A TRANSACTION MODEL AND IMPLEMENTATION BASED ON MESSAGE EXCHANGE FOR GRID COMPUTING

Zou Yali, Ling Hong
School of Management, Fudan University, Shanghai, China 200433

Wu Yonghua
Department of Computer Science and Engineering, Shanghai JiaoTong University, Shanghai, China 200030

Keywords: Grid computing, Transaction process, Message exchange.

Abstract: While grid computing is extending to apply to the commercial area, the need for transaction process that ensures the dependency of grid computing is more urgent. This paper describes how short-lived and long-lived transactions can be implemented with Web Service Resource Framework (WSRF) by proposing a grid transaction process model (GridTP). It can manage and monitor the state of transaction participants via the message exchange among those participants, and it can use atomic or cohesion transaction coordination algorithms to coordinate the transaction participants to achieve a mutually agreed outcome.

1 INTRODUCTION

Grid computing has been widely accepted as a promising solution to sharing large-scale resources and accomplishing collaborative tasks (Foster *et al*, 2001). With the newly proposed grid computing framework WSRF (Web Service Resource Framework) (Czajkowski *et al*, 2004), the enterprise field applications of grid have attracted more attentions. Transaction, a mechanism ensuring all the participants of some activity to achieve a mutually agreed outcome, determines the successful implementation of grid computing in practical applications. Traditional transactions which hold ACID (Atomicity, Consistency, Isolation, and Durability) properties lock the resources involved and the coordinator possesses the absolutely control on participants. However, transaction for Grid Services challenging the traditional researches, because Grid Services which are loosely coupled and heterogeneous often take a long time due to business latency or user interaction. And Coordinator cannot lock involved resources because of the highly autonomy of Grid Services.

Our aim in this paper is to design a GridTP model that facing the challenge. The proposed model support both short-lived and long-lived transactions using the same set of WSRF-based messages. And GridTP should be used for both Grid Service providers and client side applications.

The rest of this paper is organized as follows: In section 2 we simply review the related works. In section 3 we describe the model of GridTP. In section 4 we give a practical implementation of GridTP. Finally, conclusions and future researches are discussed in section 5.

2 RELATED WORK

There are some researches had discussed standards and models concerning the traditional distributed transaction such as X/Open DTP (Distributed Transaction Process) (Jeong & Lew, 1998). But few of them can effectively support long-lived transaction. And WS-Coordination and WS-Transaction (Cabrera *et al*, 2002) describe a Web Services transaction framework that can accommodate multiple coordination protocols. But its coordination for business activity is too complex and has not been described in detail. Business Transaction Protocol (BTP) (Ceponkus *et al*, 2002)

Yali Z., Hong L. and Yonghua W. (2006).
A TRANSACTION MODEL AND IMPLEMENTATION BASED ON MESSAGE EXCHANGE FOR GRID COMPUTING.
In *Proceedings of WEBIST 2006 - Second International Conference on Web Information Systems and Technologies - Internet Technology / Web Interface and Applications*, pages 225-228
DOI: 10.5220/0001240502250228

defines a set of messages exchanged between coordinator and participants. However, the BTP cannot well support the dynamic of grid transaction and lacks of flexible recovery mechanisms.

Our implementation is based on XML message exchange technology, soheterogeneous Grid Services can easily communicate with each other. Our implementation is also based on WSRF, which can simply locate and access stateful resources.

3 GRID TRANSACTION PROCESS MODEL

GridTP model is made up of following components: message dispatcher, state message set, coordinator, participant and client (Figure 1). The message dispatcher, state message set, coordinator, participant are Grid Services or resources deployed in WSRF platform and the necessary parts of GridTP. The client component is provided to simplify the work when the client applications want to use GridTP to support transactional Grid Services. It's an extended part of GridTP provides some APIs for transactional operations such as start a new transaction.

Now we give a detail introduction to each component as follow:

Message dispatcher. It receives and parses XML messages sent by other components, and then dispatch the messages to different components according to the message type and destination.

State message set. It contains all the participants' addresses and states. This message set is updated and managed by message dispatcher component.

Coordinator. It uses atomic or cohesion algorithms to coordinate transaction according to different transaction type.

Participant. It is a transactional Grid Service running in WSRF environment and follows the programming pattern of GridTP.

Client. The client component facilitates using the GridTP to support transactional Grid Service.

4 IMPLEMENTATION OF GRIDTP

In order to describe clearly, we give the following definitions:

Definition 1. Business Message (BM) is a 5-tuple {T,TxID,S,R,MSG}.T represents message type; TxID represents current transaction ID; S represents the address of message sender; R represents the address of message receiver while MSG is the part used for transferring the parameter defined by applications.

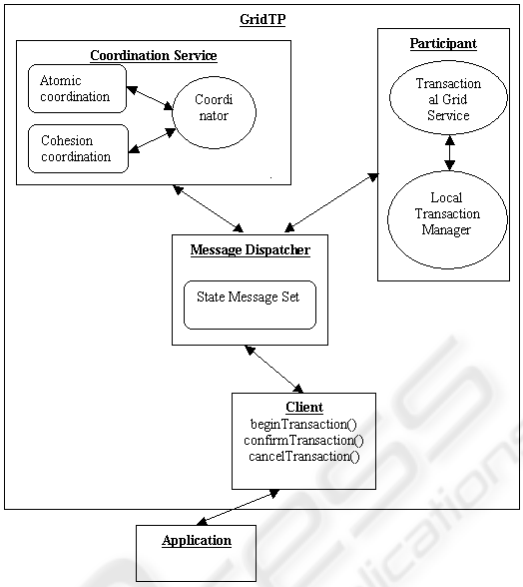

Figure 2: The WS-Resource description of Status Message

Definition 2. Transaction Message (TM) is a 5-tuple {T,TxID,S,R,OP}. T represents message types; TxID represent current transaction ID; S represents the address of message sender; R represents the address of message receiver; OP is transactional operation.

Definition 3. Transaction State (TS) is a 3-tuple {TxID,P,S}. TxID represents transaction ID; P represents the address of participant; S represents the state value.

Definition 4. Atomic Transaction (AT) is the transaction that all participants have to commit synchronously or abort entirely, and need apply atomic coordination algorithm to coordinate.

Definition 5. Cohesion Transaction (CT) is the transaction that allows some candidates to abort while others to commit, used for coordinating long-lived transactions.

4.1 Coordination of Atomic Transaction

Initiation of transaction: Application can request an atomic transaction, so the client component sends a TM-Begin message to the message dispatcher, and the message dispatcher dispatches the message to an available coordinator. Finally the coordinator invokes local transaction manager to create a new transaction and returns a TS-Begun message to message dispatcher.

Participants joining in: Participant sends a TM-Enroll message which contains the transaction ID to the message dispatcher. And according to the transaction ID the message dispatcher dispatches message to the corresponding coordinator to deal with. Finally the coordinator returns a TS-Enrolled message to message dispatcher, and returns to the participant at the end.

Preparation for the transaction: When coordinator receives the request asking for transaction preparing, it fetches out the state message set, and sends a TM-Prepare message to all the participants in this set.

Committing of transaction: When coordinator receives the request asking for transaction committing, it fetches the state message set and checks the state of all the participants. Once all the participants are TS-Prepared, then the coordinator sends a TM-Commit message to all the participants, otherwise, sends a TM-Rollback message. Finally client application will get TS-Cancelled or TS-confirmed messages, while the former means the transaction is cancelled, and the latter means transaction executes successfully. Algorithms used to coordinate the atomic transaction are two phase commit protocol (2PC) (Gray & Reuter, 1993).

4.2 Coordination of Cohesion Transaction

The initiation and joining of cohesion transaction are similar as atomic transaction. But cohesion transaction allows some candidates to abort while others to commit. So we need to determine a final confirm set according to the business logic, and only the participant in this set can be entirely success or fail. In addition, cohesion transaction allows participants commit ahead of the whole transaction, and need compensation transaction to cancel transaction behaviour.

Committing of transaction: Client component via coordinator checks state message set, and confirms final transaction whether can be committed according to the final confirm set. If can be committed, then sends a message TM-Confirm to the participants, the transaction successes. If cannot be committed, then sends a message TM-Cancel to all the participants. When participant receives the message, then operates the compensate transaction for the committed part.

Compensation of transaction: If the committed participant cannot get the TM-Confirm message or the TM-cancel message sent by the coordinate within certain times, then automatically invokes compensation transaction to recover.

Here we give the algorithms we used to coordinate cohesion transaction.

Arithmetic 1: Cohesion transactions coordinator

```
CTCoordinatorTM_Process{
     switch(ReceivedTM){
          case TM-Begin:
   create a new TM;
   return TS-Begun;
          case TM-Enroll:
   add participant to State Message Set;
   return TS-Enrolled;
          case TM-Commit:
   foreach TS in(State Message Set){
    p=get participant address from TS;
      if(p in Final Confirm Set){
        state=get state of p;
        if(state==TS-Cancelled){
 send TM-Cancel to all participants;
  return TS-Cancelled;}}}
 send TM-Confirm to all participants;
  return TS-Confirmed;}}
```

Arithmetic 2: Cohesion transactions participant

```
CTParticipantTM_Process{
     switch(ReceivedTM){
          case TM-Commit:
   generate compensation transaction;
   commit transaction;
   return TS-Committed;
          case TM-Cancel:
   execute compensation transaction;
   return TS-Cancelled;
          case TM-Confirm:
   return TS-Confirmed;}}
```

4.3 WS-Resource Description of Message Set

We simply model the state message set as stateful resource (Figure 2) in WSRF environment. We describe the model as follows:

Service Interface defines the service of visiting and managing the stateful message resource. These services are described by the wsdl files provided by schema.

WSRF specification is supported. Let WSRF container knows the resources support which protocols. There are mainly two protocols:

WS-ResourceLifetime: Means this resource need container being managed according to the WS-ResourceLifetime specification.

WS-ResourceProperty: Describes state message set. Client component can visit and manage this resource via WS-ResourceProperty specification.

```
<ws-resource
xmlns="http://grid.sjtu.edu.cn/gridtp/StatusMsgSet/
description/2005-8-10 ">
<resource-description>
<name>StatusMsgSet</name>
<txid>StatusMsgSet</txid>
<schema>counter.wsdl<schema>
<ejb-link>Counter</ejb-link>
<ws-ResourceLifetime>true
</ws-ResourceLifetime>
<ws-ResourceProperty>
<property name="msgSet"
type="java:Set" qname="msgSet">
</ws-ResourceProperty>
</resource-description>
</ws-resource>
```

4.4 The Process of System Operation

As Figure 3 shows, the application via client component interface initiates a transaction, and client requests message dispatcher initiating transactions. After dispatcher parses the request message of initiating transaction, it sends message to the available coordinator. Finally coordinator initiates an atomic or cohesion transaction according to the type of transaction, and returns TS-Begun message contains a transaction ID to the message dispatcher. Message dispatcher will maintain a related state message set for this transaction. Henceforth, all the messages among the participants related with this transaction will be dispatched via message dispatcher, and the state message set related with this transaction will changed.

When client application requests committing transaction, client component sends a TM-Confirm message to message dispatcher. Message dispatcher dispatches this message to coordinator to deal with. When coordinator receives message, it will get corresponding state message set according to the transaction ID contained in message, and then sends a TM-Prepare message to every participants. After receiving TS-Prepared responding of every participant, it further sends a TM-Confirm message to every participant. At the end, when it gets the TS-Confirmed messages of all the participants, it returns TS-Confirmed message to Client component.

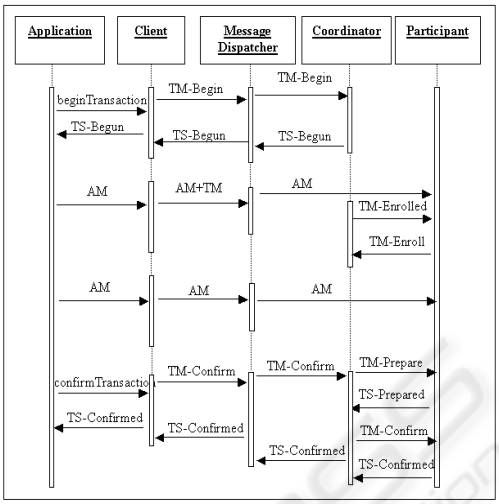

Figure 3: The Process of System Operation.

5 CONCLUSION AND FUTURE WORK

We have proposed a GridTP model to support both short-lived and long-lived transactions using the same set of simple WSRF-based messages, without breaking the architecture's design. We also need to work for a more flexible compensation mechanism to recover failed transaction in any situation. Other works like security and QoS (Quality of Service) also should be considered in future research.

REFERENCES

Cabrera,F., Copeland,G., & Freund,T. et al.(2002).Web services coordination (ws-coordination).

Cabrera,F., Copeland,G., & Cox, B. et al.(2002). Web services transaction (ws-transaction).

Ceponkus, A. et al. (June 2002). Business transaction protocol v1.0. Retrieved August 11, 2005, from http://www.oasis2open.org/committees/download.php/1184/2002206203.BTP-cttee-spec-1.0.pdf.

Czajkowski, K., Ferguson, D., Leymann, F., Nally, M., Sedukhin, I., Snelling, D., Storey, T., Vambenepe, W., & Weerawarana. S. (March 2004). Modeling stateful resources with web services v1.1.

Foster,I., Kesselman,C., & Tuecke, S.(2001).The anatomy of the grid: Enabling scal-able virtual organizations. *International Journal of High Performance Computing Application,,5.*

Gray, J. & Reuter, A. (1993).*Transaction Processing: Concepts and Techniques.* Morgan Kaufmann.

Jeong,I.C., & Lew,Y.C.(1998). Dce (distributed computing environment) based dtp (distributed transaction processing). In *Information Networking, (ICOIN-12) Proceedings*, January 1998, 701-704.