Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 34 (2008) 2921–2936
Expert Systems with Applications
www.elsevier.com/locate/eswa

A computing coordination based fuzzy group decision-making (CC-FGDM) for web service oriented architecture

Min-Jen Tsai *, Chen-Sheng Wang

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

Abstract

As the demands for faster data processing and enterprise computing are increasing, the traditional client/server architecture has gradually been replaced by Grid computing or the peer-to-peer (P2P) model which can share the content or resources over the network. In this paper, a new computing architecture – computing power services (CPS) – has been applied to utilize web services and business process execution language for overcoming the issues about flexibility, compatibility and workflow management. CPS is a lightweight web services based computing power-sharing architecture, and suitable for enterprise computing tasks which can be executed in the batch processes within a trusty network. However, a real-time load balance and dispatching mechanism is needed for distributed-computing architecture like CPS in order to handle computing resources efficiently and properly. Therefore, a fuzzy group decision-making based adaptive collaboration design for CPS is proposed in this paper to provide the real-time computation coordination and quality of service. 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. This scheme increases the overall computing performance and shows stability for the dynamic environment.

Keywords: Business process execution language; Fuzzy decision-making; Web services

1. Introduction

There are widely increasing demands for faster data processing and enterprise computing, traditional client/server architecture has gradually been replaced by Grid computing or peer-to-peer (P2P) model (Loo, 2003) which can share content or resources over the network. Though both distributed-computing models are suitable to perform distributive tasks, small medium enterprise (SME) still encounter the issues of security, motivation, flexibility, compatibility, and workflow management during implementation (Tsai, 2004). Therefore, computing power services (CPS) architecture is proposed to address these issues with assumption to deploy CPS in a trusty network.

CPS is based on service oriented architecture (SOA) to utilize web services and business process execution language (BPEL) to overcome the issues of P2P about flexibility, compatibility and workflow management. Web services have gained acceptance to integrate business applications inside enterprise because of the characteristics of loosely-coupling and open standard support. Within CPS architecture, three roles are defined as the computing node, computing requester and coordinator to act as the role of service consumer, service provider, and service register in SOA, respectively. The detailed operation will be introduced in Section 2.

Since CPS aims to run tasks independently among computing units which voluntarily provide their spare resources, how to coordinate resources within a distributed environment is crucial to system performance. Therefore, resource allocation problem can be categorized as a decision problem by choosing the best alternatives with most available resources like memory and CPU time for CPS.

* 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 © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.05.017

Group decision-making (GDM) has been an important research area for industries and academies. Several decision models such as the analytic hierarchy process (AHP) with multiplicative preference relation (Saaty, 1980) and the resolution process of GDM (RPGDM) with fuzzy preference relation (Chiclana, Herrera, & Herrera-Viedma, 1998; Chiclana, Herrera, & Herrera-Viedma, 2001; Kacprzyk, 1986) have shown good performance in many applications. Since the RPGDM uses the fuzzy set (Zadeh, 1983) to handle the uncertainties and help translating precise information, it is similar to the uncertain behaviour of computing nodes in CPS which allows users to start or stop program locally. However, only RPGDM is not enough to handle the dynamic situation for CPS and this study will develop a novel coordinating mechanism called CC-FGDM to implement the resource-sharing module within the CPS.

This paper will be organized as following. Section 2 will explain the distributed architectures of CPS and RPGDM. In Section 3, problems of load-balancing and resource coordination will be discussed. A novel computing coordination based on fuzzy group decision-making abbreviated to CC-FGDM is proposed in Section 4. The implementation and performance comparison of CC-FGDM is analyzed in Section 5 and conclusion will be given in Section 6.

2. Literatures review

This section will be dedicated to review related researches in distributed-computing, CPS architecture, computing performance benchmarking and fuzzy group decision-making.

2.1. Distributed-computing

There are two main categories of distributed-computing that is P2P computing and Grid computing. P2P computing depends on a central server to divide a computing task into several tasks, which will be assigned to voluntary computers for execution and returning computing results to the central server. In Grid computing architecture, computer network is considered as a virtual computer to shuffle resources for computation. There are four aspects which differs P2P computing from Grid computing.

Resource management: The resource management of Grid computing is generally a static and hierarchical structure in which the proprietary software is needed to search for resources. Conversely, the resources of P2P computing are such more dynamically grouped together with high flexibility.

Participant: Since Grid computing is originally intended to solve complicated scientific computation with large amount of data, it is implemented as a virtual super computer by grouping computers in different geographic area in order to provide a widespread, secure and collaborative resource-sharing environment. Therefore, participating computers are generally high performance mainframes or workstations and connected in high-speed network. However, it is often required to apply for accounts in order to use the computing resource and the strict requirement will limit the number of participating nodes.

On the other hand, most participating nodes of P2P computing are personal computers with different capable computing power. Those participating nodes are directly connected to provide their own disk space or CPU time without account constraint. Hence, the number of participating nodes in P2P computing is often larger than the one in Grid computing.

Reliability: Because most resources in Grid computing environment reside on few participating nodes by a centric and hierarchical structure, whole structure is more stable than the one in P2P computing environment where nodes can be in and out dynamically without constraints.

Workflow management: Several workflow management systems such as GridAnt, GSFL, Pegasus, Karajan or GFMS (Amnuaykanjanasin & Nupairoj, 2005) have been developed in Grid computing. However, specific propriety languages are required to design workflow in these systems for service coordination. Moreover, what those languages are not compatible with each other makes interoperability among systems impossible. Therefore, Amnuaykanjanasin and Nupairoj (2005) proposed a common standard to implement workflow management in Grid computing in order to make one design running in different workflow systems by using an open standard such as BPEL. As for workflow management in P2P, although workflow applications are inherently distributed, most of traditional workflow systems are based on client–server technology in order to satisfy the functional requirement of the system (Yan, Yang, & Raikundalia, 2006). Therefore, only few workflow management systems such as SwinDeW (Yan et al., 2006) are proposed to facilitate workflow in P2P architecture.

2.2. CPS architecture

CPS is based on the architecture of web services and it inherits the characteristics of service oriented architecture (SOA) which consists of three participants that are service requester, service provider and service broker (Fig. 1). The description of three roles will be explained as follows.

2.2.1. Role definition of CPS

The role of coordinator: The coordinator acts as a service broker to fairly negotiate between the computing node (service requester) and computing requester (service provider). Its main function is to maintain a list to record the URL and requirement of computing requester. This list will be created when the computing requester publishes its web service in coordinator. If computing node asks for tasks through the coordinator, the coordinator will assign the URL of computing requester from the list to computing node by the round-robin order. Afterwards, the computing node will use the assigned URL to communicate with the computing requester directly.

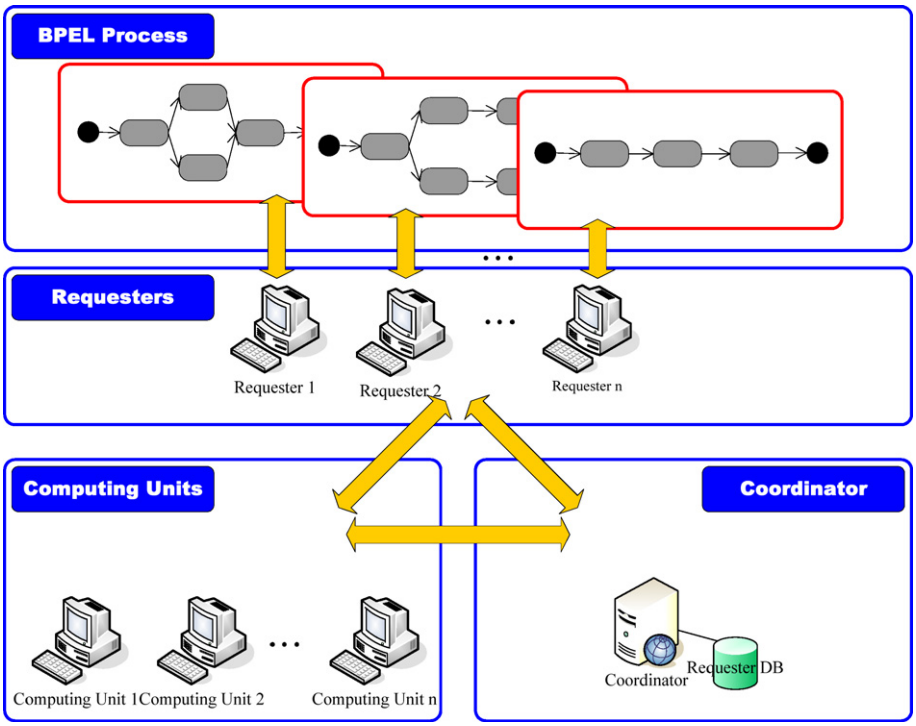

Fig. 1. Diagram of CPS components.

In addition, the function of account and audit management will be also implemented by the coordinator for the authentication reason. This role is corresponding to the role of UDDI in SOA.

The role of computing requester: Before the request, the requesters should design their experiment processes by a visualized BPEL development environment as shown in Fig. 1 and publish their requirement through the coordinator. In addition, it will assign tasks to computing nodes using the workflow management capability.

The role of computing node: Computing node is responsible for executing computation. The computing node inquires coordinator to ask for tasks when it has spare computing resource. After getting the requester's URL of web services, it negotiates with the requester to download the tasks description along with the required task files. The node then starts to execute tasks and replies results to the requester when tasks are finished. Such scenario will continue until all assigned tasks are completed.

2.2.2. Layer description

Tsai (2004) first systematically introduced the CPS architecture as the five layers modules and this study will extended the design. The five layers are power-sharing, communication, contract, service discovery and user layers. Excluding user layer, the other four layers comprise the P2P power-sharing middleware which is the core of CPS. The interaction among roles and layers of CPS are demonstrated in Fig. 2.

The user layer: A GUI interface is provided for users in this layer to facilitate workflow design and management. While it is implemented at the beginning of requester for workflow design and management, it provides a mechanism for users to participate or stop in sharing their spare resources.

The resource-sharing layer: This layer corresponds to the description layer of web services. It describes the interaction between requesters and computing nodes.

The communication layer: This layer uses the communication mechanism of web services, i.e. SOAP to bridge the components of CPS.

The contract layer: The interaction between computing node and requester is contracted in this layer. The contract presented in XML format is composed of task description and URL of requester.

The service discovery layer: This layer is implemented in coordinator as a broker agent to match requester with computing nodes. The coordinator does not involve in computation and works like UDDI in SOA.

2.3. Fuzzy group decision-making

Because of the behaviour of computing nodes in CPS which allows users to start or stop program at their demand, RPGDM which is a fuzzy group decision-making method will be used to handle such uncertainty during the resource collaboration in this paper.

The RPGDM collects opinions about possible alternatives to a problem from a group of experts to find their consensus. If consensus level is below a predefined threshold, experts will provide their new opinions to resolve conflicts. Chiclana et al. (1998, 2001) proposed using three steps that

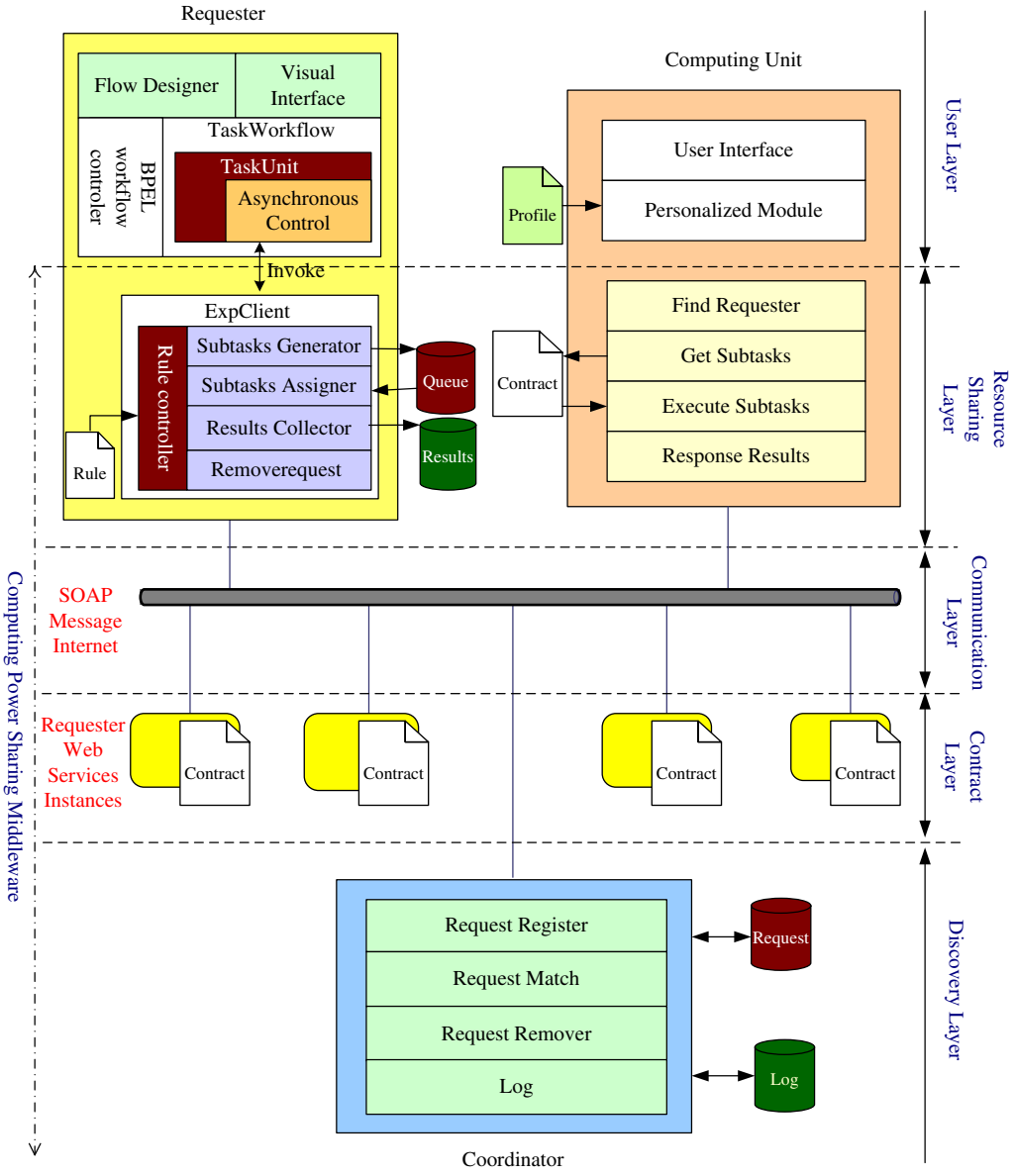

Fig. 2. Diagram of CPS architecture.

are transformation, aggregation and exploitation phases in whole decision-making processes. Before we introduce the procedure of the RPGDM, we need define the decision-making problem first.

Suppose that there exists a finite set of alternatives $A = \{a_1, a_2, \ldots, a_n\}$, $n \geqslant 2$. The RPGDM will select the alternatives from best to worst using the information which are provided by the finite set of experts $E = \{e_1, e_2, \ldots, e_m\}$, $m \geqslant 2$. For each expert, the information is used to evaluate preference level among elements in the alternative set. According to Chiclana et al. (2001), the experts' preferences over the set of alternatives may be represents by one of four ways: a preference ordering among alternatives, fuzzy preference relation, a multiplicative preference relation, and a utility function. While the RPGDM bases on fuzzy preference relation to aggregate preference information, a preference ordering among alternatives is the easiest way to use. We will use preference ordering as the evaluation way of preference by using the performance indexes about CPU, memory, or execution time. Therefore, it is easy to permute the preference order over alternative set according to the values gathered by our resource coordination module.

2.3.1. The transformation phase

As RPGDM adopts the fuzzy preference relations as a way to aggregate preference information, we need to transform our evaluating approach, preference ordering, to the fuzzy preference relation. The fuzzy preference relation is defined as P^k_{ij}, $1 \leqslant i, j \leqslant n$, $1 \leqslant k \leqslant m$ which describes the degree to which an alternative a_i is preferred to an alternative a_j by expert e_k. A transformation function presented in

Chiclana et al. (1998) from preference order to fuzzy preference relation is given as follows:

$$p_{ij}^k = \frac{1}{2}\left(1 + \frac{o_j^k - o_i^k}{n-1}\right) \tag{1}$$

Here o_j^k represents the preference order of alternative a_j in alternative set by expert e_k.

2.3.2. The aggregation phase

After we transform the preference ordering for each expert to fuzzy preference relation, an aggregating operation will be applied to get a collective preference relation $P_{ij}^C,\ 1 \leqslant i,j \leqslant n$. The RPGDM uses fuzzy quantifiers to represent the fuzzy majority concept, and the Ordered Weighted Average (OWA) operator to aggregate preference information in order to obtain P_{ij}^C. The criteria of majority is traditionally defined as a threshold number of individuals, whereas fuzzy majority is a soft majority concept. It is calculated via a fuzzy logic based calculus of linguistically quantified propositions (Kacprzyk, 1986; Yager, 1988).

The OWA operator which was proposed by Yager (1988) defines a family of aggregating operators between the Min and Max operators. Before we use the OWA operator, a weighting vector $W = \{w_1, w_2, \ldots, w_m\},\ w_i \in [0,1],\ \sum_{i=1}^m w_i = 1$ must be defined to aggregate the preference value collection $\{P_{ij}^1, P_{ij}^2, \ldots, P_{ij}^m\}$ for P_{ij}^C. An OWA operator of dimension m is a function with fuzzy quantifier Q

$$F_Q(P_{ij}^1, P_{ij}^2, \ldots, P_{ij}^m) = \sum_{k=1}^{m} w_k \cdot p_k \tag{2}$$

in which p_k is the kth largest value in preference value collection $\{P_{ij}^1, P_{ij}^2, \ldots, P_{ij}^m\}$.

How to get a weighting vector is a key point when OWA operator is applied. In Wiki (2006), an operator with non-decreasing proportional quantifier Q is given by the formula

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right), \quad i = 1, \ldots, n, \quad \text{where}$$

$$Q(r) = \begin{cases} 0 & r < a \\ \frac{r-a}{b-a} & a \leqslant r \leqslant b \\ 1 & r > b \end{cases} \quad a, b, r \in [0,1].$$

For example, if "at least half" quantifier $[0, 0.5]$ is used, a weighting vector of dimension 4 will be $\left\{0, 0, \frac{1}{2}, \frac{1}{2}\right\}$.

2.3.3. The exploitation phase

In this phase, the RPGDM uses two fuzzy ranking methods, quantifier guided non-dominance degree (QGNDD) and quantifier guided dominance degree (QGDD) to identify priority order of alternatives. While QGNDD represents the degree to which each alternative is not dominated by a fuzzy majority of remaining alternatives, QGDD quantifies the dominance degree to which an alternative has over a fuzzy majority of the others. The QGNDD of alternative a_i will be computed by the following expression

$$\text{QGNDD}_i = F_Q(1 - \max\{p_{ji}^c - p_{ij}^c\},\ j = 1, \ldots, n,\ j \neq i) \tag{3}$$

As for QGDD, we compute it by

$$\text{QGDD}_i = F_Q(p_{ij}^c,\ j = 1, \ldots, n,\ j \neq i) \tag{4}$$

With these two dominance degree, the choosing process will use the rules as follows to obtain the priority order of alternatives.

(1) If there exists UND element (i.e. QGNDD(a_i) is equal to 1), the ranking method QGDD will be used to obtain the priority order.
(2) Otherwise, QGNDD is the ranking method.

3. Problem analysis

3.1. Problems of load-balancing

Since tasks are assigned randomly to computing nodes within the distributed-computing architecture like CPS, load unbalancing occurs during the operations among the computing nodes. In order to improve the system efficiency by reducing the executing time and dispatch the job reasonably, load-balancing issues will be discussed in the following:

Estimation of performance efficiency and load balance: Since computation within the distributed-computing environment is performed dynamically and cooperatively by participating nodes, it is necessary to monitor the latest load and performance efficiency in each node. Such monitoring is especially important for P2P like environment in which nodes contain a wide variety of computing power, it will not be enough to just assign tasks to the first available node but to the first appropriate and available node after considering load in each node.

However, the estimation of node loading is not easy since its value is a crisp and the loading comparison is relative and fuzzy.

Computing coordination: How to coordinate resources is a critical issue and can be categorized as in dynamic or static way. The main difference between them is whether the latest loading status is considered during the coordination. Static coordination such as random or round-robin order uses predefined scheme to assign tasks without up-to-date loading information, whereas dynamic method chooses the best node according to the latest loading statuses.

Executing stability: Many research articles which discussed load-balancing in distributed-computing are often focused on performance efficiency such as response time, throughput or execution time. However, executing stability is also important and is defined as the same task could be

finished in a tolerance time range when it is repeatedly executed under similar conditions in a distributed-computing environment. This feature will help the computation requester to predict when a task will be completed.

Adaptive coordination: The common factors to evaluate loading status are CPU usage, CPU processing time or memory usage and they should be considered in all not by a single factor. Therefore, the mechanism to find out the relation between factors and task type is urgently needed for distributed-computing. This mechanism will adapt weighting in the formula of estimation load status according to the task type. For example, while CPU-intensive task is executed, task is assigned to those nodes with better CPU-related factors. Conversely, memory-intensive task is assigned according to memory-related factors.

3.2. Possible solutions

Three following mechanisms are proposed to address the above issues in this research.

Performance benchmark: To use dynamic coordination, the latest performance information must be evaluated before making decision. Performance information can be collected by three ways that are standard performance benchmark, performance evaluation strategy and system performance indexes. Standard performance benchmark such as SPEC CPU2000 uses specific core set of programs to accurately get loading information. However, it spends too much time to perform the benchmark testing which is not practical in a P2P like environment. For example, SPEC CPU2000 is time consuming operation by executing a core set of programs (Null & Lobur, 2003). Similarly, performance evaluation strategy uses evaluating strategy and experiments to get more loading information but still spends too much time to perform benchmark. Compared to previous approaches, system performance indexes which can be obtained directly from the computing units are more appropriate in distributed-computing.

Fuzzy group decision-making: Since system performance indexes provide useful knowledge about system loading and each index has its own view about the system, it is natural to designate these indexes as a group of experts to decide which node is the best to assign the task. Besides, nodes can join CPS voluntarily to provide resources for executing tasks by allocating spare memory and CPU time. Therefore, resource coordination problem can be categorized as a group decision-making problem for choosing the best nodes with most available resources.

Regression and correlation analysis: Since the task type can be CPU-intensive or memory-intensive, weighting for each performance index can not be same for different task type while making coordination decision. Moreover, contribution of each index to overall performance is varied. Thus, regression and correlation analysis will be applied to decide weighting of each index by using historical and latest loading information in order to improve the performance stability. Such analysis and weighting adjustment should be dynamically performed during the resource coordination.

4. Fuzzy group decision-making coordinating mechanism

According to the discussion in previous sections, the mechanism of computing coordination-based on fuzzy group decision-making abbreviated to CC-FGDM is proposed and applied to CPS in this paper. In order to offer the flexibility during the implementation, CC-FGDM will be implemented by three separate modules which first integrates RPGDM as the decision-making rule, system performance indexes as the benchmarks, and coefficients of correlation and determination to decide weighting of indexes. The CC-FGDM mechanism is operated by four policies that are information retrieval strategy, task assignment strategy, task reassignment strategy and dynamic

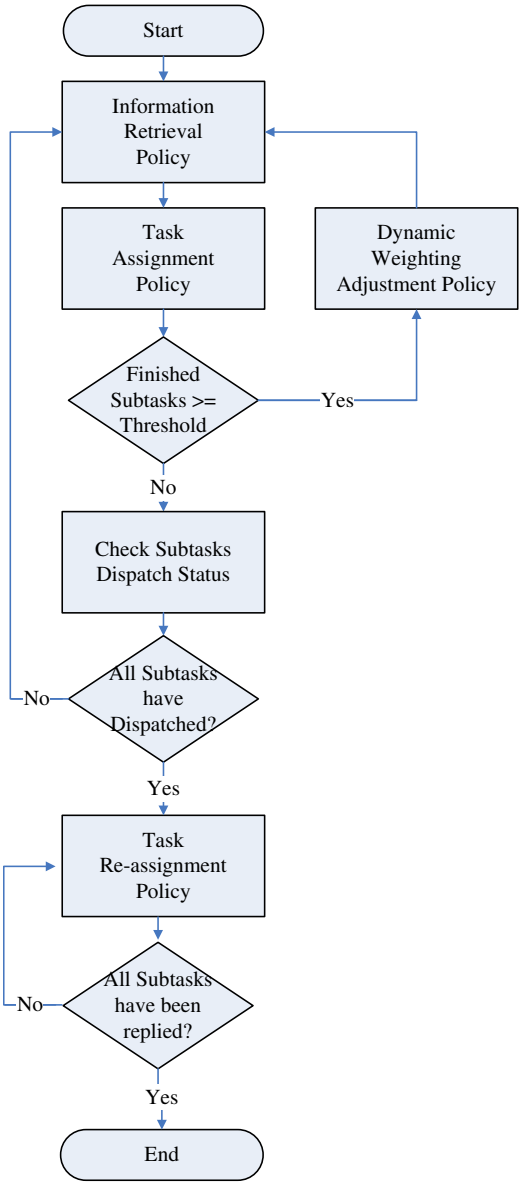

Fig. 3. The flow diagram of CC-FGDM.

weighting adjustment strategy. Fig. 3 illustrates the flow diagram and relationship among these strategies.

4.1. Information retrieval strategy

The purpose of information retrieval strategy is to monitor the loading of each node and detect whether a node is available by using the following indexes.

- CPU-relative indexes
  - CPU clock rate
    Normally, CPU is benchmarked by clock rate in hertz. It means how many instructions are executed per second. The higher CPU value means better performance in general.
  - CPU usage percentage
    This index depicts CPU's processing time versus idle time. The higher value means CPU has higher loading.

- Memory-relative indexes
  - Memory usage percentage
    This index shows usage percentage of physical memory in a node. The higher value means higher usage of memory.

- Other indexes
  - Average execution time
    The higher value of this index entails that a task has longer execution time. It reflects directly the latest performance of a node.

4.2. Task assignment strategy

This strategy which is a core part of CC-FGDM uses loading records from information retrieval strategy to assign task to the most appropriate node. It integrates RPGDM (Chiclana et al., 1998, 2001) to make group decision and will go through four phrases including problem definition, transformation, aggregation and exploitation to choose the best node. The following section will explain the operation of these four phrases.

1. Problem definition phrase
   Since this research models resource coordination as the problem of finding the best alternative according to opinions from a group experts, the alternative set A and the expert set E will be defined as

   $A = \{\text{available computing nodes}\}$

   $E = \{\text{CPU clock rate, CPU usage percentage, memory usage percentage, average task-executing time}\}$

2. Transformation phrase
   The purpose of this phrase is to transform different representation of preference from experts with varied background to the same representation in order to calculate consensus degree among them. Since system performance indexes are considered as experts in this research which are rational and intuitively represented by numerical value, preference between two indexes will be represented by ordering according to their values. However, the proposed group decision-making method uses fuzzy preference relation to aggregate preference, formula (1) will be used in this phrase to transform preference from ordering to fuzzy relation.
3. Aggregation phrase
   In order to get consensus degree among experts, fuzzy preference of individual expert will be aggregated to form a fuzzy composite preference relation. There are two aggregating actions in this phrase to get better result. The first action is to aggregate preference from the first three elements in the expert set E, whereas the other action is then to aggregate result of the first action with preference evaluated by average task-executing time. The two actions are described in details as follows.
   The first aggregation: Before the threshold of dynamic weighting adjustment is reached, this action is to aggregate preferences evaluated by CPU clock rate, CPU usage percentage and memory usage percentage by RPGDM aggregating method. It will use OWA operator with identity quantifier $Q(r) = r$ to aggregate preferences. That means equal weighting $[1/3, 1/3, 1/3]$ will be initially applied to aggregation.
   If threshold is reached, dynamic weighting adjustment strategy will be called to calculate weights for those three indexes. Then, the following formula will be used to aggregate preferences from indexes.

   $$F(P_{ij}^1, P_{ij}^2, \ldots, P_{ij}^m) = \sum_{k=1}^{m} w_k \cdot p_{ij}^k,$$

   $$W = \{w_1, w_2, \ldots, w_m\}, \quad w_i \in [0,1], \quad \sum_{i=1}^{m} w_i = 1$$

   where p_{ij}^k means fuzzy preference relation given by expert e_k preferring alternative a_i over a_j.
   The second aggregation: The OWA operator with identity quantifier will be used to aggregate result of the first action with preference evaluated by average task-executing time.
4. Exploitation phrase
   This phrase will rank computing nodes by QGNDD and QGDD methods in which formula (3) and (4) correspondingly are used to obtain QGNDD and QGDD values. In general, a node with higher value is better to assign more tasks. Therefore, while a node with the lowest value will be assigned one task, the other nodes will be assigned as many tasks as multiple of the lowest value.

4.3. Task reassignment strategy

In Stallings (2003), task moving strategy is proposed to move tasks from high-loading nodes to low-loading nodes

in order to prevent nodes overwhelmed. However, if the number of tasks to move is far beyond loads which a destination node can afford, thrashing node will activate another movement to lower loads of that node to make overpowering condition worse. Hence, task reassignment strategy is needed to address this issue. The purpose of task reassignment strategy is to make whole project finished as soon as possible by allocating a running task to another node with less average executing time. It will not stop running a task in one node but try to running it in another node.

4.4. Dynamic weighting adjustment strategy

This strategy is implemented during the first aggregation of task assigning strategy when threshold of dynamic weighting is reached. At that time, weighed average operation but not OWA is used to sum up preference of each expert. Before using this strategy, weighting to each expert will be decided by coefficient of correlation and coefficient of determination.

While all the tasks have been processed and replied to the requester, the whole processes are completed.

5. Implementation and performance analysis

5.1. Implementation

In this study, CC-FGDM will be implemented on CPS architecture. Fig. 4 depicts relationship among five layers of CPS after adding CC-FGDM with QOS mechanisms which are marked by bolder lines.

1. "Generate QoS" module in computing power-sharing layer
   "Generate QoS" module built into client program in computing node is in charge of executing information retrieval strategy to report the latest load of a node. The report enveloped by XML protocol is sent out per 2 min.
2. QoS message in communication layer
   QoS message is the loading information retrieved by "Generate QoS" module. This message marked up by XML is shown in Fig. 5. In addition to report the load information, this message also serves as the heartbeat message to notify the computation requester that the node is alive and available.
3. "QoS information" data table in discovery layer
   A new data table "QoS information" is added into database in coordinating unit to record QoS message from computing node. "Computation coordination" module will use this data table to execute task assignment strategy.
4. "Computation coordination" module in computing power-sharing layer
   "Computation coordination" module is responsible for executing task assignment, task reassignment and dynamic weighting adjustment strategy of CC-FGDM. The threshold of activating dynamic weighting is default to 30 in the experiment. "Computation coordination" module will use the fuzzy group decision-making method in Section 4.2 to rank alternatives and assign tasks.

5.2. Performance analysis

5.2.1. Experimental procedures and environment

In order to test the performance of CC-FGDM, a digital watermarking algorithm with wavelet filter bank selection is performed (Tsai, 2004). Discrete wavelet transform (DWT) based watermark algorithm makes use of filters to filtrate and construct the signals of a digital image. Among filters, analysis filters are used for distinguishing between the low frequency signals and the high frequency signals in a digital image; synthesis filters are used for constructing image based on the low frequency signals and the high frequency signals. Those filters are selected by the following formulas:

$$h0(-z)g0(z) + h1(-z)g1(z) = 0,$$
$$h0(z)g0(z) + h1(z)g1(z) = 2.$$

The decomposition structure is shown as Fig. 6. After the signals of a digital image with size of m by n pass through the two filters, $h0$ and $h1$, the low and high frequency will be distinguished. Hence, the size of two frequency signals must do downsizing for keeping the size in m by n. In Fig. 7, on the left side, it analyzes the image by rows; and on the right side, it analyzes the image by columns. Through process of decomposition, the image will be restructured as Fig. 7. In general, the decomposition process will be repeated for four or five times by using different analysis filters.

The DWT-based digital watermark algorithm consists of decomposition, embedding, reconstruction, and detection procedures (Cox, Kilian, Leighton, & Shamoon, 1997; Tsai, Wang, Yang, & Yang, 2006). 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.

In Fig. 8, it is a decomposition example of Lena image using DWT. The watermark was embedded in the band 9 as circled area. Because filter is a key component of DWT-based digital watermark algorithm, large volume of computing power is required to search the best filter among lots of filter groups. Therefore, this filter bank selection experiment is an appropriate case implementation for CC-FGDM in CPS.

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

Since CPS is aimed to provide computing resources by using existing computers in the trusty intranet, experiments

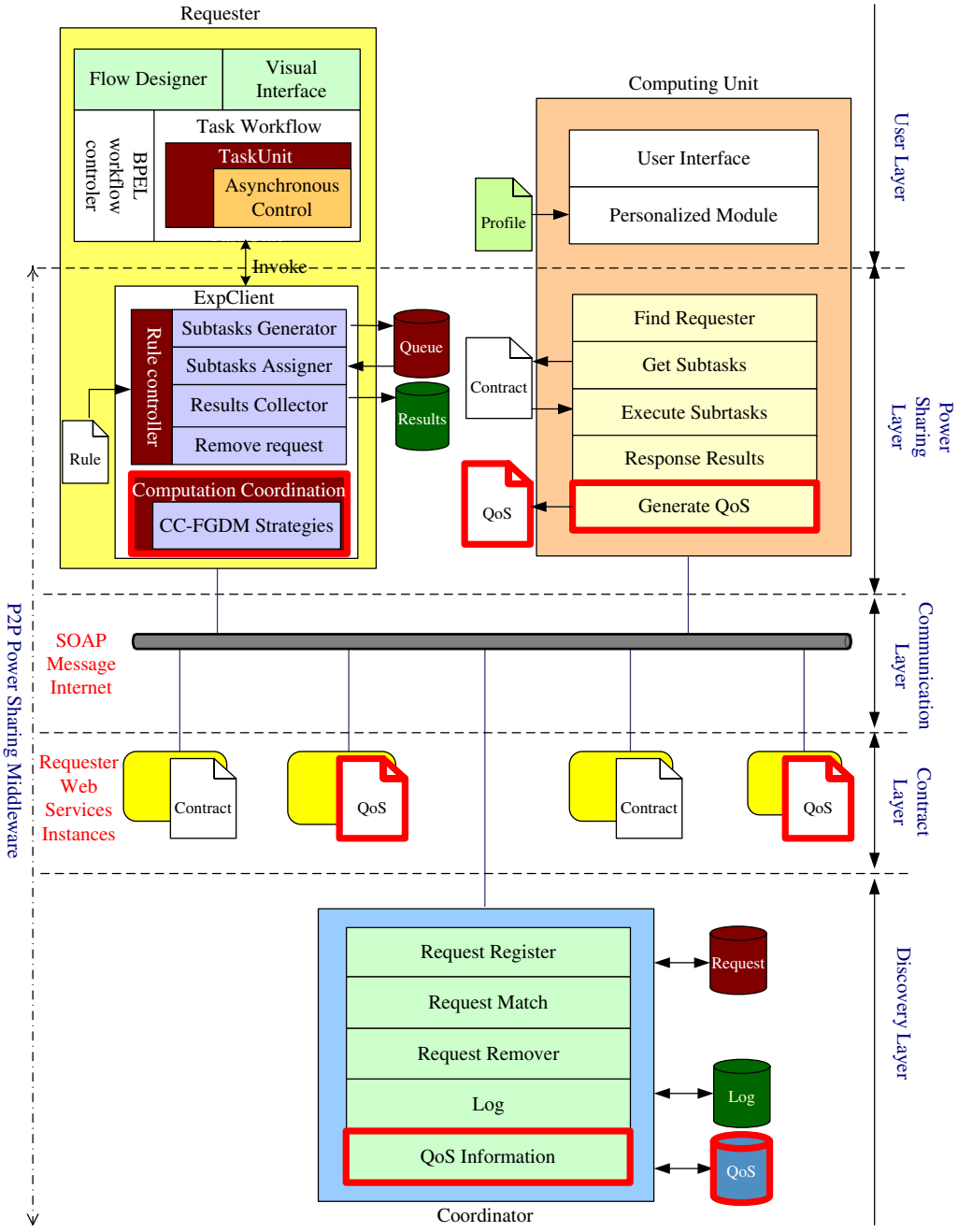

Fig. 4. CPS architecture with CC-FGDM.

are designed to use five computers with different hardware specification as shown in Table 1. All experiments will use the same computer nodes under different condition to participate in computation environment.

Results of executing experiment by four coordinating algorithms whose meanings are shown in Table 2 will be compared and analyzed in the following section. Meanwhile, CPS with fuzzy coordination mechanism will be implemented with fixed weighting and dynamic weighting, respectively, by using CC-FGDM in this paper.

CPS with WFCFS Coordination uses WRRS algorithm (Wang, Doherty, & Dyck, 2002; Wiki, 2006; Zheng, Shu, & Chen, 2005) which is an algorithm to solve an issue of nodes on varied hardware platforms. It assigns tasks according to computing capability by giving different weighting. The more capability a node has, the more weighting it is given and tasks is assigned in each service round. At first, initial weighting IW(N_i) is set by hardware capability. During the runtime, initial weighting will be adjusted according to loading of node which is represented by the following parameters:

- Usage rate of CPU: Cpu(N_i)%.
- Usage rate of memory: Memory(N_i).

```
<QoS>
  <ID>
    <IP>140.113.72.170</IP>
  </ID>
  <CPU>
    <Usage>60(%)</Usage>
    <Speed>3200(MHz)</Speed>
  </CPU>
  <Memory>
    <Usage>50(%)</Usage>
    <Physical>512(MB)</Physical>
    <Virtual>768(MB)</Virtual>
  </Memory>
  <Network>
    <RoundTripTime>8(ms)</RoundTripTime>
  </Network>
</QoS>
```

Fig. 5. XML message format.

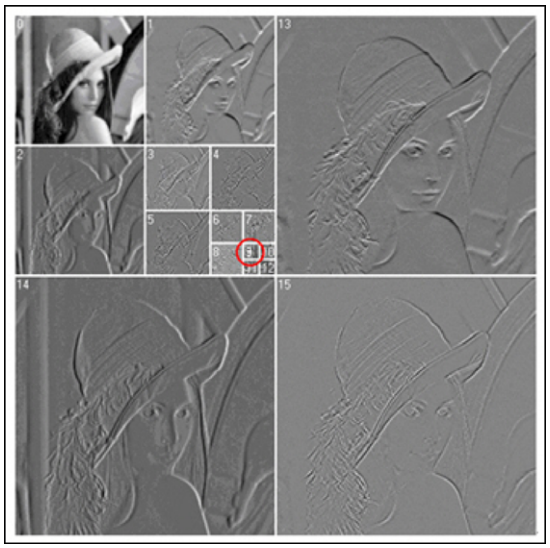

Fig. 8. An example of decomposition using DWT.

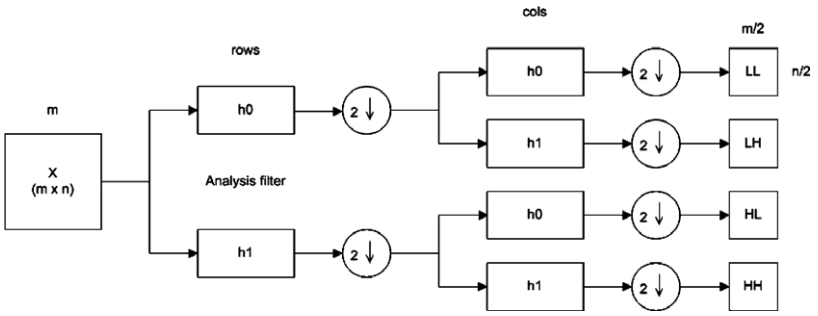

Fig. 6. The decomposition structure.

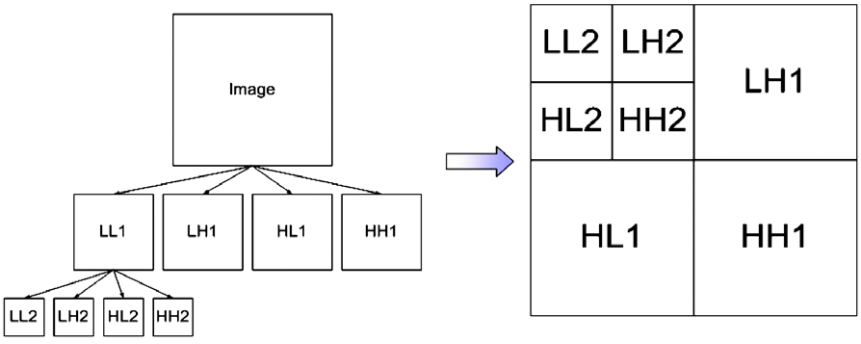

Fig. 7. The pyramid of DWT based watermark.

- Current network flow: $T(N_i)$.
- Access rate of the disk I/O: $\mathrm{Io}(N_i)\%$.
- Response time: $\mathrm{Rt}(N_i)$.
- Processing amount: $\mathrm{Pr}(N_i)$.

The loading of node $\mathrm{Load}(N_i)$ is evaluated according to these items of information by formula as follows. π_i in formula is weighting for different parameters and $\sum_{i=1}^{n}\pi_i = 1$.

Table 1
Hardware specification of five computing nodes in the experiment

Nodes	CPU clock rate (GHz)	Memory (MB)
A	2.4G (Pentium 4)	1024
B	3.2G (Pentium 4)	512
C	1.0G (Pentium 3)	256
D	3.2G (Pentium 4)	512
E	0.67G (Pentium 3)	384

Table 2
Comparison of coordinating mechanism

Coordinating mechanism	Description
CPS w/o coordination	There is no coordinating algorithm in this mechanism. Each computing node is assigned a task each time. After finishing the task, it will request for another task from the requester. This mechanism is similar to first-come-first-served algorithm
CPS w/coordination	A simple coordinating algorithm is used in this mechanism. This algorithm has 180-s threshold to decide whether task reassignment strategy should be activated or not
CPS w/WFCFS coordination	This mechanism uses WRRS (weighted round-robin scheduling) algorithm to assign tasks
CPS w/fuzzy coordination	This algorithm is proposed in CC-FGDM to let requester of CPS dynamically assign task according to load information of computing nodes

$$\begin{aligned}\mathrm{Load}(N_i) = {} & \pi_1 * \mathrm{Cpu}(N_i)\% + \pi_2 * \mathrm{Memory}(N_i) \\ & + \pi_3 * T(N_i) + \pi_4 * \mathrm{Io}(N_i)\% + \pi_5 * \mathrm{Rt}(N_i) \\ & + \pi_6 * \mathrm{Pr}(N_i) \end{aligned} \quad (5)$$

According to load $\mathrm{Load}(N_i)$, the following formula is used to adjust initial weighting to get interim weighting $W(N_i)$

$$W(N_i) = A * \mathrm{IW}(N_i) + B * (\mathrm{Load}(N_i) - \mathrm{IW}(N_i))1/4 \quad (6)$$

By using formula (5) and (6) to adjust weighting dynamically, number of tasks to assign is decided according to weighting. For example, if there are three nodes named A, B, C with weighting 4, 3, 2, respectively, the number of tasks assigned to node A, B, C will be 4, 3, 2 in round-robin order AABABCABC.

5.2.2. Experiment I

The purpose of this experiment is to observe how tasks are assigned according to loading of computing nodes by adjusting weight of performance parameters. The only mechanism of CPS w/fuzzy coordination by dynamic weighting is used in this experiment to justify the object.

There are 762 tasks assigned to computing nodes in which SuperPI (2005) program is running randomly to simulate as a normal work load in the experiment. SuprerPI consumes the CPU usage at a random ratio which could take up to 100% of the usage as the maximum ratio. In order to observe how tasks are assigned, computing node D in Table 1 is selected to monitor the operation of assignment and weight-adjustment. The experimental results are plotted in Fig. 9 which depicts weighting for system performance parameters that are CPU clock rate, CPU usage rate and memory usage rate in the experiment. Fig. 10 shows the relation between loading of computing node D and quantity of assigned tasks.

Since DWT-based algorithm is a CPU-intensive task, it is demonstrated that the performance parameters of CPU clock rate and CPU usage rate are given more weight in

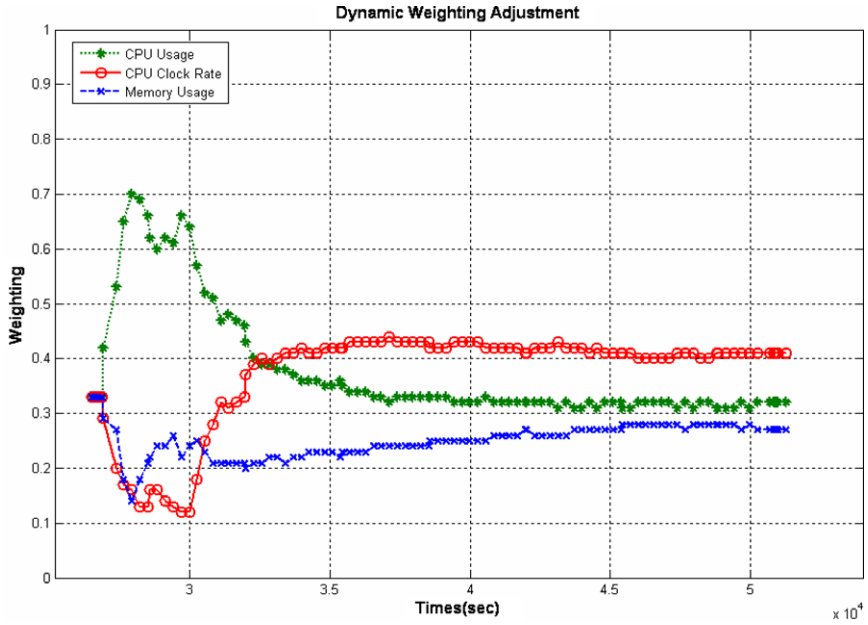

Fig. 9. The weighting plots of system performance parameters.

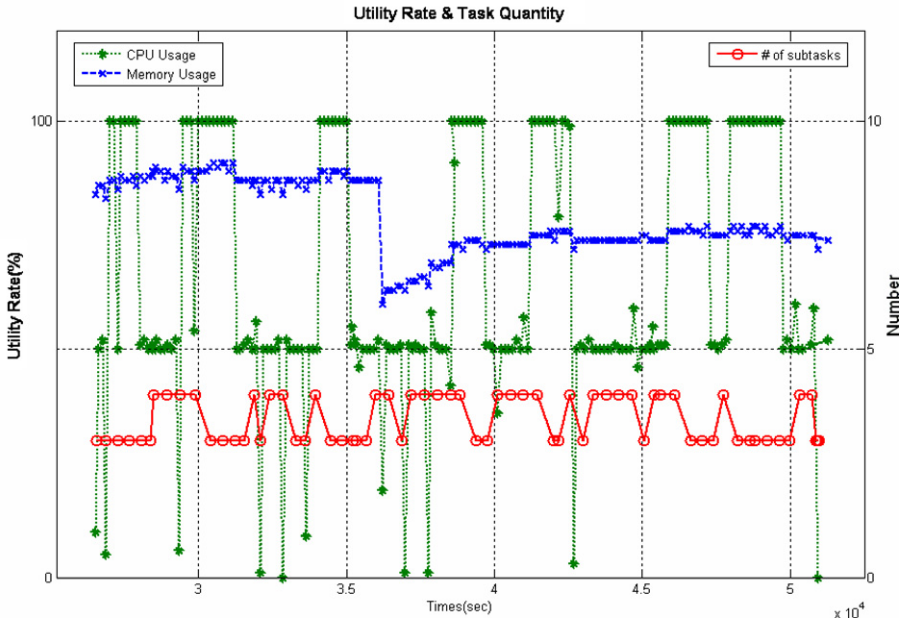

Fig. 10. The relation between loading of computing node D and quantity of assigned tasks.

Fig. 9. That is to say these values and quantity of tasks can be recorded and graphed to demonstrate their correlation. To further explore their relationship, CPU usage ratio and memory usage ratio are shown in Fig. 10 and it is true that the quantity of assigned tasks will be decreased if CPU usage ratio is close to 100% when SuperPI program is executed in computing node. This illustrates that the proposed coordination mechanism is capable of dynamically adjusting quantity of tasks according to the loading of computing node.

In addition, the BEPL capability can be demonstrated in Fig. 11 which shows concurrent parallel processing. The requester can use the visualized workflow control to monitor the whole processes. It is a big advantage for the information management.

5.2.3. Experiment II

Experiment II is to compare the performance of different coordination schemes mentioned in Table 2. When CPS w/Fuzzy Coordination is used, two weighting schemes are compared, one is the fixed weighting of $[1/3, 1/3, 1/3]$ and the other one is the dynamic weighting. It is assumed here that all computing nodes are dedicated to perform the assigned tasks in Experiment II. The experiment is composed of four sub-experiments in which there are 50, 100, 200, and 300 tasks, respectively, needed to be assigned for execution. Each sub-experiment will be run three times for getting average results in Fig. 12. The difference of execution time and system performance is tabulated in Table 3.

Form Fig. 12 and Table 3, the proposed method of CPS w/fuzzy coordination by dynamic weighting does not have too much advantage in execution time over other methods except in 100-tasks sub-experiment. The reason to have just little advantage is that when computing node is dedicated to computation, execution time is more dependent on hardware features like CPU clock rate. Therefore, coordination mechanism specialized in allocating resource does not perform well since it costs extra expense in doing the computation before assigning the tasks.

5.2.4. Experiment III

The objective of Experiment III is also to compare the performance among different coordination methods except SuperPI program will be executed randomly in computing node. The purpose to use SuperPI is to simulate the normal working situation of computing nodes which share their extra resource for CPS. Same condition in Experiment II is applied here and Fig. 13 is depicting the graph of execution time vs. number of task for different coordinating methods, whereas Table 4 is showing the execution time comparison between CPS w/fuzzy coordination by dynamic weight with other four coordination methods.

Unlike in Experiment II, proposed coordinating method has achieved better performance in execution time than other methods. Over 10% of improvement in execution time is gotten when proposed method is compared with method without using coordination or method with WFCFS algorithm. Since SuperPI program is executed randomly as a background job, execution time will be more dependent on coordinating capability of methods to deal with the uncertainty of loading by SuperPI. The experiment results demonstrate CC-FGDM by dynamic weight has the best coordinating capability among other methods.

5.2.5. Experiment IV

Except execution time, the stability of distributed-computing architecture is also important because it can provide the range of the expected execution time for users. Therefore, this experiment is implemented to compare stability

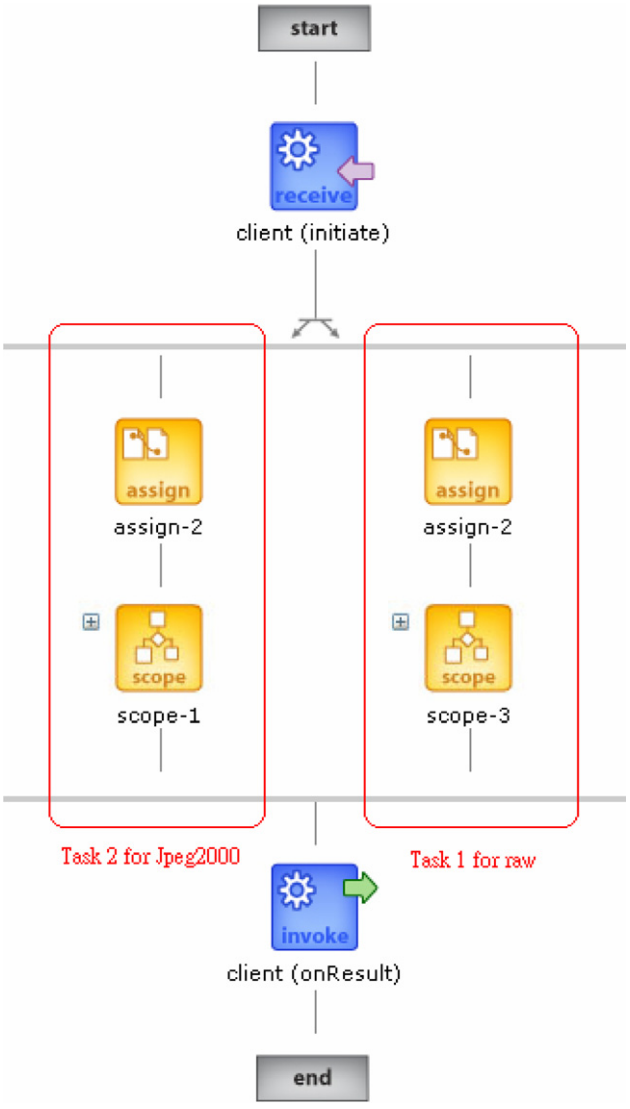

Fig. 11. The visualized work flow control by using BPEL.

of different coordinating schemes. There are 300 tasks assigned for executing 20 times in this experiment. Besides, CPS w/fuzzy coordination method will be implemented by using fixed weight [1/3, 1/3, 1/3] and dynamic weight in order to compare the stability under different weighting condition. While the average and standard deviation of execution time of experiments are tabulated in Table 5, Fig. 14 displays the detailed execution time for comparison. According to standard deviation, CPS w/fuzzy coordination method by fixed or dynamic weighting has smaller deviation than other two methods.

In addition, it is observed that CPS w/fuzzy coordination method by dynamic weighting has better stability because its weighting is decided by real-time execution data which is the up-to-date loading information from computing nodes.

5.2.6. Analysis

After scrutinizing results of above experiments, it is found that proposed CC-FGDM with dynamic weighting has better performance and is more stable than other mechanism no matter whether experiment is executed in the dedicated computing or disturbed-computing environment under SuperPI.

In addition, the architecture provides mechanism for dispatching computational job requests through a central coordinator. The scheduling of the jobs listed in the coordinator is done by CC-FGDM. The scalability of the system with multiple brokers or coordinators is possible since the layered structure of web services. The requester can assign the sub-tasks to different computing units while it keeps track the availability with the computing units. CC-FGDM allows the requester to re-assign the tasks for different computing units thorough the contact establishment. Under such approach, CPS could be easily extended for multiple coordinators and multiple requesters.

6. Discussion and conclusion

6.1. Discussion

Impact of coordinating mechanism to requester: CC-FGDM goes through fuzzy transformation, aggregation and exploitation phrases to get the best alternative according to preferences over alternatives from experts. If number of experts is m and number of alternatives is n, number of comparison in each phrase will be $mn(n-1)$, $mn(n-1)$ and $n(n-1)$, respectively. By using representation of time complexity will be $O(mn^2)$. Because set of alternative is defined as available nodes, the more nodes are involved, the longer executing time is. When clock rate of CPU and one comparative operation is assumed to 2.0 GHz and two clock time, time to finish one comparative operation will be 1 ns. Therefore, when 10^5 nodes are ranked, several 10 s are necessary. Comparison with 180 s on average when one task is executed in DWT-based operation, executing time for coordinating 10^5 nodes is very apparent. However, CPS is supposed to be run in a trusty network of SME where there are generally <10,000 nodes, so the time consumed to coordination will be several 10 ms which are really short compared to 180 s.

Impact of coordinating mechanism to coordinator: XML QoS message is used to report loading information from computing node to requester routinely in CC-FGDM. Although such information is important to coordination, whether it will impact operation in computing node is necessary to discuss from system and network aspect.

From system aspect, loading parameters such as clock rate of CPU, CPU usage and memory usage in XML QoS message are directly came from performance indexes of system. They can be easily gotten by one memory access instruction. As for executing time, it is necessary to record start and end time to get executing duration, two memory access instructions are needed to do subtraction operation which has little impact to system.

From the network point of view, one XML message as shown in Fig. 5 requires only 256 bytes and is sent out

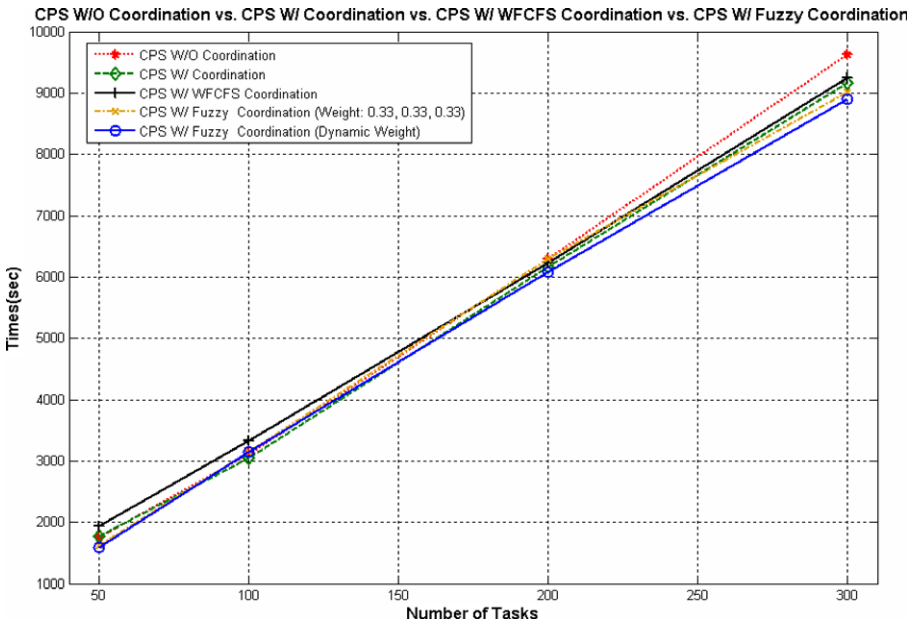

Fig. 12. Execution time vs. number of tasks for different coordinating methods.

Table 3
Difference of execution time and performance for different coordinating methods

Quantity of tasks	Difference (s) and increased (%)							
	CPS w/fuzzy coordination (1/3, 1/3, 1/3) vs. CPS w/fuzzy coordination (dynamic weight)		CPS w/WFCFS coordination vs. CPS w/fuzzy coordination (dynamic weight)		CPS w/coordination vs. CPS w/fuzzy coordination (dynamic weight)		CPS w/o coordination vs. CPS w/fuzzy coordination (dynamic weight)	
	Difference	Increased (%)	Difference	Increased (%)	Difference	Increased (%)	Difference	Increased (%)
50	35.3	3	346.7	22	181	12	163.3	11
100	9.3	0.2	193	6	−90	−3	−34	−2
200	213.3	4	138.3	2	72.7	2	216.7	4
300	116.7	2	343.7	4	263.7	3	737	9

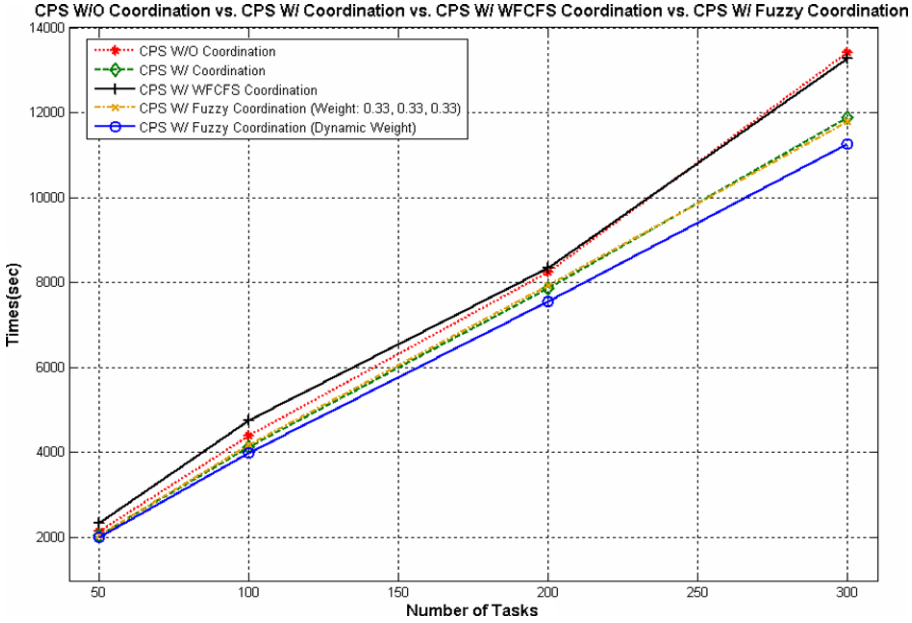

Fig. 13. Execution time vs. number of tasks for different coordinating methods when SuperPI is implemented.

Table 4
Difference of execution time and performance for different coordinating methods

Quantity of tasks	Difference (s) and increased (%)							
	CPS w/fuzzy coordination (1/3, 1/3, 1/3) vs. CPS w/fuzzy coordination (dynamic weight)		CPS w/WFCFS coordination vs. CPS w/fuzzy coordination (dynamic weight)		CPS w/coordination vs. CPS w/fuzzy coordination (dynamic weight)		CPS w/o coordination vs. CPS w/fuzzy coordination (dynamic weight)	
	Difference	Increased (%)	Difference	Increased (%)	Difference	Increased (%)	Difference	Increased (%)
50	48.3	3	335	17	37	2	132.3	7
100	185	5	770.3	20	207.3	6	394.3	10
200	378.3	5	779	11	294	4	678.3	10
300	526	5	2001	18	615.3	6	2162.7	20

Table 5
Statistic comparison of execution time

Statistic	CPS w/o coordination	CPS w/coordination	CPS w/ WFCFS	CPS w/fuzzy (1/3, 1/3, 1/3)	CPS w/fuzzy (dynamic weight)
Average	12,722.1	12,110.65	13,508	11,992	11,538
Standard deviation	1385.8	578.4	885.82	375.4	243.2

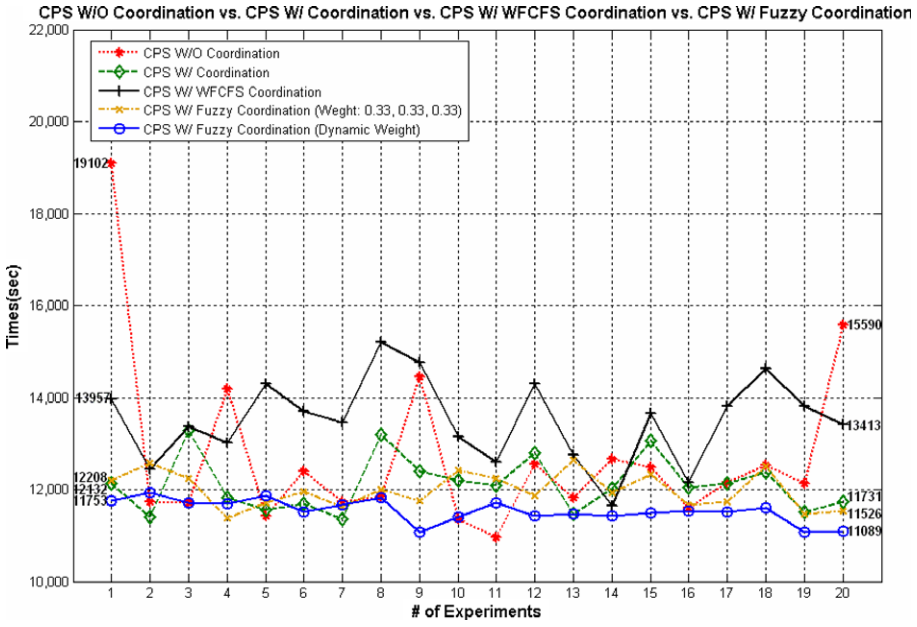

Fig. 14. Execution time of tasks assigned for running 20 times.

per 2 min, it does not use too much bandwidth when DWT-based computation does not return large size of result. However, if a task indeed returns large size of result, a steady and route message like XML QoS message will disturb each other. Moreover, it will make requester think computing node is not available (busy). Since CC-FGDM will be executed only when computing node has returned result and its queue is empty, this means loading information is just needed when coordination is underway without considering heartbeat function of XML message. Hence, when bandwidth is shrinking, length of time to send XML message can be prolonged to alleviate network congestion.

Impact of weighting to performance and stability: RPGDM used in CC-FGDM depends on OWA to aggregate preferences over alternatives to prevent extreme preference of one expert. Therefore, selection of OWA operator is very important in RPGDM. However, system performance indexes are defined as experts in this research which are rational and impartial, so weight for each expert can be dynamically calculated by coefficient of correlation and determination to improve the overall executing performance and stability. This mechanism is justified by the experimental results.

6.2. Conclusion

The mechanism of computing coordination-based on fuzzy group decision-making (CC-FGDM) is proposed in this paper. It integrates the resolution process of group

decision-making (RPGDM) to coordinate resource in sharing environment within a trusty network. CC-FGDM is implemented on web-service based computing power-sharing architecture with the capability to handle the real-time load balance and dispatching with quality of service. The experimental results illustrate that executing time and stability of proposed mechanism improve the overall performance in the disturbed computing environment. Therefore, CPS with CC-FGDM is applicable in enterprise network to efficiently utilize available computing nodes to reduce total executing time and provide more stability for computation-intensive tasks.

Acknowledgements

This work was supported by the National Science Council in Taiwan, Republic of China, under Grants NSC94-2416-H009-018 and NSC95-2416-H009-027. The authors like to thank Chi-Le Chen and Po-Yu Yang for data collection and simulation works.

References

Amnuaykanjanasin, P., & Nupairoj, N. (2005). The BPEL orchestrating framework for secured grid services. *International Conference on Information Technology: Coding and Computing, 1*, 348–353.

Chiclana, F., Herrera, F., & Herrera-Viedma, E. (1998). Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations. *Fuzzy Sets and Systems, 97*(1), 33–48.

Chiclana, F., Herrera, F., & Herrera-Viedma, E. (2001). Integrating multiplicative preference relations in a multipurpose decision-making model based on fuzzy preference relations. *Fuzzy Sets and Systems, 122*(2), 277–291.

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.

Kacprzyk, J. (1986). Group decision-making with a fuzzy linguistic majority. *Fuzzy Sets and Systems, 18*(2), 105–118.

Loo, A. W. (2003). The future of peer-to-peer computing. *Communications of the ACM, 46*(9), 57–61.

Null, L., & Lobur, J. (2003). *The essentials of computer organization and architecture*. Sudbury, MA: Jones & Bartlett Publishers, Inc.

Saaty, T. L. (1980). *The analytic hierarchy process*. New York: McGraw-Hill.

Stallings, William (2003). *Computer organization and architecture: Designing for performance* (6th ed.). NJ, Prentice Hall: Upper Saddle River.

SuperPI, (2005). <http://www.super-computing.org/> Accessed December, 2005.

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., Wang, C. S., Yang, P. Y., & Yang, C. Y. (2006). A collaborated computing system by web services based P2P architecture. *Lecture Notes in Computer Science, 3865*, 194–204.

Wang, Y., Doherty, J. F., & Dyck, R. E. (2002). A wavelet-based watermarking algorithm for ownership verification of digital images. *IEEE Transactions on Image Processing, 11*(2), 77–88.

Wiki (2006). Weighted round-robin scheduling. <http://kb.linuxvirtual-server.org/wiki/Weighted_Round-Robin_Scheduling> Accessed June 2006.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems Man and Cybernetics, 18*(1), 183–190.

Yan, Jun, Yang, Yun, & Raikundalia, G. K. (2006). SwinDeW-a p2p-based decentralized workflow management system. *IEEE Transactions on Systems, Man and Cybernetics, 36*(5), 922–935.

Zadeh, L. A. (1983). A computational approach to fuzzy quantifiers in natural languages. *Computing and Mathematics with Applications, 9*(1), 149–184.

Zheng, Shijue, Shu, Wanneng, & Chen, Guangdong (2005). A load balanced method based on campus grid. *IEEE International Symposium on Communications and Information Technology, 2*, 1516–1519.