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 is proposed 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 have 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 collecting 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
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. Event and Events Composition

We initiate our discussion 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∈f(A), e.g. x∈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≠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∪B}; Domain_{A∪B}; Alias_{A∪B}; Type_{A∪B}; Time_{A∪B}; Stimulation_{A∪B}; Location_{A∪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 Type_{A∪B} and Stimulation_{A∪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×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×B)×C=A×(B×C).

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

(A×B)×C=(<ID_{A∪B}; Domain_{A∪B}; Alias_{A∪B}; Type_{A∪B}; Time_{A∪B}; Stimulation_{A∪B}; Location_{A∪B}>×<ID_{B∪C}; Domain_{B∪C}; Alias_{B∪C}; Type_{B∪C}; Time_{B∪C}; Stimulation_{B∪C}; Location_{B∪C}>)

where we discuss the problem.

**Corollary 2.1** Let E_1, E_2, ..., E_m be m Events, the E_1×E_2×E_3×...×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×E_2×E_3×...×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∪B}; Domain_{A∪B}; Alias_{A∪B}; Type_{A∪B}; Time_{A∪B}; Stimulation_{A∪B}; Location_{A∪B}>. Meanwhile the ID_{A∪B} and Alias_{A∪B} present new ID and Alias respectively. The Type_{A∪B} can be coded by Type_A or Type_B and Stimulation_{A∪B}=<|s| s∈Stimulation(A) or Stimulation(B)> and Location_{A∪B}=<|p| p∈Location(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→B.

The A→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→B)→C=A→(B→C).

Proof: it is similar with the theorem 2.1.
Corollary 2.2 Let \( E_1, E_2, \ldots, E_m \) 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 \( E_1, E_2, \ldots, E_m \) be \( m \) Events, if \( E_1 \rightarrow E_2 \rightarrow E_3 \rightarrow \ldots \rightarrow E_m \) is still an Event then speak of \( E_1, E_2, \ldots, E_m \) as cause-effect flow.

**Definition 2.7 (synchronization).** Let \( A = \langle IDA; \text{Domain}_A; \text{Alias}_A; \text{Type}_A; \text{Time}_A; \text{Stimulation}_A; \text{Location}_A \rangle, B = \langle IDB; \text{Domain}_B; \text{Alias}_B; \text{Type}_B; \text{Time}_B; \text{Stimulation}_B; \text{Location}_B \rangle \), if the following 5-conditions are met:

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

and construct a new Event \( C = \langle IDA\_A; \text{Domain}_A-B; \text{Alias}_A-B; \text{Type}_A-B; \text{Time}_A-B; \text{Stimulation}_A-B; \text{Location}_A-B \rangle \), meanwhile both \( IDA\_A-B \) and \( \text{Alias}_A-B \) present new ID and new Alias respectively with respect to original ID and Alias. The \( \text{Stimulation}_A-B \) is a set of event operations. The result of applying a finite number composite operation is \( \text{CEP architecture} \).

Theorem 2.3: Both synchronous and asynchronous operation are associative individually, i.e. \( \text{CEP architecture} = \Theta(\text{CEP architecture}) \), \( |A|[B]|C| = |A||B||C| \).

**Proof:** it is similar with the theorem 2.1.

**Theorem 2.4:** Both synchronous and asynchronous operation are commutative individually, i.e. \( \text{CEP architecture} = \Theta(\text{CEP architecture}) \).

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

Corollary 2.3 Let \( E_1, E_2, \ldots, E_m \) be \( m \) Events, the both \( E_1 \rightarrow E_2 \rightarrow E_3 \rightarrow \ldots \rightarrow E_m \) 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. \( (A \rightarrow B))|C = |A||C = |B ||C | \).

**Proof:** We only need to prove the two sides of the equation \( (A \rightarrow B))|C = |A||C = |B ||C | \) 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 \( A = \langle IDA; \text{Domain}_A; \text{Alias}_A; \text{Type}_A; \text{Time}_A; \text{Stimulation}_A; \text{Location}_A \rangle, B = \langle IDB; \text{Domain}_B; \text{Alias}_B; \text{Type}_B; \text{Time}_B; \text{Stimulation}_B; \text{Location}_B \rangle \) and \( \text{Location}_A \neq \text{Location}_B \), if \( A \rightarrow B \) and \( A \rightarrow C \) are result of \( A \) synchronously collaborating \( B \) or \( C \). It is feasible to le...
A_d||A_b, then both A_d and A_b must be couple of negotiated events.

Proof: consider the E_1||E_2||(A_1||A_2), then the couple of negotiated events to be cared, other events can be cancelled such that E_1||E_2||(A_1||A_2)||A_m = (A_1||A_2)||A_m, \( \angle(A_d||A_b) \), where \((A_d,A_b)\) are couples of negotiated events. Again, all those negotiated event couples have already are expressed in \( \angle(A_d||A_b) \), therefore we can cancel out those remained expression so that the theorem 2.7 hold.

Corollary 2.4: If the couple of negotiated events \((A_d||A_b)\) in theorem 2.7 are cause-effect event flow, i.e. E_1=(A_1,A_2,A_3,...,A_m) be a Horizontal CEP model, make an asynchronous composite E_1||E_2||(A_1||A_2)||...||(A_m||A_m) \( \rightarrow \) (A_1||A_2)||...||(A_m||A_m) \( \rightarrow \) (A_1||A_1) \( \rightarrow \) A_1 \( \rightarrow \) A_2 \( \rightarrow \)... \( \rightarrow \) A_d \( \rightarrow \) A_d, let \( \Pi(A_d||A_b) = (A_1||A_1) \rightarrow A_2 \rightarrow A_2 \rightarrow...\rightarrow A_d \rightarrow A_b \), then both A_d and A_b 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 \( E=(A_1,A_2,A_3,...,A_m) \)
- **Output:** \( \angle(A_d||A_b) \) where \((A_d,A_b)\) is couple of negotiated events.

begin
for each \( A_i \in \{A_1, A_2, A_3, \ldots, A_m\} \) do // \( A_1, A_2, A_3, \ldots, A_m \) is data structure of \( E \)
    for each \( A_j \in \{A_1, A_2, A_3, \ldots, A_m\} \) do
        if \( \text{(Stimulation}(A_d),\text{Stimulation}(A_b)) \in \text{Negotiation} \) on then
            \( B_i[j]=A_d||A_b \); //create upper level asynchronous event flow.
            \( \text{Sum} = \text{Sum} + 1 \); // 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 \( E=(A_1,A_2,A_3,...,A_m) \) and a statistic report of events flow.
- **Output:** \( \angle(A_d||A_b) \) where \((A_d,A_b)\) is couple of negotiated events.

begin
    for each \( A_d \) do
        \( IP_d = \text{edit}(A_d) \); //parse the \( A_d \) to get IP and interface of upper node of CEP model.
        for each \( IP_a \) do
            \( \text{push}(IP_d, A_d, \text{report}(A_d)) \);
        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 \( O(n^2) \) 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. Background

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.](image)

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 Subscription

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>
<thead>
<tr>
<th>Num</th>
<th>Type</th>
<th>CPU</th>
<th>System</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cluster</td>
<td>2.4GHZ Linux</td>
<td>Hadoop-1*</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>3.0GHZ+4Core</td>
<td>Windows</td>
<td>Self-made</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Num</th>
<th>Data volume(T)</th>
<th>Cluster ime(min)</th>
<th>PC Time(min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>1333</td>
<td>883</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>1433</td>
<td>1433</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>1550</td>
<td>1833</td>
</tr>
<tr>
<td>4</td>
<td>500</td>
<td>1633</td>
<td>2233</td>
</tr>
<tr>
<td>5</td>
<td>1024</td>
<td>3166</td>
<td>3700</td>
</tr>
</tbody>
</table>

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

```java
public class EPClassifyMapper extends Mapper<LongWritable, Text, IntWritable> {
    private FilterPattern pattern;

    public class EPClassifyMapper extends Mapper<LongWritable, Text, IntWritable> {
        ... // initial code
        protected void map(LongWritable key, Text values, Context context) {
            // ... // initial code
            if(pattern.isGroup()) {
                String temp = regex;
                for(int i=0;i<regex.length;i++) {
                    if(regex[i].equals(pattern[index])) {
                        StringBuffer buffer = new StringBuffer();
                        for (int i=0;i<regex.length;i++) {
                            if (regex[i].equals(pattern[index])) {
                                buffer.append(regex[i]);
                                if (pattern[i].indexOf(str) != -1) {
                                    context.write(key, value);
                                }
                            }
                        }
                    }
                }
            }
        }
    }

    public class EPClassifyReducer extends Reducer<Text, IntWritable, IntWritable> {
        protected void reduce(Text key, Iterator<IntWritable> values, Context context) {
            ... // initial code
            if(key.equals(pattern)) {
                context.write(key, value);
            }
        }
    }
```

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.

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].

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 applicer. 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 \( \lambda \) is more larger in pushing pattern than getting pattern with respect to the meaning of \( \lambda = n \pi p \), i.e. the total experimental times product the probability for sample). Therefore, we can compare their performance on \( \lambda = 0.5, 0.6 \) for getting service pattern and \( \lambda = 1, 2, 3 \) 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 larger value of $\lambda$ than getting service pattern with respect to the meaning of $\lambda$. Therefore, we can compare their performance on $\lambda=0.5, 0.6$ for getting service pattern and $\lambda=1, 2, 3$ for pushing service with constant times and stochastic times at constant time, i.e. $t=12, c=12, \lambda=5, 10, 20$ 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.

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 the 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 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 notifies 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 weak support for event pattern mining. Analogous research works published in [21, 22].

As a new research area the big data processing has become 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 mathemathic 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


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.

Huiqian Zhao was born in September 1960 in Anshan City, Liaoning Province, China. From September 1979 to July 1983, he graduated from the Mathematics Department of Liaoning University with a bachelor of science degree. 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.