A Multi-layered Scheme for Distributed Simulations on the Cloud Environment

Shichao Guan, Robson Eduardo De Grande, and Azzedine Boukerche,
PARADISE Research Laboratory - School of Information Technology and Engineering
University of Ottawa, Canada
Email: sguan049@uottawa.ca, rdgrande,boukerch@site.uottawa.ca

***Abstract*—In order to improve simulation performance and to integrate simulation resources among geographically distributed locations, the concept of distributed simulation is proposed. Several types of distributed simulation standards, such as DIS and HLA, are established to formalize simulations and achieve reusability and interoperability of simulation components. To implement these distributed simulation standards and to manage the underlying system of distributed simulation applications, we employ Grid Computing and Cloud Computing technologies. These tackle the details of operation, configuration, and maintenance of simulation platforms in which simulation applications are deployed. However, for modelers who may not be familiar with the management of distributed systems, it is challenging to make a simulation-run-ready environment among different types of computing resources and network environments. In this article, a new multi-layered cloud-based scheme is proposed for enabling modeling and simulation based on different distributed simulation standards. This scheme is designed to ease the management of underlying resources and to achieve rapid elasticity that can provide unlimited computing capability to end users; it considers energy consumption, security, multi-user availability, scalability, and deployment issues. A mechanism for handling diverse network environments is described; by adopting it, idle public resources can be easily configured as additional computing capabilities for the local resource pool. A fast deployment model is built to relieve the migration and installation process of this platform. An energy-saving strategy is utilized to reduce the consumption of computing resources. Security components are implemented to protect sensitive information and block malicious attacks in the cloud. In the experiments, the proposed scheme is compared with its corresponding grid computing platform; the cloud computing platform achieves similar performance, but incorporates many advantages that the Cloud can provide.**

Keywords*—*Cloud Computing, DIS, HLA, Availability, Energy Consumption, Usability, Elasticity, Distributed Simulations.

I. Introduction

Compared with the traditional computer simulation, distributed simulation greatly reduces the length of time required for executing the simulation and achieves larger computing capabilities that are able to support geographically distributed complex simulations [1]. In addition, multiple participants from geographical distributed locations can be embedded in the distributed virtual environments (DVEs) in the simulation scenarios. Several approaches have been designed for distributed simulation systems, such as the Distributed Interactive Simulation (DIS) [2]; its successor – the High Level Architecture (HLA) [3]; and the Data Distribution Service for Real-Time Systems (DDS) [4]. The DIS is mainly designed for military simulations; it introduces design concepts such as interoperability, autonomy, and dead reckoning that prosper to the development of distributed commercial applications. HLA, initially developed by the Department of Defence in the United States, is an extensively-used framework that provides reusability and interoperability for distributed simulations. By implementing this framework, modeling and simulation have been enabled for supporting analysis, engineering, military, entertainment, education, and various other applications which can be linked to live systems, collectively defined as federates. DIS and HLA have been also developed as IEEE standards for modeling and simulation. DDS, managed by the Object Management Group, is used as a messaging middleware standard that supports data-centric simulations, enabling seamless, timely, scalable, and dependable distributed data sharing.

Coordinated by the aforementioned standards and frameworks, large-scale distributed simulations can be deployed to geographically distributed computing resources. However, within the specifications of the standards and frameworks, mechanisms for executing simulations are not provided for the underlying systems. For each physical computation element, it is necessary to carefully perform a series of configurations to build the simulation-run-ready environments so that the simulations can be executed properly.

In order to meet the need for handling the underlying system, Grid Computing [5] is introduced to manage the shared resources and the scheduling distributed simulations. A Grid system functions as a coordinated resource sharing system that provides services such as security, resource management, information queue, and data management for distributed simulations. The Globus Toolkit (GT) [6] is defined as a de-facto middleware standard for Grid Computing that helps to reach seamless interoperability. Compared to traditional computing simulation systems, Grid-based simulation platforms – with the help of Grid services [7], overcome limitations in terms of distribution of resources, dynamic access, security, organization, and collaboration.

Even though the Grid eases the management of underlying systems for distributed simulations, there are still problems with its ability to handle resources in a fine-grained manner. As a result, Grid-based platforms cannot reallocate resources

Manuscript received November 15, 2014; revised March 6, 2015.

during runtime while fully supporting multi-users. To tackle these issues with the underlying system, Cloud Computing [8] is employed as a new approach for distributed simulations. According to the analyses in [9] – [12], there are several reasons to migrate distributed simulations from the Grid to the Cloud:

- *Resource sharing*: Cloud provides resources on demand during runtime, while Grid emphasizes fair sharing of resources across organizations.
- *Virtualization*: For Grid, virtualization mainly covers the softlayer, which concerns data and programming. For Cloud, in addition to the softlayer, virtualization covers hardware resources. By enabling abstraction and encapsulation of raw physical resources, Cloud can provide better isolation and manageability for fine-grained resources.
- *Scalability*: Grid scales primarily by increasing the number of working nodes. Cloud offers automatic resizing of virtualized hardware.
- *Payment*: Grid services are typically billed using a fixed rate, while Cloud users have the option of a pay-per-use flexible payment model.

Although Cloud Computing has not been standardized to inner interfaces for accessing resources, which may lead to an inconvenient situation if computing resources are managed by different cloud service providers, it is still a promising approach for managing distributed systems and executing distributed simulations. Compared with these Grid services, the Cloud services, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), are agile and flexible, resulting in a greater ease of control of granularity of services.

Due to the advantages mentioned above, many cloud simulation platforms have been proposed, incorporating various aspects into their design such as job scheduling, monitoring, security, and other aspects, as shown in [46]. However, these existing cloud platforms are implemented based on local resources. In this case, to end users, the capabilities available for provisioning are limited to the size of its local resource pool. This is not in accordance with the essential characteristic of elasticity in Cloud Computing, according to the definition from the National Institute of Standards and Technology (NIST) [13]. In addition, the issue related to energy consumption is not fully addressed, particularly for distributed simulation applications that require a relatively long period of time to execute. The migration & deployment model of the cloud simulation platform itself is also not fully discussed.

In this paper, a multi-layered cloud simulation scheme is proposed, concerning usability, elasticity, energy consumption, and fast deployment. Components are designed and implemented to provide unlimited computing resources to end users by coordinating public computing resources during runtime. Energy-aware elements are utilized to reduce unnecessary energy cost from computing resources. A deployment model is designed to accelerate the migration and deployment process of the cloud simulation platform. In addition, the scheme contains functions for job scheduling, monitoring, and a friendly web-based graphic interface to ease the configuration, operation, and maintenance of the underlying system.

The reminder of the paper is organized as follows. In Section II, state-of-the-art related research work is shown and discussed. In Section III, the proposed scheme is described, including architecture, key components, and functioning. In Section IV, several experiments are presented and discussed in terms of energy consumption, management efficiency, and performance. In the last section, the proposed solution design and experimental results are summarized, and directions for future works are provided.

II. Related Work

In order to facilitate the development of distributed simulations, especially HLA-based simulations on the Grid platform, much effort has been made in the area of service provisioning, heterogeneity, resource distribution, and load management.

In the HLAGrid [14], a Federate-Proxy-RTI architecture is proposed, enabling federates to be exposed as Grid services. In this case, simulation resources can be configured, maintained and scaled much more easily, hiding the details of the underlying platforms. In the HLA-GRID-REPAST [15], the HLA-REPAST middleware coordinates the HLA-GRID, providing an agent-based solution that enables distributed simulations requiring computing resources and data sets from geographically distributed locations. In [16], a management framework is designed for HLA-based simulations in grid environments, enabling HLA Management Service, Migration Support Service and Broker Support Service. In [17], the architecture of Load Management System (LMS) is proposed, combining an existing resource sharing system with the RTI of HLA. In this case, Grid services can be straightforwardly utilized for job scheduling and federate migration. In [18], a conservative approach is proposed to balance the load during runtime. In [19], a hierarchical dynamic load balancing scheme is designed to handle non-dedicated resources in the distributed system. In [20], a distributed scheme is proposed that addresses the communication load between distributed resources. In [21], a predictive model is proposed. In this distributed dynamic scheme, communication load can be balanced by a proximity analysis of federate interactions.

The novelty of the Cloud Computing system offers many features and characteristics that the Grid cannot support, as mentioned in Section I. As a result, many cloud-based simulation platforms are proposed for the distributed simulations, focusing on on-demand resource provisioning, virtualization management, security, monitoring, optimization of data processing, synchronization, and other aspects.

The MapReduce [41] programming model is a widely used scheduling approach in the cloud environment. This model formalizes the division and parallelization of processes for computation tasks across multiple computing resources, achieving fault tolerance, locality optimization, and load balancing. This approach facilitates cloud applications such as data mining, machine learning, and sorting. However, this scheme cannot achieve a low enough latency to guarantee the performance of distributed simulation applications, particularly those that

TABLE I. COMPARISONS IN CLOUD SIMULATION PLATFORMS

Cloud Platform	Scheduling	Security	Usability	Energy Consumption	Experiment Analysis	Protability	Architecture Design
MapReduce[41]	Yes				Yes		
Aurora System[22]	Yes				Yes		
TW-SMIP[23]	Yes				Yes		
CSim[24]	Yes						Yes
CDS[25]		Yes			Yes		Yes
Cloud Simulation System[27]		Yes	Yes				Yes
On-demand Simulation Cloud[26]					Yes		Yes
Latency in Simulation Cloud[29]					Yes		
Security in Simulation Cloud[43]		Yes			Yes		
Proposed	Yes	Yes	Yes	Yes	Yes	Yes	Yes

require intensive communications and frequent message exchanges.

The Aurora system [22] also focuses on the task scheduling. This system is introduced to address the issue of small message communication in the cloud. In this system, a master/worker paradigm is implemented to aggregate small simulation communication messages, which are bundled and redirected to different destination resources. With the automatic bundling of small communication messages during the simulation execution, the cloud's high bandwidth network can be better utilized.

The Time Warp Straggler Message Identification Protocol (TW-SMIP) [23] is advanced to handle optimistic synchronization. In the TW-SMIP, heartbeat messages are used to detect straggler messages. The straggler message identification is implemented to reduce the amount of rollbacks that may be caused by asymmetrical or uneven processing loads. In addition, in this protocol, three types of SMIP protocols are proposed to distribute heartbeat messages that concern congestion, autonomous administration and communication overhead respectively.

Cloud-based Simulation (CSim) [24] concerns the idle CPU cycle issue. In this proposed scheme, each processor is virtualized as two CPUs (VCPU) – one foreground and one background. When the higher-priority foreground VCPU begins an idle cycle, the lower-priority, background VCPU begins working. According to the author, the foreground VCPU loses less than 4% performance due to this type of job scheduling, but many idle CPU cycles can be utilized by the background VCPU. Based on this two-tier structure, four job scheduling algorithms are also proposed in this scheme to improve the performance of parallel simulations on the cloud.

In the Cloud-based Distributed Simulation system (CDS), a new Cloud-RTI middleware based on traditional RTI is proposed [25]. With the encapsulation of traditional RTI, this Cloud-based RTI is provided by the use of web services. In this protocol, optimization in the process of registration operations is also made by its Management Center. For security, this system deploys a hierarchical identity-based cryptography and a role-based access control scheme. In the access control scheme, the root Key Generation Center (KGC), domain KGC, and sub-level domain KGC are proposed, tackling the inconvenience created when accessing services from different domains. With the layered KGC design, isolation of domain services is more easily achieved.

The Cloud Simulation System in [27] attempts to draw a full-scale picture of the cloud-based simulation system. In terms of security, it integrates portal security agents, access control, and self-organization to enhance protection of sensitive data. The security control domain is defined with different levels of privilege, in attempts to avoid malicious behaviours. A management system is created for various resources, recording global ID and life-cycle and attempting to ease resource allocation. Individuation virtual desktop technology, multi-user oriented dynamic building technology, and automatic composition technology of simulation models are used to support multi-user operation and simulation environment deployment. A parallel engine is defined to support fine-grained resources and tackle the sub-model issue in one federate. In this engine, lookahead is defined to extend the time interval of model parallelism, and tread-level simulation engine instances are created for large-scale simulations. Monitoring and evaluation technology is implemented for detecting abnormal behaviours on the platform and providing information for further optimization decisions.

In [26], the authors propose an on-demand HLA based simulation environment on the cloud. In [29], time latency is compared between Cloud-based RTI and native Portico RTI [35], and it shows that, within a certain range of length of update data, time latency is acceptable on the cloud.

In a series of related works [43] – [45], the authors discuss how to monitor the requirements in shared cloud platforms as well as how to deal with the profile-aware attack in IaaS cloud.

As shown in Table I, the aforementioned cloud simulation schemes promote many aspects of distributed simulations in the cloud environment. However, these schemes show limited details in terms of the deployment procedure for the proposed cloud simulation platform itself, which is challenging for modelers who may not be familiar with the management of the distributed system. As a result, the cloud-based simulation is relatively hard to popularize due to the complex configuration and maintenance of diverse raw resources which constitute the fundamental computing capability of the cloud environment. In addition, the current cloud simulation platforms mainly focus on the private cloud deployment model, which means only limited local physical raw resources are virtualized and coordinated in the cloud resource pool. In this case, to end users, the compute utility seems inadequate, particularly for large-scale distributed simulations. Meanwhile, the energy consumption issue is not fully addressed and discussed for the distributed simulation in the cloud resource pool, which may lead to substantial waster when the scale of computing resources is increased.

In this paper, a multi-layered elastic cloud simulation scheme is proposed, which supports the capability of automatic deployment, reduction of energy consumption, and scaling outwards. A deployment management scheme is proposed to tackle the deployment and migration aspects of the cloud simulation scheme, providing a flexible and agile approach for building backup replicas for the current platform. The energy consumption issue is considered, and solutions are proposed and implemented. The scaling process is implemented, which responds rapidly to requirements from cloud users, providing unlimited computing resources to end users. The scaling components support idle public instances such as physical desktops, workstations, laptops, or public cloud instances from different cloud providers such as AWS, Google App Engine, or Azure, which can be located in geographically separate regions.

III. Cloud Simulation Platform

A multi-layered platform is proposed to support diverse distributed simulation standards and to hide the management of underlying details. A deployment model is designed for the easy installation, migration, and duplication of the platform. A web-based graphic portal with safety strategies allows users to obtain access to the platform workbench through lightweight terminals. A self-managed simulation resource pool is implemented that supports, stores, and configures simulation-related softlayer resources automatically for the user's direct usage. An integration and virtualization approach is designed that can handle hardlayer resources as plug-in components elastically, requiring no direct configurations. An Energy-saving mechanism is also utilized to reduce unnecessary energy cost.

As shown in Figure II, this Cloud Simulation Platform consists of five layers: the Raw Resource Layer, the Integration and Virtualization Layer, the Simulation Function Layer, the User Management Layer, and the Deployment Management Layer. Specifically, the deployment mechanism, security mechanism, migration mechanism, and resource scheduling mechanism will each be illustrated in detail with the description of related key features of each layer.

A. Deployment Management Layer

Fast and automatic deployment and migration of the proposed scheme in new environments is a priority. To achieve this, the Simulation Resource Deployer, the Virtual Resource Deployer, and the Cloud Infrastructure Deployer (as depicted in Deployment Management Layer of Figure II) cooperate to backup current platform status, pack and transmit core resources to new environments, and deploy the platform based on its previous saved status.

1) Simulation Resource Deployer: The Simulation Resource Deployer focuses on simulation-related package handling. These simulation-related packages are managed by the Simulation Storage Manager (in Layer 3, Figure II) and are stored in both cloud and block storage. When platform packaging starts, the Simulation Resource Packager is triggered by the packaging process, and it calls the Simulation Storage Manager

Algorithm 1: Cloud Simulation Platform Packaging and Installation Algorithm

```
1  Require: $SIGN\_SR, SIGN\_VR, SIGN\_CP.$
2  Note: $CID - Cloud\ Infrastructure\ Deployer,$
   $IPC - Installation\ Parameter\ Collector,$
   $OD - OpenStack\ Deployer,$
   $CSDP - Cloud\ Simulation\ Platform\ Deployer,$
   $SVRS - Simulation/Virtual\ Resource\ Scheduler.$
   $UM - UserManager$
3  if $SIGN\_CP == packaging\_start$
4  then
5      $SIGN\_SR \Leftarrow SIGN\_CP$
6      $SIGN\_VR \Leftarrow SIGN\_CP$
7      $CID.stop(UM, SRM, VRM, SIGN\_CP)$
8      while $SIGN\_CP! = packaging\_ready$
9      do
10         $SRpacks \Leftarrow SRP.search(SRM, SIGN\_SR)$
11         $SRpacks.save()$
12         $VRpacks \Leftarrow VRP.search(VRM, IM, SIGN\_VR)$
13         $VRpacks.save()$
14         if $SIGN\_SR == ready$ && $SIGN\_VR == ready$
           then
15             $CID.add(SRpacks, VRpacks)$
16             $CID.save()$
17             $SIGN\_CP \Leftarrow packaging\_ready$
18         else
19             $SIGN\_CP \Leftarrow recheck$
20             $SIGN\_SR \Leftarrow SIGN\_CP$
21             $SIGN\_VR \Leftarrow SIGN\_CP$
22     $SIGN\_CP \Leftarrow transmitting$
23 if $SIGN\_CP == installation\_start$
24 then
25     $CID.auth()$
26     $IP \Leftarrow IPC.search(comp, st, nw)$
27     $OD.install(IP)$
28     $CSPD.install(IP)$
29     $SVRS.sche(SRI, VRI)$
30     while $SIGN\_CP! = installation\_end$
31     do
32         $SRI.unpack(SRpacks)$
33         $SRI.register(SRpacks)$
34         $VRI.unpack(VRpacks)$
35         $VRI.register(VRpacks)$
36         if $SIGN\_SR == installation\_end$
37         && $SIGN\_VR == installation\_end$
38         then
39             $CID.test()$
40             $CID.end()$
41         else
42             $SIGN\_CP \Leftarrow recheck$
43             $SIGN\_SR \Leftarrow SIGN\_CP$
44             $SIGN\_VR \Leftarrow SIGN\_CP$
```

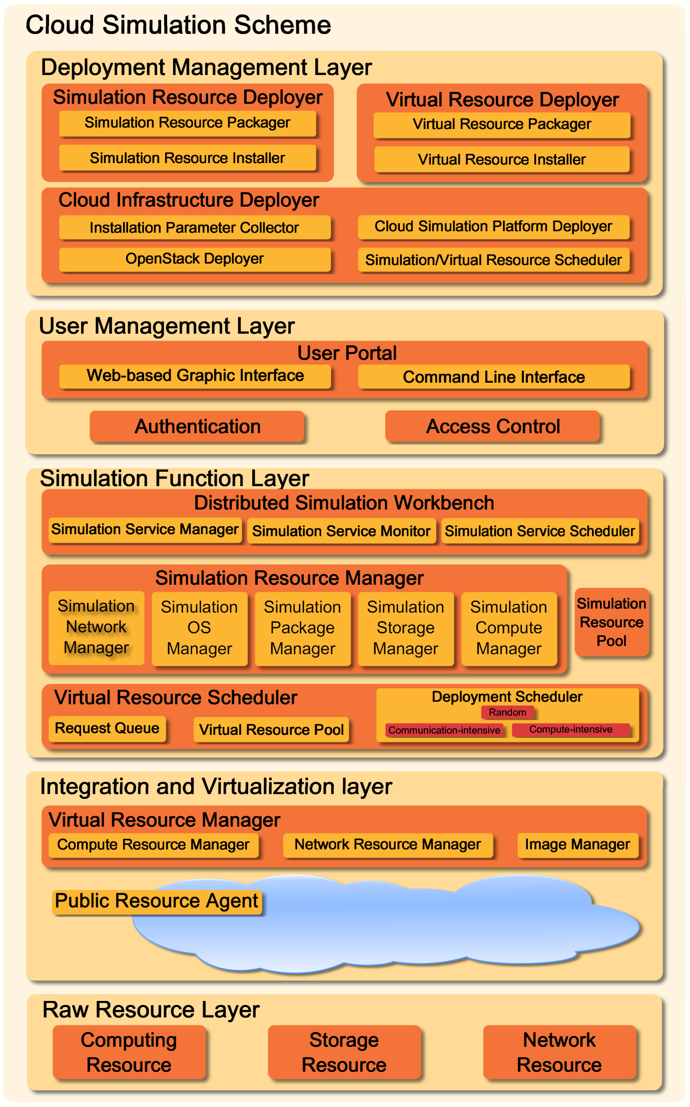

Cloud Simulation Scheme
Deployment Management Layer
Simulation Resource Deployer
Simulation Resource Packager
Simulation Resource Installer
Virtual Resource Deployer
Virtual Resource Packager
Virtual Resource Installer
Cloud Infrastructure Deployer
Installation Parameter Collector
OpenStack Deployer
Cloud Simulation Platform Deployer
Simulation/Virtual Resource Scheduler
User Management Layer
User Portal
Web-based Graphic Interface
Command Line Interface
Authentication
Access Control
Simulation Function Layer
Distributed Simulation Workbench
Simulation Service Manager
Simulation Service Monitor
Simulation Service Scheduler
Simulation Resource Manager
Simulation Network Manager
Simulation OS Manager
Simulation Package Manager
Simulation Storage Manager
Simulation Compute Manager
Simulation Resource Pool
Virtual Resource Scheduler
Request Queue
Virtual Resource Pool
Deployment Scheduler
Random
Communication-intensive
Compute-intensive
Integration and Virtualization layer
Virtual Resource Manager
Compute Resource Manager
Network Resource Manager
Image Manager
Public Resource Agent
Raw Resource Layer
Computing Resource
Storage Resource
Network Resource

Fig. 1. Cloud Simulation Framework

to find all packages that require packing. After the search is completed, the Simulation Resource Packager compresses all necessary packages, waiting for transmission. When installation begins, the Simulation Resource Installer is invoked by the installation main program. It decompresses simulation-related packages and stores these packages in a temporary folder. After the Cloud Infrastructure is properly deployed, the Simulation Resource Installer calls Simulation Resource Manager to register the packages currently stored in the temporary folder. Finally, these packages are removed from a temporary folder and stored in the proper cloud storage and block storage again, ready for further usage.

2) Virtual Resource Deployer: The Virtual Resource Deployer functions in a similar manner as the Simulation Resource Deployer. It is responsible for the handling of the simulation units – the virtual instances. When packaging begins, it calls the Virtual Resource Scheduler (in Layer 3, Figure II) and the Virtual Resource Manager (in Layer 4, Figure II) to record the real-time statuses of all computing instances in the entire system. The current statuses of virtual instances are saved by creating images. The Virtual Resource Packager classifies these images according to their owners. During the installation process, the Virtual Resource Installer first checks the Authentication status of the image owners, and then unpacks the images based on image users. Before the image can be used in the new environments, the Virtual Resource installer must trigger the Virtual Resource Scheduler to register the images and rebuild computing instances accordingly. These are created with the same instance type and network topology as in the original cloud simulation platform.

3) Cloud Infrastructure Deployer: The Cloud Infrastructure Deployer disposes components related to cloud middleware and virtualization tools. The configuration of these components are highly based on the new targeting environments. The Installation Parameter Collector is designed to collect relevant parameters required during the installation for the new physical environment, such as compute capability, Hard Disk capacities, and network characteristics. These parameters are passed to the *OpenStack Deployer* and the Cloud Simulation Platform Deployer to install the core components and services as shown in Figure II. After this step, the Simulation/Virtual Resource Scheduler triggers the Simulation Resource Installer and the Virtual Resource Installer to handle relevant softlayer and hardlayer deployment respectively.

4) Cloud Simulation Platform Packaging and Installation Algorithm: The entire packaging and installation process is shown in Algorithm 1. The packaging program begins in line 3 and ends in line 22. After the packaging main process is initialized, the Cloud Infrastructure Deployer calls the Simulation Resource Manager, the Virtual Resource Manager, and the User Portal to stop accepting new tasks, as shown in line 7. Once the current simulation tasks and the virtual instance scheduling tasks are completed, two sub-processes are triggered by the main program to package simulation resources and virtual resources at the same time. On one hand, the Simulation Resource Packager finds the required simulation-related packages in the Simulation Resource Pool with the help of the Simulation Resource Manager. These packages are then copied and compressed from the cloud storage. On the other hand, the virtual computing resources are handled by the Virtual Resource Packager. The Virtual Resource packager calls the Virtual Resource Manager to find the virtual instances, and invokes the Image Manager to build images to save the status of each instance. After both sub-programs are completed as shown in line 14, the packed simulation resources and virtual resources are registered to the main packaging process in the Cloud Infrastructure Deployer. Finally, the Cloud Infrastructure Deployer packs external tools such as the cloud middleware and virtualization components, ready to transmit.

The installation process contains three steps: the cloud infrastructure installation, the simulation resource installation, and the virtual resource installation. The main installation process first invokes the Cloud Infrastructure Deployer to build the basic cloud platform. Before the real installation process begins, the installation parameters that are required during the cloud infrastructure installation are collected by the Installation Parameter Collector. The collecting process may need the authorization of the system administrator before it can begin. Then, the *OpenStack Deployer* and the Cloud Simulation Platform Deployer build core components in the Integration and Virtualization Layer and main services in the Simulation Function Layer. After this, The Simulation/Virtualization Resource Scheduler calls the Simulation Resource Installer and the Virtual Resource Installer to install simulation resources and virtual resources. The Simulation Resource Installer and the Virtual Resource Installer work simultaneously, uncompressing, storing, and registering relevant packages in the new environments. After these two sub-processes are finished, in the end, the Cloud Infrastructure Deployer tests the functions of key services, as well as the availability of the simulation pool and the virtual resource pool, ensuring the correct installation.

B. *User Management Layer*

This layer focuses on security issues in the cloud environment. In this layer, an authentication and access control mechanism is proposed to protect users against threats from both inside the cloud and outside the cloud. In addition, this layer also provides a web-based graphic portal and a command-line interface, through which users can access the simulation resources and the computing capability of the proposed cloud simulation platform. This user portal enables users to design, code, analyze and test complex distributed simulations via lightweight terminals such as laptops, tablets, or even smart phones.

1) Authentication and Access Control Algorithm: Algorithm 2 shows the authentication and access control process. After the user registers in the platform, as shown in line 2, the cloud firewall adds the new user to the acceptance list. In this stage, the user receives authorization to access the shared materials in the Simulation Resource Pool. Users can take advantage of softlayer resources in the pool which gathers and presents codes, templates, solutions, applications, and experiences shared by other cloud users. In order to secure the user-defined simulations in the Simulation Resource Pool, a unique private key is created and stored by the simulation owner, which is required when the user edits or updates its private simulations and application. An X-token, controlled by the Virtual Resource Manager, is provided to enhance the security of user-defined virtual instances. This access key allows the user to access private virtual instances in the Virtual Resource Pool, as shown in line 11. It is required when users create, access, modify, or delete instances. The X-tokens and the passwords function as a two-layered security mechanism, stored separately in the platform. On one hand, if one user is accidentally attacked and loses the control of its instances due to the leak of username/password, the attacker cannot tap into the data inside the user-defined instances due to the lack of the X-tokens; on the other hand, if the X-tokens are stolen, users can stop or suspend their instances immediately through the control console (login by username/password), and then reset the X-tokens. The default lifespan of the X-tokens is 24 hours, which is set the same as the identity components in the internal cloud middleware [30]. With encryption, X-token and virtualization technology, users on the cloud are better isolated and protected so that malicious operations from cloud users or system administrators inside the cloud can be more easily blocked, ensuring that at any time, one virtual instance can only belong to one cloud user for its exclusive usage.

C. *Simulation Function Layer*

This layer provides core functions and services that naturally enable and support different types of distributed simulation standards, such as HLA (High Level Architecture) [3], DIS (Distributed Interactive Simulation) [2], and DDS (Data Distributed Service) [4], which include the Distributed Simulation

Algorithm 2: Authentication & Access Control Algorithm

```
1  Require: $username/password, private\_key, x\_token$
2  $User.signup()$
3  $User.fw.ad(user)$
4  if $User.auth() == True$
5  then
6      $User.access.add(SRP.r)$
7      if $User.auth(key)$
8      then
9          $User.access.add(SRP.w)$
10         if $User.auth(x\_token) is\ not\ expired$ then
11             $User.access.add(VRP.w)$
12         else
13             $User.auth.update()$
14 else
15     $User.access.del()$
```

Workbench, the Simulation Resource Manager, and the Virtual Resource Manager.

1) Distributed Simulation Workbench: The Distributed Simulation Workbench offers users a self-defined distributed simulation foreground, enabling users to focus on design, analysis and testing without further concerns regarding the configuration and maintenance of the underlying physical system. Based on the user's privilege, the Simulation Service Manager furnishes and coordinates simulation services for users, which include an online modeling service, a simulation analysis service, a fault tolerance service, and a load relocation service. The on-line modeling service is designed for modelers to code, submit, execute, and debug through a web-based lightweight interface. Other services can be lightly applied by this interface. Simulation analysis service records and provides end users with the results of simulation execution, simulation-related system logs, and simulation performance statistics, such as CPU load, simulation execution time, and network bandwidth usage, helping to reveal potential issues in the user-designed programs and applications. Fault tolerance service protects user-defined simulations from computation errors, storage errors and network errors, providing rollback and failover mechanisms. The Virtual Machine (VM) migration is utilized as an approach for simulations to recover from failed physical resources. The load relocation service supports user's ability to reschedule virtual instances and modify instance types or network topology during run-time, simplifying the set of comparison experiments. The statuses of these services are tracked by the Simulation Service Monitor. In addition, the Simulation Service Monitor detects and updates the states of the physical machines that periodically compose these simulation services. If any physical error occurs, the Simulation Service Monitor is responsible for blocking relevant resources and informing the cloud simulation platform administrator. The Simulation Service Scheduler manages and maintains the background execution queue for the simulation service requests from multi-users, ensuring proper execution orders of simulations. In addition, the Simulation Service Scheduler calculates the energy consumption of the entire platform and implements energy efficient policies, reducing energy consumption by migrating and centralizing virtual instances and releasing idle physical machines.

2) Simulation Resource Manager: The Simulation Resource Manager concerns the network, OS, software dependency, storage, and computation aspects of the distributed simulations. It also manages the Simulation Resource Pool, which gathers, presents, and shares codes, templates, solutions, and applications of diverse distributed simulation standards among the cloud users. The Simulation network manager handles the network model for the given distributed simulations. It currently supports unicast, multicast, and broadcast approaches used by most distributed simulation standards for communication. In addition, it works with the Virtual Resource Scheduler to initialize the virtual instance's network topology for self-defined virtual instances. The simulation OS Manager controls and stores a list of images of OS for booting virtual instances. Users are able to modify system settings, such as firewall rules, partitioning of the hard disk, software repository, and other system information before the virtual instances are booted. The Simulation Package Manager is responsible for arranging the simulation of dependent packages. This component natively supports standards including HLA, DIS, and DDS, and it automatically configures and installs dependency packages that each corresponding implementation may further require. In order to enable the diversity of distributed simulations, the Simulation Package Manager also allows users to manage and define self-designed packages for specific scenarios. The Simulation Storage Manager provides two types of storage in the background for simulations: cloud storage and block storage. Cloud storage is mainly used as replicas to backup resources in the Simulation Resource Pool in the event that local resource storage servers fail. In addition, these cloud replicas can be easily shared among users inside and outside the platform. The block storage saves all necessary information that this cloud simulation platform requires during runtime and it utilizes the Network File System (NFS) to accelerate the scheduling process of simulation resources. The Simulation Compute Manager concerns the compute capability for the current distributed simulation. It communicates with the Simulation Service Monitor to check local compute resources. If local resources are not sufficient for current simulations, it calls the Virtual Resource Manager and the Public Resource Agent to enable additional public computing resources to fulfil the computation requirements.

3) Virtual Resource Manager: Due to the diversity of modeling and simulation applications, computing resource requirements differ. The Virtual Resource Scheduler is utilized to control the granularity of resources so that the need of diverse simulation applications can be better met. Based on the requirement information from users, the Virtual Resource Scheduler helps to arrange the simulation environment based on the parameters of the computing unit, such as the number of virtual processors, memory size, network topology, and the quantity of virtual machines. The Virtual Resource Scheduler invokes corresponding components in the Virtual Resource Manager and in the Public Resource Agent, discovering proper computing resources and instantiating virtual computing instances in the Virtual Resource Pool.

4) Scheduling Algorithm and Dynamic VM Reallocation: Several resource scheduling algorithms are provided for instance creation, as shown in Algorithm 3: random, computation-intensive, and communication-intensive. N in line 1 defines the number of virtual resources that require deployment in simulations. The default value is 0, which means the current available resources are already adequate, and there is no need for additional virtual resources to be scheduled from the Virtual Resource Pool. In this context, IT in line 1 is short for the instance type. This parameter expresses the type of virtual instances that may be further initiated and utilized. It contains computing and storage capability information such as processors, RAM and hard disk. SA, which is first shown in line 1, stands for the type of the scheduling algorithm. In Random, as shown from line 4 to line 14, the Virtual Resource Scheduler randomly creates instances. First, from the Virtual Resource Pool, a queue of all current available resources are listed. Then, based on the number of available resources, a random integer number n is selected according to a normal distribution. The resource whose index number is n in the resource queue is chosen as the destination physical host. After this selection, the request to boot the virtual instance is queued in the background and then redirected to the destination's physical host. Finally, the virtual instance is instantiated on the targeting physical host, and the system records and updates the resource information in the Virtual Resource Pool. At this time, one round of scheduling ends and the next round is ready to begin. Differently from the random scheduling, the computation and communication-intensive approaches make use of a greedy technique for allocation of resources. In computation-intensive scheduling, shown from line 15 to line 24, a current available resource queue is first created in the same manner as the random scheduler. Then, the resource whose current computing capability is the largest in the queue is selected as the destination resource. In this mode, large virtual instance types are particularly chosen as IT for simulations involving large computation tasks. With this type of scheduling and the large virtual instance type, the rounds of scheduling can be noticeably reduced. In the communication-intensive mode shown between line 25 and line 35, the first resource in the current available resource queue is selected as the destination host. Then, this selected physical host attempts to schedule and boot as many virtual instances as possible on itself until its compute capability is exhausted. In this way, communication-intensive simulation entities can be likely to exist in the same physical host, so that the communication distance of simulation entities is reduced. After each round of scheduling, a virtual instance migration may be triggered by the Simulation Service Scheduler, reallocating virtual instances and centralizing simulation tasks. In this case, idle physical resources can be released and the energy consumption of the system can be reduced.

Algorithm 3: Scheduling Algorithm & Dynamic VM Reallocation

```
1  Require: $N, IT, SA$
2  $N\_update()$
3  $IT\_update()$
4  if $SA\ is\ Random\ Scheduling$
5  then
6      while $N! = 0$ do
7          $ava\_res\_que \Leftarrow res\_pool.search(IT)$
8          $n \Leftarrow random(0, ava\_res\_que.index)$
9          $des\_res \Leftarrow ava\_res\_que.get(index = n)$
10         $boot\_que.add(des\_res, IT)$
11         $sys.rec()$
12         $sys.update()$
13         $N\_update()$
14         $VM.realoc()$
15 if $SA\ is\ Compute - intensive\ Scheduling$
16 then
17     while $N! = 0$ do
18         $ava\_res\_que \Leftarrow res\_pool.sort(IT)$
19         $des\_res \Leftarrow ava\_res\_que.find\_low\_load()$
20         $boot\_que.add(des\_res, IT)$
21         $sys.rec()$
22         $sys.update()$
23         $N\_update()$
24         $VM.realoc()$
25 if $SA\ is\ Communicate - intensive\ Scheduling$
26 then
27     while $N! = 0$ do
28         $ava\_res\_que \Leftarrow res\_pool.search(IT)$
29         $des\_res \Leftarrow ava\_res\_que.get(index = 1)$
30         while $des\_res\ is\ not\ exhausted\ or\ N! = 0$ do
31             $boot\_que.add(des\_res, IT)$
32         $sys.rec()$
33         $sys.update()$
34         $N\_update()$
35         $VM.realoc()$
```

D. Integration and Virtualization Layer

In order to utilize the raw computing capacity from diverse resources, the Integration and Virtualization Layer is designed to coordinate these resources and implement virtualization technologies. This layer contains two core components: the Virtual Resource Manager and the Public Resource Agent. Although these components are not visible to end users, they play an important role in automatic configuration, management and maintenance of the underlying raw resources while hiding complex details of system management.

1) Virtual Resource Manager: The Virtual Resource Manager plays a crucial role in integrating physical resources, which includes three sub-managers: the Compute Resource Manager, the Image Manager, and the Network Manager. The Virtual Resource Manager maintains and manages the cloud middleware and virtualization tools. Based on the user-defined requirements from the aforementioned Virtual Resource Scheduler, the Virtual Resource Manager calls its relevant sub-managers to process compute-related, storage-related, or network-related management programs. These programs work as based for building the experimental or execution environment. The Compute Resource Manager deals with the establishment of virtual instances based on the selected scheduling algorithm and the characteristics of the user-defined virtual instances. The Image Manager handles the operating system and simulation-related soft-layer issues. According to various implementations of different distributed simulation standards, the Image Manager pre-arranges the operation system and the simulation-related packages for the virtual instances before it is booted from the Compute Resource Manager. The Network Manager controls the network topology of the virtual instances. In this case, these virtual instances can be instantiated in the same physical host or among multiple adapters, based on simulation requirements. In addition, the Network Manager maintains the key bridge that is designed to support communications between local virtual instances and public instances. It virtualizes the public portal of the cloud simulation platform, binding the virtual portal and public tunnels.

2) Public Resource Agent: In order to achieve rapid elasticity in the cloud simulation platform, the Public Resource Agent is designed to increase computational capabilities during runtime. This is achieved by introducing public physical resources such as instances in the public cloud (eg., Amazon EC2 instances). To coordinate the local resources and public resources, the diversity of the public network should be taken into careful consideration. In the public network, multicast and other approaches of package transmission protocols, which are widely used in many distributed simulation standards, may not be well supported by routers, switches, or hubs, which are in the communication path between cloud instances deployed over different remote networks. For instance, routers drop multicast messages by default if the destination resource and the original resource are not set in the same subnet. The Public Resource Agent is introduced to manage the public networks to meet the demands of distributed simulation. First, the Public Resource Agent builds specific virtual tunnels for each public resource based on the physical network, penetrating the public Internet and encapsulating packages and messages that need to be sent and received. After the virtual tunnels are initiated, the relevant Firewall rules are modified to support the established communication among local instances and public instances, redirecting packages and messages to the proper destinations. The Public Resource Agent listens to the Virtual Resource Scheduler and analyzes the simulation requirements defined by users, so it only invokes public instances if local current computational capability are not sufficient and the Service Level Agreement of the user is not violated.

3) Simulation Execution Algorithm: In Algorithm 4, the underlying simulation execution process is described step-by-step. First, when the Virtual Resource Scheduler receives simulation tasks from the user, the identity and priority of a user is verified, as shown in line 3. This ensures that the user can reach full access to the Simulation Resource Pool and the Virtual Resource Pool. The detailed authentication and access control process is demonstrated in Algorithm 2. Then, based on the information collected from the Simulation Service Monitor, the Virtual Resource Scheduler decides whether it is necessary to invoke public instances. On one hand, if local computing capabilities are sufficient (as shown in lines 5 to 10), the Public Resource Agent is not contacted. In this situation, the Virtual Resource Manager calculates the number of virtual instances that must be instantiated and collects user-defined parameters such as SA, IT, and simulation related packages. The Virtual Resource Manager then boots the instances in line 9, as illustrated in Algorithm 3. On the other hand, if local computing resources do not meet the requirements, the Public Resource Agent is called to arrange additional computing capabilities, as shown in lines 12 to 17. In this case, after the local computing resources are exhausted, the Public Resource Agent configures, links and bridges the public instances to local user groups, enabling communications among local virtual instances and public instances in the same user-defined group. In this stage, the virtual resources are properly prepared, and are ready to execute softlayer commands. Before the distributed simulation applications begins to run, the simulation scripts are supposed to map to destination instances by the Simulation Resource Manager, as shown in lines 20 to 25. The Simulation Resource Manager pairs the index of user-defined simulation packages in the Simulation Resource Pool and the index of virtual instances in the user's private Virtual Resource Pool, delegating simulation executing packages to destination instances. After each round of simulation, the executing packages can be remapped to different instances for comparison study. After the simulations are completed, as shown in lines 26 to 32, the simulation-related packages and logs are uploaded to Hard Disk storage and cloud storage. The status of computing resources is saved in the system as images and the computing capabilities can properly store and reused later for a future instantiation.

E. Raw Resource Layer

The Raw Resource Layer is a pool of untreated resources, such as computing resources, storage resources, and network resources. The whole cloud simulation platform is built on top of these resources. In this layer, physical hosts are not configured and coordinated, as they contain only an operating system and basic software that such an operating system brings.

IV. Experiments and Result Analysis

The prototype of this platform has been implemented and deployed on a cluster, on one single machine, and on the Amazon EC2 instance respectively to test and compare the performance. The cluster was composed of 20 nodes, and each node of the cluster contained a Quad Core 2.40GHz Intel(R) Xeon(R) CPU and 8 gigabytes of DIMM DDR RAM memory. All nodes were interconnected through a Myrinet optical network that allowed data transmission up to 2 gigabytes per second. The single machine setup contained a Quad Core 2.40GHz Intel CPU (4700MQ) and 16 gigabytes of RAM. VMware Workstation [32] was used as the virtualization tool, which created one management node and two computing nodes. The details of the nodes are shown in Table II. For public cloud instances in our experiments, T2.micro instance from AWS [28] was employed, which provided a basic

Algorithm 4: Simulation Execution Algorithm

```
1  Require: $VRP.w, N, IT, SA, P.$
2  Note: $VRP - Virtual\ Resource\ Pool,$
   $P - Simulation\ related\ Packages,$
   $VRS - Virtual\ Resource\ Scheduler,$
   $VRM - Virtual\ Resource\ Manager,$
   $SRM - Simulation\ Resource\ Manager,$
   $CRM - Compute\ Resource\ Manager,$
   $IM - Image\ Manager,\ NM - Network\ Manager,$
   $PRA - Public\ Resource\ Agent,$
   $SRP - Simulation\ Resource\ Pool.$
3  while $!VRS.end()\ \&\&\ User.access(VRP.w)$ do
4      while $VRS.cur(VRP) <= VRS.req()$ do
5          if $VRS.req() <= VRS.cur(VRP.loc())$
6          then
7              $VRM.N.set() \Leftarrow VRS.req() - VRS.cur(VRP))$
8              $VRM.IM.set(P) \Leftarrow SRM.map(SRP)$
9              $VRM.boot(VRM.CRM(N, IT),$
               $VRM.NM(SA), VRM.IM(P))$
10             $VRS.update()$
11         else
12             $VRM.N.set() \Leftarrow$
               $VRS.req() - VRS.cur(VRP.loc())$
13             $VRM.IM.set(P) \Leftarrow SRM.map(SRP)$
14             $VRM.boot(VRM.CRM(VRS.cur(VRP.loc()), IT),$
               $VRM.NM(SA), VRM.IM(P))$
15             $PRA.set(VRM.N, PRA.getpair(PL),$
               $VRM.IM(P))$
16             $VRM.NM.bridge.setgroup(VRM.cur(user),$
               $PRA.cur(user))$
17             $VRS.update()$
18     $VRS.end() \Leftarrow User.def()$
19 while $!SRM(end)\&\&User.access(VRP.w)$ do
20     $VRM.insList.set(user) \Leftarrow$
       $VRM.cur(VRP.user.simX)$
21     $SRM.simList.set(user) \Leftarrow$
       $SRM.upload(SRP.user.simX)$
22     $SRM.mapSimPair(VRM.insList(user),$
       $SRM.simList(user))$
23     $SRM.execute()$
24     $SRM(end) \Leftarrow User.def()$
25 if $SRM(end)$
26 then
27     $SRM.upload()$
28     $SRM.SRP.save()$
29     $VRM.IM.save()$
30     $VRM.recycle()$
31     $User.access \Leftarrow NONE$
```

configuration: one VCPU and 1 GB of memory. Based on different deployment environments, instance types, scheduling algorithms, and experiments were implemented to evaluate the energy consumption, resource provision efficiency, and simulation performance.

The first experiment concerned the preparation time of a simulation environment, including the computing resource scheduling time, raw resource providing time, and softlayer resource packaging time. The computing resource scheduling time involves the components of the whole platform; the scheduling time starts when the User Portal receives the user's resource requests and ends after these requests are properly stored, recorded, and processed in the Request Queue managed by the Virtual Resource Manager. The rare resource provision process includes the selection of computing resources in the raw computing pool, the mapping of OS and computing resources, and the establishing of instances. The softlayer resource packaging procedure focuses on the building of a simulation-run-ready environment, which contains the simulation-related package dependency analysis, package searching, package configuration, and package installation. In this experiment, the Linux system was used as the operating system for the virtual instances; CentOS 6 [33] and Fedora 17 [34] were used. The simulation applications used one testing restaurant simulation application implemented in Portico java-based RTI [35] as the HLA simulation; one simple Sender/Receiver application implemented in OpenDIS [36] as the DIS simulation; and one HelloWorld application of OpenSplice [37] as the DDS simulation. The virtual instance type for the cluster and the single machine employed in this experiment is shown in Table III. In this experiment, random scheduling was selected as the scheduling algorithm for the computing instances.

TABLE II. SINGLE MACHINE ENVIRONMENT

Host	VCPU	RAM	Bandwidth	Hard Disk
Management	2	2 G	100 Mbps	30 G
Computing 1	4	4 G	100 Mbps	40 G
Computing 2	1	2 G	100 Mbps	20 G

TABLE III. VIRTUAL RESOURCE LIST

Virtual Instance Type	VCPUs	RAM(MB)	Hard Disk(GB)
Mini	1	512	0
Small	1	1024	10G
Medium	2	2048	20G
Large	4	4096	40G
Self-defined	1 to 4	512 to 4096	0 to 40G

The results of the first experiment are shown in Table IV, V, and VI. $Time\_sche$ refers to the computing resource scheduling time, and $Time\_f17$ and $Time\_cen6$ indicate the raw resource providing time with a Fedora 17 OS and a CentOS 6 OS respectively in Tables IV and V. As the results show, on one hand, the cluster mode achieves a better performance in resource scheduling due to the better computing capability of the management node. On the other hand, the raw resource provision time of the single machine mode is much shorter because its scale of computing resource pool is much smaller, containing only two computing nodes. In this case, the raw resources can be recycled, regenerated, and reused efficiently after they are released for each round. The results in Table VI show the softlayer resource packaging time. The softlayer packing process occurs on the management node for both single machine mode and cluster mode. The process time depends on the size of the packages and the compute capability of the management node. Because the CPU in the cluster is better, the process time of the cluster is shorter. The package size of

TABLE IV. COMPUTING RESOURCE PREPARATION IN THE SINGLE MACHINE MODE

Virtual Resource Type	Time_sche(s)	Time_f17(s)	Time_cen6(s)
Mini	2.064	3.445	3.945
Small	2.278	3.705	3.998
Medium	2.162	3.753	4.011

TABLE V. COMPUTING RESOURCE PREPARATION IN THE CLUSTER MODE

Virtual Resource Type	Time_sche(s)	Time_f17(s)	Time_cen6(s)
Mini	1.389	16.871	17.539
Small	1.421	17.278	17.932
Medium	1.322	17.502	18.244
Large	1.433	17.892	18.577

DIS, DDS and HLA differs, which leads to different process time in each mode. This indicates that based on the proposed cloud simulation scheme a simulation-run-ready environment can be completely brought up and ready for execution within 250 seconds in the worst case among the three simulation standards. Different types of distributed simulation standards, operating systems, and computing instances are automatically configured, instantiated and managed based on the simulation requirements.

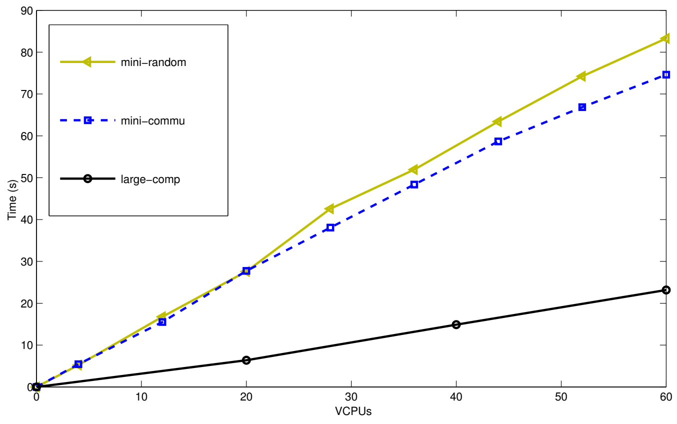


Fig. 2. Scheduling Algorithm Performance Evaluation

The second experiment evaluated the performance of scheduling algorithms, shown in Algorithm 3, which were natively supported in this cloud scheme when simulations scaled up in the cluster mode. In this experiment, mini virtual instances and large virtual instances were introduced as computing instance examples. The scheduling time recorded in this experiment was the same as T_sche, as explained in the first experiment.

The second experiment consisted in evaluating the delay on

TABLE VI. PACKAGES HANDLING PROCESS

Platform Mode	Time_hla(s)	Time_dis(s)	Time_dds(s)
Single Node	189.845	141.728	234.013
Cluster	144.672	97.412	189.844

TABLE VII. CLOUDSIM SIMULATION PARAMETERS

Scale	20 computing nodes and 1 management node
Number of VM Required	10 (Small, Medium, and Large)
VM Migration Time	23 Seconds
VM Scheduling Time	1.4 Seconds
Rated Power	355 W
Scheduling Time Interval	60 Minutes
Scheduling Algorithm	Random Scheduling

using different scheduling techniques. As depicted in Figure 2, when simulations scaled up, the time of scheduling increased in an almost linear trend, except for some slight fluctuations caused by background load. The background load originated from the cloud system, which had to continue running its services for periodically detecting and updating the status of the physical machines that composed the environment. Such services coordinated the resources by observing and recording their power status, network availability, current free RAM, and other characteristics. As the results show, Computation-intensive scheduling (from line 25 to line 35 in Algorithm 3) achieved the best performance, as it minimized the rounds of resource scheduling. As for random scheduling (from line 4 to line 14 in Algorithm 3) and communication-intensive scheduling (from line 15 to line 24 in Algorithm 3), the latter presented overall performance with an improvement of 10 percent. This difference was generated because in the communication-intensive scheduling, the search queue for available resources in the resource queue was shorter – the first available physical host in the resource queue was selected as the destination host. For random scheduling, the average scheduling search length was $que.len/2$, which was much longer than the communication-intensive. Another reason for obtaining these results was due to the built-in patten in recording scheduling information. The Communication-intensive scheduling recorded only when one physical host was exhausted while random scheduling needed to record immediately after the conclusion of each scheduling round. For different types of scheduling algorithms in this platform, the scheduling time for one instance was less than 1.5 seconds. As described in [47], the proposed script-based solution can save modelers substantial amount of time to deploy large scale complex simulation environment compared to a manual deployment.

The third experiment addressed the energy consumption of the proposed scheme. Due to the lack of temperature sensors in the hardware, CloudSim [40] was employed to estimate and observe the performance of the energy-saving approach in the scheme. The parameters in the experiment, such as the VM migration time and VM scheduling time, were based on the real test results of the machines in the cluster, shown in Table VII. The simulation used a long-term computing-intensive simulation that consumed almost 100% CPU and ran for approximately 24 hours. The type of the computing instances used in this experiment are the same as the ones listed in Table III.

Tables VIII and IX show the results of the simulation task scheduling and executing processes with and without energy-

TABLE VIII. SCHEDULING WITH MIGRATION

Round	1 and 2	3 and 4	5 and 6	7 and 8	9 and 10
Current VM Type	S, M	S, S	S, M	L, S	L, M
Migration	1	1	2	0	0
Current Power	712.538	1058.662	1429.614	2485.944	3551.349
Current Hosts in Use	1	2	2	4	4

TABLE IX. SCHEDULING WITHOUT MIGRATION

Round	1 and 2	3 and 4	5 and 6	7 and 8	9 and 10
Current VM Type	S, M	S, S	S, M	L, S	L, M
Current Power	1065.54	2487.13	3909.884	5333.634	6758.49
Current Hosts in Use	2	4	6	8	10

aware components. As the results indicate, the proposed energy-saving scheme can reduce large amount of energy by centralizing simulation tasks and releasing idle computing resources. In the experiment scenario, 47.45% energy is saved in total 10 hours on average. With the proposed distributed simulation based energy-aware components, the migration consumed approximately only 300 seconds due to the migration for all the sample simulation applications during the period of 10 hours. As a result, this approach is viable and beneficial for distributed simulations that run for periods of time in the scale of hours.

The fourth experiment concerned the performance loss in simulation execution that this multi-layered cloud platform may cause when compared with the native Grid platform. In the experiment, comparisons are made with the proposed simulation Cloud deployed on our different established scenarios: on the cluster mode, on the single machine mode, and on the public cloud computing instances from AWS [28]. In the cluster mode, the large virtual resource type (as shown in Table III), was used to boot fifteen virtual instances on the cloud while the grid used fifteen physical hosts directly. In the single machine mode, a small computing instance type, shown in Table III, was used for the cloud while the grid used the Host Computing 2 as shown in Table II. The simulation application implemented in the experiment was based on the Portico RTI restaurant example [35], the same as the first experiment. In the application, the sample federate produced a controlled synthetic load, which initially involves creating a pseudo-random number x following a normal distribution and then introducing a recursive procedure for x turns, to exhaust the computational capability for the instance in which it ran. After the computation, the sample federates communicated with each other about their computation results. Different iterations were used to measure the simulation execution performance.

The results for this fourth experiment are depicted in Figure 3. Regarding the experiments in the cluster mode, which involved both communication tasks and computing tasks, the Grid outperforms the Cloud in terms of the simulation execution time: when the experiment reached 200 iteration rounds, the performance of the Grid was 3.94% better than the Cloud. The reason for the performance loss of the Cloud is the utilization of Virtualization Technology (VT) in the Integration and Virtualization Layer, according to [38]. As for the experiments in the single machine mode that only

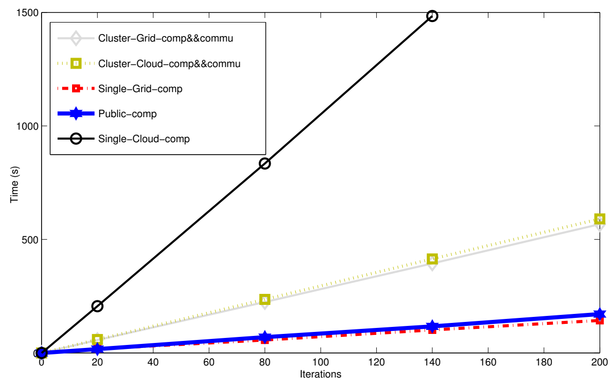


Fig. 3. Simulation Performance between Cloud Platform and Grid Platform

contained computing tasks, the performance of the Grid was much better: when the iteration number reached 140, the Grid performed approximately 15 times better than the Cloud. This result was obtained due to the nested virtualization for the cloud scheme in the single machine mode. As a result, an additional layer of abstraction and encapsulation in a pre-existing virtualized environment was created. This may influence the performance of executing simulations, according to the discussion and results from [39]. It is worth mentioning that for the public instance mode that contained the same computing task as the Grid in the single machine mode, the performance of the Amazon public cloud instance was only 19.8% slower than the latter when the iteration round reached 200, which indicates that the public cloud is very promising for holding computation-intensive distributed simulations.

Based on the results of the aforementioned four experiments, we can observe that the proposed multi-layered cloud simulation platform enables distributed simulations in the cloud environments, providing elasticity, automatic management of diverse resources, fast deployment, security, and reduction of energy costs. There is little performance loss in terms of the simulation execution time when compared with the native Grid platform, mainly due to the delays caused by the virtualization of the Cloud. However, this overhead is acceptable due to the benefits of Cloud Computing listed in the early sections of this paper.

V. CONCLUSION

In this paper, a layered cloud platform was proposed for the distributed simulations. Concerning usability, energy consumption, security, reliability, and elasticity, relevant components are implemented to support fast deployment, to ease the management of underlying resources, to reduce energy usage, and to enable fine-grained resource handling during runtime. The design of a multi-layered scheme in different scenarios was described. In the experiments, the performance of this cloud simulation platform was evaluated and discussed in detail. Based on the experimental results, we conclude that the use of cloud technologies is a promising method for facilitating distributed simulations, especially when the network environment presents greater efforts towards optimization and performance.

ACKNOWLEDGMENTS

This work is partially supported by the Canada Research Chairs Program and NSERC Canada.

REFERENCES

[1] Fujimoto, Richard M. *Parallel and Distributed Simulation Systems.* Wiley Interscience, January 2000.

[2] *IEEE Standard for Distributed Interactive Simulation–Application Protocols*. IEEE Computer Society. 1278.1, 2012.

[3] S. I. S. C. (SISC). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) Framework and Rules.* IEEE Computer Society, September 2000.

[4] *Data Distribution Service for Real-time Systems.* Object Management Group (OMG). URL: http://www.omg.org/spec/DDSII.2.

[5] Foster, Ian and Kesselman, Carl and Tuecke, Steven. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations.* Journal of the High Performance Computing Applications, 15(3): 200-222, 2001.

[6] *Globus*. University of Chicago, February 2008. URL: http://www.globus.org/.

[7] Foster, Ian and Kesselman, Carl and Nick, Jeffrey M and Tuecke, Steven. *Grid services for distributed system integration.* IEEE Computer, 35(6): 37-46, 2002.

[8] Armbrust, Michael and Fox, Armando and Griffith, Rean and Joseph, Anthony D and Katz, Randy and Konwinski, Andy and Lee, Gunho and Patterson, David and Rabkin, Ariel and Stoica, Ion and Zaharia, Matei. *A view of cloud computing.* Communications of the ACM, 53(4): 50-58, 2010.

[9] Foster, Ian and Zhao, Yong and Raicu, Ioan and Lu, Shiyong. *Cloud computing and grid computing 360-degree compared.* in: Proc. of the Grid Computing Environments Workshop, page: 1-10, 2008.

[10] Vaquero, Luis M and Rodero-Merino, Luis and Caceres, Juan and Lindner, Maik. *A break in the clouds: towards a cloud definition.* ACM SIGCOMM Computer Communication Review, 39(1): 50-55, 2008.

[11] Buyya, Rajkumar and Yeo, Chee Shin and Venugopal, Srikumar. *Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities.* in: Proc. of the High Performance Computing and Communications, HPCC'08, page: 5-13, 2008.

[12] Villegas, David and Rodero, Ivan and Fong, Liana and Bobroff, Norman and Liu, Yanbin and Parashar, Manish and Sadjadi, S Masoud. *The role of grid computing technologies in cloud computing.* Handbook of Cloud Computing, page: 183-218, 2010.

[13] Mell, Peter and Grance, Tim. *The NIST definition of cloud computing.* National Institute of Standards and Technology, 53(6): 50, 2009.

[14] Xie, Yong and Teo, Yong Meng and Cai, Wentong and Turner, Stephen John. *Service provisioning for HLA-based distributed simulation on the Grid.* in: Proc. of the Principles of Advanced and Distributed Simulation, page: 282-291, 2005.

[15] Theodoropoulos, Georgios and Zhang, Yi and Chen, Dan and Minson, Rob and Turner, Stephen John and Cai, Wentong and Xie, Yong and Logan, Brian. *Large scale distributed simulation on the grid.* in: Proc. of the Cluster Computing and the Grid, 2(63): 1-8, 2006.

[16] Rycerz, Katarzyna and Bubak, Marian and Malawski, Maciej and Sloot, Peter. *A framework for HLA-based interactive simulations on the grid.* Journal of the Simulation, 81(1): 67-76, 2005.

[17] Cai, Wentong, Stephen John Turner, and Hanfeng Zhao. *A load management system for running HLA-based distributed simulations over the grid.* in: Proc. of the Distributed Simulation and Real-Time Applications, page: 7-14, 2002.

[18] Boukerche, Azzedine and Das, Sajal K. *Reducing null messages overhead through load balancing in conservative distributed simulation systems*. Journal of Parallel and Distributed Computing, 64(3): 330-344, 2004.

[19] Boukerche, Azzedine and De Grande, Robson Eduardo. *Dynamic load balancing using grid services for hla-based simulations on large-scale distributed systems.* in: Proc. of the IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, page: 175-183, 2009.

[20] De Grande, Robson Eduardo and Boukerche, Azzedine. *Distributed dynamic balancing of communication load for large-scale HLA-based simulations*. in: Proc. of the Computers and Communications, page: 1109-1114, 2010.

[21] Robson Eduardo De Grande and Azzedine Boukerche. *Predictive dynamic load balancing for Large-Scale HLA-based simulations*. in Proc. of the IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, page: 4-11, 2011.

[22] Fujimoto, Richard M and Malik, Asad Waqar and Park, Alfred J. *Parallel and distributed simulation in the cloud.* SCS M&S Magazine, 3: 1-10, 2010.

[23] Malik, Asad Waqar and Park, Alfred and Fujimoto, Richard M. *Optimistic synchronization of parallel simulations in cloud computing environments.* in: Proc. of the IEEE Conference on Cloud Computing, page: 49-56, 2009.

[24] Liu, Xiaocheng and Qiu, Xiaogang and Chen, Bin and Huang, Kedi. *Cloud-based Simulation: the State-of-the-art Computer Simulation Paradigm.* in: Proc. of the ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation, page: 71-74, 2012.

[25] He, Heng and Li, Ruixuan and Dong, Xinhua and Zhang, Zhi and Han, Hongmu. *An Efficient and Secure Cloud-Based Distributed Simulation System.* Journal of the Applied Mathematics & Information Sciences, 6(3): 729-736, 2012.

[26] Jung, In-Yong and Han, Byong-John and Jeong, Chang-Sung. *Provisioning On-Demand HLA/RTI Simulation Environment on Cloud for Distributed-Parallel Computer Simulations.* Mobile, Ubiquitous, and Intelligent Computing, page: 329-334, 2014.

[27] Li, Bo Hu and Chai, Xudong and Hou, Baocun and Yang, Chen and Li, Tan and Lin, Tingyu and Zhang, Zhihui and Zhang, Yabin and Zhu, Wenhai and Zhao, Zenghui. *Research and application on cloud simulation.* in: Proc. of the 2013 Summer Computer Simulation Conference, page: 34(1)-34(14), 2013.

[28] *Amazon EC2 Instances*. Amazon Web Services. URL: http://aws.amazon.com/ec2/instance-types/.

[29] Feng, Shaochong and Di, Yanqiang and Meng, Zhu Xianguo. *Remodeling traditional rti software to be with paas architecture.* in: Proc. of the Computer Science and Information Technology (ICCSIT), 1: 511-515, 2010.

[30] *OpenStack Grizzly*. OpenStack. January, 2013. URL: http://www.openstack.org/software/grizzly/

[31] Kivity, Avi and Kamay, Yaniv and Laor, Dor and Lublin, Uri and Liguori, Anthony. *kvm: the Linux virtual machine monitor.* in: Proc. of the Linux Symposium, 1: 225-230, 2007.

[32] Ward, Brian. *The book of VMware: the complete guide to VMware workstation.* No Starch Press, 2002.

[33] *Centos Image.* Community ENTerprise Operating System. URL: http://openstack.redhat.com/.

[34] *Fedora Image.* Fedora. URL: http://berrange.fedorapeople.org/images/.

[35] Pokorny, Tim and Fraser, Michael. *poRTIco*. 2009. URL: http://www.porticoproject.org/.

[36] *OpenDIS*. URL: http://open-dis.sourceforge.net/.

[37] *OpenSplice*. URL: http://www.prismtech.com/opensplice.

[38] Barham, Paul and Dragovic, Boris and Fraser, Keir and Hand, Steven and Harris, Tim and Ho, Alex and Neugebauer, Rolf and Pratt, Ian and Warfield, Andrew: Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5): 164-177, 2003.

[39] Ben-Yehuda, Muli and Day, Michael D and Dubitzky, Zvi and Factor, Michael and Har'El, Nadav and Gordon, Abel and Liguori, Anthony and Wasserman, Orit and Yassour, Ben-Ami. *The Turtles Project: Design and Implementation of Nested Virtualization.* in: Proc. of the OSDI. 10: 203-216, 2010.

[40] Calheiros, Rodrigo N and Ranjan, Rajiv and Beloglazov, Anton and De Rose, César AF and Buyya, Rajkumar. *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.* Journal of the Software: Practice and Experience 41(1): 23-50, 2011.

[41] Dean, Jeffrey and Ghemawat, Sanjay. *MapReduce: simplified data processing on large clusters.* Communications of the ACM, 51(1): 107-113, 2008.

[42] DeCandia, Giuseppe and Hastorun, Deniz and Jampani, Madan and Kakulapati, Gunavardhan and Lakshman, Avinash and Pilchin, Alex and Sivasubramanian, Swaminathan and Vosshall, Peter and Vogels, Werner. *Dynamo: amazon's highly available key-value store.* ACM SIGOPS Operating Systems Review, 41(6): 205-220, 2007.

[43] Lazri, Kahina and Laniepce, Sylvie and Ben-Othman, Jalel. *Reconsidering Intrusion Monitoring Requirements in Shared Cloud Platforms.* in: Proc. of the Availability, Reliability and Security (ARES), page: 630-637, 2013.

[44] Lazri, Kahina and Laniepce, Sylvie and Ben-Othman, Jalel. *When Dynamic VM Migration Falls Under the Control of VM Users.* in: Proc. of the Cloud Computing Technology and Science (CloudCom), page: 395-402, 2013.

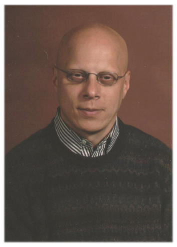
[45] Lazri, Kahina and Laniepce, Sylvie and Zheng, Haiming and Ben-Othman, Jalel. *AMAD: Resource Consumption Profile-Aware Attack Detection in IaaS Cloud.* in: Proc. of the Utility and Cloud Computing (UCC), page: 379-386, 2013.

[46] Heilig, Leonard and VoB, Stefan. *A scientometric analysis of cloud computing literature.* Journal of the Cloud Computing, 2(3): 266-278, 2014.

[47] Talwar, Vanish and Milojicic, Dejan and Wu, Qinyi and Pu, Calton and Yan, Wenchang and Jung, Gueyoung. *Approaches for service deployment*. Journal of the Internet Computing, 9(2): 70-80, 2005.

Shichao Guan received his B.Sc. in Computer Science from the Beijing Electronic Science and Technology Institute, China in 2010 and M.Sc. in ECE from the University of Ottawa, Canada in 2015. He is now a Ph.D. student at PARADISE Research Laboratory at University of Ottawa, Canada. His research interests include Cloud Computing, distributed simulations, Mobile Cloud, and performance evaluation.

Robson Eduardo De Grande is currently the Network Manager of the NSERC DIVA Strategic Research Network and a Senior Research Associate at the School of Electrical Engineering and Computer Science, University of Ottawa, Canada. He received his Ph.D. degree at the University of Ottawa in 2012 and his M.Sc. and B.Sc. degrees in Computer Science from the Federal University of Sao Carlos, Brazil in 2006 and 2004, respectively. He served as TPC Chair of IEEE/ACM DS-RT 2014. His research interests include large-scale distributed and mobile systems, Cloud computing, high performance computing, performance modeling and simulation, computer networks, vehicular networks, and distributed simulation systems.



Azzedine Boukerche (FIEEE, FEiC, FCAE, FAAAS) is a full professor and holds a Canada Research Chair position at the University of Ottawa (Ottawa). He is the founding director of the PARADISE Research Laboratory, School of Information Technology and Engineering (SITE), Ottawa. Prior to this, he held a faculty position at the University of North Texas, and he was a senior scientist at the Simulation Sciences Division, Metron Corp., San Diego. He was also employed as a faculty member in the School of Computer Science, McGill University, and taught at the Polytechnic of Montreal. He spent an year at the JPL/NASA-California Institute of Technology, where he contributed to a project centered about the specification and verification of the software used to control interplanetary spacecraft operated by JPL/NASA Laboratory. His current research interests include Wireless Ad Hoc and sensor networks, wireless networks, mobile and pervasive computing, wireless multimedia, QoS service provisioning, performance evaluation and modeling of large-scale distributed systems, distributed computing, large-scale distributed interactive simulation, and parallel discrete-event simulation. He has published several research papers in these areas. He served as a guest editor for the Journal of Parallel and Distributed Computing (special issue for routing for mobile ad hoc, special issue for wireless communication and mobile computing, and special issue for mobile ad hoc networking and computing), ACM/Kluwer Wireless Networks, ACM/Kluwer Mobile Networks Applications, and Journal of Wireless Communication and Mobile Computing. He has been serving as an Associate Editor of ACM Computing Surveys, IEEE Transactions on Parallel and Distributed systems, IEEE Transactions on Vehicular Technology, Elsevier Ad Hoc Networks, Wiley International Journal of Wireless Communication and Mobile Computing, Wileys Security and Communication Network Journal, Elsevier Pervasive and Mobile Computing Journal, IEEE Wireless Communication Magazine, Elseviers Journal of Parallel and Distributed Computing, and SCS Transactions on Simulation. He was the recipient of the Best Research Paper Award at IEEE/ACM PADS 1997, ACM MobiWac 2006, ICC 2008, ICC 2009 and IWCMC 2009, and the recipient of the Third National Award for Telecommunication Software in 1999 for his work on a distributed security systems on mobile phone operations. He has been nominated for the Best Paper Award at the IEEE/ACM PADS 1999 and ACM MSWiM 2001. He is a recipient of an Ontario Early Research Excellence Award (previously known as Premier of Ontario Research Excellence Award), Ontario Distinguished Researcher Award, Glinski Research Excellence Award, IEEE CS Golden Core Award, IEEE Canada Gotlieb Medal Award, IEEE ComSoc Expectional Leadership Award, IEEE TCPP Excpetional Leadership Award. He is a cofounder of the QShine International Conference on Quality of Service for Wireless/Wired Heterogeneous Networks (QShine 2004). He served as the general chair for the Eighth ACM/IEEE Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, and the Ninth ACM/IEEE Symposium on Distributed Simulation and Real-Time Application (DSRT), the program chair for the ACM Workshop on QoS and Security for Wireless and Mobile Networks, ACM/IFIPS Europar 2002 Conference, IEEE/SCS Annual Simulation Symposium (ANNS 2002), ACM WWW 2002, IEEE MWCN 2002, IEEE/ACM MASCOTS 2002, IEEE Wireless Local Networks WLN 0304; IEEE WMAN 0405, and ACM MSWiM 9899, and a TPC member of numerous IEEE and ACM sponsored conferences. He served as the vice general chair for the Third IEEE Distributed Computing for Sensor Networks (DCOSS) Conference in 2007, as the program cochair for GLOBECOM 20072008 Symposium on Wireless Ad Hoc and Sensor Networks, and for the 14th IEEE ISCC 2009 Symposium on Computer and Communication Symposium, and as the finance chair for ACM Multimedia 2008. He also serves as a Steering Committee chair for the ACM Modeling, Analysis and Simulation for Wireless and Mobile Systems Conference, the ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, and IEEE/ACM DSRT.