Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 36 (2009) 3101–3118
Expert Systems with Applications
www.elsevier.com/locate/eswa

Distributed computing power service coordination based on peer-to-peer grids architecture

Min-Jen Tsai *, Yin-Kai Hung

Institute of Information Management, National Chiao Tung University, 1001 Ta-Hsueh Road, Hsin-Chu 300, Taiwan, ROC

Abstract

As computational speed, storage capacity, and communication technologies steadily progress to the advanced levels, many business employments have utilized these improvements to aggregate distributed resources in various enterprise collaborating functionalities. Recently, grid computing has been identified as a critical technology by industry for enterprise computing and business-to-business computing. The concept of peer-to-peer (P2P) network is also evolving to an expanded usage in distributed networks for sharing the resources like the content files or real-time data for enterprise applications. Therefore, it is natural to include the grids and P2P to support environments that have features of both limiting case. In this paper, a new computing design – NaradaBrokering based Computing Power Services (NB-CPS) – has been applied to utilize the P2P grid to integrate the computational grids, distributed objects, P2P networks under the hybrid environment. In this study, the approach has been applied to analyze the robustness of digital watermark by filter bank selection and the performance can be improved in the aspect of speedup, stability and processing time. NB-CPS is not only suitable for executing computing works which are able to run in batches, but also be able to solve current issues in Web Services based Computing Power Service (WS-CPS) such as system resilience, fault tolerance, efficiency of job scheduling and the instability in congested network environment.

Keywords: Distributed computing; Grid; NaradaBrokering; Peer-to-peer; Web services

1. Introduction

As computational speed, storage capacity, and communication technologies steadily advance, increasingly large, complex, and resource-intensive applications are being developed both in research institutions and commercial industry. The power of network, storage, and computing resources has important implication for successful operation to the academy research and enterprise business (Casanova, 2002. Recently, grid computing has been identified as a critical technology by industry for enterprise computing and business-to-business computing (Foster, Kesselman, & Tuecke, 2001). The improvements in wide-area networking make it possible to aggregate distributed resources in various collaborating institutions and to form what have come to be known as computational grids (Foster, 2002). To date, most grid applications have been in the area of scientific computing as scientists world-wide are resorting to numerical simulations and data analysis techniques to investigate increasingly large and complex problems.

Due to the cost considerations, small and medium-sized enterprises (SME) generally like to apply the P2P computing which is defined as the sharing of computer resource and services by exchanging the information and content files, processing cycles, cache storage, and disk storage for data files. Through sharing the computation load and the computing power by the existing desktop and network, it allows the enterprise to reduce the cost and efficiently leverage their total computation powers and benefit the entire business. Comparing to P2P and grid, both are for sharing of varied resources, ranging from programs, files, and computation capabilities to computers, sensors, and

* Corresponding author. Tel.: +886 3 571 2121x57406; fax: +886 3 572 3792.
E-mail address: mjtsai@cc.nctu.edu.tw (M.-J. Tsai).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.01.050

networks in different property and characteristic with benefits, it is natural to include the grids and P2P to support hybrid environments that have features of both limiting case.

Based on previous studies, we have proposed the Web Services based Computing Power Service System (WS-CPS) Tsai et al., 2006 to fulfill SME distributed computing needs. However, there are still some issues in WS-CPS system needed to be further investigated such as: (1) The requirement of flexibility and fault tolerance mechanism in system architecture. (2) In high-throughput environment, there is a need of network bandwidth control for quality service. (3) The priority mechanism for multi-task demand situation. (4) The improvement of task allocation and re-allocation strategy.

To address these problems, we purpose a novel design named NaradaBrokering based Computing Power Service System (NB-CPS) which is using NaradaBrokering as the distributed computing middleware (NaradaBrokering Project, 2006). NB-CPS provides a lightweight distributed computing environment to gather computing powers in SME and is suitable for executing computing works in batch format. In addition, it also provides the visual interface to design and monitor the computing tasks.

The proposed design has been applied to analyze the robustness of digital watermark by filter bank selection (Tsai, 2004) and there is not only a significant upgrade in operational efficiency, but also effective task allocation and exception handling mechanisms in multi-tasks with different priorities situation.

This paper will be organized as following: Section 2 will briefly explain the distributed architectures of WS-CPS and the issues need to be solved. In Section 3, solutions for WS-CPS' problems will be recommended and the proposed NB-CPS will be explained in Section 4. The implementation and performance comparison of NB-CPS and WS-CPS is analyzed in Section 5 and conclusion will be given in Section 6.

2. Issues with WS-CPS

This section will briefly introduce WS-CPS and some issues should be further studied for improvement.

2.1. The roles of WS-CPS and the operational procedures

WS-CPS is bases on the architecture of web services, it inherits the characteristics of Service Oriented Architecture (SOA) which consists of three participants that are service requester, service provider and service broker. The description of three roles is as following:

1. *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 major 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. Afterwards, 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.
2. *The role of computing power requester*: The requesters should design their experiment processes and publish their requirements with the coordinator. In addition, it will assign the subtasks to the computing unit with the work flow control capability.
3. *The role of computing unit*: This role is responsible for executing computation. It will inquire the coordinator for the job while it is idle. After getting back the requester's URL of Web services, it negotiates with the requester to download the subtasks along with the required data. When the subtask is finished, it will respond the result to the requestor and the whole procedure will continue until all subtasks are completed.

The operational procedures of WS-CPS are shown in Fig. 1 with the following steps:

Step 1: Requestor uses BPEL to design the entire process.
Step 2: Requestor registers its task to coordinator.
Step 3: Coordinator confirms the request.
Step 4: Computing unit registers its computing power to coordinator.
Step 5: Coordinator sends requestors' addresses to computing units.
Step 6: Computing Unit makes the contract to the requestor.
Step 7: Requestor transfers the tasks to computing units.
Step 8: Computing unit sends back the result after execution.
Step 9: Requestor combines all results while the job is completed.

2.2. The issues of WS-CPS

There are some issues which are needed to be improved for WS-CPS as following:

1. *The requirement of flexibility and fault tolerance mechanism in system architecture*. From the systemic point of view, every node in the peer-to-peer distributed computing infrastructure should have equivalent important position. In other word, every role should be able to be played on any node in the network. However, nodes in WS-CPS communicating with each other through the web service and managing the process by BPEL make

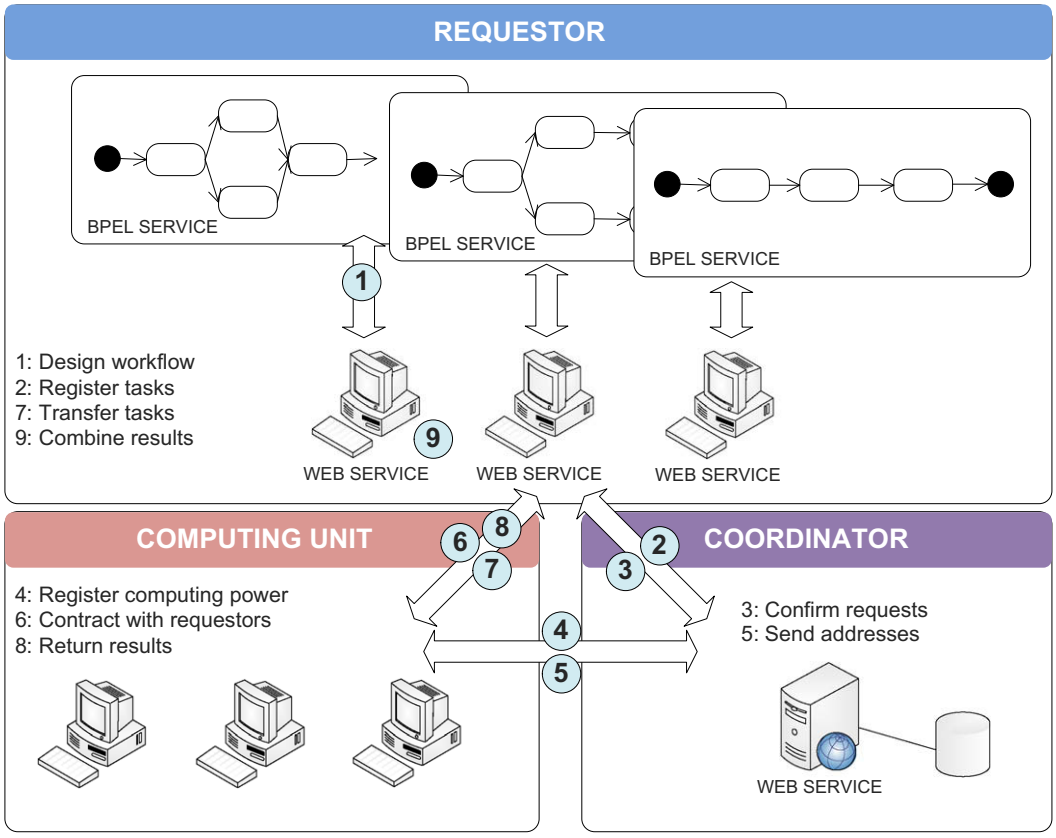

Fig. 1. The flow diagram of WS-CPS system.

roles fixed in the system. Therefore, the whole system would be stopped for operation if the role of coordinator is not functioning. Therefore, the backup or mirror sites should be able to take over the down machines and continue the operation when the above mentioned situation happens for the distributed computing system. Certain flexibility and fault tolerance mechanism is required.

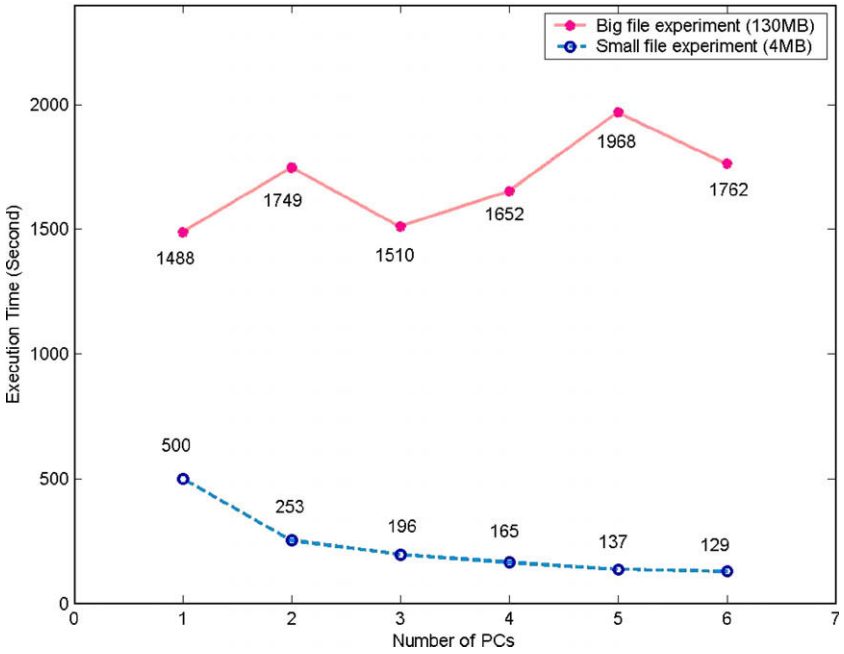

Fig. 2. The execution time with different size of transmitted files in WS-CPS.

2. *In high-throughput environment, there is a need of network bandwidth control for quality service*. Fig. 2 is the plot which shows the execution time for small and large file transmission in WS-CPS. While the transmission file is small, the transmission time is decreasing when the computing units is increasing. However, the transmission time and the computing units are increasing when the transmission file is as large as 130 MB. Due to the constraints of the network bandwidth, there is in practical a limit for the transmitted file size. On the other hand, the number of computers increasing causes the whole network congested for file transmission. Therefore, a proper network bandwidth control for quality service is required for stability consideration.
3. *The priority mechanism for multi-task demand situation*. In WS-CPS, there is no design for different priority service which results no differentiation in service requests. In fact, there usually exist urgent requests for enterprise and the request of priority service is necessary for quality service and VIP level customers.
4. *The improvement of task allocation and re-allocation strategy*. While the task is allocated for the computing units, the overall time to get the task done and receive the reply is actually composed by two independent time component: the transmission time and the actual processing time. During the data transmission period, the computing unit should keep processing other sub-tasks without wasting time for another job if the task can be parallel processed. Therefore, it will be the optimal solution by allocating all subtasks to all idle computing units (Plank et al., 2001) and keeping all units busy till the whole task is done. Under such condition, certain job queue and pre-fetch scheme should be applied to keep the computing units busy.

On the other hand, the distributed computing network is a dynamic environment. The actual processing results could be differed from the forecasting ones after the initial task allocation and certain adjustment is necessary. For example, there are eight subtasks for three computing units as allocated in Fig. 3 and apparently PC1 performs better among others. While the sub-task 7 is completed in PC1, the overall job completing time will be T2 instead T3 if the sub-task 8 is re-allocated from PC3 to PC1. A good prediction mechanism is needed for the job re-allocation strategy.

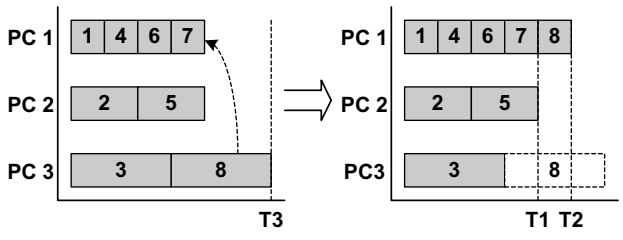

Fig. 3. The task re-allocation process in an ideal environment.

3. Proposed solutions for WS-CPS

To solve the issues mentioned in the previous section, there are proposed solutions for the new design called NB-CPS. The detailed information will be explained as following.

3.1. The application of NaradaBrokering middleware to build flexible and fault tolerant computing sharing environment

To make the system with flexibility and fault tolerance, the entire messaging infrastructure should be re-designed. In this study, we apply NaradaBrokering as the messaging infrastructure to substitute web service structure in WS-CPS and propose the NB-CPS design. NaradaBrokering is a Java based open source distributed messaging infrastructure purposed by Indiana University Community Grids Lab., Publish/Subscribe Model (Pub/Sub) Banavar et al., 1999; Segall and Arnold, 1997; Viktor et al., 2003; is the main communication structure in NaradaBrokering. It provides two closely related capabilities. First, it provides a message-oriented middleware (MoM) that facilitates communications between entities through the exchange of messages. Secondly, it provides a notification framework by efficiently routing messages from the originators to only the registered consumers of the message in question (Fox et al., 2003). As shown in Fig. 4, NaradaBrokering provides a JMS standard API for developers to create distributed applications, and furnishes an intermediary layer for message transmission.

To offer the characteristics of scaling, load balancing and failure resiliency system, NaradaBrokering is implemented on a network of cooperating brokers under the hierarchical structure, where a broker is part of a cluster that is part of a super-cluster which can be demonstrated in Fig. 5. Therefore, the distributed broker network can scale to support the increase in these aggregated entities and the addition of brokers aids the scaling which will not degrade performance by increasing communication

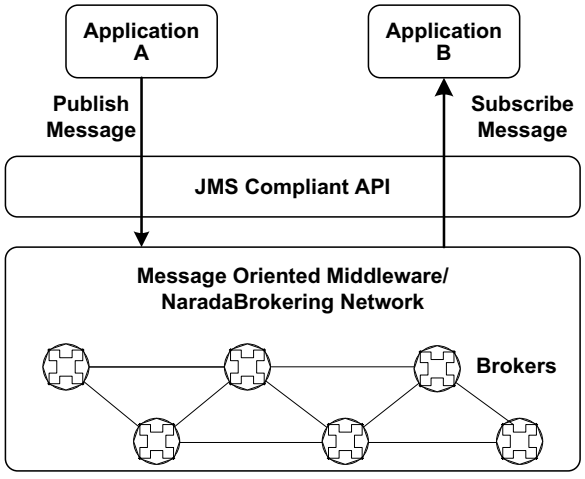

Fig. 4. System architecture of NaradaBrokering.

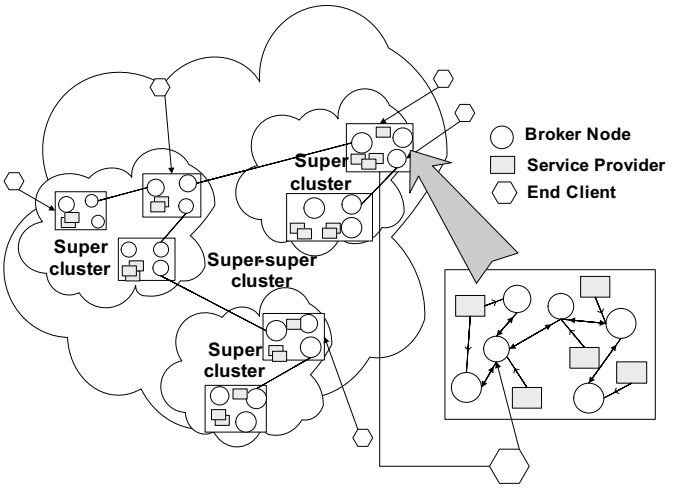

Fig. 5. NaradaBrokering network.

pathlengths or ineffective bandwidth utilizations between broker nodes within the system (Fox et al., 2003). Besides, the disseminations pertain to routing content, queries, invocations etc. to the relevant destinations in an efficient manner. The routing engine at each broker ensures the paths traversed within the broker network to reach destinations are along efficient paths that eschew failed broker nodes. The messaging infrastructure is able to communicate across firewall, DHCP and NAT boundaries. It also supports various types of protocol such as TCP/IP, UDP, RTP, RMI, HTTP and SOAP.

To use NaradaBrokering as the distributed computing network middleware, it can resolve the flexibility and failure resiliency issues. From the system point of view, the nodes can play different roles by registering different topic and easily substitute the broken nodes to continue the system functionality in NB-CPS. However, such flexibility and adjustment is not possible for web service based architecture.

3.2. The bandwidth flow control mechanism by FCFS reservation strategy

As noted in the previous section, WS-CPS is instable and will have high tendency to failure due to the network congestion. To resolve this issue, we design a flow control mechanism as shown in Fig. 6 which likes a network gate embedded within the requestor while the transmitting file is large. The gate control will allow the transmission with the permitted reservation and the reservation queue inside the requester is based on the FCFS strategy as the dispatching policy. In Fig. 6, the communication between the computing unit B and the requester is allowed since it has the permitted reservation while computing unit A is waiting for the permission since its reservation is still in the queue.

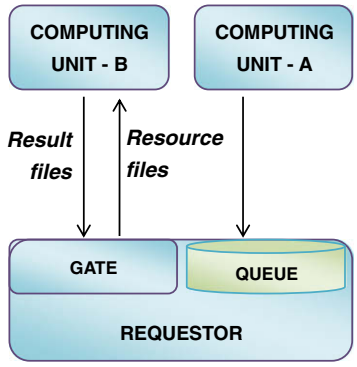

Fig. 6. The network bandwidth control strategy.

3.3. Weighted priority based task selection

NB-CPS uses weighted priority strategy (Erradi et al., 2006) to select the requested tasks for the scheduler which means the requested tasks with higher priority will have higher chance to be selected than the ones with lower priority. For example in Fig. 7, the task from requestor "A" with priority "3" will have 3 times higher probability than the task from requestor "B" with priority "1". The unsigned tasks will also have the priority waiting appreciation scheme to avoid the dispatching starvation.

3.4. Use K-means prediction process to resolve the task allocation and reallocation strategy

In a distributed computing environment, the actual execution time will be different in different computing units. Even executing the same task by the same computing unit, the execution time will be different due to the dynamic system loading during the execution period. After all tasks are allocated, it is evident that some of the computing units will finish the jobs earlier and some are not. If we can predict the remaining execution time of unfinished tasks correctly, it will be quite natural to reallocate the unfinished tasks from a slow computing unit to a fast one whose expected completion time will be shorter than the remaining execution time at the slow computing unit. However, such a predication process requires the historical data with trustful estimation within a dynamic environment. A dependable scheme of allocation and reallocation strategy is required for NB-CPS.

Rationally, the execution time will be longer if the computing unit is in a high loading level and a task during the execution will practically rely more heavily on a specific

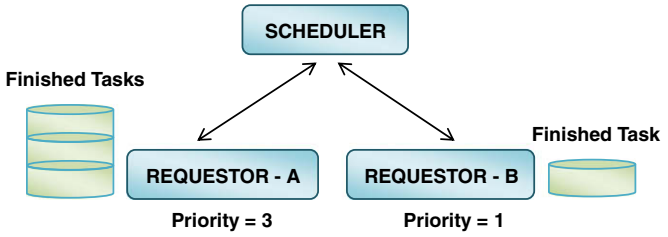

Fig. 7. The weighted priority based task selection strategy.

system performance indicator named "critical performance indicator". For example, the critical performance indicator of Linpack (Linpack, 2006) is CPU usage. Tasks with frequent file transmission such as Napster, their critical performance indicator will be the network bandwidth. Summing up the related works from Wang et al. (2007), the author analyzed the historical executing data and used value function to assign different weights to different system performance indicators to further find out the loading level of a computing unit. The authors in Tsai and Wang (2008) dynamically adjusted system performance indicators by using CC-FGDM for WS-CPS. Those methods generally analyze the historical data of the whole system in order to give the same weighting of the performance indicator for whole system. However, such an approach ignores the difference between each participating party. For example, a computer of high CUP speed with high loading level may have a longer execution time than a computer of slower CPU speed with low loading level. While analyzing the historical data of these two machines, the weightings of the performance indicator of CPU speed and loading level should be different for each individual computing unit.

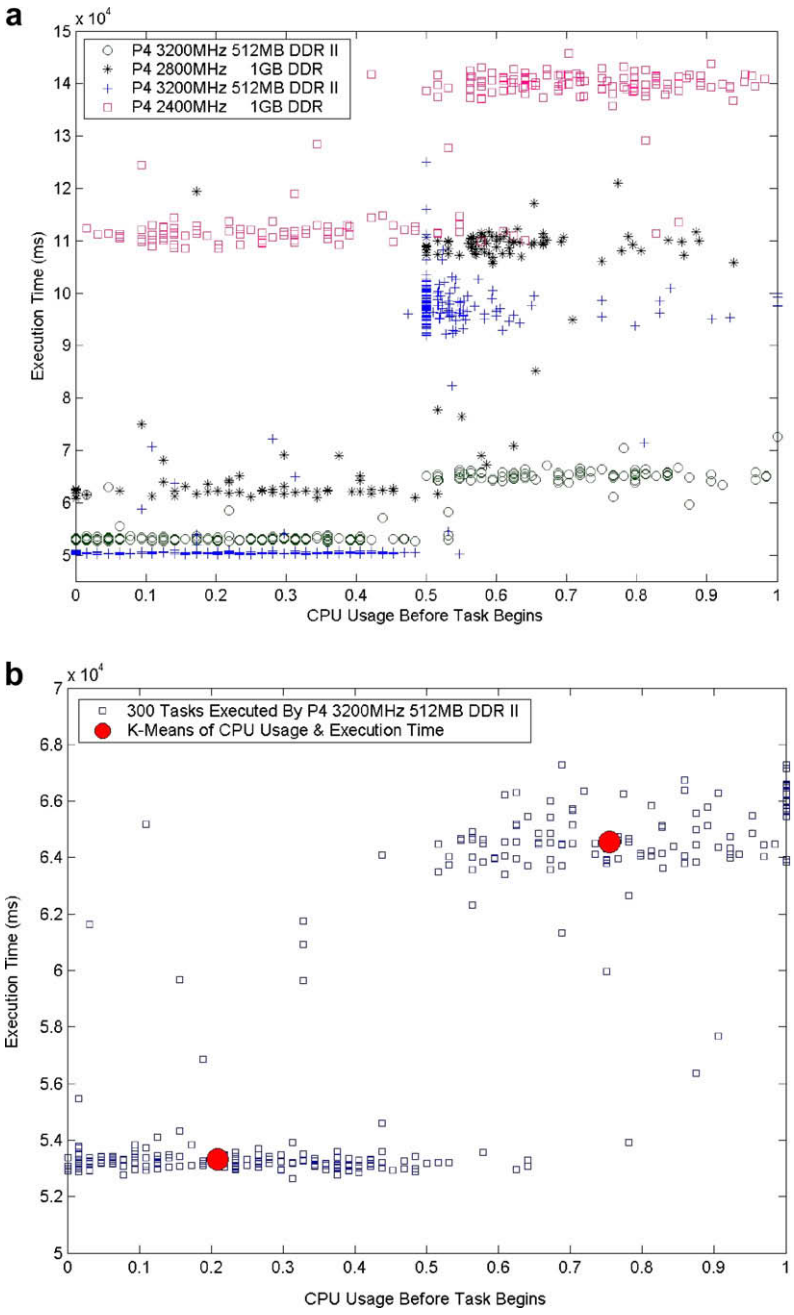

Fig. 8. (a) The execution time under different loading level for four computing units. (b) The execution time under two loading levels for one computing unit. The red dot is the mean value in each loading level. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

From Fig. 8a, it shows the historical data of the execution time and the CPU usage percentage for a distributed computing system with four computers. It is hard to differentiate those data and find a general pattern for each computer. However, a low loading and high loading pattern for only one computer is easily distinguished in Fig. 8b. To further assist the classification for the loading, support vector machine based classification (Chang & "Lin, 2001) is applied here for accurate clustering. Therefore, we adopt *K*-means clustering algorithm (MacQueen, 1967) for NB-CPS since it can be used to group the historical executing data which includes values of critical performance indicator and the execution time. The formula is in Eq. (1), where k is the number of clustering from SVM calculation, S_i is the ith group, X_j is the jth event value in S_i, u_i is the mean value for group S_i

$$V = \sum_{i=1}^{k} \sum_{X_j \in Si} |X_j - \mu_i| \tag{1}$$

The system critical performance indicator is obtained in NB-CPS by comparing the correlation values of the processing time versus the usage. The formula of correlation is in Eq. (2), where X is the indicator usage, $\bar{X}$ is the average usage, Y is the processing time and $\bar{Y}$ is the average processing time

$$\rho = \frac{\sum (X - \overline{X})(Y - \overline{Y})}{\sqrt{\sum (X - \overline{X})^2} \sqrt{\sum (Y - \overline{Y})^2}} \tag{2}$$

If the correlation value between execution time and system indicator value greater than 0.7, the system indicator will be claimed as the critical performance indicator for the task. Using such a characteristic, we can analyze each computer unit's behavior and the executing pattern from our verification shows two clear clusters which represent high and low system loading levels for CPU usage from SVM classification. Therefore, the execution time for low and high loading pattern will be used for task allocation strategy.

The necessary condition of *K*-means prediction process is to gather enough historical execution information for good estimation. Therefore, the prediction will be more accurate if the computing unit is involved in the distributed computing system for a long enough time period.

4. Proposed NB-CPS design

To sum up the proposed solutions in the previous section, we purpose the NaradaBrokering based computing power service system (NB-CPS) in this study. NB-CPS is a distributed computing platform based on NaradaBrokering (NaradaBrokering Project, 2006) based on P2P grids and provides a lightweight distributed computing environment for SMEs. The comparison between NB-CPS and WS-CPS will be explained in Section 4.1 and the detailed architecture, design and operational processes will be introduced in Sections 4.2–4.4, respectively.

4.1. The comparison of message transmission infrastructure in NB-CPS and WS-CPS

WS-CPS is a web service based computing power sharing platform, all the data transmissions are communicated through the standards of WSDL, SOAP, XML and UDDI. Such infrastructure not only improves the reuse of services but also reduces the ease-to-use and flexibility of system. The requirement of setting up web service server and UDDI increases the entry barrier and causes system roles fixed in specific nodes. Therefore, a role cannot be easily replaced when it crashes and the role regenerating processes is hard due to the environment restriction. On the other hand, the transmissions of NB-CPS between nodes are reached by Pub/Sub model and nodes in NB-CPS system play different roles through subscribing to different templates. While certain role disappears in the system, different node can use the Pub/Sub model to claim the substitution of the missing node and continue the normal function easily.

In NB-CPS system, every event is triggered by node-state change or message receiving. As shown in Fig. 9, a message structure is composed by three parts: "header" which records routing information, "template" which describes the subject and the type of the message, and "payload" which contains the content of the message. Every node must connect to a broker by a unique identification number. Fig. 9 shows a message that a computing unit (id: 1111) asks the requestor (id: 2222) for tasks with the event description of "AskForTask". This computing unit sends its performance indicators to the requestor along with the same message.

Since all messages must pass through brokers in NB-CPS, the cost of transmission and the system performance will be decreasing if large transmission files are transmitting across nodes. Thus, messages are categorized into two different types of "event message" and "resource files". As shown in Fig. 10, the transmission between nodes is peer-to-peer when a message is notified as "resource files" type to lower the transmission costs in NB-CPS.

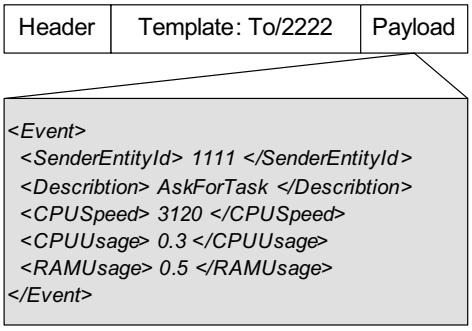

Fig. 9. An example message of NB-CPS.

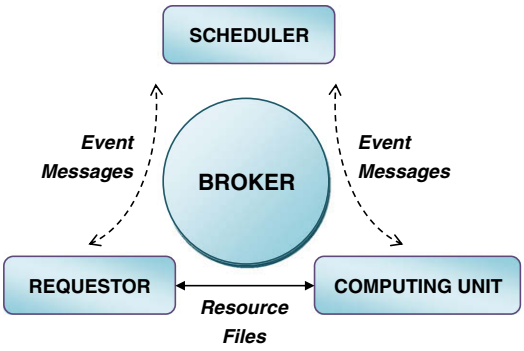

Fig. 10. Two different message transfer modes in NB-CPS.

4.2. System architecture of NB-CPS

There are fours roles in NB-CPS architecture: requestor, scheduler, computing unit and broker as shown in Fig. 11. The system functionality can be categorized into four layers: user layer, computing power sharing layer, monitor layer and communication layer. The enterprise employee can design and monitor their tasks at the user layer, conduct task scheduling, allocating execution and outcome reply in the computing power sharing layer. Each participating node could monitor other nodes' operation in monitor layer and the communication among nodes takes place in the communication layer. Detailed procedures for each role in each layer will be explained next.

4.3. Roles in NB-CPS

Any distributed computing system usually contains one or more nodes which have the computing requirements (referred to as requestor in NB-CPS), and one or more nodes that can share their computing resources (referred to as computing unit in NB-CPS). There should be a role of resource allocator to deal with complex transactions between these two roles (referred to as scheduler in NB-CPS). In addition, there are independent brokers which are responsible for the transmission of all messages to ensure the successful communication. Under such a P2P grid environment, a node can be a requestor, computing unit or scheduler at the same time. Any node leaves or disappears from the system will be monitored by broker and replaced immediately. Roles of NB-CPS are as following:

- *Requestor*

Requestor is the entry point for external users. Users define task's quantity, resource, priority...etc. to initiate the entire request process. Functions of requestor in each layer are introduced next.

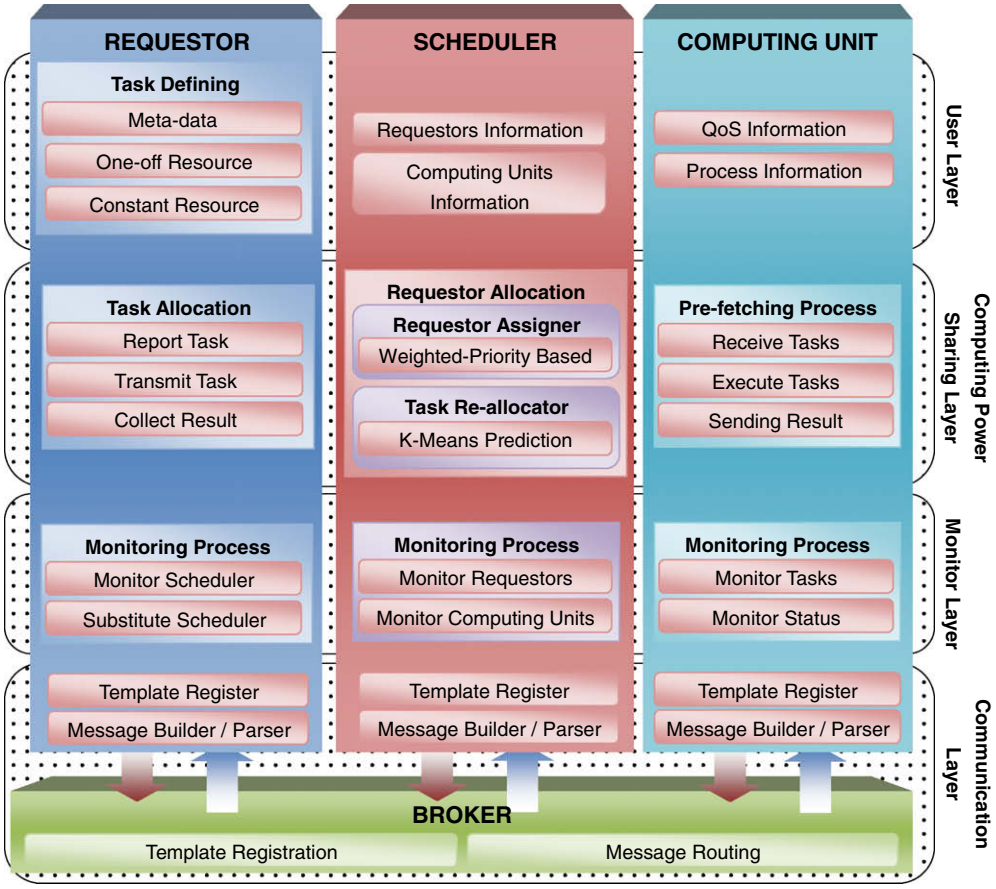

Fig. 11. The system architecture of NB-CPS.

Table 1
The computing task execution table

Meta-data	One-off resources	Constant resources
Broker IP	Executable files	Constant files
Task's name	One-off files	
Priority	Result files	
Quantity		

User layer: Users design their tasks in the user layer. Since NB-CPS is suitable for executing computing works which are able to be executed in batch format, this kind of task will contain three different types of resources as "meta-data", "one-off resource" and "constant resource" as shown in Table 1. "Meta-data" contain the detailed information associate with the task like task name, priority, quantity and broker IP,...etc. Resources used by every subtask are named "constant resource" and used only once per subtask are named "one-off resource". Therefore, constant resources can be transmitted at the beginning of the process and one-off resources can be transmitted along with each assigned subtask to reduce the transmission cost.

Computing power sharing layer: Requestors should elaborate their computing requirements to the system scheduler. After computing units assigned by scheduler, requester will transmit the task required information to computing units directly. In addition, requestor will get the associated information about the computing unit so it will analyze the computing unit performance continuously.

Monitor layer: Requestors periodically send testing messages to ensure the scheduler is still alive. If the scheduler does not respond to the requester within a given time, NB-CPS will consider that the scheduler is missing and will be replaced by the requestor who finds this event first. The replace process will be detailed later.

Communication layer: Requestors broadcast messages to brokers in communication layer in order to discover and connect to the system scheduler thereafter.

- *Scheduler*

To increase efficiency and reduce complexity in the distributed computing system, a resource allocation role is necessary while there are multiple requestors and computing units at the same time. Multiple schedulers are possible for NB-CPS while the network system is growing. Currently NB-CPS uses the "Centralized Scheduling Model" which coordinates all resources by a single scheduler to increase efficiency of the system due to the scale constraint of the network system in SME.

When the first requestor submits its request in NB-CPS, it will be assigned as a scheduler in the same time until the node leaves the system. This means there will be many requestors but only the first requestor will be the scheduler. Therefore, the scheduler is not there alone in the system but coexists with a requestor. The purpose of such design is applying the P2P characteristic that each node plays equal importance and every node could be the requestor, computing unit and be the scheduler as well. Since all the requestors will continuously communicate with the scheduler, the first requestor who finds the scheduler missing will be elected as the scheduler in NB-CPS. Functions of the scheduler are detailed as follows:

User layer: NB-CPS provides a visualized user interface to monitor working progresses which include tasks' information, task completed percentage and states of computing units.

Computing power sharing layer: Scheduler is responsible for task allocation. It allocates requestors' tasks to available computing units according to the task priority. NB-CPS uses "Weighted Priority Based Task Allocation Strategy" (Erradi et al., 2006) and tasks with higher priority will be allocated first. After all tasks are allocated, scheduler will start the task re-allocation process to utilize computing powers.

Monitor layer: To ensure all requestors are still alive, scheduler will monitor the requestors by recording the timestamp of triggering messages sent by requestors. If the interval of two trigging messages larger then the predefined threshold, NB-CPS will consider that the requestor is already gone, and its remaining tasks will not be allocated any more.

Communication layer: Scheduler discovers and connects to requestors and computing units through the broadcasting messages by brokers in the communication layer.

- *Computing unit*

The computing unit is the role which can share its computing resource in NB-CPS. Once the registered computing unit is idle, it will send its state and request for job execu-

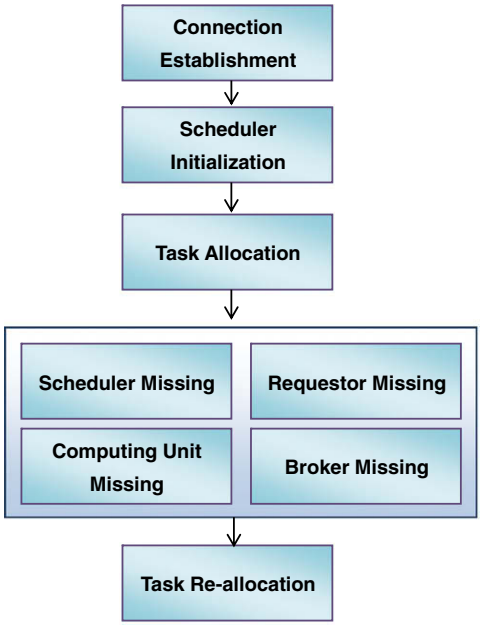

Fig. 12. The events list of NB-CPS.

tion to the scheduler. The scheduler will then assign an enrolled requestor to the computing unit so both parties can start to communicate for task execution. Functions of computing units are detailed as follows:

User layer: NB-CPS provides a visualized interface for users to monitor system conditions and working progress of the computing unit. The computing unit can decide to stop sharing its computing resource at any time and it can join or leave NB-CPS at its will.

Computing power sharing layer: When a computing unit is idle, it reports its computing power to the scheduler and communicates with an assigned requestor to get task's data, execute the task, and send the result back. As we mentioned earlier, there are one-off resources and constant resources from the requester. Constant resources are sent to the computing unit first and one-off resource will be sent next. For the needs of task re-allocation strategy, the computing units will send its current system performance indicators along with result to the requester.

Monitor layer: Computing unit monitors its current executing tasks. If the job is a sub-task, the data transmission of "Pre-fetching Process" will start which will pre-fetch the necessary data of the next sub-task and send back the previous processing result to the requestor.

Communication layer: Computing units discover and connect the system scheduler through broadcasting messages by brokers in the communication layer.

4.4. NB-CPS event flow diagram

There are many processes in NB-CPS and they could be described by the basic event flow diagram as shown in Fig. 12. All the processes must start with connection establishment, scheduler initialization, task allocation, some crashing handling events and end with task re-allocation. Since NaradaBrokering provides broker crashing handling

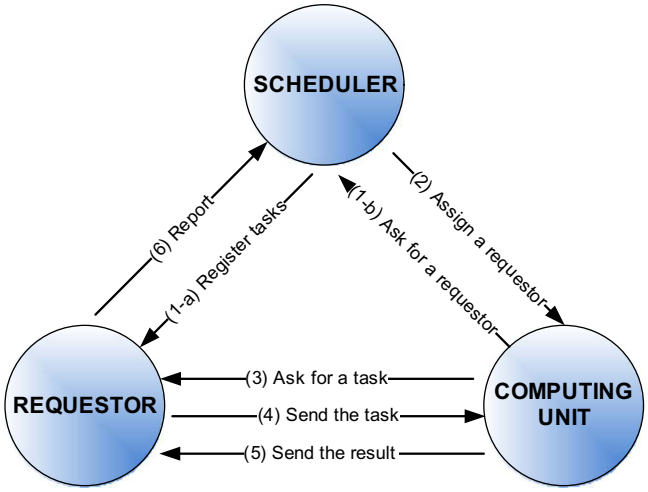

Fig. 14. The task allocation process.

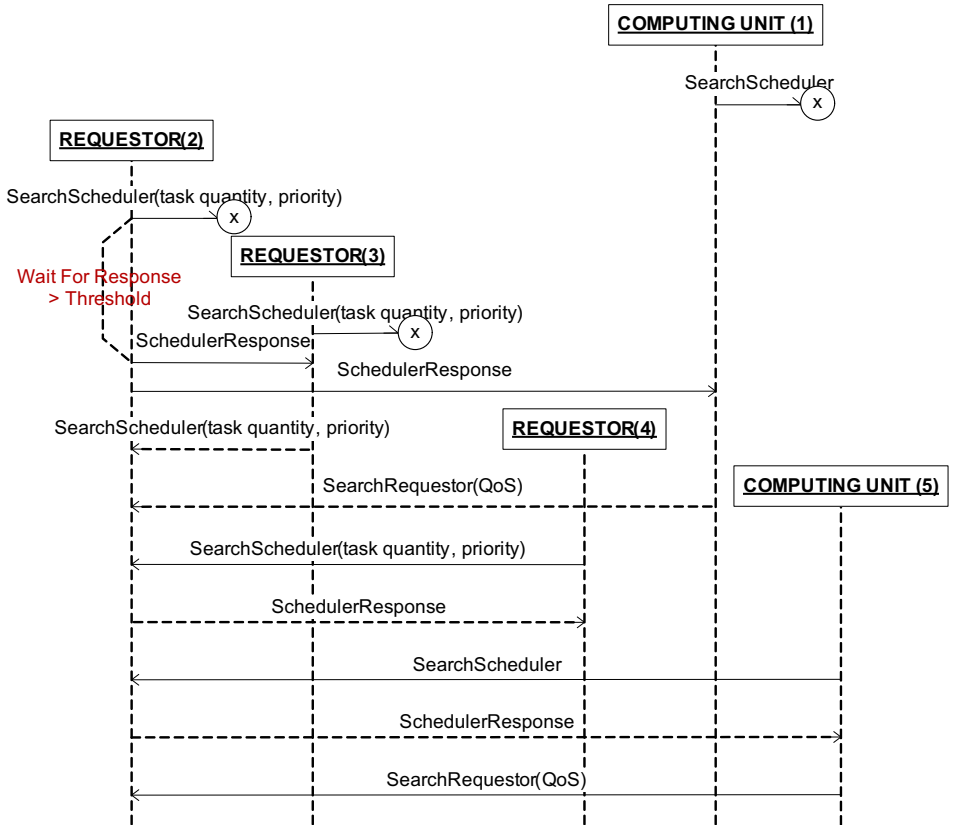

Fig. 13. The scheduler initialization process.

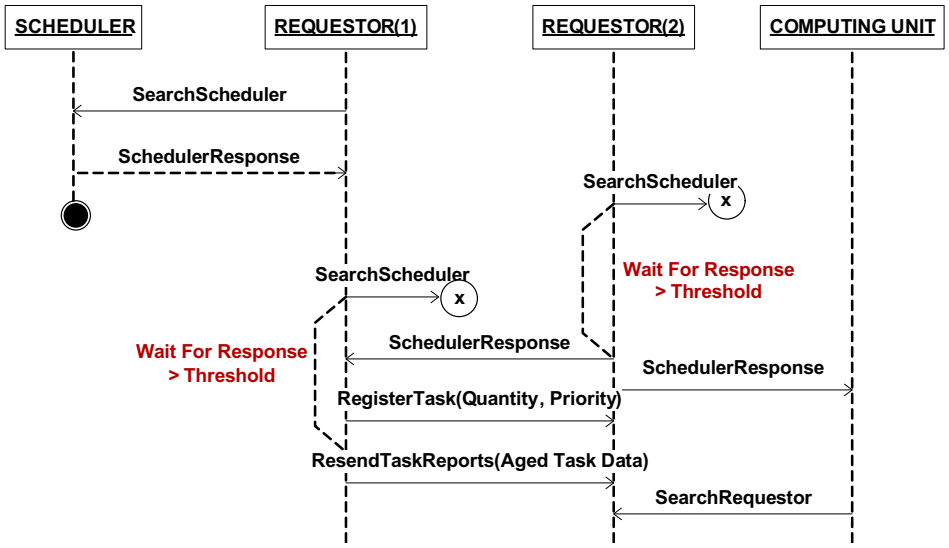

Fig. 15. The process of handling the scheduler missing event.

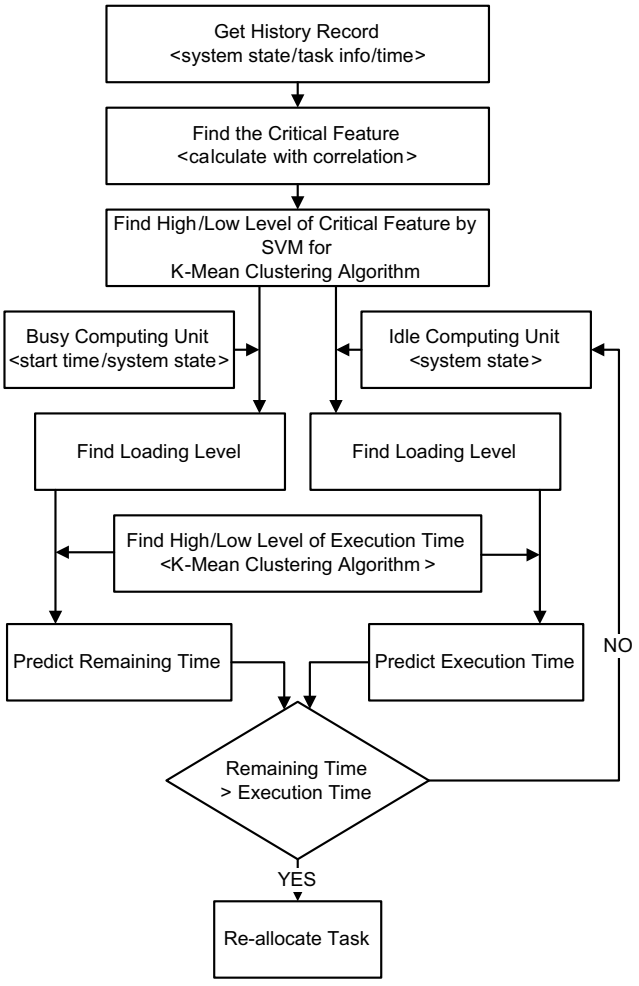

Fig. 16. The *K*-means prediction process.

mechanisms, the broker crashing events will not be discussed here. The events mentioned in Fig. 12 will be explained next.

- *Connection establishment*

Each node has to enter NB-CPS network and connect to a broker with a unique identification number. To play the roles in the system, each node has to subscribe to related templates. For example, a requestor has to subscribe to the template "ToRequestor". When a node wants to broadcast its message to all requestors in the system, it can simply publish its message through the template "ToRequestor". Other than that, each node has to subscribe to the template "To/X" where X is represented as its unique identification number. When a node wants to send a message to a node with the identification number "1111", it can simply publish its message through the template "To/1111".

- *Scheduler initialization*

The scheduler initialization process and interacting mechanism shown in Fig. 13 will explain how a requester knows there is already an existing scheduler and interacts with other roles. Supposing computing unit "1" enters the system first (the number "1" represents its identification number), it searches for the scheduler and reports its availability by broadcasting through the template "ToScheduler". At that moment, there is no scheduler in the system, thus, computing unit (1) will not be responded.

Table 2
The requirements of system environment

Characters	Environment
Requestor	Windows XP/2000/2003
Scheduler	Sun JAVA Runtime Environment1.4–1.5.11
Computing Unit	Microsoft Access 2000
Broker	Windows XP/2000/2003
	Sun JAVA Runtime Environment1.4 – 1.5.11
	NaradaBrokering 1.2.1

Next, requestor "2" comes into the system and searches for the scheduler by broadcasting its computing request through the template "ToScheduler". Due to the absence of the scheduler, requestor (2) will become a scheduler after a pre-defined waiting time and broadcast the role entitlement in the system (computing unit "1" and requestor "3") through the templates "ToComputingUnit" and "ToRequestor". After receiving this announcement in the system, requestors will resend their computing requirements to the scheduler and computing unit will resend their availability to the scheduler too. Requestor "4" and computing unit "5" who comes into the system subsequently will be responded by the scheduler after their requests are sent.

- *Weighted priority based task allocation process*

When there are un-allocated tasks, a message of computing unit's availability will trigger the task allocation process as shown in Fig. 14 for scheduler. The scheduler selects tasks according to their priorities. For example, suppose there are requestor A and B with priorities 5 and 1. When the task selection begins, A's task will have 5 times of possibility to be selected than B's task. After selecting a task, the identification number of the task's owner will be sent to the computing unit. If the computing unit haven't executes the requestor's task before, constant resources of the task will be sent first. In NB-CPS system, the file resources will be transmitted through peer-to-peer format and computing units will make reservations to requestors through Pub/Sub model before the P2P file transmission.

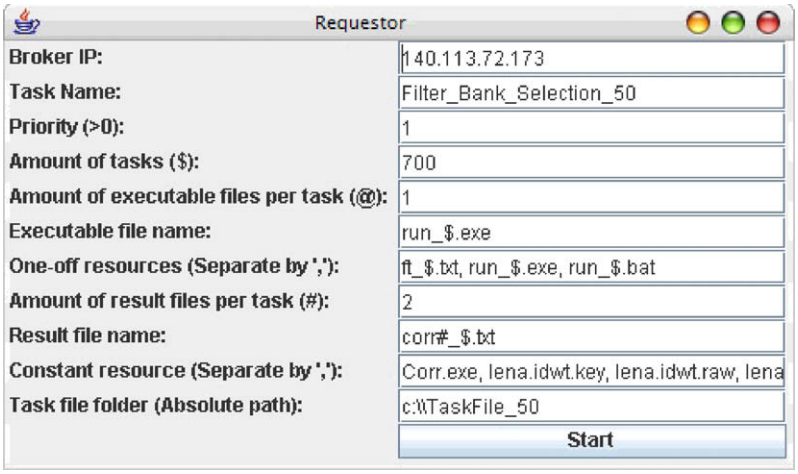

Fig. 17. The user interface of a system requestor.

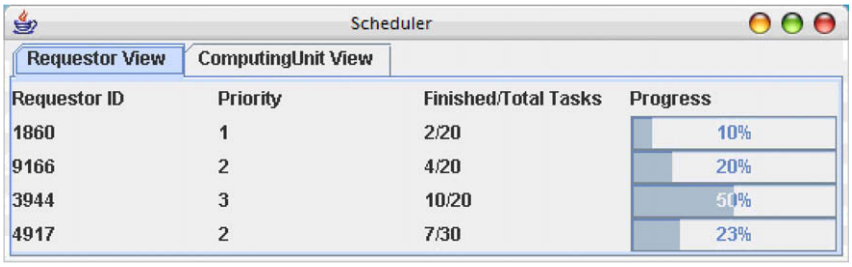

Fig. 18. The user interface of system scheduler monitoring system requestors.

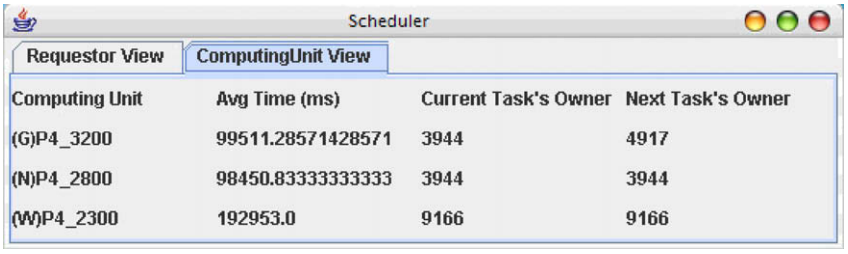

Fig. 19. The user interface of system scheduler monitoring computing units.

- *The missing scheduler event handling process*

To avoid the catastrophe of the entire system if the scheduler is missing unexpectedly, there is a missing scheduler event handling process in NB-CPS system and an example is shown in Fig. 15. Each requestor sends a testing message to the scheduler periodically. If the scheduler doesn't respond within a certain time period (five times longer than the time interval of two testing messages), the scheduler will be considered missing and will be replaced by a requestor who discovers the event first (requestor "2" in Fig. 15). The new scheduler will broadcast its new position to other nodes and associated required information for the new scheduler will be sent from the existing requesters.

- *The missing requestor event handling process*

To ensure all requestors are still alive, scheduler monitors requestors by recording the timestamp of testing messages sent by requestors. If the interval of two testing messages are larger then a threshold, the scheduler will consider that the requestor already left and its remaining tasks will not be allocated any more.

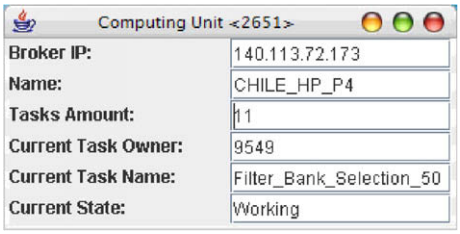

Fig. 20. The user interface of a computing unit.

- *The missing computing unit event handling process*

When a requestor sends a task to a computing unit, this timestamp will be recorded. When all other tasks of this requestor are finished, and the execution time of the unfinished task reaches a threshold (5 times than the average execution time), the tasks will be reported to the scheduler and will be reassigned to other available computing units.

- *The task re-allocation strategy using K*-means prediction process

While all subtasks are allocated, the task re-allocation process will be triggered if there are available computing units and busy ones. The goal of task re-allocation is to let idle computing units with higher performance acquire tasks which are executed by computing units with less computing powers to reduce the total execution time. The scheduler will first gather all unfinished subtasks' information including subtask's name, subtask's serial numbers, identification numbers of computing units, starting time and system status from the requestor. When an idle computing unit asks for new sub-tasks, the *K*-means prediction process will be triggered to re-allocate unfinished tasks as shown in Fig. 16.

The scheduler first analyzes historical data of finished subtasks to obtain the critical performance indicators of the tasks by calculating the correlation between the values of each performance indicator from each execution time. The K-means prediction process next divides the values

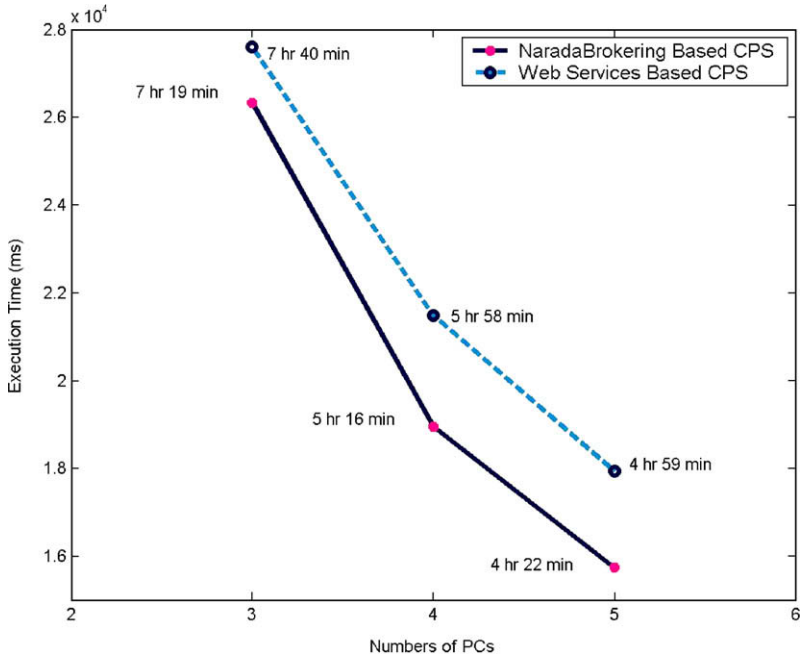

Fig. 21. The comparison of execution time between WS-CPS and NB-CPS with dedicated computing units.

of the critical performance indicators of historical data into two groups: high loading level and low loading level. From these two loading levels, the execution time of these two levels will be calculated accordingly.

Since the task reallocation process is triggered, the scheduler will estimate the system loading level of the idle computing unit and predict its execution time. If the remaining time of an unfinished task is longer than the predicted execution time of an idle computing unit, the task will be re-allocated. While multiple jobs could be reallocated, high priority subtasks will be reallocated first.

5. System implementation and performance analysis

The NB-CPS architecture is implemented in this study with open source software and the system requirement is shown in Table 2. The current system has completed the testing on JRE 1.4 to JRE 1.5.11. Fig. 17 shows the user interface of the requestor which includes the information of a broker IP address, priority of its task, resource's file name of subtasks, and so on. Multiple executable files of a subtask are also acceptable in the system in order to perform the sequential tasks. Fig. 18 shows the user interface

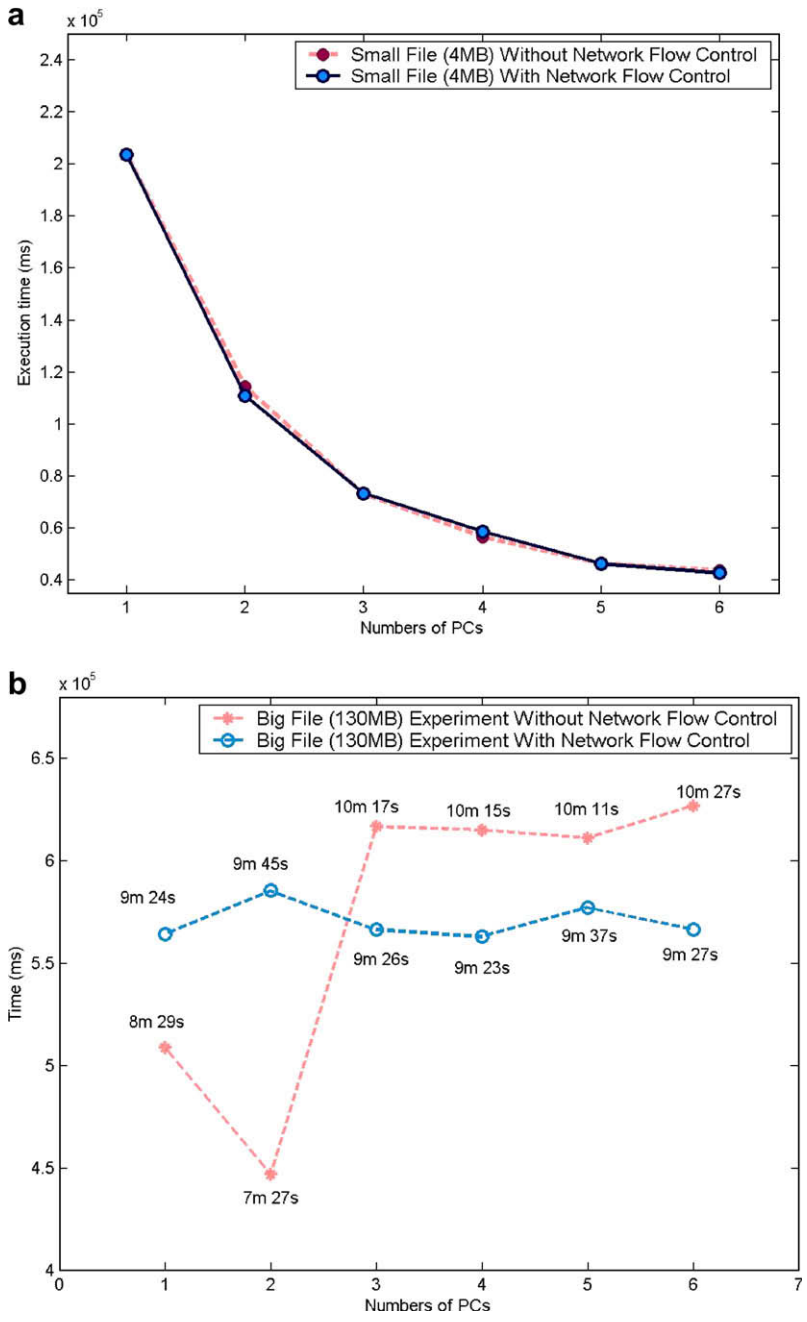

Fig. 22. (a) The execution time of 4MB file size with/without network flow mechanism for NB-CPS. (b) The execution time of 130 MB file size with/without network flow mechanism for NB-CPS.

of the scheduler who is monitoring the operation of the requestors with the progress status of each task. Fig. 19 shows the user interface of the scheduler who is monitoring the computing units with the current status. The user interface of a computing unit is shown in Fig. 20 with its current working state.

5.1. Distributed computing coordination experiment with a single requestor

In order to test the performance of the NB-CPS system, a digital watermarking algorithm with wavelet filter bank selection is performed (Tsai, 2004). The DWT-based digital watermark algorithm consists of decomposition, embedding, reconstruction, and detection procedures (Cox, Kilian, Leighton, & Shamoon, 1997; Tsai, 2004). Through the comparison of original watermark and embedded watermark from the attacked image, a similarity function based on correlation statistics is calculated and the authority of the digital image can be verified. By using the algorithm, digital copyright properties can be well protected and the ownership information can be preserved under attacks.

The wavelet filter-evaluation algorithm test total 76,177 filters which are grouped into several tasks with 100 filters

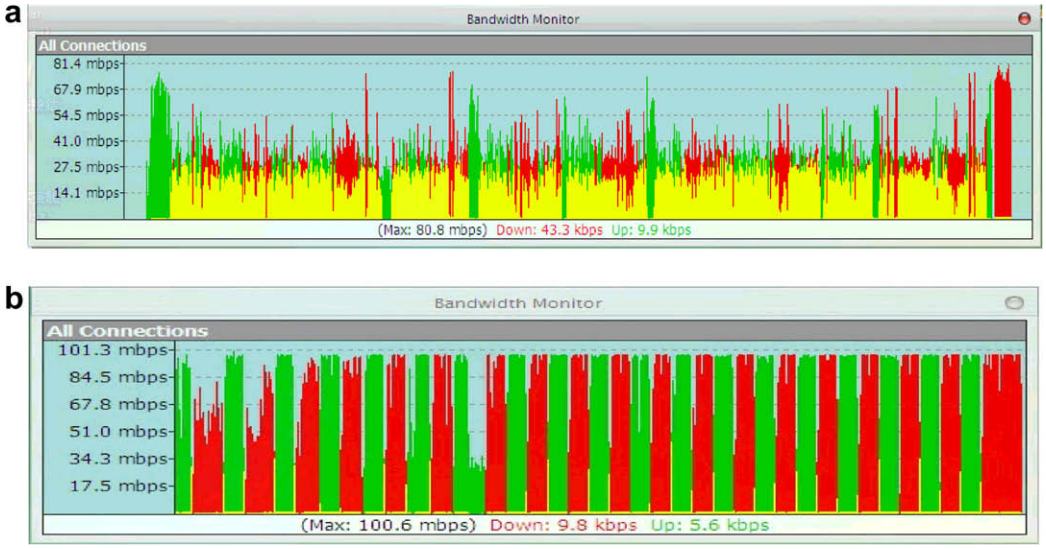

Fig. 23. (a) The network throughput demonstration without flow control mechanism. (b) The network throughput demonstration with flow control mechanism where red represents downloading flow, green represents uploading flow and yellow represents the flow of downloading and uploading at the same time. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

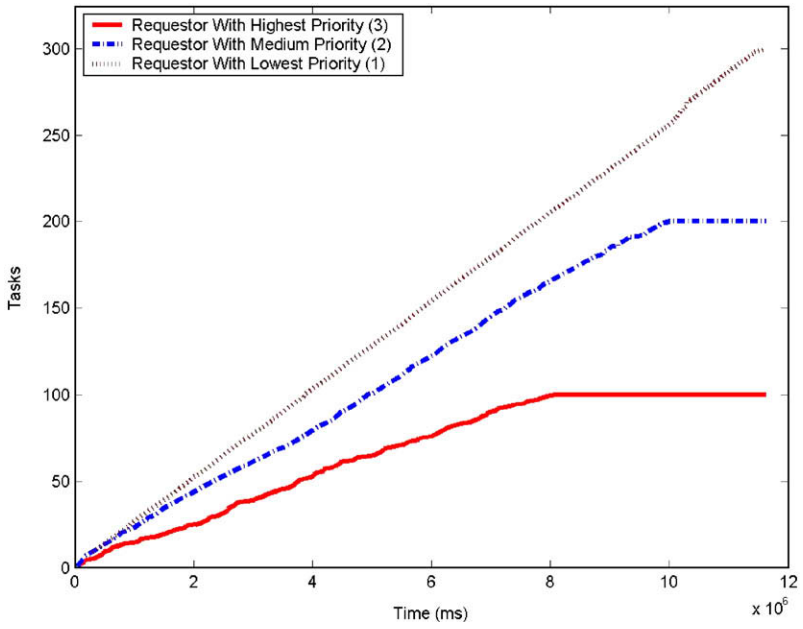

Fig. 24. The accumulated number of completed tasks with different priorities.

in each task. Each task is to extract the watermark from a watermarked raw image and JPEG2000 image by different filters and to calculate the correlation coefficient of two images. The task is highly dependent on the computing power and filters can be operated in parallel, thus, this experiment is suitable for the verification of the system coordination.

The computing units we choose are with similar performance (Intel Pentium 4 3.2 GHz and 512 MB DDRII) and they are all dedicated for the computing assignment. After steadily increasing the number of computing units in NB-CPS system, the total execution time is decreasing gradually. By comparing the NB-CPS with WS-CPS, we can find the NB-CPS has better system efficiency from Fig. 21 by about 10–15% improvement in reducing the total processing time.

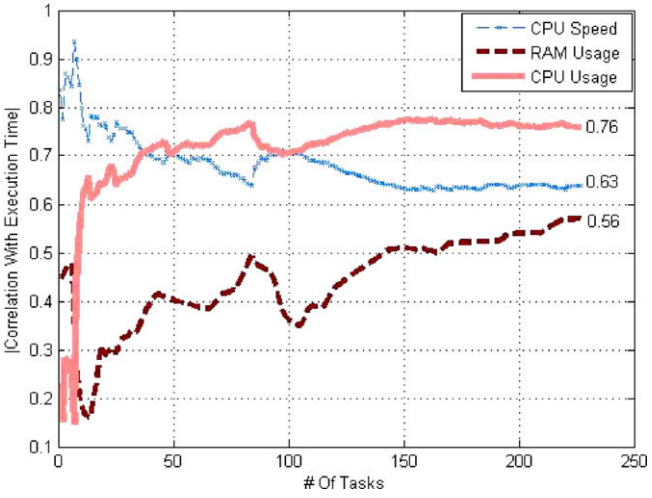

Fig. 25. Different performance indicators correlated with execution time.

5.2. The large file transmission experiment

As we mentioned previously that the WS-CPS exists the issue of large file transmission, we have proposed a network flow control mechanism for NB-CPS to deal with this problem and the simulation is performed under a network where the bandwidth is up to 100 MB/s. The operation is straightforward that the requestor will send 20 files of equal size to the available computing units and the computing units will directly send back the files to the requester. The transmission time versus the number of computing unit is shown in Fig. 22a, where the file size is 4 MB each transmitted through NB-CPS. Apparently, the processing time is decreasing while the number of computer units increases since the requestor could send files to available computing units while there are more available computing units. In Fig. 22a, the execution time for NB-CPS with or without flow control is almost identical which demonstrates the flow control does not incur any delay for system performance.

However, the simulation with the transmitted file size of 130 MB will have significantly different results as shown in Fig. 22b. Since large files consume more network bandwidth and the I/O time of the computing unit, the network transmission is congested when the amount of the computing units reaches the threshold ("3", in this case) and the execution time increases simultaneously. Fig. 23a demonstrates the network bandwidth utilization is not in peak value while there is no flow control mechanism. On the other hand, the network flow control mechanism helps the system with reliable flow transmission as shown in

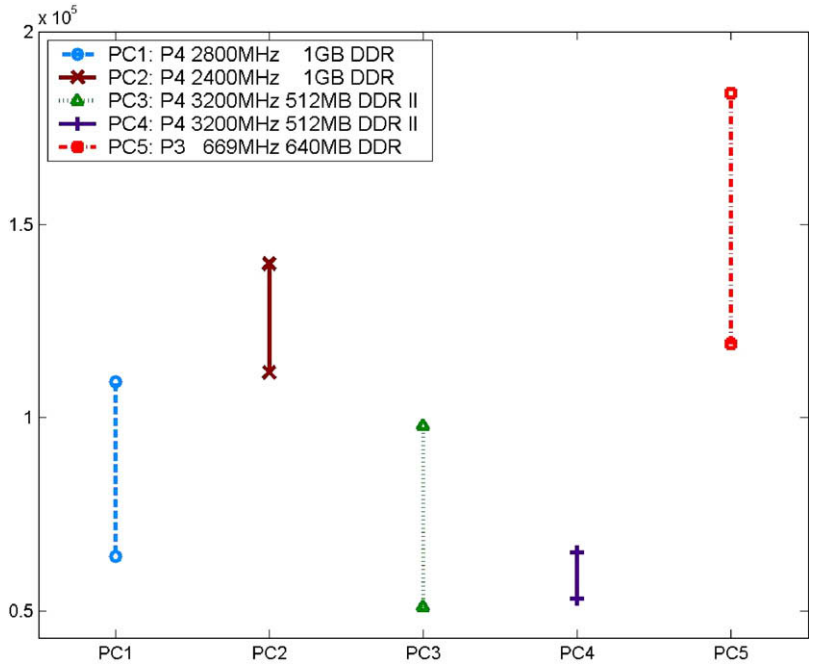

Fig. 26. The predicted execution time of 5 PCs with different loading levels.

Fig. 22b where the transmission time is almost the same when the computing units increase and utilize the network bandwidth efficiently which is demonstrated in Fig. 23b and the network flow almost remains at the peak speed. Therefore, the network flow control mechanism could achieve reliable quality of service for NB-CPS.

5.3. The verification of weighted priority based task allocation

While there are multiple requestors with different priority tasks, the simulation involves 3 requestors with three different priorities "1", "2" and "3" tasks for the filter bank selection experiment with 100 subtasks each. Priority 3's task has 3 times higher probability than 1's task and 1.5 higher probability than 2's task to be selected in the weighted priority based task allocation. As shown in Fig. 24. The accumulated number of completed tasks is increasing as times grows and the priority 3's tasks completed first, the priority 2's job completed next. However, NB-CPS system can manage multiple requesters with multiple priority tasks and the weighted priority bask task allocation could guarantee there is no starvation situation for low priority tasks from Fig. 24.

5.4. The verification of K-means prediction process

As noted in the previous section, NB-CPS collects the historical executing information and predicts the execution time of the task using *K*-means prediction process. It is essential to have accurate findings of the critical performance indicator for the task. In Fig. 25, it shows the correlation values of the system performance indicators when a computing unit executes the filter bank selection experiment under a dynamic environment. From the plot, "CPU usage" has the highest correlation value at 0.76 than CPU speed and RAM usage which should become the critical performance indicator of the task.

To verify the *K*-means prediction process, 5 computers with different capacity randomly running Super PI (Super, 2005) to simulate the dynamic environment is performed. Based on the SVM classification, two clustering has been grouped for each computing unit and the average execution time at the low and high loading are shown respectively in Fig. 26. Use this data for the *K*-means prediction process, we have performed 1000 times of the task allocation for NB-CPS and the average prediction error is calculated by the Eq. (3)

$$\text{Average prediction error} = \frac{\sum_{j=1}^{5}\sum_{i=1}^{200}\frac{|TP_ij - TR_ij|}{TR_ij}}{1000} \tag{3}$$

where TP_{ij} is the predicted processing time of task i at computing unit j and TR_{ij} is the actual processing time of task i at computing unit j. The average prediction error is only 11.0% which shows that "*K*-means prediction process" is an efficient approach of predicting the execution time.

It is also import to compare the performance between the "*K*-means prediction process" and the CC-FGDM of WS-CPS (Wang et al., 2007). CCC-FGDM is a fuzzy group decision-making technique which assigns each system performance indicator the same weighting which is generated by analyzing the historical records through regression analysis. The objective of this experiment is to find out whether "*K*-means prediction process" or CC-FGDM can effectively predict which computer has the minimum execution time. Same simulation environment for 5 computing units under dynamic environment is performed and the wrong prediction rate for *K*-means predication

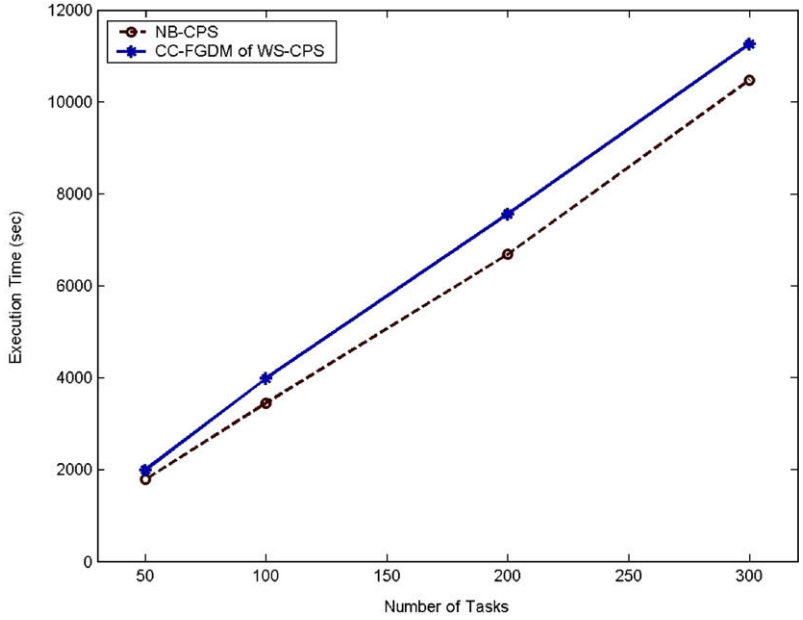

Fig. 27. The execution time comparison between NB-CPS and CC-FGDM under dynamic environment.

process is 1.77% but CC-FGDM is 22.8% for reallocation. Therefore, the data demonstrates *K*-means prediction could have sufficient prediction capability. The overall processing time comparison is shown in Fig. 27. While the task number is increasing, the execution time is increasing for both schemes but NB-CPS is always with shorter processing time than CC-FGDM of WS-CPS approach by about 10% improvement. Therefore, we can conclude the NB-CPS design has superior performance than WS-CPS techniques under dynamic environment.

6. Conclusion

In this article, NaradaBrokering Based Computing Power Service (NB-CPS) is proposed which is not only resolve the issues in WS-CPS such as system resilience, fault tolerance, efficiency of job scheduling and the instability in congested network environment, but also efficiently execute computation intensive works. NB-CPS sufficiently utilizes the P2P grids integration with the bandwidth flow control mechanism by FCFS reservation strategy, weighted priority based task selection, SVM classification and *K*-means prediction process to resolve the task allocation strategy. Experiments confirmed that NB-CPS has superior effectiveness in single requestor situation, multiple requestors situation, interfered environment, and in high traffic environment.

Acknowledgements

This work was supported by the National Science Council in Taiwan, Republic of China, under Grant NSC95-2416-H009-027 and NSC96-2416-H009-015.

References

Banavar, G., et al. (1999). An efficient multicast protocol for content-based publish-subscribe systems. In *Proceedings of the IEEE international conference on distributed computing systems*.

Casanova, H. (2002). Distributed computing research issues in grid computing. *ACM SIGACT News Distributed Computing, Column, 8*, 50–70.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: A library for support vector machines, software. Available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

Cox, I. J., Kilian, J., Leighton, F. T., & Shamoon, T. (1997). Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing, 6*(12), 1673–1687.

Erradi, A., et al. (2006). Differential QoS support in Web Services Management. In *International conference of Web services, ICWS '06* (pp. 781–788).

Foster, I. (2002). The grid: A new infrastructure for 21st century science. *Physics Today, 55*(2), 42.

Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organization. *International Journal of High Performance Computing Applications, 15*(3).

Fox, G., et al. (2003). NaradaBrokering: A middleware framework and architecture for enabling durable peer-to-peer grids. In *Proceedings of the international middleware conference*.

Linpack (2006). <http://www.netlib.org/linpack/>.

MacQueen, J. B. (1967). Some methods for classification and Analysis of Multivariate Observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, pp. 281–297). Berkeley: University of California Press.

Plank, J. S. et al. (2001). Processor allocation and checkpoint interval selection in cluster computing systems. *Journal of Parallel and Distributed Computing, 61*(11), 1570–1590.

Segall, B., Arnold, D. (1997). Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings AUUG97* (pp. 243–255).

Super, P. I. (2005). Available at: <http://www.super-computing.org/> (Accessed December, 2005).

The NaradaBrokering Project (2006). Indiana University. <http://www.naradabrokering.org/>.

Tsai, M. J. (2004). Filter bank selection for the ownership verification of wavelet based digital image watermarking. In *Proceedings of the 2004 international conference on image processing* (Vol. 5, pp. 3415–3418).

Tsai, M. J. et al. (2006). A collaborated computing system by web services based P2P architecture, computer supported cooperative work in design II. *Lecture Notes in Computer Science, 3865*, 194–204.

Tsai, M. J., & Wang, C. S. (2008). A computing coordination based fuzzy group decision making (CC-FGDM) for web service oriented architecture. *Expert Systems With Applications, 34*(4), 2921–2936.

Viktor, S., Eide, E., et al. (2003). Extending content-based publish/subscribe systems with multicast support, Technical Report 2003–03, Simula Research Laboratory, pp. 1 (July 2003).

Wang, S. C. et al. (2007). A hybrid load balancing policy underlying grid computing environment. *Computer Standards and Interfaces, 29*, 161–173.