A Simply Energy-efficient Migration Algorithm of Processes with Virtual Machines in Server Clusters

Ryo Watanabe [1*], Dilawaer Duolikun[1], Tomoya Enokido[2], and Makoto Takizawa[1]
[1]*Hosei University, Koganei, Tokyo 1848584 Japan*
ryo.watanabe.4h@stu.hosei.ac.jp, dilewerdolkun@gmail.com, makoto.takizawa@computer.org
[2]*Rissho University, Shinagawa, Tokyo 1418602 Japan*
eno@ris.ac.jp

Abstract

Virtual services on computation resources like CPUs and storages are supported to applications by using virtual machines in server clusters. Virtual machines with application processes can migrate from host servers to guest servers. In this paper, we discuss a process migration approach to reducing total electric energy consumption of servers by taking advantage of virtual machines in a cluster. We newly propose a Simple Globally-Energy-Aware Migration (SGEAM) algorithm. Here, a host server is first found to perform a process issued by an application in a cluster and then the process is performed on a virtual machine. Next, a virtual machine migrates from a host server to another guest server in order to reduce the electric energy consumption of servers. Here, the amount of computation to be performed by current processes on a virtual machine is simply estimated only by using the number of the current processes and then the total electric energy to be consumed by each server is estimated. Then, a guest server where a virtual machine migrates from a host server is selected so that the total electric energy to be consumed by not only the host server and guest server but also the other servers can be reduced. We show the total electric energy consumption of servers can be reduced in the SGEAM algorithm compared with other non-migration and migration algorithms in the evaluation.

Keywords: Computation model, Virtual machine migration, SGEAM algorithm, Simple estimation model, Power consumption model.

1 Introduction

Information systems are getting scalable and huge amount of electric energy is consumed by servers in scalable clusters [1]. We have to reduce electric energy consumed by servers in scalable clusters like cloud computing systems [1] in order to realize eco society [2]. There are approaches to reducing the electric energy consumption of servers [3, 4, 5, 6, 7, 8, 9, 10]. One is a hardware-oriented approach. Here, energy-efficient hardware devices like CPUs [11] are developed and used. Another approach is the macro-level approach [4, 5, 12] where we try to reduce total electric energy consumed by servers to perform application processes issued by clients. Types of power consumption models are proposed, which give electric power [W] to be consumed by a server to perform application processes. In one approach, one energy-efficient server is selected in a cluster to perform a process issued by a client in types of server selection algorithms [13, 12, 14]. In another migration approach [15], a process on a host server migrates to an energy-efficient guest server which is expected to consume smaller electric energy than the host server. However, it is not easy to migrate various types of application processes among heterogeneous servers with different architectures and operating systems. Cloud computing systems

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 8:2 (June 2017), pp. 1-18
*Corresponding author: Graduate School of Science and Engineering, Hosei University, 3-7-2 Kajino-cho, Koganei-shi, Tokyo, 1848584, Japan, E-mail: ryo.watanabe.4h@stu.hosei.ac.jp, Tel: +81-42-387-6218

support applications with virtual computation service by using virtual machines [16]. A virtual machine can migrate from a host server to another guest server while processes are being performed on the virtual machine, i.e. live migration [16]. In our previous studies [15], if a process is issued to a cluster, a virtual machine is selected to perform the process by estimating the termination time of each current process performed with other processes. Then the electric energy consumption of each server to perform every current process and the new process is estimated. If a host server consumes more electric energy than expected, a virtual machine migrates from the host server to another guest server which is expected to consume the smaller electric energy than the host server. The termination time of each process on each server is estimated by using the computation laxity of the process and computation rate of the server. However, it is not easy to collect information of processes like computation laxites on each server. It also takes time to do the estimation of the termination time of each process by using the information on processes and virtual machines.

In papers [17], the simple virtual machine migration (SVM) and modified SVM (MSVM) algorithms are proposed to reduce the computation complexity to do the estimation. Here, the total amount of computation of a virtual machine is given as the summation of computation laxites of each resident processes on the virtual machine. Then, the termination time of each process on a virtual machine is simply estimated by considering the amount of computation of each virtual machine and without considering the termination time of each process. In various types of applications, it is not easy, maybe impossible to obtain the amount of computation of each process. In this paper, the same amount VC of computation is assumed to be performed by each process. Then, we newly propose a simple method to estimate the termination time of processes only by using the number of current processes performed on each server. We also propose a *Simple Globally-Energy-Aware Migration* (*SGEAM*) algorithm to not only find a virtual machine to perform a process issued by a client but also to migrate a virtual machine to another server so that the total electric energy consumption of the servers can be reduced by using the simple estimation model in this paper.

We evaluate the SGEAM algorithm compared with other non-migration and migration algorithms. The SGEAM algorithm is computationally simple since only number of current processes on each server is used to estimate the electric energy consumption of servers. We show the total electric energy consumption and active time of servers and the average execution time of processes can be more reduced in the migration algorithms than the non-migration algorithms. Especially, the electric energy consumption of servers is mostly reduced in the SGEAM algorithm.

In section 2, we overview related studies. In section 3, we present a system model. In section 4, we present the power consumption and computation models. In section 5, we discuss how to simply estimate the electric energy consumption of each server. In section 6, we propose the SGEAM algorithm. In section 7, we evaluate the SGEAM algorithm.

2 Related Studies

It is critical to reduce the electric energy consumption of information systems, especially servers in scalable clusters [1, 2]. There are approaches to reducing the electric energy consumption of servers in a cluster. One approach is the hardware-oriented approach where energy-efficient hardware components like CPUs [11] and storages (SSD) [18] are developed and used in various types of computers like servers. The electric power consumption of a server depends on not only hardware devices but also software components, especially application processes. Hardware devices of a server consume electric energy by performing application processes. In our macro-level approach [19, 4], we aim at reducing total electric energy consumed by a whole server to perform application processes without considering the electric energy consumption of each hardware device. In order to design, implement, maintain,

and evaluate energy-efficient information systems, we first have to define a formal power consumption model of a server which gives electric power [W] to be consumed by the server to perform application processes for one time unit. We first measure electric power which each server consumes to perform types of application processes like computation, storage, and communication types of application processes [4] using the UWMeter [20] by which the electric power consumption of a server can be measured every 100 [msec]. By abstracting parameters which mostly dominate the electric power consumption of a server, types of power consumption models are proposed. In the SPC (simple power consumption) model [19, 4, 12], a server s_t consumes the maximum electric power $maxE_t$[W] if at least one process is performed, otherwise the minimum electric power $minE_t$. Thus, the electric power consumed by a server s_t to perform application processes is either maximum electric power $maxE_t$ or minimum electric power $minE_t$. A server with a one-core CPU follows the SPC model. A server is equipped with one or more than one multi-thread CPU. In the MLPC (multi-level power consumption) model [14, 9] and MLPCM (MLPC model of a server with Multiple CPUs) model [21, 10], the electric power consumption of a server to perform application processes depends on the numbers of active CPUs, active cores, and active threads. The types of power consumption models of a server are also proposed to perform communication [19] and storage [22, 23, 14] types of application processes. In the power consumption model for communication processes [19], the electric power of a server depends on the total transmission rate at which the server transmits data to clients. The computation models of a server to perform application processes [19, 4, 21, 14, 10, 9] are also proposed, which give how long it is expected to take to perform a process concurrently with other processes on the server. In the MLPCMS (MLPCM model for storage processes) [24], the power consumption of a server the perform storage type of application processes which manipulate data in storage like HDD is given as summation of the power consumption given by MLPCM model and electric power consumed by storage.

In order to reduce the electric energy consumption of servers, types of algorithms [4, 12, 6, 21, 14, 10, 9] are proposed to select a server in a cluster to perform an application process issued by a client. The termination time of each current application process is estimated by simulating the execution of every current application process on a server. A server to perform an application process is selected based on a power consumption model so that the total electric energy consumption of the servers can be reduced in a cluster. In the LEA (locally-energy-aware) algorithm [10], a server which is expected to consume the minimum electric energy to perform the new application process and every current application process. In the GEA (globally energy-aware) [10] and simple GEA (SGEA) [25] algorithms, the total electric energy to be consumed by not only a host server of the new process but also the other servers is tried to be reduced.

A process migration approach is also discussed to reduce the electric energy consumption of servers in a cluster [26, 3, 15, 27]. Here, a process on a host server migrates to another server if the host server is expected to consume more electric energy than expected, for example, because the server is overloaded. The energy-efficient replication and migration models of an application process are also discussed to not only increase the reliability and availability of the system but also reduce the electric energy consumption of the servers [28]. The more number of replicas of an application process are performed, the more reliable and available but the more amount of electric energy is consumed since replicas are performed on the more number of servers. The electric energy consumed by servers to perform replicas is reduced by differentiating the start time of each replica [28]. However, it is not easy to migrate types of processes to servers with various types of architectures and operating systems.

Virtual machines are widely used to support applications with virtual computation service in a cluster of servers like KVM [16] and VMware [29]. Here, applications use computation resources like CPUs and storages like HDDs of servers by using virtual machines independently of what computation resources are supported by what servers in a cluster. The average execution time of current application processes depends on the number of the current processes but is independent of the number of virtual machines

on a server [30]. Furthermore, virtual machines with application processes can easily migrate to guest servers independently of architectures and operating systems of servers without suspending processes. Especially, virtual machines with processes can migrate to servers in the live migration [16]. In this paper, we discuss how to migrate processes to other servers by using the migration technologies of virtual machines in order to reduce the electric energy consumption of servers.

The other approach is the shut-down approach [31]. As the number of application processes performed in a cluster decreases, some server might get idle. Even idle servers consume electric energy as presented in papers [4, 5, 12]. In server clusters, especially data centers, servers which are not required to perform application processes, for example, idle servers where no process is performed are shut down to reduce the total electric energy consumption of servers in the clusters. Here, servers are restarted if more number of servers are required to handle application processes depending on the number of current application processes as discussed in paper [31]. This approach is useful in the client-server model like cloud computing systems and data centers where all the servers are controlled in a centralized manner. In this paper, we consider a distributed system where each server is autonomous and there is no centralized coordinator like peer-to-peer (P2P) model [32, 4]. Here, it is not easy to shut down and restart servers since we have to do the negotiation with owners or administrators of each server. In our approach, we discuss how to select an energy-efficient server in a cluster to perform an application process issued by a client and to migrate virtual machines to energy-efficient servers. We do not discuss how to shut down and restart servers in this paper.

3 System Model

A cluster S is composed of servers $s_1, \ldots, s_m$ ($m \geq 1$) and supports applications on clients with virtual computation service on computation resources by using virtual machines $vm_1, \ldots, vm_v$ ($v \geq 1$) like KVM [16]. Here, applications can use computation resources like CPUs and storages in a cluster without being conscious of which servers support what computation resources. If a client issues a request to a cluster S, one virtual machine vm_h is selected. An application process p_i to handle the request is created. The application process p_i is performed on the virtual machine vm_h and the virtual machine vm_h sends a response to the client on termination of the process.

In this paper, a *process* means an application process which uses CPU resource and which is performed on a server. We assume every process on a server s_t is performed on some virtual machine resident on the server s_t. A server s_t is composed of np_t (≥ 1) homogeneous CPUs. Each CPU is composed of cc_t (≥ 1) homogeneous cores. There are nc_t ($= np_t \cdot cc_t$) cores in the server s_t. Each core supports the same number ct_t (≤ 2) of threads. The total number nt_t ($= np_t \cdot cc_t \cdot ct_t$) of homogeneous threads on a server s_t are supported to applications. An *active* thread is a thread where at least one process is performed. An *active* core, CPU, and server are ones where at least one thread is active, i.e. at least one process is performed. A virtual machine is allocated with some number of threads in a server.

A process p_i performed on a virtual machine vm_h is a *resident* process of the virtual machine vm_h. A virtual machine vm_h is *active* iff (if and only if) at least one process is performed on the virtual machine vm_h. Otherwise, the virtual machine vm_h is *idle*. In an active server s_t, at least one process is performed on a resident virtual machine. A server which hosts at least one resident virtual machine is *engaged*. A server where there is no resident virtual machine is $free$. $VCP_h(\tau)$ indicates a set of resident processes performed on the virtual machine vm_h at time τ. Time when a virtual machine vm_h gets idle is *idled* time. Time when an idle virtual machine gets active is *activated* time. A virtual machine vm_h on a host server s_t is a *resident* virtual machine on the host server s_t. $SVM_t(\tau)$ denotes a set of resident virtual machines on a server s_t at time τ. A virtual machine vm_h is *smaller* than a virtual machine vm_k at time τ iff $|VCP_h(\tau)| < |VCP_k(\tau)|$, i.e. more number of process are performed on the virtual machine vm_k than

the virtual machine vm_h. A virtual machine vm_h is the smallest iff the number of resident processes is the fewest in a server s_t. Let $CP_t(\tau)$ be a set of processes performed on a server s_t at time τ. In this paper, we assume every process is on a virtual machine.

A virtual machine vm_h can migrate from a host server s_t to another guest server s_u while processes are being performed without suspending the processes, i.e. live migration [16]. First, memory area of a virtual machine vm_h on a host server is created on another guest server s_u. On issuing a migration command [16] on the host server s_t, the memory state of a virtual machine vm_h is first transferred to the guest server s_u while resident processes are being performed on the host server s_t. On termination of the state transfer to the guest server s_u, the processes are resumed on the virtual machine vm_h and the state of virtual machine vm_h changed after the state is transferred to the guest server s_u. Then, the processes on the virtual machine vm_h are restarted on the server s_u.

4 Power Consumption and Computation Models

The electric power consumption $E_t(\tau)$ of a server s_t to perform processes is $minE_t + ap_t(\tau) \cdot bE_t + ac_t(\tau) \cdot cE_t + at_t(\tau) \cdot tE_t$[W] where $ap_t(\tau)(\leq np_t)$, $ac_t(\tau)(\leq nc_t)$, and $at_t(\tau)(\leq nt_t)$ are numbers of active CPUs, active cores, and active threads, respectively, of the server s_t at time τ [10]. The electric power consumption $NE_t(n)$ [W] of a server s_t to concurrently perform n processes at time τ is given in the MLPCM model as follows [14, 10]:

$$NE_t(n) = \begin{cases} minE_t & \text{if } n = 0. \\ minE_t + n \cdot (bE_t + cE_t + tE_t) & \text{if } 0 \leq n \leq np_t. \\ minE_t + np_t \cdot bE_t + n \cdot (cE_t + tE_t) & \text{if } np_t < n \leq nc_t. \\ minE_t + np_t \cdot bE_t + nc_t \cdot cE_t + n \cdot tE_t & \text{if } nc_t < n \leq nt_t. \\ maxE_t (= minE_t + np_t \cdot bE_t + nc_t \cdot cE_t + nt_t \cdot tE_t) & \text{if } n > nt_t. \end{cases} \quad (1)$$

The electric power consumption $E_t(\tau)$ [W] of a server s_t is assumed to be $NE_t(n)$ for number n $(= |CP_t(\tau)|)$ of current processes at time τ. If $n \geq nt_t$, every thread is active and a server s_t consumes the maximum electric power $maxE_t$, i.e. $NE_t(n) = maxE_t$. An idle server s_t consumes the minimum electric power $minE_t$, i.e. $NE_t(0) = minE_t$. For example, in a server DL360p Gen8 of two Intel CPUs Xeon E5-2667 v2 [11], $minE_t = 126.1$, $bE_t = 30$, $cE_t = 5.6$, $tE_t = 0.6$, $maxE_t = 301.1$ [W]. Here, each the two CPUs supports eight cores and each core supports two threads, i.e. $np_t = 2$, $nc_t = 16$, and $nt_t = 32$.

It takes T_{ti} time units [tu] to perform a process p_i on a thread of a server s_t. If only a process p_i is exclusively performed on a server s_t without any other process, the execution time T_{ti} of the process p_i is minimum $minT_{ti}$. In a cluster S of servers $s_1, \ldots, s_m$ $(m \geq 1)$, $minT_i$ shows a minimum one of $minT_{1i}$, $\ldots$, $minT_{mi}$. If $minT_{fi} = minT_i$, a thread of a server s_f is $fastest$ in a cluster S. A server s_f with a fastest thread is $fastest$.

We assume one virtual computation step [vs] is performed on a thread of a fastest server s_f for one time unit [tu], i.e. the maximum thread computation rate $maxCRT_f = 1$ [vs/tu]. The total number VC_i of virtual computation steps to be performed by a process p_i is defined to be $minT_i$ [tu] $\cdot$ $maxCRT_f$ [vs/tu] = $minT_i$ [vs]. The maximum computation rate $maxCR_{ti}$ of a process p_i on a server s_t is VC_i / $minT_{ti}$ [vs/tu] (≤ 1). For a fastest server s_f, the maximum computation rate $maxCR_{fi}$ of a process p_i is VC_i / $minT_{fi} = minT_i$ / $minT_i = 1$ [vs/tu]. On a thread of a server s_t, $maxCR_{ti} = maxCR_{tj} = maxCRT_t$ [vs/tu] for every pair of processes p_i and p_j.

The server computation rate $CR_t(\tau)$ of a server s_t of time τ is $at_t(\tau) \cdot maxCRT_t$ where $at_t(\tau)(\leq nt_t = np_t \cdot cc_t \cdot ct_t)$ threads are active. The maximum server computation rate $maxCR_t$ [vs/tu] (≤ 1) of a server

s_t is the summation $nt_t \cdot maxCRT_t$ of maximum thread computation rates of nt_t threads. As presented here, $at_t(\tau) = n$ if $n \leq nt_t$ and $at_t(\tau) = nt_t$ if $n > nt_t$ where $n = |CP_t(\tau)|$. The server computation rate $NCR_t(n)$ is equally allocated to each current process p_i at each time τ. Suppose a pair of processes p_i and p_j are performed and totally n processes are concurrently performed on a server s_t at time τ. Here, the process computation rates $NCR_{ti}(n)$ and $NCR_{tj}(n)$ of the processes p_i and p_j, respectively, are the same. The server computation rate $NCR_t(n)$ [vs/tu] of a server s_t to concurrently perform n processes is given according to the server computation rate $CR_t(\tau)$ as follows:

$$NCR_t(n) = \begin{cases} n \cdot maxCRT_t & \text{if } n \leq nt_t. \\ maxCR_t & \text{if } n > nt_t. \end{cases} \quad (2)$$

The process computation rate $NCR_{ti}(n)$ of each process p_i performed concurrently with $(n-1)$ processes on a server s_t is $NCR_t(n)$ / n:

$$NCR_{ti}(n) = \begin{cases} maxCRT_t & \text{if } n \leq nt_t. \\ maxCR_t \,/\, n & \text{if } n > nt_t. \end{cases} \quad (3)$$

The number $NCR_{ti}(n)$ of virtual computation steps of a process p_i are performed for one time unit where n processes are concurrently performed. Figure 1 shows the process computation rates $NCR_{ti}(n)$ and NCR_{ui} of a process p_i on a pair of servers s_t and s_u, respectively. The server s_t supports n_t threads and the server s_u supports nt_u $(= 4 \cdot nt_t)$ threads where the maximum thread computation rate $maxCRT_t$ of the server s_t is double of the server s_u, i.e. $maxCRT_t = 2 \cdot maxCRT_u$. The straight and dotted lines show the process computation rates $NCR_{tu}(n)$ and $NCR_{ui}(n)$, respectively. The process computation rate $NCR_{ti}(n)$ on the server s_t is two times longer than $NCR_{ui}(n)$ of the server s_u for $n \leq n_t$. The process computation rate $NCR_{ti}(n)$ decreases with $nt_t \cdot maxCRT_t/n$ for $n > n_t$ and gets equal to $NCR_{ui}(n)$ for $n = 2 \cdot nt_t$. On the other hand, the process computation rate $NCR_{ui}(n)$ is a constant $maxCRT_u$ for $n \leq nt_u (= 4 \cdot nt_t)$ and decreases with the rate $nt_u \cdot maxCRT_u (= 2 \cdot nt_t \cdot maxCRT_t)/n$ for $n > 4 \cdot nt_t$. $NCR_{ui}(n) > NCR_{ti}(n)$ for $n > 2 \cdot nt_t$.

Suppose a process p_i starts at time st and ends at time et. Here, $\sum_{\tau=st}^{et} CR_{ti}(\tau) = VC_i$ [vs] $(= maxCRT_f \cdot minT_i = minT_i)$ where s_f is the fastest server. VC_i shows the total amount of computation to be performed by a process p_i. The computation laxity $lc_{ti}(\tau)$ [vs] is the number of virtual computation steps [vs] to be performed in the process p_i on a server s_t after time τ. At time τ a process p_i starts, $lc_{ti}(\tau) = VC_i$. Then, the computation laxity $lc_{ti}(\tau)$ is decremented by the process computation rate $CR_{ti}(\tau)$ at each time τ, i.e. $lc_{ti}(\tau + 1) = lc_{ti}(\tau) - CR_{ti}(\tau)$ at each time τ. If $lc_{ti}(\tau + 1) \leq 0$, the process p_i terminates at time τ.

The process computation rate $NCR_{ti}(n)$ of each process p_i on a server s_t depends on the number n of processes performed concurrently with the process p_i. This means, the process computation rate $NCR_{ti}(n)$ is independent of the number of resident virtual machines on the server s_t [30].

5 Simple Estimation of Process Termination Time

A client issues a request process p_i to a set VM of virtual machines $vm_1, \ldots, vm_v$ $(v \geq 1)$ in a cluster S of servers $s_1, \cdots s_m$ $(m \geq 1)$. First, one host virtual machine vm_h in the set VM is selected to perform the process p_i. Furthermore, each virtual machine vm_h can migrate from a host server s_t to a guest server s_u [16]. Here, n is total number $|CP_t(\tau)|$ of current processes of a server s_t at time τ. As presented in the preceding section, the computation laxity $lc_{ti}(\tau)$ of each process p_i is initially VC_i. The computation laxity $lc_{ti}(\tau)$ is decremented by the process computation rate $NCR_{ti}(n)$ at each time τ. If $lc_{ti}(\tau) -$

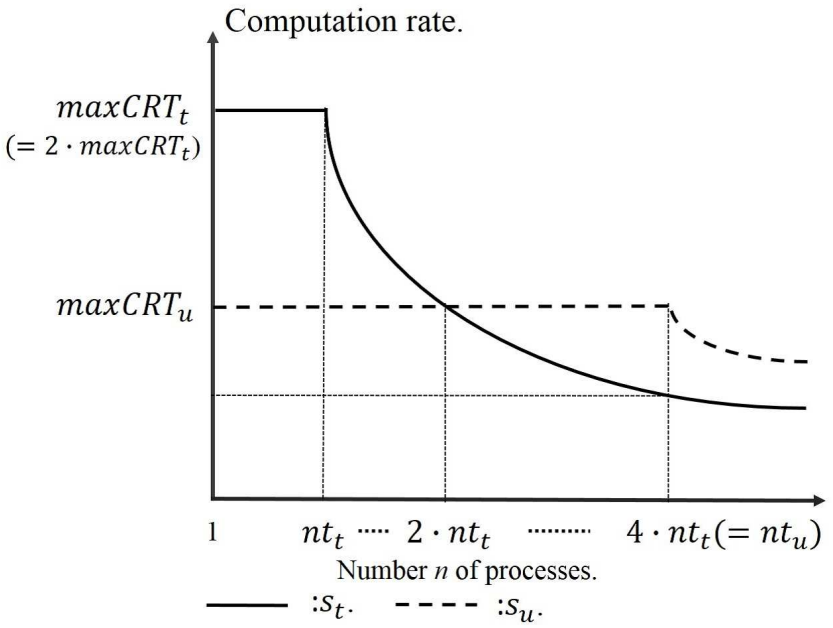

Figure 1: Process computation rate.

$NCR_{ti}(|CP_t(\tau)|) \leq 0$, the process p_i terminates at time τ. By using this model, we can estimate when every current process to terminate.

In well-formed applications, only fixed types of processes are issued in a cluster. Here, we can get the total amount VC_i of virtual computation steps of each process p_i which is the minimum execution time $minT_i$ obtained by performing the process without any other process on the fastest server s_f. However, it is not easy, maybe impossible to get the total amount VC_i of computation of each process p_i in various types of applications. It also takes time to estimate the termination time of each current process p_i on every virtual machine vm_h of a server s_t by using the computation laxity $lc_{ti}(\tau)$ and the process computation rate $NCR_{ti}(n)$ as discussed in papers [15, 14, 10, 9, 33]. In order to simplify the estimation, we assume each current process p_i ($\in VCP_h(\tau)$) on a virtual machine vm_h finishes the half of the total computation VC_i by time τ. Here, the total virtual computation steps $slc_t(\tau)(= \sum_{p_i \in CP_t(\tau)} VC_i/2)$ have to be furthermore performed on a server s_t at time τ. The expected termination time ET_t [tu] of a server s_t is given as follows:

$$\begin{aligned} ET_t &= slc_t(\tau)/NCR_t(n) \\ &= \sum_{p_i \in CP_t(\tau)} (VC_i/2)/NCR_t(n). \end{aligned} \quad (4)$$

This means, it is expected to take ET_t time units [tu] to perform every current resident process on a host server s_t. The expected electric energy consumption EE_t [W tu] of a server s_t to perform n processes at time τ is obtained by multiplying the execution time ET_t [tu] by the electric power consumption $NE_t(n)$ [W] (formula (1)) to be given as follows:

$$\begin{aligned} EE_t &= ET_t \cdot NE_t(n) \\ &= (NE_t(n)/NCR_t(n)) \cdot slc_t(\tau). \end{aligned} \quad (5)$$

Here, $NE_t(n)/NCR_t(n)$ [W tu/vs] indicates the electric energy consumption of a server s_t to perform one virtual computation step.

Next, suppose one new process p_i starts on a server s_t at time τ. The expected termination time NET_t and expected electric energy consumption NEE_t of a server s_t to perform both n current processes and the new process p_i are given as follows:

$$\begin{aligned} NET_t &= (slc_t(\tau) + VC_i)/NCR_t(n+1). \quad (6) \\ NEE_t &= NET_t \cdot NE_t(n+1) \\ &= (slc_t(\tau) + VC_i) \cdot NE_t(n+1)/NCR_t(n+1). \quad (7) \end{aligned}$$

It is difficult to obtain the total amount VC_i of virtual computation steps of each process p_i in types of applications. In this paper, we assume VC_i to be a constant, $VC_i = 1$ for simplicity. That is, every process p_i is assumed to have same amount of virtual computation steps. The simplified server laxity $sslc_h(\tau)$ of a server s_t is just defined to be the half of the number n (=$|CP_t(\tau)|$) of current processes which are performed at time τ, i.e. $sslc_t(\tau) = n/2$.

Suppose n processes are performed on a server s_t and k new processes are issued to the server s_t. Here, the total amount of computation to perform n current processes is n / 2 since the half of the computation of each current process is assumed to finish. The total computation to perform k new processes is k since the whole computation of each of the k new processes has to be performed. Here, totally (n /2 + k) computation has to be performed on the server s_t. The server computation rate of the server s_t is $NCR_t(n+k)$ since $(n+k)$ processes are concurrently performed. The expected termination time $SET_t(n, k)$ [tu] of a server s_t to perform n current processes and k new processes is given as follows:

$$SET_t(n, k) = (n/2+k)/NCR_t(n+k). \quad (8)$$

The expected electric energy consumption $SEE_t(n, k)$ [W tu] of a server s_t to perform n current processes and k new processes is give as follows:

$$\begin{aligned} SEE_t(n, k) &= SET_t(n,k) \cdot NE_t(n+k) \\ &= (n/2+k) \cdot NE_t(n+k)/NCR_t(n+k). \quad (9) \end{aligned}$$

6 Energy-efficient Migration of Virtual Machines

6.1 Virtual Machine Selection (VMS) Algorithm

We consider a cluster S of servers $s_1, \ldots, s_m$ ($m \geq 1$).We assume a set of virtual machines $vm_1, \ldots, vm_v$ ($v \geq 1$) are supported to applications in the cluster S. First, suppose a client issues a process p_i to the cluster S at time τ. A pair of variables n_t and nv_h show the numbers $|CP_t(\tau)|$ and $|VCP_h(\tau)|$ of processes performed on each server s_t and each virtual machine vm_h, respectively, at time τ. As discussed in the SGEA algorithm [25], not only a host server s_t to perform a new process p_i but also the other servers consume electric energy. For example, even if another server s_u is idle, the minimum electric power $minE_u$ is consumed by the server s_u as shown in formula 1. Hence, we take into consideration the electric energy consumed by not only a host server s_t to perform a new process p_i but also the other servers where just current processes are performed.

In this paper, we newly propose an SGEA migration (SGEAM) algorithm. The SGEAM algorithm is composed of virtual machine selection (VMS) and virtual machine migration (VMM) algorithms. Suppose n_t (≥ 0) (= $|CP_t(\tau)|$) processes are concurrently performed on each server s_t at time τ when a process p_i is issued to a cluster S. First, a process p_i is assumed to be issued to a host server s_t. Here,

both one new process p_i and n_t current processes are performed on the server s_t. Let NT_t be expected termination time $SET_t(n_t, 1)$ [tu] and NE_t be expected electric energy consumption $SEE_t(n_t, 1)$ [W tu] for a host server s_t while a new process p_i is assumed to be performed. Next, suppose only n_t current processes are performed on a server s_t, i.e. no new process is performed. Here, the expected termination time ET_t is $SET_t(n_t, 0)$ and the expected electric energy consumption EE_t is $SEE_t(n_t, 0)$ for the server s_t.

Then, we consider another server $s_u (u \neq t)$ while a new process p_i is issued to a host server s_t. The expected electric energy consumption NEE_{tu} of another server s_u ($\neq s_t$) for a host server s_t is calculated by the following function $FE(EE_u, NT_u, NT_t, minE_u)$:

$$FE(E, T, TT, mE) = \begin{cases} E + (TT - T) \cdot mE & \text{if } TT \geq T. \\ E \cdot TT / T & \text{if } TT < T. \end{cases} \tag{10}$$

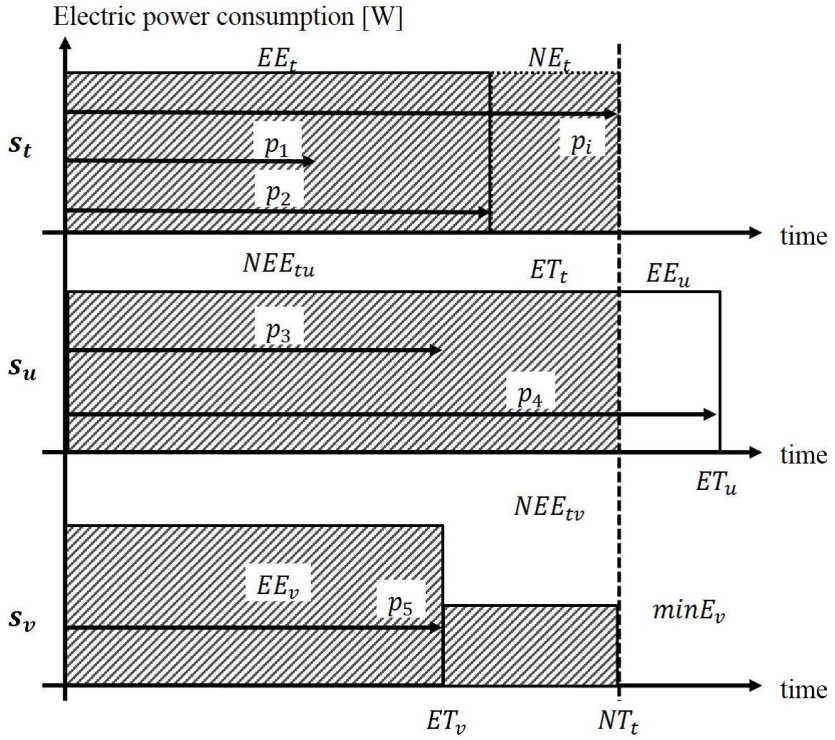

Figure 2: Expected electric energy consumption.

In Figure 2, there are three servers s_t, s_u, and s_v. EE_t, EE_u, and EE_v are expected electric energy $SEE_t(n_t, 0)$, $SEE_u(n_u, 0)$, and $SEE_v(n_v, 0)$ to be consumed by the servers s_t, s_u, and s_v, respectively, to perform current n_t, n_u, and n_v processes, respectively. ET_t, ET_u, and ET_v are expected termination time of the servers s_t, s_u, and s_v, respectively. Next, suppose a process p_i is performed on the server s_t. NE_t and NT_t show the expected electric energy consumption and expected termination time of the server s_t, respectively, to perform both the process p_i and the n_t current processes, i.e. $NE_t = SEE_t(n_t, 1)$ and $NT_t = SET(n_t, 1)$. NEE_{tu} shows the expected electric energy $FE(EE_u, NT_u, NT_t, minE_u)$ to be consumed by the server s_u by time NT_u from the current time. Since $NT_t < ET_u$, the server s_u consumes the electric energy $NEE_{tu} = EE_u \cdot NT_t / ET_u$ by time NT_t. On the other server s_v, every current process terminates by time ET_v before time NT_t. After time ET_v, the server s_v just consumes the minimum electric power $minE_v$ by time NT_t. Hence, $NEE_{tv} = FE(EE_v, NT_v, NT_t, minE_v) = EE_v + (NT_t - ET_v) \cdot minE_v$

since no process is performed after time ET_v. Total expected electric energy TEE_t to be consumed by the servers s_t, s_u, and s_v is $NE_t + NEE_{tu} + NEE_{tv}$ which is shown by hatched area in Figure 2.

The expected total electric energy consumption TEE_t of all servers for a host server s_t to newly perform a process in a cluster S is given as follows:

$$TEE_t = NE_t + \sum_{u=1,\ldots,m(\neq t)} NEE_{tu}. \tag{11}$$

If a client issues a process p_i to a cluster S, a virtual machine vm_h on a host server s_t is selected for the new process p_i by the following virtual machine selection (VMS) algorithm:

[Virtual machine selection (VMS) algorithm] A client issues a process p_i at time τ.

1. **select** a host server s_t, whose expected total electric energy consumption TEE_t is minimum in cluster S.
2. **select** a smallest virtual machine vm_h on the selected host server s_t, where the number nv_h of resident processes is minimum.
3. **perform** the process p_i on the virtual machine vm_h of the host server s_t.

The execution time of each process depends on the total number of processes concurrently performed but is independent of the number of resident virtual machines on a server as presented in paper [30]. Hence, a host server s_t is first selected where the new process p_i is to be performed at step 1. At step 2, the new process p_i is performed on the smallest virtual machine vm_h in order to balance the load among resident virtual machines on the host server s_t. Thus, new processes issued by clients are performed on virtual machines selected in the VM selection algorithm.

6.2 Virtual Machine Migration (VMM) Algorithm

Each time a process is issued to a cluster S, one virtual machine is selected and the process is performed on the selected virtual machine in the VMS algorithm. It may take longer time to perform a process than estimated in the estimation model SET. Some server may consume more electric energy. In order to furthermore reduce the electric energy consumption of servers and the average execution time of processes, a virtual machine on such a heavily energy-consuming server s_t migrates to another server s_u in this paper. Here, the number of processes on the server s_t is reduced and the server s_t consumes smaller electric energy.

Each engaged server s_t where there is at least one resident virtual machine is periodically checked. If migration conditions are satisfied on an engaged server s_t at time τ, one active resident virtual machine vm_h on the server s_t is selected and migrates to another server s_u which is expected to consume smaller electric energy. Variables n_t and nv_h show the numbers $|CP_t(\tau)|$ and $|VCP_h(\tau)|$ of resident processes on a server s_t and a virtual machine vm_h at time τ, respectively.

Let EE_t show the expected electric energy consumption and ET_t be the expected termination time of each server s_t to perform n_t current processes. Here, the expected electric energy consumption EE_t is obtained as $EE_t = SEE_t(n_t, 0)$ and the expected termination time ET_t is calculated as $ET_t = SET_t(n_t, 0)$. Next, suppose a virtual machine vm_h migrates from a host server s_t to a guest server s_u. Here, a collection of $nv_h(\geq 1)$ processes on the virtual machine vm_h move from the server s_t to the server s_u by the migration of the virtual machine vm_h. Since the nv_h processes leave the server s_t, the server s_t is expected to consume the smaller electric energy NE_t $(= SEE_t(n_t - nv_h, 0))$ than EE_t $(NE_t < EE_t)$ and to finish every current process by time NT_t $(= SET_t(n_t - nv_h, 0))$ earlier than time ET_t $(NT_t < ET_t)$. In turn, more number $n_u + nv_h$ of processes than the number n_u of current processes are performed on the guest server s_u. The

expected electric energy consumption NE_{tu} and expected termination time NT_{tu} of the guest server s_u are given as follows:

$$NE_{tu} = SEE_u(n_u + nv_h, 0). \quad (12)$$
$$NT_{tu} = SET_u(n_u + nv_h, 0). \quad (13)$$

The server s_t and another server s_v ($v \neq t$, $v \neq u$) also consume the electric energy by time NT_{tu}. As presented in formula (7), the expected electric energy consumption NEE_t and NEE_{tuv} of a host server s_t and a server s_v ($\neq s_t, s_u$), respectively, are given as follows, where the virtual machine vm_h migrates from the host server s_t to the guest server s_u:

$$NEE_t = FE(NE_t, NT_t, NT_{tu}, minE_t) \quad (14)$$
$$NEE_{tuv} = FE(EE_v, ET_v, NT_{tu}, minE_v) \quad (15)$$

The expected total electric energy consumption $TMEE_{tu}$ where a virtual machine on a host server s_t migrates to a guest server s_u is given as follows:

$$TMEE_{tu} = NEE_t + NE_{tu} + \sum_{v=1,...,m(\neq t, \neq u)} NEE_{tuv}. \quad (16)$$

The following virtual machine migration (VMM) algorithm is periodically performed for each server s_t to find a resident virtual machine vm_h and a guest server s_u to which the virtual machine vm_h to migrate in a cluster S:

[Virtual machine migration (VMM) algorithm]
X = a set of engaged servers which host at least one virtual machine in a cluster S;
for each engaged server s_t **in** X, { $EE_t = SEE_t(n_t, 0)$; and $ET_t = SET_t(nt_t, 0)$; }
while ($|X| > 0$)
{
select a server s_t **in** X **where** EE_t is largest;
select a smallest active virtual machine vm_h on the host server s_t, i.e. nv_h is minimum;
$NE_t = SEE_t(n_t - nv_h, 0)$;
$NT_t = SET_t(n_t - nv_h, 0)$;
for each server s_u ($\neq s_t$) **in** S,
{ /* vm_h migrates from s_t to s_u */
$NE_{tu} = SEE_u(n_u + nv_h, 0)$;
$NT_{tu} = SET_u(n_u + nv_h, 0)$;
$NEE_t = FE(NE_t, NT_t, NT_{tu}, minE_t)$; /* formula (14) */
/* s_t is a host server and s_u is a guest server */
for each server $s_v (\neq s_t, \neq s_u)$ **in** S,
{
$NEE_{tuv} = FE(EE_v, ET_v, NT_{tu}, minE_v)$; /* formula (15) */
};
$TMEE_{tu} = NEE_t + NE_{tu} + \sum_{v=1,...,m(\neq t \neq u)} NEE_{tuv}$;
}; /* **for** s_u end */
select a server s_u **where** $TMEE_{tu}$ is minimum in S;
if s_u is found, {
migrate vm_h **from** s_t **to** s_u;

if s_u is **in** X, $X = X - \{s_u\}$;
}; /* **if** end */
$X = X$ - $\{s_t\}$;
}; /* **while** X end */

First, the expected electric energy consumption EE_t of each engaged server s_t to perform every current process is obtained. Here, $EE_t = SEE_t(n_t, 0)$ where nt_t is the number of current processes performed on the server s_t. X is a set of engaged servers where at least one virtual machine exists in the cluster S. A server s_t whose expected electric energy consumption EE_t is largest is selected in the set X. Here, nv_h processes are performed on each virtual machine vm_h. A smallest active virtual machine vm_h is then selected on the selected server s_t. The expected electric energy consumption NE_t of the server s_t is obtained, which the virtual machine vm_h leaves, i.e. $NE_t = SEE_t(n_t - nv_h, 0)$. The expected electric energy consumption NE_{tu} of another server s_u to which the virtual machine vm_h migrates from the server s_t is obtained, where nv_h processes on the virtual machine vm_h restart, i.e. $NE_{tu} = SEE_u(n_u + nv_h, 0)$. Here, every process on the server s_u is expected to terminate by time NT_{tu}. We obtain the total electric energy to be consumed until time NT_{tu} by not only the server s_t but also another server s_v ($v \neq t$, $v \neq u$). The expected electric energy consumption NEE_t and NEE_{tuv} are obtained as $FE(NE_t, NT_t, NT_{tu}, minE_t)$ and $FE(EE_v, ET_v, NT_{tu}, minE_v)$, respectively. If the total expected electric energy consumption $TMEE_{tu}$ (= $NEE_t + NE_{tu}$ + $\sum_{v=1,\ldots,m(\neq t, \neq u)} NEE_{tuv}$) of all servers is minimum for a server s_u, the virtual machine vm_h can migrate from the host server s_t to the guest server s_u since the total electric energy $TMEE_{tu}$ to be consumed by not only a pair of the host server s_t and the guest server s_u but also the other servers can be reduced.

7 Evaluation

We consider a cluster S of four real servers DSLab2, DSLab1, Sunny, and Atria (m = 4) denoted by s_1, s_2, s_3, and s_4, respectively, in our laboratory. Initially, each server s_t hosts two virtual machines and each virtual machine vm_h is free, i.e. no process is performed. Thus, there are totally eight virtual machines vm_1, ..., vm_8 (v = 8) in the cluster S. The servers DSLab2 (s_1) and DSLab1 (s_2) are equipped with two and one Intel Xeon E5-2667 v2 CPU, respectively. The servers Sunny (s_3) and Atria (s_4) are equipped with one Intel Xeon E5-2620 CPU and one Intel Corei7-6700K CPU, respectively. The performance parameters like maximum thread computation rate $maxCRT_t$ and electric energy consumption parameters like minimum power consumption $minE_t$ of each server s_t are shown in Tables 1 and 2, respectively. For example, the server DSLab2 (s_1) supports totally 32 threads and Atria s_4 supports eight threads. The threads of the servers s_1 and s_2 are the fastest, i.e. maximum thread computation rates $maxCRT_1 = maxCRT_2$ = 1 [vs / tu].

The total electric energy consumption TE_t [W tu] and active time TAT_t [tu] of each server s_t is obtained on the simulation. Active time of each server s_t is time when the server s_t is active, i.e. at least one process is performed. The total active time TAT_t of each server s_t is also obtained in the simulation.

In the simulation, n (> 0) processes p_1, ..., p_n are randomly issued to the cluster S. The starting time $stime_i$ of each process p_i is randomly taken from time 0 to $xtime$ - 1. Here, $xtime$ is 1,000 time units [tu]. In fact, one time unit [tu] is 100 [msec] [9]. The minimum execution time $minT_i$ is randomly taken from 5 to 10 [tu]. This means, the amount VC_i of computation steps of each process p_i is 5 to 10 [vs]. Each process p_i starts at time $stime_i$ and terminates at time $etime_i$. The termination time $etime_i$ is obtained in the simulation. The execution time PET_i of a process p_i is $etime_i - stime_i$ + 1 [tu]. The simulation ends at time $etime$ when every process terminates, i.e. $etime = max(etime_1, \ldots, etime_n)$. For each server s_t, the electric energy consumption $E_t(\tau)$ is pbtained at each time τ. The total electric energy consumption

EE_t of each server s_t is calculated as $\sum_{\tau=0}^{etime} E_t(\tau)$.

The time-based simulator is implemented by using SQL [34] on the Sybase database system [35]. Parameters of servers and processes are stored in the database. States of servers and processes, e.g. electric power consumption of each server and computation laxity of each process obtained for each time unit are also stored in the database. By accessing to the database in SQL, evaluation parameters like the total electric energy consumption of the servers and the average execution time of the processes are obtained.

We consider the random (RD), round robin (RR), and simple globally energy-aware (SGEA) [25] algorithms as non-migration type algorithms and migration RD (RDM), migration RR (RRM), and SGEAM algorithms as migration type algorithms in the evaluation. If a new process p_i is issued, one server s_t is selected in the cluster S. In the RD, RR, and SGEA algorithms, virtual machines do not migrate and stay on host servers. In the RD algorithm, one virtual machine is randomly selected. In the RR algorithm, each virtual machine vm_h is deployed on a server s_t where $t = (h \bmod n) + 1$. Then, a virtual machine vm_h is selected for a new process after a virtual machine vm_{h-1} is selected. In the SGEA algorithm, a server s_t is selected to perform a new process p_i, where the total electric energy consumption TEE_t (formula (11)) is minimum. Then, a smallest resident virtual machine vm_h on the server s_t where minimum number of processes are performed is selected to perform the process p_i. Virtual machines do not migrate and stay on host servers.

In the RDM and RRM algorithms, a virtual machine is selected for a new process in the same way as the RD and RR algorithms. In the SGEAM algorithm which is proposed in this paper, a virtual machine is selected in the same way as the SGEA algorithm. Furthermore, each server is checked every five time units [tu] in the RDM, RRM, and SGEAM algorithms. If the migration condition is satisfied for a pair of a host server s_t and another server s_u, a smallest resident virtual machine vm_h migrates from the host server s_t to the guest server s_u so that the expected total electric energy $TMEE_{tu}$ (formula (16)) of the servers is minimum. Thus, virtual machines migrate to more energy efficient servers in the RDM, RRM, and SGEAM algorithm.

Figure 3 shows the total electric energy consumption TE [W tu] of the four servers $s_1, \ldots, s_4$ in the cluster S for number n of processes. $TE = TE_1 + \cdots + TE_4$. The total electric energy consumption TE of the servers can be reduced in the migration type RDM, RRM, and SGEAM algorithm compared with the non-migration RD, RR, and SGEA algorithms, respectively. Thus, we can reduce the total electric energy consumption of servers by migrating virtual machines to more energy-efficient servers. The total electric energy consumption TE of the SGEAM algorithm is the smallest in the algorithms, about 10 % smaller than the RD, RR, RDM, and RRM algorithms and about 3 % smaller than the SGEA algorithm.

Figure 4 shows the total active time TAT [tu] of the four servers $s_1, \ldots, s_4$ for number n of processes. Here, $TAT = AT_1 + \cdots + AT_4$. In the migration type RDM, RRM, and SGEAM algorithms, the total active time TAT is shorter than the RD, RR, and SGEA algorithms, respectively. This means, servers are less loaded in the migration algorithms than the non-migration algorithms. Because some virtual machine migrates from a host server to a less loaded server if the host server is overloaded. The total active time TAT of the servers can be shortened by migrating virtual machines in the cluster. The total active time TAT of the servers in the SGEAM algorithm is shorter than the RD, RR, RDM, and RRM algorithms but is longer than the SGEA algorithm.

Figure 5 shows the average execution time AET [tu] of the n processes. $AET = (PET_1 + \cdots + PET_n)/n$. The average execution time AET of the migration type RDM, RRM, and SGEAM algorithms is shorter than the non-migration RD and RR algorithms. However, the SGEAM algorithm supports a little bit longer average execution time AET than the non-migration SGEA algorithm.

In conclusion, the SGEAM algorithm supports the smallest electric energy consumption in the algorithms. In the SGEAM algorithm, the active time of servers and execution time of processes are shorter than the RD, RR, RDM, and RRM algorithms while a little bit longer than the SGEA algorithm.

Table 1: Performance parameters.

parameters	DSLab2 (s_1)	DSLab1 (s_2)	Sunny (s_3)	Atira (s_4)
np_t	2	1	1	1
nc_t	8	8	6	4
nt_t	32	16	12	8
$maxCRT_t$ [vs/tu]	1.0	1.0	0.5	0.7
$maxCR_t$ [vs/tu]	32	16	6	5.6

Table 2: Electric energy parameters of servers.

parameters	DSLab2 (s_1)	DSLab1 (s_2)	Sunny (s_3)	Atira (s_4)
$minE_t$ [W]	126.1	126.1	87.2	41.3
$maxE_t$ [W]	301.1	207.3	131.2	89.5
bE_t [W]	30	30	16	15
cE_t [W]	5.6	5.6	3.6	4.7
tE_t [W]	0.8	0.8	0.9	1.1

8 Concluding Remarks

We have to now reduce the electric energy consumption of information systems, especially server clusters. In this paper, we newly proposed the virtual machine migration approach to reducing the electric energy consumption of servers in a cluster. Here, processes on a server migrate to another server by using the virtual machine technologies which are widely used in clusters. A virtual machine on a host server migrates to a guest server which is expected to consume smaller electric energy while processes are being performed. In this paper, we first proposed the simple model to estimate the execution time of each process where only the number of current processes on each server is used. Then, the expected electric energy to be consumed by each server is obtained by using the simple estimation model. Next, we newly proposed the SGEAM algorithm composed of virtual machine selection (VMS) and migration

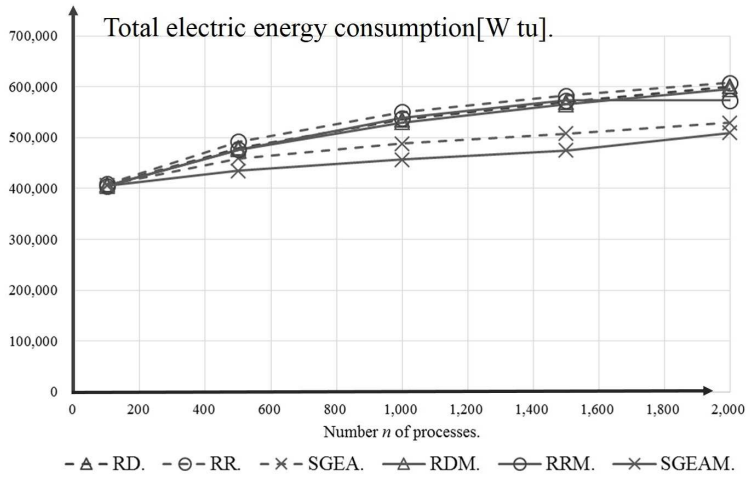

Figure 3: Total electric energy consumption of servers ($m = 4$, $v = 8$).

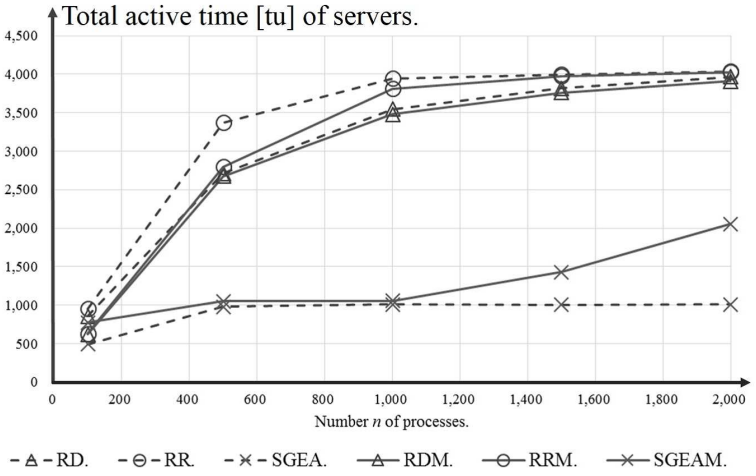

Figure 4: Total active time of servers ($m = 4$, $v = 8$).

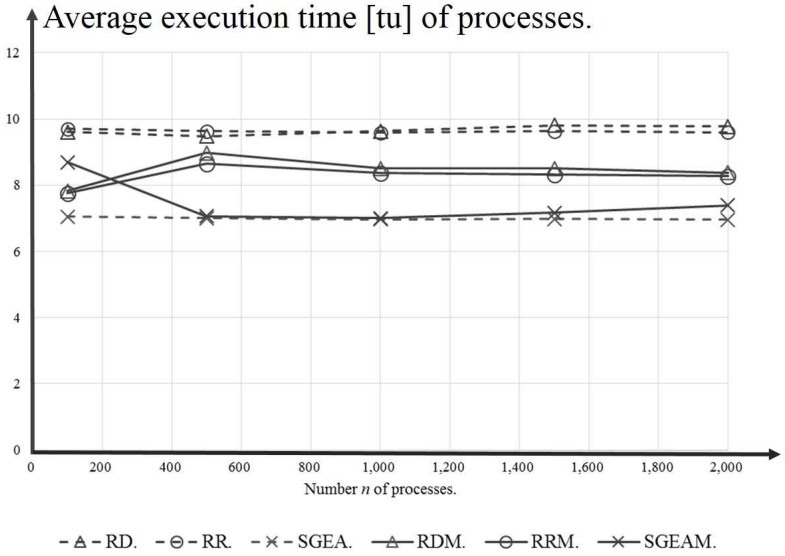

Figure 5: Average execution time of processes ($m = 4$, $v = 8$).

(VMM) algorithm. Here, a virtual machine on an energy-efficient server is first selected to perform a process each time a client issues a process. Every server is periodically checked in the VMM algorithm. Here, virtual machines migrate to more energy-efficient servers. A server which heavily consumes electric energy can get less loaded since a collection of processes on a virtual machine leaves the server for another server. In addition, the total amount of computation to be done by resident processes of each virtual machine is simply estimated only by using the number of resident processes on each virtual machine. In the evaluation, we showed the total electric energy consumption of servers can be mostly reduced in the SGEAM algorithm compared with non-migration and other migration algorithms. For example, the total electric energy consumption of servers in the SGEAM algorithm is reduced to 90 % of the non-migration random (RD), round-robin (RR), migration RD (RDM), and migration RR (RRM)

algorithms. On the other hand, the total active time of the servers and the average execution time of the processes in the SGEAM algorithm are longer than the non-migration SGEA algorithms while shorter than the other algorithms. We are now considering how to reduce the total active time of servers and the average execution time of processes in the SGEAM algorithm.

Acknowledgment

This work was supported by JSPS KAKENHI grant number 15H0295.

References

[1] "Google green," http://www.google.com/green/,[Online; Accessed on June 1, 2017].

[2] WIKIPEDIA, "United nations climate change conference (COP21)."

[3] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa, "Power consumption models for migrating processes in a server cluster," in *Proc. of the 17th International Conference on Network-Based Information Systems (NBiS'14), Salerno, Italy*. IEEE, September 2014, pp. 15–22.

[4] T. Enokido, A. Aikebaier, and M. Takizawa, "A model for reducing power consumption in peer-to-peer systems," *IEEE Systems Journal*, vol. 4, no. 2, pp. 221–229, June 2010.

[5] T. Enokido, A. Aikebaier, and M. Takizawa, "An integrated power consumption model for communication and transaction based applications," in *Proc. of the 25th International Conference on Advanced Information Networking and Applications (AINA'11), Singapore, Singapore*. IEEE, March 2011, pp. 627–636.

[6] T. Enokido, A. Aikebaier, and M. Takizawa, "An extended simple power consumption model for selecting a server to perform computation type processes in digital ecosystems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1627–1636, January 2014.

[7] T. Enokido, A. Aikebaier, and M. Takizawa, "Evaluation of the extended improved redundant power consumption laxity-based (eirpclb) algorithm," in *Proc. of the 28th International Conference on Advanced Information Networking and Applications (AINA'14), Victoria, British Columbia, Canada*. IEEE, May 2014, pp. 940–947.

[8] T. Enokido and M. Takizawa, "Power consumption and computation models of virtual machines to perform computation type application processes," in *Proc. of the 9th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS'15), Blumenau, Brazil*, July 2015, pp. 126–133.

[9] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa, "Power consumption and computation models of a server with a multi-core CPU and experiments," in *Proc. of IEEE the 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA'15), Gwangju, South Korea*. IEEE, March 2015, pp. 217–223.

[10] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa, "Multi-level computation and power consumption models," in *Proc. of the 18th International Conference on Network-Based Information Systems (NBiS'15), Taipei, Taiwan*. IEEE, September 2015, pp. 40–47.

[11] "Intel xeon processor 5600 series: The next generation of intelligent server processors white paper."

[12] T. Enokido, A. Aikebaier, and M. Takizawa, "Process allocation algorithms for saving power consumption in peer-to-peer systems," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 6, pp. 2097–2105, August 2011.

[13] T. Enokido and M. Takizawa, "Energy-efficient delay time-based process allocation algorithm for heterogeneous server clusters," in *Proc. of IEEE the 29th International Conference on Advanced Information Networking and Applications (AINA'15), Gwangju, South Korea*. IEEE, March 2015, pp. 279–286.

[14] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa, "Evaluation of energy-aware server selection algorithm," in *Proc. of the 9th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS'15), Blumenau, Brazil*. IEEE, July 2013, pp. 318–325.

[15] D. Duolikun, T. Enokido, and M. Takizawa, "Asynchronous migration of process replicas in a cluster," in *Proc. of the 29th International Conference on Advanced Information Networking and Applications (AINA'15), Gwangju, South Korea*. IEEE, March 2015, pp. 271–278.

[16] WIKIPEDIA, "Kernel-based virtual machine," https://en.wikipedia.org/wiki/Kernel-basedVirtualMachine, [Online; Accessed on June 1, 2017].

[17] R. Watanabe, D. Duolikun, T. Enokido, and M. Takizawa, "Energy-aware virtual machine migration models in a scalable cluster of servers," in *Proc. of IEEE the 31st International Conference on Advanced Information Networking and Applications (AINA'17), Taipei, Taiwan*. IEEE, March 2017, pp. 85–92.

[18] WIKIPEDIA, "Solid-state drive: SSD," https://en.wikipedia.org/wiki/Solid-state-drive, [Online; Accessed on June 1, 2017].

[19] T. Enokido, A. Aikebaier, M. Takizawa, and S. M. Deen, "Power consumption-based server selection algorithms for communication-based systems," in *Proc. of the 13th International Conference on Network-Based Information Systems (NBiS'10), Takayama, Gifu, Japan*. IEEE, September 2010, pp. 201–208.

[20] "Meta protocol corp: UWmeter," http://www.metaprotocol.com/UWmeter/UWmeter_TOP.html, [Online; Accessed on June 1, 2017].

[21] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa, "Energy-efficient virtualization of threads in a server cluster," in *Proc. of the 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA'15), Krakow, Poland*, November 2015, pp. 288–295.

[22] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa, "Design and evaluation of algorithms to energy-efficiently select servers for storage and computation processes," in *Proc. of the 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS'16), Fukuoka, Japan*. IEEE, July 2016, pp. 162–169.

[23] A. Sawada, S. Nakamura, H. Kataoka, T. Enokido, and M. Takizawa, "Algorithm to energy-efficiently select a server for a general process in a scalable cluster," in *Proc. of the 31th International Conference on Advanced Information Networking and Applications of Workshops (WAINA'17), Taipei, Taiwan*. IEEE, March 2017, pp. 138–145.

[24] A. Sawada, S. Nakamura, T. Enokido, and M. Takizawa, "Eco model of storage-based servers," in *Proc. of the 17th International Conference on Network-Based Information Systems Workshops (NBiS'15), Taipei, Taiwan*. IEEE, September 2015, pp. 407–411.

[25] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa, "Simple energy-aware algorithms for selecting a server in a scalable cluster," in *Proc. of the 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA'17), Taipei, Taiwan*. IEEE, March 2017, pp. 146–153.

[26] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa, "Power consumption model for redumdantly performing mobile-agents," in *Proc. of the 8th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS'14), Birmingham, United Kingdom*. IEEE, July 2014, pp. 185–190.

[27] D. Duolikun, S. Nakamura, T. Enokido, and M. Takizawa, "An energy-efficient process migration approach to reducing electric energy consumption in a cluster of servers," *International Journal of Communication Networks and Distributed Systems*, vol. 15, no. 4, pp. 400–420, October 2015.

[28] T. Enokido, A. Aikebaier, and M. Takizawa, "The evaluation of the improved redundant power consumption laxity-based (irpclb) algorithm in homogeneous and heterogeneous clusters," in *Proc. of the 7th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS'13), Taichung, Taiwan*. IEEE, July 2013, pp. 91–98.

[29] Q. Xilong and X. Peng, "An energy-efficient virtual machine scheduler based on cpu share-reclaiming policy," *International Journal of Grid and Utility Computing*, vol. 6, no. 2, pp. 113–120, March 2015.

[30] R. Watanabe, D. Duolikun, T. Enokido, and M. Takizawa, "An eco model of process migration with virtual machines in clusters," in *Proc. of the 18th International Conference on Network-Based Information Systems (NBiS'16), Ostrava, Czech Republic*. IEEE, September 2016, pp. 292–297.

[31] R. Bianchini and R. Rajamony, "Power and energy management for server systems," *IEEE Computer*, vol. 37, no. 11, pp. 68–74, November 2004.

[32] L. Barolli and F. Xhafa, "Jxta-overlay: A p2p platform for distributed, collaborative and ubiquitous computing," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 6, pp. 2163–2172, June 2011.

[33] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa, "Energy-aware clusters of servers for storage and computation applications," in *Proc. of the 30th International Conference on Advanced Information Networking and Applications (AINA'16), Crans-Montana, Switzerland*. IEEE, March 2016, pp. 400–407.

[34] C. J. Date, *An Introduction to Database Systems (8 Edition)*. Pearson, August 2003.

[35] "SAP company," https://www.sap.com/index.html, [Online; Accessed on June 1, 2017].

Author Biography

Ryo Watanabe received his BE in Advanced Sciences form Hosei University, Japan, in 2017. He is now a Master's student at Systems engineering in Hosei University, Japan.

Dilawaer Duolikun received his BE in Information Science and Engineering from XinJiang University, China, in 2010 and received his M.E Degree in Advanced Sciences from Hosei University in 2016. He is now a Ph.D student at Systems engineering in Hosei University, Japan. He won the best paper awards at CISIS-2014 and NBiS-2016. His research interests include distributed systems, networks, and eco distributed systems.

Tomoya Enokido received BE, ME, and Ph.D from Tokyo Denki University, Japan in 1997, 1999, and 2003, respectively. After working for NTT Data Corporation and Tokyo Denki University, he joined Rissho University in 2005. He is now a full professor in the Faculty of Business Administration, Rissho University. His research interests include distributed systems and networks. He is a member of IEEE.

Makoto Takizawa is currently a full professor of Hosei University. He received his BE and ME in Applied Physics and DE in Computer Science from Tohoku University, Japan. He was on the Board of Governors (2003 - 2008) and is a Golden Core member of the IEEE Computer Society. He is a fellow of IPSJ. He chaired many international conferences like ICDCS (1998, 2002). He founded IEEE AINA. His research interests include distributed systems and networks. He is a member of IEEE, ACM and IPSJ.