System Specification-based Design of Cloud Data Centre and DEVS Simulation for Availability Evaluation

Ji-Yeon Kim and Hyung-Jong Kim*

Department of Information Security
Seoul Women's University
621 Hwarangro, Nowon-Gu, Seoul, 137–774, Republic of Korea
Email: {jykim07, hkim}@swu.ac.kr

In the cloud computing environment, all IT resources are located in centralized cloud data centres (CDCs) which integrate and allocate resources using virtualization techniques. To ensure that a large number of service requests are handled in a stable manner, a CDC needs to evaluate its availability in advance and improve availability by compensating for shortcomings in infrastructure and operational policy. However, it is difficult to design a CDC due to high complexity of structure and behaviour, so that we could not evaluate its availability under various conditions. In this paper, we design a CDC by making use of system specification method, and propose a simulation model for evaluating an availability of a CDC considering its infrastructure and resource management scheme. By making use of system specification method in design level, we can define a hierarchical and modularized system, representing characteristics of a system's structure and behaviour. Our simulation model is designed based on DEVS (Discrete EVent system Specification) formalism, and we use an experimental frame (EF) concept, which provides a flexible experimentation environment by separating the environment from target models. In addition, to verify our model's design and execution, we implement the model using DEVSJAVA and simulate under different scenarios.

Keywords: *Cloud computing, Data centre, System specification, DEVS (Discrete EVent system Specification) formalism, Modeling and simulation, Availability*

ACM Classifications: *C.2.4 Distributed Systems (Cloud computing), C.4 Performance of systems (Availability), I.6 Simulation and modeling*

1. Introduction

Cloud computing is an internet-based computing concept, in which shared IT resources in a large cloud data centre (CDC) are provided to user devices on demand (Paquette *et al*, 2010; Wang *et el*, 2011). Because IT resources are equivalent to a "service" in the cloud computing environment, users can access and use the services from any internet connection even though they may not have access to local IT resources (The art of service, 2008).

Recently, high-speed internet and mobile devices, such as smartphones, have become more accessible. As a consequence, many people have started using cloud services because of the limitations of mobile devices, such as low processing power, poor battery life, and low data

* Corresponding author: Hyung-Jong Kim

Manuscript received: 14 August 2012
Communicating Editor: Seok-Hun Kim

storage capacity. However, this has led to an increase in the number of service requests to CDCs. Thus, to handle a large number of service requests at all times using limited resources, CDC availability must be ensured (Song *et el*, 2006). In cloud computing, availability refers to the capacity for processing cloud services normally within the specified time period. We can also define availability as the capacity for operating the virtualized resources of a CDC, because IT resources are allocated dynamically to the service requests using virtualized physical resources. However, availability evaluation in the context of cloud computing has hardly been studied.

Khurshid *et al* (2009) measured a cloud's performance during data transfer using a cloud computing testbed. However, the limitation of this study is that it focused on internal cloud architectures such as topology and file system. In addition, the fact that this study did not consider virtualization techniques implies that the conclusions derived may not be true for a commercial cloud. Most previous studies related to the availability of data centres focused on scheduling algorithms and power-based management schemes to improve the effectiveness of data centres. Kong *et al* (2010) proposed a task-scheduling scheme for virtualized data centres by considering a tradeoff between availability and performance, and Marwah *et al* (2009) and McAllister *et al* (2008) proposed a model and computation strategies for a data centre based on the concept of energy efficiency. However, because most studies do not describe the implementation of their respective proposed models, it is difficult to apply such models to other data centres that may have different infrastructures or operational policies. In addition, these models cannot be applied to CDCs because they do not consider the characteristics of cloud services.

In this paper, we design a CDC based on system specification method, a system theory based formal representation method. By using the method, we can represent the overall CDC's structure and behaviour, so that we can design a hierarchical and modularized CDC. Also, we propose a simulation model for evaluating the availability of a CDC considering its infrastructure, resource management schemes and cloud service type. Our model is designed using DEVS (Discrete EVent system Specification) formalism suggested by Zeigler *et al* (2000), and we implement the model using DEVSJAVA (Zeigler and Sarjoughian, 2005). In addition, we simulate and evaluate the availability under three different scenarios to verify our model. The main contributions of this paper are as follows.

1. The model enables CDCs to assess their availability considering their infrastructure and operational policy.
2. The model enables CDCs to create and verify resource management schemes by taking into account system capacity, service type, and security incidents; therefore, CDCs can derive an optimal resource allocation scheme.
3. The model makes CDCs to implement a cost-effective infrastructure by considering the amount of service requests.

The remainder of this paper is organized as follows. Section 2 provides a background to a CDC and our simulation model by explaining concepts such as system specification method, DEVS formalism, EF and a VM migration scheme. In Section 3, we design a CDC based on system specification method. In Section 4, we design a simulation model for evaluating an availability of a CDC, and describe detailed methods and parameters involved in creating the model. In Section 5, we develop three simulation scenarios and simulate the availability evaluation. Finally, the conclusions and future work are presented in Section 6.

2. Background

2.1 System Specification

System specification is a formal representation method for representing structure and behaviour of systems. There are eight levels of system specification, I/O frame, I/O relation observation, I/O function observation, I/O system, iterative specification, structured system specification, nonmodular coupled multicomponent system, and modular coupled network of systems. For simulation purposes, the more structured levels are most practical. A coupled system specification at the structured system level is a structure N (Zeigler *et al*, 2000):

$$N = < T, X_N, Y_N, D, \{M_d | d \in D\}, EIC, EOC, IC> \quad (1)$$

where X_N is a multivariable set of inputs, Y_N is a multivariable set of outputs, D is a set of components, EIC is an external input coupling, EOC is an external output coupling, IC is an internal coupling, and M_d is a multicomponent system:

$$M_d = (T, X_d, Y_d, \Omega, Q, \Delta, \Lambda) \quad (2)$$

where T is a time base, X_d is an input values set, Y_d is an output values set, Q is a set of states, Δ is a global state transition function, and Λ is an output function.

2.2 DEVS Formalism and Experimental Frame

Discrete event system specification (DEVS) is a formalism for the modeling and analysis of discrete event systems that allows for the hierarchical and modular modeling of discrete event systems. DEVS provides the basis for system modeling by defining general characteristics of systems based on inputs, states, outputs, and functions.

An experimental frame (EF) is a specification of the conditions under which a system is observed or experimented with (Zeigler *et al*, 2000). There are two viewpoints on the EF concept. The first viewpoint looks at a frame as a definition of the type of data elements that go into a database. The second views a frame as a system in itself that interacts with the system of interest under specified conditions. Under the second viewpoint, a frame is characterized by its implementation as either a measurement system or an observer (Zeigler *et al*, 2000). Further, under the second approach, an EF consists of a generator, an acceptor, and a transducer. The generator stimulates the system under investigation in a known, desired manner. The acceptor monitors an experiment to ensure that the desired conditions are satisfied, and the transducer observes and analyzes the system outputs. The EF concept provides a structure for specifying the simulation conditions that are to be observed during a given analysis (Veena, 2005).

2.3 High Availability of XenServer

Citrix XenServer has high availability (HA), which ensures that applications are accessible most of the time and recover quickly from localized failures so that users experience little or no interruption (Citrix Systems, 2009b). If the HA function is active, XenServer monitors and detects a failed virtual machine (VM) of a host in the resource pool and moves the VM to another host. In this paper, we use the HA function of XenServer as a resource management scheme for CDC, which imparts a degree of realism to our simulation model by reflecting the actual mechanism of a virtualized environment. In our simulation model, the HA function is implemented in the <<Server Manager (SM)>> model, including the migration mechanism, as summarized in Table 1.

Stage		Description
0	Pre-Migration	Selecting new host B for migration
1	Reservation	Initializing a container on the target host B
2	Stop and copy	Suspending VM on host A and synchronizing the states of all remaining VMs to Host B
3	Commitment	Releasing VM state on Host A
4	Activation	Starting VM on Host B

Table 1: The migration mechanism of XenServer

3. System Specification-based Design of a CDC

Virtualization is a key technology in enabling cloud computing, and therefore, in transforming the face of the modern data centre (Cloud security alliance, 2009). To evaluate the availability of a CDC, we have to design a CDC considering its infrastructure associated with the capacity for operating the virtualized resources. Figure 1 shows a CDC's components and their relations between input and output.

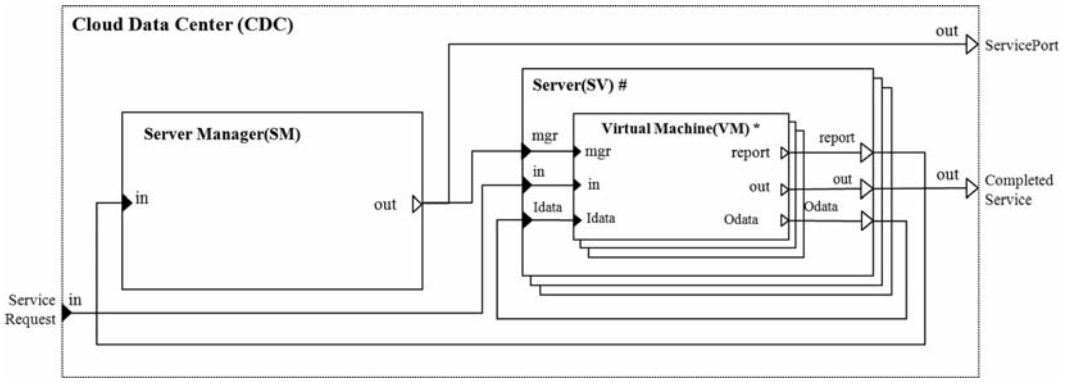

Figure 1: Block diagram representation of a CDC

The <<CDC>> functions to integrate and manage the resources of a CDC. These functions are contained in the CDC as virtualization technology. The <<CDC>> comprises the <<Server Manager (SM)>> and a large number of <<Server (SV)>>. The <<SM>> controls resources according to the operational policy and a large number of <<SV>> handle service requests from users by creating and executing VMs by service type. The CDC specification at the structured system level is shown in Table 2.

4. Development of a Simulation Model for Availability Evaluation

In this section, we design a simulation model for evaluating the availability of a CDC. We designed the dynamic characteristics of low-level models of the simulation model and the interactions among these models using the DEVS formalism. Figure 2 shows the structure of our simulation model, which consists of two models, the <<EF>> and the <<CDC>>. The <<CDC>> is

System

$CDC = < T, X_{CDC}, Y_{CDC}, D, \{M_{SM}, M_{SV} \mid SM, SV \in D\}, EIC, EOC, IC >$

$X_{CDC} = (IPorts_{CDC}, ServiceRequest)$

$Y_{CDC} = (OPorts_{CDC}, ServicePort \times CompletedService)$

Coupling

$EIC \subseteq \{((CDC, in), (SV, in) \mid in \in IPorts_{CDC}, SV \in D, in \in IPorts_{SV})\}$

$EOC \subseteq \{((SM, out), (CDC, out) \mid SM \in D, out \in OPorts_{SM}, out \in OPorts_{CDC})\}$

$EOC \subseteq \{((SV, out), (CDC, out) \mid SV \in D, out \in OPorts_{SV}, out \in OPorts_{CDC})\}$

$IC \subseteq \{((SM, out), (SV, mgr) \mid SM, SV \in D, out \in OPorts_{SM,} mgr \in IPorts_{SV})\}$

$IC \subseteq \{((SV, report), (SM, in) \mid SV, SM \in D, report \in OPorts_{SV,} in \in IPorts_{SM})\}$

$IC \subseteq \{((SV, Odata), (SV, Idata) \mid SV \in D, Odata \in OPorts_{SV}, Idata \in IPorts_{SV})\}$

Constraints

$\forall ((CDC, in), (SV, in)) \in EIC: range_{in}(X_{CDC}) \subseteq range_{in}(X_{SV})$

$\forall ((SM, out), (CDC, out)) \in EOC: range_{out}(Y_{SM}) \subseteq range_{out}(Y_{CDC})$

$\forall ((SV, out), (CDC, out)) \in EOC: range_{out}(Y_{SV}) \subseteq range_{out}(Y_{CDC})$

$\forall ((SM, out), (SV, mgr) \in IC: range_{out}(Y_{SM}) \subseteq range_{mgr}(X_{SV})$

$\forall ((SV, report), (SM, in) \in IC: range_{report}(Y_{SV}) \subseteq range_{in}(X_{SM})$

$\forall ((SV, Odata), (SV, Idata) \in IC: range_{Odata}(Y_{SV}) \subseteq range_{Idata}(X_{SV})$

Components

$M_{SM} = < T, X_{SM}, Y_{SM}, \Omega, Q, \Delta, \Lambda >$

$M_{SV} = < T, X_{SM}, Y_{SM}, D, \{M_{VM} \mid M_{VM} \in D\}, EIC, EOC, IC >$

System

$SV = < T, X_{SV}, Y_{SV}, D, \{ M_{VM} \mid VM \in D\}, EIC, EOC, IC >$

$X_{SV} = (IPorts_{SV}, MigrationCommand \times ServiceRequest \times MigrationData)$

$Y_{SV} = (OPorts_{SV}, ResourceInfo \times CompletedServices \times MigrationData)$

Coupling

$EIC \subseteq \{((SV, mgr), (VM, mgr) \mid mgr \in IPorts_{SV}, VM \in D, mgr \in IPorts_{VM})\}$

$EIC \subseteq \{((SV, in), (VM, in) \mid in \in IPorts_{SV}, VM \in D, in \in IPorts_{VM})\}$

$EIC \subseteq \{((SV, Idata), (VM, Idata) \mid Idata \in IPorts_{SV}, VM \in D, Idata \in IPorts_{VM})\}$

$EOC \subseteq \{((VM, report), (SV, report) \mid VM \in D, report \in Oports_{VM}, report \in Oports_{SV})\}$

$EOC \subseteq \{((VM, out), (SV, out) \mid VM \in D, out \in Oports_{VM}, out \in Oports_{SV})\}$

$EOC \subseteq \{((VM, Odata), (SV, Odata) \mid VM \in D, Odata \in Oports_{VM}, Odata \in Oports_{SV})\}$

Constraints

$\forall ((SV, mgr), (VM, mgr)) \in EIC: range_{mgr}(X_{SV}) \subseteq range_{mgr}(X_{VM})$

$\forall ((SV, in), (VM, in)) \in EIC: range_{in}(X_{SV}) \subseteq range_{in}(X_{VM})$

$\forall ((SV, Idata), (VM, Idata)) \in EIC: range_{Idata}(X_{SV}) \subseteq range_{Idata}(X_{VM})$

$\forall ((VM, report), (SV, report)) \in EOC: range_{report}(Y_{VM}) \subseteq range_{report}(Y_{SV})$

$\forall ((VM, in), (SV, in)) \in EOC: range_{in}(Y_{VM}) \subseteq range_{in}(Y_{SV})$

$\forall ((VM, Odata), (SV, Odata)) \in EOC: range_{Odata}(Y_{VM}) \subseteq range_{Odata}(Y_{SV})$

Components

$M_{VM} = < T, X_{VM}, Y_{VM}, \Omega, Q, \Delta, \Lambda >$

Table 2: A CDC specification at the structured system level

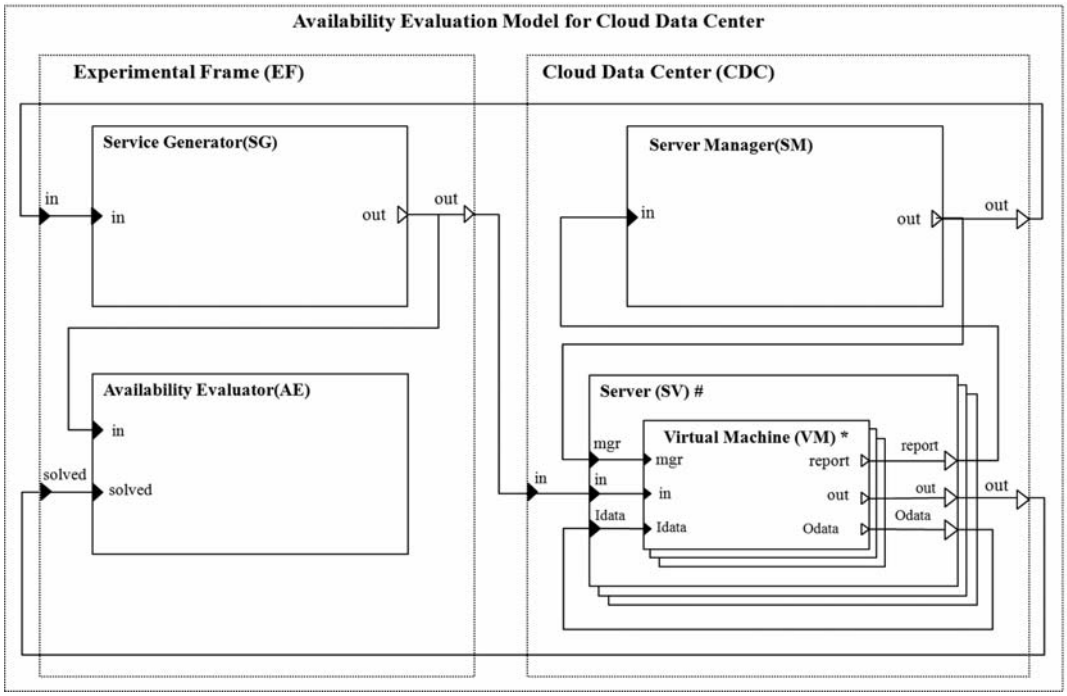

Figure 2: Availability evaluation model for cloud data centre

a target model that we want to evaluate availability and the <<EF>> is an experimentation environment model that has built-in simulation scenarios and evaluation mechanisms.

4.1 Modeling of a CDC

To model the <<CDC>> considering its real environment, we define parameters related to infrastructure and policies; these parameters are summarized in Table 3. When a <<SV>> receives a new service request through its "in" port, it creates a VM that has the default resources. Then, the model allocates a service to the VM by service type, and reports the resource status of the VM to the <<SM>> through the "report" port. If there is no problem in handling a service request, the VM processes the service and sends the results to the << AE>> through the "out" port. Conversely, if the VM receives migration commands from the <<SM>> through the "mgr" port, it sends the data that is being processed in the VM to a new destination server through "Odata" port. A VM with greater resources will be created in the destination server. The resources of a VM are determined by the parameter *Migration_Ratio*, listed in Table 3.

As shown in Figure 3, resource statuses can be reported from the <<SV>> models to the <<SM>>. When the <<SM>> receives a status message, it checks whether a VM in the server should be migrated by comparing the resource status with *Migration_Threshold*, listed in Table 3. In our simulation model, we set the policy parameters in the <<SM>> because this model needs to know the migration policies. In the <<SM>> shown in Figure 2, the "in" port receives resource statuses and sends migration commands through the "out" port.

Classification	Component	Parameters	Data type	Description
Infrastructure	Server (SV)	Server_Num	integer	Number of servers(#)
		CPU	double	Server specifications (Monitoring index)
		Memory	double	
		Bandwidth	double	
		HardDisk	double	
	Virtual Machine (VM)	Service_Num	integer	Number of services (*)
		Processing_Time	double	Processing time by service type
		Default_CPU	double	Default allocation size of resources by service type
		Default_Memory	double	
		Default_Bandwidth	double	
		Default_HardDisk	double	
Policy	Server Manager (SM)	Migration_Threshold	double	Migration threshold by service type
		Migration_Ratio	double	Increment ratio of resources for migration

Table 3: Parameters of cloud data centre

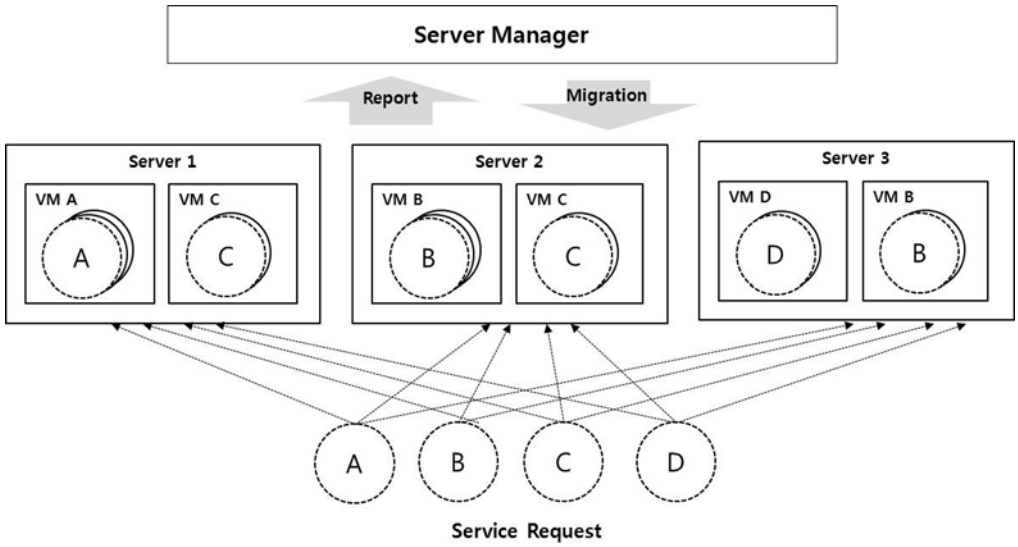

Figure 3: Resource management through <<Server Manager (SM)>>

4.2 Modeling of an EF

The <<EF>> consists of two models, the <<Service Generator (SG)>> and the <<Availability Evaluation (AE)>>. As far as the <<SG>> is concerned, cloud services are generated and sent to the <<CDC>> and the <<AE>> through the "out" port to provide input for simulation. The <<AE>>

Parameter	Data Type	Description
Service_Type	integer	Types of cloud services
Interarrival_Time	double	Time interval between generation of service requests
Job_Size	double	Size of cloud service
Processing_Time	double	Time interval between start and finish of service

Table 4: The parameters of the <<Service Generator (SG)>> model

evaluates the availability of the <<CDC>> by collecting result messages from the <<SV>> and comparing these messages with the generated services. For modeling the <<SG>>, as summarized in Table 4, we need to define parameters that reflect characteristics of cloud services because this model generates the input for our simulation model.

In our simulation model, we have assumed that the cloud service is an IaaS (Infrastructure as a Service) and that the users of this service want to store their data on the cloud. In this scenario, *Job_Size* refers to the size of data stored on the hard disk and we set a fixed *Processing_Time* regardless of *Job_Size*. This is because if a path between a user and a hard disk is configured, data can be transmitted to the hard disk using the configured path and such data transmission does not consume computing power. In the <<SG>>, the "in" port receives information about migration from the <<SM>>, which provides information about the destination server according to the service type.

In the <<AE>>, the "in" port receives generated services from the <<SG>> and the "solved" port receives information about fulfilled service requests. In our simulation model, there are two availability indices, average turnaround time and average throughput. Turnaround time is the time interval between request and fulfillment of service request and throughput is the number of fulfilled service requests per unit time.

5. DEVS Simulation

To verify our simulation model, we implemented the simulation model using DEVSJAVA, which provides a library for implementing DEVS-based models. Figure 4 shows the DEVSJAVA implementation of our model. There are three <<SV>> models that can deal with three types of service, a <<SM>>, and an <<EF>> in our simulation model. In this section, we describe a simulation that was performed under different scenarios.

5.1 Simulation Scenarios

The CDC deals with service requests in various manners according to its resource management policy. Thus, we first created simulation scenarios by considering various policy cases of CDC. Then, we compared the availability in various simulation scenarios by analyzing the simulation results. In this study, we used three resource allocation scenarios, as shown in Figure 5.

In scenario 1, the CDC does not use virtualization, and each server deals with only one type of service. Thus, although individual service types can have multiple servers, individual servers cannot handle various service types because of the lack of server resources.

In scenario 2, individual servers can handle multiple service types by using multiple VMs and allocating one service request to one VM. If a VM cannot handle a service request because of the lack of the VM resources, migration will occur. If all server resources are used by a single VM, migration cannot occur and the system status will be the same as that in scenario 1.

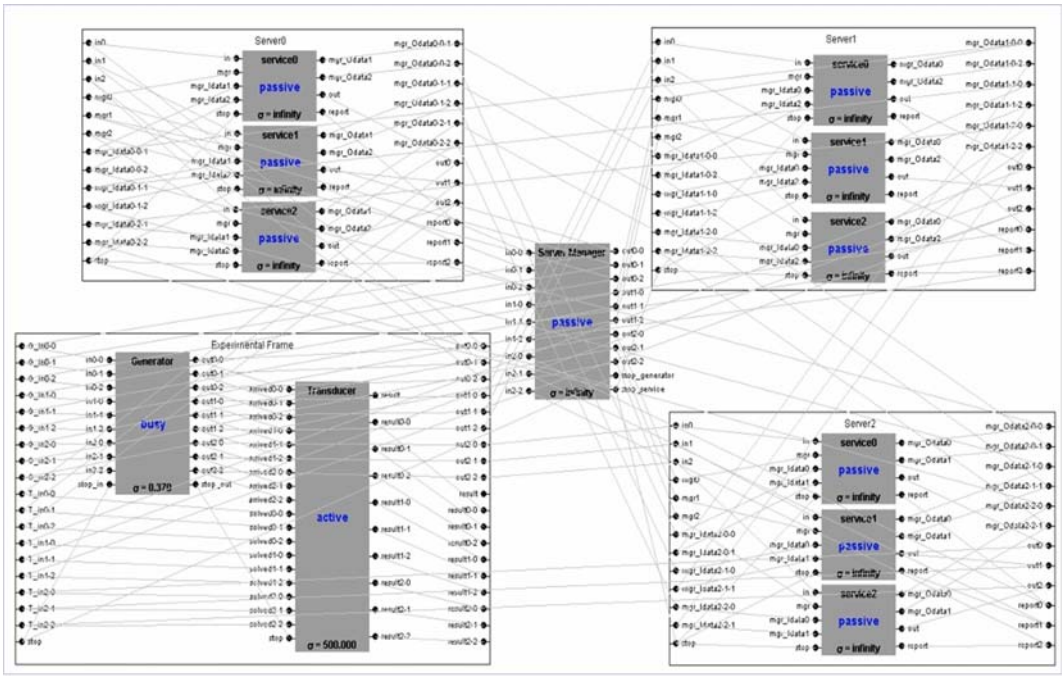

Figure 4: Screenshot of the DEVSJAVA implementation of the simulation model

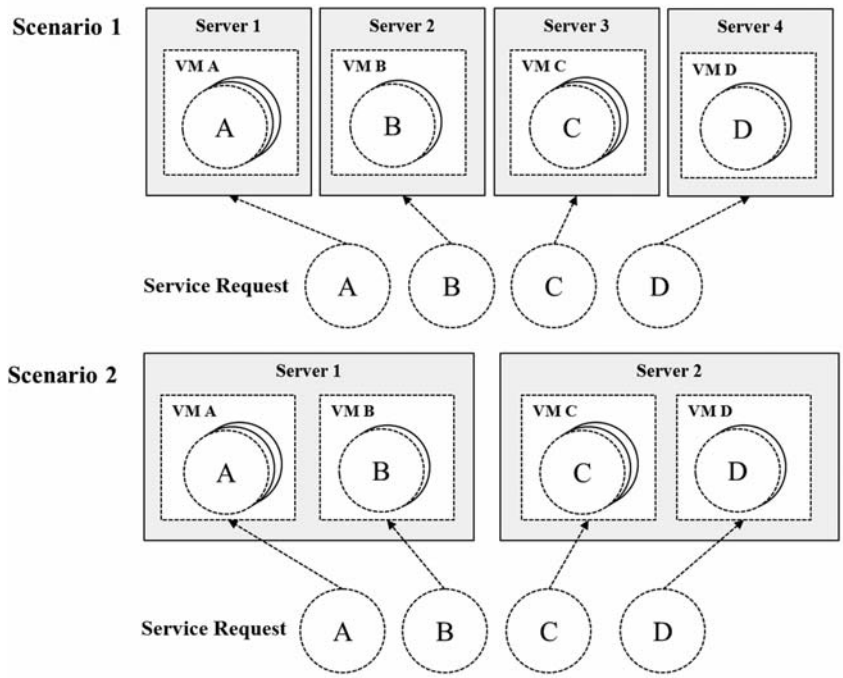

Figure 5 (*part*): Resource allocation scheme for each scenario

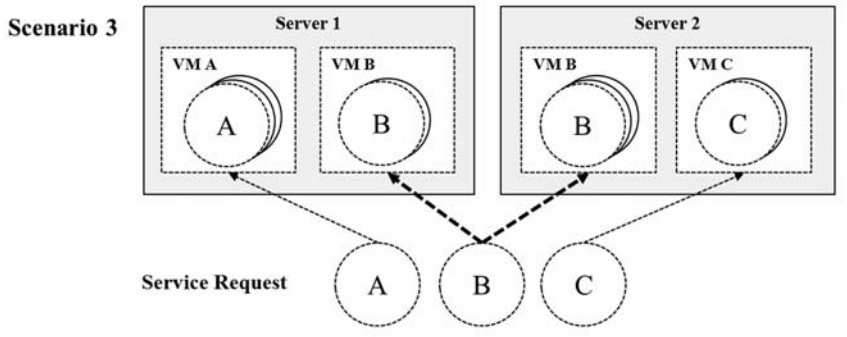

Figure 5 (*part*): Resource allocation scheme for each scenario

In scenario 3, a service request can be allocated to VMs of multiple servers. To use this scheme in CDC, the network interface cards (NICs) of multiple servers should be connected to each other. In the XenServer paradigm, such a network of servers is called "bonded network."

In this simulation, we set the simulation parameters as listed in Table 5. There are three types of services in the simulation model, and we set different *Processing_Time_** according to the service

Parameter	**Value**
Server_Num	3
CPU	3.0
Memory	4.0
Bandwidth	100.0
HardDisk	670.0
Service_Num	3
Processing_Time_*	Exponential(10)
Interarrival_Time	Exponential(5)
Default_CPU_#	<Scenario 1> Default_CPU_# = CPU, Default_Memory_# = Memory, Default_Bandwidth_# = Bandwidth, Default_HardDisk_# = HardDisk <Scenario 2 and Scenario 3> Default_CPU_# = CPU * Migration_Ratio / Service_Num, Default_Memory_# = Memory * Migration_Ratio / Service_Num, Default_Bandwidth_# = Bandwidth * Migration_Ratio / Service_Num, Default_HardDisk_# = HardDisk * Migration_Ratio / Service_Num
Default_Memory_#	
Default_Bandwidth_#	
Default_HardDisk_#	
Migration_Threshold	0.7
Migration_Ratio	0.5
Service_Type	1~Service_Num
Job_Size	5.0

Table 5: Simulation parameters

types by using an exponential function. The amount of default resources such as *Default_CPU_#*, *Default_Memory_#*, *Default_Bandwidth_#*, and *Default_HardDisk_#* are set according to scenarios. In the case of scenario 1, the four parameters are set to the size of server resources such as CPU, memory, hard disk, and bandwidth. In the other scenarios, the CDC administrator can determine the amount of default VM resources by taking into account *Service_Num*, *Migration_Ratio*, and so on.

5.2 Simulation Results

In our simulation, service requests are generated until lack of CDC resources prevents further migration. When the simulation is ended, we can determine the average turnaround time by computing collected data such as generated service requests and completed service requests in the <<AE>>. Figure 6 shows a plot of the average turnaround time and the average throughput in the three scenarios.

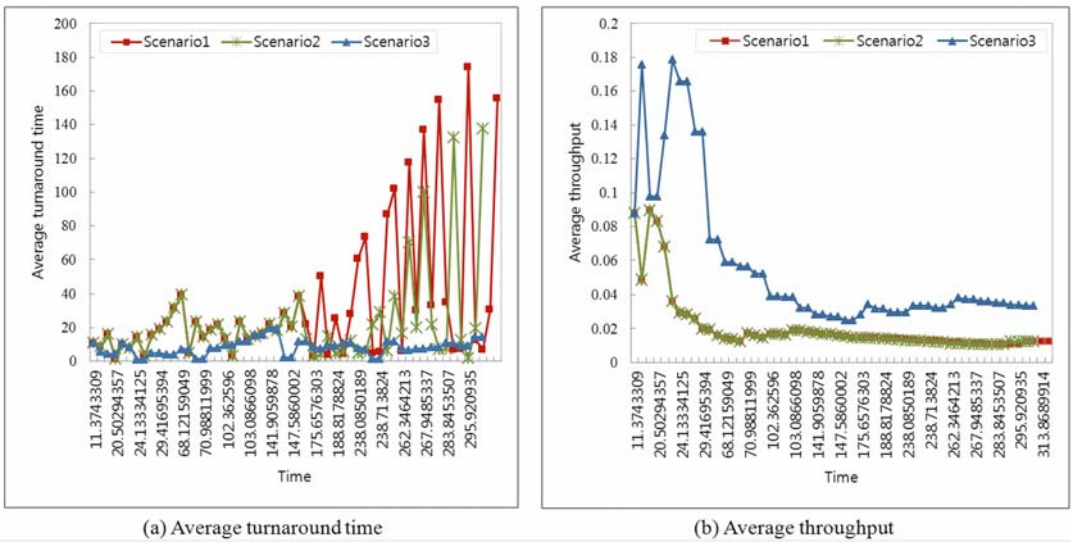

Figure 6: Simulation results of the average turnaround time and the average throughput

As time passes, the turnaround times increase rapidly in the order of scenarios 1, 2, and 3 as shown in Figure 6(a). Scenarios 2 and 3 allow for the flexible use of resources because of their migration policies; however, scenario 1 does not have a migration policy, and therefore, turnaround time rapidly increases because of server overload. Although the turnaround time in scenario 2 increases more slowly than that in scenario 1, it is still longer than that in scenario 3, because in scenario 2, a service request can be handled by only one VM. In the case of scenario 3, a VM can handle all service requests within a short period of time until the end of simulation of scenario 1.

Figure 6(b) shows the number of completed services per unit time for the three scenarios. As time passes, all the throughputs of all the scenarios decreased because of server overload. Among the three scenarios, scenario 3 has the highest throughput at all times. In our simulation, throughput cannot be zero even if the CDC is unable to perform migration because of resource scarcity. This is because the <<SM>> determines the amount of available resources in advance. If the available resources are less than *Job_Size*, it does not accept further service requests and terminates the simulation.

In our simulation results, we observe that among the three scenarios, scenario 1 has the longest turnaround time and the lowest throughput, whereas scenario 3 has the shortest turnaround time and the highest throughput. This implies that scenario 1 has the lowest availability, whereas scenario 3 has the highest. In Section 5.1, because scenario 1 had no resource management policy and scenario 3 had various resource management policies, we can infer that the implementation of our model is sound reflecting the robustness of the derived parameters and the resource management scheme.

6. Conclusion

In this paper, we designed a hierarchical and modularized cloud data centre (CDC) using system specification method and proposed a simulation model for evaluating the availability of a CDC. We developed the simulation model considering not only infrastructure such as virtualized servers of the CDC but also its operational policy. In our modeling, we used a DEVS formalism which specifies discrete event systems in a hierarchical modular manner, and derived CDC parameters in detail to suggest a detailed method for modeling a CDC.

To verify the simulation model's execution, we implemented the model using DEVSJAVA, which provides a library for DEVS simulation. In addition, we created three different scenarios considering various resource allocation schemes of a CDC, and performed a simulation. From the simulation results, we can see that the scenario 1, to which the virtualization technique was not applied, has the lowest availability and scenario 3, which has various schemes such as migration policy and virtualization techniques, has the highest availability. The simulation results show that our model is well-designed and it works. Our model can be used for developing an evaluation environment for various CDCs reflecting the characteristics of each CDC, and we can incorporate additional CDC-related modeling parameters, because the model is designed based on the DEVS formalism.

As future work, we will derive additional availability indices of the <<EF>> model and modeling parameters of the <<CDC>> model by researching performance indices and CDC characteristics. In addition, we need to simulate the model under practical scenarios by considering commercial IaaS services with the complex form of service distribution methods and a priority of services. Furthermore, to use our simulation model in the other types of cloud services such as SaaS (Software as a Service) and PaaS (Platform as a Service), we will refine and expand our simulation model.

Acknowledgement

This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2062654).

References

CATBIRD NETWORKS (2008): Virtualization Security: The Catbird Primet. Scotts Valley, CA 95066: Catbird Networks Inc.

CHREYH, R. (2009): An Internet-based Searchable Repository for DEVS Models and their Experimental Frames. M.S., Carleton University.

CITRIX SYSTEMS (2009a): How does Xen Work? Version 1.0. Technical report. http://xen.org/files/Marketing/HowDoesXenWork.pdf, Accessed 18-Nov-2011.

CITRIX SYSTEMS (2009b): The three levels of high availability-balancing priorities and cost. White paper. http://deliver.citrix.com/go/citrix/CtxMarathonXS5WP, Accessed 18-Nov-2011.

CITRIX SYSTEMS (2010): Citrix XenServer 5.6 Software Development Kit. 1.0 Edition. Developer's guide. http://support.citrix.com/servlet/KbServlet/download/23832-102-645656/XenServer-5.6.0-sdk.pdf, Accessed 18-Nov-2011.

CLOUD SECURITY ALLIANCE (2009): Security Guidance for critical areas of focus in cloud computing. https://cloudsecurityalliance.org/research/initiatives/security-guidance, Accessed 18-Nov-2011.
HERTONG, S., CHOKCHAI, L. and RAJA, N. (2006): Availability Modeling and Evaluation on High Performance Cluster Computing Systems. *Journal of Research and Practice in Information Technology* 38(4): 317–335.
KHURSHID, A., AL-NAYEEM, A. and GUPTA, I. (2009): Performance Evaluation of the Illinois Cloud Computing Testbed. Technical Report. Ilinois digital environment for access to learning and scholarship.
KONG, X., LIN, C., JIANG, Y., YAN, W. and CHU, X. (2010): Efficient dynamic task scheduling in virtualized data centres with fuzzy prediction. *Journal of Network and Computer Applications* 34(4): 1068–1077.
LIU, Y., GERSTEIN, M. and ENGELMAN, D.M. (2004): Transmembrane protein domains rarely use covalent domain recombination as an evolutionary mechanism. *Proceedings of the National Academy Sciences of the United States of America* 101(10): 3495–3497.
LLUIS. P.J, PEDRO, G.L., MARC, S.A. and HERRERA, B. (2011): Towards the design of optimal data redundancy schemes for heterogeneous cloud storage infrastructures. *Computer Networks* 55(5): 1100–1113.
MARWAH, M., MACIEL, P., SHAH, A., SHARMA, R., CHRISTIAN, T., ALMEIDA, V., ARAUJO, C., SOUZA, E., CALLOU, G., SILVA, B., GALDINO, S. and PIRES, J. (2009): Quantifying the Sustainability Impact of Data Centre Availability. *GreenMetrics '09*, Seattle, WA, ACM Sigmetrics.
McALLISTER, S., CAREY, V.P, SHAH, A., BASH, C. and PATEL, C. (2008): Strategies for effective use of exergy-based modeling of data centre thermal management systems. *Microelectronics Journal* 39: 1024–1029.
PAQUETTE, S., JAEQER, P.T. and WILSON, S.C. (2010): Identifying the security risks associated with governmental use of cloud computing. *Government Information Quarterly* 27(3): 245–253.
THE ART OF SERVICE (2008): Cloud Computing Foundation complete Certification kit.
VEENA, M. (2005): Development of Experimental Frame and Abstract Devs Models to SupportScope Network Expansion. M.S, University of Arizona.
WANG, S.S., YAN, K.Q. and WANG, S.C. (2011): Achieving efficient agreement within a dual-failure cloud-computing environment. *Expert Systems with Applications* 38(1): 906–915.
YILMAZ, L. and ORENL, T. (2010): Agent-Directed Simulation and Systems Engineering. Wiley-VCH.
ZEIGLER, B.P. (1995): Object Oriented Simulation with Hierarchical, Modular Models.
ZEIGLER, B.P., PRAEHOFER, H. and KIM, T.G. (2000): *Theory of Modeling and Simulation*. 2nd edition, Academic Press.
ZEIGLER, B.P. and SARJOUGHIAN, H.S. (2005): Introduction to DEVS Modeling and Simulation with JAVA: Developing Component-Based Simulation Models.
ZHANG, Y., SU, A.J. and JIANG G. (2011): Understanding data centre network architectures in virtualized environments: A view from multi-tier applications. *Computer Networks* 55(9): 2196–2208.

Biographical Notes

Ji-Yeon Kim *received her BSc and PhD degrees in information security engineering from Seoul Women's University in 2007 and 2013, respectively. She is currently a postdoctoral researcher in the Department of Electrical and Computer Engineering at Carnegie Mellon University. Her research interests include cloud computing security, VoIP security, cybersecurity and M&S (modeling and simulation) methodology.*

Ji-Yeon Kim

Hyung-Jong Kim *has been a faculty member of Seoul Women's University since March 2007. He worked as a principal researcher at Korea Information Security Agency (KISA) from 2001 to 2007. He received his information engineering BSc (1996) degree from Sungkyunkwan University, Korea. Also, he received his MSc (1998) and PhD (2001) from the Electrical Computer Engineering Department of Sungkyunkwan University. He worked in the CyLab Korea at CMU (Carnegie Mellon University) as a visiting scholar from 2004 to 2006. His research interests include cloud computing security, VoIP security, privacy protection and simulation modeling methodology.*

Hyung-Jong Kim