Decentralized Workflow Execution for Virtual Enterprises in Grid Environment

Wei Tan, Yushun Fan
Dept. Automation, Tsinghua University, Beijing 100084, China
E-mail: tanwei@mails.tsinghua.edu.cn, fanyus@tsinghua.edu.cn

Abstract

With the emergence of global market and virtual enterprises, coordination of business processes in dispersed organizations by distributed workflow execution will take on more importance. At the same time, grid workflow is now been regarded as a fundamental service in grid environment. In this paper we give a real life example to see how distributed workflow technology can be used in virtual enterprises in grid environment. We propose a novel distributed workflow execution architecture, and we also address the key technologies in implementing this architecture, i.e., the procedure of distributed process execution and the model partition method. With the real life example we demonstrate how to use the proposed method to support distributed workflow execution inside virtual enterprises. Our method can be used in many scenarios such as service oriented computing and the grid, to serve as the process management platform. We also stress that future research should focus on making further investigations on the model partition policy, i.e., how to partition the model and allocate task to different sites to improve the overall system performance.

1. Introduction

For a long time, workflow has been the key technology in enabling business process reengineering and business process automation [1]. Traditionally, workflow management systems (WfMSs) serve as the process management tool inside an enterprise, and are often built with a centralized manner, for example, the Client/Server or Browser/Server architecture. With the emergence of global market and virtual enterprises, decentralized coordination of processes in dispersed organizations will take on more importance. We know that virtual enterprises are typically based on volatile partnership, loosely-coupled collaboration, with an aim to exploit profit from rapidly changing business opportunities. So there is a growing need for a workflow platform to support the collaboration among entities inside a virtual enterprise. At the same time, grid computing is emerging as a key technical architecture for large scale, end to end application integration, grid and Service Oriented Computing also accelerates the development of distributed workflow technology [6]. Grid workflow are the combination of grid and workflow technology, providing build-time and run-time support to large-scale sophisticated e-science and e-business processes in many complex scientific and business applications [10]. Therefore, decentralized workflow execution is a key technology in grid workflow system due to the autonomy and heterogeneity of grid applications.

Under this circumstance, many distributed workflow system were developed. Replicated servers and server clusters are used to address the required levels of scalability and fault tolerance in commercial workflow systems, which can be seen as a primary and pragmatic solution to distributed workflow execution [2]. The Exotica project [3] proposes a completely distributed architecture in which a set of autonomous nodes cooperates to complete the execution of a process, with persistent message queue as its information transmission technique. Aalst introduces his well-known WF-net model to the inter-organizational paradigm [4]. In [5], innovative approaches to support decentralized process enactment with the Peer to Peer (P2P) technology are presented. In [6], AI planning techniques are used to generate grid workflows in a decentralized environment. The author of [8] proposed a dynamic grid workflow model which concentrates on component reusing.

Our work focuses on the execution procedure and model partition techniques for distributed workflow execution, which can be seen as an original effort in the field of distributed workflow management. We believe our method can be used in many scenarios besides grid environment, to serve as the process management platform. In this paper we first give a real life example of distributed workflow execution. Then
we propose a novel distributed workflow execution architecture for virtual enterprises in grid environment, and discuss the key technology in implementing this architecture, i.e., distributed process execution procedure and model partition method. We demonstrate how the workflow model in the real life example is partitioned and executed among multiple sites. Finally we conclude the paper and point our future research issues.

2. Scenario

In this section we will give a real life example to illustrate the usage of distributed WfMSs.

XBike is a company offering bike customization services. As a small and medium sized enterprise(SME), XBike concentrates on its core competence in designing/delivering various customized bikes, and outsources its production and logistic business to its partner on the grid.

Although we still use the term department to indicate the performers of each individual tasks, we emphasize that actually the stock, the production and the logistics department are independent partners collaborate with XBike based on contracts on the grid.

 destinates whether this order is feasible or not. If it is feasible, an order confirmation letter is sent to the customer, and the bike will be produced by the production department, packaged and finally delivered to the customer by the logistics department. Otherwise if the order is not feasible due to some reason, an order modification letter with suggestions will be sent to the customer and the process terminates.

Fig. 1 illustrates the bike customization process of XBike. When a customer wants to place an order, he visits XBike’s website and fills in the required information, including the customer information, bike information and some extra specifications. Then a bike customization process starts. First, the sales department checks the order and completes possible missing fields. Afterwards three tasks are performed in parallel: the financial department calculates the price, the stock department checks the stock, and the design department checks the technical feasibility. After all these three steps completes, the system decides whether this order is feasible or not. If it is feasible, an order confirmation letter is sent to the customer, and the bike will be produced by the production department, packaged and finally delivered to the customer by the logistics department. Otherwise if the order is not feasible due to some reason, an order modification letter with suggestions will be sent to the customer and the process terminates.

We can see that traditional WfMS is not suitable to handle the workflow process illustrated above. First of all, the process itself is distributed among geographically dispersed business partners, and the involved workflow applications are inherently distributed. Secondly, the reliability of the centralized system cannot be guaranteed since there can be a single point of failure. Furthermore, when performing data-intensive tasks, the central workflow server must retrieve a large volume of data from a remote site and manipulate them in the server side; this can bring about unnecessary data transfer overhead.

In Section 3 we will come up with a novel architecture, the components of it, and its advantages. We can see that this architecture address well the issues we mentioned in last paragraph.

3. System Architecture and Execution Procedure

3.1 System Architecture

In order to support distributed workflow execution and to provide a more scalable and agile platform than
existing approaches did, we propose a novel distributed workflow execution architecture (see Fig. 2).

The main feature of our architecture is:

1. No central control. Each workflow process is instantiated at the site where the first task of the process is executed. During its execution, the process is partitioned into fragments which will be handled by some other sites. By this means the process can be handled by multiple sites, which makes the system more scalable and real parallelism can be achieved.

2. More flexible task allocation scheme. The execution sites for tasks are designated before the task is going to be executed, so the task allocation is just in time.

3. Reduced information transfer between sites. By sending fragments to designated sites, we need not transfer a large volume of data to the workflow server’s site, and this can greatly improve the efficiency in data-intensive processes.

Our architecture is built based on the P2P scheme, which means that the components at each site are identical. We call all the components at one site as a workflow suite, and the main components of each workflow suite are shown as follows.

**Process modeler**

Process Modeler allows the users at this site to build workflow models. For example, the Sales Department of XBike can build an order processing model at the workflow suite at its own site.

Here we give a brief introduction to the workflow model we adopted. We extend the WF-net [4] proposed by Van der Aalst to model the data and pre-conditions for tasks. Our model, known as CWF-net [7], is a kind of colored Petri net with the structure characteristics of a WF-net. The color of token identifies the process instance as well as the relevant data items. There is at most one token in each place, each arc expression represents the initial token in the source place, and the guard functions on transitions are used to model the conditional dependencies between tasks.

**Task Allocator**

Task Allocator receives task allocation requests and allocates tasks to specific sites at which they will be performed. When a workflow model is being built, each task is associated with two set of sites, i.e., the Suggested Sites and Mandatory Sites. As their names indicate, the Suggested Sites are the sites which the task are advised to be allocated to, while the Mandatory Sites are the sites which the task are obliged to be allocated to.

For example, the task Tech. Check is suggested to be allocated to one of the several sites in the Design Department, while the task Send confirmation letter is mandatory to be allocated to the site at which the email server locates.

**Task monitor**

Task monitor keeps a log on the allocated sites for each task. We stress that although the proposed architecture is based on P2P collaboration, the execution track (i.e., the execution site of each task) must be integrally stored on some site (usually the initiating site of the workflow process). This module is useful when tokens (which representing workflow relevant data and control privilege) should transfer from one fragment to another in the AND-join case, as we’ll see later. More importantly, this log is quite necessary in collaborative environment in grid because of the charge and security issue.

**Task client**

Task Client maintains a task list for each site, so workflow users at this site can access this task list and manipulate tasks in the list.

**Fragment pool**

Fragment pool keeps all the information about the fragments allocated to this site.

**Fragment manager**

Fragment manager serves as the engine to drive the workflow process. When a new fragment is put into the fragment pool, the fragment manager checks whether the first task of this fragment can be executed at once. If the condition is valid (for example, in the AND-join case, all the preceding tasks have been completed), the fragment manager put this task to the task list to be executed. When this task is completed, new succeeding fragments will be generated and sent to the fragment pools at their designated sites. By this means the execution is propagated until all the tasks have been accomplished.
In the next sub section we’ll present the execution procedure in detail.

3.2 Distributed Workflow Execution Procedure

The distributed workflow execution procedure is explained as follows.
1. A workflow model is built on some site.
2. When a new process instance is to be established, the original fragment is sent to the site designated to execute the first task. And this site acts as the management site of this process instance, it maintains the task execution track and sometimes it acts as the task allocator.
3. Task execution.
   3.1. The site executes the first task of the fragment, and then this fragment is no longer active.
   3.2. Report to the task monitor.
   3.3. If there are un-executed tasks remained, partition the remaining part to construct new active fragments.
4. Send fragments to their designated sites.
   4.1. Allocate one site for each new fragment
   4.2. Send each new fragment to its designated site
5. For each fragment, follow Step 3 and Step 4.
6. This procedure terminates when there are no active fragments.

4. Workflow Model Partition Method

In this section we will give a dynamic workflow model partition method for the well-structured acyclic WF-nets, and show how it can be utilized in the decentralized workflow execution architecture. Due to space limit we are not going to elaborate on the model partition algorithms and implementation details. Instead, we concentrate on the concept and method we use for model partition and execution.

To make it simple and clear, first we introduce some concepts which will be used later in the following part of this paper.

- **Fragment**
  A fragment is a partition of a workflow model, the first task (or transition) of this partition is called the source transition. A fragment contains one source transition, all the transition reachable from this source transition and all the input and output places of all these transitions.

- **RSF**
  In a fragment $F$ we define all the transitions reachable from a given transition $t$, together with the input/output places of these transitions, as the Reachable Sub Fragment of $t$, denoted as $RSF(t, F)$.

**TRRSF**

The Transition Restricted Reachable Sub Fragment of $F$, started from transition $t$, restricted by transitions in $T_r$, is denoted as $TRRSF(t, F, T_r)$. Simply speaking, $TRRSF(t, F, T_r)$ is to cut the parts reachable from all the transitions in $T_r$ from $RSF(t, F)$. This concept is used to avoid information redundancy when AND-join/AND-split block is encountered.

Now we give a brief explanation of the control flow of the dynamic workflow model partition method, which partitions the workflow model into fragments dynamically.

First let’s discuss the situation that the source transition $t_s$ of $F$ has only one output place $p$. If $p$ has only one output transition, we can just cut off $t_s$ from $F$ and get a subsequent fragment. Else if $p$ has multiple output transitions, then each of the output transitions is the source transition of a new fragment, so multiple fragments are obtained. Note that only one of them can really be enabled and executed since they are mutual exclusive.

Another situation, which is more difficult to tackle, is that when $t_s$ has multiple output places. In this situation each output place forms at least one new fragment, and these fragments are to be executed in parallel. Since all these parallel fragments will be executed and will be synchronized at some task, the information of the tasks after the AND-join transition is not necessarily carried by all parallel fragments. So we take some measure to avoid information redundancy when multiple fragments are generated, i.e. we introduce the idea of Transition Restricted Reachable Sub Fragment, and when we do model fragmentation, the set $T_r$ is updated constantly to prohibit the unnecessary spanning of sub fragments.

5. Decentralized Workflow Execution for the Real Life Example

In this section we’ll demonstrate how the bike customization workflow model in Fig. 1 is fragmented and performed among the multiple workflow servers (sites) located at different departments (partners), and we’ll also show how the proposed fragmentation method have increased the flexibility and productivity of the bike customization business. Fig. 3 illustrates the Petri net model of the bike customization process.
Now let’s discuss the distributed execution procedure of the bike customization workflow (see Fig. 4). When a new order is placed, a workflow fragment \( F_1 \) is created (\( F_1 \) is identical to the model in Fig. 3). \( F_1 \) is sent to one of the workflow servers in the sales department to be executed. When task \( t_1 \) is completed, three fragments are generated. First, fragment \( F_2 \) is generated, containing all the tasks follow \( t_1 \), i.e., \( RSF(t_2,F_1) \). And in \( F_2 \), task \( t_2 \) is the AND-join transition of \( t_1 \), then \( t_3 \) is added to \( T_s \), i.e., \( T_s = \{ t_3 \} \). So two fragments (i.e., \( F_3 \) and \( F_4 \)) are generated by the function \( TRRSF(t_3,F_2,T_s) \) and \( TRRSF(t_4,F_2,T_s) \), respectively. As we’ve seen, \( TRRSF(t_5,F_3,T_s) \) contains all the tasks reachable from \( t_5 \). \( F_5, F_3 \) and \( F_4 \) are sent to the financial, stock and design department respectively, where they will be handled by three workflow servers and thus real parallelism is achieved. Note that \( F_3 \) is handled by a different company to which the production business is outsourced.

We know that \( t_2, t_3 \) and \( t_4 \) are all data-intensive tasks, i.e., when performing these tasks, a large volume of data (the financial, stock and technical data) is needed. In traditional mode, the central workflow server must retrieve these data from a remote site and manipulate them locally. While in the dynamic fragmentation mode, we just forward the fragment to the data site to be executed, by this means unnecessary data transfer is avoided and further more, data security is guaranteed. When \( t_2, t_3 \) and \( t_4 \) have all been completed, fragment \( F_5 \) is generated. And when task \( t_5 \) in \( F_5 \) is completed, \( F_6 \) and \( F_7 \) are generated. Since \( F_6 \) and \( F_7 \) are mutual exclusive, only one of them will eventually be executed according to whether the order is feasible or not. So when \( t_5 \) is completed, only the viable fragment will be forwarded. If the order is feasible, \( F_8 \) is generated. And when task \( t_7 \) is completed (order confirmation letter is sent to the customer), three fragments (i.e., \( F_8, F_9 \) and \( F_{10} \)) will be created. \( F_8 \) and \( F_9 \) are sent to the production department, where the bike is produced and packaged. At the same time, \( F_{10} \) is sent to the logistics department, where the transportation is arranged. \( F_{11} \) is generated when task \( t_7 \) and \( t_8 \) are completed. Similarly, \( F_{12} \) is generated when task \( t_{10} \) is completed (the bike is produced and packaged), and then the bike is delivered to the customer by the logistics department.

With this example, we can see that the dynamic model fragmentation method can help XBike company to coordinate its business processes which span over several partners. And this approach has increased the flexibility of workflow execution, reduced data transfer overload and enhanced the data integrity.

6. Conclusion

In this paper we investigate how distributed workflow technology can be used in virtual enterprises in grid environment. We propose a novel distributed workflow execution architecture for virtual enterprises, and discuss key technologies in implementing this architecture, i.e., the procedure of distributed process execution and the model partition method. Through a real life example we demonstrate how to use the proposed architecture and method to support distributed workflow execution in virtual enterprises.

Future research issues include developing a prototype system based on grid platform (for example, Globus [9]), and make further investigations on the model partition policy, i.e., how to partition the model and allocate task to different sites to improve system performance. We also want to do some formal analysis on the decentralized workflow model in future work.
Fig. 4 Dynamic model partition of the bike customization process

Acknowledgements

This work was granted by National Science Foundation of China (No. 60274046).

References


