An Intelligent Adaptation System based on a Self-growing Engine

Jehwan Oh, Seunghwa Lee and Eunseok Lee

School of Information and Communication Engineering, Sungkyunkwan University
300 Chunchun jangahn Suwon, 440-746, Korea
{hide7674, jbmania, eslee}@selab.skku.ac.kr

Abstract. In this paper, a self-growing engine based adaptation system, which automatically decides the more efficiency plan about the assigning of jobs, in a mobile, grid computing environment, is proposed. Recently, research relating to grid computing has become an important issue, achieving certain goals by sharing the idle resources of computing devices, and overcoming various constraints of the mobile computing environment. In this domain, most existing research assigns work only by considering the status of resources. Hence the situation of assigning work to a peer having relatively low work efficiency, is possible. The proposed system considers various contexts and selects the most suitable peer. In addition, the system stores the history of the work result, and if the same request occurs in the future, a peer is selected by analyzing the history. In this paper, a prototype used to evaluate the proposed system is implemented, and the effectiveness of the system is confirmed through two experiments.

1 Introduction

In modern society, the handheld device is quickly becoming widespread, its usage and explosive growth can be attributed to the rapid development of wireless network technologies. The computing power of these handheld devices is continually increasing, relative to the growth of associated technologies. However, handheld devices have limited capacity because of a potable feature. For this reason, applications for handheld devices to some extent, are limited, and developers must create applications offering limited services. The expectations of users are increasing, demanding the same the quality of service as the desktop computer. Grid computing technologies, creating distributed processing, operating over a number of low capacity computers, can perform large amounts of work, which is difficult to process in an individual computing device. This technology overcomes the various constraints of handheld devices, and creates large scale applications over a wireless environment.

In grid computing, the allocation of work to the various devices is a significant problem. Most existing research adopts a method of allocating the work only by considering the resource status of the participating peers. Therefore, problems could arise of assigning work to peers, which have relatively low work efficiency. For instance, the processing time of a peer which has the byte code for work, even if the peer has insufficient resources, may still be faster than the processing time of a peer which has sufficient resources, but does not have the byte code.

To cope with this problem, the proposed system selects the most suitable peer through considering current variable context (resource usage pattern, available resource, network bandwidth, existence of the byte code), therefore more efficient distribution of work is made possible. In addition, the proposed system monitors the processing results for work and stores these values in the history DB. Then, the system computes the average values for each factor and retrieves the more significant context factor. The system assigns the weight value to the selected factor, selects the peer with the most similar context values, and average values among the current participating peers, assigning the work to the selected peer. Next, the results better than average values are stored in the history DB. The system has self learning feature, selecting the more suitable host according to the time passed.

In this paper, a prototype is implemented. In order to evaluate the proposed system, the effectiveness of the system is confirmed through two experiments.

The paper is organized as follows: Section 2 introduces related work. Section 3 describes the comprehensive structure and the behavior process of the proposed system. In Section 4, details of the operational process and algorithm are presented. System evaluation through the implementation of a prototype is described in Section 5. Finally, Section 6 concludes the paper and identifies future work.

2 Related Work

Research overcoming the various constraints has been studied in various laboratories. Firstly, research relating to adaptation, which is adjusting the quality of service, and parameters of the application or system components, has been actively processed [1][2][3][4]. However, this method focuses on the quality of service offered in the past.

Grid computing is becoming an important issue, achieving various goals by sharing idle resources of diverse computing devices. The system can perform large amounts of work, which is difficult to process for individual computing devices. Representative studies include Globus [7] and Condor [8]. However, these studies focus on the wired environment and therefore are not suitable for the mobile environment. Studies dealing with the problems of the mobile environment include the following; Xiaohui Gu *et al.* [6] proposes the Adaptive offloading System, performing distributed processing in a surrogate computer, having relatively rich memory to overcome the limited memory available in mobile devices. Another study, Junseok Hwang *et al.*[10]proposes middleware based on a mobile proxy structure that can execute jobs submitted to mobile devices, in-effect making a grid consisting of mobile devices. However, these systems only consider the available resources of each peer when allocating the jobs. Thus, the possibility of work being assigned to a peer with relatively low work efficiency, exists.

In this paper, an intelligent system is proposed, this system is based on resource usage patterns, existence of the byte code, as well as available resources, and has a self learning feature.

In the subsequent section, the proposed system applying these concepts, is described in detail.

3 Proposed System

3.1 System Framework

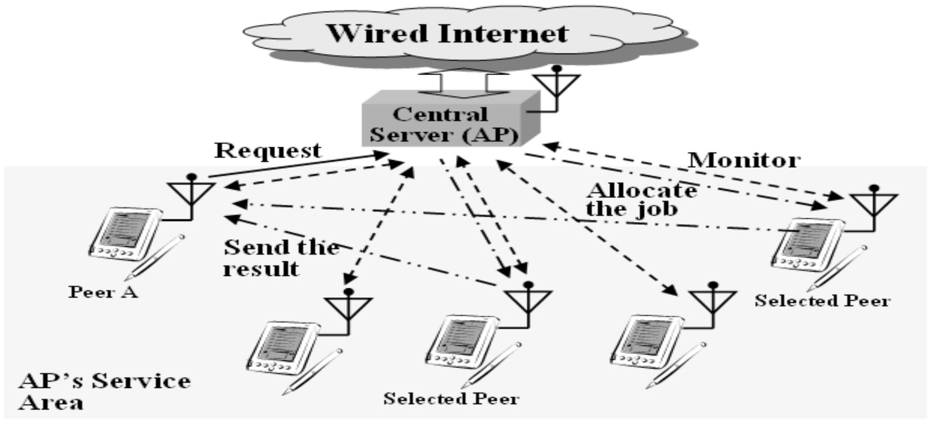

Fig. 1. Overall Architecture of Proposed System

This paper proposes the intelligent adaptation system, which continually improves the method of allocating jobs, by using a self growing engine, in order to efficiently allocate jobs. As presented in Fig. 1, the central server monitors/stores the context information of peers. If peer A requests services from the server, then the server selects the peers which can perform the requested services according to various computation processing, allocating the jobs to selected peers. In addition, the system stores the history of the work result, and if the identical request is issued in the future, it selects the most suitable peer by analyzing the history.

3.2 System Components

As presented in Fig.2 the component of the proposed system is composed of two parts; the Client Module (CM), which is embedded in each peer, and the Server Module (SM), which is embedded in the central server. The SM is located at or close to a wireless network access point. The role of each principal component is as follows.

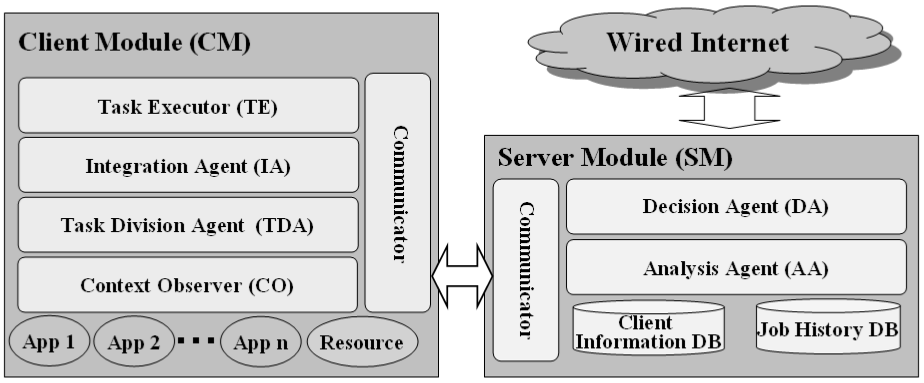

Fig. 2. Components of the Proposed System

- *Context Observer (CO):* gathers the variety context information including resource usage pattern and transmits this information to the *SM*
- *Task Division Agent (TDA):* divides the self-executing jobs , entrusted depending on gathered context
- *Integration Agent (IA)*: integrates the received job results from the external agent, and delivers the integrated results to the application
- *Task Executor (TE)*: executes the jobs required from an external peer
- *Communicator (client)*: performs the interaction with the *SM* or peers
- *Communicator (sever)*: performs the interaction with the *CM* in each peer
- *Analysis Agent (AA):* selects the job history DB corresponding to the requested job from the client from the selected job history DB, and computes the average value of a client's condition values when a client performs the each job. In addition, the *AA* stores the context information received from each client, in the Client Information DB. The AA then analyzes the results of the executed job by the selected peer, and updates the job history DB with the result.
- *Decision Agent (DA):* decides peers to be allocated based on various computations and on the peer's condition analyzed by the *AA*
- *Client Information DB*: The location where various context information (resource usage pattern, available resource, network bandwidth, existence of the byte code) of each participating peer is stored. This information is periodically updated.
- *Job history DB*: The location where the executed result each service is stored.

3.3 Overall System Behavior

A sequence diagram corresponding to the overall system behavior is presented in Fig.3

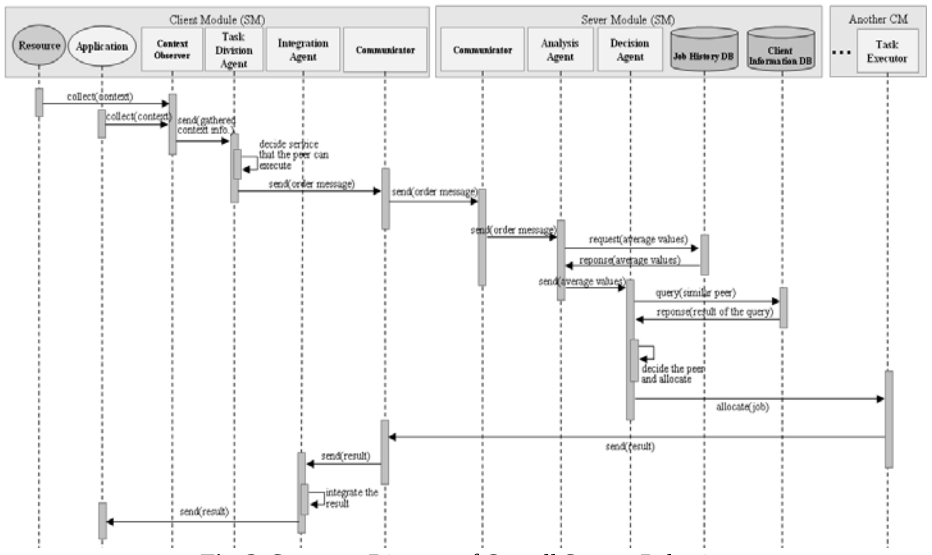

Fig. 3. Sequence Diagram of Overall System Behavior

Firstly, when a user connects to the server, the CO continually gathers the current resource status including the resource usage pattern, and transmits this information to the SM. Then, the AA receives this information, and computes the score for this context, storing this score value in the Client Information DB.

Next, when a user executes the application, the TDA decides the jobs to be executed at the current resource status. The jobs, which cannot be executed, are entrusted to the AA via a communicator. The AA retrieves the suitable condition of the peer to execute the Jobs in the Job History DB. The DA finds the peer based-on the retrieved condition of peer AA in Client Information DB. The selected peer is allocated to the jobs. The TE of the selected peer performs the jobs, and then transmits the executed results to the CM requesting the jobs. The IA of the CM receives and integrates the results. Finally, the IA transmits the integrated result to the application.

3.4 Selection Phase of the Peer via Computation

In the case of the server allocating the requested jobs from the client only by considering the resource status of each participating peer, the case of assigning work to a peer which has a relatively low work efficiency, could exist. For instance, the processing time of a peer which has the byte code for the work may have insufficient resources, but achieves results faster than the processing time of a peer having sufficient resources, but does not have the byte code. To cope with these problems, the proposed system selects the most suitable peer through considering more relevant current context as follows:

- The current available (idle) resources of each participating peers such as CPU, RAM, Battery.
- The network bandwidth of each participating peer

- The changing probability of resources computed via the resource usage pattern analysis of each participating peer
- The existence of the byte code, which will be the completed job

This information is collected in the CM, and periodically stored in the Client Information DB. Then each factor is computed to a relative score using the formulas presented in Table 1.

Table 1. The Sample Formulas to Compute the Each Factor

Context type	Formula	Example	Score
Idle resource	$\frac{CPU_{available\ Rate} \times CPU_{clock}}{10000} = CPU_{level}$	$1 - \frac{20 \times 400}{10000} = 0.2$	2
	$\frac{RAM_{freeSpace}}{RAM_{size}} = RAM_{level}$	$\frac{20}{64} = 0.312$	4
	:	:	:
The Changing Probability of Resource	$\frac{(x_1 - m)^2 + (x_2 - m)^2 + \cdots + (x_n - m)^2}{n}$	$1 - \frac{3.26}{3} = 0.92$	10
:	:	:	:

The SM can obtain the final score of each device using a formula as follows. Then, the SM selects the device, which obtained the highest final score, allocating the jobs to the selected device. The each formula is differently created to become range of values from 0.01 to 1.00.

$$CPU_{level} \cdot w_1 + RAM_{level} \cdot w_2 + \cdots + probability \cdot w_n = Final\,Score \tag{1}$$

```
(REQUEST
  :sender(agent-identifier :name CM1@HIDE:1099/)
  :receiver (set(agent-identifier :name SM@selab:1099/))
  :context "(action translator)and
            (action image_compression)
        ...

)
```

```
(INFORM
  :sender(agent-identifier :name CM1@HIDE:1099/)
  :receiver (set(agent-identifier :name SM@selab:1099/))
  :context "(service_name(translator)),(entrusted_device(tester76)),
            (execution_time(3.5)),
        ...

)
```

Fig. 4. Example of ACL message, which is requesting the jobs from the CM to the SM, informing the completion of work from the CM to the SM

3.4 Learning Phase via Self Growing Engine

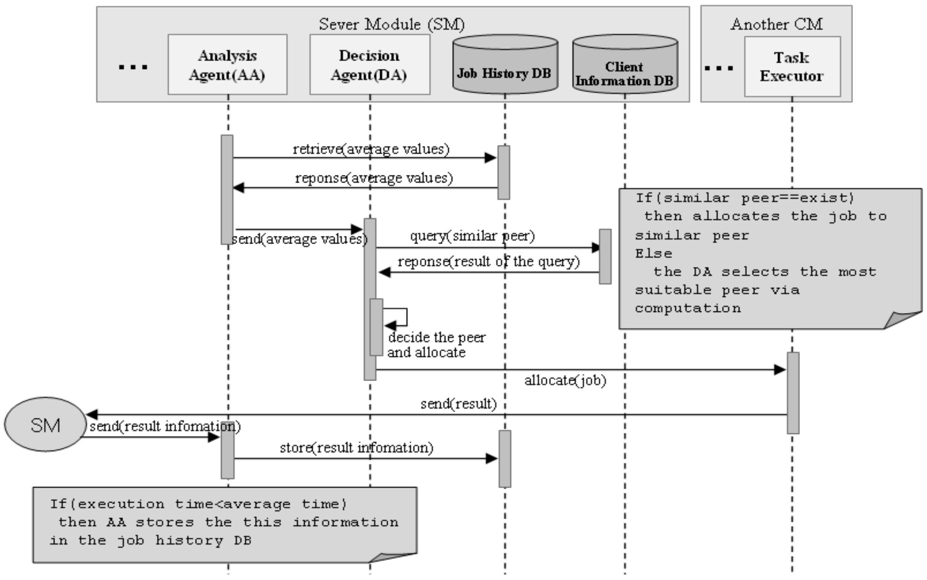

Fig. 5. Sequence Diagram of Learning Phase

The selecting phase of the specific peer is described via computation, in the previous section, when the CM requests the jobs from the SM. In this section, the self learning of the most suitable peer using the history for the job result, is introduced.

When the peer requests a job, the AA retrieves the average value of the performed results in the Job History DB. The AA transmits the retrieving result to the DA. The DA assigns the weight to the element having the highest value among the average values, and finds the peer, which is similar to this value in the Client Information DB. If a similar peer exists, the SM allocates the jobs to the similar peer. However, if a similar peer does not exist, the DA selects the most suitable peer via computation, described in the previous section, and the selected peer is allocated the jobs. The TE of the selected CM performs the jobs, then transmits the executed results to the CM requesting the jobs. The CM receiving the result informs the time for executing work and the condition of the device performing the jobs to the AA. The AA analyzes the received information from the CM. If the processing time is less than the average time, the AA stores this information in the job history DB. The AA re-computes the average value in the job history DB.

Table 2 presents sample result data for the translator service in the Job history DB. The average values mean that the condition is the suitable condition of the peer executing the translator service.

Table 2. Sample of the Result Data for the Translator Service in the Job history DB

Device name	Condition	Execution time	Date
tester 34	7, 5, 4, 1, 4, 8	3.23	20050704
tester 21	4, 4, 5, 2, 3, 4	3.74	20050621
tester 28	3, 9, 4, 1, 4, 7	3.59	20050520
:	:	:	:
Average value	**4.7, 7.3, 6.1, 1.4, 7.2, 3.4**	**3.82**	

4 System Evaluation

Each prototype module for the system evaluation was implemented mainly using Embedded Visual C++. In this experiment, the CM and SM are both implemented on the desktop PC, connected by wired Internet. The CM, is tested using a PDA simulator offered by the Embedded Visual Studio. The executed application is 'Remote Video Medical Treatment' for the E-healthcare scenario, studied in the laboratory. The divided jobs are 'Translator', 'Image Converter', and 'Video Converter' respectively.

The efficiency of the proposed system is confirmed via two experiments. Firstly, the time for processing jobs is compared by considering the resource status, with the time for processing the jobs. The information of 10 peers is stored in the Client Information DB, and requests generating the translator service are generated in a peer.

Consequently, the proposed system selects the peer using factors such as network bandwidth, existence of job byte code, and available resources. The processing time of the peer selected by the proposed system is superior to the processing time of a peer selected when only considering the resource status. The result for this experiment is presented in Table 3.

Table 3. Comparison of Processing Time

	Condition of Selected Peer	Processing Time (s)
The existing method	6, 8, 9	5.6
The proposed method	4, 7, 8, 1, 8, 9	4.3

Secondly, a change in time for processing the job according to the number of times that process the same job increases, is observed. The translator service is executed 100 times using the PDA simulator, and the average of the processing times and the conditions of the PDA simulators is computed. Initially the average values are stored in the Job history DB. The peer repeatedly requests the translator service from the SM.

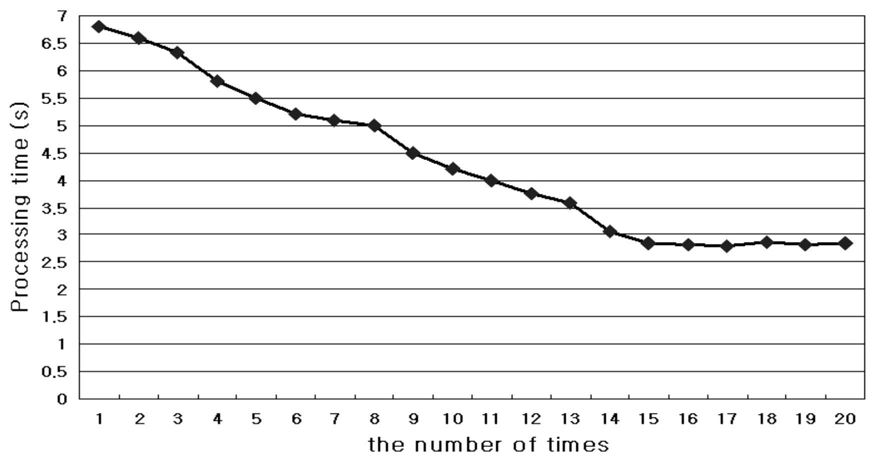

Fig. 6. Decrease of the processing time through a repeat execution of the translator service

Fig. 6 presents the experiment results. When the peer initially requests the translator service, the processing time is similar to the average processing time. Accordingly, as the number of times that the peer executes the translator service increases, the processing time decreases. However, when the peer executes the translator service 15 times, the processing time does not decrease. This result means that the SGE finds the most efficient condition for executing the translator service.

It is confirmed that the proposed system is more efficient than the traditional method of allocating jobs.

5 Conclusion

In this paper, a self learning engine based adaptation system to more efficiently plan and improve the current system, in the mobile grid computing environment, is presented. The efficiency of the proposed system is confirmed via two experiments. The various aspects of the proposed system are expected to solve the constraints discussed in wireless computing, resulting in a more convenient wireless computing environment.

Additionally we see a number of other key future research areas. First is the investiga-tion of more diverse context that may occur in mobile environment. Second is study for more efficient algorithms for gathering and analyzing context. Last is study for more optimal size of history data.

References

1. Alvin T.S. Chan, Siu-Nam Chuang, "MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing", IEEE Transaction on Software Engineering. Vol. 29, No.12 pp.1072-1085, Dec.2003.

2. Brian Noble, “System Support for Mobile, Adaptive Applications”, IEEE Personal Communications Vol.7 No.1, Feb.2000
3. Wai Yip Lum, Francis C. M. Lau, “User-Centric Content Negotiation for Effective Adaptation Service in Mobile Computing”, IEEE Transaction on Software Engineering. Vol. 29, No.12 pp.1100-1111, Dec.2003
4. Vahe Poladian, João Pedro Sousa, David Garlan, Mary Shaw, “Dynamic Configuration of Resource-Aware Services”, 26th International Conference on Software Engineering (ICSE’04), pp.604-613, May.2004
5. Richard S. Sutton, Andrew G. Barto, ‘Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)’, The MIT Press, Mar.1998
6. Xiaohui Gu, Klara Nahrstedt, Alan Messer, Ira Greenberg, Dejan Milojicic, “Adaptive offloading for pervasive computing”, Pervasive Computing, IEEE Volume 3, Issue 3, pp.66-73, July-Sept. 2004
7. http://www.globus.org/
8. http://www.cs.wisc.edu/condor/
9. Gabrielle Allen, David Angulo, Ian Foster, Gerd Lanfermann, Chuang Liu, Thomas Radke, Ed Seidel and John Shalf, “The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment”, Journal of High-Performance Computing Applications, Volume 15, no. 4 2001
10. Junseok Hwang; Aravamudham, P., “Middleware services for P2P computing in wireless grid networks”, Internet Computing, IEEE, Volume 8, Issue 4, pp.40-46, July-Aug. 2004