Collaborative Process Execution for Service Composition with StarWebService

Bixin Liu YuFeng Wang Bin Zhou Yan Jia

National University of Defense Technology, ChangSha, China
{bxliu, yfwang, binzhou,yanjia}@nudt.edu.cn

Abstract. This paper gives a brief introduction to a collaborative process execution mechanism for service composition implemented in StarWebService system. The main idea is to partition the global process model of composite service into local process models for each participant, so as to enable distributed control and direct data exchange between participants. It is a novel solution to the issues of scalability and autonomy of centralized execution.

1 Introduction

With the growing trend of service oriented computing, composition of web services has received much interest to support dynamic inter-enterprise application integration. Many research plans and projects [1,2,3] are established on this attractive theme.

A main approach to realize service composition is to orchestrate the constituent services according to a high-level business process model. Current strategies mainly follow the work of workflow technology and propose centralized composition engine to execute the process model [1]. However we believe that service composition targets at large scale information systems in open environments, which demands that execution efficiency will not decrease with the increase of participant services, exchanged data volumes and concurrent requests. A centralize engine will have inherent limits in scalability. Moreover, centralization conflicts the autonomy of business partners and prevents them from accommodating to exceptions in an autonomous manner.

In this paper, we present the collaborative process execution mechanism provided by StarWebService that allows a composite service to be carried out on multiple nodes. By virtue of distributed scheduling control and direct communications, better scalability can be achieved, as well as autonomy of participating services.

2 StarWebService Overview

StarWebService is a research project of StarMiddleware Group [5] which provides a service-oriented integration infrastructure for enterprise applications. It bridges the gap between traditional middleware technologies and web services seamlessly, and

enables flexible and scalable inter-enterprise integrations by composing services into high level business process.

Primary components of StarWebService system include: the web service runtime environment based on Bus-Container-Service architecture; bi-directional application gateways for exporting various middleware-based enterprise resources (like CORBA objects, EJBs) as web services, as well as importing web services as specific middleware resources; and collaborative service composition engine for cross-organizational integrations. The first two are abridged from this paper for space limit.

3 Collaborative Process Execution

The overall concept is a strategy of "divide and collaborate". Process model of a composite service is partitioned into several fragments and deployed to several peer composition engines. At run time, those engines collaborate with each other and schedule the composite service in a decentralized manner.

3.1 Global Process Model vs. Local Process Model

Generally speaking, the process model of composite service captures control flows and data flows among activities which map to invocations on different constituent services. It is a logical centralized view of the contract between participants (including all the constituent services and the composite service itself). So we call it *global process model* (GPM) for the composite service.

Ideas of partitioning business process to enable distributed execution have been proposed in [2,3]. But they divide the process model into fragments consisting of only one activity (task), and then distribute them over a set of selected nodes responsible for carrying them out. However, we argue that a business process usually involves complex conversations within participants who carry out more than one activity. So we adopt a relative course-grained policy and group all the activities and interactions with respect to one participant together. It's also presented as a process but only consists of activities that are assigned to specific participant. We call it *local process model (LPM) for the participant* in the composition. A composition involving N constituent services can be decomposed into N+1 LPMs.

Compared with GPM, LPM only concerns the behavior of a specific participant in the context of a composition, such as whom it speaks with, when to perform its operations, and how to exchange messages. The logic relations among activities in LPMs accord with those in GPM. In fact, a LPM can be understood as the projection of GPM on a specific participant.

Algorithm can be developed to deduce LPMs from GPM. Its main idea is to partition activities into several subsets in terms of their performers and then to relate them by analyzing their causal and data dependencies in GPM. Explicit activities are added to represent communications between LPMs if two activities are data-dependent in GPM but belong to different LPMs. Sec. 4 will give an example of GPM and generated LPMs. Details of the algorithm will be presents in other papers.

3.2 Service Composition Engine

The service composition engine in StarWebService consists of the following parts:

– *Partition module* implements the algorithm that divides the GPM of a composite service into LPMs for its participants.

– *Deploy module* distributes a process model to composition engine in a target node and deploys it in the service runtime environment for execution.

– *Execution module* manages the execution of processes by event-driven mechanism and maintains contexts for process instances.

Note that deployed processes are wrapped as services with published WSDL documents and can be invoked through SOAP messages. Furthermore, the partition module and the deploy module are also implemented as special services deployed in the system to enable communication between composition engines.

3.3 Collaborative Execution with StarWebService

Instead of executing GPM in a centralized engine, composite service execution with StarWebService involves multiple nodes which respectively execute a LPM for the composition. The whole procedure can be illustrated as two phrases.

In the prepare phrase, GPM of the composite service is pre-processed by the partition module and LPMs for every participant are generated. After the concrete constituent service providers for each participant are decided, their deploy modules are invoked to transmit and deploy LPMs to their composition engine. LPM for the composite service itself is deployed at the node that wishes to host it, which is called *master node*, others *participant nodes*. Now it's ready for execution.

The master node reacts on the customs' requests and initiates the composite service. Execution module in the master node then load its LPM and schedules activities defined in. In most cases, the master node does few things but invokes participants' local processes and then waits for their termination. Sometimes callback interfaces are necessary. For the participant nodes, their LPMs are activated as response to invocations from the master node. Actions taken on both master and participant nodes and communications among them are strictly conducted by their LPMs, which assures compatible behaviors with each other. Thus, each of the involved engines schedules a part of activities in the composition and collaborates with each other through peer-to-peer communication.

4 An Example

We refer to the purchase order process example in BPEL4WS specification [4] to demonstrate our idea. It is composed of invoice service (IS), shipper service (SS) and production scheduling service (PSS), with GPM depicted in the left of Fig1. LPMs for 4 participants in the composition are shown in the right. Take the LPM of the shipper for instance. It specifies the activities happened on the shipper site. It requests the shipper information to decide the shipper after the order arrives. Then it notifies the invoice service of shipping information in parallel with logistics arrangement. At last

the shipping schedule is sent to the production scheduling service to end the process.

Both the GPM and LPM can be specified with existing business process modeling languages. BPEL4WS is adopted by StarWebService for process definition and engine implementation. Fig 2 shows part of BPEL definition for the shipper's LPM.

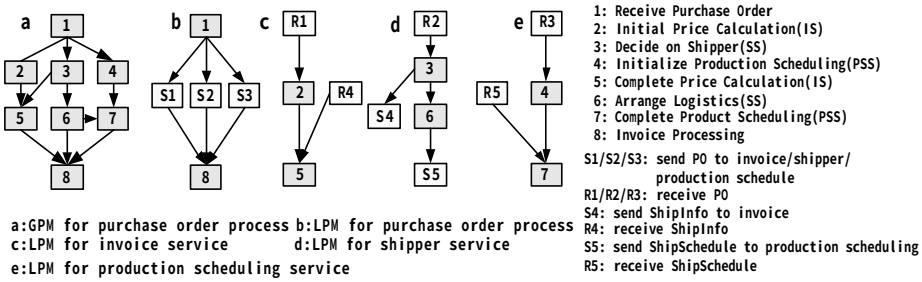

Fig. 1. GPM and LPMs of the purchase order process

```
<sequence>
  <receive partner="purchaseOrderProcess"
        portType="purchaseOrderprocessPT"
       operation="sendPO"   container="PO">
  <assign>...</assign>
  <invoke  partner="shippingProvider"
         portType="shippingPT"
        operation="requestShipping"
        inputContainer="shippingRequest"
        outputContainer="shippingInfo">
  <flow>
    <invoke partner="invoiceProcess"
        portType="invoiceProcessPT"
        operation="sendShippingPrice"
        inputContainer="shippingInfo"/>
  <sequence>
    <recieve partner="shippingProvider"
        portType="invoiceProcessPT"
        operation="sendSchedule"
        container="shippingSchedule"/>
    <invoke
      partner="productionScheduleProcess">
      portType="productionScheduleProcessPT">
      operation="sendShippingShedule"
      container="shippingSchedule"/>
  </sequence>
 </flow>
</sequence>
```

Fig. 2. Part of BPEL4WS definition for the shipper's LPM

5 Conclusions

We propose a collaborative process execution mechanism for service composition in this paper. By distributing the control to multiple nodes and enabling direct communication between distributed composition engines, it provides a novel solution to scalable and autonomous inter-enterprise integration.

Reference

1. Senthilanand Chandrasekaran, John A. Miller, etc. "Composition, Performance Analysis and Simulation of Web services". *Electronic Markets Volumn13(2)*, Nov.2003
2. Roger Weber, Christoph Schuler, Patrick Neukomm, etc.. "Web Service Composition with O'Grape and Osiris". In Proc. of 29th VLDB Conference, Berlin, Germany, 2003
3. Boualem Benatallah, Marlon Dumas, etc. "Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services". Proceedings of ICDE02, 2002
4. BPEL4WS specification 1.0 http://www.ibm.com/,2002
5. StarMiddleware Group, http://www.starmiddleware.net