# Elastic Streaming Semantic Engine for Web of Thing Appliations

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 sensor data collection and communication, heterogeneous and real-time IOT data is increasing rapidly. To make full use of the data, lots of web technologies such as semantic technologies and web service are introduced to effectively organize, publish, integrate and process the sensor data. Thus the web infrastructure integrates the IOT world and Web world into a unified Web of Thing(WOT). While most of the existing efforts are focused on the modeling, annotation, and representation of WOT data, there has been little work focusing on the background processing of large-scale streaming WOT data. In the paper, we present an elastic streaming semantic processing framework for WOT applications. The proposed framework efficiently capture and model different scenarios for all kinds of WOT applications. We have preliminarily implemented an engine 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 WOT applications.**

***Keywords*-WOT, Semantic Web, IOT, Streaming, Big Data, Spark.**

## I. Introduction

With the advances in sensor data collection and communication technologies, increased number of sensors have been deploying world-wide, which builds a global sensing system. 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) 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, logistics and environmental monitoring. Such a stunning number of devices will generate massive data. These sensor data can help us to observe the surroundings, learn patterns; we can also learn concepts and associate these concepts with shapes and patterns. Then we are able to generalize, recognize unseen patterns and infer new ones.

To make full use of the senses and have a better understanding of the world, a natural next step would be to find a way to connect the sensor things with existing web of data. The web infrastructure is the ideal means to integrate these two worlds to a unified Web of Thing. Presently, much efforts have been put on this area from Internet of Things(IOT) to Web of Things(WOT). 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 for M2M and IOT-oneM2M, whose purpose and goal is to develop technical specifications which address the need for a common WOT Service Layer by reusing existing web standards and protocols.

However, the characteristics of WOT, including heterogeneity, variety, volume and real-time, pose a series of challenges to integrate WOT data and deploy WOT applications effectively. The Semantic Web technologies are viewed as a key for the development of WOT. Figure 1 shows the generic functional model of M2M for supporting semantics in the specification of oneM2M study on abstraction and semantics enablement. In specific, it serves as the following several purposes: First, the data access layer provides connections with a device and a gateway for accessing M2M data. Secondly, the abstraction and semantics layer provide us with a good way to resolve the problems of inter-operability and integration within this heterogeneous world of M2M devices by defining and reusing some standard semantic concepts. Besides, the Semantic Web provides a seamless interface to facilitate the interactions of M2M data and other existing web of data such as Linked Data, DBpedia, LinkedGeodata, various kinds of Web Services. At last, the service layer provides an interface for various WOT applications by semantic processing technologies, including semantic query, mash-up and so on.

Currently, most of the existing works aim at annotating and integrating the WOT data by providing corresponding description ontology, there has been little work focusing on the background processing of large-scale real-time WOT data for common WOT applications. For example, ontologies such as the W3Cs SSN ontology (Lefort et al., 2011; Compton et al., 2012) have been developed, which offers a number of constructs to formally describe not only the sensor resources but also the sensor observation and measurement data. In the paper, we present an elastic

streaming semantic processing framework for WOT applications. The proposed framework efficiently capture and model different scenarios for various WOT applications. We have preliminarily implemented an engine 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 WOT applications.

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

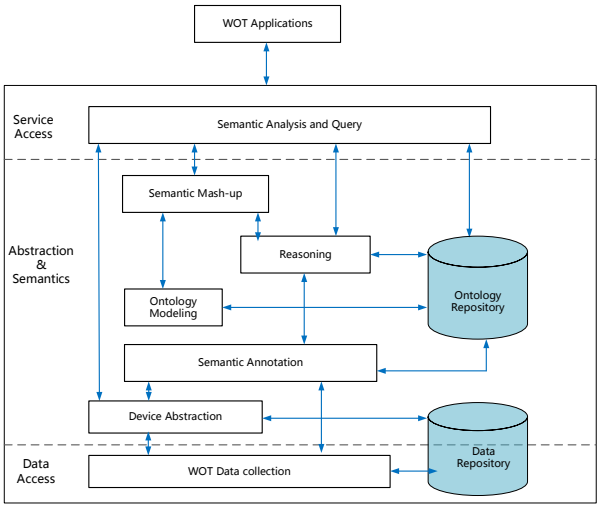


Figure 1. Generic Functional Semantic Model for Supporting WOT

## II. RELATED WORK

To the best of our knowledge, our framework and system are the first work addressing various elastic semantic processing for large-scale streaming WOT data, including WOT semantic mash-up, WOT semantic query and semantic reasoning. But our approaches touch on a number of areas.

### A. WOT Modeling and Ontology

One key research topic in WOT is to represent the "things" by standard vocabularies and schemas.For example, 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. Many works have proposed semantic model for representing sensors and data. Ontologies such as the W3C's SSN ontology have been developed. In particular, these ontologies provide metadata for numerical, spatial, temporal, and other semantic objects. Similar works for sensor metadata description also include Sensor Data Ontology (SDO) , Sensei O&M and SensorML. Meanwhile, there are ontologies like CSIRO Sensor Ontology and OntoSensor designed to describe semantic sensor specification and knowledge. In order to improve ontology reusability and inter-operability, some of them extend their ontology based on IEEE's Suggested Upper Merged Ontology (SUMO) as a root definition of general concepts and associations for semantic modeling. Other ontologies include LinkedGeo ontology in the spatial domain, Uniprot ontology in the biology domain, FOAF ontology in the social domain and so on.

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

### B. Semantic WOT Applications

Gyrard proposes a semantic-based Machine-to-Machine Measurement approach (M3) to automatically combine, enrich and reason about WOT data to provide promising cross-domain WOT applications, such as naturopathy application based on multiple datasets. The approach also presents a hub for cross-domain ontologies and datasets. [20] and [21] apply the WOT in the generic agriculture and healthcare context management. SSEO [22] is developed to enable semantic indexing, machine-processable event detection and data exchange for smart space modeling. Other applications include CONON [23], CoOL [24] and CoBrA [25].

Most of the work aim at building WOT applications in a specific domain, failing to provide a generic semantic WOT processing framework. And the applications also did not deal with some important challenges for WOT 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, C-SPARQL, CQELS, etc. 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 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 WOT applications. Moreover, these works only focus on the semantic query for linked stream data, ignoring other common demands of WOT applications, including semantic mash-up, reasoning and so on.

## III. Proposed Framework And Elastic Processing Engine

In this section, we propose our elastic streaming semantic processing framework for WOT applications and elaborate the Elastic Processing Engine to explain how it provides the capabilities for performing various WOT applications.

### A. Framework

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

*1)* ***Physical Entities Layer****:* Physical entities layer is located in the lowest layer of the framework, which is responsible for generating 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 WOT 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 data from the physical devices by the semantic annotation of proxy software. The abstraction layer aims at hiding the complexity of variety of devices and environments by providing a single and 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. In the real-world WOT applications, data take 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. Our work mainly focus on the layer. We will discuss the layer later in detail. The layer is responsible for receiving outer requests, creating corresponding virtual entities, interacting with static web of data, and real-time returning service results.

### B. Elastic Processing Engine

*1) WOT Data Model:* All WOT data is modeled as the RDF Stream. Every virtual entity aggregates the RDF streams from corresponding several windows. A window extracts from its sensor stream the latest elements , which is considered by the virtual entity. Besides the streaming WOT data, some applications need auxiliary background knowledge, such as Linked Open Data, DBpedia, LinkedGeo data and so on. Related definitions are as follows:

*Definition 1:* **RDF Stream(S).** The basic data unit for RDF Stream is ($< s, p, o >$,timestamp). 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)= $\{D_i{}^0, D_i{}^1, D_i{}^2,..., D_i{}^(t-1)\}$denotes the RDF stream data of the ith sensor within the latest t logical time unit(second, minute, hour, number...). Windows are sliding when they are progressively advanced of a given STEP. For example, the size of window and sliding window for $W_0$ in the figure 2 are 2 and 1, respectively.

*Definition 3:* **Virtual Window(V).** Virtual window aggregates the related data needed by a virtual entity, which includes the corresponding window data 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 $V_i$.

*2) WOT Semantic Mash-up:* Semantic Mash-up is one of the most common and important demands in WOT domain since many WOT applications rely on the task. It provides functionalities to support new services through by obtaining semantic information through semantic descriptions from multiple existing WOT 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 PM10,PM2.5,O3,CO,SO2,NO2 sensors.

The mash-up task is formalized via the concepts of filtering and recombination. Given a RDF Stream S=$\{S_0, S_1, S_2,...,S_N\}$, N is the number of sensors in the stream. A filtering f is a function f: S$\rightarrow$ T which maps the primitive RDF stream to a triple set T=$\{$(s,p,o)$\}$. A recombination r is a mapping r: 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

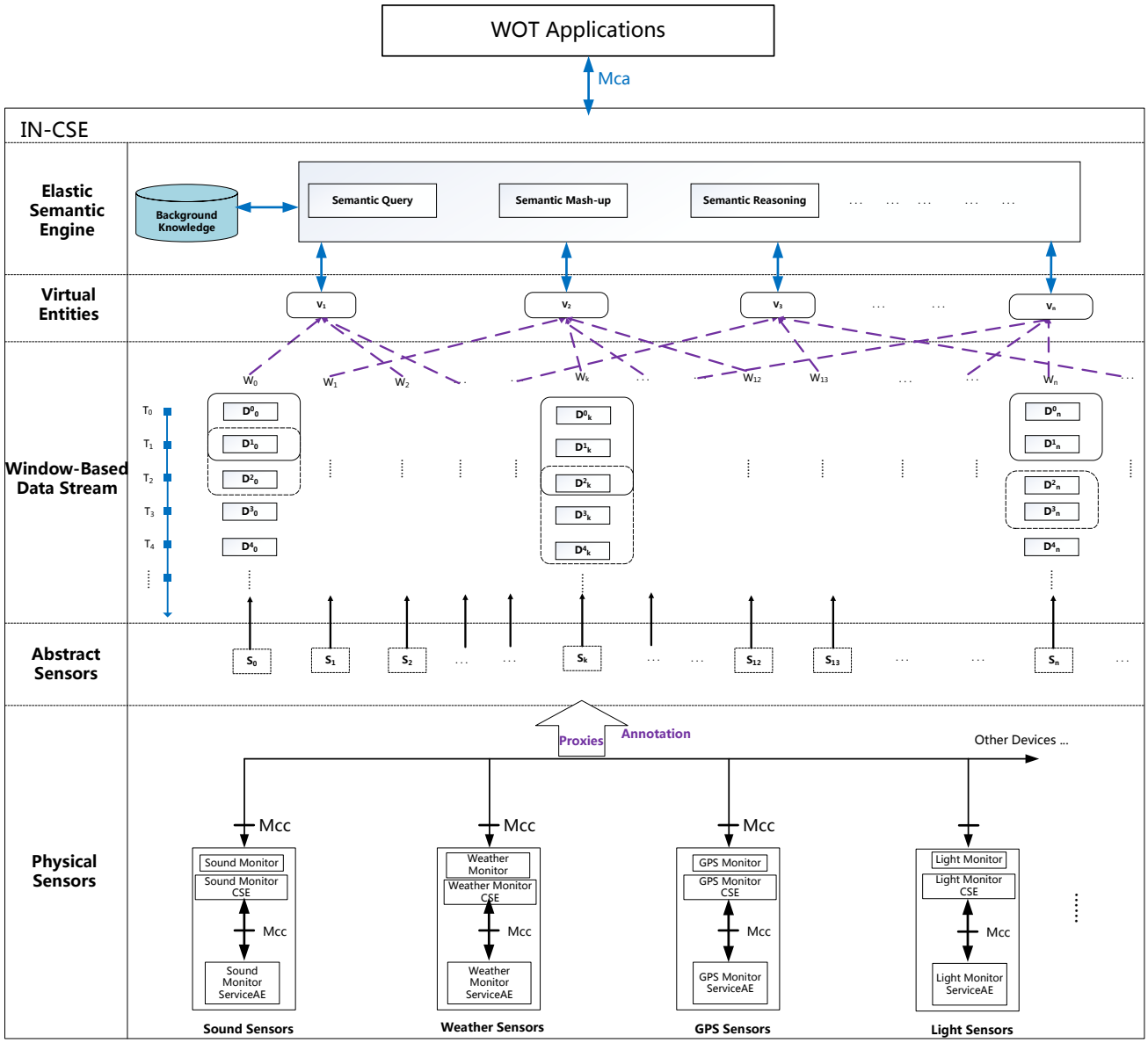


Figure 2. Semantic Processing Framework For M2M Applications

sensors. r denotes the mapping formula for computing the individual AQI.

*3) WOT Semantic Query:* WOT Semantic Query is a basic function for WOT applications. It enhances the WOT discovery mechanism, to allow locating and linking resources or services based on their semantic information. For example, "Get the temperature of the room 1", "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 $\mu$ is a partial function $\mu$ : V $\rightarrow$ T which gives the bindings for all the variables of a query. Evaluation occurs when a 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($\mu$) as the subset of V where $\mu$ is defined (i.e., the domain of $\mu$ and use the notation $\mu$(x) to refer to the bindings of variable x in $\mu$.

*4) WOT 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 WOT 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 WOT applications can be seen as the process of applying the reasoning rules in WOT data to derive new facts. We denote as W, B respectively the data of sensor streams and background knowledge. $F$ represents a set of facts that we want to $\gamma$ contains a set of rules $\gamma$=$\{R_1, R_2, R_3, ...\}$. 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 use case of WOT domain. It is designed to facilitate the smart real-time monitoring to the home environment. We first give a overview of the use case, then three concrete application examples are introduced briefly.

### 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 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 home 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 considering 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 or sensors related to home environment, 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 applications correspond to three kinds of WOT applications: WOT Semantic Mashup, WOT Semantic Query, WOT Semantic Reasoning.

### B. Data Model

As the paper mainly focuses on the background streaming data processing for WOT applications, we do not create a complex and comprehensive ontology to semantically annotate the all various WOT 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 model 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.

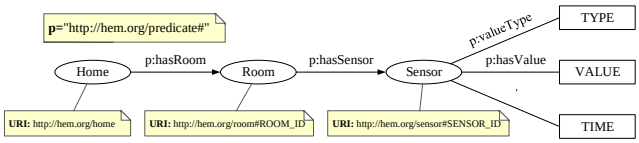


Figure 3. The Simple Concept Model for Home Environment Monitoring

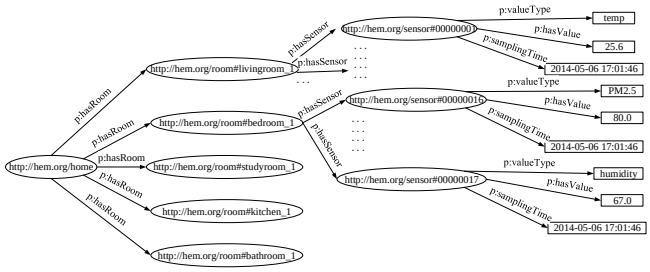


Figure 4. A Snapshot of the Stream Knowledge Model

### C. Scenarios

*1) Semantic Mashup:IAQI and AQI:* Outdoor AQI is available to us for years, while little attention is paid on indoor AQI, which is quite 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 WOT semantic mashup 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(IAQL) for a pollutant. Equation 1 gives the computing formula for $IAQI_p$ based

on []. 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

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

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

*2) Semantic Query:S:*

*3) Semantic Reasoning:$I_{hc}$:* emergency detection

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

## V. Model Implementation and Evaluation

In this section, we briefly introduce the implementation of the M2M application framework. Then extended experiments is performed to evaluate the system from the functionality and scalability.

### A. Model Implementation

Followed by proposed framework, 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 M2M applications. We use the DStream(Discreted Stream) to model the window stream and implement related functions by translating the to the operators of SPARK including "filter" and "join" transformation .

Presently, we have preliminarily implemented the M2M query subsystem, mashup subsystem and reasoning subsystem. Every subsystem acts as a module of our elastic semantic processing engines. Once a M2M application requests service, corresponding subsystem will run a continues Spark job to return

### B. Evaluation

*1) 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).

*2) Functionality Evaluation:*

*3) Scalability Evaluation:* 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 4 and Figure 5 show the tread 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$\tilde{5}$0). 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 compute the standard deviation $\sigma$ of $TD_{SA}$.

Equation 4 computes the system's throughout(Q): 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.

$$Q = \frac{1}{1024} \times \frac{0.5N_s}{TD_{SA}} \quad (4)$$

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

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 :

Firstly, we can get the correlation among relevant variables: $TD_SA$, Q, $N_s$. From the graphs 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 velocity of the 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 $\sigma$ will also increase evidently because the $TD_SA$ 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 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 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.

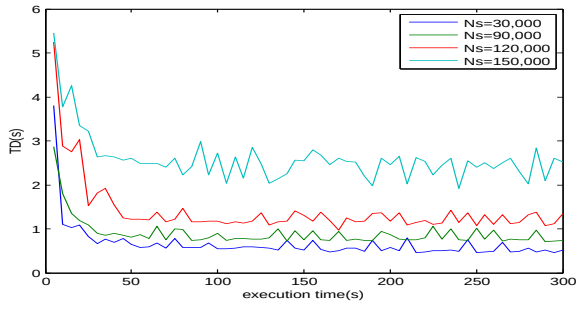


Figure 5. TD in single node

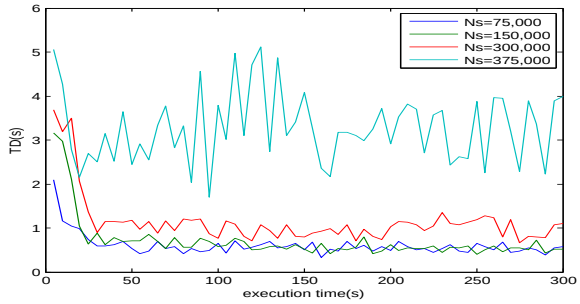


Figure 6. TD in cluster

Table I
THROUGHPUT IN SINGLE NODE

| $N_s$ | TD(s) | TD/D | $\sigma$ | 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 II
THROUGHPUT IN CLUSTER

| $N_s$(k) | TD(s) | TD/D | $\sigma$ | 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 |

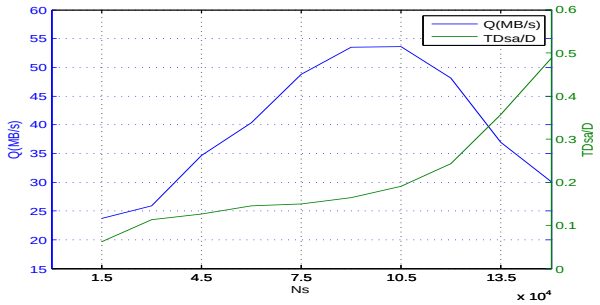


Figure 7. Throughput and TD with Increased Sensors in Single Node

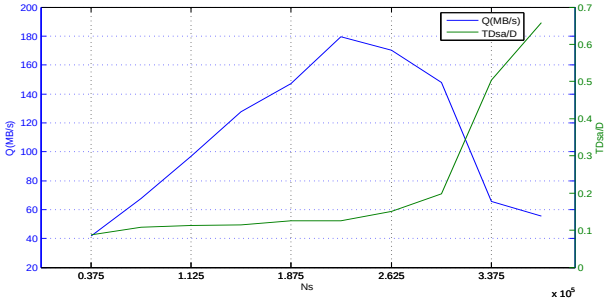


Figure 8. Throughput and TD with Increased Sensors in Cluster

## VI. CONCLUSION