2012 IEEE 4th International Conference on Cloud Computing Technology and Science

Using Cloud Technologies for Large-Scale House Data in Smart City

Shintaro YAMAMOTO, Shinsuke MATSUMOTO, Masahide NAKAMURA
Graduate School of System Informatics, Kobe University, JAPAN
1-1 Rokkodai-cho, Nada-ku, Kobe, Hyogo 657-8501, Japan
Email:shintaro@ws.cs.kobe-u.ac.jp, {shinsuke, masa-n}@cs.kobe-u.ac.jp

***Abstract*—In the smart city environment, a wide variety of data are collected from sensors and devices to achieve value-added services. In this paper, we especially focus on data taken from smart houses in the smart city, and propose a platform, called Scallop4SC, that stores and processes the large-scale house data. The house data is classified into log data or configuration data. Since the amount of the log is extremely large, we introduce the Hadoop/MapReduce with a multi-node cluster. On top of this, we use HBase key-value store to manage heterogeneous log data in a schemaless manner. On the other hand, to manage the configuration data, we choose MySQL to process various queries to the house data efficiently. We propose practical data models of the log data and the configuration data on HBase and MySQL, respectively. We then show how Scallop4SC works as a efficient data platform for smart city services. We implement a prototype with 12 Linux servers. We conduct an experimental evaluation to calculate device-wise energy consumption, using actual house log recorded for one year in our smart house. Based on the result, we discuss the applicability of Scallop4SC to city-scale data processing.**

***Keywords*- smart city; smart house; data platform; Hadoop; HBase;**

I. INTRODUCTION

Smart city is a next-generation city planning that aims to achieve sustainable society with ICT technologies [1][2][3]. In a smart city, various kinds of data are collected from sensors and devices. The data include, for instance, traffic conditions of roads, power consumption of houses, health status of inhabitants, and environmental measures. These data are analyzed with advanced data processing technologies to achieve *smart city services*. A wide variety of services are expected, such as traffic optimization [4], community-based energy saving [5], local economic trend analysis [6], entertainment [7], community-based health care [8], disaster prevention [9] and agriculture support [10].

Although there are many challenges to realize such smart city services, we especially focus on *house data* in this paper. The house data refers to any data collected from smart houses to achieve the smart city services. Typical house data include energy consumption (of a whole house or every device), status of appliances (e.g., TV channels, set temperature of air-conditioner) and environmental measures (e.g., room temperature, humidity). These data are recorded with date and time, characterizing *dynamic* contexts and history of a house. We call such house data, *house log data*.

In addition to that, smart city services may require *static* meta-data like, address of a house, name and type of an appliance, people living in a house, etc. We call such static data, *house configuration data*. Although security and privacy issues must be addressed carefully, the house data contain crucial information to achieve beneficial services.

The goal of this study is to implement a data platform that can store and process the large-scale house data efficiently. For this, we tackle two challenges in this paper.

The first challenge is to consider how to manage large-scale house data. Especially, the house log becomes huge, since it is collected periodically from various kinds of appliances and sensors deployed in a number of houses. Therefore, we propose a scalable data platform, called *Scallop4SC (SCALable LOgging Platform for Smart City)*, to store and process the large-scale house data. In Scallop4SC, we extensively use the *Hadoop/MapReduce* [11] to process large-scale log data. Since it is not practical to manage heterogeneous house log with a rigorous data schema, we employ *HBase key-value store* [12] to manage the house log in a schemaless way. Scallop4SC also uses MySQL [13] for efficient queries to the house configuration data.

The second challenge is to conduct data modeling of Scallop4SC. It is difficult to enumerate all possibilities how a variety of services use the house data stored in the platform. Hence, a solid model is crucial to make Scallop4SC independent of specific services or applications. For this, we propose practical data models of the house configuration and the house log on MySQL and HBase, respectively.

Based on the idea, we implement a prototype of Scallop4SC as a Hadoop cluster comprising 12 Linux servers. Using the prototype, we have conducted an experimental evaluation to calculate device-wise energy consumption, using actual house log recorded for one year in our smart house environment. Based on the result, we discuss the applicability of Scallop4SC to city-scale data processing. Assuming a middle-class smart city with 62,000 households, each of which sends house log of 30 appliances every minute, it is shown that Scallop4SC can process daily batch around one hour, using a native MapReduce program.

The concept of Scallop4SC has been originally published in our fast-abstract paper [14] (one-page length). This paper delivers our contribution as a full-length paper, which makes substantial changes to the previous version.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

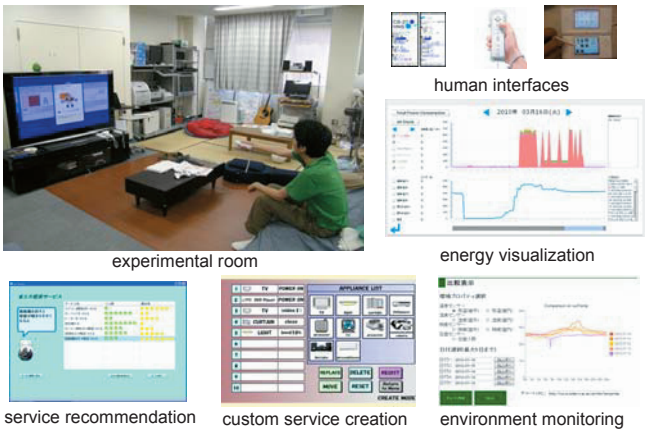

Figure 1. CS27-HNS and applications

II. Preliminaries

A. Smart City and Smart House

Smart city is a next-generation city planning that aims to achieve sustainable society with ICT technologies. The principle of the smart city is to gather data of the city first, and then to provide appropriate services based on the data. Thus, a variety of data are collected from sensors, devices, cars and people across the city. The smart city is not only the theoretical concept. Recently, several pilot projects have been started in various cities around the world, such as Yokohama, Singapore and Amsterdam [1][2][3].

Data from houses are essential to achieve the smart city services, since a house is a primary construct of a city. It is expected in the near future that various kinds of house data will be obtained using technologies of *smart houses*.

Our research group is also developing an home network system, called *CS27-HNS*, as a core technology of the smart house [15][16]. Figure 1 shows our experimental room and related applications. In CS27-HNS, household appliances like TVs, DVDs, lights, air-conditioners are connected to a network. There are also sensors like temperature, humidity, brightness, human-detection, etc. Orchestrating these devices via network implements various value-added services. CS27-HNS can also obtain data from these devices through APIs, and send them to external network clouds. We have been storing house data from all appliances and sensors for years, characterizing the *history* of our research lab.

B. House Log and House Configuration

We assume that there are a number of smart houses in a smart city, and that every house sends its house data to the external cloud to use beneficial smart city services[1]. For this, there are two kinds of house data. The one is data that dynamically changes the value as time goes. For example, power consumption, status of an appliance, room temperature, and so on. These kinds of data are often associated with date and time, and recorded as log. So, we call them *house log*. Another is static data explaining configuration of houses, including address of a house, IDs of devices, appliance names and types, floor plan, inhabitant information, etc. We call these data *house configuration*.

A smart city service is implemented by using both *house log* and *house configuration*. For example, suppose a service that calculates today's power consumption in Nada ward. The service first identifies all houses in Nada ward from house configuration, then retrieves all logs of power consumption for the houses, and finally sums up the values.

C. Challenges of Managing House Data

In this paper, we address the following two challenges.

Challenge C1: Store and Process Large-Scale Heterogeneous House Log: In a smart city, every smart house sends the house log periodically to the cloud. Although the data size of each log is small, the number of houses multiplying by the number of devices within a house yields vast amounts of log data. For example, when a house sends house log of 30 appliances every one minute, 43,200 records will be obtained within one day. With a small city containing 25,000 houses, the total number of daily records exceeds one billion. Moreover, the house log are mixed with various types of data. Thus, we need a scalable data platform that can store and process such large-scale and heterogeneous house log.

Challenge C2: Construct Solid Data Model of House Data: It is expected that a variety of services use the house data in different ways. Since we cannot imagine all possible queries in advance, we need to define a solid data model for the house data, which is independent of specific services or applications. Especially, a data model of the house configuration must be considered carefully, since the query is often specified with static attributes (e.g., city, house, device, people, etc.). As far as we know, there is no reference data model that specifically manages the house data within the context of smart city.

III. Scallop4SC: Platform for Large-scale House Log in the Smart City

A. System Architecture

To cope with Challenge C1, we propose a data platform, called *Scallop4SC* (SCALable LOgging Platform for Smart City), in this section. To store and process the large-scale and heterogeneous house log, Scallop4SC extensively uses the Hadoop technology. More specifically, it employs Hadoop/MapReduce [11] for parallel data processing of the house log. We also introduce HBase key-value store [12] to manage heterogeneous data in a schemaless fashion. Thus, the two cloud technologies fit well the nature of the house log management.

[1]Although security and privacy issues must be addressed carefully, these are beyond the scope of the paper.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

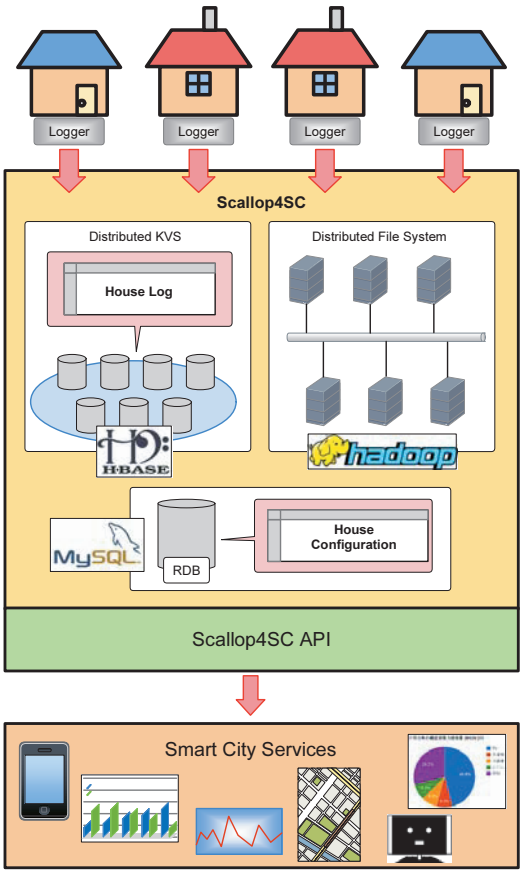

Figure 2. Overall architecture of Scallop4SC

Scallop4SC also contains a relational database (MySQL [13]) to enable flexible queries towards the house configuration data. Thus, Scallop4SC takes a hybrid configuration of SQL and No-SQL databases.

Figure 2 shows the overall architecture of Scallop4SC. In each smart house, house log is taken by a *house logger*, and sent to Scallop4SC. The house log is then stored in HBase of Scallop4SC. If necessary by any services, the house log is analyzed by Hadoop. The house configuration data is stored in the MySQL database. We suppose that the configuration data is input by the owner of a smart house, when the house is newly registered in a smart city, or any information is updated.

The house data stored in the platform are provided for external services and applications, via *Scallop4SC API*. For a given query, the API retrieves necessary data from HBase and MySQL. We suppose that the API should be exhibited as a platform independent Web service.

B. Type of House Log

In the finest granularity, the house log is taken from an appliance or a sensor deployed in a smart house. We classify such house log into the following three types.

1) **Energy Log:** It characterizes the history of energy consumption. The energy involves electricity, water, gas, and so on. For example, "2012-07-28 12:34:56 Power of TV is 600W." is a energy log.
2) **Device Log:** Log taken from appliances, involving the history of status and operations of appliances. For example, "2012-07-28 12:34:56 TV is off." is a device log generated from a TV status. On the other hand, "2012-07-28 12:35:00 TV is turned on." is a device log generated from a TV operation.
3) **Environment Log:** Log taken from sensors, characterizing the environmental context of a house. For example, "2012-07-28 12:34:56 temperature is 24 degree." is an environment log taken from a temperature sensor. Similarly, "2012-07-28 12:34:56 number of people is 3." is log taken from a people counter.

C. Collecting House Log

In our architecture, a house log is collected by a house logger in a smart house. For a device, the logger first obtains data by executing appropriate API of the device. For instance, in case of CS27-HNS, a status of a TV is obtained by REST Web-API http://cs27-hns/TVService/getStatus. Then, the logger associates the obtained data with the current date, time, log type, house ID and device ID. Finally, the logger sends the house log record to Scallop4SC.

The house logger *periodically* queries designated appliances and sensors, and uploads the log to Scallop4SC. The period should be adjusted considering application requirements and network capacity. If the period is short, it is good for real-time applications, but it yields a large amount of log. If the period is long, the data size becomes small, but it might miss some events.

D. Storing House Log in HBase

Since the house log includes heterogeneous type of data, it is unpractical to determine strict data schema in advance. Therefore, we extensively use the *schemaless* data management with the simple key-value of HBase. For instance, all the five log in Section III-B are different from each other. However, HBase can flexibly include them as follows.

Key	Value
2012-07-28T12:34:56.cs27.Energy.tv01	{value:600, unit:W}
2012-07-28T12:34:56.cs27.Device.tv01	{status:[power:off]}
2012-07-28T12:35:00.cs27.Device.tv01	{operation:on()}
2012-07-28T12:34:56.cs27.Env.temp2	{value:24.0, unit:celsius}
2012-07-28T12:34:56.cs27.Env.pcount3	{value:3, unit:people}

In this example, we define the key as a concatenation of [date, time, house ID, log type, device ID], which forms a unique string of every house log. As for the value, each log defines own hash to describe the content. The key-value store can receive benefit of high scalability and easy duplication. Instead, complex queries for data search is significantly limited. For this, we need to devise the data modeling, as will be shown in Section IV.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

E. Storing House Configuration in MySQL

In this paper, we assume that the house configuration manages the following information for the smart city services.

1) **House Information:** Manages information of all smart houses registered in a smart city.
2) **Device Information:** Manages information of all devices (e.g., appliances, sensors, equipments) deployed in each smart house.
3) **Personal Information:** Manages information of people living in each smart house.

The house configuration data is not so large or heterogeneous as the house log. It is therefore convenient to store them in RDB like MySQL. However, to allow various applications and services to access the data, a solid data model is required. The details will be discussed in Section IV.

F. Using Hadoop for Processing Large-Scale House Data

A specific nature of smart city services is to use *global statistics* of multiple houses in the city, rather than locally focusing on individual houses. For example, questions addressed by the services include; "Q1: How much energy was used today by all air-conditioners in the city?", "Q2: How many houses are watching TV channel #10 now?", "Q3: What time did 80% of all houses turn off lights?", "Q4: What is the average room temperature of the city?", etc. To answer most these questions, we do not need very sophisticated algorithms with complex dependencies. Rather than that, we apply simple statistical calculation for a huge amount of records of house log. Hadoop/MapReduce is good at that kinds of jobs, since they can be paralleled.

The basic strategy of the data processing is as follows. For a given query q, we first identify a set of devices D_q matching q, based on the house configuration in MySQL. Then, for the house log in HBase, we derive a set L_q of house log, where $l \in L_q$ is a log of a device $d \in D_q$. Finally, we apply MapReduce to L_q. For example, let us consider the above Q1. From the house configuration in MySQL, we first identify device IDs of all air-conditioners in the city. Then, from house log in HBase, we retrieve all energy logs with those device IDs and date of today (Map phase). Finally, we sum up the value of the electric consumption (Reduce phase).

For the implementation of data processing programs, we consider native Java MapReduce and Pig [17].

G. Scallop4SC API

Scallop4SC provides API to allow external applications and services to access the house data. We consider the following three types.

1) **SimpleDataAPI:** Provides simple database operations to get, insert, update, delete data.
2) **StatisticAPI:** Provides generic statistical operations like sum, average, maximum, minimum, etc.

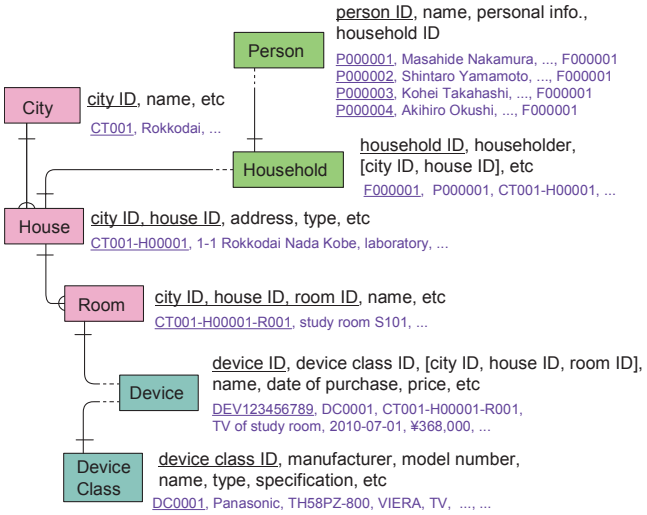

Figure 3. Data model of house configuration

3) **CustomMapReduceAPI:** Receives a custom MapReduce program, executes the program, and returns the result.

The design and implementation of the API is currently under way.

IV. Constructing Data Model for House Data

To cope with Challenge C2, we conduct data modeling for house data in this section. We present two data models. The first data model is for the house configuration, and the second data model is for the house log.

A. Data Model of House Configuration

Since the house configuration is stored in MySQL, we conduct the conventional data modeling for RDB. Figure 3 shows the proposed data model for the house configuration, which is depicted by an *entity-relationship diagram (ERD)*.

This ERD follows a special notation presented in [18]. A square represents an entity, and data items are aligned to the right. An underlined item represents a primary key, whereas [] represents a composite foreign key. Instances are represented under each entity. Between entities, there may be a parent-child relationship (depicted in $+$—$\in$), or a reference relationship (drawn by $+$—$\cdots$).

The proposed data model is carefully designed so as to involve three primary information of the house configuration (see Section III-E).

House Information: The house information is stored in structured entities: *City, House* and *Room*. City entity represents information of every instance of smart city. Each city has many houses. A house is explained by an address, floor plan, type, etc. A house has multiple rooms. A room is the minimal construct of the house information, where devices are installed or deployed. As shown in the ERD, there are parent-child relationships between [City and

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

House] and [House and Room], implying children cannot exist if a parent does not exist.

Device Information: The device information is defined by two entities: *Device* and *Device Class*. Device entity represents every instance of devices, whereas Device Class defines a class of device. A device is explained by a device ID, a device class, naming for description, and miscellaneous information like date of purchase, price of purchase, etc. More importantly, a device refers to a room where the device is deployed or installed, allowing applications to identify the device by location, house, owner, etc. A device class is defined by meta-data commonly referred by multiple instances. It includes a manufacturer, a model number, a type, a specification.

Personal Information: The personal information is defined by two entities: *Person* and *Household*. Person entity defines an individual by personal information like name, age, etc. Each person is bound with a household. Household entity defines a household living in a smart house, which has a reference to a house. If the household is moved to another house, the reference will be changed.

Since the data model of the house configuration is implemented as a data schema in MySQL, we can use powerful data search with SQL. For example, to obtain device IDs of all the air-conditioners in a city CT001, we just specify an SQL statement as follows:

```
SELECT `deviceID` from 'DEVICE' WHERE
    `deviceClassID` IN (
         SELECT `deviceClassID` from
         'DEVICECLASS' WHERE
              `type` = 'air-conditioner'
     )
```

B. Data Model of House Log

The house log is managed by HBase in Scallop4SC. The key-value store achieves high scalability and easy duplication for large-scale data. However, it lacks flexible search queries as SQL performs. We need to devise the data model so that applications can access the house log efficiently.

A naive data model with HBase is the one shown in Section III-D. However, this model has the following problems.

(P1) Search queries are limited significantly: Due to constraint of HBase, data are basically searched by *prefix search* of row key. In the naive model, it is possible to search the house log by date. However, we cannot search them by device or log type, without specifying the prefix items.

(P2) Data are not evenly distributed: The row key with *monotonically increasing values* tends to concentrate data records in a single server [19]. Thus, the house log records are not evenly distributed to multiple servers in the Hadoop cluster, which causes a performance bottleneck.

To cope with these problems, we design a data model with two kinds of HBase tables: *HouseLog* and *HouseIndex*.

Table I shows the HouseLog table. It stores the house log with two kinds of column families: *Data* and *Info*. Data column stores contents of the log, whereas Info column stores meta-data commonly used by any kinds of house log. By applying a *value filter* to Info, we can filter specific rows by value. For the row key, we use a *hash encoded string* (with timestamp) to distribute data rows evenly to multiple servers. The prefix search for the row key is not applied directly to HouseLog. Instead, we use the HouseIndex table.

Table II shows the HouseIndex table, which stores a pointer (i.e., index) to each house log data. The row key is constructed from *permutation* of (1) time stamp, (2) log type, (3) house ID, and (4) device ID, which allows applications to search house log by any kinds the four attributes. Column family hlkey contains the row key of HouseLog Table. In Table II, all $24\ (= 4!)$ entries point the same house log record (46018eab). Thus, an application first applies the prefix search (by any of date, house, log type or device) to the HouseIndex table, and obtains row keys of the corresponding house log records. Then, the application gets the house log records from the HouseLog table, by specifying the row keys obtained.

Thus, the proposed data model well circumvent the problems P1 and P2.

V. Implementation

Based on the above idea, we have implemented a prototype of Scallop4SC, with a Hadoop cluster of 12 Linux servers. The platform and middleware used in the prototype are; Vine Linux 6.0, Java SE 7u5, Hadoop 0.20.2+737, HBase 0.90.3 and Pig 0.9.1. Table III shows configuration of the cluster. HDP00 is a *master node* that manages distributed data processing centrally. HDP01, HDP02 and HDP03 are *ZooKeepers*, which supports decentralized cooperation between the nodes [20]. The cluster is made up from low-end PCs with two different types of processors (Pentium 4 and Athlon 64)[2].

VI. Experimental Evaluation

A. Overview of Experiment

We conduct an experiment to evaluate performance of the proposed Scallop4SC prototype. The performance is evaluated by a *batch job*, which accumulates power consumption of every device in CS27-HNS. In the experiment, we use actual house log recorded for years in CS27-HNS (see Section II-A). Within Scallop4SC, the job is implemented by two versions of programs. The one is native Java MapReduce program, the other is a Pig script [17]. Just for reference, we also implement the job in SQL, by importing partial data to MySQL. We compare processing time of the three.

[2]The reason why we used such a bit out-of-date machines is because we just collected unused machines in our laboratory. With the latest machine with more memory, Scallop4SC would show much better performance.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

Table I
HOUSE LOG TABLE

Row Key	Column Families							
	Data:	Info:						
hash()@<timeStamp>		date:	time:	device:	house:	unit:	location:	type:
46018eab@2012-07-28T12:34:56	600	2012-07-28	12:34:56	tv01	cs27	W	living room	Energy
53cd5fad@2012-07-28T12:34:56	[power:off]	2012-07-28	12:34:56	tv01	cs27	status	living room	Device
8afd4abd@2012-07-28T12:35:00	on()	2012-07-28	12:34:56	tv01	cs27	operation	living room	Device
e85afaad@2012-07-28T12:34:56	24.0	2012-07-28	12:34:56	temp3	cs27	celsius	kitchen	Environment
13ffadbc@2012-07-28T12:34:56	3	2012-07-28	12:34:56	pcount3	cs27	people	living room	Environment
:	:	:	:	:	:	:	:	:

Table II
HOUSE INDEX TABLE

Row Key	Column Family
	hlkey:
[timeStamp.type.house ID.device ID]!	hash()@<timeStamp>
2012-07-28T12:34:56.Energy.cs27.tv01	46018eab@2012-07-28T12:34:56
2012-07-28T12:34:56.Energy.tv01.cs27	46018eab@2012-07-28T12:34:56
2012-07-28T12:34:56.cs27.Energy.tv01	46018eab@2012-07-28T12:34:56
2012-07-28T12:34:56.cs27.tv01.Energy	46018eab@2012-07-28T12:34:56
2012-07-28T12:34:56.tv01.Energy.cs27	46018eab@2012-07-28T12:34:56
2012-07-28T12:34:56.tv01.cs27.Energy	46018eab@2012-07-28T12:34:56
Energy.2012-07-28T12:34:56.cs27.tv01	46018eab@2012-07-28T12:34:56
Energy.2012-07-28T12:34:56.tv01.cs27	46018eab@2012-07-28T12:34:56
Energy.cs27.2012-07-28T12:34:56.tv01	46018eab@2012-07-28T12:34:56
Energy.cs27.tv01.2012-07-28T12:34:56	46018eab@2012-07-28T12:34:56
Energy.tv01.2012-07-28T12:34:56.cs27	46018eab@2012-07-28T12:34:56
Energy.tv01.cs27.2012-07-28T12:34:56	46018eab@2012-07-28T12:34:56
cs27.2012-07-28T12:34:56.Energy.tv01	46018eab@2012-07-28T12:34:56
cs27.2012-07-28T12:34:56.tv01.Energy	46018eab@2012-07-28T12:34:56
cs27.Energy.2012-07-28T12:34:56.tv01	46018eab@2012-07-28T12:34:56
cs27.Energy.tv01.2012-07-28T12:34:56	46018eab@2012-07-28T12:34:56
cs27.tv01.2012-07-28T12:34:56.Energy	46018eab@2012-07-28T12:34:56
cs27.tv01.Energy.2012-07-28T12:34:56	46018eab@2012-07-28T12:34:56
tv01.2012-07-28T12:34:56.Energy.cs27	46018eab@2012-07-28T12:34:56
tv01.2012-07-28T12:34:56.cs27.Energy	46018eab@2012-07-28T12:34:56
tv01.Energy.2012-07-28T12:34:56.cs27	46018eab@2012-07-28T12:34:56
tv01.Energy.cs27.2012-07-28T12:34:56	46018eab@2012-07-28T12:34:56
tv01.cs27.2012-07-28T12:34:56.Energy	46018eab@2012-07-28T12:34:56
tv01.cs27.Energy.2012-07-28T12:34:56	46018eab@2012-07-28T12:34:56
:	:

Table III
CONFIGURATION OF SCALLOP4SC PROTOTYPE

Node Name	Node Type		Spec	
	Hadoop	HBase	CPU	Mem.
HDP00	Master	HRegionMaster	Pentium4 3.00GHz	400MB
HDP01	Slave	Zookeeper	Pentium4 3.00GHz	1GB
HDP02	Slave	Zookeeper	Pentium4 3.00GHz	1GB
HDP03	Slave	Zookeeper	Pentium4 3.00GHz	1GB
HDP04	Slave	HRegionServer	Pentium4 3.00GHz	1GB
HDP05	Slave	HRegionServer	Pentium4 3.00GHz	1GB
HDP06	Slave	HRegionServer	Pentium4 3.00GHz	1GB
HDP07	Slave	HRegionServer	Pentium4 3.00GHz	1GB
HDP08	Slave	HRegionServer	Pentium4 3.00GHz	1GB
HDP09	Slave	HRegionServer	Athlon 64 3800+	1GB
HDP10	Slave	HRegionServer	Athlon 64 3800+	1GB
HDP11	Slave	HRegionServer	Athlon 64 3500+	1GB

B. Target House Data

In CS27-HNS, we are collecting various house log over two years, including environmental measures, appliance status and power consumption. In the experiment, we use the *power consumption log*, which is being recorded every six seconds from 32 appliances. The total number of records is currently reaching 460 millions. The batch job in the experiment is to accumulate total power consumption from the log for *each* of the 32 appliances.

To see the performance over the data size, we vary the term of house log as follows: one month, two months, three months, six months, one year, two years, three years and four years. For the cases of three and four years, we duplicate the existing data to adjust the size.

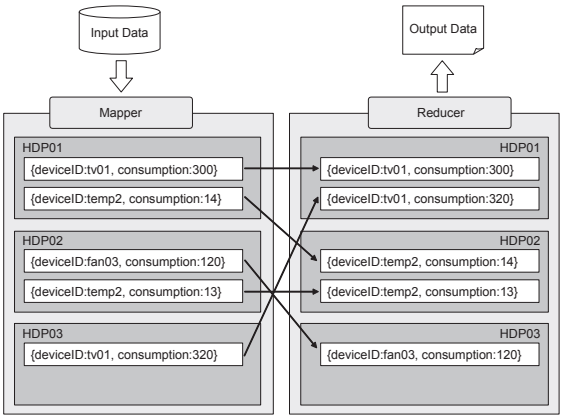

Figure 4. MapReduce for calculating device-wise power consumption

C. Implementing Batch Job

Native MapReduce Program: The house log data is processed by a native MapReduce program written in Java. The flow of data processing is shown in Figure 4. In Map phase, the consumption logs distributed over several nodes are converted into key-values with deviceID as a primary key. In Shuffle phase, data with the same deviceID are aggregated in the same node. Finally, in Reduce phase, the power consumption is accumulated for each deviceID. The lines of code of the Java MapReduce in this experiment is 57, except blank and comment lines.

Pig Script: We can do the same job using a high-level language Pig. Using Pig, a developer can perform large-scale data analysis with Hadoop, easily and interactively. An example script used in the experiment is shown as follows.

```
1 records = LOAD 'hbase://HouseLogTable'
2  USING HBaseStorage('info:device data:*','..')
3  AS (id:bytearray, deviceID:chararray,
4   power:double);
```


978-1-4673-4510-1/12/$31.00 ©2012 IEEE

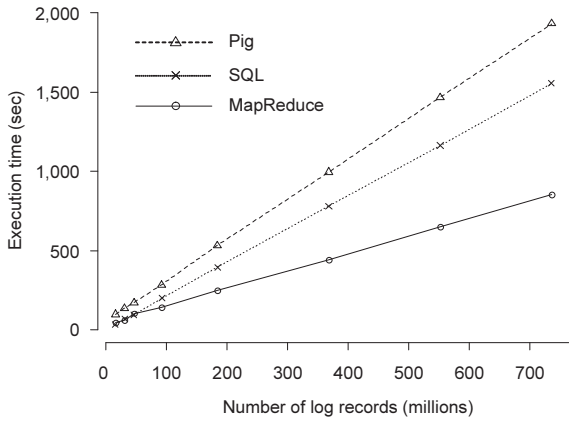

Figure 5. Result of experiment

```
grouped_records = GROUP records BY deviceID;
sum = FOREACH grouped_records
 GENERATE group, SUM(records.power);
STORE sum INTO 'Result.txt';
```

Conventional SQL: Just for a reference, we implement the same job in SQL. The device-wise power consumption can be easily calculated by "Sum" built-in function with "GROUP BY" statement.

```
SELECT `DeviceID`, Sum(`powerConsumption`)
 FROM `ConsumptionLogTable` GROUP BY `DeviceID`
```

D. Result

The experimental result is shown in Figure 5. The x-axis represents the number of house log records, and the y-axis plots the execution time of the batch job. Points from left to right correspond to the data of one month, two months, three months, six months, one year, two years, three years and four years, respectively.

The execution time for every method was approximately proportional to the number of data records. SQL with one month achieved the best performance. However, as the data size increased, the performance was good in the order of MapReduce, SQL and Pig. The processing time by SQL for the largest data (four years log) was about 26 minutes. In contrast, MapReduce program could complete the data processing in about half time (14 minutes). Though Pig used the same Scallop4SC infrastructure as MapReduce did, it was no match for other two methods.

E. Feasibility for City-Scale House Data

Based on the result, we simulate the feasibility of Scallop4SC for actual scale of smart cities. As candidates of the smart city, we choose three different scale of municipalities: *Kobe City*, *Nada Ward* and *Rokkodai Town*, which are local to our affiliation. Kobe City is government-decreed city in Japan, comprising 700,000 households. Nada Ward is one of nine wards in Kobe City, and there are 62,000 households. Rokkodai Town is a town where Kobe University is located, and there are 1,200 households.

Table IV
ESTIMATED PROCESSING TIME OF SCALLOP4SC FOR SMART CITIES

Logging Period	Kobe City	Nada Ward	Rokkodai Town
Case 1 (every 1 min)	730 min	65 min	1 min
Case 2 (every 6 sec)	7,298 min	651 min	12 min

In the simulation, we assume that every house in a smart city takes the house log from 30 devices, and sends the data periodically to Scallop4SC. As for the data transmission period, we consider two cases:

- (Case 1) House log is sent every one minute, assuming statistic monitoring services.
- (Case 2) House log is sent every six second, assuming real-time services.

Our interest here is how much time is required to calculate device-wise energy consumption every day. We estimate the number of daily house log records for each smart city, and derive the estimated data processing time with the MapReduce program based on the experimental result.

Table IV shows the estimated processing time for one-day house log data for each city. In Case 1, time taken for Kobe City is more than 12 hours, which is impractical for a daily batch job. The result shows that Kobe City is too large for the current prototype of Scallop4SC to manage it as a smart city. As for Nada Ward and Rokkodai Town, Scallop4SC is well capable of processing the data within reasonable time. Thus, a ward-scale smart city would be appropriate for the current implementation of Scallop4SC.

In Case 2, even Nada Ward takes 10 hours, as the data size grows significantly. Thus, for such a real-time logging, we need to *scale out* Scallop4SC, by adding more computing nodes. Appropriate data transmission period may vary among smart city services. It is also a challenging problem to determine right number of servers from a given smart city configuration. We leave them our future work.

VII. RELATED WORK

Research and development of smart city services are now a hot topic in various domains, such as traffic [4], energy saving [5], economics [6], entertainment [7], health care [8], disaster prevention [9] and agriculture [10], as seen in Section I. These existing studies basically focus on how the services can contribute to the human society, and what features the services should provide. Few of them consider concrete data platform to manage large-scale data to implement the service. Al-Hader et. al. [21] discuss the importance of smart city infrastructure. However, detailed design and implementation are not given yet.

As for the house data model, Japanese smart house information standardization forum (eSHIPS) has proposed a reference data schema in [22]. However, the data schema is designed specifically for energy-saving applications of smart houses. It does not suppose city-scale data size or heterogeneous data type, as we addressed in this paper.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

Yokohama Smart City Project (YSCP) [1] is conducting an experiment with common households. It demonstrates community-scale energy-saving activities using HEMS and BEMS. It appears that the collected house log is energy log only. Consideration of large-scale and heterogeneous house data seems to be a future challenge.

The proposed Scallop4SC currently deals with house data only. However, we consider it not difficult to adapt it to other kinds of data, contributing to efficient development of the above smart city services.

VIII. Conclusion

In this paper, we have presented a data platform, called Scallop4SC, which stores and processes large-scale house data in smart city. Scallop4SC extensively uses Hadoop MapReduce and HBase to manage large-scale house log. We have proposed concrete data models for house configuration and house log. We have also implemented a prototype of Scallop4SC and evaluated through an experiment. The feasibility simulation has shown that Scallop4SC can sufficiently manage simple data aggregation of ward-scale smart cities, where every house sends log of 30 appliances every minute.

In our future work, we are currently developing Scallop4SC APIs, to allow external application to access the platform more easily. We will also develop a method to configure Scallop4SC from given information of smart city.

Acknowledgments

This research was partially supported by the Japan Ministry of Education, Science, Sports, and Culture [Grant-in-Aid for Scientific Research (C) (No.24500079), Scientific Research (B) (No.23300009)], and Kansai Research Foundation for technology promotion.

References

[1] City of Yokohama, "Yokohama smart city project," http://www.city.yokohama.lg.jp/ondan/english/.

[2] R. G. Hollands, "Will the real smart city please stand up?" *City: analysis of urban trends, culture, theory, policy, action*, vol. 12, no. 3, pp. 303–320, 2008.

[3] A. Mahizhnan, "Smart cities: The singapore case," *Cities*, vol. 16, pp. 13–18, 1999.

[4] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, "Optimizing traffic lights in a cellular automaton model for city traffic," *Phys. Rev. E*, vol. 64, p. 056132, 2001.

[5] S. Massoud Amin and B. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *Power and Energy Magazine, IEEE*, vol. 3, no. 5, pp. 34–41, 2005.

[6] IBM, "Apply new analytic tools to reveal new opportunities," http://wwiw.bm.com/smarterplanet/nz/en/business_analytics/article/it_business_intelligence.html.

[7] I. Celino, S. Contessa, M. Corubolo, D. Dell'Aglio, E. D. Valle, S. Fumeo, and T. Krüger, "Urbanmatch - linking and improving smart cities data," in *Linked Data on the Web (LDOW2012)*, 2012.

[8] IBM, "eHealth and collaboration - collaborative care and wellness," http://www.ibm.com/smarterplanet/nz/en/healthcare_solutions/nextsteps/solution/X056151Y25842W14.html.

[9] C. Hu and N. Chen, "Geospatial sensor web for smart disaster emergency processing," in *International Conference on GeoInformatics (Geoinformatics2011)*, 2011, pp. 1–5.

[10] Y. Cho, J. Moon, and H. Yoe, "A context-aware service model based on workflows for u-agriculture," in *International Conference on Computational Science and Its Applications (ICCSA2010)*, vol. 6018, 2010, pp. 258–268.

[11] D. Borthakur, "The hadoop distributed file system: Architecture and design," http://hadoop.apache.org/common/docs/r0.18.0/hdfs_design.pdf, 2007.

[12] A. Khetrapal and V. Ganesh, "Hbase and hypertable for large scale distributed storage systems," http://www.uavindia.com/ankur/downloads/HypertableHBaseEval2.pdf, 2006.

[13] MySQL, http://www.mysql.com/.

[14] S. Yamamoto, H. Seto, S. Matsumoto, and M. Nakamura, "Scallop4SC: Data platform for storing and processing large-scale house log in smart city," in *Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT2010)*, 2012 (to appear), fast abstract.

[15] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. Matsumoto., "Constructing home network systems and integrated services using legacy home appliances and web services," *International Journal of Web Services Research*, vol. 5, no. 1, pp. 82–98, 2008.

[16] H. Igaki, H. Seto, M. Fukuda, and M. Nakamura, "Mashing up multiple logs in home network system for promoting energy-saving behavior," in *Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT2010)*, vol. CDROM, 2010.

[17] Apache Pig, http://pig.apache.org/.

[18] K. Watanabe, *Hanbai Kanri System de Manabu Modeling Koza*. Shoeisha, 2008 (in Japanese).

[19] Apache Software Foundation, "Apache hbase reference guide," http://hbase.apache.org/book/book.html, 2012.

[20] T. White, *Hadoop*. O'Reilly Japan, 2010.

[21] M. Al-Hader and A. Rodzi, "The smart city infrastructure development & monitoring," *Theoretical and Empirical Researches in Urban Management*, vol. 4, no. 2(11), pp. 87–94, May 2009.

[22] eSHIPS, "Study on the interim report based on information use smart house activities (in japanese)," http://www.jipdec.or.jp/dupc/forum/eships/results/doc/eships_fy22_sum_report.pdf.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE