

# PROPOSED OF A LOAD BALANCING METHOD FOR DATA INTENSIVE APPLICATIONS ON A HYBRID CLOUD ACCOUNTING FOR COST INCLUDING POWER CONSUMPTION

Yumiko Kasae<sup>1</sup>, Masato Oguchi<sup>1</sup>

<sup>1</sup>*Ochanomizu University, 2-1-1 Ohtsuka, Bunkyo-ku, Tokyo 112-8610, Japan  
yumiko@ogl.is.ocha.ac.jp, oguchi@computer.org*

**Keywords:** Hybrid Cloud, Power Consumption, Middleware, Load Balancing, Evaluation of Costs, Data-Intensive Application

**Abstract:** Based on the recent explosive increase of information in computer systems, we need a system that can efficiently process large amounts of data with limited resources. In this paper, we propose a method to implement such a system in its Hybrid Cloud environment, implemented as Middleware. Using this Middleware, the user can not only efficiently process large amounts of data but can also control monetary costs, including power consumption, by setting parameters. Furthermore, we evaluate the total costs, calculated by Execution Time, Public Cloud's Metered Rates and Charge of Power Consumption on the Private Cloud when running our Middleware.

## 1 INTRODUCTION

Due to the recent explosive increase of information in computer systems, we need a system that can efficiently process a large amount of data with limited resources. To achieve this, using cloud computing is effective, and it has spread throughout the world. However, this diffusion has caused power consumption to increase in the equipment used to build a cloud. Moreover, according to the global eco-conscious trend, we should reduce the power consumption of computer systems.

To reduce power consumption, it is possible to develop power-saving computer systems and air-conditioning equipment. However, it is often difficult to realize such a power-saving system from the hardware perspective. We thus also require a software effort to this end. In this research, focusing on a hybrid cloud in cloud computing, which is a combination of public and private clouds, we propose a method that can both process a large amount of data and control monetary costs, including power consumption. We have implemented the system as middleware.

The middleware has two characteristics. First, as a target job, we have used a data-intensive application. The middleware measures disk I/O periodically as an indicator for load-balancing decisions. Second, the middleware aims to realize power-saving load balancing. This middleware considers two types of costs:

time and monetary cost. The time cost is the execution time, and the monetary cost is the sum of the charge of a public cloud at a metered rate and the charge of power consumption on a private cloud. The power consumption charge on a public cloud is included in the charge at a metered rate because it is difficult for a user to know the actual power consumption charge on a public cloud; it should therefore be included in the charge at a metered rate.

In this middleware, it is possible to realize power-saving oriented load balancing by controlling the rate of private/public cloud usage. By changing the parameters, users can choose to lay weight on either execution time or monetary cost, which includes power consumption. In this paper, we have measured and calculated execution time, the charge of a public cloud at a metered rate, and the power consumption charge on a private cloud while executing the application with our middleware. Using these results, we have estimated the time and monetary costs. Moreover, we have evaluated these costs when users decide the importance of both costs.

The remainder of this paper is organized as follows. Section 2 introduces related research studies. Section 3 shows an experimental system of the hybrid cloud that we have constructed in this study. Section 4 describes our proposed method and introduces how to implement it as middleware. Section 5 introduces the results of the execution using our middle-

ware. Section 6 evaluates our middleware, and Section 7 presents concluding remarks.

## 2 RELATED WORKS

Previous researchers have discussed load balancing in the cloud computing (Gueyoung Jung and Pu, 2008) and (Evangelia Kalyvianaki and Hand, 2009). In these papers, however, CPU-intensive applications were used as targets of load balancing jobs, and not as data-intensive applications. In computing-centric applications, like some scientific calculations, performing appropriate load balancing based on the CPU load of each node is possible. In this research, however, we have used a data-intensive application as a target of the jobs. In such a case, because the CPU is often in the I/O waiting state, load balancing is almost impossible by CPU load. In this research, we have used the disk I/O as a load indicator.

In data-intensive applications, load balancing middleware was also developed using the amount of disk access for load decisions (Shiori Toyoshima and Oguchi, 2011). This middleware, based on disk access, provided dynamic load balancing between public clouds and a local cluster and performed the optimal job placement. As we have further developed middleware by introducing user-specified parameters, it becomes possible to reduce the monetary costs of load balancing, including power consumption.

Power saving in cloud computing has also been actively investigated. Unlike this study, researchers have discussed an approach to power saving when executing CPU-intensive applications in a cloud (Kazuki Mochizuki, 2011), (Che-Yuan Tu, 2010). Another study (C.Peoples and S.McClean, 2011) examined power saving efforts for a cloud datacenter. Our study aims to save power in entire clouds, including private clouds. Researchers (Luna Mingyi Zhang, 2010) proposed a scheduling algorithm considering power consumption and a job's execution time. However, these studies differ from our study, especially because we have focused on the point at which total costs in the hybrid cloud are discussed, consisting of job execution time, public cloud charge at a metered rate, and power consumption charge on a private cloud. In addition, we have used data-intensive applications as target jobs.

## 3 ENVIRONMENT FOR EVALUATION

In this paper, we have used the cloud-building software, Eucalyptus (Daniel Nurmi, 2010) to build two cloud systems. Eucalyptus, as Virtualization Software, supports Xen and KVM. In previous research that investigated characteristics of virtualization software (Kasae and Oguchi, 2011), Xen was found superior to KVM in performance, especially when data-intensive jobs are executed in a Eucalyptus-built cloud. Therefore, we have used Xen in this experiment.

Figure 1 shows the organization of the Eucalyptus system. Eucalyptus has three layers: Cloud Controller (CLC), Cluster Controller (CC), and Node Controller (NC). CLC, with a compatible interface EC2, performs control based on the content requested by the user. CC performs network control between an instance and an external cloud or among instances. Instance refers to a virtual machine provided by the cloud. NC performs instance control.

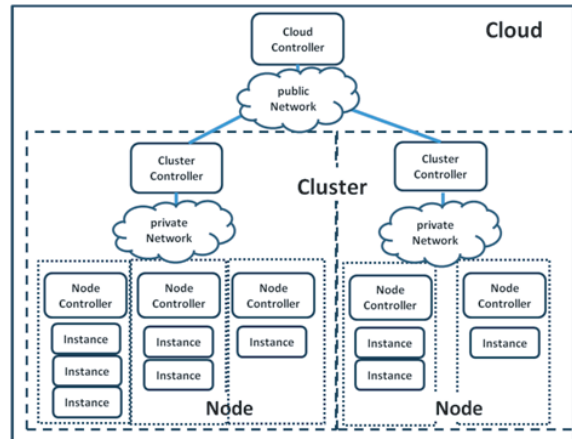


Figure 1: The system of Eucalyptus

We have built an emulated hybrid cloud environment in our laboratory. Figure 2 shows an experimental system developed in this research. In the Hybrid Cloud experimental system, the Private Cloud has a Frontend Server that performs the CLC and CC and four Node Servers, in which instances are created. Similarly, the Public Cloud has 1 Frontend Server unit and 4 Node Server units. The Private and Public Clouds are connected to Dummynet, which artificially causes delay.

Tables 1 and 4 provide specifications for each physical machine. The two Frontend Servers have two network ports: one is connected to the Node Servers, and the other to an external network. With

this configuration, each Node Server is independent from the external network.

To measure power consumption in this environment, we have used the watt-hour meter SHW3A(SHW3A, 2008)., which is a high-precision power meter produced by the System Artware Company in Japan. After plugging an electric product into SHW3A, the power consumption is instantly measured and displayed. This study only measures the Private Cloud’s Node power consumption. In reality, it is difficult for users to know the amount of power consumed by a Public Cloud. Instead, in the Public Cloud, we have assumed that the power consumption charge is included in the Public Cloud charge at a metered rate. This is discussed later. The measured power consumption is collected by a monitoring PC.

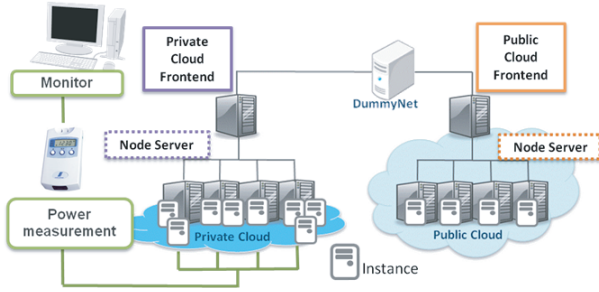


Figure 2: The System for Experiment

Table 1: Private Cloud Frontend

OS	Linux 2.6.38/Debian GNU/Linux 6.0
CPU	Intel(R) Xeon(R) CPU @ 3.60GHz
Memory	4GByte
Disk	141GByte

Table 2: Private Cloud Node

OS	Linux 2.6.32-xen-amd64 and xen-4.0-amd64 /Debian GNU/Linux 6.0
CPU	Intel(R) Xeon(R) CPU @ 3.60GHz
Memory	4GByte
Disk	222GByte

Table 3: Public Cloud Frontend

OS	Linux 2.6.38/Debian GNU/Linux 6.0
CPU	Intel(R) Xeon(R) CPU @ 2.40GHz
Memory	1GByte
Disk	72GByte

Table 4: Public Cloud Node

OS	Linux 2.6.32-xen-amd64 and xen-4.0-amd64 /Debian GNU/Linux 6.0
CPU	Intel(R) Xeon(R) CPU @ 3.60GHz
Memory	4GByte
Disk	141GByte

## 4 PROPOSED LOAD BALANCING METHOD AND MIDDLEWARE IMPLEMENTATION

### 4.1 Load Balancing Method used in Hybrid Cloud

This section explains the middleware that implements the proposed load balancing method. This Middleware performs load balancing of data-intensive jobs during runtime. It estimates the number of jobs by monitoring the disk I/O of the private and public clouds. In a sequence where data-intensive application jobs are submitted consecutively, after all resources in the private cloud are fully utilized, the next job is distributed to the public cloud’s resources. In the public cloud, after fully utilizing resources for submitted jobs, new instances in public cloud are also acquired, and their resources are used for the next jobs.

Because this middleware performs load balancing after using up the resources, the Total Cost (discussed below) can be reduced and the jobs are processed efficiently. In addition, by changing parameter settings according to the user’s instructions, power-saving load distribution can also be achieved.

This middleware, namely with shell scripts and C programs running on the Frontend of a Private Cloud, contains Monitor and Dispatch Units. The Monitor Unit collects information on the disk I/O periodically, using the *dstat* command. The Dispatch Unit, based on the disk I/O information collected at the Monitor Unit, receives jobs and submits them to private or public cloud as load balancing.

### 4.2 Disk I/O Saturation Decision

This middleware uses the disk I/O to decide on resource exhaustion because it focuses on data-intensive jobs. For data-intensive jobs, as the system often waits for I/O processing, it is difficult to determine the CPU load. The disk I/O thus decides the system load.

Figure 3 shows a graph of execution time and disk I/O when data-intensive jobs are thrown in every two

seconds. According to Figure 3, as the number of jobs increases, the value of disk I/O becomes saturated. When this occurs, the execution time becomes longer compared to the baseline condition, which is not the case with the saturated disk I/O.

This middleware thus is used only within a range where resource usage is not saturated. The middleware periodically measures the disk I/O value. When the disk I/O exceeds a saturation value (hereafter 'S') designated times, the instance is considered saturated, and load balancing to another instance is performed. When the disk I/O falls below the S value, the instance is not considered saturated. This is one of the most suitable methods to estimate saturation because the disk I/O value is unstable.

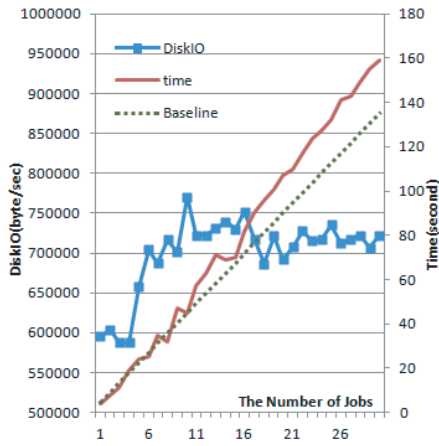


Figure 3: DiskIO and Execution Time of executing Data Intensive Application

Users can determine the saturation level parameters of this middleware. Users can control load balancing by weighting either the execution time or monetary cost, which includes power consumption. Changing the saturation level means changing the following two parameters: "the number of times the instance's disk I/O exceeds the S value (hereafter 'M')" and "the number of times the instance's disk I/O falls below the S value (hereafter 'L')".

If the M value decreases and the L value increases, jobs are easily considered saturated, which makes heavy use of the public cloud. In other words, as more monetary costs exist, job processing times become faster. Conversely, if the M value increases and the L value decreases, jobs are less likely to be saturated, and most jobs are processed in the private cloud. In this case, though the job processing time becomes longer, monetary costs, including the energy consumption charge, can be lowered.

### 4.3 Evaluation Indices of Middleware

Considering load distribution in a cloud, many tasks can be executed in parallel using many instances on a public cloud, and the overall execution time is thus expected to be shortened. However, if many instances on public cloud are employed, though the power consumption charge on a private cloud is suppressed, the public cloud charge at a metered rate increases.

To achieve load balancing on hybrid clouds while accounting for power saving, it is important to allocate resources based not only on execution time but also on the cost and power consumption. As an evaluation index for this middleware, the time and monetary costs are thus considered. The time cost is the job execution time. The monetary cost is sum of the power consumption charge on a private cloud and the public cloud charge at a metered rate.

### 4.4 Middleware Algorithms

In our experiments, we assume that jobs are consecutively submitted into the system.

- (1) Receive jobs that are consecutively submitted.
- (2) Check all instances' disk I/O in the Private Cloud. If the higher priority instance's disk I/O value is not saturated, run the jobs in that Private Cloud instance and return to Step (1).  
If all Private Cloud instances are saturated, proceed to Step (3).
- (3) Examine whether the disk I/O value of the high priority instance in the Public Cloud is saturated. If the instance is not saturated, run the jobs in that instance and return to Step (1). If not found, go to Step (4).
- (4) Rent one instance in the Public Cloud, run the jobs in that instance, and return to Step (1).

First, the middleware checks the status of the disk I/O on the private cloud; it next examines the status of the disk I/O on a higher priority instance on the public cloud. By executing jobs at a vacant status instance with higher priority, the middleware realizes well-balanced load balancing while considering time and monetary costs.

## 5 MAIN EXPERIMENTS AND MEASUREMENTS

### 5.1 Experiment Overview

In this experiment, as shown in Figure 4, after generating one instance in every Node server in both clouds, load balancing was performed in eight total instances. Table 5 shows the performance of every instance. In the public cloud, the instances are generally available without restriction. In this experiment, the total number of thrown jobs is capable of load balancing within the public cloud's 4 instances.

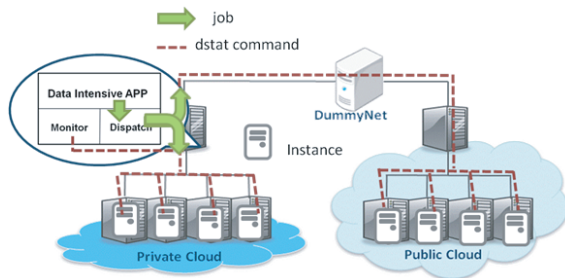


Figure 4: The System For Experiment

Table 5: Instance

OS	Linux 2. 6. 27. 21-0.1-xen / x86_ 84 GNU / CentOS 5.3
CPU	Intel(R) Xeon(R) CPU @ 3.60GHz 1 core
Memory	1024MByte
Disk	10GByte

In the experiment, the saturation level varies by changing the M values, as in Table 6, but maintains a fixed L value, described in Chapter 4. In this saturation level, if the level is small, the load is easily distributed, and if the level is large, the load is hard to be distributed.

Table 6: The Load Balance Level

Level	Value of L	Value of M
I	5	2
II	5	3
III	5	4
IV	5	5
V	5	6
VI	5	7
VII	5	8
VIII	5	9
IX	5	10

When varying the saturation level, we thus measure the job processing time, metered public cloud costs and private cloud's power consumption rates. We then evaluate these costs.

### 5.2 Data Placement

For load balancing when using the cloud, it is important to consider data placement. This experiment considers placement using a block storage attached to a cloud or remote access to local storage from a public cloud instance, which assumes that we have already placed the instance data. Data placement will be discussed in detail in a future work.

### 5.3 About Populated Jobs

As a job to bring into this experiment, we sought a data-intensive application and used pgbench(pgbench, 1999)., which is the PostgreSQL benchmark. Pgbench is a simple tool benchmark that is bundled with PostgreSQL. Tatsuo Ishii created the first version, published in 1999 by the PostgreSQL mailing list in Japan. The program now comes with a contrib, now distributed with the PostgreSQL source code. Because this benchmark uses a basic server-side database based on TPC-B, the performance can be judged by the number of transactions allowed per second in this experiment to create a PostgreSQL database on the local 6 Gbyte in all instances. The number of clients in this job is 1, and number of transactions is 500. We were thrown jobs every two seconds 200 times. The processing time per job is approximately 9 seconds. In this experiment, we assumed that independently data-intensive small jobs, including pgbench, were continuously thrown.

### 5.4 Measurement Result

Figures 5, 6, and 7 show the measurements. Figure 5 shows the job processing time. Figure 6 shows the costs of the metered public cloud. Figure 7 shows the private cloud's power consumption. In each graph, the horizontal axis shows the number of submitted jobs.

Figure 5 shows the entire time until all jobs are submitted. Figure 6 shows the public cloud charge. This is computed by multiplying the metered charge by the number of rented public cloud instances plus the job execution time. The public cloud's metered charge is based on the metered charge AmazonEC2; instances in this study had performance calculated as \$ 0.5 per hour. Figure 7 shows the power consumption until all jobs are submitted to the private cloud.



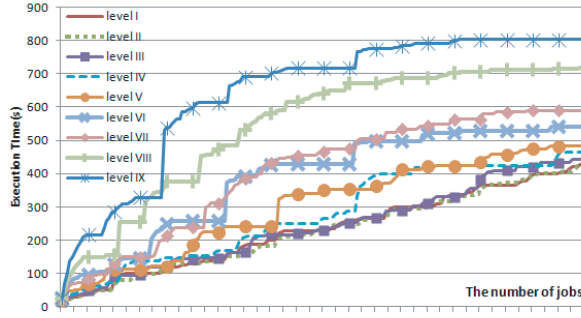


Figure 5: Execution Time

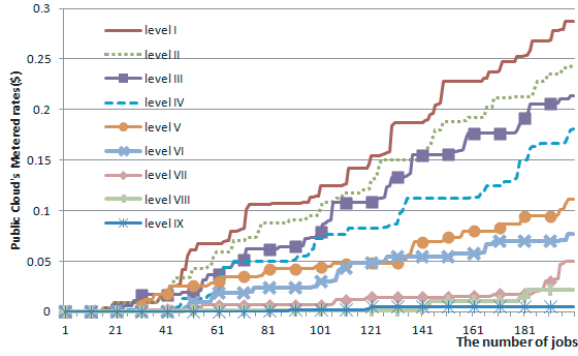


Figure 6: Public Cloud's Metered rates

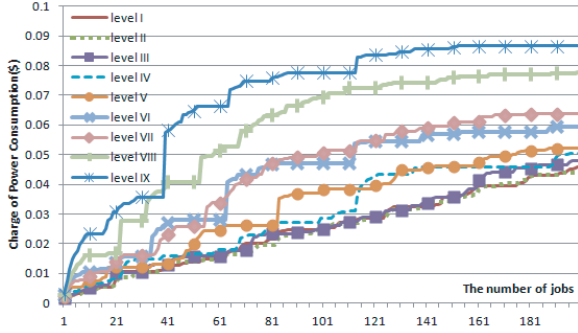


Figure 7: Charges of Power Consumption on Private Cloud

Power consumption rates, with reference to TEPCO's electricity rates, were calculated as \$ 0.5 per 1kwh.

In Figure 5's graph, despite equal processing times in levels I and II, the metered charge's level I is higher than its level II in Figure 6's graph. For this job, load balancing can have enough resources to reach level II. The saturation level of level I is wasting public cloud resources. From saturation levels III to IX, the metered charge is lower as the level increases, as in Figure 6, indicating that load balancing becomes difficult. Accordingly, job processing also takes time.

As Figures 7 and 5 are similar, the processing time clearly has a significant impact on energy consumption. By varying saturation levels in this middleware, we thus realized ways to control the power consumption rates in a private cloud, metered rates in a public cloud and job execution time.

In addition, Figure 8 indicates the execution time per job and the Disk I/O value when running the middleware.

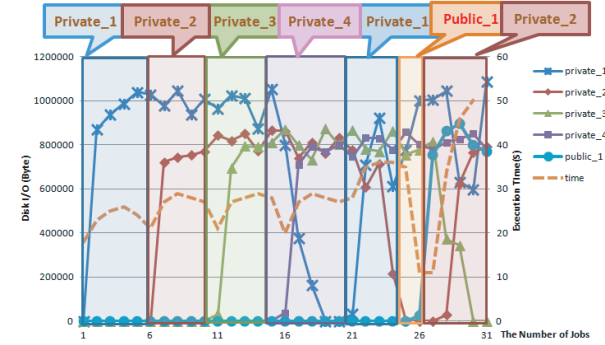


Figure 8: the load balancing control corresponding to Disk I/O

First, an instance named private cloud 1 is thrown to the job. If private cloud 1 is determined to be saturated by the saturated Disk I/O value, the job is submitted to private cloud 2. After saturating Private Cloud 2, the job is thrown to Private Cloud 3. After saturating Private Cloud 3, load balancing sends the job to Private Cloud 4. At this time, private cloud 1's Disk I/O saturation is alleviated, and the next job is put into a private cloud. In this condition, as had been saturated with all instances in the private cloud, the following jobs are load balanced to the public cloud. Load balancing can thus be achieved by having the middleware control the Disk I/O.

## 6 EVALUATING MIDDLEWARE IMPLEMENTATION USING THE PROPOSED METHOD

### 6.1 Evaluation Overview

In this section, based on the results of measurements in Chapter 4, we have evaluated the middleware. In this middleware, consider the following equation as an evaluation index.

$$\text{Total cost} = F * T_{total} + (T_R * N_R * C_R + P_L * C_L)$$

$T_{total}$ : Execution time of Total Jobs

$T_R$ : Execution time on the Public Cloud

$N_R$ : Number of Instances used on the Public Cloud  
 $C_R$ : Public Cloud Charges  
 $P_L$ : Power Consumption on the Private Cloud  
 $C_L$ : Charges of Power Consumption on Private Cloud  
 $C_L$ : Power Consumption Charges on the Private Cloud  
 $F$ : The Factor for Converting Money into Time

The first term represents the time cost of the execution time. The second term represents the monetary cost, which is the sum of the power consumption rates on a private cloud and the charge of metered costs on public cloud. We have considered the power consumption charge on a public cloud, which includes the metered cost, because it is difficult for a user to know the actual charge. If users know how much power is consumed by the public cloud, the users' motivation determines whether to try to reduce its power.

Factor  $F$  is intended to be converted into monetary and time costs. The users use factor  $F$  to decide how to balance the time and monetary costs. In this chapter, we first discuss the financial costs using the metered and power rate pricing. Based on the monetary costs considered, we then evaluate the total cost by varying the factor  $F$  in the above formula.

## 6.2 Discussion of Monetary Costs

In this study, we have considered the monetary cost, which is the sum of the metered public cloud charges and power consumption rates of the private cloud. These two amounts are not always constant, as many cloud provider's recent metered rates have not necessarily been constant. Moreover, as such providers increase in the future, metered rates may decrease based on price competition, but may be higher now. This is also true for energy consumption rates. Based on these facts, we have considered the various financial costs when evaluating this experiment, as the prices of both energy consumption and metered rates vary. In the experiment, we have converted the metered charges as \$0.5 per instance and power consumption rates as \$0.5 per 1 kwh. In this evaluation, as in Figure 9, metered rates varied from \$0.5 to \$1.5, and power consumption rates varied from \$0.5 to \$3.0. When changing one value, the other was fixed at its initial value.

Figure 9 shows that, for most pricing, the financial costs of level 1 were larger. In this experiment, this means that the metered costs are more expensive than the energy consumption charge. However, by varying the amount of \$3.0, which is a high power consumption charge, we have found relatively equal costs at every level. In this pricing, the metered charges form the majority of the monetary cost at low levels, but at higher levels, the metered rates are smaller,

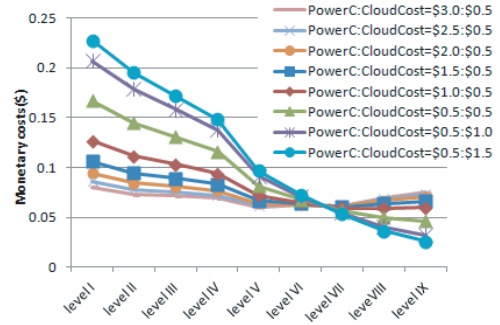


Figure 9: The Monetary costs

and increase the percentage of electricity consumption charges. The total financial cost has not changed. Figure 9 presents the typical results: PowerConsumption: CloudCost = \$3.0: \$0.5, \$1.5: \$0.5, \$0.5: \$0.5, \$0.5: \$1.0, \$0.5: \$1.5. We have assumed that these results are representative examples. Using these monetary costs, we have evaluated the total cost.

## 6.3 Evaluation of Total Cost

When evaluating the total cost, we have used the formula described in section 5.1. In this formula, factor  $F$  is important. Using factor  $F$ , the total job execution time is converted into monetary costs. In this evaluation, the  $F$  value has changed between  $1/20$ ,  $1/200$ ,  $1/2000$  and  $1/20000$ . Using  $F=1/20$  means that the user considers processing time to be the most important cost, whereas using  $F=1/20000$  means that the user considers monetary cost to be the most important. Figures 10, 11, 12, and 13 show that the Total Cost of factor  $F$  varies. In addition, as discussed in Section 5.2, monetary cost pricing is also changed in these figures.

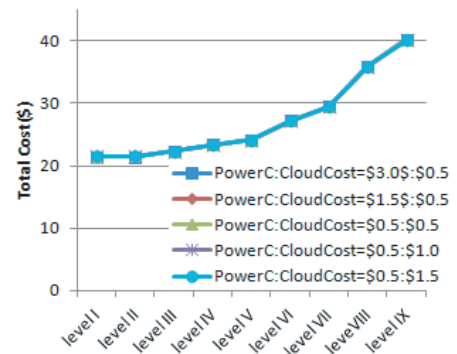


Figure 10: Total Cost (  $F=1/20$  )

As Figure 10 is the result that considered processing time to be the most important cost, there is little change in the pricing of each. This result primarily in-

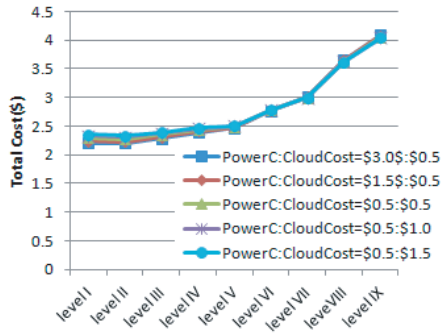


Figure 11: Total Cost ( F = 1/200 )

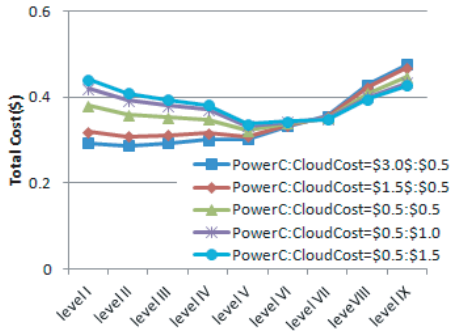


Figure 12: Total Cost ( F = 1/2000 )

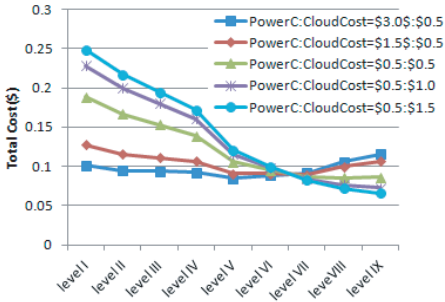


Figure 13: Total Cost ( F = 1/20000 )

indicates that the result depends only on the processing time. For this reason, to provide the time-critical load distribution processing by the user requires setting the level I parameters, at the lowest saturation level. This provides a reasonable result. Figure 11 was compared to Figure 10, but when considering the financial cost, there is almost no difference between two graphs. In Figure 11, as in Figure 10, the user specifying level I, which is the lowest saturated level, can reduce the total cost.

However, these results are different in Figure 12. This result is a cost evaluation, which suppresses processing time and monetary costs. Figure 12 indicates a difference in Total Cost when examining the pricing. PowerConsumption: CloudCost = \$0.5: \$0.5, \$0.5: \$1.0, \$0.5: \$1.5, if the power consumption of

the private cloud charges are fixed, and the pricing of the rates change metered is change, the total cost is the smallest in level VI or level V.

In this experiment, there is almost no difference in the number of jobs when considering the total cost, which is sum of the time and monetary costs, even when executing the load distributed between level I and V, and we also find that this has the lowest cost. In this pricing, if the user set up parameters of level V from those in level I, the middleware thus provides load balancing that controls both Time and Monetary Costs. PowerConsumption: CloudCost = \$0.5: \$0.5, \$0.5: \$1.0, \$0.5: \$1.5 such that if the private cloud charges power consumption is fixed and the rate change metered pricing changes, the total cost is the smallest in levels V or VI.

When considering the Total Cost of the importance of metered rate pricing, these results indicate that the middleware, following the user-set parameters for levels V or VI, execute load balancing that reduces both Time and Monetary Costs.

In Figure13, because it is little consider of time cost, and largely reflected the impact of monetary cost. Figure13 does not greatly consider time cost, which largely reflects the impact of monetary cost. That is the Total Cost of monetary cost-consciousness.

PowerConsumption: CloudCost = \$3.0: \$0.5, \$1.5: \$0.5 such that if the monetary costs are dominated by power consumption rate pricing on a private cloud, the Total Cost creates little difference between levels. This pricing causes no change in the Total Cost when executing this middleware, as the pricing is more important to monetary costs, even following the user parameter settings. In addition, PowerConsumption: CloudCost = \$0.5: \$0.5, \$0.5: \$1.0, \$0.5: \$1.5 such that if the private cloud rate power consumption is fixed and the metered cost rate has changed, as in level IX, and if most jobs were executed in the Private Cloud without using the Public Cloud, the user can control the Total Cost. Therefore, when considering the Total Cost of the importance of metered rate pricing and Monetary Cost, the middleware, following the user-set parameters for level IX, executes load balancing that reduces Monetary Costs.

## 7 CONCLUSIONS

In this research, especially focusing on hybrid clouds, we have proposed a method that can process large amounts of data and control the monetary cost, which includes power consumption. We have also implemented the System as Middleware. Using this



Middleware, the user can not only efficiently process large amounts of data, but also control the monetary cost, which includes power consumption, by setting parameters. By varying the parameters to run the middleware, we have measured and calculated processing time, public cloud metered rates and power consumption charge on a private cloud. We have evaluated the total cost by calculating the sum of the costs and the financial time costs. This evaluation showed that this middleware perform load balancing can reduce costs if the actual user sets the parameters. To reduce load balancing in both the Time and Financial costs, we have demonstrated that this middleware is successfully implemented.

## ACKNOWLEDGEMENTS

This work is partly supported by the Ministry of Education, Culture, Sports, Science and Technology, under Grant 22240005 of Grant-in-Aid for Scientific Research. The authors would like to thank to Drs. Atsuko Takefusa, Hidemoto Nakada, Ryousei Takano, and Tomohiro Kudoh at the National Institute of Advanced Industrial Science and Technology (AIST) for the conscientious advice and help with this work.

## REFERENCES

- Che-Yuan Tu, Wen-Chieh Kuo, W.-H. T. Y.-T. W. S. S. (2010). A power-aware cloud architecture with smart metering. pages 497–503. Parallel Processing Workshops (ICPPW), 2010 39th International Conference.
- C.Peoples, G. and S.McClean (2011). Energy-aware data centre management. pages 1–5. Communications (NCC), 2011 National Conference.
- Daniel Nurmi, Rich Wolski, C. G. G. O. S. S.-L. Y. D. Z. (2010). The eucalyptus open-source cloud-computing system. pages 62–73. Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference.
- Evangelia Kalyvianaki, T. C. and Hand, S. (2009). Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In Proc. 6th International Conference on Autonomic Computing and Communications (ICAC2009).
- Gueyoung Jung, Kaustubh R. Joshi, M. A. H. R. D. S. and Pu, C. (2008). Generating adaptation policies for multi-tier applications in consolidated server environments. pages 23–32. In Proc. 5th IEEE International Conference on Autonomic Computing (ICAC2008).
- Kasae, Y. and Oguchi, M. (2011). Evaluation of energy consumption using private cloud system with various condition. In Proc. Multimedia, Distributed, Cooperative, and Mobile Symposium(DICOMO2011).
- Kazuki Mochizuki, K. S. (2011). Evaluation of optimal resource allocation method for cloud computing environments with limited electric power capacity. pages 1–5. Network-Based Information Systems (NBIS), 2011 14th International Conference.
- Luna Mingyi Zhang, Keqin Li, Y.-Q. Z. (2010). Green task scheduling algorithms with speeds optimization on heterogeneous cloud servers. pages 76–80. Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on and Int'l Conference on Cyber, Physical and Social Computing (CPSCom).
- pgbench (1999). <http://www.postgresql.org/docs/current/static/pgbench.html>.
- Shiori Toyoshima, S. Y. and Oguchi, M. (2011). Middleware for load distribution among cloud computing resource and local cluster used in the execution of data-intensive application. *DBSJ Journal, Vol.10, No.1*.
- SHW3A (2008). <http://www.system-artware.co.jp/shw3a.html>.