Journal of Systems Architecture 54 (2008) 944–956

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Execution coordination in mobile agent-based distributed job workflow execution

Yuhong Feng, Wentong Cai [*]

Parallel and Distributed Computing Centre, School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 16 April 2007
Received in revised form 8 January 2008
Accepted 7 April 2008
Available online 26 April 2008

Keywords:
Grid computing
Distributed job workflow execution
Mobile agent
Execution coordination

ABSTRACT

Mobile agent-based distributed job workflow execution requires the use of execution coordination techniques to ensure that an agent executing a subjob can locate its predecessors' execution results. This paper describes the classification, implementation, and evaluation of execution coordination techniques in the mobile agent-based distributed job workflow execution system. First, a classification of the existing execution coordination techniques is developed for mobile agent systems. Second, to put the discussion into perspective, our framework for mobile agent-based distributed job workflow execution over the Grid (that is, MCCF: Mobile Code Collaboration Framework) is described. How the existing coordination techniques can be applied in the MCCF is also discussed. Finally, a performance study has been conducted to evaluate three coordination techniques using real and simulated job workflows. The results are presented and discussed in the paper.

1. Introduction

Complex scientific applications can generally be viewed as job workflows in which subjobs (i.e., nodes or activities) represent application components and dependencies represent the interactions between the components. In a Service Oriented Architecture (SOA), common application components are usually deployed as services. A workflow engine coordinates the data and control flow amongst multiple services and allows the enactment of workflows to take advantages of distributed services and resources.

Data intensive scientific applications, such as bioinformatics [21] and digital imaging survey (e.g., Sloan Digital Sky Survey (SDSS) [1]), often involve diverse, high volume, and distributed data sets. They can be generally expressed as a workflow of a number of analysis modules, each of which acts on specific sets of data and performs cross multidisciplinary computations.

To reduce the communication overhead caused by data movement and to provide decentralized control of execution during the workflow enactment, the Mobile Code Collaboration Framework (MCCF) is developed to map the execution of subjobs to the distributed resources and to coordinate the subjobs' execution at runtime according to the abstract workflow provided by users [16]. Light-weight Mobile Agent (LMA) [6] and Code-on-Demand (CoD) [18] techniques are adopted in the development of the MCCF, so that an analysis module in data intensive scientific applications can be executed at a computational resource close to where the required data set is located.

The MCCF, which does not have a centralized engine, is different from the existing scientific workflow engines (e.g., Condor's DAGMan,[1] SCIRun[2]). When multiple data independent subjobs can be executed concurrently, replicas of an exiting LMA will be generated so that there is one LMA for each subjob. The LMAs will then be migrated to the different computational resources for the execution of these data independent subjobs in parallel. Because of the data dependencies in a job workflow, before an LMA executes a subjob, it needs to locate the execution results of its predecessors. When multiple concurrently executing subjobs have a common immediate successor, only one of the corresponding LMAs should be selected for the latter's execution. Others should be discarded. Due to the lack of a centralized engine, execution coordination is therefore required in the MCCF. So, the objective of the paper is to investigate and evaluate execution coordination technique for the MCCF.

Execution coordination of mobile agents is a well researched area. There are many techniques that have been developed. Section 2 gives a classification of these techniques according to how they can be applied in the mobile agent-based distributed job workflow execution. To further analyze their applicability, the MCCF and how these coordination techniques can be implemented in the MCCF are presented in Section 3. A comparative, experimental study of these coordination techniques is presented in Section 4. Section 5 concludes the paper and outlines the future work.

2. Classification of execution coordination techniques

According to how a subjob obtains the locations of its predecessors' execution results, the execution coordination

* Corresponding author. Tel.: +65 67904600; fax: +65 67926559.
E-mail address: aswtcai@ntu.edu.sg (W. Cai).

[1] http://www.cs.wisc.edu/condor/dagman/.
[2] http://software.sci.utah.edu/scirun.html.

1383-7621/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2008.04.011

techniques for mobile agnets (MAs) can be classified into *indirect* and *direct* coordinations. In indirect coordination, agent interacts with a shared third party for subjob result sharing [8]. There is no explicit communication between two mobile agents executing subjobs. That is, a subjob obtains the locations of its predecessors' execution results through accessing the shared memory. Indirect coordination can be further categorized into *co-residing coordination* and *global shared memory based coordination*.

When the co-residing coordination technique is used, two MAs interact only when they co-reside on the same host. MA interaction is performed either through a shared local memory associated with the hosting environment (e.g., *blackboard based* [10], *whiteboard based* [5], and *meeting oriented* [10] techniques) or through introducing an additional local agent in the co-residing host (e.g., *constrained rendezvous coordination* proposed in [9]). These kinds of coordination techniques are fully decentralized. The increased network traffic produced by highly communicative agents, e.g., negotiating agents, in large distributed systems can be reduced by allowing agents to meet at the same host before commencing communication. A runtime decision making between the remote communication and the migration has been proposed in [29], using the prediction model inspired by the inductive reasoning and the bounded rationality principles [4]. However, they may not be suitable for data-intensive computations. During the course of a job workflow execution, a MA may need to communicate many other MAs for the locations of the required subjob execution results, especially when a subjob has a large set of predecessors. Compared to the direct coordination techniques (to be discussed later), requiring two distributed MAs to move to the same host for coordination may introduce more overhead for the job workflow execuiton.

Global shared memory-based coordination can be classified into *middle agent-based coordination* [24] (which is also named as *client/server-based coordination*) and *Linda like coordination* [10]. LIME [28] is an extension of the linda like approach. It adopts an effective context-awareness [7] mechanism and facilitates an agent with an effective responding to context changes so as to improve its adaptability. When these coordination techniques are applied, after a subjob completes its execution, the location of its result needs to be published in a global shared memory. The subjob's successors can then obtain the location through accessing the shared memory.

When a direct coordination technique is used, MAs interact with each other directly or with the resources that belong to the interacting MAs [8]. When direct coordination takes place, an explicit communication between agents is initiated, i.e., the mobile agent must explicitly name its partner for a communication before it takes place. After a subjob completes its execution, its corresponding MA will notify partner MAs the location of the subjob's execution result, so that the subjob's successors can locate the result. Depending on the mechanism for locating communication partners, existing direct coordination techniques are classified into *centralized lookup infrastructure-based coordination* and *decentralized coordination.*

In centralized lookup infrastructure-based coordination, a centralized server is used for MAs' location update and retrieval. An MA keeps known partners' location information. It contacts the partners using the known locations first, and contacts the centralized server for the up to date partner location only when it finds that its partner's location is out of date (e.g., when the partner is unreachable). According to how an MA locates known partners, this type of coordination techniques can be classified into *home-based coordination* [27], *forwarding-based coordination* [12] and *hierarchical-based coordination* [32].

- The home-based approach simply adopts a centralized discovery server to look up the agent's current location. The centralized discovery server is named as "home" and maintains a dynamic name and location database of the MAs.
- The forwarding-based approach keeps a forwarding pointer to the next host of its itinerary on the current host of the MA. Each sender has to know its communication partner's original host, on which the communication partner is created. A global server is usually used to maintain this information. The frequent updating of location caused by agent migration is eliminated in this method, which reduces traffic around the server. There are three important implementations of forwarding-based coordination: *sendbox-based scheme*, *proxy-based scheme*, and *mailbox-based scheme*. In the sendbox scheme [26], each computational resource exploits a sendbox to offer the message sending and forwarding capability for current resident agents. In the proxy-based scheme [19], each agent is associated with a proxy. If an agent, say sender agent, wants to communicate with another agent, say receiver agent, it must first obtain a proxy to the receiver agent. The proxy of the receiver agent routes the message to its corresponding agent.In the mailbox-based scheme [12], each agent is equipped with a mailbox that buffers the messages sent to it. The mailbox itself is also a mobile object. But it is a reactive mobile object with less migration frequency than the MA and can be located on a different host to where the MA is located. The sender sends messages to the receiver's mailbox, and the mailbox forwards the message to MA based on the pull or push mechanism. A centralized server can be used to keep track of the location of the mailboxes. An improved approach has been recently proposed to enable the urgent message get priority processing for mailbox-based approach [13].
- The hierarchical-based approach organizes the discovery servers in a hierarchical fashion, e.g., a tree in which MAs are treated as leaves. Internal nodes of the tree maintain names of agents with their corresponding locations.

The decentralized coordination includes *flooding-based coordination* [20], *distributed hash table (DHT)-based coordination* [22], and *contact list-based coordination* [30]. When these coordination techniques are adopted, an MA locates its partners using either locally maintained information or a flooding technique.

- Using the flooding-based approach, the sender broadcasts an agent discovery request to all of its neighbors, and then the discovery requests will be generated by these neighbors until the destination partner agent is located. This coordination technique generates excessively flooding messages which will degrade the overall system's performance considerably and cause a large amount of traffic over the network.
- Using the DHT-based approach, a community is formed for each group of cooperative agents. An MA would keep track and maintain partial states of a global structure which represents the member location of the community. No discovery infrastructure is involved. However, with dynamic MA replication, disposal, and migration,[3] the maintenance of the DHT overlay is costly. In addition, the average number of hops required for sending a message in this approach is in the order of $\log n$ (where n is the number of computational resources), this can increase the time for message delivery.
- Using the contact list-based approach, each MA maintains a list of all its partners' locations. Before an MA is migrated or discarded, it will notify its partners so that they can update their contact lists accordingly. In order to ensure the message is received by the partners, the MA cannot be migrated until it gets

[3] These concepts will be further described in Section 3.1.