A Simple Server Selection Algorithm to Reduce Electric Power for Storage and Computation Processes

法政大学大学院理工学・工学研究科紀要 Vol.59(2018 年 3 月) 法政大学

A Simple Server Selection Algorithm to Reduce Electric Power for Storage and Computation Processes

澤田充広
Atsuhiro SAWADA
Supervisor: Professor Makoto Takizawa

法政大学大学院理工学研究科システム理工学専攻修士課程

Abstract
In application processes like Web and databases, files and databases are manipulated on servers. Thus, both CPU and storage resources are used to perform application processes. In this paper, we propose a computation model to give the expected termination time of each application process. Then, we propose an SGEAG (Simple Globally-Energy-Aware for General processes) algorithm to select a server to perform a new process issued by a client, which is expected to consume the minimum electric energy to perform not only the new process but also every current process. In the evaluation, we show the electric energy consumed by servers and the average execution time of processes can be more reduced in the SGEAG algorithm than the other algorithms.
Key Word: *Power consumption model; Computation model; improved MLCMG model; SGEAG algorithm; General processes; Storage operations;*

1 Introduction

In order to realize eco society, we have to reduce the electric energy consumed in information systems [10], [30], especially scalable systems like cloud computing systems [22] and IoT [11]. In hardware-oriented approaches to reducing the electric energy consumption of servers, energy-efficient devices like CPUs [14] [23] and storages [13] are developed. On the other hand, in our macro-level approach [6] [7] [8], we aim at reducing the total amount of electric energy [J] consumed by a whole server to perform application processes without considering the power consumption of each hardware device. In this paper, a *process* means an application process to be performed on a server.

In this paper, we propose an improved computation model of a server to perform general processes which use both CPU resources and storage drives. The MLCMG model [26] is improved so that computation and storage operations performed by each application process are uniformly handled. Here, computation steps and storage manipulation steps of each process are virtualized into virtual computation steps. Then, virtual computation rate and storage manipulation rate are defined in terms of virtual computation steps to show how much computation and storage manipulation are performed on a server for one time unit. Based on the improved MLCMG model, we newly propose an SGEAG (Simple Globally-Energy-Aware for General processes) algorithm to select a host server to perform general processes. Here, the total electric energy to be consumed by not only the host server but also the other servers is minimized as discussed in the GEAG algorithm. We evaluate the SGEAG algorithm compared with the random and round-robin algorithms in the simulation studies. We show the total electric energy consumption and total active time of servers in a cluster and the average execution time of processes can be more reduced in the SGEAG algorithm than the other algorithms in the evaluation.

In section 2, we present the power consumption model and improved computation model of a server to perform general processes. In section 3, we propose the SGEAG algorithm to select a host server. In section 4, we evaluate the SGEAG algorithm.

2 Multi-level Power Consumption and Computation Models

2.1 MLPCM model

A cluster S is composed of m (≥ 1) servers $s_1, \ldots, s_m$. Each server s_t is composed of np_t (≥ 1) homogeneous CPUs [14]. Each CPU is composed of cc_t (≥ 1) homogeneous cores. Each core supports ct_t (≤ 2) threads. Let ncp_t and nt_t be the total numbers of cores and threads of a server s_t, i.e. $ncp_t = np_t \cdot cc_t$ and $nt_t = ncp_t \cdot ct_t$. An active thread is one where at least one process is performed. An active server is a server where at least one thread is active, Let $CP_t(\tau)$ be a set of processes performed on a server s_t at time τ.

The electric power consumption $E_t(\tau)$ [W] of a server s_t to perform computation processes is $minE_t + ap_t(\tau) \cdot bE_t + ac_t(\tau) \cdot cE_t + at_t(\tau) \cdot tE_t$ where $ap_t(\tau)(\leq np_t)$, $ac_t(\tau)(\leq ncp_t)$, and $at_t(\tau)(\leq nt_t)$ are numbers of active CPUs, cores, and threads, respectively, at time τ in the MLPCM model [18] [19].

Next, suppose n processes are performed on a server s_t. It is more convenient and practical to give the electric power consumption of a server s_t in terms of number n of current processes performed. Here, let $nap_t(n)$, $nac_t(n)$, and $nat_t(n)$ be numbers of active CPUs, cores, and threads of a server s_t where n computation processes are performed. Processes are fairly allocated with CPUs, cores, and threads in a scheduling algorithm like round-robin algorithm [20]. Hence, we can assume $nap_t(n) = n$ if $n \leq np_t$, else np_t. $nac_t(n) = n$ if $n \leq ncp_t$, else ncp_t. $nat_t(n) = n$ if $n \leq nt_t$, else nt_t. The electric energy consumption $CE_t(n)$ of a server s_t to perform n processes is given as follows:

$$\begin{aligned} CE_t(n) &= minE_t + nap_t(n) \cdot bE_t \\ &+ nac_t(n) \cdot cE_t + nat_t(n) \cdot tE_t. \end{aligned} \quad (1)$$

The maximum electric power consumption $CE_t(n)$ is $maxCE_t$ [W] = $minE_t + np_t \cdot bE_t + ncp_t \cdot cE_t + nt_t \cdot tE_t$ where $n \geq nt_t$.

Next, we consider general processes which read data in a file while doing the computation for simplicity. $CP_t(\tau)$ is a set of processes which use CPU at time τ. $RP_t(\tau)$ is a set of processes which read data in storages on a server s_t at time τ. If a process p_i uses both CPU and storage at time τ, $p_i \in CP_t(\tau)$ and $p_i \in RP_t(\tau)$. $PP_t(\tau)$ is a set of all processes performed on a server s_t at time τ, i.e. $PP_t(\tau)$ = $CP_t(\tau) \cup RP_t(\tau)$. The electric power computation $E_t(\tau)$ [W] of a server s_t to perform general processes at time τ is given in the MLPCMG (MLPCM for General processes) model [27] as follows;

$$\begin{aligned} &E_t(\tau) = CE_t(|CP_t(\tau)|) + \delta_t(\tau) \cdot RE_t \\ &\text{where } \delta_t(\tau) = 1 \text{ if } |RP_t(\tau)| > 0, \text{else } 0. \end{aligned} \quad (2)$$

The maximum electric power consumption $maxE_t$ of a server s_t is $maxCE_t$ + RE_t where $|CP_t(\tau)|$ ($\geq nt_t$) processes use CPU and at least one process accesses to storages.

We measure the electric power consumed by a pair of servers, DSLab (Two Intel Xeon CPUs, 128GB memory, 4.8TB storage) s_D and Atria (Intel Core i7 CPU, 16GB memory, 2.0TB storage) s_A in our laboratory. The DSLab server s_D supports thirty two threads and the Atria server s_A supports eight threads. Figure 1 shows the electric power consumption of the servers s_D and s_A to perform n general processes and computation processes. Each general process uses both CPU and reads data of 1 [GB] in a file. $C + R$ shows that n general processes are performed and C means only n computation processes are performed. As shown in Figure 1, $minE_D = 126.1$, $bE_D = 30$, cE_D = 5.6, tE_D = 0.8, and $maxCE_D$ = 301.1 [W] for the DSLab server s_D and $minE_A$ = 41.3, bE_A = 15, cE_A = 4.7, tE_A = 1.1, and $maxCE_A$ = 89.5 [W] for the Atria server s_A. RE_D = 21 and $maxE_D$ = 322.1 [W] for the DSLab server s_D and RE_A = 2 and $maxE_A$ = 91.5 [W] for the Atria server s_A.

2.2 MLCMG model

Each general processes p_i is composed of read operations as storage operations in addition to computation operations. Let cp_i and sp_i denote subsequences of computation and storage operations of a process p_i, respectively.

A process p_i is at a time performed on a thread of a server s_t in a cluster S of m ($\geq$ 1) servers $s_1, \ldots, s_m$.

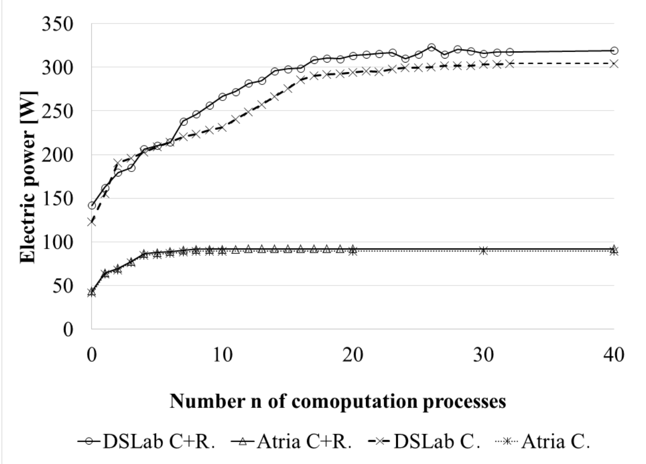

Figure1: Electric power consumption.

$minCT_{ti}$ shows the minimum execution time of computation operations, i.e. cp_i of a process p_i on a thread of a server s_t, where only the process p_i is performed on a thread without any other process. $minCT_i$ indicates the minimum one of the minimum execution time $minCT_{1i}$, $\cdots$, $minCT_{mi}$ on the servers $s_1, \ldots, s_m$, respectively. A thread of a server s_f is $fastest$ in a cluster S if $minCT_i$ = $minCT_{fi}$ for every process p_i. Here, the server s_f is referred to as $fastest$ in a cluster S. We assume one virtual computation step is performed on a thread of a fastest server s_f for one time unit [time unit (tu)]. That is, the thread computation rate CRT_f of a fastest server s_f is one [vs/tu]. The thread computation rate CRT of a cluster S is CRT_f of the fastest server s_f (CRT = CRT_f = 1). A server s_t supports the thread computation rate CRT_t = ($minCT_i$ / $minCT_{ti}$) $\cdot$ CRT = $minCT_i$ / $minCT_{ti}$ [vs/tu] ($\leq CRT$ = 1). If the computation operations cp_i of a process p_i are performed on a fastest server s_f without any other process, it takes $minCT_{fi}$ = $minCT_i$ [tu]. Hence, the amount VC_i of computation of a process p_i is defined to be $minCT_i$ [tu] $\cdot$ CRT_f [vs / tu] = $minCT_i$ [vs]. Thus, $minCT_i$ shows the total number VC_i of virtual computation steps.

The server computation rate $NSR_t(\tau)$ [vs/tu] of a server s_t at time τ is $at_t(\tau) \cdot CRT_t$ where $at_t(\tau)$ ($\leq nt_t$) is the number of active threads and nt_t is the total number of threads. As discussed in papers [16]-[19], the maximum computation rate $maxCR_t$ of a server s_t is nt_t $\cdot$ CRT_t. $maxCR$ shows the maximum one of maximum computation rates $maxCR_1, \ldots, maxCR_m$ of the servers $s_1, \ldots, s_m$. Computation resources are assumed to be uniformly allocated to each process on a server s_t in the round-robin (RR) algorithm [20]. The server computation rate $NSR_t(n)$ of a server s_t where n (= $|CP_t(\tau)|$) processes are performed is given as follows:

$$NSR_t(n) = \begin{cases} n \cdot CRT_t \text{ if } n \leq nt_t. \\ nt_t \cdot CRT_t (= maxCR_t) \text{ if } n > 0. \end{cases} \quad (3)$$

We assume each process p_i is fairly allocated with computation resources. The process computation rate $NPR_{ti}(n)$

of a process p_i performed concurrently with (n - 1) processes on a server s_t is $NPR_{ti}(n)$ = $NSR_t(n)$ / n.

For simplicity, we assume each process p_i only reads data in a file f_i different from every other process p_j ($f_i \neq f_j$). Here, b_i shows the size [B] of a file f_i. Let $maxRR_t$ indicate the maximum read rate [B/tu] of a server s_t. Suppose only storage operations sp_i of a process p_i are performed to just read data in a file f_i of the server s_t. It takes RT_{ti} time units [tu] for the process p_i to read every data in a file f_i. The minimum read time $minRT_{ti}$ of a process p_i to read every data in a file f_i is b_i / $maxRR_t$ [tu] where only the process p_i is performed without any other process on a server s_t. The read rate $NRR_t(n)$ [B/tu] of a server s_f where n processes reading data in storages are performed is given as follows:

$$NRR_t(n) = maxRR_t. \tag{4}$$

Each process p_i reads data in a file f_i at the rate $NRR_{ti}(n)$ where n processes read data in the storages:

$$NRR_{ti}(n) = NRR_t(n)/n. \tag{5}$$

The more number of processes read data, the smaller read rate of each of the processes.

In this paper, we take a fastest server s_f as a *canonical* server in a cluster S. The canonical read time cRT_i [tu] of a process p_i is defined to be $minRT_{fi}$ on the fastest server s_f. We assume one virtual computation step is performed on the canonical server s_f to read data of size $maxRR_f$ [B] for one time unit [tu]. That is, the canonical virtual read rate cRR is defined to be one [vs/tu]. Hence, cRR [vs/tu] $\cdot$ cRT_i [tu] ($= cRT_i$) virtual computation steps [vs] are performed to read every data in a file f_i on the fastest server s_f. The number VR_i of virtual computation steps of a process p_i to read every data in a file f_i is cRR [vs/tu] $\cdot$ cRT_i [tu] $= cRT_i = minRT_{fi}$ [vs]. The virtual read rate VRR_t [vs/tu] of a server s_t is defined as follows;

$$\begin{aligned} VRR_t &= cRR \cdot maxRR_t/maxRR_f \\ &= maxRR_t/maxRR_f. \end{aligned} \tag{6}$$

The virtual read rate VRR_f is assumed to be cRR ($= 1$) for the fastest server s_f. Even if a server s_t is slower than the fastest server s_f, i.e. $CRT_t < CRT_f$, the virtual read rate VRR_t may be larger than VRR_f depending on the read rates $maxRR_t$ and $maxRR_f$ of the storage drives. The virtual read rate $VRR_{ti}(n)$ of a process p_i on a server s_t where n (≤ 1) read processes are performed is VRR_t / n.

The canonical execution time cT_i [tu] of a process p_i in terms of minimum computation time $minCT_i$ and canonical read time CRT_i is defined as follows;

$$cT_i = minCT_i + cRT_i(= minRT_{fi}). \tag{7}$$

It takes cT_i [tu] to perform the process p_i on the fastest server s_f. The amount VP_i of virtual computation steps of a process p_i is defined to be VC_i + VR_i (= $minCT_i$ + cRT_i) [vs].

The computation ratio cr_i and the read ratio rr_i of a process p_i are VC_i / VP_i and VR_i / VP_i, respectively. Here, cr_i + rr_i = 1. In this paper, we assume the ratios of the numbers of virtual computation steps and virtual read steps of a process p_i performed for each time unit are invariant cr_i and rr_i, respectively. Let ncp_t be the number $|CP_t(\tau)|$ of processes which do the computation and nrp_t be the number $|RP_t(\tau)|$ processes which read data in storages. We assume virtual computation operations and virtual storage operations are uniformly distributed in a sequence of operations of each process. We define the virtual computation rate $VCR_{ti}(\tau)$ [vs/tu] and virtual read rate $VRR_{ti}(\tau)$ [vs/tu] of a process p_i as follows:

[Virtual computation (VC) and read (VR) rates]

$$VCR_{ti}(ncp_t) = \begin{cases} cr_i \cdot CRT_t \textbf{ if } ncp_t \leq nt_t. \\ cr_i \cdot maxCR_t/ncp_t \textbf{ if } ncp_t > nt_t. \end{cases} \tag{8}$$

$$VRR_{ti}(nrp_t) = rr_i \cdot VRR_t/nrp_t. \tag{9}$$

The variables vc_i and vr_i show the virtual computation laxity and virtual read laxity of a process p_i, respectively. That is, vc_i and vr_i virtual computation steps of a process p_i have to be still performed now. At time τ a process p_i starts on a server s_t, $vc_i = VC_i$ ($= minCT_i$) and vr_i $= VR_i$ ($= cRT_i$) [vs]. At each time τ, vc_i and vr_i are decremented by the virtual computation rate $VCR_{ti}(ncp_t)$ and virtual read rate $VRR_{ti}(nrp_t)$ [vs/tu], respectively. If both the laxities vc_i and vr_i are equal to or smaller than zero, the process p_i terminates at time τ.

Initially, a variable PP_t = ϕ for each server s_t. PP_t shows a set of processes performed on a server s_t.

[Virtual computation (VC) model]

```
at each time $\tau$,
  for each server $s_t$ in a cluster $S$, {
  /* $PP_t$ show $PP_t(\tau)$ */
    for every new process $p_i$ to be started on $s_t$, {
        $PP_t = PP_t \cup \{p_i\}$;
        $vc_i = VC_i$;
        $vr_i = VR_i$;
    }; /* for process $p_i$ end */
    for each process $p_i$ in $PP_t$, { /* decrement laxity */
        $vc_i$ = $vc_i - VCR_{ti}(\tau)$;
        $vr_i$ = $vr_i - VRR_{ti}(\tau)$;

        if $vc_i \leq 0$ and $vr_i \leq 0$, /* $p_i$ terminates */
          $PP_t$ = $PP_t - \{p_i\}$;
    }; /* for process $p_i$ end */
  }; /* for server $s_t$ end */
```

2.3 Estimation model

We discuss how to estimate at time τ the electric energy consumption of a server s_t to perform all the current processes and the termination time of each current process. In the estimation models discussed by previous studies [26] - [24], it takes time to calculate the expected termination time of each process and the laxity of each process has to be collected on a server. In this paper, we consider a model to simply estimate the termination time of each process on a server. We assume each current process p_i on a server s_t finishes the half $VC_i/2$ of the virtual computation steps and $VR_i/2$ of virtual read steps at time τ.

Suppose a set $NP_t(\tau)$ of nn_t (≥ 0) processes are newly issued to a server s_t where a set $PP_t(\tau)$ of current processes are performed by using CPU resources at time τ. Let n_t, ncp_t, and nrp_t be the numbers $|PP_t(\tau)|$, $|CP_t(\tau)|$, and $|RP_t(\tau)|$ of processes, respectively. Here, $ncp_t \leq n_t$, $nrp_t \leq n_t$ and $n_t \leq ncp_t + nrp_t \leq 2 \cdot n_t$. The total number VSC_t [vs] of virtual computation steps to be performed by all the current processes and new processes is $\sum_{p_j \in CP_t(\tau)} VC_j/2 + \sum_{p_j \in NP_t(\tau)} VC_j$ on a server s_t. The computation rate of a server s_t to perform ncp_t current processes and nn_t new processes is $NSC_t(ncp_t + nn_t)$ [vs/tu]. The expected execution time ECT_t of a server s_t to perform every computation operation is VSC_t / $NS_t(ncp_t + nn_t)$ [tu]. The total number VSR_t of virtual manipulation steps to be performed by all the current and new processes is on a server s_t $\sum_{p_j \in RP_t(\tau)} VR_j/2 + \sum_{p_j \in NP_t(\tau)} VR_j$ [vs] which are performed at rate $NRR_t(nt_t + nn_t)$ ($= maxRR_t$) [vs/tu]. Here, the expected execution time ERT_t to perform all the storage operations is VSR_t / $NRR_t(nrp_t + nn_t)$ [tu]. Thus, the expected termination time ECT_t and ERT_t of computation operations and storage operations of every current process in $PP_t(\tau)$ and every new process in $NP_t(\tau)$ are given as follows:

$$ECT_t = (\textstyle\sum_{p_j \in CP_t(\tau)} VC_j/2 + \sum_{p_j \in NP_t(\tau)} VC_j\ ) /NSR_t(ncp_t + nn_t). \tag{10}$$

$$ERT_t = (\textstyle\sum_{p_j \in RP_t(\tau)} VR_j/2 + \sum_{p_j \in NP_t(\tau)} VR_j\ ) /NRR_t(nrp_t + nn_t). \tag{11}$$

The expected electric energy consumption EE_t [W · tu] of a server s_t is given as follows:

$$EE_t = ECT_t \cdot CE_t(ncp_t + nn_t) + ERT_t \cdot RE_t. \tag{12}$$

In order to make the estimation easier, we assume cr_i is a constant $cr = \alpha$ (≤ 1), $rr_i = rr = 1 - \alpha$ for each process p_i. We also assume $VC_i = VC = \alpha$, and $VR_i = VR = 1 - \alpha$ for every process p_i. The expected termination time of computation and read operations of a server s_t where n (= $|PP_t(\tau)|$) processes are concurrently performed and k (= $|NP_t(\tau)|$) new processes are issued to the server s_t are given as follows:

$$MECT_t(n_t,\, k) = \alpha \cdot (n_t/2 + k)/NCR_t(n_t + k). \tag{13}$$

$$MERT_t(n_t,\, k) = (1 - \alpha) \cdot (n_t/2 + k)/NRR_t(n_t + k). \tag{14}$$

The expected electric energy consumption $MEE_t(n_t, k)$ [W · tu] of a server s_t is summation of electric energy $MECT_t(n_t, k)$ to be consumed by performing computation operations and electric energy $MERT_t(n_t, k)$ to be consumed by performing storage operations is given as follows:

$$MEE_t(n_t,\, k) = MECT_t(n_t, k) + MERT_t(n_t, k). \tag{15}$$

A server s_t gets idle if every computation operation and manipulation operation of every processes finishes. Hence, the expected termination time $MET_t(n_t, k)$ [tu] of a server s_t is given as follows:

$$MET_t(n_t,\, k) = max(MECT_t(n_t, k),\, MERT_t(n_t, k)). \tag{16}$$

3 Server Selection Algorithms

A client issues a process p_i to a cluster S of servers s_1, ..., s_m ($m \geq 1$). Then, one host server s_t is selected for performing the process p_i. The expected electric energy consumption EE_t and expected termination time ET_t of the server s_t are calculated by the simple estimation procedures $MEE_t(|PP_t(\tau)|, 1)$ and $MET_t(|PP_t(\tau)|, 1)$, respectively.

First, in the SLEAG (Simple Locally-Energy-Aware for General processes) selection algorithm [5], a host server s_t is selected in the cluster S, whose expected electric energy consumption EE_t to perform not only the new process p_i but also every current process in the set $PP_t(\tau)$ is minimum as follows:

[SLEAG algorithm]
```
for each server $s_u$ in a cluster $S$,
    $EE_u = MEE_u(|PP_u(\tau)|, 1)$;
select a host server $s_t$ where $EE_t$ is minimum in $S$;
perform a process $p_i$ on $s_t$;
```

As discussed in papers [18], [19], another server s_u ($\neq s_t$) where a new process p_i is not performed also consumes the electric energy while the process p_i is performed on the host server s_t. Even if the server s_u is idle, i.e. no process is performed, the server s_u just consumes the minimum electric power $minE_t$. We propose an $SGEAG$ ($Simple\ Globally\text{-}Energy\text{-}Aware\ for\ General\ processes$) algorithm to select a host server s_t to perform a new process p_i. Here, we take into account the total electric energy consumption of not only a host server s_t of a new process p_i but also the other servers in a cluster S. For a process p_i, a host server s_t is selected in the SGEAG algorithm as follows:

[SGEAG algorithm]
```
for each server $s_t$ in a cluster $S$, {
/* not only every current process but also a new process $p_i$
to be performed on $s_t$ */
    $NT_t = MET_t(|PP_t(\tau)|, 1)$; /* execution time */
    $NE_t = MEE_t(|PP_t(\tau)|, 1)$; /* energy */
/* only every current process to be performed on $s_t$*/
    $ET_t = MET_t(|PP_t(\tau)|, 0)$; /* execution time */
    $EE_t = MEE_t(|PP_t(\tau)|, 0)$; /* energy */
}; /* for $s_t$ end */
for each server $s_t$ in $S$, {
        for each server $s_u$ ($\neq s_t$) in $S$, {
            if $ET_u < NT_t$, /* $s_u$ terminates before $NT_t$ */
                    $NEE_u$ = $EE_u$ + $minE_u \cdot (NT_t - ET_u)$
            else $NEE_u$ = $EE_u \cdot NT_t$ / $ET_u$;
        }; /* for $s_u$ end */
        $GE_t$ = $NE_t$ + $\sum_{s_v \in S-\{s_t\}} NEE_v$;
} /* for $s_t$ end */
select a server $s_t$ where $GE_t$ is minimum in $S$;
perform a process $p_i$ on $s_t$;
```

First, we obtain the expected electric energy NE_t to be consumed by each server s_t to perform not only every current process but also a new process p_i by the procedures NT_t = $MET_t(|PP_t(\tau)|$, 1) and NE_t = $MEE_t(|PP_t(\tau)|$, 1). In addition, we obtain the expected electric energy consumption EE_t of each server s_t to perform only every current process by the procedures ET_t = $MET_t(|PP_t(\tau)|$, 0) and EE_t = $MEE_t(|PP_t(\tau)|$, 0). For each host server s_t, we obtain the expected electric energy NEE_u to be consumed by each other server s_u until time NT_t when every process terminates on the server s_t. An idle server s_u, i.e. $ET_u = 0$ consumes the electric energy $NEE_u = minE_u \cdot NT_t$. If every current process terminates on a server s_t before time NT_t, i.e. $ET_u < NT_t$, the server s_t consumes the electric energy $minE_t \cdot (NT_t - ET_u)$ after every current process terminates. Hence, the server s_u consumes the electric energy NEE_u = EE_u + $minE_u \cdot (NT_t$ - $ET_u)$. If $ET_u \leq NT_t$, NEE_u = $EE_u \cdot NT_t$ / ET_u. If a process p_i is to be performed on a server s_t, all the servers in the cluster S are expected to totally consume the electric energy GE_t = NE_t + $\sum_{s_u(\neq s_t)\in S} NEE_u$. A server s_t where the expected total electric energy consumption GE_t is minimum is selected as a host server of a process p_i.

4 Evaluation

4.1 Environments

A cluster S is composed of m ($\geq$ 1) real servers s_1, ..., s_m where n ($\geq$ 1) processes p_1, ..., p_n are performed. In the evaluation, we consider four real servers $s_1, \dots, s_4$ in our laboratory. A pair of the servers s_1 and s_4 are equipped with two and one Intel Xeon E5-2667 v2 CPU, respectively. A pair of the equipped with one Intel Corei7-6700K and Intel Xeon E5-2620 CPU, respectively. The performance parameters like thread computation rate CRT_t and electric energy parameters like the minimum electric power $minE_t$ of each server s_t are shown in Table 1. The threads of the servers s_1 and s_4 are the fastest, $CRT_1 = CRT_2 = 1$ [vs/tu]. The maximum computation rate $maxCR_4$ of the server s_4 is 32 $\cdot$ 1 = 32 [vs/tu] since thirty two threads are supported. On the other hands, $maxCR_1$ = 16 since the server s_1 supports sixteen threads. The server s_2 supports the thread computation rate CRT_2 = 0.7 and the maximum computation rate $maxCR_2 = 5.6$ since eight threads are equipped. The server s_3 is the slowest where CRT_3 = 0.5 and $maxCR_3$ = 0.5 $\cdot$ 12 = 6.0 since twelve threads are supported. The server s_4 is taken as a canonical server where virtual read rate VRR_1 = 1 since the server s_4 is the fastest.

The number n ($>$ 1) of processes p_1, ..., p_n are performed on the servers $s_1, \dots, s_4$. The parameters of the processes are shown in Table 2. The starting time $stime_i$ of each process p_i is randomly taken from 0 to $xtime$ - 1 [tu]. In the evaluation, $xtime$ is 1,000 [tu], i.e. 100 [sec]. The minimum computation time $minCT_i$ of each process p_i is randomly taken from mT (= 5) to xT (= 20) [tu]. The total number VC_i [vs] of virtual computation steps of each process p_i is $minCT_i$. The canonical read time cRT_i [tu] is $\alpha \cdot minCT_i$. In the evaluation, α = 0.025. The minimum execution time $minT_i$ of a process p_i is $minCT_i$ + cRT_i = (1 + α) $\cdot$ $minCT_i$ = 1.025 $\cdot$ $minCT_i$ [tu]. The total amount VR_i of virtual read steps of a process p_i is cRT_i $\cdot$ cRR = cRT_i. At time τ a process p_i starts, the virtual computation laxity vc_i is VC_i (= $minCT_i$) and the virtual read laxity vr_i is VR_i (= cRT_i) [vs]. At each time τ, the laxity vc_i and vr_i are decremented by the rates $VCR_{ti}(\tau)$ and $VRR_{ti}(\tau)$, respectively. In the simulation, the electric energy consumption EE_t [W $\cdot$ tu] and active time AT_t [tu] of each server s_t are obtained. The active time AT_t is time when the server s_t is active, i.e. at least one process is performed. The total electric energy consumption TEE of servers is $EE_1 + \cdots + EE_4$ and the total active time TAT is $AT_1 + \cdots + AT_4$. In the simulation, the termination time $etime_i$ of each process p_i is obtained. Here, the execution time et_i of the process p_i is $etime_i - stime_i$ + 1. The average execution time AET of n processes is (et_1 + $\cdots$ + et_n / n.

The random (RD), round-robin (RR), and SGEAG algorithms are performed on the same pair of a server configuration and a process configuration. In the RD algorithm, a server s_t is randomly selected for each new process p_i. In the RR algorithm, after a server s_t is selected for a previous process, a server s_{t+1} is selected for a next process. In the SGEAG algorithm, one host server s_t is selected where the total electric energy GE_t of all the servers $s_1, \dots, s_m$ is minimum. Then, 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 for each server s_t in each process configuration in the simulation for each algorithm. $etime$ is time when the simulation ends, i.e. a maximum one of $etime_1, \cdots, etime_n$.

4.2 Evaluation results

Figure 2 shows the total electric energy consumption TEE [W $\cdot$ tu] of the four servers $s_1, \dots, s_4$ for number n of processes in each algorithm. The total electric energy consumption TEE of the servers s_1, ..., s_4 is EE_1 + $\cdots$ + EE_4 where EE_t is the electric energy consumption of each server s_t. The total electric energy consumption TEE of the SGEAG algorithm is minimum in the algorithms. For example, the total electric energy consumption TEE of the SGEAG algorithm is 8% and 11% smaller than the RD and RR algorithms for n = 1,000 and n = 2,000, respectively. As shown here, the total electric energy consumption TEE of the servers can be reduced in the SGEAG algorithm compared with the RD and RR algorithms.

Figure 3 shows the total active time TAT of the servers $s_1, \dots, s_4$ for number n of processes. The total active time TAT is AT_1 + $\cdots$ + AT_n where AT_t is the active time of each server s_t. The total active time TAT of the SGEAG algorithm is shorter than the RD and RR algorithms. For example, the total active time TAT of the SGEAG algorithm is about 50 % of the total active time TAT of the RR and RD algorithms for n = 1,500. This meas, the servers are less loaded in the SGEAG algorithm than the RD and RR algorithms.

Figure 4 shows the average execution time AET [tu] of n processes $p_1, \ldots, p_n$ on the four servers $s_1, \ldots, s_4$. The average execution time AET of the SGEAG algorithm is shorter than the RD and RR algorithms. For example, the average execution time AET of the n processes in SGEAG algorithms is about 35% shorter than the RD and RR algorithms for n = 2,000. The average execution time AET of the RD and RR algorithms exponentially increases as the number n of processes increases. In the SGEAG algorithm, the average execution time AET almost linearly increases.

As shown in Figures 2, 3, and 4, the total electric energy consumption TET and total active time TAT of the servers and the average execution time AET of the processes can be more reduced in the SGEAG algorithm than the RD and RR algorithms.

Table. 1: Parameters of servers.

parameters	DSLab1 (s_1)	Atria (s_2)	Sunny (s_3)	DSLab2 (s_4)
np_t	1	1	1	2
ncp_t	8	4	6	8
nt_t	16	8	12	32
CRT_t [vs/tu]	1.0	0.7	0.5	1.0
$maxCR_t$ [vs/tu]	16	5.6	6.0	32
VRR_t [vs/tu]	1.0	0.56	0.8	1.0
$minE_t$ [W]	126.1	41.3	87.2	126.1
$maxE_t$ [W]	222.7	91.5	136.2	322.1
bE_t [W]	30	15	16	30
cE_t [W]	5.6	4.7	3.6	5.6
tE_t [W]	0.8	1.1	0.9	0.8
RE_t [W]	21	2	5	21

Table. 2: Parameters of processes.

parameters	values
n	number of processes $p_1, \ldots, p_n$.
α	computation - read ratio factor (α = 0.025).
$minCT_i$ [tu]	minimum computation time (5 $\sim$ 20).
cRT_i [tu]	canonical read time ($\alpha \cdot minCT_i$).
$minT_i$	$minCT_i + cRT_i = (1 + \alpha) \cdot minCT_i$.
VC_i [vs]	$minCT_i$ [vs].
VR_i [vs]	$\alpha \cdot minCT_i$ [vs].
$stime_i[tu]$	starting time of p_i ($0 \leq st_i < xtime$ - 1) [tu].
$xtime[tu]$	simulation time (= 1000) [tu].

5 Concluding Remarks

In this paper, we newly proposed the improved computation MLCMG model of a server to perform general processes which use both CPU and storages. In the improved MLCMG model, the expected termination time of each process is simply estimated by using only the number of processes performed on each server. Then, we proposed the SGEAG algorithm to select a host server to perform a process issued by a client. Here, a host server s_t is se-

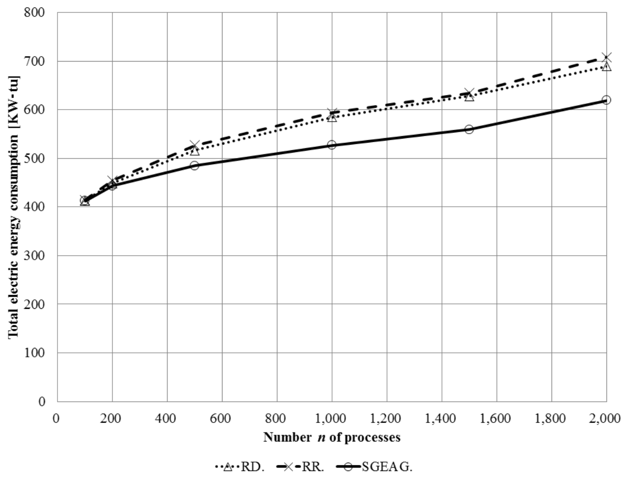

Figure2: Total electric energy (TEE) (α = 0.025, m = 4).

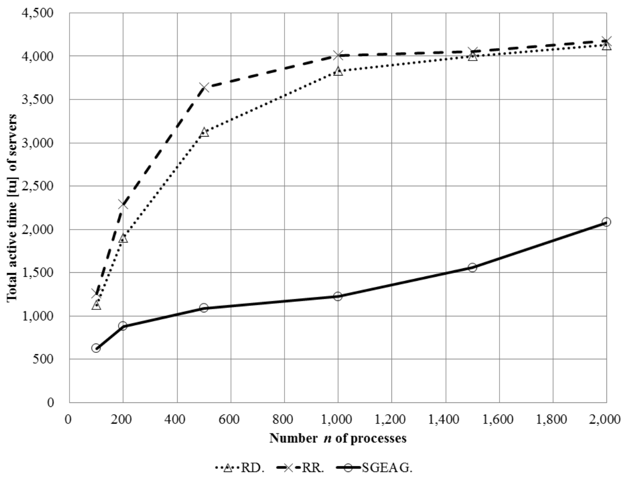

Figure3: Total active time (TAT) (α = 0.025, m = 4).

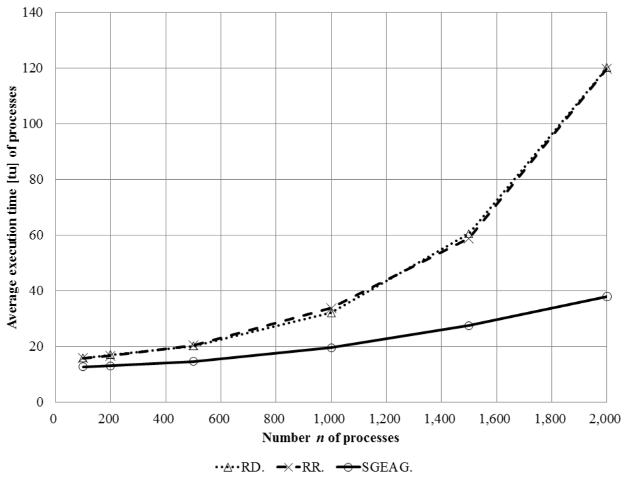

Figure4: Average execution time (AET) (α = 0.025, m = 4).

lected to perform a new process issued by a client so that the total electric energy to be consumed by the host server s_t and the other servers is minimized. We evaluated the SGEAG algorithm compared with the RD and RR algorithms in the simulation. We showed the total electric energy consumption and total active time of servers and the average execution time of processes can be reduced in the SGEAG algorithm compared with the RD and RR algorithms. The SGEAG algorithm implies the smallest electric energy consumption of the servers. The average execution time of the processes is also shorter in the SGEAG algorithm than the RD and RR algorithms. The SGEAG algorithm is the best since the electric energy consumption of the servers and the shortest average execution time of the processes are smaller than the RD and RR algorithms.

Acknowledgment

The author would like to express endless appreciation to my supervisor, Professor Makoto Takizawa, for his kindness, support, and instruction. He always gave the best support and help to the author.

References

[1] R. Bianchini and R. Rajamony: Power and Energy Management for Server Systems, *IEEE Computer*, vol. 37, no. 11, 2004, pp. 68-74.

[2] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa: Energy-aware Passive Replication of Processes, *Journal of Mobile Multimedia*, vol. 9, no. 1&2, 2013, pp.53-65.

[3] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa: Power Consumption Models for Migrating Processes in a Server Cluster, *Proc. of the 17th International Conference on Network-Based Information Systems (NBiS-2014)*, 2014, pp.15-22.

[4] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa: Energy-Efficient Dynamic Clusters of Servers, *Journal. of Supercomputing*, vol.71, issue 5, 2015, pp.1642–1656.

[5] D. Duolikun, H. Kataoka, T. Enokido, and M. Takizawa: Simple Algorithms for Selecting an Energy-efficient Server in a Cluster of Servers *International Journal. of Communication Networks and Distributed Systems (IJCNDS)*, 2017, (Accepted).

[6] T. Enokido, A. Aikebaier, and M. Takizawa: A Model for Reducing Power Consumption in Peer-to-Peer Systems, *IEEE Systems Journal*, vol.4, issue 2, 2010, pp.221-229.

[7] 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, 2011, pp.2097–2105.

[8] 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, issue 2, 2014, pp.1627-1636.

[9] T. Enokido, A. Aikebaier, and M. Takizawa: An Integrated Power Consumption Model for Communication and Transaction Based Applications, *Proc. of IEEE the 25th International Conference on Advanced Information Networking and Applications (AINA*-2011), 2011, pp. 627-636.

[10] S. Ghemawat, H. Gobioff, and S.-T. Leung: "The Google Files System," in *Proc. of the 19th ACM Symposium on Operating Systems Principles*, (*SOSP′*03), 2003, pp.29-43.

[11] J. Hoeller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, and D. Boyle: From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence. Elsevier, 2014, 352 pages.

[12] HP server: http://h50146.www5.hp.com/products /servers/proliant/system_pdf/dl360pgen8.pdf.

[13] T. Inoue, A. Aikebaier, T. Enokido, and M. Takizawa: Power Consumption and Processing Models of Servers in Computation and Storage Based Applications, *Journal of Mathematical and Computer Modeling*, vol.58, issue 5&6, 2013, pp.1475-1488.

[14] Intel Xeon Processor 5600 Series: The Next Generation of Intelligent Server Processors, white paper [online]. Available: http://www.intel.com/content/www/us/en/processors /xeon/xeon-5600-brief.html, 2010.

[15] Job Scheduling Algorithms in Linux Virtual Server, http://www.linuxvirtualserver.org/docs /scheduling.html, 2010.

[16] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa: 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 Workshops (WAINA*-2015), 2015, pp.318-325.

[17] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa: Evaluation of Energy-Aware Server Selection Algorithms, *Proc. of the 9th international Conference on International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS*-2015), 2015, pp.318-326.

[18] H. Kataoka, D. Duolikun, T. Enokido, and M. Tak-

izawa: Multi-level Computation and Power Consumption Models. *Proc. of the18th International Conference on Network-Based Information Systems (NBiS-2015)*, 2015, pp.40-47.

[19] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa: Energy-aware Server Selection Algorithms in a Scalable Cluster, $Proc.\ of\ IEEE\ the\ 30th\ International\ Conference\ on\ Advanced\ Information\ Networking\ and\ Applications$ ($AINA$-2016), 2016, pp.565-572.

[20] Linux distribution. Available: http://itpro.nikkeibp.co.jp/article/COLUMN/20120223 /382669/.

[21] Metaprotocol Corp: "UWmeter" Online'. Available: http://www.metaprotocol.com/UWmeter /Feautures.html, 2011.

[22] Natural Resources Defense Council (NRDS): Data Center Efficiency Assessment - Scaling up Energy Efficiency across the Data Center Industry: Evaluating Key Drivers and Barriers. http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf, 2014.

[23] Qualcomm snapdragon. Available: http://www.qualcomm.co.jp/products/snapdragon,

[24] A. Sawada, S. Nakamura, T. Enokido, and M. Takizawa: Eco Model of Storage-based Servers. *Proc. of the NBiS Workshops*, 2015, pp.407-411.

[25] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa: Energy-aware Clusters of Servers for Storage and Computation Applications. *Proc. of IEEE the 30th International Conference on Advanced Information Networking and Applications (AINA-2016)*, 2016, pp.400-407.

[26] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa: Energy-aware Clusters of Servers for Storage and Computation Applications. *Proc. of the 10-th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2016)*, 2016, pp.162-169.

[27] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa: Energy-aware Server Selection Algorithms for Storage and Computation Processes. *Proc. of the 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (BWCCA-2016)*, 2016, pp.45-56.

[28] A. Sawada, S. Nakamura, T. Enokido, and M. Takizawa: Algorithm to Energy-efficiently Select a Server for a General Process in a Scalable Cluster. *Proc. of IEEE the 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA-2017)*, 2017, pp.400-407.

[29] A. Sawada, S. Nakamura, T. Enokido, and M. Takizawa: Simple Energy-Aware Algorithms to Selecting a Server for Storage and Computation Processes in a Cluster. *Proc. of the 11th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2017)*, 2017, pp.98-109.

[30] United Nations Framework Convention on Climate Change (UNFCCC) Available: https://en.wikipedia.org/wiki/Kyoto Protocol.