A Big Data Management System for Energy Consumption Prediction Models

Wonjin Lee*
Dept of Computer Science
Kyonggi University

Byung-Won On†
Dept of Statistics and Computer Science
Kunsan National University

Ingyu Lee‡
Smart Grid Research Center
Advanced Institutes of Convergence Technology

Jungin Choi§
Smart Grid Research Center
Advanced Institutes of Convergence Technology

ABSTRACT

In this work, we develop a prototype about a big data management system for storing, indexing, and searching for *huge-scale energy usage data*. Rather than existing, commercial relational databases such as Oracle and IBM-DB2, this system is able to provide us with high availability and performance at low cost. It is also able to manage unstructured data and store big data in distributed environment. In addition, using data access APIs, target data is quickly retrieved from our proposed system. To utilize our prototype system, we also propose an energy consumption prediction model based on penalized linear regression-based map/reduce algorithms. Then, we exploit discriminate features with respect to time stamp. Finally, given a time stamp (e.g., 2014-01-05 12:01:08), our proposed learning model will give us a predicted value about the energy usage (e.g., 90 watt) at that time. According to our experimental results obtained from about 7.5 million records, each of which consists of an energy usage and time stamp during three months in 2014, it turns out that our prediction model can predict real values that are very close to actual energy usage at that time, and is about 1.72 times faster than in a single machine.

1 INTRODUCTION

Big data is too huge and complex to process using current database management system tools. With the advent of new technologies such as smart devices, social networking service, and cloud computing, data, regardless of transaction data or unstructured data (e.g., location information, short messages, text, friendship links, etc.), have been created and stored anytime and anywhere. According to the recent report [10], the total size of data in the world is 1.8×10^{21} Bytes in 2011. The amount of data has been rapidly increased in two times per year, and further we expect that it will be about 44 times in 2020 than now. In May 2011, Mckinsey & Company pointed out that big data will become the next frontier agenda for innovation, competition, and productivity. If big data technology is applied to industries such as automobiles and shopping malls, it can help both profit-making and job creation. As part of this effort, the recent study is to apply big data technology (e.g., big data collection, integration and cleaning, databases for managing big data, and data mining and visualization) to the field of energy usage management and energy efficiency. In general, it is reported that big data tend to have four properties: volume, velocity, variety, and complexity. The volume stands for bulky data – e.g., larger than a few hundreds Giga Bytes. The others mean real-time data creation, unstructured data, linked databases, respectively. If a given data has at least two properties, we can view such a data as *Big Data*. In case of energy usage data (power data in short), we can say that it is one of big data because it has two properties – volume and velocity. A large number of sensors are located in houses, buildings, factories, and the other places. Sensors collect energy usage data in near real-time. Currently, to manage this energy data, commercial relational database management systems such as Oracle and IBM-DB2 have been widely employed in industrial complex. However, the data coverage of such database tools is a few hundreds Giga Bytes at most. In addition, it is so hard to support high availability with a low budget. It should also provide us with high performance. Nowadays since the amount of data is rapidly increased and stored to cloud servers, available disk space should be easily added to the servers at low cost. It is also not effective to use the existing relational databases to manage *unstructured* data [1].

In this paper, we first propose a big data management system for managing large-scale power data. The size of data that we considered is about 4 Tera Bytes. The aim of our research is to present how to build such a big data management system. Our proposed system consists of a cluster infrastructure, automatic data collection, data loading to a distributed file system, a distributed database for managing large-scale power data, and data access APIs. We designed and implemented our prototype system to validate our approach in which big data technology can be effectively applied to the energy efficiency domain. In our system, large-scale power data are well indexed and stored in a distributed database. Further, using data access APIs, end-users can quickly search for target data from our proposed system. This system also provides us with high availability as well as high performance with a low budget. And it will be able to manage unstructured data well unlike the existing commercial database management systems such as Oracle and IBM-DB2.

In addition, to utilize our prototype system, we address an energy consumption prediction problem. We first develop a prediction model based on a least square linear regression-based map/reduce algorithms. Then, after exploiting 32 discriminate features with respect to time stamp, we make machine learning with the features using training sets. Finally, we show the effectiveness of our prediction model using test sets. In our work, given a time stamp – e.g., 2014-01-05 12:01:08, our proposed model will give us a predicted value about the electricity usage (e.g., 90 watt) at that time. In our experiments, we used about 7.5 million records consisting of pairs of (electricity usage, time stamp) during three months – from January until March 2014. Our experimental results validate that our prediction model works effectively but yet efficiently. For the

*e-mail: ckask12@kyonggi.ac.kr
†e-mail: bwon@kunsan.ac.kr, Corresponding Author
‡e-mail: inlee@gmail.com
§e-mail: drjichoi@gamil.com

[1] It is known that about 90% of data created in recent time is unstructured data.

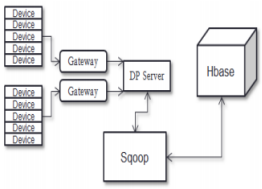


Figure 1: The flow chart of data collection in our system.

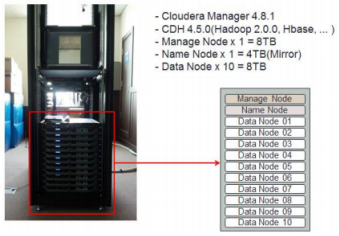


Figure 2: A screen-shot image about our cluster system.

detail, we will discuss the experimental results in Section 3.

This paper consists of the followings. Section 2 describes details of the proposed prototype system for managing big data about energy usage domain. We also describe empirical results on our energy consumption prediction model in Section 3. Subsequently, Section 4 describes existing methods related to our proposal. Concluding remarks and future plans are followed in Section 5.

2 Main Proposal

2.1 Big Data Management System for Power Data

2.1.1 Collection of Power Data

In our context, power data means the *electricity usage per 15 minutes* from four industrial complex in Korea. Those are Sihwa machinery-industrial complex, Gwangju circulation market, Gwangju tech valley, and Incheon circulation market. In Sihwa machinery-industrial complex, there are 392 sensor devices from which power data have been created in real time. Similarly, there are 216, 440, and 824 sensors in the other complex, respectively. A gateway collects and aggregates these power data from 50 sensors in each industrial complex. Then, it sends the aggregated power data to a data portal server. Since each complex has one data portal server, there are four data portal servers in our system. Finally, the aggregated power data are stored to PostgreSQL database in each data portal server. Since PostgreSQL is a relational database management system, we are unable to store the power data in Hadoop Distributed File System (HDFS) in person. To address this problem, we use Sqoop open source tool [1]. Using Sqoop, all of the power data in four data portal servers are automatically stored in a distributed database management system such as HBase in our big data management system. We will also describe the detail of HBase in Section 2.1.4. Until now, the volume of data in our big data management system is about 4 Tera Bytes (TB) and these power data have been appended to our system. Figure 1 shows the flow chart of collecting energy usage data in our system.

2.1.2 Cluster Architecture

Since power data such as electricity usage per 15 minutes in a good many buildings and factories are huge, such data are unlikely to be stored in a single machine. A cluster system, as a standard architecture for such problem, has been widely used in numerous institutes and companies these days. In general, cluster systems are defined as a cluster of general-purpose computers in which they are connected each other via commodity network such as Ethernet and each machine is based on Linux operating systems. Recent cluster systems have about $16 \sim 64$ nodes per rack and any pair of nodes in a rack is supported by 1 Gbps physical layer. In addition, $2 \sim 10$ Gbps backbone is set up between racks. In 2011, it was known that Google had 1 million machines [10]. Given a big data that is unable to be stored in a single machine, it should be first divided into pieces of data. We call each piece of data either "chunk" or "block." The size of each block is about 64M bytes or so. Each block should be stored in a particular node within a cluster system. Compared to processing data in a single machine, this procedure is fairly challenging because of the following questions:

1. How to distribute computation within the cluster system?
2. How to make it easy to write distributed programs?
3. How to handle if some machines fail?

For instance, it is known that a computer may be working normally during three years. If our clustering system consists of 1,000 nodes, it is expected that one node fails every day. In the end, we are no longer able to use pieces of data that have stored in the failed node. To cope with this issue, a block is stored multiple times to different nodes within the cluster system for reliability. In addition, it will take time to copy data over a network. Therefore, if blocks are stored in a node, all of the blocks are *locally* processed in the node. We call it a "map" task. For example, let us suppose that a file is divided into ten blocks, and then each block is stored in a node. Thus each node has a block in it. Next, a map task is executed in each node. Then, the results of ten map tasks are aggregated to only a few nodes within the cluster system. For instance, the ten results are sent to one node in which a "reduce" task carried out, performing *global process* in order to provide us with a final result. This map/reduce model will be able to reduce network traffic within the cluster system. Furthermore, the communication interface between map and reduce tasks is a pair of key and value. This approach will make it easy to write programs in parallel. Finally, the cluster system provides us with global name space so that end-users properly appreciate that the cluster system acts as a single machine. They even do not know that a big file is divided to multiple chunks of entire data and to see which machine includes particular pieces of data.

In our big data management system, to support this distributed environment, we first constructed a cluster system as shown in Figure 2. By and large, our cluster system consists of a *manager node*, a *master node*, and ten *data nodes*. Since this cluster system has one IP address which corresponds to the MAC address of the manager node, clients outside the cluster system can only communicate with the manager node. The master node controls the cluster system, whereas data nodes mainly store blocks in them. Each node is a workstation server having Intel Xeon 2.20GHz CPU and six cores per CPU. We can say that our system will eventually have 66 nodes if we view one core as a data node. Each node also has 8TB Hard Disk Drive (HDD) so the total size of storage space in the system is 88 Tera Bytes (TB). The master node has 32GB main memory, while the size of main memory is 24GB main memory in the data nodes.

2.1.3 Distributed File System

We used Linux operating system (e.g., CentOS release 6.4) on each node in the cluster system. In addition, to support the distributed environment, we installed Hadoop Distributed File Systems (HDFS) and MapReduce Framework on each machine in the cluster system [1]. In particular, we installed and configured Hadoop name node daemon on the master server and Hadoop data node daemon on each data node. Hadoop is a open source project regarding

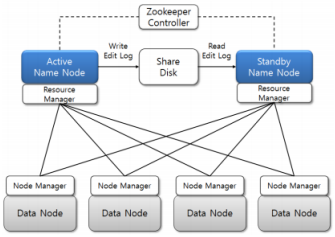


Figure 3: Hadoop distributed file system.

High Availability Distributed Object-Oriented Platform. Overall, Hadoop ecosystem is composed of data storage, data processing, data access, management, and applications. In the data storage level, there are HDFS and HBase. HDFS is a distributed file system, while HBase is a NoSQL (Not Only SQL) database based on HDFS. HDFS plays an important role in posing data nodes in the cluster system as a single storage device. End-users do not properly appreciate that a big file is automatically divided into blocks and then the blocks are stored in proper nodes, considering network traffic and load balancing of data nodes in the cluster system. The MapReduce framework is in the data processing layer. Through this framework, programmers can write distributed programs using Java which are executed in parallel. However, Hadoop also supports the very users that are not familiar with Java, to access big data. Hive, Pig, and Avro are the data access modules. For example, a user can write a simple program using query language like query clauses in MySQL (e.g., SELECT-FROM-WHERE clauses). Then, Hive automatically translates the query clause to a sequence of map and reduce tasks so as to get or put data to HDFS. The automatically-generated map and reduce tasks are executed in parallel on multiple data nodes within the cluster system. In the management level, there are Zookeeper and Chukwa. Zookeeper is required for distributed configuration service, synchronization service, and naming registry for large distributed systems. In the meanwhile, Chukwa gives us a service to quickly search for target data. Finally, as applications, there are machine learning tools like Mahout, RHadoop for data mining and visualization, data warehousing tools, and so on.

In the following section, we will discuss the detail of HDFS which consists of a name node and data nodes in the cluster system.

Name Nodes: Hadoop name node daemon is installed on the master node in the cluster system. The name node divides a file to blocks and then send blocks to particular data nodes. It always keeps an eye on the number and size of blocks stored in each data node, and then determines proper data nodes. Next, the name node usually stores meta data about where blocks are stored. In addition, for reliability, a block is replicated three times (by default) and stored in different data nodes within the cluster system. In our system, since we used Hadoop 2.2, we can avoid the failure of the entire system. In Hadoop 1.2, there is only one name node physically. If the name node fails, we can no longer use the system. On the other hand, in our big data management system, there are two name nodes. One is active and the other is stand-by. In our system, the tenth data node is also set up to be the stand-by name node.

As shown in Figure 3, the active name node regularly writes edit log to the shared disk called the journal node. Then, the stand-by name node gets them from the journal log. When the active name node fails for any reason, the stand-by name node detects the failure of the current, active name node through Zookeeper, and then the stand-by name node is switched to the active name node instead of current name node. This failover process keeps the name node going in a seamless way.

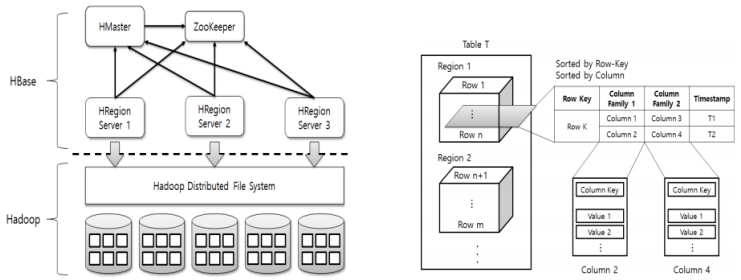
Data Nodes: Hadoop data node daemon is located on each data node in the cluster system. In general, each data node has a few thousands blocks, each of which contains about 64 Mega Bytes (MB) on average, and has a few hundreds Giga Bytes (GB) in it. The data nodes usually execute map and reduce tasks. A map task is executed with blocks stored in the node. Then, the results of map tasks are combined to only a few data nodes. Then, reduce tasks perform global processes to provide us with a final result. Often, data nodes merge multiple blocks when the number of blocks is increased to improve an efficient search service. In addition, to improve performance, blocks are often split when the size of the block is large.

In particular, there are two main daemons on each data node: node managers and application masters. These daemons are usually communicating with the resource manager running on the active name node. The resource manager is responsible for scheduling tasks as well as allocating resources like tasks. It regularly communicates with the node manager using heart bits to check if the node manager is still alive or not. The node manager estimates the size of context, meaning each data node's memory buffer including a sequence of tasks. If the node manager determines that a data node has enough context to execute tasks allowed by the resource manager, it asks a corresponding application master to perform the task. Application masters manage tasks inside the context of a data node, and they also ask the resource manager to obtain tasks to be executed.

2.1.4 Distributed Database

We installed Hadoop Database (HBase) in [1]. HDFS is enough if data is rarely updated in place and read/append operations are mainly used in an application. However, HBase is needed when random updates are common in another application. Unlike the existing relational database management systems such as Oracle, IBM-DB2, MySQL, etc., HBase is one of non-relational database management systems. We call it NoSQL database. This means that HBase does not support join operations. Instead, the aim of HBase is to sort large-scale data by time stamp and to quickly search for particular data. In addition, since NoSQL databases do not have any relational schema, it is easy to manage unstructured data such as text data. In addition, they have been designed to satisfy two properties: partition tolerance and either consistency or availability. Therefore, NoSQL databases may not perfectly guarantee the integrity of data. However, without the cease of service, data storage space can be added to NoSQL databases so big data is easily stored in the database without paying an enormous sum of money. Unlike popular NoSQL databases such as Big Table, Cassandra, MongoDB, REDIS, etc., HBase is especially based on Hadoop Distributed File Systems. In other words, data is actually stored in HDFS. On the other hand, HBase has occupied a buffer space in memory. The buffer contains popular blocks that are frequently used in some application. If the application continues to initiate update operations, HBase first goes to the buffer, finding the corresponding blocks and then updating the block. If the buffer is full, proper blocks are stored to HDFS by a certain buffer scheduling policy (e.g., LRU).

In practice, HBase is composed of HMaster and HRegion server daemons. HMaster is running on the active name node. The HMaster usually stores meta data about where blocks are stored. On the other hand, the HRegion servers are located in data nodes. Each HRegion server has a buffer, called MemStore, that includes a couple of HRegions. A HRegion is corresponding to a relation table in RDBMS. As shown in Figure 4, each HRegion consists of row-key, column-family, and time-stamp. A row-key has a couple of column-family that has a set of columns. HBase is also based on Zookeeper that controls HRegion servers. If certain HRegion servers fail in HBase, Zookeeper starts to recover the HRegion servers which are



(a) HBase system architecture (b) Region table type

Figure 4: HBase: Distributed database in Hadoop.

out of order.

2.1.5 Data Access APIs

In the final step, to search for particular data and then to get them, we implemented a web page in which users create queries (e.g., create or delete tables, insert records, etc.) and get records, which they want, stored in our big data management system. We used JDBC-like libraries (as .jar files) with which HBase provides us.

2.2 Energy Consumption Prediction Model

In this section, the goal of our research is to propose an effective energy consumption prediction model. The prediction model is based on linear regression model.

2.2.1 Linear Regression Model

Given a matrix $X = (X_1, ..., X_p) = (X_{ij})_{n \times p} \in R^{n \times p}$ and response Y, we build a least square linear model by minimizing the Residual Sum of Squares

$$RSS(\alpha, \beta) = (Y - \alpha 1 - X\beta)^T (Y - \alpha 1 - X\beta) \quad (1)$$

In Eq. 1, α and β stand for intercept and coefficients, and X and Y indicate an independent variable and a dependent one. According to the recent study in [13], $RSS(\alpha, \beta)$ can be summarized as follows.

$$RSS(\alpha, \beta) = T_1 + T_2 \quad (2)$$

$$T_1 = Y^T Y - n\overline{Y}^2 - 2(Y^T X - n\overline{Y}(\overline{X_1}, ..., \overline{X_p}))D^{-1}\beta \quad (3)$$

$$T_2 = \beta^T D^{-1}(X^T X - n(\overline{X_1}, ..., \overline{X_p})^T (\overline{X_1}, ..., \overline{X_p}))D^{-1}\beta \quad (4)$$

For $RSS(\alpha, \beta)$ in Eq. 2, the values of n, $Y^T Y$, $X^T Y$, $\overline{Y}$, $\overline{X_i}$, and $X^T X$ should be computed in parallel using MapReduce framework in our big data management system. Table 1 shows our MapReduce algorithm based on penalized linear regression.

Map Task	for each sample (x, y) key = random(0,1,...,k-1) calculate value = $\{n, Y^T Y, X^T Y, \overline{Y}, \overline{X_i}, X^T X\}$ emit (key, value)
Reduce Task	for each (key, value list) chunk = aggregate whole value list emit(key, chunk)

Table 1: The MapReduce algorithm for penalized linear regression

2.2.2 Feature Selection

For our experiments, we collected power data between January 2014 and March 2014. The power data is a set of records, each of which is a pair of electricity usage per 15 minutes and time stamp. An example of records is (electricity usage = 90 watt, 2014-01-05 12:01:08). The number of records is about 7,500,000 during the three months. From all of the records, we exploited a total of 32 features, each of which is day of the week, national holiday, or 24 hours. For instance, let us suppose that a given time stamp is 2014-01-05 12:01:08. Since 2014-01-05 was Sunday, the 'Sunday' feature is 1, while the 'Monday' feature is 0. Similarly, the other day of week features are 0. 2014-01-05 was not a national holiday so the 'national holiday' feature is 0, and the feature between 12:00 and 13:00 is 1, and 0 otherwise. In this way, a feature vector with respect to 2014-01-05 12:01:08 is {1,0,0,0,0,0,0,0, ..., 1, ...}, where the first and 20-th features are 1, meaning Sunday and the time interval between 12:00 and 13:00.

Finally, we grouped all of the records into six groups. One is a group of records related to Sunday and national holiday in January 2014. We denote this dataset by 2014-01-Holiday. Another is a group of records including weekdays in January 2014. We also label this data set as 2014-01-Weekdays. In this way, we generated the other four data sets – e.g., 2014-02-Holiday, 2014-02-Weekdays, 2014-02-Holiday, and 2014-02-Weekdays.

2.2.3 Prediction

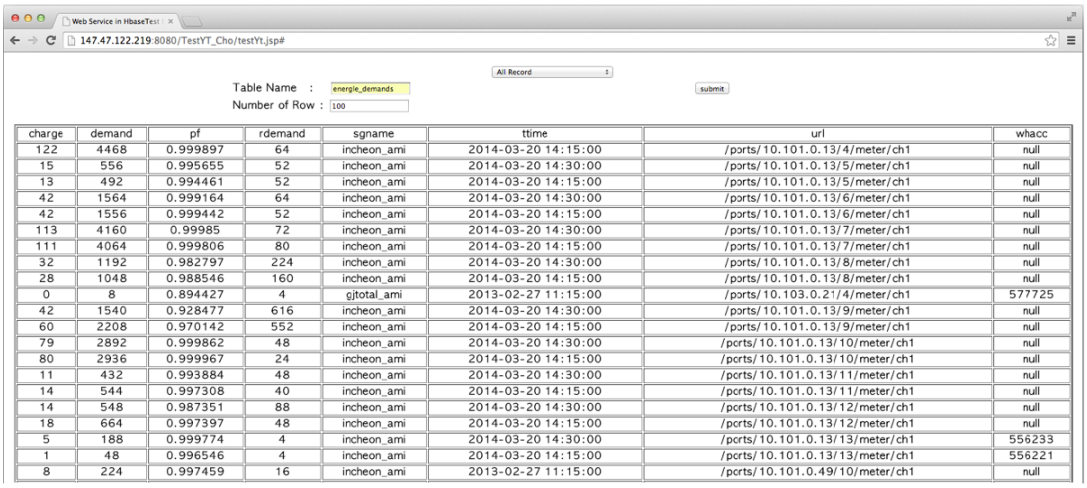
For each data set, we constructed the linear regression model as discussed in Section 2.2.1. Then, we estimated intercept and coefficient values α and β using the linear regression model. Next, given a feature vector in each test set, compute Y indicating the predicted value of the electricity usage.

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_{32} X_{32} \quad (5)$$

3 Experimental Results

3.1 Demonstration of our Big Data Management System

Figure 5 shows a web browser through which a user can take a look at some records in our big data management system including about 4TB energy usage data which have been collected in four industrial complex in Korea since 2013. To search for some energy usage data, users can visit to the web site in http://147.47.122.219:8080/TestYT_Cho/index.html. In the main web page, a user can have a look at the list of tables (HRegions) which are stored in our big data management system. First of all, select one of tables. Next, enter a table name for which he or she wants to search, and then put the number of records to be displayed in the web page. In Figure 5, the table name is "energle_demand" and the number of rows in the table is 100. Then, press the submit button. After that, the user can see top-100 records in the energle_demand table. Each record consists of eight fields – charge, demand, pf, rdemand, sgname, ttime, url, and whacc. Charge means



charge	demand	pf	rdemand	sgname	ttime	url	whacc
122	4468	0.999897	64	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/4/meter/ch1	null
15	556	0.995655	52	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/5/meter/ch1	null
13	492	0.994461	52	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/5/meter/ch1	null
42	1564	0.999164	64	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/6/meter/ch1	null
42	1556	0.999442	52	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/6/meter/ch1	null
113	4160	0.99985	72	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/7/meter/ch1	null
111	4064	0.999806	80	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/7/meter/ch1	null
32	1192	0.982797	224	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/8/meter/ch1	null
28	1048	0.988546	160	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/8/meter/ch1	null
0	8	0.894427	4	gjtotal_ami	2013-02-27 11:15:00	/ports/10.103.0.21/4/meter/ch1	577725
42	1540	0.928477	616	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/9/meter/ch1	null
60	2208	0.970142	552	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/9/meter/ch1	null
79	2892	0.999862	48	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/10/meter/ch1	null
80	2936	0.999967	24	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/10/meter/ch1	null
11	432	0.993884	48	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/11/meter/ch1	null
14	544	0.997308	40	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/11/meter/ch1	null
14	548	0.987351	88	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/12/meter/ch1	null
18	664	0.997397	48	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/12/meter/ch1	null
5	188	0.999774	4	incheon_ami	2014-03-20 14:30:00	/ports/10.101.0.13/13/meter/ch1	556233
1	48	0.996546	4	incheon_ami	2014-03-20 14:15:00	/ports/10.101.0.13/13/meter/ch1	556221
8	224	0.997459	16	incheon_ami	2013-02-27 11:15:00	/ports/10.101.0.49/10/meter/ch1	null

Figure 5: A screen-shot image of query results using Google Chrome web browser.

electric bill. The unit of the field is cent. Demand and rdemand indicate active power and wattless power, respectively. In the third field in the table, pf is power factor, defined as the ratio of the real power flowing to the load, to the apparent power in the circuit, and is a dimensionless number between -1 and 1. The sgname field contains gateway names. Ttime is time stamp in the format of YYYY-MM-DD Hours:Minutes:Seconds. Each url stands for each sensor's IP address and port number.

3.2 Results of our Energy Consumption Prediction Model

Figure 6 shows the results of six data sets – (1) 2014-01-Weekday dataset, (2) 2014-02-Weekday dataset, (3) 2014-03-Weekday dataset, (4) 2014-01-Holiday dataset, (5) 2014-02-Holiday dataset, and (6) 2014-03-Holiday dataset. The 2014-01-Weekday dataset means pairs of records, each of which consists of active power per 15 minutes and time stamp (YYYY-MM-DD Hours:Minutes:Seconds) which were created in weekdays around January 2014. On the other hand, the 2014-01-Holiday dataset includes all records that were created in only holidays during January 2014. In each dataset, we can see three lines in color. The orange line stands for *actual*, active power. The blue line means the active power value *predicted* by our prediction model. The green line indicates the difference value between the actual, active power and the predicted one. If the green line lies close to zero, it implies that predicted values are closely similar to corresponding actual, active power values. Thus, we can validate the effectiveness of our prediction model based on linear regression-based map/reduce algorithms. On the whole, for all of the six datasets, it is consistently observed that our predicted values are very similar to actual, active power values. Note that the difference is insignificant. We also measured both mean difference and standard deviation values about the six datasets, and hence reported in Table 2. In all of the datasets that we used (i.e., about 7,500,000 records between January 2014 and March 2014), the maximum active power value is 6,000,000, while the minimum value is 1,000,000. The difference between max and min values is 5,000,000. According to Table 2, the average difference between actual and predicted values is less than 500,000 that is much less than that between max and min values in actual power data. However, unexpectedly, we can also observe that the difference between actual and predicted values is higher in all holiday datasets, compared to that of all weekday datasets.

Finally, we measured the execution time of our prediction model to process about 7,500,000 records on a single machine and our cluster system. The running time is 5 minutes 10 seconds on the single machine, whereas that is 3 minutes on the cluster system. If we execute our model on the cluster system, the running time is about 1.72 times faster than that on the single machine.

Dataset	Standard deviation	Mean difference
2014-01-Weekdays	373,149	124,774
2014-02-Weekdays	507,200	134,495
2014-03-Weekdays	320,022	100,843
2014-01-Holidays	3,416,968	527,897
2014-02-Holidays	3,479,047	471,278
2014-03-Holidays	3,035,376	459,588

Table 2: The mean difference and standard deviation values between actual and predicted values in the six datasets.

4 Related Work

Lai et. al. [9] integrated appliance and activity recognition mechanism for IoT energy management system. The authors presented a management service layer for the recognition of current household appliances. The latter not only establishes communication services among various appliances, but also deduces human activities conducted for context data using Naive Bayes from the electric appliances in use and the variation of its states. The authors focus on studying the characteristics of the house appliances. Dejan et. al [6] proposed an architecture, describing its key components and depicting in scenarios its usage with the goal of enabling facility management which enables informed business decisions by following enterprise strategies as well as considering the volatility of the available energy excess or shortage. The authors' main contribution is to provide the architecture and concept of the energy efficiency in smart grid era. In [5], the authors experiments the performance of the different levels of energy data aggregation. Thousands of smart meters are aggregated, by usage of the collected energy readings from a real-world trial. Using a selected data set, the traditional database system (row-based) performance is compared to the emerging column-based approach in order to assess the suitability for real-time analytic in such scenarios. The main contributions of the authors is to show that the in-memory column-based database system is more suited to aggregate energy data. According to the authors in [11], reducing forecast errors can be achieved by clustering prosumers, but prediction algorithms are still sensitive to smaller levels of aggregation. The authors also in-

(a) 2014-01-Weekday dataset (b) 2014-02-Weekday dataset

(c) 2014-03-Weekday dataset (d) 2014-01-Holiday dataset

(e) 2014-02-Holiday dataset (f) 2014-03-Holiday dataset

Figure 6: Results of our energy consumption prediction model.

vestigated the role of distributed storage in residential areas as well as a mean towards creating groups of prosumers that feature better forecast energy behavior. Several other researches have been done related to the energy big data systems and several companies [2, 4] provide services on energy big data analysis. However, as authors aware, this is the first paper to gather and analyze the real massive power data and to study the usage characteristics based on the timing properties. Our prediction model shows a decent performance and can be used to plan an efficient energy usage plan.

5 Concluding Remark and Future Work

In this paper, we propose a proof-of-concept prototype about a big data management system for managing large-scale energy data. For this, we present how to build this system in distributed environments. Our proposed system gives to us better ideas to high availability as well as high performance at low cost. These points of our system are better than existing, commercial databases like Oracle and IBM-DB2. To utilize our proposal, we also tackled a energy consumption prediction problem that is very significant problem to the field of energy efficiency. In our approach, we propose novel prediction model based on penalized linear regression-based map/reduce algorithms. Our model will be executed in parallel on the big data management system. Given a particular time stamp, our learning model will predict the active power value. Then, in our experimental section, we showed the effectiveness of our model. In addition, our learning model is about 1.72 faster than that on a single machine.

As our future direction in our research, above all, we have a plan to add about 20 data nodes to the current system. For better energy consumption prediction, we would like to exploit various features rather than time stamp-based features. In addition, we will extend our linear regression-based map/reduce algorithms to various machine learning algorithms such as SVM, Decision Tree, and so on. Finally, we will improve the user interface of our current prototype system.

Acknowledgement

This research was supported by the Energy Efficiency & Resources of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea Government Ministry of Trade, Industry & Energy (No. 20132010101800).

References

[1] Apache Software Foundation, “Hadoop 2.2”, http://hadoop.apache.org/ (2014)

[2] AutoGrid, http://www.AutoGrid.com (2014)

[3] Berges, M., Goldman, E., Matthews, H., and Soibelman, L., “Training load monitoring algorithms on highly sub-metered home electricity consumption data”, Tsinghua Science and Technology 13, Supple, 0 (2008), pp.406–411

[4] Big Data Energy, http://www.bigdataenergyservices.com (2013)

[5] Ilic, D., Karnouskos, S., and Wilhelm, M., “A comparative analysis of smart metering data aggregation performance”, IEEE International Conference on Industrial Informatics, Bochum, Germany, July 2013

[6] Ilic, D., Karnouskos, S., Silva, P., and Detzler, S., “A system for enabling facility management to achieve deterministic energy behaviour in the smart grid era”, International Confernce on Smart Grids and Green IT systems (SmartGreen 2014), 2014

[7] Karnouskos, S., Colombo, A., Lastra, J., and Popescu, C., “Towards the energy efficient future factory”, IEEE International Conference on Industrial Informatics (INDIN 2009)

[8] Karnouskos, S., Ilic, D., and Silva, P., “Assessment of an enterprise energy service platform in a smart grid city pilot”, International Conference on Industrial Informatics (INDIN), 2013

[9] Lai, C., Lai. Y., Tianruo, L., and Chao, H., “Integration of IoT energy management system with appliance and activity recognition”, IEEE International Conference on Green Computing and Communications (GreenCom 2012), DOI 10.1109

[10] Leskovec, J., “Mining massive datasets”, http://web.stanford.edu/class/cs246/slides/01-mapreduce.pdf (2013)

[11] Lic, D., Karnouskos, S., and Silva, P., “Improving load forecast in prosumer clusters by varying energy storage size”, IEEE PowerTech 2013, June 2013

[12] Santaferraro, J., “Offloading analytics”, Business Intelligence Journal, 17(4):43–48 (2012)

[13] Yang, K., “Simple one-pass algorithm for penalized linear regression with cross-validation on MapReduce”, e-Print arXiv:1307.0048 (2013)

[14] Yin, J., Kulkarni, A., Purohit, S., Gorton, I., and Akyol, B., “Scalable real time data management for smart grid”, Middleware Industry Track Workshop 2011