An Energy-Efficient Process Replication Algorithm in Virtual Machine Environments

Tomoya Enokido and Makoto Takizawa

Abstract Server cluster systems are widely used to realize fault-tolerant, scalable, and high performance application services with virtual machine technologies. In order to provide reliable application services, multiple replicas of each application process can be redundantly performed on multiple virtual machines. On the other hand, a server cluster system consumes a large amount of electric energy since multiple replicas of each application process are performed on multiple virtual machines. It is critical to discuss how to realize not only reliable but also energy-efficient server cluster systems. In this paper, we propose the redundant energy consumption laxity based (RECLB) algorithm to select multiple virtual machines for redundantly performing each application process in presence of server faults so that the total energy consumption of a server cluster and the average computation time of each process can be reduced. We evaluate the RECLB algorithm in terms of the total energy consumption of a server cluster and the average computation time of each process compared with the basic round-robin (RR) algorithm.

1 Introduction

Various types of business and industrial information services like data centers [10] require scalable, high performance, and fault-tolerant information systems like cloud computing systems [10]. These computing systems are realized virtual machines [5] with server cluster systems [2, 3, 4]. A server cluster system is composed

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

Makoto Takizawa
Department of Advanced Sciences, Faculty of Science and Engineering,
Hosei University, Tokyo, Japan
e-mail: makoto.takizawa@computer.org

© Springer International Publishing AG 2017

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_10

of a large number of servers and multiple virtual machines are installed in each server in order to increase the resource utilization of servers. In fault-tolerant information systems, application processes which provide application services have to be reliably performed in presence of server faults [8]. One way to provide a fault-tolerant application service is that multiple replicas of each application process are performed on multiple virtual machines in a server cluster. However, a large amount of electric energy is consumed in a server cluster system since replicas of each application process are performed on multiple virtual machines which are performed on multiple servers. It is necessary to realize not only fault-tolerant but also energy efficient server cluster systems with virtual machines as discussed in Green computing [10].

In order to design energy-efficient server cluster systems [2, 3, 4], it is necessary to define a computation model and power consumption model of servers to perform application processes on virtual machines. In our previous studies [5], we measured power consumption of servers to perform *computation type application processes* (*computation processes*) which mainly consumes CPU resources of servers and derived the computation model of a virtual machine and power consumption model of a server to perform computation processes on multiple virtual machines from the experimentations. In this paper, we consider computation processes.

In this paper, we propose the *redundant energy consumption laxity based* (*RECLB*) algorithm to select multiple virtual machines for redundantly performing each application process in presence of server faults so that the total energy consumption of a server cluster and the average computation time of each process can be reduced. In this paper, we assume some servers in a server cluster might stop by fault. If a server stops by fault, every virtual machine performed on the server stops. Hence, replicas of each computation process have to be performed on multiple virtual machines which are performed on different servers in a serve cluster. Here, if at least one virtual machine is operational, a computation process is successfully performed even if some servers stop by fault. In the RECLB algorithm, a set of multiple virtual machines where the total energy consumption laxity of a server cluster is the minimum is selected for redundantly performing multiple replicas of each computation process. We evaluate the RECLB algorithm in terms of the total energy consumption of a homogeneous server cluster and computation time of each request process compared with the basic round-robin (RR) algorithm [9]. The evaluation results show the total energy consumption of a homogeneous server cluster and computation time of each request process can be more reduced in the RECLB algorithm than the RR algorithm.

In section 2, we define the computation model of a virtual machine and power consumption model of a server. In section 3, we discuss the RECLB algorithm. In section 4, we evaluate the RECLB algorithm compared with the RR algorithm.

2 System Model

2.1 Computation Model of a Virtual Machine

Let S be a cluster of servers $s_1, ..., s_n$ ($n \geq 1$). Let C_t be a set of cores $c_{1t}, ..., c_{lt}$ ($l \geq 1$) and nc_t be the total number of cores in a server s_t. We assume Hyper-Threading Technology [7] is enabled on a CPU. Let TH_t be a set of threads $th_{1t}, ..., th_{qt}$ ($q \geq 1$) in a server s_t. Let ct_t be the number of threads on each core c_{ht} in a server s_t. Threads $th_{(h-1)\cdot ct_t+1}, ..., th_{h\cdot ct_t}$ ($1 \leq h \leq l$) are bounded to a core c_{ht}. Let nt_t be the total total number of threads in a server s_t, i.e. $nt_t = nc_t \cdot ct_t$. Let V_t be a set of virtual machines $VM_{1t}, ..., VM_{qt}$ ($q \geq 1$) in a server s_t. Each virtual machine VM_{kt} holds one virtual CPU and is bounded to a thread th_{kt} in a server s_t. In this paper, we assume any virtual machine does not migrate to another server in a server cluster S. A virtual machine VM_{kt} is referred to as *active* iff (if and only if) the virtual machine VM_{kt} is initiated on a thread th_{kt} and at least one process is performed on the virtual machine VM_{kt}. A virtual machine VM_{kt} is *idle* iff the virtual machine VM_{kt} is initiated on a thread th_{kt} but no process is performed on the virtual machine VM_{kt}. A virtual machine is *stopped* iff the virtual machine is not initiated on any thread. A core c_{ht} is referred to as *active* iff at least one virtual machine VM_{kt} is active on a thread th_{kt} in the core c_{ht}. A core c_{ht} is *idle* if the core c_{ht} is not active.

We consider *computation processes*, where CPU resources are mainly consumed. A term *process* stands for a computation process in this paper. Let rd^i be the *redundancy* of a process p^i. A notation p^i_{kt} stands for a *replica* of a process p^i performed on a virtual machine VM_{kt}. On receipt of a process p^i, the load balancer K selects a set VMS^i ($|VMS^i| = rd^i$) of virtual machines in the server cluster S and forwards the process p^i to every virtual machines VM_{kt} in the set VMS^i as shown in Figure 1. We assume servers in a server cluster S might stop by fault. Let NF be the maximum number of servers which concurrently stop by fault in the cluster S. If a server s_t stops by fault, every virtual machine VM_{kt} performed on the server s_t stops. Hence, replicas of each process p^i have to be performed on rd^i virtual machines performed on different servers in a server cluster S. We assume $NF + 1 \leq rd^i \leq n$ and replicas of each process p^i are performed on rd^i virtual machines performed on different servers. This means, each client cl^i can receive at least one reply r^i_{kt} from a virtual machine VM_{kt} even if NF servers stop by fault in a server cluster S. On receipt of a process p^i, a replica p^i_{kt} is created and performed on a virtual machine VM_{kt}. Then, the virtual machine VM_{kt} sends a reply r^i_{kt} to the load balancer K. The load balancer K takes only the first reply r^i_{kt} and ignores every other reply.

Replicas which are being performed and already terminate at time τ are *current* and *previous*, respectively. Let $CP_{kt}(\tau)$ be a set of current replicas on a virtual machine VM_{kt} at time τ and $NC_{kt}(\tau)$ be $|CP_{kt}(\tau)|$. Let T^i_{kt} be the total computation time of a replica p^i_{kt} [msec]. $minT^i_{kt}$ shows the minimum computation time of a replica p^i_{kt} where the replica p^i_{kt} is exclusively performed on a virtual machine VM_{kt} and the other virtual machines are not active in a server s_t. We assume $minT^i_{1t} = minT^i_{2t} = \cdots = minT^i_{qt}$ in a server s_t, i.e. the maximum computation rate of every virtual machine

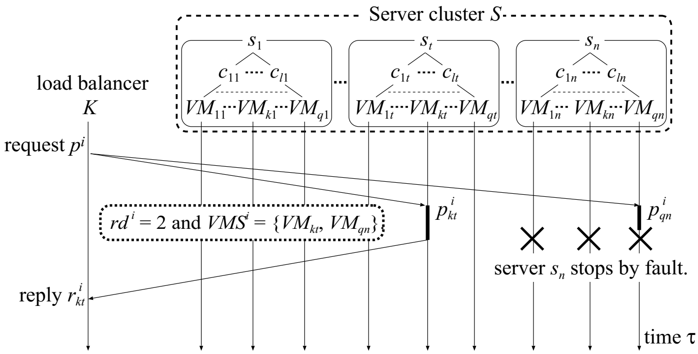

Fig. 1 System model.

VM_{kt} in the server s_t is the same. $minT^i = min(minT^i_{k1}, ..., minT^i_{kn})$. $minT^i = minT^i_{kt}$ on the fastest server s_t. We assume one virtual computation step is performed for one time unit on a virtual machine VM_{kt} in the fastest server s_t. That is, the maximum computation rate $Maxf_{kt}$ of the fastest virtual machine VM_{kt} is 1 [vs/msec]. We assume $Maxf_{1t} = Maxf_{2t} = \cdots = Maxf_{qt}$ in a server s_t. $Maxf = max(Maxf_{k1}$, ..., $Maxf_{kn})$. A replica p^i_{kt} is considered to be composed of VS^i_{kt} virtual computation steps. $VS^i_{kt} = minT^i_{kt} \cdot Maxf = minT^i_{kt}$ [vs].

The computation rate $f^i_{kt}(\tau)$ of a replica p^i_{kt} performed on a virtual machine VM_{kt} at time τ is defined as follows [5]:

$$f^i_{kt}(\tau) = \alpha_{kt}(\tau) \cdot VS^i/(minT^i_{kt} \cdot NC_{kt}(\tau)) \cdot \beta_{kt}(nv_{kt}(\tau)). \quad (1)$$

Here, $\alpha_{kt}(\tau)$ is the *computation degradation ratio* of a virtual machine VM_{kt} at time τ ($0 \leq \alpha_{kt}(\tau) \leq 1$). $\alpha_{kt}(\tau_1) \leq \alpha_{kt}(\tau_2) \leq 1$ if $NC_{kt}(\tau_1) \geq NC_{kt}(\tau_2)$. $\alpha_{kt}(\tau) = 1$ if $NC_{kt}(\tau) \leq 1$. Here, $\alpha_{kt}(\tau)$ is assumed to be $\varepsilon_{kt}^{NC_{kt}(\tau)-1}$ where $0 \leq \varepsilon_{kt} \leq 1$. The maximum computation rate $maxf^i_{kt}$ of a replica p^i_{kt} is VS^i_{kt} / $minT^i_{kt}$ ($0 \leq f^i_{kt}(\tau) \leq maxf^i_{kt} \leq 1$) where the replica p^i_{kt} is exclusively performed on a virtual machine VM_{kt} and only VM_{kt} is active on a core. Let $nv_{kt}(\tau)$ be the number of active virtual machines on a core which performs a virtual machine VM_{kt} at time τ. Let $\beta_{kt}(nv_{kt}(\tau))$ be the *performance degradation ratio* of a virtual machine VM_{kt} at time τ ($0 \leq \beta_{kt}(nv_{kt}(\tau)) \leq 1$) where multiple virtual machines are active on the same core. $\beta_{kt}(nv_{kt}(\tau)) = 1$ if $nv_{kt}(\tau) = 1$. The formula (1) means the computation rate $f^i_{kt}(\tau)$ of each replica p^i_{kt} performed on a virtual machine VM_{kt} at time τ decreases as the number of current replicas increases on the virtual machine VM_{kt}. The computation rate of a virtual machine decreases as the number of active virtual machines increases on the same core. The computation rate of a virtual machine performed on a core c_{ht} is independent of active virtual machines performed on another core c_{ft} ($h \neq f$).

Suppose that a replica p^i_{kt} starts and terminates on a virtual machine VM_{kt} at time st^i_{kt} and et^i_{kt}, respectively. Here, $T^i_{kt} = et^i_{kt}$ - st^i_{kt} and $\sum_{\tau=st^i_{kt}}^{et^i_{kt}} f^i_{kt}(\tau) = VS^i$. At time st^i_{kt} a replica p^i_{kt} starts, the computation laxity $lc^i_{kt}(\tau) = VS^i$ [vs]. The computation laxity $lc^i_{kt}(\tau)$ [vs] of a replica p^i_{kt} at time τ is given as follows:

$$lc^i_{kt}(\tau) = VS^i - \sum_{x=st^i_{kt}}^{\tau} f^i_{kt}(x). \quad (2)$$

2.2 Power Consumption Model of a Server

$E_t(\tau)$ shows the electric power [W] of a server s_t at time τ. Let $maxE_t$ and $minE_t$ be the maximum and minimum electric power [W] of a server s_t, respectively. Let $ac_t(\tau)$ be the number of active cores in a server s_t at time τ. $minC_t$ shows the electric power [W] where at least one core c_{ht} is active on a server s_t. Let $mv_t(\tau)$ be the number of virtual machines which concurrently migrate from a server s_t to the other servers at time τ. Let cE_t be the electric power [W] where one core gets active on a server s_t. Let mE_t be the electric power [W] where one virtual machine migrates from a server s_t to the other server s_u.

The electric power $E_t(\tau)$ [W] of a server s_t to perform processes on virtual machines at time τ is given as follows [5]:

$$E_t(\tau) = minE_t + \sigma_t(\tau) \cdot (minC_t + ac_t(\tau) \cdot cE_t) + \lambda_t(\tau) \cdot (mv_t(\tau) \cdot mE_t). \quad (3)$$

Here, $\sigma_t(\tau) = 1$ if at least one core c_{ht} is active on a server s_t at time τ. Otherwise, $\sigma_t(\tau) = 0$. $\lambda_t(\tau) = 1$ if at least one virtual machine migrates from a server s_t to another server at time τ. Otherwise, $\lambda_t(\tau) = 0$.

The total energy consumption $TE_t(\tau_1, \tau_2)$ [Ws] of a server s_t from time τ_1 to τ_2 is $\int_{\tau 1}^{\tau_2} E_t(\tau)\, d\tau$. The processing power $PE_t(\tau)$ [W] of a server s_t at time τ is $E_t(\tau)$ - $minE_t$. The total processing energy consumption $TPE_t(\tau_1, \tau_2)$ of a server s_t from time τ_1 to τ_2 is $\int_{\tau 1}^{\tau_2} PE_t(\tau)\, d\tau$.

3 Selection Algorithm

The total processing energy consumption laxity $tpel_t(\tau)$ [Ws] shows how much electric energy a server s_t has to consume to perform every current replica on every active virtual machine in the server s_t at time τ. Suppose a load balancer K receives a new request process p^{new} and allocates a replica p^{new}_{kt} to a virtual machine VM_{kt} performed on a server s_t at time τ. Here, the replica p^{new}_{kt} is added to the current replica set $CP_{kt}(\tau)$ of a virtual machine VM_{kt}, i.e. $CP_{kt}(\tau) = CP_{kt}(\tau) \cup \{p^{new}_{kt}\}$. Let $\mathbf{CP}_t(\tau)$ be a family $\{CP_{1t}(\tau), ..., CP_{qt}(\tau)\}$ of current replica sets of every virtual machine VM_{kt} in a server s_t at time τ. The total energy consumption laxity $tpel_t(\tau)$ of a server s_t at time τ is obtained by the **ELaxity**(s_t, τ) procedure [6]:

```
ELaxity(s_t, τ) { /* a term VM stands for a virtual machine. */
    if every CP_kt(τ) = φ in CP_t(τ) and mv_t(τ) = 0, return(0);
    for each core c_ht in a server s_t, {
        nv = the number of active VMs on c_ht at time τ;
        mv = the number of VMs migrating from c_ht at time τ;
        if nv ≥ 1, ac_t(τ) = ac_t(τ) + 1; /* count of active cores on s_t.*/
        mv_t(τ) = mv_t(τ) + mv; /* count of VMs migrating from s_t.*/
        for each VM_kt on a core c_ht, {
            for each p^i_kt ∈ CP_kt(τ), {
                lc^i_kt(τ + 1) = lc^i_kt(τ) - f^i_kt(τ);
                if lc^i_kt(τ + 1) ≤ 0, CP_kt(τ) = CP_kt(τ) - {p^i_kt};
            }
        }
    }
    tpel_t(τ) = E_t(τ) - minE_t; /* processing power consumption */
    return(tpel_t(τ) + ELaxity(s_t, τ + 1);)
}
```

We discuss the *redundant energy consumption laxity based* (*RECLB*) algorithm to select multiple virtual machines to redundantly perform each process in presence of server fault so that the total processing energy consumption of a server cluster and average computation time of each replica can be reduced. Let $TPE^S_{kt}(\tau)$ be the total processing energy consumption laxity of a server cluster S where a replica p^i_{kt} of a new request process p^i is allocated to a virtual machine VM_{kt} at time τ. Suppose a load balancer K receives a new request process p^i at time τ. Then, the load balancer K selects a set VMS^i of rd^i virtual machines in a server cluster S by the following procedure **RECLB**(p^i, τ):

```
RECLB(p^i, τ) { /* a term VM stands for a virtual machine. */
    VMS^i = φ;
    while (rd^i > 0) {
        for each VM_kt in a server cluster S, {
            CP_kt(τ) = CP_kt(τ) ∪ {p^i};
            TPE^S_kt(τ) = Σ^n_{t=1} ELaxity(s_t, τ);
        }
        vm = a virtual machine VM_kt where TPE^S_kt(τ) is the minimum;
        VMS^i = VMS^i ∪ {vm};
        S = S - {s_t}; /* the server s_t which performs the virtual machine vm is removed. */
        rd^i = rd^i - 1;
    }
    return(VMS^i);
}
```

Suppose there are three servers s_1, s_2, and s_3 in a server cluster S, i.e. $S = \{s_1, s_2, s_3\}$. Each server s_t is equipped with a single-core CPU and two threads th_{1t} and th_{2t} are bounded to the single-core. Hence, two virtual machines VM_{1t} and VM_{2t} are performed on each server s_t. Suppose a load balancer K receives a new request process p^i at time τ and the redundancy rd^i of the process p^i is two ($rd^i = 2$). Then, the load balancer K calculates the total processing energy consumption laxity $TPE^S_{kt}(\tau)$ where a replica p^i_{kt} of the process p^i is allocated to each virtual machine VM_{kt} ($k = \{1, 2\}$ and $t = \{1, 2, 3\}$) according to the **ELaxity**(s_t, τ) procedure. Suppose $TPE^S_{11}(\tau)$ is the minimum, i.e. $TPE^S_{11}(\tau) \leq TPE^S_{kt}(\tau)$ ($k = \{1, 2\}$ and $t = \{1, 2, 3\}$). Then, the load balancer K includes a virtual machine VM_{11} into the set VMS^i ($= \{VM_{11}\}$) and removes the server s_1 which performs the virtual machine VM_{11} from the server set S ($= \{s_2, s_3\}$). Next, the load balancer K calculates the total processing energy consumption laxity $TPE^S_{kt}(\tau)$ where a replica p^i_{kt} is allocated to each virtual machine VM_{kt} ($k = \{1, 2\}$ and $t = \{2, 3\}$) in the server set S ($= \{s_2, s_3\}$). Suppose $TPE^S_{23}(\tau)$ is the minimum. Then, the load balancer K includes a virtual machine VM_{23} into the set VMS^i ($= \{VM_{11}, VM_{23}\}$) and removes the server s_3 which performs the virtual machine VM_{23} from the server set S ($= \{s_2\}$). Here, $rd^i = |VMS^i| = 2$. The load balancer K forwards the process p^i to a pair of virtual machines VM_{11} and VM_{23} in the set VMS^i.

4 Evaluation

We evaluate the RECLB algorithm in terms of the total processing energy consumption of a homogeneous server cluster S and average computation time of each process p^i compared with the basic round-robin (RR) [9] algorithm.

A homogeneous cluster S is composed of five servers $s_1, ..., s_5$ ($n = 5$) as shown in Table 1. Every server s_t ($1 \leq t \leq 5$) follows the same computation model and the same power consumption model. Every server s_t is equipped with a dual-core CPU ($nc_t = 2$). Hyper-Threading Technology [7] is enabled on a CPU in every server s_t. Two threads are bounded for each core in a server s_t, i.e. $ct_t = 2$. The number of threads nt_t in each server s_t is four, i.e. $nt_t = nc_t \cdot ct_t = 2 \cdot 2 = 4$. $minE_t = 14.8$ [W], $minC_t = 6.3$ [W], $cE_t = 3.9$ [W], $mE_t = 1.25$ [W], and $maxE_t = 33.8$ [W]. The parameters of each server s_t are obtained from the experiment [5]. Each virtual machine VM_{kt} holds one virtual CPU and is bounded to a thread th_{kt} in a server s_t ($k = 1, ..., 4$ and $t = 1, ..., 5$). Hence, there are twenty virtual machines in the server cluster S. Every virtual machine VM_{kt} follows the same computation model as shown in Table 1. The maximum computation rate $Maxf_{kt}$ is 1 [vs/msec]. The parameter ε_{kt} in the computation degradation ratio $\alpha_{kt}(\tau)$ is 1. The performance degradation ratios $\beta_{kt}(1) = 1$ and $\beta_{kt}(2) = 0.5$. The parameters of each server s_t and each virtual machine VM_{kt} are obtained from the experiment [5]. We assume the fault probability fr_t for every server s_t is the same $fr = 0.1$. We assume any virtual machine does not migrate to the other servers.

Table 1 Parameters of each server s_t and each virtual machine VM_{kt}.

Server	nc_t	ct_t	nt_t	$minE_t$	$minC_t$	cE_t	mE_t	$maxE_t$
s_t	2	2	4	14.8 [W]	6.3 [W]	3.9 [W]	1.25 [W]	33.8 [W]

Virtual machine	$Maxf_{kt}$	ε_{kt}	$\beta_{kt}(1)$	$\beta_{kt}(2)$
VM_{kt}	1 [vs/msec]	1	1	0.5

($k = 1, ..., 4$, $t = 1, ..., 5$, and $minT^i = 1$ [msec]).

The number m of processes $p^1, ..., p^m$ ($0 \leq m \leq 10{,}000$) are issued in the simulation. The starting time of each process p^i is randomly selected in a unit of one millisecond between 1 and 60 [sec]. The minimum computation time $minT^i$ of every process p^i is assumed to be 1 [msec]. This means it takes one milli-second to exclusively perform a replica p^i_{kt} on a virtual machine VM_{kt} if the other virtual machines are not active on the same core where the virtual machine VM_{kt} is performed. We assume the redundancy rd^i for each process p^i is the same rd (= $\{1, 2, 3, 4, 5\}$) and $N_{fault} = rd - 1$ holds.

Let $TPEC^m_{tm}(rd)$ be the total processing energy consumption [Ws] to perform the total number m of processes ($0 \leq m \leq 10{,}000$) with redundancy rd (= $\{1, 2, 3, 4, 5\}$) obtained in the tmth simulation. The total processing energy consumption $TPEC^m_{tm}(rd)$ is measured 5 times for each redundancy rd (= $\{1, 2, 3, 4, 5\}$) and each number m of processes. The average total processing energy consumption $ATPEC^m(rd)$ [Ws] to perform the total number m of processes with redundancy rd is calculated as $\sum_{tm=1}^{5} TPEC^m_{tm}(rd) / 5$. Here, $ATPEC^m_{algo}(rd)$ stands for the average total processing energy consumption with an algorithm type $algo \in \{\text{RECLB, RR}\}$ to perform the total number m of processes with redundancy rd. Figure 2 shows the average total processing energy consumption $ATPEC^m_{algo}(rd)$ of the RECLB and RR algorithms where the fault probability fr for every server s_t is 0.1. In the RECLB and RR algorithms, the average total processing energy consumption increases as the number m of processes increases. In the RECLB algorithm, a virtual machine VM_{kt} where the total processing energy consumption laxity of a server cluster S is the minimum is selected for each replica. Hence, the average total processing energy consumption to perform the number m of processes can be more reduced in the RECLB algorithm than the RR algorithm for each redundancy rd.

The computation time T^i for each process p^i is the computation time T^i_{kt} of a replica p^i_{kt} earliest committed in the replicas of the process p^i. The computation time T^i for each process p^i is measured 5 times for each redundancy rd and each number m of processes. Let $T^{i,m}_{tm}(rd)$ be the computation time T^i [msec] of a process p^i obtained in the tm-th simulation for redundancy rd and total number m of processes. Here, $AT^m_{algo}(rd)$ stands for the average computation time [msec] of each process with an algorithm type $algo \in \{\text{RECLB, RR}\}$ to perform the total number m of processes with redundancy rd. The average computation time $AT^m_{algo}(rd)$ is calculated as $\sum_{tm=1}^{5}\sum_{i=1}^{m} T^{i,m}_{tm}(rd) / (m \cdot 5)$. Figure 3 shows the average computation time

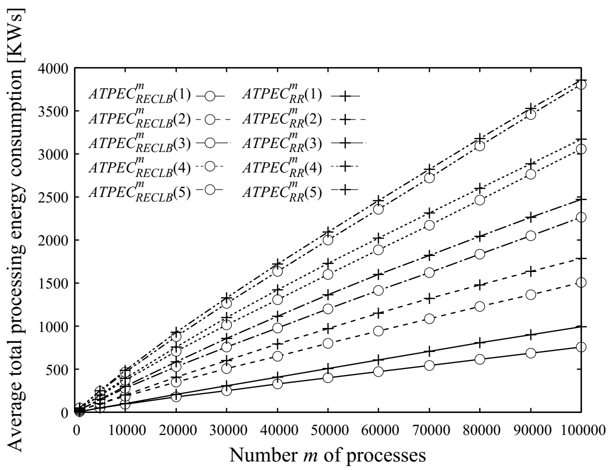

Fig. 2 Average total processing energy consumption $ATPEC^m_{algo}(rd)$ in the homogeneous server cluster S (number m of processes are issued in 60 [sec] and $fr = 0.1$).

$AT^m_{algo}(rd)$ in the RECLB and RR algorithms where the fault probability fr for every server s_t is 0.1. The average computation time $AT^m_{algo}(rd)$ increases as the number m of processes and redundancy rd increase in the RECLB and RR algorithms. For $1 \leq rd \leq 4$ and $0 \leq m \leq 10{,}000$, the average computation time in the RECLB algorithm is the same as the RR algorithm. For $rd = 5$ and $0 \leq m \leq 10{,}000$, the average computation time in the RECLB algorithm can be more reduced than the RR algorithm since the computation resources in the homogeneous server cluster S can be more efficiently utilized in the RECLB algorithm than the RR algorithm.

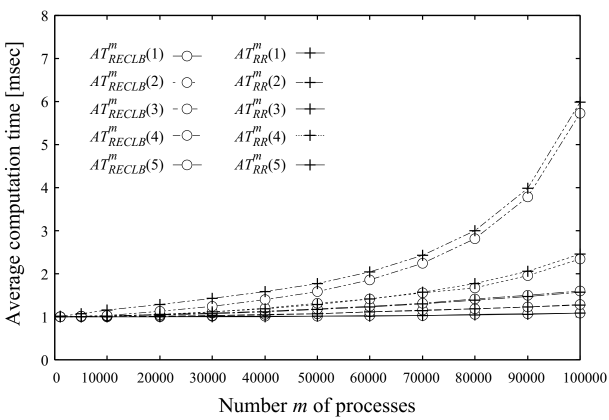

Fig. 3 Average computation time $AT^m_{algo}(rd)$ of each process in the homogeneous server cluster S (number m of processes are issued in 60 [sec] and $fr = 0.1$).

Following the evaluation, we conclude the RECLB algorithm is more useful in a homogeneous server cluster than the RR algorithm.

5 Concluding Remarks

In this paper, we proposed the RECLB algorithm to select multiple servers for redundantly performing each computation process issued by a client in presence of server fault so that the total energy consumption of a server cluster and the average computation time of each process can be reduced. In the RECLB algorithm, a set of multiple virtual machines where the total processing energy consumption laxity of a server cluster is the minimum is selected to perform multiple replicas of each process. We evaluated the RECLB algorithm in terms of the total energy consumption of a homogeneous server cluster and computation time of each process compared with the RR algorithm. The average total processing energy consumption of the homogeneous server cluster and computation time of each process are shown to be more reduced in the RECLB algorithm than the RR algorithm.

References

1. 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).
2. Enokido, T., Aikebaier, A., and Takizawa, M.: Process allocation algorithms for saving power consumption in peer-to-peer systems. IEEE Trans. on Industrial Electronics, **58**(6), pp. 2097–2105, (2011).
3. Enokido, T. and Takizawa, M.: Integrated power consumption model for distributed systems IEEE Trans. on Industrial Electronics, **60**(2), pp. 824–836, (2013).
4. 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 Trans. on Industrial Informatics, **10**(2), pp. 1627–1636, (2014).
5. Enokido, T. and Takizawa, M.: Power consumption and computation models of virtual machines to perform computation type application processes. Proc. of the 9th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2015), pp. 126–133, (2015).
6. Enokido, T. and Takizawa, M.: An energy-efficient load balancing algorithm to perform computation type application processes for virtual machine environments. Proc. of the 18th International Conference on Network-Based Information Systems (NBiS-2015), pp. 32–39, (2015).
7. Intel: Intel xeon processor 5600 series : the next generation of intelligent server processors. http://www.intel.com/content/www/us/en/processors/xeon/xeon-5600-brief.html, (2010).
8. Lamport, R., Shostak, R., and Pease, M.: The byzantine generals problems. ACM Trans. on Programing Language and Systems, **4**(3), pp.382–401, (1982).
9. LVS project: Job scheduling algorithms in linux virtual server. http://www.linuxvirtualserver.org/docs/scheduling.html, (2010).
10. Natural Resources Defense Council (NRDS): Data center efficiency assessment - scaling up energy efficiency across the data center lndustry: Evaluating key drivers and barriers -. http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf, (2014).