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

Abstract: As demands of data processing and computing power are increasing, existing information system architectures become insufficient. Some organizations try to figure out how to keep their systems work without purchasing new hardware and software. Therefore, a Web-services-based model which shares the resource over the network like a P2P network will be proposed to meet this requirement in this paper.

In addition, this paper also discusses some problems about security, motivation, flexibility, compatibility and workflow management for the traditional P2P power sharing models. Our new computing architecture - Computing Power Services (CPS) - will aim to address these problems. For the shortcomings about flexibility, compatibility and workflow management, CPS utilizes Web Services and Business Process Execution Language (BPEL) to overcome them. Because CPS is assumed to run in a reliable network where peers trust each other, the concerns about security and motivation will be negated.

In essence, CPS is a lightweight Web-Services-based P2P power sharing environment and suitable for executing computing works in batch in a reliable network.

I. Introduction

In the era of host computing, almost everything is done by mainframe computers. Processing in the mainframe often becomes a bottleneck in the information systems. Therefore, it forces enterprises to spend more money in upgrading mainframe system in order to keep up with increasing demands of computing power. Then, the client-server architecture is proposed to address such issue. The client-server architectures shift processing burden to the client computers. By workload sharing, client-server systems can maintain efficiency of the information systems while reducing the budget for computing resources.

Although client-server architectures have gained wide acceptance, increasing maintenance cost after system deployment push many companies to search for another ways to improve their processing power again without more investment in new hardware and software in a competitive market. In the meantime, some people try to think about how to use existing resource in the company such as 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 and benefit by lower costs and faster processing times.

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. Thus, computer resources such as file system, network bandwidth and computing power are shared in some current P2P models. But there are some problems in existing P2P models, like lack of ability to customize computing tasks, workflow management etc.

To address these problems, this paper presents CPS - a lightweight Web-Services-based P2P power sharing environment which is suitable for executing computing works in batch in a reliable network. The architecture relies on BPEL to provide workflow management and on Oracle's BPEL Process Designer to provide a visual development environment. CPS also benefits from the characteristics of Web services, which are an open standard and loosely coupled.

II. Peer-To-Peer Model

The term "peer-to-peer" (P2P) refers to a class of systems and applications that collect distributed resources to perform a critical function in a decentralized manner. The resources could be computing power, data (storage and content), network bandwidth, and presence (computers, human, and other resources) [6]. Generally, there are three features in the P2P system: [1]

- A Computer can act as either client or server in the system.
- It allows users to make use of the collective power in the network
- A user benefits from lower costs and faster processing times in the system.

By employing P2P model, an organization can accumulate the existing resources to more powerful resources to satisfy increasing demands of processing power with economic expense. No matter how old the computer is, how narrow the bandwidth is and how less the storage is, many a little may make a mickle in the P2P network and it is what P2P model want to do.

There are two main categories of P2P system currently. One is file sharing (Napster) model and another is distributed computing (CPU power sharing) model [1].

File sharing model

Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, December 5-9, 2005, pp. 156 - 166.

According to [6], 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 to provide the features of file exchange, highly available safe storage, anonymity, and manageability.

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 [6]. Other famous model like E-Donkey, eMule and Bittorrent are also the examples of P2P file sharing systems.

Distributed computing

Another model of P2P system is distributed computing. This model tries to combine computing power to satisfy processing demands. It can shorten a long processing time without upgrading processing equipments. 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 [3].

Distributed computing is often implemented in large-scale scientific researches. A famous one is SETI@home [5]. Up to October, 2005, this project has a consolidated power of about 40 TeraFLOPs/s (Thousands of Billions of floating point operation per second), collected from more than five million registered user machines [12].

In general, works which will be solved in a distributed computing system need to be split into small independent parts. Then, each part will be done by the specific software which is downloaded from central server and is run on participant computer. The results will be collected by a central server. Because the results are collected, the tasks are assigned by a central server and no direct communication occurs between participant computers (peers), someone argues that this architecture is not a purely P2P architecture [6].

III. Problems Analysis

Problems about distributed computing

Because distributed computing can integrate computing power over network to meet high processing demands such as large-scale scientific computing and process efficiently, it is suitable to process complex tasks. Nevertheless there are some problems which make small-scale organizations hard to pursue this architecture. These issues are discussed as follows.

- Security

Security in the distributed computing model is based on trust. Participants must completely trust the research organization before they download the programs because to allowing unknown programs running on your own computer is greatly exposed 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 misuse, but if the patrons of P2P project are famous like Intel and the University of Oxford sponsoring the Cancer Research Project in UD, reliance on safety of their computers will be enhanced.

- Motivation

Participants who take part in distributed computing only want to make some contribution to the world and do not ask any pay back. 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? Someone argue that those equipments are exclusive assets for companies and it is not necessary to do something that is not beneficial to them. Similarly, general participants do not hope to join the projects with commercial purposes. To gain the participant's confidence and attract them to participate, some famous P2P systems like UD - a cancer research project - announce their research results do not belong to any commercial originations.

- Flexibility

In order to contribute, participants must download the specific program which is developed for that project and install it on their computers to donate CPU power. Once that program needs to be updated to do new research, the tightly-coupled relationship between participating program and central server would make it hard to update all the programs around the world efficiently. In addition, such kind of project can not let participants design their own tasks and execute them in the system. Although grid computing provide such service, a command-line but not a visual interface to use the service will make participants feel less friendly.

- Compatibility

Compatibility of participating programs across different platforms is another problem. Some participating programs, such as ones in UD project are only run on NT-compatible platforms. However, there are a large number of workstations that use Unix or other as operating systems. It will be a pity that those workstations can’t join this project due to compatibility. Although some distributed computing systems like SETI@home solve this problem by developing different versions of the program for different platforms, it will increase the maintenance cost as many versions of the program must be developed and updated.

- Workflow Management

What most of Grid or P2P distributed computing middleware focuses on is performance, workload balance or

stability but hardly workflow management. With workflow management, a complicated job which is composted of many small tasks can be executed in parallel or sequentially. 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 software development solution based on Services Oriented Architecture (SOA) in Figure 1. A Web-services-based system can inherit the features of Web Services which are loosely coupled and open standard. If a computing system is implemented by Web Services, it will not only improve flexibility of software updates in the system because of loosely-coupled relationship between service consumer and provider but also be easily to communicate because of open standard.

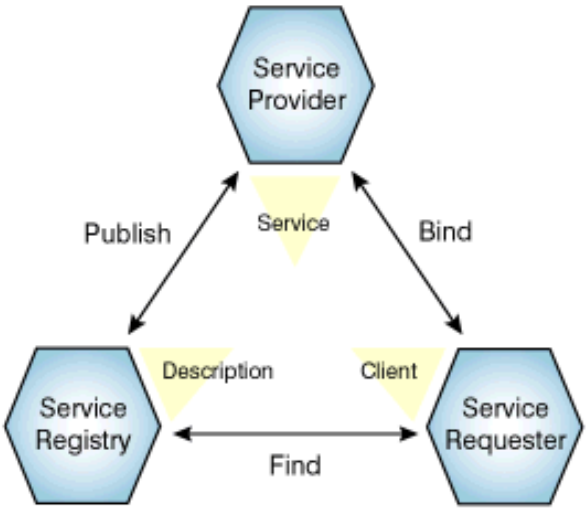

Figure 1 Architecture of Web Services [4]

Web Services is a message-based architecture and the interaction between services can be synchronous or asynchronous. In essence synchronous Web Services is not suitable for distributed computing system because it is hard to estimate the processing time in the system and it may cause timeout exception. Therefore, asynchronous Web Services [7,8,9] should be applied in the system to avoid over-time exception because it is usual to wait for response until tasks finished in distributed computing environment. The problem of flexibility will be addressed by using asynchronous Web Services to implement distributed computing system.

Business Process Execution Language (BPEL)

BPEL [9,13] is a de facto standard of Web Services composition and integrated by IBM and Microsoft from WSFL and XLANG. It has the characteristics such visual development, workflow management, exception and transaction handling and compatibility with Web Services [2].

Based on the characteristics of Web Services, BPEL is suitable to solve the issues of a visual development environment and workflow management in distributed computing system. In addition, a complicated job can be tackled by Web services composition which BPEL aims to address and can be easily executed by BPEL engine.

Implementation in a trusty network

As discussed in last section, security in distributed computing system is based on trust. So, it will be easy for famous and large originations to sponsor P2P distributed computing project but not for small-scale and medium-scale companies. However, if we think from the perspective of an organization, 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.

In this paper Trusty network is defined as a network where peers 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. Our lightweight distributed computing system is assumed to implement in a trusty network. Thus, the concerns about security and motivation can be negated.

IV. The Architecture of Computing Power Services

Based on the possible solutions in last section, this paper proposes the CPS architecture, which is a Web-services-based P2P architecture as shown in Figure 2. It provides users a platform to design the business processes and control workflow of the processes by using the visual characteristics of BPEL. The architecture is assumed to be implemented in the trusty 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 the computing requester as service provider.

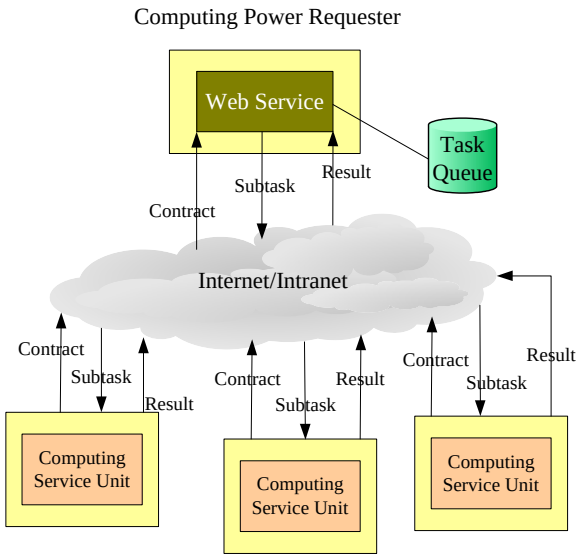

Figure 2 Architecture Diagram of CPS

The Roles of CPS Architecture

Because CPS bases on the architecture of Web services, it will inherit the characteristics of SOA 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.

- 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.

- The role of Computing Power Requester

The requesters design their processes by a BPEL visual development 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.

- 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 Figure 3 below.

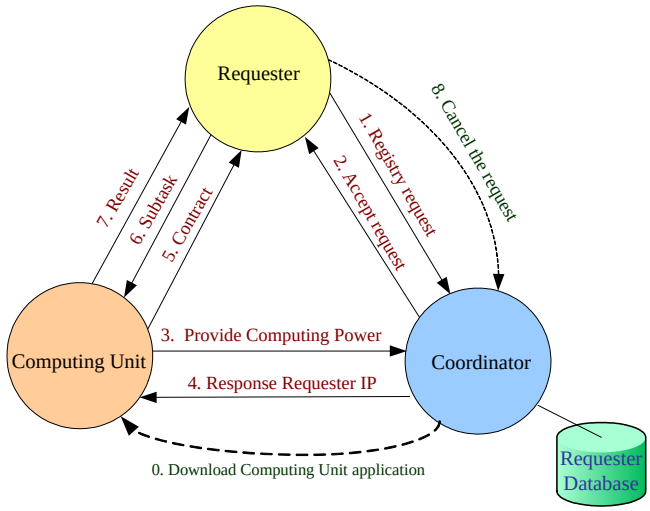

Figure 3 Operation Diagram of CPS

The System Architecture of CPS

The Figure 4 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.

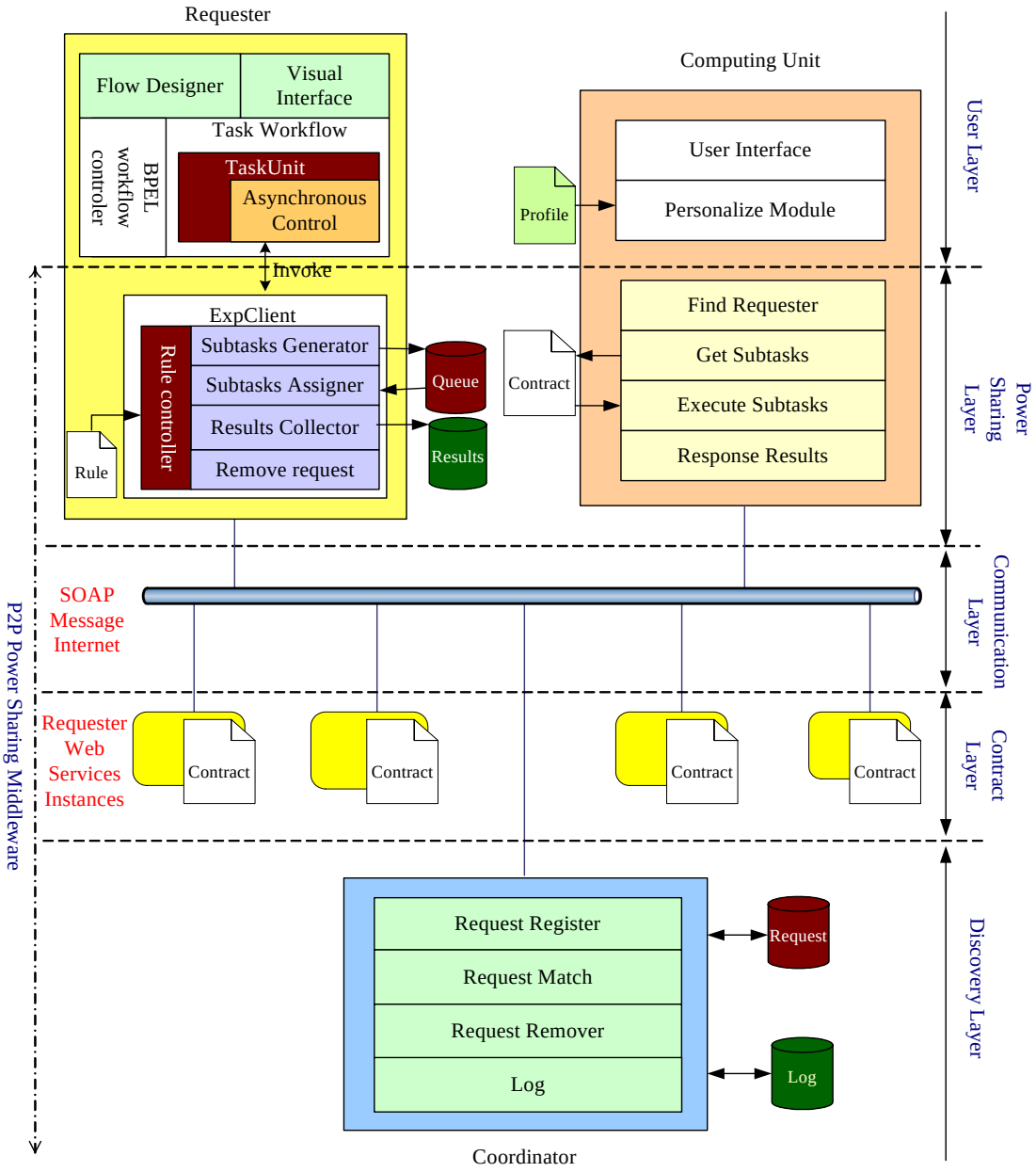

Figure 4 The Layer Diagram of CPS Architecture

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 Figure 5 shows.

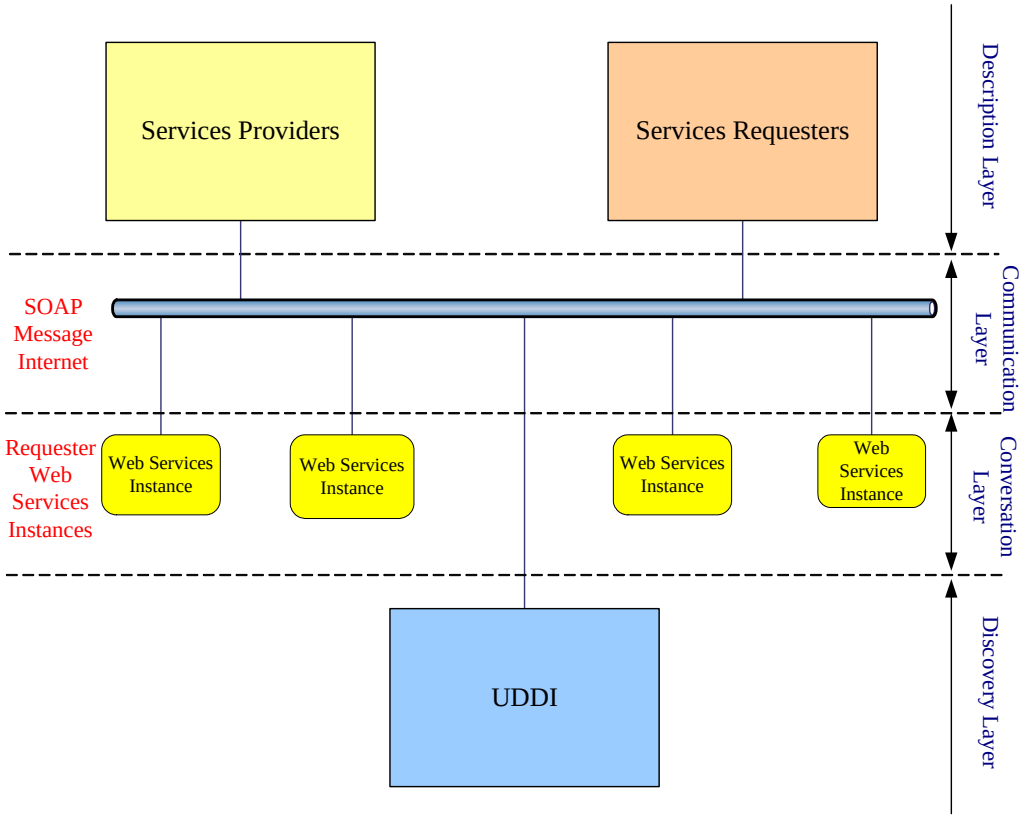

Figure 5 The Diagram of P2P Middleware in the CPS Architecture

- 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 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 Service Discovery Layer

The coordinator operates in this layer as a broker agent for the requester and computing unit. The coordinator will not involve the computing.

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 management. 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 Figure 6 describes the interaction between TaskUnit and P2P CPS middleware.

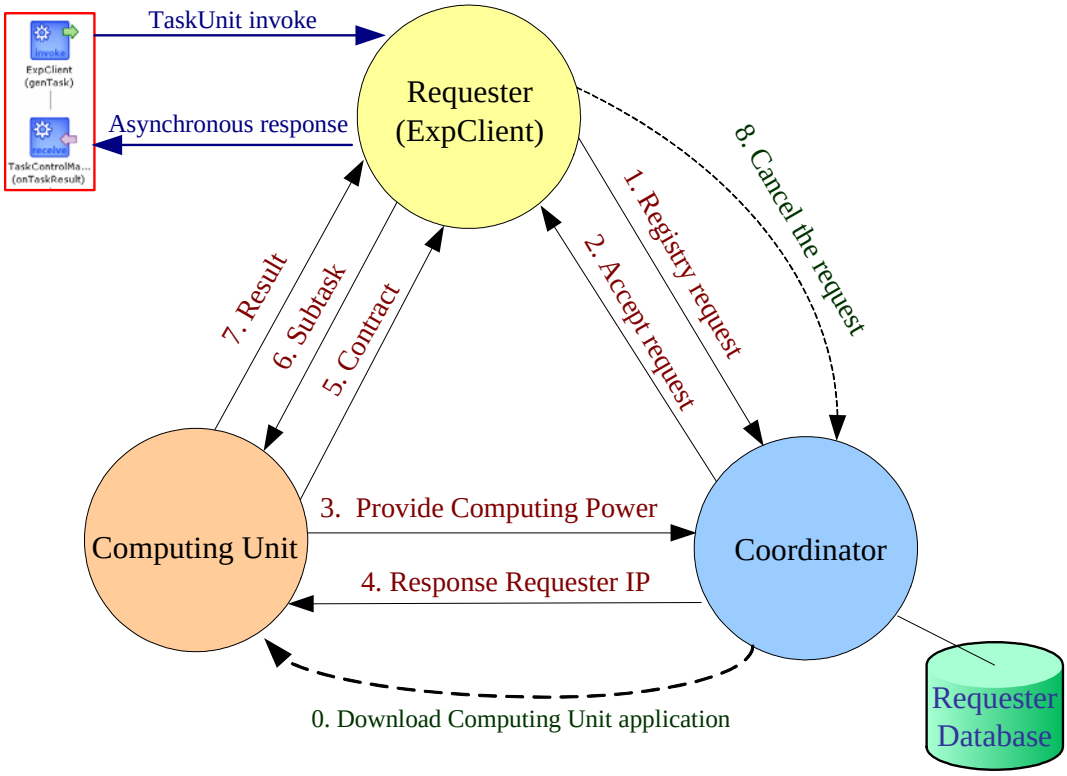

Figure 6 The Interaction Between TaskUnit and P2P Middleware

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 Figure 7. 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.

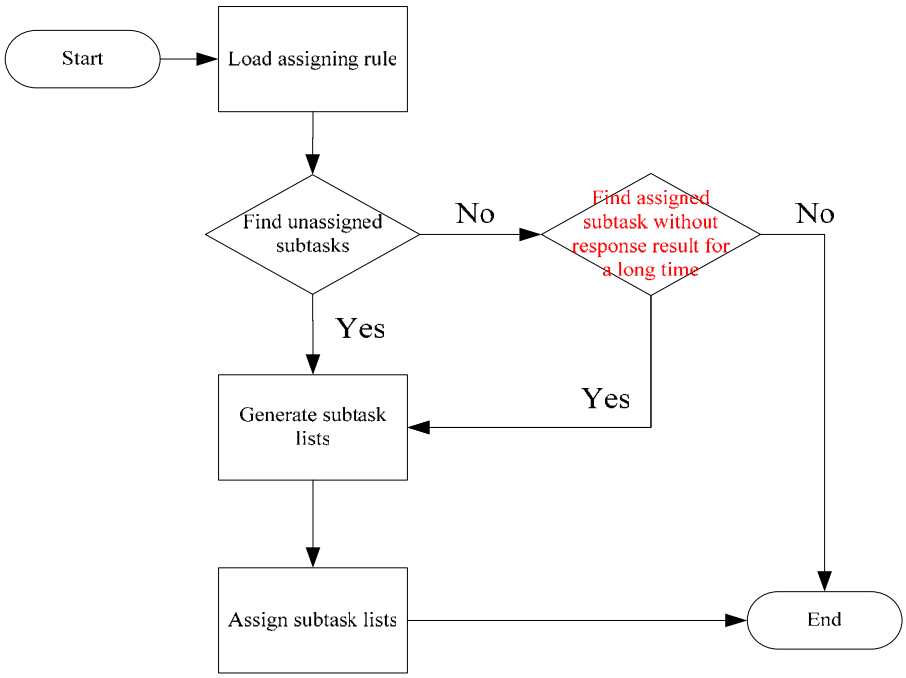

Figure 7 The Flow Chart of Assigning Subtasks at the End of Requester

The Roll-Back Mechanism at the End of Computing Unit

As Figure 8 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.

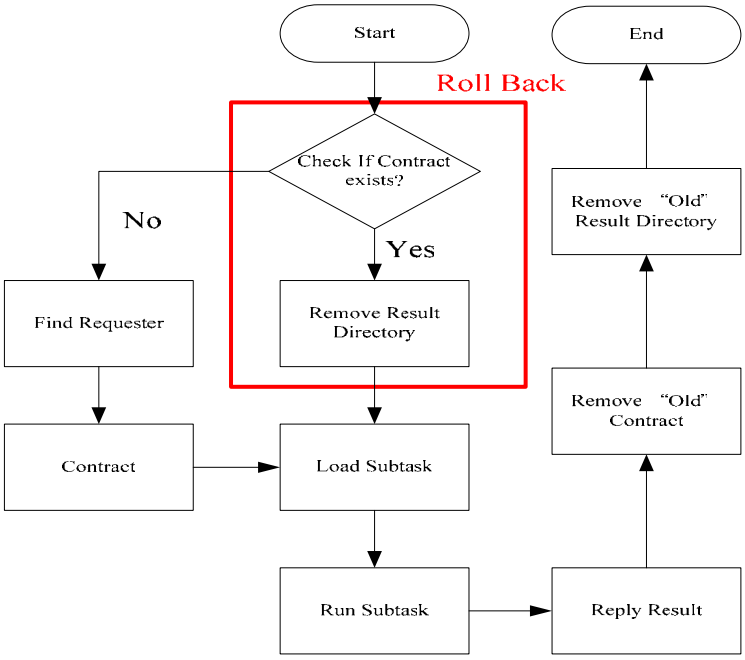

Figure 8 The Flow Chart of Executing Subtasks and Roll-back at the End of Computing Unit

V. Implementation and Result

System Implementation

As Figure 9 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 [11] 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.

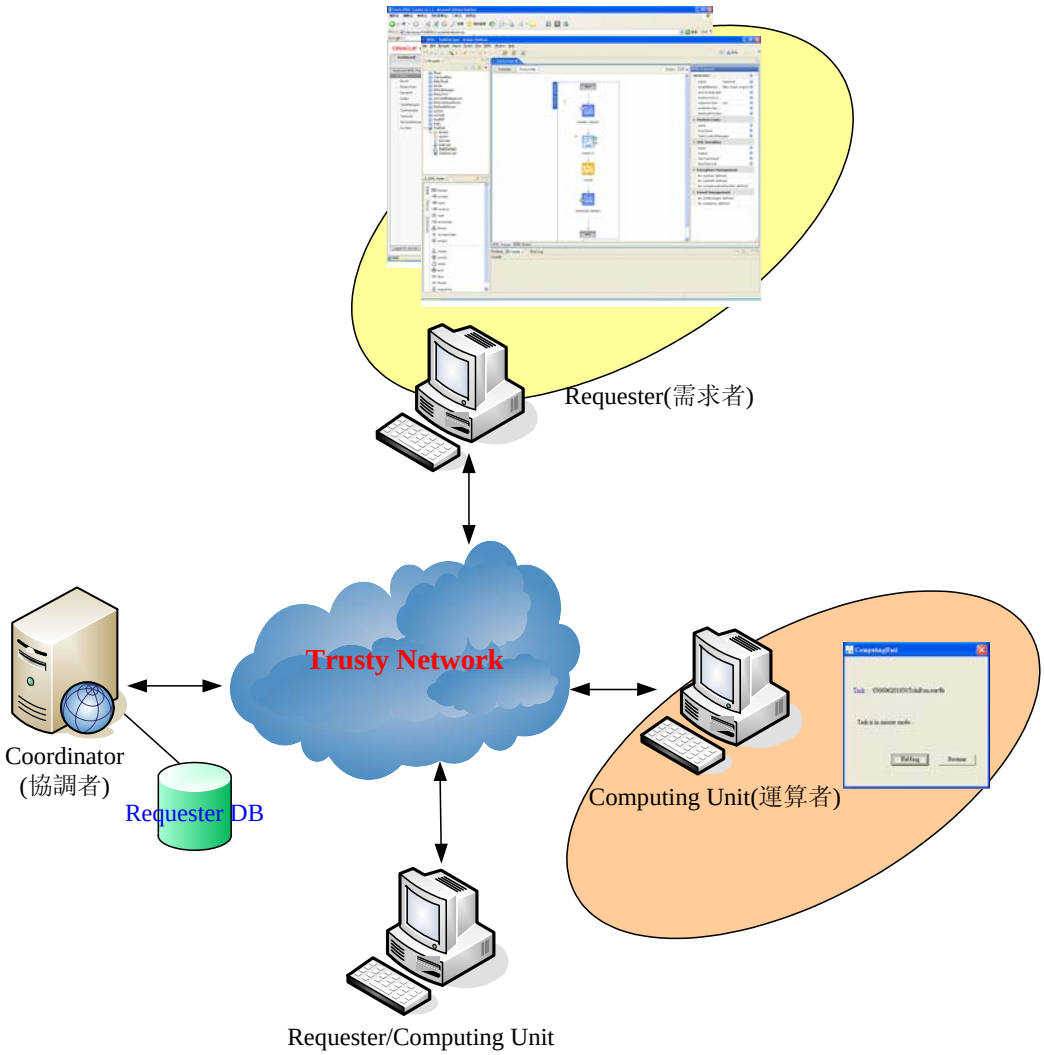

Figure 9 The Deployment Diagram of CPS

Result

To verify the architecture, a lab is arranged to test CPS on executing the program from [14]. 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 Figure 10.

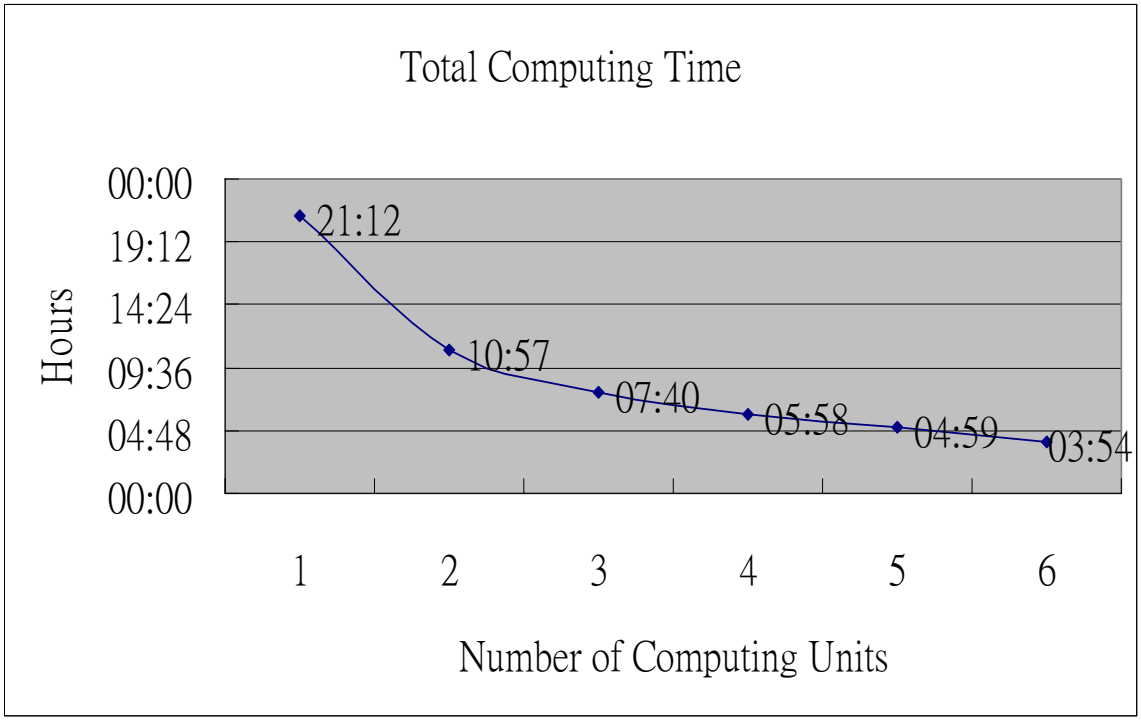

Figure 10 The Computing Time to Finish the Lab

According to [14], 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.

VI. 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 Figure 11, 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 Figure 11 (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 Figure 11(b).

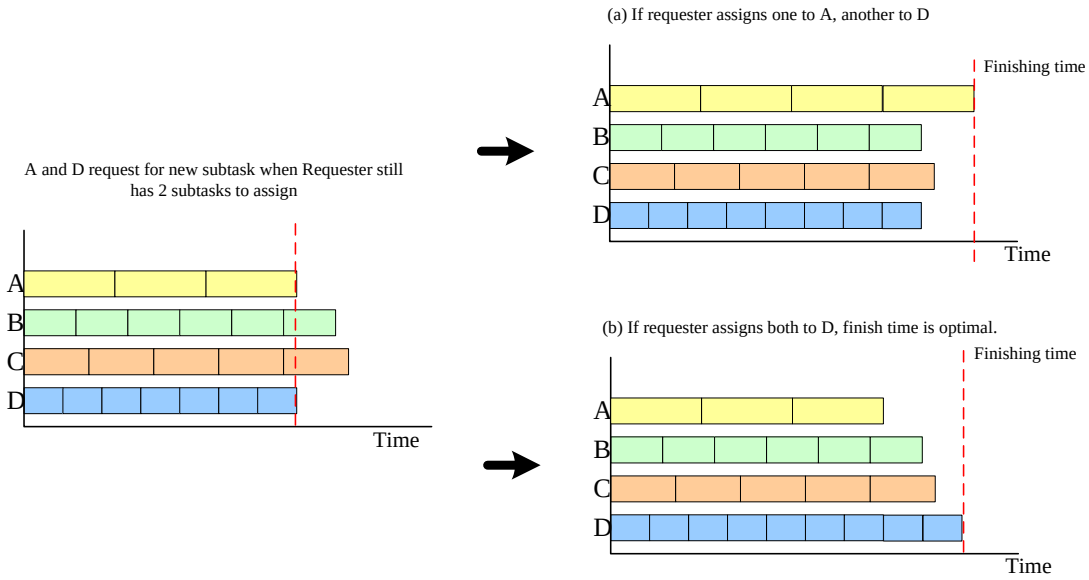

Figure 11 The Optimized Task Assignments of a Process

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.

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

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

[3] Cavallar, S. et al, “Factorization of a 512-bit RSA Modulus.”, Proceedings of EuroCrypt 2000, Bruges (Brugge), Belgium.

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

[5] David P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. ” SETI@home: An Experiment in PublicResource Computing.” , Communications of the ACM, 45, 2002.

[6] Dejan S. Milojicic “Peer-to-Peer Computing” HP Laboratories Palo Alto March 2002

[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] 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

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

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

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

[12] SETI@home, “Current Total Statistics”, http://seticlassic.ssl.berkeley.edu/totals.html

[13] 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/

[14] 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