A Process Migration Approach to Energy-efficient Computation in a cluster of Servers

DILAWAER, Duolikun / ディラワリ, ドリクン

(開始ページ / Start Page)
1
(終了ページ / End Page)
9
(発行年 / Year)
2016-03-24
(学位授与年月日 / Date of Granted)
2016-03-24
(学位名 / Degree Name)
修士(工学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

2015 年度修士論文

論文題名　A Process Migration Approach to Energy-efficient Computation in a Cluster of Servers

指導教員　滝沢　誠

大学院理工研究科

システム工学専攻修士課程

14R6103

ディラワリ　ドリクン

氏名　迪拉瓦尔　多里坤

***Abstract*—Application processes have to be efficiently performed on servers in a cluster with respect to not only performance but also energy consumption. In this paper, we consider a process migration (MG) approach to energy-efficiently performing application processes on servers in a cluster. First, a client issues an application process to a server in a cluster. A process performed on a current server is migrated to another server if the server is expected to consume smaller electric energy to perform the process than the current server and the deadline constraint on the process is satisfied on the server. In the evaluation, the total energy consumption of servers is shown to be smaller and the average execution time of each process to be shorter in the MG algorithm than the round robin and random algorithms.**

***Keywords*-Energy-aware cluster; Power consumption model; Computation model; Process migration; Energy-efficient process migration;**

I. Introduction

In a cluster of servers like cloud computing systems [16], [19], application processes have to be efficiently preformed on servers in terms of not only performance but also energy consumption. The power consumption models of a server to perform types of applications are purposed in papers [8], [9], [10], [11], [12].

In papers [1], [13], [14], the energy-aware active replication of a process [2] on multiple servers is discussed. In order to reduce the electric energy consumption of a server cluster, the algorithm where the other replicas are forced to terminate once one replica successfully terminates is discussed [13]. Furthermore, every replica is not simultaneously started as discussed in the paper [14]. In papers [4], [5], the passive replication [2] of a process is discussed to reduce the total energy consumption of a cluster, where only a primary replica of the process is performed. In papers [21], [22], a mobile agent approach is discussed where a process manipulates databases while moving around servers. Here, a mobile agent is passively replicated, where a primary replica of the mobile agent is performed while moving around servers and the other secondary replicas are not performed.

In this paper, a process performed on a server is migrated to another server to efficiently perform the process in terms of performance and energy consumption. A client first issues a request process to a server s_t in a server cluster. Than, the process is performed on the server s_t. Even if the server s_t is lightly loaded when the process is started, the server s_t might be later overloaded and consume more electric energy and longer time to perform the process. Here, suppose another server s_u is expected to consume smaller electric energy to perform up the process than the current server s_t. In addition, the deadline constraint of the process is satisfied even if the process is migrated to the server s_u. Here, the process is migrated to the server s_u and performed on the server s_u. We discuss how to estimate electric energy to be consumed by a server to perform all the current processes and how to estimate when each current process terminates under an assumption that no additional process starts. By using the estimation models of electric energy consumption and termination time, we discuss the energy-efficient migration (MG) algorithm for each process to decide on whether the process stays on the current server or is migrated to another server. If a process can be energy-efficiently performed on another server s_u than the current server, the process is migrated to the server s_u.

We evaluate the MG algorithm in terms of total energy consumption of a cluster and average execution time of each process compared with random (RD) and round-robin (RR) algorithms. We show the total electric energy consumption of the cluster can be reduced and average execution time of each process can be shorter in the MG algorithm than the other algorithms.

In section II, we present how to estimate the power consumption of servers and the execution time of each process. In section III, we discuss the MG algorithm to select a server in a cluster for each process. In section IV, we evaluate the MG algorithm in terms of total energy consumption of the cluster and average execution time of a process.

II. EXPECTED COMPUTATION AND POWER CONSUMPTION

A. *Expected computation*

The more number of processes are concurrently performed on a server, the longer time it takes to perform each of the processes. We take the simple computation (SC) model [7], [9], [10] to perform processes on a server. Suppose a cluster S is composed of servers $s_1, \ldots, s_n$ ($n \geq 1$). It takes $minT_{ti}$ [sec] to exclusively perform a process p_i without any other process on a server s_t. Let $minT_i$ be the minimum one of $minT_{1i}$, …, $minT_{ni}$ to perform exclusively a process p_i on serves s_1, …, s_n, respectively, in the cluster S.

The normalized maximum computation rate $maxF_{ti}$ ($\leq$ 1) of the process p_i is $minT_i$ / $minT_{ti}$ on the server s_t. The normalized computation rate $F_{ti}(\tau)$ ($\leq maxF_{ti}$) of a process p_i shows how much amount of computation of the process p_i is performed on the server s_t at time τ [9], [10], [12]. Let p_{ti} denote a process p_i performed on a server s_t. Suppose a process p_{ti} starts at time st and ends at time et. Here, $\sum_{\tau}^{et} = s_t\ F_{ti}(\tau)\ d\tau = minT_i$ [sec]. Let $CP_t(\tau)$ be a set of processes concurrently performed on a server s_t at time τ. The computation rate $F_t(\tau)$ of a server s_t at time τ is $\sum_{p_{ti} \in CP_t(\tau)} F_{ti}(\tau)$. The computation rate $F_t(\tau)$ is assumed to be fairly allocated to each current process p_i, i.e., $F_{ti}(\tau) = F_t(\tau)$ / $\mid CP_t(\tau) \mid$. $maxF_t$ indicates the maximum computation rate of a server s_t. If only a process p_i is exclusively performed on a server s_t at time τ, $F_t(\tau) = F_{ti}(\tau) = maxF_t$. $maxF_{ti}$ shows the maximum computation rate of a process p_i on a server s_t. Here, $maxF_{ti} = minT_i$ / $minT_{ti}$ ($\leq$ 1) for every process p_i. The more number of processes are concurrently performed at time τ, the smaller computation rate $F_t(\tau)$.

[Computation rate] $F_t(\tau) = \alpha_t(\tau) \cdot maxF_t$.

Here, $\alpha_t(\tau)$ (> 0) is the degradation function of a server s_t. Here, $\alpha_t(\tau) = 1$ if $CP_t(\tau) \leq maxN_t$, else $a_t^{CP_t(\tau)-1}$ in this paper. The constant a_t is a degradation factor ($a_t \leq 1$). That is, the execution time T_{it} of a process p_{it} is linearly increases as the number of processes concurrently performed with the process p_i if $CP_t(\tau) \leq maxN_t$. For example, $maxN_t = 200$ and $\alpha_t = 0.99$ in the evaluation of this paper.

Suppose a process p_{ti} starts on a server s_t at time st_i. The computation $\sum_{\tau=st_i}^{\tau} F_{ti}(\tau)\ d\tau$ of the process p_{ti} is already performed before time τ. The *computation laxity* $lc_{ti}(\tau)$ is $minT_i - \sum_{\tau=st_i}^{\tau} F_{ti}(\tau)$ which of has to be furthermore performed on the server s_t after time τ. At each time τ, $lc_{ti}(\tau + 1) = lc_{ti}(\tau) - F_{ti}(\tau)$. If τ the computation laxity $lc_{ti}(\tau_t)$ gets 0, the process p_{ti} terminates.

B. Expected energy consumption

In this paper, a term *process* stands for a application process. In the simple power consumption (SPC) model [1], [8], [9] of a server, the electric power consumption $E_t(\tau)$ of a server s_t at time τ is either the minimum $minE_t$ or the maximum $maxE_t$. If at least one process is performed on a server s_t at time τ [W], $E_t(\tau) = maxE_t$. Otherwise, $E_t(\tau) = minE_t$. The total electric energy $TE_t(\tau_1, \tau_2)$ consumed by a server s_t from time τ_1 to time τ_2 is $\sum_{\tau=\tau_1}^{\tau_2} E_t(\tau)$ [Ws].

For each current process p_{ti} in the set $CP_t(\tau)$, the computation laxity $lc_{ti}(\tau)$ has to be furthermore performed on a server s_t after time τ. As discussed in papers [9], [10], [11], we can estimate termination time by when each current process p_{ti} in $CP_t(\tau)$ is expected to terminate on a server s_t if no additional process is performed on the server s_t after time τ according to the SC model [9], [10]. In this paper, one unit time is 100 [msec] since we can measure the power consumption of a server every 100 [msec] [9], [10]. The expected termination time $ETP(s_t, CP_t(\tau), p_i, \tau)$ is given as time τ_t in the following procedure:

```
$lc = lc_{ti}(\tau)$;    /* laxity of $p_{ti}$ */
$\tau_i = \tau$;    /* current time */
while ( $lc > 0$)
do {
  $lc = lc - F_{ti}(\tau_t)$;
  $\tau_i = \tau_i + 1$;
}; /* $p_{ti}$ terminates at $\tau_i$ */
$CP_t(\tau_i + 1) = CP_t(\tau_i) -$ {$p_{ti}$};
```

Here, the normalized computation rate $F_{ti}(\tau)$ at time τ is $\alpha_t(\tau) \cdot maxF_t$ / $\mid CP_t(\tau) \mid$ as discussed in the preceding subsection. The computation rate $F_{ti}(\tau)$ monotonically decreases as the number of processes concurrently performed on a server s_t increases at each time τ.

A variable lc_i shows the computation laxity of a process p_{ti} and CP denotes a set $CP_t(\tau)$ of current processes on a server s_t. The expected termination time $ET(s_t, CP_t(\tau), \tau)$ by when every process in a current process set $CP_t(\tau)$ is obtained as time τ_t by the following procedure:

```
$CP = CP_t(\tau)$;
$lc_i = lc_{ti}(\tau)$ for each process $p_{ti}$ in $CP$;
$\tau_t = \tau$;    /* current time */
while ($CP \neq \varphi$)
do {
    for each process $p_{ti}$ in $CP$
    do {
      $lc_i = lc_i - F_{ti}(\tau_t)$;    /* $p_{ti}$ is performed */
     if $lc_i = 0$, $CP = CP -$ {$p_{ti}$}; /* $p_{ti}$ terminates */
    };
    $\tau_t = \tau_t + 1$;
};
```

Every current process in $CP_t(\tau)$ is expected to terminate by time τ_t under an assumption that no process additionally starts after time τ. Here, the server s_t is expected to consume the amount $EE(s_t, CP_t(\tau), \tau)$ of electric energy to perform every current process in the current process set $CP_t(\tau)$ at time τ. The expected energy consumption $EE(s_t, CP_t(\tau), \tau)$ is $(\tau_t - \tau) \cdot maxE_t$ to perform all the current processes of time τ on a server s_t.

III. Server Selection

A. Process migration

Suppose a cluster S is composed of multiple servers $s_1, \ldots, s_n$ ($n \geq 1$) and clients which are interconnected in an underlying reliable network N. Each server s_t supports clients with computation service.

A client c_s first finds a server s_t in the cluster S and issues the process p_i to a server s_t. Every process p_i is assumed to do the computation in this paper. The process p_i is performed on the server s_t. Then, the process p_i is migrated to another server s_u as shown in Figure 1. If the process p_i terminates on the server s_u, the reply is sent to the client c_s. Here, the process p_i is referred to as *migrated* and the servers s_t and s_u are *migrated servers* of the process p_i.

A process on a current server s_t is migrated to another server s_u in a cluster S so that not only some performance requirement of the process p_i like deadline constraint dl_i is satisfied but also the electric energy to be consumed by the servers_u is smaller than the server s_u. We discuss migration conditions that a process on one server is migrated to another server. Suppose a process p_i is performed on a server s_t at time τ. There are two ways to perform the process p_i [Figure 2]:

1 The process p_i is performed on the server s_t without migrating to another server.
2 The process p_i is perform to another server s_u.

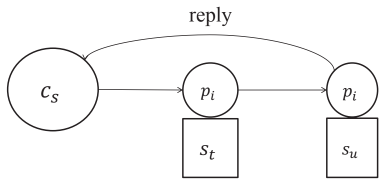

Figure 1. Migration of a process.

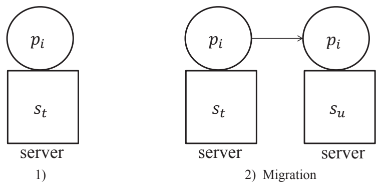

Figure 2. Process migration.

First, suppose that the process p_i stays on the server s_t at time τ. Here, the server s_t is expected to consume electric energy $EE(s_t, CP_t(\tau), \tau)$ to perform all the current processes $CP_t(\tau)$ of time τ. It is expected for every process in the set $CP_t(\tau)$ to terminate on the server s_t by time $ET(s_t, CP_t(\tau), \tau)$ and for each process p_i in $CP_t(\tau)$ to terminates at time $ETP(s_t, CP_t(\tau), p_i, \tau)$.

Next, suppose the process p_i is migrated to the server s_u from the current server s_t at time τ. The energy consumption of the server s_t is expected to decrease to $EE(s_t, CP_t(\tau) - \{p_i\}, \tau)$ because one current process p_i leaves the server s_t. The process p_i has to be transmitted to the server s_u. It is assumed to take δ_i time units to migrate the process p_i on a server to another server. Hence, the process p_i starts on the server s_u at time $\tau + \delta_i$ after the process p_i is transmitted from the other server s_t to the server s_u at time τ. On the other hand, the server s_u consumes more amount of electric energy because the process p_i is additionally performed after time $\tau + \delta_i$. The server s_u is expected to consume total energy $EE(s_u, CP_u(\tau+\delta_i) \cup \{p_i\}, \tau+\delta_i)$ [Ws] to perform the process p_i and current processes $CP_u(\tau + \delta_i)$ of time $\tau + \delta_i$. The expected termination time of the process p_i and every current process on the server s_u at time $\tau + \delta_i$ is also changed with $ET(s_u, CP_u(\tau + \delta_i) \cup \{p_i\}, \tau + \delta_i)$.

We have to obtain the current process set $CP_u(\tau + \delta_i)$ on a server s_u at time $\tau + \delta_i$. Current processes in the set $CP_u(\tau)$ are performed on the server s_u from time τ to time $\tau + \delta_i$. The computation laxity $lc_{uj}(\tau)$ of each process p_{uj} in $CP_u(\tau)$ is decremented by the normalized computation rate $F_{uj}(\tau)$. If the computation laxity $lc_{uj}(\tau')$ gets 0 at time τ' ($\tau \leq \tau' \leq \tau + \delta_i$), the process p_{uj} is removed in the process set $CP_u(\tau + \delta_i)$. The current process set $CP_u(\tau + \delta_i)$ is estimated by the following procedure:

```
for $x = \tau, \cdots, \tau + \delta_i$
do { $F = \alpha_t(\tau) \cdot maxF_t$ / $|CP_u(x)|$;
        for   every process $p_{uj}$ in $CP_u(x)$
        do {
            $lc_{uj}(x+1) = lc_{uj}(x) - F$;
         if   $lc_{uj}(x+1) = 0$,
             $CP_u(x+1) = CP_u(x)$ - $\{p_{uj}\}$;
         };
};
```

B. Server selection

A process p_i on a current server s_t can be migrated to another server s_u if the following migration (MG) conditions are satisfied:

[Migration conditions]

1 [Energy condition] $EE(s_t, CP_t(\tau) - \{p_i\}, \tau) < EE(s_u, CP_u(\tau + \delta_i) \cup \{p_i\}, \tau + \delta_i)$.
2 [Performance condition 1] $ETP(s_u, CP_u(\tau + \delta_i) \cup \{p_i\}, p_i, \tau + \delta_i) + \delta_i \leq dl_i - \tau$.
3 [Performance condition 2] $ETP(s_u, CP_u(\tau + \delta_i) \cup \{p_i\}, p_i, \tau + \delta_i) + \delta_i \leq ETP(s_t, CP_t(\tau), p_i, \tau)$.

The energy condition indicates that a smaller amount of electric energy is consumed by a server s_u than a current server s_t. In addition to the energy condition, a process p_i has to satisfy the following performance conditions.

The first Performance condition shows that a process p_i has to terminate by the deadline dl_i. The second Performance condition means that it has to take a shorter time to perform every current process on a server s_u than a current server s_t if the process p_i on the server s_t is migrated to the server s_u. In Figure 3, if a process p_i is performed on a server s_t at time τ, the process p_i is expected to terminate at time $\tau_2 = ETP(s_t, CP_t(\tau), p_i, \tau)$. If the process p_i on the server s_t is migrated to a server s_u at time τ, the process p_i is expected to terminate at time $\tau_1 = ETP(s_u, CP_u(\tau+\delta_i) \cup \{p_i\}, p_i, \tau + \delta_i)$. Here, the computation time to perform the process p_i can be reduced if the process p_i is migrated to the server s_u, i.e. $(\tau_2 - \tau) > (\tau_1 - \tau)$.

Suppose the first condition is not satisfied. Suppose the deadline dl_i of a process p_i is specified as performance constraint. If $ETP(s_u,\ CP_u(\tau + \delta_i) \cup \{p_i\},\ p_i,\ \tau + \delta_i) + \delta_i \leq dl_i - \tau$, the process p_i can be expected to terminate on the server s_u by the deadline dl_i. Hence, the process p_i can be migrated to the server s_u. Otherwise, the process p_i might not terminate by the deadline dl_i if the process p_i is migrated to the server s_u.

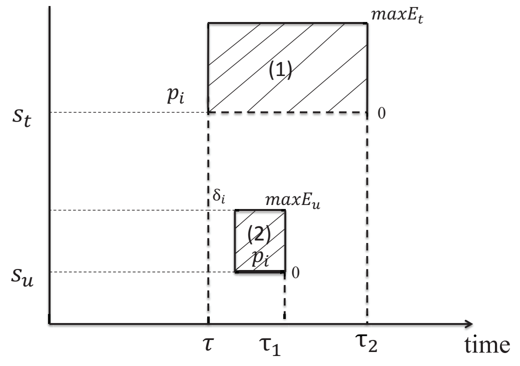

Figure 3. Expected termination time.

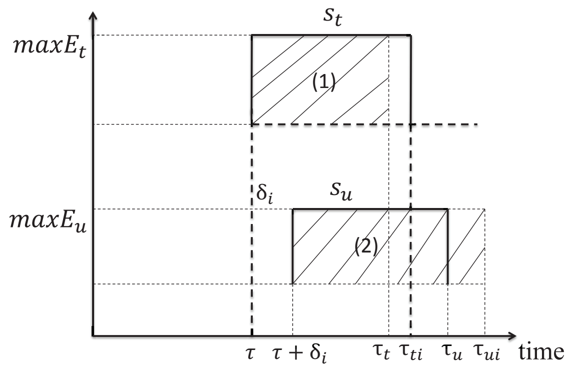

Figure 4. Expected energy consumption.

In Figure 4, τ_{ti} shows time by when every current process in $CP_t(\tau)$ terminates, i.e. $\tau_{ti} = ET(s_t,\ CP_t(\tau),\ \tau)$ and $\tau_u = ET(s_u,\ CP_u(\tau),\ \tau)$ where a process p_i is performed on the server s_t at time τ. Suppose the process p_i on the server s_t is migrated to the server s_u. Since the process p_i is not performed on the server s_t after time τ, the expected termination time τ_t of all the processes in $CP_t(\tau)$ is $ET(s_t,\ CP_t(\tau) - p_i,\ \tau)$. Here, $\tau_{ti} < \tau_t$ since the process p_i is migrated to the server s_u. The process p_i starts on the server s_u at time $\tau + \delta_i$. The expected termination time τ_{ui} of processes in $CP_u(\tau + \delta_i)$ is $ET(s_u,\ CP_u(\tau + \delta_i) \cup \{p_i\},\ \tau + \delta_i) + \delta_i$. $\tau_{ti} < \tau_t$ since the process p_i is additionally performed. The hatched areas (1) and (2) show the total energy consumption of the servers s_t and s_u, respectively, where the process p_i is migrated to the server s_u.

If there are multiple servers which satisfy the migration conditions, a server s_u where the expected energy consumption $EE(s_u,\ CP_u(\tau + \delta_i) \cup \{p_i\},\ \tau + \delta_i)$ is minimum is selected in the cluster S.

A server s_u is selected for a process p_i with a deadline constraint dl_i on a current server s_t at time τ as follows:

```
    $E = EE(s_t,\ CP_t(\tau),\ \tau)$;
    $T = dl_i - \tau$;   /* deadline of a process $p_i$*/
**for** each server $s_u$ in a cluster $S$
**do** {
    **if** ($EE(s_u,\ CP_u(\tau + \delta_i) \cup \{p_i\},\ \tau + \delta_i) < E$) {
      **if** ($ETP(s_u,\ CP_u(\tau + \delta_i) \cup \{p_i\},\ p_i\ \tau + \delta_i)$ +
        $\delta_i < T$) {   /* deadline is satisfied */
      $E = EE(s_u,\ CP_u(\tau + \delta_i) \cup \{p_i\},\ \tau + \delta_i)$;
      $T = ET(s_u,\ CP_u(\tau + \delta_i) \cup \{p_i\},\ \tau + \delta_i)$;
      $s = s_u$;
      };
    };
};
```

The MG conditions are checked every γ_i time units if a more number of processes are performed than a process p_i starts on a server s_t. Here $\gamma = maxT_i$ / 4.

IV. Evaluation

A. Environnent

We evaluate the energy-efficient process migration (MG) algorithm in terms of total energy consumption and total execution time. We consider a cluster S composed of n servers $s_1,\ \cdots,\ s_n$. Each server s_t follows the simple power consumption model [9], [10] with maximum power consumption $maxE_t$ and minimum power consumption $minE_t$. In this evaluation, $maxE_t$ is randomly taken out of 1,000 to 2,000 [W] and $minE_t$ is randomly taken out of 800 to 1,000 [W] for each server s_t. In each server s_t, the maximum normalized computation rate $maxF_t$ is randomly taken out of 0.5 to 1.0. The degradation constant α_t = 1 for $CP_t(\tau) \leq maxN_t$ and $maxN_t$ = 200. For $CP_t(\tau) > maxN_t$, α_t is randomly taken out of 0.99 to 1.0. The computation rate $F_t(\tau)$ of a server s_t is given $a_t^{l-maxN_t-1} \cdot maxF_t$ for number $l = |\ CP_t(\tau)\ |$ of processes concurrently performed at time τ as presented in this paper.

Totally l ($\geq$ 1) processes are performed on servers in the cluster S. For each process p_i, the starting time st_i is randomly taken from 0 to $xtime$. In this evaluation, the simulation time $xtime$ is 10,000 time units. One time unit is assumed to be 100 [msec]. That is, $xtime$ = 10,000 [msec].

The minimum computation time $minT_i$ of each process p_i is randomly taken out of 10 to 20 time units. The simulation ends if every process terminates.

In the evaluation, we consider three selection algorithms, random (RD), round robin (RR), and energy-efficient process migration (MG) algorithms to select a server for each process p_i. In the RD algorithm, one server is randomly selected for each process p_i in the clusters of n servers. In the RR algorithm, a server s_1 is selected for a first process. A server s_2 is selected for a next coming process. Thus, a server s_t is selected for a process after a server s_{t-1}. Here, t shows t modulo $n + 1$. In the evaluation, the servers in the cluster S are randomly ordered. In the MG algorithm, a server s_t whose expected power consumption is minimum is selected for each process p_i. The process p_i is performed on the server s_t. Every $\gamma_i = minT_t$ / 4 time units the process p_i checks if a more number of processes are concurrently performed than the process p_i starts on a server s_t. If so, the migration (MG) conditions are checked. If a server s_u which satisfies the MG conditions, i.e. s_u is expected to consume a smaller amount of electric energy to perform processes than the current server s_t, the process p_i is migrated to the server s_u. The delay time δ_i to migrate the process p_i to another server is the half of the maximum minimum computation time, i.e. $\delta_i = 20$ / $2 = 10$ time units.

B. Evaluation results

The cluster S is composed of n (≥ 1) servers $s_1, \cdots, s_n$. Figures 5 and 6 show the total energy consumption [Ws] of the servers $s_1, \cdots, s_n$ to perform l processes on servers of the cluster S in the MG, RR, and RD algorithms for $n = 8$ and 24, respectively. As shown in Figures 5 and 6, the total energy consumption of the servers is smaller in the MG algorithm than the RR and RD algorithms. The RR and RD algorithms imply almost the same energy consumption. For example, the total energy consumption in the MG algorithm is about 70% in the RR and RD algorithms for $l = 1{,}400$ for $n = 8$ as shown in Figure 5. For $n = 8$, every server is heavily loaded. For $n = 24$, since servers are less loaded, processes can be migrated to other servers so that the total energy consumption is reduced. Hence, the energy consumption of the MG algorithm is less reduced for $n = 8$ than $n = 24$. For example, the total energy consumption of the MG algorithm is about 60% of the RR and RD algorithms for $n = 24$ as shown in Figure 6.

Figure 7 shows the average execution time of each process p_i for $n = 8$. The average execution time is shorter in the MG algorithm than the RR and RD algorithms. The average execution time of the MG algorithm does not change if more number of processes are performed.

Figure 8 shows the number of processes which are migrated on eight servers ($n = 8$) in the MG algorithm. There is no process which migrates to another server for $l < 400$. For example, about 20% of the processes are migrated for $l = 1{,}000$ while about 75% of the processes are migrated for $l = 1{,}600$.

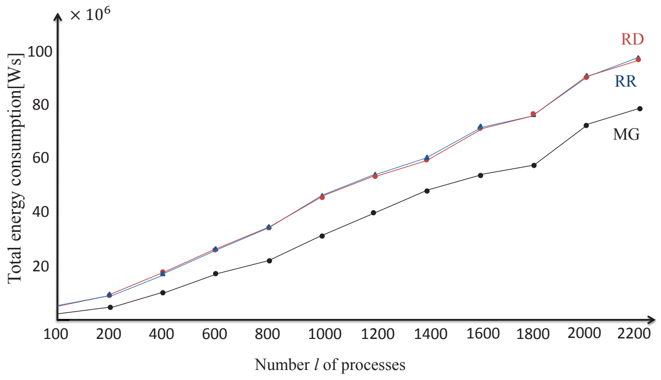

Figure 5. Total energy consumption ($n = 8$).

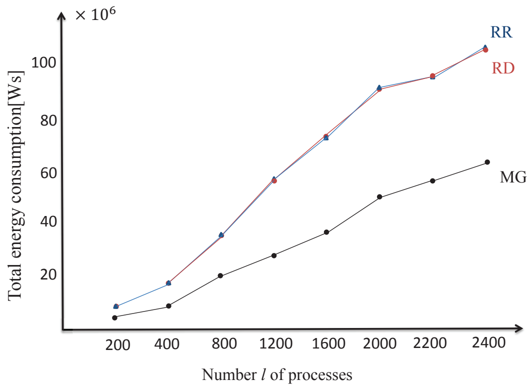

Figure 6. Total energy consumption ($n = 24$).

Figure 9 indicates how many number of servers each migrated process is migrated to in the sixteen servers ($n = 16$) for number l of processes in the MG algorithm. The average number of migrated servers is about 2.2 for each migrated process. This means, each migrated process is performed on two servers out of sixteen servers. In the evaluation, each process p_i checks the migration conditions four times, i.e. $\gamma_i = maxT_i$ / 4.

Figure 10 shows the total energy consumption of n servers in the cluster S to perform 1,600 processes ($l = 1{,}600$). In the MG algorithm, the total energy consumption decreases as the number n of servers increases. In the MG algorithm implies smaller electric energy is consumed than the RR and RD algorithms.

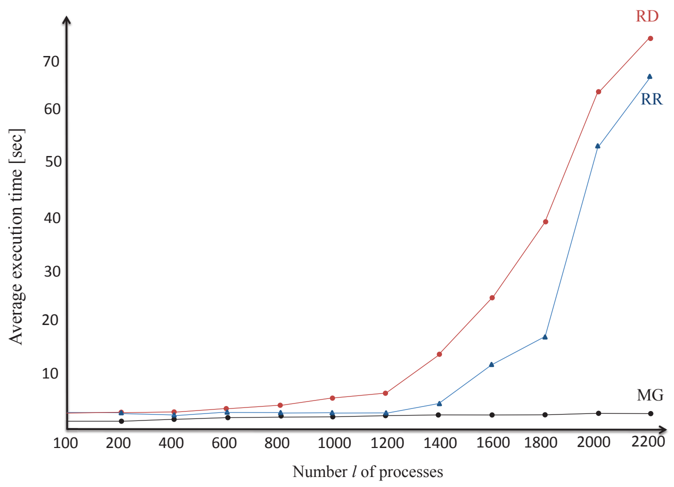

Figure 7. Average execution time of a process (n = 8).

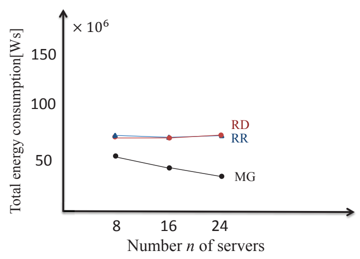

Figure 10. Total energy consumption (l = 1,600).

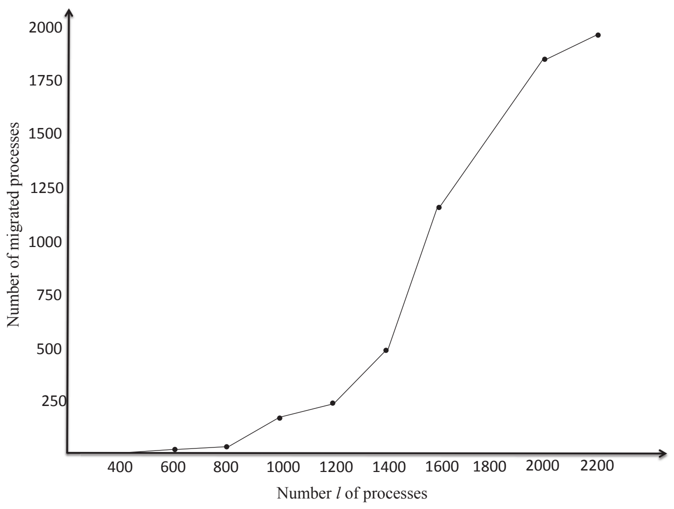

Figure 8. Number of migrated processes in the MG protocol (n = 8).

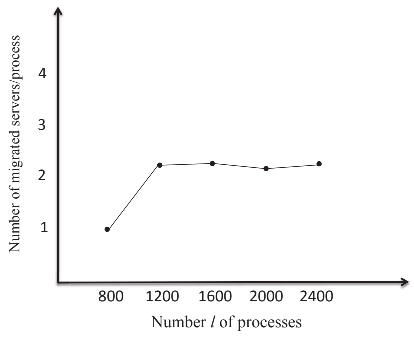

Figure 9. Number of migrated servers in the MG protocol (n = 16).

Figure 11 shows the average execution time of each process on n servers where 1,600 processes are performed (l = 1,600). The average execution time of each process is shorter in the MG algorithm than the RR and RD algorithms. For n = 8, each server is more loaded. Here, the average execution time of the MG algorithm is one fifth and one tenth of the RR and RD algorithms, respectively. In this evaluation, the migration time δ_i of each process p_i is assumed to be $minT_i$ / 2. The shorter migration time δ_i, the shorter average execution time of each process p_i.

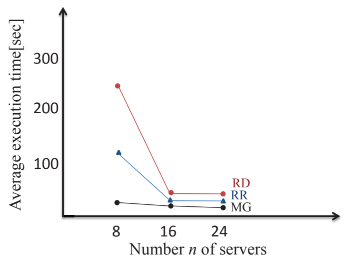

Figure 11. Average execution time of a process (l = 1,600).

Figures 12 and 13 show the total energy consumption and anerage execution time of the MG algorithm for migration time, respectively, where n = 8 and l = 800. δ_i = 10 [sec] means δ_i = $maxT_i$ / 2. The total energy consumption and average execution time are similar for every delay time.

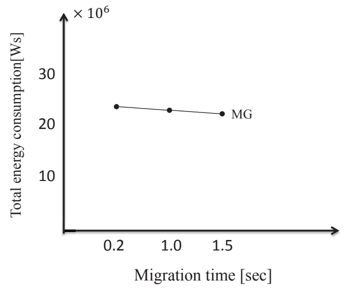

Figure 12. Total energy consumption ($n = 8$, $l = 800$).

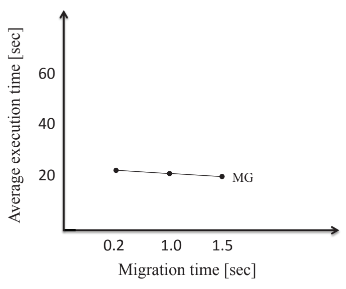

Figure 13. Average execution time ($n = 8$, $l = 800$).

V. Concluding Remarks

In this paper, we discuss the energy-efficient process migration (MG) algorithm for realizing energy-efficient executions of processes in a cluster of servers. Based on the SC and SPC models [8], [9], [10], we discussed how to obtain the expected energy consumption of a server to perform all the current processes. We also discussed how to estimate the expected termination time of each current process. We persented the migration (MG) conditions that a process is migrated from a current server to another server by estimating the energy consumption of a server and the termination time of current processes. If the process is expected to be more energy-efficiently performed on another server, the process is migrated to the server. Here, a most energy-efficient server is selected for a process. In the evaluation, we showed the total energy consumption of servers to perform processes can be smaller in the MG algorithm than the random (RD) and round-robin (RR) algorithms. The average execution time of each process can be also reduced in the MG algorithm compared with the RR and RD algorithms.

Acknowledgment

First of all, the author would like to express endless appreciation to his supervisor, Professor Makoto Takizawa, for his kindness, support, and instruction. He is the one of the person who has the most influence in authors life, the author has study many thing not only about how to do research but also how to be a good person, and how to do things correctly. He has always gave the best support and help to the author when it needed.

References

[1] A. Aikebaier, T. Enokido, and M. Takizawa : Energy-Efficient Computation Models for Distributed Systems, *Proc. of the 12th International Conference on Network-Based Information Systems* (*NBiS-2009*), page: 424-431, 2009.

[2] P. A. Bernstein and N. Goodman : The Failure and Recovery Problem for Replicated Databases, *Proc. of the 2nd ACM Symposium on Principles of Distributed Computing*, page: 114-122, 1998.

[3] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair : Distributed Systems Concepts and Design, *4th ed.*, Addison-Wesley, 2012.

[4] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa : Dynamic Clusters of Servers for Reducing Electric Power in P2P Overlay Networks, *Proc. of the 16th International Conference on Network-Based Information Systems* (NBIS-2013), page: 149-155, 2013.

[5] D. Duolikun, A. Aikebaier, T. Enokido, L. Barolli, and M. Takizawa : Energy-efficient Passive Replication of a Process in Mobile Environment, *Proc. of the 11th International Conference on Advances in Mobile Computing and Multimedia*, page: 416, 2013.

[6] D. Duolikun, H. Hama, A. Aikebaier, T. Enokido, and M. Takizawa, : Group Communication Protocols for Scalable Groups of Peers, *Proc. of the AINA-2013 Workshop (WAINA-2013)*, page: 1027-1032, 2013.

[7] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa : Power Consumption Models for Migrating Processes in a Server Cluster, *Proc. of the NBiS-2014, CD-ROM, 2014.*

[8] T. Enokido, A. Aikebaier, S. M. Deen, and M. Takizawa : Power Consumption-based Server Selection Algorithms for Communication-based Systems. *Proc. of the 13th International Conference on Network-based Information Systems* (*NBiS-2010*), page: 201-208, 2010.

[9] 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, page: 221-229, May 2010.

[10] T. Enokido, A. Aikebaier, and M. Takizawa : Process Allocation Algorithms for Saving Power Consumption in Peer-to-Peer Systems, *IEEE Transactions on Industrial Electronics* (TIE), vol.58, no. 6, page: 2097 - 2105, June 2011.

[11] T. Enokido and M. Takizawa : An Extended Power Consumption Model for Distributed Applications, *Proc. of IEEE the 26th International Conference on Advanced Information Networking and Applications* (*AINA-2012*), page: 912 - 919, 2012.

[12] T. Enokido and M. Takizawa : An Integrated Power Consumption Model for Distributed Systems, *IEEE Transactions on Industrial Electronics* (*TIE*), *vol*.60, *no*.2, page: 824-836, 2013.

[13] T. Enokido, A. Aikebaier, and M. Takizawa : An Energy-Efficient Redundant Execution Algorithm by Terminating Meaningless Redundant Processes, *Proc. of IEEE the 27th International Conference on Advanced Information Networking and Applications* (*AINA-2013*), page: 1-8, 2013.

[14] T. Enokido, A. Aikebaier, and M. Takizawa : The Evaluation of the Improved Redundant Power Consumption Laxity-Based (IRPCLB) Algorithm in Homogeneous and Heterogeneous Clusters, *Proc. of the 7th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2013*), page: 91-98, 2013.

[15] T. Enokido, K. Suzuki, A. Aikebaier, and M. Takizawa : Process Allocation Algorithm for Improving the Energy Efficiency in Distributed Systems, *Proc. of IEEE the 24th International Conference on Advanced Information Networking and Applications* (*AINA-2010*), page: 142-149, 2010.

[16] S. Ghemawat, H. Gobioff, and S. T. Leung : The Google File System, *Proc. of ACM 19th Symposium on Operating System Principle* (*SOPI 03*), page: 29 - 43, 2003.

[17] T. Inoue, M. Ikeda, T. Enokido, A. Aikebaier, and M. Takizawa : A Power Consumption Model for Storage-based Applications, *Proc. of the Fifth International Conference on Complex, Intelligent, and Software Intensive Systems* (*CISIS-2011*), page: 612 - 617, 2011.

[18] T. Inoue, A. Aikebaier, T. Enokido, and M. Takizawa : Algorithms for Selecting Energy-efficient Storage Servers in Storage and Computation Oriented Applications, *Proc. of IEEE the 26th International Conference on Advanced Information Networking and Applications* (*AINA-2012*), page: 217 - 224, 2012.

[19] K. H. Kim : Reward-based Allocation of Cluster and Grid Resources of Imprecise Computation Model-based applications, *International Journal of Web and Grid Services Systems* (*IJWGS*), *vol*.9, *no*.2 page: 140 - 171, 2013.

[20] D. Lange and M. Oshima : Programming and Deploying Java Mobile Agents with Aglets, *Addison Wesley,* 1983.

[21] Y. Tanaka, N. Hayashibara, T. Enokido, and M. Takizawa : A Mobile Agent Model for Fault-Tolerant Manipulation on Distributed Objects, *International Journal of Cluster Computing (IJCC), vol*.10, *no*.1, page: 81-93, 2007.

[22] Y. Tanaka, T. Enokido, and M. Takizawa : Design and Implementation of Transactional Agents, *International Journal of Wireless and Mobile Computing (IJWMC), vol*.4, *no*.2, page: 126-135, 2010.