Energy-aware Migration of Virtual Machines in a Cluster

Dilawaer Duolikun, Shigenari Nakamura, Ryo Watanabe, Tomoya Enokido, and Makoto Takizawa

Abstract In order to realize eco-society, we have to reduce the electric energy consumed by servers. Virtual machines are now widely used to support applications with virtual computation service in server clusters. Here, a virtual machine can migrate to a guest server while processes are being performed. In the EAMV algorithm we previously proposed, the termination time of each process on each virtual machine has to be estimated. However, it is not easy to obtain the state of each process and takes time to calculate the expected termination time. In this paper, we newly propose a virtual machine migration (VMM) algorithm where termination time of each virtual machine is estimated without considering each process. We evaluate the VMM algorithm and show the total electric energy consumption and active time of servers and the average execution time of processes can be reduced in the VMM algorithm compared with non-migration algorithms. The VMM algorithm is simpler than the EAMV algorithm

Dilawaer Duolikun
Hosei University, Tokyo, Japan, e-mail: dilewerdolkun@gmail.com

Shigenari Nakamura
Hosei University, Tokyo, Japan, e-mail: nakamura.shigenari@gmail.com

Ryo Watanabe
Hosei University, Tokyo, Japan, e-mail: ryo.watanabe.4h@stu.hosei.ac.jp

Tomoya Enokido
Rissho University, Tokyo, Japan, e-mail: eno@ris.ac.jp

Makoto Takizawa
Hosei University, Tokyo, Japan, e-mail: makoto.takizawa@computer.org

L. Barolli et al. (eds.), *Advances on Broad-Band Wireless Computing, Communication and Applications*, Lecture Notes on Data Engineering and Communications Technologies 2, DOI 10.1007/978-3-319-49106-6_3

1 Introduction

We have to reduce the electric energy consumed in information systems, especially server clusters [27] to realize eco society [26]. In order to discuss how to reduce the electric energy consumption of servers in a cluster, we first need a power consumption model which shows how much electric power a server consumes to perform application processes. Types of power consumption models are proposed in our previous studies [12, 13, 14, 20, 21, 22]. The power consumption models are also proposed for communication [16] and storage [17] types of application processes. In order to reduce the electric energy consumption, types of server selection algorithms [7, 13, 14, 17, 20, 21, 22] are proposed. Here, a server to perform a request process is selected so that the expected total electric energy consumption of the servers can be reduced. A process migration approach is also discussed where processes migrate to more energy-efficient servers [4, 5, 6, 7, 11]. However, it is not easy to migrate types of processes to servers with various architectures and operating systems.

A server cluster provides applications with virtual computation service by using virtual machines like KVM [25] and VMware [27]. Applications processes can be performed on a virtual machine without being conscious of what servers are included in a cluster. A virtual machine on a host server can migrate to a guest server while processes are being performed on the virtual machine [25]. The EAMV (Energy-Aware Migration of Virtual machines) algorithm [10] is proposed to select a virtual machine for a request process and migrate a virtual machine to another guest server. Here, the termination time of every current process on each virtual machine has to be estimated to obtain the expected electric energy consumption of the servers. In this paper, we newly propose a virtual machine migration (VMM) algorithm which is simpler than the EAMV algorithm. As discussed in paper [28], the average execution time of processes depends on the total number of current processes on a server and is independent of the number of virtual machines. A server is selected for a request process which is expected to consume the minimum electric energy to perform the process and every current process. Then, a virtual machine where the minimum number of processes are performed is selected in the selected server. If a server is expected to consume more electric energy to perform processes, one virtual machine is selected in the server, where the maximum number of processes are performed. Then, the selected virtual machine migrates to another guest server which is expected to consume smaller electric energy. We evaluate the VMM algorithm compared with non-migration algorithms. In the evaluation, we show the total electric energy consumption and active time of servers and the average execution time of processes are reduced in the VMM algorithm.

In section 2, we present a model of virtual machines. In section 3, we discuss power consumption and computation models of a server with virtual machines. In section 4, we propose the VMM algorithm. In section 5, we evaluate the VMM algorithm.

2 System Model

A cluster S is composed servers s_1, ..., s_m ($m \geq 1$). A server s_t is equipped with a set CP_t of np_t (≥ 1) homogeneous CPUs, cp_{t0}, ..., cp_{t,np_t-1}. Each CPU cp_{tk} is composed of nc_{tk} (≥ 1) cores $c_{tk0}, \ldots, c_{tk,nc_{tk}-1}$. Each core c_{tki} supports a set $\{th_{tki0}, \ldots, th_{tki,ct_{tki}-1}\}$ of threads ($ct_{tki} \geq 1$). Here, $nc_{tk} = nc_t$ and $ct_{tki} = ct_t$ for each core ct_{tki}. A server s_t supports processes with the total number nt_t of threads, where $nt_t = np_t \cdot ct_{tk} \cdot nc_t$.

A server s_t is modeled to support processes with vt_t (≥ 1) virtual processors $vt_{t0}, \ldots, vt_{t,vt_t-1}$. In this paper, every virtual processor is homogeneous in each server s_t. Applications can use virtual processors to perform processes without being conscious of which thread in which core of which CPU is supported. One virtual processor is at a time allocated to a process p_i [24]. In this paper, we assume each virtual processor is in a one-to-one correspondent relation with one thread. Hence, $vt_t = nt_t$. A virtual processor is *active* if at least one process is performed, otherwise *idle*. A server is *active* if and only if (iff) at least one virtual processor is active, otherwise *idle*. In this paper, a *process* means an application process to be performed on a server, which uses CPU.

A cluster S supports applications with a set VM of virtual machines $\{VM_1, \ldots, VM_v\}$ ($v \geq 0$). Each virtual machine VM_h is supported with virtual processors of a server s_t. Here, s_t is a *host* server of the virtual machine VM_h and VM_h is a *resident* virtual machine of the server s_t. $SVM_t(\tau)$ shows a set of resident virtual machines on a host server s_t and $HS_h(\tau)$ denotes a host server of a virtual machine VM_h at time τ. $VP_h(\tau)$ ($\subseteq VP_t$) shows a subset of virtual processors on a host server s_t, which are allocated to a virtual machine VM_h at time τ. One virtual machine VM_h ($\in VM$) on a host server s_t is selected for a process p_i issued by a client. Then, the process p_i is performed on the virtual machine VM_h. Here, the process p_i is a *resident* process of the virtual machine VM_h. $VCP_h(\tau)$ shows a set of resident processes of a virtual machine VM_h at time τ. A virtual machine VM_h is *active* at time τ if $|VCP_t(\tau)| > 0$, i.e. at least one process is performed, otherwise *idle*. $CP_t(\tau)$ is a set of all the resident processes performed on virtual machines of a server s_t at time τ, i.e. $CP_t(\tau) = \cup_{VM_h \in SVM_t(\tau)} VCP_h(\tau)$.

A virtual machine VM_h on a host server s_t can migrate to a guest server s_u. First, a copy of memory of a virtual machine VM_h is created on a guest server s_u. On issuing a migration command [25] on the host server s_t, the memory state of VM_h is first transferred to the server s_u while processes are being performed. On termination of the state transfer to the host server s_u, the processes are resumed on the virtual machine VM_h and the state of VM_h changed after the state transfer is transfered to the server s_u. Then, the processes on the virtual machine VM_h are restarted on the server s_u.

3 Power Consumption and Computation Models

3.1 MLPCM and MLC Models

The electric power consumption $E_t(\tau)$ [W] of a server s_t with multiple CPUs to perform computation processes at time τ is given as follows [23]:

[Multi-Level Power Consumption with Multiple CPUs (MLPCM) model]

$$E_t(\tau) = minE_t + \sum_{k=0}^{np_t-1}\{\gamma_{tk}(\tau)\ [bE_t + \sum_{i=0}^{nc_t-1} \alpha_{tki}(\tau)(cE_t + \beta_{tki}(\tau)\ tE_t)]. \quad (1)$$

Here, $\gamma_{tk}(\tau) = 1$ if a CPU cp_{tk} is active. Otherwise, $\gamma_{tk}(\tau) = 0$. That is, $E_t(\tau) = minE_t$ [W] in an idle server. $\alpha_{tki}(\tau) = 1$ if a core c_{tki} is active on a CPU cp_{tk}. Otherwise, $\alpha_{tki}(\tau) = 0$. $\beta_{tki}(\tau)$ ($\leq ct_t$) is the number of active threads on a core c_{tki}.

In Linux operating systems, processes are allocated to nt_t ($\geq$ 1) virtual processors, in the round-robin (RR) algorithm [24]. The, electric power consumption $CE_t(n)$ [W] of a server s_t to concurrently perform n ($\geq$ 1) processes at time τ is given in the MLPCM model [23] as follows:

[MLPCM model]

$$CE_t(n) = \begin{cases} minE_t\ if\ n = 0. \\ minE_t + n \cdot (bE_t + cE_t + tE_t)\ if\ 1 \leq n \leq np_t. \\ minE_t + np_t \cdot bE_t + n(cE_t + tE_t)\ if\ np_t < n \leq nc_t \cdot np_t. \\ minE_t + np_t \cdot (bE_t + nc_t \cdot cE_t) + nt_t \cdot tE_t\ if\ nc_t \cdot np_t < n < nt_t. \\ maxE_t\ if\ n \geq nt_t. \end{cases} \quad (2)$$

In this paper, we assume $E_t(\tau) = CE_t(|CP_t(\tau)|)$ for each server s_t. The total electric energy $TE_t(st, et)$ [J] consumed by a server s_t from time st to time et is TE_t (st, et) = $\sum_{\tau=st}^{et} E_t(\tau)$.

It takes T_{ti} [sec] to perform a process p_i on a thread in 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, i.e. T_{ti} = $minT_{ti}$. In a cluster S of servers s_1, $\ldots$, s_m ($m \geq 1$), $minT_i$ shows a minimum one of $minT_{1i}, \ldots, minT_{mi}$. That is, $minT_i$ = $minT_{fi}$ on the fastest thread which is on a server s_f in the cluster S. Here, the server s_f is referred to as $fastest$. We assume one virtual computation step [vs] is performed on the fastest server s_f for one time unit [tu]. This assumption means, the maximum computation rate $maxCRT_f$ of a fastest server s_f is assumed to be one [vs/sec]. Here, $maxCRT = maxCRT_f$. On another slower server s_t, $maxCRT_t \leq maxCRT_f$ (= 1). The total number VC_i of virtual computation steps to be performed in a process p_i is defined to be $minT_i$ [sec] $\cdot$ $maxCRT_f$ [vs/sec] = $minT_i$ [vs] where a server s_f is the fastest. The maximum computation rate $maxCR_{ti}$ of a process p_i on a server s_t is VC_i / $minT_{ti}$ [vs/sec] ($\leq$ 1). On a fastest server s_f, $maxCR_{fi}$

= $maxCRT$ = 1. For every pair of processes p_i and p_j on a server s_t, $maxCR_{ti}$ = $maxCR_{tj}$ = $maxCRT_t$ (≤ 1). The maximum computation rate $maxCR_t$ of a server s_t is $nt_t \cdot maxCRT_t$.

[Multi-level computation (MLC) model] [20, 21, 22] The computation rate $CR_{ti}(\tau)$ [vs/sec] of a process p_i on a server s_t at time τ is given as follows:

$$CR_{ti}(\tau) = \begin{cases} maxCR_t \;/\; |CP_t(\tau)| & if\ |CP_t(\tau)| > nt_t. \\ maxCRT_t & if\ |CP_t(\tau)| \leq nt_t. \end{cases} \tag{3}$$

Suppose a process p_i on a server s_t starts at time st and ends at time et. Here, $\sum_{\tau=st}^{et} CR_{ti}(\tau) = VC_i$ [vs]. At time τ a process p_i starts on a server s_t, the computation laxity $plc_{ti}(\tau)$ of a process p_i is VC_i. At each time τ, $plc_{ti}(\tau)$ is decremented by the computation rate $CR_{ti}(\tau)$.

[Computation of a process p_i]

1. At initial time τ the process p_i starts, $plc_{ti}(\tau) = VC_i$;
2. At each time τ, $plc_{ti}(\tau+1) = plc_{ti}(\tau)$ - $CR_{ti}(\tau)$;
3. Then, if $plc_{ti}(\tau+1) \leq 0$, p_i terminates at time τ;

3.2 Computation Model of a Virtual Machine

Let p_{hi} show a process p_i performed on a virtual machine VM_h of a server s_t. $plc_{hi}(\tau)$ is the computation laxity $plc_{ti}(\tau)$ of a process p_{hi} on the server s_t at time τ. The *virtual machine* (*VM*) *laxity* $vlc_h(\tau)$ [vs] of a virtual machine VM_h at time τ is defined to be the summation of computation laxities of the resident processes of VM_h:

- $vlc_h(\tau) = \sum_{p_i \in VCP_h(\tau)} plc_{hi}(\tau)$.

The *server laxity* $slc_t(\tau)$ [vs] of a server s_t is the summation of VM laxities of virtual machines hosted by the server s_t at time τ:

- $slc_t(\tau) = \sum_{VM_h \in SVM_t(\tau)} vlc_h(\tau)$.

The *VM* computation rate $VCR_h(\tau)$ [vs/sec] of a virtual machine VM_h is defined as follows:

[Virtual machine (VM) computation ratio] The *VM* computation rate $VCR_h(\tau)$ of a virtual machine VM_h on a server s_t at time τ is given as follows:

$$VCR_h(\tau) = \begin{cases} maxCR_t \cdot |VCP_h(\tau)| \;/\; |CP_t(\tau)| & if\ |CP_t(\tau)| > nt_t. \\ |VCP_h(\tau)| \cdot maxCRT_t & if\ |CP_t(\tau)| \leq nt_t. \end{cases} \tag{4}$$

Here, $VCR_h(\tau) \leq VCR_k(\tau)$ if $|VCP_h(\tau)| \leq |VCP_k(\tau)|$ for every pair of different virtual machines VM_h and VM_k on a same server s_t. $VCR_h(\tau)$ / $VCR_k(\tau)$ =

$|VCP_h(\tau)| \,/\, |VCP_k(\tau)|$. The computation rate $CR_{ti}(\tau)$ of each process p_i depends on the total number $|CP_t(\tau)|$ of processes but is independent of the number $|SVM_t(\tau)|$ of virtual machines of a host server s_t [28].

The VM laxity $vlc_h(\tau)$ of a virtual machine VM_h and the server laxity $slc_t(\tau)$ of a server s_t which hosts VM_h are manipulated as follows:

[VM computation (VMC) model]

$VCP_h = VCP_h(\tau)$;
while ($VCP_h \neq \phi$) {

1. **for** each process p_i on a virtual machine VM_h in $SVM_t(\tau)$, i.e. $p_i \in VCP_h$, $plc_{hi}(\tau+1) = plc_{hi}(\tau) - VCR_h(\tau) \,/\, |VCP_h|$;
2. **if** plc_{hi} $(\tau + 1) \leq 0$, p_i terminates at time τ and $VCP_h = VCP_h - \{p_i\}$;
3. $vlc_h(\tau+1) = vlc_h(\tau) - VCR_h(\tau)$;
4. **if** $vlc_h(\tau+1) \leq 0$, every process on VM_h terminates, i.e. VM_h gets idle;
5. $\tau = \tau + 1$;

}; /* **while** end */

A virtual machine VM_h is referred to as *terminate* if VM_h gets idle, i.e. no process is performed on VM_h In this paper, we estimate the termination time ET_t and electric energy consumption EE_t of a server s_t to perform every process by considering active virtual machines, not each process as follows:

[Virtual machine computation (VMC) model]

VMEST $(s_t, \tau; EE_t, ET_t)$
input s_t; τ;
output EE_t; ET_t;
{ $ncp = |CP_t(\tau)|$; /*number of processes on s_t*/
  $vlc = 0$;
  $SVM = SVM_t(\tau)$; /* set of virtual machines on s_t */
  $x = \tau$;
  $EE_t = 0$;
/* obtain laxity vlc of the server s_t */
  for each virtual machine VM_h **in** SVM, /* VM laxity of VM_h */
    $vlc_h = vlc_h(\tau)$ $(= \sum_{p_i \in VCP_h(\tau)} plc_i(\tau))$;
    $ncp_h = |VCP_h(\tau)|$; /*number of processes on VM_h*/
    $vlc = vlc + vlc_h$; /* server laxity of s_t */
  }; /* **for** end */
  while ($SVM \neq \phi$) {
    $EE_t = EE_t + CE_t(ncp)$; /* electric energy */
    for each virtual machine VM_h **in** SVM, {
      $vlc_h = vlc_h - VCR_h(\tau)$; /* VM laxity is decremented */
      if $vlc_h \leq 0$, /*VM_h gets idle, i.e. terminates */ {
        $SVM = SVM - \{VM_h\}$;
        $ncp = ncp - ncp_h$;
      } **else** $vlc = vlc - vlc_h$; /*decrement server laxity*/
    }; /* **for** end */

$x = x + 1$; /* time advances */
}; /* **while** end */
$ET_t = x - 1$; /* every VM terminates, i.e. gets idle on s_t*/
};

Here, the VM computation rate $VCR_h(\tau)$ of a virtual machine VM_h depends on how many number of processes are totally performed on VM_h. The more number of processes are performed on a virtual machine VM_h, the larger VM computation rate $VCR_h(\tau)$. Here, it is noted we do not consider the termination time of each process p_i and only consider each virtual machine.

4 A Virtual Machine Migration (VMM) Algorithm

A client issues a process p_i to virtual machines $VM_1, \ldots, VM_v$ ($v \geq 1$) in a cluster S. The expected electric energy consumption EE_t and expected termination time ET_t of a server s_t to perform every current process on the virtual machines are obtained by the procedure **VMEST** $(s_t, \tau; ET_t, EE_t)$. Then, one virtual machine VM_h on a server s_t is selected to perform a process p_i as follows:

[VM selection]
for each server s_u in a cluster S, **VMEST** $(s_u, \tau; EE_u, ET_u)$;
$MS = \{s_u \mid EE_u$ is minimum in $S\}$;
select s_t in MS where $|CP_t(\tau)|$ is minimum;
select a virtual machine VM_h in s_t where $|VCP_h(\tau)|$ is minimum;

Then, the process p_i is performed on the selected virtual machine VM_h in the selected host server s_t.

A server s_t is *overloaded* at time τ iff $|CP_t(\tau)| > maxNCP_t$. For example, the computation rate $CR_{ti}(\tau)$ of each process p_i should be larger than $\alpha \cdot maxCR_t$. Since $CR_{ti}(\tau) < \alpha \cdot maxCR_t$, $CR_{ti}(\tau) = nt_t \cdot maxCR_t \,/\, |CP_t(\tau)|$, $nt_t \cdot maxCR_t \,/\, maxNCP_t$ $= \alpha \cdot maxCR_t$. Hence, $maxNCP_t = nt_t \,/\, \alpha$. A server s_t more overloaded than a server s_u if $|CP_t(\tau)| \,/\, maxNCP_t > |CP_u(\tau)| \,/\, maxNCP_u$.

First, an *overloaded* server s_t is selected whose expected electric energy EE_t is the largest in a cluster S. Then, a virtual machine VM_h is selected in the selected server s_t, $VM_h \in SVM_t(\tau)$, where the number $|VCP_h(\tau)|$ of processes performed on VM_h is minimum.

[VM selection in s_t]
for each server s_u in a cluster S, **VMEST** $(s_u, \tau; EE_u, ET_u)$;
$OS = \{s_u \mid s_u$ is overloaded and $SVM_u(\tau) \neq \phi$ in $S\}$;
while ($OS \neq \phi$)
{
select s_t whose EE_t is maximum in OS;
while (s_t is overloaded)
{

select a virtual machine VM_h in $SVM_t(\tau)$ where $|VCP_h(\tau)|$ is minimum;
select a server s_u where $|CP_u(\tau)|$ is minimum and which is not overloaded;
if not found, **break**;
migrate VM_h **from** s_t **to** s_u;
}; /* **while** end */
OS = OS − $\{s_t\}$;
}; /***while** end */

5 Evaluation

We evaluate the VMM algorithm in terms of the total electric energy consumption TEE [J] and total active time TAT [sec] of servers and the average execution time AET [sec] of processes compared with the random (RD), round robin (RR), and NVM (non-migration of virtual machines). In the NVM algorithm, a virtual machine is selected in the same VM selection algorithm as the VMM algorithm but no virtual machine migrates. In the RD algorithm, one virtual machine VM_h is randomly selected. In the RR algorithm, a virtual machine VM_h is selected after a virtual machine VM_{h-1}. In the RD, RR, and NVM algorithms, every virtual machine VM_h does not migrate. In the VMM algorithm, each virtual machine VM_h migrates to a guest server.

There are m heterogeneous servers $s_1, \ldots, s_m$ in a cluster S. The power consumption parameters like $minE_t$ and $maxE_t$ [W] and the performance parameters like $maxCRT_t$ and $maxCR_t$ of a server s_t are randomly taken as shown in Table 1. There are a set VM of v (≥ 1) virtual machines $VM = \{VM_1, \ldots, VM_v\}$. In the evaluation, $m = 6$ and $v = 8$.

The number n (≥ 1) of processes $p_1, \ldots, p_n$ are randomly issued to the cluster S. In the simulation, one time unit [tu] is assumed to be 100 [msec]. In each process configuration PF_{ng}, the minimum execution time $minT_i$ of each process p_i is randomly taken from 5 to 10 [tu], i.e. 0.5 to 1.0 [sec]. The amount VS_i [vs] of virtual computation steps of each process p_i is $minT_i$ as discussed in this paper. The start time $stime_i$ of each process p_i is randomly taken from 0 to $xtime$ - 1. The simulation time $xtime$ is 200 [tu] (= 20 [sec]). The simulation is time-based. We randomly generate four process configurations $PF_{n1}, \ldots, PF_{n4}$ of the processes $p_1, \ldots, p_n$.

We randomly generate four server configurations $SF_1, \ldots, SF_4$ of the servers $s_1, \ldots, s_m$ ($m = 6$). In each server configuration SF_k, the parameters of each server s_t are randomly taken. We also generate four VM configurations $VF_1, \ldots, VF_4$ of the virtual machines $VM_1, \ldots, VM_v$ ($v = 8$). In each VM configuration VF_l, initially each virtual machine VM_h is randomly deployed on a server. For each combination of the configurations SF_k, VF_l, and PF_{ng}, the electric energy consumption EE_t and active time AT_t of each server s_t and the execution time ET_i of each process p_i are obtained.

Figure 1 shows the total electric energy consumption (TEE) [J] of six servers $s_1, \ldots, s_6$ ($m = 6$) with eight virtual machines $VM_1, \ldots, VM_8$ ($v = 8$) for number n of

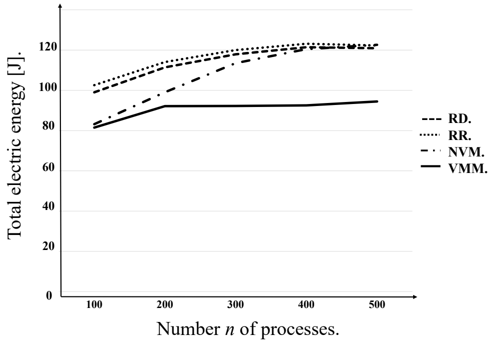

Fig. 1 Total electric energy consumption.

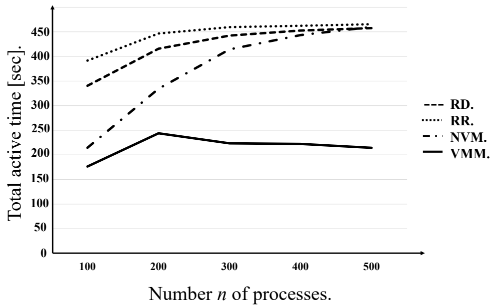

Fig. 2 Total active time of servers.

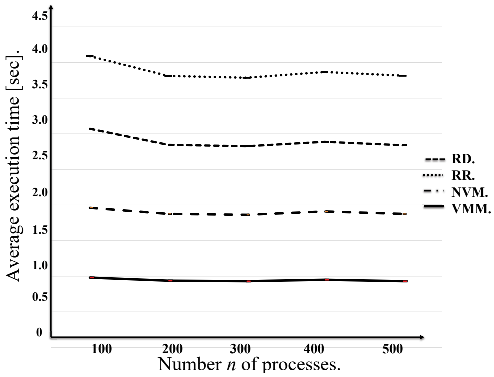

Fig. 3 Average execution time of processes.

processes. TEE is the summation $EE_1 + \ldots + EE_m$. As shown in Figure 1, TEE of the VMM algorithm is smaller than the other non-migration algorithms. Thus, TEE can be reduced in the VMM algorithm.

Figure 2 shows the total active time (TAT) [sec] of six servers ($m = 6$) for the number n of processes. TAT is $AT_1 + \ldots + AT_m$. TAT in the VMM algorithm is shorter than the other algorithms. This means, the servers are more lightly loaded in the VMM algorithm than the other algorithms.

Figure 3 shows the average execution time (AET) [sec] of the number n of processes. AET is $(ET_1 + \ldots + ET_n) / n$. AET of the VMM algorithm is shorter than the other algorithms.

As shown here, the total electric energy consumption and active time of servers and average execution time of processes can be reduced in the VMM algorithm.

Table 1 Parameters.

parameters	values
m	number of servers $s_1, \ldots, s_m$ (≥ 1).
np_t	number of CPUs (≤ 2).
nc_t	number of cores (1, 2, 4, 6)/ CPU.
ct_t	threads/core (≤ 2)
nt_t	number of threads (= $ct_t \cdot np_t \cdot nc_t$).
$maxCRT_t$ [vs/tu]	0.5 $\sim$ 1.
$maxCR_t$ [vs/tu]	$nt_t \cdot maxCRT_t$.
$minE_t$ [W]	80 $\sim$ 100.
$maxE_t$ [W]	$minE_t$ $+E$ ($50 \leq E \leq 150$).
bE_t [W]	($maxE_t$ - $minE_t$) / (4 $\cdot$ np_t).
cE_t [W]	5 $\cdot$ ($maxE_t$ - $minE_t$) / (8 $\cdot$ $np_t \cdot nc_t$).
tE_t [W]	($maxE_t$ - $minE_t$) / (8 $\cdot$ nt_t).
n	number of processes $p_1, \ldots, p_n$ (≥ 1).
$minT_i$ [tu]	minimum computation time of p_i (0.5 $\sim$ 1.0).
VS_i [vs]	$VS_i = minT_i$.
$stime_i$	starting time of p_i ($0 \leq st_i < xtime$ - 1).
$xtime$	simulation time (= 200 [tu] = 20 [sec]).
v	number of virtual machines $VM_1, \ldots, VM_v$.

6 Concluding Remarks

In this papers, we proposed the VMM algorithm to reduce the electric energy consumption of servers in a cluster. A virtual machine migrates from a host server to a guest server if the guest server is expected to consume smaller electric energy than the host server. The termination time of every current process is estimated for each virtual machine without considering each process The computation time of the VMM algorithm is smaller than the other algorithms. In the evaluation, we showed the total electric energy consumption and active time of servers and the average execution time of processes can be reduced in the VMM algorithm compared with the non-migration algorithms.

Acknowledgements This work was supported by JSPS KAKENHI grant number 15H02695.

References

1. Aikebaier, A., Enokido, T., and Takizawa, M.: Energy-Efficient Computation Models for Distributed Systems, Proc. of the 12th International Conference on Network-Based Information Systems (NBiS-2009), pp.424-431, (2009).
2. Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G.: Distributed Systems Concepts and Design, 4th ed., Addison-Wesley, (2012).
3. Duolikun, D., Aikebaier, A., Enokido, T., and Takizawa, M.: Energy-aware Passive Replication of Processes, Journal of Mobile Multimedia, **9**(1&2), pp.53–65, (2013).
4. Duolikun, D., Aikebaier, A., Enokido, T., and Takizawa, M.: Power Consumption Models for Redundantly Performing Mobile-Agents, Proc. of the 8th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2014), pp.185-190, (2014).
5. Duolikun, D., Aikebaier, A., Enokido, T., and Takizawa, M.: Power Consumption Models for Migrating Processes in a Server Cluster Proc. of the 17th International Conference on Network-Based Information Systems (NBiS-2014), pp.155-162, (2014).
6. Duolikun, D., Enokido, T., and Takizawa, T.: Asynchronous Migration of Process Replicas in a Cluster, Proc. of IEEE the 29th International Conference on Advanced Information Networking and Applications (AINA-2015), pp.271-278, (2015).
7. Duolikun, D., Enokido, T., and Takizawa, T.: Energy-Efficient Replication and Migration of Processes in a Cluster, Proc. of the 9th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2015), pp.118–125, (2015).
8. Duolikun, D., Aikebaier, A., Enokido, T., and Takizawa, M.: Energy-Efficient Dynamic Clusters of Servers, Journal of Supercomputing, **71**(5), pp.1642–1656, (2015).
9. Duolikun, D., Watanabe, R., Enokido, T., and Takizawa, T.: A Model for Migration of Virtual Machines to Reduce Electric Energy Consumption, Proc. of the 10th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2016), pp.160-166, (2016).
10. Duolikun, D., Watanabe, R., Enokido, T., and Takizawa, T.: A Model for Migration of Virtual Machines to Reduce Electric Energy Consumption, Proc. of the 19th International Conference on Network-based Information Systems (NBiS-2016), CD-ROM, (2016).
11. Duolikun, D., Nakamura, S., Enokido, T., and Takizawa, M.: An Energy-efficient Process Migration Approach to Reducing Electric Energy Consumption in a Cluster of Servers, International Journal of Communication Networks and Distributed Systems, **15**(4), pp.400-420, (2015).
12. Enokido, T., Aikebaier, A., Deen, S, M., and Takizawa, M.: Power Consumption-based Server Selection Algorithms for Communication-based Systems, Proc. of the 13th International Conference on Network-based Information Systems (NBiS-2010), pp.201-208, (2010).
13. Enokido, T., Aikebaier, A., and Takizawa, M.: A Model for Reducing Power Consumption in Peer-to-Peer Systems, IEEE Systems Journal, **4**(2), pp.221-229, (2010).
14. Enokido, T., Aikebaier, A., and Takizawa, M.: Process Allocation Algorithms for Saving Power Consumption in Peer-to-Peer Systems, IEEE Transactions on Industrial Electronics, **58**(6), pp.2097-2105, (2011).
15. Enokido, T. and Takizawa, M.: An Extended Power Consumption Model for Distributed Applications, Proc. of IEEE the 26th International Conference on Advanced Information Networking and Applications (AINA-2012), pp.912-919, (2012).
16. Enokido, T. and Takizawa, M.: An Integrated Power Consumption Model for Distributed Systems, IEEE Transactions on Industrial Electronics, **60**(2), pp.824-836, (2013).
17. Enokido, T., Aikebaier, A., and Takizawa, M.: An Extended Simple Power Consumption Model for Selecting a Server to Perform Computation Type Processes in Digital Ecosystems, IEEE Transactions on Industrial Informatics, **10**(2), pp.1627-1636, (2014).
18. Enokido, T. and Takizawa, M.: An Energy-Efficient Load Balancing Algorithm for Virtual Machine Environment to Perform Communication Type Application Processes, Proc. of IEEE the 30th International Conference on Advanced Information Networking and Applications (AINA-2016), pp. 392-399, (2016).

19. Ghemawat, S., Gobioff, H., and Leung, S, T.: The Google File System, Proc. of ACM the 19th Symposium on Operating System Principle (SOPI 03), pp.29-43, (2003).
20. Kataoka, H., Duolikun, D., Enokido, T., and Takizawa, T.: Power Consumption and Computation Models of a Server with a multi-core CPU and Experiments, Proc. of IEEE the 29th International Conference on Advanced Information Networking and Applications (AINA-2015), pp.217-222, (2015).
21. Kataoka, H., Duolikun, D., Enokido, T., and Takizawa, T.: Evaluation of Energy-Aware Server Selection Algorithms, Proc. of the 9th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2015), pp.318–325, (2015).
22. Kataoka, H., Duolikun, D., Enokido, T., and Takizawa, T.: Multi-level Computation and Power Consumption Models, Proc. of the 18th International Conference on Network-based Information Systems (NBiS-2015), pp.40–47, (2015).
23. Kataoka, H., Sawada, A., Duolikun, D., Enokido, T., and Takizawa, T,: Energy-aware Server Selection Algorithms in a Scalable Cluster, Proc. of IEEE the 30th International Conference on Advanced Information Networking and Applications (AINA-2016), pp. 565-572, (2016).
24. Negus, C. and Boronczyk, T.: CentOS Bible, ISBN: 978-0-470-48165-3, (2009).
25. Rosa, J., D., la.: KVM Virtualization in RHEL 6 Made Easy, Dell Linux Engineering, (2011).
26. 2015 United Nations Climate Change Conference (COP21). https://en.wikipedia.org/wiki/2015 United Nations Climate Change Conference.
27. VMware Virtualization, http://www.vmware.com/jp.html
28. Watanabe, R., Duolikun, D., Enokido, T., and Takizawa, M.: An Eco Model of Process Migration with Virtual Machines, Proc. of the 19th International Conference on Network-based Information Systems (NBiS-2016), pp.292–297, (2016).