Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), July Edition, 2013
Volume 3, Issue 7

Power Management of Server Clusters via Machine Learning and Passive Traffic Measurement

Satoru Ohta and Takehito Hirota

***Abstract*—Network services are often provided by server clusters. From the perspective of operational expenditure and the global environment, the power consumption of server clusters should be decreased. This is possible by operating the minimum number of computers required to realize a sufficiently good performance against changes in load. To do this, it is necessary to accurately determine the number of computers that should be turned on or off for the measured load metrics. This number should be determined by estimating multiple load metrics because a single metric does not adequately represent the statuses of various bottlenecked resources. In addition, decision rules should be appropriately updated if there are changes in the service content or computer specifications. To satisfy these requirements, this study proposes a machine learning approach as a method of determining the number of server computers. Another technical requirement for power management is that the load metrics should be measured nonintrusively for the OS or hardware of the cluster computers. From this viewpoint, we employ traffic parameters as the metrics that reflect resource consumptions. These traffic parameters are passively measured on a machine that is separate from the server cluster.**

This paper first explains the machine learning approach to determine the number of computers. The implementation of the approach is then presented. The effectiveness of the scheme is confirmed experimentally.

***Index Terms*— machine learning; power management; server cluster; measurement; traffic**

I. Introduction

To satisfy a large number of requests, network services are often provided by a server cluster constructed with multiple computers. The power management of a server cluster [1] is important from the viewpoint of reducing operational expenditure and CO2 emissions. Because the amount of load assigned to a cluster changes during the daytime, power consumption can be reduced by turning off server computers with light loads and turning on computers with heavy loads [2]. This is achieved by measuring the load on the cluster, determining which computers should be turned on or turned off, and finally, turning on/off computers.

The key to realizing the efficient power management of a server cluster lies in determining which computers need to be turned on and off on the basis of the measured load. In previous studies, this on/off decision was made by comparing a single or a few measured load metrics with fixed thresholds. For example, the method discussed in [2] compares the utilization of the CPU, disk interface, and network interface with thresholds. However, an on/off decision based on a comparison between load metrics and thresholds does not always provide accurate results. This is because the optimal threshold depends on the service content and machine specifications. For example, the same value of CPU utilization leads to good performance for some contents but bad performance for other contents. This happens because the performance is affected by the consumption of different resources depending on the service content.

To achieve an adequate on/off decision, multiple metrics that reflect the utilizations of various resources must be monitored. With this approach, it is favorable that the metrics are measured nonintrusively because of the robustness against excessive loads or failure. Some utilization information is available through the OS, which runs on a server computer. However, when the computer performance is significantly degraded owing to overload or failures, the utilization information may not be successfully obtained through the OS. Another reason is the compatibility of the software/hardware platform. The application interface used to extract resource utilization will differ based on the OS or hardware platform of a server computer. Thus, different measurement software must be developed for each OS or hardware. This increases the development time and cost. Therefore, it is necessary to employ the metrics, which can be measured independent of the hardware/software platform of the server computers.

The relationship between the nonintrusively monitored metrics and the on/off decision is not simple, and thus it is necessary to develop a mechanism that automatically determines this relationship. Machine learning technology is very promising for this purpose.

Machine learning is an important tool that can be used to discover a rule hidden in obtained data [3]. Machine learning

Manuscript received July 9, 2013.

Satoru Ohta is with the Department of Information Systems Engineering, the Faculty of Engineering, Toyama Prefectural University, 5180 Kurokawa, Imizu-shi, Toyama 939-0398, Japan (phone: +81-768-56-7500; fax: +81-768-56-6172; e-mail: ohta@pu-toyama.ac.jp).

Takehito Hirota is with Hokuriku Tsushin Kogyo Co., 1-7-23, Aiden-machi, Toyama-shi, Japan (e-mail: hirota1332@hokutsu.co.jp).

techniques has been successfully applied to various network/system management applications, including attack detection [4], spam mail filtering [5], and application classification [6]. If a machine learning technique is used to make the on/off decision, it is possible to find the decision rule hidden in multiple monitored load metrics. In addition, a decision rule can flexibly adapt to changes in service content or machine specifications by re-learning for the new environment. Thus, a machine learning approach is promising for determining the on/off status of computers from multiple load metrics.

This paper proposes a power management method for server clusters. The proposed method employs the traffic parameters as the load metrics. These parameters can be nonintrusively obtained from captured packets. The proposed method employs a machine learning technique to make the on/off decision. For the machine learning technique, a method based on the decision tree model [3], [7] is examined. The paper explores the method by which a machine learning framework is applied to the power management of a server cluster. Load metrics and performance criteria are also presented. The proposed method was implemented and tested on an experimental server cluster. Through experiments, it was shown that the proposed method successfully reduces the power consumption of the server cluster while offering a sufficiently good performance.

The rest of the paper is organized as follows. Section II reviews the previous related work. In Section III, we explore the problem addressed in this paper. Section IV proposes the machine learning approach for determining the number of computers required on the basis of the measured load metrics. The implementation of the proposed method is described in Section V. Section VI experimentally evaluates the feasibility and effectiveness of the proposed method. Finally, Section VII concludes the paper.

II. Related Work

Power management techniques for server clusters have been reported in [2], [8-10]. In [2], two implementations called "power-aware cluster-based network server" and "power-aware OS for clusters" are examined. Both implementations turn computers in a cluster on and off on the basis of the load. The load metrics are the utilizations of the CPU, disk interface, and network interface. These metrics are measured on each cluster node by means of the `/proc` files of the Linux OS. By setting the threshold at 90% for these utilizations, the method determines whether a computer should be turned on or off. However, the accuracy of this decision procedure is insufficient because the most appropriate threshold depends on the service content or hardware specifications.

Another power management method for a server cluster is examined in [8]. This method measures parameters including the CPU utilization, request queue length, and TCP connection rate, called "throughput." Among these parameters, the allotment of resources is determined by comparing the CPU utilization with a target value. The target value is modified if the request queue length exceeds its threshold. Using this modification, the method can flexibly set the target value that corresponds to the variations of the bottlenecked resources. However, it is not easy to find the optimal threshold for the queue length.

References [2] and [8] show the experimental results performed on prototype systems, whereas [9] and [10] report simulation studies. Of these, [9] formulates the power management method as an optimization problem by assuming the M/M/1 model for the response time. In the algorithm used in [9], the on/off decision is executed by comparing the request rate with a threshold, which is theoretically derived on the basis of the M/M/1 assumption. However, it is very unlikely that the relationship between the request rate and response time is exactly modeled by the M/M/1 system. Another formulation is found in [10], which minimizes the sum of the backlog cost, power cost, and reconfiguration cost for the current job backlog, thermal condition, and number of CPUs. To solve this problem, they presented a heuristic that determines the number of CPUs. However, the number of backlog jobs may not be an appropriate metric of the response time for the variety of jobs whose processing times are very different. It is also uncertain whether the exact number of backlog jobs is measurable in a cluster.

III. Problem Description

Most existing methods determine the computers that should be turned on or off by comparing a single or a few load metrics with predefined threshold values. However, this approach is problematic in that the optimal threshold may change depending on the requested service content. Thus, a unique threshold cannot be guaranteed to always offer good performance for any requested content. This is because the performance bottleneck of the server computer significantly depends on the requested service content. Therefore, even for the same load metric value, the performance degrades for some service contents, whereas it does not degrade for other service contents. Thus, it is inadequate to determine the on/off status of server computers by comparing the metrics and thresholds.

For example, let us assume that a web service is provided by a cluster. Then, suppose that the page data includes a set of large files and CGI scripts that require heavy computation. For this situation, consider that CPU utilization is used as the load metric. Then, the performance degradation will be accurately found by measuring the CPU utilization if most requests are directed to the CGI scripts. However, if the requests are directed to the large files, the performance will be degraded to a much smaller CPU utilization value. This happens because the bottlenecked resource is different in these cases. In the case of CGI scripts, CPU utilization may be an adequate metric because the computing power of the CPU is the bottleneck. Meanwhile, because the bottleneck is the read speed of the hard

disk for large files, the performance when obtaining large files degrades for smaller CPU utilization. Therefore, it is difficult to determine the need to add/remove a computer by comparing the CPU utilization to a threshold.

The above problem is better understood by analyzing the result of the following experiment that was performed on two PCs, with the Linux OS, connected by a 1 Gb/s Ethernet. The `apache` Web server was running on one PC, whereas `httperf` [11] was executed as a client program on the other PC. The server PC provides three file sets: 100 HTML files of size 1 KB, 100 HTML files of size 1 MB, and a PHP script that computes prime numbers and returns the result in a table format. For this configuration, the requests were generated for each file set on the client PC. From the output of `httperf`, the average time to establish a TCP connection and the average bit rate on the network interface were obtained. At the same time, the CPU utilization on the server PC was recorded by the `top` command performed in the batch mode. The PCs have a Celeron 3.06 GHz processor and 512 KB of memory.

Fig. 1 shows the relationship between the connection establishment time and CPU utilization. From the figure, it is observed that the performance degrades at a very different value of CPU utilization, depending on the file set. For the PHP script, the connection establishment time quickly increases when the CPU utilization increases to 92%. On the other hand, the connection establishment time increases for a much smaller utilization of 44% for the 1 KB file set. For the 1 MB file set, the utilization at which the degradation occurs is smaller and is 6%. This result suggests that it is impossible to determine the appropriate threshold of CPU utilization that is effective for every file set; the threshold optimized for the PHP script is very large for 1 MB files, whereas that optimized for the 1 MB files is very small for the script.

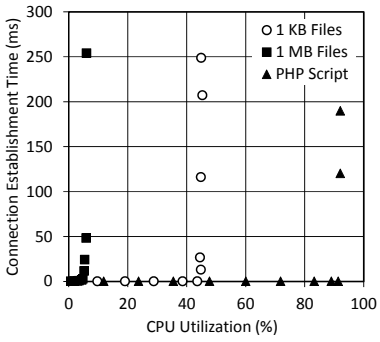

Fig. 1. Server performance versus CPU utilization.

In Fig. 2, the connection establishment time is plotted against the bit rate of the network interface. As shown in the figure, the bit rate at which the connection establishment time increases differs greatly with the file set. Therefore, it is also difficult to decide appropriate thresholds for bit rates. Suppose that the thresholds for the CPU utilization and network interface bit rate are optimized for the PHP script and the 1 MB files, respectively. Then, these thresholds may work well for the PHP script and 1 MB files. However, the thresholds are not effective if most requests are directed to the 1 KB files, because neither of the thresholds is optimal for the 1 KB files. Therefore, it is inadequate to individually set thresholds for multiple metrics.

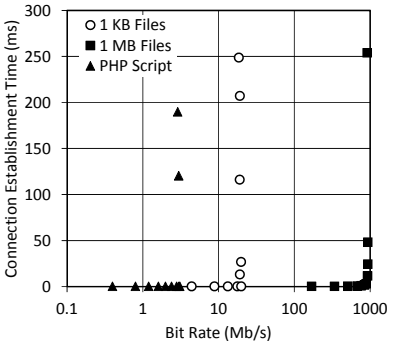

Fig. 2. Server performance versus the bit rate on the network interface.

The above results were obtained because the consumptions of different resources cause performance degradation. Fig. 1 implies that the consumption of CPU utilization limits the performance for the PHP script. Moreover, Fig. 2 suggests that the capacity of the network interface determines the performance for the 1 MB files. For the 1 KB files, it is likely that the performance is affected by the consumption of some resource other than the CPU or network interface. There are various resources that may affect the performance. These include the CPU, network interface, bus, disks, memory, buffers for TCP/IP, and request queue buffer for TCP. In addition, server software settings, for example, the maximum number of concurrently served clients (MaxClients in `apache`), may also affect the performance. Thus, for an accurate on/off decision, it is necessary to monitor multiple load metrics that reflect the consumptions of those various resources.

If multiple metrics are used, it becomes necessary to consider the complicated relationships among the metrics for the on/off decision. Using the measured metric values, the method must systematically find the computers that should be turned on or off. In addition, the mapping rule between the metric values and on/off decision must be updated against the changes in the service content and computer specifications, and it is necessary to develop an algorithm that can perform this update.

To satisfy these requirements, the machine learning approach is considered promising. Using this approach, the mapping among multiple metrics and the number of required servers is algorithmically discovered from training data. Thus, it is expected that the on/off state of the computers will be accurately determined by utilizing multiple metrics, each of which is associated with a different server bottleneck. Moreover, even if the server specifications or service content change, it is easy to update the decision rule by relearning the data created for the new environment.

IV. Proposed Method

This section explores the machine learning approach to determine the on/off state of computers by measuring multiple

load metrics.

For simplicity, this study assumes that the specifications of each computer in the cluster are identical. It is also assumed that the same volume of traffic is uniformly distributed to every computer. Strictly speaking, the power consumed on a computer depends on the load. However, the power variation caused by the load is small in comparison with that caused by turning the computer on or off. Thus, it is assumed that power consumption is determined by the number of computers currently turned on. Based on these assumptions, the on/off decision for the computers is reduced to the problem of determining how many computers should be turned on.

Hereafter, this paper considers the case where the cluster provides the World Wide Web (WWW) service. However, the proposed method is also applicable to other services.

A. Machine Learning Approach

Suppose that a system takes one of the n states $s_1, \ldots, s_n$. We can observe m attributes $a_1, \ldots, a_m$, which provide some information about the unknown current state. The value of each attribute is determined according to some probability distribution, which depends on the current state. Then, the purpose of machine learning is to build a classifier that estimates the unknown current state from the measured attribute values. Assume that we have training data that include the vectors of the actually occurred state and the attribute values measured under that state. For this situation, there exists a machine learning algorithm that builds a classifier from the training data.

This study applies the above machine learning framework to determine the optimal number of computers that should be turned on. For this application, the states are the optimal number of computers needed for a sufficiently good performance, whereas the attributes are the load metric values. By employing this technique, it becomes possible to decide the exact number of computers by considering the relationship among the load metrics. It is also possible to easily update the classifier against changes in machine specifications or service content by relearning.

There are several machine learning approaches, such as the artificial neural network, the naïve Bayes, and the Bayesian network. Among these approaches, this study examines the method based on a decision tree, which is often used in this field [3], [7]. For the program that generates the decision tree classifier, c4.5 [7] is employed. Program c4.5 has been examined in various studies in the network management field [4], [12], [13] and therefore is considered to be sufficiently reliable. It is also reported that the decision tree based method is advantageous because of its low computational cost [14]. This advantage is very important for the real time/online power management of clusters.

B. Server Load Metrics

Previous studies regarding the power management of server clusters often used a metric measured on each server computer. However, this study employs traffic parameters that are passively monitored on a computer that is separate from the server cluster. This scheme considerably simplifies the implementation because it is not necessary for the node to gather the metrics obtained at the computers in the cluster by means of polling. Another advantage is that the scheme requires neither processing on the server computers nor communication between the computers. Therefore, the measurement is robust even if the load on the server or network is excessive. Moreover, the traffic attributes are independent of the software or hardware platform used in the server cluster. This decreases the time and cost of developing the measurement software for different types of platforms.

The employed traffic attributes are as follows:

- Byte rate of the traffic sent to and received from the server cluster, R_B.
- Packet rate of the traffic sent to and received from the server cluster, R_P.
- TCP connection establishment rate, R_C.
- TCP SYN loss rate, R_S.
- Number of flows, N_F

Among the above metrics, the TCP SYN loss rate R_S is defined as follows. Let N_S be the number of TCP SYN messages sent to the server cluster during the measurement period. Moreover, let N_A be the number of SYN ACK messages sent from the server cluster, then

$$R_S = \frac{N_S - N_A}{N_A} \tag{1}$$

Metric R_s is considered to be effective for estimating the server condition [15].

The number of flows is defined as the number of TCP connections existing in the last Δt s. The method of counting flows is explored in [16]. In this study, Δt is set to 1 s and the average value over a specified time period is used.

The above metrics are relevant to the resource usage and performance. The byte (or bit) rate obviously represents the utilization of a network interface. Moreover, the number of packets that can be processed by the interface is often limited for short packets. Thus, the packet rate provides useful information that reflects the network interface performance. It is also clear that the establishment of a TCP connection consumes some of the resources of a server computer. This implies that the connection rate is related to the resource utilization. Another resource that significantly affects the performance is the TCP request queue buffer. The TCP SYN is lost when the TCP request queue buffer has overflowed. This significantly increases the response time [17]. Thus, the TCP SYN loss rate is a reliable metric that indicates the TCP request queue buffer. Finally, the number of flows is also important because an excessive number of flows causes a decreased throughput of each TCP connection.

C. Performance Criteria

The purpose of applying machine learning is to determine the optimal number of server computers required to achieve a sufficiently good performance. For this purpose, it is necessary

to clarify the criteria required for a sufficiently good performance. In this study, the server performance is considered to be sufficiently good if both of the following conditions are satisfied from the viewpoint of the WWW application:

- The average time of the TCP connection establishment is less than 1 s.
- The average bit rate achieved on the application layer from the server to a client is greater than 10 Mb/s.

These criteria were determined because of the following reasons. If the first condition is satisfied, the connection will be established much faster than the time required by a user to accept the Web page response. The satisfaction of the second condition means that the bit rate is sufficiently large to perform, for example, movie streaming with DVD quality. Thus, it is believed that the conditions are rational as performance criteria to serve a Web page that includes a movie.

In the experiment, the satisfaction of the above conditions is determined by the output of the measurement tool (`httperf`) performed on the client PC. The tool estimates the average time of the TCP connection establishment. Thus, the first condition can be examined directly from its output. The tool also shows the average download time. Thus, the bit rate of the application layer is easily calculated by the ratio of the page data size and the download time.

V. Implementation

A power management system for a small cluster was implemented to experimentally confirm the feasibility and effectiveness of the proposed approach. The functions of the system are as follows:

- Measurement of the load metrics.
- Feeding the metrics values to the c4.5 decision tree classifier.
- Counting the number of server computers to be turned off or on.
- Modification of the on/off status.

Fig. 3 illustrates the system.

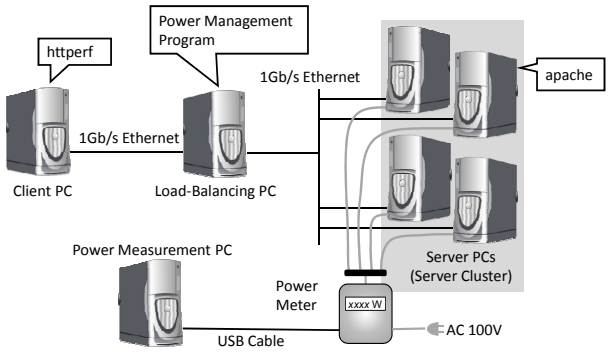

Fig. 3. Experimental configuration of the proposed method.

As shown in Fig. 3, the system consists of a load balancing PC and a server cluster. The power management program is performed on the load balancing PC. The program is implemented on the Linux OS using C language. The load is shared among server computers by setting the Destination Network Address Translation (DNAT) [18] on the load balancing machine.

The power management program executes the following procedure repeatedly:

(1) Measure the load metrics R_B, R_P, R_C, R_S, and N_F for a specified time T_m. This measurement is performed by capturing TCP/HTTP packets through the `pcap` library [19]. The metrics R_B and R_P are computed by the size and number of the packets captured during T_m. By counting the number of SYN ACK messages among the captured packets, R_C is obtained. Similarly, R_S is calculated from the numbers of TCP SYN and SYN ACK messages included in the captured packets. The average flow number N_F is computed from the flow identifier shown in the packet header and the packet arrival time every 1 s. Then, it is averaged over the period T_m. Thus, these metrics are obtained for the packets captured by the `pcap` library.

(2) Obtain the number of server computers, s, by inputting the load metrics R_B, R_P, R_C, R_S, N_F into the c4.5 decision tree.

(3) Let t denote the number of server computers that are currently turned on. If $t < s$, execute (4). If $t > s$, execute (5). Otherwise, return to (1).

(4) Turn on $s - t$ server computers that are currently sleeping. This is performed using the Wake-On-LAN mechanism. By executing `ping`, check whether the computers have been successfully turned on. After confirming the operation, modify the DNAT setting and distribute traffic to the newly added computers.

(5) Modify the DNAT setting and shut down the traffic to $t - s$ server computers that are currently on. Remotely log into these computers using `ssh` and set the `hibernate` command to be executed after T_w seconds. The time T_w is set to a sufficiently large value to avoid turning off the computer that holds the TCP connections.

The structure and flow of the power control program is depicted in Fig. 4.

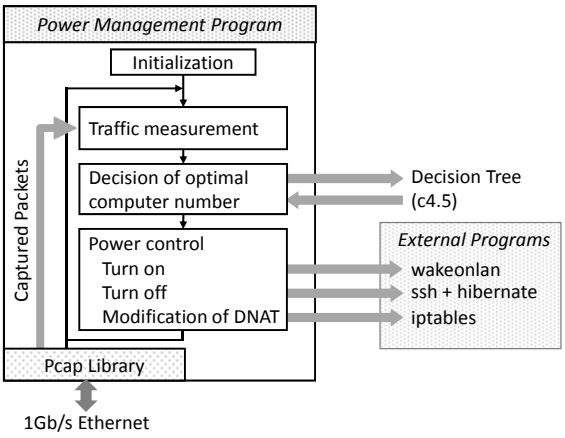

Fig. 4. Structure of the power management program.

VI. EVALUATION

A. Experiment Configuration

By using the power management program, experiments were performed for a small server cluster. As shown in Fig. 3, the experimental network is configured with four server PCs, a load balancing PC, a client PC, and a power measurement PC. A WWW server program (`apache`) runs on each server PC. On the load balancing PC, the traffic from the client is distributed to the server PCs by DNAT and the power control program is executed. The requests for page data are generated on the client PC by performing `httperf`. The output of `httperf` is used to estimate two performance parameters: the average TCP connection establishment time and the average bit rate from the server to the client. The power consumption of the server PCs is monitored by a power meter. The result is sent to the power measurement PC through the USB interface. The Linux OS runs on the PCs, with the exception of the power measurement PC on which Microsoft Windows XP is used. The server PCs have an Intel Core2Duo 2.2 GHz CPU and 2 GB of memory.

The server PCs provide six sets of page data files, as shown in Table 1. The purpose of employing different file sizes is to make various resources (e.g., hard disks and network interfaces) the performance bottlenecks. By adding the PHP script that needs extensive computations, it becomes possible to test the situation where the computational power of the CPU is the bottleneck. The training data is generated for these very different file sets with the aim being that the system will support a broad range of service requests.

TABLE 1.
FILE SETS PROVIDED BY THE SERVERS FOR TRAINING DATA.

File Sets	Content
#1	100 HTML files, size of each file: 10 KB
#2	100 HTML files, size of each file: 100 KB
#3	100 HTML files, size of each file: 1 MB
#4	100 HTML files, size of each file: 10 MB
#5	100 HTML files, size of each file: 100 MB
#6	PHP script that sends an html document with randomly selected 500000 words: each word consists of 2 characters

B. Training Data

The training data were obtained as follows. Requests for the page data are generated for each file set by executing `httperf` with the connection rate R_i ($0 \le i < 20$). The measured load metrics and the outputs of `httperf` are written to files. This procedure is repeated while changing the number of server computers from 1 to 4. For each number of server computers, the output of `httperf` is checked to see whether the performance criteria described in Section 4.C are satisfied. Let s denote the optimal number of server computers for the pair of a file set and a connection rate. Then, s is obtained as the minimum number of computers that satisfies the criteria. Let $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$ be the sets of the observed load metrics when the number of server computers is 1, 2, 3, and 4, respectively. Then, an instance of the training data is defined as the vector of ($\mathbf{x}_s$, s).

The connection rate R_i was determined as follows. First, the maximum connection rate that satisfies the criteria is found for each file set by turning on all four server computers. Let R_M be the maximum connection rate discovered with this procedure. Then, the rate R_i is defined as follows:

$$R_0 = 0.1R_M \tag{2}$$

$$R_i = R_0 + i(R_M - R_0)/20 \tag{3}$$

By setting the connection rate as above, there always exists an optimal number of computers that satisfy the performance criteria. Simultaneously, it becomes possible to examine the characteristics of the system for a considerably wide range of connection rates.

The performance of the server cluster is influenced by the connection rate and the probability distribution of the connection interval. The distribution of the connection interval is set to the exponential distribution.

The period for generating the requests is set to 10 min for the file sets. This period is determined by considering the time needed by the system to be in a stationary state. The load metrics are computed from the captured packets within 60 s. Thus, ten sets of load metrics are obtained during the request generation period. From these load metric sets, four sets obtained during the stationary state are used for the training data. The employed measurement program is identical to the load measurement part of the power management program described in Section V.

By generating the data as above, 480 data instances were obtained from six file sets, 20 connection rates, and four load metric sets. In addition to these data instances, 20 instances for no traffic load are included in the training data. This additional instance represents the condition where the number of computers should be one if there is no traffic load. Thus, the training data includes a total of 500 data instances. By inputting this training data to the c4.5 program, a decision tree classifier is constructed.

C. Experimental Results

The performance of the service and electrical power consumed by the server cluster were measured by operating the power management program. From the result, we checked whether power savings were realized while satisfying the performance criteria.

The experiments were executed while offering page requests for the file sets shown in Table 2. To confirm the robustness of the proposed method against the unlearned page requests, we used the file sets that were not used for the training data.

TABLE 2.
FILE SETS PROVIDED BY THE SERVERS FOR DYNAMIC LOAD.

File Sets	Content
#7	100 HTML files, size of each file: 30 MB
#8	100 HTML files, size of each file: 3 MB
#9	PHP script that sends an html document with randomly selected 10000 words: each word consists of 100 characters

For each file set shown in Table 2, the maximum rate R_M was first determined using the same procedure as that for obtaining the training data. The dynamic load was generated by executing `httperf` on the client PC while altering the connection rate as follows. Initially, `httperf` was started with a connection rate of $0.09R_M$. Then, `httperf` was launched every 24 min while increasing the connection rate by $0.09R_M$ on each occasion. The maximum connection rate was $0.9R_M$. After the connection rate reached the maximum, `httperf` was started every 24 min while decreasing the connection rate by $0.09R_M$ each time. Fig. 5 shows the characteristics of the load generated using this procedure.

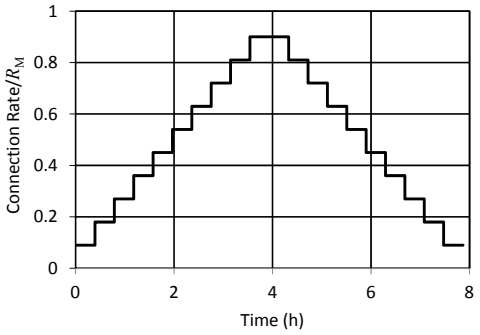

Fig. 5. Characteristics of the dynamic load.

In the experiment, the parameters T_m and T_w were set to 600 s and 120 s, respectively.

While generating the load shown in Fig. 5 to the server cluster controlled by the proposed method, the performance was measured. As a result, the performance notably degraded for the period when the load increased. This happened because of the delay associated with the measurement period T_m. That is, when the classifier estimates the number of computers, the actual current load becomes larger than that measured during the previous T_m. Thus, the number of computers is underestimated for the next period.

To prevent the above-mentioned problem, the power control program was slightly modified. Let s denote the number of computers estimated by the classifier, and let n be the total number of available server computers. Then, if $s < n$, $s + 1$ instead of s is used as the number of computers to be turned on. Because this modification provides a margin for the server capacity, it is expected that the performance degradation will be avoided.

Fig. 6 plots the number of server computers against the time for which the modified power management program was used. The figure shows the case when the connection request was given to file set #7. Fig. 6 shows that the number of computers is small for the light load period and large for the heavy load period. Thus, the figure implies that an adequate number of computers was turned on or off depending on the changes to the offered load.

Obviously, this control reduces the power consumption compared with the case where the power management is not performed. This is clearly shown in Figs. 7–9, which compare the power consumption with the proposed method and that obtained by turning on all four server computers. Figs. 7, 8, and 9 show the characteristics for file sets #7, #8, and #9, respectively. The average power consumption with the proposed method is about 64%–70% in the cases when all four server computers are turned on. This confirms that a considerable power reduction is obtainable by employing the proposed method.

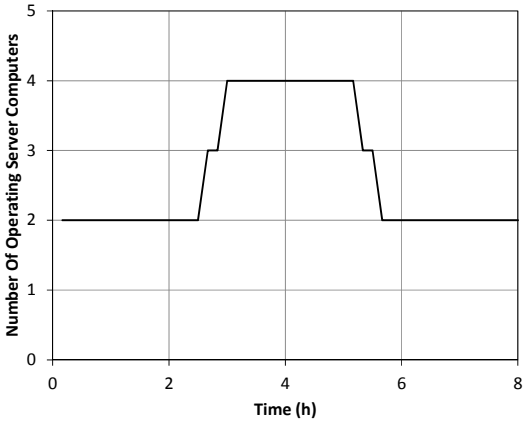

Fig. 6. Number of operating computers versus time for file set #7.

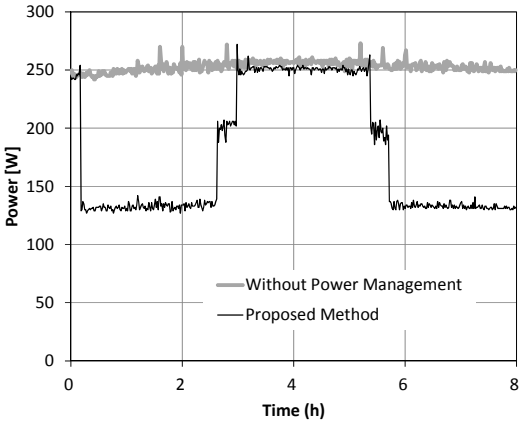

Fig. 7 Power consumptions for file set #7.

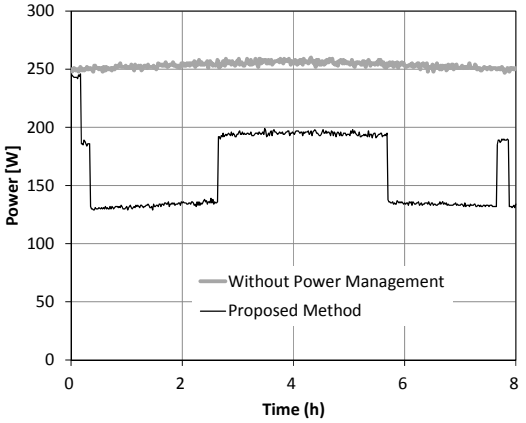

Fig. 8 Power consumptions for file set #8.

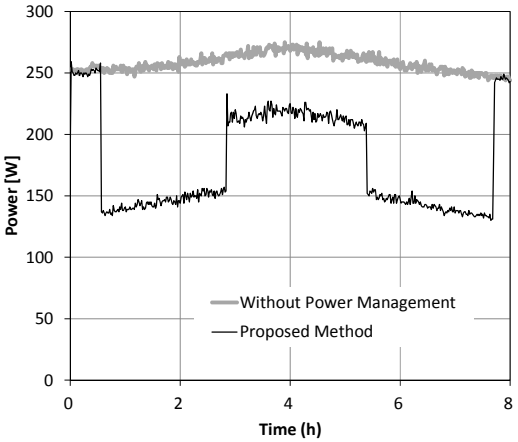

Fig. 9 Power consumptions for file set #9.

Figs. 10, 11, and 12 show the average bit rate of a TCP connection when the power management program was executed while feeding the dynamic load to the file sets #7, #8, and #9, respectively. As shown in the figures, the bit rate was greater than 10 Mb/s (which is the criterion for sufficiently good performance) in every time period. This implies that the number of server computers was correctly determined by the machine learning mechanism.

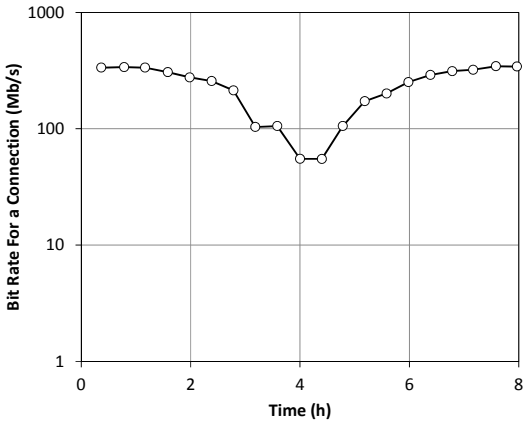

Fig. 10 Average bit rate of a connection versus time for file set #7.

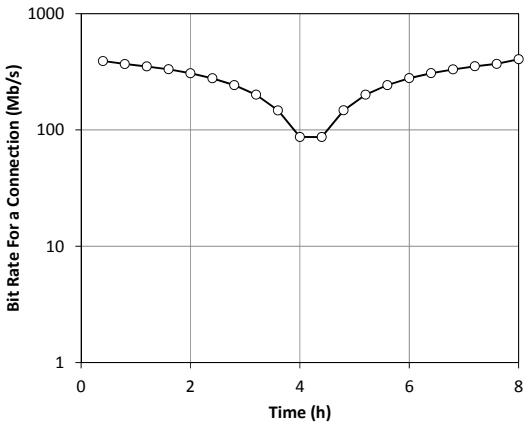

Fig. 11 Average bit rate of a connection versus time for file set #8.

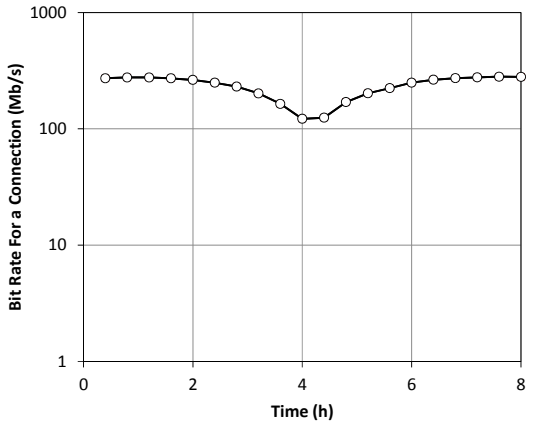

Fig. 12. Average bit rate of a connection versus time for file set #9.

Additionally, the connection establishment time is always less than 1 s. Thus, the performance was also sufficiently good for this criterion.

From the above result, it is concluded that the proposed method correctly determines the number of computers required for a sufficiently good performance and successfully reduces the power consumption of the server computers.

If the number of computers to be turned on is determined by simply comparing a metric with a threshold, it is difficult to achieve effective power management as is the case with our proposed method. For example, consider the case where the management system determines the number of server computers to be turned on by comparing the bit rate with thresholds. In this case, it is essential to know the relationship between the bit rate and number of required server computers. Figs. 13, 14, and 15 plot this relationship obtained from the training data for file sets #4, #5, and #6, respectively. These figures show that the characteristics are very different for the file sets. Let r_s denote the maximum bit rate when the number of required server computers is s. Then, r_1 is about 660 Mb/s for file set #4, whereas it is 370 Mb/s for file set #5. For file set #6, r_1 is much smaller and is about 33 Mb/s. Thus, if the threshold is determined by the characteristics for file set #4, it will be too large for file sets #5 and #6. This causes an underestimation of the number of computers and performance degradation if clients request mainly the data of file sets #5 and #6. Moreover, if the threshold is determined according to the characteristics for file set #6, it will be too small for file sets #4 and #5. Thus, the number of computers will be overestimated, and the power will not be adequately reduced for file sets #4 and #5. As shown by these characteristics, it is impossible to determine an optimal threshold that always provides good results for different file sets. In contrast, the proposed method offers a considerable reduction in the power consumption and a sufficiently good performance for different file sets. This confirms the need for and the advantage of the proposed method.

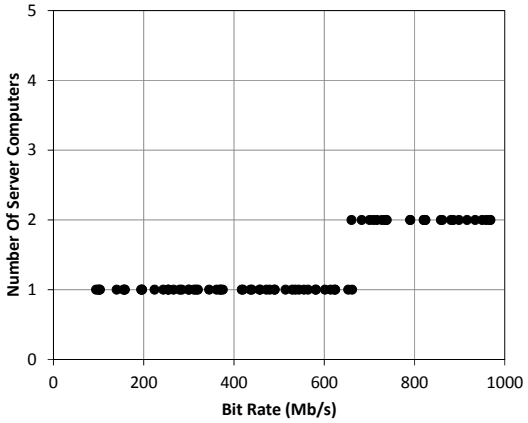

Fig. 13 Relationship between the bit rate and number of server computers to be turned on for file set #4.

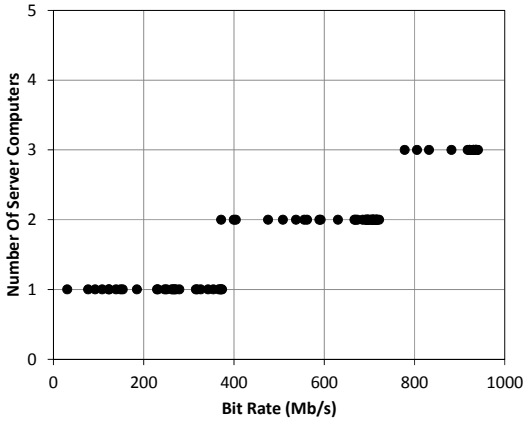

Fig. 14 Relationship between the bit rate and number of server computers to be turned on for file set #5.

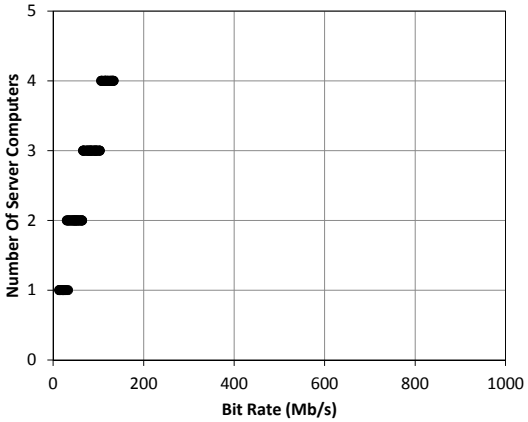

Fig. 15 Relationship between the bit rate and number of server computers to be turned on for file set #6.

VII. Conclusion

This paper proposed a power management method for server clusters, which provide, for example, the WWW service over the Internet. The proposed method employs a machine learning approach that extracts the information required for the on/off decision from passively measured traffic metrics. The proposed method is advantageous because it correctly performs the on/off decision for a wide range of requested service contents by discovering the shortages or excesses of resource consumptions from multiple traffic metrics. An additional feature of the proposed method is that the traffic metrics are measured passively. Thus, the method is nonintrusive to the server cluster computers and is independent of the hardware or software platform. This characteristic is advantageous from the perspective of robustness and compatibility.

The paper clarified the method for estimating the number of server computers to be turned on based on the passively measured traffic metrics while using the machine learning technique. The power management program based on this concept was implemented on a PC with a Linux OS for a small cluster while assuming the WWW service. The feasibility and effectiveness of the proposed method was experimentally evaluated. The results show that for different types of page datasets, the method successfully reduces the power consumption without degrading the service performance.

References

[1] R. Bianchini and R. Rajamony, "Power and energy management for server systems," *Computer*, 37, 11, Nov. 2004, pp. 68-74.

[2] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," in *Proc. Workshop on Compilers and Operating Systems for Low Power*, Barcelona, Spain, 2001.

[3] I. H. Witten, E. Frank, and M. A. Hall, *Data mining: practical machine learning tools and techniques - 3rd edition*. Burlington, MA: Morgan Kaufmann, 2011.

[4] S. Ohta, R. Kurebayashi, and K. Kobayashi, "Minimizing false positives of a decision tree classifier for intrusion detection on the Internet," *Journal of Network and Systems Management*, 16, 4, Aug. 2008, pp. 399-419.

[5] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, 10, 5, Sep. 1999, pp. 1048-1054.

[6] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *Proc. LCN'05*, Sydney, 2005, pp. 250-257.

[7] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.

[8] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *Proc. 18th Symposium on Operating Systems Principles*, Banff, Canada, 2001, pp. 103-116.

[9] L. Wang and Y. Lu, "Efficient power management of heterogeneous soft real-time clusters," in *Proc. 2008 Real-Time Systems Symposium*, Barcelona, Spain, 2008, pp. 323-332.

[10] L. Mastroleon, N. Bambos, C. Kozyrakis, and D. Economou, "Autonomic power management schemes for internet servers and data centers," in *Proc. IEEE Globecom 2005*, St. Louis, MO, USA, 2005, pp. 943-947.

[11] D. Mosberger and T. Jin, "httperf – A tool for measuring web server performance," *ACM SIGMETRICS Performance Evaluation Review*, 26, 3, Dec. 1998, pp. 31-37.

[12] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive Bayes vs decision trees in intrusion detection systems," in *Proc. SAC '04*, Nicosia, Cyprus, 2004, pp. 420-424.

[13] W. Li and A. W. Moore, "A machine learning approach for efficient traffic classification," in *Proc. MASCOTS'07*, Istanbul, Turkey, 2007, pp. 310-317.

[14] S. Marsland, *Machine Learning – An Algorithmic Perspective*, CRC Press, 2009.
[15] S. Ohta and R. Andou, "WWW server load balancing technique employing passive measurement of server performance," *ECTI Trans. EEC*, 8, 1, Feb. 2010, pp. 59-66.
[16] S. Zhu and S. Ohta, "Real-time flow counting in IP networks: strict analysis and design issues," *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, 3, 2, Feb. 2012, pp. 7-17.
[17] C.-H. Tsai, K. G. Shin, J. Reumann, and S. Singhal, "Online web cluster capacity estimation and its application to energy conservation," *IEEE Trans. on Parallel and Distributed Systems*, 18, 7, July 2007, pp. 932-945.
[18] R. Russell. (2002, January). Linux 2.4 NAT HOWTO, Available: http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html
[19] TCPDUMP/LIBPCAP Repository, Available: http://www.tcpdump.org/

Satoru Ohta received the B.E., M.E., and Dr. Eng. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 1981, 1983, and 1996, respectively.

In 1983, he joined NTT, where he worked on the research and development of cross-connect systems, broadband ISDN, network management, and telecommunication network planning. Since 2006, he has been a professor in the Department of Information Systems Engineering, the Faculty of Engineering at Toyama Prefectural University, Imizu, Japan. His current research interests are network performance evaluation, power management of network systems, algorithms for sensor networks, and management of virtual machines.

Dr. Ohta is a member of the IEEE, IEICE, and ECTI. He received the Excellent Paper Award in 1991 from IEICE.

Takehito Hirota received the B.E. and M.E. degrees from Toyama Prefectural University, Toyama, Japan, in 2009 and 2011. He joined Hokuriku Tsushin Kogyo Co. in 2011. While he was with Toyama Prefectural University, he was engaged in the study on the power management of server clusters.