Simulation Modelling Practice and Theory 26 (2012) 135–150

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Cloud-based computer simulation: Towards planting existing simulation software into the cloud

Xiaocheng Liu [a,b,*], Qiang He [a], Xiaogang Qiu [a], Bin Chen [a], Kedi Huang [a]

[a] System Simulation Lab, Mechatronics and Automation School, National University of Defense Technology, Changsha, Hunan Province 410073, China
[b] Center for Distributed and High Performance Computing, School of Information Technologies, The University of Sydney, NSW 2006, Australia

ARTICLE INFO

Article history:
Received 6 July 2011
Received in revised form 16 April 2012
Accepted 2 May 2012
Available online 28 May 2012

Keywords:
Cloud computing
M&S
Modeling as a service
Simulation as a service
Parallel Discrete Event Simulation

ABSTRACT

Cloud computing provides various IT resources to users transparently in an easy-to-use, on-demand, cheap and pay-as-you-go manner. It frees users of burdens associated with managing computing resources and facilities, and reduces or even eliminates the capital outlays in hardware. These advantages are helpful to make simulation techniques become more accessible to users. Cloud-based computer Simulation (CSim) is becoming more and more attractive to Modeling and Simulation (M&S) practitioners. The primary challenge associated with CSim is how to make it practical. There are only a few of CSim research work proposed in the literature. To our best knowledge, none of them has tackled this question in full scale. This paper presents our work on planting existing simulation software into the cloud. The needs and the architecture of CSim, the development of CSim services, a modified modeling method and a novel simulation execution mode are proposed. In particular, some guidelines of making effective use of computing resources are developed after extensive experimentation. These guidelines provide the scheduler in CSim with useful approaches.

1. Introduction

Simulation techniques are based on Information Technology, Similarity Principal, Modeling Theory, System Engineering and technologies related to application domains. Simulation techniques utilize simulation models of real/conceptual systems for dynamic experimentation. They have been commonly recognized as new developed technologies after theoretic research and experimental research, which can also make contributions to impact and change the world [1,2]. As an important subject in simulation techniques, computer simulation (shortened as simulation in the rest of this paper) is defined as a hybrid technology of using computer science and technology to build simulation models and then performing experimentation on the models under various environments and conditions. It has advantages such as high efficiency, high security, scalability, and flexibility. It has become an important tool for design, analysis, and evaluating systems (especially complex systems), and has been playing important roles in domains of astronautics, military, economy, medicine and entertainment, with great success [3]. Computing infrastructure served as the underlying supporting resource in the computer simulation, whose architecture has critical influences on the way of building and executing simulations. Hence, analogue machine-based simulation, mainframe-based simulation, distributed simulation, parallel simulation and grid-based simulation were developed in consequence.

* Corresponding author at: System Simulation Lab, Mechatronics and Automation School, National University of Defense Technology, Changsha, Hunan Province 410073, China. Tel.: +86 84573389 8021.
E-mail address: nudt200203012007xcl@gmail.com (X. Liu).

1569-190X/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2012.05.001

In cloud computing, the IT service provider provides computing resources to consumers as a utility similar to electricity; the consumers can access the IT resources easily and pay for the ones they consumed [4,5]. Nowadays, cloud computing has been recognized as the third revolution in the IT industry [6], it has made significant changes to the way of providing and obtaining IT services. Cloud computing is more and more appealing because: (1) From the technique perspective, key techniques in cloud computing such as virtualization technology, automatic deployment, resource management, web service, SOA [7], high performance I/O, high-speed internet are well developed and have been widely applied in various domains. In addition, the rapid development of the internet backbone with increasing bandwidth and coverage and the popularization of light-weight devices (such as cell phone, hand computer, laptop and PDAs) with embedded internet connectivity make the internet ubiquitous. (2) From the IT policy of main countries/regions perspective, the success of cloud computing in the industrial world has drawn the attentions of the countries all over the world. Since 2008, many countries/regions have developed their cloud computing plans. For example, RACE, Nebula and CCM projects in the USA [8,9]; G_Cloud Project in UK [10]; Kasumigaseki Cloud Plan in Japan [11]; Green Growth Strategy in Korea [12]; Auspicious Cloud Project in China [13]. Besides, NATO and European have also proposed their cloud computing plans [14,15].

We believe cloud computing can also bring computer simulation into CSim because CSim can well alleviate some pains faced by the M&S practice [16,17]. The pains include: large capital outlays in hardware but low utilization, high complexity in building simulation system, high labor cost in simulation software maintenance etc. Howerver, CSim is not such straightforward yet, some questions are still worth deep investigation such as: (1) what is the form of CSim? (2) how to implement CSim and (3) what are the key issues involved in the implementation? Unlike previous research work that focused on some specific aspects in CSim, this paper focuses on the questions mentioned above and makes the following contributions:

- We propose the architecture of CSim.
- We improve current modeling method and simulation execution mode for CSim.
- We present the process of planting existing simulation software in the cloud, using a Parallel Discrete Event Simulation (PDES) engine as an example.
- We conclude some guidelines of making effective use of computing resources through detailed experiments, to provide the scheduling in CSim with powerful approaches.
- We suggest the direction of further improvement in the CSim research.

The rest of this paper is organized as follows: Section 2 presents the architecture of CSim. The detailed process of planting existing PDES engine into the cloud is introduced in Section 3. Experiments are proposed in Section 4, which developed some guidelines of making effective use of computing resources in CSim. Related work is shown in Section 5. We conclude our work in Section 6 and finally some open issues and future work are discussed in Section 7.

2. The architecture of CSim

2.1. The workflow of simulation [18]

As demonstrated in Fig. 1, simulation roughly consists of the following steps: *Modeling*, *Execution* and *Analysis*.

At the *Modeling* stage, *mathematical models* of the components in the system under study are extracted by the *Modelers* first. Then the *simulation models* of these mathematical models are built with the support of *Modeling tools* and *Testing tools*. Finally, the simulation models are committed into the *Simulation Resource Repository* with the help of *Managers*.

At the *Execution* stage, a *simulation scenario* is identified by the *Analysts* using the *Scenario editor*, then the scenario is submitted into the Simulation Resource Repository. With the aid of *Simulation Execution tools*, the *Execution Operators* execute the simulation according to the simulation scenario and save the simulation results into the Simulation Resource Repository.

The *Analysis* can be performed during a simulation run (on-line) or after it (off-line). At the *Analysis* stage, the *Analysts* perform Data Collecting, Online Analysis, Offline Analysis, Situation Display, and Online Directing and Adjusting.

2.2. The architecture of CSim

As illustrated in Fig. 2, CSim is composed of three major parts: *user*, *browser* in cloud terminals (cell phone, notebook, desktop etc.) and *simulation cloud*. The users of the simulation cloud mainly consist of Modelers, Execution Operators, Analysts, and Managers mentioned in Section 2.1. The simulation cloud provides clients with various simulation tools in the form of web service. It is a SaaS type cloud, which can be also called as SIMaaS (SIMulation as a service) in M&S [19]. The simulation cloud is composed of two layers: SIMaaS and VCE (Virtualized Computing Environment).

The SIMaaS layer consists of the Cloud Manager Module (*CMM*), the simulation cloud *website* and the simulation *service* layer. CMM is the "Operating System (OS)" of the simulation cloud, it includes modules such as *web − page management*, *user management*, *security management*, *service management* and *resource allocator* etc. The resource allocator schedules tasks (especially the PDES runs) involved in all stages of the simulation. It maps the simulation tasks onto suitable processors within the processor pool and controls the allocation of the computing resources. The scheduling in the resource allocator is apparently very important, and the scheduling algorithm should maximize the utilization of the computing resources

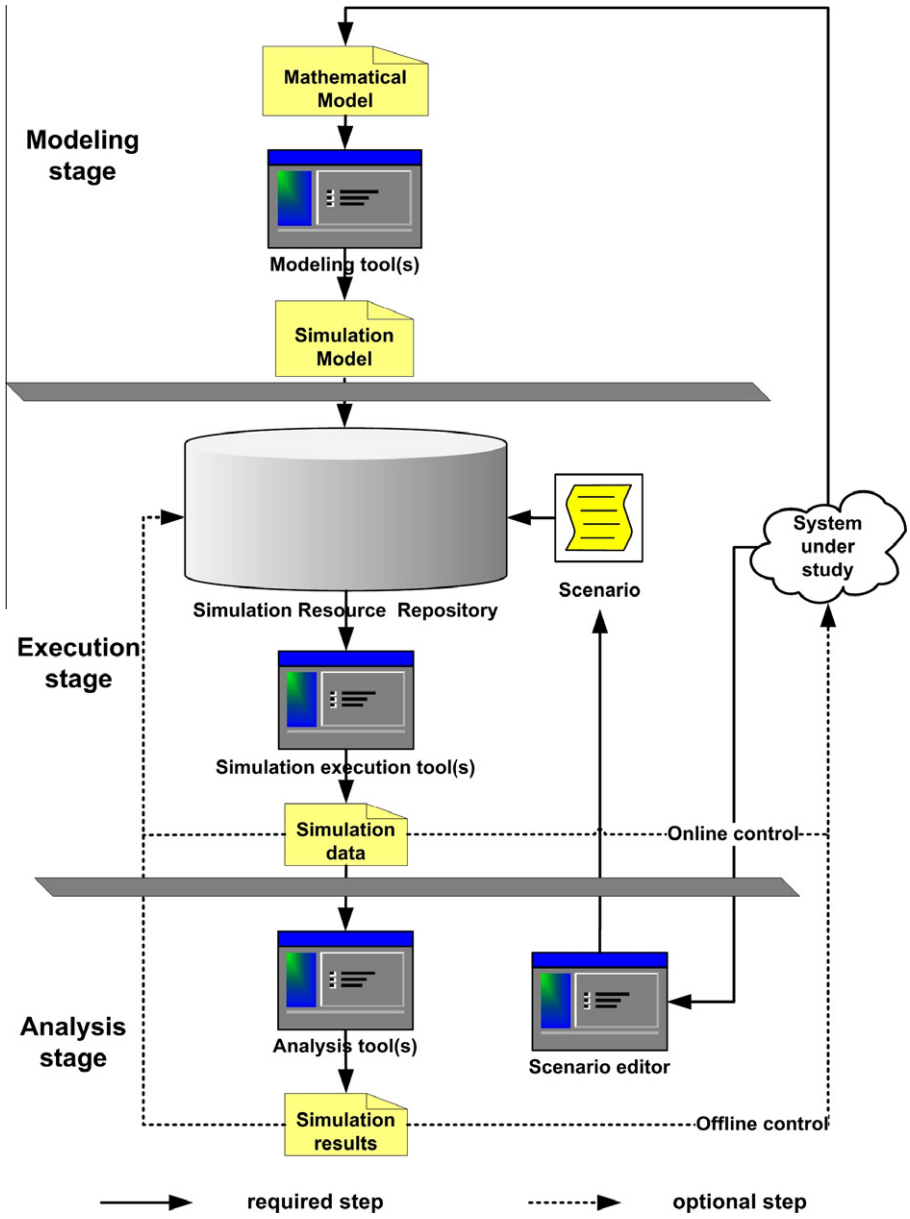


Fig. 1. The workflow of simulation.

while taking good care of the responsiveness [20], it also should produce good response time to cost ratio on the other hand [21]. The resource allocator is the "kernel" of the cloud OS. SIMaaS provides users with modeling services (Modeling as a Service, *MaaS*), execution services (Execution as a Service, *EaaS*) and analysis services (Analysis as a Service, *AaaS*) under the control of the CMM. MaaS, EaaS and AaaS all adopt SOA concept, and deliver functionalities in the form of web services through internet. MaaS includes the modeling service and the VV&A (Verification, Validation and Accreditation. Although VV&A is in fact involved in all phases of M&S, for simplicity, we put it in MaaS category) service etc. EaaS essentially includes only the execution service. The experiment design service, the resource deployment service, the simulation monitoring service, the simulation migration service and the load balancing service are optional sub-services of the execution service. AaaS includes the scenario design service, the data collecting service, the simulation display service, the on-line/off-line analysis service and the adjusting and directing service.

The *VCE* provides the services in SIMaaS with virtualized computing resources. It is constituted of the VCE interface, the virtual resource layer and the physical resource layer. The VCE interface provides the resource allocator with API to access the VCE. The physical resources include clusters, desktops, database servers, file servers, network etc. The virtual resource layer reallocates the physical resources using virtualization technology to increase resource utilization. The VCE can be built

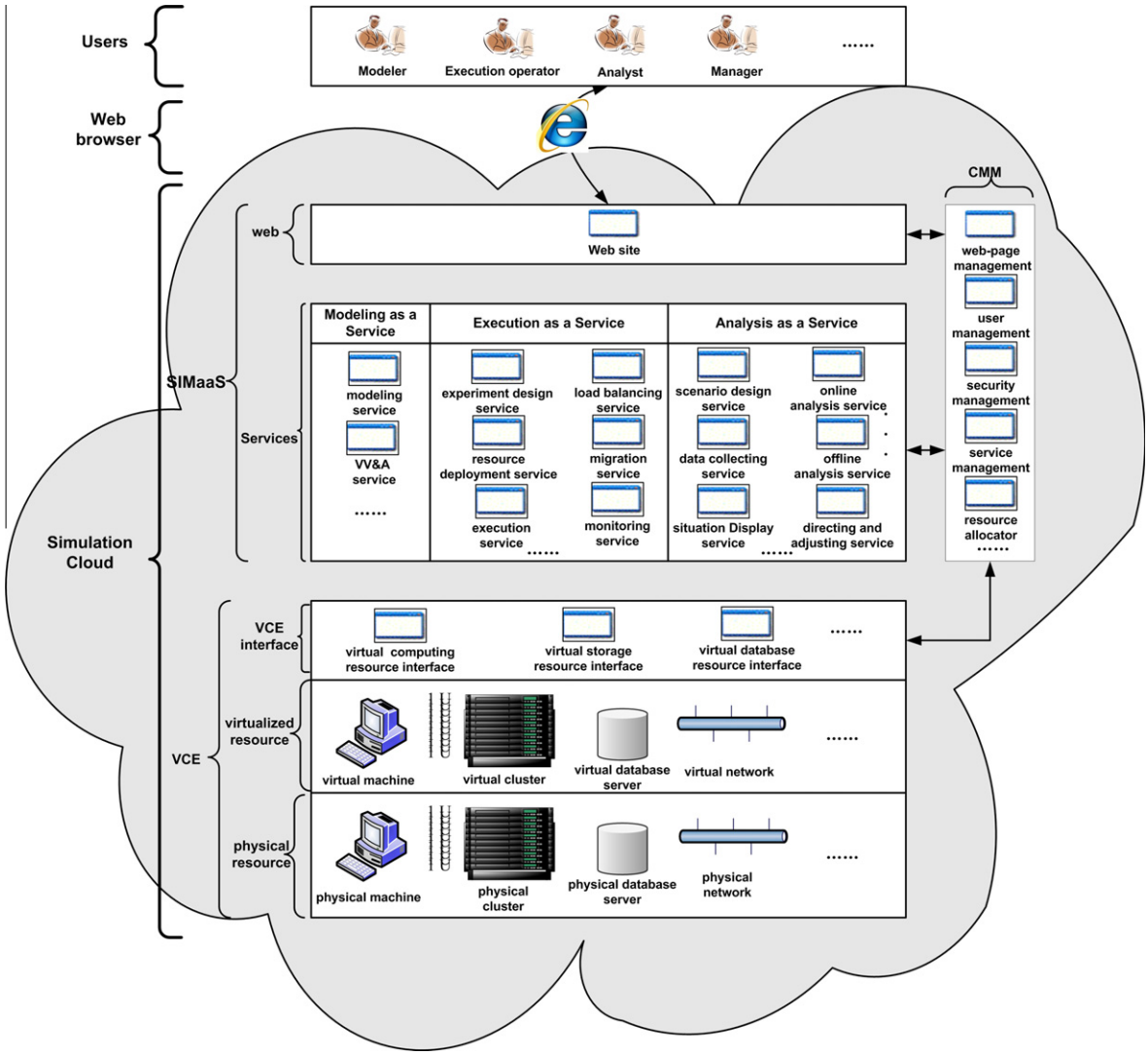


Fig. 2. The architecture of CSim.

by users themselves or outsourced in public IaaS clouds. When a simulation cloud whose VCE is provided by the users themselves, we call it a private simulation cloud; when a simulation cloud whose VCE is provided by public IaaS providers, we call it a public simulation cloud; otherwise we call it a hybrid simulation cloud.

3. Plant existing PDES engine into the cloud

The services in the MaaS, the AaaS and the experiment design service in the EaaS are usually independent interaction–intensive services. During the execution of these services, the consumption of computing resources is comparatively stable and not many. Thus the implementation and the scaling of these services are relatively easy. The execution service in the EaaS is computation-intensive and more complex, it is the key service in CSim. This section introduces the implement of the execution service by planting a PDES engine into the cloud as an example.

3.1. Preparation work

3.1.1. Build a local VCE

Although only in some cases that has the VCE to be provided by users themselves. On the one hand, any service which will be deployed into the public IaaS needs to be tested at a local VCE first; on the other hand, there is a need to build private/hybrid simulation clouds in practice. So, it is necessary to build a VCE aiming at simulation.

Some open source choices such as OpenNebula [22] and Eucalyptus [23] are useful but somewhat overstaffed for CSim, as the virtualization technology used in CSim is simple (mainly CPU and memory virtualization). In addition, a light-weight manager has extra benefits in software integration and simulation optimization. So we implemented a VCE manager for CSim using Xen [24], libvirt [25] and virt-manager [26]. The manager supports computing resources management via API and GUI. Similar work can be obtained from [27]. In addition, the design and implementation of the simulation database and the simulation repository are proposed in [28,29] respectively.

3.1.2. Understand the development of a cloud-based service [30]

The provision paradigm of most existing simulation software is shown as Fig. 3a), each user has his own copy of software installed on his own physical computing resources. The relationship among the user, the software and the computing resources is fixed and the software has no scalability.

With the introduction of virtualization technology in cloud computing, the development and deployment of software have experienced significant changes. Virtual Machine (VM, including software) serves as the minimal connector between the VCE and the developer, has become a standard media in the software deployment. Developers can deploy and redeploy software without connecting to physical servers. More importantly, virtualization makes the underlying computing resources programmable, the developers are able to use the API provided by the VCE to request/release the computing resources to scale-up/down their applications.

To more efficiently utilize cloud computing, in addition to *function module*, cloud-based software (web service) should has *monitoring module* and horizontal *scalability module* (shortened as scalability module) at least, as shown in Fig. 3b). When the request number of the service or the requirement for computing resources by the service itself change, with the help of the monitoring module, computing resources are requested/released by the scalability module.

3.2. Plant existing PDES engine into the cloud

This section presents the process of planting our existing PDES engine into the cloud, the plantation consists of the following steps: adjust the structure of the model to fit the features of cloud computing, adjust the simulation execution mode, add the horizontal scalability module to achieve service scalability, develop the resource allocator. Sections 3.2.1 and 3.2.2 detail the adjustments, Section 3.2.3 describes the logic of scalability module, Section 3.2.4 introduces the PDES scheduling scheme in the resource allocator.

3.2.1. Adjustment of the simulation execution mode

A PDES program can be viewed as a collection of sequential discrete event simulation programs (namely Logical processes, LPs) executing on different processors that communicate by sending time stamped messages to each other [31]. From an object-oriented point of view, each LP contains a set of simulated entities.

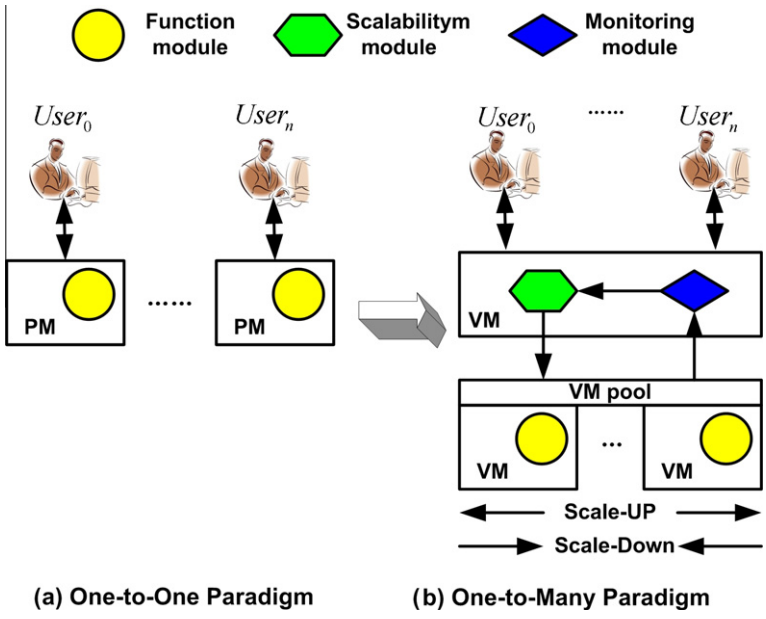


Fig. 3. Conversion of provision paradigm of simulation software.

In cloud computing (especially in the hybrid cloud), in order to effectively use the computing resources, computation (execution of some functions) may be distributed in different nodes/clouds at the function granularity rather than the class (entity) granularity.

In traditional simulation execution, the simulated entities are deployed onto different nodes, as depicted in Fig. 4a), the computation of a simulated entity is fixed on its host computer, and load balancing can only be achieved via migrating simulated entities. In order to comply with the function granularity of computation, we propose an Entity server/Calculation server (*E*/*C*)-based simulation execution mode (as illustrated in Fig. 4b), where *E* denotes the number of the entity servers and *C* denotes the number of the calculation servers. This mode is the traditional mode, where *C* equals to 0. In the *E*/*C* mode, simulated entities are distributed on the entity servers, the calculation distributor puts some calculation tasks into the calculation server when the loads of the entity servers reach a threshold during the simulation run. The load balancing in the *E*/*C* mode can be achieved by migrating simulated entities and reassigning calculation tasks as well.

3.2.2. Adjustment of the model structure

In PDES, component-based modeling is a state-of-the-art approach with increasing popularity recently. We have proposed a modeling method which is based on BOM (Base Object Model) simulation component to build simulated entity model in [32]. In this approach, the model is constituted of the model description file, the model configuration file and the model execution file. The model description file describes the format of the inputs and outputs, as well as the interfaces of them. The model configuration file stores configuration parameters which will be read by the model before and/or during the simulation execution. The model execution file is a dynamic linked library that contains the implementation of the interfaces defined in the model description file.

In the E/C execution mode, as the computation must be implemented at the function granularity therefore we divide the model execution file into two parts which are the execution framework file and the computation file respectively, as depicted in Fig. 5. These two files are all dynamic linked library and from the PEDS engine's point of view, the execution framework file equals the former model execution file but all functions are realized in the computation file. This modeling method only differs in the file structure of the model, no additional work has been introduced. Moreover, models can be reused at the function level rather than the class level.

3.2.3. The workflow of horizontal scaling

Horizontal scaling of the execution service in CSim can be divided into two parts as illustrated by Fig. 6: (1) Scaling when the number of service requests changes. (2) Scaling when the load changes during the execution of an individual execution service.

When the simulation cloud receives an incoming execution service request, it plans (using static load balancing algorithms) the number and configuration of entity servers and calculation servers required by the request first. Then it groups and maps the simulated entities onto entity servers. It dynamically changes the number and configuration of entity servers and calculation servers using dynamic load balancing algorithms during the simulation run. Simulation migration (not only migrating simulated entities, but may also migrating calculation or migrating VMs) is employed to perform the load balancing.

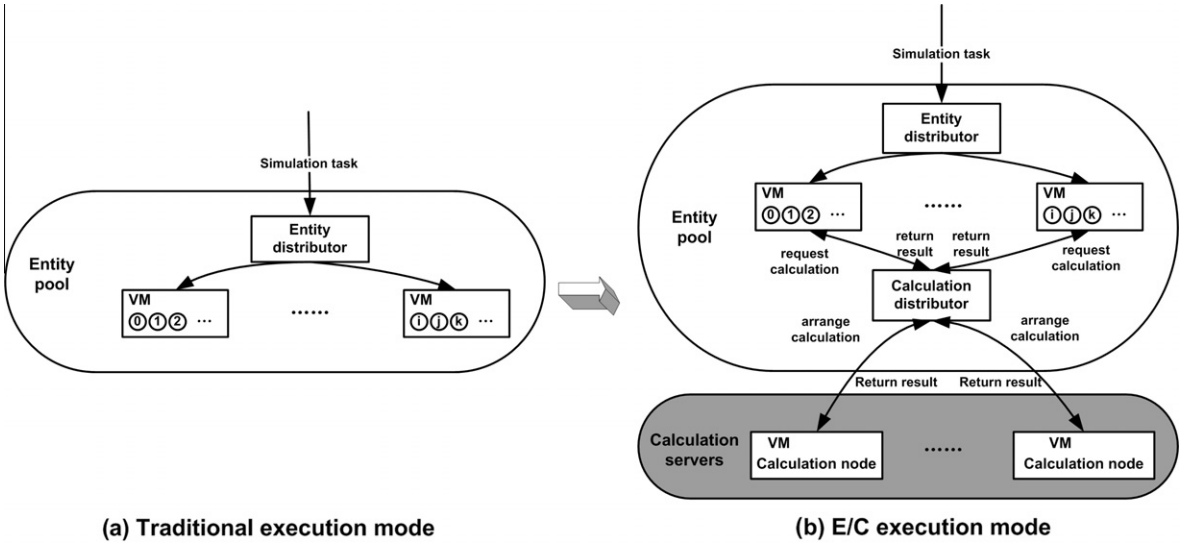


Fig. 4. The execution mode in CSim.

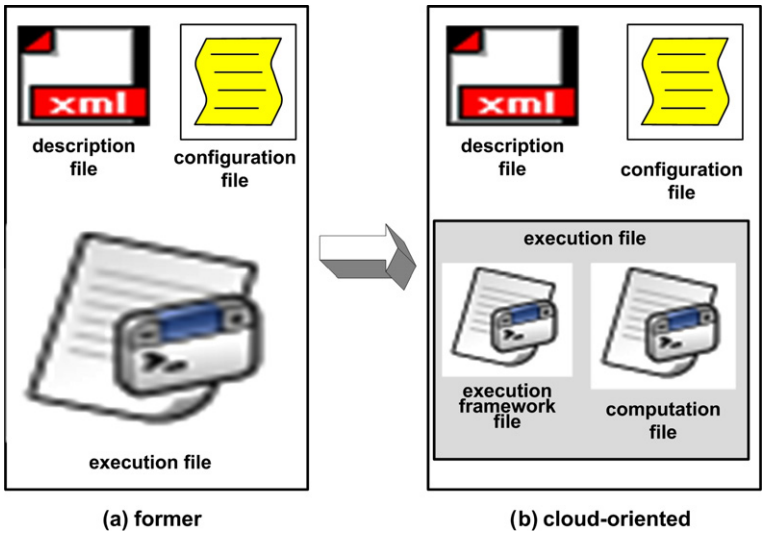


Fig. 5. The structure of cloud-oriented model.

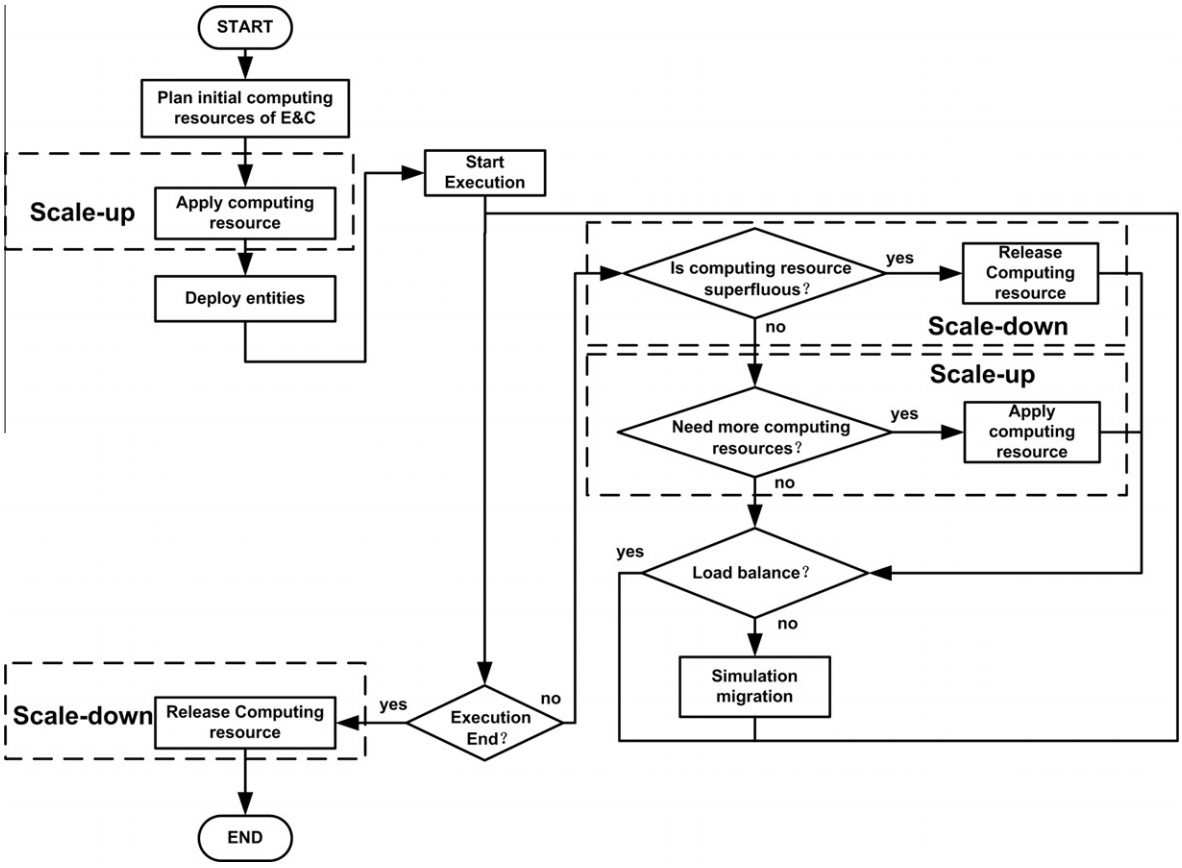


Fig. 6. The workflow of scalability.

3.2.4. PDES scheduling scheme in the resource allocator

Parallel program scheduling refers to two levels. One is the scheduling within a parallel application; the other one is among different parallel applications. Our scheduling method complies with the rules widely used in parallel application scheduling [33] while considering the features of PDES applications. PDES applications often show the following features:

Table 1
Time consumption in PM and VM (unit:second).

Node no.	$L = 20$, $n = 3$			$L = 20$, $n = 2$			$L = 20$, $n = 1$		
	PM	VM	LOSS (%)	PM	VM	LOSS (%)	PM	VM	LOSS (%)
1	252.696	261.076	3.21	85.932	88.855	3.29	82.606	85.452	3.33
2	157.831	161.815	2.74	52.734	54.315	2.91	40.751	41.964	2.89
3	157.053	160.373	2.07	51.980	53.176	2.25	27.457	28.034	2.06

- The loads of the processes change dynamically during the simulation run, thus simulation migration may be employed to cope with the unbalance within an application. The simulation migration is as always well supported by most PDES engines and the price for it is cheap. It has been widely used in the dynamic load balancing, fault-tolerance, rollback, etc.
- The processes within a multi-processes PDES application cannot consume the whole capacity of its host CPU due to the communication and synchronization among processes. Thus an opportunity for PDES workload consolidation appears.
- Simulation migration can also provides a powerful mean to the scheduling among different PDES applications to further improve the utilization of computing resources.

More specificcally, traditional load balancing approaches are employed when scheduling within a PDES application; a migration and consolidation based scheduling is used when performing PDES applications scheduling. Our initial evaluation shows that: in most cases, even without any information about the computing resource consumption pattern of a PDES workload, our migration and consolidation based scheduling can get better performance than EASY (Extensible Argonne Scheduling sYstem) [34]; in all cases, our scheduling outperforms FCFS (First-Come-First-Serve) very much. The consolidation method used in our scheduling will be introduced in Section 4.3.2. If CSim involves the public IaaS, other aspects (such as the budget constraint, the feature of the renting VM instances, the characteristic of the workload, the cost-to-performance efficiency [21] etc.) should be taken into account.

4. Guidelines to effectively utilize computing resources

Making good use of computing resources is a key to the successful application of CSim, it is the main goal of the scheduling in CSim. This section presents some new guidelines that could promote effective use of computing resources by some experimentations.

4.1. Experiment environment

The hardware platforms used in the experiments are: (1) A cluster consists of four nodes, each one is equipped with Intel pentium4 3.0G CPU and 1 GB RAM, interconnected by 1 GB Ethernet; (2) A desktop computer with Intel Core Duo E8400 dual core CPU at 3.00G and 3.4 GB RAM; (3) Amazon Elastic Compute Cloud (EC2) [35], a public IaaS that provides different types of VMs. The software configuration of each machine (both VMs and PMs) is CentOS (32-bit 2.6.18 linux kernel), mpich21.3.2p1, Xen 3.1 and KD-PaSEC which is a PEDS engine developed by system simulation Lab at National University of Defense Technology (China) [36].

The experiments employ PHOLD [37] which is a classical model in PDES for simulation. The test simulation system is consisted of 500 PHOLD model objects (namely simulated entities) which are denoted as $\{e_1, e_2, \ldots, e_{500}\}$. The simulation time is set to 300 s, and the simulation logic of each entity (e_i) are:

- Initialization rules: (1) Plan the events: generate an event whose timestamp is 1 s to each of these n entities in the entity set $\left\{e_{N_i^1}, e_{N_i^2}, \ldots e_{N_i^n}\right\}$, where: $N_i^j = \begin{cases} i+j, & if\ i+j < 500 \\ i+j-500, & otherwise \end{cases}$, $1 \leqslant j \leqslant n$ Bigger n means more complex interactions among the entities. (2) Calculate the next target to generate event to, where: $N_i^{next} = \begin{cases} N_i^n + 1, & if\ N_i^n + 1 < 500 \\ N_i^n + 1 - 500, & otherwise \end{cases}$.
- Event processing: (1) Process the event for L ms time consumption. In most of the cases, simulation computation only utilizes CPU and RAM, hence, this L ms load does not involve any disk I/O processing. (2) Generate an event whose timestamp is $CurrentTime + 1$ to $e_{N_i^{next}}$. (3) Calculate the next target to generate event to, where: $N_i^{next} = \begin{cases} N_i^{next} + 1, & if\ N_i^{next} + 1 < 500 \\ N_i^{next} + 1 - 500, & otherwise \end{cases}$.

4.2. Virtualization in private SIMaaS

4.2.1. Adoption of the virtualization technology

Virtualization technology multiplexes physical resources at the granularity of an entire OS and is able to provide performance isolation between them. It facilitates the management of computing resources and moreover increases the utilization

of computing resources. But there is a price to pay for this flexibility and efficiency–running a full OS is more heavy-weight than running a process, thus virtualization leads performance degradation to some extent. Although various experiments in the literature have evaluated the performance overheads of virtualization under different conditions [38–41]. This section aims at further convincing M&S practitioner to adopt the virtualization technology when building private simulation clouds.

This experiments are performed in the cluster aiming at finding performance loss caused by virtualization. Table 1 shows the time consumption (average time consumption in 3 runs, the same in the rest of the paper). The comparison of performance loss under conditions (with different event interaction complexity, VMs and PMs) is given in the table as well. The results show that the maximum loss in performance caused by virtualization is 3.33%, which is accordant with other performance overhead testing results mentioned in [38–41].

Such a low loss in performance for multiplexing physical resources is very valuable in multiple cases. For a simple instance, given two heterogeneous tasks, say, $Task_1$ (a Windows-based application) and $Task_2$ (a Linux-based application) as illustrated in Fig. 7. We assume that the two tasks are submitted at t_1 and t_2 and the time consumption of $Task_1$ is quit less than $t_2 - t_1$. Without the presence of the virtualization technology, at least two physical machines have to be equipped to fulfill them, but only one is enough by running VMs (with different OS images) on the same physical machine.

4.2.2. Tradeoff between performance and isolation

There is a price to pay for isolation by running a full OS other than running a process. The more the guest OSs the greater performance loss. We compare the performance of running multiple simulation runs in their own guest OS against running them on the same native OS.

The experiments are performed in the desktop machine, Table 2 shows the loss in performance. Results show that isolation causes heavier performance loss. In practice, if there is no isolation requirement, simulation tasks should be deployed in as less VMs as possible.

4.3. Consolidation in CSim

As mentioned in Section 3.2.4, the tasks often cannot consume the total capacity of the computing resources in practice. Hence, the opportunity for task consolidation appears in CSim. In a few cases, the computing resource comsumption pattern can be known, but in other cases, this information is hard to obtain.

4.3.1. Task consolidation with known resource usage pattern

We explore the consolidation with known resource usage pattern in this section. This kind of task often exists in SIMaaS (especially in MaaS and AaaS). The experiments are performed in the desktop machine, and four modified PHOLD models with $L = 20$, $n = 0$ are employed to simulate four tasks of this type. Their computing resource usage patterns are illustrated in Fig. 8. Seven simulation task arrangement policies are described as following:

- Policy #1: involves one physical machine, $Task_1$, $Task_2$, $Task_3$ and $Task_4$ are executed sequentially.
- Policy #2: involves one physical machine, $Task_1$, $Task_2$, $Task_3$ and $Task_4$ are executed concurrently.
- Policy #3: involves one physical machine, $Task_1$, $Task_2$ are executed concurrently firstly, and then $Task_3$, $Task_4$ are executed concurrently after $Task_1$, $Task_2$ are all completed.
- Policy #4: involves one physical machine, $Task_1$, $Task_4$ are executed concurrently firstly, and then $Task_2$, $Task_3$ are executed concurrently after $Task_1$, $Task_4$ are all completed.
- Policy #5: involves two physical machines, $Task_1$ and $Task_4$ are executed sequentially on one physical machine, $Task_2$ and $Task_3$ are executed sequentially on another physical machine at the same time.
- Policy #6: involves two physical machines, $Task_1$ and $Task_2$ are executed concurrently on one physical machine, $Task_3$ and $Task_4$ are executed concurrently on another physical machine at the same time.
- Policy #7: involves two physical machines, $Task_1$ and $Task_4$ are executed concurrently on one physical machine, $Task_2$ and $Task_3$ are executed concurrently on another physical machine at the same time.

Tables 3 and 4 show the time consumption under these arrangement policies. From them, if all the tasks have to complete in 1000*s*, then policy #2 which involves only one physical machine and consumes 742.256*s* is the best choice; if all the tasks have to complete in 700*s*, then policy #7 which involves two physical machines and consumes 399.473*s* is the best choice.

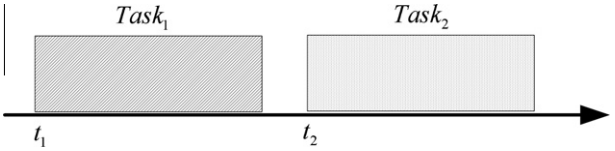
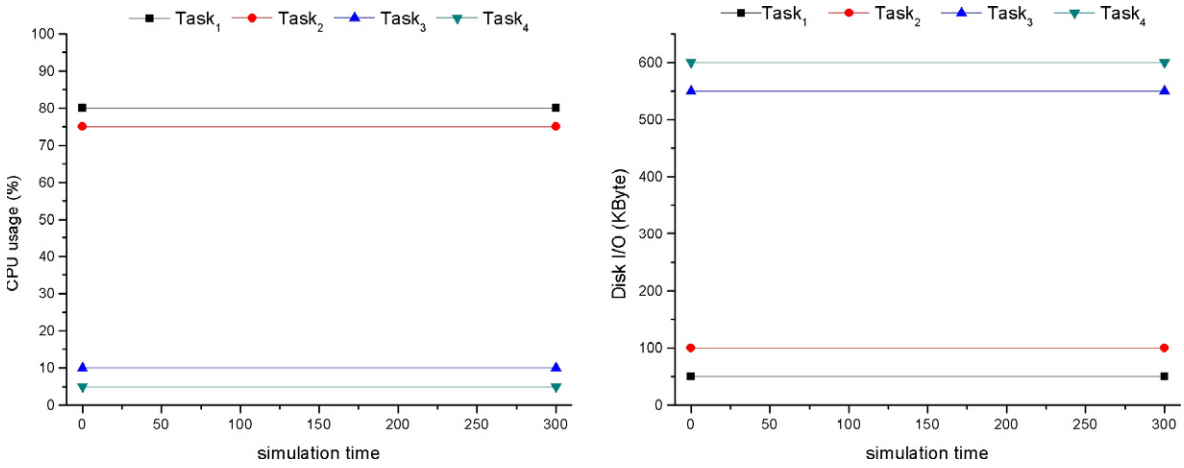


Fig. 7. An example of two tasks.

Table 2
Simultaneous runs on VM and PM (unit:second).

parameter	Run in PM	Run in VMs	Loss (%)
$L = 20, n = 5$	231.333	246.096	6.00
$L = 20, n = 8$	469.973	500.544	6.11
$L = 60, n = 3$	687.110	727.107	5.51
$L = 60, n = 5$	490.831	512.544	4.24



(a) CPU usage pattern of the tasks (b) Disk I/O pattern of the tasks

Fig. 8. Computing resource usage patterns of the tasks.

Table 3
Time consumption of different arrangements on one machine (unit: second).

Policy	Time consumption of $Task_1$	Time consumption of $Task_2$	Time consumption of $Task_3$	Time consumption of $Task_4$	Total time consumption
#1	114.241	125.372	342.193	378.231	960.037
#2	227.431	240.778	701.452	742.256	742.256
#3	202.745	213.862	652.771	673.261	887.123
#4	140.413	374.117	157.228	399.473	773.59

Table 4
Time consumption of different arrangements on two machines (unit:second).

Policy	Time consumption of PM#1	Time consumption of PM#2	Total time consumption
#5	114.241 + 378.231 = 492.472	125.372 + 342.193 = 467.565	492.472
#6	Max (202.745,213.862)=213.862	Max (652.771,673.261)=673.261	673.261
#7	Max (140.413,374.117)=374.117	Max (157.228,399.473)=399.473	399.473

Results show that reasonable task consolidation can significantly increase the utilization of computing resources while meeting the time constraint.

4.3.2. Task consolidation with unknown resource usage pattern

We investigate the consolidation without the resource usage pattern in this section. This kind of simulation task often exist in EaaS, for example, detailed computing resource usage pattern in most PDES applications is often hard to get in practice. The idea in such kind of task consolidation is that: we only focus on the control of CPU and leave other computing resource alone. This is reasonable because it captures the following two facts: (1) PDES is often a CPU-intensive application; (2) all the operations on other computing resources consume CPU cycles, thus the process with higher CPU priority also owns higher priority to use other computing resources.

In the experiments here, we introduce two more typical PDES programs which are picked from Section 2 (Program1, P1) and Section 6 (Program2, P2, which is an uneven workload) in [42] respectively. We create seven different applications out of

Table 5
Applications involved in the experiments.

Application (program)	Average processor utilization				Elapsed time (s)
	Processor 1 (%)	Processor 2 (%)	Processor 3 (%)	Node 4 (%)	
A1(P1)	99.0	...	...	...	302.3
A2(P1)	58.5	91.2	...	...	189.3
A3(P1)	60.5	36.0	60.5	...	190.0
A4(P2)	44.2	42.3	43.6	...	261.0
A5(P2)	64.8	62.8	62.9	...	761.5
A6(P2)	65.7	66.0	65.0	66.7	578.8
A7(P2)	71.0	69.8	70.0	71.0	779.8

Table 6
Application execution time comparison with VM collocation.

Application	w/o Collocation	Collocation w/o priority		Collocation w/ priority	
		fg	bg	fg	bg
A1	**302.3**	608	606	**307**	606
A2	**189.3**	379	380	**191**	375
A3	**190**	351	351	**203**	314
A4	**261**	397	395	**271**	405
A5	**761.5**	1271	1268	**789**	1350
A6	**578.8**	961	960	**590**	1011
A7	**779.8**	1333	1331	**794**	1414

Table 7
Time consumption under different execution modes (1) (unit:second).

Execution mode	$n = 8$				$n = 5$			
	$L = 10$	$L = 20$	$L = 60$	$L = 120$	$L = 10$	$L = 20$	$L = 60$	$L = 120$
1/0	361.894	691.237	1989.11	3958.01	228.127	425.434	1222.38	2475.63
2/0	230.663	430.899	1231.07	2853.31	144.214	269.678	769.178	1520.26
1/1	321.44	562.646	1304.45	2379.12	202.063	354.686	816.106	1483.38
3/0	229.061	425.537	1214.831	2400.58	142.221	265.326	759.452	1496.91
2/1	216.872	391.091	991.732	1832.92	135.382	244.497	619.877	1144.57
1/2	292.509	526.274	1007.39	1687.14	187.12	333.6	625.34	1077.93

the two programs and run each application using 1–4 processor(s), as shown in Table 5. In these applications, the only difference among A1, A2 and A3 is the number of processors used, the difference among A4, A5, A6 and A7 includes the number of processors used as well as parameters used in each application. The average processor utilization and the elapsed time of each application are also shown in the table.

For the processor pool, we partition each processor into two tiers by pinning two Virtual CPUs (VCPUs) on the processor. These two VCPUs belong to two different VMs. By running processes in the two VMs, each processor can support two processes from different applications at the same time, thus may improve the CPU utilization of the physical processor. We use the Credit Scheduler [43] in Xen to control how collocated VMs share the physical CPU.

In the first experiment, we collocate two VMs in each physical processor and give these VMs the same CPU priority (default setting in Xen). In the second experiment, one VM is assigned a higher CPU priority (by using Credit Scheduler to set the VM a weight of 10,000) and the other is assigned a lower CPU priority (a weight of 1). We call the high priority one foreground VM (*fg*) and the low priority one background VM (*bg*). In this setting, the background VM only runs when the foreground VM is idle. We run the same application in both foreground VM and background VM. When the application in the foreground VM finishes, the background VM has a chance to use the computing resources. Table 6 shows the results of the two experiments. Further study on the impact of task consolidation to the performance of applications running in the foreground VMs shows the same pattern as well.

Throughout the experiments, we made the following observations:

- Priority-based VM collocation incurs trivial performance impact (less than 4% performance degradation) to applications running in the high priority VMs. It brings better resource utilization than the one that without priority control. It also incurs lower context switching cost and network I/O contention between foreground and background VMs.

Table 8
Time consumption under different execution modes (2) (unit:second).

Execution mode	$n = 3$				$n = 1$			
	$L = 10$	$L = 20$	$L = 60$	$L = 120$	$L = 10$	$L = 20$	$L = 60$	$L = 120$
1/0	139.38	261.076	749.652	1450.8	46.893	85.452	250.279	488.812
2/0	86.834	161.815	461.56s	911.407	22.521	41.964	126.576	247.127
1/1	121.998	212.612	487.842	896.992	41.259	71.675	168.823	299.061
3/0	85.31	160.373	453.859	869.493	14.897	28.034s	81.692	163.413
2/1	81.757	156.935	372.948	687.261	21.311	38.592	100.031	213.402
1/2	112.967	201.031	374.761	633.305	37.238	67.008	126.5	182.797

- When a foreground VM runs an application with high CPU utilization, collocating a VM to run in its background does not benefit either the foreground or the background application due to that context switching incurs overhead and the background VM has very small chance to get physical resource to run.
- When a foreground VM runs an application with low CPU utilization, the application running in the collocated background VM can get significant share of physical resources to run. Thus a balanced VMs match is helpful to resource allocation.

In the experiments here we use the CPU scheduler in Xen to control the CPU priority of the process, we can also use the process priority mechanism in common OS to perform the work. The observations are taken into account when the resource allocator makes a scheduling decision.

4.4. Selection of the execution mode

The experiments here are performed on the cluster, aiming at analyzing the effect of the execution mode on the performance. Tables 7 and 8 show the time consumption under different conditions.

Fig. 9 presents the speedup ratio over L under different execution modes. It can be observed that: (1) the speedup ratio at 1/1 mode is almost the same as 3/0 mode with $\{n = 8, l = 120\}$, $\{n = 5, l = 120\}$, $\{n = 3, l = 120\}$; (2) 1/1 mode that involves only two machines is more efficient; (3) 3/0 mode is much more efficient when $n = 0$, but 1/2 and 2/1 mode are more efficient when $n \neq 0$. In summary, for different interaction complexity and workload, the performance demonstrates significant difference. So the number and configuration of entity server and calculation server should be well estimated for a simulation run. In addition, the entity migration, calculation migration and VM migration should be carefully selected as well after relocating the entity and calculation server.

4.5. Study of the performance on the public IaaS

Sometimes, CSim needs to hire computing resources from the Public IaaS. Public IaaS providers usually provide VMs with fixed configurations under different price. However, it is not true that high price VM can deliver the expected performance. Hence when running simulations on the public IaaS, the computing efficiency in different VMs must be well understood. Suitable VMs must be carefully chosen based on the features of the simulation task to optimize the ratio between the cost and the execution time.

The experiments are performed on EC2. A great deal of insight work is required to obtain the entire knowledge (such as the CPU performance, the memory speed, the Disk I/O performance, the network bandwidth, and the variance of these computing resources) about practicing simulation upon EC2. Information in [44–47] may be helpful for this purpose. In this paper, we employed EC2 small instance (m1.small) and High-CPU medium instance (c1.medium) for example. Table 9 presents the time cost using these two different VM instances when $n = 8, L = 10$. The results indicate: (1) $\frac{1634.82s}{178.631s} = 20.791 \gg 2 = \frac{\$0.17}{\$0.085}$, it means that the cost and execution time of c1.medium is much better than that of m1.small; (2) $\frac{1634.82s}{1562.76s} = 1.046 \ll 2 = \frac{2instances}{1instance}$, $\frac{178.631s}{159.231s} = 1.122 \ll 2 = \frac{2instances}{1instance}$, it means that it may not be worthy to increase the number of VMs to reduce the execution time in both c1.medium and m1.small situations. Aspects such as the budget constraint, the time constraint and the affinity (between a workload and the VM instances), the cost-to-performance efficiency should be taken into account when making resource allocation. [21,48] have proposed some valuable considerations on this topic.

5. Related work

The research of CSim is still at its early stage at present, and it is a new research area in M&S. Only little research related to CSim can be found, and most of them focused on specific aspects of it.

Bohu Li et al. [49] proposed a primitive thinking about CSim based on Grid-based simulation, and summarized 12 key technologies for the implement of CSim. DU Jin [50] presented an idea of introducing cloud computing into training simulation. CIMdata Inc [51] put forward an approach using cloud computing for simulation and analysis from a commercial

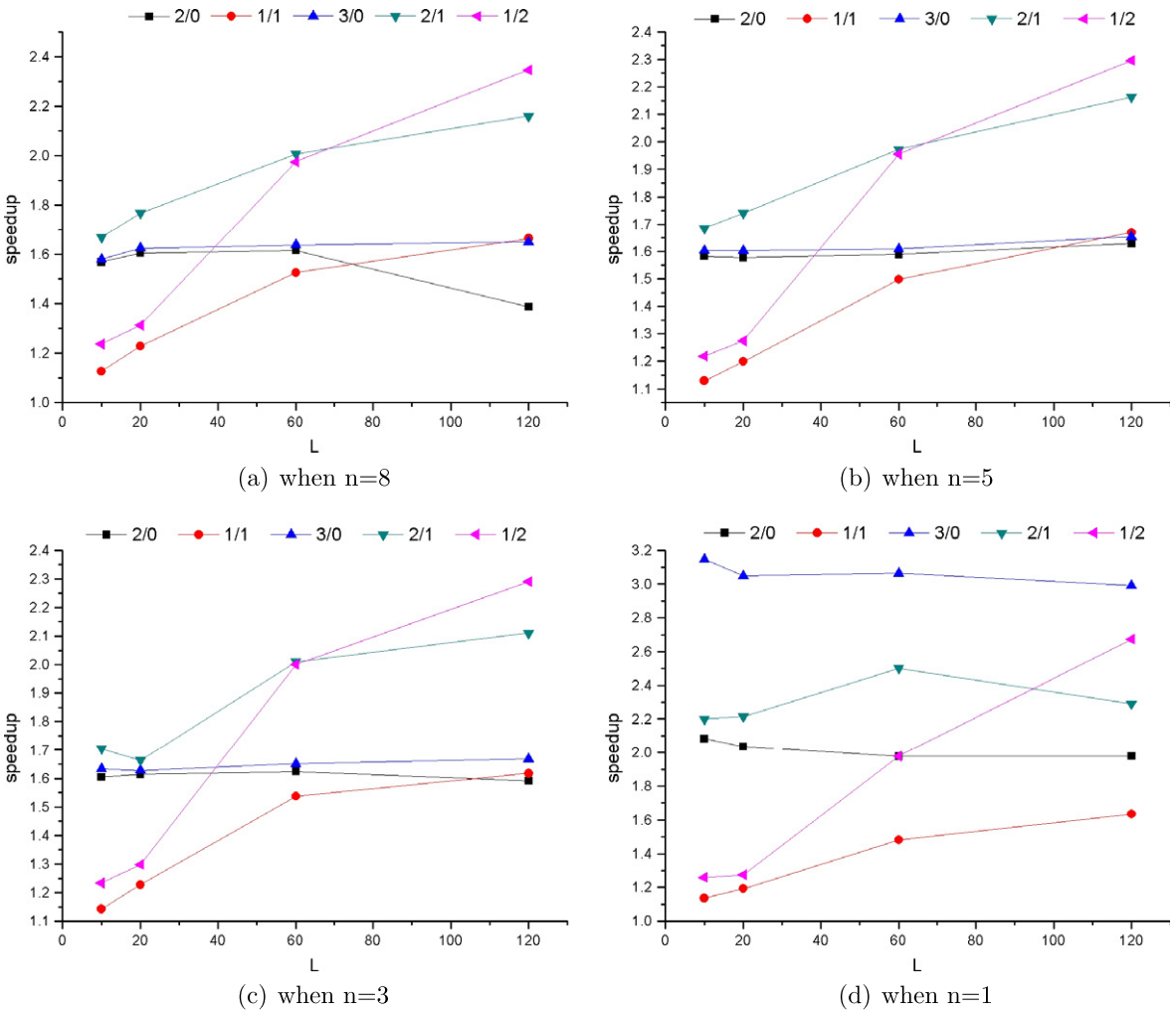


Fig. 9. Speedup ratio over L under different execution modes.

Table 9
Time consumption when using different EC2 instances (unit:second).

Instance no.	m1.small ($0.085 per hour)	c1.medium ($0.17 per hour)
1	1634.82	178.631
2	1562.76	159.231
3	1561.34	156.825

product design's point of view, insight analysis of the advantages in the cloud-based simulation was also given. Richard M.Fujimoto et al. [16] highlighted the benefits and important challenges associated with executing Parallel and Distributed Simulation (PADS) in cloud environments. Some possible solutions were suggested in the paper as well. In addition, the paper also reviewed a Master/Worker simulation execution paradigm. Asad Waqar Malik et al. [17] discussed a cloud architecture for Time Warp (TW-SMIP) which is an optimistic synchronization protocol intended to address concerns about interference and communication delays that are inherent in cloud computing environments. Gabriele DAngelo [48] discussed some points on PADS in the public cloud. In particular, the need for new evaluation metrics and adaptive mechanisms proposed for multi-agent systems were described in the paper. Feng et al. [52] proposed their work on remodeling traditional RTI software to be with PaaS architecture. The software construction, sub-component functions and the method of interacting based on the remodeled RTI software on the WAN were studied in the paper. Jason Scott Bolin [8] gave a detailed use case analysis for adopting cloud computing in army test and evaluation. Yoshiki Kato et al. [53] described a successful use case which introduced how to use the cloud-based M&S to assist decision-making in politics and economics. Eva Pajorov et al. [54] discussed how to use cloud computing for the simulation display.

Although there are only a few papers about CSim in the past, encouraging achievements about adopting key technologies of cloud computing (such as Web service, SOA and Grid) into M&S have been proposed. Bo Hu Li et al. [55] presented the design and implement of a simulation grid prototype named Cosim-Grid0.1 based on the concept of Grid. James Byrne et al. [56] gave a review of the area of Web-Based simulation (WBS), explored the advantages and disadvantages of WBS over classical simulation systems. A classification of different sub- and related-areas of WBS, an exploration of technologies that enable WBS, and the evolution of the Web in terms of its relationship to WBS were discussed in the paper as well. [57–59] detailed the use of SOA in M&S. In which, [57] put forward a concept of simulation service system based-on SOA using HLA (High Level Architecture) as an example; [58] used the Java platform to implement DEVS (Discrete Event Specification) over a SOA framework; the design of a service-oriented simulation software framework was discussed in [59]. Bo Hu Li et al. [60] presented a virtualized Grid-based simulation platform which adopted virtualization technology in the grid-based simulation.

6. Conclusions

This paper describes the work of planting our existing simulation software into the cloud. More concretely, this paper discusses the needs for CSim, presents the design and implementation of CSim, and emphasises on some guidelines for the effective utilization of computing resources.

From the user's point of view, CSim brings many benefits to the consumers. One of the most important facts is that CSim makes the use of simulation techniques easier and with lower cost, hence CSim can extend the applicability of simulation techniques. We believe that CSim will become a major trend in M&S.

From the simulation cloud provider's point of view, CSim is the integration of some mature technologies such as simulation techniques, virtualization technology and web service technologies. The difficulty and risk of using it are low. Effective utilization of computing resources is a key goal of the scheduling in CSim. During the scheduling within a simulation run, not only the entity migration but also the calculation migration and the VM migration can be performed to achieve load balance; virtualization and workload consolidation provide powerful means of optimizing resource allocation for the scheduling among simulation runs.

7. Open issues and future work

This paper presents the initial research in CSim. A CSim prototype system as well as the issues to be considered in the development of simulation services are proposed. Some directions of our future research are summarized as follows:

- Optimistic time synchronization mechanisms for CSim [17]: The decrease in I/O performance of VMs in the cloud [45] increases the time cost in I/O operations (such as entity status saving and recovery) in CSim; the shortage of network bandwidth results in decrease in the performance of optimistic time policy. Therefore, optimistic time synchronization algorithms should be adjusted according to the features of clouds.
- Load balancing in the simulation run: Due to the changes in execution mode and the adoption of virtualization, new balancing techniques such as calculation migration and virtual machine migration will increase the efficiency as well as the complexity of the existing load balance algorithms.
- Scheduling of simulation tasks in the public IaaS: the scheduling of parallel applications in the public cloud is a tricky question in CSim, the problem of how to make tradeoff between the execution time and the budget constraint requires deeper study. Seminal discussions proposed in [21,48] are valuable for this purpose.
- Execution in GPU/CPU based computing framework [61]: GPU/CPU mixed architecture is the mainstream in future supercomputer, hence has good prospect in the future cloud computing. Some of supercomputers [62] adopted such architecture. Moreover, EC2 also released the GPU cluster instance at the end of 2010 [35].
- Implementation and summary of model as service, algorithm as service: a lot of widely used algorithms and models are available in many domains. Gathering these models and algorithms and delivering them as services will increase the reusability of them and the development efficiency as well.

Acknowledgement

This work was supported by NSFC awards 91024030. A lot of thanks should be given to referees and editors, their valuable comments greatly improved the quality of the manuscript. The authors would also like to express their sincere appreciation to Yefeng Wang and Ge Huang for their polishing work.

References

[1] J. Dongarra, D. Reed, R. Bajcsy, M. Fernandez, J. Griffiths, R. Mott, C. Johnson, A. Inouye, W. Miner, M. Matzke, et al., Computational Science: Ensuring America's Competitiveness, 2005.
[2] K. Delic, M. Walker, Emergence of the academic computing clouds, Ubiquity 2008 (2008) 1.

[3] Y. Zhang, General Purpose Parallel Discrete Event Simulation Environment and the Study of Relevant Techniques, Ph.D. Thesis, National University of Defense Technology, 2008. (in Chinese).
[4] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications 1 (2010) 7–18.
[5] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., Above the Clouds: A Berkeley View of Cloud Computing, EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28 (2009).
[6] CCIDConsulting, White Paper of Chinese Cloud Computing Industry. <http://www.techweb.com.cn/special/download/cloudbook.pdf> (in Chinese, accessed 2011).
[7] E. Marks, M. Bell, Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology, 2006.
[8] J. Bolin, Use Case Analysis for Adopting Cloud Computing in Army Test and Evaluation, Technical Report, DTIC Document, 2010.
[9] D.C. Team, The US Army is Developing its Own Private Cloud. <http://www.dreamsimplicity.com/component/content/article/1179.pdf> (accessed 2011).
[10] Cabinetoffice, Gcloud Programme. <http://www.cabinetoffice.gov.uk/resource-library/uk-government-ict-strategy-resources> (accessed 2011).
[11] M. of Internal Affairs, Communications, Digital Japan Creation Project (ICT Hatoyama Plan): Outline. <http://www.soumu.go.jp/main_sosiki/joho_tsusin/eng/Releases/Topics/pdf/090406_1.pdf> (accessed 2011).
[12] U.N.E. Programme, Overview of the Republic of Koreas National Strategy for Green Growth. <http://www.unep.org/PDF/PressReleases/201004_UNEP_NATIONAL_STRATEGY.pdf> (accessed 2011).
[13] Economy, I.C. of Beijing, Auspicious Cloud Project. <http://210.75.193.70/zdzttj/sjxwwj/sjxwwj/201111/t20111112_20239.htm> (in Chinese, accessed 2011).
[14] IBM, Nato and IBM to Use Cloud Technology for Improved Command and Control. <http://www-03.ibm.com/press/us/en/pressrelease/33285.wss> (accessed 2011).
[15] K. Jeffery, H. Schubert, B. Neidecker-Lutz, The Future of Cloud Computing Opportunities for European Cloud Computing Beyond 2010, Expert Group Report, Public Version 1, 2010.
[16] R. Fujimoto, A. Malik, A. Park, Parallel and distributed simulation in the cloud, International Simulation Magazine, Society for Modeling and Simulation 1 (2010).
[17] A. Malik, A. Park, R. Fujimoto, Optimistic synchronization of parallel simulations in cloud computing environments, IEEE International Conference on Cloud Computing, 2009. CLOUD'09, IEEE, 2009, pp. 49–56.
[18] Z. Ke, Q. Xiao-gang, P. Chun-guang, C. Bin, H. Ke-di, Design and implementation of distributed simulation experiment management system, Journal of System Simulation (2008) (in Chinese).
[19] R. Smith, Simulation in the 21st Century. <http://www.peostri.army.mil/CTO/FILES/RSmith_Simulation21.pdf> (accessed 2011).
[20] Z. Papazachos, H. Karatza, The impact of task service time variability on gang scheduling performance in a two-cluster system, Simulation Modelling Practice and Theory 17 (2009) 1276–1289.
[21] I. Moschakis, H. Karatza, Evaluation of gang scheduling performance and cost in a cloud computing system, Journal of Supercomputing (2011) 1–18.
[22] Opennebula. <http://www.opennebula.org/start> (accessed 2011).
[23] Eucalyptus. <http://www.eucalyptus.com/> (accessed 2011).
[24] Xen. <http://www.eucalyptus.com/> (accessed 2011).
[25] Libvirt. <http://libvirt.org/> (accessed 2011).
[26] Virt-Manager. <http://virt-manager.et.redhat.com/> (accessed 2011).
[27] D. Nurm, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, The eucalyptus open-source cloud-computing system, 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009. CCGRID09, IEEE, 2009, pp. 124–131.
[28] J. Ru-sheng, Q. Hai-quan, Q. Xiao-gang, H. Ke-di, Research on design and application of HLA simulation result database, Journal of System Simulation 2 (2006) (in Chinese).
[29] W. Ming, Q. Xiao-gang, L. Bao-hong, Description Research for Simulation Resource of HLA, Computer (2007) 11 (in Chinese).
[30] R. Barga, Introduction to cloud computing architecture, ACM Sigact News 40 (2009).
[31] B. Zeigler, H. Praehofer, T. Kim, Theory of Modeling and Simulation, vol. 100, Academic Press, 2000.
[32] X. Liu, Q. He, Z. Zhong, C. Peng, A rapid way of developing bom-based-model, Second International Conference on Computer Modeling and Simulation, 2010. ICCMS'10, vol. 2, IEEE, 2010, pp. 508–512.
[33] D. Feitelson, L. Rudolph, U. Schwiegelshohn, K. Sevcik, P. Wong, Theory and practice in parallel job scheduling, in: Job Scheduling Strategies for Parallel Processing, Springer, pp. 1–34.
[34] A. Mu'alem, D. Feitelson, Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling, IEEE Transactions on Parallel and Distributed Systems 12 (2001) 529–543.
[35] Amazon AWS. <http://aws.amazon.com/> (accessed 2011).
[36] Q.H. et al., Research and implement of parallel simulation engine for component based-on bom, Journal of National University of Defense Technology (in press). (in Chinese).
[37] R. Fujimoto, Performance of Time Warp Under Synthetic Workloads, 1990.
[38] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, ACM, pp. 164–177.
[39] A. Menon, J.R. Santos, Y. Turner, G.J. Janakiraman, W. Zwaenepoel, Diagnosing Performance Overheads in the Xen Virtual Machine Environment-Network. <http://www.usenix.org/events/vee05/full_papers/p13-menon.pdf> (accessed 2011).
[40] P. Padala, X. Zhu, Z. Wang, S. Singhal, K. Shin, Performance Evaluation of Virtualization Technologies for Server Consolidation, HP Labs Technical Report, 2007.
[41] G. Diwaker, G.R.C. Ludmila, Xenmon: Qos Monitoring and Performance Profiling Tool. <http://www.hpl.hp.com/techreports/2005/HPL-2005-187.pdf> (accessed 2011).
[42] Y. Lin, Parallelism analyzers for parallel discrete event simulation, ACM Transactions on Modeling and Computer Simulation (TOMACS) 2 (1992) 239–264.
[43] CreditScheduler. <http://wiki.xen.org/xenwiki/CreditScheduler>.
[44] C. Evangelinos, C.N. Hill, Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's ec2, Cloud Computing and Its Applications 2008 (CCA'08). <http://www.cca08.org/papers/Paper34-Chris-Hill.pdf> (accessed 2012).
[45] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. Wasserman, N. Wright, Performance analysis of high performance computing applications on the amazon web services cloud, in: 2nd IEEE International Conference on Cloud Computing Technology and Science, IEEE, pp. 159–168.
[46] R. Masud, High Performance Computing with Clouds. <http://ix.cs.uoregon.edu/raihan/HPC_with_Clouds_Raihan_Masud.pdf> (accessed 2011).
[47] J. Schad, J. Dittrich, J. Quiane-Ruiz, Runtime measurements in the cloud: observing, analyzing, and reducing variance, Proceedings of the VLDB Endowment 3 (2010) 460–471.
[48] G. D'Angelo, Parallel and distributed simulation from many cores to the public cloud, 2011 International Conference on High Performance Computing and Simulation (HPCS), IEEE, 2011, pp. 14–23.
[49] L. Bo-hu, C. Xu-dong, H. Bao-cun, L. Tan, Z. Ya-bin, Y. Hai-yan, H. Jun, D. Yan-qiang, H. Ji-jie, S. Chang-feng, et al, Networked modeling and simulation platform based on concept of cloud computing-cloud simulation platform, Journal of System Simulation 21 (2009) (in Chinese).
[50] D. Jin, The application research of cloud computing in military simulation, Computer Knowledge and Technology 6 (2010) (in Chinese).
[51] Cimdata, Using the Cloud for Simulation and Analysis – A New Approach. <http://www.cimdata.com/publications/pdf/Commentary_Autodesk%20CAE%20and%20Cloud%207Dec2010.pdf> (accessed 2011).

[52] S. Feng, Y. Di, Z. Meng, et al, Remodeling traditional rti software to be with paas architecture, 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 1, IEEE, 2010, pp. 511–515.
[53] Y. Kato, H. Yamaki, Y. Asai, Gpgcloud: model sharing and execution environment service for simulation of international politics and economics, Principles of Practice in Multi-Agent Systems (2009) 616–623.
[54] E. Pajorová, L. Hluchỳ, Complicated simulation visualization based on grid and cloud computing, Cooperative Design, Visualization, and Engineering (2010) 211–217.
[55] B. Li, X. Chai, Y. Di, H. Yu, Z. Du, X. Peng, Research on service oriented simulation grid, in: Autonomous Decentralized Systems, 2005. ISADS 2005. Proceedings, IEEE, pp. 7–14.
[56] J. Byrne, C. Heavey, P. Byrne, A review of web-based simulation and supporting tools, Simulation Modelling Practice and Theory 18 (2010) 253–276.
[57] H. Qiang, H. Jian-guo, H. Jian, A simulation service system based on soa, Computer Simulation (2007) 05.
[58] S. Mittal, J. Risco-Martín, B. Zeigler, Devs/soa: a cross-platform framework for net-centric modeling and simulation in devs unified process, Simulation 85 (2009) 419.
[59] G. Shao, R. McGraw, Service-oriented simulations for enhancing situation awareness, in: Proceedings of the 2009 Spring Simulation Multiconference, Society for Computer Simulation International, pp. 48.
[60] H. Liu, H. Su, Y. Zhang, B. Hou, L. Guo, X. Chai, S. Zhan, Study on virtualization-based simulation grid, 2010 International Conference on Measuring Technology and Mechatronics Automation, IEEE, 2009, pp. 685–689.
[61] H. Park, P. Fishwick, A gpu-based application framework supporting fast discrete-event simulation, Simulation 86 (2010) 613.
[62] Top500 Supercomputer Sites. <http://www.top500.org/site/systems/3154> (accessed 2011).