Processing IoT Data with Cloud Computing for Smart Cities

Hae Sun Jung, Chul Sang Yoon, Yong Woo Lee
School of Electrical & Computer Engineering the University of Seoul
South Korea
banyasun@uos.ac.kr, csyoon2014@uos.ac.kr, ywlee@uos.ac.kr

Jong Won Park, Chang Ho Yun
The Smart-city Consortium
South Korea
comics77@gmail.com
touch011@gmail.com

ABSTRACT: *A smart city requires the intelligent management of infrastructure like the Internet of Things (IoT) devices in order to provide smart services that improve the quality of human life. To obtain the information needed to implement smart city services, stream reasoning is used to intelligently process the big data stream constantly generated from IoT devices. However, there are constraints associated with the real-time processing of large streams of big data from the smart city infrastructure. In this paper, we propose a stream reasoning system model for the smart city application, which was implemented using real-time big data processing technology in the smart city middleware. We use Apache Kafka, a message processing system, and Apache Storm, a real-time distributed processing system, to overcome the constraints associated with real-time processing. We evaluate the performance of our system implementation by measuring the throughput per second and the maximum capacity of the experimental system.*

Keywords: Stream Reasoning, Smart City, Smart City Middl, are; Apache Kafka, Apache Storm, Real-time Distributed Processing, Big Data.

Received: 1 May 2017, Revised 6 June 2017, Accepted 12 June 2017

1. Introduction

A smart city is envisioned as a futuristic city that is pervaded by information and communication technologies (ICT) that provide smart services to the citizens anytime and anywhere through smart devices, resulting in an improved quality of urban life [1].

A study by Gartner Inc. estimates that 1.6 billion connected things will be used in smart cities in 2016, an increase of 39 percent from 2015 [2]. Internet of Things (IoT) has enabled the mapping of real world objects to Physical Things and virtual objects to Virtual Things and connected the two via the Internet. This represents the future Internet infrastructure that provides a variety of services through the linkage of objects, data, and people across physical and virtual spaces [3].

To implement smart services in a smart city setup, it is necessary to analyze the big data transferred from smart IoT devices to support situation-aware and intelligent decision making. Stream reasoning technology is necessary to recognize the situation and handle the big data generated constantly in real-time. This is a state-of-the-art technology defined as the "logical reasoning in real time on gigantic and inevitably noisy data streams in order to support the decision process of extremely large numbers of concurrent users" [4].

In our previous study, we proposed a three-tier paradigm to efficiently manage a smart city and implemented a smart city system called UTOPIA. UTOPIA is composed of a smart city infrastructure tier, a smart city middleware tier, and smart city portal tiers. Among these, the smart city middleware is the core tier which plays the role of a brain or soul of a person. Hence, the prototype is termed Smart and Outstanding Ubiquitous Lounge (SOUL). SOUL uses a stream reasoning technology to provide services such as fire accident management which require real-time processing and complex reasoning. Here, the stream data used by SOUL changes rapidly and is huge in volume. It is a huge challenge to continuously perform the stream reasoning in real-time for large volumes of rapidly changing big data streams.

In this paper, we propose a stream reasoning system model and implement the system using real-time big data processing in the smart city middleware tier. The stream reasoning system is implemented using the mass-messaging system, Apache Kafka [5], and the real-time distributed processing system, Apache Storm [6], to overcome constraints related to real-time processing. We evaluate the performance of our implementation by measuring the throughput per second and the maximum capacity of the experimental system. Research contribution of this paper is as follows.

1) In this study, we propose a stream reasoning system for implementation in the smart city middleware tier that can handle rapidly changing big data stream in real-time. 2) Our approach is based on real-time distributed processing techniques that can handle large data streams and provide scalability. 3) Finally, we evaluate the performance of the developed system and demonstrate its usefulness.

The rest of this paper is organized as follows. Section 2 describes the studies related to stream reasoning using cloud computing in the smart city IoT environment. Section 3 describes the smart city systems and middleware as previously investigated in our previous study. Section 4 describes the process of stream reasoning in the smart city system and Section 5 evaluates the performance of the implemented system. Section 6 concludes this study.

2. Related Works

It is hard to find stream reasoning methods that can intelligently process the large number of big data streams that are continuously generated from the many IoT devices in a smart city. However, various methods for continuously processing a stream of big data in real-time have been proposed.

Apache Storm, Apache S4 [7], and Apache Samza [8] have been developed for real-time processing. Apache Spark [9] is a fast and generic engine for large-scale data processing with micro batch and in-memory processing. Streams reasoning studies on cloud computing clusters using a distributed processing platform for big data are in progress.

J. Hoeksema et al. [10] propose a parallel method utilizing a computer cluster and the Yahoo S4 framework to provide a high-throughput for stream reasoning. They also present a set of low-level predicates operating on a stream and an encoding of RDFS reasoning and C-SPARQL query answering using these predicates.

X. Chen et al. [11] discuss a large-scale real-time semantic processing framework and implement an elastic distributed streaming engine for IOT applications. The proposed engine efficiently captures and models different scenarios for all kinds of IOT applications based on the distributed computing platform SPARK. Through experimental evaluation, they demonstrate that the proposed system can scale for a large number of sensor streams and different types of IoT applications.

A.I. Maarala et al. [12] develop a semantic reasoning system by applying state-of-the-art semantic technologies to an IoT environment. They evaluate the scalability and real-time response of different reasoning approaches, including single reasoners, distributed reasoners, mobile reasoners, and a hybrid of these reasoners. They also evaluate different data aggregation strategies for integrating the distributed IoT data in order to implement the reasoning processes.

The above-mentioned studies increase the processing speed of stream reasoning simply by using real-time cloud computing technologies. However, there are still some limitations because the advancements are not implemented on the middleware. The method proposed in this study is capable of handling big data streams from heterogeneous IoT devices. Based on the smart city middleware, real-time cloud as well as Storm can be applied, depending on the application characteristics. Our approach also guarantees the reliability and scalability of data transfer by using Apache Kafka, a message processing system.

3. Smart City System and Middleware

Smart city refers to a future city that combines ICT in existing cities to manage urban facilities for sustainable urban development and to provide intelligent services to citizens.

In our previous study [13], we proposed and implemented a three-tier architecture termed UTOPIA that is capable of intelligently managing smart city infrastructure to offer various smart applications to citizens.

Fig. 1 shows the architecture of UTOPIA. The three tiers of the UTOPIA design interlock with each other and manage the smart city effectively. UTOPIA is composed of a smart city infrastructure tier, a smart city middleware tier, and smart city portal tiers. Among these, the smart city middleware is the core tier that plays the role of a brain or soul of a person and has been termed SOUL.

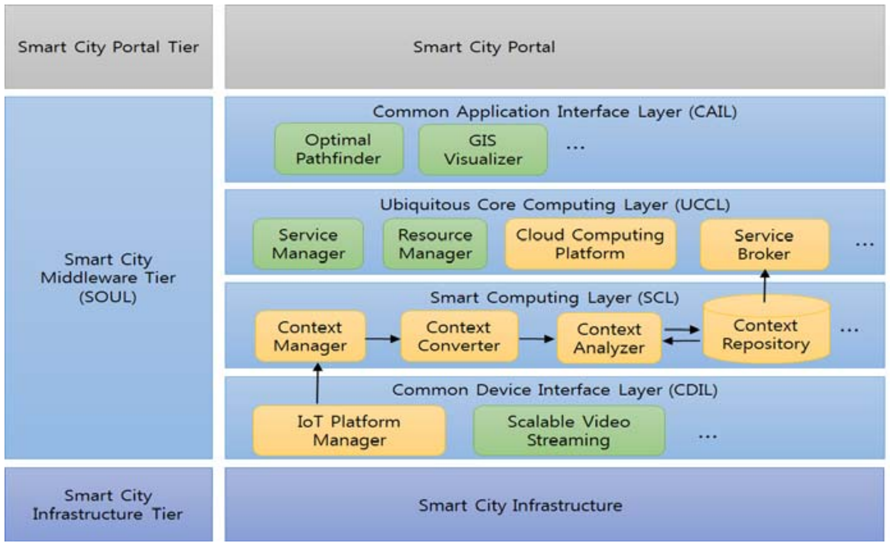

Figure 1. Stream Reasoning System in SOUL

SOUL processes the data generated from the smart city infrastructure tier intelligently and provides services through the smart city portal tier. It consists of a Common Device Interface Layer, Smart Computing Layer, Ubiquitous Core Computing Layer, and Common Application Interface Layer. The Common Device Interface Layer receives data from sensors in the infrastructure area and transfers it to the Smart Computing Layer. It supports various types of sensors and communication protocols and provides a common interface for heterogeneous Ubiquitous Sensor Network (USN).

The Smart Computing Layer is a critical layer of the stream reasoning process and consists of a Context Converter, Context Analyzer, Context Repository and Context Manager. The Smart Computing Layer converts the raw data into RDF/ OWL format, interprets it using predefined rules, stores the inferred data, and supports continuous query on the context. The Context

Converter receives raw sensor data from the Common Device Interface Layer, performs filtering based on the filter policy, and converts it into a RDF/OWL format that follows the context ontology model and is compatible for stream reasoning. The Context Analyzer performs stream reasoning by inferring RDF/OWL context instances according to predefined rules. It supports distributed processing by utilizing cloud computing in the Ubiquitous Core Computing Layer to improve the performance of the stream reasoning. Cloud computing is implemented based on OpenNebula [14] and Haizea [15]. The Context Repository stores a context ontology model, the context instances, the rules, and an inferred context. The Context Manager interfaces with the Ubiquitous Core Computing Layer, interprets a query from the Service Broker in the Ubiquitous Core Computing Layer and performs queries. The context ontology model and the rule file are pre-loaded into memory when the system is initialized and the context instance is loaded into memory at run time.

4. Stream Reasoning

In this paper, we study a distributed system that can support high-volume, real-time processing of the continuous query obtained from stream reasoning of the big data stream. We use a large distributed messaging system, Apache Kafka, and a distributed processing technology from real-time cloud computing, Apache Storm, that are well-suited for stream reasoning systems. Kafka is used to receive the real-time transmission of a large-capacity data stream in the data acquisition component. It is specialized messaging system that is suitable for real-time data stream transmission, in order to handle large messages. It models the point to be operated at as a cluster for scale-out and exhibits high availability by easily handling the large amounts of data generated in the smart city. In addition, Kafka meets the high-volume, real-time, and low transmission delay requirements for stream reasoning. It uses a lightweight protocol as compared to existing messaging protocols such as AMQP protocol and JMS API and is therefore suitable for stream data transmission between IoT devices and the SOUL.

Storm is a real-time distributed analysis system that can define the stream to read data and includes processing logic to handle the stream using the topology. There are two types of components: the spout and the bolt. The spout generates the stream from a source external to the topology. The bolt performs all processing operations of the topology and converts it into the necessary stream in the next step. It may also be necessary to gain the necessary steps to functionally separate the data stream processing tasks that define multiple bolts and perform each vault. The structure and processes of Storm is very efficient for real-time processing of stream reasoning and the inference mechanisms.

Fig. 2 shows the process flow of the stream reasoning system which is implemented in SOUL.

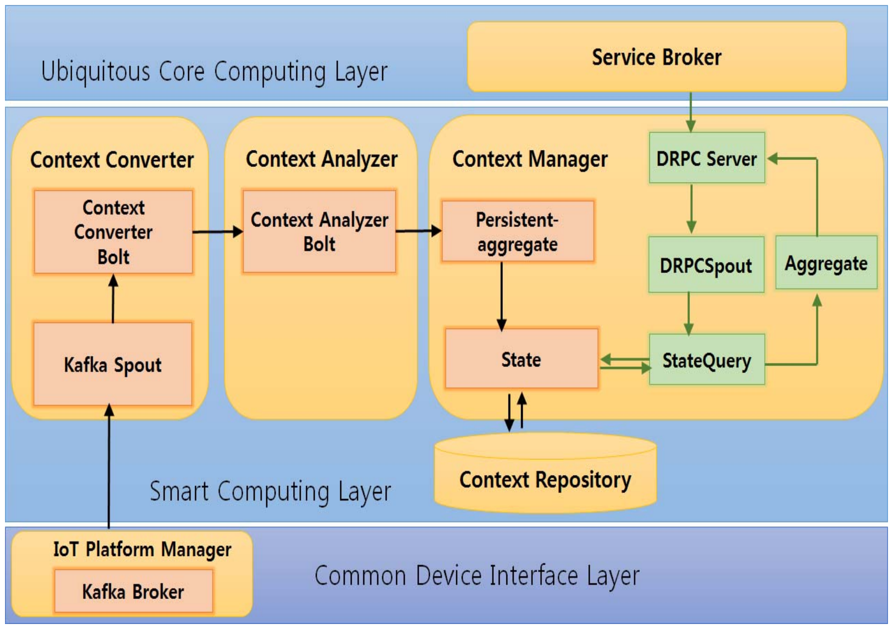

Figure 2. Stream Reasoning System in SOUL

A Kafka Broker is deployed in the IoT Platform Manager on the Common Device Interface Layer to transfer the data into the Context Converter on the Smart Computing Layer. The Context Converter consists of a Kafka Spout and a Context Converter Bolt. The Kafka Spout implements a Kafka consumer in Storm and accepts the data stream transmitted from the IoT Platform Manager. The Context Converter Bolt converts the OWL ontology instances using the OWL ontology model in order to infer the received data stream. Fig. 3 shows the OWL domain ontology defined in this study.

The Context Analyzer performs the inference operation using the OWL ontology instances and user-defined rules, including the Context Analyzer Bolt. The Context Analyzer Bolt performs OWL reasoning using the Apache Jena Framework [16], while the OWL Reasoner uses Pellet [17]. User-defined rules are configured to receive user input in order to enable dynamic change during run-time. The Context Manager transmits the resulting inference from the Context Analyzer to the Context Repository and interfaces with the Service Broker on the Ubiquitous Core Computing Layer. Persistent aggregation on each node is responsible for passing the results of the inference into State.

State refers to the memory storage that stores the state required to process the continuous query that is passed in Distributed Remote Procedure Call (DRPC) [18]. In order to perform a continuous query on the results of the inference in the continuous stream reasoning process, the status should be stored for each moment. The window method is used to maintain and query the status of each stream for a given moment. DRPC techniques are used to implement the continuous queries on Storm.

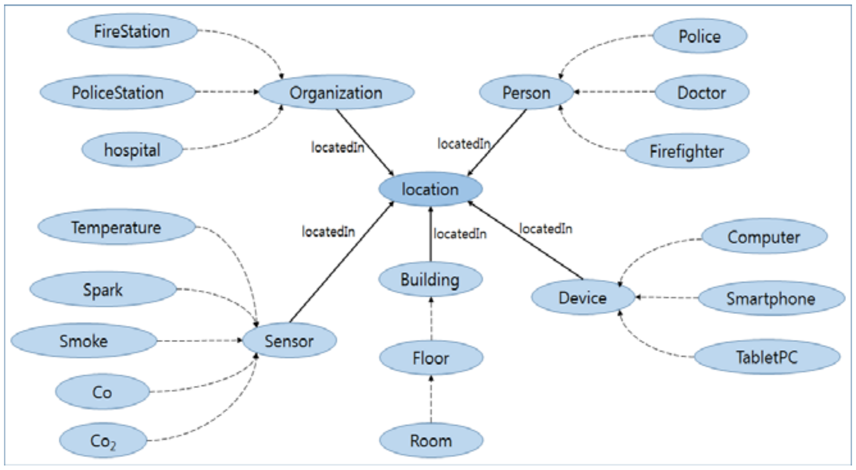

Figure 3. The OWL domain ontology model

The DRPCServer performs RPC in the Storm topology in the role of a server that receives a command from the Service Broker and returns the results to the Service Broker. DRPCSpout passes the query received from the DRPCServer into StateQuery. StateQuery queries the current state value of the window in the State using the transferred query and transmits the results to the Aggregate. The Aggregate transmits the received results to the DRPCServer, and finally, the DRPCServer delivers the results of continuous queries to the Service Broker. The Context Repository is the repository that permanently stores the state value that was loaded into the memory of the State and the inferred ontology instance for each moment. The Service Broker is a part of the Ubiquitous Core Computing Layer and interfaces with the Smart Processing Layer. In order to obtain the status information required by the smart city application, the Service Broker requests a continuous query and receives the results.

5. Performance Evaluation

So far we have described the process of stream reasoning in the smart city system.

In this section, we evaluate the performance of the proposed system. We use clusters to perform a stream reasoning in a distributed environment; fourteen of them were used in the experiment. The CPU specifications are: Intel core i5 760 2.8 Ghz, DDR3 8 GB RAM, 500 GB 7200 rpm hard disk, 1 Gbps Ethernet network, CentOS 6.7 operating system, and a 64-bit server. Kafka Broker and Zookeeper are installed on three nodes, Nimbus is installed on one node, and Storm Supervisor is installed on ten nodes in the cluster. A stream data generator is used to generate a large data stream. In this experiment, we generate 1,000,000 sensor data streams and measure the processing time and throughput per second while the transmission rate of the data generated from the data generator changes.

It is difficult to evaluate stream reasoning using the same method as a general batch processing experiment because stream reasoning is a real-time process. In order to evaluate the stream reasoning system, we measure the sensor stream data transfer amount per second with respect to the sensor stream data transfer time for a fixed number of sensor stream data. Based on the measured sensor stream data transfer amount per second, the total processing time and the total throughput per second are measured. Fig. 4, 5, and 6 show the experimental results. Fig. 4 shows the transmission amount of the sensor stream data per second when transferring the entire 1,000,000 sensor data into a stream. The stream transfer amount per second and the stream transfer time are found to be inversely related. It is the inverse relationship shrinking of the stream transfer amount per second, according to increasing the stream transfer time because of sending the data in a fixed number according to increasing the transfer time.

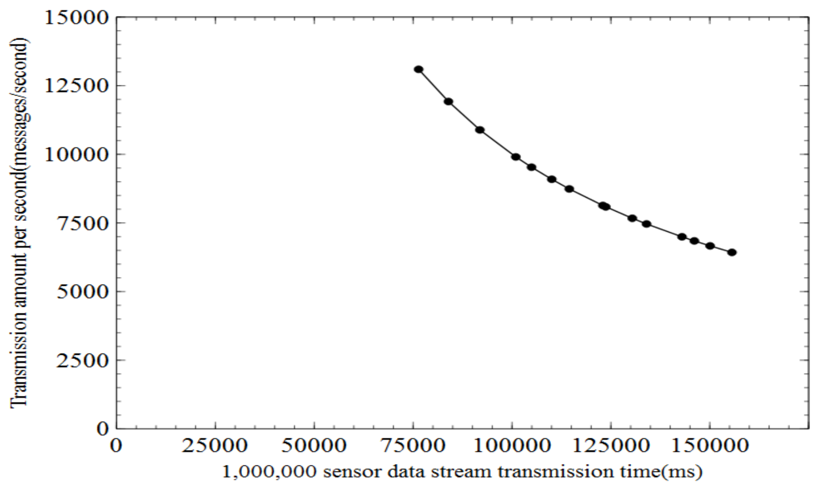

Figure 4. The transmission amount of the sensor stream data

Fig. 5 shows the total processing time with respect to the sensor stream data transmission amount per second prior to operation in the Y-axis of Fig. 4. Processing time is least when the sensor stream data transmission is 10,000 per second, and subsequently the processing time is minimal but growing.

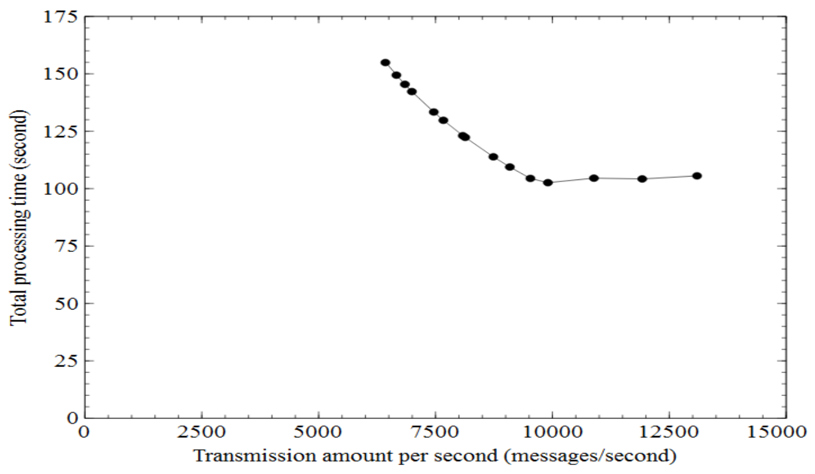

Figure 5. Total processing time

Fig. 6 shows the total throughput per second with respect to the transmission rate of the sensor stream data. A transmission rate of 10,000 messages/second exhibits the highest overall throughput per second, and thereafter it can be seen that the throughput per second decreases. From Fig. 5 and 6, we can conclude that the maximum processing capacity of the experimental system is approximately 10,000 stream data per second.

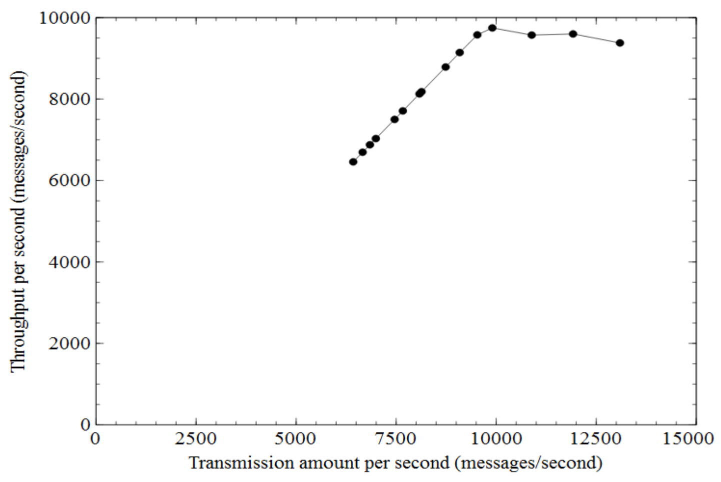

Figure 6. Total Throughput

6. Conclusions

This paper presents a stream reasoning system model for the smart city application, which was implemented using real-time big data processing technologies in the smart city middleware. Stream reasoning systems for the smart city application must possess high scalability, high throughput, and low processing delay. In order to satisfy these requirements, the real-time big data processing technology was implemented using a combination of Apache Kafka and Apache Storm. The stream reasoning system was implemented using these technologies on the smart city middleware SOUL. In order to perform a continuous query on the results of the inference from the continuous stream reasoning process, the status should be stored for each moment. In order to query the status for each moment in the stream, the state was maintained using the window method. Storm DRPC techniques were used to implement the continuous query in Storm. In order to evaluate the model and measure the maximum performance in the experimental system, we measured the total processing time and the throughput per second. In an actual system implementation, this will help to determine the amount of computing power of a cluster or cloud system. In the future, we will experiment with measuring the latency time for a continuous query as well as the data processing of the stream reasoning system.

Acknowledgment

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Korea Ministry of Science, ICT & Future Planning (2015R1C1A1A02036461). This study was also supported by the Smart City Consortium (10561) and Seoul Grid Center.

References

[1] Lee, Y. W. (2016). Smart-city, European Union Parliament Seminar, May 2013, [Online]. Available from: http://www.europarl.europa.eu/document/activities/cont/201305/20130514ATT66084/20130514ATT66084EN.pdf 2016.06.01

[2] Meulen, R., Woods, V (2016). Gartner Says Smart Cities Will Use 1.6 Billion Connected Things in 2016, Gartner Newsroom, Dec. 2015, [Online]. Available from: http://www.gartner.com/newsroom/id/3175418 2016.06.01

[3] Rose, K. . Eldridge, S.,. Chapin, L (2015).The Internet of Things: An Overview," Report of Internet Society, Oct. 2015.

[4] Valle, E. D., Ceri, S., Harmelen, F. v., Fensel, D (2009). It's a Streaming World! Reasoning upon Rapidly Changing Information,

IEEE Intelligent Systems, vol. 24, p. 83–89, Nov. 2009.

[5] Kreps, J. Narkhede,N. Rao, J. (2011). Kafka: a Distributed Messaging System for Log Processing, *In*: Proc. NetDB workshop (NetDB 2011), Jun. 2011.

[6] Apache Storm, [Online]. Available from: http://storm.apache.org/ 2014. 05.05

[7] Apache S4, [Online]. Available from: http://incubator.apache.org/s4/ 2014.03.02

[8] Apache Samza, [Online]. Available from: http://samza.incubator.apache.org/ 2016.01.15

[9] Apache Spark, [Online]. Available from: http:// http://spark.apache.org/ 2014.10.20

[10] Hoeksema J., Kotoulas, S (2011). High-performance Distributed Stream Reasoning using S4, *In*: Proc. The First International Workshop on Ordering and Reasoning (OrdRing 2011), p. 1-12, Oct. 2011.

[11] X. Chen, H. Chen, N. Zhang, J. Huang, and W. Zhang, (2015). Large-Scale Real-Time Semantic Processing Framework for Internet of Things, *International Journal of Distributed Sensor Networks*, 11 (10) 1-11, Oct. 2015.

[12] Maarala, A. I. Su, X. Riekki, J.(2016). Semantic Reasoning for Context-aware Internet of Things Applications, *IEEE Internet of Things Journal*, 3 (4) 1-13, Aug. .

[13] Jung, H. S. . Jeong, C. S. Lee,Y. W Hong, P. D. (2009). An Intelligent Ubiquitous Middleware for U-City: SmartUM, *Journal of Information Science and Engineering*, 25, p. 375-388, Mar. 2009.

[14] OpenNebula Homepage, [online]. Available from: http://opennebula.org/. 2015.12.01

[15] Sotomayor, B. Montero, .R. S., Llorente, I. M.Foster, I. (2009). Virtual Infrastructure Management in Private and Hybrid Clouds, *IEEE Internet Computing*, 13 (5) 14-22.

[16] Apache Jena, [online]. Available from: https://jena.apache.org/ 2014.09. 30

[17] Sirin, E.Parsia, B. Grau, B. C. Kalyanpur, A. Katz.Y. (2007). Pellet: A practical OWL-DL reasoner, Web Semantics: Science, Services and Agents on the World Wide Web, 5 (2) 51-53, Jun. 2007.

[18] Storm Distributed RPC, [online]. Available from: http://storm.apache.org/releases/current/Distributed-RPC.html 2015.03.05