# Using Materialized View as a Service of Scallop4SC for Smart City Application Services

Shintaro YAMAMOTO, Shinsuke MATSUMOTO,
Sachio SAIKI, and Masahide NAKAMURA

Kobe University,
1-1 Rokkodai-cho, Nada-ku, Kobe, Hyogo 657-8501, Japan
{shintaro@ws.cs, shinsuke@cs, sachio@carp, masa-n@cs}
.kobe-u.ac.jp

**Abstract.** Smart city provides various value-added services by collecting large-scale data from houses and infrastructures within a city. However, it takes a long time and man-hour and needs knowledge about big data processing for individual applications to use and process the large-scale raw data directly. To reduce the response time, we use the concept of *materialized view* of database, and materialized view to be as a service. And we propose materialized view to be as as service (*MVaaS*). In our proposition, a developer of an application can efficiently and dynamically use large-scale data from smart city by describing simple data specification without considering distributed processes and materialized views. In this paper, we design an architecture of MVaaS using MapReduce on Hadoop and HBase KVS. And we demonstrate the effectiveness of MVaaS through three case studies. If these services uses raw data, it needs enormous time of calculation and is not realistic.

**Keywords:** large-scale: house log: materialized view: high-speed and efficient data access: MapReduce: KVS: HBase;

## 1 Smart City and Scallop4SC

### 1.1 Smart City and Services

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. A smart city provides various value-added services, named *smart city services*, according to the situation by big data within a city. Promising service fields include energy saving [1], traffic optimization [2], local economic trend analysis [3], entertainment [4], community-based health care [5], disaster control [6] and agricultural support [7].

The size and variety of gathered data become huge in general. Velocity (i.e., freshness) of the data is also important to reflect real-time or latest situations and contexts. Thus, the data for the smart city services is truly *big data*.

Due to the limitation of storage, the conventional applications were storing only necessary data with optimized granularity. Therefore, the gathered data was application-specific, and could not be shared with other applications.

The limitation of the storage is relaxed significantly by cloud computing technologies. Thus, it is now possible to store various kinds of data as they are, and to reuse the raw data for various purposes. We are interested in constructing a data platform to manage the big data for smart city services.

### 1.2 Scallop4SC (Scalable Logging Platform for Smart City)

We have been developing a data platform, called *Scallop4SC*, for smart city services [8][9]. Scallop4SC is specifically designed to manage data from *houses*. The data from houses are essential for various smart city services, since a house is a primary construct of a city. In near future, technolgies of smart homes and smart devices will enable to gather various types of house data.

Scallop4SC basically manages two types of house data: *house log* and *house configuration*. The house log is history of values of any dynamic data measured within a smart home. Typical house log includes power consumption, status of an appliance and room temperature. The house configuration is static meta-data explaining a house. Examples include house address, device ID, floor plan and inhabitant names.

Figure 1 shows the architecture of Scallop4SC. For each house in a smart city, a logger measures various data and records the data as house log. The house log is periodically sent to Scallop4SC via a network. Due to the number of houses and the variety of data, the house log generally forms big data. Thus, Scallop4SC stores the house log using *HBase* NoSQL-DB, deployed on top of *Hadoop* distributed processing. On the other hand, the house configuration is static but structural data. Hence, it is stored in *MySQL* RDB to allow complex queries over houses, devices and people.

*Scallop4SC API* (shown in the middle in Figure 1) provides a basic access method to the stored data. Since Scallop4SC is an application-neutral platform, the API just allows basic queries (see [9]) to retrieve the *raw data*. Application-specific data interpretation and conversion are left for individual applications.

### 1.3 Introducing Materialized View in Scallop4SC

In general, individual applications use the smart city data in different ways. If an application-specific data is derived from much of raw data, the application suffers from expensive data processing and long processing time. This is because the application-specific data conversion is left to each application. If the application repeatedly requires the same data, the application has to repeat the same calculation to the large-scale data, which is quite inefficient.

To cope with this, we introduced *materialized view* in Scallop4SC, as shown in the lower part of Figure 1 [10]. The application-specific data can be considered as a *view*, which looks up the raw data based on a certain query. The materialized view is constructed as a table, which *caches* results of the query in advance.

Note, however, that the raw data in Scallop4SC is very large, and that we cannot use SQL for HBase to construct the view. Therefore, in [10] we developed a Hadoop/MapReduce program for each application, which efficiently converts the raw data into application-specific data. The converted data is stored in an HBase table, which is used as a materialized view by the application. Our experiment showed that the use of materialized

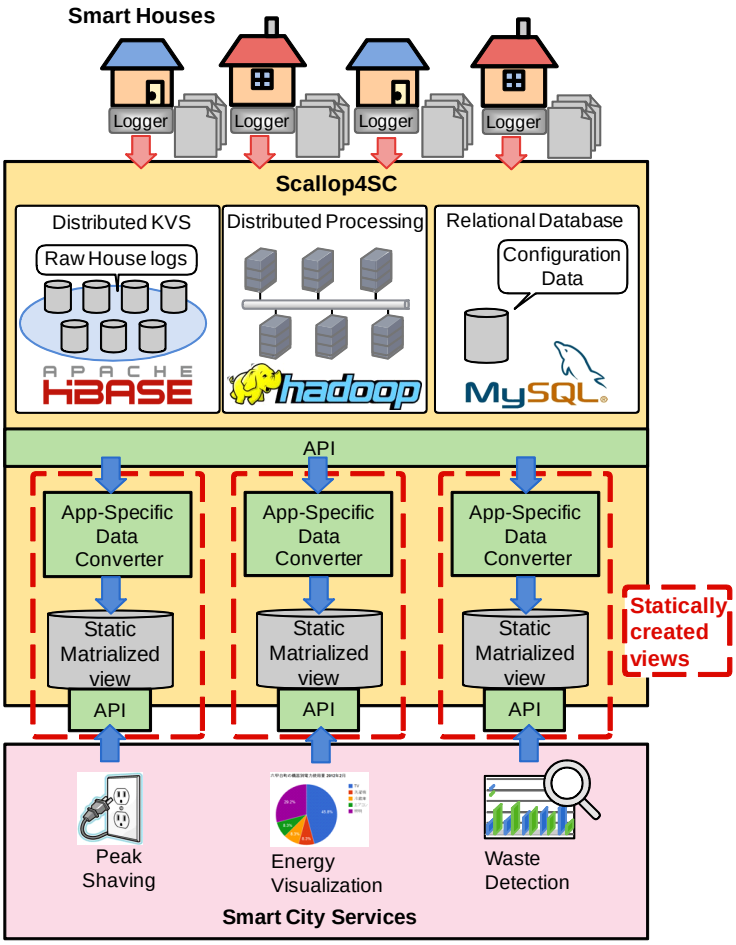


**Fig. 1.** Scallop4SC with static MVs

view significantly reduced computation cost of applications and improved the response time.

A major limitation of the previous research is that the MapReduce program was statically developed and deployed. This means that each application developer has to implement a proprietary data converter by himself. The implementation requires development effort as well as extensive knowledge of HBase and Hadoop/MapReduce. It is also difficult to reuse the existing materialized views for other applications. These are obstacle for rapid creation of new applications.

## 2 MaterializedView as a Service for Large-Scale House Log

### 2.1 Materialized View as a Service (MVaaS)

To overcome the limitation, we propose a new concept of *Materialized View as a Service (MVaaS)*. MVaaS encapsulates the complex creation and management of the materialized views within an abstract cloud service. Although MVaaS can be a general concept for any data platform with big data, this paper concentrates the design and implementation of MVaaS for *house log* in Scallop4SC.

Figure 2 shows the new architecture of Scallop4SC with MVaaS. A developer of a smart city application first gives an order in terms of *data specification*, describing what data should be presented in which representation. MVaaS of Scallop4SC then dynamically creates a materialized view appropriate for the application, from large-scale house log of Scallop4SC. Thus, the application developer can easily create and use own materialized view without knowledge of underlying cloud technologies.

In the following subsections, we explain how MVaaS converts the raw data of house log into application-specific materialized view.

### 2.2 House Log Stored in Scallop4SC

**Table 1.** Raw Data: House Log of Scallop4SC

| **Row Key** | **Column Families** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Data:** | **Info:** | | | | | | |
| (dateTtime.type.home.device) | | date: | time: | device: | house: | unit: | location: | type: |
| 2013-05-28T12:34:56.Energy.cs27.tv01 | 600 | 2013-05-28 | 12:34:56 | tv01 | cs27 | W | living room | Energy |
| 2013-05-28T12:34:56.Device.cs27.tv01 | [power:off] | 2013-05-28 | 12:34:56 | tv01 | cs27 | status | living room | Device |
| 2013-05-28T12:34:56.Environment.cs27.temp3 | 24.0 | 2013-05-28 | 12:34:56 | temp3 | cs27 | Celsius | kitchen | Environment |
| 2013-05-28T12:34:56.Environment.cs27.pcount3 | 3 | 2013-05-28 | 12:34:56 | pcount3 | cs27 | people | living room | Environment |
| 2013-05-28T12:35:00.Device.cs27.tv01 | on() | 2013-05-28 | 12:35:00 | tv01 | cs27 | operation | living room | Device |
| : | : | : | : | : | : | : | : | : |

First of all, we briefly review the data schema of the house log in Scallop4SC (originally proposed in [8]).

Table 1 shows an example of house logs obtained in our laboratory. To achieve both *scalability* for data size and *flexibility* for variety of data type, Scallop4SC stores the house log in the HBase key value store. Every house log is stored simply as a pair of key (**Row Key**) and value (**Data**). To store a variety of data, the data column does not have rigorous schema. Instead, each data is explained by a meta-data (**Info**), comprising standard attributes for any house log.

The attributes include *date* and *time* (when the log is recorded), *device* (from what the log is acquired), *house* (where in a smart city the log is obtained), *unit* (what unit should be used), *location* (where in the house the log is obtained) and *type* (for what the log is). Details of device, house and location are defined in an external database of house configuration in MySQL (see Figure 2). A row key is constructed as a concatenation of date, time, type, home and device. An application can *get* a single data (i.e., row) by specifying a row key. An application can also *scan* the table to retrieve multiple rows

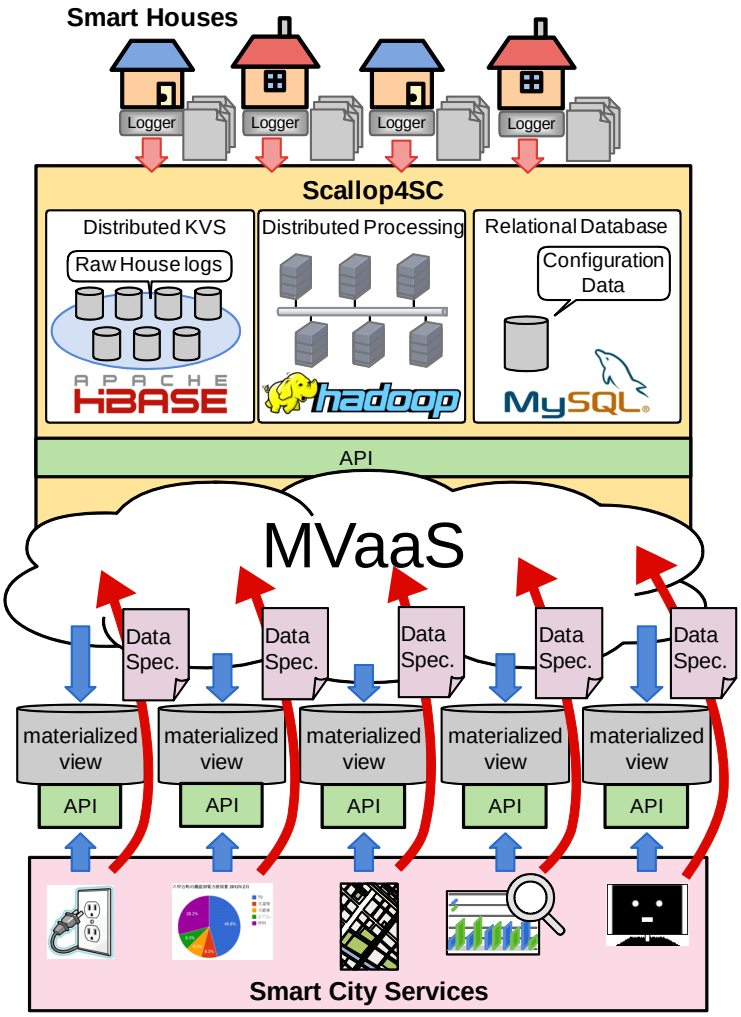


**Fig. 2.** Extended Scallop4SC

by *prefix match* over row keys. Note that complex queries with SQL cannot be used to search data, since HBase is a NoSQL database.

For example, the first row in Table 1 shows that the log is taken at 12:34:56 on May 28, 2013, and that the house ID is cs27 and log type is Energy, and that the deviceID is tv01. The value of power consumption is 600 W. Similarly, the second row shows a status of tv01 where power is off.

We assume that these logs are used as *raw data* by various smart city services and applications.

### 2.3 Converting Raw Data into Materialized View

In order to create materialized views in response to applications, it needs converting data from raw data and importing the materialized view to create the materialized view. Here is how to create materialized view from raw data to meet the requirements of a developer of an application.

First, a developer of an application describes a data specification and gives it to MVaaS. Next, MVaaS generates MapReduce converter to convert raw data into data suitable for the application, depending on data specification. Finally, MVaaS practices MapReduce Converter on Hadoop, and stores created data into a materialized view.

For example, in a case to create a view in RDB collecting value calculated total power consumption of each device by one day, we use a SQL command as follows:

```
1 CREATE VIEW view_name AS
2 SELECT date,device,SUM(Data) FROM houselog
3 WHERE type=Energy,unit=W
4 GROUP BY date,device;
```

From the above example, we can see that typical views consist of a phase for filtering data (a part of WHERE), a phase for grouping data (a part of GROUP BY), and a phase for aggregating data (a part of SUM). We want to realize similar commands for house logs on HBase to be NoSQL.

Figure 3 shows flowchart of creating a materialized view collecting value calculated total power consumption of each device by one day. First, raw data are narrowed to refer to power consumption (Filter), then grouped by data and device (Grouping), next aggregate sum total against each grouped data, finally imported into a materialized view.

## 3 Case Study

We demonstrate the effectiveness of MVaaS through three case studies. If these services uses raw data, it needs enormous time of calculation and is not realistic. But by using MVaaS, a developer of an application can use materialized views only to describe simple data specifications.

### 3.1 Power Consumption Visualization Service

The service collects data of power consumption from smart house and visualizes the use of energy from various viewpoints (e.g. houses, towns, devices, current power consumption, passage of past power consumption, etc.). This service is intended to raise user's awareness of energy saving by intuitively showing the current and past usage of energy.

Here, we will examine about one of simple power consumption visualization services as a case study. The service is to visualize power consumption for each device by one hour in one house.

Thus, this application needs logs that type is Energy, unit is W, house is a house (cs27), row key consists time and device, and value is total sum of power consumption, like Table 2.

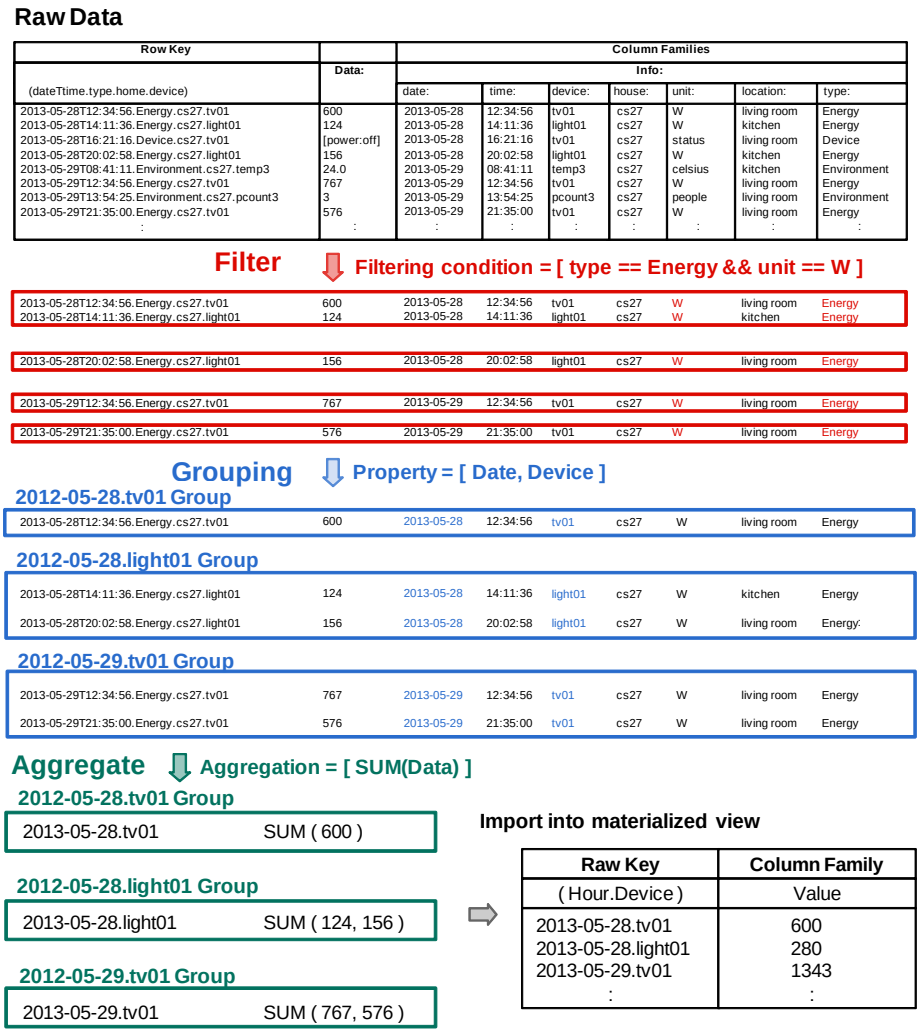


**Fig. 3.** Flowchart of converting raw data into materialized view

### 3.2 Wasteful Energy Detection Service

This service automatically detects wasteful electricity and notifies users using power consumption data and sensors data in a smart house. Furthermore, a user can review the detected waste occurred in the past.

Here, we will examine about one of wasteful energy detection service as a case study. The service is to visualize power consumption of an air conditioner with temperature in the room.

Thus, this application needs two logs. First one is logs about power consumption of an air conditioner. Another is logs about temperature in the room. Power consumption

**Table 2.** Materialized view of Power Consumption Visualization Service

| Row Key | Column Families |
|---|---|
| (dateTtime.device) | **Data:** |
| 2013-05-28T10.tv01 | 784415 |
| 2013-05-28T10.light01 | 1446 |
| 2013-05-28T10.light02 | 21321 |
| 2013-05-29T10.aircon01 | 54889 |
| 2013-05-28T11.tv01 | 51247 |
| 2013-05-28T11.light01 | 7742 |
| 2013-05-28T11.light02 | 3288 |
| 2013-05-28T11.aircon01 | 65774 |
| : | : |

logs are type is Energy, unit is W, house is a house (cs27), location is a room (living room), device is air conditioner (aircon01), row key consists time, and value is total sum of power consumption, like Table 3(a). Temperature logs are type is Environment, unit is Celsius, house is a house (cs27), location is a room (living room), device is temperature sensor (temp1), row key consists time, and value is average of temperature, like Table 3(b).

**Table 3.** Materialized view of Wasteful Energy Detection Service

(a) Power consumption of the air conditioner

| Row Key | Column Families |
|---|---|
| (dateTtime) | **Data:** |
| 2013-05-28T10 | 75426 |
| 2013-05-28T11 | 11548 |
| 2013-05-28T12 | 21859 |
| : | : |

(b) Temperature

| Row Key | Column Families |
|---|---|
| (hour) | **Data:** |
| 2013-05-28T10 | 27.0 |
| 2013-05-28T11 | 23.4 |
| 2013-05-28T12 | 28.2 |
| : | : |

### 3.3 Diagnosis and Improvement of Lifestyle Support Service

This service urges users to improve their lifestyles and advises users on health with analyzing the daily device logs. For example, by using light log in user's house the service advises on time of sleep.

Here, we will examine about one of simple power consumption visualization services as a case study. The service is to visualize sleep period of time from logs. For

example, the period of time in which all lights are off with pelple in the room is expected to be a sleep period of time.

Thus, this application needs two logs. First one is logs about status of devices. Another is logs about presence of people in the room. Status of devices logs are type is Device, unit is status, house is a house (cs27), row key consists date, room and device , and value is time in which devices is off, like Table 4(a). Presence of people in the room logs are type is Environment, unit is peolple, house is a house (cs27), row key consists date and room, and value is time in which people exist, like Table 4(b).

**Table 4.** Materialized view of Diagnosis and Improvement of Lifestyle Support Service

(a) Status of lights

| **Row Key** | **Column Families** |
|---|---|
| (date.location.device) | **Data:** |
| 2013-05-28.living room.tv01 | 21:12:44, 21:13:43, 21:14:57,... |
| 2013-05-28.bedroom.light01 | 23:34:56, 23:35:56, 23:36:57,... |
| 2013-05-29.living room.tv01 | 20:22:16, 20:23:14, 20:24:17,... |
| 2013-05-29.bedroom.light01 | 21:46:23, 21:47:24, 21:48:50,... |
| 2013-05-30.living room.tv01 | 23:56:21, 23:57:21, 23:58:24,... |
| 2013-05-30.bedroom.light01 | 23:57:23, 23:58:21, 23:59:20,... |
| : | : |

(b) Period of time of people in the room

| **Row Key** | **Column Families** |
|---|---|
| (date.location) | **Data:** |
| 2013-05-28.living room | 18:12:44, 18:13:43, 18:14:57,... |
| 2013-05-28.bedroom | 22:34:56, 22:35:56, 22:36:57,... |
| 2013-05-29.living room | 19:22:16, 19:23:14, 19:24:17,... |
| 2013-05-29.bedroom | 21:36:23, 21:37:24, 21:38:50,... |
| 2013-05-30.living room | 16:56:21, 16:57:21, 16:58:24,... |
| 2013-05-30.bedroom | 20:57:23, 20:58:21, 20:59:20,... |
| : | : |

In this case study, we thought the case using two materialized views. But this two materialized views can be combined into one. If we combine materialized views, number of data will decrease and it will be easy to visualize, whereas extensibility and reusability will not maintain.

## 4 Conclusion

In this paper, we proposed an architecture of MVaaS that allows various applications to efficiently and dynamically use large-scale data. The method is specifically applied to a data platform, Scallop4SC, for the large-scale smart city data. In the MVaaS, a developer of an application can efficiently and dynamically access large-scale smart city data only by describing data specifications for application. And we demonstrated the effectiveness of MVaaS through three case studies. If these services uses raw data, it needs enormous time of calculation and is not realistic.

Our future works include implement of MVaaS, as well as an efficient operation method of MapReduce converter.

## 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)].

## References

1. Massoud Amin, S., Wollenberg, B.: Toward a smart grid: power delivery for the 21st century. Power and Energy Magazine, IEEE **3**(5) (2005) 34–41
2. Brockfeld, E., Barlovic, R., Schadschneider, A., Schreckenberg, M.: Optimizing traffic lights in a cellular automaton model for city traffic. Phys. Rev. E **64** (2001) 056132
3. IBM: Apply new analytic tools to reveal new opportunities. `http://wwiw.bm.com/smarterplanet/nz/en/business_analytics/article/it_business_intelligence.html`
4. Celino, I., Contessa, S., Corubolo, M., Dell'Aglio, D., Valle, E.D., Fumeo, S., Krüger, T.: Urbanmatch - linking and improving smart cities data. In: Linked Data on the Web (LDOW2012). (2012)
5. IBM: eHealth and collaboration - collaborative care and wellness. `http://www.ibm.com/smarterplanet/nz/en/healthcare_solutions/nextsteps/solution/X056151Y25842W14.html`
6. Hu, C., Chen, N.: Geospatial sensor web for smart disaster emergency processing. In: International Conference on GeoInformatics (Geoinformatics2011). (2011) 1–5
7. Cho, Y., Moon, J., Yoe, H.: A context-aware service model based on workflows for u-agriculture. In: International Conference on Computational Science and Its Applications (ICCSA2010). Volume 6018. (2010) 258–268
8. Yamamoto, S., Matumoto, S., Nakamura, M.: Using cloud technologies for large-scale house data in smart city. In: In International Conference on Cloud Computing Technology and Science (CloudCom2012). (December 2012) 141–148
9. Takahashi, K., Yamamoto, S., Okushi, A., Matsumoto, S., Nakamura, M.: Design and implementation of service api for large-scale house log in smart city cloud. In: In International Workshop on Cloud Computing for Internet of Things (IoTCloud2012). (December 2012) 815–820
10. ISE, Y., YAMAMOTO, S., MATSUMOTO, S., NAKAMURA, M. In: International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2013). (2012 (to appear))