Journal of Parallel and Distributed Computing 133 (2019) 40–50

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

On construction of a big data warehouse accessing platform for campus power usages

Chih-Hung Chang [a], Fuu-Cheng Jiang [b], Chao-Tung Yang [b,*], Sheng-Cang Chou [b]

[a] College of Computing and Informatics, Providence University No. 200, Sec. 7, Taiwan Boulevard, Shalu Dist., Taichung City 43301, Taiwan, ROC
[b] Department of Computer Science, Tunghai University No. 1727, Sec.4, Taiwan Boulevard, Xitun District, Taichung City 40704, Taiwan, ROC

HIGHLIGHTS

- Real-time power monitoring platform based on the smart meter.
- Data collection and storage method using the Hadoop subsystem and Hive data warehouse with Spark.
- Query-response efficiency testing of Apache Hive, Apache Spark, and Impala.
- Performance testing on the statistical query function and data ETL processing of Apache Hive, Apache Spark, and Impala.

ARTICLE INFO

Article history:
Received 1 October 2018
Received in revised form 22 February 2019
Accepted 15 May 2019
Available online 24 June 2019

Keywords:
Internet of Things
Big data warehouse
Smart meter
Data ETL
Real-time processing

ABSTRACT

With the emerging of the Internet of Things (IoT) technology, we can analyze and real-time monitoring the power consumption data of buildings or equipment. However, over time, many power data will accumulate to even the terabyte level. Traditional data processing techniques will not handle. It is a challenge to monitor and process the data in a reasonable time. In this paper, we construct an efficient and real-time power monitoring platform, based on open source techniques. The power-data of buildings or equipment is provided through smart meters equipped inside campus buildings. The technologies of data collected and stored on the server side were handled by the Hadoop ecosystem and the data process to Hive data warehouse is executed by the Spark. On the particulars of system evaluation, we make some experiments to compare with the performance of record query, use three different technologies separately: Apache Hive, Apache Spark, and Impala had been evaluated regarding query-response performance. Moreover, for the performance estimation on the data ETL processing and SQL functions. The relevant experiments have been conducted on the same three software modules as well.

1. Introduction

Big data and the Internet of Things (IoT) are sparked the world to explore more benefit of these technologies. The combination of big data and IoT align the technologies in the best possible way for data analyzing. However, when the organizations intend to implement data analysis using these two technologies, they should concern about the architecture. Traditional architecture cannot handle the massive amount of data]generated by multiple sensors. Through deploying a massive amount of smart meters on organization or campus. The log data of smart meters will update in every two seconds. The collecting data from the sensors will accumulate many historical log data. Thus, the data volume of processed shows rapid growth. It can be expected the performance of the traditional relational database will be unable to satisfy the system demands. As the data size would be increased to hundreds of gigabytes, even up to Terabytes magnitude. It is necessary to build a high-performance data processing environment to serve the demands of the storage and manage the data from the smart meters, long-term historical data, and to analyze electricity consumption.

Hadoop is an open-source framework that allows the user stored and processed large volume data in a distributed environment across clusters of computers using simple programs. Hadoop can be constructed by a single server to thousands of computers, each computer offering its capabilities of computation and storage. Hadoop has been the mainstream framework of the cloud computing technology, which HDFS and MapReduce are the two core technologies. Hadoop manages the data by breaking up the files into blocks and distributes them to the nodes of the Hadoop cluster. In general, Hadoop has better scalability,

* Corresponding author.
E-mail addresses: ch.chang@gm.pu.edu.tw (C.-H. Chang), admor@thu.edu.tw (F.-C. Jiang), ctyang@thu.edu.tw (C.-T. Yang), minicp9523@gmail.com (S.-C. Chou).

https://doi.org/10.1016/j.jpdc.2019.05.011
0743-7315/

reliability, and usage efficiency of equipment than traditional data processing. Hive is a data warehouse system tool built on the top of Hadoop that can be used to handle structured and semi-structured data. After changing the overall storage architecture, the data stored in HDFS can be managed more easily.

In this manuscript, an architecture of a Big Data Warehouse Accessing Platform for Campus Electricity Loads Usages will be proposed. In this case, we concern three purposes as follows:

- To design and implement a real-time electricity load monitoring platform.
- To visualize the real-time monitoring power usage using Hue web user interface.
- To test and evaluate the dig data SQL query-response efficiency, and performance evaluations for the proposed data warehouse and processing platform.

The rest of this paper is organized as follows. Section 2 describes background materials and relevant research works. Section 3 demonstrates the proposed system architecture which includes big data processing functions like extraction feature, data transformation, and load data processing function. The related performance evaluation and experimental results with analysis are presented in Section 4. Section 5 gives a discussion and summary to the proposed system and future works.

2. Background review

In this section, we will discuss related background knowledge.

2.1. Internet of things (IoT)

Internet of Things (IoT) [19] is about every object, including the general items, animals and even people are equipped with a UID (Unique Identifiers). The data and information about objects hooked up on any computer networks, like wired-LAN or Wireless LAN, can be shared directly through the Internet. It is no longer to rely on the interaction between people or people and machines. The future will be the world of machine-to-machine (M2M), directly by the machine to complete a variety of work on the machine. From a practical point of view, the concept of IOT can be divided into three-tier architecture, from the bottom to the upper layer are respectively sensing layer, network layer, and application layer.

2.2. Cloud computing

The term Cloud Computing comes from Google CEO Eric Schmidt, who presented the idea for the first time on August 9, 2006, at the SES San Jose. According to the definition of National Institute of Standards and Technology (NIST) [7] on Cloud Computing on May 2012: cloud computing is a model that provides ubiquitous, convenience, on-demand, and shared the resource that can be rapidly provisioned and released with minimal management effort or service provider interaction. It is composed of five essential characteristics, three service models and four deployment models.

Five essential characteristics:

- On-demand self-service,
- Anytime, anywhere access by any network device,
- Resource pooling,
- Quick redeployment, and
- Can be monitored and measured.

Three service models:

- Infrastructure as a Service (IaaS): IaaS is the way that users can use the computing resources, such as the processor, storage capacity, and network through renting to cloud service providers but not buy hardware and build their own infrastructure.
- Platform as a Service (PaaS): PaaS is a cloud computing service to deliver hardware and software tools for those customers need for application development.
- Software as a Service (SaaS): Consumers utilize software available or data stored in the cloud, but without managing cloud infrastructure and programming execution environment. No longer do customers need to install software on their computer, therefore reducing maintenance works and software support issues.

Four deployment models:

- Public Cloud,
- Private Cloud,
- Hybrid Cloud, and
- Community Cloud.

The definition of cloud computing is showing as Fig. 1.

2.3. Big data

Big data [8,12,15,20,24] is a term that has been in use since the 1990s. As the data volume grows explosively and ubiquitously, the traditional techniques for data processing applications are inadequate to deal with ever-growing data volumes. Almost 90 percent of the data in the world was generated during the past two years. According to International Data Corporation (IDC), in 2013 there are 4.4 zettabytes data in the world, and they predict it will soon reach 44 zettabytes in 2020. Big Data is also a way to deal with a great volume of the structured, semi-structured and unstructured data.

Bioinformation analysis is also a big issue for big data. Such EEG analysis [11,21]. In Chen et al. [10] they are using a hierarchical parallel processing framework over a GPU cluster to process big data. The other solution is using GPU to speed up the processing performance, such as Chen et al. [11], they use a multiway parallel factor data analysis of the electroencephalography (EEG), which is using general-purpose computing on the graphics processing unit.

2.4. Hadoop ecosystem

Apache Hadoop [1] is an open-source software framework that is been widely applied for big data processing at present. Start from the Google File System paper which was published in October 2003 and the paper of MapReduce. The Hadoop framework is constructed on the top of the Hadoop Distributed File System (HDFS). Hadoop provided a reliable and distributed processing system. Hadoop runs the MapReduce [2,14] programming model, which divides data files into the same block size, these can be deployed and processed parallelly in the computing cluster. Hadoop is designed to serve parallel computing and enhance the computing capability of a single server to multiple computers.

The HDFS [3,9] is a distributed file system designed to run on low-cost hardware and provides high throughput access to very large data application data (larger to terabytes or petabytes). HDFS has the highly fault-tolerant capability which data are

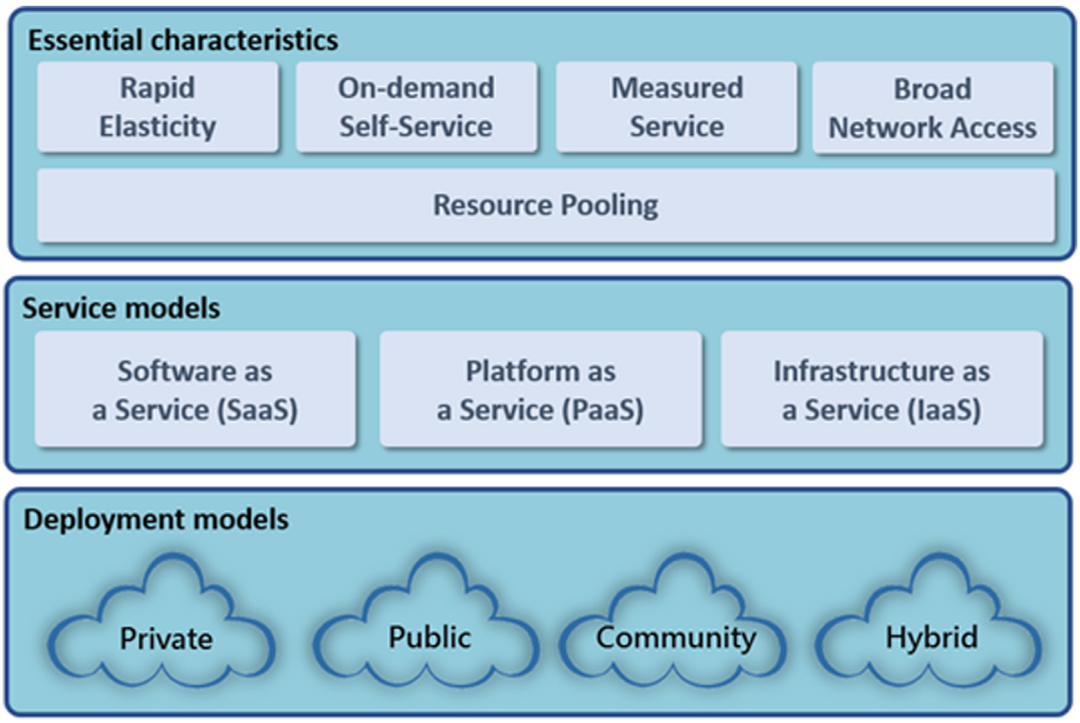


Fig. 1. Definition of cloud computing.

stored on multiple machines redundantly to ensure their durability to failure and high availability to parallelize applications [26]. HDFS has many resemblances with other existing distributed file systems but is not alike in some points. One striking difference is HDFS's write-once-read-many model that it simplifies concurrency control and data coherency to enhance high-throughput access [27]. Another distinct property of HDFS is that it usually deployed processing program near the data instead of moving the data to the location of programs. HDFS limit data writing to one writer at one time rigorously. HDFS consists of the interconnected node clusters in which the files and directories. The HDFS cluster consists of a node called Namenode, and the other nodes called Datanode. In HDFS, the Namenode manages file System namespace operations, such as opening, closing, and renaming files and directories. The Namenode also images the block of data to the Datanode, which processes read and write requests from the HDFS client. The Datanode also creates, deletes, and replicates blocks of data based on the instructions in the Namenode. Fig. 2 shows the architecture of HDFS.

Apache Hive [4] is an open-source data warehouse project of Apache Software Foundation. Hive has integrated data storing, querying and managing for large datasets. Hive is a data warehouse applications build upon of the Hadoop MapReduce. It let users deal with the data stored in Hive same as the data is stored in a regular database. HiveQL is SQL-like language let user storage and requests data in Hive. Hive lets the user who has experience using traditional RDBMS to run familiar queries on MapReduce [22]. Hive's advantages are as follows:

- Very powerful at big data storing
- Easy to learn and understand
- Portable, multiple data views
- Used with and DBMS system with vendor
- Well defined standards exist and using relational databases
- High performance, integrates with Java

Fig. 3 shows the relation between Hadoop and Hive.

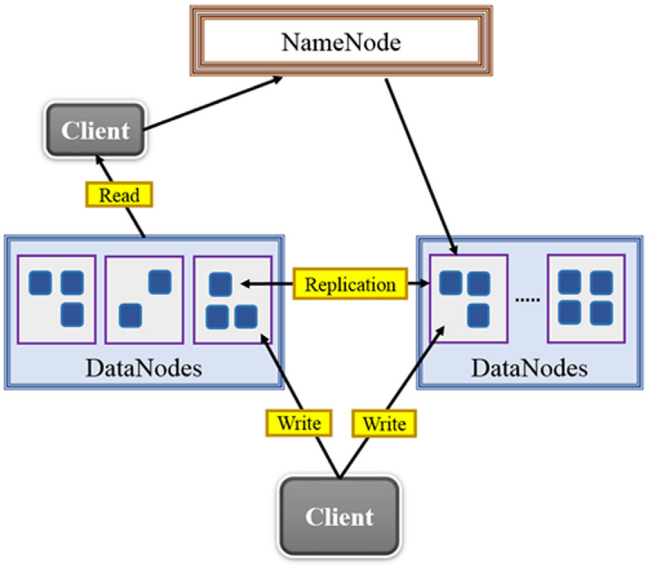


Fig. 2. The architecture of HDFS.

Sqoop [5,23] is a tool transferring data between traditional relational database and NoSQL. With Hadoop MapReduce parallel characteristic, Sqoop can speed up data migration through batch processing. Sqoop is an efficient tool to migrate data from a relational database to Hive, HDFS, and HBase; it is also supporting table importing for the full table and incremental table (see Fig. 4).

2.5. In-memory processing framework

Apache Spark [28] is an open source framework for large scale data processing. Compare with MapReduce of Hadoop, Spark used

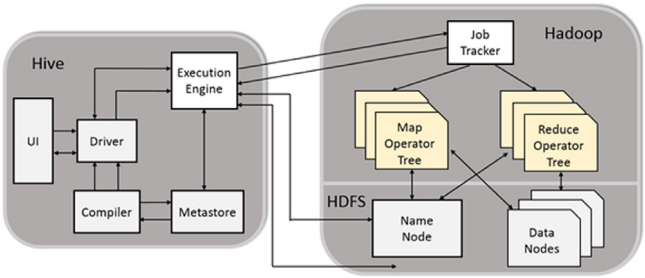


Fig. 3. Relation between Hadoop and Hive.

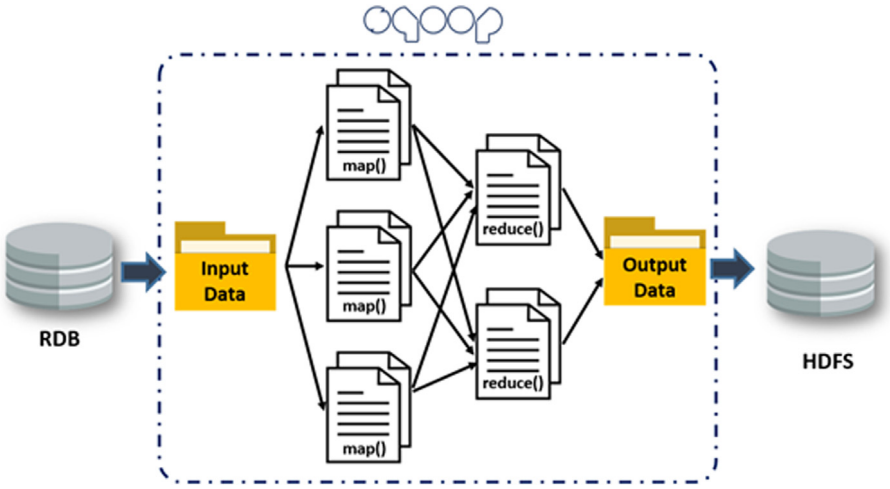


Fig. 4. Sqoop workflow.

In-memory technology to achieve speed up the performance to 100x faster for certain applications. Spark allows the user to load data and process on the memory of the cluster and query it repeatedly. In Spark environment include a cluster manager and a set of the distributed storage system. For the storage cluster, Spark supports standalone, Hadoop YARN, or Apache Mesos.

In particular, Spark SQL [6] is using to execute SQL queries in distributed in-memory computations. Spark SQL can also utilize to read data from a Hive environment.

Impala [16] is an open source. It provides low latency and high-performance analytic queries on Hadoop. For Hive user, Impala utilizes the same metadata and ODBC drivers. It also supports SQL to query data stored in HDFS and HBase without re-inventing the implementation or transformation. However, Impala is memory based which does not run heavy data operations effectively, such as joins operation.

2.6. Related works

The data analysis of the smart meter is often bundled with socio-economic information in the analysis, such as instrument geography, weather, and user behavior. It lets the data source very large and complex to analyze. Liu Xiufeng et al. [17] raise a mechanism to integrate information and extracting data from the smart meters. They also implemented a platform for large data handle and data mining and visual analysis of the result by the web portal. Their system is a hybrid architecture that uses Spark or Honeycomb for big data processing and uses the Madelib toolkit to perform in-database data analysis in a Posture SQL.

In addition to the advantages of Apache Spark and Hadoop HDFS compatibility, as well as the use of distributed memory technology, it also allows for repeated calculations of data from in-memory cached data, since the primitives in Spark memory provide performance for some applications. As the experimental result of Y.Z. Yan et al. [25] in the previous show the Spark achieve high performance for both batch and streaming data, the performance is faster than Hadoop, forasmuch we use Spark as this big data process framework.

About the OLAP technology in a big data application, a big data analysis for power equipment monitoring system is a good practice. The platform includes an online analysis of Hive-based relationships, online analysis of Impala-based relationships, and multi-dimensional online analysis based on HBase. Solving the problem of high cost and slow query speed of distributed relational analysis data model connection, Wang et al. [13], a power equipment condition monitoring data model based on non-connected level coding technology is proposed. In order to optimize performance, they reduce the number of connections, the level information in the dimension table is encoded and compressed into the fact table.

SQL is an exceptional purpose of the programming language that has been utilized to process information in relational databases for a long time. Despite the fact that SQL is not appropriate for the complex analysis. SQL has been using numerous venture engineers and business experts as a result of its ease of use. In Ilias Mavridis et al. [18], they made experiments of distributed SQL querying real data using Apache Hive and Spark SQL. Through numerous analyzes, they presumed that Spark SQL has more efficieny than Hive. The reason is Spark SQL uses cache of tables in memory to optimize efficiency.

3. System design and implementation

In this section, we present the architecture of our proposed electricity load data warehouse and the ETL service of data warehousing.

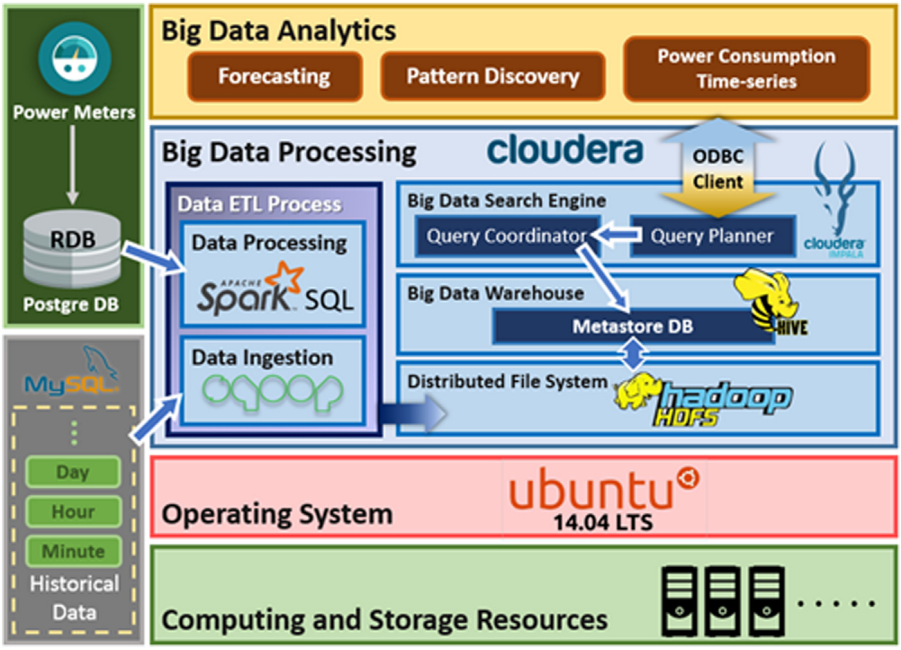


Fig. 5. The overview of system architecture.

3.1. System architecture

The objective of this system must be received and deal with electricity data per second from the sensors all over campus for power monitoring, notification system, analysis, and related administrate functions. Therefore, the scalability, performance, agility and adaptability of the system are very crucial. To achieve these abilities, the proposed system has multi-layer architecture, such as data generation and collection, data processing, and data analysis for the vast amount of power data. This multi-layer architecture is shown in Fig. 5.

The lowest layer of Fig. 5 is responsible for computing and storing resources. This layer is constructed with one master node with 10 processors and 128 GB (Gigabyte) memory and four slave nodes with 8 processors and 16 GB memory. Chapter 4 will explain the hardware specification in detail. Fig. 6 shows a summary of system hardware and network specification.

In the second layer is the OS layer, we use Ubuntu Linux 14.04 LTS as the operating system. The third layer is a major part of the system. This layer consisted of the big data ETL process, storage, and search engine. The data source of the ETL process includes raw data from smart meters and the historical data which be stored in MySQL database. For data ingestion, we use Spark SQL as a job scheduling module to gather the raw data on smart meters; Apache Sqoop is used as a transfer tool to directly ingest historical data that stored in MySQL to a data warehouse which construction by Hive.

In the part of big data search engines, Impala integrates with the Hive Metastore database to share databases and tables between two modules. Impala provides high-performance and low-latency SQL operations such as Hive or for data stored in Hadoop file format. In order to obtain high-performance data analysis in the future, such as pattern discovery, power consumption analysis of time series, Impala plays a major role in OLAP services.

In the term of power information collection, we used the wireless multifunction power meter: WPM-100 (as Fig. 7) to gather data, such as voltage, current, power, power factor, and frequency, and so on. It can easily deploy in buildings and transmit through the wireless to reduce the overhead of network cabling.

Historical instrumentation, data can be transferred from relational databases to HDFS via Sqoop. Sqoop is a data transfer tool that accelerates the data migration process using Hadoop MapReduce parallelism to transferee data from traditional relational database. It not only supports transferring data from MySQL, Hive and HBase to HDFS.

The data ETL service transfers raw data to the data warehouse through ETL procedures. The raw data consist of real-time data from the data center and data from the campus building and accomplished by periodically executing the data ETL service. Fig. 8 shows the use case diagram of the data ETL service.

To achieve real-time presentation on front-end web user-interface, the process of calculating raw data in different kinds of processed data period is required. By the reason of the efficiency of Spark, Scala is our first choice to program the periodic processing application. There are several types of data that is needed to generate, such as accumulated power data in each minute, hour, day, month, and year. Another benefit of adopting spark as back-end processing is to reduce the burden of the front-end.

3.2. System implementation

In this subsection, we illustrate the big data warehouse service clusters which consist of 4 physical machines, 1 machine as a master, 3 machines as the computing nodes to establish the big data platform, which constructed by Spark, Sqoop2, Hive, and Impala.

In the big data warehouse and processing platform architecture, we used the recent version of CDH (a Cloudera's open source platform) as a big data service platform. The reason we did not use native Hadoop is that CDH not only offers more stability than the native Hadoop but also easy to use platform monitoring and administration. In this environment, each node in the cluster can be observed instantly via the web-based user interface in the Cloudera Manager. Furthermore, the Cloudera Manager permits administrator to append new cluster or host manually and select custom services to deploy.

Through the Hue, the querying records inside the Hive table becomes more interactive. The Hue not only assists to operate data in the Hadoop but also offers the relevant dynamical search dashboard with Solr. The most important is Hue also supports

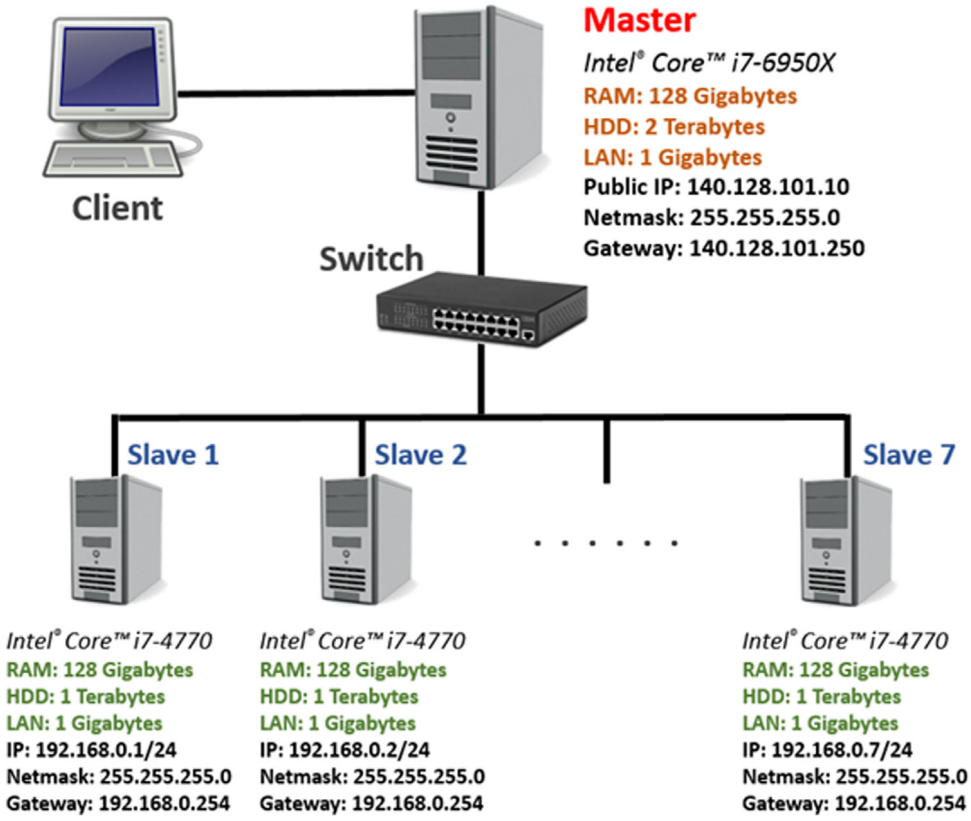


Fig. 6. The detail of computing and storage resources layer.

Fig. 7. WPM-100 wireless multifunction power meter.

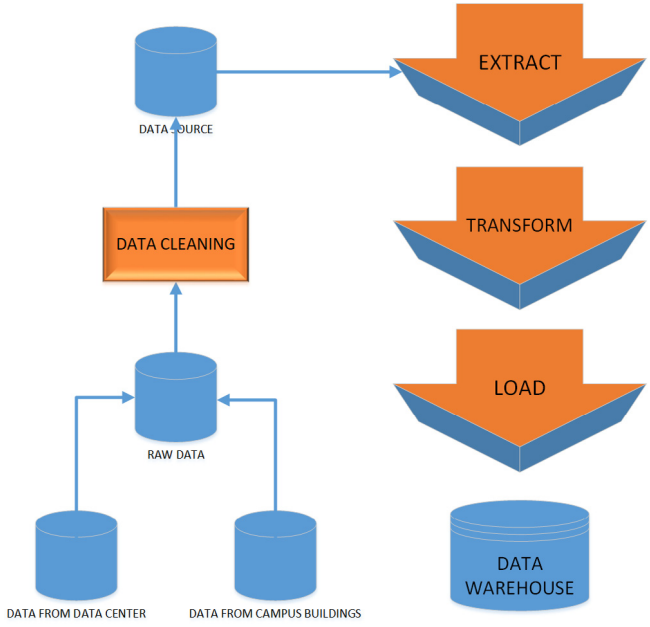


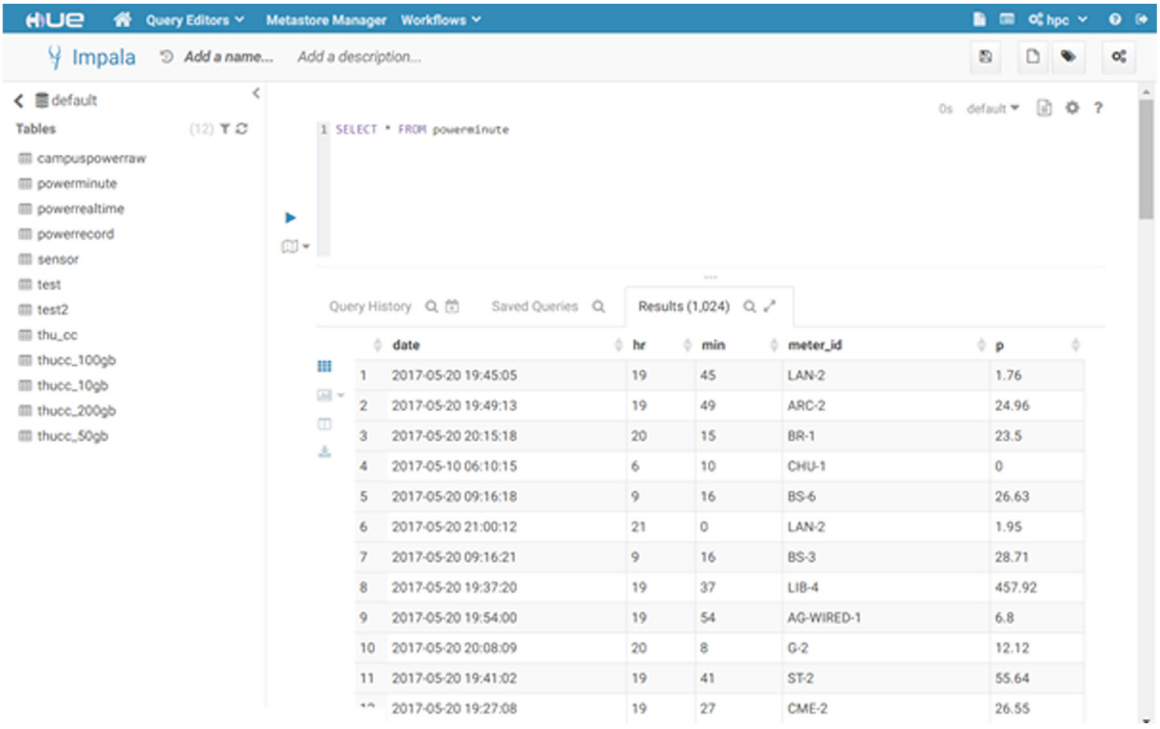
Fig. 8. The scenario of data ETL service.

interactive query of HiveQL and Impala. Fig. 9 presents the Hue web user interface and Fig. 10 shows Hue job browser interface.

4. Evaluation and experimental results

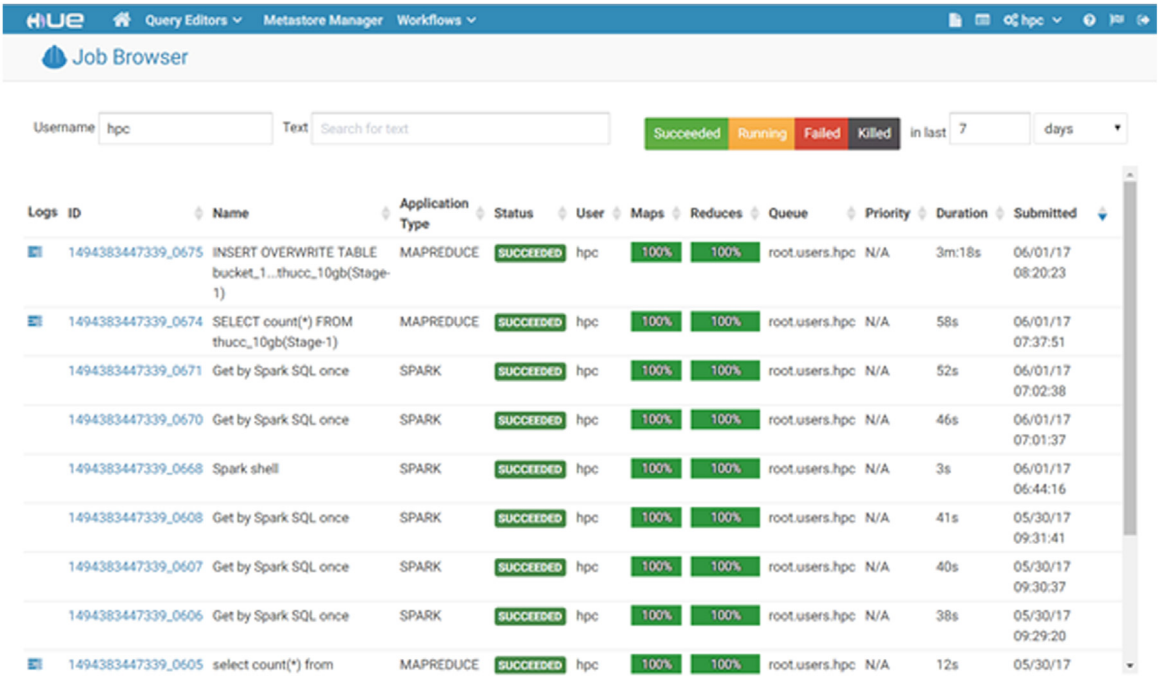
In this section, we conduct the testing and evaluation with the following details:

1. Compare record counting speed of table created in Hive and Impala
2. Compare the searching speed in a single condition query between HiveQL, Spark SQL, and Impala.
3. The Execute time of ETL Application between Hive and Spark
4. The Execute time of ETL Application with different number of processing nodes
5. The Execute time of SQL procedure in different numbers of processing nodes



	date	hr	min	meter_id	p
1	2017-05-20 19:45:05	19	45	LAN-2	1.76
2	2017-05-20 19:49:13	19	49	ARC-2	24.96
3	2017-05-20 20:15:18	20	15	BR-1	23.5
4	2017-05-10 06:10:15	6	10	CHU-1	0
5	2017-05-20 09:16:18	9	16	BS-6	26.63
6	2017-05-20 21:00:12	21	0	LAN-2	1.95
7	2017-05-20 09:16:21	9	16	BS-3	28.71
8	2017-05-20 19:37:20	19	37	LIB-4	457.92
9	2017-05-20 19:54:00	19	54	AG-WIRED-1	6.8
10	2017-05-20 20:08:09	20	8	G-2	12.12
11	2017-05-20 19:41:02	19	41	ST-2	55.64
	2017-05-20 19:27:08	19	27	CME-2	26.55

Fig. 9. Hue web user interface.



Logs	ID	Name	Application Type	Status	User	Maps	Reduces	Queue	Priority	Duration	Submitted
	1494383447339_0675	INSERT OVERWRITE TABLE bucket_1...thucc_10gb(Stage-1)	MAPREDUCE	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	3m:18s	06/01/17 08:20:23
	1494383447339_0674	SELECT count(*) FROM thucc_10gb(Stage-1)	MAPREDUCE	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	58s	06/01/17 07:37:51
	1494383447339_0671	Get by Spark SQL once	SPARK	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	52s	06/01/17 07:02:38
	1494383447339_0670	Get by Spark SQL once	SPARK	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	46s	06/01/17 07:01:37
	1494383447339_0668	Spark shell	SPARK	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	3s	06/01/17 06:44:16
	1494383447339_0608	Get by Spark SQL once	SPARK	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	41s	05/30/17 09:31:41
	1494383447339_0607	Get by Spark SQL once	SPARK	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	40s	05/30/17 09:30:37
	1494383447339_0606	Get by Spark SQL once	SPARK	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	38s	05/30/17 09:29:20
	1494383447339_0605	select count(*) from	MAPREDUCE	SUCCEEDED	hpc	100%	100%	root.users.hpc	N/A	12s	05/30/17

Fig. 10. Hue job browser.

6. Performance compare the Read/Write operation on Hive and Spark
7. Memory utilization compare on 4, 5, 6, 7 nodes between Spark and Impala

The experimental environment and the results of the proposed cloud intelligent campus energy monitoring system are described. After building the proposed system, we have collected about 400 GB of data. In order to make the evaluation and experiments easy to be measured, the data size has been divided into 10 GB, 50 GB, 100 GB, and 200 GB.

4.1. Experimental environment

This section presents our software and hardware specification of our environment. Our system is implemented with 8 physical servers which interconnected by Gigabit Ethernet to build a

Table 1
Software specification.

Software	Version
Cloudera manager	5.10.1
Hadoop	2.6.0-cdh5.10.1
HDFS	2.6.0-cdh5.10.1
Hive	1.1.0-cdh5.10.1
YARN	2.6.0-cdh5.10.1
Spark	2.1.0-cdh5.10.1
Sqoop2	1.99.5-cdh5.10.1
Impala	2.7.0-cdh5.10.1
Hue	3.9.0-cdh5.10.1

Table 2
Hardware specification of experimental environment.

ID	CPU	RAM	HDD	Num of cores
1	i7-4770@3.40 GHz	16 GB DDR3	1 TB	8
2	i7-4770@3.40 GHz	16 GB DDR3	1 TB	8
3	i7-4770@3.40 GHz	16 GB DDR3	1 TB	8
4	i7-4770@3.40 GHz	16 GB DDR3	1 TB	8
5	i7-4770@3.40 GHz	16 GB DDR3	1 TB	8
6	i7-4770@3.40 GHz	16 GB DDR3	1 TB	8
7	i7-4770@3.40 GHz	16 GB DDR3	1 TB	8
8	i7-6950X@3.00 GHz	128 GB DDR3	2 TB	10

computing cluster. Table 1 presents the software specification of our environment. Table 2 shown the hardware specification of our experimental environment.

4.2. Performance evaluation

In order to compare record counting speed of table created, we implemented two frameworks, including Impala and Hive to figure out how performance-intensive a record counting would be at the time when the table is created in Impala and Hive. Fig. 11 describes the result of record counting speed. From this experiment, we found that the Impala has better performance because the Impala can do the file compression in a better way than Hive in case of table creating. A table that is produced in the Hive has spent more time than Impala in any number of records. By comparison, Impala daemon is started at boot time, and each Impala node caches all of its metadata to reuse for future queries against the same table. Thence, Impala is always ready to execute a query. In the term of certification perspective, we inspected with Impala SQL COUNT Function in tables with a different number of records. The scope of verifying data is from 56 to 1120 million.

The response time of the data query is an important performance indicator in the real-time power management system. In order to evaluate the efficiency of searching for records in different conditions. We examine the performance of three modules, a single condition query experiment was being derived from different scales of data. Fig. 12 is the comparison results of HiveQL, Spark SQL and Impala SQL that are searching speed in a single condition query.

We can see in Fig. 12 the result of Impala SQL still has a larger gap compared to HiveQL and Spark SQL. The major reason is the difference of Impala SQL with HiveQL and Spark SQL is that Impala SQL produces small part files than HiveQL and Spark SQL while running a query task in the same cluster. As this reason, the process time of shuffle of Spark is going to spend less than the other modules.

4.3. The execution time of ETL application

In the item of the data ETL processing efficiency, the execution time of data ETL application is an important indicator. The speed

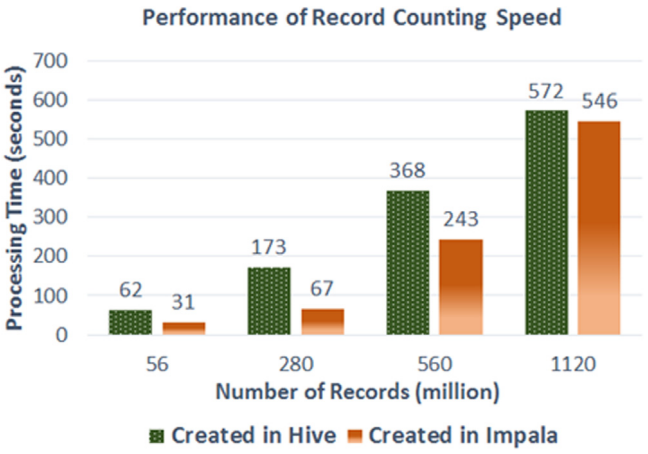


Fig. 11. Comparing record counting speed of table being created in Hive and Impala.

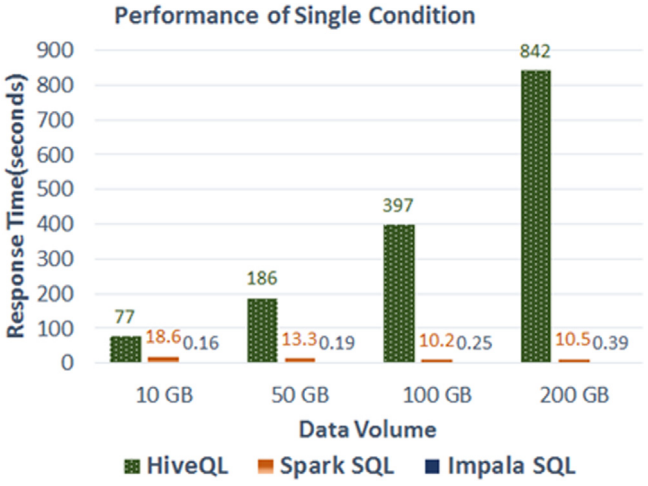


Fig. 12. Searching speed in a single condition query results of HiveQL, Spark SQL, and Impala SQL.

of data ETL will directly influence the speed of data visualization. In Fig. 13, we implemented 2 data ETL applications in Hive and Spark to compare the time of extract data from a data resource, transfer format, and load into a Hive table. As described in Fig. 13, we observe that Spark is rapid more than 16x for ETL application and the latency time while processing data would reduce to less than one minute.

To break out how the number of slave nodes how to influence executing time, we run the same data ETL application in the same server cluster using a different number of slave nodes. In Fig. 14 we can see that when the number of slave nodes increases, the more execution time would be expended. It is because Hadoop is getting data from external data, it will take 3 copies in default before storing them into HDFS. Therefore, Hadoop needs more time to divide the data into more nodes to run a parallel mission. When the more slave nodes in the cluster, the more time latency while data initialization.

4.4. The processing time of the SQL procedure

In general, most of the website would put calculating or statistical procedures on the website server and use several script

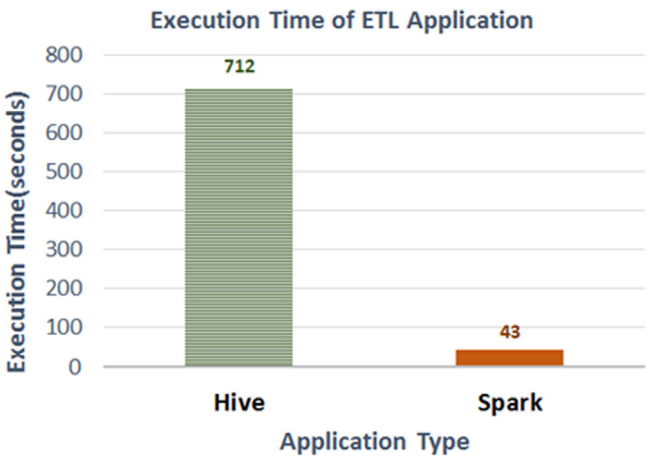


Fig. 13. Execution time of ETL application between Hive and Spark.

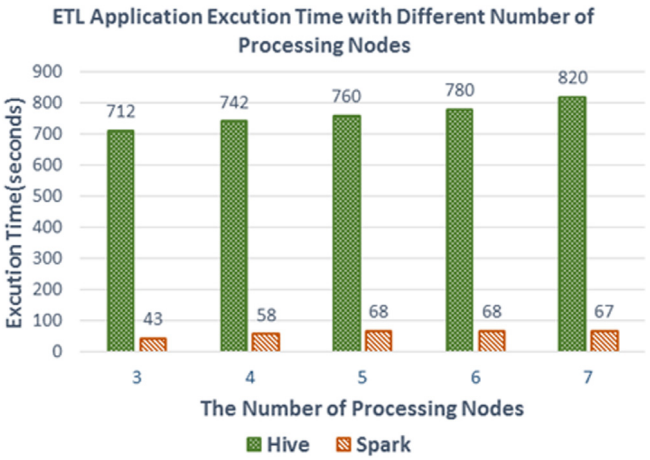


Fig. 14. Execution time of ETL application with different number of processing nodes.

languages to compute the information they need. As the purpose to achieve a rapid presentation on the front-end website, those SQL procedures must be computed quickly in the background cluster. Fig. 15 shows the execution time of the SQL procedure in the computing cluster with different numbers of slave nodes. We observe that the executing time of Spark is by degrees approaching a stable status in three computing resource. The Hive executing time is reduced but still spent far more time than Spark.

Because of the previous experiments, we observed that the number of processing node will directly influence the execute time of data ETL and SQL query operation. Thence, we also compared the execute time trend between read/write operations on Hive and Spark, Fig. 16 shows the result. We can observe, when slave nodes increase, the performance is improved both in Hive and Spark. But Spark still has better performance than Hive in the same conditions.

4.5. Resource utilization

In terms of resource utilization, the processing time of multiple condition query is long enough for presenting the resource utilization of the cluster. Because Spark and Impala are both memory intensive frameworks. Thus, we compared the memory usage of Spark and Impala frameworks, by presenting the experimental utilization results for the same manipulation statement.

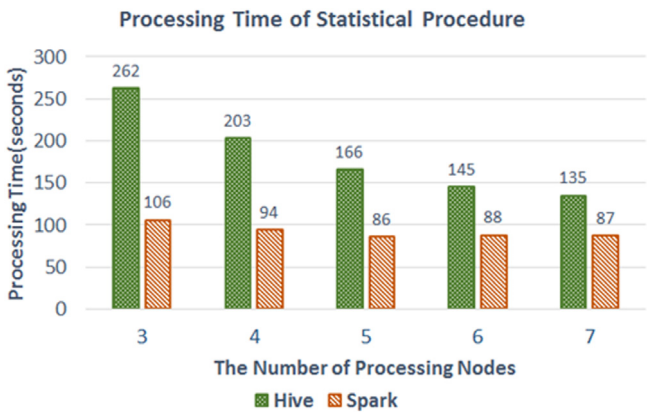


Fig. 15. Processing time of SQL procedure in different numbers of computing nodes.

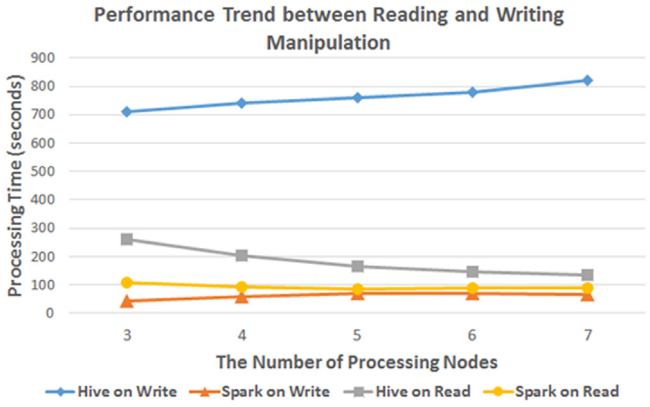


Fig. 16. Performance trend between read/write manipulation on Hive and Spark.

As Fig. 17 shows, we can observe the memory usage for the current workload in relation to the total amount of memory. We can see that Impala does better memory usage than Spark. As we refer before Impala uses more effectively the main memory and realize better resource utilization.

5. Conclusions

In this paper, we deploy a big data warehouse and processing platform of electricity load data and make experiments of data ETL processing for data warehousing with several big data modules. We integrated Hadoop as a storage subsystem, Hive as a data warehouse, Spark as a data ETL tool, and Impala as a big data search engine as a multi-layer architecture. The generic electricity-data source is provided by the smart meters equipment with real-time data. Hadoop subsystem handles data collection and storage, while Spark manages data ingestion to Hive data warehouse to evaluate the query-response performance of our data warehouse and processing platform, we conduct several experiments in the different data quantity.

The major contributions of this paper can be concluded in 2 points: (1) Using open source to construct a multi-layer architecture. The design objective had been addressed by establishing the real-time electricity load-monitoring platform with high efficiency, high feasibility, and low-cost technologies in a cluster environment. We also apply a big data warehouse and ETL process of data cleansing, customization, reformatting, integration into our proposed data warehouse and processing platform.

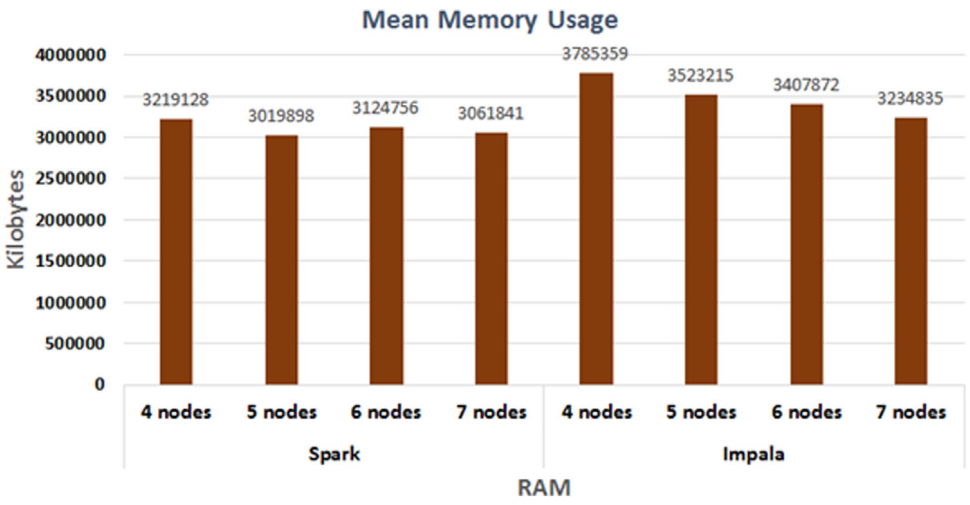


Fig. 17. Memory utilization.

(2) The test and evaluation experiments had been executed to verify the query-response efficiency, and performance evaluations for our proposed data warehouse and processing platform.

In the future works, we are going to adapt the restful API between a front-end website and to complete the data warehouse and processing platform. In addition, we will also improve the system by adopting different types of data, such as semi-structured data (such as system logs or service logs) to obtain more implied phenomenon of power use. Finally, to confirm the system's scalability, we will append more hosts and monitor the condition of each host to enhance the robustness of system.

Acknowledgments

This work is supported in part by the Ministry of Science and Technology, Taiwan ROC, under Grant Numbers MOST 104-2221-E-029-010-MY3, MOST 105-2634-E-029-001, MOST 106-2622-E-029-002-CC3, and MOST 106-2221-E-126-012-MY2.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.jpdc.2019.05.011.

References

[1] Apache Software Foundation: Welcome to Apache™ Hadoop. (2014) http://hadoop.apache.org/index.pdf (accessed on April 30, 2018).
[2] Apache Software Foundation: Mapreduce. (2014) https://hadoop.apache.org/docs/r2.6.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html (accessed on April 30, 2018).
[3] Apache Software Foundation: The Hadoop Distributed File System: Architecture and Design. (2014) http://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html (accessed on April 30, 2018).
[4] Apache Software Foundation: Apache Hive. (2014) https://hive.apache.org (accessed on April 30, 2018).
[5] Apache Software Foundation: Apache Sqoop. (2018) http://sqoop.apache.org/ (accessed on April 30, 2018).
[6] Apache Software Foundation: Spark SQL, Dataframes and Datasets Guide. (2016) http://spark.apache.org/docs/1.6.0/sql-programming-guide.html (accessed on April 30, 2018).
[7] Peter Mell and Timothy Grance : The NIST Definition of Cloud Computing, NIST Special Publication, 2011, 800-145.
[8] Tom White: Hadoop: The Definitive Guide, O'Reilly Media, Inc., 2012.
[9] Farag Azzedin, Towards a scalable HDFS architecture, in: Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013, 2013, pp. 155–161.
[10] Dan Chen, Yangyang Hu, Lizhe Wang, Albert Y. Zomaya, Xiaoli Li, H-PARAFAC: Hierarchical parallel factor analysis of multidimensional big data, IEEE Trans. Parallel Distrib. Syst. 28 (4) (2017) 1091–1104.
[11] Dan Chen, Xiaoli Li, Lizhe Wang, Samee U. Khan, Juan Wang, Ke Zeng, Chang Cai, Fast and scalable multi-way analysis of massive neural data, IEEE Trans. Comput. 64 (3) (2015) 707–719.
[12] Min Chen, Shiwen Mao, Yunhao Liu, Big data: A survey, Mob. Netw. Appl. 19 (2) (2014) 171–209.
[13] Wang Dewen, Zhou Qing, A method of distributed on-line analytical processing of status monitoring big data of electric power equipment, Zhongguo Dianji Gongcheng Xuebao/Proc. Chin. Soc. Electr. Eng. 36 (19) (2016) 5111–5121.
[14] Jens Dittrich, Jorge-arnulfo Quian, Efficient big data processing in Hadoop MapReduce, Proc. VLDB Endow. 5 (12) (2012) 2014–2015.
[15] Junqing Fan, Jining Yan, Yan Ma, Lizhe Wang, Big data integration in remote sensing across a distributed metadata-based spatial infrastructure, Remote Sens. 10 (1) (2018) 7.
[16] Ke Li, Fei Su, Xinzhou Cheng, Weiwei Chen, Kejing Meng, The Research of Performance Optimization Methods Based on Impala Cluster. 2016, pp. 336–341.
[17] Xiufeng Liu, Per Sieverts Nielsen, A hybrid ICT-solution for smart meter data analytics, Energy 115 (2016) 1710–1722.
[18] Ilias Mavridis, Helen Karatza, Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark, J. Syst. Softw. 125 (2016) 133–151.
[19] Diane J. Skiba, The Internet of Things (IoT), Nurs. Educ. Perspect. 34 (5) (2015) 63–64.
[20] Weijing Song, Lizhe Wang, Yang Xiang, Albert Y. Zomaya, Geographic spatiotemporal big data correlation analysis via the Hilbert-Huang transformation, J. Comput. System Sci. 89 (2017) 130–141.
[21] Yunbo Tang, Dan Chen, Lizhe Wang, Albert Y. Zomaya, Jingying Chen, Honghai Liu, Bayesian tensor factorization for multi-way analysis of multi-dimensional EEG, Neurocomputing 318 (2018) 162–174.
[22] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, Raghotham Murthy, Hive - A petabyte scale data warehouse using Hadoop, in: Proceedings - International Conference on Data Engineering, 2010, pp. 996–1005.
[23] Sahithi Tummalapalli, Venkata rao Machavarapu, Managing MySQL cluster data using cloudera impala, Procedia Comput. Sci. 85 (2016) 463–474.
[24] Lizhe Wang, Ke Lu, Peng Liu, Rajiv Ranjan, Lajiao Chen, IK-SVD: Dictionary learning for spatial big data via incremental atom update, Comput. Sci. Eng. 16 (4) (2014) 41–52.
[25] Chao-Tung Yang, Shuo-Tsung Chen, Yin-Zhen Yan, The implementation of a cloud city traffic state assessment system using a novel big data architecture, Cluster Computing 20 (2) (2017) 1101–1121.
[26] Chao-Tung Yang, Wen-Chung Shih, Guan-Han Chen, Shih-Chi Yu, Implementation of a cloud computing environment for hiding huge amounts of data, in: Proceedings of the 2010 International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2010, pp. 1–7.
[27] Chao-Tung Yang, Wen-Chung Shih, Chih-Lin Huang, Implementation of a distributed data storage system with resource monitoring on cloud computing, in: International Conference on Grid and Pervasive Computing, 2012, pp. 64–73.

[28] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica, Spark: Cluster computing with working sets, HotCloud (2010) 10.10–10:95.

Chih-Hung Chang, Ph.D., received his Ph.D. degrees in Computer Science from Feng Chia University in 2004. Currently, he is an associate professor at Providence University. He is member of the IEEE Computer Society. His research interests include software engineering, cloud service, and big data . He has published more than 140 papers in journals, book chapters and conference proceedings. He served as local arrangement chair for COMPSAC 2015, and program chair for TANET 2017. His email address is ch.chang@gm.pu.edu.tw.

Fuu-Cheng Jiang, worked as a design engineer with the Aeronautical Research Lab., Chung Shan Institute of Science and Technology (CSIST) when he was assigned to a partnership project at General Dynamic, Fort Worth, Texas. Currently, he is a member of faculty in the department of computer science at Tunghai University in Taiwan. Professor Jiang was the recipient of the Best Paper Award at the 5th International Conference on Future Information Technology 2010 (FutureTech 2010), which ranked his paper first among the 201 submittals. He has served dozens of the TPC for worldwide international conferences like BWCCA 2010, ICCCT 2011-2012, IEEE CloudCom 2012, IEEE BIOCAS-2013, NPC 2014, IEEE SPICES 2015, IoT 2015, CCBD 2015, CUTE 2016, IC4S 2016-2017, IoT4TD 2017, ICSICCS 2018, IEEE SC2 2014-2019 and also the Session Chair of CSE2011 and IEEE ICCE-Taiwan 2014, Publication Chair of NPC 2014, the Demo/Poster Chair of IOV 2016 and Publication Chair of TANET 2017. Moreover, he served as journal reviewer of the Computer Journal, Ad Hoc Networks, Journal of Network and Computer Applications (JNCA), Journal of Supercomputing (JOS), Journal of Internet Technology (JIT), Wireless Networks, International Journal of Communication Systems (IJCS), Wireless Networks, Future Generation Computer Systems (FGCS) and IEEE Transactions on Cloud Computing. Professor Jiang has served several journals as an associate editor like American Journal of Artificial Intelligence (AJAI, ISSN 26399717), CIP-JWCMCN Journal and WCSN (ISSN 25207725), and an assistant editor on the Cloud-Link Editorial Team of IEEE society. His research interests include network modeling, services on cloud computing, wireless sensor networks, IoT, artificial intelligence and simulation.

Chao-Tung Yang received a B.Sc. degree in Computer Science from Tunghai University, Taichung, Taiwan, in 1990, and the M.Sc. degree in Computer Science from National Chiao Tung University, Hsinchu, Taiwan, in 1992. He received the Ph.D. degree in Computer Science from National Chiao Tung University in July 1996. In August 2001, he joined the Faculty of the Department of Computer Science at Tunghai University as an Associate Professor. He is a full Professor started in August 2007 and as a Distinguished Professor in August 2015. He is serving in a number of journal editorial boards, including Future Generation Computer Systems, International Journal of Communication Systems, KSII Transactions on Internet and Information Systems, Journal of Cloud Computing, IJ- CLOSER, International Journal of Next-Generation Computing (IJNGC), Dr. Yang has published more than 300 papers in journals, book chapters and conference proceedings. His present research interests are in Cloud computing and Big data, Parallel and multicore computing, and Web-based applications. He is both a member of the IEEE Computer Society and ACM. He is also both a member of IICM and TACC in Taiwan. His email address is ctyang@thu.edu.tw.

Sheng-Cang Chou, MS, received his M.Sc degrees in Computer Science at Tunghai University in 2018. His present research interests are in cloud and grid computing, His email address is minicp9523@gmail.com.