Elastic Streaming Semantic Engine for Internet of Things

Xi Chen
Department of Computer Science
Zhejiang University
Hangzhou, China
xichen@zju.edu.cn

Huajun Chen
Department of Computer Science
Zhejiang University
Hangzhou, China
huajunsir@zju.edu.cn

Jue Huang
Department of Computer Science
Zhejiang University
Hangzhou, China
juehuang@zju.edu.cn

***Abstract*—With the rapid advances in wireless sensor data collection and communication, heterogeneous and real-time IOT(Internet of Things) data is increasing explosively, which makes it possible to develop and deploy ubiquitous IOT-based applications in various aspects of our daily life. However, the characteristics of IOT sensor data, including heterogeneity, variety, volume and real-time, pose a series of challenges to effectively organize, publish, and process the sensor data. The Semantic Web technologies are viewed as a key for the development of IOT. While most of the existing efforts are mainly focused on the modeling, annotation, and representation of IOT data, there has been little work focusing on the background processing of large-scale streaming IOT data. In the paper, we present a general semantic processing framework and implement an elastic distributed streaming engine for IOT applications. The proposed engine efficiently capture and model different scenarios for all kinds of IOT applications based on popular large-scale distributed computing platform SPARK. Based on the engine, a typical use case on home environment monitoring is given to illustrate the efficiency of our engine. The results show that our system can scale for large number of sensor streams with different types of IOT applications.**

***Keywords*-IOT, Semantic Web, Streaming, Big Data, Wireless Sensor Network.**

I. Introduction

With the rapid advances in wireless sensor data collection and communication, increased number of IOT data is increasing explosively, which builds many massive wireless sensor networks(WSN). It is predicted that within the next decade billions of devices (Cisco predicts that the number of the Internet connected devices will be around 50 Billion by 2020) [1] will generate myriad of real world data for many applications and services in a variety of areas such as smart grids, smart homes, e-health, automotive, transport and environmental monitoring. Such a stunning massive and widespread data can help us to observe the surroundings, learn patterns and have a better understanding of the world, which will construct a more intelligent world.

To make full use of the senses to implement a deeper web intelligence, a natural next step would be to unify and process IOT data by existing mature web infrastructure and protocols. Presently, much efforts have been put on this area from Internet of Things(IOT) to Web. The W3C founded the Web of Things Community Group aiming at accelerating the adoption of Web technologies such as semantic technologies as a basic for enabling services for the combination of IOT with rich descriptions of web data and the context in which they are used. The industry also initiates the standards oneM2M[1], whose goal is to develop technical specification which is used to address the need for a common IOT Service Layer by reusing existing web standards and protocols, including RDF, HTTP, Restful.

However, the characteristics of IOT data, including heterogeneity, variety, volume and real-time, pose a series of challenges to effectively organize, publish, and process the sensor data. The Semantic Web technologies are viewed as a key for the development of IOT. Figure 1 shows the generic functional model of oneM2M for supporting semantics in the specification of oneM2M study on abstraction and semantics enablement [2]. In specific, it serves as the following several purposes: First, the abstraction and semantics layer provide us with a good way to resolve the problems of interoperability and integration within this heterogeneous world of IOT data by defining and reusing some standard semantic concepts. Then, the Semantic Web provides a seamless interface to facilitate the interactions of IOT data and other existing web of data such as Linked Data [3], DBpedia [4], LinkedGeodata [5], various kinds of data from Web Services. At last, the service layer provides an interface for various IOT applications by semantic processing technologies, including semantic mash-up, query, reasoning and so on.

Currently, most of the existing works aim at the annotation and definition of various WSNs by providing corresponding description ontology. For example, ontologies such as the W3Cs Semantic Sensor Network(SSN) ontology have been developed, offering a number of constructs to formally describe not only the sensor resources but also the sensor observation and measurement data [6]. However, there has been little work focusing on the background processing of large-scale streaming IOT data. In the paper, we present a general semantic processing framework for IOT applications. The proposed framework efficiently capture and model different scenarios for various IOT applications. We have implemented an elastic streaming engine based on

[1]www.onem2m.org

popular large-scale distributed computing platform SPARK [7]. Based on the engine, a typical use case on home environment monitoring is given to illustrate the efficiency of our engine. The results show that our system can scale for large number of sensor streams with different types of IOT applications.

The remaining of this paper is organized as follows. Section 2 outlines related work. In Section 3, we introduce the general framework and processing engine for IOT applications. Section 4 describes a typical use case in home environment monitoring. Section 5 presents our experiments and results. Finally, we conclude the work in Section 6.

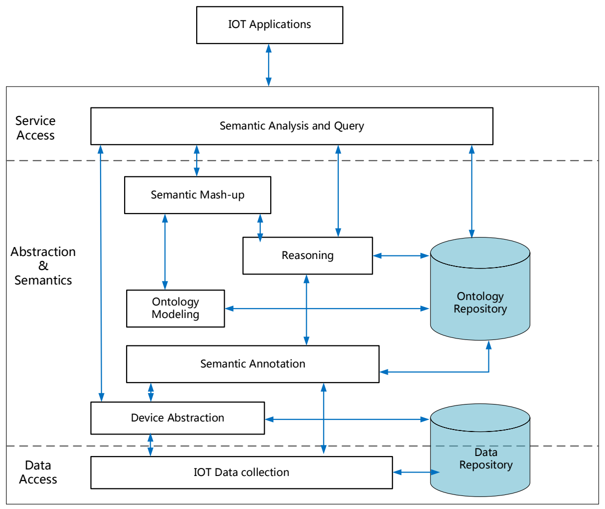


Figure 1. Generic Functional Semantic Model for Supporting IOT applications

II. RELATED WORK

To the best of our knowledge, our framework and system are the first work addressing various semantic processing tasks for large-scale streaming IOT data, including IOT semantic mash-up, semantic query and semantic reasoning. There is some related work as follows.

A. IOT Modeling and Ontology

One key research topic in IOT is to represent the "things" by standard vocabularies and schemas. Semantic Sensor Web (SSW) is a technology in which sensor data is semantic annotated for inter-operability and also provides contextual information for situational knowledge [8]. Many works have proposed semantic model for representing sensors and data. Ontologies such as the W3C's SSN ontology have been developed [6]. These ontologies provide metadata for numerical, spatial, temporal, and other semantic objects. Similar works for sensor metadata description also include Sensor Data Ontology (SDO) [9] and SensorML [10].

These works mainly focus on semantic annotation for the inter-operability of IOT by defining an unified and standard ontology, paying no attention to the high-level semantic IOT applications.

B. Semantic IOT Applications

Gyrard proposes a semantic-based Machine-to-Machine Measurement approach (M3) to automatically combine, enrich and reason about IOT data to provide promising cross-domain IOT applications, such as naturopathy application based on multiple datasets [11]. The approach also presents a hub for cross-domain ontologies and datasets. [12] and [13] apply the IOT in the generic agriculture and healthcare context management. SSEO [14] is developed to enable semantic indexing, machine-processable event detection and data exchange for smart space modeling. Other applications include CONON [15], CoOL [16] and CoBrA [17].

Most of the work aim at building IOT applications in a specific domain, failing to provide a generic semantic IOT processing framework. And they also did not deal with some important challenges for IOT data, such as the real-time and scalability.

C. Stream Processing for Semantic data

There are several semantic stream data processing engines, including Streaming SPARQL [18], C-SPARQL [19], CQELS [20]. Streaming SPARQL extends SPARQL to process data streams. C-SPARQL defines an extension of SPARQL whose distinguishing feature is the support of continuous queries, i.e. queries registered over RDF data streams and then continuously executed. CQELS is a native and adaptive query processor for unified query processing over Linked Stream Data and Linked Data.

However, most of these works only focus on the semantic query for linked stream data, ignoring other common demands of IOT applications, including semantic mash-up, reasoning and so on. What is more, the systems are designed to run on a single machine, while our system goes beyond that and specifically focuses on the common scalability issues for IOT applications.

III. PROPOSED FRAMEWORK AND ELASTIC PROCESSING ENGINE

In this section, we propose the general semantic framework for IOT applications and elaborate the elastic processing engine to explain how it provides the capabilities for performing various IOT applications.

A. Framework

Figure 2 shows the architecture of our semantic processing framework. In general, it consists of five parts: Physical Entities Layer, Abstract Entities Layer, Window-Based Data Stream Layer, Virtual Entities Layer and Elastic Semantic Engine Layer.

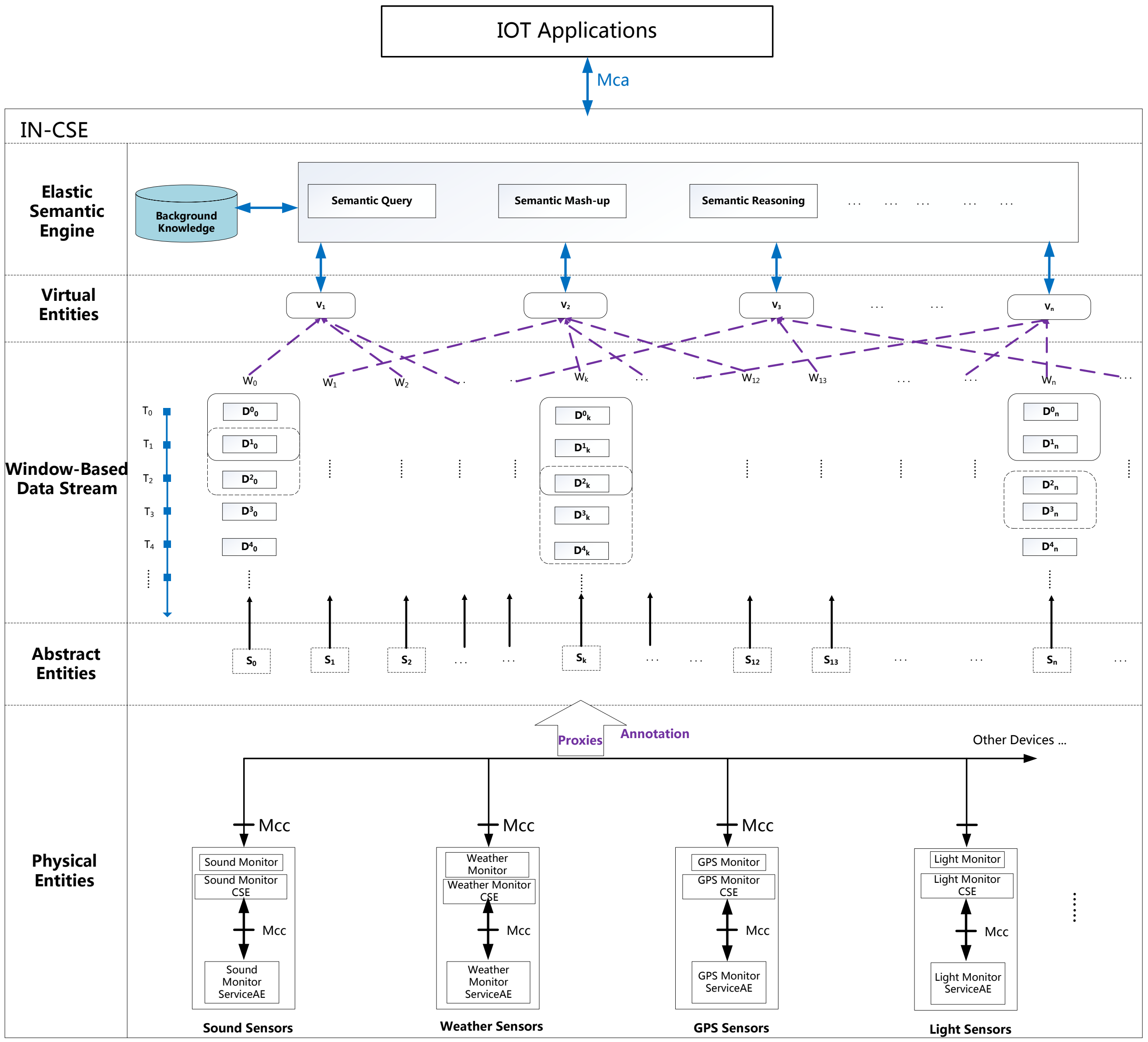


Figure 2. General Streaming Semantic Processing Framework For IOT Applications

*1) **Physical Entities Layer**:* Physical entities layer is located in the lowest layer of the framework, which is responsible for collecting raw sensor data in real-time. Every physical entity represents a tangible element that can be sensed by sensors that are deployed in the oneM2M Field Domain environment, and that is not specific to a particular IOT application in this environment. According to the oneM2M project standardization, every kind of sensors are be organized by logical entity (AE) and common services entity (CSE), which provide application logic and common services, respectively.

*2) **Abstract Entities Layer**:* Abstract Entities Layer is responsible for receiving and implementing the abstraction for the physical devices by the semantic annotation of proxy software. The abstraction layer aims at hiding the complexity of devices and environments by providing a standard format to represent devices. So from the view of upper layer, all the heterogeneous physical sensors can be seen as unified data streams.

*3) **Window-Based Data Stream Layer**:* The layer focuses on extracting related data streams into the windows according to the demands of upper applications. In the real-world IOT applications, data takes the form of continuous streams instead of the form of finite data sets stored in a traditional repository. This is the case for traffic monitoring, environment monitoring, disaster management, telecommunication management, manufacturing, and many other domains. Every sensor corresponds a window with a certain size.

*4) **Virtual Entities Layer**:* Virtual Entities Layer aggregates the related window data required by every virtual entity. Virtual entity is a new resource created by multiple window data streams, which is used to accomplish a application service. For our latter use case, if user in a home requests the service for Discomfort Index(DI), a new virtual entity will be generated through aggregating corresponding home appliance sensors(such as temperature, heater, air cleaner sensors and so on). Then we can get the service of DI by the virtual entity.

*5) **Elastic Semantic Engine Layer**:* The elastic semantic engine layer is the key of the architecture. The layer is responsible for receiving outer requests, creating corresponding virtual entities, interacting with static web of data, and real-time returning continuous results. Our work mainly focuses on the layer. We will discuss it in detail in the next section.

B. Elastic Streaming Processing Engine

Our elastic semantic streaming processing engine provides various common IOT services by constructing corresponding virtual entities, including IOT semantic Mash-up, query and reasoning. Every virtual entity aggregates the RDF streams from corresponding several windows. A window extracts the latest elements from the sensor stream. Besides the streaming IOT data, some applications need auxiliary background knowledge, such as Linked Open Data, DBpedia, LinkedGeo data and so on.

In this part, related definitions are first given, then we elaborate on the 3 functional modules of the engines.

Definition 1: **RDF Stream(S).** The basic data unit for RDF Stream is a quad ($< s, p, o, t >$). A RDF Stream is defined as an ordered sequence of pairs, where every pair is constituted by multiple basic data units, denoted as $D_i{}^j$, i represents the identification ID of certain sensor, j is the timestamp. For example, S_i=\{$D_i{}^0$, $D_i{}^1$, $D_i{}^2$,...,$i \in \mathbb{N}$\} denotes the RDF stream data of the ith sensor.

Definition 2: **Window(W).** A window is a subset of the RDF Stream given a time range t. W_i(t)= \{$D_i{}^0$, $D_i{}^1$, $D_i{}^2$,..., $D_i{}^{t-1}$,$t \in \mathbb{N}$\} denotes the RDF stream data of the ith sensor within the latest t logical time units. For example, W_0(3) can represent the data of Sensor$_0$ within the last 3 seconds.

Definition 3: **Virtual Window(V).** Virtual window aggregates the related data needed by a virtual entity, which includes the corresponding window and background knowledge $B = (s, p, o)$. A Virtual Window can be denoted as $V_i = \{W^i, B^i\}$, which W^i represents the set of windows aggregated by the Virtual Window V_i, B^i is the the set of background knowledge bases needed by the V_i.

1) IOT Semantic Mash-up: Semantic Mash-up is one of the most basic demands in IOT domain since many IOT applications rely on the task. It provides functionalities to support new services by aggregating multiple disperse resources. For example, "compute the indoor air quality index(AQI) of a room" is a typical Mash-up application, which needs to accomplish the task based on various window data sources including the PM2.5, O3, CO and so on.

The Mash-up task is formalized via the concepts of filtering and recombination. Given a set of RDF Streams S=\{S_0, S_1, S_2,...,S_N\}, N is the number of sensor streams. A filtering is a function υ: S$\rightarrow$ T which maps the primitive RDF streams to a triple set T=\{(s,p,o)\}. A recombination is a mapping γ: T $\rightarrow$ T' which transforms the T to T' based on the mash-up formulas. Take indoor AQI for example, T represents the triples containing the concentration of related sensors. γ denotes the mapping formula for computing the individual AQI.

2) IOT Semantic Query: IOT Semantic Query is a common function for IOT applications. It enhances the IOT discovery mechanism, to allow locating and linking resources or services based on their semantic information, such as "Get the temperature of the room$_0$", "Get the all rooms whose PM2.5 $\geq$ 80".

The query task is formalized via the concept of mapping. We denote as I, B, L, V respectively the domains of IRIs, blank nodes, literals, and variables which are all disjoint. We also define T = (I $\cup$ B $\cup$ L). A mapping μ is a partial function μ : V $\rightarrow$ T which gives the bindings for all the variables of a query. Evaluation occurs when a input graph pattern (denoted as P) in the query is matched against a Virtual Window (V). P is a set of triple patterns t = (s, p, o) such that s, p, o $\in$ (V $\cup$ T). We then define dom(μ) as the subset of V where μ is defined (i.e., the domain of μ) and use the notation μ(x) to refer to the bindings of variable x in μ.

3) IOT Semantic Reasoning: Reasoning is a mechanism to derive a new implicit knowledge from semantically annotated data and to answer complex user query. It can be implemented as a piece of software to be able to infer logical consequences from a set of asserted facts or axioms. Many IOT services belong to the application type. For example, we can infer the human comfort index based on the temperature and humility, the dangerous level of gas leaking and so on.

The reasoning task for IOT applications can be seen as the process of applying the reasoning rules in IOT data to derive new facts. We denote as W, B respectively the windows of sensor streams and background knowledge. F represents a set of facts that we want to γ contains a set of rules γ=\{$R_1, R_2, R_3, ...$\}. Thus the reasoning task is formalized as $\delta : (W, B) \xrightarrow{\gamma} F$.

IV. Use Case: Home Environment Monitoring

In this section, we give a common IOT use case. It is designed to facilitate the smart real-time monitoring to the home environment. We first give a overview of the use case, then the data model is presented. At last, three concrete application examples are introduced.

A. Overview

Nowadays, people are paying much attention to the environmental problems since we are facing a series of serious environmental pollution, such as smog disaster, water pollution and so on. To deal with these challenges, governments deploy lots of outdoor monitoring stations to capture and publish real-time environmental information to the public. However, there is limited work in the indoor environmental monitoring due to a lack of sensor devices and processing infrastructure.

With the popularity of smart home appliances (e.g., heater, air conditioner, humidifier, air cleaner, etc.) equipped with environment sensors (e.g., sensors for temperature, humidity, CO1, CO2, VOC, etc.), large volumes of data from all aspects of indoor environment is available, which makes it possible to implement various home monitoring applications including emergency detection, indoor air quality index and so on. Presently, many commercial companies such as Huawei, Cisco, Intel and Telecom are planing to deploy and develop related hardware and software infrastructure to provide similar services.

Our use case considers the scenario: Suppose a number of households in a city have installed relevant smart appliances, they want to get a series of environment monitoring services from the supplier, including Indoor Air Pollution Index (API), Indoor Sensor Discovery, Human Comfort Index (I_{hc}). The three cases correspond to the above three kinds of IOT applications: IOT Semantic Mashup, IOT Semantic Query, IOT Semantic Reasoning.

B. Data Model

As the paper mainly focuses on the background streaming data processing for IOT applications, we do not create a complex ontology to semantically annotate the all various IOT data. Conversely, we design a simple concept model for the home environment monitoring scenario (see Figure 3). The model captures three types of resources: Home, Room, Sensor. The label under the resource denotes the its URI. "p" is the namespace of the properties. Every sensor entity has three properties: type, value, time.

Figure 4 shows a snapshot of the stream knowledge graph based on the concept model. Every home contains multiple rooms(living room, bedroom, kitchen and so on). Every room is equipped with 15 kinds of sensors, including Temperature, Humility, Illumination, Volume, PM10, PM2.5, O3, CO, SO2, NO2 and so on.

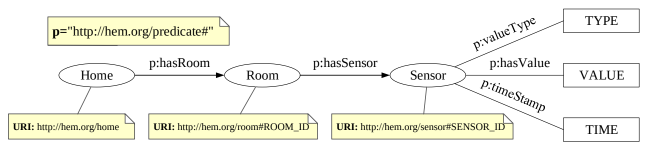


Figure 3. The Simple Concept Model for Home Environment Monitoring

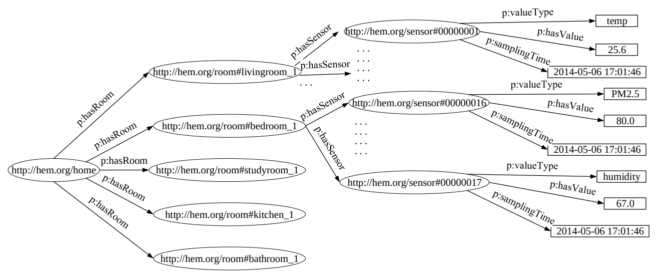

http://hem.org/home
p:hasRoom
http://hem.org/room#livingroom_1
http://hem.org/room#bedroom_1
http://hem.org/room#studyroom_1
http://hem.org/room#kitchen_1
http://hem.org/room#bathroom_1
p:hasSensor
http://hem.org/sensor#00000001
http://hem.org/sensor#00000016
http://hem.org/sensor#00000017
p:valueType
p:hasValue
p:samplingTime
temp
25.6
PM2.5
80.0
humidity
67.0
2014-05-06 17:01:46

Figure 4. A Snapshot of the Stream Knowledge Graph

C. Scenarios

1) Streaming Semantic Mash-up: Outdoor AQI is available to us for years, while little attention is paid on indoor AQI, which is also important both for customers and device suppliers. For customers, the indoor AQI can help them keep track of the latest situation of the house and the top air pollutants. For suppliers, the AQI data can help them monitor their devices and have a better understanding of the needs of different customers.

The indoor AQI task is a typical IOT semantic Mash-up application since it needs to integrate multiple window data sources and combine them to complete a specific work. The indoor AQI task is composed of two phases. The first step is to compute the Individual AQI(IAQI) for a pollutant. Equation 1 gives the computing formula for $IAQI_p$ based on [21]. The parameters in the equation are relevant with the specific pollutant. Related pollutants include PM10, PM2.5, O_3, CO, SO_2, NO_2. Then these IAQIs will be sorted based on Equation 2. The final result will be displayed by a sequence of IAQIs in a descending order, which will keep customers informed of the noticeable pollutants. For more details, readers can refer [21].

$$IAQI_p = \frac{IAQI_{Hi} - IAQI_{Lo}}{BP_{Hi} - BPLo}(C_p - BP_{Lo}) + IAQI_{Lo} \quad (1)$$

$$AQI = \max\{IAQI_1, IAQI_2, IAQI_3, ..., IAQI_n\} \quad (2)$$

2) Streaming Semantic Query: The streaming semantic query provides us with a basic function to discover and query the IOT resources in real time. We implement the following 6 query examples to illustrate its applications (Table I). All streaming queries are showed in the Appendix A.

Table I
IOT STREAMING QUERY EXAMPLES

Query	Goal
Q1	Query the type and value of all sensors in a specific room
Q2	Query the value of PM2.5 sensor in a specific room
Q3	Query the value of all CO sensors and corresponding room ID
Q4	Query the value of all SO_2 sensors and corresponding sensor ID
Q5	Query all the room ID and sampling time with humidity sensor
Q6	Query the temperature and sampling time in a specific room

3) Streaming Semantic Reasoning:
Reasoning is ubiquitous in the IOT environment. We can derive corresponding conclusions if certain data streams trigger reasoning rules. For example, we can get warnings if some pollutants' concentration exceed normal range, such as temperature and CO. Other reasoning examples include inferring the health status and sleep quality of a person based on some wearable devices such as smart band.

Here we give the example of reasoning the human comfort level. The application will help us to keep track of the conditions of indoor rooms, so further automatically adjust the indoor environment and remind us to prevent heatstroke or cold in time. Specifically, two kinds of sensor streams(temperature and humanity) will be first integrated to compute the human comfort index according to Equation 3. Then based on reasoning rules in Appendix B, the level of human comfort will be derived.

$$I_{HC} = T - 0.55(1 - H_R)(T - 58) \tag{3}$$

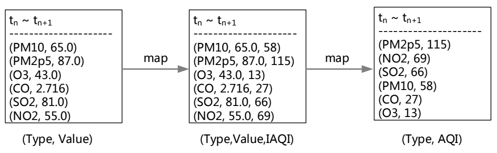
V. EXPERIMENT AND EVALUATION

In this section, we introduce the experiments and evaluations. First the experimental environment is briefly presented including the configuration and data. Then extended experiments are performed to evaluate the system's functionality and scalability.

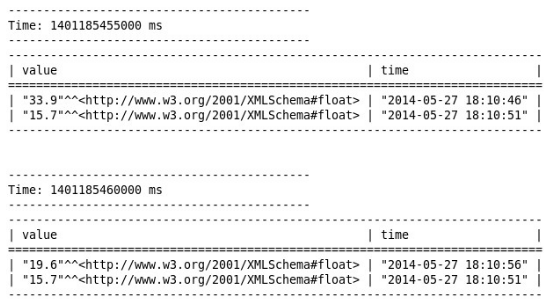
A. Experiment Setup

Configuration: The experiment is implemented on a Spark cluster with three machines. Each node has 16 GB DDR3 RAM, 8-core Intel Xeon(R) E5606 CPUs at 2.13GHz, 1.5TB disk. The nodes are connected by the network with the bandwith of 1000M/s. All the nodes use CentOS6.4 with the softwares JDK-1.7.0, Scala-1.10.1 and Spark-0.9.0.

Data: The experimental data is generated by our stream data generator whose schema is based on the concept model in figure 3. The main parameters of the generator are R and T, denoting the number of home and sampling time, respectively. The number of home is in proportion to the number of sensor denoted by N_s(N_s=15*5*R, 15 and 5 represent the number of sensors in a room and rooms in a home). Sampling time stimulates the rate of sensor stream. The average data size generated by a sensor within a sampling time is 0.5(KB). Data generator and all the source codes are available in [2].



(a) A Snapshot of Indoor AQI Streaming Mash-up



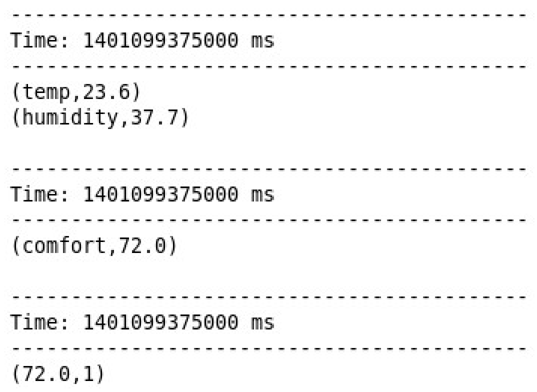
```
-------------------------------------------
Time: 1401185455000 ms
-------------------------------------------

----------------------------------------------------------------------------
| value                                           | time                   |
============================================================================
| "33.9"^^<http://www.w3.org/2001/XMLSchema#float> | "2014-05-27 18:10:46" |
| "15.7"^^<http://www.w3.org/2001/XMLSchema#float> | "2014-05-27 18:10:51" |
----------------------------------------------------------------------------


-------------------------------------------
Time: 1401185460000 ms
-------------------------------------------

----------------------------------------------------------------------------
| value                                           | time                   |
============================================================================
| "19.6"^^<http://www.w3.org/2001/XMLSchema#float> | "2014-05-27 18:10:56" |
| "15.7"^^<http://www.w3.org/2001/XMLSchema#float> | "2014-05-27 18:10:51" |
----------------------------------------------------------------------------
```

(b) A Snapshot of Indoor CO Streaming Query



```
-------------------------------------------
Time: 1401099375000 ms
-------------------------------------------
(temp,23.6)
(humidity,37.7)

-------------------------------------------
Time: 1401099375000 ms
-------------------------------------------
(comfort,72.0)

-------------------------------------------
Time: 1401099375000 ms
-------------------------------------------
(72.0,1)
```

(c) A Snapshot of Indoor Human Comfort Streaming Reasoning

Figure 5. The Snapshot of Indoor Environment Monitoring Scenarios

B. Functional Evaluation

For functional experiment, we first briefly introduce the implementation of the elastic streaming semantic engine. Then based on the proposed engine, 3 kinds of representative IOT applications are presented to illustrate the functional characteristics of our system.

We builded our elastic streaming processing engine based on the efficient in-memory cluster computing framework-SPARK, which provides us with rich data abstraction and operation abstraction to meet the needs of various IOT applications. For the data model, we use the DStream(Discreted Stream) to model the window streams. For the processing model, operators provided by SPARK such as "filter" and

[2]www.github.com/hualichenxi/Semantic-IOT-Engine/DataGenerator

”map ”are translated into the processing primitives to effectively implement the different IOT scenarios.

Presently, we have preliminarily implemented the IOT mash-up subsystem, query subsystem, and reasoning subsystem. Every subsystem acts as a module of our elastic semantic processing engines. Once a IOT application requests service, corresponding subsystem will be activated to run a Spark job to return continues results. Figure 5 shows partial running results of the previous 3 use cases. For more results, readers can access [3].

C. Scalability Evaluation

For scalability experiment, we use the AQI use case to illustrate the performance of our engine by increasing the number of sensor streams. During the process of spark streaming execution, we will write a total delay(TD) into the log file after the data in a time slice has been processed completely. The parameter records the total time from receiving window data to output final results. In our experiment, the time slice(D) is set as 5 seconds and we will run the program for 300 seconds. That is to say, 60 TD will be written into the log file.

Figure 6 and Figure 7 show the trend of the processing time(TD) in single node and cluster with varied sensors. For single node experiment, the number of sensors is varied from 15,000 to 150,000. For cluster experiment, the number of sensors is varied from 75,000 to 750,000. The two figures only show the processing time for parts of sensors so that we can recognize the broken line well. From both figures, in the beginning of executing a spark job, TD is not stable(0∼50) for all varied sensors. After a while, TD will stay in a comparatively stable level. Here we choose these TD in the last 250s and compute their average value denoted as TD_{SA}. To capture the fluctuation of the processing time, we also compute the standard deviation σ of TD_{SA}.

Equation 4 computes the system's throughout Q(MB/s): 0.5 is the average data size generated by a sensor in a sampling time, N_S is the number of sensors, TD_{SA} is the processing time.

$$Q = \frac{1}{1024} \times \frac{0.5N_s}{TD_{SA}} \tag{4}$$

$$Sizeup = \frac{computing \quad time \quad for \quad processing \quad m \times data}{computing \quad time \quad for \quad processing \quad data} \tag{5}$$

Table II and Table III show the execution results in single node and cluster. Figure 6 and Figure 7 show the throughput and processing time with increased sensors. We can conclude the following results from the tables.

Firstly, we can get the correlation among relevant variables: TD$_{SA}$, Q, N$_s$. From Figure 8 and 9, we can see that TD$_{SA}$ and Q increase with the increasing numbers of sensors. But when the ratio of TD$_{SA}$ to D reaches a certain value, the throughput will decrease rapidly. This is because the computing capability of the system will not be able to catch up with the rate of the input data stream. As a result, the delay caused by the last time slice will have an effect on the next process and the system will become more and more unstable. The evident increasing of σ also reflects the result that TD$_{SA}$ has a larger fluctuation. The correlation analysis can help us control the rate of input stream and set proper time slice to accommodate the ability of the system.

Secondly, Table II and Table III show that our system achieves high throughputs: more than 53MB/s and 175 MB/s in single node and cluster. Benefiting from the system's elastic processing ability, it can concurrently process more than 300,000 sensor streams efficiently.

At last, the tables show that our system achieves excellent scalability. For both single and cluster configuration, the sizeup (see Equation 5) of m times input is much less than m. Especially for the cluster, when the input stream increases by 6 times($N_s = 262,500$), the processing time only increases by less 2(1.72) times. The results mean that the TD increases much more slowly than input data size and our system works better in processing larger input stream. At the same time, the two tables also show that the processing capability of cluster is much better than single node. For example, when the number of input stream sensors is 150,000, the execution time in cluster is 0.575, compared to 2.440 in single node. The best throughput in cluster (175MB/s) is more than 3 times of the one in single node(53MB/s). It proves our system achieves good flexibility and elastic scalability: It can adapt to various different application scenarios and requirements by adding computing nodes.

To sum up, the results demonstrate excellent scalability regarding both the size of input stream and number of nodes.

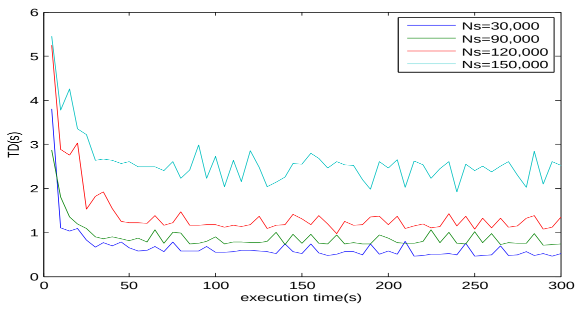


Figure 6. Processing Time(TD) with Increased Sensors in Single Node

[3]www.github.com/hualichenxi/Semantic-IOT-Engine/Experiment

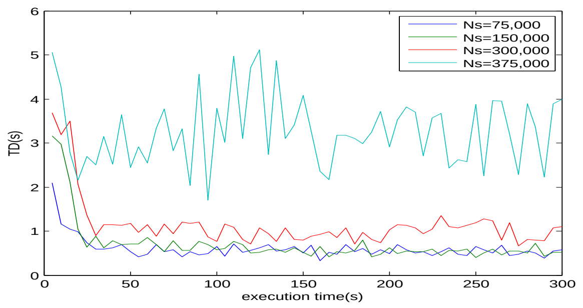


Figure 7. Processing Time(TD) with Increased Sensors in Cluster

Table II
THROUGHPUT AND SIZEUP IN SINGLE NODE

N_s	TD(s)	TD/D	σ	Q(MB/s)	Sizeup
15,000	0.308	6.17%	0.06584	23.759	1
30,000	0.566	11.32%	0.08999	25.879	1.836
45,000	0.635	12.70%	0.09619	34.605	2.06
60,000	0.727	14.54%	0.11524	40.305	2.36
75,000	0.751	15.02%	0.11416	48.769	2.44
90,000	0.821	16.42%	0.10602	53.518	2.66
105,000	0.956	19.12%	0.11979	53.635	3.10
120,000	1.218	24.37%	0.13368	48.092	3.95
135,000	1.785	35.70%	0.19057	36.930	5.79
150,000	2.440	48.81%	0.24599	30.012	7.92

Table III
THROUGHPUT AND SIZEUP IN CLUSTER

N_s(k)	TD(s)	TD/D	σ	Q(MB/s)	Sizeup
37,500	0.438	8.76%	0.08675	41.825	1
75,000	0.543	10.85%	0.08929	67.474	1.24
112,500	0.566	11.32%	0.10169	97.025	1.29
150,000	0.575	11.49%	0.10676	127.452	1.31
187,500	0.623	12.46%	0.13582	146.922	1.42
225,000	0.627	12.54%	0.13051	175.160	1.40
262,500	0.753	15.07%	0.14888	170.142	1.72
300,000	0.991	19.83%	0.17464	147.753	2.26
337,500	2.519	50.37%	0.56164	65.431	5.75
375,000	3.294	65.89%	0.78245	55.583	7.52

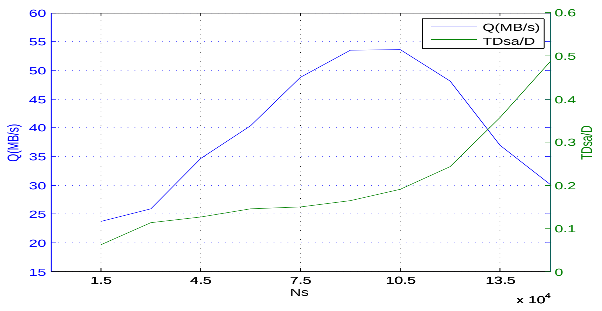


Figure 8. Throughput and TD with Increased Sensors in Single Node

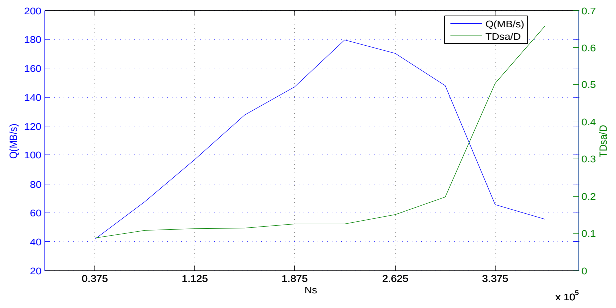


Figure 9. Throughput and TD with Increased Sensors in Cluster

VI. CONCLUSION AND FUTURE WORK

To effectively process large-scale streaming IOT data, the paper presents a general semantic processing framework for various IOT applications. Followed by the framework, we have implemented an elastic streaming engine based on popular large-scale distributed computing platform S-PARK. Based on the engine, a typical use case on home environment monitoring is given to illustrate the efficiency of our engine. The results show that our system can scale for large number of sensor streams with different types of IOT applications. For future work, we are planing to deploy spatial semantic support in our distributed semantic engine to process all kinds of location-based IOT applications such as taxi service, parking service and so on.

REFERENCES

[1] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, 2011.

[2] V. Cackovic and Z. Popovic, "Abstraction and semantics support in m2m communications," in *Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on.* IEEE, 2013, pp. 404–408.

[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," 2009.

[4] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia-a crystallization point for the web of data," *Web Semantics: science, services and agents on the world wide web*, vol. 7, no. 3, pp. 154–165, 2009.

[5] S. Auer, J. Lehmann, and S. Hellmann, *Linkedgeodata: Adding a spatial dimension to the web of data.* Springer, 2009.

[6] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog *et al.*, "The ssn ontology of the w3c semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 17, pp. 25–32, 2012.

[7] M. Zaharia, M. Chowdhury, T. Das, A. Dave, and Ma, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 2–2.

[8] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic sensor web," *Internet Computing, IEEE*, vol. 12, no. 4, pp. 78–83, 2008.

[9] M. Eid, R. Liscano, and A. El Saddik, "A universal ontology for sensor networks data," in *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on*. IEEE, 2007, pp. 59–62.

[10] M. Botts and A. Robin, "Opengis sensor model language (sensorml) implementation specification," *OpenGIS Implementation Specification OGC*, pp. 07–000, 2007.

[11] A. Gyrard, C. Bonnet, and K. Boudaoud, "Enrich machine-to-machine data with semantic web technologies for cross-domain applications," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 559–564.

[12] S. Hu, H. Wang, C. She, and J. Wang, "Agont: Ontology for agriculture internet of things," in *Computer and Computing Technologies in Agriculture IV*. Springer, 2011, pp. 131–137.

[13] D. Ejigu, M. Scuturici, and L. Brunie, "An ontology-based approach to context modeling and reasoning in pervasive computing," in *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on*. IEEE, 2007, pp. 14–19.

[14] Z. Li, C.-H. Chu, W. Yao, and R. A. Behr, "Ontology-driven event detection and indexing in smart spaces," in *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*. IEEE, 2010, pp. 285–292.

[15] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology based context modeling and reasoning using owl," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*. Ieee, 2004, pp. 18–22.

[16] T. Strang, C. Linnhoff-Popien, and K. Frank, "Cool: A context ontology language to enable contextual interoperability," in *Distributed applications and interoperable systems*. Springer, 2003, pp. 236–247.

[17] H. Chen, T. Finin, and A. Joshi, "Using owl in a pervasive computing broker," DTIC Document, Tech. Rep., 2005.

[18] A. Bolles, M. Grawunder, and J. Jacobi, *Streaming SPARQL-extending SPARQL to process data streams*. Springer, 2008.

[19] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus, "C-sparql: Sparql for continuous querying," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 1061–1062.

[20] D. Le-Phuoc, H. N. M. Quoc, C. Le Van, and M. Hauswirth, "Elastic and scalable processing of linked stream data in the cloud," in *The Semantic Web–ISWC 2013*. Springer, 2013, pp. 280–297.

[21] H. 2012, "Chinese air quality index specifications [s]," Ph.D. dissertation, 2012.

APPENDIX

We provide the SPARQL queries used in the experimental section of streaming semantic query:

A. *IOT Streaming Semantic Query Examples*

PREFIX p: $< http : //hem.org/predicate\# >$.
Q1: SELECT ?type ?value FROM STREAM $< http : //hem.org/streams/home >$ WHERE { $< http : //hem.org/room\#bedroom_1 >$ p:hasSensor ?sensor . ?sensor p:valueType ?type . }
Q2: SELECT ?value FROM STREAM $< http : //hem.org/streams/home >$ WHERE { $< http : //hem.org/room\#bedroom_1 >$ p:hasSensor ?sensor . ?sensor p:hasValue ?value . ?sensor p:valueType 'PM2.5' . }
Q3: SELECT ?room ?value FROM STREAM $< http : //hem.org/streams/home >$ WHERE { ?room p:hasSensor ?sensor . ?sensor p:hasValue ?value . ?sensor p:valueType 'CO' . }
Q4: SELECT ?sensor ?value FROM STREAM $< http : //hem.org/streams/home >$ WHERE { ?sensor p:hasValue ?value . ?sensor p:valueType 'SO2' . }
Q5: SELECT ?room ?time FROM STREAM $< http : //hem.org/streams/home >$ WHERE { ?room p:hasSensor ?sensor . ?sensor p:hasValue ?value . ?sensor p:valueType 'humidity' . ?sensor p:samplingTime ?time . }
Q6: SELECT ?value ?time FROM STREAM $< http : //hem.org/streams/home >$ WHERE { $< http : //hem.org/room\#bedroom_1 >$ p:hasSensor ?sensor . ?sensor p:hasValue ?value . ?sensor p:valueType 'temp' . ?sensor p:samplingTime ?time . }

B. *IOT Streaming Semantic Reasoning Example*

Table IV
HUMAN COMFORT TABLE

HC Index Range	Level	HC Description
86∼88	+4	very hot
80∼85	+3	hot
76∼79	+2	warm
71∼75	+1	slightly warm
59∼70	0	comfotable
51∼58	-1	slightly cool
39∼50	-2	cool
26∼38	-3	cold
≤ 25	-4	very code