A WEB-SERVICES-BASED P2P COMPUTING-POWER SHARING ARCHITECTURE

Chen-Sheng Wang, Po-Yu Yang, Min-Jen Tsai

Institute of Information Management

National Chiao Tung University

1001 Ta Hsueh Road, Hsinchu, Taiwan

Phone: 886-3-571-2121 ext 57417 Fax: 886-3-572-3792

cswang.iim92g@nctu.edu.tw, pyang.iim92g@nctu.edu.tw, mjtsai@cc.nctu.edu.tw

Chen-Sheng Wang, Po-Yu Yang, Min-Jen Tsai

A WEB-SERVICES-BASED P2P COMPUTING-POWER SHARING ARCHITECTURE

ABSTRACT

As demands of data processing and computing are increasing, current information system architectures become insufficient. Some organizations try to keep their systems work without purchasing new hardware and software. A peer-to-peer (P2P) model which shares the resource over the network will be proposed to meet this requirement in this paper.

In addition, this paper discusses some problems about security, motivation, flexibility, compatibility and workflow management for the traditional P2P power sharing models. Thus, we propose a new computing architecture Computing Power Services (CPS) that aims to solve the problems discovered. It utilizes Web Services and Business Process Execution Language (BPEL) to overcome the shortcomings about flexibility, compatibility and workflow management. CPS is limited to a trust network where peer and peer trust each other. Thus, the concerns about security and motivation are negated.

CPS is a lightweight Web-Services-based P2P power sharing environment, and suitable for executing computing works which are able to run in batch in a trusty network. The architecture relies on BPEL which provides a visualized development environment and workflow control management.

INTRODUCTION

In the past mainframe systems, almost everything is done by mainframe computers. Processing in the mainframe becomes a bottleneck in the information systems. It forces enterprises to keep pumping money into mainframe upgrades in order to maintain efficiency of information system under increasing processing demands. Client/server architecture is an alternative to improve mainframe system. Client/server architectures shift processing burden to the client computer. Through workload sharing, client/server systems can maintain efficiency of the information systems while reducing the budget for computing resources.

Client/server architectures have gained wide acceptance, but many companies are searching for ways to improve their processing power again without more investment in new hardware and software in a competitive market. Some people try to think about how to use current resource in the origination, like idle computers or free storage space to reach the goal. This new approach is called peer-to-peer systems. It allows users to make use of collective power in the network. The benefits are lower costs and faster processing times for everyone involved.

Since P2P model is a system that allows users to share their resources with each other over the network, it is a matter to think about what kinds of computer resources can be shared. Some computer resources, such as files system, network bandwidth and computing power are shared in some current P2P models. But there are still some problems in current P2P models, like lack of ability to customize computing tasks, workflow control management and etc.

According to these problems, the paper presents a lightweight Web-Services-based P2P power sharing environment, and suitable for executing computing works which are able to run in batch in a trusty network. The architecture relies on BPEL which provides a visualized development environment and workflow control management.

PEER-TO-PEER MODEL

The term “peer-to-peer” (P2P) refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized manner. The resources encompass computing power, data (storage and content), network bandwidth, and presence (computers, human, and other resources) [1]. Generally, there are three features in the P2P system: [2]

- Computers can now act as both clients and servers
- P2P system allows users to make use of the collective power in the network
- The benefits of P2P system are lower costs and faster processing times for everyone involved

It can combine current resources to more powerful one by using P2P model. For the origination, using P2P system can integrate existing resources to satisfy increasing demands and economize on processing power upgrades. No matter how old the computers are, how narrow the bandwidth is and how less the storage is, many little may make a mickle by using P2P system. It is what P2P model want to do.

There are two kinds of P2P systems at present. One is file sharing model, another is distributed computing. [2]

File sharing model

Content storage and exchange is one of the areas where P2P technology has been most successful. Distributed storage systems based on P2P technologies are taking advantage of the existing infrastructure. Some systems provide the user with a potentially unlimited storage area by taking advantage of redundancy. The duplication and redundancy in P2P systems help ensuring reliability of data. [1]

Napster is the first P2P file sharing application that jump started the P2P area. Napster uses the centralized directory model to maintain a list of music files, where the files are added and removed as individual users connect and disconnect from the system. Users submit search requests based on keywords such as “title,” “artist,” etc. Napster has been quite popular. It has had more than forty million client downloads and has led to numerous variants of file-sharing applications. [1] Famous model like E-Donkey, eMule and Bittorrent are other examples of P2P file sharing systems.

Distributed computing

Another model is distributed computing. This model tries to combine computing power to satisfy processing demands. It can shorten a long processing time without processing equipments upgrades. For example, in January 1999, a system with the help of several tens of thousands of Internet computers broke the RSA challenge [DES-III] in less than 24 hours using a distributed computing approach. [1]

Distributed computing is implemented in large-scale scientific researches. A famous one is SETI@home [SETI@home 2001]. Now, it has a consolidated power of about 25 Tflop/s (Thousands of Billions of floating point operation per second), collected from more than three million registered user machines. [1]

Generally, a distributed computing system works which needs to split the computational problems to be solved into small independent parts. Each of the parts is done by individual computer and the results are collected by a central server. Individual can participates in the project by downloading participant software from central server.

Distributed computing is very successfully used by P2P systems. Many systems like SETI@home, United Devices, Einstein@Home and etc, have thousands of participators and contribute much in scientific researches.

PROBLEMS ANALYSIS

Problems about distributed computing

Distributed computing can integrate computing power over network to meet high processing demands, such as large-scale scientific computing. Using distributed computing system can raise efficiency of processing and make it possible to computing complex task. But there are some problems for small scale organizations which are hard to pursue. These issues are discussed as follows.

- Security

Security of running distributed computing system is based on trusty. Participants must completely trust the research organization before they download the programs. Allowing program running on computer is greatly increases vulnerability to security breaches. A malicious attacker may add or delete files on the computer, or connect to other computers and perform illegal operations by attacking vulnerability on the computer. It is very difficult to secure P2P applications against such misuses, but if the sponsors of P2P project are famous like Intel and the University of Oxford sponsor the Cancer Research Project in UD. It can give participants enough trusty to assure them the safety of their network.

- Motivation

Participants who participate distributed computing only want to make some contribution to world and do not ask any repayment. In many companies, there are thousands of idle computers on 5:00 PM between 9:00 AM. Why do we use them to process something? Some scholars present that there may be an exclusive property in companies. These companies use their assets to do nothing that helpless to their selves. Similarly, general participants do not hope to join the projects that have commercial purposes. Some famous P2P systems like the Cancer Research Project in UD, announce their research results do not belong to any commercial originations and it can improve the participant's confidence.

- Flexibility

The participant program of the P2P distributed computing system is important. Participants must download and install it on their computers and donate their processing time. Once, if the programs need to be updated, the tightly coupled of procedure language would make it hard to update all the programs efficiently. In addition, it is impossible for participants simply to sponsor project to be computed. Some grid computing provide a command line mode to reach such a goal, but not a visual interface that people feel more friendly.

- Compatibility

Compatibility across different platforms is another problem. Some participant programs, such as UD are only works on NT-compatible platforms. There are a large number of workstations that use Unix or other operating systems. It is a pity that workstations cannot join this project due to compatibility problems. Some distributed computing systems like seti@home solve the compatibility problem by using different versions for different platforms, but it increases the cost of maintenance as many versions must be kept and updated.

- Workflow Control

What most Grid or P2P distributed computing middleware focuses on is performance, workload balance or stability but hardly workflow control management. Workflow control management can execute a complicated workflow which is composted by parallel or dependent sequence. At present, it is not natively supported by most models.

Available Solutions

According to the issues mentioned above, the paper presents some available solutions or technologies to address these problems.

Asynchronous Web Services

Web Services is a solution of distributed Services Oriented Architecture (SOA). A Web Services-Based system can inherent the features of Web Services which are loosely coupled and open standard. A distributed computing system implemented by Web Services can improve flexibility of software updates of system because of loosely coupled.

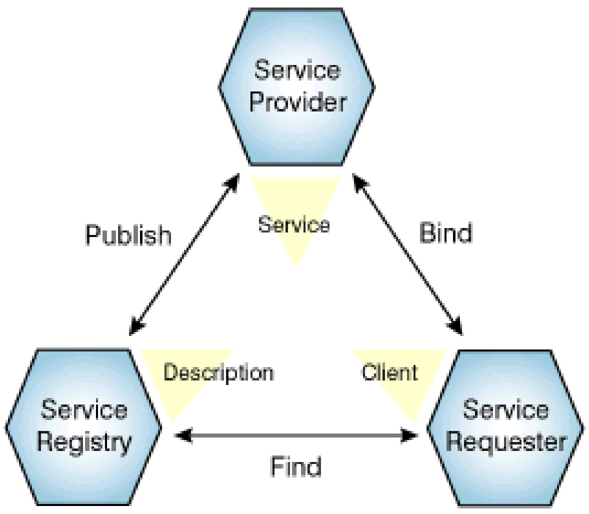

Figure 3–1 Architecture of Web Services [4]

Web Services is a message-based architecture and the interaction between services is based on messages exchanges. General synchronous Web Services is not suitable for distributed computing because it is hard to estimate the processing time of the distributed computing systems and it may cause over-time exception. So, it is an answer to utilize asynchronous Web Services to avoid over-time exception. Besides, asynchronous Web Services that response until tasks finished is more reasonable in distributed computing environment. It solves the problem of flexibility that use asynchronous Web Services to implement distributed computing system.

Business Process Execution Language (BPEL)

BPEL is a standard of Web Services composition and integrated by IBM and Microsoft from WSFL and XLANG they released before. It has characteristics like visualization, workflow control management, exception and transaction handling and compatible with Web Services. [6] BPEL is wide accepted in the industry.

It provides a sample and visual development environment that using BPEL and BPEL design tools to plan the workflow of distributed computing task. Addition to, a complicated workflow composed by parallel or dependent sequence can be executed by BPEL engine. Finally, BPEL utilized in the distributed computing systems improves the flexibility of computing task designed and provides the ability of workflow control management.

Implementation in a trusty network

Security of running distributed computing system is based on trusty and fame. So, it is easy for famous and large originations to sponsor P2P distributed computing project but not for small and medium scale businesses. The reason is that it is hard for them to be trusted by masses. No harm in thinking conversely, how about using idle computers in the origination to process what the origination want to compute. In other words, it is implementation in a trusty network.

Trusty network defined in this paper is a network where peer and peer trust each other. No matter the intranet of an origination or computer network of the friends, it can be classified as trusty network if the peers in the network trust each other. A lightweight distributed computing system is limited to implementation in a trusty network. Thus, the concerns about security and motivation can be negated.

THE ARCHITECTURE OF COMPUTING POWER SERVICES

Based on the possible solutions in last section, this paper proposes the architecture of Computing Power Services (CPS), which is a Web-services-based P2P architecture. It provides users a platform to design the business processes and control workflow of the processes by using the graphicable characteristics of BPEL. The architecture is assumed to be implemented in the trusted network to execute the computation-intensive tasks by using the idle computing power in the enterprises.

The Model of Web-Services-Based Power Sharing

The key point of CPS is how to assign the jobs in distributed computing environment. Intuitively, the computing requester should search for the computing units and give them the tasks to do. If CPS is implemented so, each computing unit will need to publish a Web service as accessing point. It will mean an application server will be necessary to host a Web service.

However, such environment will be too complicated for users to provide computation and it will discourage users to participate the project. Hence, to comply with the concept of thin client and encourage users to provide their computing power, this paper makes the computing unit as service requester and

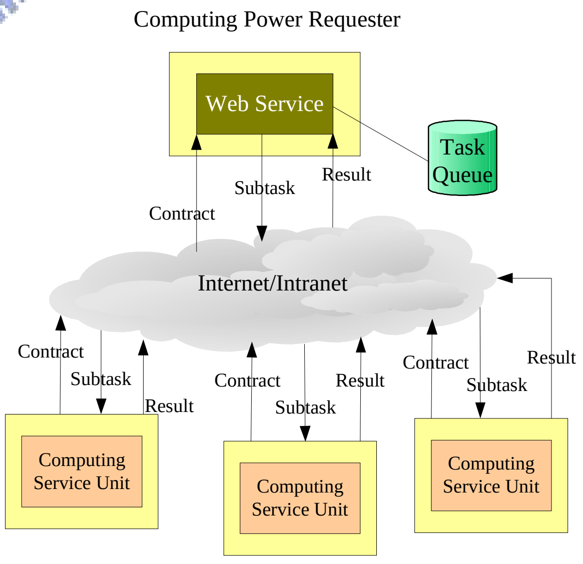

Figure 4.1 The diagram of CPS Architecture

The Roles of CPS Architecture

Because CPS bases on the architecture of Web services, it will inherit the characteristics of SOA (Service Oriented Architecture) which consists of 3 participants that are service requester, service provider and service broker. However, in order to make the program developed in CPS as thin as possible, 3 participating roles will be changed slightly to meet the requirement discussed in last section. The description of 3 roles will be explained in the following section.

l The role of Coordinator

The coordinator acts as a service broker to fairly mediate between the computing unit (service requester) and computing requester (service provider). Its main function is to maintain a list which records the URL and requirement of computing requester. This list will be created when the computing requester publishes its Web service in the coordinator. If computing unit asks for the subtasks through the coordinator, the coordinator will assign the URL of computing requester in the list to computing unit by round-robin mechanism. Afterward, the computing unit will use the specified URL to communicate with the computing requester directly.

In addition, the function of account and auditing management will be implemented at the end of coordinator. This role is corresponding to the role of UDDI in SOA.

l The role of Computing Power Requester

The requesters design their processes by a BPEL graphical environment. After designing, the requester will publish their requirement at the end of coordinator. If computing unit asks for the subtasks through the coordinator, the coordinator will assign the URL of computing requester in the list to computing unit by round-robin mechanism.

Afterward, the computing unit will use the specified URL to communicate with the computing requester directly.

In addition, the function of account and auditing management will be implemented at the end of coordinator. This role is corresponding to the role of UDDI in SOA.

l The role of Computing Unit

This role is responsible for execute computation. It will inquire the coordinator to ask for the job when it is idle. After getting back the requester's URL of Web services, it negotiates with the requester to download the task and required files for that task. Then, it starts to execute the task and respond the result to the requester when the task is finished. The whole procedure will continue until all tasks are done.

The interaction among roles and the operating procedures of CPS are described by the figure below.

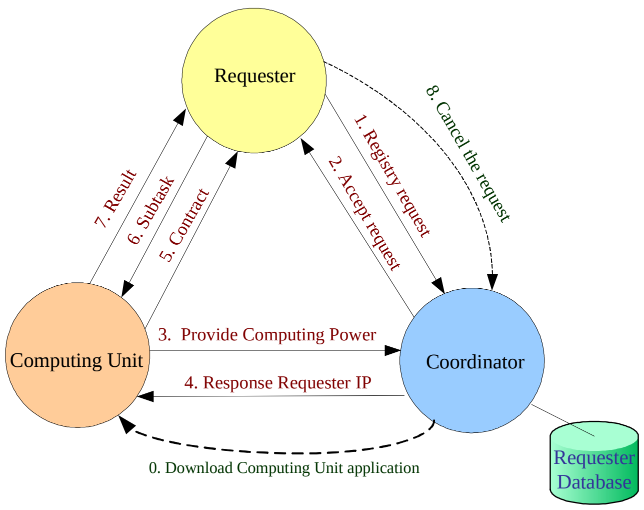

Figure 4.2 The operation diagram of CPS

The System Architecture of CPS

The figure 4-3 is the diagram of CPS architecture. By functionality, the architecture is divided into 5 layers, which are the User Layer, the Power

Sharing Layer, the Communication Layer, the Contract Layer and the Discovery Layer.

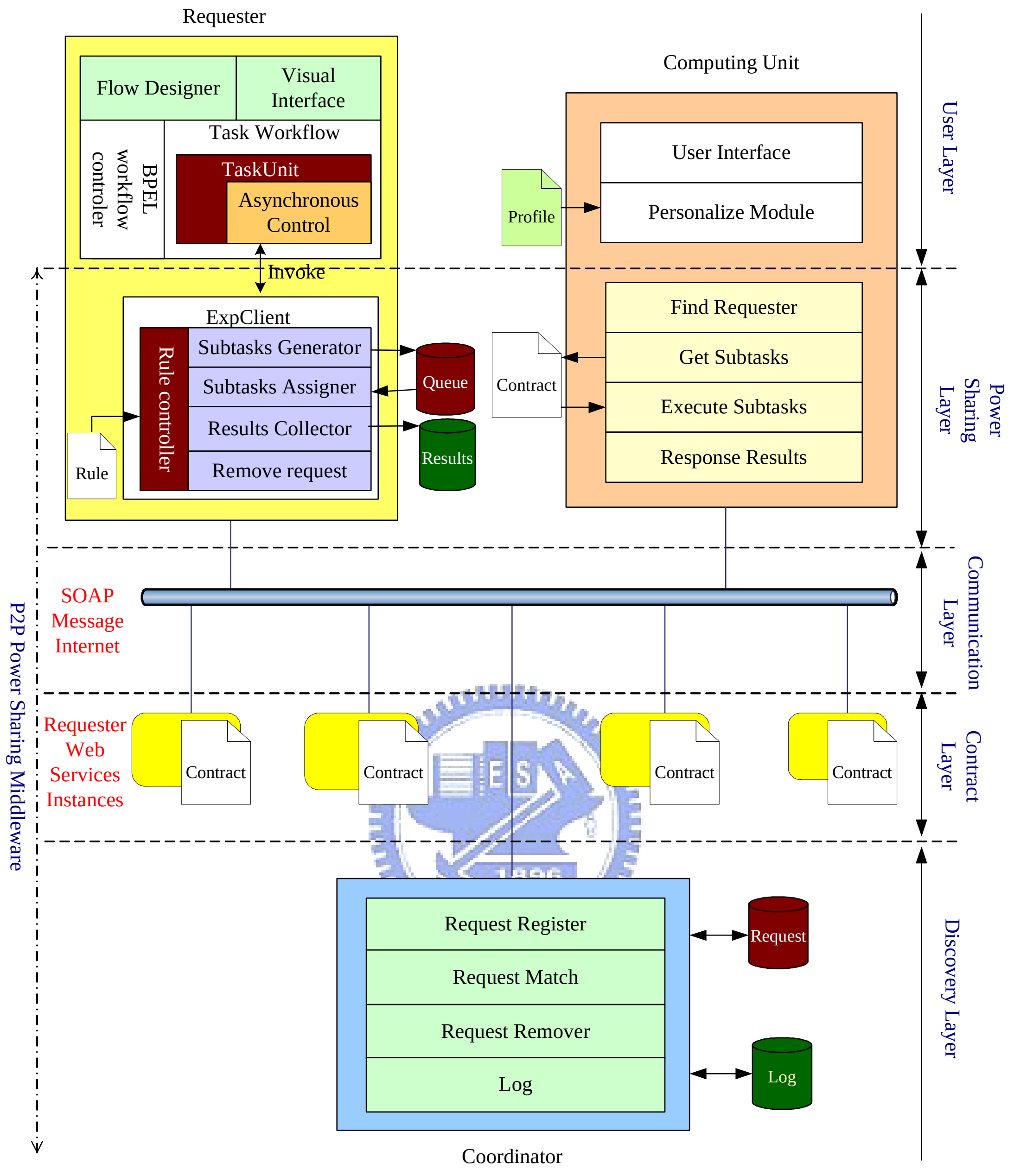

Figure 4.3 The layer diagram of CPS Architecture

P2P Power Sharing Middleware

Excluding the User Layer, the other 4 layers comprise the P2P Power Sharing middleware which is the core of CPS. Because CPS is based on Web services, the middleware is also established by the protocols of Web services as the following figure depicts.

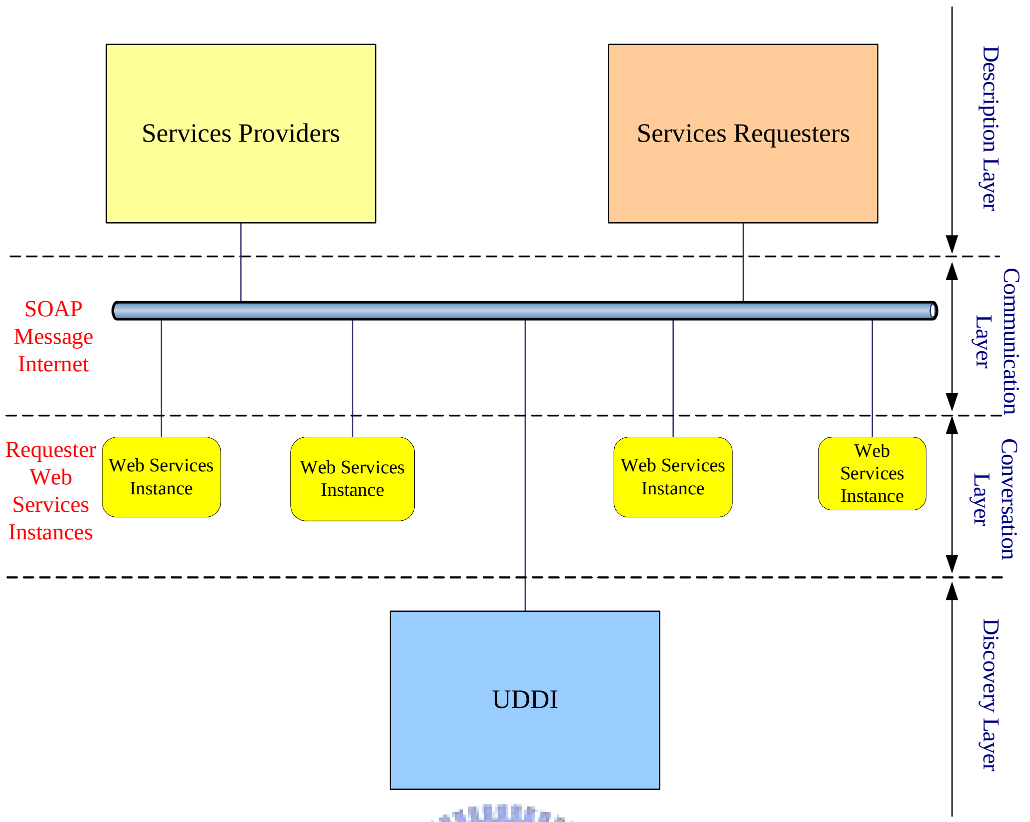

Figure 4.4 The diagram of P2P middleware in the CPS architecture

- The Power Sharing Layer

This layer corresponds to the Description Layer of Web services. It describes the interaction between the requester and the computing unit.

- The Communication Layer

This layer uses the communication mechanism of Web services, i.e. SOAP.

- The Contract Layer

The conversation between a computing unit and the requester will be defined by the contract in this layer.

- The Layer of Service Discovery

The coordinator operates in this layer as a broker agent for the requester and computing unit. The

coordinator will not involve the c

- The User Layer

The users access the whole architecture in this layer. This layer will be implemented at the end of requester and computing unit. While, at the end of computing unit, it provides an interface to control the execution of the program, it will allow user to design the BPEL process at the end of the requester. Besides, it also provide GUI interface to facilitate the designing and managing the workflow of the process.

The Interaction between User Layer and P2P CPS Middleware

By using BPEL as a language to develop the process, CPS provides the environment of visual development and the capability of workflow control. However, BPEL doesn't support the distributed computing. Therefore, this paper develops a TaskUnit program which interacts with P2P CPS middleware to address this issue.

Actually, TaskUnit is a process developed by using BPEL. It can be viewed as the process of task dispatcher to provide the capability of distributing computing. It comprises of 2 modules. While One module is to invoke a ExpClient Web service, another module will asynchronous receives the result sending by P2P CPS middleware.

The following figure 4.4 describe the interaction between TaskUnit and P2P CPS middleware.

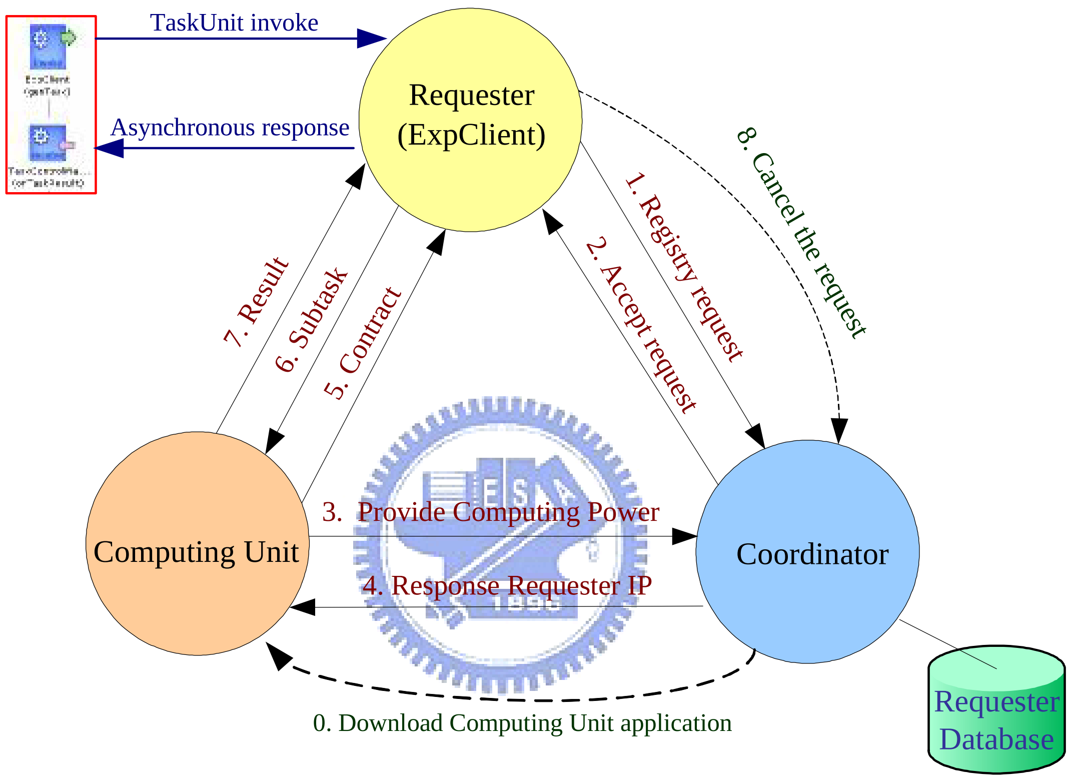

Figure 4.4 The interaction between TaskUnit and P2P middleware

The Mechanism of Exception Handling

There are two possible exceptions when CPS operates. The one is that a user closes the program at the end of computing unit, the other is that the computing unit can not finish the task before time-out. To address both exceptions, CPS employs the mechanism of task reassigning after time-out and roll-back at the end of computing end.

The Assigning Rule at the End of Requester

The flowchart of assigning subtasks at the end of requester is indicated in the figure 4.5. In general, the mechanism of assigning subtasks will distribute the unassigned subtasks to the computing unit. If all subtasks are assigned, the requester will use the mechanism of reassigning subtasks to find the time-out tasks. The length of time-out timer will be defined in the assigning rules.

Although CPS can handle the breach of contracts by using the mechanism of reassigning subtasks, it will cost more resources to redo the tasks.

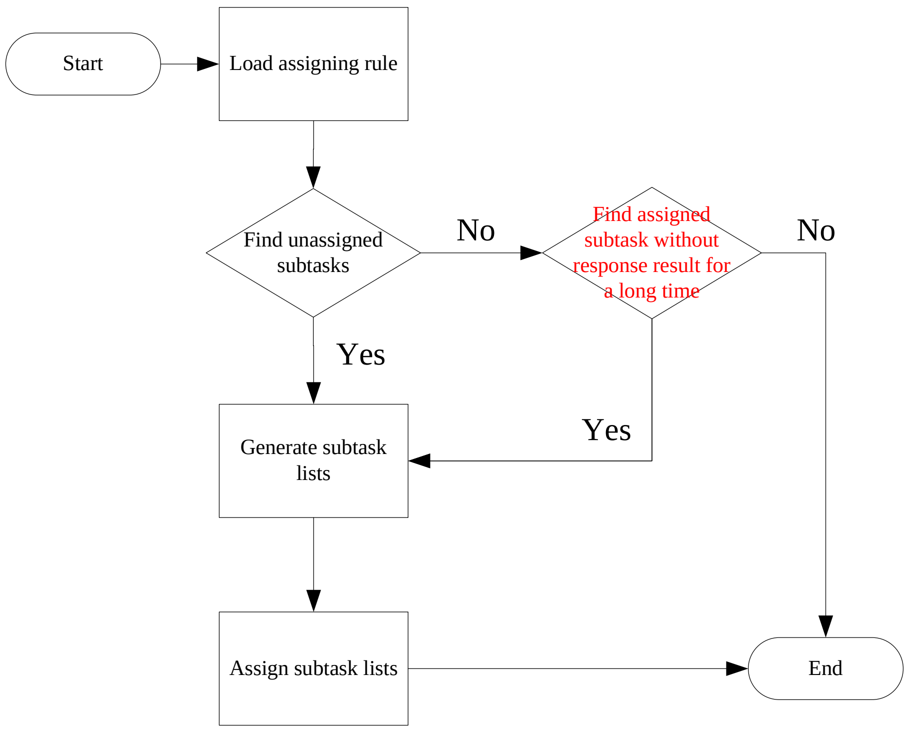

Figure 4.5 The flow chart of assigning subtasks at the end of requester

The Roll-Back Mechanism at the End of Computing Unit

As Figure 4.6 depicts, the program at the end of Computing Unit will perform computation according to the contract. After finishing the subtask, it will reply the result to the requester, terminate the contract and remove results. If the program is abnormally terminated, the contract still exists at the end of Computing Unit. Therefore, as soon as the program starts, it will verify existence of the contract. If it does, the program will remove the previous result and do computation again. Although this paper adopts the conservative way to roll back the computation, it guarantees finishing the contract.

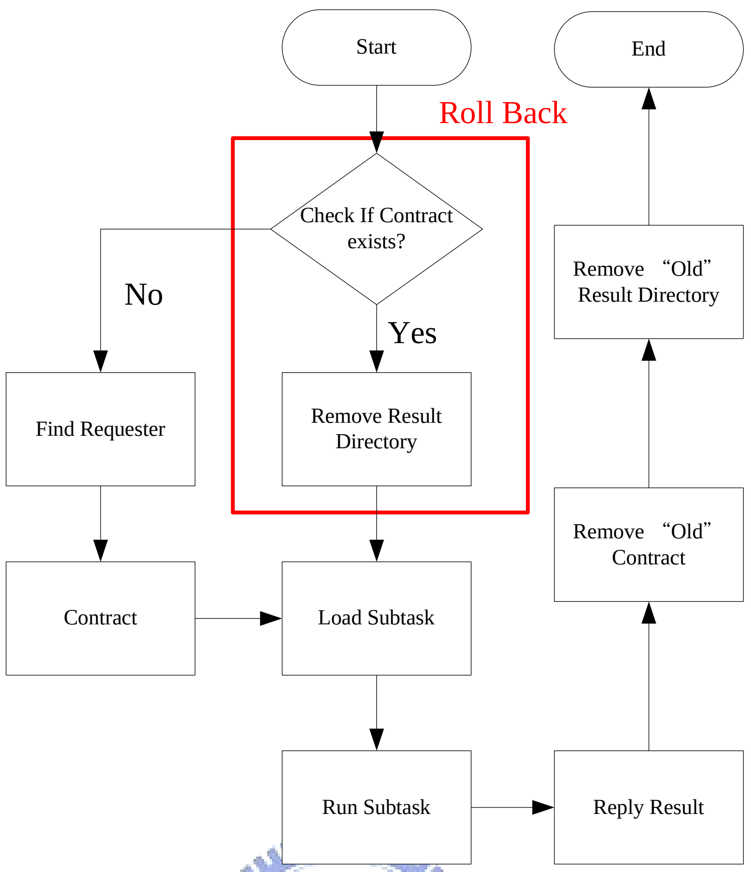

Figure 4.6 The flow chart of executing subtasks and roll-back at the end of computing unit

IMPLEMENTATION AND RESULT

System Implementation

As the diagram depicts, CPS is implemented in the trusted network to utilize the idle computing power. The coordinator publishes a Web service to provide the list of requiring computing power as the access point of CPS. As for the requester, it uses Oracle Process Manager Server to host BEPL engine and Oracle PM designer with Eclipse to provide GUI interface for designing and management. Meanwhile, a low-priority program is run at the end of the computing unit to execute the task from the requestor. The purpose to lower the priority of a program is to avoid impacting the routine work of the computing unit.

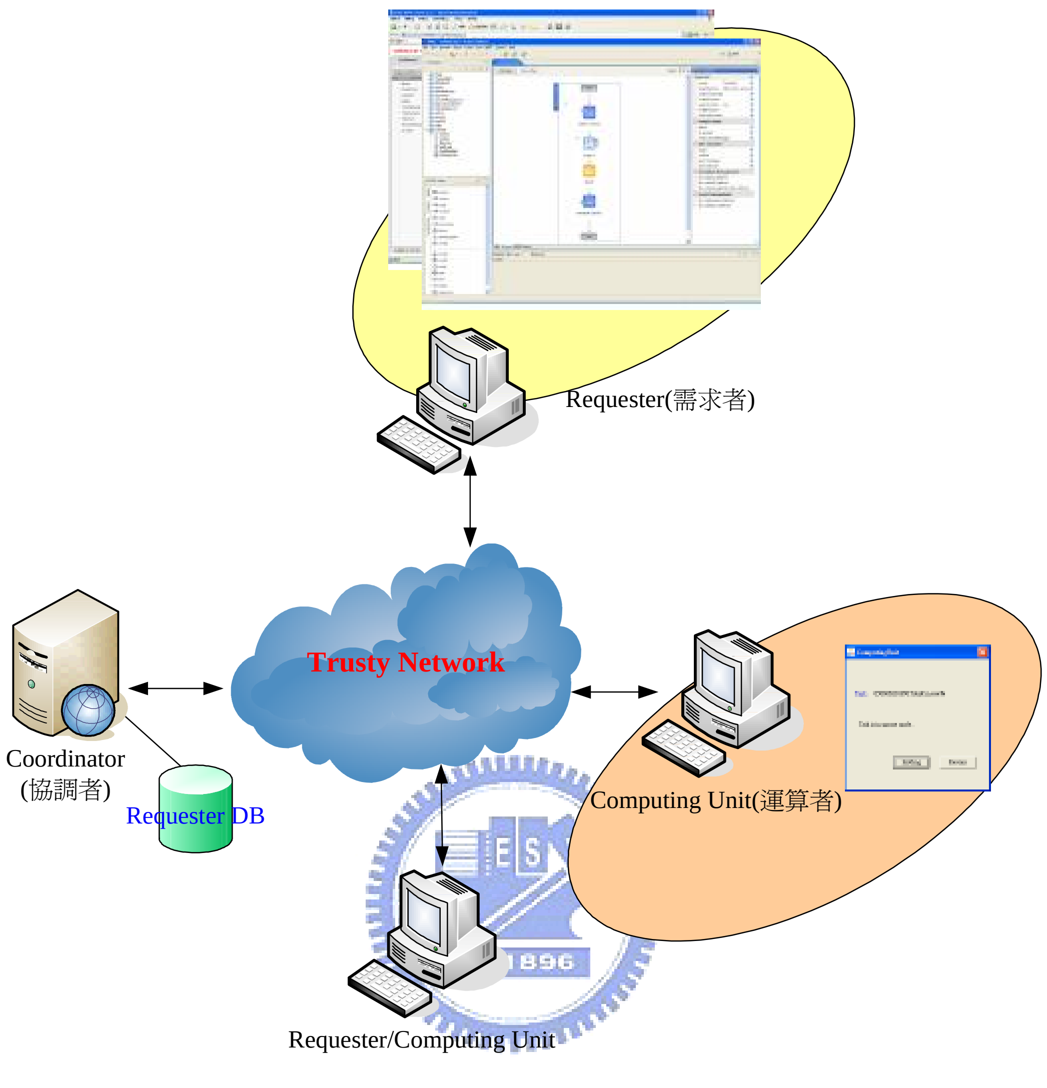

Figure 5.1 The deployment diagram of CPS

Result

To verify the architecture, a lab is arranged to test CPS on executing the program from [5]. This program will extract the watermark by using 76,177 filers which will be grouped into several subtasks with a group having 100 filters. By using the similar computers at the end of computing unit, the total computing time versus the number of computers involved to finish the lab is graphed in the figure 5.1.

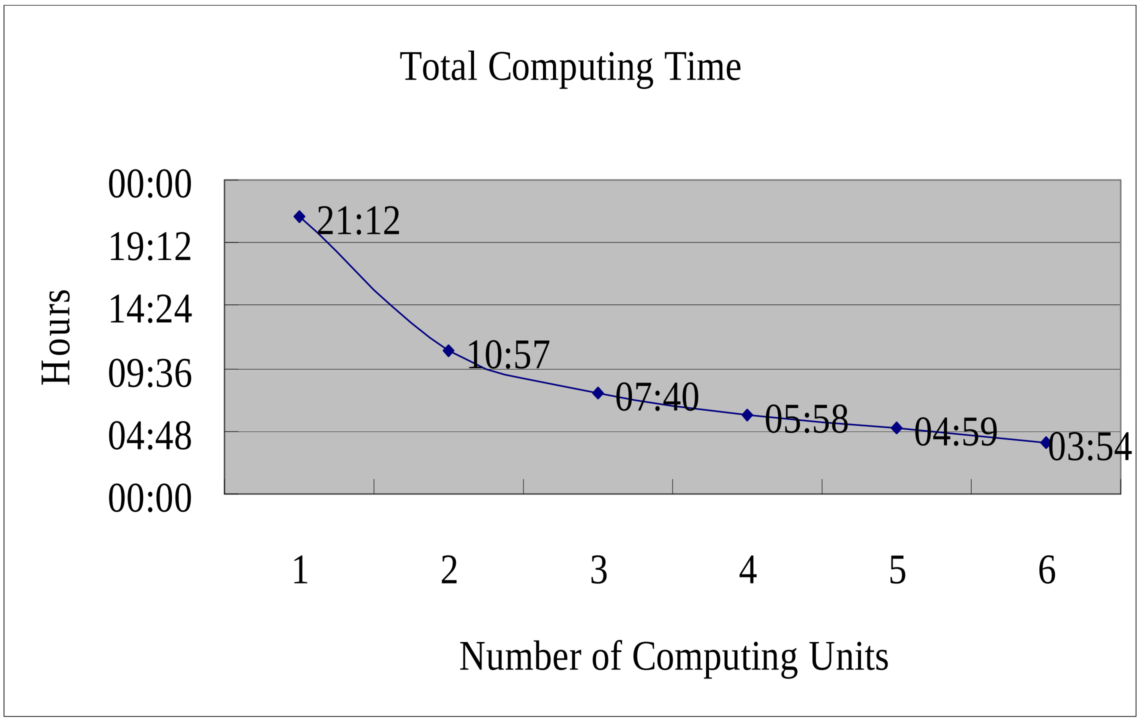

Figure 5.2 The computing time to finish the lab

According to [5], one computer will spend about 20 hours to finish the specified lab. If the number of filters is increased to 1,628,250 and each group consists of 500 filters, one computer will need 18 days to finish the lab. However, CPS will shorten the computing time to 2 days 14 hours 13 seconds to do the same lab. The deducting ratio is almost 1/9. As a result, CPS indeed helps executing a computation-intensive task.

CONCLUSIONS

This paper presents the architecture of CPS which employs the protocols of Web services to address the flexibility issues of current P2P computing, uses BPEL to control the workflow of the process and provides a user-friendly environment to design the process. In addition, the architecture is assumed to perform in the trusted network to avoid the security issue.

Such a lightweight architecture is especially applicable to the batch programs which need intensive computing power and appropriate to the enterprises which can efficiently utilize their computing power after the office hours. Besides, it also provides a graphical designing and management environment to the enterprises.

Future Works

- Workflow Management

Currently, the mechanism of Roll-Back will redo the unfinished task when an exception occurs. The purpose to do so is to guarantee the contract is performed exactly. However, it will consume more resources to finish the same task. Therefore, how to efficiently continue the interrupted task will be a future work to address.

- Process Optimization

Although CPS shortens the computing time, is it an optimized solution? In the following figure 6.1, there are 4 computing units A, B, C and D assigned to execute a process. Each colored block means the computing time needed to finish one subtask. If A and D ask the requester to assign a new subtasks at the same time when only 2 subtasks are left to finish, the requester will assign one subtask to A and D when the round-robin mechanism is used. Then, the total computing time will be depicted in the figure 6.1(a). However, if the dynamic mechanism such as assigning the subtask according to the previous computing time, both subtasks should be assigned to D because it finishes the subtask faster. Then, the more optimized solution is concluded in the figure 6.1(b).

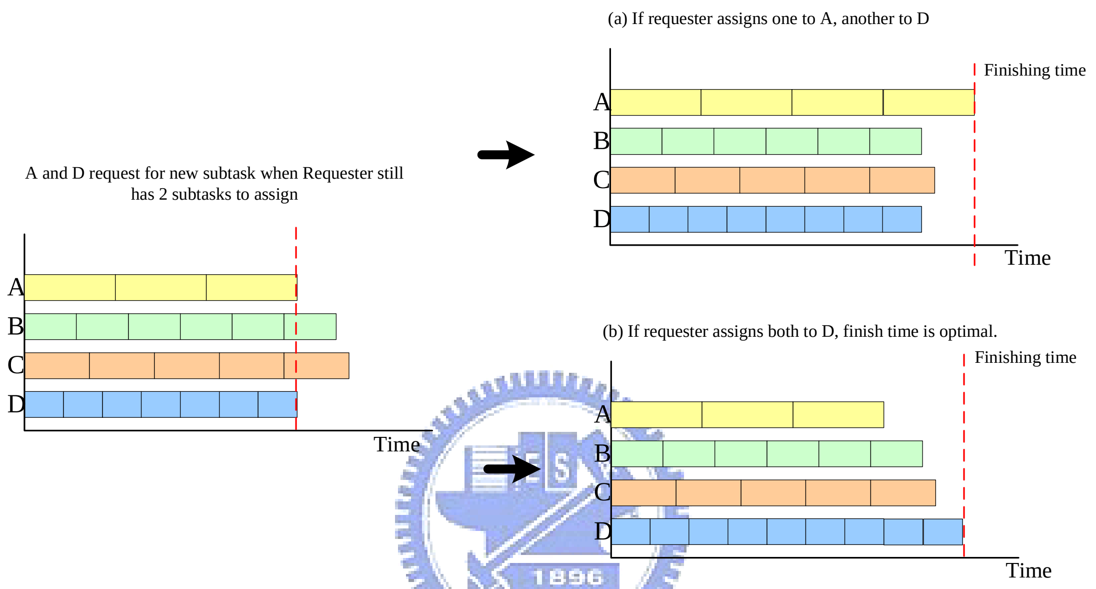

Figure 6.1 The optimized task assignments of a process

In addition, BPEL provide the capability to run the subtasks in parallel. Hence, the work to find the more optimized solution which finishes the parallel process is a topic deserved to research.

l BPEL Virtual Machine

In essence, BPEL could be though as programming language of the process. Therefore, BPEL virtual machine could be designed to execute the process. Doing so, the BPEL virtual machine will be downloaded to the computing unit and the subtask will be the fragment of BPEL document that could be executed in the virtual machine. Then, the issue of cross platform could be addressed.

REFERENCES

[1] Dejan S. Milojicic "Peer-to-Peer Computing" HP Laboratories Palo Alto March 2002

[2] Alfred W. Loo "The Future of Peer-to-peer Computing" Communications of The ACM September 2003 Vol.46,No.9 P.57-.61

[3] Dan Gisolfsi, Web Services Architect , Solutions Architect, IBM jStart Emerging Technologies 01 Apr 2001 http://www-106.ibm.com/developerworks/webservices/library/ws-arc1/

[4] Tony Andrew, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein et al. “Business Process Execution Language for Web Services” http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[5] Tseng, L. H. (2003). The wavelet packet transform based watermarking for the digital image ownership verification. Unpublished master’s thesis, National Chiao-Tung University, Taiwan

[6] Biplav Srivastava, Jana Koehler “Web Service Composition - Current Solutions and Open Problems”

[7] Holt Adams “Asynchronous operations and Web services: A primer on asynchronous transactions” http://www-106.ibm.com/developerworks/library/ws-asynch1.html , 2002

[8] Oracle Lab Segments “Oracle BPEL Process Manager Training” http://otn.oracle.com/bpel , August 2004

[9] Nikola Milanovic and Miroslaw Malek“Current Solutions for Web Service Composition” IEEE INTERNET COMPUTING NOVEMBER DECEMBER 2004, P.51-59

[10] Matt Powell “Asynchronous Web Service Calls over HTTP with the .NET Framework” http://msdn.microsoft.com/library/en-us/dnservice/html/service09032002.asp?frame=true September 9, 2002

[11] Matt Powell “Server-side Asynchronous Web Methods” http://msdn.microsoft.com/library/en-us/dnservice/html/service10012002.asp?frame=true October 2, 2002