Migration Models for Energy-efficient Computation of Processes in Server Clusters

著者	DILAWAER Duolikun
著者別名	ディラワリ ドリクン
その他のタイトル	サーバクラスタでの低消費電力化のための移行モデルの研究
page range	1-84
year	2018-03-24
学位授与番号	32675甲第428号
学位授与年月日	2018-03-24
学位名	博士(工学)
学位授与機関	法政大学 (Hosei University)
URL	http://hdl.handle.net/10114/14080

HOSEI UNIVERSITY

Migration Models for Energy-efficient Computation of Processes in Server Clusters

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

Doctor of Engineering
(Systems Engineering and Science)

in Advanced Sciences

by

Dilawaer Duolikun

Dissertation committee:

Professor Makoto Takizawa, Committee Chair
Professor Tamai Tetsuo
Professor Takao Miura
Professor Leonard Barolli

2018

Migration Models for Energy-efficient Computation of Processes in Server Clusters

This dissertation of Dilawaer Duolikun
is approved and is acceptable
in quality and form for publication:

Makoto Takizawa 滝沢

Professor Makoto Takizawa, Committee Chair

玉井哲雄

Professor Tamai Tetsuo

三浦孝夫

Professor Takao Miura

Professor Leonard Barolli

Hosei University
2018

Contents

List of Figures

List of Tables

Acknowledgments

The author has received tremendous amount of support from so many people upon completing the degree that he cannot enumerate all of them at this moment.

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

Professor Tetsuo Tamai, Professor Takao Miura, Professor Leonard Barolli (Fukuoka Institute of Technology), Professor Isamu Shioya, Professor Tomoya Enokido (Rissho University) and Distributed Systems laboratory members of Hosei university, gave the author helpful discussions, instruction, and comments. The author would like to dedicate this work to all of them, because this work could not be done without all of them.

Most importantly, the author would like to acknowledge Dr. Alisher Akber for introducing me to Professor Makoto Takizawa.

Lastly, the author would like to express his appreciation and love to his wife and son (Kadiriya Yalkun and Feroz Dilawaer), his parents (Dolkun Mahmutjan and Gulnar Hanbaba), father-in-law and mother-in-law (Yalkun Yasin and Gulpari Abdulla), elder brother and his family (Diyar Dolkun, Maryam Omar, Nawruzbek Diyar and Nawranbek Diyar), cousin and her family (Rishat Tursun and Marzuka Zakir) and all other relatives for giving him behind-the-scenes support over many years. Words cannot express author's love to them.

Curriculum Vitae

Dilawaer Duolikun (Dilewer Dolkun)

2010	B.E. in Computers and Information Science, Xinjiang University, China.
2016	M.E. in Systems Engineering, Graduate School of Advanced Sciences, Hosei University, Japan
2018	Ph.D. in Systems Engineering of Advanced Sciences, Hosei University, Japan

Field of Study

Energy-aware dynamic migration of virtual machines, and distributed systems.

Abstract

It is critical to reduce electric energy to be consumed in every area in order to realize eco society. Huge amount of electric energy is consumed by servers in scalable clusters like cloud computing systems in information systems. We have to reduce electric energy consumed in information systems, especially server clusters to realize eco society. There are multiple approaches to reducing the electric energy consumption of servers like hardware oriented approach like developing energy-efficient CPUs. In this thesis, we newly propose a macro-level approach where we aim at reducing the total electric energy consumption of a whole server to perform application processes without considering how much electric energy each hardware device consumes. In order to discuss how to reduce the electric energy consumption of servers in a cluster, we first need a power consumption model which shows how much electric power a server consumes to perform application processes. In this thesis, a term $process$ means an application process to be performed on a server. First, we measure the electric power consumed by types of servers to perform types of application processes. Then, by abstracting parameters which mostly dominate the electric power consumption of a server, we newly propose a power consumption model named an MLPCM (Multi-Level Power Consumption with Multiple CPUs) model where the electric energy consumption of a server depends on numbers of active CPUs, cores, and threads to perform application processes.

In information systems, processes requested by applications on clients have to be performed on servers so that not only QoS (quality of service) requirements like response time and throughput are satisfied but also the total electric energy consumed by servers to perform processes has to be reduced. Based on the power consumption and computation models, algorithms to select a most energy-efficient server to perform a process issued by a client. Here, some servers might be overloaded and consume more electric energy than expected. In this thesis, we newly propose a process migration approach where processes migrate from host servers to more energy-efficient servers in a cluster. First, we propose an MG (MiGration) algorithm. Here, each process on a host server migrates to another guest server if the total electric energy to be consumed by both the servers can be reduced. However, it is not easy to migrate process among servers due to heterogeneity of the servers.

Virtual machines are now widely used to support applications with virtual computation service in clusters like cloud computing systems, which is independent of heterogeneity and distribution of servers. Furthermore, a virtual machine on a host server can migrate to a guest server while processes are being performed on the virtual machine, i.e. live migration. For example, if the host server of a virtual machine is heavily loaded and a guest server is less loaded, the virtual machine migrates to the guest server. Processes can easily migrate from servers to servers by using virtual machines even if the servers are heterogeneous. We first propose a static type of migration algorithm, an EAMV (Energy-Aware Migration of Virtual machines) algorithm where not only a virtual machine is selected to perform a process issued by a client but also a virtual machine migrates to a guest server which is expected to consume smaller electric energy to perform processes on the virtual machine. Here, a collection of virtual machines are invariant. Next, we consider a cluster where virtual machines are dynamically created and dropped depending on number of processes performed on server. We propose a DVMM (Dynamic Virtual Machine Migration) algorithm to reduce the total electric energy consumption of servers. If an application issues a process to a cluster, a most energy-efficient host server is first selected and the process is performed on a least-loaded virtual machine on the host server. Then, a virtual machine is selected on a host server and the virtual machine migrates from a host server to a guest server so that total electric energy consumption of the servers can be reduced if the host server is expected to consume more electric energy.

In the evaluation, we obtain the total electric energy and active time of servers and the average execution time of processes compared with non-migration algorithms in the simulation. We show not only the total electric energy consumption and active time of servers but also the average execution time of processes can be reduced in the DVMM algorithm compared with the other algorithms.

Keywords: Energy-efficient computation, Virtual machine, Power consumption model, Energy-aware dynamic migration of virtual machines,

Chapter 1

Introduction

We have to reduce the electric energy consumed in information systems to realize eco society as discussed in the Kyoto protocol [34], COP21 [58], and COP23 [59]. Especially, huge amount of electric energy is consumed by servers in scalable server clusters like cloud computing systems [21, 30] and Internet of Things (IoT) [45]. Hence, it is critical to reduce the electric energy consumed by servers in clusters since servers consume more electric energy than clients and other IOT devices like sensors. There are approaches to reducing the electric energy consumption of computer, especially a server. In the hardware-oriented approach, energy-efficient hardware devices like CPUs [33, 3] and storages like SSD [53] are developed and used in servers.

The electric power consumption of a server depends on not only hardware devices but also software components, especially application processes because the hardware devices are activated and consume electric energy to perform software components. In our macro-level approach to reducing the electric energy consumption of servers [21, 22], we aim at reducing the total electric energy [J] consumed by a whole server to perform application processes. We do not discuss how much electric energy each hardware device consumes. It is more significant to make clear how much amount of electric energy each server totally consumes from application software's point of view. In order to discuss how to reduce the electric energy consumption of servers in a cluster, we first have to define a power consumption model which gives how much electric power [W] a server totally consumes to perform application processes. Types of power consumption models of a server to perform computation, communication, and storage types of application processes are proposed in our previous studies [22]. In the MLPC (Multi-Level Power Consumption) model [36, 37], the electric power consumption of a server depends on the number of active threads. In this thesis, we newly propose an MLPCM (MLPC model of a server with Multiple CPUs) model [38, 39] of a server with multiple CPUs. Here, the electric power consumption of a server to perform application processes depends on the numbers of active CPUs, active cores, and active threads of the server. In addition to the power consumption model, we have to make clear how long it takes to perform each application process concurrently with other application processes on a server. In this thesis, we propose an MLCM (Multi-Level Computation model with Multiple CPUs) model [38] which gives the execution time of each application process performed on a server with multiple CPUs. By using the MLPCM and MLCM models, we can estimate the electric energy to be consumed by a server to perform application processes. In this thesis, we propose a model to simply estimate the execution time of each application process only using the total number of application processes currently performed on a server based on the power consumption model. In order to reduce the electric energy consumption of servers in a cluster, types of server selection algorithms are proposed [14, 22, 23, 27, 36, 37, 38, 39]. Here, each time a client issues a request to a cluster, a host server is selected in a cluster and an application process is created to handle the request on the host server so that the total electric energy consumption of the servers can be reduced. A $process$ means an application process issued by an application and is performed on a server in this thesis.

A process migration approach is also discussed in addition to selecting a host server for each application process [12, 14, 16]. Here, a server is selected to perform an application process issued by clients in server selection algorithms. After application processes are performed on servers, servers might be overloaded and consume more electric energy than expected. In this thesis, we newly propose a process migration approach to reducing the electric energy consumptions. Here, processes migrate to more energy-efficient guest servers while the processes are being performed. However, it is not easy to migrate types of application processes to other servers in a heterogeneous cluster where architectures and operating systems of servers are different.

A server cluster provides applications with virtual computation service by using virtual machines like KVM [44] and VMware [60]. Applications processes can be performed on a virtual machine without being conscious of what servers support what computation resources in a cluster, i.e. independent of heterogeneity and distribution of servers. Furthermore, a virtual machine on a host server can migrate to a guest server while processes are being performed on the virtual machine, i.e. live migration [44]. In our previous studies [20, 57], types of energy-efficient migration algorithms are proposed, where a virtual machine migrates from a host server to a more energy-efficient guest server. First, we discuss a *static* cluster where a set of virtual machines are invariant while each virtual machine migrates to guest servers in a cluster. In this thesis, we propose an EAMV (Energy-Aware Migration of Virtual machines) algorithm to migrate virtual machines in a static cluster. Hence, the more number of application processes are issued to a cluster, the more number of application processes are performed on each virtual machine. Here, a virtual machine might not find a guest server since too many number of application processes are performed on the virtual machine to migrate to another server.

Next, we consider a *dynamic* cluster where virtual machines are dynamically created and dropped depending on number of processes to be performed on servers. We newly propose a dynamic virtual machine migration (DVMM) algorithm for a dynamic cluster in this thesis. Here, if an application process is issued to a cluster, a host server is first selected, which is expected to consume smallest electric energy. Then, a virtual machine is newly created or selected in existing virtual machines and the application process is performed on the selected virtual machine. Each virtual machine migrates from the host server to a guest server if an application processes on the virtual machine can be more energy-efficiently performed on the guest server.

We evaluate the MG, EAMV, and DVMM algorithms proposed in this thesis in terms of the total electric energy consumption of servers and average execution time of application processes performed on the servers compared with non-migration algorithms in the simulation. We develop a simulation to measure the electric energy consumption of a server to perform application processes. We show the total electric energy consumption and total active time of servers can be mostly reduced and average execution time of application processes can be mostly shortened by the DVMM algorithm in the evaluation.

The remaining part of this thesis is organized as follows.

In chapter 2, we overview research studies related with this thesis. Energy-efficient hardware devices like CPUs are developed in industries. In this thesis, we do not consider the electric power consumption of each hardware device like a CPU. Especially, we propose a macro-level approach where the total electric energy consumed by a whole server is considered to perform application processes.

In chapter 3, we present a model of servers and virtual machines. A server is composed of CPUs and each CPU is composed of multiple cores. Each core supports one or two threads. A cluster of servers supports applications virtual services on resources like CPU, memory, and storages through virtual machines on servers. Here, applications use computation resources without being conscious of which server supports the computation resources. Furthermore, a virtual machine can migrate from a host server to another guest server while application processes are being performed, i.e. live migration,

In chapter 4, we propose the power consumption and computation models of a servers, which give electric power consumed by a server to perform application processes and execution time of each application process on a server. Here, we propose an MLPCM (Multi-Level Power Consumption with Multiple CPUs) model as a power consumption model of a server which is composed of multiple CPUs. We also propose an MLPC (Multi-Level Power Consumption) model of a server which gives the execution time of each current application process on a server. By using the MLPCM and MLPC models, we can estimate the execution time of each application process and electric energy to be consumed by each server to perform application processes.

In chapter 5, we consider a process migration approach to energy-efficiently performing an application process on servers in a cluster. If an application process is issued by a client, a server is selected to perform the application process where the expected electric energy consumption of the server is the smallest in a cluster. Furthermore, an application process migrates from a host server to another guest server if the host server is expected to consume more electric energy than expected. We show the electric energy consumption of servers in the process migration approach can be reduced compared with the non-migration approaches.

In chapter 6, we discuss the process migration approach by using virtual machines in a cluster. An application processes are able to easily migrate from host servers to guest servers even if servers are heterogeneous, e.g. diffrent architectures and operating systems. First, we consider a static cluster where the membership of virtual machines is not changed, i.e. the number of virtual machines is invariant. In this thesis, we propose an EAMV (Energy-Aware Migration of Virtual machines) algorithm to select a virtual machine to perform a request process issued by a client and to migrate a virtual machine to another guest server in order to reduce the total electric energy consumption of servers in a cluster.

In chapter 7, we consider a dynamic cluster whose membership is dynamically changed depending on number of processes performed in a cluster. In this thesis, we propose a DVMM (Dynamic Virtual Machine Migration) algorithm. In the DVMM algorithm, virtual machines are dynamically created and dropped depending on the number of application processes. Initially, there is no virtual machine on each server in a cluster. A server is first selected to perform an application process in the same way as the EAMV algorithm. Then, a smallest virtual machine where the fewest number of application processes are performed is selected on the server. If more number of application processes on the virtual machine are performed than some number, a new virtual machine is created to perform the application process. Otherwise, the application process is performed on the smallest virtual machine.

In chapter 8, we evaluate the MG, EAMV, and DVMM algorithms proposed in this thesis in terms of total electric energy consumption and total active time of servers and the average execution time of application processes. We develop a time-based simulator on a database in SQL to evaluate the algorithms, especially measure the electric energy consumption of servers to perform application processes. The total electric energy consumption and active time of servers and the average execution time of application processes in the dynamic DVMM algorithm can be mostly reduced compared with the EAMV algorithm and non-migration algorithms.

In chapter 9, we conclude this thesis. In this thesis, we newly propose the macro-level approach to reducing the total electric energy consumed by information system. Then, we newly propose the power consumption model and the computation model of a server to perform application processes. Based on the power consumption and computation models, we propose the MG, EAMV, and DVMM algorithms to migrate application processes to more energy-efficient servers by taking advantage of virtual machines supported by server clusters. Our proposed macro-level approach, power consumption and computation models, and algorithms to select servers and migrate virtual machines are theoretical and practical foundations to discuss models, architecture, algorithms, implementation, and evaluation of eco information systems.

Chapter 2

Related Studies

Information systems are now getting scalable like cloud computing systems [8] and IOT (Internet of Things) [45]. Servers in these information systems consume more electric energy [J] than clients, personal computers (PCs) and other IoT devices like sensor and actor nodes. Hence, it is critical to reduce the electric energy to be consumed by servers in clusters. There are approaches to reducing the electric energy consumption of servers in a cluster. One approach is a hardware-oriented approach where energy-efficient hardware devices like Intel CPU [33] and AMD CPU [3] and storages (SSD) [53] are developed. For example, energy-efficient CPUs like Intel Xeon [33], AMD Ryzen [3], and ARM [4] are developed by industries and are used in computers like servers. The electric energy consumption of computers like servers and personal computers (PCs) is now decreasing according to the these energy-efficient hardware devices, especially energy-efficient CPUs.

The electric power consumption [W] of a server depends on not only hardware devices but also software components, especially application processes. Hardware devices are activated by software components, i.e. processes and consume electric energy. A server thus consumes electric power to perform application processes. In our macro-level approach [21, 22] to reducing the electric energy consumption of a server, we rather aim at reducing the total electric power consumed by a whole server to perform application processes without considering the power consumption of each hardware device. In order to design, implement, and evaluate energy-efficient information systems, we first need a formal power consumption model of a server to perform application processes. A power consumption model gives electric power [W] to be consumed by a whole server to perform application processes. In our approach, we first measure electric power which each server consumes to perform types of application processes like computation, storage, communication, and general types of processes [22] by using the electric power meter UWMeters [46] as shown in Figure 2.1. Here, the electric power consumption [W] of a server can be measured every 100 millisecond. By abstracting parameters which mostly dominate the electric power consumption of a server, we define types of power consumption models.

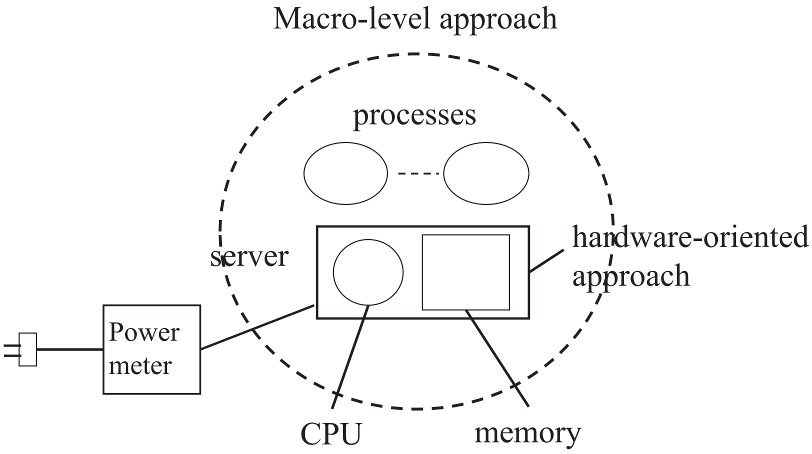

Figure 2.1: Macro-level approach.

First, we consider a computation type of an application process which consumes CPU resources like scientific computation. The SPC (Simple Power Consumption) model [21, 22, 23] is proposed as a power consumption model of a server to perform application processes of a computation type. The SPC model is the first power consumption model of a server to give the total electric power. In the SPC model, a server s_t consumes the maximum electric power $maxE_t$ [W] if at least one application process is performed. If no process is performed on a server s_t, the server s_t consumes the minimum electric power $minE_t$ [W]. Thus, the electric power consumption of a server s_t is either the maximum $maxE_t$ or minimum $minE_t$ as shown in Figure 2.2. A server with a one-thread CPU follows the SPC model.

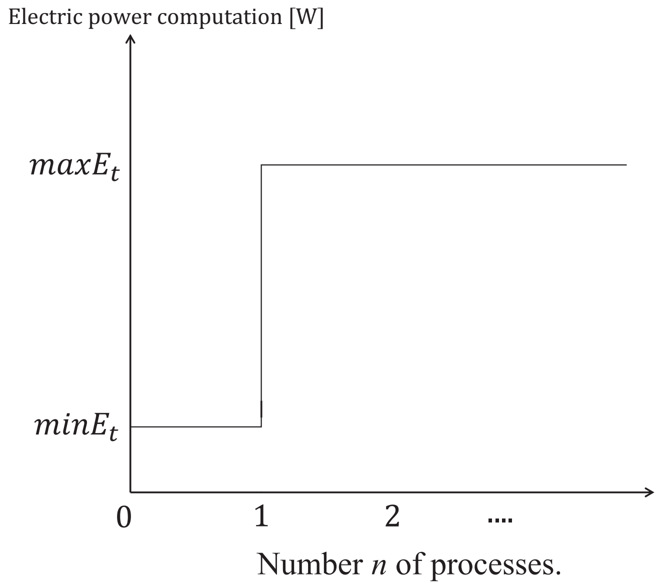

Figure 2.2: SPC model.

The MLPC (multi-level power consumption) model [36, 37] is furthermore discussed as a power consumption model of a server with a multi-thread CPU. A CPU is composed of one or more than one core. Each core supports threads, usually one or two threads. A thread is *active* where at least one application process is performed. A core is *active* if at least one thread is active. Here, the electric power consumption of a server depends on numbers of active cores and active threads of the server.

A server is currently equipped with multiple multi-thread CPUs. The MLPCM (MLPC model of a server with Multiple CPUs) model [38, 39] is also proposed for a server with multiple CPUs to perform application processes which mainly use CPU resource. In the MLPCM model, the electric power consumption of a server to perform application processes depends on the number of active CPUs, active cores, and active threads.

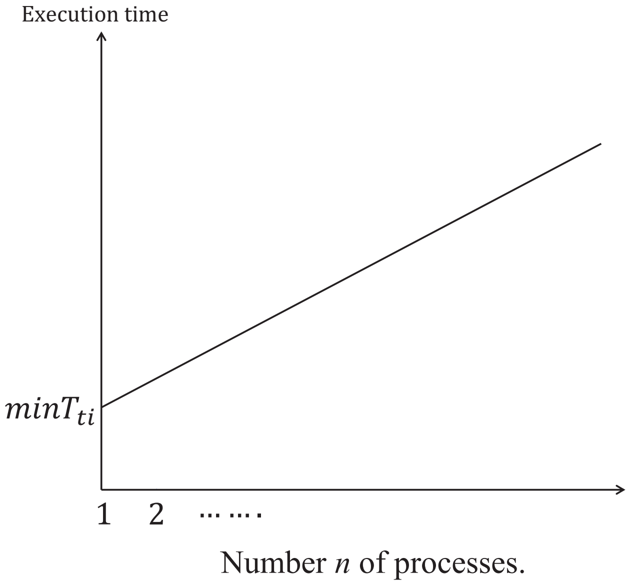

Figure 2.3: SC model.

The power consumption models are also proposed for communication [21, 22] and storage [32, 49, 52] types of application processes. In papers [21, 22], the power consumption model of a download server is proposed. A algorithms to select a server in a cluster are also proposed. Here, it is shown the electric power consumed by a server to transfer data depends on the total transmission rate at which the server transmits data to clients. The power consumption of the communication device to transfer data in files is not so large compared with CPUs.

In a storage type of application process, files in storage drives are manipulated like database and web applications. The power consumption model of a storage server to perform storage application processes is proposed [50, 51]. Here, the electric power consumption of a server is some constant larger than the power consumption to perform computation application processes.

In addition to the power consumption models, we need a computation model of a server which gives the execution time of each current application process. In the SC (Simple Computation) model [21, 22, 23], the execution time of each application process is proportional to the number of application processes concurrently performed with the application process as shown in Figure 2.3. $minT_{ti}$

shows the minimum execution time of an application process p_i on a fastest server s_t, i.e. only the process p_i is performed. The minimum execution time $minT_i$ of a process p_i in a cluster is a minimum of $minT_{1i}, \ldots, minT_{mi}$ of servers $s_1, \ldots, s_m$, respectively. The minimum execution time $minT_i$ shows the total amount VC_i of computation of an application process p_i. The total amount VC_i of computation of the application process p_i is decremented by the computation rate for each time unit. Thus, the execution time of each application process on a server can be estimated by using the computation model. A server with a single thread follows the SC model.

An application on a client issues a request process to a cluster of servers. On receipt of the request from an application on the client, a load balancer selects a server and forwards the request to the server. Then, an application process is created and then performed on the server. Here, a server which is expected to consume smaller electric energy has to be selected to perform the application process.

In order to reduce the electric energy consumption of servers in a cluster, types of algorithms [14, 22, 23, 27, 36, 37, 38, 39] are proposed to select a server in a cluster to perform an application process. Here, a server to perform an application process is selected based on the power consumption and computation models so that the total electric energy consumption of the servers can be reduced in a cluster. However, it takes time to simulate the execution of each application process to estimate the electric energy consumption of the server by using the computation laxity of each application process and the computation rate of the server.

A process migration approach is also discussed to reduce the electric energy consumption of a cluster of servers. Here, an application process on a host server migrates to another server if the host server is expected to consume more electric energy than expected, e.g. because the server is overloaded [11, 12, 13, 14, 16]. In order to increase the reliability and availability of a system, an application process is replicated to replicas on multiple servers. The more number of replicas are performed, the more reliable and available the system is. However, the more electric energy is consumed by the more number of servers. The energy-efficient replication and migration ways of an application process are also discussed not only to increase the reliability and availability of the system but also to reduce the electric energy consumption of the servers [14, 26]. However, it is not easy to migrate types of application processes to servers with various types of architectures and operating systems.

Virtual machines are widely used to support applications with virtual computation service in a cluster of servers, e.g. KVM [44], VMware [61]. Here, applications use computation resources like CPUs and storages like HDDs by using virtual machines independently of what servers support what computation resources. Furthermore, virtual machines where application processes are performed can easily migrate to guest servers independently of architectures and operating systems of servers. In this paper, we discuss how to migrate application processes to servers by using the migration of virtual machines to reduce the electric energy consumption of servers.

In clusters like data centers, servers which are not required to perform application processes, for example, lightly loaded, are shut down to reduce the electric energy consumed. Servers are restarted if more number of servers are required to perform application processes depending on the traffic as discussed in paper [6]. This is the shut-down approach. It is efficient and useful to take this shut-down approach in the client-server model like cloud computing systems where the servers are controlled in a centralized manner. In this paper, we rather consider a distributed system where each server is autonomous like peer-to-peer (P2P) model [5, 22, 47]. Here, it is not easy to shut down and restart servers since we have to do the negotiation with owners or administrators of each server. In our approach, we discuss how to select an energy-efficient server in a cluster to energy-efficiently perform an application process issued by a client and do not discuss how to shut down and restart servers.

In wireless sensor networks (WSNs) [2, 5], it is critical to reduce the electric energy consumption of sensor nodes since the sensor nodes work by using the electric energy supplied by buttery. Energy-efficient ad hoc routing protocols are proposed and evaluated [54, 43].

Chapter 3

System Model

3.1 Servers

Current information systems like cloud computing systems [8] are based on the server-client model. A cluster S is composed of servers s_1, ..., s_m $(m \geq 1)$ and clients which are interconnected in reliable high speed networks. We assume an underlying network supports a pair of servers with non-loss, non-duplication, and sending-order delivery of messages, i.e. delivery of messages in sending order like TCP [28]. We also assume every server is reliable, i.e. does not suffer from fault in this thesis. Every server is always properly operational.

An application on a client first issues a request to a cluster S. One server s_t is selected in the cluster S. For example, a server is selected by a load balancer in the round-robin algorithm. An application process to handle the request is created on the selected server s_t. Then, the application process is performed on the server s_t. On termination of the process, the server s_t sends a reply to the client.

Each server s_t is equipped with a set CP_t of np_t (≥ 1) homogeneous CPUs, cp_{t0}, ..., cp_{t,np_t-1} as shown in Figure 3.1. Each CPU cp_{tk} is composed of cc_t (≥ 1) homogeneous cores c_{tk0}, ..., c_{tk,cc_t-1}. Each core c_{tki} supports a set $\{th_{tki0}$, ..., $th_{tki,ct_t-1}\}$ of ct_t (≥ 1) homogeneous threads. Usually, ct_t is two, i.e. a core supports two threads. A server s_t thus supports processes with the total number nt_t $(= np_t \cdot cc_t \cdot ct_t)$ of threads on nc_t $(= np_t \cdot cc_t)$ cores. Each process is at a time allocated to one thread i.e. performed on a thread [48]. Multiple processes can be concurrently performed on each thread. An $active$ thread is a thread where at least one process is performed. If no process is performed on a thread, the thread is $idle$. An active core is a core where at least one thread is active. In an idle core, no thread is active. An active server is a server where at least one thread is active, i.e. at least one process is performed. An *idle* server is a server where no thread is active, i.e. no process is performed.

There are types of application processes [23], as presented in the preceding chapter:

1. Computation processes.
2. Communication processes.
3. Storage processes.
4. General processes.

A computation type of application process is an application process which uses CPU resource. A computation process does the computation like scientific computation. In the communication type of application process, communication resources are used. For example, an application process transmits a file to a client like FTP (File Transfer Protocol) application. In the storage type of application processes, data in storage devices like HDD and SDD are manipulated. In general processes, both CPU and storages are manipulated like web and database applications.

In this thesis, we consider computation processes as application processes. A term *process* means a computation type of an application process to be performed on a server, which uses CPU resource.

3.2 Virtual Machines

A cluster S supports applications with virtual computation service through virtual machines as supported by cloud computing systems [44]. Applications can use computation resources on servers without being conscious of what servers support the resources and independently of the heterogeneity and distribution of servers like operating systems and architectures. This means, applications can easily use resources even on a cluster of heterogeneous servers.

Suppose a set VM of virtual machines $vm_1, \ldots, vm_v$ ($v \geq 1$) are supported to applications in a cluster S. Each virtual machine vm_h is supported with threads of a server s_t. Here, the server s_t is referred to as a *host* server of the virtual

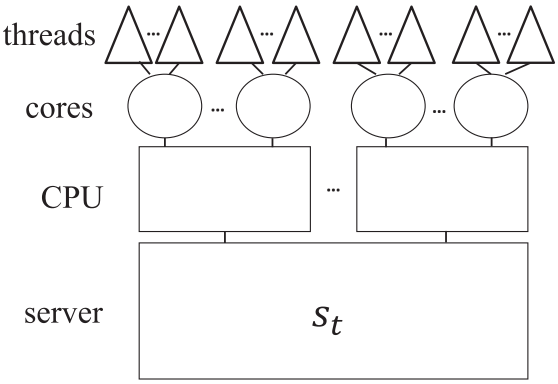

Figure 3.1: Server.

machine vm_h and the virtual machine vm_h is a $resident$ virtual machine of the server s_t. $VM_t(\tau)$ shows a set of resident virtual machines on a host server s_t and $HS_h(\tau)$ denotes a host server of a virtual machine vm_h at time τ. A process p_i performed on a virtual machine vm_h is a $resident$ process of the virtual machine vm_h. $VCP_h(\tau)$ shows a set of resident processes of a virtual machine vm_h at time τ. A virtual machine vm_h is *active* at time τ if $|VCP_t(\tau)| > 0$, i.e. at least one process is performed on the virtual machine vm_h, otherwise *idle*. Time τ when an active virtual machine vm_h gets idle is referred to as $idled$ time of the virtual machine vm_h. That is, some process is performed by the time τ and no process is performed after the time τ. $CP_t(\tau)$ is a set of all the resident processes performed on virtual machines of a server s_t at time τ, i.e. $CP_t(\tau) = \cup_{vm_h \in SVM_t(\tau)} VCP_h(\tau)$. In this thesis, we assume every application process is performed on a virtual machine, not directly performed on a host server. A server s_t where at least one virtual machine resides, i.e. $|VM_t(\tau)| > 0$, is an $engaged$ server. An engaged server s_t is $active$ if at least one resident virtual machine of the server s_t is active. A server s_t is $free$ if no virtual machine resides on the server s_t. A virtual machine vm_h is $smaller$ than a virtual machine vm_k (vm_k is larger than the vm_h) ($vm_h < vm_k$) at time τ if $|VCP_h(\tau)| < |VCP_k(\tau)|$. That is, more number of processes are performed on a larger virtual machine vm_k than a smaller virtual machine vm_h ($vm_k > vm_h$). A pair of virtual machines vm_h and vm_k are equivalent ($vm_h \equiv vm_k$) if $|VCP_h(\tau)| = |VCP_k(\tau)|$. $vm_k \geq vm_h$ if $vm_k > vm_h$ or $vm_k \equiv vm_h$. An idle virtual machine vm_h is the smallest sine $|VCP_k(\tau)| = 0$.

A virtual machine vm_h on a host server s_t can migrate to a guest server s_u while resident processes are performed in the live migration [Figure 3.2]. For example, a virtual machine vm_h on a host server s_t can migrate to a guest server s_u by issuing a following migration command **virsh** on the host server s_t in KVM (Kernel-based Virtual Machine) [44].
“ **virsh** migrate $-$live nVM qemu+ssh://$destinationURL$/system”
A virtual machine vm_h on a host server s_t migrates to a guest server s_u by issuing a migration command on the host server s_t. Another type of migration is offline migration. Here, every process is terminated on a virtual machine. Then, the virtual machine migrates to a guest server. In this thesis, we consider the live migration of virtual machines.

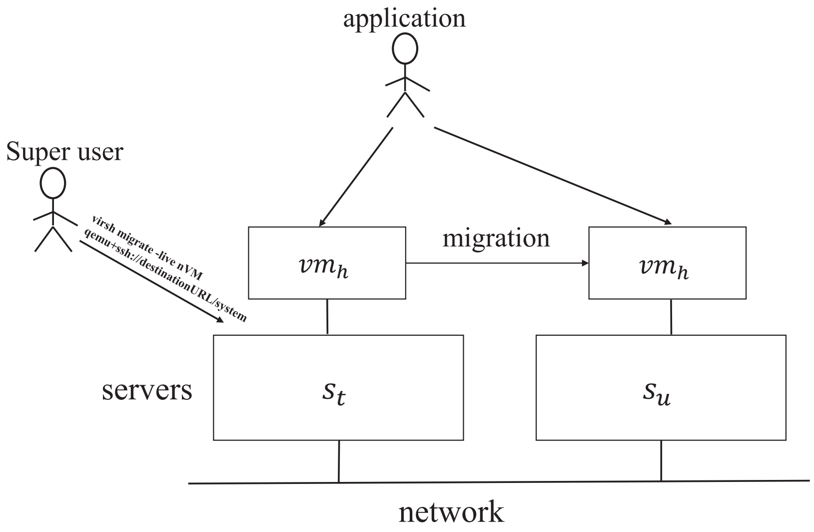

Figure 3.2: Virtual machine migration.

First, a copy of memory of the virtual machine vm_h is created on a guest server s_u. Then, the virtual machine vm_h migrates to the server s_u. On issuing the following migration command the virtual machine vm_h from the host server s_t to the guest server s_u.

First, the memory state of the virtual machine vm_h on the host server s_t is transfered to the guest server s_u while processes on the virtual machine vm_h are being performed. On termination of the memory state transfer, the processes are suspended at time τ_t and the state of the virtual machine vm_h changed after the memory copy, i.e. dirty pages, is transfered to the server s_u. Then, the virtual machine vm_h is resume, i.e. the processes are restarted on the server s_u at time st_u. Thus, the processes are not performed for time $et_t - st_u$ which is down time. The time $st_u - st_t$ is the migration time. The migration command is only allowed to be issued by the superuser of a host server.

A system is composed of servers $s_1, \ldots, s_m$ and clients which are interconnected in reliable networks. First, an application on a client issues a request to a cluster S of servers $s_1, \ldots, s_m$ ($m \geq 1$) as shown in Figure 3.3. On receipt of a request from a client, a load balancer L selects a host server s_t in the cluster S. Then, one virtual machine vm_h is selected on the host server s_t. A process is created to handle the request on the virtual machine vm_h and is performed on the virtual machine vm_h. Even if the virtual machine vm_h migrates from the host server s_t to another guest server s_u, the application can take usage of the process on the virtual machine vm_h without being conscious of which server the process is performed on. Then, a reply is sent back to the application of the client on termination of the process.

3.3 Performance of Virtual Machines

We consider the overhead of virtual machines on servers in terms of the average execution time of processes and time to migrate a virtual machine with processes. We consider a server s_t and a virtual machine vm_h resident on the server s_t. The server s_t is equipped with a CPU (Intel Corei7-6700K) where CentOS7 [48] is installed as an operating system. The virtual machine vm_h is equipped with 2GB memory storage by KVM [44] and supports CentOS7.

We first measure the average execution time of n (≥ 0) processes which are performed on the virtual machine vm_h and are directly performed without any

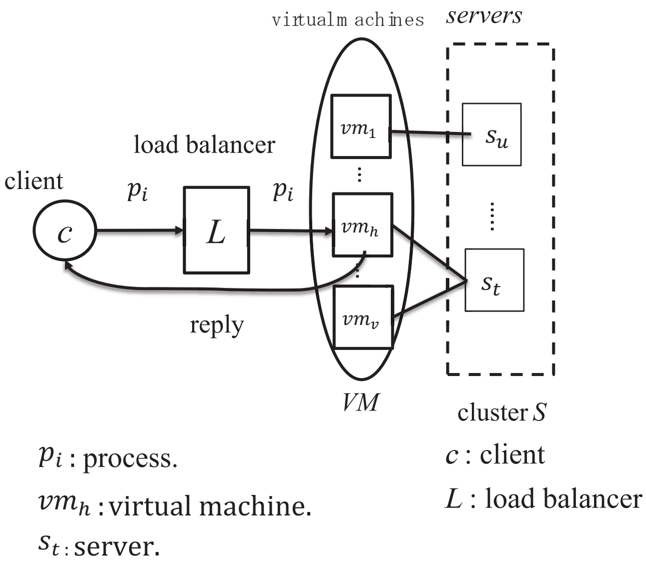

Figure 3.3: Cluster of servers.

virtual machine on the server s_t. First, n processes p_1, …, p_n are created by forking a process p. It takes 2.1 [sec] to perform the process p on the server s_t without any other process. It is minimum execution time $minT_i$ of the processes p_i. Each process p_i waits until specified time τ after the process p_i is created by the fork mechanism. Then, all the processes $p_1, \ldots, p_n$ start at time τ. Figure 3.4 shows the average execution time of the n processes on the virtual machine vm_h and on the server s_t. The dotted line and straight line show the average execution time of the processes which are performed on the virtual machine vm_h of the server s_t and are performed directly on the server s_t, respectively. As shown in Figure 3.4, the average execution time of the n processes on the virtual machine vm_h is about 10% longer than the processes are directly performed on server s_t. This means, the average execution time of the processes depends on the number n of processes performed on the server s_t even if the processes are performed on virtual machines.

Next, we measure the migration time of the virtual machine vm_h. In addition to the server s_t, we use another server s_u with a CPU (Intel core i5 E97378) in our laboratory. The virtual machine vm_h with n processes migrates from the server s_t to the server s_u in the live migration of KVM. That is, n processes are being performed on the virtual machine vm_h while the virtual machine vm_h migrates from the server s_t to the server s_u. On the server s_t, the migration command is issued at time τ_1 and ends at time τ_2. The migration time of the virtual machine vm_t is defined to be $\tau_2 - \tau_1$. The virtual machine vm_h is composed of memory 1 [GB]. A pair of the servers s_t and s_u are connected in a 1Gbps local area network. Figure 3.5 shows the migration time of the virtual machine vm_h with n processes. As shown in Figure 3.5, the migration time of the virtual machine vm_h on the host server s_t to migrate to the guest server s_u is about 11[sec]. The migration time is independent of number n of processes which are performed on the virtual machine vm_h.

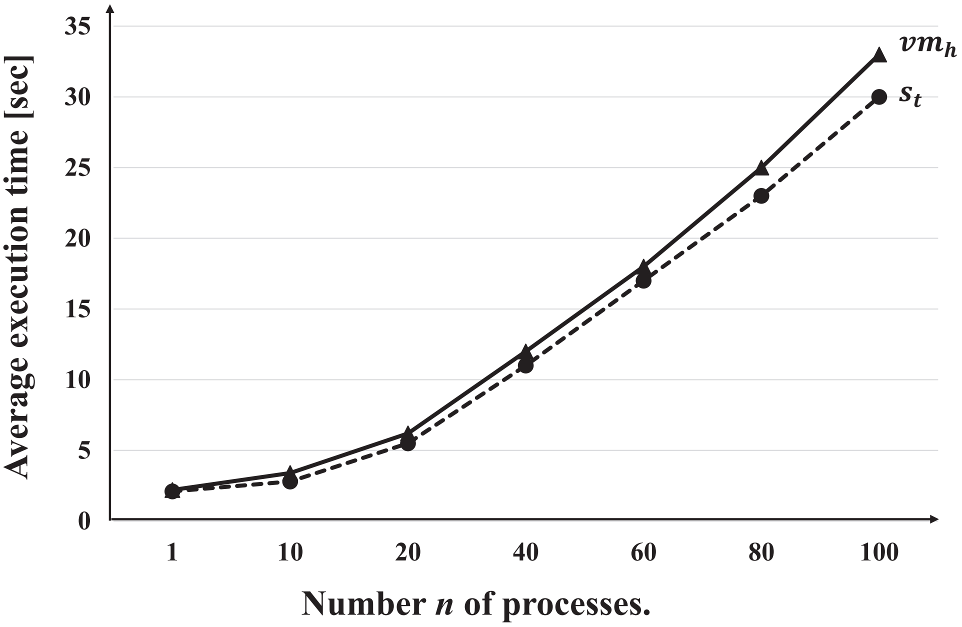

Figure 3.4: Average execution time.

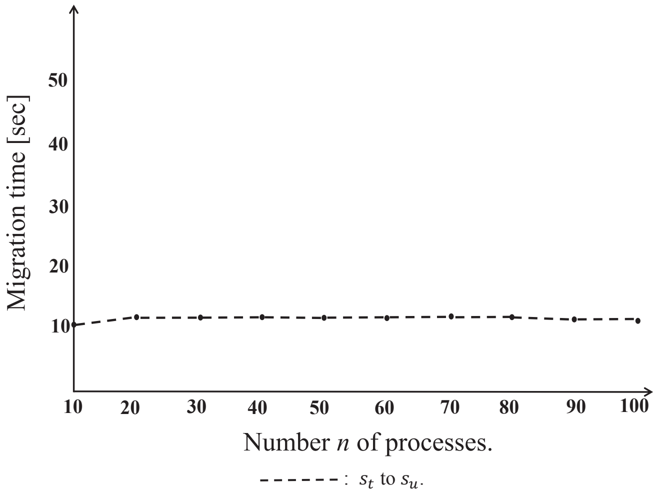

Figure 3.5: Migration time.

Chapter 4

Power Consumption Models and Computation Models

4.1 Power Consumption Model

First, we would like to propose a power consumption model of a server which gives the electric power to be consumed by a server to perform application processes by the macro-level approach. The power consumption model of a server gives how much electric power [W] the server consumes to perform application processes. The power consumption model plays an essential role to design, implement, and evaluate models and algorithms to reduce the electric energy consumption of information systems.

An application on a client c_s issues a request to the cluster S of servers $s_1, \ldots,$ s_m $(m \geq 1)$ as shown in Figure 4.1. A load balancer L selects a host server s_t in the cluster S and sends a request to the server s_t. An application process p_i is created on the server s_t to handle the request and the process p_i is performed on the server s_t. On termination of the process p_i, the server s_t sends a reply to the client c_s. In this thesis, we consider a computation type of application process which uses CPU resource. A term $process$ means a computation type of application process in this thesis.

A server s_t is composed of np_t (≥ 1) homogeneous CPUs $cp_{t0}, \ldots, cp_{t,np_t-1}$. Each CPU cp_{tk} is composed of cc_t (≥ 1) homogeneous cores $c_{tk0}, \ldots, c_{tk,cc_t-1}$. Each core c_{tkh} supports the same number ct_t of homogeneous threads, usually ct_t is one or two. A server s_t supports totally nc_t $(= np_t \cdot cc_t)$ homogeneous cores and nt_t (= $nc_t \cdot ct_t$) homogeneous threads $tr_{tk0}, \ldots, tr_{tk,nt_t-1}$. An $active$ thread is a thread where at least one process is performed. A thread where no process is performed is $idle$. Each process is at a time performed on one thread.

Let $CP_t(\tau)$ be a set of processes concurrently performed on a server s_t at time τ. An active server is a server where at least one thread is active. In an active server, at least one process is performed, i.e. $|CP_t(\tau)| > 0$. Suppose a process p_i is performed on a host server s_t of the process p_i. Here, the process p_i is resident process on the host server s_t.

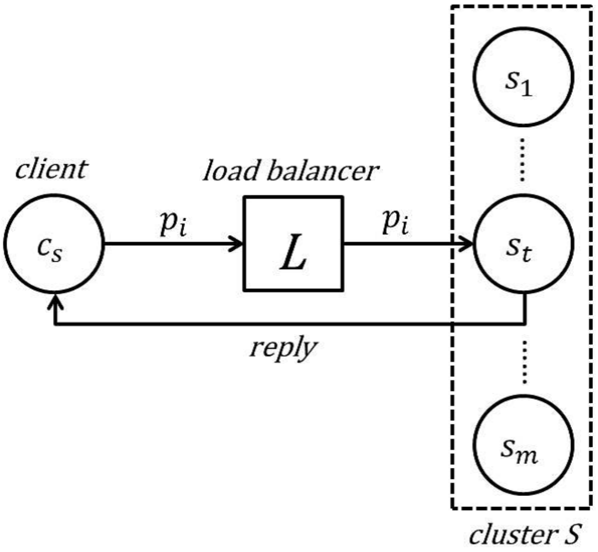

Figure 4.1: Cluster of servers.

The electric power consumption $E_t(\tau)$ [W] of a server s_t to concurrently perform processes in the set $CP_t(\tau)$ at time τ is given as follows [37]:

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

$$E_t(\tau) = minE_t + \sum_{k=0}^{np_t-1} \{\gamma_{tk}(\tau) \cdot [bE_t + \sum_{i=0}^{cc_t-1} \alpha_{tki}(\tau) \cdot (cE_t + \sum_{h=0}^{ct_t-1} \beta_{tkih}(\tau) \ \cdot tE_t)]. \tag{4.1}$$

Here, $\gamma_{tk}(\tau)$ = 1 if a CPU cp_{tk} is active at time τ ($k < np_t$). Otherwise, $\gamma_{tk}(\tau)$ = 0. $\alpha_{tki}(\tau)$ = 1 if a core c_{tki} is active on a CPU cp_{tk} at time τ ($i < cc_t$). Otherwise,

$\alpha_{tki}(\tau)$ = 0. $\beta_{tkih}(\tau) = 1$ if the hth thread on a core c_{tki} is active ($h < ct_t$). Otherwise, $\beta_{tkih}(\tau) = 0$.

Let $ap_t(\tau)$, $ac_t(\tau)$, and $at_t(\tau)$ be numbers of active CPUs, active cores, and active threads in a server s_t at time τ, respectively. Here, $ap_t(\tau) \leq np_t$, $ac_t(\tau)$ $\leq nc_t$ (= $np_t \cdot cc_t$), and $at_t(\tau) \leq nt_t$ (= $np_t \cdot cc_t \cdot ct_t$). The electric power consumption $E_t(\tau)$ (formula (1)) is also given as follows:

$$E_t(\tau) = minE_t + ap_t(\tau) \cdot bE_t + ac_t(\tau) \cdot cE_t + at_t(\tau) \cdot tE_t. \quad (4.2)$$

The maximum electric power $maxE_t$ to be consumed by a server s_t is $minE_t$ + $np_t(\tau) \cdot bE_t$ + $nc_t(\tau) \cdot cE_t$ + $nt_t(\tau) \cdot tE_t$ [W] where every thread is active. That is, at least one process is performed on every thread. Even if more number of processes than the total number nt_t of threads are performed on a server s_t at time τ, the server s_t consumes the maximum electric power $E_t(\tau) = maxE_t$. If no process is performed on a server s_t at time τ, the server s_t consumes the minimum electric power $E_t(\tau) = minE_t$ [W].

In Linux operating systems [48], processes are allocated to threads on a server s_t in the round-robin (RR) algorithm. Here, a first process is performed on a first thread of the core c_{t00} of the first CPU cp_{t0}. A next process is performed on a first thread of the second core c_{t10} of the second CPU cp_{t1}. If one thread of each core is active, a process is performed on the second thread of the first core. Thus, processes are allocated to threads, cores, and CPUs in a server s_t so that processes are uniformly distributed to threads in the server s_t. Here, the electric power $CE_t(n)$ [W] consumed by a server s_t to concurrently perform n (= $|CP_t(\tau)|$) processes at time τ is assumed to be given as follows [39, 40]:

[Power consumption for n processes] [Figure 4.2]

$$CE_t(n) = \begin{cases} minE_t \text{ if } n = 0. \\ minE_t + n \cdot (bE_t + cE_t + tE_t) \text{ if } 1 \leq n \leq np_t. \\ minE_t + np_t \cdot bE_t + n \cdot (cE_t + tE_t) \text{ if } np_t < n \leq nc_t. \\ minE_t + np_t \cdot bE_t + nc_t \cdot cE_t + nt_t \cdot tE_t \text{ if } nc_t < n \leq nt_t. \\ maxE_t \text{ if } n > nt_t. \end{cases} \quad (4.2)$$

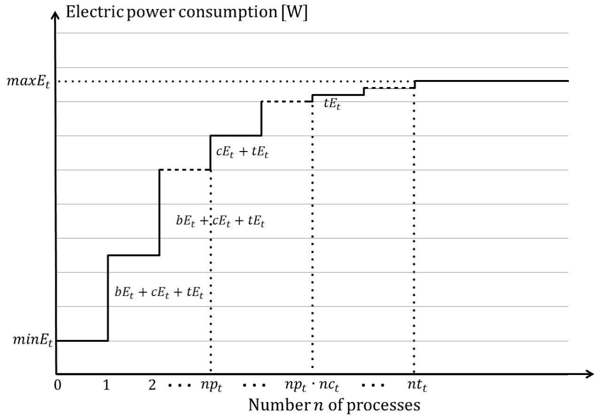

Figure 4.2: MLPCM model.

The electric power consumption $E_t(\tau)$ [W] of a server s_t at time τ is assumed to be $CE_t(n)$ in this paper, where n is the number $\mid CP_t(\tau) \mid$ of processes concurrently performed. If more number n of processes than the total number nt_t of threads are performed on a server s_t, $n \geq nt_t$, the server s_t just consumes the maximum electric power $maxE_t$ independently of the number n of processes. Thus, the server s_t follows the SPC model as long as $n \geq nt_t$ or $n = 0$. For $0 < n < nt_t$, the electric power consumption of a server s_t depends on the number n of processes.

We measure the electric power consumption of the DSLab server s_t [9] with CentOS7 [48] to concurrently perform n processes in our laboratory every 100 [msec] by using the electric power meter UWmeter [46]. A process p is forked to n child processes $p_1, \ldots, p_n$ [48]. Each child process p_i waits by a system call syscalls until specified time τ and all the n processes $p_1, \ldots, p_n$ simultaneously start at the time τ. Figure 4.3 shows the electric power consumption [W] measured where n (≥ 0) processes are concurrently performed. The server s_t is equipped with two Intel Xeon E5-2667 v2 CPUs ($np_t = 2$). Each CPU is composed of eight cores ($cc_t = 8$) and each core supports two threads ($ct_t = 2$). Totally, $nt_t = np_t \cdot cc_t \cdot ct_t = 2 \cdot 8 \cdot 2 = 32$ threads are supported for processes on $nc_t = np_t \cdot cc_t = 2 \cdot 8 = 16$ cores. The electric power consumption $minE_t = 126.1$, $bE_t = 30$, $cE_t = 5.6$, $tE_t = 0.6$, and $maxE_t = 301.1$ [W] as shown in Figure 4.3. If one process is performed ($n = 1$), the server s_t consumes $minE_t + bE_t + cE_t + tE_t = 162.3$ [W]. For $n = 2$, both the CPUs are active. Hence, the server s_t consumes the electric power $minE_t + 2 \cdot (bE_t + cE_t + tE_t) = 198.5$ [W]. For $n = 3$, two cores of one CPU and one core of another CPU are active. Here, $minE_t + (bE_t + 2 \cdot (cE_t + tE_t)) + (bE_t + cE_t + tE_t) = minE_t + 2 \cdot bE_t + 3 \cdot (cE_t + tE_t) =$ 204.7 [W]. For $n \geq$ 16, every core on every CPU is active. For 16 $\leq n \leq$ 32, every core is active and n threads are active. The server s_t consumes the electric power $2 \cdot (bE_t + 8 \cdot cE_t) + n \cdot tE_t$. For $n \geq$ 32, every thread is active and the server s_t consumes the maximum electric energy $maxE_t = minE_t + 2 \cdot bE_t + 8 \cdot cE_t + 32 \cdot tE_t =$ 425.1 [W]. Thus, the MLPCM model given by formula (2) holds for a real server as shown in Figure 4.3.

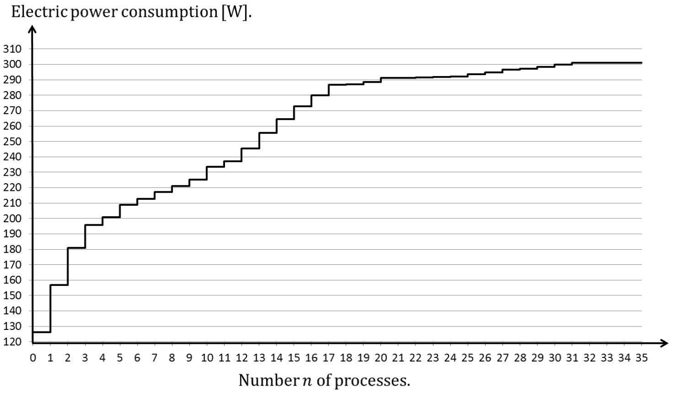

Figure 4.3: Electric power consumption of DSLab server.

4.2 Computation Model

Next, we propose a computation model of a server which gives the execution time of each process. Processes issued by clients are performed on servers in a cluster S. Each process is at a time performed on a thread of a server. It takes T_{ti} time units [tu] to perform a process p_i on a thread of a server s_t. If only a process p_i is performed on a thread of a server s_t without any other process, the execution time T_{ti} of the process p_i is shortest, i.e. T_{ti} = $minT_{ti}$. The more number of processes are performed with a process p_i on a thread, the longer time it takes to perform the process p_i. Let $minT_i$ be a minimum one of minimum execution time $minT_{1i}$, $\ldots$, $minT_{mi}$ of a process p_i on servers $s_1, \ldots, s_m$. That is, $minT_i$ is the minimum

execution time $minT_{fi}$ on the fastest thread which is on a server s_f. Here, a server s_f with the fastest thread is referred to as $fastest$ in a cluster S.
We make the following assumption:

1. One virtual computation step [vs] is performed on the thread of the fastest server s_f for one time unit [tu] in this paper.

2. The thread computation rate CRT_f of a fastest server s_f is one [vs/tu] in a cluster S, $CRT_f = 1$.

3. The thread computation rate CRT_t of a server s_t is defined to be $(minT_i$ / $minT_{ti}) \cdot CRT_f$ = $minT_i$ / $minT_{ti}$ [vs/tu] $(\leq CRT_f)$.

It is not easy to measure the total amount of computation of each process. Hence, we introduce a $virtual\ computation\ step$ (VS). On a fastest server s_f, one $(= CRT_f = minT_i$ / $minT_{ti})$ virtual computation step is performed for one time unit [tu] on a thread. On another server s_t, $minT_i$ / $minT_{ti}$ $(= CRT_t)$ (≤ 1) virtual computation steps are performed on a thread for one time unit.

The maximum server computation rate $maxCR_t$ $(\leq nt_t)$ of a server s_t is $nt_t \cdot CRT_t$ where nt_t is the number $(= np_t \cdot cc_t \cdot ct_t)$ of threads of the server s_t. The maximum server computation rate $maxCR_t$ of a server s_t shows the maximum throughput of the server s_t. If there are multiple fastest servers, a server s_f whose $maxCR_f$ is largest is taken as a fastest server in a cluster S.

The total number VC_i of virtual computation steps of a process p_i is $minT_i$ [tu] $\cdot$ CRT_f [vs/tu] = $minT_i$ [vs] for a fastest server s_f. Thus, $minT_i$ shows the total amount of computation of each process p_i. For a pair of processes p_i and p_j, p_i is $longer$ than p_j (p_j is shorter than p_i) $(p_i > p_j)$ if and only if (iff) $minT_i > minT_j$. It takes longer time to perform a process p_i than a process p_j on a server if $p_i > p_j$. A pair of processes p_i and p_j are equivalent $(p_i \equiv p_j)$ if $minT_i = minT_j$. $p_i \geq p_j$ iff $p_i > p_j$ or $p_i \equiv p_j$. The maximum computation rate $maxCR_{ti}$ of a process p_i on a server s_t is VC_i / $minT_{ti}$ = $minT_i$ / $minT_{ti}$ (≤ 1). Hence, for every pair of processes p_i and p_j on a server s_t, $maxCR_{ti}$ = $maxCR_{tj}$ = CRT_t.

The $server\ computation\ rate$ $CR_t(\tau)$ 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 CRT_t is the thread computation rate of the server s_t. We assume the computation CPU resource is fairly allocated to each current process in every server s_t. Hence, each process p_i is performed at the process computation rate $CR_{ti}(\tau) = CR_t(\tau)$ / $\mid CP_t(\tau) \mid$ where processes in the process set $CP_t(\tau)$ are concurrently performed at time τ.

The process computation rate $CR_{ti}(\tau)$ ($\leq CRT_t$) indicates the computation rate [vs/tu] of a process p_i on a server s_t at time τ.

[MLCM (Multi-Level Computation with Multiple CPUs) model] The process computation rate $CR_{ti}(\tau)$ [vs/tu] of a process p_i on a server s_t to perform process at time τ is given as follows:

$$CR_{ti}(\tau) = \begin{cases} nt_t \cdot CRT_t \,/\, |CP_t(\tau)| & \text{if } |CP_t(\tau)| > nt_t. \\ CRT_t & \text{if } |CP_t(\tau)| \leq nt_t. \end{cases} \tag{4.3}$$

The process computation rate $NCR_{ti}(n)$ of a process p_i on a server s_t, where n processes are performed is given as follows:

$$NCR_{ti}(n) = \begin{cases} maxCR_t(= nt_t \cdot CR_t(\tau))/n & \text{if } n > nt_t. \\ CRT_t & \text{if } n \leq nt_t. \end{cases} \tag{4.4}$$

Figure 4.4 shows the process computation rate $NCR_{ti}(n)$ of a process p_i on a server s_t where n processes are concurrently performed. If a fewer number of processes than the total number nt_t of threads are performed, the process computation rate $NCR_{ti}(n)$ of each process p_i is the thread computation rate CRT_t since only the process p_i is performed on a thread. If more number of processes than the total number nt_t of threads are performed, at least one process is performed on every thread. For example, if 2 $\cdot$ nt_t processes are performed on a server s_t, two processes are performed on each thread. Here, the process computation rate $NCR_{ti}(n)$ of each process p_i is CRT_t / 2. Thus, the process computation rate $NCR_{ti}(n)$ of each process decreases as the number of processes concurrently performed increases.

Suppose there are a pair of servers s_u and s_v with numbers nt_u and nt_v of threads, respectively, and each thread of the servers s_u and s_v supports the same thread computation rate as the server s_t, i.e. CRT_u = $CRT_v = CRT_t$. Suppose the server s_t supports nt_u (= nt_t / 2) threads and the server s_u supports nt_v (= nt_t / 4) threads for the server s_t. A process p_i is performed on each of the servers concurrently with ($n - 1$) processes, i.e. totally n processes are concurrentlly performed. In Figure 4.4, the straight line shows the process computation rate $CR_{ti}(\tau)$ of the process p_i on the server s_t and a pair of dashed and dotted lines show the process computation rates $CR_{ui}(\tau)$ and $CR_{vi}(\tau)$ of the process p_i on the servers s_t and s_u, respectively, for number n of processes.

We measure the execution time of processes which are performed on the server DSLab [9] which supports thirty two threads (nt_t = 32) on a pair of CPUs. Figure

4.5 shows the average execution time T of n processes $p_1, \ldots, p_n$. The processes are created by forking a process p to n child processes $p_1, \ldots, p_n$ as presented in the preceding section. The n processes are concurrently performed on the server. The minimum execution time $minT_i$ of each process p_i is 0.9 [sec] on the DSLab server. The average execution time of the n processes is almost the same for $n \leq 32$ since at most one process is performed on each thread. The average execution time linearly increases as the number n of processes increases for $n > 32$. The process computation rate $CR_{ti}(\tau)$ of each process p_i is $minT_i$ / T ($\leq$ 1) where T is the execution time of the process p_i. Figure 4.5 shows the computation model (formula (4.4)) holds on a real server.

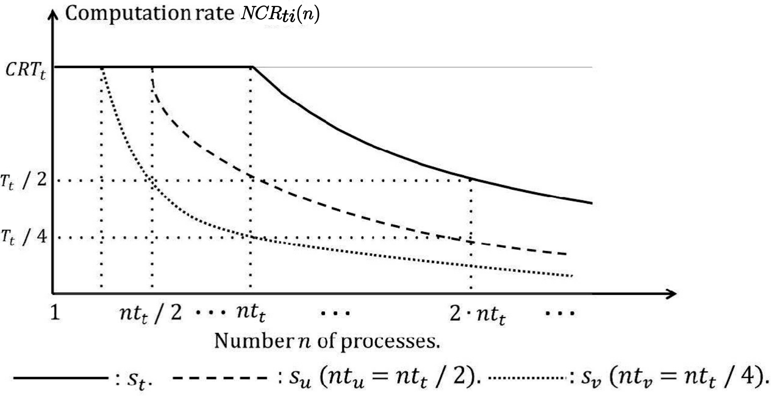

Figure 4.4: Process computation rate of a process p_i.

Suppose a process p_i on a server s_t starts at time st and ends at time et. Here, $\sum_{\tau=st}^{et} CR_{ti}(\tau)$ = VC_i [vs] (= $minT_i$) shows the total amount of computation, i.e. total number of virtual computation steps to be performed by a process p_i. Figure 4.6 shows the process computation rates $CR_{ti}(\tau)$ and $CR_{tj}(\tau)$ of processes p_i and p_j, respectively, which are performed on the same thread of a server s_t. Here, a pair of the processes p_i and p_j start at time τ_1 and τ_2, respectively. Then, the processes p_i and p_j terminate at time τ_3 and τ_4, respectively. The process p_i is performed at the thread computation rate CRT_t from time τ_1 to time τ_2 since only one process, i.e. p_i is performed on the thread. The process computation rate $CR_{ti}(\tau) = CRT_t$ for $\tau_1 \leq \tau < \tau_2$. At time τ_2, the process p_i starts on the thread. Here, since a pair of the processes p_i and p_j are performed, the processes p_i and

p_j are performed at the same computation rate CRT_t / 2. $CR_{ti}(\tau) = CR_{tj}(\tau) = CRT_t$ / 2 for $\tau_2 \leq \tau < \tau_3$. At time τ_3, the process p_i terminates and then only the process p_j is performed. Here, the computation rate of the process p_j increases to the thread computation rate CRT_t. $CR_{tj}(\tau) = CRT_t$ for $\tau_3 \leq \tau_1$ The hatched area shows the total computation, i.e. total number VC_i of virtual computation steps of the process p_i, VC_i = $\sum_{\tau=\tau_1}^{\tau_3} CR_{ti}(\tau)$.

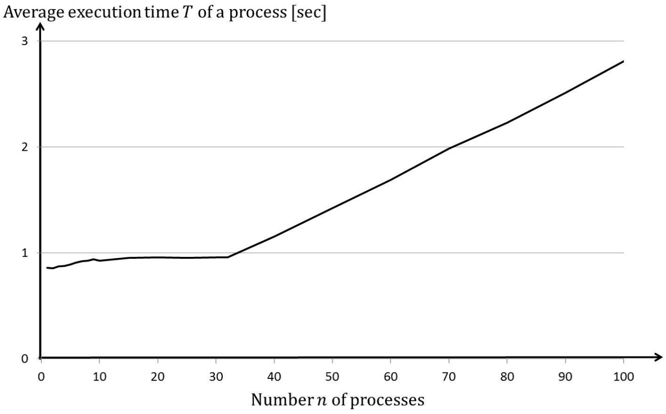

Figure 4.5: Average execution time of processes on DSLab server.

The computation of each process p_i is modeled as follows:
[Process computation model of a process p_i]

1. At time τ a process p_i starts on a server s_t, the computation laxity $lc_{ti}(\tau)$ of a process p_i is VC_i.

2. At each time τ, the computation laxity $lc_{ti}(\tau)$ of a process p_i is decremented by the process computation rate $CR_{ti}(\tau)$, i.e. $lc_{ti}(\tau + 1) = lc_{ti}(\tau) - CR_{ti}(\tau)$.

3. If the computation laxity $lc_{ti}(\tau + 1)$ of a process p_i gets equal to or smaller than zero, the process p_i terminates at time τ.

The computation laxity $lc_{ti}(\tau)$ of a process p_i is initially VC_i $(= minT_i)$ at time τ the process p_i starts. Then, the computation laxity $lc_{ti}(\tau)$ is decremented by the process computation rate $CR_{ti}(\tau)$ at each time τ. The more number of processes are performed, the smaller process computation rate $CR_{ti}(\tau)$. If the computation laxity $lc_{ti}(\tau) - CR_{ti}(\tau)$ gets zero or smaller than zero, the process p_i terminates at time τ.

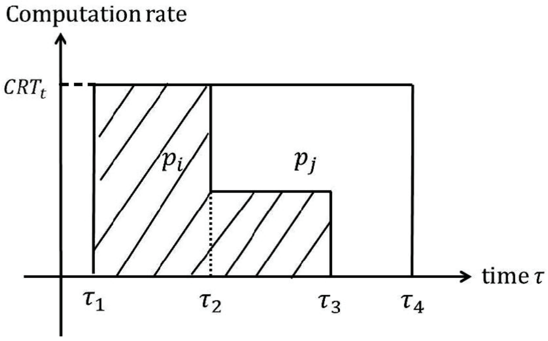

Figure 4.6: Computation rates of processes p_i and p_j.

Chapter 5

Process Migration

5.1 Models to Estimate Electric Energy Consumption

In this chapter, we discuss how a process migrates from a host server to a gust server where the process can be more energy-efficiently performed. We take the SC (Simple computation) and SPC (Simple Power Consumption) models. Here, the maximum computation rate $maxCR_t$ of a server s_t is the thread computation rate CRT_t. In the simple power consumption (SPC) model [1, 21, 22] 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 $TE_t(\tau_1, \tau_2)$ consumption consumed by a server s_t from time τ_1 to time τ_2 is $\sum_{\tau=\tau_1}^{\tau_2} E_t(\tau)$ [Wtu] where tu shows one time unit.

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 [22, 23, 24], 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 [22, 23]. In this paper, one unit time is 100 [msec] since we can measure the power consumption of a server every 100 [msec] [22, 23]. The expected termination time $ETP(s_t, CP_t(\tau), p_i, \tau)$ is given as time τ_t in the following procedure:

```
$ETP$ $(s_t, CP_t(\tau), p_i, \tau)$ {
        $lc = lc_{ti}(\tau)$;   /* laxity of $p_i$ on server $s_t$ */
```

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

Here, the computation rate $CR_{ti}(\tau)$ of a process p_i at time τ is $CR_t(\tau)$ / | $CP_t(\tau)$ | on a server s_t as discussed in the preceding chapter. Here, $CR_t(\tau)$ is the computation rate of a server s_t. 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:

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

Every current process in the set $CP_t(\tau)$ 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.

5.2 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 as presented. The process p_i is performed on the server s_t. Then, the process p_i migrates from the host server s_t to another server s_u as shown in Figure 5.1. 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 a pair of the servers s_t and s_u are $migrated\ servers$ of the process p_i.

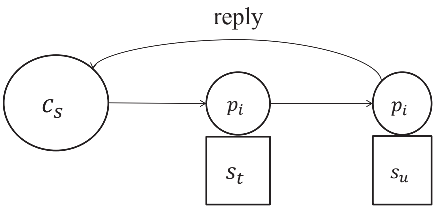

Figure 5.1: Migration of a process.

A process on a host server s_t migrates 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 host server s_u. We discuss migration conditions that a process on one host server migrates to another server. Suppose a process p_i is performed on a host server s_t at time τ. There are two ways to perform the process p_i [Figure 5.2]:

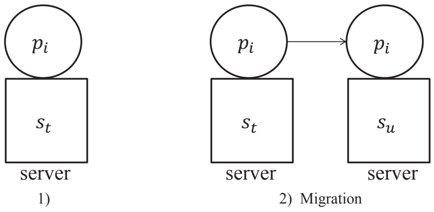

Figure 5.2: Process migration.

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

2 The process p_i is performed on another server s_u by migrating the process p_i from the host server s_t to the server s_u.

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 in the set $CP_t(\tau)$ 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 the set $CP_t(\tau)$ to terminates at time $ETP(s_t, CP_t(\tau), p_i, \tau)$.

Next, suppose the process p_i migrates 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 in the set $CP_u(\tau + \delta_i)$ 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 the set $CP_u(\tau)$ is decremented by the computation rate $CR_{uj}(\tau)$ of the process p_j on the server s_u. 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 {   $C = CR_u(x)$ / $|CP_u(x)|$;
       for  every process $p_{uj}$ in $CP_u(x)$
       do {
          $lc_{uj}(x+1) = lc_{uj}(x) - C$;
        if  $lc_{uj}(x+1) = 0$,
           $CP_u(x+1) = CP_u(x)$ - $\{p_{uj}\}$;
        }; /* for end */
}; /* for $x$ end */
```

5.3 Server Selection

A process p_i on a host server s_t can migrates 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 host server s_t if the process p_i on the host server s_t is migrated to the server s_u. In Figure 5.3, 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 migrates 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 migrates 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 migrate from the host server s_t to the server s_u. Otherwise, the process p_i might not terminate by the deadline dl_i if the process p_i migrates to the server s_u.

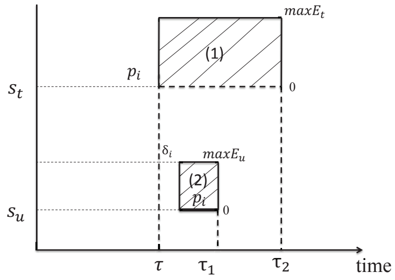

Figure 5.3: Expected termination time.

In Figure 5.4, τ_{ti} shows time by when every current process in the set $CP_t(\tau)$ 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 host server s_t at time τ. Suppose the process p_i on the host server s_t migrates 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 the set $CP_t(\tau)$ is $ET(s_t, CP_t(\tau) - p_i, \tau)$. Here, $\tau_{ti} < \tau_t$ since the process p_i migrates from the host server s_t 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

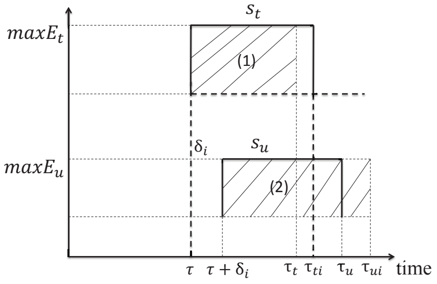

Figure 5.4: Expected energy consumption.

$ET(s_u, CP_u(\tau+\delta_i) \cup \{p_i\}, \tau+\delta_i)$ + δ_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 migrates from the host server s_t 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 host server s_t at time τ as follows:

[Process migration]

```
        $E = EE(s_t, CP_t(\tau), \tau)$;
        $T = dl_i - \tau$;   /* 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 */
          $E = EE(s_u, CP_u(\tau + \delta_i) \cup \{p_i\}, \tau + \delta_i)$;
          $T = ET(s_u, CP_u(\tau + \delta_i) \cup \{p_i\}, \tau + \delta_i)$;
          $s = s_u$;
          };
       }; /* for end */
    };
```

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 host server s_t. Here $\gamma = maxT_i$ / 4. If the MG conditions are satisfied on the server s_t. The process p_i migrates to a guest server s.

Chapter 6

Static Migration of Virtual Machines

6.1 Computation Model of a Virtual Machine

6.1.1 Computation model

It is not easy for processes to migrate among types of servers, e.g. heterogeneous servers with different operating systems and architectures. For example, it is difficult, almost impossible for a C process to migrate from a Linux server to a Windows server. A cluster like cloud computing systems support applications with virtual computation services through virtual machines like KVM [44]. We consider a process migration way to migrate processes by using virtual machines. Processes issued by clients are performed on virtual machines in a cluster. The process computation rate $CR_{ti}(\tau)$ of each process p_i depends on the total number $|CP_t(\tau)|$ of processes but is independent of the number $|SVM_t(\tau)|$ of virtual machines of a host server s_t. Let p_{hi} show a resident process p_i which is performed on a virtual machine vm_h of a server s_t.

The $virtual\ machine\ (VM)\ laxity\ vlc_h(\tau)$ [vs] of a virtual machine vm_h at time τ is defined to be the summation of computation laxities of the resident processes of the virtual machine vm_h:

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

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

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

Hence, the VM computation rate $VCR_h(\tau)$ [vs/sec] of a virtual machine vm_h is defined as follows:

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

$$\begin{cases} maxCR_t \cdot |VCP_h(\tau)| \;/\; |CP_t(\tau)| & \text{if } |CP_t(\tau)| > nt_t. \\ CRT_t \cdot |VCP_h(\tau)| & \text{if } |CP_t(\tau)| \leq nt_t. \end{cases} \tag{6.1}$$

That is, $CR_{ti}(\tau) = CR_{tj}(\tau)$ for every pair of resident processes p_i and p_j of virtual machines on a server s_t. Here, $VCR_h(\tau) \leq VCR_k(\tau)$ if $|VCP_h(\tau)| \leq |VCP_k(\tau)|$ for every pair of resident virtual machines vm_h and vm_k on a same server s_t. $VCR_h(\tau)$ / $VCR_k(\tau) = |VCP_h(\tau)|$ / $|VCP_k(\tau)|$ for every pair of resident virtual machines vm_h and vm_k on a server.

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

[VM computation (VMC) model on a virtual machine VM_h of a server s_t]

```
while ($|VCP_h(\tau)| > 0$)
{
    for every process $p_i$ which starts on $vm_h$ at time $\tau$, {
        $VCP_h(\tau) = VCP_h(\tau) \cup \{p_i\}$;
        $VCP_h(\tau + 1) = VCP_h(\tau)$;
    /* for end */
  for each process $p_i$ on a virtual machine $vm_h$, i.e. $p_i \in VCP_h(\tau)$,
  {
      $plc_i(\tau + 1) = plc_i(\tau) - VCR_h(\tau) / |VCP_h(\tau)|$;
      if $plc_{hi}(\tau + 1) \leq 0$; /* $p_i$ terminates at time $\tau$ */
          $VCP_h(\tau + 1) = VCP_h(\tau + 1) - \{p_i\}$;
  }; /* for end */
  $vlc_h(\tau + 1) = vlc_h(\tau) - VCR_h(\tau)$;
  $\tau = \tau + 1$;
}; /* while end */
```

At each time τ, if a process p_i starts on a virtual machine vm_h, the process p_i is added to a set $VCP_h(\tau)$. Then, for every process p_i on the virtual machines vm_h, the computation laxity of the process p_i is decremented by the process computation rate $VCR_t(\tau)$ / $|VCP_t(\tau)|$, i.e. $CR_t(\tau)$ / $|CP_t(\tau)|$. Here, $plc_i(\tau + 1)$ = $plc_i(\tau)$ − $VCR_h(\tau)$ / $|VCP_t(\tau)|$. If the computation laxity $plc_i(\tau + 1) \leq 0$, the process p_i terminates and $VCP_h(\tau + 1) = VCP_h(\tau + 1) - \{p_i\}$. Then, the VM laxity $vlc_h(\tau)$ is decremented by the VM computation rate $VCR_t(\tau)$, i.e. $vlc_h(\tau + 1) = vlc_h(\tau) - VCR_h(\tau)$. If $vlc_h(\tau + 1) \leq 0$, the virtual machine vm_h gets idle.

6.1.2 Estimation model

Suppose, every resident process of the virtual machine vm_h terminates at time τ. Here, time τ is idled time of the virtual machine vm_h. Time before when at least one virtual machine is active and after when no virtual machine is active on a server s_t is referred to as *termination* time ET_t of the server s_t. EE_t shows the electric energy to be consumed by a server s_t by the termination time ET_t. We assume no new process is issued to a server s_t after time τ. We estimate the termination time ET_t and electric energy consumption EE_t of a server s_t to perform every process by considering active virtual machines, not each process, in the following procedure **VMEST** $(s_t, \tau, SVM_t(\tau); EE_t, ET_t)$:

[Virtual machine computation (VMC) model]

```
VMEST $(s_t, \tau, SVM; EE_t, ET_t)$
input  $s_t$,  $\tau$,  $SVM$; /*set of VMs */
output $EE_t$;  $ET_t$;
{ $ncp = |CP_t(\tau)|$; /*number of processes on $s_t$*/
    $vlc = 0$;
    $x = \tau$; /* current time */
    $EE_t = 0$;
  /* obtain laxity $vlc$ of the server $s_t$ */
   for each virtual machine $vm_h$ in $SVM$,
   {
       /* $VM$ laxity of $vm_h$ */
     $vlc_h = vlc_h(\tau)$ $(= \sum_{p_i \in VCP_h(\tau)} plc_i(\tau))$;
     $ncp_h = |VCP_h(\tau)|$; /*number of processes on
     $vm_h$*/
```

```
    $vlc = vlc + vlc_h$; /* server laxity of $s_t$ */
  }; /* for end */
  while ($SVM \neq \phi$)
  {
    $EE_t = EE_t + CE_t(ncp)$; /* electric energy */
    for each virtual machine $vm_h$ in $SVM$,
    {
     $vlc_h = vlc_h - VCR_h(x)$;
     /* VM laxity is decremented */
     if $vlc_h \leq 0$, /*$vm_h$ gets idle */ {
          $SVM = SVM - \{vm_h\}$;
          $ncp = ncp - ncp_h$;
      } else $vlc = vlc - vlc_h$; /*decrement server laxity*/
     }; /* for end */
     $x = x + 1$; /* time advances */
   }; /* while $SVM$ end */
   $ET_t = x - 1$; /* every virtual machine gets idle on $s_t$*/
};
```

Here, the VM computation rate $VCR_h(\tau)$ of a virtual machine vn_h depends on how many number of processes are totally performed on the virtual machine vm_h at time τ. The more number of processes are performed on a virtual machine vm_h, the larger VM computation rate $VCR_h(\tau)$. Here, it is noted we do not consider the termination time of each process p_i and only consider the idled time of each virtual machine. This means, the computation complexity of the estimation gets simpler by our approach.

6.2 Energy-aware Virtual Machine Migration

A client issues a process p_i to a set VM of virtual machines $vm_1, \ldots, vm_v$ ($v \geq 1$) in a cluster S. We have to discuss a pair of algorithms on virtual machines:

1. VM selection (VMS): a virtual machine vm_h is selected to perform a new process p_i issued by a client.
2. VM migration (VMM): a virtual machine vm_h in a server s_t is selected to migrate to another server s_u.

6.2.1 VM selection (VMS) algorithms

First, a process p_i is issued to a cluster S by an application on a client. We discuss how to select a virtual machine vm_h to perform the process p_i in a cluster S [Figure 6.1]. In the first random VM selection (RDVMS) algorithm, a virtual machine vm_h is randomly selected in a set VM of virtual machines in a cluster S.

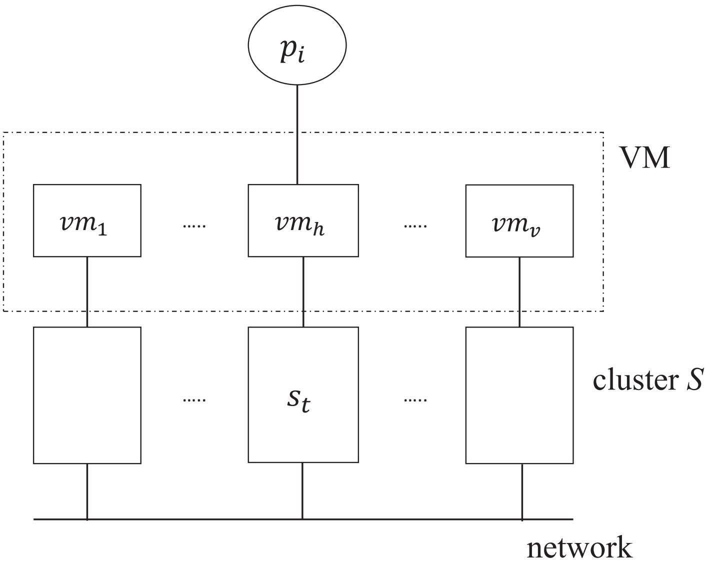

Figure 6.1: VM selection.

[Random VM selection (RDVMS)]

1. Randomly select a virtual machine vm_h in the virtual machine set VM.

In another random server selection (RDSS) algorithm, a server s_t which hosts virtual machines is randomly selected. Then, a smallest virtual machine vm_h where the number $|VCP_h(\tau)|$ of resident processes is minimum is selected in the selected server s_t.

[Random server selection (RDSS)]

1. Randomly select an engaged server s_t which hosts at least one virtual machine in a cluster S.
2. Select a smallest resident virtual machine vm_h of the server s_t where $|VCP_h(\tau)|$ is minimum, i.e. the fewest number of processes are performed.

In the second way, a virtual machine is selected in a round-robin (RR) manner. In a round-robin VM selection (RBVMS) algorithm, a virtual machine vm_h is selected in the virtual machine set VM as follows:

[Round-robin VM selection (RBVMS)]

1. Select a virtual machine vm_h in the round-robin (RR) algorithm, i.e. vm_h is selected after a virtual machine vm_{h-1} is selected.

In another round-robin server selection (RBSS) algorithm, an engaged server s_t which hosts at least one virtual machine is first selected. Then, a smallest resident virtual machine vm_h where the number $|VCP_h(\tau)|$ of processes is minimum is selected in the selected server s_t.

[Round-robin server selection (RBSS)]

1. Select an engaged server s_t in the round-robin (RR) algorithm, i.e. s_t is selected after s_{t-1} is selected.
2. Select a resident virtual machine vm_h of the server s_t where $|VCP_h(\tau)|$ is minimum, i.e. the fewest number of processes are performed.

In the third way, a virtual machine is selected so that the processing load is balanced among virtual machines. A smallest virtual machine vm_h is selected where the number of processes is minimum in a load-efficient VM selection (LVMS) algorithm.

[Load-efficient VM selection (LVMS)]

1. Select a smallest virtual machine vm_h where $|VCP_h(\tau)|$ is minimum in the set VM.

In another load-efficient server selection (LSS) algorithm, a server s_t where the number $|CP_t(\tau)|$ of processes is minimum is first selected. Then, a resident virtual machine vm_h of the server s_t is selected.

[Load-efficient server selection (LSS)]

1. Select a smallest engaged server s_t where $|CP_t(\tau)|$ is minimum in a cluster S.
2. Select a smallest virtual machine vm_h on the host server s_t where $|VCP_h(\tau)|$ is minimum.

In the last way, a virtual machine is selected so that the electric energy consumption of the servers can be reduced. The expected electric energy consumption EE_t and expected termination time ET_t of a server s_t to perform every current process on the virtual machines are obtained by the procedure **VMEST** $(s_t, \tau; ET_t, EE_t)$. Then, one virtual machine vm_h on a server s_t is selected to perform a process p_i in an energy-efficient VM selection (EVMS) algorithm. Here, a smallest virtual machine vm_h is selected as follows:

[Energy-efficient VM selection (EVMS)]

1. **for** each server s_u in a cluster S, **VMEST** $(s_u, \tau; EE_u, ET_u)$;
2. $MS = \{s_u \mid EE_u$ is minimum in $S\}$;
3. **select** s_t in MS where $|CP_t(\tau)|$ is minimum;
4. **select** a smallest virtual machine vm_h in the selected server s_t where $|VCP_h(\tau)|$ is minimum;

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

6.2.2 VM migration (VMM) algorithms

Next, virtual machines migrate to guest servers. We first have to discuss the following points in the VMM algorithms.

[VM condition]

1. When servers are checked if a resident virtual machine migrates to a guest server.
2. Conditions to migrate virtual machines on host servers.

If the migration conditions are satisfied, a virtual machine vm_h on a host server s_t migrates to another guest server s_u as shown in Figure 6.2. Here, we have to discuss the following points:

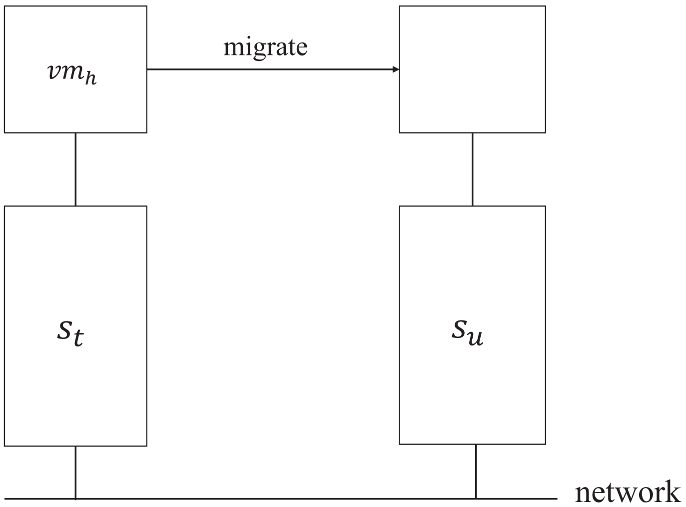

Figure 6.2: VM migration.

[VM migration (VMM)]

1. Host server (HS) selection: one engaged server s_t which hosts at least one virtual machine is selected.
2. Virtual machine selection: one resident virtual machine vm_h on a selected server s_t is selected.
3. Guest server (GS) selection: one guest server s_u is selected, to which the virtual machine vm_h is to migrate from the server s_t.
4. Virtual machine migration (VMM): the virtual machine vm_h migrates from the host server s_t to the guest server s_u.

Each engaged server s_t is periodically checked and the expected electric energy consumption EE_t of each server s_t to perform every current process is obtained. Then, a host server s_t is selected, where the expected electric energy consumption EE_t is maximum in a cluster S.

[VM condition]

1. Each engaged server s_t is periodically checked and the expected electric energy consumption EE_t of the server s_t obtained by the procedure **VMEST** $(s_u, \tau, SVM_t(\tau); EE_t, ET_t)$.

2. A server s_t whose expected electric energy consumption EE_t is maximum is selected. Then one virtual machine is selected on the server s_t and migrates to another server.

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

[Energy-efficient VMM (EVMM) selection on s_t]

```
$HS$ = a set of engaged servers in a cluster $S$;
for each engaged server $s_u$ in $HS$,
    VMEST $(s_u, \tau; EE_u, ET_u)$;
while ($HS \neq \phi$)
{
    select $s_t$ whose $EE_t$ is maximum in $HS$;
    select $vm_h$ on $s_t$ where $|VCP_h(\tau)|$ is minimum;
    VMEST $(s_t, \tau, SVM_t(\tau) - \{vm_h\}; NE_t, NT_t)$;
    for each server $s_u$ in $S$ ($s_u \neq s_t$)
     {
        VMEST $(s_u, \tau, SVM_u(\tau) \cup \{vm_h\}; NE_u, NT_u)$;
    }; /* for end */
    select a server $s_u$ where $(EE_u + EE_t) - (NE_t + NE_u)$ ($> 0$) is the largest;
    if found,
    {
     migrate $vm_h$ from $s_t$ to $s_u$;
     $SVM_t(\tau) = SVM_t(\tau) - \{vm_h\}$;
```

```
        $SVM_t(\tau) = SVM_t(\tau) \cup \{vm_h\}$;
        $EE_t = NE_t$; $EE_u = NE_u$;
      };
      **else** $HS = HS - \{s_t\}$;
}; /* **while** end */
```

Chapter 7

Dynamic Migration of Virtual Machines

7.1 Simple Estimation Model

In order to select a host server where a process is to be performed, we have to estimate the execution time of each current process on each server. In the estimation model [22, 38] discussed in the preceding chapter, the number VC_i of virtual computation steps of each process p_i is collected and then the expected termination time of each process p_i with the other processes is calculated by the computation model. However, it takes time to estimate the termination time of each process on each server s_t [17, 18, 19].

In this paper, we propose a simple model to estimate the termination time of each process p_i on a server s_t. We assume each process p_i has the same total number of virtual computation steps, $VC_i = VC$ as discussed in paper [42]. In this thesis, we assume VC =1. We also assume that the half VC / 2 (= 1 / 2) of the total number of virtual computation steps of each current process p_i is finished. Here, suppose n $(= |CP_t(\tau)|)$ processes are currently performed on a server s_t. The total amount of computation to be performed by the n processes is n / 2. If k processes newly start on the server s_t, the number of virtual computation steps of the k new processes is k. Hence, totally (n / 2 + k) virtual computation steps [vs] are considered to be performed on the server s_t. It takes (n / 2 + k) / $NPR_t(n + k)$ time units [tu] to perform n current processes and k new processes on a server s_t. Hence, the expected termination time $SET_t(n, k)$ [tu] and expected electric energy consumption $SEE_t(n, k)$ [Wtu] of each server s_t to perform both

n current processes and k (≥ 0) new processes are given as follows:

$$SET_t(n, k) = (n/2 + k)/NPR_t(n + k). \tag{7.1}$$

$$SEE_t(n, k) = SET_t(n, k) \cdot CE_t(n+k) = (n/2+k) \cdot CE_t(n+k)/NPR_t(n+k). \tag{7.2}$$

In the estimation model, only the number n of current processes is used to estimate the electric energy consumption of each server s_t. In addition, the computation to estimate the termination time and electric energy consumption of a server is simple. Hence, it is easy to estimate the electric energy to be consumed by a server. Thus, the estimation model to estimate the electric energy consumption of a server is practical even in a cluster.

7.2 Dynamic Virtual Machine Selection (DVMS) Algorithm

In our previous studies [17, 18, 19, 55], the EAVM (Energy-Aware Virtual machine Migration) algorithm is proposed where virtual machines migrate from host servers to guest servers [19]. Here, the number v of virtual machines is fixed independently of number of processes performed on servers. Here, the more number of processes are issued to a cluster, the more number of processes are performed on each virtual machine. If a large number of processes are performed on a virtual machine vm_h, the virtual machine vm_h may not be able to migrate from a host server to another guest server s_u since the virtual machine vm_h is too large to be performed on the guest server s_u.

We propose a $dynamic\ virtual\ machine\ migration$ ($DVMM$) algorithm [19], where virtual machines are dynamically created and dropped depending on the number of processes on host servers so that virtual machines can anytime migrate to other servers. Let VM be a set of virtual machines in a cluster S of servers $s_1, \ldots, s_m$ ($m \geq 1$). VM_t shows a set of resident virtual machines on each server s_t ($t = 1, \ldots, m$). Initially, $VM = \phi$ and $VM_t = \phi$ for every server s_t. Suppose a process p_i is issued to a cluster S at time τ. Here, we assume n_t ($= |CP_t(\tau)| \geq 0$) processes are concurrently performed on each server s_t. Let nv_h show the number $|VCP_h(\tau)|$ of resident processes on each virtual machine vm_h at current time τ.

One server s_t is selected in the cluster S and then a virtual machine vm_h is created or selected in the host server s_t depending on the number of processes. In addition, virtual machines migrate from host server to guest servers. Idle virtual machines are also dropped on a server to reduce the number of virtual machines. The DVMM algorithm is thus composed of the following algorithms.

1. DVMS (Dynamic Virtual Machine Selection).
2. DVMD (Dynamic Virtual Machine Drop).
3. DVMM (Dynamic Virtual Machine Migration).

First, we discuss the DVMS algorithm to select a host server s_t and then a virtual machine is selected on the server s_t to perform a process p_i issued by a client.

[Dynamic VM selection (DVMS)] A process p_i is issued to a cluster S.

1. Select a host server s_t to perform the process p_i whose expected electric energy consumption $SEE_t(n_t, 1)$ to perform both the process p_i and n_t current processes is minimum in the cluster S.
2. If the server s_t is free, i.e. there is no resident virtual machine, go to 5.
3. Select a smallest virtual machine vm_h resident on the server s_t, i.e. nv_h is minimum.
4. If $nv_h \leq maxNVM_t$, perform the process p_i on the virtual machine vm_h of the server s_t [Figure 7.1].
5. Otherwise, create a new virtual machine vm_k on the server s_t [Figure 7.2], i.e. $VM_t = VM_t \cup \{vm_k\}$ and $VM = VM \cup \{vm_k\}$ and perform the process p_i on the virtual machine vm_k.

As discussed in papers [55, 56], the average execution time of processes is independet of number of virtual machines and just depends on number n_t of current processes performed on a host server s_t. Hence, a server s_t is first selected in the

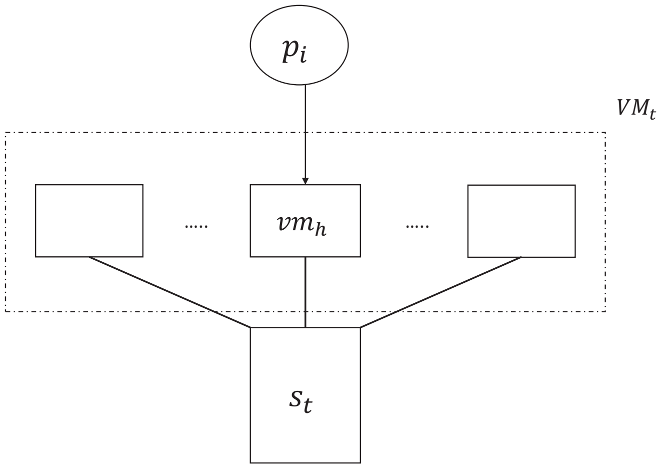

Figure 7.1: VM selection.

cluster S, whose expected electric energy consumption $SEE_t(n_t, 1)$ is minimum to perform both n_t current processes and one new process p_i issued by a client. A new virtual machine vm_h is created to perform a new process p_i on a selected server s_t if $VM_t = \phi$, i.e. there is no virtual machine on the server s_t. Processes are thus issued to the smallest virtual machine vm_h on the selected server s_t. If the number nv_h of processes on the smallest virtual machine vm_h is larger than a constant value $maxNVM_t$, i.e. $nv_h > maxNVM_t$, a new virtual machine vm_k is created to perform the process p_i on the server s_t. Otherwise, the smallest resident virtual machine vm_h is selected on the server s_t and the process p_i is performed on the server s_t.

Thus, the number of virtual machines monotonically increases as a new process is issued to the cluster. We have to reduce the number of virtual machines. Each server s_t is periodically checked if there is an idle resident virtual machine. If there is an idle virtual machine vm_h on the server s_t, the virtual machine vm_h is dropped as follows:

[Dynamic VM drop (DVMD)]

1. An idle virtual machine vm_h on a server s_t is dropped.

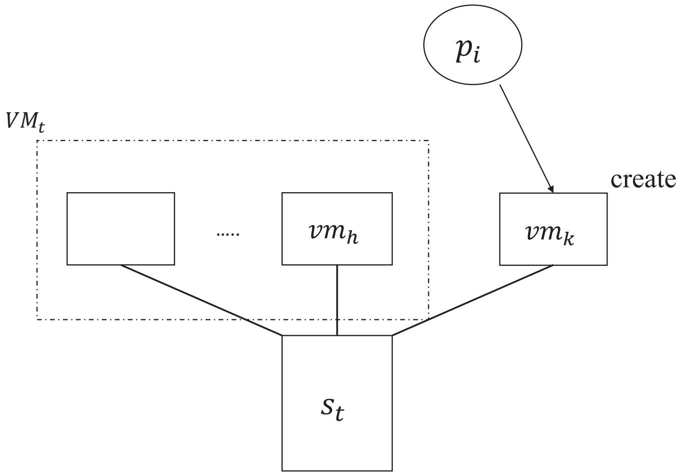

Figure 7.2: VM creation.

2. $VM_t = VM_t - \{vm_h\}$ and $VM = VM - \{vm_h\}$;

In the VM selection (VMS) algorithm, if a new process is issued to a cluster S, a virtual machine vm_h is created or selected on a server depending on the number of processes performed on the server. On the other hand, each server s_t is periodically checked. If there is an idle virtual machine vm_h on the server s_t, the idle virtual machine vm_h is dropped on the server s_t.

Thus, virtual machines are dynamically created and dropped depending on the number of processes performed in the cluster S. The more number of processes are performed, the more number of virtual machines on the servers. This means, the number vn_h of resident processes on each virtual machine vm_h is reduced so that the virtual machine vm_h can anytime migrate from the host server to another server.

7.3 Dynamic Virtual Machine Migration (DVMM) Algorithm

Processes issued by clients are performed on servers as discussed in the preceding section. In addition to selecting a host server for each new process issued by a client, virtual machines migrate from host servers to guest servers in order to reduce the electric energy consumption of servers depending on the number of current processes.

Each engaged server s_t is periodically checked if an active virtual machine vm_h on the server s_t is to migrate to another guest server so that the electric energy consumption of the servers can be reduced.

[Dynamic VM migration (DVM)] The following procedure is periodically performed to migrate virtual machines to another guest server for each engaged host server s_t where at least one virtual machine resides:

1. Obtain the expected electric energy consumption $EE_u = SEE_u(n_u, 0)$ and expected termination time $ET_u = MET_u(n_u, 0)$ of every server s_u to perform only n_u current processes ($u = 1, \ldots, m$).

2. Select a smallest virtual machine vm_{th} in the set VM_t on the host server s_t. Obtain the expected electric energy consumption $NE_t = SEE_t(n_t - nv_{th}, 0)$ and termination time $NT_t = SET_t(n_t - nv_{th}, 0)$ of the server s_t to perform every process after the virtual machine vm_{th} migrates to another server.

3. Obtain the expected electric energy consumption $NE_{tu} = SEE_u(n_u + nv_{th}, 0)$ and expected termination time $NT_{tu} = SET_u(n_u + nv_{th}, 0)$ of each server s_u to perform not only n_u current processes but also nv_{th} processes on the virtual machine vm_{th} ($u = 1, \ldots, m$, $u \neq t$).

4. Select a server s_u where $EE_t + EE_u > NE_t + NE_{tu}$ and NE_{tu} is minimum ($u \neq t$) [7.3]. If found, the virtual machine mv_{th} migrates from the host server s_t to the guest server s_u. Otherwise, no virtual machine migrates from the server s_t.

For each engaged server s_t in a cluster S, the DVM migration (DVMM) algorithm is performed. At step 1, the expected electric energy consumption EE_u and expected termination time ET_u of each server s_u including the host server s_t

are obtained where only n_u current processes are performed. Here, we assume no resident virtual machine migrates to or from the server s_u, $EE_u = SEE_u(n_u, 0)$ and $ET_u = SET_u(n_u, 0)$. At step 2, a smallest resident virtual machine vm_{th} is selected on the server s_t. The virtual machine vm_{th} is a candidate to migrate from the server s_t to another guest server. The expected electric energy consumption NE_t of the server s_t is obtained by the estimation procedure $SEE_t(n_t - nv_{th}, 0)$, where the virtual machine vm_{th} migrates from the server s_t to another server. That is, nv_{th} processes on the virtual machine vm_{th} leave the server s_t. Next, the virtual machine vm_{th} migrates from the host server s_t to another server s_u. Here, nv_{th} processes carried by the virtual machine vm_{th} are performed on the guest server s_u in addition to n_u current processes. Hence, the expected electric energy consumption NE_{tu} of each guest server s_u ($\neq s_t$) is obtained as $NE_{tu} = MEE_t(n_u + nv_{th}, 0)$. If $EE_t + EE_u > NE_t + NE_{tu}$, the total electric energy to be consumed by a pair of the servers s_t and s_u is reduced to $NE_t + NE_{tu}$ by migrating the virtual machine vm_{th} from the host server s_t to the guest server s_u. We find a guest server s_u where the total electric energy consumption $NE_t + NE_{tu}$ of both the host server s_t and the guest server s_u can be mostly reduced and the electric energy consumption NE_{tu} of the guest server s_u is minimum. If such a server s_u is found, the virtual machine vm_{th} migrates from the host server s_t to the guest server s_u.

Suppose there are a pair of servers s_t and and s_u as shown in Figure 7.3. ET_t and ET_u show a pair of termination time of the servers s_t and s_u, respectively, where no virtual machine on the host server s_t migrates to the guest server s_u. That is, some current process is performed on the server s_t before time ET_t and no process is performed after time ET_t. In Figure 7.3, a pair of the dotted area show the electric energy EE_t and EE_u to be consumed by the servers s_t and s_u to perform every current process, respectively.

Next, a virtual machine vm_h on the host server s_t migrates to the guest server s_u. Here, processes resident on the virtual machine vm_h move from the host server s_t to the guest server s_u. Hence, the other processes on the server s_t terminate at time NT_t before time ET_t. On the other hand, more number of processes are performed on the server s_u since the resident processes of the virtual machine vm_h are additionally performed. The hatched area of the server s_t in Figure 7.3 shows the electric energy consumption of the servers s_t and s_u to perform the resident processes of the virtual machine vm_h. By migrating the virtual machine vm_h to the guest server s_u, the server s_t consumes the smaller electric energy. On the other hand, the electric energy consumption of the server s_u increases since the virtual machine vm_h with nv_h processes newly come. The area obtained by removing the hatched area from the doted area shows the electric energy consumption NE_t of the server s_t. On the other hand, the area obtained by adding the hatched area to the dotted area shows the electric energy computation NE_{tu} of the server s_u. The doted area of the server s_u in Figure 7.3 shows the electric energy consumption $NE_t + NE_{tu}$ to perform the processes on the virtual machine vm_h. If $NE_t + NE_{tu}$ is smaller than $EE_t + EE_u$ the virtual machine vm_h can migrate from the server s_t to the server s_u.

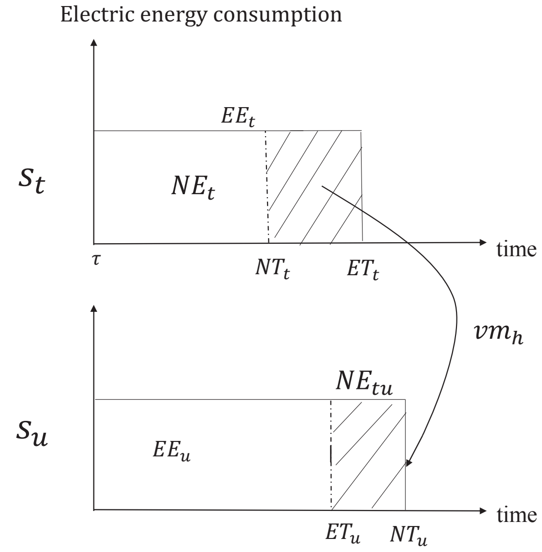

Figure 7.3: Electric energy consumption.

Chapter 8

Evaluation

8.1 Process Migration

8.1.1 Environment

We first evaluate the energy-efficient process migration (MG) algorithm in terms of total energy consumption and total execution time of servers. We consider a cluster S which is composed of m ($\geq$ 1) servers $s_1, \cdots, s_m$. Each server s_t follows the simple power consumption (SPC) model [22, 23] with maximum power consumption $maxE_t$ and minimum power consumption $minE_t$. In this evaluation, the maximum power consumption $maxE_t$ is randomly taken out of 100 to 200 [W] and the minimum power consumption $minE_t$ is also randomly taken out of 80 to 100 [W] for each server s_t. In each server s_t, the maximum computation rate $maxCR_t$ 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 $CR_t(\tau)$ of a server s_t is given $a_t^{l-maxN_t-1} \cdot CRT_t$ for number $n = \mid CP_t(\tau) \mid$ of processes concurrently performed at time τ as presented in this paper.

Totally n ($\geq$ 1) processes are performed on the servers $s_1, \cdots, s_m$ 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 [tu]. 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 at time $etime$ when every process terminates. Here, $etime \geq xtime$.

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 as a host server of each process p_i in the clusters of m servers $s_1, \cdots, s_m$. 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 $m + 1$. In the MG algorithm, a server s_t whose expected electric power consumption is minimum is selected to perform each process p_i. The process p_i is performed on the selected host server s_t. Every $\gamma_i = minT_t$ / 4 time units, it is checked from the process p_i 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. the server s_u is expected to consume a smaller amount of electric energy to perform processes than the host server s_t, the process p_i migrates to the guest server s_u. The delay time δ_i to migrate the process p_i from host server s_t to another guest server s_u is the half of the maximum minimum computation time, i.e. $\delta_i =$ 20 / 2 $=$ 10 time units.

8.1.2 Evaluation results

The cluster S is composed of m ($\geq$ 1) servers $s_1, \cdots, s_m$. Figures 8.1 and 8.2 show the total energy consumption [Wtu] of the servers $s_1, \cdots, s_m$ to perform n processes on servers of the cluster S in the MG, RR, and RD algorithms for $m =$ 8 and 24, respectively. As shown in Figures 8.1 and 8.2, the total electric energy consumption of the m servers is smaller in the MG algorithm than the RR and RD algorithms. The RR and RD algorithms imply almost the same electric energy consumption. For example, the total electric energy consumption of the servers in the MG algorithm is about 70$\%$ of the RR and RD algorithms for 1,400 processes ($n =$ 1,400) for eight servers ($m =$ 8) as shown in Figure 8.1. For $m =$ 8, every server is heavily loaded. For twenty four servers ($m =$ 24), since three times more number of servers are less loaded than $m =$ 8. Hence, processes can migrate to other servers so that the total energy consumption of the servers is reduced. Hence, the energy consumption of the servers in the MG algorithm is less reduced for $m =$ 8 than $m =$ 24. For example, the total electric energy consumption of the servers in the MG algorithm is about 60$\%$ of the RR and RD algorithms for $m =$ 24 as shown in Figure 8.2.

Figure 8.3 shows the average execution time of the n processes for eight servers ($m =$ 8). The average execution time of the processes is shorter in the

MG algorithm than the RR and RD algorithms. In the RR and RD algorithms, the average execution time of the processes drastically increases for more number of processes than 1,200 ($n >$ 1,200). However, the average execution time of the processes in the MG algorithm does not change even if more number of processes are performed. Because each process can migrate to a more energy-efficient server if the host server is overloaded in the MG algorithm.

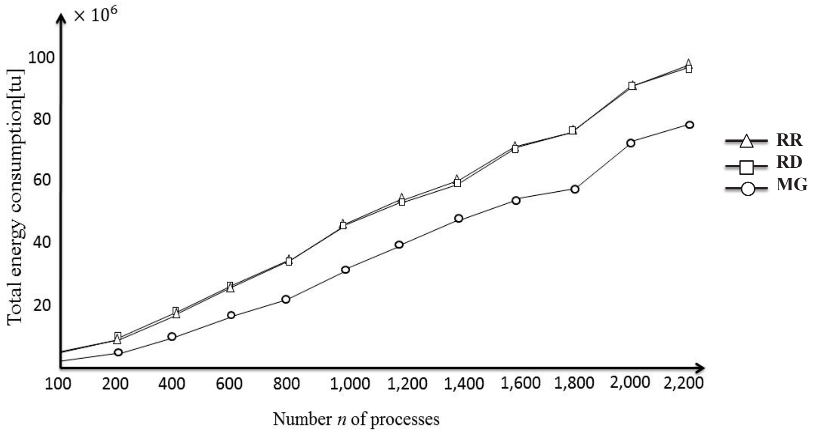

Figure 8.1: Total energy consumption of servers (m = 8).

In the MG algorithm, processes migrate form host servers to guest servers to reduce the electric energy consumption. Figure 8.4 shows the number of processes which migrates on eight servers (m = 8) in the MG algorithm. There is no process which migrates to another guest server if a fewer number n of processes are performed ($n <$ 400). For n = 400, processes migrate from host server to guest server. For example, about 20% of the processes migrate for n = 1,000 while about 75% of the processes migrate for n = 1,600. Thus, the more number of processes are performed, the more number of processes migrate so that the total electric energy consumption of the server can be reduced.

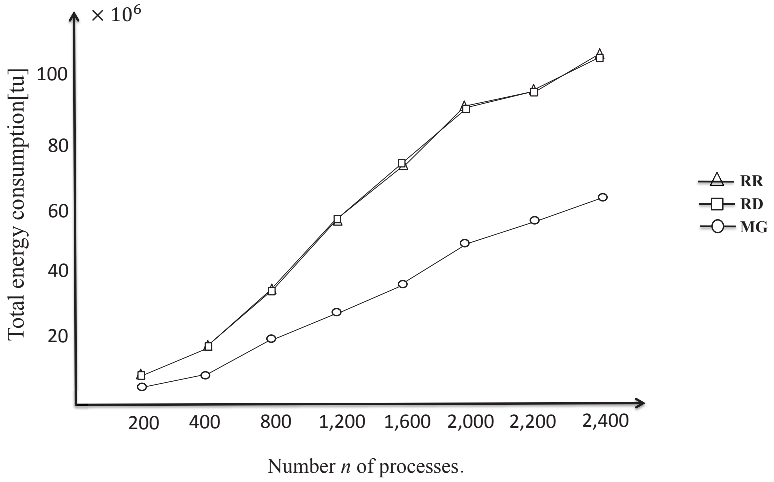

Figure 8.2: Total energy consumption of servers ($m = 24$).

Figure 8.5 shows the total electric energy consumption of m servers $s_1, \cdots, s_m$ in the cluster S to perform 1,600 processes ($n = 1{,}600$). In the RR and RD algorithms, the electric energy consumption of the servers does not change even if the number m of servers increases. In the MG algorithm, the total energy consumption of the servers decreases as the number m of servers increases. In the MG algorithm, the smaller electric energy is consumed by the servers than the RR and RD algorithms.

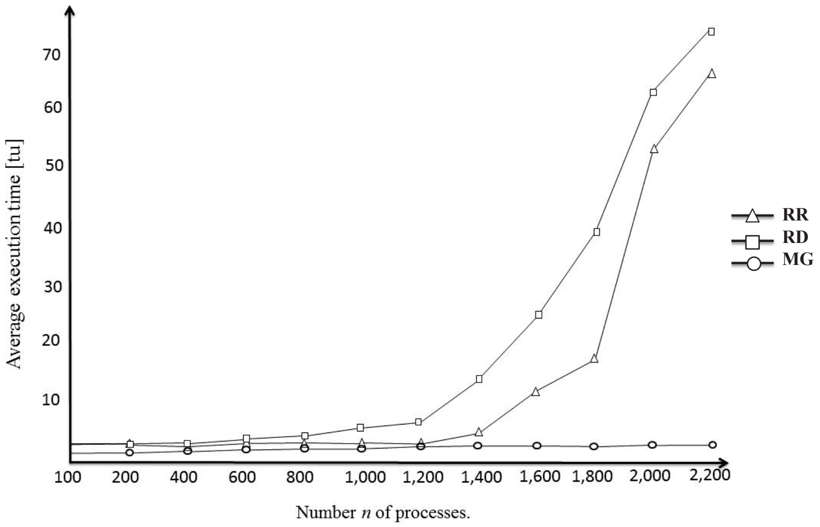

Figure 8.3: Average execution time of processes (m = 8).

8.2 Static Virtual Machine Migration

8.2.1 Environment

Next, we evaluate the migration algorithms of virtual machines. We first evaluate the static migration algorithm, EAMV (Energy-Aware Migration of Virtual Machines) algorithm in terms of the total electric energy consumption TEE [J] and total active time TAT [tu] of servers and the average execution time AET [tu] of processes compared with the non-migration random (RD), round robin (RR) algorithms. In this paper, we consider the EAMV algorithm which takes usage of the EVMS algorithm to select a virtual machine and EVMM algorithm to migrate a virtual machine from a host server to a guest server. A virtual machine is selected in the EVMS algorithm. In the RD algorithm, one virtual machine vm_h is randomly selected. In the RR algorithm, a virtual machine vm_h is selected after a virtual machine vm_{h-1} is selected. In the RD and RR algorithms, every virtual machine vm_h does not migrate and stays on one server. In the EAMV algorithm, each virtual machine vm_h migrates to a guest server if a migration condition is satisfied.

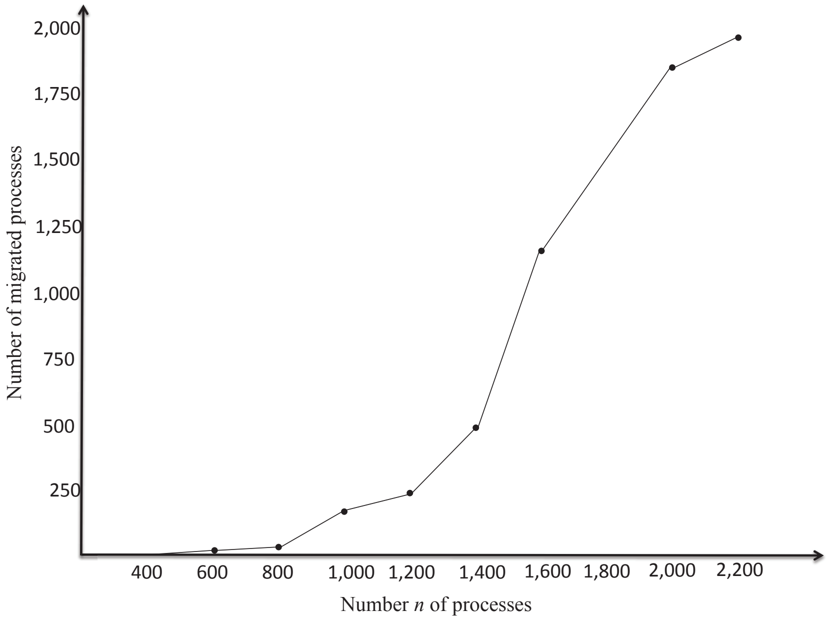

Figure 8.4: Number of migrated processes in the MG algorithm (m = 8).

There are m ($\leq$ 1) heterogeneous servers $s_1, \ldots, s_m$ in a cluster S. The electric power consumption parameters like $minE_t$, $maxE_t$, cE_t, bE_t, and tE_t [W] and the performance parameters like thread computation rate CRT_t and number nt_t of threads of each server s_t are randomly taken as shown in Table 8.2. There are a set VM of eight virtual machines $vm_1, \ldots, vm_8$ and $VM = \{vm_1, \ldots, vm_v\}$. In the evaluation, we consider six servers (m = 6) and eight virtual machines (v = 8).

The number n ($\geq$ 1) of processes $p_1, \ldots, p_n$ are randomly issued to the cluster S. In the simulation, one time unit [tu] is assumed to be 100 [msec] since the electric power of a server is measured by using the electric power meter UWmeter [46]. Each process configuration PF_{ng} includes a pair ¡p_i, $minT_i$, $stime_i$¿ where p_i starts at time $stime_i$. The minimum execution time $minT_i$ of each process p_i is randomly taken from 5 to 10 [tu], i.e. 0.5 to 1.0 [sec]. The amount VS_i [vs] of virtual computation steps of each process p_i is $minT_i$ as discussed in this paper. The start time $stime_i$ of each process p_i is randomly taken from 0 to $xtime$ - 1. The simulation time $xtime$ is 200 [tu] (= 20 [sec]). The simulation is time-based. We randomly generate four process configurations $PF_{n1}, \ldots, PF_{n4}$ of the processes $p_1, \ldots, p_n$.

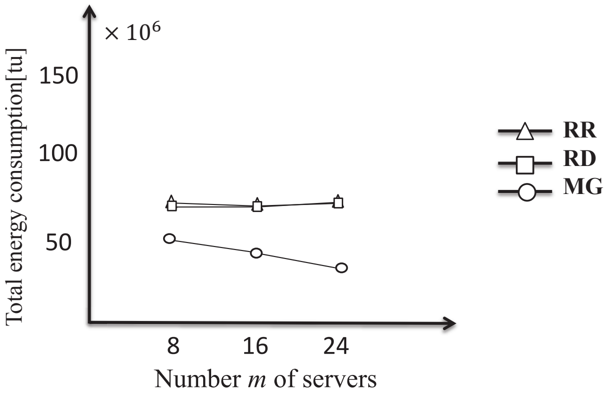

Figure 8.5: Total energy consumption (n = 1,600).

We randomly generate four server configurations SF_1, ..., SF_4 of the six servers s_1, ..., s_m (m = 6). In each server configuration SF_k, the parameters of each server s_t like minimum electric power $minE_t$, number nt_t of threads, and thread computation rate CRT_t are randomly taken.

We also generate four VM configurations VF_1, ..., VF_4 of the virtual machines vm_1, ..., vm_v (v = 8). In each VM configuration VF_l, initially each virtual machine vm_h is randomly deployed on a server s_t (l = 1, ..., 4).

For each combination of the configurations SF_k, VF_l, and PF_{ng}, the electric energy consumption EE_t and active time AT_t of each server s_t and the execution time ET_i of each process p_i are obtained in the simulation. Then, the total electric energy consumption TEE of servers $s_1, \cdots, s_m$ are calculated as $EE_1 + \ldots + EE_m$ and average execution time AET of the processes $p_1, \cdots, p_n$ as ($ET_1 + \ldots + ET_n$ / n). The average active time AT of the server s_1, ..., s_m is calculated as ($AT_1 + \ldots + AT_m$ / m).

Table 8.1: Parameters.

parameters	values
m	number of servers $s_1, \ldots, s_m$ ($\geq$ 1).
np_t	number of CPUs ($\leq$ 2).
nc_t	number of cores (1 $\sim$ 8)/ CPU .
ct_t	threads/core (= 2).
nt_t	number of threads (= 2 $\cdot$ np_t $\cdot$ nc_t).
CRT_t	0.5 $\sim$ 1 [vs/tu].
$maxCR_t$	nt_t $\cdot$ $maxC_t$ [vs/tu].
$minE_t$	80 $\sim$ 100 [W].
$maxE_t$	100 $\sim$ 200 [W].
bE_t	($maxE_t$ - $minE_t$) / (4 $\cdot$ np_t) [W]
cE_t	5 $\cdot$ ($maxE_t$ - $minE_t$) / (8 $\cdot$ np_t $\cdot$ nc_t) [W]
tE_t	($maxE_t$ - $minE_t$) / (8 $\cdot$ nt_t) [W]
n	number of processes $p_1, \ldots, p_n$ ($n \geq 1$)
$minT_i$	minimum computation time of a process p_i (5 - 10 [tu])
VS_i	5 $\sim$ 10 [vs] (VS_i = $minT_i$)
$stime_i$	starting time of p_i ($0 \leq st_i < xtime$ - 1)
$xtime$	simulation time (= 200 [tu] = 20 [sec])
v	number of virtual machines vm_1, . . ., vm_v ($v = 8$)

8.2.2 Evaluation results

Figure 8.6 shows the total electric energy $TEE = EE_1 + \ldots + EE_m$ [Wtu] of the four servers s_1, . . ., s_m ($m = 4$) for number n of processes with eight virtual machines vm_1, . . ., vm_8 ($v = 8$). As shown in Figure 8.6, the total electric energy TEE of the RD algorithm is almost the same as the RR algorithm. The total electric energy consumption TEE of the four servers s_1, . . ., s_4 in the EAMV algorithm is smaller than the other non-migration RD and RR algorithms. For example, only 40% of the electric energy of the RD and RR algorithm is consumed in the EAMV algorithm for $n \geq$ 800. In the EAMV algorithm, a virtual machine on a host server s_t migrates to another guest server s_u if the host server s_t is expected to consume more electric energy to perform processes than the guest server s_u. Hence, the total electric energy consumption TEE of the m servers s_1, . . ., s_m can be reduced in the EAMV algorithm compared with the non-migration RR and RD algorithms.

Figure 8.7 shows the total active time TAT [tu] of the four servers s_1, . . ., s_4

for the number n of processes with eight virtual machines vm_1, ..., vm_8. TAT = AT_1 + ... + AT_n. The active time of a server s_t means time when the server s_t is active, i.e. at least one process is performed on the server s_t. The total active time TAT of the RD algorithm is almost the same as the RR algorithm, because processes are uniformly allocated to the servers. The total active time TAT of the four servers (m = 4) in the EAMV algorithm is shorter than the other non-migration RR and RD algorithms. For example, the total active time TAT of the servers s_1, ..., s_4 of the EAMV algorithm is half of the RR and RD algorithms. This means, the servers are more lightly loaded in the EAMV algorithm than the other RR and RD algorithms.

Figure 8.8 shows the average execution time AET [tu] of the number n of processes p_1, ..., p_n where m = 4 and v = 8. The average execution time AET is (ET_1 + ... + ET_n) / n. The average execution time AET of the processes p_1, ..., p_n in the EAMV algorithm is shorter than the RD and RR algorithms. In the simulation, the average minimum execution time (= ($minT_1$ + ... + $minT_n$) / n) of the processes is 8.0 [tu] as shown in Table 8.2. As shown in Figure 8.8, the average execution time AET of RD algorithm is almost same as the RR algorithm. In the EAMV algorithm, the average execution time AET of the n processes is shorter than 10 [tu] for $n \leq$ 300 and 11 [tu] for n = 400. On the other hand, the average execution time AET of the n processes is about 50 and 100 [tu] in the RD and RR algorithms for n = 300 and n = 400, respectively. By migrating virtual machines to servers, the average execution time AET of the n processes can be thus reduced in the EAMV algorithm compared with the non-migration RR and RD algorithms.

8.3 Dynamic Virtual Machine Migration

8.3.1 Environment

We consider the dynamic migration algorithm, DVMM (Dynamic Virtual Machine Migration) algorithm where virtual machines are dynamically created and

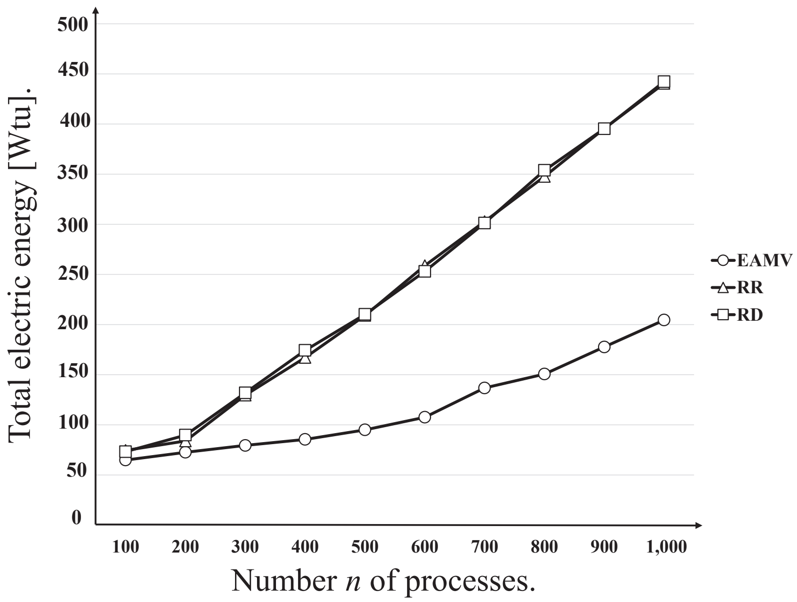

Figure 8.6: Total electric energy consumption.

dropped depending on number of processes. The DVMM (Dynamic Virtual Machine Migration) algorithm is evaluated in terms of the total electric energy consumption TEE [Wtu] and total active time TAT [tu] of servers and the average execution time AET [tu] of processes compared with the migration type EAMV (Energy-Aware Migration of Virtual Machine) [19] algorithm and a pair of the non-migration types of random (RD) and round robin (RR) algorithms.

In the RD algorithm, one virtual machine vm_h is randomly selected in a set VM of virtual machines. In the RR algorithm, a virtual machine vm_h is selected after a virtual machine vm_{h-1} is selected in the virtual machine set VM. Thus, virtual machines are serially selected. In the RD and RR algorithms, every virtual machine vm_h just stays on the host server. A virtual machine vm_h on an energy-efficient server is first selected to perform a process issued by a client. Then, virtual machines migrate to energy-efficient servers so as to reduce the total electric energy consumption of the servers in the EAMV and DVMM algorithms. In the RD, RR, and EAMV algorithms, the number v of virtual machines are invariant where there are eight virtual machines vm_1, . . ., vm_8 ($v = 8$) in a cluster S. If a process p_i is newly issued to a cluster S, one virtual machine is selected to perform the process p_i. In the EAMV and DVMM algorithms, each server s_t

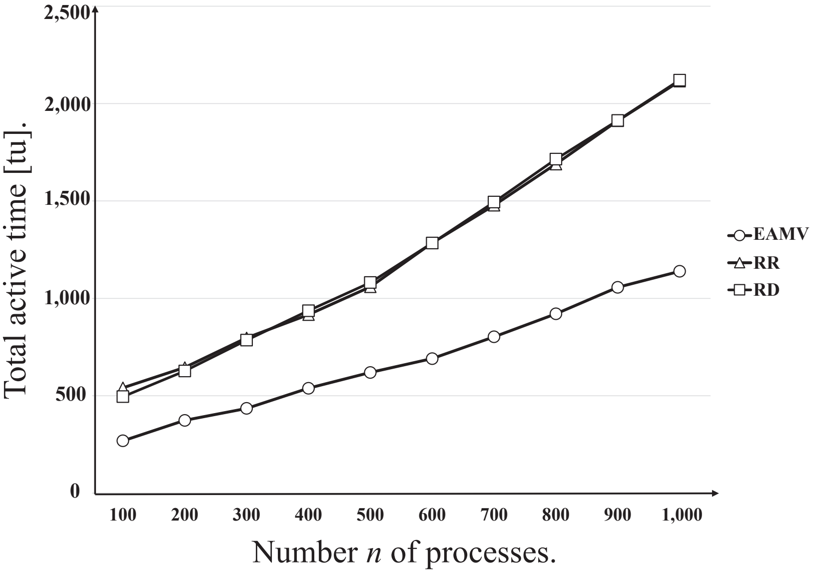

Figure 8.7: Total active time of servers.

is checked every σ time units. Then, a smallest resident virtual machine is selected and migrates to another guest server s_u if the electric energy consumption of the servers s_t and s_u can be reduced. The electric energy to be consumed by the servers is estimated by using the more sophisticated way in the EAMV algorithm than the DVMM algorithm. However, it takes a longer time to do the computation to estimate the electric energy consumption. In the DVMM algorithm, the estimation procedure is simple since just the number n_t of current processes of each server s_t is used. In the DVMM algorithm, virtual machines are dynamically created and dropped depending on the number of processes. Hence, the total number n of virtual machines is changed depending on the number of processes performed. Initially, there is no virtual machine on each server s_t, i.e. $VM_t = \phi$ and $VM = \phi$ in the cluster S.

There are four server configurations SF_1, . . ., SF_4 (m = 4). In each server configurations SF_i, the power consumption parameters like minimum electric energy consumption $minE_t$ and core energy consumption cE_t [W] and the performance parameters like thread computation rate CRT_t [vs/tu] and number nt_t of threads of each server s_t are randomly taken as shown in Table 8.2.

The number n ($\geq$ 1) of processes $p_1, \ldots, p_n$ are randomly issued to the cluster

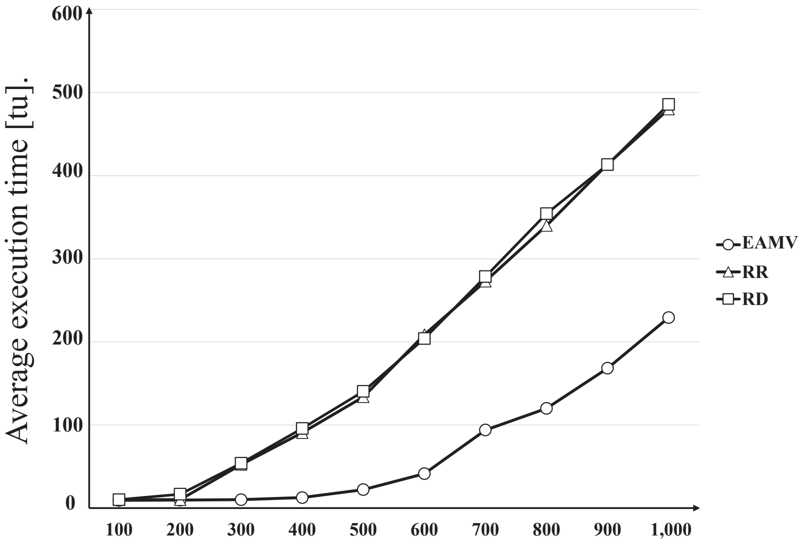

Figure 8.8: Average execution time of processes.

S. One server s_t is selected for each process p_i in the cluster S. One time unit [tu] is assumed to be 100 [msec]. In each process configuration PF_{ng}, the minimum execution time $minT_i$ of each process p_i is randomly taken from 5 to 10 [tu]. The amount VC_i [vs] of virtual computation steps of each process p_i is $minT_i$, i.e, 5 to 10. The starting time $stime_i$ of each process p_i is randomly taken from time 0 to $xtime$ - 1. The simulation time $xtime$ is 1,00 [tu] (= 10 [sec]). Thus, in each process configuration PF_{ng}, a tuple ¡p_i, $minT_i$, $stime_i$¿ is randomly taken for each process p_i. We randomly generate four process configurations $PF_{n1}, \ldots,$ PF_{n4} of the n processes $p_1, \ldots, p_n$.

In the EAMV algorithm, eight virtual machines $vm_1, \ldots, vm_8$ ($v = 8$) are randomly deployed on the four servers $s_1, \ldots, s_4$. Four virtual machine configurations $VF_1, \ldots, VF_4$ are generated in the EAMV algorithm. For each combination of the configurations SF_k, PF_{ng}, and VF_h (k, g, $h = 1, \ldots, 4$), 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.

In the DVMM algorithm, virtual machines are dynamically created and dropped. Each server is checked every five time units, i.e. $\sigma = 5$, to drop idle virtual machines on the server. The electric energy consumption EE_t and active time AT_t

of each server s_t and the execution time ET_t of each process p_i are obtained for each pair of server and process configurations SF_k and PF_{ng} (k, g = 1, . . ., 4) in the simulation.

8.3.2 Evaluation results

First, the total electric energy consumption $TEE = EE_1 + \ldots + EE_4$ [Wtu] of the servers is considered. Figure 8.9 shows the total electric energy consumption TEE for number n of processes. $maxNVM_t$ = 10 and σ = 5 in the DVMM algorithm. If the number nv_h of processes on a smallest virtual machine vm_h is larger than $maxNVM_t$, a new virtual machine is created on the server s_t. The total electric energy consumption TEE of the m (m = 4) servers s_1, . . ., s_m in the RD algorithm is the same as the RR algorithm. As shown in Figure 8.6, the total electric energy consumption TEE of the server in the DVMM algorithm is smaller than the other algorithms. For example, only 40% of the electric energy of the servers in the RD and RR algorithms and 70% of the EAMV algorithm are consumed in the DVMM algorithm. In the DVMM and EAMV algorithms, the total electric energy consumption TEE of the servers s_1, . . ., s_4 can be reduced compared with the non-migration RD and RR algorithms. In the EAMV algorithm, since the total number v of virtual machines is invariant, i.e. v = 8, the more number n of processes are issued, the more number of processes are performed on each virtual machine. This means, even if a server consumes more electric energy, no resident virtual machine of the server can migrate to another server since too many number of processes are performed on the virtual machine to migrate to another server. On the other hand, virtual machines are dynamically created and dropped in the DVMM algorithm. The more number of processes are issued, the more number of virtual machines. Hence, the servers consume smaller electric energy in the DVMM algorithm than the EAMV algorithm.

Figure 8.10 shows the toal electric energy consumption TEE of the four servers s_1, . . ., s_4 in the DVMM algorithm for $maxNVM_t$ where n (= 100, 500, 1,000) processes are performed. The larger $maxNVM_t$ gets, the smaller total electric energy TEE is consumed by the servers for $n \geq$ 100. For example, the total electric energy consumption of servers s_1, . . ., s_4 where $maxNVM_t$ is 20 is about 10% smaller than $maxNVM_t$ = 5, for n = 1,000. For n = 100, the total electric energy consumption TEE of the servers does not change even if $maxNVM_t$ changes.

Figure 8.11 shows the total active time TAT [tu] of the servers s_1, . . ., s_4 for the number n of processes. The active time AT_t of each server s_t means time

Table 8.2: Parameters.

parameters	values
m	number of servers $s_1, \ldots, s_m$ ($\geq$ 1).
np_t	number of CPUs ($\leq$ 2).
cc_t	number of cores (1 $\sim$ 8)/ CPU .
ct_t	threads/core (= 2).
nt_t	number of threads (= 2 $\cdot$ $np_t \cdot cc_t$).
CRT_t	0.5 $\sim$ 1 [vs/tu].
$maxCR_t$	$nt_t \cdot maxCR_t$ [vs/tu].
$minE_t$	80 $\sim$ 100 [W].
$maxE_t$	100 $\sim$ 200 [W].
bE_t	($maxE_t$ - $minE_t$) / (4 $\cdot$ np_t) [W].
cE_t	5 $\cdot$ ($maxE_t$ - $minE_t$) / (8 $\cdot$ $np_t \cdot nc_t$) [W].
tE_t	($maxE_t$ - $minE_t$) / (8 $\cdot$ nt_t) [W].
n	number of processes $p_1, \ldots, p_n$ ($n \geq 1$).
$minT_i$	minimum computation time of a process p_i (5 - 10 [tu]).
VS_i	5 $\sim$ 10 [vs] (VS_i = $minT_i$).
$stime_i$	starting time of p_i (0 $\leq$ st_i $<$ $xtime$ - 1).
$xtime$	simulation time (= 200 [tu] = 20 [sec]).
v	number of virtual machines vm_1, . . ., vm_v ($v = 8$).

when the server s_t is active, i.e. at least one process is performed on the server s_t. The total active time TAT of the servers is $AT_1 + \ldots + AT_4$. The total active time TAT of the servers in the RD algorithm is the same as the RR algorithm. The total active time TAT of the servers $s_1, \ldots, s_4$ in the DVMM algorithm is longer than the EAMV algorithm while TAT is shorter than the RD and RR algorithms. For example, the total active time TAT of the servers in the DVMM algorithm is about 60% of the total active time TAT of the RR and RD algorithms and is 10% longer than the EAMV algorithm. This means, the servers are more lightly loaded in the migration type DVMM and EAMV algorithms than the non-migration RR and RD algorithms.

Figure 8.12 shows the average execution time AET [tu] of the number n of processes $p_1, \ldots, p_n$. The average execution time AET of the processes $p_1, \ldots, p_n$ is $(ET_1 + \ldots + ET_n)$ / n where ET_i is the execution time of each process p_i. The average execution time AET of the n processes in the RD algorithm is the same as the RR algorithm since process are uniformly issued to the servers $s_1, \ldots, s_4$. The average execution time AET of the n processes in the DVMM algorithm is about half of the other algorithms. By dynamically creating virtual machines and migrating virtual machines to more energy-efficient servers, the average execution time AET of the n processes can be thus reduced in the DVMM algorithm compared with the non-migration RR and RD algorithms.

Figure 8.13 shows the numbers of virtual machines created and dropped and the number of migrations in the DVMM algorithm for number n of processes. The more number n of processes are issued by clients, the more number of virtual machines are created and dropped. In addition, virtual machines migrate more frequently among servers. It takes time and consumes electric energy to drop virtual machines. We have to reduce the number of virtual machines dropped.

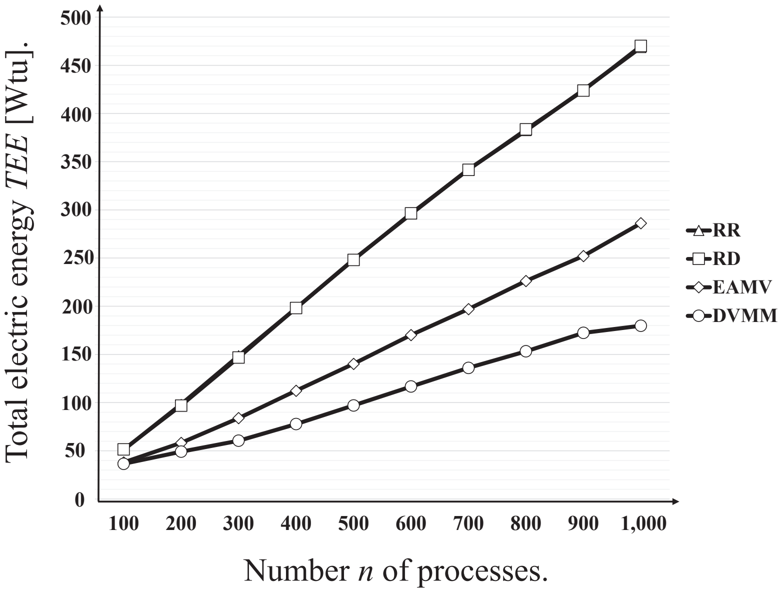

Figure 8.9: Total electric energy consumption ($m = 4$, $\sigma = 5$, $maxNVM_t = 10$).

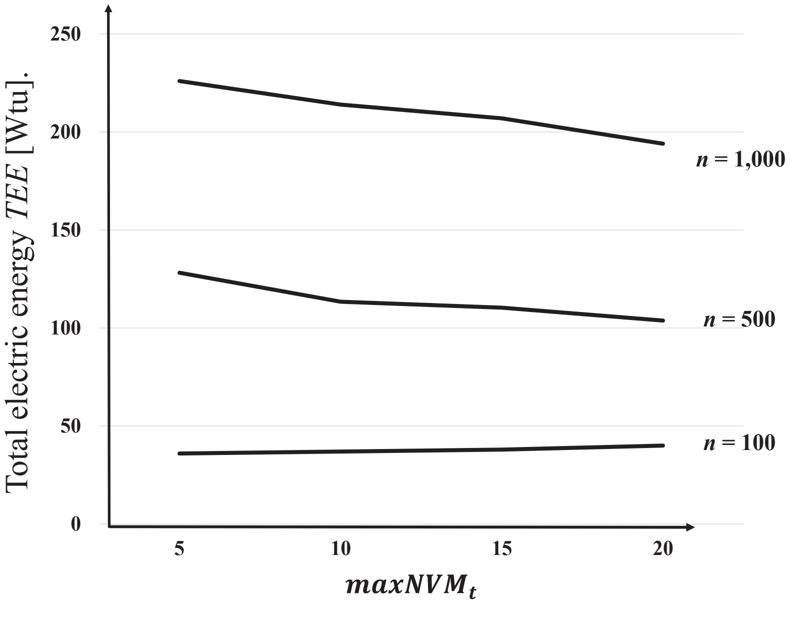

Figure 8.10: Total electric energy consumption ($m = 4$, $\sigma = 5$).

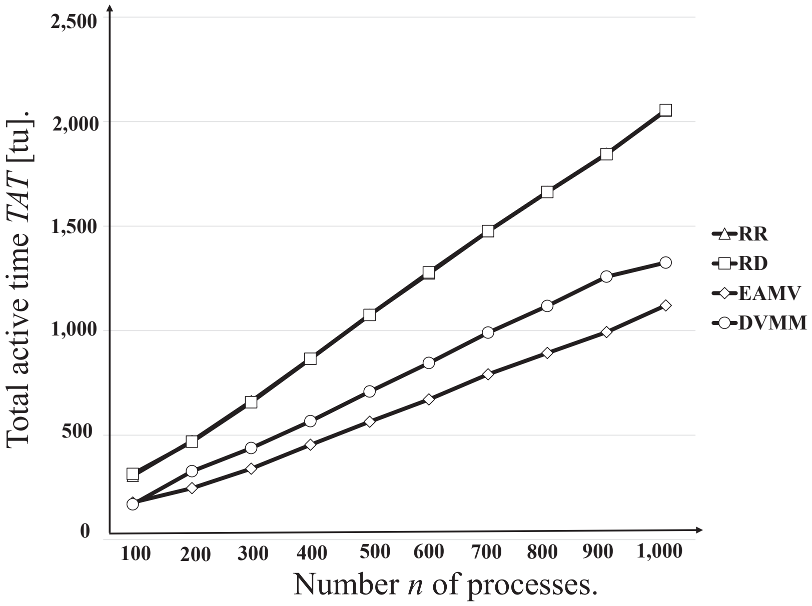

Figure 8.11: Total active time of servers ($m = 4$, $\sigma = 5$, $maxNVM_t = 10$).

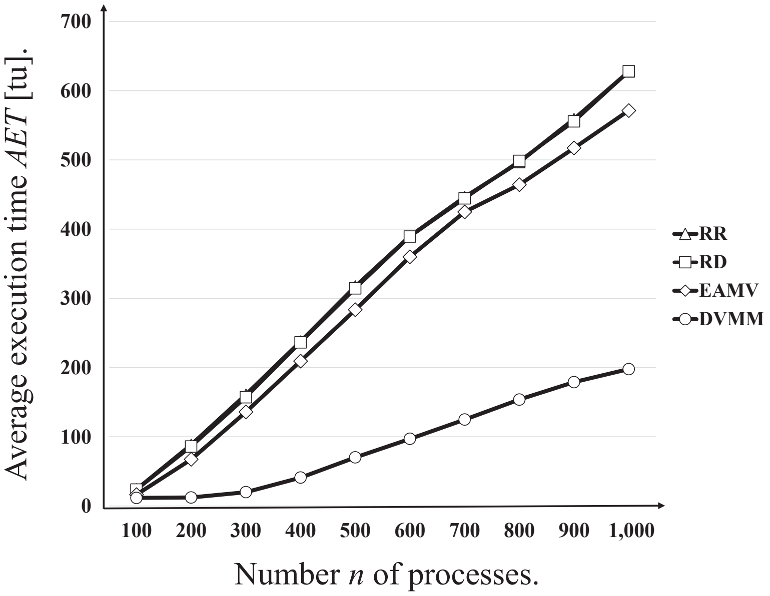

Figure 8.12: Average execution time of processes ($m = 4$, $\sigma = 5$, $maxNVM_t = 10$).

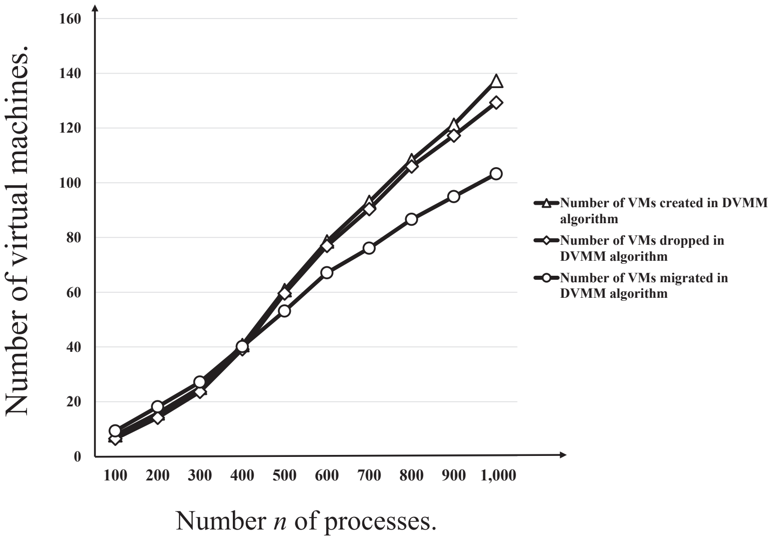

Figure 8.13: Numbers of virtual machines created and dropped ($m = 4$, $\sigma = 5$, $maxNVM_t = 10$).

Chapter 9

Conclusions and Future Studies

9.1 Conclusions

In order to realize eco society, it is critical to reduce the electric energy consumption of information systems. Especially, servers in scalable clusters like cloud computing systems consume huge electric energy compared with clients. In this thesis, we take the macro-level approach to reducing the electric energy consumption of servers. Here, we aim at reducing the total electric energy consumed by servers to perform application processes issued by applications. First, we measured the electric power [W] consumed by types of servers to perform types of application processes. Then, parameters which mostly dominate the electric power consumption of a server. Then, we newly proposed the MLPCM (MLPC model of a server with Multiple CPUs) model as a power consumption model which gives electric power to be consumed by a server to perform application processes. We made clear the power consumption of a server s_t depends on the number of active threads. If no application process is performed, a server s_t consumes the minimum electric power $minE_t$. If equal to or more number of processes than the total number nt_t of threads are performed, the server s_t consumes the maximum electric power $maxE_t$. Then, we proposed the MLCM (Multi-Level Computation model with Multiple CPUs) model as a computation model of a server which gives the expected termination time of each application process performed with other application processes. The computation rate of each process is constant, i.e. the thread computation rate if a fewer number of processes than the total number of threads are concurrently performed on the server. Otherwise, the process computation rate decrease as the number of concurrent processes increases. The

MLPCM and MLCM models give the formal basis to consider how to reduce the electric energy consumption of servers at the macro level.

Each application on a client issues a request to a cluster of servers. One server is selected and a process to handle the request is created on the server. The process is performed on the server. In this thesis, we newly proposed the process migration approach to reducing the electric energy consumption of servers in addition to selecting an energy-efficient server to perform each application process. First, we discussed how to migrate each process on a host server to another energy-efficient guest server. We proposed the MG (Process Migration) algorithm to decide which process on which server to migrate to which server, so that the total electric energy to be consumed by the servers is reduced. Based on the MLPCM and MLCM models, the expected termination time of each process is obtained by decrementing the computation laxity by the computation rate.

Secondly, we discussed virtual machine migration algorithms where we take advantage of virtual machine technologies which are used to support virtual service in clusters like cloud computing systems. It is not easy to migrate types of processes among heterogeneous servers with different architectures and operating system. Processes on a virtual machine on a host server can easily migrate to another guest server independently of heterogeneity of servers. In the virtual machine migration algorithms, a pair of static and dynamic migration algorithms of virtual machines are newly proposed. In the static migrations of virtual machines, the number v of virtual machines is invariant in a cluster independently of number of processes performed. We proposed the EAMV (Energy-Aware Migration of Virtual machines) algorithm where each virtual machine migrates from host server to another guest server. We also proposed the DVMM (Dynamic Virtual Machine Migration) algorithm to dynamically migrate virtual machines among servers. Here, virtual machines are dynamically created and dropped depending on number of processes performed on the servers. The more number of processes are performed, the more number of virtual machines are created. In the EAMV algorithm, it takes time to estimate the expected termination time of each process since the computation of each current process has to be simulated by decrementing the computation laxity of each process by the process computation rate. In order to make the estimation simpler, we proposed the simple estimation model where only number n_t of processes on each server s_t and the number nv_h on a virtual machine vm_h to migrate. by using the simple estimation model, a virtual machine on a host server and a guest server to which the virtual machine migrates are selected so that the total electric energy to be consumed by the host and guest servers can be reduced.

Lastly, we evaluated the MG, EAMV, and DVMM algorithms which we proposed in this thesis, in terms of the total electric energy consumption of servers, the active time of servers, and the average execution time of processes in the simulation. In order to do the simulation, we developed the time-based simulator by which the electric energy consumption and active time of each server and the execution time of each process are obtained. The simulator is implemented by taking advantage of a relational database and SQL. In the evaluation, the total electric energy of servers can be mostly reduced in the dynamic migration algorithm DVMM compared with non-migration algorithms RR (Round-Robin), RD (Random), SGEA (Simple Globally Energy-Aware), and static migration algorithm EAMV.

In this thesis, we newly proposed the power consumption model of a server with multi-thread CPUs to perform application processes. Then, we newly discussed the migration approach to reducing the electric energy consumption of servers in a cluster. We proposed novel algorithms to select energy-efficient servers to perform an application process and migrate processes to more energy-efficient servers by taking advantage of virtual machine technologies. The models and algorithms which we newly discussed and proposed in this thesis. They are the theoretical foundations to design, implement, and evaluate energy-efficient information systems.

9.2 Future Studies

In this thesis, we proposed migration types of the MG, EAMV, and DVMM algorithms to reduce the electric energy consumption of servers in clusters and evaluated the algorithms in terms of total electric energy consumption of servers and average execution time of processes in the simulation. We would like to implement and evaluate the algorithms, which we proposed, in real server clusters, especially scalable clusters.

Information systems are composed of various types of nodes like sensors [2] and actors like robots in addition to servers and clients as discussed in IoT (Internet of Things) [45]. In the IoT system, there are fog nodes between devices and clouds of servers. Data and computation are stored and used in a cloud. In the IoT system, data and computation are distributed to not only servers in a cloud but also fog nodes. We would like to make a power consumption model of fog nodes and IoT devices in the IoT system.

Bibliography

[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*), 2009, pp.424–431.

[2] L. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E Cayirci, 'A Survey on Sensor Networks,' *IEEE Communication Magazines*, 2002, Vol. 40, No. 8, pp.102–114.

[3] AMD Ryzen. (2017) *https://www.amd.com/en/ryzen*.

[4] Processors Arm. (2017) *https://www.arm.com/products/processors*

[5] L. Barolli and F. Xhafa. 'JXTA-OVERLAY: A P2P Platform for Distributed, Collaborative and Ubiquitous Computing,' *IEEE Transactions on Industrial Electronics*, Vol. 58, No. 6, 2011, pp. 2163-2172.

[6] R. Bianchini and R. Rajamony. 'Power and Energy Management for Server Systems,' *IEEE Computer*, Vol.37, No.11, 2004 pp.68-74.

[7] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair. 'Distributed Systems Concepts and Design (4th Edition),' Addison Wesley, 2012, 983 pages.

[8] Ray J. Rafaels. 'Cloud Computing: From Beginning to End,' Create Space Independent Publishing Platform. 2015, 152 pages.

[9] DSLab URL. *http://www.http://takilab.org/*.

[10] 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.

[11] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa. 'Power Consumption Models for Redundantly Performing Mobile-Agents,' *Proc. of the 8th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2014*), 2014 pp.185-190.

[12] 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.155–162.

[13] D. Duolikun, T. Enokido, and M. Takizawa. 'Asynchronous Migration of Process Replicas in a Cluster,' *Proc. of IEEE the 29th International Conference on Advanced Information Networking and Applications* (*AINA-2015*), 2015, pp.271–278.

[14] D. Duolikun, T. Enokido, and M. Takizawa. 'Energy-Efficient Replication and Migration of Processes in a Cluster,' *Proc. of the 9th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2015*), 2015, pp.118–125.

[15] D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa. 'Energy-Efficient Dynamic Clusters of Servers,' *Journal of Supercomputing*, Vol.71, No.5, 2015, pp.1642–1656.

[16] D. Duolikun, S. Nakamura, T. Enokido, and M. Takizawa. 'An Energy-efficient Process Migration Approach to Reducing Electric Energy Consumption in a Cluster of Servers,' *International Journal of Communication Networks and Distributed Systems*, Vol.15, No.4, 2015, pp.400–420.

[17] D. Duolikun, R Watanabe, T. Enokido, and M. Takizawa. 'A Model for Migration of Virtual Machines to Reduce Electric Energy Consumption,' *Proc. of the 8th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2016*), 2016 pp.160-166.

[18] D. Duolikun, R Watanabe, T. Enokido, and M. Takizawa. 'Energy-aware Migration of Virtual Machines in a Cluster,' *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2016*), 2016, 283-287.

[19] D. Duolikun, R Watanabe, T. Enokido, and M. Takizawa. 'A Model for Energy-aware Migration of Virtual Machines,' *Proc. of the 19th International Conference on Network-Based Information Systems* (*NBiS-2016*), 2016, pp.50-57,

[20] D. Duolikun, T. Enokido, and M. Takizawa. 'An Energy-aware Algorithm to Migrate Virtual Machines in a Server Cluster,' *International Journal of Space-Based and Situated Computing* , Vol.7, No.1, 2017, pp.32–42.

[21] 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*), 2010, pp.201–208.

[22] T. Enokido, A. Aikebaier, and M. Takizawa. 'A Model for Reducing Power Consumption in Peer-to-Peer Systems,' *IEEE Systems Journal*, Vol.4, No.2, 2010, pp.221–229.

[23] 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.

[24] 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*), 2012, pp.912–919.

[25] T. Enokido and M. Takizawa. 'An Integrated Power Consumption Model for Distributed Systems,' *IEEE Transactions on Industrial Electronics*, Vol.60, No.2, 2013, pp.824–836.

[26] 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*), 2013, pp.91-98.

[27] T. Enokido, A. Aikebaier, and M. Takizawa. 'An Extended Simple Power Consumption Model for Selecting a Server to Perform Computation Type Processes in Digital Ecosystems,' *IEEE Transactions on Industrial Informatics*, Vol.10, No.2, 2014, pp.1627–1636.

[28] W. R. Stevens. 'TCP/IP illustrate,' Addison Wesley, 1994, 1017 pages.

[29] E. Elnozahy, M. Kistler, R. Rajamony. 'Energy-Efficient Server Clusters,' *Power-Aware Computer Systems*, Vol. 2325, 2003, pp.179-197.

[30] S. Ghemawat, H. Gobioff, and S. T. Leung. 'The Google File System,' *Proc. of ACM the 19th Symposium on Operating System Principle* (*SOPI 03*), 2003, pp.29–43.

[31] HP DL-360p server. *https://www.hpe.com/h20195/v2/GetPDF.aspx/c04123167.pdf*, 2015.

[32] T. Inoue, A. Aikebaier, T. Enokido, and M. Takizawa. 'Evaluation of Energy-aware Server Selection Algorithm,' *Journal of Mathematical and Computer Modeling*, Vol.58, No.5&6, 2013, pp.1475–1488.

[33] 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.

[34] Kyoto protocol. *http://unfccc.int/kyoto protocol/items/2830.php*, 1997.

[35] A. Kipp, T. Jiang, J. Liu, M. Fugini, M. Vitali, B. Pernici, and I. Salomie. 'Applying green metrics to optimise the energy consumption footprint of IT service centres,' *International Journal of Space-Based and Situated Computing* , Vol.2, No.3, 2012, pp.158–174.

[36] 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* (*AINA-2015*), 2015, pp. 217–222.

[37] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa. 'Evaluation of Energy-Aware Server Selection Algorithms,' *Proc. of the 9th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2015*), 2015, pp.318–325.

[38] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa. 'Multi-level Computation and Power Consumption Models,' *Proc. of the 18th International Conference on Network-based Information Systems* (*NBiS-2015*), 2015, pp.40–47.

[39] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa. 'Energy-efficient Virtualisation of Threads in a Server Cluster,' *Proc. of the 10th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2015*), 2015, pp.288-295.

[40] H. Kataoka, A. Sawada, 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.

[41] H. Kataoka, A. Sawada, D. Duolikun, T. Enokido, and M. Takizawa. 'Energy-efficient Virtualisation of Threads in a Server Cluster,' *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2016*), 2016, pp.573–584.

[42] H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa. 'Simple Energy-aware Algorithms for Selecting a Server in a Scalable Cluster,' *Proc of AINA-2017 workshops* (*WAINA-2017*), 2017, pp.146-153.

[43] E. Ogawa, S. Nakamura, T. Enokido, and M. Takizawa. 'An Energy-saving Unicast Routing Protocol in Wireless Ad-hoc Network,' *Proc. of the 11th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2017*), 2017, CD-ROM.

[44] J. D. la Rosa. 'KVM Virtualization in RHEL 6 Made Easy,' *Dell Linux Engineering*, 2011.

[45] A. McEwen and H. Cassimally, 'Designing the Internet of Things,' Wiley, 2013 338 pages.

[46] Meta Protocol Corp: "UWmeter", *http://www.metaprotocol.com/UWmeter/Feautures.html*, 2011.

[47] T. Mori, M. Nakashima, and T. Ito. 'SpACCE: a Sophisticated Ad hoc Cloud Computing Environment Built by Server Migration to Facilitate Distributed Collaboration,' *International Journal of Space-Based and Situated Computing*, Vol.2, No.4, 2012, pp.230–239.

[48] C. Negus and T. Boronczyk. 'CentOS Bible,', 2009.

[49] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa. 'Energy-aware Server Selection Algorithms for Storage and Computation Processes,' *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2016*), 2016, pp.45–56.

[50] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa. 'Design and Evaluation of Algorithms to Energy-efficiently Select Servers for Storage and Computation Processes,' *Proc. of the 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2016)*, 2016, pp.162–169, .

[51] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa. 'Selection Algorithms to Select Energy-efficient Servers for Storage and Computation Processes,' *Proc. of the 19th International Conference on Network-Based Information Systems* (*NBiS-2016*), 2016, pp.218-225.

[52] A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, and M. Takizawa. 'Algorithm to Energy-efficiently Select a Server for a General Process in a Scalable Cluster,' *Proc of AINA-2017 Workshops* (*WAINA-2017*), 2017, pp.138-145.

[53] SSD. *http://searchsolidstatestorage.techtarget.com/definition/SSD-solid-state-drive*, 2014.

[54] M. Sugino, S. Nakamura, T. Enokido, and M. Takizawa. 'Trustworthiness-based Broadcast Protocols in Wireless Networks,' *Proc. of the 18th International Conference on Network-Based Information Systems* (ıtNBiS-2015), 2015, pp.125–132 .

[55] R. Watanabe, D. Duolikun, T. Enokido, and M. Takizawa. 'An Eco Model of Process Migration with Virtual Machines in Clusters,' *Proc. of the 19th International Conference on Network-based Information Systems* (*NBiS-2016*), 2016, pp.292-297.

[56] R. Watanabe, D. Duolikun, T. Enokido, and M. Takizawa. 'An Eenergy-efficient Migration Model of Process with Virtual Machines in a Server Clusters,' *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Apprications* (*BWCCA-2016*), 2016, pp.33-44.

[57] R. Watanabe, D. Duolikun, T. Enokido, and M. Takizawa. 'Energy-aware Virtual Machine Migration Models in a Scalable Cluster of Servers,' *Proc. of IEEE the 31st International Conference on Advanced Information Networking and Applications* (*AINA-2017*), 2017, pp.85-92.

[58] 2015 United Nations Climate Change Conference (COP21), 2015.

[59] 2017 United Nations Climate Change Conference (COP23), 2017.

[60] Linux. 'Job Scheduling Algorithms in Linux Virtual Server,', 2013.

[61] Q. Xilong and X. Peng. 'An Energy-efficient Virtual Machine Scheduler Based on CPU Share-reclaiming Policy,' *International Journal of Grid and Utility Computing*, Vol.6, No.2, 2015, pp. 113–120.

List of Publications

1. Refereed Journal papers

1. D. Duolikun, A. Aikebaier, T. Enokido, M. Takizawa: “Energy-aware Passive Replication of Processes,” *International Journal of Mobile Multimedia*, Vol. 9, No. 1&2, 2013, pp.53-65, DOI: 10.1145/2536853.2536865.
2. D. Duolikun, A. Aikebaier, M. Takizawa: “A Hybrid Clock Group Communication Protocol,” *International Journal of Adaptive and Innovative Systems*, Vol.2, No.1, 2014, pp.59-72, DOI: 10.1504/IJAIS.2014.062048.
3. D. Duolikun, A. Aikebaier, T. Enokido, M. Takizawa: “Design and Evaluation of a Quorum-based Synchronization of Multimedia Replicas,” (*IJAHUC*), Vol.17, No.2&3, 2014, pp.100-109, DOI: 10.1504/IJAHUC.2014.065773.
4. D. Duolikun, T. Enokido, M. Takizawa: “Energy-efficient Dynamic Cluster of Servers,” *International Journal of Supercomputing*, Vol. 71, No.5, 2015, pp.1642-1656, DOI: 10.1007/s11227-014-1261-3.
5. D. Duolikun, S. Nakamura, T. Enokido, M. Takizawa: "An Energy-efficient Process Migration Approach to Reducing Electric Energy Consumption in a Cluster of Servers," *International Journal of Communication Networks and Distributed Systems*, Vol.15, No.4, 2015, pp.400-420, DOI: 10.1504/IJCNDS.2015.072404.
6. T. Enokido, D. Duolikun, M. Takizawa: “An Extended Improved Redundant Power Consumption Laxity-Based (EIRPCLB) Algorithm for Energy Efficient Server Cluster Systems,” *World Wide Web Journal*, Vol.18, No.6, 2015, pp.1603-1629, DOI: 10.1007/s11280-014-0315-z.
7. T. Enokido, D. Duolikun, M. Takizawa: “The Delay Time-based Server Selection Algorithm for Energy-efficient Redundant Execution of Processes,” *International Journal of Communication Networks and Distributed Systems*, Vol.15, No.4, 2015, pp. 366-385, DOI: 10.1504/IJCNDS.2015.072401.
8. R. Watanabe, D. Duolikun, T. Enokido, and M. Takizawa: “A Simply Energy-efficient Migration Algorithm of Processes with Virtual Machines in Server Clusters,” *International Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*(*JoWUA*), Vol. 8, No.2, pp. 1-18, 2017, DOI:10.22670/JOWUA.2017.06.31.001.
9. D. Duolikun, T. Enokido, M. Takizawa: “An Energy-aware Algorithm to Migrate Virtual Machines in a Server Cluster,” *International Journal of Space-Based and Situated Computing* (*IJSSC*), Vol. 7, No, 1, 2017, DOI: 10.1504/IJSSC.2017.10004986.
10. H. Kataoka, S. Nakamura, D. Duolikun,T. Enokido, M. Takizawa: “Multi-level Power Consumption Model and Energy-aware Server Selection Algorithm,” *International Journal of Grid and Utility Computing* (*IJGUC*), Vol.8, No. 3, 2017, pp.201-210, DOI: 10.1007/S12652-017-0541-1.
11. D. Duolikun, T. Enokido, M. Takizawa: “Dynamic Migration of Virtual Machines to Reduce Energy Consumption in a Cluster,” International Journal of Grid and Utility Computing (IJGUC), 2017, (Accepted).
12. D. Duolikun, H. Kataoka, T. Enokido, 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).
13. T. Enokido, D. Duolikun, M. Takizawa: “An Energy-Aware Load Balancing Algorithm to Perform

Computation Type Application Processes in a Cluster of Servers," *International Journal of Web and Grid Services (IJWGS*), 2017, (Accepted).

2. Refereed International Conference Papers

1. D. Duolikun, A. Aikebaier, T. Enokido, M. Takizawa: "Ordered Delivery of Messages in Group Communication Protocols," *Proc. of the 15th International Conference on Network-Based Information Systems* (*NBiS-2012*), Melbourne, Australia, September, 2012, pp.397-401.
2. D. Duolikun, M. Takizawa: "Communication Protocols in Layered Groups with Heterogeneous Clocks," *Proc. of the 7th International Conference on Broadband, Wireless Computing, Communication and Applications* (*BWCCA-2012*), Victoria, Canada, November, 2012, pp.568-572.
3. D. Duolikun, H. Hama, A. Aikebaier, T. Enokido, M. Takizawa: "Group Communication Protocols for Scalable Groups of Peers," *Proc. of IEEE the 27th International Conference on Advanced Information Networking and Applications* (*AINA-2013*), Barcelona, Spain, March, 2013, pp. 1027-1032.
4. D. Duolikun, A. Aikebaier, T. Enokido, M. Takizawa: "A Scalable Group Communication Protocol on P2P Overlay Networks," *Proc. of the 7th International Conference on Complex, Intelligent, and Software Intensive Systems* (*CISIS-2013*), Taichung, Taiwan, July, 2013, pp.428-433.
5. D. Duolikun, A. Aikebaier, T. Enokido, L. Barolli, M. Takizawa: "Energy-efficient Passive Replication of a Process in Mobile Environment," *Proc. of International Conference on Advances in Mobile Computing & Multimedia* (*MoMM-2013*), Vienna, Austria, December, 2013, pp.416-424.
6. D. Duolikun, A. Aikebaier, T. Enokido, M. Takizawa: "Experimentation of Group Communication Protocols," *Proc. of the 16th International Conference on Network-Based Information Systems* (*NBiS-2013*), Gwangju, Korea, September, 2013, pp.476-481.
7. D. Doulikun, A. Aikebaier, T. Enokido, M. Takizawa: "Energy-Efficient Dynamic Clusters of Servers," *Proc. of the 8th International Conference on Broadband, Wireless Computing, Communication and Applications* (*BWCCA-2013*), Compiegne, France, October, 2013, pp 253-260.
8. D. Doulikun, A. Aikebaier, T. Enokido, M. Takizawa: "Group Communication Protocols Based on Hybrid Types of Logical and Physical Clocks," *Proc. of the 8th International Conference on Broadband, Wireless Computing, Communication and Applications* (*BWCCA-2013*), Compiegne, France, October, 2013, pp 494-499.
9. D. Duolikun, A. Aikebaier, T. Enokido, M. Takizawa: "Energy-Aware Replication Models of Mobile Agents," *Proc. of IEEE the 28th International Conference on Advanced Information Networking and Applications* (*AINA-2014*), Victoria, Canada, May, 2014, pp.1132-1139.
10. D. Duolikun, A. Aikebaier, T. Enokido, and M. Takizawa: "Power Consumption Models for Redundantly Performing Mobile-Agents," *Proc. of the 8th International Conference on Complex, Intelligent, and Software Intensive Systems* (*CISIS-2014*), Birmingham, UK, July 2014, pp.185-190 **(Best paper award)**.
11. D. Duolikun, A. Aikebaier, T. Enokido, 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*), Salerno, Italy, September, 2014, pp.15-22.
12. D. Duolikun, T. Enokido, M. Takizawa: "A Process Migration Approach to Energy-efficient Computation in a Cluster of Servers," *Proc. of the 9th International Conference on Broadband and Wireless Computing,*

Communication and Applications (*BWCCA-2014*), Guangzhou, China, November, 2014, pp. 191-198.

13. D. Duolikun, T. Enokido, M. Takizawa: “Asynchronous Migration of Process Replicas in a Cluster,” *Proc. of IEEE the 29th International Conference on Advanced Information Networking and Applications* (*AINA-2015*), Gwangju, Korea, March, 2015, pp.271-278.
14. D. Duolikun, T. Enokido, M. Takizawa: "Energy-Efficient Replication and Migration of Processes in a Cluster,” *Proc. of the 9th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2015*), Blumenau, Brazil, July 2015, pp.118–125.
15. D. Duolikun, H. Kataoka, S. Nakamura, T. Enokido, Makoto Takizawa: “Energy-Aware Migration and Replication of Processes in a Cluster,” *Proc. of the 9th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2015*), Krakow, Poland, November, 2015, pp.283-287.
16. D. Duolikun, R. Watanabe, H. Kataoka, S. Nakamura, T. Enokido, M. Takizawa: “An Energy-aware Migration of Virtual Machines,” *Proc. of IEEE the 30th International Conference on Advanced Information Networking and Applications* (*AINA-2016*), Crans-Montana, Switzerland, March 2016, pp.557-564.
17. D. Duolikun, R. Watanabe, T. Enokido, and M. Takizawa: "A Model for Migration of Virtual Machines to Reduce Electric Energy Consumption,” *Proc. of the 10th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2016*), Fukuoka, Japan, July 2016, pp.159-166.
18. D. Duolikun, R. Watanabe, T. Enokido, and M. Takizawa: “A Model for Energy-Aware Migration of Virtual Machines,” *Proc. of the 19th International Conference on Network-based Information Systems* (*NBiS-2016*), Ostrava, Czech, September 2016, pp.50-57 **(Best paper award)**.
19. D. Duolikun, S. Nakamura, R. Watanabe, T. Enokido, M. Takizawa: “Energy-aware Migration of Virtual Machines in a Cluster,” *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2016*), Asan, Korea, November 2016, pp.283-287.
20. D. Duolikun, R. Watanabe, T. Enokido, M. Takizawa: “An Eco Migration of Virtual Machines in a Server Cluster,” *Proc. of IEEE the 31st International Conference on Advanced Information Networking and Applications* (*AINA-2017*), Taipei, Taiwan, March 2017, pp.1098-1105.
21. D. Duolikun, T. Enokido, and M. Takizawa: “Energy-aware Dynamic Migration of Virtual Machines in a Server Cluster,” *Proc. of the 11th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2017*), Torino, Italy, July 2017, pp.70-81.
22. D. Duolikun, R. Watanabe, T. Enokido, M. Takizawa: “An Eco Algorithm for Dynamic Migration of Virtual Machines in a Server Cluster,” *Proc. of the 20th International Conference on Network-Based Information Systems* (*NBiS-2017*), Toronto, Canada, August 2017, pp. 42-54.
23. D. Duolikun, R. Watanabe, T. Enokido, M. Takizawa: “Energy-aware Dynamic Migration of Virtual Machines in a Server Cluster,” *Proc. of the 12th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2017*), Barcelona, Spain, November 2017, pp.161-172.
24. D. Duolikun, R. Watanabe, T. Enokido, M. Takizawa: “An Eco Migration of Virtual Machines in a Server Cluster,” *Proc. of IEEE the 32nd International Conference on Advanced Information Networking and Applications* (*AINA-2018*), Cracow, Poland, May 2018, (Accepted).
25. H. Nakayama, D. Duolikun, A. Aikebaier, T. Enokido, M. Takizawa: “Causal Order of Application Events is P2P Publish/Subscribe Systems,” *Proc. of the 9th International Conference on Broadband and Wireless*

Computing, Communication and Applications (*BWCCA-2014*), Guangzhou, China, November, 2014, pp.444-449.

26. H. Nakayama, D. Duolikun, A. Aikebaier, T. Enokido, M. Takizawa: "Situation-Aware Group Communication Protocols," *Proc. of the 8th International Conference on Complex, Intelligent, and Software Intensive Systems* (*CISIS-2014*), Birmingham, UK, July, 2014, pp.415-420.
27. H. Nakayama, D. Duolikun, T. Enokido, M. Takizawa: "A P2P Model of publish/Subscribe Systems," P*roc. of the 9th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2014*), Guangzhou, China, November, 2014, pp.383-388.
28. H. Nakayama, D. Duolikun, T. Enokido, M. Takizawa: "Selective Delivery of Event Messages in Peer-to-Peer Topic-Based Publish/Subscribe Systems," *Proc. of the 18th International Conference on Network-based Information Systems* (*NBiS-2015*), Taipei, Taiwan, September, 2015, pp.379-386.
29. H. Honda, S. Nakamura, D. Doulikun, T. Enokido, M. Takizawa: "Reduction of Unnecessarily Ordered Messages in Scalable Group Communication," *Proc. of the 18th International Conference on Network-based Information Systems* (*NBiS-2015*), Taipei, Taiwan, September, 2015, pp.99-106.
30. H. Kataoka, D. Duolikun, T. Enokido, M. Takizawa: "Multi-level Computation and Power Consumption Models," *Proc. of the 18th International Conference on Network-based Information Systems* (*NBiS-2015*), Taipei, Taiwan, September, 2015, pp.40-47.
31. H. Nakayama, D. Duolikun, T. Enokido, M. Takizawa: "Synchronization of Peers in Peer-to-Peer Publish/Subscribe Systems," *Proc. of the 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (*IMIS-2015*), Blumenau, Brazil, July, 2015, pp.252-259.
32. H. Kataoka, D. Duolikun, T. Enokido, M. Takizawa: "Evaluation of Energy-Aware Server Selection Algorithms," *Proc. of the 9th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2015*), Blumenau, Brazil, July, 2015, pp.318-325.
33. A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, M. Takizawa: "Power Consumption and Computation Models of a Storage Server," *Proc. of the 9th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2015*), Krakow, Poland, November, 2015, pp.472-477.
34. H. Kataoka, D. Duolikun, T. Enokido, M. Takizawa: "Energy-Efficient Virtualization of Threads in a Server Cluster," *Proc. of the 9th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2015*), Krakow, Poland, November, 2015, pp.288-295.
35. H. Nakayama, D. Duolikun, T. Enokido, M. Takizawa: "Causally Ordered Delivery of Event Messages with Keyword Vectors in P2P Publish/Subscribe Systems," *Proc. of IEEE the 28th International Conference on Advanced Information Networking and Applications* (*AINA-2015*), Gwangju, Korea, March, 2015, pp.534-541.
36. K. Kouno, D. Duolikun, T. Enokido, M. Takizawa: "Broadcast Protocols in Wireless Networks," *Proc. of IEEE the 28th International Conference on Advanced Information Networking and Applications* (*AINA-2015*), Gwangju, Korea, March, 2015, pp.272-277.
37. A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, 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*), Crans-Montana, Switzerland, March, 2016, pp.400-407

38. H. Kataoka, A. Sawada, D. Duolikun, T. Enokido, 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*), Crans-Montana, Switzerland, March, 2016, pp.565-572.
39. H. Nakayama, D. Duolikun, T. Enokido, M. Takizawa: “Reduction of Unnecessarily Ordered Event Messages in Peer-to-Peer Model of Topic-Based Publish/Subscribe Systems,” *Proc. of IEEE the 30th International Conference on Advanced Information Networking and Applications* (*AINA-2016*), Crans-Montana, Switzerland, March, 2016, pp.1160-1167.
40. H. Kataoka, A. Sawada, D. Duolikun, T. Enokido, M. Takizawa: “Energy-Aware Algorithms to Select Servers in Scalable Clusters,” *Proc. of the 10th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2016*), Fukuoka, Japan, July, 2016, pp.308-315.
41. A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, M. Takizawa: “Design and Evaluation of Algorithms to Energy-Efficiently Select Servers for Storage and Computation Processes,” *Proc. of the 10th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2016*), Fukuoka, Japan, July, 2016, pp.162-169.
42. H. Kataoka, A. Sawada, D. Duolikun, T. Enokido, M. Takizawa: “Multi-level Power Consumption and Computation Models and Energy-Efficient Server Selection Algorithms in a Scalable Cluster,” *Proc. of the 19th International Conference on Network-based Information Systems* (*NBiS-2016*), Ostrava, Czech, September, 2016, pp.210-217.
43. A. Sawada, H. Kataoka, D. Duolikun, T. Enokido, M. Takizawa: “Selection Algorithms to Select Energy-Efficient Servers for Storage and Computation Processes,” *Proc. of the 19th International Conference on Network-based Information Systems (NBiS-2016*), Ostrava, Czech, September, 2016, pp.218-225.
44. R. Watanabe, D. Duolikun, T. Enokido, M. Takizawa: “An Eco Model of Process Migration with Virtual Machines,” *Proc. of the 19th International Conference on Network-based Information Systems* (*NBiS-2016*), Ostrava, Czech, September, 2016, pp.292-297.
45. A. Sawada, H. Kataoka D. Duolikun, T. Enokido, M. Takizawa, “Energy-aware Server Selection Algorithms for Storage and Computation Processes,” *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2016*), Asan, Korea, November 2016, pp.45-56.
46. R. Watanabe, D. Duolikum, T. Enokido, M. Takizawa, “An Energy-efficient Migration Model of Process with Virtual Machines in a Server Clusters,” *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2016*), Asan, Korea, November 2016, pp.33--44.
47. R. Watanabe, D. Duolikun, T. Enokido, M. Takizawa: “Energy-aware Virtual Machine Migration Models in a Scalable Cluster of Servers,” *Proc. of IEEE the 31st International Conference on Advanced Information Networking and Applications* (AINA-2017), Taipei, Taiwan, March 2017, pp.85-92.
48. T. Enokido, D. Duolikun, M. Takizawa: “Energy-Efficient Quorum Selection Algorithm for Distributed Object-Based Systems,” *Proc. of the 11th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2017*), Torino, Italy, July 2017, 31-42.
49. A. Sawada, D. Duolikun, T. Enokido, 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 Complex, Intelligent and Software Intensive Systems* (*CISIS-2017*), Torino, Italy, July 2017, pp.98-109.

50. R. Watanabe, D. Duolikun, T. Enokido, M. Takizawa: "An Energy-efficient Migration Algorithm of Virtual Machines in Server Clusters," *Proc. of the 11th International Conference on Complex, Intelligent and Software Intensive Systems* (*CISIS-2017*), Torino, Italy, July 2017, pp.94-105.
51. R. Watanabe, D. Duolikun, T. Enokido, M. Takizawa: "A Simple Energy-Aware Virtual Machine Migration Algorithm in a Server Cluster," *Proc. of the 20th International Conference on Network-based Information Systems* (*NBiS-2017*), Toronto, Canada, August, 2017, pp.55-65.
52. R. Watanabe, D. Duolikun, T. Enokido, M. Takizawa: "A Simple Migration Algorithm of Virtual Machine in a Server Cluster," *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2017*), Barcelona, Spain, November 2017, 149-160.
53. T. Enokido, D. Duolikun, M. Takizawa: "An Energy Efficient Load Balancing Algorithm Based on the Active Time of Cores," *Proc. of the 11th International Conference on Broadband and Wireless Computing, Communication and Applications* (*BWCCA-2017*), Barcelona, Spain, November 2017, pp.185-196.
54. R. Watanabe, D. Duolikun, T. Enokido, M. Takizawa: "Eco Migration Algorithms of Processes with Virtual Machines in a Server Cluster," *International Conference on Emerging Internet, Data and Web Technologies* (*EIDWT-2018*), Tirana, Albania, March 2018, (Accepted).
55. R. Watanabe, D. Duolikun, T. Enokido, M. Takizawa: "Energy-aware Virtual Machine Migration Models in a Scalable Cluster of Servers," *Proc. of IEEE the 32nd International Conference on Advanced Information Networking and Applications* (AINA-2018), Cracow, Poland, May 2018, (Accepted).

3. Awards

1. **Best paper award**, "Power Consumption Models for Redundantly Performing Mobile-Agents," *the 8th International Conference on Complex, Intelligent, and Software Intensive Systems* (*CISIS-2014*), Birmingham, UK, July 2014.
2. 2015 年度**中国新疆籍優秀自費留学生賞**, Tokyo, Japan, November, 2015.
3. **Best paper award**, "A Model for Energy-Aware Migration of Virtual Machines," *the 19th International Conference on Network-based Information Systems* (*NBiS-2016*), Ostrava, Czech, Sept. 2016.

4. Grants

1. 公益財団法人 NEC C&C 財団 2017 年度（平成 29 年度）外国人研究員助成.
   (NEC C&C FOUNDATION Grants for Non-Japanese Researchers (Grants for Fiscal Year 2017)).
2. 独立行政法人日本学術振興会 特別研究員-DC2 (平成 30 年度).
   (Japan Society for the Promotion of Science Research Fellowship for Young Scientists-DC2 (Grants for Fiscal Year 2018)).