Detecting Dataraces for BPEL-based Services using Weighted Vector Clock

Xuehong YANG†, Junfei HUANG, Yunzhan GONG

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract

BPEL is the most popular business process specification language, so it is crucial to ensure the quality and reliability of BPEL-based services. The dead-path elimination semantic determines that the datarace detection must adopt a new technique. Timestamps notation is an effective way to solve unexpected execution like determining the potential causality among events in concurrent programs, so in this paper we focus on the timestamps which can be used to analyze the data races among activities. The vector clock is used to judge whether BPEL activity pairs is concurrent or not. And the weighted value of event is used to improve the accuracy of the data races detection for BPEL process which consists of link semantic (transition condition and join condition) and path condition, the weighted value is calculated using abstract interpretation. Finally, experimental results show the efficiency of this approach.

Keywords: Web Services; BPEL; Web Services Composition; Vector Clock; Data Races

1. Introduction

Service composition is one promising application in service-oriented architecture (SOA). Unfortunately, we still lack complete understanding of how to analyze and test the service composition systems, even if substantial research efforts have been made on this topic. Testing thoroughly a service composition remains to be a challenge. A composted web service consists of a set of web services that cooperated with each other to achieve a common goal which satisfies customers’ demands or gets a value-added service. The main characteristic of the composition is that the services communicate only by exchanging messages over a communication network such as SOAP which may with unpredictable (but finite) transmission delays—BPEL[1] supports long-running transactions[9]. So execute actually is not feasible for testing of BPEL-based services. Concurrency and synchronization make BPEL process suffers from time-dependent data races due to the erroneous use of flow and link structure.

For concurrent system, time related information is often an important factor. Vector clock has several interesting properties [2, 3], for example, its mathematical structure is similar to Minkoski’s relativistic space-time in the sense that captures the causal histories [4]. The concurrent and distributed natures make BPEL-based services difficult to understand and test. During the execution of a BPEL process, a concurrent activity has no knowledge about other activities, and because of the impacts of dead-path elimination (DPE)
semantic, it makes the detection of data races difficult. S. Chen [14] proposed a static detection which is based on graph theory, but it cannot deal with the concurrent activity nested in branch activity. J.Cui [13] designs an automatic tool for detecting BPEL data races, but it omits the link condition. So Li [15] proposes a method combines happens-before and Lockset for static data races detection, but it does not consider path condition and DPE which may make the concurrent activity dead. Just because of the imprecision of static analysis, in order to detect data races among concurrent activities, this paper proposes a new datarace detection method which extends the traditional vector clock mechanism proposed “weighted vector clock”, the weight value is related to the variables in BPEL process, its abstract domain is derived based on abstract interpretation. This method not only considers happen-before relationship among activities, but also path condition and DPE, which can resolve the imprecision of the above methods.

2. Basic Knowledge Review

2.1. BPEL Language

The commands of the BPEL language are called “activity”, i.e. the “activity” is the basic constructing unit of the BPEL. Activities can be partitioned into two groups: basic activities and structure activities which can also be nested in any levels. The whole structure of a process contains the following four sections:

- Declaration of the relationships to external partners according to the attributes of role.
- Declaration of the variables and correlation sets used by the process.
- Declaration of handlers used by the process.
- Declaration of the main body of the process behavior.

Dead-path elimination (DPE) is a typical feature of BPEL language, when the suppressJoinFailure attribute of the process or activity is set to ‘yes’, and then DPE can be enabled. Each BPEL activity may have some incoming and/or outgoing links; each link is associated with a transition condition. If no explicit transition condition is specified, the default value is true. The activity in BPEL can also correlate a join condition which is a Boolean expression over the status (true, false, unset) of the incoming links of the activity. Once all the incoming links have get status (true or false) value, the join condition can be evaluated. If the join condition evaluates to true, the activity is executed. Otherwise it can not executed and all the status of its outgoing links are set to false, this situation will propagate downstream until a true join condition is reached, then the corresponding activity can execute. This is called “dead-path elimination”. Here, if the activity or link is not executed because of the DPE, it is called “dead”.

2.2. Weighted Vector Clock

A vector clock (VC) system is a timestamp mechanism that exactly tracks causality among events produced by a distributed computation which was introduced in 1988. It was development as an extension to Lamport’ time to handle complex interactions in concurrent system—vector clock as an algorithm is used to generate the partial ordering of events and detect causality violations [5], including datarace.

In concurrent program, one particularly hard bug to detect is ‘datarace’. A datarace occurs when the programmer does not correctly synchronize the access to a variable which is being manipulated by more than one thread. This can leave the variable in an unexpected or inconsistent state [7]. In BPEL-based web services, the activity may be dead because of dead-path elimination, so “weighted vector clock” (WVC)
which adds a weighted value to vector clock is proposed. WVC is defined as a 2-tuple \( WVC = \langle vc, w \rangle \) where \( vc \) is a vector clock with a dimension equal to the maximum degree of parallelism during execution, \( w \) is the weighted value on the edges of events which is the condition of event. By assigning a vector of integers \( vc[1 \ldots n] \) to each process \( P_\alpha \), the properties of \( WVC.vc \) are listed following: For two events \( e \) and \( f \), \( e \in P_\alpha, f \in P_\beta \),

1. If \( e \) and \( f \) has precedent relation, then the following condition is true:
   \[
   e \rightarrow f \Leftrightarrow vc_i(e)[i] \leq vc_j(e)[i]
   \]
2. If \( e \) and \( f \) happens concurrently, noted by \( e \parallel f \), the following condition is true:
   \[
   e \parallel f \Leftrightarrow \begin{cases} 
   vc_j(f)[i] < vc_i(f)[i] \\ 
   vc_j(e)[j] < vc_j(f)[j]
   \end{cases}
   \]

2.3. Weighted Temporal Constraint Diagram

The “happen-before” relationship can be used in the dynamic testing which can compare the causal relation among events [8]. With this idea in mind, we extend it to detect data races in BPEL-based services statically; “static” means that we collect the execution information of process without run any instances. BPEL uses link construct to express the control dependency relationship among concurrent activities which is like the fork/join relation in Java. Since BPEL cannot create variable and activity dynamically, the read/write variable and all concurrent activities can be calculated statically, so we can statically collect the temporal relations between the activities which have deep effects to improve the accuracy of static analysis.

In order to detect data races in BPEL-based services with the consideration of DPE and path condition, the new mechanism “weighted vector clock” (WVC) is used to model the relationships among activities. The weighted temporal constraint diagram is constructed based on the BPEL control flow graph which is not explained detail in this paper.

The weighted temporal constraint graph (WTCG) in figure 1 depicts a BPEL-based service process with three direct sub-processes (you can refer to the process in figure 3). The dotted line indicates the synchronization relation, and the associated vectors assigned to each event illustrate the execution history. From vector clock, people can determine causal relationships of activities; some researches concluded that vector time is a “knowledge-based” interpretation [10]. For example, if \( V(e) \) is the vector timestamp of an event occurring in \( P_\alpha \), then \( V(e)[k] \) is the number clock of events in \( P_\alpha \) which \( e \) “knows about”, where “\( x \) knows about \( y \)” is synonymous to “\( y \) is in the causal history of \( x \)”. But because of the path condition and
link semantic may affect the execution of event in BPEL process, some of events cannot occur simultaneously, such as event invokeVIP and assignDiscount in figure 1, because sum>=1000 and sum<1000 are conflict.

3. BPEL Extended Clock Modeling

3.1. Visualization Modeling for BPEL

A weighted vector clock (WVC) system is a mechanism that associates timestamps and weighted value with events, by comparing the properties of events we can know whether those events are causally related (or which one comes first). BPEL is a concurrent language, what is important is usually not when activities happen but in which order they happen. If two events are not executed properly, there may be a datarace.

**Definition 1.** A process execution, P, is a triple <E, T ⃗→, WVC>, where E is a finite set of events, T ⃗→ is an irreflexive partial order on E called the causal relationship, and WVC is the extended vector clock properties of events. The causal relation T ⃗→ ⊆ E × E is the smallest transitive relation satisfying:

1. a, b ∈ E and in the same sub-process P if WVC(a).vc<=WVC(b).vc and WVC(a).w ∧ WVC(b).w=true, then a T ⃗→ b, i.e. a occurred before b. Here WVC(a).vc is the vector clock value, WVC(a).w is the execution condition of event, i.e., the weighted value.
2. If a ∈ E is the source event of synchronization link L’ in a sub-process and b ∈ E is the target event of synchronization of the same link L’ by another sub-process, and also WVC(a).w ∧ WVC(b).w=true, then a T ⃗→ b.
3. If a T ⃗→ b and b T ⃗→ c, then a T ⃗→ c, i.e. causal relations are transitive.

Note T ⃗→ is non-reflexive, symmetric and transitive.

**Definition 2:** the concurrency relation || ⊆ E × E satisfies the following condition which is not transitive. a||b iff ¬(a T ⃗→ b) ∧ (b T ⃗→ a)

3.2. Weighted Value (OCURRENCE CONDITION) of Event

In BPEL, besides the loops and conditional branches, link’ complete semantic including transition condition and join condition also affect the execution of events, and also the different branches combination will make some activity unreachable. Whenever an event is to be invoked, its precondition must be satisfied. So we have to evaluate the occurrence condition for each event— weighted value.

Service composition is based on SOAP message, the message is depended on customer’ input or external partner service, so we cannot get the concrete value statically and the domain of variable may be infinite. So we adopt the idea of abstract interpretation, i.e., the abstract value of variable interval. WVC.w is a logic AND over the conditions along the path to the current event including condition expression of structure activity in BPEL(ce), transition condition(tc) and join condition(jc), along the path, the abstract interval of variable may change by merging the three conditions, so if the domain of variable is not null, then we say the result of the condition is true, otherwise is false, after all conditions get values according to the current abstract interval of variable, we can judge the result of WVC.w, if it is true, the event can execute, or not execute if its value is false. Such as for the event invokeDiscount in figure 1, we can get the flowing expression:
Because the branch condition is conflict, $WVC.w(\text{invokeDiscount})=false$, so the invokeDiscount can never be executed.

3.3. Weighted Vector Clock (WVC) Properties

In “weighted vector clocks”, the clock value is calculated as following, $pre(WVC.vc(e_j))$ is the precedent event of the current event $e_j$. For $WVC.vc(e_{i,k})[j]$ indicates the jth vector in $WVC.vc(e_{i,k})$ of $e_{i,k}$ (1≤j≤p; p is maximum degree of parallelism), $e_{i,k}$ indicates the kth event in sub-process i.

$$WVC.vc(e_{i,k})[j] = \begin{cases} \max(pre(WVC.vc(e_{j,k}))[j]) + 0 & j \neq i \text{ and } k \neq 0 \\ \max(pre(WVC.vc(e_{j,k}))[j]) + 1 & j = i \text{ and } k \neq 0 \\ 0 & j \neq i \text{ and } k = 0 \\ 1 & j = i \text{ and } k = 0 \end{cases}$$

The most important function of WVC is that we can determine whether two activities can concurrent or not with implicit execution history. For any two events $a_i \rightarrow b_j$ iff $WVC.vc(a_i) \leq WVC.vc(b_j)$ and $WVC.w(a_i) \land WVC.w(b_j)=true$. Two activities are parallel if and only if $\neg (e_i \rightarrow e_j)$ and $\neg (e_j \rightarrow e_i)$.

Based on the above abstraction structures, we define event is a 4-tuple $p=<\text{threadID}, WVC, \text{Type}, \text{Object}>$, $\text{Type} \in \{R, W\}$ denotes reading and writing operation separately. WVC is the extended vector clock of current event; threadID is the ID of concurrent thread which helps not consider the events in the same thread. Each object contains all the attributes of activity.

Just as the discussion above, the weighted value can greatly reduce the false race pairs. For data races, if two events are concurrent and their weighted values are satisfied, then we say datarace may occur.

A directed edge in time diagram represents the causality between two dots $e_i$ and $e_j$. An edge $e_i$ is weighted by a condition $w(e_i)$, meaning that the event represented by $e_i$ will be executed if $e_i$ is completed and $w(e_i)$ is evaluated to true. The w has to satisfy the following two properties:

**Property 1.** If $e_i$ is an activity without branch and $e_j$ is an immediate successor of $e_i$, i.e. $(e_i, e_j) \in E$, then: $w(e_i)=true$.

**Property 2.** If $e_i$ is a branch event, and $(e_{i1}, \ldots, e_{ik})$ are the k immediate successors of $e_i$, then:

$$w(e_{im}) \land w(e_{ih})=false, \text{ for all } m, h \in \{1, \ldots, k\} \text{ and } m \neq h.$$  

Consider the reachability of event, the weight constraint is derived as $WVC.vc(e_{i,k}) = \wedge_{k=1,\ldots,i-1} WVC.vc(e_{i,k+1})$ : i is the number of e.

Besides the weighted value, there is another aspect affect the result—the Isolated scope, if the “Isolated” attribute is set to yes, then the variable will be accessed sequentially, and when we transform the BPEL code to EMF model, we set the corresponding attribute of activity is true. So the enhanced datarace relation is as follow:
4. Experiments

The overall analysis framework for BPEL-based services data races (AF4WS) is shown in figure 2. In order to do datarace detection automatically, parsing is essential because of the difficult to analyze process code directly, here we use Eclipse Modeling Framework (EMF) to model different BPEL elements. Based EMF model we can construct the according BPEL control flow graph, the temporal relation and constraints information can be derived from the BPEL CFG, and construct the weighted temporal constraint diagram, so from the derived weighted vector clock graph, we can analysis the data races problems.

4.1. Case Study

Figure 3 is an online book order process after a simple modification. There are three tasks execute concurrently: 1) calculate the discount and the final amount to be paid; 2) calculate the vouchers the customer can get; 3) calculate the VIP user can get discount if the purchase amount is more than 1000. InvokeFinalSum and InvokeVIP are nested in the “Isolated scope” activity, and the attribute Isolated is set to yes.
The weighted temporal constraint diagram is constructed in figure 1. Here we omit the main process because it has no effects to the datarace. From the WTCG and our datarace formula, we can see there is no data races. But algorithm [14] will conclude that invokeFinalSum and invokeCoupons have a datarace because it not consider “Isolated” attribute in BPEL. Algorithm [15] will say assignDiscount and invokeVIP have a datarace for variable discount, because it does not consider the path condition. So our method can reduce false positive in data races detection.

4.2. Experimental Results
In order to evaluate the efficient and precision of this method, two processes are used in the experiments, namely Loan Approval (LA), Online Book Order (OBO). First, the traditional vector clock mechanism (Without weighted) is used. The second experiment is our presented technique “Weighted Vector Clock (WVC)”. Table 1 gives the dataraces number of the different methods and the number of activities in the two example processes. Among the two methods, we are focus on whether our method could eliminate some false positives generated by path condition and link semantic. One significant advantage of our approach is that it considers the path condition and link semantic which will improve the accuracy sharply and reduce the number of comparison.

5. Conclusion
SOA brings us the benefit of reuse existed web services, but it makes the validation and testing difficult at the same time. In this paper we have formally defined the causal relationships and presented the timestamps for handling complex web services so that the dataraces in BPEL-based services can be detected. The
method not only considers the path condition, link complete semantic, but also the isolated scope attribute. Through weighted vector clock, the dynamic executing information including the order and condition of event execution can be derived, Of course, improvements are also needed to be done in the future, such as improve the accuracy of abstract interpretation, and the slicing technique can be used to slice the redundant events.

Table 1 The Result of Detection

<table>
<thead>
<tr>
<th>Process</th>
<th>Activity Number</th>
<th>The Number of Datarace (Without Weighted)</th>
<th>The Number of Datarace (Our Approach)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBO</td>
<td>18</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>LA</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Acknowledgement

This work is supported by the National Natural Science Foundation of China (NO.91018002).

References