Framework for Workflow Parallel Execution in Grid Environment

Lican Huang

Institute of Network & Distributed Computing,
Zhejiang Sci-Tech University,
Hangzhou, P.R. China

Ministry's intelligent virtual research center, Zhejiang University
{huang_lican@}@yahoo.co.uk

Abstract. This paper presents a framework for Web Service based workflow optimisation in Grid environment. In this framework, parallel service is introduced as adaptor and container of parallel block of workflow. In parallel service, workflow is divided into multi sub-workflows, which are dispatched and executed in parallel at different workflow engines in different grid nodes. By this strategy, the performance of workflow can be enhanced greatly. In this paper, we also give a strategy how to guarantee the correctness in complicated network environment by redundant computing.

Keywords: workflow optimisation, Grid, e-Science, workflow Parallelization.

1 Introduction

Web service technologies are widely used in industry and research institutes. This brings in a variety of Web service composition languages and a variety of workflow engines available, some of which implement the same description language, but differ in their configuration, specific deployment description, and performance[1]. Grid[2] aims to solve resource and information sharing problems between virtual organizations. A variety of workflow management systems for Grid computing have been reported in literature. These range from portal-based interface for users to connect components together, to complete systems that allow composition and deployment of a set of services. Often these systems are categorized as being a Problem Solving Environment(PSE)[3]. In many ways, a PSE is seen as a mechanism to integrate different software construction and management tools and application specific libraries, within a particular problem domain. Often PSE and Grid Computing Environment are interchangeable – as PSE research predates the existence of Grid infrastructure. A key element within a PSE is therefore the provision of a workflow engine that enables a set of services (provided by one or more scientist) to be executed, in the context of a particular application. Workflow in PSE often is called as scientific workflow[4]. Scientific workflow is the workflow technology used by scientists to access and analyze

data in scientific research. Scientific workflow often requires support for large data volumes, high throughput computation or high performance computation.

Scientific workflow usually takes long time to execute, from several hours to several months. Therefore, scientific workflow optimisation is an important issue. There are several kinds of workflow optimisation: Static optimisation by re-order of workflow[5], Dynamic optimisation by selecting optimal service from multi semantic Web Services based on machine's status such as CPU speed, CPU load monitored just before the service is executed[6] and Workflow parallelization by dividing workflow into sub workflows which are executed in multi workflow engines.

Workflow parallelization for optimisation in Grid environment is a challenge issue. We need to discover the optimal workflow engines to execute sub-workflows. The workflow engines are differ in workflow languages, configuration, auto-deployment, etc. And the performance of workflow engines are differ in the quality and workload of host machines at the pre-execution point. We also need to solve fault-resistance in the complex network environment.

We present a framework for workflow optimisation in Grid environment. The discovery of suitable workflow engines is based on domain specific P2P - VIRGO [7], which is hybrid of structural and un-structural P2P technologies. We use parallel service as a container of sub workflows, which contains workflow parallelization, workflow engine discovery,dispatcher of sub workflows, collector of sub results, and integration of sub results into final result. The fault-resistance is solved by executing multi copies of sub-workflows in Grid environment.

The structure of this paper is as follows: Section 2 describes framework for workflow parallel execution in Grid environment; Section 3 presents implementation of prototype; and finally we give conclusions.

2 Framework for Workflow Parallel Execution in Grid Environment

In the workflow applications, the parallel blocks are re-written as parallel service as Figure 1 shows. Parallel service serves as adaptor and container of parallel block in workflow script. The script describing workflow block and parameters of the block are input into parallel service as parameters. After sub workflows are dispatched and executed in multi workflow engines, the results are collected and integrated into final result, and output as parameters into the successive Web service in the workflow script.

The parallel service contains workflow parallelization, workflow engine discovery, dispatcher,collector,and integrator shown in Figure 2. Workflow parallelization parallelizes the parallel block of workflow into multi blocks of sub workflows, which can be auto-deployed and executed in workflow engines. Workflow engine discovery discovers suitable workflow engines, which are active, support for corresponding workflow languages and lightweight workload, etc. Dispatcher dispatches sub workflows into multi Grid nodes which host workflow engines. Collector collects the results sent by the Grid nodes which have executed these sub workflows. Integrator integrates these results into a whole.

Fig. 1. workflow parallelization

Fig. 2. The components of parallel service

Figure 3 shows the sequence of the framework of workflow parallel execution. The parallel blocks of workflow are transformed into parallel services in the workflow scripts at design time. Then the scripts are deployed. When client invokes workflow engine, the workflow engine will invoke services according to the script. When it encounters the parallel blocks, the engine will invoke parallel service. The parallel service first divides the parallel block into several sequential sub workflow scripts. Then, the parallel service discovers the suitable workflow engines, and then dispatches these sub workflow scripts into the Grid nodes which host the suitable workflow engines. After that, the parallel service collects the executed results by the Grid nodes and integrates these results into a whole. Finally, the parallel service sends the result to the workflow engine.

Fig. 3. The sequence of workflow parallelization

2.1 Workflow Parallelization

This module parallelizes workflow as multi sub workflow blocks. As Figure 1 shows, the block of workflow is parallelized as six sub blocks. This module also adds input and output parameters into sub workflow scripts. It concerns about analysis of workflow dependence graph. It may involve with the strategies for parallelism at multiple levels of workflow granularities in order to optimize workflow execution[8].

2.2 Workflow Engine Discovery

In the cluster of workflow engines, to find out the most suitable engine for a specific task is an important issue. Workflow engines are different in workflow languages, in the vendors and in performance, etc. The process of workflow engine discoverer is as follows: When workflow engine discoverer discovers the locations of workflow engines according to properties of required workflow engines (see Table 1) such as supported workflow languages, auto-deploy,operation system of hosts, etc., the host machines will invoke the processes of Ganglia[9] to get their performance data such as host workload, CPU speed, etc. Then workflow engine discoverer will collect information of suitable workflow engines which lists the locations, language, performance, availability, active time, failure frequency and so on for workflow engines. These workflow engines are optimally re-ordered by

their performance data. Finally dispatcher of parallel service will dispatch the workflow sub blocks into those suitable workflow engines by the rule that the workflow sub blocks are dispatched into those optimal workflow engines.

Table 1 shows some terminologies about workflow engine.

Table 1. Workflow engine description terminologies

workflow language
workflow engine
workflow engine stability
whether workflow engine is auto-deployed?
performance of workflow engine
host performance
Operation system of workflow engine host
etc.

Compared with discovering Web services with UDDI, there are no existed standards to discover the workflow engines. We here use VIRGO P2P network to discover workflow engines. VIRGO is a domain-related hierarchical structure hybriding un-structural P2P and structural P2P technology. VIRGO consists of prerequisite virtual group tree, and connections cached by least-recently used and minimum distance replacement strategies. The workflow properties are published into the nodes of VIRGO network which are the hosts of the workflow engines. The workflow engines are classified as domain catalogues. The nodes containing workflow engines join VIRGO network as the same domains as the classifications of workflow engines. The discovery of workflow engine uses VIRGO lookup protocol[7][10] to find the suitable workflow engines. The main advantage using VIRGO based distributed workflow engines discovery are that all hosts of suitable workflow engines can concurrently get performance data by Ganglia and the discovery is effective when there are a large amount of Grid nodes.

2.3 Dispatcher

When workflow parallelization has divided workflow into many sub workflows, the dispatcher dispatches these sub workflows to the suitable workflow engines. The suitable workflow engines are chosen based on the strategies as following:

1. sub workflows are dispatched into different workflow engines.
2. the workflow engines which host many Web services of sub workflow prefer to those which invoke Web service remotely.
3. the highest performance of workflow engines are preferred to be chosen. The workload, CPU speed, etc of workflow engines are obtained by monitor such as Ganglia.
4. the key sub workflows are dispatched into several copies of workflow engines to solve the fault-resistance problems.

2.4 Collector

After dispatcher dispatched sub workflows to the workflow engines, and all workflow engines deployed and executed these sub workflows, the collector will receive the results sent by the sender of those workflow engines.

2.5 Integrator

When all results are collected by collector, the integrator will integrate the results into a whole. It will produce the parameters of output as the input parameters of the successor Web services.

3 Implementation of Prototype

We have implemented the prototype of parallel service. The parallel service is a standard Web service taking as adaptor of parallel blocks of workflow. We use XML document to represent the parallel block of workflow. In this XML document, the script of workflow block and the output interfaces of parameters are described.

We have implemented two classes –MasterNode, which dispatches subworkflow, and ExecuteNode, which executes subworkflow.

Class MasterMode includes functions of parallelization, which divides parallel block into subworkflows, discoveryWorkflowEngine, which discovers the optimal workflow engines, Dispatcher, which dispatches subworkflows into the optimal workflow engines, Collector, which collects the results from workflow engines, and Integrater, which integrates the results into the whole result.

Class ExecuteNode includes functions of Receiver, which receives the message containing subworkflow script from Master Node, deployer, which automatically deploys the subworkflow script in workflow engine, Executer, which invokes sub-workflow and get the result, and Sender, which sends the result to Master Node.

The following is pseudo code of parallelServiceProxy service:

```
public String[]parallelServiceProxy(String
XMLparallelXML,String[]parameters)
{
    subworkflowList =
        masternode.parallelization(XMLparallelXML,parameters);
    workflowEngines =
        masternode.discoveryWorkflowEngine(XMLparallelXML);
    masternode.Dispatcher(subworkflowList, workflowEngines);
    collectResults= masternode.Collector( message);
    results = masternode.Integrater(collectResults );
    return results;
}
```

In the above, `discoveryWorkflowEngine` has following pseudo code:

```
public String[] discoveryWorkflowEngine( String XMLparallelXML )
{
    WorkflowEngineMeta = parse(XMLparallelXML);
    Send WorkflowEngineMeta to virgo-lookup protocol;
    workflowEnginesXML = get_response_of_virgo-lookup;
    workflowEnginesList=DecisionWorkflowEngine(workflowEnginesXML);
    /*re-order optimally workflow engines by performance data*/
    return workflowEnginesList;
}
```

The pseudo code of Receiver in Execute Node is as following:

```
public void Receiver(Message msg)
{
    subworkflowscript = parse_workflowscrip(msg);
    parameters = parse_paremters(msg);
    WebServiceURL = deployer(subworkflowscript,parameters);
    result = Executer(WebServiceURL);
    msg = modifyMessage(result);
    Sender(msg) to Master Node;
}
```

Every host machine of workflow engine installed virgo software. The functions of `discoveryWorkflowEngine`, `Dispatcher` and `Collector` in `MasterNode` class and functions of `Receiver`, `Sender` in `ExecuteNode` use virgo socket port to send and receive messages. The VIRGO node uses Tomcat as Web Server and Axis as Web service container. Every node implements Lookup protocol and VIRGO structure maintenance, and matchmaker and Register for Web Service and workflow engine. The messages are XML formed. The message type in the message is used to distinguish kinds of messages and indicates how to handle these messages.

We use BPEL4WS as experimental workflow language and Activebpel as workflow engines. The operation system for host machines of workflow engine is Linux. We have used simple example to do the experiments.

4 Conclusions

We have presented a framework for workflow parallel execution for workflow optimisation in Grid environment. In this framework, parallel workflow block is divided into multi sub-workflows, which are dispatched and executed in parallel at different workflow engines in different grid nodes. We also give a strategy how to guarantee the correctness in complicated network environment by redundant computing. We use parallel service as adaptor and container of parallel block of workflow. We insert parallel services into workflow script at design time as agents which execute multi sub workflows in parallel. Workflow engine discovery

is based on VIRGO protocol to effectively find suitable workflow engines. We also implemented the prototype of the framework. In the near future, we will use a real optimal workflow example from Bioinformatics to demonstrate the enhance effectiveness of Research via the strategy presented in this paper.

Acknowledgments

This paper is supported by Research Funds of Zhejiang Sci-Tech Univeristy (No:0604229). Author also thanks anonymous reviewers for their comments to modify the paper.

References

1. L. Huang, D. W. Walker, O. F. Rana and Y. Huang, "Dynamic Invocation, Optimisation and Interoperation of Service-oriented Workflow", CCGRID2005 WIP section, Cardiff 2005
2. I. Foster, and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *International Journal of Supercomputer Applications*, 11(2): 115-128, 1997
3. O. Bunin, Y. Guo, and J. Darlington, "Design of Problem-Solving Environment for Contingent Claim Valuation", Proceedings of EuroPar, LNCS 2150, Springer Verlag, 2001
4. Bertram Ludascher, et.al, "Scientific workflow management and the Kepler system: Research Articles", *Concurrency and Computation: Practice & Experience*, 18(10): 1039 - 1065 , 2006
5. Yash Patel, Andrew Stephrn McGough and John Darlington, "Grid Workflow Scheduling In WOSE", UK e-Science Programme All Hands Meeting 2006 (AHM2006)
6. L. Huang, D. W. Walker, Y. Huang and O. F. Rana , "Dynamic Web Services Selection for Workflow Optimization", UK e-Science Programme All Hands Meeting 2005 (AHM2005)
7. L. Huang, "VIRGO: Virtual Hierarchical Overlay Network for Scalable Grid Computing", Proc. European Grid Conference (EGC2005), in LNCS 3470, pp911-921, February 14-16, 2005, Amsterdam, Netherlands
8. Michael J Pan and Arthur W Toga, "Multi-granularity parallelization for scientific workflow management", [http://www.loni.ucla.edu/twiki/pub/Pipeline/ Research-Publications/parallelStrategies.pdf](http://www.loni.ucla.edu/twiki/pub/Pipeline/Research-Publications/parallelStrategies.pdf)
9. Ganglia, 2006 <http://ganglia.sourceforge.net/>
10. L. Huang, "A P2P service discovery strategy based on content catalogues", the 20th CODATA International Conference, Beijing, China, 23-25 October