Towards the Design of an Interoperable Multi-Cloud Distributed Simulation System

Dan Liu  
EECS - University of Ottawa  
Ottawa, ON, Canada  
dliu049@uottawa.ca

Robson E. De Grande  
EECS - University of Ottawa  
Ottawa, ON, Canada  
rdgrande@site.uottawa.ca

Azzedine Boukerche  
EECS - University of Ottawa  
Ottawa, ON, Canada  
boukerch@site.uottawa.ca

ABSTRACT
Simulations over Cloud environments introduce additional benefits from Cloud Computing to conventional distributed simulations, including elasticity on computation resource, cost saving on investment, and convenience of service accessibility. There exist some works that attempt to apply Cloud computing on distributed simulation. However, there is one significant drawback on those works: lack of interoperability across Cloud platforms, which limits the usability and flexibility of distributed simulation over Cloud environment substantially. Thus, we propose a novel interoperable multi-Cloud distributed simulation system. This system is based on existing approaches in deploying distributed simulation systems over the Cloud environment. Our proposed system integrates Cloud computing to conventional distributed simulations, addressing the interoperability issues of distributed simulation on Cloud environments and enhancing the capability of traditional HLA-based simulations.

Keywords: Distributed Simulations, Cloud computing, Interoperability.

1 INTRODUCTION
As a new approach to accessing computation resource, Cloud computing provides a novel and potential computation paradigm, which has been extending and improving the standard computation models substantially. Through Cloud Computing, users can access a centralized and elastic computation resource pool conveniently via the network. These resources are available in the format of services and under an on-demand model. This new resource provisioning method has been introducing new evolutions in the conventional information technology industry.

The advantages of Cloud computing consist of one important factor that drives research and development works to apply Cloud Computing to conventional distributed simulations. On the other hand, High Level
Architecture (HLA) introduces several factors that benefit the development and execution of distributed simulations through interoperability and reusability (Simulation Interoperability Standards Committee 2000), which can be further improved through Cloud computing paradigm. The flexibility in elastic resource supply considerably benefits the traditional distributed simulations.

In fact, efforts have been made in this sense, showing potential opportunities in this area and significant aggregation of value to simulation systems. However, there are various issues and problems in such existing works. Among them, interoperability is a major issue that limits the development of distributed simulations, as well as any distributed application, on multi-Cloud environments (Diallo et al. 2017). The interoperability issue concerns the execution of distributed simulations across multiple different Cloud platforms. This multi-Cloud execution involves the deployment distributed simulations on a platform consisting of several underlying Cloud Computing services, in which there might be private Cloud or public Cloud providers, such as Amazon EC2. All the existing solutions or schemes are specified to dedicated Cloud systems, which possibly limits the utilization and integration of computation resources from different Cloud platforms.

Based on the requirements in the context of distributed simulations and the restricting characteristics of current Cloud Computing platforms, we propose an interoperable HLA-based simulation system over a Cloud environment, named ISSC. The architecture of the proposed simulation system aims to enable execution of simulations based on large-scale HLA-based distributed simulations across multiple Cloud platforms. The framework defines a standard interfacing among Cloud environments, which allows establishing a single common virtual environment where all resources and services are accessible and fully responsive.

The remainder of this paper is organized as follows. Section 2 describes the related works, including research on simulation on Cloud computing and interoperability on Cloud Computing. Section 3 details proposed simulation system. Section 4 introduces the experiments and discusses the related results. Finally, Section 5 summarizes the paper and presents directions for future work.

2 RELATED WORKS

Efforts from academia and industry have been attempting to run conventional simulations using Cloud Computing paradigm. A few schemes and solutions have been proposed and implemented, providing valuable examples and experience of exploring this new area.

2.1 Distributed Simulations on Cloud Environments

Indeed, there exist research works that have been already investigating the challenges of establishing distributed simulations, or at least porting the existing ones, on Cloud environments (Hüning et al. 2016, Padilla et al. 2014, Fortmann-Roe 2014). These efforts can be considered initial attempts towards achieving interoperability in Cloud-based distributed simulations.

2.1.1 COSIM-CSP

The work described in (Li et al. 2009) defined a Cloud simulation platform prototype, named COMSIM-CSP. The prototype is developed from conventional Grid simulation, applying Cloud computing on the traditional distributed simulation and integrating a set of related technology and methods. The employed technologies include virtualization, pervasive computing, and high-performance computing (Li et al. 2009) (Li et al. 2012). The motivation of this work consists of resolving issues of conventional distributed simulations based on Grid computing to improve and strengthen existing Grid simulation systems (Li et al. 2009).
The work introduces an architecture of the Cloud simulation platform, which constitutes of a resource/capability layer, Cloud simulation platform layer, and service application layer. The resource/capability layer is in charge of integrating hardware and software resources for upper layers. The Cloud simulation platform layer enables the control of simulation execution. Based on the Cloud simulation platform layer, the service application layer entirely fulfills multi-user support, which implements four different usage modes (Li et al. 2012).

The implemented prototype employs (Li et al. 2011) a series of technologies and mechanisms to implement the platform prototype and to support the various functionalities of the architecture (Li et al. 2009) (Li et al. 2012). The features include technologies that extend over a pervasive portal and simulation resource service scheduling (Li et al. 2012). Based on the implemented Cloud simulation platform prototype, a group of simulation applications has been developed, which shows acceptable functionality and performance from the perspective of simulators.

2.1.2 Cloud-based Distributed Simulation System

To improve deficiencies of HLA and enhance its capabilities, efforts have been applied to integrate Cloud computing with Grid-based distributed simulations (He et al. 2012). Consequently, this Cloud-based distributed simulation system, namely CDS, has been designed to satisfy the new requirement of large-scale distributed simulation effectively (Li et al. 2011). The proposed simulation system introduces the Cloud computing to provide a service-oriented simulation platform, addressing some vital issues of HLA and former grid-based distributed simulation. The improvement brought by CDS contains enhanced load balancing capability of the simulation, realizing sharing and reuse of simulation resources, efficient applicability on wide area network, on-demand service model of simulation resources, and reliable security guarantee (He et al. 2012).

CDS is consist of six primary layers, including simulation application layer, simulation Cloud portal/tool player, Core service layer, Cloud infrastructure layer, and Heterogeneous resource layer (He et al. 2012). Heterogeneous resource layer plays a fundamental role in the whole architecture since it encapsulates and integrates hardware and software resources, including computation, storage, and network resources to provide underlying physical resources to upper layers. Core service layer provides a series of essential functionalities by a group of components (He et al. 2012). Simulation model layer contains a repository of simulation models in the Cloud simulation system, which are employed by other components to build application-oriented simulations. Simulation Cloud portal/tool layer plays the role of a portal of the Cloud simulation system, providing an interface to simulators. Simulation application layer encloses various simulation applications built on the Cloud simulation platform.

2.2 Interoperability on Cloud Computing

The interoperability on Cloud computing is considered critical for the further development of the Cloud ecosystem, including services and applications (Pahl et al. 2013). The work on interoperability on Cloud computing involves some standards on interoperability, as well as middleware and software libraries targeting interoperability.

2.2.1 Standards on Interoperability on Cloud Computing

The fast growth of Cloud computing raises the requirement of standards on interoperability on Cloud computing. There have been several related standards proposed by different organiza-
tions (Machado et al. 2009). Among these standards, OCCI stands as a leading open source Cloud computing interface/standard, playing an important role in the ecosystem of Cloud computing.

Cloud Infrastructure Management Interface, CIMI, consists of another leading open standard on interoperability on Cloud computing, is proposed by Distributed Management Task Force (DMTF) (Mell and Grance 2010, Armbrust et al. 2010). CIMI is designed to provide a series of the standardized definitions of resources on different Cloud platforms and related operations on resources. The standard also covers interactions over Clouds platforms so that Cloud consumers can access and manage Cloud resources from various Cloud platforms.

The core resources of CIMI model include systems, machines, volumes, and networks. Systems in the CIMI model define groups of resources which can be managed as one single unit (Armbrust et al. 2010). Machines in the CIMI model represent compute instances which encapsulate a group of computation resources, such as CPU and memory. A volume in the CIMI model is a storage container, which can be associated with machines. A network in CIMI model implements a connected network involved with a group of machines.

### 2.2.2 Middleware/Libraries on Interoperability on Cloud Computing

There are also many middleware/libraries on interoperability on Cloud computing developed by open source organizations or commercial companies that complement the aforementioned standards.

a) LibCloud. LibCloud is an open source programming library written in Python which aims to provide a unified API interacting with different Cloud service providers (Rajaprabha 2013). The purpose of the library is to facilitate the development of application working over various Cloud platforms.

b) Fog. Fog is another open source programming library written in Ruby, similar as LibCloud, which is developed to utilize multiple Cloud provider services from a standardized API (Petcu and Vasilakos 2014).

c) JClouds. JClouds is an open source multi-Cloud toolkit that enable interoperability over several Clouds (Petcu et al. 2011). It also allows developers to use Cloud-specific features with full control.

d) DeltaCloud. DeltaCloud is a Ruby API originally proposed and developed by Red Hat and the Apache Software Foundation (Bist et al. 2013, Brogi et al. 2014, Monteiro et al. 2011). It provides a unified abstract REST-based API for the management of services deployed across Cloud platforms.

### 2.3 Summary

From the previous related discussion, we know there is much effort of planting existing distributed simulation on Cloud computing. However, one significant drawback of those conventional distributed simulation platforms/systems is the lack of interoperability across different Cloud platforms. They are limited on specific Clouds platform which means it is impossible to move across different Cloud platforms. This reduces the capability of encapsulation and integration of resources from various Cloud platform greatly, which degrades the portability and compatibility of those platform/system.

In that sense, OCCI and CIMI are two similar standards focusing on the interoperability of Cloud computing that differ in some aspects. First, OCCI is designed to an extensible standard while CIMI is specific to just IaaS domain. Second, OCCI provides a group of abstractly defined models in an inheritable architecture while CIMI provides a group of completely defined and independent models. Thus, due to the better flexibility OCCI provides, this work employs it to implement the interoperability of distributed simulation system on Cloud environment.
3 INTEROPERABLE MULTI-CLOUD DISTRIBUTED SIMULATION SYSTEM (ISSC)

In this paper, we propose a simulation system that targets the interoperability in multi-Cloud environments named ISSC. The system aims at to integrate Cloud computing with traditional distributed simulations, providing an easy-of-use, unified interface. Cloud computing introduces versatility through elastic and portable resource provision and management, offering theoretically infinite computation capability at lower costs. The proposed system features unique interoperability over Cloud platforms, which means it can be deployed on various Cloud platforms and perform all kinds of interactions across these Cloud platforms.

3.1 Overview

With prepared simulation environments and a convenient user interface, ISSC provides a one-stop portal for simulators. ISSC includes benefits inherited from the Cloud computing paradigm, such as cost reduction, productivity, reliability, scalability, and elasticity. However, multi-Cloud application deployment suffers from interoperability challenges. Consequently, ISSC employs OCCI to overcome the lack of interoperability over Cloud platforms, offer the ability to integrate resources from various Cloud. This cross Cloud integration allows for alllying public Cloud platforms, such as Amazon EC2, to private Clouds, such as built by OpenNebula and OpenStack standards. This feature enables scalability, expandability, and flexibility of Cloud-based simulations significantly.

At present, ISSC supports HLA distributed simulations, containing specifically designed virtual machines and enclosing custom computing resources and simulation images. The simulator can deploy various complex HLA simulations with the prepared simulation environment and simulation portal.

3.2 Architecture

As demonstrated in Figure 1a, ISSC comprises four parts: simulation portal, frontend, backend and Cloud service. Simulation portal is the simulation entry and interface of ISSC, which provides a single entrance for simulators to deploy and perform simulations on this simulation system. The frontend is the back support to the simulation portal, which accepts and processes requests from simulation portal. The backend plays the role of a bridge between various Cloud service and the frontend, which encloses computation resources.
from Cloud platforms. Frontend and backend are constructed based on the rOCCI framework, a ruby OCCI framework, through which frontend and backend communicate with the OCCI language. Cloud service represents various underlying Cloud service provider, which can be any Cloud platform supported by OCCI.

This architecture adopts a distributed deployment model. Components are deployed on geographically and logically different servers, which increases the flexibility of the architecture significantly. Usually, the Simulation Portal and the Frontend part components are deployed together on the same server while a group of Backend components is deployed on a set of different servers. Underlying Cloud services are located at same servers with Backend servers or separately.

3.3 Simulation Portal and Frontend

The simulation portal and frontend are deployed together. Both of these components play the role of interacting with simulators and responding to them. It is also in charge of communicating with various backend which provides computation resource required by simulation tasks.

As demonstrated in Figure 2a, the Frontend is built on Apache HTTP Server, Fusion Passenger, Ruby on Rails, and rOCCI framework. Apache HTTP Server is employed to offer a web interface of the simulation platform to end-users. The web service is implemented via Ruby on Rails and is loaded through Phusion Passenger, a Ruby application deployment server which is encapsulated in Apache HTTP Server. With the underlying rOCCI framework, the Frontend interacts with the backend, speaking OCCI language.

3.4 Backend

The backend acts as the bridge between the frontend of the simulation system and underlying Cloud providers. The backend is responsible for processing requests from the frontend and responding to a particular client with property information. Moreover, the backend is in charge of transferring messages to and collecting related data from Cloud platforms. Through the backend, ISSC delivers tasks to a Cloud platform, monitors Cloud platforms, and manages them via proprietary APIs of each platform.

As shown in Figure 2b, the backend adopts a distributed architecture. Apart from the frontend, backend servers are deployed separately in a one-to-one model, which means that one dedicated backend server serves one particular background Cloud platform, not being able to bind to more than one Cloud platform. Any two backend servers cannot also be deployed in one physical server together. However, the backend server does not necessarily need to be deployed in the physical server within the respective Cloud platform;
it can be used in a separate server physically apart from the Cloud platform. Finally, all the distributed backend servers construct an abstract backend layer over the entire multi-Cloud environment.

3.5 Cloud Service

The Cloud service layer corresponds to the underlying Cloud platforms providing computation resources of ISSC. Through OCCI, various Cloud platforms can be integrated into ISSC. Moreover, the current implementation of ISSC, at present, supports two Cloud service platforms: OpenNebula and Amazon EC2.

4 EXPERIMENTAL RESULTS AND ANALYSIS

A series of experiments were performed on both native Cloud platform and ISSC-enabled environment. The native Cloud platform consisted of a private Cloud built though OpenNebula while the ISSC-enabled environment comprised a mix of the OpenNebula private Cloud and Amazon EC2 public Cloud. The experiments were conducted to observe performance, which involved time overhead of ISSC on top of the native Cloud platform, the execution time of simulation task, and execution time of simulation application. The experimental scenarios, as well as the configuration parameters, have been defined according to the experimental setup described in (Diallo et al. 2016) to evaluate the performance of the proposed system.

4.1 Experimental Environment

As shown in Figure 1b, the prototype is built on a hybrid system, enclosing one frontend server and two backend servers. The frontend server and two backend servers are deployed on virtual machines (VMs), one per each server, created by Parallels Desktop 10 on a Mac Mini Server with Mac OS X 10.10.3 Operating System. Frontend server and backend servers run Ubuntu 14.04 LTS Server Operation System. One backend server, OpenNebula Backend, is connected to a Private Cloud built on three Dell Studio computer servers using OpenNebula 4.10.2. Another backend server, Amazon Backend, is connected to Amazon AWS EC2 public Cloud service.

4.2 Time Overhead of ISSC

The first category of experiments was designed to observe the time overhead of ISSC compared to the native Cloud platform. It refers to the extra time cost which ISSC generates. A java application was designed and implemented to transfer raw data between two VMs, which was employed to test the data communication performance between VMs. Two VMs are deployed on OpenNebula while another two VMs with the same specification are deployed on ISSC. Same experiments are carried out on both two groups of VMs respectively.

The Java test application transferred data between two nodes according to many parameter sets. Three parameters were used in the experiments: message_size, message_frequency, and message_time. Three groups of experiments were performed. In the first group of experiments, message frequency varied from 10 messages per second (msg/s) to 200 msg/s, with fixed message_size (1000*1000 Bytes) and message_time (10s). Message frequency kept varying from 10 msg/s to 200 msg/s, with the same message time (10s) but different message sizes (100*1000 Bytes and 10*1000 Bytes) respectively in the second and third group of experiments. The results contained averages obtained 10 executions of each parameter combination.

As depicted in Figure 3, the time cost of data transfer increases steadily when message frequency grows from 10 msg/s to 200 msg/s with different message sizes. The time overheads of ISSC on top of the native
Cloud platform differ according to message frequency and message size. If message size is set to 10000 bytes, there is no time overhead between the native Cloud platform and ISSC. If message size is set to 100000 bytes, the time overhead can be ignored when message frequency is less than 90 msg/s while the time overhead grows slowly when message frequency exceeds 90 msg/s. If message size is set to 1000000 bytes, the time overhead varies between 8.3% and 25.1%. As a result, the results show the overhead grows linearly with the scale of the system.

4.3 Execution Time of Simulation Tasks

The second category of experiments observed the execution time of distributed simulation tasks on the native Cloud platform and ISSC. In this category of the experiment, Portico RTI 2.0 was employed to design and perform HLA-based distributed simulations. Computation intensive simulation tasks and communication intensive tasks were performed in this execution time evaluation.

In these experiments, 10 VMs were deployed on ISSC hybrid environment and the native Cloud platform to carry out simulation tasks. The hybrid environment consisted of five VMs on OpenNebula backend and five VMs on Amazon backend. Same simulation tasks were also performed on 5 VMs deployed on the native Cloud platform. This setup observed the scenario where computation resources are limited in the native Cloud platform where only limited VMs are available. By contrast, ISSC enabled access to other resource pools through the interoperability feature.

4.3.1 Computation Intensive Analysis

This first set of experiments examined execution times of a group of computation intensive simulation tasks. Thus, ten federates were created to performed controlled highly complex floating point calculations. A parameter is used to determine the scale of computation load, which ranged from $10 \times 10^6$ to $100 \times 10^6$ calculations (iterations).

As shown in Figure 4a, the execution times of tasks increase with fixed rate when the number of iterations grows. Due to the difference of capabilities among VMs on different backends, the execution time on ISSC is slightly less than the native Cloud platform, about 7%, when 10 VMs are employed to perform simulation tasks. If the number of VMs is limited to 5, the execution times on the native Cloud platform reach 200% of ISSC approximately. This result illustrates the great benefit of accessing several Cloud platforms through the interoperability introduced by ISSC.
4.3.2 Communication Intensive Analysis

The second set of experiments examine execution times of a group of communication intensive simulation tasks. Ten federates are created to carry out simulation tasks where the number of simulation steps and message_size are used to control volume of data that is transmitted over an entire simulation run. The total execution time of a simulation task primarily depends on the value of these two parameters. In some groups of experiments, steps vary from 10 to 100 with different fixed message_size, which is delimited by 10^6 bytes, 10^5 bytes, and 10^4 bytes.

As described in Figure 4b, the performance in the native Cloud platform is better than ISSC when executing communication intensive simulation tasks, showing some variations depending on message size. The performance gap is below 10% at most of the time when message size is set to 10^4 bytes while it can reach 30% in some points when message size is set to 10^5 bytes. The gaps originate from the latency of network and the communication overhead among various backends. However, the situation is different when the number of VMs is limited to 5 on the native Cloud platform. The results show that the native Cloud platform is around 10% better than ISSC at most of the time but is worse than ISSC from 14.1% to 23.3% when message size is set to 10^4 bytes.

4.3.3 Effect of Virtual Machines Deployment on Execution Time of Simulation Task

Another group of experiments is conducted to investigate the effect of different VMs deployment on the backends. Thus, ISSC environment consists of two backends: OpenNebula backend and Amazon AWS backend. Ten VMs are built on the ISSC prototype in this group of experiments. In a group of scenarios, the deployment of VMs on backends varies from 10 VMs on OpenNebula backend and 0 VMs on Amazon backend to 0 VMs and 10 VMs on Amazon VMs. The distributed simulations include computation intensive tasks and communication intensive tasks. Two groups of computation intensive tasks are performed: 10 \times 10^6 and 100 \times 10^6 iterations. Three groups of communication intensive tasks are realized: 100 steps with 10^6 bytes, 10^5 bytes, and 10^4 bytes message sizes.

As depicted in Figure 5a, the execution time of computation intensive tasks decrease when the number of VMs deployed on OpenNebula drops. The ISSC performance is best when there are 10 VMs in Amazon and 0 VMs in OpenNebula, 11% and 12% better with two different numbers of iterations respectively. The reason is the contrasting processing power of VMs in different backends. In terms of communication intensive tasks, the execution time is most when there are 5 VMs in OpenNebula and 5 VMs in Amazon. The
performance is best when there are 10 VMs in Amazon. This behavior is attributed to the communication delay among VMs deployed on different Cloud platforms.

4.4 Performance of Simulation Application

The last category of experiments observes the performance of executing an HLA-based distributed simulation application under different scenarios in ISSC. Sushi Restaurant is employed in these experiments to conduct the performance evaluation. The simulation application is implemented based on Pitch pRTI (Nan and Eusgeld 2011, Tang et al. 2010). The pRTI middleware suite, including the central RTI component (CRC) and the local RTI components (LRC), are installed on each VMs. Each role of the sushi restaurant corresponds to one federate in the simulation application. One central RTI component and 5 federates are deployed on those 6 VMs, which means one dedicated VM for the central RTI component and one VM for each federate. Same experiments are performed on ISSC with different ratios of VMs deployed on two backend servers. In a group of scenarios, the deployment of VMs on backends varies from 10 VMs on OpenNebula backend and 0 VMs on Amazon backend to 0 VMs and 10 VMs on Amazon VMs. According to the characteristics of this simulation application. The results correspond to an average of 10 execution per each parameter setup, with a varying number of incoming serving customers, from 10 to 100.

As described in Figure 5b, the time costs of simulation increase along with the growth in the number of servings. The execution time also displays a curve when the deployments of VMs change. When the number of serving customer rises to 100, the curve displays an apparent radian value with a peak at the middle of the curve and two radians at the beginning and end of the curve. The time cost of simulation is the highest when the number of VMs deployed on OpenNebula backend server is closest to the number of VMs deployed Amazon backend server. To be more specific, the value is about 472000ms when 3 VMs are deployed on OpenNebula backend server and 3 VMs are deployed on Amazon backend server, which is about 2% more than the value when all 6 VMs are deployed on OpenNebula backend server, and 4.4% more than the value when all 6 VMs are deployed on Amazon backend server. This behavior originates from the communication delay among distant VMs and the application-specific simulation federate dependencies.

5 CONCLUSION

In this paper, we have proposed a novel interoperable HLA-based simulation system in a Cloud environment named ISSC. The system applies Cloud computing on conventional simulations, bringing pseudo-infinite
computation capabilities, cost saving on investment, portability and flexibility on deployment, and accessibility and elasticity on services. ISSC allowed a feasible and practical solution for the deployment and execution of HLA-based simulations over various Cloud services and the interoperability across different Cloud platforms. A prototype of ISSC has been implemented as a proof of concept to allow a performance analysis of ISSC. Experiments have been conducted and showed that there are benefits in reliably and efficiently enabling multi-Cloud environments for distributed simulations.

As future work, we intend to extend the supported Cloud platforms, expanding the scenarios where ISSC is applied. Also, extra features will be incorporated, such as network approaches for extending adaptability.

REFERENCES


**AUTHOR BIOGRAPHIES**

**DAN LIU** is currently a M.Sc. at the University of Ottawa, Canada. He received his B.Sc. of Engineering in Computer Science and Technology from Lanzhou University in 2007. His research interests include distributed simulations and cloud computing. His email address is dliu049@uottawa.ca.

**ROBSON E. DE GRANDE** is a Senior Research Associate at the University of Ottawa, Canada. He received his Ph.D. degree at the University of Ottawa in 2012. His research interests include distributed simulation systems, Cloud computing, high performance computing, and computer networks. His email address is rdgrande@site.uottawa.ca.

**AZZEDINE BOUKERCHE** is a distinguished professor and holds a Senior Canada Research Chair Tier 1 position at the University of Ottawa. He was a recipient of the IEEE Canada K. Gotlieb Computer Gold Medal Award, IEEE-CS Golden Core Award, and Ontario Distinguished Research Award. His current research interests include context-interpretation-based systems for emergency class applications, sensor networks, vehicular networks, wireless multimedia, distributed and mobile computing, and mobile cloud networking. boukerch@site.uottawa.ca.