A Strategy for Timely Long Running Transactions Management Extended from WS-Transaction

Lin Qing, Aziz Nasridinov, Jeongyong Byun
Department of Computer Engineering, Dongguk University
e-mail :qinglin9@gmail.com, aziz_nasridinov@yahoo.com, byunjy@dongguk.ac.kr

WS –트랜잭션의 확장된 적시적 장기수행 트랜잭션 관리 전략

림 청, 아지즈 나스리디노프, 변정용
동국대학교 컴퓨터공학

Abstract

Timely management is an important activity in long running transactions. This can be seen in many Supply Chain scenarios where insufficient time management could lead to expensive compensations. Therefore, effective strategy for timely management in long running transactions is needed. In this paper, we propose our solution to detect and avoid violating deadline based on a comparison strategy of predicted complete time and required deadline. Three approaches are used. Firstly, we build Open Nested Transaction Model (ONTM) where deadline of top-level transactions are managed through a flexible time adjustment of lower level transactions. Secondly, workflow is introduced to WS-Transaction framework which supports the algorithm to predict complete time and spot delaying activities. Thirdly, a rule-based coordinator is developed to decide the confirmation or cancelation of participants' ongoing activities. We also discuss a workflow example for implementation.

1. Introduction

Most business-to-business applications require transactional support in order to guarantee consistent outcome and correct execution. These applications often involve long-running computations, loosely coupled systems, and components that don't share data, location, or administration. It's difficult to incorporate atomic transactions within such architectures [1].

WS-BusinessActivity [2], one of WS-Transaction components, is intended for long-duration, ACID-relaxed transactions among loosely-coupled systems and asynchronous communication between them. It ensures consistency by compensation. So that locking resources is neither impractical nor desirable. In addition, WS-BusinessActivity coordination type is based on the (ONTM) [3]. Transactions in this model can form a tree (of arbitrary height) of "sub-transactions". The sub transactions may commit independently of each other without having to wait for the root transaction to commit.

Workflow technology and SOA's Java Workflow Tooling project (JWT) is a key technology for process-based application, are now widely used in information systems, either as autonomous systems (like Lotus, Ultimus Workflow, etc.) or as parts of larger systems (like the workflow systems included in the ERPs like SAP, Oracle, etc.). Workflow management systems provide the foundation for defining and executing business processes. However, a key performance index, which should receive much attention to reduce failure rate of meeting deadline, is time management for business workflow, especially for timely long running transaction. For example, Supply Chain Management is a real case in which violating deadline would result in enormously expensive compensations.

In this paper, we propose our solution to detect and avoid violating deadline based on a comparison strategy of predicted complete time and required deadline. We exploited three approaches to do it. Firstly, we build Open Nested Transaction Model (ONTM) where deadline of top-level transactions are managed through a flexible time adjustment of lower level transactions. Secondly, workflow is introduced to WS-Transaction framework which supports the algorithm to predict complete time and spot delaying activities. Thirdly, a rule-based coordinator is developed to decide the confirmation or cancelation of participants' ongoing activities. Eventually, our strategy helps reduce the risk of compensation that the business activities cannot afford.

The remaining of this paper as following: Part 2 is related work, and Part 3 is our motivating scenario. The solution is explained in Part 4, 5 and 6. Part 7 makes conclusion and highlights future works.

2. Related Work

The idea of time management in Web Service transactions is studied by many researchers.

For example, researchers in [5] presented the RT-Llama SOA middleware to reserve CPU bandwidth in advance for each service in the process to ensure the process can meet its end-to-end deadline. This solution is suitable for media

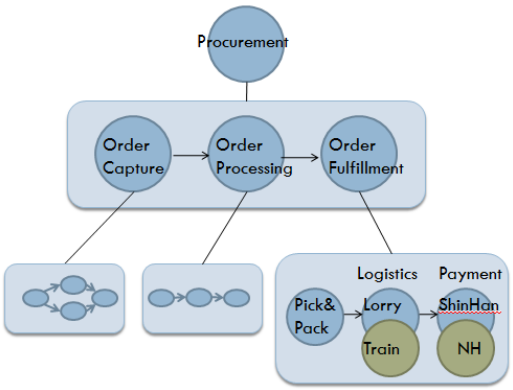
streaming SOA application but not for real time business activity. In contrast, in our paper, more reasonable solution is raised for management of timely Web Service transaction.

In the following paper [6], they proposed a very generic semantic framework to solve the problems in Web Service transaction. Our paper will present a detailed solution for the complete time prediction and deadline insurance.

Paper [7] is our previous research. It also tries to solve the delay problem of transaction. Reservation resource method was used as a way to reduce the risk of compensation. Actually this approach is more close to a half locking mechanism. However, for long running concurrent conflict is not the only reason that will lead to delay. And reservation approach may result in unavailability of the resource because it is a highly pessimistic method. In comparison, this paper ties to manage transaction processing that goes beyond resource allocation.

3. Motivating Scenario

We set our scenario in a business activity, procurement. Samsung factory orders 1000 Intel CPUs from Intel Cooperation with an imposed time constrain for this procurement because factory hopes not to miss manufacturing plan. The order is processed by a series of heterogeneous and distributed systems.



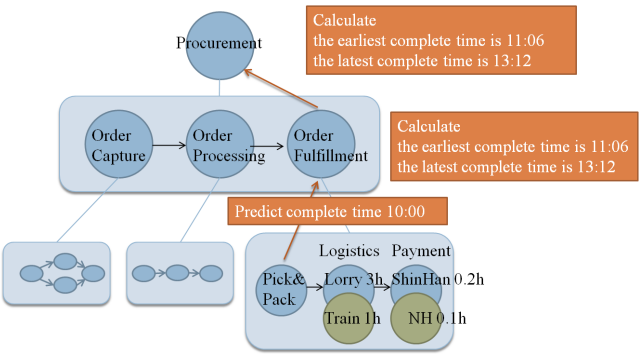
(Figure 1) Nested Transaction Model Building for Procurement Activity

Figure 1 shows our ONTM built for procurement activity. The top transaction is named *Procurement*, and shallow blue panel denotes a sub transaction but also contains a unit of works. Inside the panel, black arrows mean a workflow including sequential, parallel and other operations for substitution.

Any delay from one of sub tasks will result in the overall latency, so the whole transaction may need to be cancelled and decrease customer's satisfaction. Since those sub tasks are completed by distributed and autonomous service providers, it is impossible to apply a central control to solve latency problem through a precise scheduling approach.

In this case, we raise our idea of predict complete time range. And we encourage more alternative participants to enroll in the transaction. Because when it predicts a delay, an alternative service provider with a shorter processing time can be called up. In this way, we could avoid violating deadline.

It is illustrated with an example in the scenario shown in Figure 2. Now it is in the period of order fulfillment, and *Pick&Pack* task is completing. The participant, who is doing packing job, reports its coordinator about predict end time, 10:00. *OrderFulfillment* activity's coordinator recalculates end time, and reports to an upper coordinator. Therefore, an upstream complete time report and downstream verifying response is our critical approach for Web Services communication between fully distributed enterprises to detect a delayed sub transaction. It may reside in the very bottom of the hierarchy.



(Figure 2) Message Communication Mechanism for dynamic prediction of end time

Subsequently, *OrderFulfillment* coordinator calculates three sub tasks complete time period, 11:06 to 13:12. So will *Procurement* coordinator do and get the estimated complete time period, 11:06 to 13:12. If *Procurement* (top transaction's) deadline is later than 13:12, any participant is supposed to satisfy timeliness. But if deadline is earlier than 11:06, and if it's hard deadline, the whole transaction will be cancelled. This is the worst case that our solution doesn't reach, while it is believed that it is better than current solutions. Since an earlier delay detection brings to an earlier compensation with less cost.

4. Rules-based Decision Making

Nearly all enterprise systems have a workflow inside to take care of particularly complex problems, and nearly all complex problems contain a transaction. In Figure 1 this is already a nested transaction model combined with workflow. The workflow explicitly describes processing order of sub transactions, which provides us a useful tool to evaluate complete time range.

Rule:

Complete time verification function is working in parent coordinators to foresee a child transaction T performed by some participant would excess time constrain. A TRUE returned value is denoted by verification (T), in contrast, a FALSE value is ¬verification (T). How the value effects coordinator's decision is set in Rules.

$$\forall T(\, active(T) \wedge verification\ (T)) \rightarrow complete \quad T$$
$$\forall T(\, active(T) \wedge \neg verification\ (T)) \rightarrow cancel \quad T$$

Active(T) represents transaction T is in "ACTIVE" state, if verification function's feedback is positive, coordinator

allows participant to complete and vice versa.

$$\forall T(\text{completing}(T) \wedge \neg\text{verification}(T) \wedge \neg\text{alternate}(T)) \rightarrow \text{Cancel } T$$

$$\forall T(\text{completing}(T) \wedge \neg\text{verification}(T) \wedge \text{alternate}(T)) \rightarrow \text{Cancel } T \wedge \text{Complete alternate}T$$

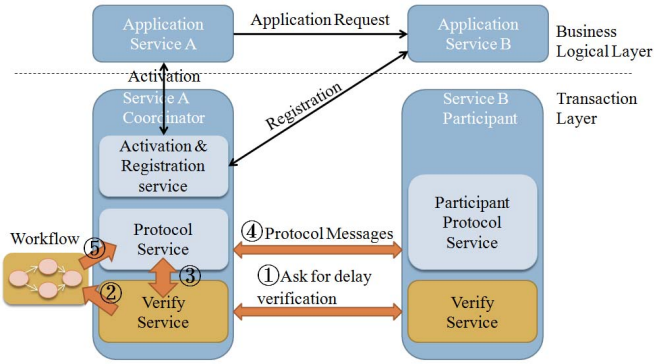
Completing(T) denotes transaction T is in state of "COMPLETING", with a predict complete time from T's participant, coordinator invokes verification function. If the result is negative, and T doesn't have substation choice, then cancel T and vice versa.

5. System Design

5.1 Architecture

Figure 4 shows the general system architecture designed for our strategy. And we extend from WS-Coordinator framework with WS-BusinessActivity protocol.
Two new components are developed:

1. Verify service. Both coordinator and participant sides are equipped with verify service to evaluate and predict a complete time range.
2. A workflow component resides in the coordinator. The workflow contains information including process sequence, general execution time periods from service providers, substitution or compensation relationship and current states of participants.



(Figure 4) System Architecture extended from WS-Coordinator Framework

The arrows in Figure 4 show interaction and cooperation between components. The arrow marked with 1 means the message flow of complete time verification request and response. My solution requires that participants have the responsibility to report the estimated termination time to their coordinator, especially when this participant is suffering from delay. As a result, coordinator could control and perceive a potential latency as early as possible.

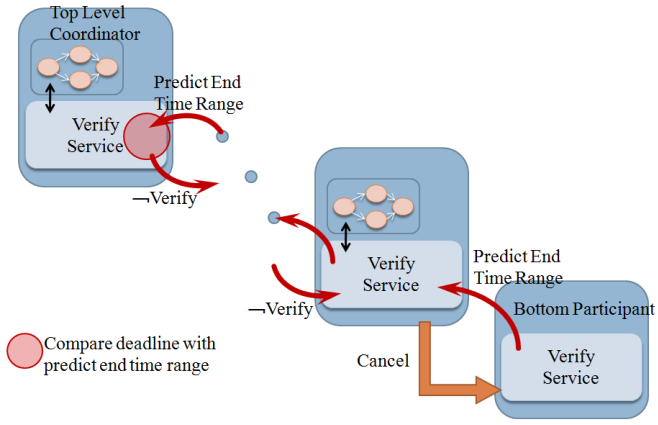
The verify function also needs to refer to workflow, so an interface should be open for verify component to read from, and this interface is highlighted in Figure. 4 arrow 2 .

Consequently, with a feedback from verify function, a rule-based decision has to influence Protocol Service through arrow 3. Thus, by protocol messages coordinator may issue a new command to participant from arrow 4.

Finally, Protocol Service updates state information about participants by arrow 5.

5.2 Upstream Verify Request and Downstream Response

Supporting flexible transaction deadline adjustment is our main innovation in this paper. In contrast with previous WS-Coordinator, our solution focuses on guarantee top level transaction's deadline, instead of every individual task. For this reason, the verify process has to experience a series of upstream and downstream service invocations if we explain this phenomenon in view of open nested transaction model.



(Figure 5) Upstream Verify Request and Downstream Response

Figure 5 illustrates the whole verifying process. From bottom child participant reporting a predict end time, the verifying chain begins. Parent coordinator calculates its predict end time, and reports to grandparent coordinator, and so forth, until this time report arrives at top. Only the top level (root) coordinator owns the right to determine whether the action that is taken in the bottom would shake the tree and suspend deadline in the air.

Generally speaking, we get many benefits from this mechanism. All that matters is coping with accidents during business processing flexibly through a more intelligent transaction manager.

6. Complete Time Estimate and Verify Algorithm

Complete time evaluation is a pivotal function for Verify Service. Tasks' sequence determines a general approach of estimation. For instance, *T1, T2…Tn* are *n* sub transactions within the scope of transaction *T*, and *ExT(Ti)* is a forecast execution time for Ti. If they are sequential tasks, the complete time is as following:

$$\text{Complete time} = \text{CurrentTime} + \sum_{i=1}^{n} ExT(Ti) \quad (1<=i<=n)$$

If they are parallel tasks, the complete time is as following:

$$\text{Complete time} = \text{CurrentTime} + \text{Max}(T1 \ldots Ti \ldots Tn) \ (1<=i<=n)$$

If they are alternative tasks, the complete time domain *(Earliest End Time, Latest End Time)* is calculated as

following:

Earliest Complete time= CurrentTime + Max (T1…Ti…Tn) (1<=i<=n)
Latest Complete time= CurrentTime + Min (T1…Ti…Tn) (1<=i<=n)

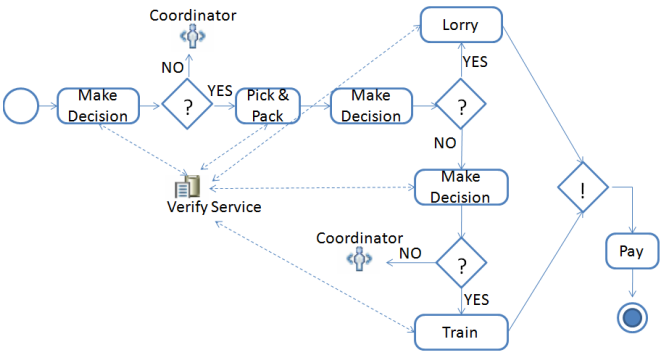
Therefore, a workflow, as one part of coordinator constructions, provides efficient information of sub transactions that under its control, such as information includes process sequence, a general execution time from service provider, substitution or compensation relationship and current state.

The returned value domain is reported to an upper level coordinator, and then coordinator runs the algorithm once again to get a new value domain. Iterate in this way several times, finally top level coordinator's verify function will compare this complete time domain with required deadline. So if deadline is earlier than *MinEndT*, which means delay probability will happen, a false verification response is passed downstream. Meanwhile decision on confirmation or cancellation of participants' work will be made based on Rules that are stated in Part 4.

7. Implementation

Eclipse SOA's Java Workflow Tooling project (JWT) provides design time, development time and runtime workflow tools. It also fosters interoperability between Business Process Management (BPM) platforms and integration in Information Systems thanks to Service Oriented Architecture (SOA). So it is considered as a suitable tool to develop the workflow inside of transaction.

The workflow management system should enhance its functionality especially in time management in order to deploy workflow management in an enterprise for managing the business processes. The time constraints and time management are very important in designing and managing workflow models.



(Figure 6) View of OrderFulfillment Workflow

Figure 6 presents the workflow for OrderFulfillment transaction. You will see that decision is made before every action is taken, and the decision refers to Verify Service. And this Verify Service is our central function that is related to time evaluation and management. This method works in the following way. If a negative result is casted, alternative Web Service will be verified and chosen as substitution. Even if there are no more candidates that the workflow can turn to, it will consult from coordinator. Moreover, during the running of every action, workflow would catch the instant and predict completion time and use Verify Service to check the risk of violating deadline.

Different workflows within different transactions can be built in this similar way. Cooperated with the management of nested structured coordinators, finally we reach our goal of time constrain control among a fully distributed and autonomous environment.

8. Conclusion and Future Work

In this paper, we tried to avoid violating deadline based on a mechanism of real time prediction of complete time. The comparison between predicted time and required deadline is considered as the criteria to detect delay and thus avoid expensive compensation. We also achieved the goal of ensuring root transaction's deadline by dynamic selection and adjustment of participants. And, we used strategy which is helpful to reduce the risk of compensation that the business activities cannot afford.

Now we have a good strategy to avoid delay by complete time monitoring. So in the future work, we will solve the delay problem for a specific scenario in Supply Chain applications. And we hope other better time management methodologies will be developed to enhance and improve the current one. Also, Web Service providers' complete time prediction methods will be discussed if a real case is combined, such as scheduling to refrain from delay and improve business activities' performance.

Reference

[1] M. Little, Web Services transactions: Past, present and future, found at: www. jboss.org/ jbosstm/ resources/ presentations/ XML2003.pdf.
[2] OASIS Web Service Business Activity (WS-BusinessActivity), found at: http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec.pdf
[3] E. B. Moss, "Nested transactions: An approach to reliable distributed computing," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, Apr. 1981, also published as MIT Laboratory for Computer Science Technical Report 260.
[4] Aldarmi S.A., Real-Time Database Systems: Concepts and Design, The University of York, 1998.
[5] OASIS Web Service Coordination (WS-Coordination) Version 1.2, found at: http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os.pdf.
[6] Panahi, Weiran Nie, Kwei-Jay Lin, "The Design and Implementation of Service Reservations in Real-Time SOA," e-Business Engineering, ICEBE '09, IEEE International Conference only, pp.129-136, 21-23 Oct. 2009
[7] Yingyuan Xiao, Hua Zhang, "SHTM: A Semantic Hierarchy Transaction Model for Web Services Transactions" IEEE computer society, 2008.
[8] LinQing, Aziz Nasridinov, Jeongyong Byun, "Reservation-based protocol extended from WS-BusinessActivity" Korea Information Processing Society Conference, 2010.