Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 506432, 11 pages
http://dx.doi.org/10.1155/2015/506432

Research Article

A Load Balancing Scheme Using Federate Migration Based on Virtual Machines for Cloud Simulations

Xiao Song, Yaofei Ma, and Da Teng

Science and Technology on Aircraft Control Laboratory, School of Automation Science, Beihang University, Beijing 100191, China

Correspondence should be addressed to Yaofei Ma; mayaofeibuaa@163.com

Received 4 June 2014; Revised 18 September 2014; Accepted 20 September 2014

Academic Editor: Minrui Fei

A maturing and promising technology, Cloud computing can benefit large-scale simulations by providing on-demand, anywhere simulation services to users. In order to enable multitask and multiuser simulation systems with Cloud computing, Cloud simulation platform (CSP) was proposed and developed. To use key techniques of Cloud computing such as virtualization to promote the running efficiency of large-scale military HLA systems, this paper proposes a new type of federate container, virtual machine (VM), and its dynamic migration algorithm considering both computation and communication cost. Experiments show that the migration scheme effectively improves the running efficiency of HLA system when the distributed system is not saturated.

1. Introduction

Nowadays, Cloud computing has been recognized as a new revolution in the IT industry. Key techniques in Cloud computing such as virtualization technology, automatic deployment, resource management, Web services, SOA, high performance I/O, and high-speed internet are well-developed and have been widely applied in various domains. Accordingly in the area of modeling and simulation (M&S), *Cloud-based computer simulation* (CSim) [1–6] has been proposed and designed to provide users with better solutions to solve M&S problems including large capital outlays in hardware but low utilization, high complexity in building simulation system, and high labor cost in simulation software maintenance.

Currently CSim research is at its preliminary stage and the pioneering works can be divided into three categories.

(i) CSim *framework* and how to plant existing simulation software into the cloud: Liu et al. [1] presents the process of deploying existing Parallel Discrete-Event Simulation (PDES) engine into the cloud, the deployment includes adjusting the structure of the model to fit the features of Cloud computing, developing the simulation execution mode, adding the horizontal scalability module to achieve service scalability, and using the resource allocator. Li et al. [2] proposed Cloud simulation platform and its prototype architecture, which implement new techniques including HLA/RTI (runtime infrastructure, software implementation of HLA) technique based on Web, resource dynamic management middleware technique based on virtualization technique.

(ii) Optimistic *time advancement algorithms* in the cloud: Jafer et al. [3] presents the state of the art in PDES, and it summarizes current research of PDES in the clouds and hardware acceleration. In order to address concerns about interference and communication delays that are inherent in Cloud computing environments, Fujimoto et al. [4, 5] consecutively studied parallel and distributed simulation in the cloud focusing on optimistic parallel simulation advancement approach named "time warp straggler message identification protocol (TW-SMIP)."

(iii) *Cloud agents* and web-based simulations: Jávor and Fur [6] proposes to solve high complexity problems using advanced Web services and Cloud computing techniques. It designs a framework in which cloud agents are composed of Web services integrating complex agent elements including large databases and different novel concepts of inference engines.

Meanwhile, one of the most important M&S standards, high level architecture (HLA) is an essential IEEE simulation interoperability standard in the area of distributed and parallel simulations. And IEEE has developed HLA implementation standards such as HLA-evolved [7–9]. With HLA and its implementation program runtime infrastructure (RTI), a lot of HLA compatible simulation systems have been developed including large-scale military HLA applications such as ModSAF [10], CCTT SAF [11], OneSAF [12, 13], and COSIM [14]. These large-scale simulations are usually hardware-consuming and software-intensive as they contain numerous models of environments and entities. Grid technology [15–17] has been used to solve the resource allocation and scheduling problem, but Grid is not fully used while cloud is more supported by many international companies such as Amazon [18], Google [19], and Softlayer [20].

As such we propose to use Cloud computing techniques to develop simulation systems, to solve the high cost and overlapped hardware and software resources problem with cloud. Cloud computing infrastructure offers numerous benefits such as reconfigurable dynamic resources and unified and simplified access to resources.

Although the works in [1–6] explored new paradigms of CSim, few papers addressed the specific steps of how to run HLA simulations in cloud, especially virtualization based large-scale military applications. Therefore, in this paper, we mainly study how to run existing large-scale HLA/RTI simulations with cloud and how to run them more efficiently.

The paper is structured as follows. In Section 2, key technology of Cloud computing, virtualization and HLA load balancing strategies are introduced and analyzed. Core part of the contribution, Section 3 introduces the proposed virtualization based HLA federation framework and runtime dynamic migration algorithm. Section 4 shows designed validation experiments and discusses results. Finally, conclusions are drawn and future works are discussed.

2. Related Work

In this section, virtualization related technologies and load balancing approaches are studied and analyzed.

2.1. Virtualization Technology and HLA Simulations. In cloud, *Virtualization* technology (VT) encapsulates the simulation application models and software packages to a relatively independent execution environment, VM, and deploys VMs to the underlying *Physical Machines (PM, or hosts)* [21]. Furthermore, by the segmentation in space and the time-sharing in time [22, 23], VT encapsulates the distribution and heterogeneity of the hardware resources, allowing multiple virtual machines to run simultaneously on one host as well as the migration of VMs among different hosts. In this way VT greatly improves the resource utilization and quality of service and is gradually implemented in the medium-scale and large-scale distributed simulation platform.

One of the key virtualization technologies is VM monitor (VMM), which connects applications and VMs with underlying hardware to control the running of multiple operating systems. VMM decouples the software from the hardware by inserting a layer of interaction between the software running in the VM and hardware, so that VMM could obtain direct control of how operating systems in VMs access underlying hardware resources [24]. The design of VMM also makes it possible to implement the functions like live migration [25] and security that are difficult to achieve in modern operating systems. Through VMM, computing resources can be encapsulated into a collection of VMs according to user's requirements, and VMs can be remapped and replicated easily. In addition, the state of VMs can be suspended and resumed at any time, and the running context can be rolled back to a checkpoint. Thus, virtualization technology provides the ability to consolidate management of distributed and heterogeneous resources by means of decoupling software and hardware, transparent access, encapsulation, live migration, and resources isolation.

Compared with the federate migration method of [17, 26], the main difference is that we use *Virtual Machines* (VMs) to be containers of federates. From the perspective of practical implementation, using cloud-based VMs to handle the resource management and dynamic migration of HLA simulation systems has the following merits compared with previous approaches.

(i) It saves programming efforts to implement migration of federates with the support of the virtualization middleware interfaces developed by many business or open-source VM vendors such as VMware [27], Opennebula [28], Eucalyptus [29], and Xen [30].

(ii) VM live (or online) migration technique makes it feasible to realize high efficient federate migration. Previous works show that VM live migration downtime is mainly proportional to the amount of the "writable working set" and could have downtime as low as 60 to 200 ms [31, 32].

(iii) VM encapsulating HLA application software packages such as RTI could be copied easily. This greatly saves the time and efforts simulation practitioners spend to configure the operational system and software environment, which is often a tricky and tedious process especially during the large-scale simulation applications deployment phase.

Therefore, it is more convenient for simulationists to develop and run large-scale simulations if we adapt the virtualization technology. That is why we study cloud-based HLA simulation in this paper. However, when we are ensured that we could use VMs to help us handle the federate migrations, we also encounter the problems that using VM to encapsulate federates create additional overhead. Some works [1, 33, 34] have designed experiments to find performance loss caused by virtualization. They found that the applications (with different event interaction complexity) ran in VM have the maximum loss of 3.33% compared to ran in Physical Machine. Although the performance loss is not high, we still have to take more measures to make HLA federation run faster. One effective way is to improve current load balancing algorithms with better heuristics which considers HLA communication cost among federates.

2.2. Load Balance Strategy in HLA and Beyond. A famous developing IEEE standard in Modeling and Simulation (M&S) society, HLA is also gradually known for its slow running characteristics. One important reason is that HLA is short of load-balance methods [17, 26, 35–38] to make the heavy load host run faster while all the other nodes must wait until the slowest federate advances so they can move to the next safe lookahead.

Current studies of HLA load balance techniques could be divided into two categories.

(i) Load Balance Algorithms for Distributed and Parallel Systems, Including HLA. Load balance strategies for distributed and parallel systems are basically adaptable for HLA, which is designed to provide a common set of framework and rules for distributed simulation. In load balance algorithms, the core concerns are computing and communication costs [17, 26, 39–43]. Our algorithm has been influenced by earlier works in three main areas: heuristics, computation, and communication cost.

Most dynamic load balancing works use heuristic algorithms because of its NP-hard nature [39]. Paper [40] studied heuristic methods for dynamic load balancing in a message-passing supercomputer. It uses easy-to-implement heuristics (choosing a node with minimum load as the migration destination node) and variable threshold in migrating processes among the multicomputer nodes. Genetic algorithms are used in [41] to dynamically distribute simulation tasks and compared with a first fit algorithm and a random allocation scheme. In this work, the test results show that genetic method outperforms the other two both in completion time and average processor utilization. In paper [39], adaptive load threshold is emphasized to suit the changing load on the system, and its basic strategy is also to ensure that heavily loaded node is balanced first with lightly loaded node. Simulated annealing algorithm is addressed in paper [42], which transforms the problem into a NP-complete problem named degree-constrained minimum spanning tree. This approach needs to adjust the heuristic factors according to the practical applications. These works use heuristics as an essential approach, which is also our basic strategy because it is simple, practical, and time-saving, especially in large-scale distributed simulation. However, most of the traditional works are one-by-one migration, while in our approach it is designed to migrate a set of interaction-aware VMs to save the cost of migration procedure latency and communication.

Moreover, papers [17, 26] study dynamic load balancing using Grid services for HLA-based large-scale simulations. These pioneering works store all the resources in a queue and organize them in ascendant order, and the migration pairs are assembled with both extremities. The algorithm is practical and simple but it does not consider the communication cost. Fujimoto et al. in [4, 5] addressed "Network traffic and communication delays are significant in current implementation of Cloud computing infrastructure." Thus, for the cloud-based large-scale simulations, communication cost must be taken into account in dynamic HLA load balancing.

All the abovementioned works are meaningful, but HLA simulation systems have some more distinct characteristics which are neglected by these researches. For instance, all interactions among HLA federates are mainly modeled and implemented with interaction classes and object classes. In most cases of military simulations, the former classes are *stochastic* and the latter classes are *periodic*. For example, the status data describing the position of a tank in Cartesian coordinates would be modeled as an object class and its instance data is updated by the tank entity at every time step. The data describing the fire of a tank entity would be modeled as an interaction class and its instance data is sent when the tank fires. Moreover, the interaction classes, object classes, and their relations of subscription and publication are all defined before the simulation starts. Therefore, despite the stochastic interaction class communications, we can still compute the communication cost of periodic object classes and these core HLA characteristics will be considered in the load balance strategy of this paper.

(ii) Federate Migration Techniques. The approach presented by Tan et al. [35, 36] accomplishes federate migration with a federate wrapper, which controls the federates execution through SugarCubes stopping and resuming a federate. The key point of this research is using the third-party mechanisms to avoid HLA federation save/restore approach. Although it gains less latency than some other works, its minimum latency is about 8 seconds which is much longer than VM approach in this paper.

To propose and implement an efficient procedure for balancing HLA simulation's load, paper [17] migrates federates through the GRAM Grids' service such as Web services grid resource allocation and management (WS GRAM) and GridFTP. To make the migration delay as negligible as possible, the authors implement the migration in two steps. In the first step, it executes in a way that federate does not stop its execution while initialization files are transferred. In the second step, the RTI methods are called to freeze the federation and the rest of data regarding the federate's running status and messages are transferred. The two migration steps are well-devised and similar strategy is implemented in this paper while using live VM migration.

Furthermore, in the companion work of [17], thesis [26] decreases the migration latency to around 0.8 seconds, which is highly efficiency. However, as we described in the previous section, VM live migration downtime is mainly proportional to the amount of the frequently updated memory and could have latency of tens of ms; thus we propose using VM as a new federate container and migration enabler in this paper.

3. Dynamic VM/Federate Migration Method

In a Cloud computing environment resources are shared among multiple users. The number and nature of the workload presented by these users can vary over time [5]. In addition, large-scale military HLA systems dynamically change their computation and communication load during their execution time [1, 17]. Thus, migration of simulation tasks is essential in CSim.

For CSim of military HLA, there are four basic concepts including entity, federate, VM, and host.

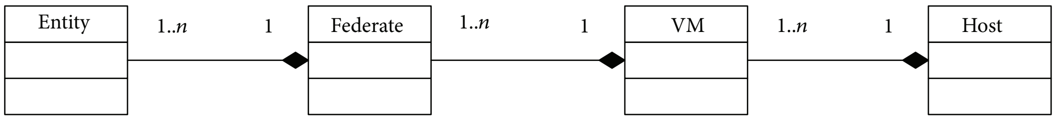

Figure 1: The mapping relations of entity, federate, VM and host.

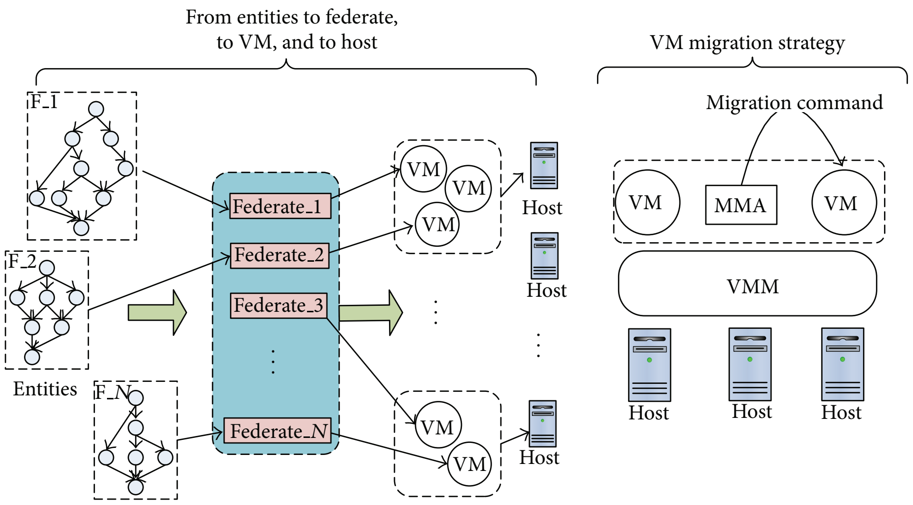

Figure 2: The strategy of clustering entities to federate, to VM, and to host.

(i) An entity often refers to a simulation object such as a tank, an airplane, and a helicopter or a car.

(ii) A federate is the encapsulated HLA form of one or more entities.

(iii) VM is the container of one or more federates because VM is the basic service unit of user's request in CSim.

(iv) Host is the container of VMs, which means one or more VM could be consolidated to one host mostly depending on the host's computation ability.

As previously mentioned, federate migration is enabled by VM in this paper. The mapping relations among the four concepts are illustrated in Figure 1.

The primary clustering strategy of multiscale HLA simulation implementation is shown in Figure 2, where VM based migration algorithm and implementation approach will be illustrated.

In the following sections we study dynamic VM migration mechanisms. Section 3.1 proposes federates' distribution and communication architecture. Section 3.2 develops the algorithm solving the problem when to migrate VMs, what VMs are to be migrated, and where the migrating VMs are to be contained.

3.1. VM Distribution and Communication Architecture. In HLA simulations, entities are often encapsulated in federates, which are then located into VMs deployed in different hosts. Each federate executes the simulation by message communication. The federate communication architecture is shown in Figure 3.

In this architecture, each VM encapsulates a local-RTI component (LRC) and there is one server-RTI running on a host. The RTI communication here is hybrid, which means that HLA global management services including time synchronizations are executed by the communications among server-RTI and LRC. Meanwhile, federate to federate communications including object instance attribute reflections and interaction instance sending/receiving are executed via LRC in peer-to-peer mode [44].

The VM/Federate distribution architecture is designed as Figure 4, each host is equipped with a VM monitor (VMM) in charge of monitoring the VMs' load and hosts' load. In our CSim, every VMM is both a load monitor and a migration executor. VMM keeps track of running state of local host and VMs. Also VMM captures the snapshots of local VMs periodically and then keeps them in a database located in local host.

Meanwhile, one host has a migration management agent (MMA) in charge of monitoring all VMs' periodic status, triggering VM migration procedure when a host is overloaded, selecting the migrated federate sets and target VMs, and sending the command. MMA monitors all VMs' periodic status by collecting status data from VMM of each host. This is a pull manner because VMM pulls data from distributed hosts to MMA. The frequency of this monitoring is set as 1/s in this paper.

3.2. VM Migration Algorithm. Based on the architecture shown in Figures 3 and 4, this section delivers a solution which handles dynamic load imbalance in HLA federations considering both computational and communication costs.

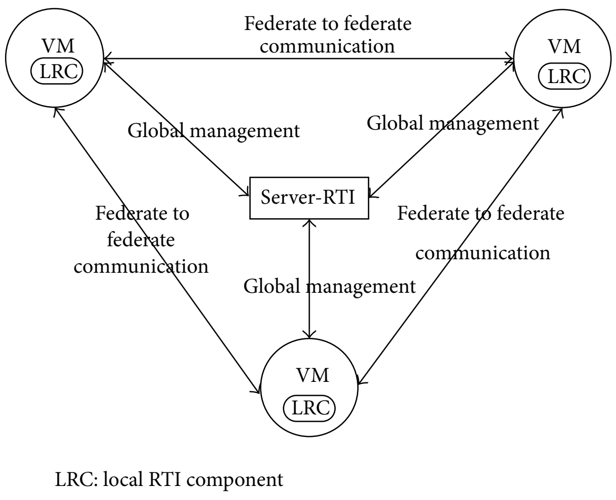

Figure 3: VM/Federate communication architecture.

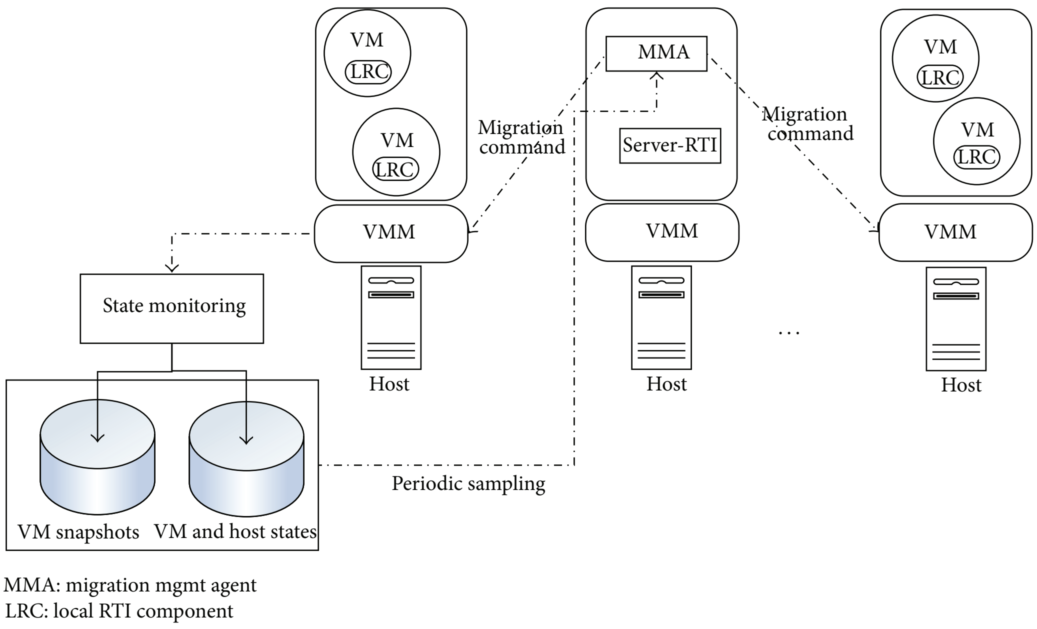

Figure 4: VM/Federate distribution architecture.

Let us start from analyzing the utilization and workload of hosts.

3.2.1. Hosts' Utilization Threshold. $U_{p_j}(t)$ is the utilization of host p_j at time t:

$$U_{p_j}(t) = \alpha * U_{\text{cpu}}(t) + (1-\alpha) U_{\text{mem}}(t), \qquad \text{Th}_{p_j} = k, \quad 0 < k \le 1, \tag{1}$$

where $U_{\text{cpu}}(t)$ is CPU utilization of host p_j (in percent) and $U_{\text{mem}}(t)$ is memory utilization of host p_j (in percent). α is a coefficient representing the relative importance between CPU utilization and memory utilization. As both CPU and memory are equally important for running VM, α is set to 0.5.

The utilization threshold host p_j is Th_{p_j} or k, which is a parameter that allows the adjustment of the effect of the method; the lower k is, the higher the possible overload is and the higher possibility of migration is. For the determination of utilization threshold, the following equation is used: $k = \lceil (1+\beta) \cdot ((\sum_{j=1}^{n} U_{p_j})/n) \rceil$, where k is the CPU utilization threshold for all hosts, U_{p_j} is load of node j, n is the number of hosts in the simulation system, and $0 \le \beta \le 0.2$ is a normalized constant. To generate moderate migration, here we set $\beta = 0.1$.

Obviously there are two load states for the hosts.

(i) If $U_{p_j}(t) \ge k$, host p_j is overloaded.

(ii) If $U_{p_j}(t) < k$, host p_j is not overloaded.

3.2.2. Load of Host and VM. At time t, load $L_{p_j}(t)$ of host p_j is computed by the following equation:

$$L_{p_j}(t) = U_{\text{cpu}}(t) * C_j, \quad (2)$$

where C_j is the computation capacity of host p_j, which is the frequency of the host's CPU mapped onto millions instructions per second (MIPS) ratings of each core [45].

In this paper we assume that each federate has variable workload throughout a simulation, but we can use preliminary experiments to test the maximum loaded VMs of a host according to their configurations. Suppose that all VMs are homogeneously configured and have the same amount of entities; the load $L_{v_{ij}}(t)$ (MIPS) of VM v_{ij} at time t is defined as follows:

$$L_{v_{ij}}(t) = \frac{L_{p_j}(t)}{n_j}, \quad (3)$$

where n_j is the number of VMs in host p_j and the assumption here also enables that we can migrate a set of VMs at one time (see algorithm studied later).

3.2.3. Communication Cost. In CSim HLA federation, federates communicate with each other through the interaction class instance and object class instance. As discussed in Section 2.2, we assume the interaction class instance is stochastically sent and object class instance is periodically updated every time step for the correctness of simulations. The communication bandwidth request (bit/s) between federate a_1 and a_2 is as follows:

$$\text{Comm}_{a_1,a_2} = \frac{1}{\text{sim\_step}} * \text{obj\_ins\_bytes} * 8, \quad (4)$$

where obj_ins_bytes is the amount of object class instances bytes exchanged every time step. This means the communication cost shown in Figure 5 is the object class instances' requirements of network bandwidth.

Then we try to compute the communication cost between host and VM and host and host. Figure 5 illustrates the interactions.

In most cases a prerequisite is that VMs in a local host can communicate much faster than VMs among different hosts. Therefore, we must consider the two cases separately. In Figure 5(a), the solid lines are communications among VMs within hosts, and the dashed lines are the communications among VMs of different hosts. After communication merging, we can get Figure 5(b), where the *c1s* represents the sum of communication costs between vm1 and host p_j.

Therefore, the communication cost at time t between the VM a_{ik} and the host p_j is defined as follows:

$$\text{Comm}_{a_{ik},p_j}(t) = \sum_{l=1}^{n_j} \text{Comm}_{a_{ik},a_{lj}}. \quad (5)$$

Then hosts p_k and p_j communication cost at time t is as follows:

$$\text{Comm}_{p_k,p_j}(t) = \sum_{i=1}^{n_k} \text{Comm}_{a_{ik},p_j}(t). \quad (6)$$

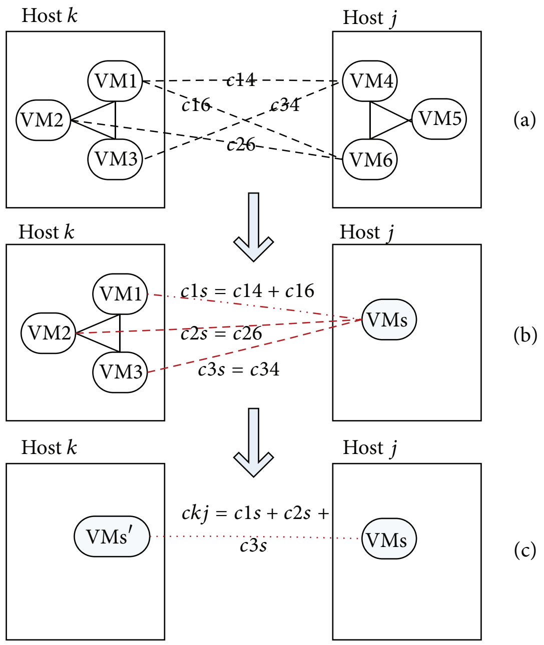

Figure 5: Merging of VM interactions in hosts.

With respect to two objectives of dynamic load balancing, reducing the load of the overloaded hosts and decreasing the interhost communication cost, a dynamic load balancing model is proposed as follows:

$$\min z(t) = \sum_{j=1}^{n} \text{Comm}_{p_k,p_j}(t), \quad \text{s.t. } U_{p_j}(t) < \text{Th}_{p_j}. \quad (7)$$

The objective function $z(t)$ is to minimize the interhost communications between host p_k with VM to be migrated and other hosts, and the constraint is that each host's computation load is below its threshold.

3.2.4. Migration Algorithm. The migration model above is a NP-hard problem. Many researchers have used heuristics to find the optimal solutions and our approach is influenced by them including the works in [17, 26, 35, 36, 39–43, 45–48]. However, compared with existing researches, our algorithm not only considers the periodical HLA object class communication cost, but also migrates a set of VMs every time decreasing the migration procedure latency compared to most one-by-one federate migration methods.

Suppose the overloaded host is p_k and the destination host selected by migration management agent (MMA) is p_j, which has least utilization in the simulation. The heuristic is to select a set of VMs from p_k to migrate to p_j, in order to reduce the load of p_k and minimize the communication cost after migration. The algorithm is illustrated in Algorithm 1.

For the proposed algorithm in Algorithm 1, the time complexity of Steps 1–4 is $O(N_{\text{ave}}^2)$ (N_{ave} is the average number of VMs per host) and the time complexity of Step 5 is $O(m)$. Thus the time complexity is $O(m * N_{\text{ave}}^2)$. Moreover, the algorithm is executed with the same frequency of MMA monitoring all VMs' periodic status, that is, 1/s.

```
Input: VM list with $m$ VMs, host list with $n$ hosts.
Output: Deployment that VM to host, $(\mathrm{VM}_i, \mathrm{host}_j \mid i \in (1, m), j \in (1, n))$.
Algorithm:
(1) At time $t$, MMA finds that host $p_k$ is overloaded and needs VM migration, where $U_{p_k}(t) > k$,
$L_{\mathrm{mig}}(t) = (U_{p_k,\mathrm{cpu}}(t) - k) * C_k$. Also $p_j$ is the least loaded host in host list. If $U_{p_j}(t) \geq k$, all hosts
are overloaded and this algorithm does not perform migration; else if $U_{p_j}(t) < k$ MMA
sends command to $p_k$ that $p_j$ is its migration destination host.
(2) Then $\min\{L_{\mathrm{migrate}}(t), \quad (k - U_{p_j}(t)) * C_j\}$ is the largest accepted migration load. The largest
accepted VM number is calculated according to $n_j^{\mathrm{mig}}(t) = \mathrm{floor}(\min\{L_{\mathrm{mig}}(t), (k - U_{pj}(t)) * C_j\}/(L_{v_{ik}}(t)))$.
(3) Calculate the communication cost $\mathrm{Comm}_{v_{ik},p_j}(t)$ between every VM of $p_k$ and host $p_j$, and the
sum of communication cost $\mathrm{Comm}_{v_{ik},p_k-v_{ik}}(t)$ between the VM in $p_k$ and the rest VMs in $p_k$. The
VM which has $\min_i(\mathrm{Comm}_{v_{ik},p_k-v_{ik}}(t) - \mathrm{Comm}_{v_{ik},p_j}(t))$ is selected into the VM set $s_{kj}(t)$. Then the
selected VM is removed from $p_k$, while $p_j$ adds the selected VM. Accordingly, the communication
relations of VMs' communication are updated.
(4) If the number of VMs in $s_{kj}(t)$ is less than $n_j^{\mathrm{mig}}(t)$, back to Step 3. Otherwise output its planned
migration set $s_{kj}(t)$ of $p_k$.
(5) If $L_{s_{kj}(t)} \leq L_{\mathrm{mig}}(t)$, VMM of host $p_k$ and $p_j$ starts the migration.
```

Algorithm 1: Communication cost based VM dynamic migration algorithm.

4. Experiment Results and Analysis

4.1. Experiment Design. To validate the effectiveness of the proposed VM based HLA simulation load balancing method in CSim, experiments have been designed and implemented. The simulations were run in a system comprising 2 nodes of Lenovo 8200t, 2 nodes of HP 6300 Pro MT, 6 nodes of HP Compaq 8000 Elite CMT, and a 100 Mbit/sec Ethernet connection among all the nodes. The node of Lenovo 8200t had an Intel i7-870 (8 cores) 2.93 GHz CPU and 8 G MEM. The node of HP 6300 had an Intel i5-3470 (4 cores) 3.2 GHz CPU and 4 G MEM. The node of Compaq 8000 had an Intel Core 2 E8400 (2 cores) 3.00 GHz CPU and 2 G MEM.

The nodes run a paravirtualized Linux CentOS 5.6 kernel as a privileged virtual machine on top of the Xen hypervisor 4.0.1 [30]. The guest virtual machines are configured to single core and run the same version of the Linux kernel as that of the privileged one. HLA platform was AST-RTI [49, 50] version 2.0 performing communication through TCP/IP connections.

Moreover, as our benchmark, a practical HLA armored force game for tactical training was developed. The game coded in C/C++ was used to conduct experiments and analyze the performance of our approach. The scenario for our experiments was a simulation of battle engagement game of red and blue tank forces, which were hierarchically organized as Platoon (P), Company (C), Battalion (B), and Regiment (R). The tank effectuated random selection of several tactical routes and engagement strategies in two-dimensional space that was within range of some military training location.

The organization structure of tank forces is illustrated in Figure 6, which shows that red forces are formed hierarchically in 3 to 3 organization. This means that every red company has 3 platoons and every platoon has 3 tanks, while, for the blue side, it is formed in 4 to 4 organization, which means every blue company has 4 platoons and every platoon has 4 tanks.

Table 1: The number of VMs in different game scenarios.

Scenario (or scale)	Number of VMs
1 red Company versus 1 blue Company	9
1 red Battalion versus 1 blue Company	18
1 red Company versus 1 blue Battalion	25
1 red Battalion versus 1 blue Battalion	34
1 red Battalion + 1 red Company versus 1 blue Battalion	38
1 red Battalion + 1 red Company versus 1 blue Battalion + 1 blue Company	43
1 red Battalion + 2 red Company versus 1 blue Battalion + 1 blue company	47
1 red Battalion + 2 red Company versus 1 blue Battalion + 2 blue Company	52
2 red Battalion versus 1 blue Battalion + 2 blue Company	57

In order to accomplish such simulations, we cluster tank entities into VMs according to their military affiliations. The abbreviations are P: Platoon, C: Company, B: Battalion, R: Regiment, r: red, b: blue.

Table 1 shows the experiments' deployment. Each VM contains one federate in the experiments because computation and communication costs are mainly due to the number of tank entities. When the number of entities in one VM is fixed, the number of federates has little impact on the VM's costs as interhost communication cost is normally much greater than local host cost.

Moreover, each red tank Company is deployed with 4 VMs, which are Platoon-1 (P-1), P-2, P-3, and Company

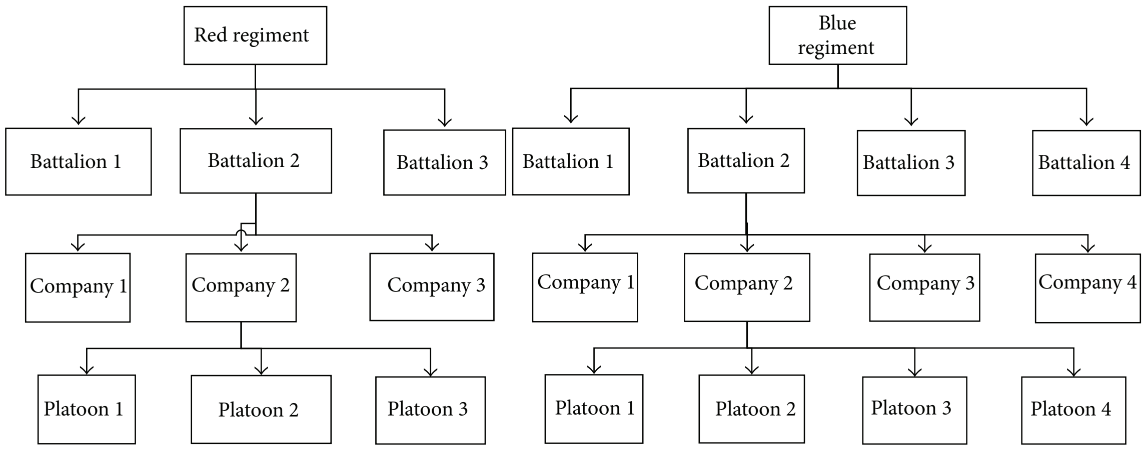

FIGURE 6: Hierarchical organizational structure of red and blue forces.

tank. Each blue tank Company is deployed with 5 VMs, which are Platoon-1, P-2, P-3, P-4, and Company tank.

To fulfill migration algorithm addressed in Section 3.2, the communication cost between VM/Federates were estimated according to periodic HLA object class instances while ignoring stochastic HLA interaction class instances. For example, rR1B1C1 needs to report its information by sending its object class instances to rR1B1 every simulation step. Its object class contains the information of ID, name, position, fuel consumption, ammunition, and so forth. According to this, we can estimate the size of its object class, for instance, 48 bytes. Assuming the simulation step is 50 ms, then the communication cost caused by the object class instance is 960 bytes/sec. By using this method, we can get the communication cost among all the federates.

4.2. Experimental Results and Analyses. In order to evaluate the proposed VM based migration algorithm's efficiency, the experiments were accomplished in two test case groups over heterogeneous, nondedicated sets of resources, applying an increasing large load to the distributed system. In the first test case group, the effectiveness of the dynamic load balancing system was observed as distributed load imbalances occur. In the second test case group, to analyze the detection of external background load, an external load is added in the system and the balancing reaction is observed.

(1) Reactivity to Load Imbalances. In this test case, all the distributed simulations were deployed based on an initial static partitioning that evenly placed the VM/Federates on the resources. However, due to the resource heterogeneity characteristics and variable federate loads, the simulation shows an uneven distribution of load, decreasing the simulations' performance. In order to evaluate the balancing system's reaction to load imbalances and the VM encapsulation's impact on simulation, the balanced VM based simulation's performance was compared with static distribution wrapped and unwrapped with VM. In this case of experiments, the system comprehended the run of the experimental scenario with a configuration of federates that ranged from 9 to 57 (see Table 1).

To provide trustworthy results, each execution time in our graphs represents the average of 20 runs. For every mean value of simulation execution time, a 95% confidence interval was evaluated. The half-widths of all confidence intervals are less than 5% of their respective mean values. According to Figure 7, the proposed dynamic balancing algorithm and VM migration improved the performance of HLA-based simulations on large-scale distributed systems in most of the experiments. When the distributed load was under 20 federates, the balancing scheme's improvement is unnoticeable or nonexistent because the simulations did not require any load balancing. In this case, the balancing just caused a small overhead (2.1%) for the distributed system, consuming computing from the resource where the MMA was deployed. A noticeable improvement was detected with experiments over 25 federates because considerable load imbalances occurred during the simulation, along with the different deployment of VMs and the heterogeneity of resources caused an imbalanced division of load. Then, a high increase in execution time in the balanced system is observed when the number of federates is over 50. This increases evidence that the distributed system is reaching a saturation point in which the balancing system cannot improve the simulation performance since all resources are becoming totally overloaded.

In addition, blue and red curves in Figure 7 show that the average overhead with VM encapsulation compared to without VM in all runs is 3.28%, which means using VM is acceptable because of two reasons. Firstly, using VM, live migration techniques saves lots of simulation programmers' efforts in realization of federate migrations. Secondly, when the number of federates is less than 50, that is, below the saturated point of the system, the average execution time saved is 22.25% compared to the static distribution runs without VMs.

(2) Detection of Background Load. In order to measure the efficiency of the load balancing system in detecting and reacting to the background loads, external jobs are generated using a tool called Stress [5]. Stress is a workload generator for POSIX systems and allows for a configurable amount

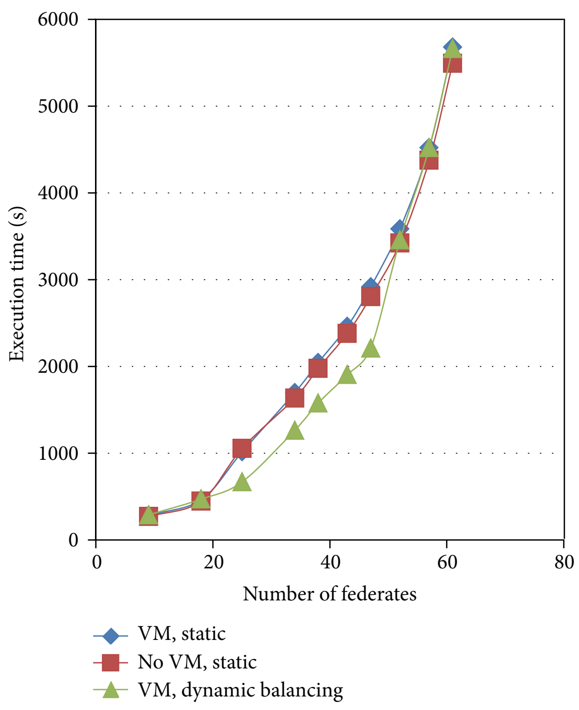

Figure 7: Dynamic balancing scheme versus a static distribution for an increasing scale of federates.

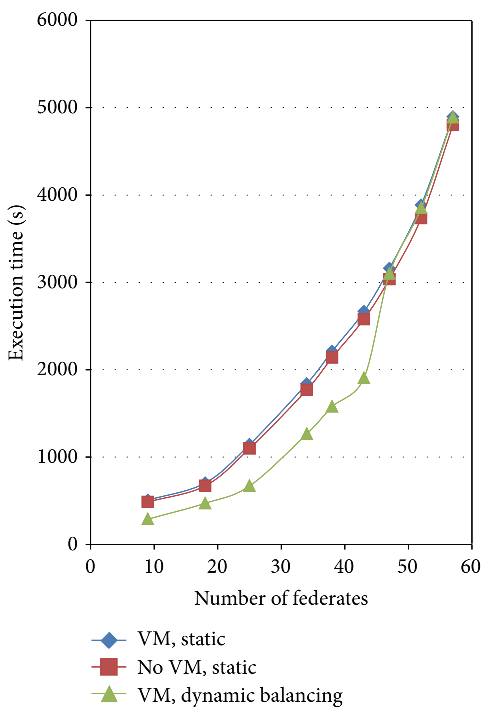

Figure 8: Capacity of the dynamic balancing scheme in detecting background load for an increasing scale of federates.

of CPU and memory stress on the system. In the test case, the federates were deployed evenly on the distributed nodes, and Stress was placed on two nodes of HP Compaq 8000 workstation. The load was 1-CPU bound, 1 I/O bound, and one memory allocator process.

As shown in Figure 8, the curves are similar to those in Figure 7 except that introduction of an external load caused an addition of execution time for experiments which have no dynamic balancing scheme. However, the saturated point is earlier (changes from 52 to 47) because of the external load imposed on the distributed system. Thus, the load balancing system presented a performance improvement detecting the external load and triggering redistribution of load only when the distributed system is not saturated.

5. Conclusions and Future Work

The paper proposes a VM based federate migration scheme for HLA system load balancing on Cloud Simulation Platform. Contribution of this work could be summed in two aspects: (i) it proposed to use VM as the container of federate. The overhead brought by VM is about 3.33% according to papers [1, 33, 34] (in our tests it is around 3.28%). (ii) It devised an algorithm of HLA load balancing under the constraints of both computational and communication costs. The experiment results show that the migration scheme effectively improved the efficiency of the HLA system with the prerequisite that the distributed system is not saturated.

As a preliminary work in Cloud computing based HLA system, this research has a lot of future work to do. Firstly, the computing granularity is still a difficult problem because VM is actually a heavy container for current resources, and if one VM contains only one federate, the federate should include as many simulation entities as possible. However, a big federate containing many entities may not be flexible to migrate for load balancing. Therefore, it is complex to design an appropriate computing granularity and this should be solved in the future. Secondly, migration algorithm should be designed to be more adapted to HLA systems. In this paper, we devised an algorithm considering both computational and communication cost. However, the algorithm neglected the stochastic interaction classes' characteristics, which may be considered in an intelligent way to enhance the efficiency of load balancing in HLA.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The research in this paper was supported by Grants 61104057 and 61473013 from the Natural Science Foundation of China and funding of the Science and Technology on Complex Land Systems Simulation Laboratory (63963). The authors thank the reviewers for their comments.

References

[1] X. Liu, Q. He, X. Qiu, B. Chen, and K. Huang, "Cloud-based computer simulation: Towards planting existing simulation

software into the cloud," *Simulation Modelling Practice and Theory*, vol. 26, pp. 135–150, 2012.

[2] B. H. Li, X. Chai, and L. Zhang, "New advances of the research on cloud simulation," in *Advanced Methods, Techniques, and Applications in Modeling and Simulation*, vol. 4 of *Proceedings in Information and Communications Technology*, pp. 144–163, 2012.

[3] S. Jafer, Q. Liu, and G. Wainer, "Synchronization methods in parallel and distributed discrete-event simulation," *Simulation Modelling Practice and Theory*, vol. 30, pp. 54–73, 2013.

[4] R. Fujimoto, A. Malik, and A. Park, "Parallel and distributed simulation in the cloud," *SCS Modeling and Simulation Magazine*, pp. 1–10, 2010.

[5] A. W. Malik, A. J. Park, and R. M. Fujimoto, "An optimistic parallel simulation protocol for cloud computing environments," *SCS M&S Magazine*, vol. 4, 2010.

[6] A. Jávor and A. Fur, "Simulation on the Web with distributed models and intelligent agents," *Simulation*, vol. 88, no. 9, pp. 1080–1092, 2012.

[7] IEEE Std 1516.1-2010, *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)*, Framework and Rules Specification, 2010.

[8] IEEE Std 1516.2-2010, *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)*, Object Model Template (OMT) Specification, 2010.

[9] IEEE Standard, *1516.1-2010—IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface Specification*, 2010.

[10] S. Radio, D. Parsons, and V. Deneen, *MODSAF Overview and MODSAF History [EB/OL]*, 2006, http://www.aiai.ed.ac.uk/~arpi/SUO/MODULES/modsaf.html.

[11] B. McEnany, "CCTT SAF functional analysis," in *Proceedings of the 4th Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, 1994.

[12] A. J. Courtemanche and R. L. Wittman Jr., "OneSAF: a product line approach for a next-generation CGF," in *Proceedings of the 11th Computer Generated Forces Conference*, IEEE Computer Society Press, Orlando, Fla, USA, 2002.

[13] One Semi-Automated Forces (OneSAF), "Operational Requirements Document (ORD) Version 1.1[EB/OL]," 2000, http://www.onesaf.net/community/.

[14] B. H. Li, X. Chai, Y. Di, H. Yu, Z. Du, and X. Peng, "Research on service oriented simulation grid," in *Proceedings of the IEEE International Symposium on Autonomous Decentralized Systems (ISADS '05)*, pp. 7–14, April 2005.

[15] I. Foster, C. Kesselman, J. M. Nick et al., *The Physiology of Grid: An Open Grid Services Architecture*, 2003.

[16] S. Tuecke, K. Czajkowski, and I. Foster, Open Grid Services Infrastructure (OGSI), 2003, http://www.ggf.org/documents/GFD.15.pdf.

[17] A. Boukerche and R. E. de Grande, "Dynamic load balancing using grid services for HLA-based simulations on large-scale distributed systems," in *Proceedings of the 13th IEEE/ACM Symposium on Distributed Simulation and Real-Time Applications (DS-RT '09)*, pp. 175–183, October 2009.

[18] Amazon AWS, 2014, http://aws.amazon.com.

[19] Google, https://cloud.google.com/.

[20] Softlayer, 2014, http://www.softlayer.com/Cloud.

[21] R. N. Rodrigo, R. Ranjan, A. Beloglazov, C. A. F. de Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[22] Intel Corporation, *System Virtualization-Theory and Implementation*, Tsinghua University Press, Beijing, China, 2009.

[23] D. Ruest and N. Ruest, *Virtualization: A Beginner's Guide*, McGraw-Hill, NewYork, NY, USA, 2009.

[24] M. Rosenblum and T. Garfinkel, "Virtual machine monitors: current technology and future trends," *Computer*, vol. 38, no. 5, pp. 39–47, 2005.

[25] C. Clark, K. Fraser, S. Hand et al., "Live migration of virtual machines," in *Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design & Implementation (NSDI '05)*, vol. 2, pp. 273–286, USENIX Association, Berkeley, Calif, USA, 2005.

[26] R. E. De Grande, *Dynamic load balancing schemes for large-scale HLA-based simulations [Ph.D. thesis]*, University of Ottawa, Ontario, Canada, 2012.

[27] VMware, 2014, http://www.vmware.com.

[28] Opennebula, 2014, http://opennebula.org/.

[29] Eucalyptus, http://www.eucalyptus.com/.

[30] Xen, http://www.xenproject.org/.

[31] C. Clark, K. Fraser, S. Hand et al., "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI '05)*, vol. 2, pp. 273–286, 2005.

[32] F. Travostino, P. Daspit, L. Gommans et al., "Seamless live migration of virtual machines over the MAN/WAN," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 901–907, 2006.

[33] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel, *Diagnosing Performance Overheads in the Xen Virtual Machine Environment-Network*, 2014, http://www.usenix.org/events/vee05/full_papers/p13-menon.pdf.

[34] G. Diwaker and G. R. C. Ludmila, XenMon: QoS Monitoring and Performance Profiling Tool, http://www.hpl.hp.com/techreports/2005/HPL-2005-187.pdf 2014.

[35] G. Tan and K. C. Lim, "Load distribution services in HLA," in *Proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '04)*, pp. 133–141, October 2004.

[36] G. Tan, A. Persson, and R. Ayani, "Migration of HLA federates," in *Proceedings of the Simulation Interoperability Workshop (SIW '05)*, San Diego, Calif, USA, 2005.

[37] W. H. Tao, *Task management and scheduling methods for grid-computing-based simulation [Ph.D. thesis]*, National University of Defense Technology, 2005.

[38] W. Cai, S. J. Turner, and H. Zhao, "A load management system for running HLA-based simulation over the grid," in *Proceedings of the 6th IEEE International Symposium on Distributed Simulation and Real Time Applications*, pp. 7–14, Fort Worth, Tex, USA, 2002.

[39] T. Alam and Z. Raza, "A dynamic load balancing strategy with adaptive threshold based approach," in *Proceedings of the 2nd IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC '12)*, pp. 927–932, Solan , India, December 2012.

[40] J. Xu and K. Hwang, "Heuristic methods for dynamic load balancing in a message-passing supercomputer," in *Proceedings of the ACM/IEEE conference Supercomputing (Supercomputing '90)*, pp. 888–897, New York, NY, USA, November 1990.

[41] A. Y. Zomaya and Y.-H. Teh, "Observations on using genetic algorithms for dynamic load-balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 9, pp. 899–911, 2001.

[42] S. Jin and B. Ren, "A novel distributed dynamic load balancing mechanism," in *Proceedings of the International Conference on Information Technology, Computer Engineering and Management Sciences (ICM '11)*, pp. 133–137, Nanjing, China, September 2011.

[43] A. Boukerche and S. K. Das, "Reducing null messages overhead through load balancing in conservative distributed simulation systems," *Journal of Parallel and Distributed Computing*, vol. 64, no. 3, pp. 330–334, 2004.

[44] M. Eklöf, M. Sparf, F. Moradi, and R. Ayani, "Peer-to-peer-based resource management in support of HLA-Based distributed simulations," *Simulation*, vol. 80, no. 4-5, pp. 181–190, 2004.

[45] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency Computation Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[46] Q. Long, J. Lin, and Z. Sun, "Agent scheduling model for adaptive dynamic load balancing in agent-based distributed simulations," *Simulation Modelling Practice and Theory*, vol. 19, no. 4, pp. 1021–1034, 2011.

[47] N. Rodrigo, R. Ranjan, A. Beloglazov, C. A. F. de Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2014.

[48] A. Murtazaev and S. Oh, "Sercon: server consolidation algorithm using live migration of virtual machines for green computing," *IETE Technical Review*, vol. 28, no. 3, pp. 212–231, 2011.

[49] Y. Xu, M. Yu, and X. Wang, "Research and development on AST-RTI," in *Systems Modeling and Simulation: Theory and Applications*, vol. 3398 of *Lecture Notes in Computer Science*, pp. 361–366, 2005.

[50] N. Li, X.-Y. Peng, M.-H. Zhang, M. Wang, and G.-H. Gong, "Multimedia communication over HLA/RTI," *Simulation Modelling Practice and Theory*, vol. 14, no. 2, pp. 161–176, 2006.