A Log Analysis System with REST Web Services for Desktop Grids and its Application to Resource Group-based Task Scheduling

Joon-Min Gil* and Mihye Kim**

Abstract—It is important that desktop grids should be able to aggressively deal with the dynamic properties that arise from the volatility and heterogeneity of resources. Therefore, it is required that task scheduling be able to positively consider the execution behavior that is characterized by an individual resource. In this paper, we implement a log analysis system with REST web services, which can analyze the execution behavior by utilizing the actual log data of desktop grid systems. To verify the log analysis system, we conducted simulations and showed that the resource group-based task scheduling, based on the analysis of the execution behavior, offers a faster turnaround time than the existing one even if few resources are used.

Keywords—Desktop Grids, Execution Behavior, Log Analysis System, REST Web Services, Resource Group-based Task Scheduling

1. INTRODUCTION

With the popular use of the Internet and the high-performance of PC resources, it is possible to build a desktop grid environment by binding the unused PC resources that are connected to the Internet [1, 2].

An important aspect in desktop grids is that each resource has a volatility property, due to free withdrawal from execution participation even in the middle of task execution. Moreover, each resource has a heterogeneity property as it has a totally different computing environment (e.g., CPU performance, memory capacity, network speed, etc.) [3]. Due to these two properties, it is not possible to estimate the completion time of all tasks. Therefore, if tasks are allocated without any consideration for the dynamic execution features of resources, execution failures will occur frequently due to the volatility and heterogeneity, and thus the turnaround time for all of the tasks becomes longer. However, execution failures can be prevented by selectively allocating tasks to those resources that are suitable for the current execution environment of desktop grids. As a result, it is necessary to analyze the execution behaviors of resources as a way to provide a stable execution environment for desktop grids.

※ This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2008-331-D00447)

Manuscript received November 2, 2011; accepted November 7, 2011.

Corresponding Author: Joon-Min Gil

* School of Computer and Information Communications Engineering, Catholic University of Daegu, Gyeongsan, Gyeongbuk, S. Korea (jmgil@cu.ac.kr)

** Dept. of Computer Science Education, Catholic University of Daegu, Gyeongsan, Gyeongbuk, S. Korea (mihyekim@cu.ac.kr)

In this paper, we implemented the log analysis system with REST web services that can extract the execution behavior of resources from actual log data in the Korea@Home system. The analysis results of the execution behavior can be used as the task allocation information to minimize the waste of resources and the execution delay in task scheduling. In particular, this study can be very useful as background on deciding the task scheduling policies for various desktop grid systems, as well as for Korea@Home. Moreover, we present Representational State Transfer (REST)-based open APIs that provide an efficient access method for clients. Using the open APIs, a client can easily acquire log analysis results via its web browser.

The remainder of this paper is organized as follows. In Section 2, we provide a brief description of the desktop grids that were used in this paper. In Section 3, we define the execution behavior of resources as availability and credibility. The log analysis system that we implemented is presented in Section 4. This section also presents in detail the modules of the log analysis system. In Section 5, we present the log analysis results and their application to resource group-based task scheduling. Section 6 concludes the paper.

2. DESKTOP GRID ENVIRONMENT

The desktop grid environment assumed in this paper is physically composed of a client (or an application submitter), a central server, and resources. The client is an entity submitting its own application to the central server. The central server takes responsibility for mediating resources and tasks and performs the functions such as task management, resource management, task allocation, etc. Each resource acts as a resource provider and executes the tasks sent by the central server during its idle cycles. Once a task is finished, the task result is sent back to the central server. The detailed steps to execute the parallel tasks that have been submitted by a client are described as follows:

Step 1 (Resource registration): Each resource registers its own information, such as CPU performance, memory capacity, OS type, etc., to the central server.
Step 2 (Application submission): A client submits their own application to the central server.
Step 3 (Task allocation): The central server allocates tasks in the task pool to resources.
Step 4 (Task execution): The resource to which tasks are assigned executes the tasks during its idle cycles. As soon as task execution is finished, it sends the task results back to the central server.
Step 5 (Results collection): The central server collects the task results that were received from resources and records them in the databases.
Step 6 (Application completion): The central server checks if all tasks are completed, and sends the final results to the client.

The steps described above are based on the client-server architecture, which has been typically used in most of the desktop grid systems (e.g., BOINC [4], XtremWeb [5], Korea@Home [6], etc.). The application that is submitted to the central server by clients, is divided into hundreds of thousands of tasks, each of which is small enough to be executed in one resource. In addition, each task is mutually independent without any dependency between each other.

3. EXECUTION BEHAVIOR OF RESOURCES

The performance of desktop grids is largely influenced by the dynamic features such as the execution join and the withdrawal of resources [2, 7]. Due to the task stops that can occur at any time even in the middle of task execution, task failures will be unavoidably encountered. Thus, the availability, that represents how much time each resource can spend executing tasks for a given time period, and the credibility that determines the trustworthiness of the task results in the presence of failures, can be considered as an important factor to enhance the overall performance of desktop grids.

Due to the dynamic features of desktop grids, it is strongly recommended that tasks should be allocated to the resources with high availability and credibility. These resources can return as many as possible credible task results within a given time period. Therefore, in this paper, we classify resources by the two factors of availability and credibility, and we analyzed the execution behavior of resources based on these two factors.

Definition 1. Availability (A): *a probability that can execute tasks in the presence of task failures.*

$$A = \frac{MTTCF}{MTTCF + MTTCR} \tag{1}$$

Definition 2. Credibility (C): *a factor for how many task results can be returned after tasks are allocated to resources.*

$$C = \begin{cases} \frac{r}{n}, & \text{if } n > 0 \\ 0, & \text{if } n = 0 \end{cases} \tag{2}$$

In Equation (1), *MTTCF* (Mean Time To Critical Failure) means the average failure time that is caused by execution failures or resource defects, and *MTTCR* (Mean Time To Correct and Recover) means the average execution time, including recovery time, after a failure occurs [8]. Thus, the availability defined as Equation (1) is calculated as the rate divided execution time by the total time that includes failure time and execution time. In Equation (2), n means the total number of the allocated tasks to resources, and r means the number of task results for the n number of tasks allocated to resources.

In this paper, we will implement the log analysis system that can classify resources by the availability and credibility presented in Definitions 1 and 2.

4. LOG ANALYSIS SYSTEM

4.1 The Execution Flow of Resources

Before implementing the log analysis system, it is important to first understand the execution flow of each resource to capture the dynamic features of desktop grids. Fig. 1 shows the execution flow of each resource in Korea@Home desktop grid system [6]. In this figure, a resource

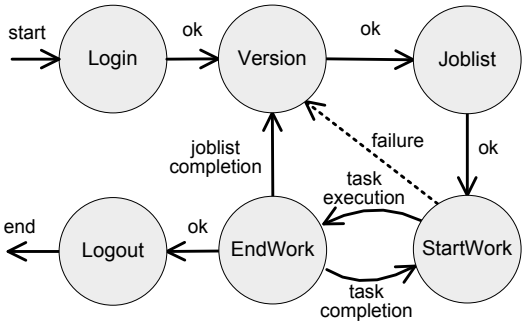


Fig. 1. The execution flow of each resource

requests for a function from the central server by message communications with an XML document and receives all the data and information needed to execute tasks from the central server.

As we can see in Fig. 1, the request-response functions include **Login**, **Version**, **Joblist**, **StartWork**, **EndWork**, and **Logout**, and are essentially required for analyzing the execution behavior of each resource.

- **Login**: this is the first phase to participate in the task execution of desktop grids. In this phase, an agent program installed in each resource performs the authentication function to check the credentials of the resource.
- **Version**: this sends a query to the central server to check if there is a newer agent program than the one already installed in a resource. If so, the update process to reinstall the newer program is performed.
- **Joblist**: this is the procedure for requesting for tasks to be executed by a resource, and the central server provides the job list for the resource as a response. It also has a download function that sends the task program and data files that are needed for task execution from the central server to the resource.
- **StartWork**: this function informs the central server of task execution once a resource is ready to execute a task.
- **EndWork**: this function informs the central server of task completion after a resource finishes task execution. It also uploads task results to the central server.
- **Logout**: this logs out of desktop grids.

The time period between **StartWork** and **EndWork** can be seen as the actual execution time for a task. As long as there is no failure during the time period, a pure execution time is the same as the actual execution time. However, if there are failures, the actual execution time is obtained by adding the pure execution time to the failure time.

Fig. 2 shows an example of XML request messages for **StartWork** and **EndWork**. XML messages of these types are exchanged between a resource and the central server for each request-response. In this paper, we extract the execution behavior of resources by parsing log information in the XML messages and analyze the availability and credibility of resources with the execution behavior.

```
<kath>
    <request>
        <startwork>
            <appid> </appid>
            <workid> </workid>
        </startwork>
    </request>
</kath>
```

(a) StartWork

```
<kath>
    <request>
        <endwork>
            <appid> </appid>
            <workid> </workid>
            <worktime> </worktime>
        </endwork>
    </request>
</kath>
```

(b) EndWork

Fig. 2. Example of XML messages

4.2 Detail Modules for the Log Analysis System

The log analysis system consists of five modules as shown in Fig. 3. The detailed function of each module is described as follows:

- **LogManager**: this receives the primitive log data from a desktop grid system as a type of file and performs preprocessing for the log data.
- **LogParser**: this takes the responsibility of filtering an unnecessary log from the primitive log data so as to only obtain information that is relevant to the execution behavior of resources.
- **XMLParser**: this extracts meaningful information for each log through an XML parsing procedure and captures the detailed information that is relevant to the execution behavior of resources.
- **LogAnalyzer**: this takes responsibility for analyzing actual availability and credibility by utilizing the detailed information extracted by XMLParser. It also takes responsibility for cooperating with the DBManager to store and update the analysis results on the availability and credibility in databases.
- **DBManager**: this stores, updates, and retrieves the availability and credibility information of resources to/from databases.

The log analysis system in this paper has been implemented with JDK 1.6.0, and JDOM 1.1 [9]. It has been used as an XML parser to extract task information from request-response XML messages. Meanwhile, we have used MySQL 5.5 [10] as the database to store and manage all the information that is relevant to the execution behavior of resources, including availability and

Table 1. REST-based open APIs

Name	Description
getAgent(agentid)	Get the information of a specific agent
getAvailability(agentid)	Get the availability of a specific agent
getCredibility(agentid)	Get the credibility of a specific agent
getAgentByTask(taskid)	Get agents that have executed a specific task
getTaskByAgent(agentid)	Get all of the tasks that have been executed by a specific agent
getCompletedTaskList()	Get a list of the completed tasks
getCompletedTaskTime()	Get the total execution time of the completed tasks

Fig. 4. Example of log data fetched by open APIs

credibility.

The log analysis system also supports REST-based open APIs that allow users to access log analysis results more easily via web browsers. The users can simply acquire log analysis results using the open APIs presented in Table 1. These open APIs have been implemented by the Restlet [11] that supports REST web services, which are based on Java. The APIs in Table 1 follow the design principles of a REST web services architecture [12, 13]. Fig. 4 shows an example of the log data fetched by the implemented open APIs in a web browser.

5. LOG ANALYSIS RESULTS AND THEIR APPLICATION TO TASK SCHEDULING

5.1 Log Analysis Results

In this paper, log analysis results have been produced based on the primitive log data of the Korea@Home system, which is the only desktop grid system in Korea. The used log data was collected for one month (Dec. 2007). Fig. 5 shows the log analysis results that were acquired by analyzing this log data. The log analysis results are depicted in this figure as an average execution time in a one-week unit. In this figure, the x- and y-axis represents the time in the unit of an hour and the total execution time by all the resources for one hour, respectively.

Upon analyzing the results of this figure, we can find that execution behavior for a weekday is different from that on the weekend. Although the execution behavior per day during the week is not exactly same as each other, it has almost similar patterns (i.e., the task execution time gradually increases from 9:00 to 12:00, and high execution time is maintained without any sharp fluc-

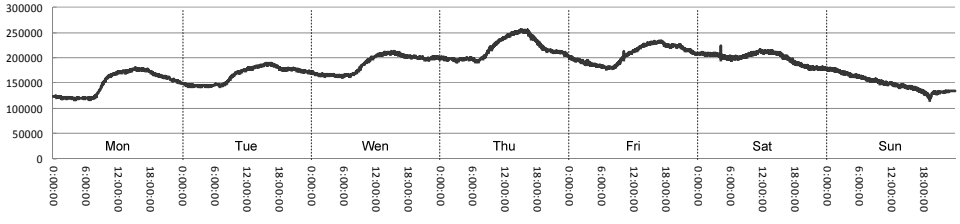


Fig. 5. Average execution time in a one-week unit

tuation between 12:00 and 15:00). The task execution time from 15:00 to 24:00 steadily decreases. The task execution time from 0:00 and 9:00 is almost flat, but comparatively lower than that of other times. In fact, this tendency is due to the working style of normal people. We can also observe from this figure that execution behavior on the weekend is different from that of a weekday (i.e., the task execution time is kept at certain value or decreases steadily regardless of the time for each day). Thus, we can see from this figure that different task scheduling policies should be applied according to the weekday and the weekend.

Meanwhile, the availability and credibility of resources can have an uplifting effect on the performance of task scheduling in desktop grids. For example, resources with relatively high availability and credibility can execute more tasks for a given time period and moreover, can return many more credible task results to the central server. On the contrary, it is highly probable that resources with low availability and credibility will fail to return credible task results within a given time period. Therefore, if tasks are properly allocated according to the availability and credibility, resources will be efficiently utilized due to the reduction of task failures, resulting in a shortened turnaround time for all tasks.

Fig. 6(a) shows the distribution of availability and credibility for all of the 3,390 resources participating in the Korea@Home system during one month (Dec. 2007). During this period, the average availability and credibility of resources were 50.62% and 62.65%, respectively. On analyzing the distribution of availability and credibility in this figure, we can see that all resources are widely scattered and resources with more than average availability and credibility have contributed a lot more to task execution than the others.

5.2 Application to Resource Group-based Task Scheduling

Here, we apply the execution behavior of weekdays and weekends to task scheduling. First, a set of resource groups is constructed with the availability and credibility of resources. Table 2 shows four sets of resource groups, R_1, R_2, R_3, and R_4. We used two kinds of thresholds, availability threshold ($\tilde{A}$) and credibility threshold ($\tilde{C}$) as a criterion to make resource groups. We determined the values of $\tilde{A}$ and $\tilde{C}$, by considering the execution behavior presented in Fig. 5. Table 3 shows the values of $\tilde{A}$ and $\tilde{C}$ for weekdays and weekends, respectively.

Let us consider task scheduling based on the resource groups that are classified by the execution behavior of resources. In this paper, we use the task scheduling policy, in which a task is first allocated to a resource in the group with the highest availability and credibility (i.e., R_1). If there are no more resources in R_1, the task is allocated to a resource in R_2. By a way of this process, tasks are allocated to resources in each group in the order of R_1, R_2, R_3, and R_4.

We conducted the simulation to evaluate the performance of desktop grids when the execution behavior of resources is applied to task scheduling. The log data from Jan. 2008 that is different from ones that have been used to extract the availability and credibility of resources was

Table 2. Four sets of resource groups by availability and credibility

Group	Availability	Credibility
R_1	High	High
R_2	High	Low
R_3	Low	High
R_4	Low	Low

Table 3. Values of $\tilde{A}$ and $\tilde{C}$

(a) weekday

Time Zone	$\tilde{A}$	$\tilde{C}$
00:00~09:00	0.8	0.8
09:00~12:00	0.65	0.7
12:00~15:00	0.5	0.6
15:00~24:00	0.6	0.7

(b) weekend

Time Zone	$\tilde{A}$	$\tilde{C}$
00:00~09:00	0.8	0.8
09:00~12:00	0.8	0.8
12:00~15:00	0.8	0.8
15:00~24:00	0.8	0.8

used in the simulation (see Fig. 6(b)). For comparison, the existing task scheduling is also simulated without any consideration for the execution behavior. We used turnaround time and the number of used resources as the performance measure. Fig. 7 shows the average turnaround time and the average number of used resources when 100-1,000 tasks are used.

From the results presented in Fig. 7(a), we can observe that the resource group-based task scheduling has a much lower turnaround time than the existing task scheduling, regardless of the number of tasks. These results are caused from the use of different criterion on availability and credibility for each time zone. Meanwhile, as the existing task scheduling does not consider the execution behavior of resources, it allocates tasks to resources regardless of the execution environment of desktop grids. Thus, if resources have high availability and credibility, the return of task results can be fast. Otherwise, the fast return of task results can hardly be expected. As a result, the resource group-based task scheduling will obtain a faster turnaround time by allocating tasks to resources according to the execution behavior of resources compared to the existing task scheduling.

The results of Fig. 7(b) show that the resource group-based task scheduling can complete entire tasks in spite of using much less resources compared to the existing task scheduling. This results from the fact that our task scheduling can allocate tasks to the resources with as high availability and credibility as possible by analyzing the execution behavior of resources and thus task failures can be handled at a low level. On the contrary, in the existing task scheduling, tasks can be allocated to the improper resources for the execution environment of each time zone and hence the chances of task failures will increase dramatically. Once a task fails, it should be allocated to other resources again. Task failures will induce the waste of resources, resulting in low resource utilization.

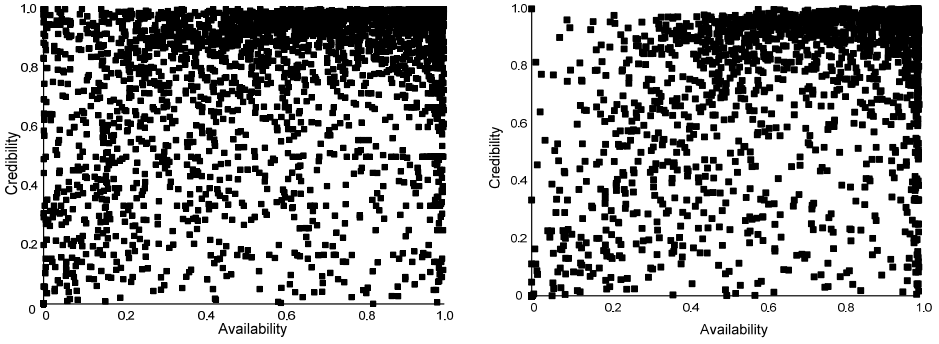


Fig. 6. Distribution of availability and credibility

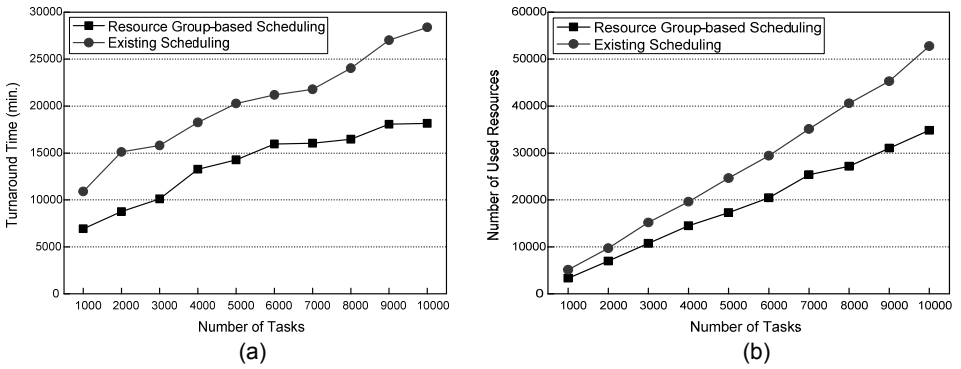


Fig. 7. Comparison of scheduling performance

In conclusion, it is seen that a fast turnaround time can be achieved with less resources if task scheduling is operated with the execution behavior of resources that have been extracted from the log analysis system of this paper.

6. CONCLUSION

In this paper, we implemented a log analysis system with REST web services, which can extract the execution behavior of resources in desktop grid systems. Actual log data in Korea@Home systems were used to analyze the execution behavior. We observed from the analysis results that there is a large difference between execution behaviors on weekdays and weekends for each time zone. In the simulation to verify the applicability of the implemented log analysis system; resource group-based task scheduling was applied to actual desktop grid systems after grouping resources according to availability and credibility. Simulation results indicated that this scheduling can achieve a faster turnaround time in spite of using fewer resources than the existing one. Therefore, we expect that the log analysis system can be usefully utilized in improving the performance of new task scheduling policies or existing ones by analyzing the task execution environment in advance before these policies are directly applied to actual desktop grid systems.

REFERENCES

[1] P. Kacsuk, R. Lovas, and Z. Németh, *Distributed and Parallel Systems in Focus: Desktop Grid Computing*, Springer, 2008.

[2] D. P. Anderson and G. Fedak, "The Computational and Storage Potential of Volunteer Computing," Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006, pp.73-80.

[3] I. Al-Azzoni and D. G. Down, "Dynamic Scheduling for Heterogeneous Desktop Grids," *Journal of Parallel and Distributed Computing*, Vol.70, No.12, 2010, pp.1231-1240.

[4] BOINC: Berkeley Open Infrastructure for Network Computing, http://boinc.berkeley.edu.

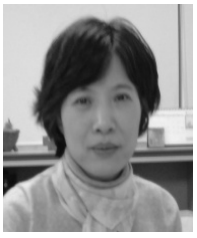
[5] XtremWeb, http://www.xtremweb.net.

[6] Korea@Home, http://www.koreaathome.org.

[7] D. Kando, G. Fedaka, F. Cappello, A. A. Chien, and H. Casanova "Characterizing Resource Availability in Enterprise Desktop Grids," *Future Generation Computer Systems*, Vol.23, 2007, pp.888-903.
[8] M. L. Shooman, *Reliability of Computer Systems and Networks*, John Wiley & Sons Inc., 2002.
[9] JDOM 1.1, http://www.jdom.org.
[10] MySQL 5.5, http://www.mysql.com.
[11] Lightweight REST framework for Java, http://www.restlet.org.
[12] S. Vinoski, "Demystifying RESTful Data Coupling," *IEEE Internet Computing*, Vol.12, No.2, 2008, pp.87-90.
[13] L. Richardson and S. Ruby, *RESTful Web Services*, O'Relly, 2007.

Joon-Min Gil

He received his B.S. and M.S. degrees in Computer Science from Korea University, Korea in 1994 and 1996, respectively. He received his Ph.D. degree in Computer Science and Engineering from Korea University, Korea in 2000. From 2001 to 2002, he was a Visiting Research Associate in the Department of Computer Science, University of Illinois at Chicago, USA. From Oct. 2002 to Feb. 2006, he was a Senior Research Engineer in Supercomputing Center at Korea Institute of Science and Technology Information, Korea. He joined Catholic University of Daegu in March 2006, where he is currently an Assistant Professor in the School of Computer and Information Communications Engineering. His research interests include cloud computing, grid computing, Internet computing, and wireless & sensor networks.



Mihye Kim

She received her Ph.D. in Computer Science and Engineering in 2003, from New South Wales University, Sydney, Australia. She is currently an Associate Professor in the Computer Science Education Department at Catholic University of Daegu, South Korea. Her research interests include knowledge management and retrieval, computer science education, digital textbooks, and cloud computing.