclusions

This paper presented a big data framework for smart grids. The concept of big data and the core components of the framework were highlighted. The framework's stages including, data acquisition, data storing and processing, data querying, and data analytics components were discussed in details. Furthermore, the functionality of Apache Hadoop platform and the features that make it suitable for the smart grid big data management and analysis were discussed.

The framework was implemented on a secure IaaS cloud-based platform, and relevant configurations and source codes were presented. Also, the interfaces between various compnent combinations that have been able to successfully communicate together are build this framework are presented. The framework was hosted on an IaaS Google cloud platform which presents a cost-effective, improved accessibility, and scalability infrastructure. The secure link between the framework's cluster nodes was established by adopting the Secure Shell version 2 (SSH) protocol. Further, a detailed configuration for the data acquisition component, flume,

was presented. For simplicity, the Cloudera Manager Hadoop distribution was used to install the essential configurations, CDH agents and other components for the cluster nodes. It was observed that the use of such open-source Hadoop distributions was comparable to installing the Hadoop components separately. However, setting up the Hadoop components separately is not likely an option for average-users. Moreover, the CDH offers graphical user interfaces that assist in setting up and monitoring the cluster nodes. Consequently, this achieves the intention of utilizing open source state-of-the-art tools, and providing an easy and cost-effective development environment for practicing engineers to replicate and develop for their demanding smart grid applications. Furthermore, the application of the framework was applied to two scenarios. The first scenario was a single-house that included micro-generators (i.e., wind turbine, PV roof panels and EV). The second scenario included a real smart metering electricity behavior data set from the Irish Social Science Data Archive for 6436 participating Irish homes and businesses.

Finally, the impact of the proposed framework goes beyond visual analytics, but in this paper the main objective was to introduce a framework that can handle the massive smart grid data. Service providers and researchers could follow the framework guideline to conduct research in smart grid big data domains; and improve the electrical grid. The application of the two scenarios and the visualization of the grid's status, suggests that this framework is feasible in performing further smart grid data analytics. As this framework was dedicated to the needs of smart grid big data, it is worth noting that the practical implementation of the framework, including necessary configuration and coding, and application, can assist in developing big data frameworks for other disciplines that can generate profit for businesses, promote the development of sciences and technology and improve decision making for government sectors.

Appendix A. Wind turbine power output

(A1) Wind turbine power output

$$P_w = \frac{1}{2}\rho A v^3 C_p \tag{A1}$$

where ρ is the air density, A is the swept area, v is the wind speed and C_p is the power coefficient.

Appendix B. PV power output

(B1) PV power output

The output power from the PV panels was calculated using the irradiance and ambient temperature by following two steps [42]:

1. The calculation of the output dc power (P_{dc}) generated from the PV panel using a PV model and the PV module data sheet:

$$T_c = T_{amb} + \left(\frac{NOCT - 20}{0.8}\right).S \tag{B1}$$

$$P_{dc} = P_{max}\left(S/1000\right)\left[1 - 0.005\left(T_c - 25\right)\right] \tag{B2}$$

where T_c is the cell temperature, T_{amb} is the ambient temperature, S is the irradiance, $NOCT$ is the operating cell temperature at Standard Test Conditions (STC: $S = 1000\,\mathrm{W/m^2}$, $T_{amb} = 25\,^{\circ}\mathrm{C}$) and P_{max} is the maximum rated power.

2. The calculation of the output ac power (P_{ac}) generated from the PV system using the manufacturer's efficiency curve:

$$P_{ac} = P_{dc} \times \eta_{mismatch} \times \eta_{dirt} \times \eta_{inverter} \tag{B3}$$

where $\eta_{mismatch}$, η_{dirt} and $\eta_{inverter}$ are array to mismatched modules, dirt loss and inverter efficiency, respectively.

References

[1] J. Gantz, D. Reinsel, The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east, IDC Anal. Future (2012).
[2] H. Hu, Y. Wen, T.-S. Chua, X. Li, Toward scalable systems for big data analytics: a technology tutorial, IEEE Access 2 (May) (2014) 652–687.
[3] X. Wu, X. Zhu, G.Q. Wu, W. Ding, Data mining with big data, IEEE Trans. Knowl. Data Eng. 26 (January (1)) (2014) 97–107.
[4] M. Simoes, S. Mohagheghi, P. Siano, P. Palensky, X. Yu, Advances in information technology for smart grids, Proc. Ind. Electron. Soc. (2013) 36–41.
[5] A. Katal, M. Wazid, R.H. Goudar, Big data: issues, challenges, tools and good practices, Proc. Contemp. Comput. (2013) 404–409.
[6] Y. Demchenko, C. deLaat, P. Membrey, Defining architecture components of the big data ecosystem, Proc. Collab. Technol. Syst. (May) (2014) 104–112.
[7] C.L. Stimmel, Big Data Analytics Strategies for the Smart Grid, CRC Press, 2015, pp. 155–169.
[8] X. He, Q. Ai, C. Qiu, W. Huang, L. Piao, H. Liu, A Big Data Architecture Design for Smart Grids Based on Random Matrix Theory, 2017, Available [Online]: https://arxiv.org/abs/1501.07329.
[9] A.A. Munshi, Y.A.I. Mohamed, Cloud-based visual analytics for smart grids big data, Proc. IEEE Innovative Smart Grid Technologies (2016).
[10] K. Slavakis, G.B. Giannakis, G. Mateos, Modeling and optimization for big data analytics: (statistical) learning tools for our era of data deluge, IEEE Signal Process. Mag. 31 (September (5)) (2014) 18–31.
[11] X. He, R.C. Qui, Q. Ai, Y. Cao, J. Gu, Z. Jin, A Random Matrix Theoretical Approach to Early Event Detection in Smart Grids, 2017, Available [Online]: https://arxiv.org/pdf/1502.00060.pdf.
[12] Y. Simmhan, S. Aman, A. Kumbhare, R. Liu, S. Stevens, Q. Zhou, V. Prasanna, Cloud-based software platform for data-driven smart grid management, IEEE/AIP Comput. Sci. Eng. 15 (4) (2013) 38–47.
[13] O.R. Team, Big Data Now: Current Perspectives from O'Reilly Radar, O'Reilly Media, Sebastopol, CA, USA, 2011.
[14] M. Grobelnik, Big Data Tutorial, 2012, Available [Online]: http://videolectures.net/eswc2012_grobelnik_big_data/.
[15] J. Manyika, et al., Big Data: The Next Frontier for Innovation, Competition, and Productivity, McKinsey Global Institute, San Francisco, CA, USA, 2011, pp. 1–137.
[16] P. Zikopoulos, C. Eaton, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGraw-Hill, New York, NY, USA, 2011.
[17] E. Meijer, The world according to LINQ, Commun. ACM 54 (August (10)) (2011) 45–51.
[18] D. Laney, 3D Data Management: Controlling Data Volume, Velocity and Variety, Gartner, Stamford, CT, USA, 2001, White Paper.
[19] P. Zikopoulos, D. deRoos, C. Bienko, R. Buglio, M. Andrews, Big Data: Beyond the Hype, McGraw-Hill, New York, NY, USA, 2015.
[20] VMware, Apache Flume and Apache Sqoop Data Ingestion to Apache Hadoop Clusters on Vmware Vsphere, VMware, CA, USA, 2013, White Paper.
[21] T. White, Hadoop The Definitive Guide, O'Reilly Media, Yahoo! Press, Sebastopol, CA, USA, 2012.
[22] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, in: Proc. IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), May, 2010, pp. 1–10.
[23] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, in: Proc. 6th Symposium on Operating System Design and Implementation (OSDI'04), December, 2004.
[24] S. Ghemawat, H. Gobioff, S.-T. Leung, The google file system, in: Proc. 19th ACM Symposium on Operating System Design and Implementation, October, 2003.
[25] V.K. Vavilapalli, et al., Apache hadoop YARN: yet another resource negotiator, Proc. 4th Symposium on Cloud Computing (2013).
[26] A. Thusoo, J.S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, Hive: a warehousing solution over a mapreduce framework, Proc. Very Large Data Bases 2 (August (2)) (2009) 1626–1629.
[27] J. Li, Design of real-time data analysis system based on Impala, in: IEEE Workshop in Advanced Research and Technology in Industry Applications, September, 2014, pp. 934–936.
[28] A. Bifet, G.D.-F. Morales, Big data stream learning with SAMOA, IEEE Data Mining Workshop (2014) 1199–1202.
[29] M.A. Rahman, M.H. Manshaei, E. Al-Shaer, M. Shehab, Secure and private data aggregation for energy consumption scheduling in smart grids, Trans. Dependable Secure Comput. 14 (June (2)) (2015).
[30] Office of the National Coordinator for Smart Grid Interoperability, NIST Framework and Roadmap for Smart Grid Interoperability Standards, 2010.
[31] N. Serrano, G. Gallardo, J. Hernantes, Infrastructure as a service and cloud technologies, Softw. IEEE 32 (2) (2015) 30–36.
[32] F. Fakhar, M.A. Shibli, Management of symmetric cryptographic keys in cloud based environment, Proc. Adv. Commun. Technol. (January) (2013).
[33] Cloudera Inc, Cloudera ODBC Driver for Impala, 2016, Available [Online]: http://www.cloudera.com/documentation/other/connectors/impala-odbc/latest/Cloudera-ODBC-Driver-for-Impala-Install-Guide.pdf.
[34] M. Lichman, UCI Machine Learning Repository, 2013 http://archive.ics.uci.edu/ml.
[35] Solar Radiation Research Laboratory (BMS), 2017, Available [Online]: http://www.nrel.gov/midc/srrl_bms.
[36] Kingspan wind, KW3 Small Wind Turbines, RAL 9005, Datasheet.

[37] SUNPOWER, E20/435 Solar Panel, SPR-435NE-WHT-D Datasheet, 2011.
[38] A.P. Robinson, P.T. Blythe, M.C. Bell, Y. Hubner, G.A. Hill, Analysis of electric vehicle driver recharging demand profiles and subsequent impacts on the carbon content of electric vehicle trips, Energy Policy 61 (2013) 337–348.
[39] M. Alonso, H. Amaris, J.G. Germain, J.M. Galan, Optimal charging scheduling of electric vehicles in smart grids by heuristic algorithms, Energies 7 (4) (2014) 2449–2475.
[40] Irish Social Science Data Archive (ISSDA), 2017, Available [Online]: www.ucd.ie/issda.
[41] M. Kornacker, et al., Impala: a modern, open-source sql engine for hadoop, Proc. Innov. Data Syst. Res. (CIDR'15) (January) (2015).
[42] G.M. Masters, Renewable and Efficient Electric Power Systems, John Wiley & Son, 2004, pp. 505–531.