Proposal and Evaluation by Simulation of Management Processing Load Control Method for Virtual Client System

Katsuyuki UMEZAWA
IT Service Division, Hitachi, Ltd.
Tokyo, 101-8608 Japan
Email: katsuyuki.umezawa.ue@hitachi.com

Hiromi GOTO
IT Service Division, Hitachi, Ltd.
Tokyo, 101-8608 Japan
Email: hiromi.goto.nh@hitachi.com

***Abstract*—In recent years, virtualization of desktops has become important as a solution for various problems caused from the cost of running a company. Our company began using thin client blade (CB) systems and thin client terminal service (TS) systems about ten years ago. There are currently about 70,000 users of these systems in our group companies. A thin client virtual PC system has been developed as a virtualization platform for the desktop. The management processing load dispersion in virtual PC system was designed for a large number of users. We previously reported a method that addresses this problem. In this paper, evaluation by simulation of a previously proposed improved method for managing the load imposed on resources in a virtual PC system showed that it effectively prevents significant load increases. The threshold for initiating management processing can be flexibly set, which enables the load in a virtual client environment to be precisely controlled. Use of this method would enable virtual clients to be managed more effectively.**

I. Introduction

According to the International Data Corporation [1], the return on investment for implementing client virtualization products of 2013 is more than 400%, and the investment payback period is approximately ten months. In other words, introducing a client virtualization product produces a benefit more than four times the investment. Moreover, the cost of the investment is returned within one year.

Our company began using thin client blade (CB) systems and thin client terminal service (TS) systems about ten years ago. There are currently about 70,000 users of these systems in our group companies. A thin client virtual PC system has been developed as a virtualization platform for the desktop. The management processing load dispersion in this "virtual desktop" was designed for a large number of users [19]. We previously reported a method that addresses this problem [21] [22]. It was designed to prevent a significant increase in the load on resources such as disk I/O, CPU, network, and memory when there is a large number of users of the virtual PC system.

We have now evaluated the proposed improved method by simulation and demonstrated that it effectively prevents a significant increase in the load.

In section 2, we describe three desktop virtualization technologies, and in section 3, we describe the load problems when using them. In section 4, we discuss related work and our previous work. The original method for load dispersion is described in section 5. We overview our previously proposed improved method for preventing high loads in section 6. Our simulation of this method and the results are described in section 7. Section 8 concludes the paper with a summary of the key points and a mention of future work.

II. Desktop Virtualization Technologies

Desktop virtualization technologies can be classified into CB systems, TS systems, and virtual PC systems, as illustrated in Figure 1.

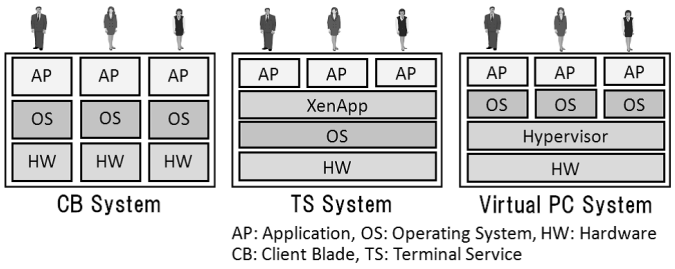

Fig. 1. Basic configurations of CB, TS, and virtual PC systems

A. Client blade systems

A CB system comprises thin PCs, called "blades," mounted in a rack. Each blade is allocated to a user. Since applications can be installed individually in each blade, an environment satisfying the needs of a user can be constructed. However, since the computing resources are not flexible, the processing capacity may be low for a single blade; however, the processing capacity of most blades can afford. In addition, managing individual client OSs can be complicated because, for example, virus measures and security patches must be implemented on each blade individually.

B. Terminal service systems

A TS system comprises a server OS on one server and applications for multiple clients on that OS. Because the server hardware resources can be flexibly used among multiple clients at the same time in a multiple desktop environment, they can be used effectively. In addition, efficient operation management

is possible because applications and data can be managed effectively. However, applications cannot be installed for each individual user.

C. Virtual PC systems

A virtual PC system comprises multiple virtual machines on one physical server controlled by a hypervisor. Each virtual machine has an OS and desktop environment. With a virtual PC system, multiple OSs operate on a server; with a TS system, there is only one OS per server. Since each OS in the virtual PC system has a virtual desktop environment, an environment that satisfies the needs of each user can be constructed because applications as well as CB systems can be installed on each individual virtual machine. Moreover, data can be controlled effectively, as with a TS system.

III. Load Problems with Desktop Virtualization

Load problems are typically encountered with desktop virtualization. OS updating, security patch application, and virus scanning (all considered to be "management processing") of virtual clients that share resources (as in the TS and virtual PC systems) increase the load on resources (disk I/O, CPU, network, memory etc.). This makes it difficult to manage a large number of virtual clients without causing a significant load.

One proposed approach to dispersing the load is to group the datastores and to group the management processes [19]. However, variations in the quantity of management processes can significantly increase the load.

Under these conditions, the manager might schedule the management processing activities on the basis of the load of the virtual client environment. Alternatively, the manager can estimate the maximum load that will be imposed by trial and error in a test environment. In addition, there may be a situation in which management processing itself cannot be done due to insufficient resources.

IV. Related Work

Several studies [2][3][4] focused on latency and bandwidth between a thin client system and server, and a number of studies [5][6][7][8][9][11] considered VM placement models. Other studies [10] investigated the performance of storage in virtualized environments. None of these studies, however, were done considered virtual desktops.

Several studies have considered virtual desktops. Shridharan et al. presented a resource defragmentation algorithm based on virtual desktop migration [12], and Calyam et al. presented a resource allocation model based on a benchmarking tool [13]. A method for managing resources in a virtual desktop environment has been investigated by several groups [14][15][16]. One finding in particular is that CPU utilization on one day is associated with the utilizations on other days [14][15]. Methods for reducing the load on shared storage by using caches have been proposed by a couple of groups [17][18].

As mentioned above, the load dispersion in the "virtual desktop" of our thin client virtual PC system was designed for a large number of users, resulting in a high system load [19]. The load was evaluated [20] by using the technique of Le Thanh Man and Kayashima [14] and [15]. Also as mentioned above, we previously reported a method [21][22] for avoiding the high load that affects the originally proposed method [19].

We have now evaluated our proposed improved method.

V. Virtual PC System

A. Overview

The virtual desktop environment of our virtual PC system accommodates 1,200 users in one data center and 5,100 users in another data centers. The plan is to build a virtual desktop environment for 30,000 users in total (three data centers, 10,000 users each).

The configurations of the current system is illustrated in Figure 2. The virtual desktop copes with hardware obstacles by using a high availability (HA) configuration of 15:1. The authentication server, virtual desktop deploy server, virtual desktop login server, and scan definition server were made redundant by using active-active configurations to deal with load dispersion and obstacles. The file server was made redundant by using N+1 configurations so that it could be used for dealing with obstacles. The number of storage management servers was determined by the number needed to manage the system.

The "Virtual Desk-top" and "Storage for System (Boot) Disk," shown at the top right of Figure 2, are related to a future discussion.

B. Previous load dispersion method of current system

1) Use of meshed storage mapping to balance load: In the originally proposed method [19], each virtual desktop controlled by a hypervisor is dispersed and assigned to multiple datastores to enable dispersion of the network load between the hypervisor and datastores. The current system executes with a load of 1/12 (for five people) even if one blade is broken. It executes with a load of 1/12 (for five people) even if the chassis is broken.

2) Use of management process grouping to balance load: In the originally proposed method [19], the disk I/O load is scattered when management processing is performed. Specifically, the virtual desktops controlled by a hypervisor are grouped so that the desktops in each group do not share a datastore.

Then, the OS or applications are updated or a virus scan is conducted for one group at a time so that datastore conflicts are avoided. The deploy management server sends appropriate instructions for doing this to either one or a few groups.

Because, in our proposed improved method, processing is carried out on a group basis, a few virtual desktops controlled by a hypervisor may share a datastore. This enables the dispersion of the network load between the hypervisor and the datastores to be better planned.

VI. Proposed Improved Method

However, a large amount of management processing would result in a high resource load, and the system could stop. In such a case, the manager must schedule the management

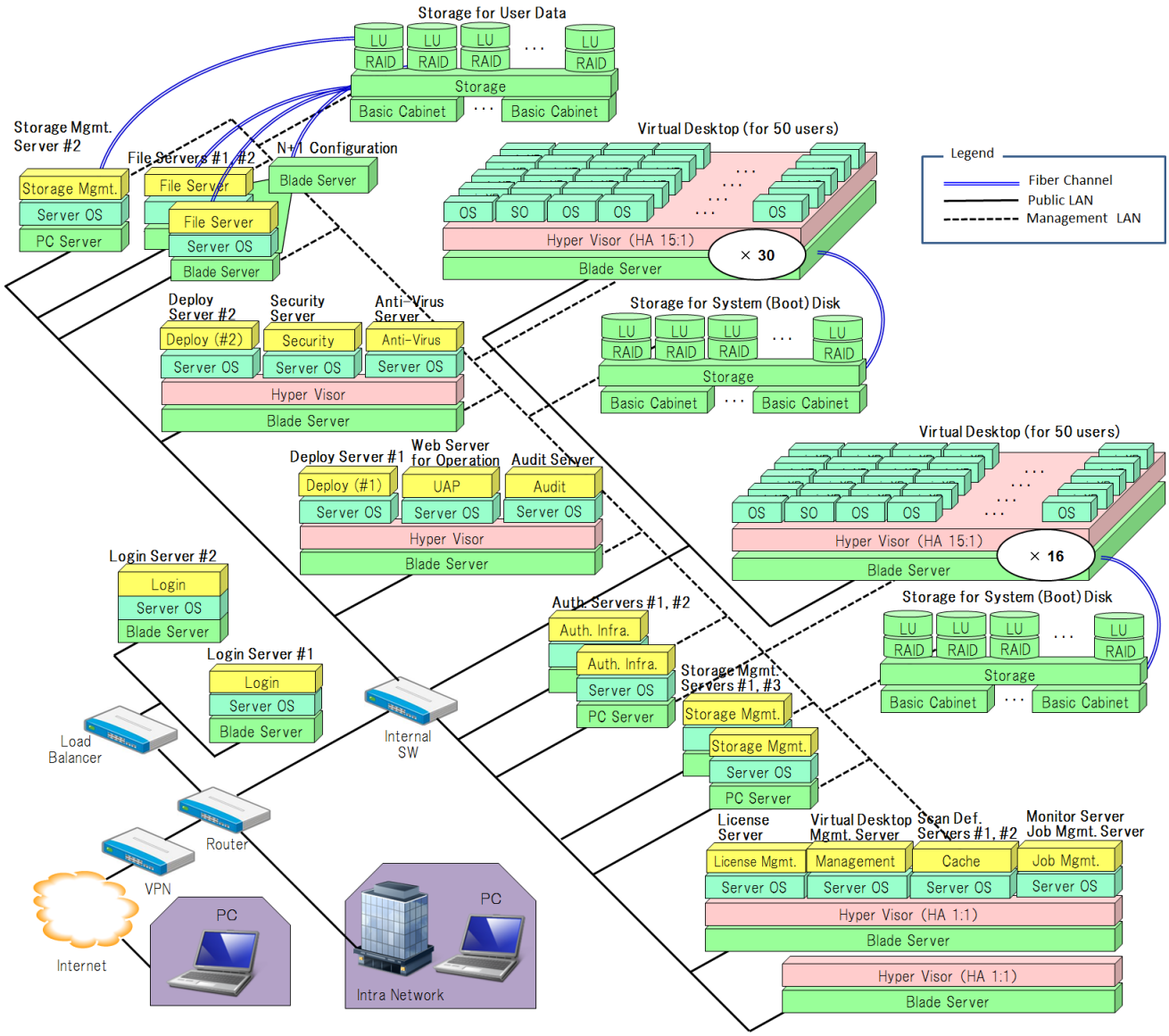

Fig. 2. Configuration of virtual PC system

processing or determine the maximum load by using a trial and error approach in a test environment, as mentioned above. Thus, the above proposed method is not sufficiently effective.

With the proposed improved method [21] [22] evaluated here, highly precise load control of the virtual client environment is achieved, and the virtual clients are appropriately and effectively managed. The concept is simple: whenever management processing is to be performed, the system checks the load situation, and only if it is lower than a threshold the processing performed.

A. Proposed system

The flow of the proposed system [21] is shown in Figure 3. [1] The management server sends a management processing instruction such as update OS, apply security patch, or conduct virus scan to each virtual desktop controlled by the hypervisor, as shown in Figure 3. [2] Before executing the instruction, each virtual desktop sends a judgment request to the server monitoring the system inquiring whether it should proceed. [3] The monitoring server observes the operation situation for the resources to be used by the virtual desktop. The server [4] compares the result with the preset threshold to determine whether the virtual desktop should proceed and [5] sends the judgment result to the desktop. The desktop carries out the processing only if it receives approval.

1) General flow of proposed system: We show a load control flow of the proposal in figure 4. The proposed load control flow is illustrated in Figure 4. The virtual client receives instructions to perform management processing (i.e., to execute a "management instruction") from a management server and sends a judgment request including the virtual client's ID to the monitoring server. The monitoring server receives the request and observes the operation situation for the virtual client. It compares each value (disk I/O, CPU load, network load,

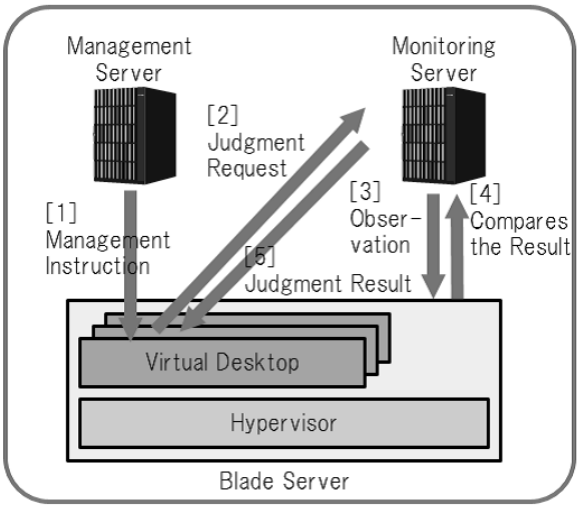

Fig. 3. Flow of proposed system

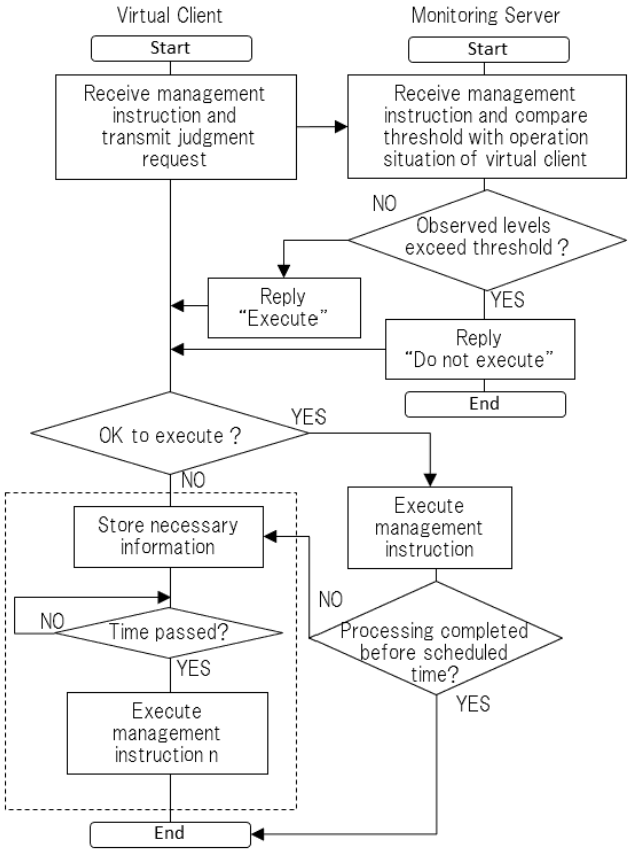

Fig. 4. General flow of proposed system

memory load, etc.) with the preset threshold values. If one or more observed levels exceed the threshold, the server notifies the virtual client that the management instruction should not be executed. Otherwise, the server notifies the client that the management instruction can be executed.

In addition, the performance judgment can be made by comparing the list of preset thresholds for every resource type

TABLE I. EXAMPLE OF THRESHOLD INFORMATION

Management instruction	Resources	6:00 – 7:00	7:00 – 8:00	...	5:00 – 6:00
Conduct virus scan	Disk I/O	4	2	...	4
	CPU	70%	40%	...	70%
	Network	70%	60%	...	80%
	Memory	65%	50%	...	65%
Deliver file	Disk I/O	5	3	...	5
	CPU	70%	40%	...	70%
	Network	50%	40%	...	60%
	Memory	85%	70%	...	85%
Back up data	Disk I/O	4	2	...	4
	CPU	70%	40%	...	70%
	Network	55%	45%	...	65%
	Memory	85%	70%	...	85%

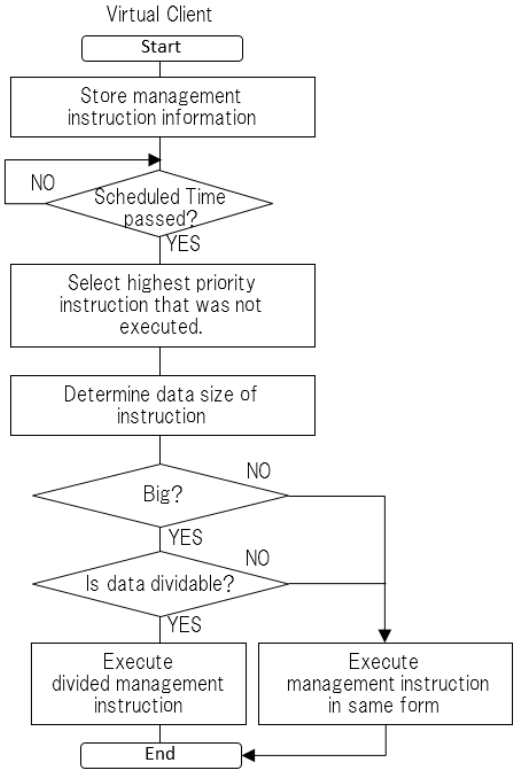

Fig. 5. Steps in restarting instruction execution

and instruction practice time, as shown in Table I, with the levels observed for the virtual client.

The virtual client executes the instruction if the judgment result sent by the monitoring server is "Execute." Execution is stopped if the previously scheduled time has passed even if the processing has not completed. Furthermore, execution does not start if the judgment is "Do not execute." In both cases, once a previously scheduled elapsed time has passed or a previously scheduled time has come, the execution process is restarted.

2) Detailed flow at time of execution: The flow marked by the dashed line in Figure 4 is shown in more detail in Figure 5. If the execution of the management instruction by a virtual client is not completed due to a "time expired" or a "Do not execute" judgment, the information needed for future

execution is saved. The task is restarted after a certain amount of time has passed.

If the previously scheduled elapsed time has passed or a previously scheduled time has come, execution of the highest priority instruction that was not executed is executed, and then, if that instruction is executed successfully, the next-highest priority instruction that was not executed is executed. The instruction information that is stored includes the priority information.

Before restarting instruction execution, the virtual client determines the data size of the instruction. If the size is "big," it judges whether the data can be divided into smaller units. If they can, the data is divided into a predetermined number of units, and execution is started. If the size is not "big", or the data cannot be divided, execution is started with the data in the same form.

VII. Evaluation

We evaluated the proposed improved method by simulating the CPU load. Usage of other resources (disk I/O, network, memory, etc.) can be similarly evaluated by simply stopping execution when at least one of two or more resources is judged to be inadequate for execution.

A. Conditions

The simulation conditions are illustrated in Figure 6 and listed in Table II.

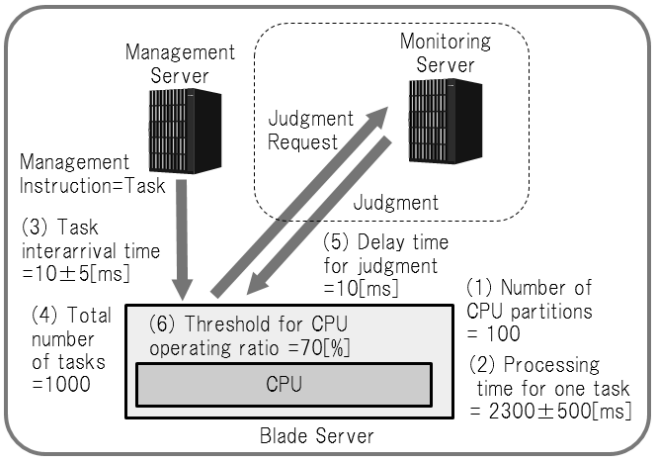

Fig. 6. Simulation Conditions

TABLE II. Simulation Conditions

Parameter	Value
(1) Number of CPU partitions	100
(2) Processing time for one task [ms]	2300
Variance in processing time	500
(3) Task interarrival time [ms]	10
Variance in interarrival time	5
(4) Number of tasks	1000
(5) Delay time for judgment [ms]	10
(6) Threshold for CPU operating ratio (%)	70

B. Results

The simulation results are plotted in Figure 8.

With the "conventional" method[1], the CPU operating ratio reached 100% immediately, and the queue steadily increased. This means that the user was soon unable to work on his or her desktop.

With the proposed improved method, the time taken to complete the 1000 tasks was about twice as long due to the CPU operating ratio being kept below the 70% threshold (which can be flexibly set). This means that at least 30% of the CPU load was available to the user for normal work. Moreover, the queue length remained zero. The proposed improved method thus effectively prevents a significant increase in the load.

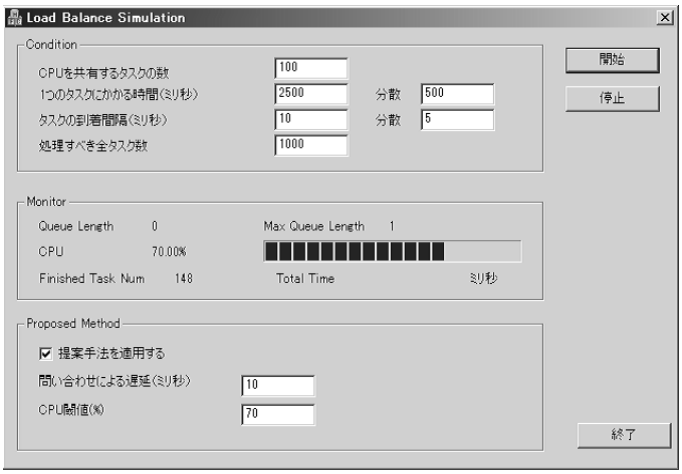

Fig. 7. Simulation screen

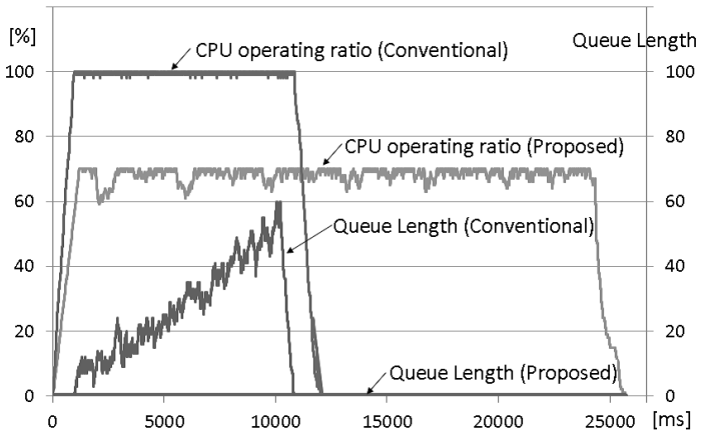

Fig. 8. Simulation results

VIII. Conclusion

Evaluation by simulation of the previously proposed improved method for managing the load imposed on resources when managing users on a large scale in a virtual PC system

[1]"conventional" means a method not to apply a proposed improved method.

showed that it effectively prevents a significant increase in the load. The ability to flexibly set the threshold for management processing enables the load in a virtual client environment to be precisely controlled. Use of this method enables virtual clients to be effectively managed. Future work includes constructing a system based on the improved method and evaluating it in a real environment.

REFERENCES

[1] "Japan client virtualization market ROI analysis," International Data Corporation Japan, 2014, http://www.idcjapan.co.jp/Press/Current/20140220Apr.html

[2] Lai, A. and Nieh, J. "On the performance of wide-area thin-client computing," ACM Transactions on Computer Systems, pp. 175–209, 2006.

[3] Tolia, N., Andersen, D., and Satyanarayanan, M. "Quantifying interactive user experience on thin clients," IEEE Computer, volume 39, pp. 46–52. 2006.

[4] Berryman, A., Calyam, P., Lai, A., and Honigford, M. "Vdbench: A benchmarking toolkit for thin-client based virtual desktop environments," 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 480–487. 2010.

[5] Malet, B. and Pietzuch, P. "Resource allocation across multiple cloud data centres," 8th International Workshop on Middleware for Grids, Clouds and e-Science (MGC), 2010.

[6] Mohammadi, E., Karimi, M., and Heikalabad, S. "A novel virtual machine placement in cloud computing," Australian Journal of Basic and Applied Sciences, volume 5, pp. 1549–1555. 2011.

[7] Piao, J. and Yan, J. "A network-aware virtual machine placement and migration approach in cloud computing," Ninth International Conference on Grid and Cloud Computing, pp. 87–92, 2010.

[8] Sonnek, J., Greensky, J., Reutiman, R., and Chandra, A. "Starling: minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration," 39th International Conference on Parallel Processing (ICPP), pp. 228–237, 2010.

[9] Sato, K., Sato, H., and Matsuoka, S. "A model-based algorithm for optimizing io intensive applications in clouds using VM-based migration," 9th IEEEACM International Symposium on Cluster Computing and the Grid (CCGRID), pp. 466–471. 2009.

[10] Gulati, A., Kumar, C., and Ahmad, I. "Storage workload characterization and consolidation in virtualized environments," 2nd International Workshop on Virtualization Performance: Analysis, Characterization, and Tools (VPACT), 2009.

[11] Zhang, Z., Xiao, L., Li, Y., Ruan, L., "A VM-based Resource Management Method Using Statistics," 18th IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 788–793, 2012

[12] Sridharan, M., Calyam, P., Venkataraman, A., and Berryman, A. "Defragmentation of resources in virtual desktop clouds for cost-aware utility optimal allocation," 4th IEEE International Conference on Utility and Cloud Computing (UCC), pp. 253–260, 2011.

[13] Calyam, P., Patali, R., Berryman, A., Lai, A., and Ramnath, R. "Utility-directed resource allocation in virtual desktop clouds," Computer Networks, volume 55, pp. 4112–4130, 2011.

[14] Le Thanh Man, C. and Kayashima, M., "Virtual Machine Placement Algorithm for Virtualized Desktop Infrastructure," IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), 2011.

[15] Le Thanh Man, C. and Kayashima, M., "Desktop Work Load Characteristics and Their Utility in Optimizing Virtual Machine Placement in Cloud," IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), 2012.

[16] Kochut, A., Beaty, K., Shaikh, H., and Shea, D., "Desktop Workload Study with Implications for Desktop Cloud Resource Optimization," IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010.

[17] Shamma, M., Meyer, D., Wires, J., Ivanova, M., Hutchinson, N., and Warfield, A., "Capo: Recapitulating Storage for Virtual Desktops," 9th USENIX Conference on File and Storage Technologies (FAST), pp. 31–45. 2011.

[18] Lagar-Cavilla, H., Whitney, J., Scannell, A., Patchin, P., Rumble, S., De Lara, E., Brudno, M., and Satyanarayanan, M., "SnowFlock: rapid virtual machine cloning for cloud computing," 4th ACM European Conference on Computer Systems, pp. 1–12. 2009.

[19] Umezawa, K., Iwashita, A., and Kato, Y., "Development of a Virtual PC Type Thin Client System," The Institute of Electronics, Information and Communication Engineers (IEICE), Information and Communication Management (ICM) Technical Report, Vol. 112, No. 378, pp. 97–102, Jan 2013.

[20] Iwashita, A., Miyake, T., Kato, Y., and Umezawa, K., "Performance Evaluation of a Virtual PC Type Thin Client System," 75th National Convention of Information Processing Society of Japan, Vol. 1, pp. 77–78, Mar. 2013.

[21] Umezawa, K. and Gotoh, H., "A Proposal of Load Control Method for Virtual Client System," The Institute of Electronics, Information and Communication Engineers (IEICE), Life Intelligence and Office Information Systems (LOIS) Technical Report Vol. 113, No. 210, pp. 49–52, Sept. 2013.

[22] Umezawa, K. and Goto, H., "Load Control System for Virtual Client System," 1st Mosharaka International Conference on Telecommunication Systems and Networks (MIC-Telecom2013), Dec. 2013.