Available online at www.sciencedirect.com
ScienceDirect
Computers in Industry 58 (2007) 381–391
COMPUTERS IN INDUSTRY
www.elsevier.com/locate/compind

Dynamic workflow model fragmentation for distributed execution

Wei Tan *, Yushun Fan

Department of Automation, Tsinghua University, 100084 Beijing, PR China

Received 18 February 2006; accepted 14 July 2006
Available online 30 August 2006

Abstract

Workflow fragments are partitions of workflow model, and workflow model fragmentation is to partition a workflow model into fragments, which can be manipulated by multiple workflow servers. In this paper a novel dynamic workflow model fragmentation algorithm is proposed. Based on the well-known Petri net formalism, this algorithm partitioned the centralized process model into fragments step by step while the process is executed. The fragments created can migrate to proper servers, where tasks are performed and new fragments are created and forwarded to other servers to be executed in succession. The advantages of the proposed dynamic model fragmentation method include the enhanced scalability by outsourcing the business functionalities, the increased flexibility by designating execution sites on-the-fly, the avoidance of redundant information transfer by judging their pre-conditions before forwarding fragments, etc. An industrial case is given to validate the proposed approach. Later some discussions are made on the correctness of the algorithm and the structural properties of the workflow model. Finally the future research perspectives are pointed out.

Keywords: Distributed workflow; Dynamic workflow model fragmentation; Petri net

1. Introduction

Workflow management is the key technology for the coordination of various business processes, such as loan approval and customer order processing [1]. By setting up the process model and enacting it in the workflow server, a workflow system can help to streamline the business process, deliver tasks and documents among users, and monitor the overall performance of the process.

Traditional workflow systems are often built on the client/server architecture, in which a single workflow server takes the responsibility for the operation of the whole process. Meanwhile, this sort of centralized systems may bring about many disadvantages. First of all, with an increasing need of relocating entire business functions to either self-owned or third-party service providers, business process outsourcing (BPO) has been the trend in management as well as IT field. When an company is leveraging technology vendors to provide and manage some of its enterprise applications, its business process may be distributed among geographically dispersed business partners; therefore 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. Last but not the least, the performance of the centralized system may be drastically degraded when there are too many process instances to handle.

The aim of distributed workflow execution is to separate one integrated workflow model into small partitions and allot them to different servers to be executed. To solve the difficulties that centralized workflow system cannot overcome, many distributed workflow systems have been designed from different approaches.

In this paragraph we give a brief introduction to the related work, a detailed comparison of these works with ours is given in Section 6. 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 [10]. The Exotica project [2] proposes a completely distributed architecture in which a set of autonomous nodes cooperate to complete the execution of a process, with persistent message queue as its information transmission technique. METEOR [3]

* Corresponding author. Tel.: +86 10 6277 6211; fax: +86 10 6278 9650.
E-mail addresses: tanwei@mails.tsinghua.edu.cn (W. Tan), fanyus@tsinghua.edu.cn (Y. Fan).

0166-3615/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2006.07.004

and Mentor [4] project are developed with similar approaches. Aalst introduces his well-known WF-net model to the inter-organizational paradigm [5]. In [6], an agent-based workflow management system is proposed. The work in [7–10] has shown the use of the mobile agents in distributed workflow execution, the mobile agents which carry parts of the process information can migrate from host to host to execute the workflow tasks. In [11,12], innovative approaches to support decentralized process enactment with the Peer to Peer (P2P) technology are presented.

However, the research work in this field mainly focuses on the design of the system architecture and the implementation technique based on specific communication mechanisms. As far as we have known, little attention has been paid to the formal method of workflow model fragmentation.

Workflow model is the basis for workflow execution. In distributed workflow execution paradigm, the whole process is to be executed at multiple sites instead of a single one. Therefore, the workflow model must be partitioned into small parts and transferred to their designated sites. We call these small parts of a workflow model *fragments*, which carry adequate information, so that they can be manipulated by any given workflow engine. *Workflow model fragmentation* is to partition a workflow model into fragments. We emphasize that model fragmentation is the basis for distributed workflow execution.

In this paper we propose a Petri net-based approach for dynamic fragmentation of a workflow model. Our approach is based on the well-known Petri net formalism. We partition the centralized process model into fragments step by step while the process is executed. The fragments created can migrate to proper servers, where tasks are performed and new fragments are created and forwarded to other servers to be executed in succession. The advantages of the proposed dynamic model fragmentation method include the enhanced scalability by outsourcing the business functionalities, the increased flexibility by designating execution sites on-the-fly, the avoidance of redundant information transfer by judging their pre-conditions before forwarding fragments, etc.

This paper is organized as follows. In Section 2, the problem to be solved in this paper is formulated. The workflow model, the centralized and distributed architecture, and some specifications of workflow fragment are introduced here. In Section 3, the dynamic fragmentation algorithm, i.e., the algorithm to create fragments during process execution is presented. In Section 4, a real case is given to illustrate the advantage of the proposed approach. In Section 5, some discussions are made. Section 6 summaries the related work and compared their approaches with ours. Section 7 concludes the paper and gives some research perspectives.

2. Problem formulation

2.1. Centralized workflow model

A centralized workflow model is a pre-requisite for distributed workflow execution. In this paper we adopt WF-net [13] proposed by Van der Aalst, as the centralized workflow model. WF-net is a special class of Petri net, which prevails in workflow modeling field because of its graphic nature and theoretical foundation. We do not use high-level Petri nets (colored Petri nets [14], for example) because in this paper we mainly focus on the issue of structural partition. At the same time, we acknowledge the need for using colored Petri net when data or resource issue is further considered, and for workflow modeling with colored Petri nets, one can refer to [15].

In this paper, a WF-net is denoted as a tuple (P, T, A), in which P is the set of places, T is the set of transitions, and A is the set of arcs. We assume that the centralized workflow model is a well-structured and acyclic WF-net, because it is reasonable to assume that in the distributed workflow paradigm, the centralized model is well structured and contains no loop. Well-structured property of a WF-net implies the balance of AND/OR-splits and AND/OR-joins, i.e., alternative flows created via an OR-split should also be joined by an OR-join; parallel flows created via an AND-split should also be synchronized by an AND-join. The definition of well-structured WF-net can be found in [16]. Acyclic property of a WF-net means that the workflow model contains no recursive flows. Meanwhile, we also mention how to deal with cyclic models in Section 4. For the properties of WF-net, one can refer to [13,16], and for the basic definitions of Petri net, one can refer to [17].

2.2. Centralized and distributed workflow execution

In traditional centralized workflow management system, there is one central workflow server takes charge of the operation of the overall process, so the workflow engine must communicate with each task performer, deliver necessary information and retrieve the outcome of each task (see Fig. 1(a)).

With the need of distributed workflow execution, many approaches have been proposed, ranging from server clusters to radically distributed architectures [10]. In this paper we present a novel view on this problem. We classify the distributed workflow execution paradigms into two categories according to the model fragmentation method used, i.e., the static paradigm and the dynamic one. In the static fragmentation paradigm [18], before the process is initiating, each task in the workflow model is designated to one workflow server (site) at which it is going to be executed. By this means the process model is naturally separated into several fragments. For example, in the workflow process in Fig. 1(b), tasks t_1 and t_2 are designated to *server* 1, t_4 is designated to *server* 2, t_3 and t_5 are designated to *server* 3, and the rest are designated to *server* 4. Thus, the workflow model is naturally divided into four fragments, i.e., f_1, f_2, f_3 and f_4.

In the static paradigm, the execution site of each task must be determined before the initiating of a process. Obviously it lacks flexibility.

Another paradigm is the dynamic one. It is stimulated by the idea that a workflow process instance can migrate to one server, executing the immediate tasks, partitioning the remaining part, and forwarding the remainder to the next servers. Generally speaking, the model is fragmented step by step with the execution of the process.