A Novel CEP Model and Its Applications in Internet of Things Big Data Processing

Jing Sun and Huiqun Zhao

***Abstract*—The Internet of Things (IoT in short) have been experienced tremendously fast development in the past decade whither in theory or practice. But one of the side effects of the IoT is that it can create an unprecedented amount of data. As the typical technology, the CEP (Complex Event Processing CEP in short) has plays an important role in the processing of dealing with IoT event flow. However, how to control the huge amount of data injected into the network and how to fast transform them to end user has met the challenge. In this paper, an CEP model is proposed for facilitating algorithm analysis and formal expression. Based on the CEP model a few algorithm for dealing with big data problem are discussed formally, including fast classification of event flow and the fast pushing of data report service. Two case studies of EPC IoT are practiced, and the experiment results show the feasibility of proposed algorithm which is based on CEP model.**

***Index Terms*—Internet of things, complex event processing, Algebra model, EPC network.**

I. Introduction

The term Internet of Things was first coined by Kevin Ashton in 1999 in the context of supply chain management [1]. In the past decade the definition has been more inclusive covering wide range of applications and IoT has become an emerging information field. There are several application domains which having been impacted by the emerging IoT. Personal and Home IoT at the scale of an individual or home, Enterprise IoT at the scale of a community, Utility IoT at a national or regional scale and Mobile IoT which is usually spread across other domains mainly due to the nature of connectivity and scale [2]. IoT employs a layered architecture which is consisted of four layers [3]: (1) *Sensing Layer* is a connection between physical and cyber worlds, such as RFID read, sensor, monitoring camera, etc. (2) *Network Layer* is the media to convey the collected data to upper layers for further processing and higher-level abstraction. (3) *Middleware Layer* is a crucial part in IoT architecture. IoT middleware is an effective integration of several major functions including communications management, devices management, data processing, semantic reasoning. (4) *Application Layer* is established based on the three lower layers and provides domain-oriented IoT applications for end-users in various application domains.

One of the obviously properties of IoT is the creation of an unprecedented amount of data [2]. The massive amount of data flowing from the environment to the Internet is a side effect of the IoT. Surprisingly, how to control the huge amount of data injected into the network from the environment is a problem so far mostly neglected in the IoT research [4].

It is obviously that the IoT data flowing processing is a kind of big data problem. How to intelligently automate the process of colleting and aggregating context information on a large scale is a big challenge. The various challenges faced in big data management include scalability, unstructured data, accessibility, real time analytics, fault tolerance and many more [5]. The CEP is a feasible approach to solve the big data problem. It is a methodology and technique paradigm for collecting, aggregating and analyzing event flows of information about things that happen (event). The goal of CEP is to identify meaningful events pattern and respond to event subscriber as quickly as possible. CEP relies on a number of techniques, including [6]:

- Event-pattern detection
- Event abstraction
- Event filtering
- Event aggregation and transformation
- Modeling event hierarchies
- Detecting relationships (such as causality, membership or timing) between events
- Abstracting event-driven processes

However, in large-scale IoT applications, the current CEP technology encounters the challenge of massive distributed data which cannot be handled by most of the current methods efficiently [7]. In this paper, we focus on the issue of event aggregating and transforming in IoT big event flows.

The paper organized as follow. In Section II a group of definition about the basic conception of event and event operation will be illuminated formally, two key definitions are the Sequence and Invoke event operator which give a basic semantic model of event flows. We also define the CEP architecture and CEP model using the concept of event operation. Two implement techniques for dealing complex event with respect to IoT big event flows are discussed in the Section III. One is how to implement fast aggregating of complex events by Hadoop framework, another is fast pushing EPC data report service. Finally, in Section IV and Section V, we give related works and research conclusion.

II. An Algebra Model of CEP

Commonly, the CEP model has been classified into three types [8]. *1)Graph-based CEP Model.* Complex events are expressed in tree structure with its leaf nodes represent the primitive events. Once the corresponding event occurs, the

Manuscript received July 12, 2019; revised October 10, 2019. This work was supported in part by Natural Science Foundation of China (Grant No: 61672041, 61370051). It is also partly supported by Beijing Government and Education Committee (Grant No. PHR201107107).

Jing Sun and Huiqun Zhao are with Computer School, North China University of Technology, Beijing 100144 China (e-mail: sunjing8248@163.com, zhaohq6625@sina.com).

doi: 10.18178/ijmlc.2019.9.6.864

occurrence will be stored in the leaf node and pushed to its father node. *2)Automata-based CEP Model.* Event is constructed into corresponding automata, set of states and transition functions in automata-based model. Complex event is successfully detected if the automata reaches an accept state according to the event history. 3)Petri-net-based CEP Model. Employ Petri_net as formal notation, the input place represents the primitive event and output place indicates the occurrence of complex event.

In this section, we advise an algebra CEP model that use algebra notation to describe and deal with the event relationship and the event processing strategy.

A. Eevent and Events Composition

We initiate our discuss with basic event description.

Definition 2.1 (Basic Event). An Event is 7-tuple: <ID; Domain; Alias; Type; Time; Stimulation; Location> where:

1) **ID** is the identifier of an Event which differs from others.
2) **Domain** is **territory** where we discuss the problem.
3) **Alias** is an **event** name, for example an EPC.
4) **Type** is a **classification** identifying one of various types of event. The Type relates with its domain, all types consist of a Type set.
5) **Time** is the moment of an event **occurs**.
6) **Stimulation** is an act of **activities** set those may stimulate an event.
7) **Location** is place where an event is stimulated (or occurs).

In the following part of this paper, we use Dom(U) to denote the universal set of Event, Dom(A) to denote a domain from Event A, and $f(A)$ denote the item f of an Event A. For example, the item Type of Event A is expressed as Type(A). In this paper, Events are represented by uppercase letters (e.g. A; B; X; Y) and activities of Event are represented by lowercase letters (e.g. a; b; x; y). Activity x in a particular item f of a Event A is denoted as $x \in f(A)$, e.g. $x \in$ Activities(A). We also use Boolean values for activities: T when an activity is active and F when it is inactive.

Definition 2.2 (Event unequal and equal). Let A and B be two Events in domain(A) and domain(B). Both A and B are unequal, denoted as $A \neq B$, if anyone of the following five conditions are met:

1) Different Domain, denote to Domain(A)≠Domain(B)
2) Different Alias, denote to Alias(A)≠Alias(B)
3) Different Type, denote to Type(A)≠Type(B)
4) Different Stimulation, denote to Stimulation(A)≠ Stimulation(B)
5) Different Location, denote to Location(A)≠ Location(B)

Conversely, if all above five couple element of event are same, then say A equal B, denoted as $A=B$.

It is reasonable that the Time is different of two equivalent events.

Now we define Event composition operation. In essence, a composite is a combination of Event as the result of successively applying the connective operation.

Definition 2.3 (Sequence). Let A=<ID_A; $Domain_A$; $Alias_A$; $Type_A$; $Time_A$; $Stimulation_A$; $Location_A$>, B=<ID_B; $Domain_B$; $Alias_B$; $Type_B$; $Time_B$; $Stimulation_B$; $Location_B$>. If the following 2-conditions are met:

1) Time(A)<Time(B)
2) Location(A)=Location(B)

and construct a new event C that C=<$ID_{A\propto B}$; $Domain_{A=B}$; $Alias_{A\propto B}$; $Type_{A\vee B}$; $Time_A$; $Stimulation_{A\vee B}$; $Location_{A=B}$>, meanwhile both $ID_{A\propto B}$ and $Alias_{A\propto B}$ present new ID and new Alias respectively with respect to original ID and Alias; The $Type_{A\vee B}$ and $Stimulation_{A\vee B}$ are new set composed by the corresponding element from Event A or Event B respectively. The $Time_C$ is same with $Time_A$ because Event A emerges before Event B, The $Location_{A=B}$ can be coded by $Location_A$ or $Location_B$. The new Event C is result of both A and B sequence composite, denoted $C=A\propto B$. Conversely, say A latter than B. Especially, if $A=B$ then $C=A$ or $C=B$.

The A∝B is an abstract of two sequence event which two physical entity worn sensor device are captured in same place.

Theorem 2.1: the Sequence operation is associative, i.e. $(A\propto B)\propto C=A\propto (B\propto C)$.

Proof: Let A, B and C be three events, the following composition can be operated.

$(A\propto B)\propto C$=(<$ID_{A\propto B}$; $Domain_{A=B}$; $Alias_{A\propto B}$; $Type_{A\vee B}$; $Time_A$; $Stimulation_{A\vee B}$; $Location_{A=B}$>)$\propto C$

=$ID_{A\propto B\propto C}$;$Domain_{A=B=C}$; $Alias_{A\propto B\propto C}$; $Type_{A\vee B\vee C}$; $Time_A$; $Stimulation_{A\vee B\vee C}$; $Location_{A=B=C}$>

and A∝(B∝C)=<ID_A; $Domain_A$;$Alias_A$; $Type_A$; $Time_A$; $Stimulation_A$; $Location_A$>∝(<$ID_{B\propto C}$; $Domain_{B=C}$; $Alias_{B\propto C}$; $Type_{B\vee C}$; $Time_B$;$Stimulation_{B\vee C}$;$Location_{B=C}$>)

=<$ID_{A\propto B\propto C}$; $Domain_{A=B=C}$; $Alias_{A\propto B\propto C}$; $Type_{A\vee B\vee C}$; $Time_A$; $Stimulation_{A\vee B\vee C}$; $Location_{A=B=C}$>

The $(A\propto B)\propto C=A\propto(B\propto C)$ is met, therefore the theorem 2.1 is true.

Based on theorem 2.1 we can rewrite a $(A\propto B)\propto C$ to $A\propto B\propto C$. this 3-events algebra expression can be expended into m events.

Corollary 2.1 Let E_1, E_2, …, E_m be m Events, the $E_1\propto E_2\propto E_3\propto\ldots\propto E_m$ is still an Event.

Obviously, the corollary 2.1 is true.

Now we define sequential event flow.

Definition 2.4 (Sequence Flow). Let E_1, E_2, …, E_m be m events, if $E_1\propto E_2\propto E_3\propto\ldots\propto E_m$ is still an event then say the sequence event of E_1, E_2, …, E_m is a sequence flow.

Definition 2.5 (Invoke). Let A=<ID_A; $Domain_A$; $Alias_A$; $Type_A$; $Time_A$; $Stimulation_A$; $Location_A$>, B=<ID_B; $Domain_B$; $Alias_B$; $Type_B$; $Time_B$; $Stimulation_B$; $Location_B$>. If the following 3-conditions are met:

1) Domain(A)=Domain(B)
2) Type(A)=Type(B)
3) Time(A)<Time(B)

and construct a new Event C that C=<$ID_{A\rightarrow B}$;$Domain_{A=B}$; $Alias_{A\rightarrow B}$; $Type_{A=B}$; $Time_A$; $Stimulation_{A\vee B}$; $Location_{A\vee B}$>, meanwhile the $ID_{A\rightarrow B}$ and $Alias_{A\rightarrow B}$ present new ID and Alias respectively, The $Type_{A=B}$ can be coded by $Type_A$ or $Type_B$ and $Stimulation_{A\vee B}=\{s|\ s\in$Stimulation($A$) or Stimulation($B$)$\}$ and $Location_{A\vee B}=\{p|\ p\in$Locaion($A$) or Location($B$)$\}$, the $Time_C$ is same with $Time_A$ because Event A emerges before Event B. The new Event C is result of Event A invoke B, denoted $C=A\rightarrow B$.

The $A\rightarrow B$ is an abstract of two sequence event which a physical entity worn sensor device is captured in different place of same domain.

Theorem 2.2: the Invoke operation is associative, i.e. $(A\rightarrow B)\rightarrow C=A\rightarrow(B\rightarrow C)$.

Proof: it is similar with the theorem 2.1.

Corollary 2.2 Let E_1, E_2, …, E_m be m Events, the $E_1 \rightarrow E_2 \rightarrow E_3 \rightarrow \ldots \rightarrow E_m$ is still an Event.

We name the $< E_1, E_2, \ldots, E_m >$ is a invoke event flow.

Definition 2.6 (Cause-Effect Flow). Let E_1, E_2, …, E_m be m Events, if $E_1 \rightarrow E_2 \rightarrow E_3 \rightarrow \ldots \rightarrow E_m$ is still an Event then speak of E_1, E_2, …, E_m as cause-effect flow.

Definition 2.7 (synchronization). Let A=<ID_A; $Domain_A$; $Alias_A$; $Type_A$; $Time_A$; $Stimulation_A$; $Location_A$>, B=< ID_B; $Domain_B$; $Alias_B$; $Type_B$; $Time_B$; $Stimulation_B$; $Location_B$>, if the following 5-conditions are met:

1) $Alias(A) \neq Alias(B)$
2) $Type(A)=Type(B)$
3) $Time(A)=Time(B)$
4) $Location(A) \neq Location(B)$
5) $(a, b) \in$ Negotiation, $a \in Stimulation(A)$ and $b \in Stimulation(B)$,

and construct a new Event C=<$ID_{A\Theta B}$; $Domain_{A=B}$; $Alias_{A\Theta B}$; $Type_{A=B}$; $Time_{A=B}$; $Stimulation_{A\Theta B}$; $Location_{A\vee B}$>, meanwhile both $ID_{A\Theta B}$ and $Alias_{A\Theta B}$ present new ID and new Alias respectively with respect to original ID and Alias. The $Stimulation_{A\Theta B}$={$(a, b) \in$ Negotiation, $a \in Stimulation(A)$ and $b \in Stimulation(B)$}, $Location_{A\vee B}$={p| $p \in Location(A)$ or $Location(B)$}. The new Event C is result of A synchronously collaborating B, denoted $C=A\Theta B$.

The collaborative operator implies that two different events occur at exactly time on negotiation in different place. The Negotiation is set of internal activities, for example, (read, write) and (send, receive).

A related event operation is *parallel* composition which the Negotiation is null.

Definition 2.8 (asynchronization). Let A=<ID_A; $Domain_A$; $Alias_A$; $Type_A$; $Time_A$; $Stimulation_A$; $Location_A$>, B= <ID_B; $Domain_B$; $Alias_B$; $Type_B$; $Time_B$; $Stimulation_B$; $Location_B$>, if the following 5-conditions are met:

1) $Alias(A) \neq Alias(B)$
2) $Type(A)=Type(B)$
3) $Time(A) \neq Time(B)$
4) $Location(A) \neq Location(B)$
5) $(a,b) \in$ Negotiation, $a \in Stimulation(A)$ and $b \in Stimulation(B)$

and construct a new Event C=<$ID_{A\|B}$; $Domain_{A=B}$; $Alias_{A\|B}$; $Type_{A=B}$;$Time_{A\vee B}$; $Stimulation_{\varphi}$; $Location_{A\vee B}$>, meanwhile both $ID_{A\|B}$ and $Alias_{A\|B}$ present new ID and new Alias respectively with respect to original ID and Alias; the $Time_{A\vee B}$ and $Location_{A\vee B}$ are new set constructed by the corresponding element from event A and event B respectively. The new Event C is result of Event A asynchronously collaborating Event B, denoted $C=A\|B$.

Theorem 2.3: Both synchronous and asynchronous operation are associative individually, i.e. $(A\Theta B)\Theta C=A\Theta(B\Theta C)$, $(A\|B)\|C=A\|(B\|C)$.

Proof: it is similar with the theorem 2.1.

Theorem 2.4: Both synchronous and asynchronous operation are commutative individually, i.e. $A\Theta B=B\Theta A$, $A\|B=B\|A$).

Proof: Based on the definition 2.7 and 2.8 the theorem 2.4 is true.

Corollary 2.3 Let E_1, E_2, …, E_m be m Events, the both $E_1\Theta E_2\Theta E_3\Theta\ldots\Theta E_m$ and $E_1\|E_2\|E_3\|\ldots\|E_m$ are still an Event individually.

Theorem 2.5: The asynchronous operation is contributive with respect to invoke operation. i.e.: $(A\rightarrow B)\|C=A\|C\rightarrow B\|C$.

Proof: We only need to prove the two sides of the equation $(A\rightarrow B)\|C=A\|C\rightarrow B\|$C is established, i.e. satisfy the quality definition given in Definition 2.2. We only prove that the fourth of the five conditions is true.

Let A=<ID_A; $Domain_A$; $Alias_A$; $Type_A$; $Time_A$; $Stimulation_A$; $Location_A$>, B=<ID_B; $Domain_B$; $Alias_B$; $Type_B$; $Time_B$; $Stimulation_B$; $Location_B$> and C=<ID_c; $Domain_c$; $Alias_c$; $Type_c$; $Time_c$; $Stimulation_c$; $Location_c$> be three events in Dom(U).

$(A\rightarrow B)\|C$=<$ID_{A\rightarrow B}$; $Domain_{A=B}$; $Alias_{A\rightarrow B}$; $Type_{A=B}$; $Time_A$; $Stimulation_{A\vee B}$; $Location_{A\vee B}$>$\|C$=<$ID_{A\rightarrow B\|C}$; $Domain_{A=B=C}$; $Alias_{A=B\|C}$; $Type_{A=B=C}$; $Time_{A\vee C}$; $Stimulation_{(A\vee B)\vee C}$; Location $_{A\vee B\vee C}$> and

$A\|C\rightarrow B\|C$=< $ID_{A\|B}$; $Domain_{A=B}$; $Alias_{A\|B}$; $Type_{A=C}$; $Time_{A\vee C}$; $Stimulation_{A\vee C}$; $Location_{A\vee C}$>$\rightarrow B\|C$ = <$ID_{A\|C\rightarrow A\|B}$; $Domain_{A=C=B}$; $Alias_{A\|C\rightarrow A\|B}$; $Type_{A=C=B}$; $Time_{A\vee C}$; $Stimulation_{(A\vee C)\vee(B\vee C)}$; $Location_{A\vee B\vee C}$>.

See above two equation, the $Simulation_{(A\vee B)\vee C}$= $Simulation_{A\|C}$ or $Simulation_{B\|C}$ and $Stimulation_{(A\vee C)\vee(B\vee C)}$ =$Stimulation_{(A\|C)}$ or $Stimulation_{(B\|C)}$. Let $ID_{A\rightarrow B\|C}$=$ID_{A\|C\rightarrow A\|B}$; $Alias_{A\rightarrow B\|C}$=$Alias_{A\|C\rightarrow A\|B}$, hence $(A\rightarrow B)\|C=A\|C\rightarrow B\|C$.

It is feasible to let $ID_{A\rightarrow B\|C}$ same with $ID_{A\|C\rightarrow A\|B}$ and $Alias_{A\rightarrow B\|C}$ same with $Alias_{A\|C\rightarrow A\|B}$ because both of them are only a namely string.

B. An Algebra Model of CEP

Based on event composite operation we can define the CEP model.

Definition 2.9 (CEP Architecture). Let U be a domain. A CEP architecture is defined as:

1) An event itself is CEP architecture.
2) The result of applying a finite number composite operation of events is CEP architecture.

CEP architecture is denoted T=<E, O>, where E is set of events and O is a set of event operations.

It is able to prove that the events with respect to all composite operators construct an algebraic system.

Theorem 2.6 Let T=<E, O> be a CEP architecture, T is an algebraic system with respect to any event composite operation $OP_i \in O$.

Proof: This theorem follows the closure property of all composite operation.

In order to facilitate big data analysis, it is necessary to define the CEP architecture.

Definition 2.10 (Horizontal/Vertical CEP Model). Let U be a domain. A Vertical CEP model is defined as T=<E, O>, where the E=(A_1, A_2, … A_{m-1}, A_m) is a tuple of m sequence events flow, i.e. each A_k is sequence events flow, and O is a set of event operations. The Horizontal CEP Model is defined as T=<E, O>, where the E=(B_1, B_2, … B_{m-1}, B_m) is a tuple of m cause-effect events flow, i.e. each B_k is cause-effect events flow, and O is a set of event operations.

The CEP architecture has the Hierarchical properties. i.e. architecture is constructed by compositing events.

Theorem 2.7: Let E_1**=(**A_1, A_2, A_3…, A_m) be a Vertical CEP model, make an asynchronization composite $E_1\|E_1$=(($A_1\|\|A_{j1}$),…,($A_m\|\|A_{jm}$), ($A_{i1}\|\|A_1$), …, ($A_{im}\|\|A_m$)) = ($A_{i1}\|\|A_{j1}$, $A_{i2}\|\|A_{j2}$, …, $A_{ik}\|\|A_{jk}$). Let the $\angle(A_{ik}\|A_{jk})$= ($A_{i1}\|\|A_{j1}$, $A_{i2}\|\|A_{j2}$,…,

$A_{ik}||A_{jk}$), then both A_{ik} and A_{jk} must be couple of negotiated events.

Proof: consider the $E_1||E_1=(A_1||E_1),\ldots,(A_m||E_1)$, $(E_1||A_1),\ldots,(E_1||A_m)$). Since only the couple of negotiated events to be cared, other events can be cancelled such that $E_1||E_1=((A_1||A_{j1}),\ldots,(A_m||A_{jm}),(A_{i1}||A_1),\ldots,(A_{in}||A_m)$, $\angle(A_{ik}||A_{jk})$, where (A_{ik},A_{jk}) are couples of negotiated events. Again, all those negotiated event couples have already are expressed in $\angle(A_{ik}||A_{jk})$ therefore we can cancel out those remained expression so that the theorem 2.7 hold.

Corollary 2.4：If the couple of negotiated events $(A_{ik}||A_{jk})$ in theorem 2.7 are cause-effect event flow, i.e. $E_1=(A_1,A_2,A_3\ldots,A_m)$ be a Horizontal CEP model, make an asynchronization composite $E_1||E_1=((A_1||A_{j1})\rightarrow\ldots\rightarrow(A_m||A_{jm})\rightarrow(A_{i1}||A_1)\rightarrow\ldots\rightarrow(A_{im}||A_m))=(A_{i1}||A_{j1}\rightarrow A_{i2}||A_{j2}\rightarrow\ldots\rightarrow A_{ik}||A_{jk})$, let $\prod(A_{ik}||A_{jk})=(A_{i1}||A_{j1}\rightarrow A_{i2}||A_{j2}\rightarrow\ldots\rightarrow A_{ik}||A_{jk})$, then both A_{ik} and A_{jk} must be couple of negotiated events.

The theorem 2.7 and corollary 2.4 can help us to deduce a useful events pattern. The designer of CEP system can set negotiation conditions with respect to domain knowledge or user need. For example, all the events of cause-effect events flow are in transaction event which must hold the atomicity of the transactional event. The following algorithm show how to classify for big data flow based on theorem 2.7.

Algorithm 2.1 Classifying events of big data flow

Input: A vertical CEP model $E=(A_1, A_2, A_3,\ldots,A_m)$

Output: $\angle A_{ik}||A_{jk}$ where (A_{ik}, A_{jk}) is couple of negotiated events.

```
begin
    for each Ai∈(A1, A2, A3, …, Am)do // (A1, A2, A3,…,Am)is data structure of E
        for each Aj∈(A1, A2, A3, …, Am )do
            if (Stimulation(Aik),Stimulation(Ajk))∈Negotiation then
                Bk[i,j]= Aik||Ajk; //create upper level asynchronous event flow.
                Sum=Sum+1; // count the number of negotiated events
        end
    end
end.
```

In order to improve event aggregation and transformation, it is the intents of this paper, an algorithm for pushing event report is proposed.

Algorithm 2.2: Pushing event report

Input: Created cause-effect flow $E=(A_{i1}, A_{i2}, A_{i3},\ldots, A_{im})$ and a statistic report of events flow.

Output: $\angle A_{ik}||A_{jk}$ where (A_{ik}, A_{jk}) is couple of negotiated events

```
begin
    for each Aik do
        IPik=edit(Aik) //parse the Aik to get IP and interface of upper node of CEP model..
        for each IPik do
            push(IPik, Aik,report(Aik);
        end
    end
end
```

The time efficiency of Algorithm 5.1 is O(HL), where H is the length of the sequence event flow and L is the maximum length of the cause-effect event flows. The time efficiency of Algorithm 5.2 is $O(n2)$ where n is the number of elements in the set for which the linear dependency is considered.

III. Application of CEP Model

In this section we demonstrate two experiments results in which the reader may better understand the CEP algebra Model.

A. Backgroud

As standards organization, the EPCglobal published a framework for developing application of EPC(Electronic Product Code, EPC in short) network, which is a kind of IoT, in the year 2007 [9]. In order to leave flexibility to it user, he only gives a series of interface for implement. The ALE(Application Layer Event, ALE in short) [10] is one of his interface through which clients may obtain filtered consolidated EPC data from a variety of RFID(Radio Frequency Identification, RFID in short) reader. It's objective is that reduces the volume of EPC data that comes directly from RFID readers into coarser "events" of interest to applications. The processing done at this layer typically involves: (1) receiving EPCs from one or more RFID readers, (2) accumulating data over intervals of time, filtering to eliminate duplicate EPCs that are not of interest, and counting and grouping EPCs to reduce the volume of data; and (3) reporting in various forms such as XML, Database and so forth. The ALE model is showed as follow.

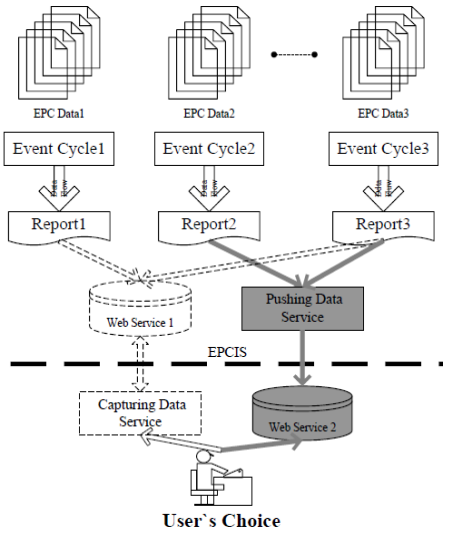


Fig. 1. Framework for processing IoT big data.

The framework showed in Fig. 1 meet the CEP algebra model perfectly, so that the algorithm 2.1 and algorithm 2.2 can guide us to accomplish Transactional Event Subscription. The transactional event is one of four event type of EPC-based IoT. It has same meaning with the concept of Data Base Transaction. Therefore, the ALE must collect and filter all sequence events flow emerged in each EPC reader for transactional event with the negotiation condition of Type (Transaction).

B. Transactional Event Subcription

In order to compare the performance of deal with huge

volume EPC data between Cluster and PC, two different experimental systems were established. The environment parameter of cluster system and PC are showed in Table I.

TABLE I: CONFIGURATION OF EXPERIMENT PLATFORM

Num	Type	CUP	System	Software
1	Cluster	2.4GHZ	Linux	Hadoop-1*
2	3.0GHZ+4Core	Windows	Self-made	

Without loss of generality, the EPC sgtin-96 and RFID IoT as experiment platform. The transactional event can be initiated as follow:

<ID=any string, Alias=sgtin-96, Domain=EPC IoT, Type=transaction, Simulation=read, Location=all point of reader>.

An EPC generator software is employed to create huge volume EPC Data in minute showed in the Table II.

TABEL II: COLLECTING AND FILTERING TIME

Num	Data volume(T)	Cluster ime(min)	PC Time(min)
1	200	1333	883
2	300	1433	1433
3	400	1550	1833
4	500	1633	2233
5	1024	3166	3700

The following program segment is main part of dealing with transactional events by Map_ Reduce function support by Apache Hadoop 0.23.10

```
public class EPClassifyMapper extends Mapper< LongWritable,Text,
Text ,IntWritable>{
...
private FilterPattern pattern;
...
public class EPClassifyMapper extends Mapper< LongWritable,Text,
Text ,IntWritable>{
...
private FilterPattern pattern;
...
protected void map(LongWritable key, Text values, Context context)
throws IOException, InterruptedException {
   .... // initial code
  if(pattern.isGroup())
{ String temp = regex;
    for(int index: pattern.getIndexs())
{if(index != -1){regexs[index] = strs[index]; }}
    StringBuffer buffer = new StringBuffer();
    for(int i=0;i<regexs.length;i++)
{buffer.append(regexs[i]);
     if(pattern.match(str))
{value.set(1);
      context.write(writeKey,value);}}}
catch(ArrayIndexOutOfBoundsException e){...}
public class EPClassifyReducer extends Reducer<Text, IntWritable,
Text, IntWritable>{
// initial code
protected void reduce(Text key, Iterable<IntWritable> values,Context
context) throws IOException, InterruptedException {
  int count = 0;
  for(IntWritable n : values)
{count += n.get();   }
    value.set(count);
    context.write(key, value);}}
```

Fig. 2. Map_Reduce code of collecting EPC pattern.

where the EPC Pattern refer to the following:

urn:epc:pat:*type_i*:*field_i_1.field_i_2.field_i_3*... .

Meanwhile the *field_i_j* take an item from *, digital, [lo-hi] and X. the pat takes the value sgtin-96[11]. For example, urn:epc:pat: sgtin-96:3.*.*.[5000-9999].

Fig. 3 is corresponding graph of Table II. Obviously, the cluster system consume less time than PC when the data volume is over 300G.

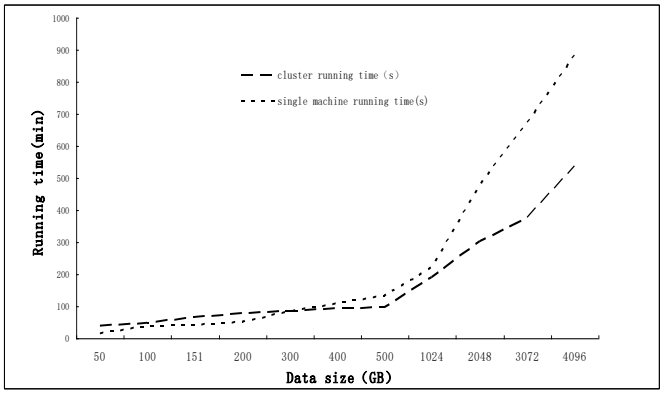


Fig. 3. Collecting and filtering time of cluster and PC.

C. Pushing EPC Data Report Service

In this section we demonstrate the pushing service have better performance than getting service based on algorithm 2.

The Web Service is a popular technical method that provides communications between two electronic devices over the World Wide Web. It is a software function provided at a network address over the web with the service always on as in the concept of utility computing [12].

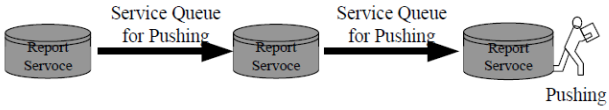


Fig. 4. Service queue for pushing.

It can identify two major classes of Web services:

- REST(Representational state transfer (REST)) compliant Web services,
- Arbitrary Web services, in which the service may expose an arbitrary set of operations.

In this paper a new WSDL schema was established in which bidirectional web service request can be described. We use a string to mark the direction in WSDL schema. The "callee" presents provider of web resources, and the "caller" presents applier. The "callee" act as original meaning of Web Service and "caller" is new Web Service pattern.

For the reason of limited pages, we only demonstrate the evidence of new pushing service pattern (demonstrated on Fig. 4) has a better performance than the old getting service pattern.

First we focus on the waiting time. It has been proved that the time of a queue waiting for service meet exponential distribution $F(t)=1-e^{-\lambda t}$, and pushing service pattern spends less residents time than getting service pattern, that means the value of λ is more larger in pushing pattern than getting pattern with respect to the meaning of $\lambda(\lambda=n\times p$, i.e. the total experimental times product the probability for sample). Therefore, we can compare their performance on λ=0.5, 0.6 for getting service pattern and λ=1, 2, 3 for pushing service with time $t\in$[0, 12].

Another performance aspect is service times. It also has been proved that the service times for a queue meet Poisson

Distribution and pushing service pattern has lager value of λ than getting service pattern with respect to the meaning of λ. Therefore, we can compare their performance on λ=0.5, 0.6 for getting service pattern and λ=1, 2, 3 for pushing service with constant times and stochastic times at constant time, i.e. t=12, c=12, λ=5, 10, 20 and n=10, 20.

Fig. 5 indicates that the rate of push service is high, and the time is relatively small, and the Fig. 6 show that the number of services has increased. Both the Fig. 5 and the Fig. 6 show the pushing service pattern has better performance than the getting service pattern.

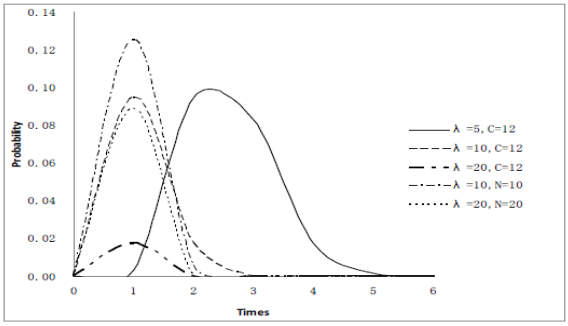


Fig. 6. Different performance in service times.

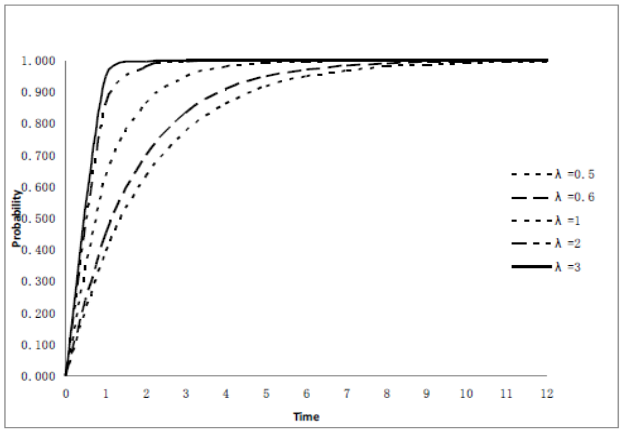


Fig. 5. Difference in waiting time.

IV. Related Work

The Internet of Things have been experienced tremendously fast development during the past decade whither in theory or practice. Jayavardhana Gubbi, *et al*. [2] given a vision on architecture, future developing directions of IoT. There are four application directions or domains: Personal and Home, Enterprise, Utilities and Mobile are categorized in his paper. As a side effect of IoT application, the Jayavardhana Gubbi also pointed out one of most important outcomes is the creation of an unprecedented amount of data, the technique for collecting, storing and processing come up against numerous challenges. Meng Ma, *et al*. [3] pay a attention to the big data processing technology of IoT, and suggest a layered architecture of IoT which is consisted of four layers, i.e. *Sensing Layer*, *Network Layer*, *Middleware Layer*, *Application Layer*. C. C. Aggarwal, *et al*. [13] described the IoT as massive real-time data sensor: "IoT is designed to connect thousands of objects in large scale. Communications between different entities in dynamic networks generate a large volume of heterogeneous data in the form of real-time, high-speed, uninterrupted data flows. Scalable storage, filtering and compression schemes are essential for efficient big data processing." In addition, the related observation and review can see such as [14]-[17].

The CEP is a prospective approach to IoT massive real-time data processing. Meng Ma, *et al*. [3] classify the CEP model for IoT event processing into four type, (1) Graph-based CEP Model. In this model, complex events are expressed in tree structure. (2) Automata-based CEP Model. All events are constructed into corresponding automata, set of states and transition functions in automata-based model. (3) *Petri-ne*t-based CEP Model. Complex events are transformed into corresponding *Petri_net* in *Petri-net*-based CEP model. Y.H. Wang, *et al*. [18] argues that the CEP technology encounters the challenge of massive distributed data because of weak processing ability by most of the current methods efficiently. A high-performance complex event processing method over distributed probabilistic event flows is proposed against the big challenge. In this paper a contributed method uses probabilistic nondeterministic finite automaton and active instance stacks to process a complex event in both single and distributed probabilistic event flows. A parallel algorithm is designed to improve the performance. However, the authors only demonstrate the high level patterns for deal with IoT event instead of contributing any confident technology and the experiment results only base on simulation software, so that the contribution is discounted. Nicholas Poul, *et al*. [19] propose a novel approach to high-level patterns of event match in CEP in contrast with database management systems (DBMSs in short). The new method can specify queries as high-level patterns of event, and if the CEP system detects complex events matching these queries the system notices clients of matches in soft real time. Omran Saleh, *et al*. [20] consider that most of the conducted works in sensor network field avert to address the wider issues of how to integrate CEP with in-network processing. The author employs the algebraic method to support sampling, filtering and preprocessing events, such as disjunction, sequence, conjunction and some forms of negation operator. All above so called algebraic operator only carry out common data arrangement and week support for event pattern mining. Analogous research works published in [21], [22].

As a new research area the big data processing has became a hot topic. Weizhong Yan, *et al*. [23] consider the Power Iteration Clustering(PIC in short) is not enough suitable to Big Data Processing, and expand PIC's data scalability by implementing a parallel power iteration clustering (p-PIC). Aditya B. Patel [24] reports the experimental work on big data problem and its optimal solution using Hadoop cluster, Hadoop Distributed File System (HDFS) for storage and using parallel processing to process large data sets using Map Reduce programming framework. Wen Xiong [25] emphasize the priority of benchmarking big data and discover much more redundancy existed in these pioneering benchmark suites and give three findings: how to remove redundancy safely, input data sets for data analysis and benchmarks can be used as academic research. Cristina L. Abad [26] employ Markov renewal process model for evaluating performance of Big Data processing, and provide two kind of algorithm for creating two workflow traces, one

for popularity and another for temporal locality. Analogous report and review published in [27], [28].

V. Conclusion

In this paper, we propose an algebraic model for CEP and demonstrate its application in IoT Big Data processing. Our contributions can be concluded as follow:

1) Proposed a mathematic CEP model that can facilitate the pre-analysis and design for Big Data processing algorithm. The contributed Horizontal/Vertical CEP Model considers event flow with sequence and cause-effect two direction, so that a complex event relationship can be described clearly. The experimental results have proved that the CEP model can support to collect and filter IoT Big Data flow.
2) In contrast with the traditional Web service, the contributed pushing service pattern can send event report to subscriber actively instead of waiting for visitor which the Web Service does. We also demonstrate our novel technique for processing Big Data flow by using Apache Hadoop, and the experiment result show that it is suitable for the CEP model.

Algebraic approach to CEP is far more complex than what a single model can describe. There are many more properties of CEP architecture that need further exploration. Here are the specific problems for our future research:

1) To implement a tool for supporting the CEP automatic analysis, involved defined event pattern filtering algorithms and the complex event report generating algorithms.
2) To apply to a large number of real cases to support and improve the approach of algebraic modeling proposed in this paper.

We believe that IoT Big Data processing system design should and can be specified and modeled with a sound theoretical framework. This paper is an attempt toward this direction.

References

[1] K. Ashton. That 'Internet of Things' thing. *RFiD Journal*. [Online]. Available: http://www.rfidjournal.com/articles/view?4986

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*. vol. 29, no. 7, pp. 1645-1660, 2013.

[3] M. Ma, P. Wang, and C. H. Chu, "Data management for internet of things: Challenges, approaches and opportunities," *GreenCom-iThings-CPSCom.2013*, pp. 1144-1151, Aug. 20-23, 2013.

[4] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, pp. 1497-1516, 2012.

[5] C. Dobre and F. Xhafa, "Intelligent services for big data science," *Future Generation Computer Systems*, 2014.

[6] O. Etzion and P. Niblett, *Event Processing in Action*, *Manning Publications*, Shelter Island New York U.S, August 2010.

[7] Y. H. Wang, K. Cao, and X. M. Zhang, "Complex event processing over distributed probabilistic event flows," *Computers and Mathematics with Applications*, vol. 66, pp. 1808-1821, 2013.

[8] M. Ma, P. Wang, and C. H. Chu, "Data management for internet of things: Challenges," *Approaches and Opportunities*, pp. 1144-1151, 2013.

[9] EPCglobal. (2010). Architecture framework final version. [Online]. Available: http://www.gs1.org/gsmp/kc/epcglobal/architecture/architecture_1_4-framework-20101215.pdf

[10] EPCglobal. (2009). The Application Level Events (ALE) Specification, Version 1.1.1. [Online]. Available: http://www.gs1.org/gsmp/kc/epcglobal/ale/ale_1_1_1-standard-core-20090313.pdf

[11] W3C. (2004). Web Services Glossary. W3C. [Online]. Available: http://www.w3.org/TR/ws-gloss/

[12] W3C. Web Services Architecture. W3C. [Online]. Available: http://www.w3.org/TR/ ws-arch/

[13] C. C. Aggarwal, N. Ashish, and A. Sheth, "The internet of things: A survey from the data-centric perspective," *Managing and Mining Sensor Data*, pp. 383-428, 2013.

[14] M. Díaz, G. Juan, O. Lucas, and A. Ryuga, "Big data on the internet of things," *IMIS*, Palermo, Italy, pp. 897-900, 2012.

[15] D. Tracey and C. Sreenan, "A Holistic architecture for the internet of things, sensing services and big data," in *Proc. 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, Delft, Netherland, 2013, pp. 546-553.

[16] Z. Ding, Xu Gao, Jiajie Xu, and Hong Wu, "IOT-StatisticDB: A general statistical database cluster mechanism for big data analysis in the internet of things," *iThings/CPSCom2013*, Beijing, China, pp. 535-54, Aug. 2013.

[17] C. G. Wang, M. Daneshmand *et al*., "Special Issue on Internet of Things (IoT): Architecture, protocols and services," *IEEE Sensors Journal*, vol. 10, no. 13, pp. 3505-3510, October 2013.

[18] Y. H. Wang, K. Cao, and X. M. Zhang, "Complex event processing over distributed probabilistic event flows," *Computers and Mathematics with Applications*, vol. 66, pp. 1808–1821, 2013.

[19] N. Poul, M. Migliavacca, and P. Pietzuch, *Distributed Complex Event Processing with Query Rewriting*, Nashville, Tennessee, USA, July 6-9, 2009.

[20] O. Saleh and K.-U. Sattler, *Distributed Complex Event Processing in Sensor Networks*, Milan Italy, pp. 23-26, June 2013.

[21] E. Ogasawara, J. Dias, D. Oliveira, F. Porto, P. Valduriez, and M. Mattoso, "An algebraic approach for data-centric scientific workflows," in *Proc. VLDB Endowment*, 2011, vol. 4, no. 12, pp. 1328–1339.

[22] J. Dias, E. Ogasawara, D. de Oliveira, F. Porto *et al*., *Algebraic Dataflows for Big Data Analysis*, pp. 150-155, October 8, 2013.

[23] W. Z. Yan, U. Brahmakshatriya, Y. Xue, M. Gilder, and B. Wise. "p-PIC: Parallel power iteration clustering for big data," *Journal of Parallel Distributed Computing*, vol. 73, pp. 352-359, 2013.

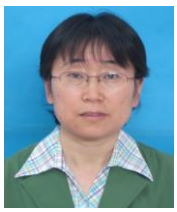
[24] A. B. Patel, M. Birla, and U. Nair, *Addressing Big Data Problem Using Hadoop and Map Reduce*, pp. 1-5, December 2012.

[25] W. Xiong, Z. B. Yu, Z. D. Bei *et al*., *A Characterization of Big Data Benchmarks*, pp. 118-125, October 2013.

[26] C. L. Abad, M. Yuan, C. X. Cai *et al*., "Generating request flows on Big Data using clustered renewal processes," *Performance Evaluation*, vol. 70, pp. 704-719, 2013.

[27] Z. B. Zheng, J. M. Zhu, and M. R. Lyu, *Service-Generated Big Data and Big Data-as-a-SERVICE: An Overview*, Santa Clara CA USA, pp. 403-410, 2013.

[28] S. Sagiroglu and D. Sinanc, *Big Data: A Review*, CTS 2013:42-47, San Diego, CA, USA, May 20-24, 2013.



Jing Sun was born in January 1968 in Tieling City, Liaoning Province, China. From September 1986 to July 7, he graduated from the Mathematics Department of Liaoning University with a bachelor of science degree. From September 1998 to July 2000, he graduated from the Department of Computer Science and Technology of Liaoning University with a master of science degree.

Her research interests are big data analytics, internet of things technology.

Huiqun Zhao was born in September 1960 in Anshan City, Liaoning Province, China. From September 1979 to July 1983, he obtained a bachelor of science degree in mathematics from Jinzhou Normal University. From September 1998 to July 2002, he studied in the Department of Computer Science and Engineering of Northeastern University and obtained a doctorate in engineering.

His research interests are big data analytics, IoT technology.