Design and Analysis of an Interoperable HLA-based Simulation System over a Cloud Environment

by

Dan Liu

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements

For the M.A.Sc. degree in

Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Abstract

Distributed simulation over Cloud environment is still a new subject. Cloud computing is expected to bring new benefits to conventional distributed simulation, including elasticity on computation resource, cost saving on investment and convenience on service accessibility. A few researches have been done on applying Cloud computing on distributed simulation. However, there are various drawbacks and limitations on those works. Lack of interoperability across Cloud platforms is one of critical drawbacks among them. It can greatly limit the usability and flexibility of distributed simulation over Cloud environment. Based on the investigation on Cloud computing and existing distributed simulation systems over Cloud environment, a novel interoperable HLA-based (High Level Architecture) simulation system over a Cloud environment, ISSC (Interoperable Simulation System over a Cloud Environment), is proposed in this thesis. ISSC aims to address the interoperability issue of simulation system across various Cloud platforms. It employs OCCI and a set of technologies, including Ruby on Rails, OpenVPN and RESTful web services, to build the interoperability across Cloud platforms. It adopts a distributed architecture to construct flexibility and expansibility of the system. The prototype and related experiments performed provides an excellent demonstration that ISSC is a reliable and effective solution on interoperable simulation system over a diverse Cloud environment.

Acknowledgments

Firstly, I would like to present my grateful appreciation to my supervisor, Professor Azzedine Boukerche of the School of Electrical Engineering and Computer Science (EECS) at The University of Ottawa, for giving me the chance to pursue my graduate study under his supervision. I am also very grateful for his patience and guidance on my thesis. Without his support, it is impossible for me the complete this thesis.

I also would like to express my deep gratitude to Dr. Robson Eduardo De Grande who always provides me selfless support. His endless help on my research and thesis gave me great assistance. It will become a valuable experience in my life.

I thank my family, my friends, my classmates during my graduate study and all colleagues in the PARADISE research laboratory. Thank you so much for your help.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The emergence of Cloud computing provides a novel and prospect computation model. It constructs an new approach by which clients are able to access a centralized, elastic and theoretically infinite computing resources pool conveniently through network, as a service, in an on-demand model. This novel computation model is considered as an excellent computation solution to future information technology industry and expected to produce a great revolution in the conventional information technology industry. In fact, during past several years, Cloud computing grew speedily and has become a common and popular computation solution and even economic model of the information technology industry, which has been presenting the great potential to pose a huge change on the conventional information technology industry.

Considering the advantages of Cloud computing, we have been managing to apply Cloud computing to conventional distributed simulations. Cloud computing is expected to bring conventional distributed simulations a great deal of new benefits,

including theoretical infinite computation capability, cost saving on investment, portability and flexibility on deployment, simpleness and convenience on environment setup, and accessibility and elasticity on service. Through introducing Cloud computing to the conventional distributed simulation, a lots of issues and bottlenecks existing in conventional distributed simulation are prospected to be improved or even resolved. It has been agreed with by a great deal of researchers and professionals that the rapid development of Cloud computing is going to boost the advance of conventional distributed simulation and related research.

In fact, some effort on applying Cloud computing to conventional distributed simulations has been made by researchers and professionals. A few schemes or solutions have been proposed and some experiments or implementations have been completed to apply Cloud computing to conventional distributed simulation. Some other researchers have been trying to plant or port existing distributed simulation over Cloud environment. Those work can be considered early effort on applying Cloud computing on conventional distributed simulations. Although those effort has shown excellent examples on exploring in this area and provides valuable experience in this area, there are different kinds of defects and drawbacks in those work which can be resolved or improved. A fact is that simulation on Cloud computing is still at an early stage. It is apparent that the research on distributed simulation on Cloud computing is still far from mature and much more effort should be devoted into this area to provide a reliable and mature solution on simulation on Cloud computing.

1.1 Motivation

Due to the fast development of Cloud computing in last a couple of years, it turns to be feasible and prospect to apply Cloud computing to conventional simulations. It is expected to introduce advantages of Cloud computing into conventional simulations, such as elasticity, cost saving and flexibility, to produce a more functional, more flexible and more efficient solution for conventional simulations.

Some research on simulation on Cloud computing has been performed which shows excellent examples on exploring in this area and provides valuable experience and clues to researchers. However, those works have various defects and drawbacks, such as lacking flexibility on deployment, which needs more effort into this area to resolve or improve and lead to a reliable and mature solution on simulation on Cloud computing.

Moreover, with the prevalence of Cloud computing, simulation on Cloud computing faces new challenges from the development of various Cloud computing platforms, including public Cloud computing service and private Cloud computing service. The bloom of various Cloud computing services boosts the application of Cloud computing in all kinds of area, whereas it also brings the interoperability issue of Cloud computing platforms. There is no solution on interoperable simulation platform over various Cloud platforms, which has been limiting the usability and flexibility of simulation on Cloud computing apparently.

A reliable, mature and interoperable solution on simulation across Cloud platforms is imperative and absolutely is a key of promoting the application of Cloud computing on conventional simulations.

1.2 Objectives

Considering the advantages of Cloud computing and the current status of development of distributed simulation, I think it would be a valuable work to apply Cloud computing to conventional distributed simulation and provide a reliable, mature and interoperable solution to interoperable distributed simulation on a Cloud environment.

Based on the characteristics of Cloud computing and conventional distributed simulation system, I am going to propose a novel architecture of interoperable HLA-based simulation system on a Cloud environment, based on which I design, implement and analyze an interoperable HLA-based simulation system on the Cloud platforms. This novel simulation system applies Cloud computing to conventional HLA-based distributed simulations, encapsulating the advantages of Clouding computing, improving or resolving some existing issues and bottleneck of current HLA-based distributed simulation and enhancing the capability and interoperability of conventional HLA-based distribute simulation. Through the great benefits brought by Cloud computing, it provides clients an easy-of-use and uniform interface to elastic and portable simulation service, with theoretical infinite computation capability in an economic cost.

This simulation system aims to build the interoperability across various Cloud platforms to enable large-scale interoperable HLA-based distributed simulation on multi Cloud platforms, which means the simulation system owns the capability of deploying simulations over various Cloud services and performing interoperable simulations across different Cloud platforms.

1.3 Contributions

In this thesis, I propose a novel architecture of interoperable HLA-based simulation system, ISSC, on a Cloud environment which provides a feasible, reliable and flexible solution to interoperable HLA-based distributed simulation on a Cloud environment. This novel simulation system brings conventional HLA-based distributed simulations lots of benefits from Cloud computing, such as theoretical infinite computation capability, cost saving on investment, portability and flexibility on deployment, simpleness and convenience on environment setup, and accessibility and elasticity on service.

Based on the proposed architecture, I design and implement a prototype of this interoperable HLA-based distributed simulation system, which provides an excellent demonstration to prove the feasibility of the concept of ISSC. A group of experiments are also designed and performed on this prototype of ISSC to analyze and research the performance and characteristics of this interoperable HLA-based distributed simulation system.

In conclusion, this novel interoperable HLA-based distributed simulation system provides a feasible and effective solution to interoperable HLA-based distributed simulations over a diverse Cloud environment which addresses the concern of interoperability on distributed simulation over various Cloud platforms.

1.4 Outlines

This thesis is going to be organized as follows. In Chapter 1, a simple introduction of this thesis is given. In Chapter 2, some basic background knowledge and information is presented. In Chapter 3, related work is discussed, including research on simulation on Cloud computing and interoperability on Cloud computing. In Chapter 4, details of the design of simulation system are described. Then in Chapter 5, related experiment and results as well as analysis on this simulation system is illustrated. Finally, Chapter 6 summarizes the conclusion and presents some future work.

Chapter 2

Background

As discussed above, the objective of this thesis is to design and analyze an interoperable simulation system over a Cloud environment. In order to carry out the discussion very well, some basic background knowledge and information is presented in this chapter, including computer simulation, HLA-based distributed simulation, Cloud computing platforms and interoperability on Cloud computing.

2.1 Computer Simulation Systems

Computer simulation is a computer technique or methodology which represents real world systems, as well as related dynamic behaviors, using computer techniques, to simulate activities of systems in the real world and produce related results used for analysis. Computer simulation is used widely in research and industry helping to design, build and imitate systems in the real world, especially some complex ones which are difficult or expensive to produce or implement. In fact, along with the great

development of computer science and technology, computer simulation has been playing a very important role in our life, from weather forecasting, pilots training, to disaster prediction and military.

According to characteristics of various simulations, simulations can be divided into two categories, continuous simulation and discrete simulation. Continuous simulation imitates systems as well as their behaviors which occur over a period of continuous time while discrete simulation imitates systems as well as their behaviors which occur at a series of discrete time spots. Among them, discrete simulations are suitable for computer simulation which can imitate discrete events in digital systems.

2.2 Distributed Simulation

Distributed simulation is one kind of computer simulation which is deployed among a group of computers connected through a distributed network. It is common that distributed simulation is employed to imitate complex network systems for study and research.

Examining characteristics of distributed simulation, we can summarize some principal benefits which distributed simulation bring us [1]:

1. Reduction on execution time [2]. Basically, this benefit is achieved through decomposing one large simulation task into a set of small sub-simulations which are deployed and executed simultaneously on a group of distributed computers connected via network. In this way, the execution time of simulation can be reduced by times depending on the scale of simulation subdividing.

2. Distribution on geography [2]. The decomposition of simulation among a group of computation nodes which may be deployed at various locations builds a virtual network where nodes interact with each other regardless of the geographical barriers. It can bring us great benefits on time and economics.

3. Integration of various simulators executing on machines from different manufacturers [2]. Distributed simulation also brings us great benefits on compatibility over various simulators from different manufactures. It integrates different simulators into one virtual simulation network which can adapt various simulators.

4. Fault tolerance [2]. Thanks to the characteristic of distributing simulation task across a set of computation nodes, distributed simulation possesses better fault tolerance. Even if some computation nodes are out of work, other computation nodes are able to take over corresponding simulations tasks to instead failed nodes.

Briefly, there are two primary categories of distributed simulation which draw most of attention: analytic simulations and virtual environment.

Analytic simulation refers to examining quantitative behaviors and data of simulated systems so that the characteristics and mechanism of simulated systems could be analyzed quantitatively. Due to the features of analytic simulations, analytic simulations are usually expected to be executed in least time. In order to simulate investigated systems, analytic simulations are designed and implemented to approaching the real systems as close as possible. Human interaction in analytic simulations is limited or eliminated to ensure the accuracy of simulations. Virtual environment is a new kind of distributed simulation which aims to build a simulated virtual environment for humans so that people could be placed in this virtual world and

behave like in the real world simulated. Considering the purpose of virtual environment simulation, this kind of distributed simulation raise a high requirement on real-time. Virtual environment is usually employed to training and entertainment. A comparison of analytic simulation and virtual environment is shown in Tab. 2.1.

Items	Analytic Simulations	Virtual Environments
Execution pacing	Typically as-fast-as-possible	Real-time
Typical objective	Quantitative analysis of complex systems	Create a realistic and/or entertaining representation of an environment
Human interaction	If included, human is an external observer to the model	Humans integral to controlling the behavior of entities within the model
Before-and-after relationships	Attempt to precisely reproduce before-and- after relationships	Need only reproduce before-and-after relationships to the extent that humans or physical components embedded in the environment can perceive them

Tab 2.1: Comparison of Analytic Simulation and Virtual Environment [1]

Briefly, two different kinds of architectures are employed in distributed simulation, client-server architecture and peer-to-peer architecture [3].

Client-server architecture place most of computation on the center computer server or centralized computation nodes which are in charge of management and monitoring of entire simulation system, while clients connect to the server to participate the simulation progress and obtain the result of simulation from the server. Thanks to centralized computation located on sever nodes, client-server architecture brings some benefits on the management of simulation system as well as convenience on maintenance and deployment. Meanwhile this model possesses a related high requirement for the computation capacity of server nodes which can become barriers for usability in specific situations. Multi-player gaming is a common application of client-server architecture.

By contrast, peer-to-peer architecture employs a different model where computation is placed across a virtual network composed of dispersively located computation nodes. This model provides a flat and wide computation architecture. In this architecture, computation load is distributed among connected peers which play equivalent roles and rely on each other to proceed the simulation. Compared to client-to-client architecture, peer-to-peer architecture possesses related low requirement for computation nodes and own related high fault tolerance, while it increases the difficulty on simulation deployment and also brings inconvenience on management and maintenance.

However, now there are still lots of challenges which the distributed simulation is facing [110]. Firstly, it remains a big challenge to model human and organizational behavior. More effort is required to address issues on modeling human and organizational behavior [11][12][13][14][15]. Secondly, the interoperability of distributed simulation is still a big issue in this area. It is expected to improve this problem by establishing a interoperable framework for plug-and-play. Thirdly, there are lots of work need to be done to improve subscription and distributed data management algorithms. And it is also a significant challenge to guarantee real time processing rates for distributed simulation. Besides, there are also some other challenges which we need to pay much attention to for distributed simulation, including maintaining consistency in multi-resolution models and component replications, guaranteeing causality and causal message delivery and ensuring repeatability and reversibility. All of those challenges requires more effort and exploration in distributed simulation.

2.3 HLA-based Distributed Simulation

Based on characteristics of distributed simulation, a couple of standards have been proposed or developed to support application of distributed simulation in various field, for example, High Level Architecture (HLA), Distributed Interactive Simulation (DIS), Test and Training Enabling Architecture (TENA), Aggregate Level Simulation Protocol (ALSP) and so on.

Among various standards supporting distributed simulation, High Level Architecture is popular one which is widely used in military and civilian applications [26][27][28][29][30]. Basically, the High Level Architecture is a general standard or framework designed for distributed simulation focusing on the interoperability and reusability of the simulation system or components, which was developed in the middle 1990's for the United States Department of Defense (DoD) [3]. One of main purposes of High Level Architecture is to build a unique and interoperable simulation environment or architecture where various simulators or simulation components can interact with each other regardless of different platforms or domains. Thanks to the interoperability and reusability brought by the High Level Architecture, existing simulators or simulation platforms can be reused and integrated into the new simulation system, which is considered to bring great saving on cost and time. Originally, the High Level Architecture is developed in the military domain, however, benefited from the well-design architecture, it presents the applicability in a wide area, from civilian and education to entertainment and industry. It is also proved to have excellent expansibility and compatibility to adapt to new platforms and systems [16][17][18][19][20].

Basically, the High Level Architecture is formally defined through several components which constitute the whole High Level Architecture together and rely on each other [41][42][43][44][45], including a set of HLA rules, Object Model Template (OMT) and Interface Specification.

HLA rules define set of constraints which regulate the behaviors and interactions of simulation elements in the system, providing a definition of properties and responsibilities of the simulation federation and related federates [5]. Object Model Template is a formalization language and mechanism used to define and represent data collection in the simulation system. It also provides a set of definition of fundamental models of the entire simulation architecture, including Federation Object Model (FOM), Simulation Object Model (SOM) and Management Object Model (MOM) [5]. Interface specification specifies the interface of the HLA run-time infrastructure (RTI), giving definitions of a set of service provided by the HLA RTI, which builds the fundament of communications among simulation elements in the simulation system [5].

In order to understand the underlying construction and principle of the High Level Architecture further, some basic conception or object are described in following paragraphs [8], including Federation, Federate, RTI, Object, Attribute, Interaction and Parameter.

Federation is a declared simulation union composed of simulation units (Federate) connected in a simulation network through an HLA RTI, which gives a basic define of the entire simulation going to be executed [8][51][52][53][54][55]. Federate represents the basic simulation unit(maybe a simulation application, a simulation computer or a compatible simulation platform) in a simulation federation, holding all

object representations, which participate in and perform specific simulation task in the simulation federation [8]. Run-time Infrastructure (RTI) is the underlying software infrastructure or middleware in the HLA framework, which enables communications and interactions among federates. RTI implements the standard interfaces of HLA framework, providing standard services which are consistent with HLA rules and by which federates participate in the simulations [8]. Object is a group of data collection transferred among simulations. Attribute represents the property of object. Interaction defines interactive operation occurring in simulation units. Parameter defines property of interaction [8].

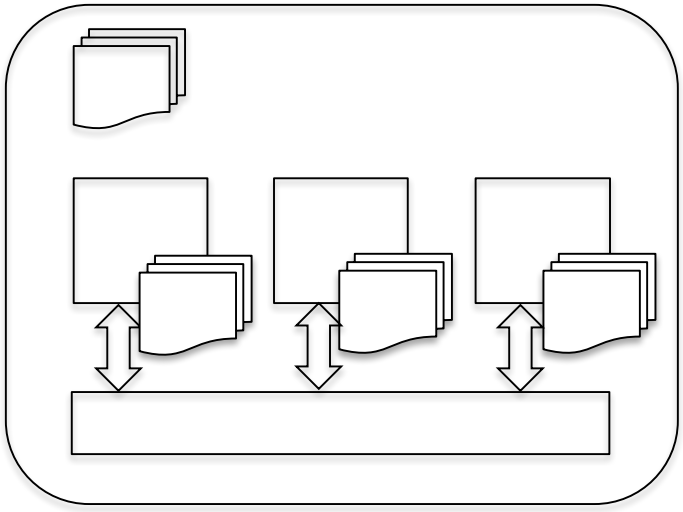

Fig 2.1: Federation and federates [9]

2.3.1 HLA Rules

Among three components of the HLA, the HLA rules are considered a fundamental end essential component of the whole architecture or framework which provides regulation and constraint of the running system. The HLA rules specify two groups of rules which define two set of regulations the federation and federates must obey [3]. The two groups

of rules regulate the behavior of the federation and federates as well as related communication and interaction among them.

The set of rules specified for the federation provide the norms of a federation behaving. There are five rules for federation. The first one is Federations shall have a Federation Object Model(FOM) in the Object Model Template(OMT) format [9]. The second one is all of objects' representation in the FOM shall be in the federates, not in the RTI [9][61][62][63][64][65]. The third one is during a federation execution, all exchange of FOM data among joined federates shall occur via the RTI [9]. The forth one is during a federation execution, joined federates shall interact with the RTI in accordance with the HLA Interface Specification [9]. The fifth one is during a federation execution, an instance attribute shall be owned by only one joined federate at any given time [9].

Accordingly, five rules are defined for federates. The first one is Federates shall have a Simulation Object Model(SOM) in the HLA Object Model Template [9]. The second one is Federates shall be able to update and/or reflect any instance attributes of objects in their SOM, and send and/or receive SOM interactions externally, as specified in their SOM [9][71][72][73][74][75]. The third one is Federates shall be able to transfer and/or accept ownership of attributes dynamically during a federation execution, as specified in their SOM [9]. The forth one is Federates shall be able to vary the conditions under which they provide updates of attributes of objects, as specified in their SOM [9]. The last one is Federates shall be able to manage local time in a way which will allow them to coordinate data exchange with other members of a federation [9].

2.3.2 Object Model Template

Object Model Template (OMT) provides the HLA a formalization language or mechanism used to define and represent data collection transferred in the simulation system. OMT gives descriptions of basic elements transferred among federates, enabling the interoperability of HLA. The information declared in OMT includes defined object models representing simulation items and architecture, as well as related attributes, associations, and interactions. It also declares definition of essential models and related key algorithms.

Generally, OMT provides a set of definition of three fundamental models of the entire simulation architecture. One Federation Object Model (FOM) corresponds to each federation [81][82][83][84][85]. It provides a description of all relevant shared data in the federation, for example, simulation objects, properties and related interaction. Similarly, one Simulation Object Model (SOM) corresponds to each federate. It provides a description of all relevant federate-specific data of the federate, including attributes and related interactions. Management Object Model (MOM) provides a description of relevant objects and interactions for the management of federation.

Normally, some mandatory basic information is required in OMT [10], including Object Class, Attribute Table, Interaction Class Structure Table, Parameter Table and FOM/SOM Lexicon. Object Class Structure Table defines the object class construction. Attribute Table defines properties of objects. Interaction Class Structure Table defines the interaction construction [96][97][98][99]. Parameter Table defines parameters of interactions. FOM/SOM Lexicon defines items in other tables.

Additionally, some other extended information may be included in OMT [10], for example Component Structure Table, Associations Table and Object Model Metadata.

2.3.3 Interface Specification

The interface specification specifies the interface of the HLA run-time infrastructure (RTI), giving definitions of a set of service provided by the HLA RTI, which builds the fundament of communications among simulation elements in the simulation system.

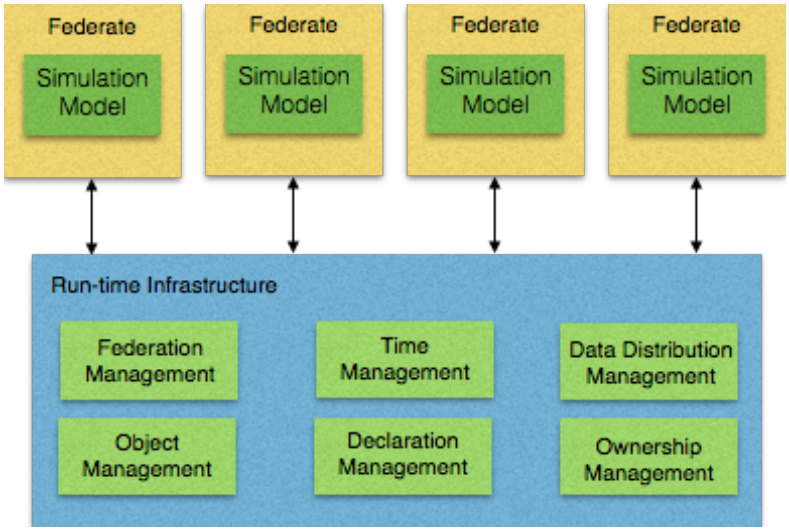

Fig 2.2: RTI and Interface Specification of HLA

Generally, six different aspects of simulation management services are defined in the interface specification [10], including Federation Management, Declaration Management, Object Management, Ownership Management, Time Management and Data Distribution Management. Federation Management provides fundamental functionality of building and manipulating a federation. Declaration Management implements basic mechanism or approach of information exchange among federates. Object Management provides fundamental functionality of building and

management of object. Ownership Management is responsible to ownership management of items in the simulation system during the simulation execution. Time Management provides synchronization mechanism of the simulation execution. Data Distribution Management Service implements data routing functionality among federates during the simulation execution.

2.3.4 Challenges and Limitations

High Level Architecture is for interaction between different types of systems such as real-time simulations and event-stepped wargames [112]. It provides a solution for distributed simulation focusing on the reusability and interoperability of simulation components and systems. However, there are some challenges and limitations blocking the development of High Level Architecture. Firstly, the flexibility of High Level Architecture could be a weakness of this architecture. Even though all of components of the simulation are HLA-compliant, they are able to interoperable with each other unless all federates agree on one same Federate Object Model (FOM) [112]. Apparently, this weakness reduces the interoperability of High Level Architecture greatly. Secondly, the complexity of the architecture and implementation also bring performance issue to High Level Architecture. It has been proved that the packets of High Level Architecture are significantly more than some other alternative standards, like Distributed Interactive Simulation, which leads to larger latencies than others consequently.

2.4 Cloud Computing

Cloud computing is a new technology prevailing recently years which integrates a set of technologies on network, large-scale computation, computation architecture and so on. Basically, it provides a novel computing model where users are able to access centralized, powerful and elastic computing resource pool through Internet by the mean of on-demand self- service [21].

The popularity of Cloud computing has produced a great effect on conventional computation model, from individual nodes computation to centralized computation. It provides a centralized, powerful and elastic computing resource pool which clients can access remotely through Internet and utilize in a mean of on-demand self-service [111]. As a novel computing model, Cloud computing allows users to set up their own customized computing environment with cheap cost, which means they do not need to pay for building up powerful computing environment and just rent from Cloud when large computing resource required, and also do not need to maintain lots of computing resource when only small computing resource required [112] [113].

According to common definition, at present, Cloud computing services are categorized to three different kinds [114]: SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service).

Namely, SaaS provides a software functionality in a service model while PaaS provides a platform bundling with a set of related functionalities in a service model. IaaS provides clients access to a wide range of infrastructural computing resources (like CPU and Storage) in a service model [114].

In fact, after a couple years of development, there has been some mature and successful application on those three areas respectively. Gmail is regarded as one of the most successful application of SaaS. Google Application Engineer (GAE) is one excellent example of PaaS application. In IaaS domain, there are more mature and famous cases from various IaaS business vendors, such as Amazon's EC2, Microsoft's Azure and Rackspace's Mosso.

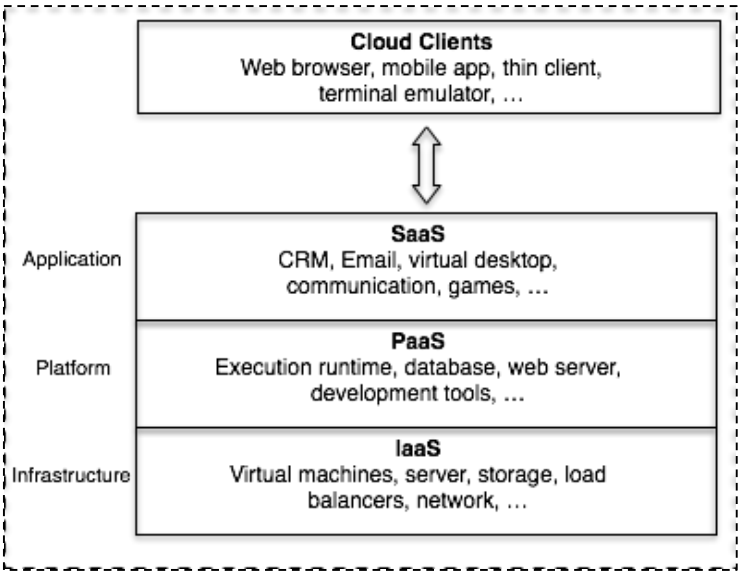

Fig 2.3: Cloud-computing layers [100]

According to deployment model, Cloud services can be categorized to three main kinds [114], including public Cloud, private Cloud and hybrid Cloud. Public Cloud is one kind of Cloud infrastructure which is provided for open use by the general public. There are some famous public Cloud service vendors: Amazon AWS EC2, Microsoft Azure and Google Compute Engine. By contrast, private Cloud is another kind of Cloud infrastructure which is provisioned for exclusive use by designated users, for example, a company or an organization. There are some popular deployment framework or software which is used to build private Cloud: OpenNebula and

OpenStack. Hybrid Cloud, namely, is a kind of Cloud infrastructure which is a composition of two or more different Cloud infrastructures which are integrated into one Cloud through related technologies.

2.5 Open Cloud Computing Interface

The Open Cloud Computing Interface (OCCI) is a set of open community-lead specifications delivered through the Open Grid Forum, for cloud computing service providers [101] [102].

It is an open source standard on interoperability of Cloud computing. Basically, it defines a set of specification to provide an interoperable interface which enables various functionalities of Cloud computing on different Cloud platforms or Cloud service providers. It was originally designed to support Cloud computing in IaaS, however, now it has been developed to a general, flexible and interoperable standard supporting IaaS and PaaS, which features a set of advantages, including interoperability, portability, integration and innovation, plus a high degree of extensibility [101].

The OCCI specification consist of a set of related documents which define various functionalities in each domain. There are three categories of documents, including OCCI Core, OCCI Renderings and OCCI Extension [102]. The OCCI Core specification defines the OCCI Core Model which provides the fundamental and core functionalities of OCCI [101]. The OCCI rendering specification contains several documents which provides a group of rendering of the OCCI Core Model. Renderings can communicate with the OCCI Core Model with different approaches, as well as extensions of the OCCI Core Model. The OCCI Extension specification includes a set

of documents which provide extension specifications of the OCCI Core Model. According to the OCCI Extension specification, extensions can be easily added to the OCCI Core Model, which extends capability of the OCCI Core Model greatly.

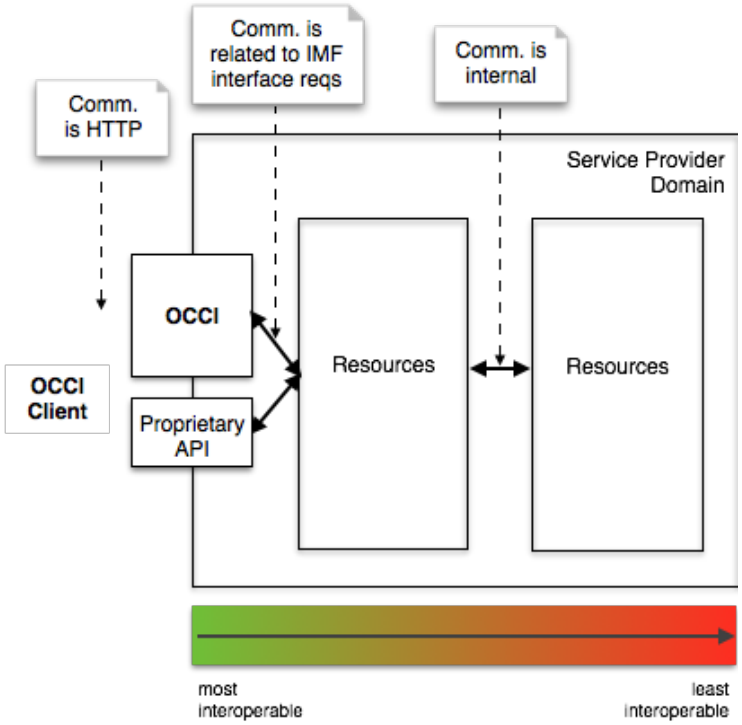

Fig 2.4: OCCI's place in a provider's architecture [101]

OCCI was initialized in March 2009 and was firstly lead by co-chairs from the once SUN Microsystems, RabbitMQ and the Universidad Complutense de Madrid [101] [111]. Today, the working group grows to an organization consisting of over 250 members and includes numerous individuals, industry and academic parties [101] [111]. Some of these members from industry that have contributed include Rackspace, Oracle, Platform Computing, GoGrid, Cisco, Flexiscale, ElasticHosts, CloudCentral, RabbitMQ, CohesiveFT and CloudCentral [101] [111]. Some of these members from academia and research that have contributed include SLA@SOI, RESERVOIR, Claudia Project, OpenStack, OpenNebula and DGSI [101] [111].

Chapter 3

Related Work

As discussed above, some effort on applying Cloud computing to conventional simulations has been made by researchers and professionals. A fact is that simulation on Cloud computing is still at an early stage. A few schemes or solutions have been proposed and some experiments or implementations have been completed. Although there are different kinds of defects and drawbacks in those work which can be resolved or improved, these research on simulation on Cloud computing which has been performed shows excellent examples on exploring in this area and provides valuable experience. In this chapter, some related research work will be introduced and discussed.

3.1 Distributed Simulation over Cloud Environment

In fact, some researchers have been trying to plant or port existing distributed simulation over Cloud environment, which can be considered early effort on applying Cloud computing on distributed simulations. It is accepted that these effort has shown excellent examples on exploring in this area and provides valuable experience. A group of related work of distributed simulation over Cloud environment will be presented in following paragraphs.

3.1.1 COSIM-CSP

In [104], the authors propose and design a Cloud simulation platform prototype, named COMSIM-CSP, which is developed from conventional grid simulation, applying Cloud computing on the traditional distributed simulation and integrating a set of related technology and methods including virtualization technology, pervasive computing and high-performance computing [104] [114] [121]. Basically, the motivation of the authors is to resolve some issues of conventional distributed simulation based on grid computing to improve and strengthen existing grid simulation systems [104].

According to the motivation, some issues or problems are expected to be resolved or improved in the new Cloud simulation platform [104], including the capability to share a considerable volume of computation resources among distributed nodes, multi-users supporting, collaboration mechanism of various simulation components, fault tolerance mechanism and security mechanism.

In order to resolve those issues or problems, the author proposes an architecture of the Cloud simulation platform as shown in Fig 3.1. Briefly, the architecture is constituted of three different layers, including resource/capability layer, Cloud simulation platform layer and service application layer.

The resource/capability layer is in charge of integrating all kinds of hardware and software resource to provide computation and simulation capability for upper layers. It poses the ability to encapsulate various computation resources and share integrated resources over distributed nodes [104].

The Cloud simulation platform layer plays a vital role in the entail architecture. It is made up of four sub layers, namely perception/connection layer, virtual resource/capability layer, core function layer and user interface layer [114]. Perception/connection layer is placed upon the resource/capability layer where the perception, connection and networking of simulation resources are completed [114]; Virtual resource/capability layer encapsulates various simulation resources into one virtual simulation resource pool using a set of virtual technology where physical resource are transformed into logic or virtual simulation resource [114]; Core function layer implements a simulation middleware which contains a group of functionality components including virtual resource/capability management component, knowledge/model/data management component, system running management component, system service evaluation component and system dynamic construction management. User interface layer provides a unique interface and portal accessing various simulation resources and services to all kinds of simulation users through web browser [114].

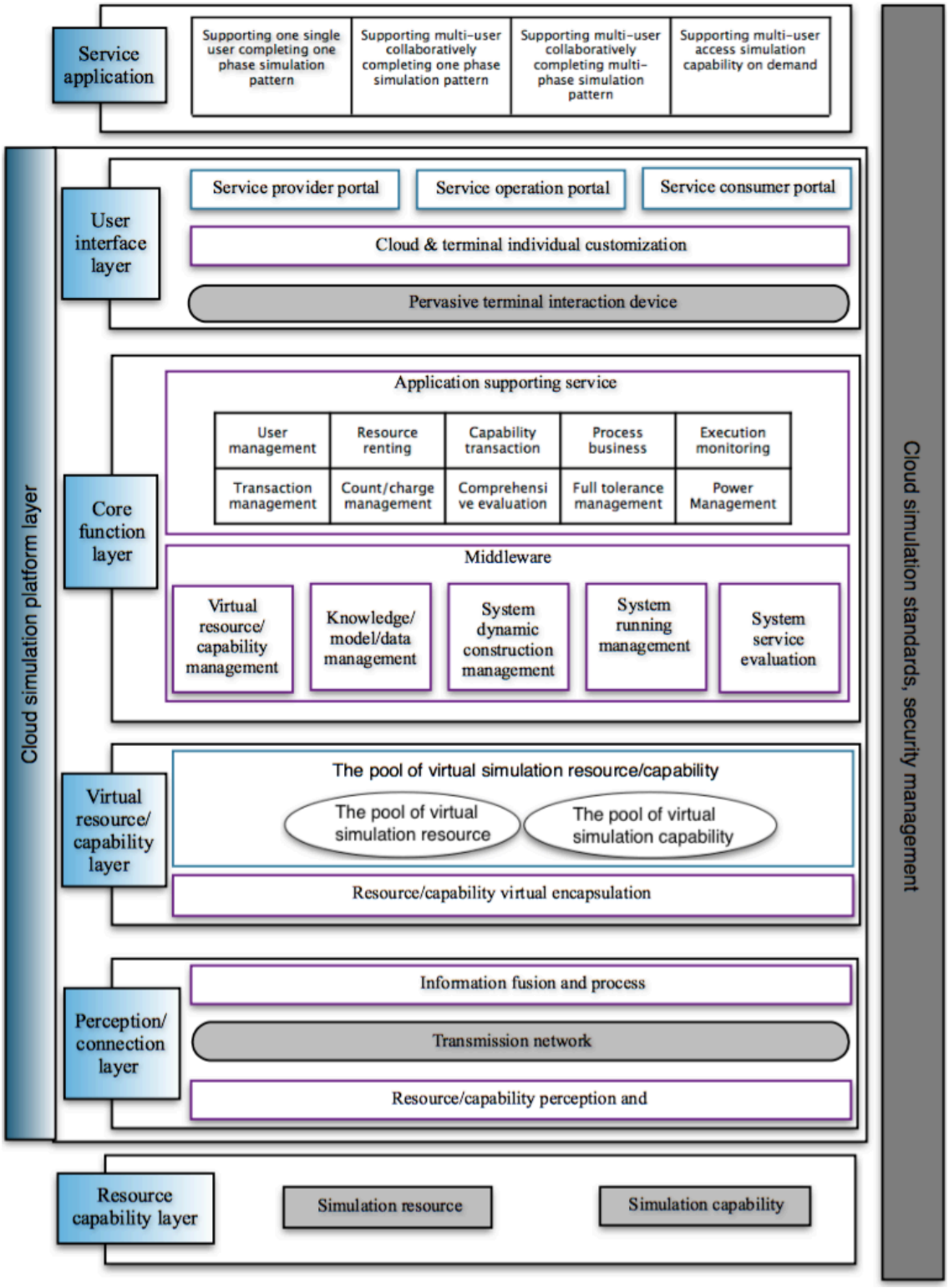

Fig 3.1: Architecture of the cloud simulation platform [114]

Based on the Cloud simulation platform layer, the Service application layer

fulfills fully multi-user supporting which implements four different using modes including single user completing one phase simulation activity, multi-user collaboratively completing one phase simulation activity, multi-user collaboratively completing multi-phase simulation activity and multi-user access simulation capability on demand [114].

According the proposed architecture of Cloud simulation platform, a Cloud simulation platform prototype, COSIM-CSP, is implemented to prove and test the feasibility and functionality of proposed architecture. In order to implement the Cloud simulation platform prototype, the author develops or employs [87] a series of technologies or mechanisms to support various functionalities of the architecture [104] [114], including pervasive portal technology, simulation resource services scheduling technology, simulation system fault tolerance and migration technology, individual virtual desktop technology, multi-user oriented dynamic building technology of virtual simulation environment, fault-tolerant migration technology for simulation resources, high-performance Cloud simulation supported co-simulation platform technology, monitoring and evaluation technology for Cloud simulation, problem solving environment technology, virtualization portal technology, web-based HLA/RTI distributed interactive simulation technology, simulation resource management technology based on web service/grid technology, simulation resource discovery technology based on semantics, simulation resource services automatic composition technology, simulation resource services scheduling technology, simulation system fault tolerant and migration technology, simulation resource virtualization technology and security mechanism and user management technology [114].

Based on the implemented Cloud simulation platform prototype, a group of simulation applications are developed or deployed which shows acceptable functionality and performance from the view of simulators.

3.1.2 CDS

In order to improve some deficiency of High Level Architecture (HLA) and enhance the capability of HLA, a group of researchers integrated Cloud computing with their work on grid-based distributed simulation [88] and accordingly proposed a new efficient and secure Cloud-based distributed simulation system [87], namely CDS, to satisfy the new requirement of large-scale distributed simulation effectively. The proposed simulation system introduces the Cloud computing into distributed simulation to provide a service-oriented simulation platform, addressing some essential issues of HLA and old grid-based distributed simulation. The improvement brought by CDS contains enhanced load balancing capability of the simulation, realizing share and reuse of simulation resources, efficient applicability on wide area network, on-demand service model of simulation resources and reliable security guarantee [88].

Introducing Cloud computing into distributed simulation, CDS employs a new-designed architecture featuring novel characteristics from Cloud computing and extending conventional HLA.

Briefly, 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, which are made up of lots of different components respectively [88].

Heterogeneous resource layer plays a fundamental role in the entail architecture encapsulating and integrating all kinds of hardware and software resources including computation resources, storage resources and network resources, to provide underlying physical resources to upper layers. The Cloud simulation system mange and discover resources from heterogeneous resource layer and port them to upper layers [88].

Core service layer provides a series of essential functionalities by a group of components [88]. The Cloud-RTI component acts as the platform providing the traditional RTI service in the Cloud model. The management center (MC) component takes the responsibility of managing different versions of RTIs and simulation models [88]. The CDS Monitor (CM) component is in charge of monitoring the entail simulation system. The resource requirement description (RRD) component is used to create resource requirement document. And according to the documents generated by RRD, the simulation scene deployment (SSD) component complete the deployment of simulation tasks across distributed nodes. The data collection and process replay (C&R) component is employed to record the dynamic status of simulation tasks and recreate the simulation process when necessary. The functionality of simulation fault tolerance and migration (T&M) component is to implement the fault tolerant and migration mechanism of the entail system which guarantees the efficient scheduling of load balancing across the entire simulation system.

Simulation model layer stores all kinds of simulation models of the Cloud simulation system which are used by other components of the Cloud simulation system to build specific simulations.

Simulation Cloud portal/tool layer plays the role of a portal of the Cloud simulation system, providing the simulation entry and interface to simulators.

Simulation application layer encloses various simulation applications built on the Cloud simulation platform.

3.2 Interoperability on Cloud Computing

While Cloud computing has been experiencing rapid growth, interoperability on Cloud computing turns out very important for the further development of the Cloud ecosystem and market [104]. Merging and integrating different Cloud platforms relies on the work on interoperability on Cloud computing. Although the research on this area is still at a relatively early stage, considerable effort has been made by lots of researchers and professionals.

The work on interoperability on Cloud computing has been done involves some standards on interoperability of Cloud computing, as well as some middleware and software libraries targeting interoperability on Cloud computing. Some popular and mature standards and middleware/libraries are going to be introduced in the following paragraphs.

3.2.1 Standards on Interoperability on Cloud Computing

The fast growth of Cloud computing rises the requirement of standards on interoperability on Cloud computing. There has been lots of related standards proposed by different organizations [103]. Some of them focus on the data communication level while some others work on the service provision level.

Among these standards, OCCI, as a leading open source Cloud computing interface/standard, plays an important role in the ecosystem of Cloud computing. It is has been discussed in Chapter 2.

Cloud Infrastructure Management Interface (CIMI), another primary open standard on interoperability on Cloud computing, is proposed by Distributed Management Task Force (DMTF) [90] [91]. CIMI is designed to provide a series of standardized definition of resource on different Cloud platforms and related operations on resources, as well as interactions across Clouds platforms so that Cloud consumers can access and mange Cloud resource from various Cloud platforms. At present, CIMI is focused on the IaaS domain. The latest version of CIMI gives a description of the architecture enclosing all components and APIs as shown in Fig. 3.3.

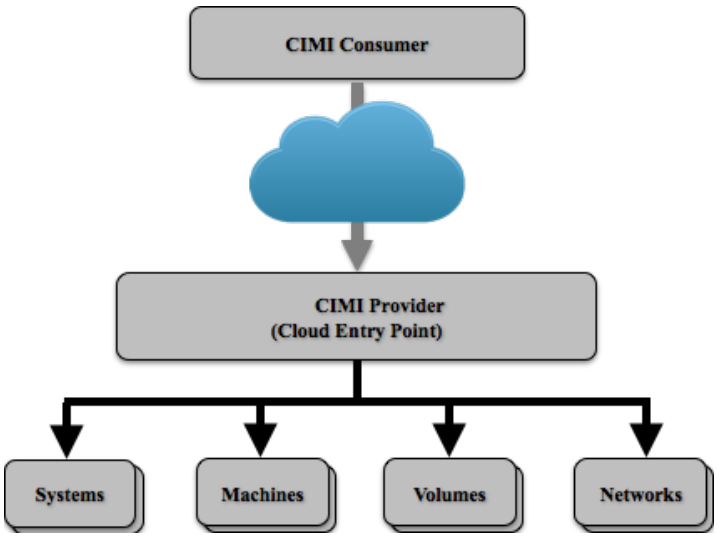

Fig 3.2: CIMI model [91]

Cloud consumers access Cloud resources through CIMI provider interface, Cloud Entry Point of the architecture, which acts as the main entry into the IaaS providers [91]. A group of domains and related constraints are defined to describe various capabilities and resources in the Cloud platform. All components or domains

provide an interface by which the system can discover various capabilities and resources automatically.

The core resources of CIMI model include systems, machines, volumes and networks. The structures of them are illustrated in Fig. 3.4 and Fig. 3.5. Systems in the

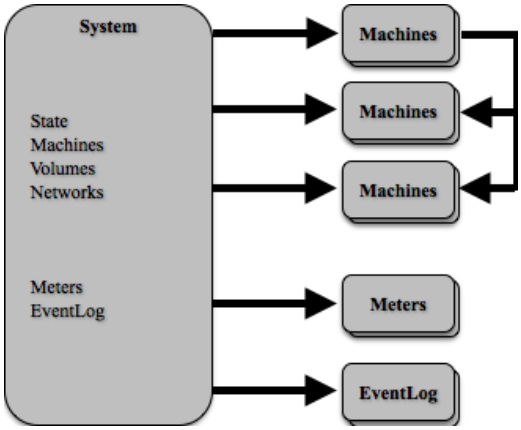

Fig 3.3: System in CIMI model [91]

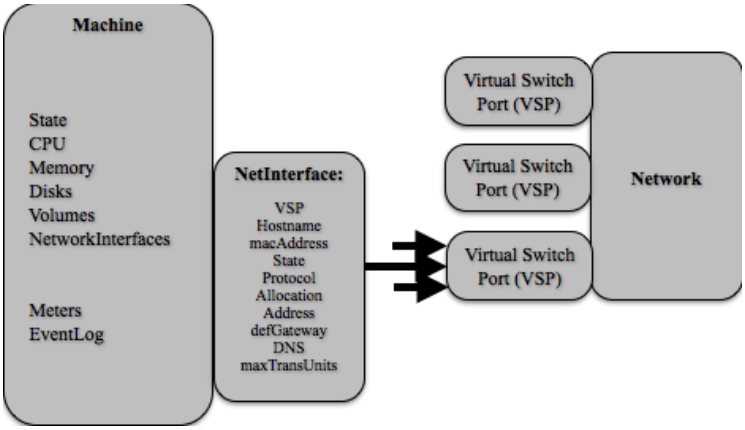

Fig 3.4: Machine in CIMI model [51

CIMI model defines groups of resources which can be managed as one single unit [91]. 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.

3.2.2 Middleware/Libraries on Interoperability on Cloud Computing

In addition to standards introduce above, there are also lots of middleware/libraries on interoperability on Cloud computing developed by open source organizations or commercial companies. Most of them involve specific programming languages which provides convenient and effective toolkits or APIs to build applications across Cloud.

3.2.2.1 LibCloud

LibCloud is a open source programming library written in Python which aims to provide a unified API interacting with different Cloud service providers [92]. The purpose of it is to facilitate the development of application working over various Cloud platforms. Originally, LibCloud was developed in 2009 by Cloudkick [93] in order to resolve the issue of interacting with multiple different Cloud provider APIs[92]. Late in 2009, the project joined Apache Incubator and then graduated from the incubator to a top level project in May of 2011 [92]. LibCloud is designed to support the management of several kinds of Cloud resources [92], including Cloud Servers and Block Storage, Cloud Object Storage and CDN, Load Balancers as a Service and DNS as a Service.

At present, LibCloud supports more than 30 Cloud providers, including Amazon EC2, Google Compute Engine and OpenNebula and so on.

3.2.2.2 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 [94]. It implements a series of Cloud services from fundament to top level, with a group of components including Collections, Requests and Mocks [100]. Collections provide a simplified interface, making clouds easier to work with and switch between [100]. An abstract interface, implemented through a set of collections, is provided for each Cloud platform. Requests allow power users to get the most out of the features of each individual cloud [100]. Requests provide an advanced interface when some complex and deep functionalities are required. An interface is also given to discover all available request of specific Cloud platform. Mocks provide a smart mechanism for tests which allow skipping overhead of operations on application instances and accordingly make testing and integrating of code easy [100].

Now there are more than 30 Cloud providers which have been supported by fog [101], such as Amazon EC2 and OpenStack.

3.2.2.3 JClouds

JClouds is an open source multi-Cloud toolkit for the Java platform which gives developers the freedom to build Java software portable and interoperable over various Clouds [102]. Moreover, it gives developers the ability to use Cloud-specific features with full control [102].

JClouds offers a set of abstract APIs as Java libraries which can be used to create applications in Java on various Cloud platforms. Some most mature of these APIs

includes ComputeService, BlobStore and LoadBalance [103]. ComputeService provide basic functionalities of instances management in Cloud. Allow users to instantiate multiple compute instances at once as well as software installation on them. BlobStore provides a simplified and portable interface to access and manage storage across various Clouds. It features presenting a straightforward map view of contained data in your storage. LoadBalancer abstracts a unified interface to manage and schedule load balancers on various Cloud platforms. It is allowed to create load balancer and associated nodes which will perform related actions of load balancing.

It provides a simplified and portable interface to access and manage storage across various Clouds. It features presenting a straightforward map view of contained data in your storage. According to [104], JClouds now supports 30 Cloud providers and Cloud software stacks including Amazon, Azure and OpenStack.

3.2.2.4 DeltaCloud

DeltaCloud is a Ruby API originally proposed and developed by Red Hat and the Apache Software Foundation [105]. It provides a unified abstract REST-based API to perform the management of services deployed across various Cloud platforms. DeltaCloud employs a set of drivers to adapter different Cloud platforms. Three different frontends are supported by DeltaCloud. They are DeltaCloud classic frontend, DMTF CIMI frontend and Amazon EC2 frontend. The architecture of DeltaCloud is described in Fig. 3.6.

The first version of DeltaCloud was released in September of 2009. In May of 2010, the project was placed into the Apache Incubator project and then graduated from

the incubator, as on top level project in October of 2011 [105]. In July of 2015, DeltaCloud was moved to the Apache Attic due to inactivity [105].

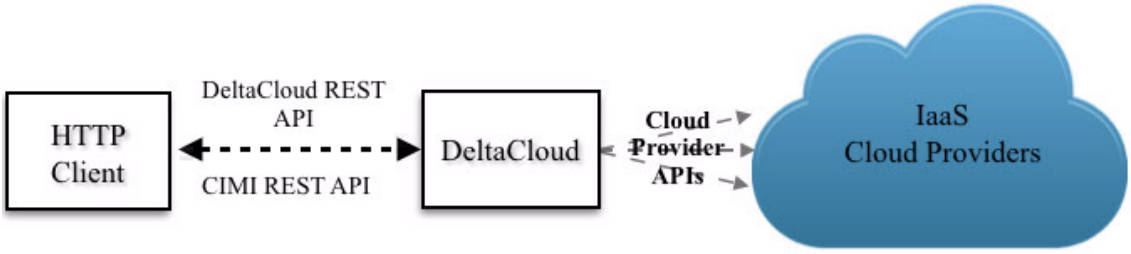

Fig 3.5: Architecture of DeltaCloud [106]

3.3 Summary

From related discussion above, we know there are much effort of planting existing distributed simulation on Cloud computing. However, there is one significant drawback of those conventional distributed simulation platforms/systems which 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. It also reduces greatly the capability of encapsulation and integration of resources from various Cloud platform. This drawback significantly degrades the portability and compatibility of Cloud simulation platform/system, posing great negative effect on the functionality and capability of the distributed simulation platform/system on Cloud environment.

Related work on interoperability on Cloud computing is considered important to improve the drawback of lack of interoperability in those platforms. Related effort of interoperability on Cloud computing are discussed in this chapter. Some standards of interoperability on Cloud computing and a group of middleware/libraries on interoperability on Cloud computing are reviewed respectively. These work can be

utilized to improve the drawback of those conventional distributed simulation platforms/systems on Cloud environment.

Among those related work, OCCI and CIMI are two similar standards focusing on the interoperability of Cloud computing. Even though both of them are managing to provide interoperable interface across various Cloud platforms, they differ in some areas. Firstly, OCCI is designed to an extensible standard which can be extended to PaaS and SaaS domains while CIMI now is specific to IaaS domain. It is apparent that OCCI can be applied to wider domains than CIMI. Secondly, OCCI provides a group of abstract defined models in an inheritable architecture while CIMI provides a group of completely defined and independent models. This feature leads to the better flexibility of OCCI than CIMI. The comparison between OCCI and CIMI is shown in Tab. 3.1.

Items	**OCCI**	**CIMI**
Open Source	Yes	Yes
Domain	IaaS, PaaS, SaaS	IaaS
Models	Abstract inherited models	Independent models

Tab 3.1: Comparison of OCCI and CIMI

In this thesis, OCCI, the leading open source Cloud computing interface/standard, is employed to implement the interoperability of distributed simulation system on Cloud environment, which is one of objectives of this thesis. In the next chapter, a new interoperable distributed simulation system over a Cloud environment, based on OCCI, will be proposed and described in details. The implementation of prototype and experiments as well as related analysis will also be discussed in later chapters.

Chapter 4

Interoperable Simulation System over a Cloud Environment (ISSC)

According to discussion in the former chapters of this thesis, it is not difficult to obtain a conclusion: distributed simulation on Cloud can bring conventional simulation great benefits from the novel Cloud computing paradigm, which is very prospective, however, the work of applying Cloud computing on distributed simulation is still at an early stage, which needs to be improved and advanced. The fact is that existing schemes or solutions of distributed simulation on Cloud environment have various limitations or drawbacks. One apparent drawback of them is the lack of interoperability across Cloud platforms, which has been posing considerable negative effect on the usability and flexibility of distributed simulation on Cloud environment.

Based on the research of existing schemes and solutions of distributed simulation on Cloud environment, and characteristics of distributed simulation and

Cloud computing, in this thesis, a novel interoperable distributed simulation system on a Cloud environment, ISSC, is proposed which integrates Cloud computing with conventional distributed simulation, providing an easy-of-use and unified interface to elastic and portable simulation resources and services, with theoretically infinite computation capability in an economic cost. This platform features the great interoperability of Cloud platforms, which means it can be deployed on various Cloud platforms and perform all kinds of interactions across these Cloud platforms.

In this chapter, the details of this simulation system will be discussed. A prototype of this simulation system has been implemented. A group of experiments and related analysis were performed on this prototype platform which will be introduced in the next chapter.

4.1 Overview

This simulation system is an interoperable simulation system over a Cloud environment. It is designed for HLA distributed simulations, adapting the new Cloud environment. With prepared simulation environments and a convenient user interface, the simulation system provides a one-stop portal for simulators. Built over Cloud environment, the simulation system adopts novel features of Cloud computing, bringing cost reduction, productivity, reliability, scalability and elasticity to simulators. Employing OCCI, it features the interoperability across Cloud platforms, offering the ability to integrate resources from various Cloud platforms into the simulation system, from public Cloud platforms, saying Amazon EC2, to private Clouds built by OpenNebula, OpenStack and

so on, which improves the scalability, expandability and flexibility of the simulation system greatly.

At present, this simulation system supports HLA distributed simulation. It provides a prepared simulation environment for HLA distributed simulation through designed virtual machines, enclosing custom computing resources and simulation images, and convenient simulation portal. Simulator can deploy various complex HLA simulations easily and conveniently with the uniform user interface and prepared simulation portal. The simulation system prepares custom computing resources and environment for users so that simulators are able to deploy and perform their simulations fast through convenient simulation portal, leaping over the great difficulty of building complex computing system and simulation environment. The prepared simulation environment integrates enough computation resources and necessary simulation tools and components including configurations, storages, simulation networks and so on. In the future, support for other distributed simulation models can be added conveniently.

4.2 Features and Scenarios

Built on a Cloud environment, this simulation system can provide much more sufficient, scalable and flexible computation resources. The Cloud computing environment features scalability and flexibility on one-demand computing service, which integrates a set of technologies on network, large-scale computation, computation architecture and so on. It offers a novel computing model where users are able to access centralized, powerful and elastic computing resource pool through network by the mean of

on-demand self-service. As a novel computing model, Cloud computing allows users to set up their own customized computing environment with cheap cost, which means they do not need to pay for building up powerful computing environment and just rent from Cloud when large computing resource required, and also do not need to maintain lots of computing resource when only small computing resource required. This simulation system employs those novel features of Cloud computing and applies on simulation computing, bringing great cost reduction and powerful computation capacity to simulators.

This simulation system employs OCCI to construct the operability over various Cloud platforms, which improves the scalability, expandability and flexibility of simulation system greatly. One fact of Cloud computing is that various Cloud platforms are independent and isolated separately. They have their own interface and API respectively. Each of them can not be interoperated and communicate with each others. Even though there are some commercial factors why they are isolated and separate, it produces great barriers among different Cloud platforms, which impacts the optimal utilization and integration of various Cloud platforms negatively. OCCI is a de facto open source standard on interoperability of Cloud computing, which provides the interoperability among different Cloud platforms. This simulation system adopts OCCI to build the ability of enclosing various computation resources from different Cloud platforms into the simulation system, which gives a solution of interoperable simulation across different Cloud platforms.

Three typical scenarios of applying this simulation system are selected to demonstrate main features of this simulation system in the following discussion.

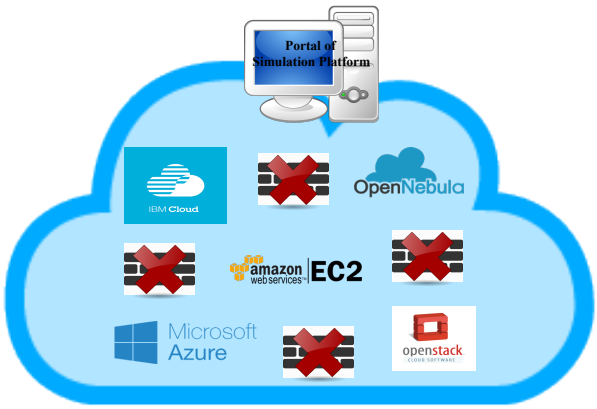

Fig 4.1: Scenario 1 of ISSC

The first scenario in Fig. 4.1 illustrates some essential features of this simulation system: powerful computation capability, elasticity and interoperability. Thanks to the novel computing model of Cloud computing, this simulation system built on a Cloud environment acquires theoretically unlimited computation capability, with an economic cost. Benefiting from characteristics of Cloud computing, the computation resources underlying this simulation system is elastic, which means the computation resources can be acquired in an on-demand model. Moreover, employing OCCI, this simulation system possesses interoperability of Cloud platforms. The interoperability of this simulation system allows more computation resources from different Cloud providers to be encapsulated into this simulation platform.

As shown in Fig. 4.2, the simulation system provides a unified interface to all simulation resources. The unified interface acts as the unique portal to the simulation system which enables simulators access all kinds of functionalities of the simulation system conveniently through one unified interface. Moreover, the unified simulation interface provides one interoperable portal to various simulation resources disturbed across different Cloud platforms and consequently it becomes easy to monitor and

manage various simulation resources on different Cloud platforms via one unified interface.

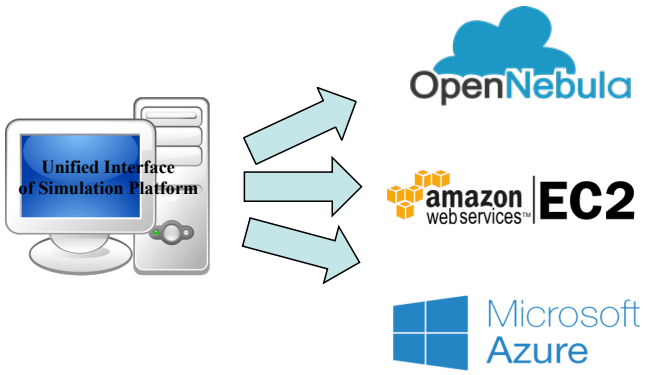

Fig 4.2: Scenario 2 of ISSC

Another typical scenario of applying this simulation system is shown in Fig. 4.3. This scenario demonstrates the scalability and extensibility of the simulation system. Owing to the interoperability of this simulation system, new Cloud resource sources, whatever imported public Cloud providers or new built private Cloud services, can be encapsulated dynamically when more computation resources are required by increasing requirement of simulations. Apparently it greatly strengthens the scalability and extensibility of the simulation system.

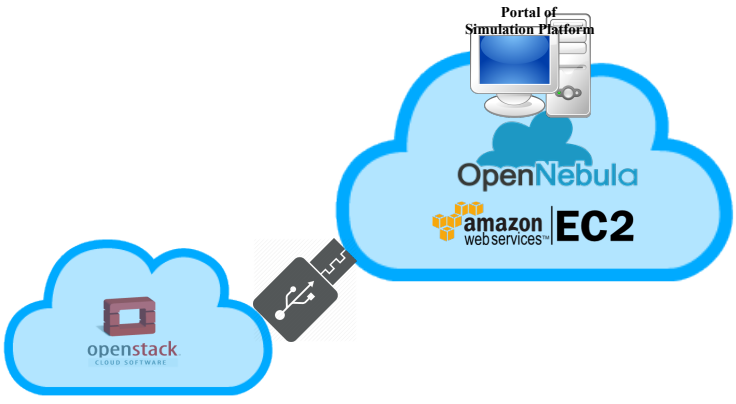

Fig 4.3: Scenario 3 of ISSC

4.3 Architecture

Based on the requirements and features discussed in former chapters, a new interoperable distributed simulation architecture is proposed. Briefly, this simulation system composes of four parts: simulation portal, frontend, backend and Cloud service. The following figure demonstrates the system architecture.

As demonstrated in Fig. 4.4, the platform is made up of four parts, simulation portal, frontend, backend and Cloud service. Simulation portal is the simulation entry and interface of the simulation system to simulators, which provides the unique entrance for simulators to deploy and perform simulations on this simulation system. The fronted 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 platform and supplies computing capacity to frontend. Frontend and backend are constructed based on rOCCI framework, a ruby OCCI framework, through which frontend and backend communicate with the OCCI language. Cloud service represents various Cloud service provider, which can be any Cloud platform supported by OCCI.

As described before, OCCI is employed to help building the interoperability of simulation system across Cloud platforms in this architecture. The rOCCI framework is fundamental components in Frontend part and Backend part. In Frontend part, the rOCCI framework provides a set of application programing interface compliant to the OCCI standard, offering fundamental functionalities of Cloud computing. Requests are translated into the OCCI language so that they can be transported to Backend part where

requests in OCCI format will be processed. The rOCCI framework in Backend part is responsible to adapt to various underlying Cloud services and translate requests in OCCI format into ones compliant to various native Cloud service interface. The functionality of this part plays the role of driver between Backend of ISSC and underlying Cloud platforms. The rOCCI framework in Backend part implements a group of drivers bridging Backend of ISSC and underlying Cloud services.

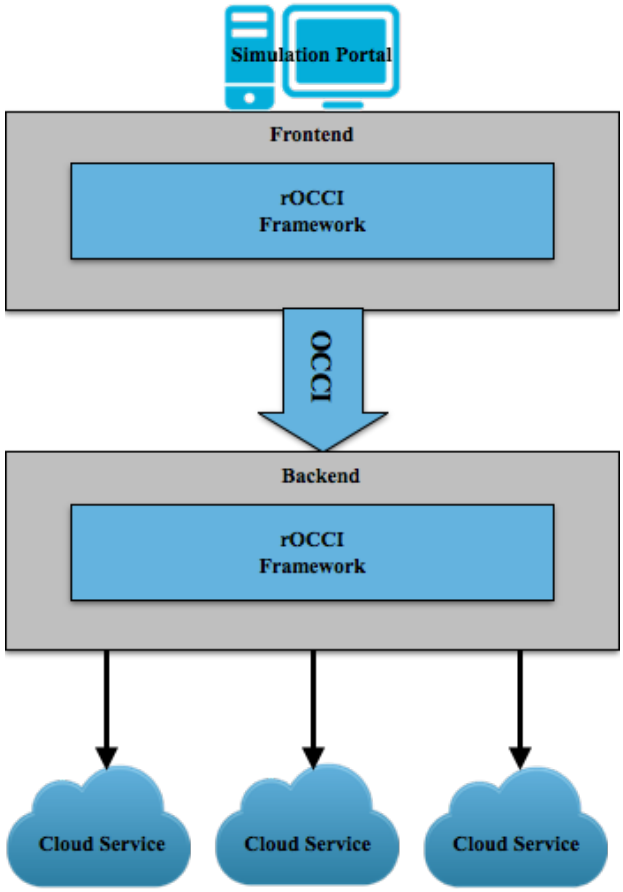

Fig 4.4: Architecture of ISSC

The architecture adopts a distributed deployment model. Components are deployed on different servers geographically and logically, which increases the flexibility of the architecture greatly. Normally, the Simulation Portal component and the Frontend part component are deployed together on the same server while a group of

Backend components are deployed on a set of different servers. Underlying Cloud services are located at same servers with Backend servers or separately.

4.3 Simulation Portal and Frontend

The simulation portal and front-end are deployed together, which plays the role of interacting with and responding to simulators. It is also in charge of communicating with various backends which provide computation resource required by simulation tasks. The architecture of this part is demonstrated in Fig. 4.5.

As demonstrated, this part is built on Apache HTTP Server, Fusion Passenger, Ruby on Rails and rOCCI framework, which reply on and coordinate with each other to providing end-user simulation service.

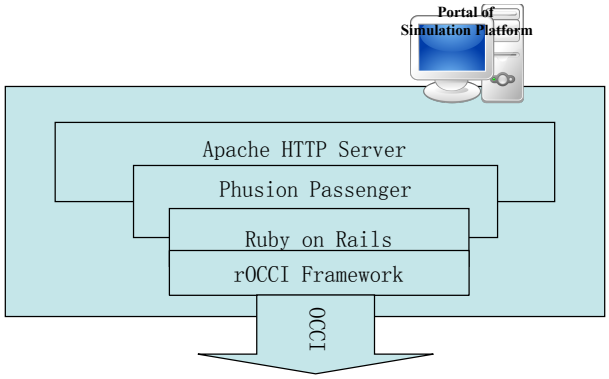

Fig 4.5: Architecture of Simulation Portal and Frontend

Apache HTTP Server is employed to offer web interface of the simulation platform to end-users. The web interface presents a unified simulation entrance to the simulation system, providing easy-to-use and convenient accesses to simulation service. With integrated prepared simulation environment for distributed simulation in the form of virtual machines enclosing custom computation resources and simulation images, the

unified interface acts as a convenient one-stop portal to the simulation system for simulators. Through the unified simulation interface, simulators are able to deploy and perform various complex simulations on this simulation system easily and conveniently. Necessary entities and components, including computation resources, related simulation tools and software, related configurations, storages and simulation networks, are prepared for required simulations, which helps simulators to leap over the great difficulty of building complex computation system and simulation environment and focus on the simulation progress and results.

The web service is implemented in Ruby via Ruby on Rails, a Ruby application framework, based on rOCCI Framework, a Ruby OCCI framework, and is loaded through Phusion Passenger, a Ruby application deployment server which is encapsulated in Apache HTTP Server. With the underlaying rOCCI framework, the front-end interacts with backends speaking OCCI language.

At present, the Simulation Portal of system owns three main entrance, namely overview, new simulation and management, providing different function entrance respectively. Thanks to the excellent extensibility and productivity of Ruby on Rails application framework, it is not difficult to add new function entrance to the web interface, which also increases the extensibility and productivity of the simulation system.

The entrance of overview functionality of simulation system presents all of available simulation resource units, virtual machines, across all enclosed Cloud platforms. Related information of virtual machines, including id, backend which indicates the Cloud platform this virtual machine is deployed, and location which

represents the specific physical network location in the simulation system, will be shown in this functionality. Through the link binding to location of virtual machines, more detailed information of each virtual machine, such as computation resources, storage resources and network resources, can be queried and displayed from associated Cloud platform.

The new simulation functionality of simulation system is another important component of ISSC. This functionality provides a portal for simulators to build a new simulation environment. It allows users to specify a group of configurations to set up a custom simulation environment for simulation tasks. Simulators are able to define the number of virtual machines which are going to be used to carry out planed simulation tasks. Several different types of virtual machine can be specified when different computation requirements are required, which shows the excellent elasticity of this simulation system. Different operation systems, saying Ubuntu, can be chosen to adapt different simulation scenarios. Depending on specified network situation, there are different options for the network type connecting virtual machines. A default network type of virtual machines, OpenVPN, is provided, which connect all created virtual machines through OpenVPN. Thanks to interoperability of this simulation system, which means computation resources can be deployed across different Cloud platforms, users can select different deployment models of virtual machines: deploying all of virtual machines on one specific Cloud platform or on different Cloud platforms. At present, this simulation system supports HLA distributed simulation which is the default option of model of simulation when building new simulation environment. It will not very difficult to be extended to support other distributed simulations if necessary.

The portal of management functionality of simulation system prototype plays an important role in the management of ISSC. It provides basic management functionality of the simulation system, including user management, backend management and image management. The user management functionality implements management functions related to user account, for example, creating new accounts and privilege management of accounts. Backend management is the component supporting the interoperability of the simulation system. Through the backend management functionality, backends binding to various Cloud providers are added into the simulation system with property configurations, which completes the integration of different Cloud platforms. Also it supports basic management of Cloud service backends, such as deleting and modification and so on. Image management portal is used to manage images of the simulation system. An image of the simulation system refers to a packaged image file which can be used to instantiate a virtual machine instance with custom configurations, saying the computation capability and storage size and so on. The images can also contain necessary simulation tools or software prepared for simulation tasks. Images are stored in the simulation system and share across the overall simulation system wherever the Cloud platform is integrated. Accordingly, simulators are able to build their own custom images with predefined configurations and prepared simulation environment, which provides a convenient and efficient approach to deliver and manage custom simulation environment for users, helping to leap over the great difficulty of setting up a complex simulation environment and focus on simulation process and simulation results.

4.4 Backend

The backend of simulation platform plays the role of performing detailed tasks associated with specific background Cloud services. It acts as the bridge between the frontend of the simulation system and underlying Cloud providers. On the one hand, the backend is the section processing request from frontend and responding to specific client with property information. On the other hand, it is responsible to transfer messages to Cloud platforms and collect related data from Cloud platforms. Through backend, the simulation system delivers missions to specific Cloud platform where detailed tasks are performed, and monitor and manage Cloud platforms via communicating required data with Cloud platforms. Along with frontend, backend implements functionalities of the simulation system. Owing to well-designed architecture, the backend also owns excellent extensibility which means it is not difficult to extend the backend with new features or functionalities.

As shown in Fig. 4.6, the backend of simulation system adopts a distributed architecture. Apart from frontend, backend servers are deployed separately in a one-one model which means one dedicated backend server serves one specific background Cloud platform. Any backend server cannot bind to more than one Cloud platform. Any two backend servers cannot also be deployed in one physical server together. However, the backend server can be deployed together with the Cloud platform in the same physical server, or deployed in a separate server physically apart from the Cloud platform. All of distributed backend servers construct an abstract backend layer which implements functionalities the backend is expected to provide.

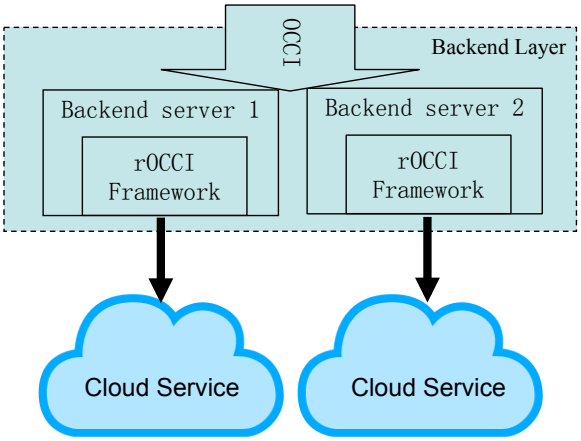

Fig 4.6: Architecture of Backend

Technically, the back server is built on the rOCCI framework, a Ruby framework of OCCI. The rOCCI framework facilitates the interaction between backend server and background Cloud platform. Corresponding with OCCI, the backend server implements related functionalities in universal APIs which ensures the interoperability of the simulation system. A group of configurations, including the type of Cloud platform, the location of Cloud platform, the authentication information of Cloud platform and so on, are used to integrate specific Cloud platform into this simulation system. The backend communicates with frontend in OCCI language while they interact with Cloud platforms with specific proprietary APIs respectively. Employing different drivers, the rOCCI framework take the responsibility of translating the OCCI language to specific proprietary languages of Cloud platforms.

4.5 Cloud Service

The Cloud service layer is the underlying Cloud platforms providing computation resources for the simulation system. Thanks to the excellent of OCCI, various Cloud

platforms can be integrated into the simulation system. The underlying Cloud platform can be a commercial public Cloud, for example Amazon EC2, or a private Cloud built by individuals or organizations. The supporting of multiple Cloud platforms illustrates the interoperability of the simulation system well, which also show the excellent scalability and extensibility of the simulation system.

Backend	Status	Explanation
OpenNebula	production	The OpenNebula Backend is a flagship product for rOCCI, routinely used at multiple sites
Amazon EC2	pilot	The EC2 backend implementation has been developed and tested with Amazon Web Services. It is currently an experimental product and the documentation has yet to be compiled.
VMWare	considered	Based on requests from multiple site, a backend for VMWare products is being considered. It is not yet decided which product in the VMWare range would be targeted.
Windows Azure	in progress	The implementation of an Azure backend is in nearly design stage. There is nothing to try out, yet.
Cloud Stack	considered	The implementation of a CloudStack backend is being considered. There is nothing to try out, yet.

Tab 4.1: rOCCI backend supports status [107]

Employing the rOCCI, a Ruby OCCI framework, the simulation system supports two Cloud service platforms: OpenNebula and Amazon EC2 now. Supporting of more Cloud service platform is going to be implemented. The latest status of rOCCI backend development is shown in Tab 4.1.

Chapter 5

Experiment Results and Analysis

According to aforementioned design and architecture, a prototype of this simulation system, ISSC, is built to prove the feasibility of this simulation system. A series of experiments are carried out on this prototype of simulation system to analyze the performance of this simulation system. In order to compare performance, experiments are performed on both the native Cloud platform (a private Cloud built using OpenNebula) and ISSC. Experiments observe and investigate times on setting up simulation environment, time overhead of ISSC on top of the native Cloud platform, time on executing simulation task and performance of executing simulation application. Based on the process of experiments and the data results of experiments, related analysis and discussion are performed.

5.1 Experiment Environment

According to the proposed architecture of ISSC, the prototype of ISSC is built as a hybrid system, enclosing one frontend server and two backend servers. The architecture of ISSC prototype is demonstrated as follow in Fig. 5.1.

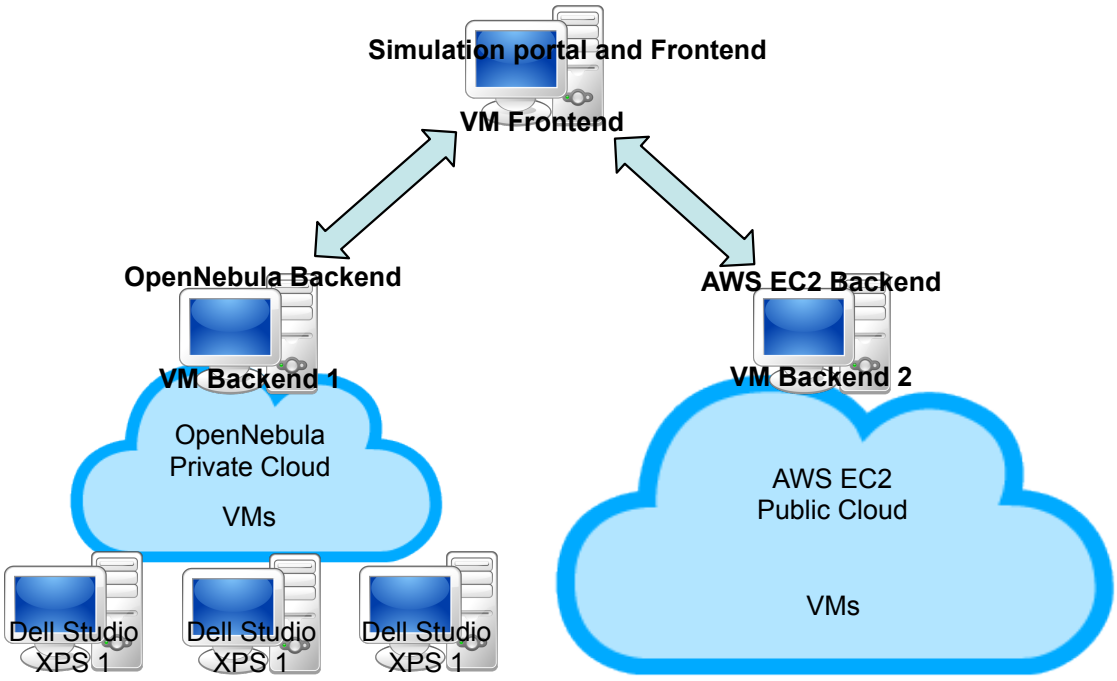

Fig 5.1: Architecture of ISSC prototype

As shown in Fig. 5.1, the prototype is composed of one frontend server and two backend servers. The frontend server is deployed on a virtual machines created by Parallels Desktop 10 on a Mac Mini Server with Mac OS X 10.10.3 Operating System. Two backend servers are deployed on two virtual machines also created by Parallels Desktop 10 on a Mac Mini Server with Mac OS X 10.10.3 Operating System. The specification of Mac Mini Server is described in Tab. 5.1.

Frontend server and backend servers run Ubuntu 14.04 LTS Server Operation System. The specification of frontend server and backend server are described in Tab. 5.2.

One backend server, OpenNebula Backend, is connected to a private Cloud built on three Dell Studio computer servers using OpenNebula 4.10.2. The specification of three Dell Studio computer servers are described in Tab. 5.3.

Item	**Specification**
CPU	Intel Core i5 3335S@2.7GHz
RAM	8GB with dual-channel DDR3@1333MHz
Network	10/100/1000 Mbps Ethernet Network
OS	Mac OS X 10.10.3

Tab 5.1: Specification of Mac Mini Server

Item	**Frontend Server**	**OpenNebula Backend Server**	**Amazon Backend Server**
VCPU	2	2	2
RAM	1GB	1GB	1GB
Network	10/100 Mbps Ethernet Network	10/100 Mbps Ethernet Network	10/100 Mbps Ethernet Network
OS	Ubuntu 14.04 LTS Server	Ubuntu 14.04 LTS Server	Ubuntu 14.04 LTS Server

Tab 5.2: Specification of Frontend Server and Backend Servers.

Items	**Specification**
CPU	Intel Core i7 920@2.66GHz
RAM	6GB with tri-channel DDR3@1066MHz
Network	10/100/1000 Gigabit Ethernet Network
OS	Fedora 23 Linux Operating System

Tab 5.3: Specification of Dell Studio Computer Servers

Another backend server, Amazon Backend, is connected to Amazon AWS EC2 public Cloud service.

Based on built prototype of ISSC, three categories of experiments are carried out. The first category is observing the time overhead of ISSC compared to the native

Cloud platform. The second category is observing the time cost on executing specified simulation tasks. The last category is observing the performance of executing different simulation applications on ISSC.

5.2 Time Overhead of ISSC

In order to examine the performance of ISSC precisely, the first category of experiments are designed to observe the time overhead of ISSC compared to the native Cloud platform. To be more specific, this overhead refers to the extra time cost which the ISSC imposes on top of the underlying native Cloud platform.

In this category of experiment, OpenNebula acts as the baseline to compare with ISSC. A java application is designed and implemented to transfer raw data between two virtual machines. This java application is employed to test the data communication performance between virtual machines. Two virtual machines are deployed on OpenNebula while another two virtual machines with the same specification are deployed on ISSC (2 virtual machines on OpenNebula backend). The specification of virtual machines is demonstrated in Tab 5.7.

Items	**Specification**
VCPU	1
RAM	1024MB
OS	Debian 8.4.0, 64-bit PC

Tab 5.4: Specification of Virtual Machine

The same experiments are carried out on those two groups of virtual machines respectively. The experiment data from virtual machines deployed on OpenNebula build up the baseline. The experiment data from virtual machines deployed

on ISSC will be compared to data from OpenNebula. Through the comparison of two group of experiment data, the time overhead of ISSC can be observed.

The java test application is designed to transfer raw data between two nodes with various speeds according to different parameter sets. There are three parameters determining data transferring speed in the java test application: message_size, message_frequence and experiment_time. The first parameter, message_size, defines the size of each message transferred between nodes; The second parameter, message_frequence, defines the frequence of sending message between nodes per second, which refers to the number of messages sent out per second. The last parameter, experiment_time represents how long the experiment last, which means how long messages are kept sending in the experiment.

In order to examine the time overhead of ISSC precisely, three groups of experiments are performed. In the first group of experiments, message frequency varies from 10 messages/s to 200 messages/s, with fixed message_szie (1000*1000 Bytes) and experiment_time (10s); Message frequency keeps varying from 10 messages/s to 200 messages/s, with the same experiment time (10s) but different message sizes (100*1000 Bytes and 10*1000 Bytes) respectively in the second and third group of experiment. Each experiment is performed 10 times on the native Cloud platform and ISSC. Student's t-distribution is employed to calculate confidence intervals under a 95% confidence level.

The data of three groups of experiments are summarized in Fig 5.2. In the figure, ISSC_100K represents data on ISSC with 10k message_size while

NATIVE_10K represents data on the native Cloud platform (underlying OpenNebula Cloud platform in this experiment) with 10k message_size.

Briefly, the time cost of raw data transfer increases steadily when message frequency grows from 10 messages/s to 200 messages/s with different message sizes. The time overheads of ISSC on top of the native Cloud platform differ in different message frequencies and message sizes. 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 messages/s while the time overhead grows slowly when message frequency exceeds 90 messages/s. If message size is set to 1000000 bytes, the time overhead varies between 8.3% and 25.1%. From the data of experiments, we can find that the time overhead is generated when message size and message frequency increase to fixed values.

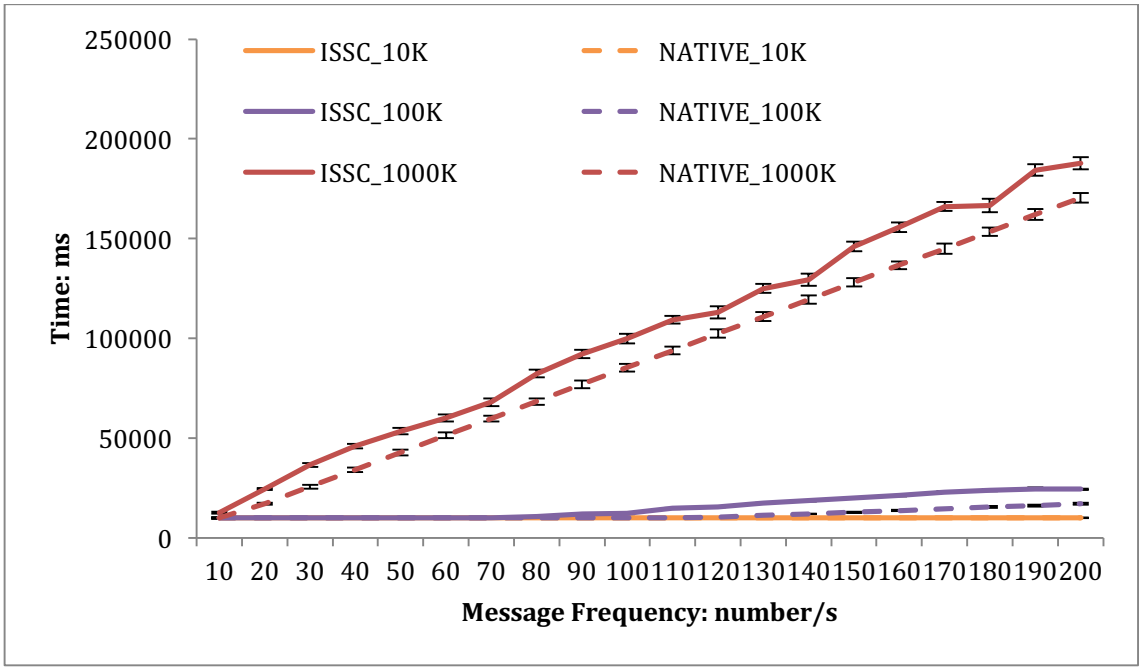

Fig 5.2: Time Overhead Summary

5.3 Execution Time of Simulation Task

The second category of experiments observe the execution time of distributed simulation tasks on the native Cloud platform and ISSC, which refers to the time cost of executing designed distributed simulation tasks on native Cloud platform and ISSC. OpenNebula, the native Cloud platform, acts as the baseline to compare with ISSC in this category of experiments.

In this thesis, portico, a simulation middleware implemented based on HLA, is employed to design and perform HLA-based distributed simulation tasks. The version of portico used to design and implement simulation tasks is 2.0.0 specified to Linux operating system.

According to characteristics of different simulation tasks, two kinds of tasks are designed and performed, including computation intensive simulation task and communication intensive task. Execution times of two kinds of simulation tasks are referred as computation time (CPT) and communication time (CMT) respectively.

In this category of experiments, 10 virtual machines are deployed on ISSC (5 virtual machines on OpenNebula backend and 5 virtual machines on Amazon backend) and the native Cloud platform respectively to carry out simulation tasks. In order to investigate the performance of ISSC from multi views, same simulation tasks are also performed on 5 virtual machines deployed on the native Cloud platform. This simulates the scenario where computation resource is limited in the native Cloud platform so that only limited virtual machines are available to perform dedicated simulation tasks, by contraries, which can be resolved by the great interoperability of ISSC perfectly. The specification of virtual machine created on each platform is shown in Tab. 5.7.

5.3.1 Computation Intensive Task

The first kinds of experiments examine execution times of a group of computation intensive simulation tasks.

Portico is employed to implement a test application that performs float calculation to imitate computation intensive simulation task. 10 federates are created to carry out simulation tasks in this test application. A parameter is used to control the number of calculation iterations, which determines the scale of computation load. The number of calculation iterations varies from 10000000 to 100000000. The experiments are performed on ISSC and the native Cloud platform respectively. Each experiment is performed 10 times on the native Cloud platform and ISSC. Student's t-distribution is employed to calculate confidence intervals under a 95% confidence level.

The related data of experiments are illustrated in Fig. 5.3. In this figure, ISSC_10VMs represents data on ISSC with 10 vms deployed while NATIVE_10VMs represents data on the native Cloud platform (underlying OpenNebula Cloud platform in this experiment) with 10 vms deployed, and NATIVE_5VMs represents data on the native Cloud platform with 5 vms deployed.

It is apparent that the execution times of tasks increase with fixed rate when the number of iterations grows. Due to the performance difference among virtual machines from different backends, the execution time on ISSC is lightly less than the native Cloud platform about 7% when 10 virtual machines are employed to perform simulation tasks. If the number of virtual machines is limited to 5, the execution times on the native Cloud platform reach 200% of ISSC approximately. This result illustrates the great benefit brought by the interoperability of ISSC very well.

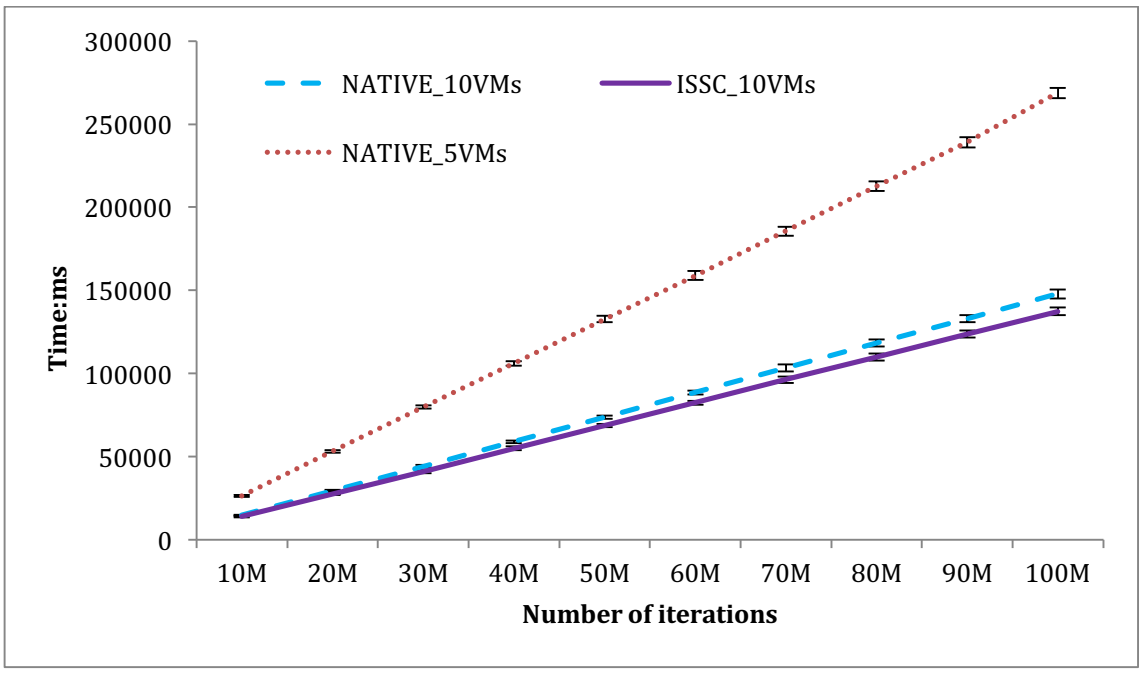

Fig 5.3: Execution Time Summary of Computation Intensive Tasks

5.3.2 Communication Intensive Task

The second kinds of experiments examine execution times of a group of communication intensive simulation tasks which refers to the time cost of message communication among simulation units during simulation.

Portico is employed to implement a communication test application used to imitate communication intensive simulation task. Similarly, 10 federates are created to carry out simulation tasks in this test application. According to characteristics of HLA simulation, two parameters are designed to control the overload of communication among federates in HLA simulation. The first parameter, steps, controls the number of simulation steps each federate need to complete in a whole HLA simulation. From the features of HLA simulation, we know that each federate in a HLA federation will synchronize with other federates in the same federation during each simulation step and

then continue to next step after all of federates of the same federation complete synchronization. The second parameter, messag_size, controls the size of message which are communicated among federate during each simulation step. It is apparent that the total execution time of simulation task primarily depends on the value of those two parameters.

In several groups of experiments, steps varies from 10 to 100 with different fixed message_size (1000000 Bytes , 100000 Bytes and 10000 Bytes). Each experiment is performed 10 times on the native Cloud platform and ISSC. Student's t-distribution is employed to calculate confidence intervals under a 95% confidence level. Based on the data of experiments, a summary is demonstrated in Fig. 5.4.

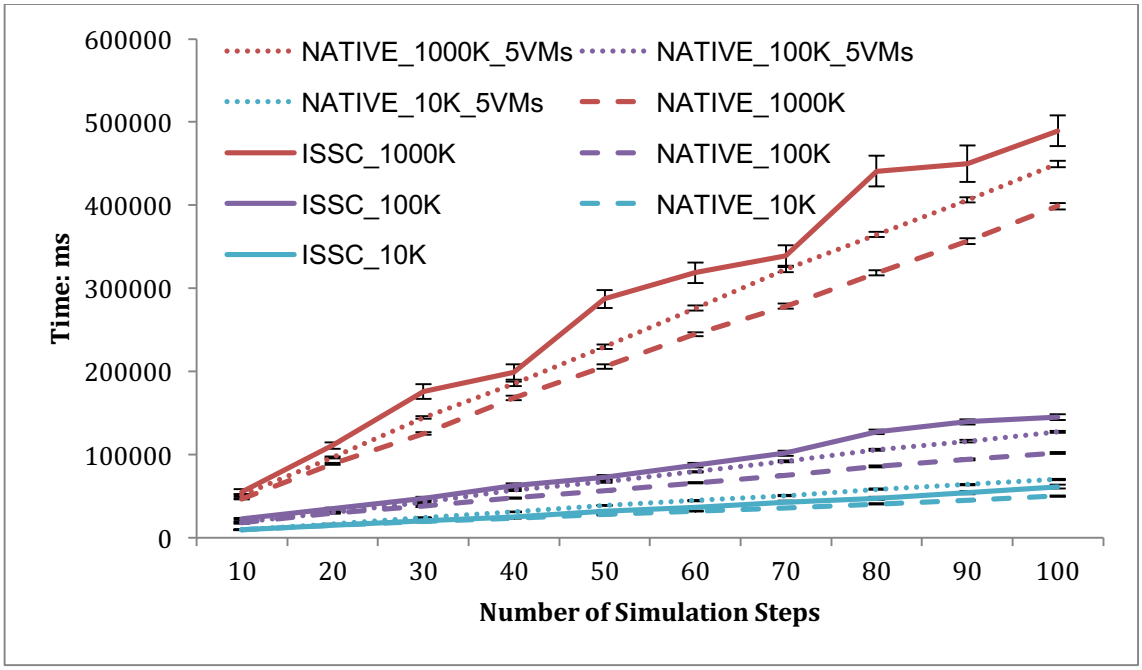

Fig 5.4: Execution Time Summary of Communication Intensive Tasks

From the summary, it can be observed that the performance in the native Cloud platform is better than ISSC when executing communication intensive simulation tasks. The performance gaps differ lightly in different message size. The performance

gap is below 10% at most of time when message size is set to 10000 bytes while it can reach 30% in some points when message size is set to 100000 bytes. The performance gaps can be attributed to the latency of network and the communication overhead among various backends during the execution of communication intensive tasks.

However, the situation is different when the number of virtual machines is limited to 5 on the native Cloud platform. In this scenario, the performance on 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 10000 bytes, which is benefited from the interoperability of ISSC.

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

Considering characteristics of ISSC, another group of experiments are designed and performed to investigate the effect of different virtual machines deployment on various backends on performance of simulation task.

In this thesis, the ISSC prototype used in experiments is consist of two backends: OpenNebula backend and Amazon AWS backend. Totally 10 virtual machines are built on this ISSC prototype. In a group of scenarios, the deployment of virtual machines on backends varies from 10 virtual machines on OpenNebula backend and 0 virtual machines on Amazon backen, to 0 virtual machines and 10 virtual machines on Amazon virtual machines.

A set of simulation tasks, including computation intensive simulation tasks and communication intensive tasks, are performed in different scenarios to investigate

the effect of different virtual machines deployment on the performance of simulation task. Two groups of computation intensive tasks are performed with two different fixed numbers of iterations (10000000 and 100000000). Three groups of communication intensive tasks are performed with fixed number of simulation steps (100) but different message sizes (1000000 bytes, 100000 bytes and 10000 bytes). Each experiment is performed 10 times on the native Cloud platform and ISSC. Student's t-distribution is employed to calculate confidence intervals under a 95% confidence level.

As shown in Fig.5.5, It can be observed that the execution time of computation intensive tasks decrease when the number of virtual machines deployed on OpenNebula drops. The performance is best when there are 10 virtual machines from Amazon and 0 virtual machines from OpenNebula, 11% and 12 % better with two different numbers of iterations respectively. As discussed in Chapter 5.3.1, the reason can be attributed to the performance difference among virtual machines from different backends.

In terms of communication intensive tasks, the execution time displays a curve when the deployments of virtual machines change. It can be found that the execution time is most when there are 5 virtual machines from OpenNebula and 5 virtual machines from Amazon. It decreases slowly when the numbers of virtual machines deployed on OpenNebula and Amazon differ. The performance is best when there are 10 virtual machines from Amazon. This performance difference can be attributed to the communication overhead among virtual machines deployed on different backend Cloud platforms. When virtual machines are deployed on two different backend server, which means the geographical and network distance among

virtual machines is larger, the time overhead of communication among virtual machines deployed on different backend server increases accordingly, which poses a negative effect on the performance of simulation. Apparently, more dispersedly virtual machines are distributed on different backend Cloud platform, worse the performance of simulation is. It is obvious that the performance of simulation is the lowest when same number of virtual machines are deployed on OpenNebula backend server and Amazon backend server respectively.

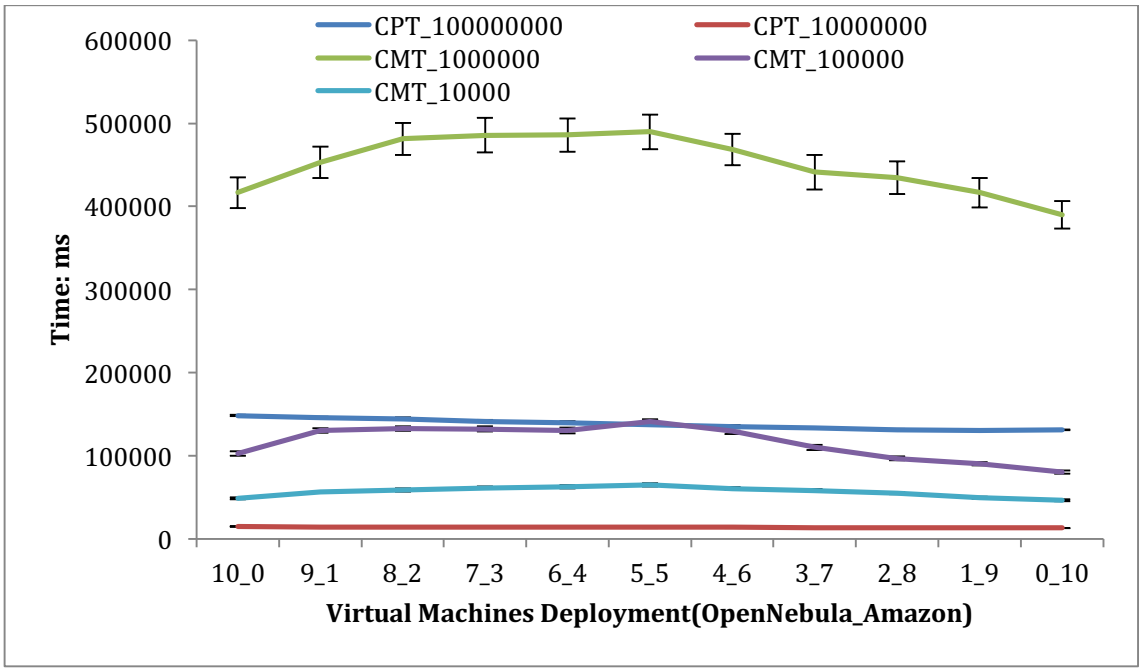

Fig 5.5: Execution Time Summary in Different Deployment

5.4 Performance of Simulation Application

The last category of experiments observes the performance of executing simulation applications on this ISSC prototype system, which refers to the time cost of executing specific HLA-based distributed simulation applications under different scenarios on

ISSC. In this category, three different simulation applications are tested on ISSC prototype platform.

5.4.1 Sushi Restaurant

The first simulation application is sushi restaurant, a HLA-based distributed simulation application, which is employed to design and perform related experiments. The simulation application, sushi restaurant, is a distributed simulation application which simulates the behaviors and operations of a sushi restaurant. It is a well-known simulation application example and widely used as benchmark to test simulation behaviors in the HLA scene/community [113]. This simulation application covers most of parts that a federation can have in HLA, such as ownership management, time management, declaration management and so on [113]. There are lots of papers or project which employ this simulation application [113] [114].

There are five kinds of roles designed and implemented in this simulation application to simulate behaviors and operations of the sushi restaurant. The first kind of role is manager of the sushi restaurant who is in charge of management of the restaurant. The manager is the person who manages employees of restaurant, schedules serving for customers and assigns tasks among other roles. The second kind of role is production which represent chiefs and related employees who are responsible to producing sushi for customers. Chiefs and related employees cooperate with each other to prepare sushi food for customer orders and provide food to transport. The third kind of role is transport which is the representation of employees and related tools responsible for transporting ordered sushi food to dedicated customers. It is the bridge

between production and consumption. The forth kind of role is consumption which represent customers at restaurant. It is the role who makes order of food and consume sushi food at restaurant. The last kind of role is viewer which plays the role of observing transactions of restaurant. It monitors all transactions of restaurant and show them in a visual approach so that we can grab the overview of operations of the whole restaurant.

With those five kinds of implemented roles of restaurant, this simulation application simulates and demonstrates all behaviors and transactions of the sushi restaurant dynamically, including producing, transporting and consuming of sushi food, which consumes computation and communication resource of nodes to complete simulation tasks. In this experiment, sushi restaurant acts as a simulation combination of computation and communication tasks.

This simulation application is implemented based on a commercial HLA simulation middleware, Pitch pRTI, which is developed by Pitch Corporation in Sweden. Pitch pRTI is an excellent and mature commercial distributed simulation RTI completely compliant to HLA standard. It has been widely used in lots of papers or projects [115] [116].

Pitch pRTI provides a platform-independent implementation of all services documented in the HLA Interface Specification [111]. The primary language used to build this middleware is Java. It can run on multiple platforms, including Windows Operating System, Sun Operation System, Linux Operating System and so on. It is a completely HLA compliant simulation middleware. The version of pRTI used in this category of experiments is v5.0.1.1 build 2077 Free specified to Linux Operating System.

The ISSC prototype used in this category of experiments is consist of two backends: OpenNebula backend and Amazon AWS backend. According to characteristics of this simulation application, totally 6 virtual machines are created to carry out this group of experiments. Each role of the sushi restaurant corresponds to one federate in the simulation application, deployed on one virtual machine. Additionally, because the pRTI middleware suite contains the central RTI component (CRC) and the local RTI components (LRC), one virtual machines is used to deploy the central RTI component. In summary, there is one dedicated virtual machine for the central RTI component and one virtual machine for each federate. The specification of virtual machine is shown in Tab. 5.7.

Similarly, same experiments are performed on ISSC with different ratios of virtual machines deployed on two backend servers. In a group of scenarios, the deployment of virtual machines on backends varies from 6 virtual machines on OpenNebula backend and 0 virtual machines on Amazon backen, to 0 virtual machines and 6 virtual machines on Amazon virtual machines.

According to the characteristics of this simulation application, several groups of experiments are designed and performed under different parameter configurations. We change the number of customers we are going to complete serving to determine the scale of this sushi simulation. The time costs of completing dedicated transactions are different depending on the numbers of customers intending to serve. A set of values of this number are tested: 200 serving customers, 100 serving customers and 10 serving customers. Each experiment is executed 10 times to approach correct

data of simulations. Student's t-distribution is employed to calculate confidence intervals under a 95% confidence level.

The summary of data is demonstrated in Fig.5.6. It is apparent that the time costs of simulation increase along with the growth of serving number. It costs around 67s to complete serving 10 customers, 470s to 100 customers and 920s to 200 customers. The time cost of 200 customers is about 1.95 times of the cost of 100 serving while the time cost of 100 customers is about 7 times of the cost of 10 customers, not reaching 10 times.

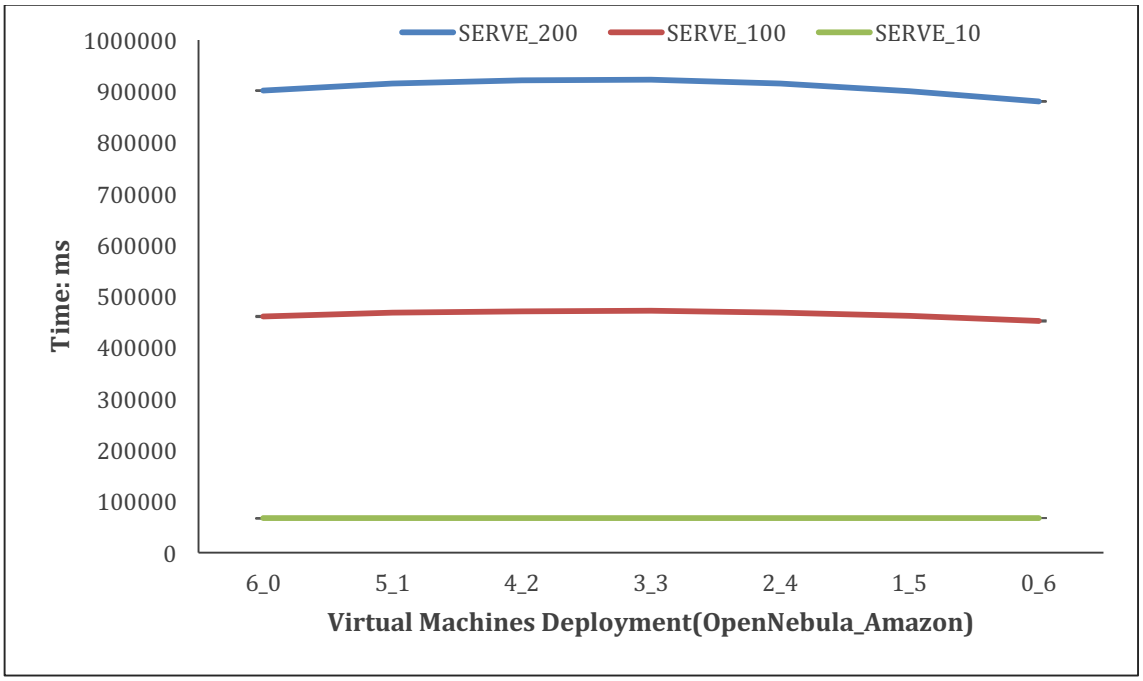

Fig 5.6: Execution Time Summary of Sushi Restaurant Simulation Application

We can also observe that the execution time displays a curve when the deployments of virtual machines change. The curve is not very apparent when the number of serving customer is only 10, with a relatively steady value of 67000ms around. However, the situation is lightly different when the number of serving customer rises to 100 and 200. The curve displays a radian with a peak at the middle of curve and

two nadirs at the beginning and end of curve. This trend turns out apparent when the number of serving customers reaches to 200. It means the time cost of simulation is highest when the number of virtual machines deployed on OpenNebual backend server closes to the number of virtual machines deployed Amazon backend server, while lowest when all of virtual machines are deployed on OpenNebula backend server or Amazon backend server. To be more specific, the value is about 922000ms to serve 200 customers when 3 virtual machines are deployed on OpenNebula backend server and 3 virtual machines are deployed on Amazon backend server, which is about 2% more than the value when all 6 virtual machines are deployed on OpenNebula backend server, and 4.7% more than the value when all 6 virtual machines are deployed on Amazon backend server. Similarly, the value is about 472000ms to server 100 customers when 3 virtual machines are deployed on OpenNebula backend server and 3 virtual machines are deployed on Amazon backend server, which is about 2% more than the value when all 6 virtual machines are deployed on OpenNebula backend server, and 4.4% more than the value when all 6 virtual machines are deployed on Amazon backend server. The time cost of simulation approaches to the peak value when the number of virtual machines deployed on two different backend server close.

This trend can be explained by the overhead of communication among virtual machines very well. When virtual machines are deployed on two different backend server, which means the geographical and network distance among virtual machines is larger, the time overhead of communication among virtual machines deployed on different backend server increases accordingly, which poses a negative effect on the performance of simulation. Apparently, more dispersedly virtual machines

are distributed on different backend Cloud platform, worse the performance of simulation is. Consequently, the performance of simulation drops to the lowest when same number of virtual machines are deployed on OpenNebula backend server and Amazon backend server respectively because the time overhead of communication among virtual machines are largest in that case.

5.4.2 Fuel Economy

The second simulation application in this category is fuel economy which is a HLA-based distributed simulation application used to investigate the economy of fuel consumption of vehicles. The fuel economy simulation is an analysis simulation application that is intended to study and compare the fuel consumption of vehicles under different conditions. To be more specific, this simulation application simulates vehicles driving in given circumstances to examine how far vehicles can drive using a limited amount of fuel so that fuel consumptions of vehicles could be evaluated [117]. It is a common simulation application widely used to verify simulation services and performance in the HLA scene/community [118].

Similar as sushi restaurant simulation application, fuel economy is implemented based on Pitch pRTI which has been introduced in the experiment of sushi restaurant simulation application.

This simulation application is made up of three different roles: master, mapviewer and vehicles. The first role is master of the fuel economy simulation federation which is in charge of management of the entire simulation federation, including setting up driving scenarios, starting and stopping the simulation and so on.

The second role is mapviewer which acts as a monitor or observer of the simulation federation. The status of vehicles, including position and fuel status, will displays on mapviewer. It is in charge of monitoring the real-time simulation federation dynamically. The third role is vehicle which simulates behaviors of vehicles in the simulation federation, including position movement and fuel consumption. Simulators can import many different vehicles corresponding to different vehicle models so that this simulation could be able to evaluate fuel consumption of different vehicle models as many as you can.

The workload on roles in this simulation application contains computation tasks and communication tasks. The role of vehicle in this simulation application keeps calculating and updating position of vehicle in real time. It also keeps calculating and updating fuel consumption of vehicle dynamically. The role of mapviewer keeps updating and rendering the position and movement of vehicle as well as the consumption of fuel in given map in real time. Roles in this simulation application keep communicating with each other to complete the simulation. The role of master sends messages to and receive messages from other roles in this simulation application to control and schedule the entire simulation procedure. The role of vehicle and the role of mapviewer keep transferring information of position and movement of vehicle and status of fuel consumption of vehicle in real time. The role of master also logs and monitors the real-time simulation information of federation dynamically so that further analysis and research on fuel consumption could be conducted. Consequently, the workload of this simulation application is a combination of computation tasks in federate nodes and communication tasks among federate nodes.

The experiment environment used in this experiments is similar as sushi restaurant simulation. The ISSC prototype is consist of two backends: OpenNebula backend and Amazon AWS backend. According to characteristics of this simulation application, totally 6 virtual machines are created to carry out this group of experiments. There is one master federate in this simulation which is deployed on one dedicated virtual machine. There is also one mapviewer federate in this simulation which is deployed on one dedicated virtual machine. Due to the limitation of maximum federates number in a federation by the free license of Pitch pRTI, three vehicle federates are deployed on three virtual machines, which means one vehicle federate is deployed on each virtual machine. And there is also one virtual machine used to deploy the central RTI component of Pitch pRTI. The specification of virtual machine is shown in Tab. 5.7.

Similarly, same experiments are performed on ISSC with different ratios of virtual machines deployed on two backend servers. In a group of scenarios, the deployment of virtual machines on backends varies from 6 virtual machines on OpenNebula backend and 0 virtual machines on Amazon backen, to 0 virtual machines and 6 virtual machines on Amazon virtual machines.

According to the characteristics of this simulation application, several groups of experiments are designed and performed under different parameter configurations. We change the amount of fuel in a vehicle to determine the scale of the simulation. With given amount of fuel, vehicles keep driving until fuel is exhausted. Apparently the time costs of completing entire fuel consumption simulation depend on the amount of fuel provide to vehicles. A set of values of amount of fuel are tested: 100

liters, 10 liters and 1 liter. Each experiment is executed 10 times to approach correct data of simulations. Student's t-distribution is employed to calculate confidence intervals under a 95% confidence level.

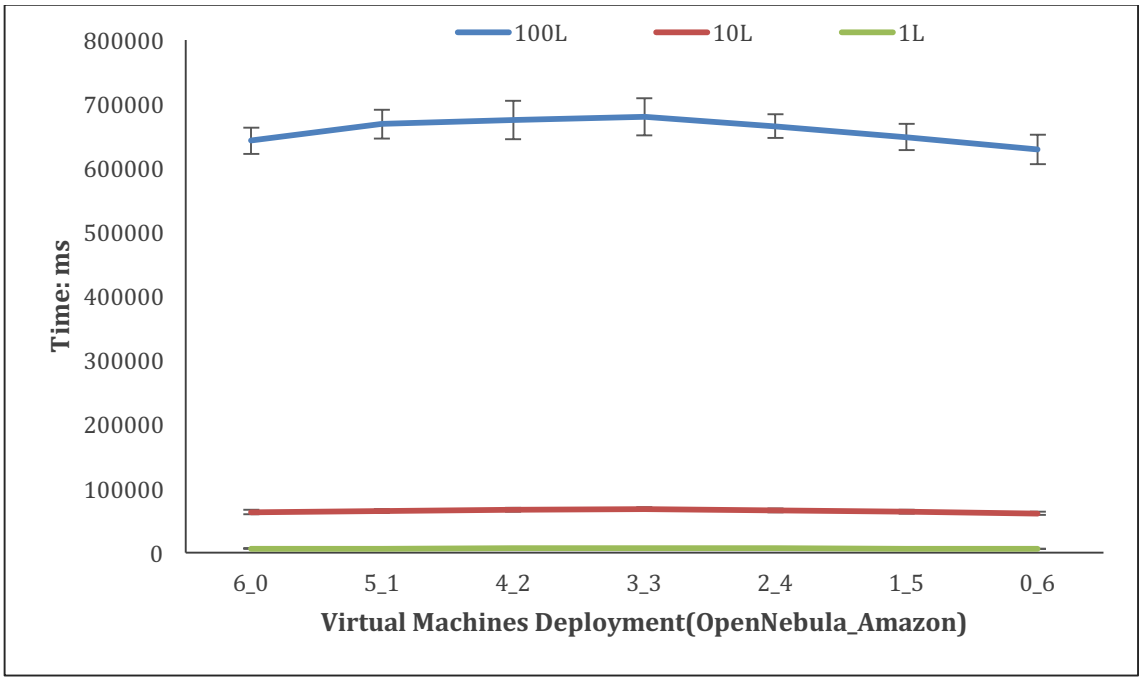

Fig 5.7: Execution Time Summary of Fuel Economy Simulation Application

The summary of data is demonstrated in Fig.5.7. There is no doubt that the time costs of simulation increase along with the growth of the amount of fuel. It costs around 6.9s with 1L fuel, 68s with 10L fuel and 680s with 100L fuel. The time cost of 100L fuel is about 10 times of the cost of 10L fuel while the time cost of 10L fuel is about 9.85 times of the cost of 1L fuel, not reaching 10 times.

The execution time also displays a curve when the deployments of virtual machines change. The curve is apparent when the amount of fuel rises to 100L. The curve displays a radian with a peak at the middle of curve and two nadirs at the beginning and end of curve. It means the time cost of simulation is highest when the number of virtual machines deployed on OpenNebula backend server closes to the

number of virtual machines deployed Amazon backend server, while lowest when all of virtual machines are deployed on OpenNebula backend server or Amazon backend server. To be more specific, the value is about 679000ms when 3 virtual machines are deployed on OpenNebula backend server and 3 virtual machines are deployed on Amazon backend server, which is about 6% more than the value when all 6 virtual machines are deployed on OpenNebula backend server, and 7% more than the value when all 6 virtual machines are deployed on Amazon backend server. The curve shows a trend that the time cost of simulation approaches to the peak value when the number of virtual machines deployed on two different backend server close.

We observed a similar trend in sushi restaurant simulation. Similarly, it can be explained by the overhead of communication among virtual machines very well. When virtual machines are deployed on two different backend server, which means the geographical and network distance among virtual machines is larger, the time overhead of communication among virtual machines deployed on different backend server increases accordingly, which poses a negative effect on the performance of simulation. Apparently, more dispersedly virtual machines are distributed on different backend Cloud platform, worse the performance of simulation is. Consequently, the performance of simulation drops to the lowest when same number of virtual machines are deployed on OpenNebula backend server and Amazon backend server respectively, which can be attributed to the increasing time overhead of communication among virtual machines.

We can also find that the curve in this simulation application is more apparent compared to sushi restaurant simulation application. The reason of this can be attributed to the difference between workload of those two simulation application. The

sushi restaurant simulation application intimate behaviors of all roles in a sushi restaurant, including producing food, transmitting food and consuming food. The major workload imposed on federates refers to the synchronization of federates in the simulation federation. Compared to sushi restaurant, the workload imposed on federates in fuel economy simulation application is larger. As described before, federates corresponding to the vehicle role in fuel economy simulation federation keeps calculation and updating position and fuel consumption of vehicle in real time. At the meantime, the mapviewer federate and vehicle federates keep communicating with each other to transfer position and fuel consumption information and then render the movement of vehicles and fuel consumption in given map dynamically. The amount of computation tasks and communication tasks in fuel economy simulation application is apparently greater than the sushi restaurant simulation application. Considering the factor of communication overhead among virtual machines discussed above and the performance difference between virtual machines from OpenNebula backend and Amazon backend, we can observe this difference on those two curves due to different amount of computation tasks and communication tasks in sushi restaurant simulation application and fuel economy simulation application.

5.4.3 Billiard

The third simulation application is the billiard simulation application. Billiard is a HLA-based distributed simulation application which imitates the billiard game. It simulates behaviors of balls in aspect of physical accuracy. To be more specific, this simulation application simulates accurate physical behaviors of balls including physical

movement of balls, collisions among balls and interactions among balls and table. Billiard is a famous simulation application in the HLA scene/community which is usually employed to investigate performance of different HLA RTI [119] [120].

Billiard simulation application is implemented based on an open source HLA simulation middleware, CERTI, which is developed by the French Aerospace Laboratory (ONERA) [121] [122]. It provides complete compatibility of HLA 1.3 specification and IEEE 1516 specification [121] [122]. As an excellent free HLA compliant RTI middleware, CERTI has been employed in many applications and projects in HLA community [121] [122] [123].

CERTI provides an efficient and platform-independent implementation of HLA services and interfaces. It is mainly implemented using C++ language as well as some Java code. CERTI can runs on several operating systems, including Linux and Windows. It comprises two major components: RTIA and RTIG. RTIA is deployed locally with federates and corresponds to an RTI ambassador process through with federates interact with RTIG remotely. RTIG represents the central component of CERTI which interacts with local RTIA to provide related distributed RTI services of HLA to federates. The version of CERTI used in this experiments is 3.5.1.

The main federate in billiard simulation application is the ball federate which models ball in the billiard simulation federation with a set of accurate position attributes, and simulates behaviors of balls including movement of balls, collisions among balls and interactions among balls and table. It is possible to simulate any number of balls in this billiard simulation federation where each ball corresponds to one ball federate in the billiard simulation federation. Each ball is initialized with a random

position and a random movement before the simulation starts. All of balls are placed inside a given table which has clear boundaries. Balls perform movements inside this given table with any possible collision with other balls and the table. The simulation ends if a stop command is received or the given simulation time is exhausted. Each ball federate is presented with a graphical interface so that the situation of entire simulation could be dynamically observed on each federate node graphically.

Each ball federate in this billiard simulation application undertakes workload of computation tasks and communication tasks. The ball federate keeps calculating and updating position of ball in real time. It also processes any possible collisions with other balls or the table and handles related update of movement and position. Each ball federate in this simulation application publishes the information of its' position and movement and subscribes the information of the information of position and movement from other balls. All of ball federates keep transferring real-time information of position and movement with each other to calculate possible collisions and dynamically render situation of the entire simulation graphically. The workload imposed on federates in this billiard simulation application combines computation tasks and communication tasks.

The experiment environment used in this experiments is similar as last two simulation applications. The ISSC prototype is consist of two backend: OpenNebula backend and Amazon AWS backend. According to characteristics of this simulation application, totally 10 virtual machines are created to carry out this group of experiments. There is one virtual machine where RTIG is deployed while there are 9 other virtual machines where one ball federates is deployed on each virtual machine. In

another word, there are totally 9 balls federates deployed in this billiard simulation federation. All 9 ball federates simulates movement inside the given table with collision with each other and boundaries of the table during the simulation procedure. The specification of virtual machine is shown in Tab. 5.7.

Same experiments are performed on ISSC with different ratios of virtual machines deployed on two backend servers. In a group of scenarios, the deployment of virtual machines on backends varies from 10 virtual machines on OpenNebula backend and 0 virtual machines on Amazon backend, to 0 virtual machines and 10 virtual machines on Amazon virtual machines.

According to the characteristics of HLA simulation, a parameter, steps which controls the number of simulation steps each ball federate need to complete in a whole HLA simulation, is employed to determine the scale of simulation. A set of values of simulation steps are tested: 100, 1000and 1000. Each experiment is executed 10 times to approach correct data of simulations. Student's t-distribution is employed to calculate confidence intervals under a 95% confidence level.

The summary of data is demonstrated in Fig.5.8. We can observe easily that the time costs of simulation increase along with the growth of simulation steps. It costs around 25s with 100 simulation steps, 250s with 1000 simulation steps and 2500s with 10000 simulation steps. From the summary, we can also find that the execution time shows a curve with changes of the deployments of virtual machines. The curve is not very apparent when the number of simulation steps is only 100, with a relatively steady value of 25s around. However, the situation is lightly different when the number of simulation steps rises to 1000 and 10000. This trend turns out clear when the number of

simulation steps reaches to 10000. The curve displays a radian with a peak at the middle of curve and two nadirs at the beginning and end of curve. It means the time cost of simulation is highest when the number of virtual machines deployed on OpenNebual backend server closes to the number of virtual machines deployed Amazon backend server, while lowest when all of virtual machines are deployed on OpenNebula backend server or Amazon backend server. We can notice that the value is about 2720s for 10000 simulation steps when 3 virtual machines are deployed on OpenNebula backend server and 3 virtual machines are deployed on Amazon backend server, which is about 23% more than the value when all 10 virtual machines are deployed on OpenNebula backend server, and 32% more than the value when all 10 virtual machines are deployed on Amazon backend server.

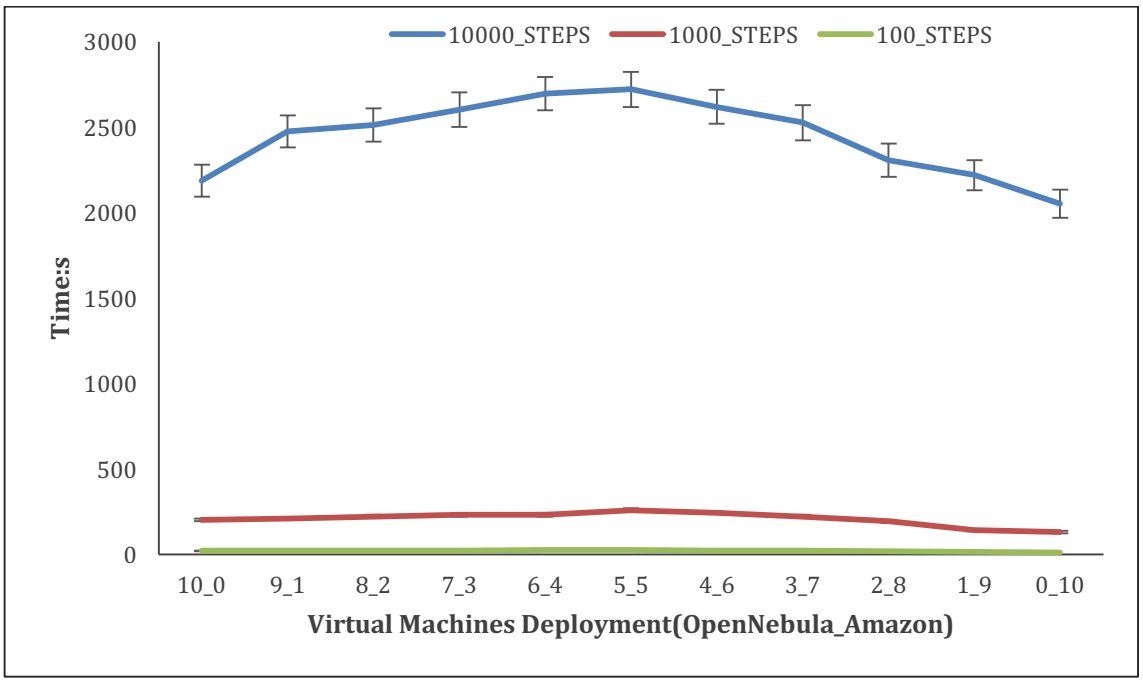

Fig 5.8: Execution Time Summary of Billiard Simulation Application

We can make some analysis on the factor of this difference on performance. Considering the characteristics of HLA simulation and this billiard simulation

application, the major reason leading to this performance difference can be attributed to the communication among federates, which means the overhead of communication among federates deployed on dispersive virtual machines geographically and logically produces the difference on performance in different deployment scenarios. When virtual machines are deployed on two different backend server, which means the geographical and network distance among virtual machines is larger, the time overhead of communication among virtual machines deployed on different backend server increases accordingly, which poses a negative effect on the performance of communication among federates deployed on corresponding virtual machines. In this simulation, each federate publishes the real-time information of its' position and movement, and also subscribes the real-time information of position and movement from other balls. All ball federates keep transferring real-time information of position and movement with each other through the entire simulation procedure so that they cloud calculate possible collisions with other balls and boundaries of the table. To be more specific, each collision between two ball federates produces a set of message communication in the simulation federation. Firstly, a message communication between two ball federates is made once a possible collision between those two balls federates is detected. Those two ball federates calculates new position and movement with information enclosed in transferred message. After the information of position and movement of ball federates involved in this collision is renewed, an operation of updating the information of position and movement of those tow ball federates is triggered in the entire simulation federation to reflect the effect of this collision on the simulation federation. In detail, each ball federate involved in this collision initializes an information update operation

in the simulation scope, which means 2 information update operations would be triggered if this collision occurs between 2 ball federates. Each information update operation involves 9 messages communication between the central RTI component and ball federates in this simulation, which means each ball federate in this simulation federation would receive an information update message from the central RTI component. In another word, there are at least 10 messages communication triggered by each possible collision in this simulation. The number of message communication is even more large if collisions among ball federates is more complex, for instance, the collision involves three ball federates. It is apparent that collisions among ball federates produces lots of workload of message communication in the simulation federation, which explains the remarkable difference on performance of this billiard simulation application under different scenarios very well. Apparently, more dispersedly virtual machines are distributed on different backend Cloud platform, worse the performance of simulation is, and vice versa. In consequence, it is reasonable that the performance of this billiard simulation drops to the lowest when the same number of virtual machines are deployed on OpenNebula backend server and Amazon backend server respectively due to the overhead of communication among virtual machines.

Compared to curves in sushi restaurant simulation application and fuel economy simulation application, the curve in this billiard simulation application is steeper apparently, which can be explained by the much more amount of computation tasks and communication tasks in billiard simulation application than sushi restaurant simulation application and fuel economy simulation application. As discussed above, each ball federate in billiard simulation application is imposed workload of computation

tasks and communication tasks. The workload of computation task imposed on the ball federate includes calculating position of the ball and processing possible collision with other balls or boundaries of the table dynamically. The workload of communication task imposed on the ball federate includes transferring real-time information of position and movement with other ball federates. Each ball federate in this simulation application publishes the information of its' position and movement and subscribes the information of the information of position and movement from other ball federates. As discussed in last paragraph, each collision among ball federates or boundaries of the table triggers an update operation of position and movement information in the entire simulation federation widely so that all ball federates could receive the latest information of position and movement. The frequency of information update operation is high due to the great deal of collision among ball federates and boundaries of the table. The amount of messages communicated corresponding to those update operations is large. It is apparent that the workload of computation tasks and communication tasks in billiard simulation application is much more than the one in sushi restaurant simulation application and fuel economy simulation obviously. Considering the communication overhead among virtual machines discussed before and the performance difference between virtual machines from OpenNebula backend and Amazon backend, we can observe different degrees of radian on those curves because of the difference on the amount of computation tasks and communication tasks in three simulation application, which demonstrates the effect of different virtual machines deployment on the performance on different simulation applications very well.

5.5 Summary

Based on experiments data above, we can summarize the results and then obtain some conclusions. Firstly, the prototype of ISSC can complete various simulation tasks very well including computation intensive tasks and communication intensive tasks. In terms of computation intensive simulation task, the performance on ISSC is better than the native Cloud platform. The performance of communication intensive simulation task on ISSC is worse than the native Cloud platform. However, when the simulation resource is limited, the situation is different. For example, if there are only 5 virtual machines available on the native Cloud platform to be used to perform simulation tasks due to limited budget and lack of interoperability, which means 2 federates are deployed on each virtual machine as demonstrated on experiments above, the performance of two kinds of simulation tasks on the native Cloud platform experience apparent drop. Especially in the computation intensive tasks, the performance on the native Cloud platform is nearly two times worse than ISSC. This situation can demonstrate the great benefit brought by the interoperability of ISSC. Secondly, from the data of executing serval simulation applications, we can find that the performance of simulation application execution is affected by the deployment of virtual machines on different backend. Generally, more dispersedly virtual machines are distributed on different backend Cloud platform, worse the performance of simulation is, and vice versa. Summarily, thanks to the interoperability of ISSC, existing computation resource can be utilized and integrated into the simulation system as a new interoperable Cloud resource

provider, enhancing the computation capability of simulation system, to overcome the limitation of computation resource, which is the value of ISSC.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, based on the investigation of Cloud computing and existing distributed simulation systemsover Cloud platform, a novel interoperable HLA-based simulation system on a Cloud environment, ISSC, is proposed. This novel simulation platform applies Cloud computing on conventional simulations encapsulating the advantages of Clouding computing. It offers clients an easy-of-use and unified interface to elastic and portable simulation resource with theoretical infinite computation capability in an economic cost. The simulation platform owns the capability of being deployed over various Cloud services and the interoperability across different Cloud platforms. This novel simulation platform brings conventional simulations many benefits, such as theoretical infinite computation capability, cost saving on investment, portability and flexibility on deployment, simpleness and convenience on environment setup, and

accessibility and elasticity on service. It also provides a feasible and effective solution on interoperable simulations over a diverse Cloud environment which addresses the concern of interoperability on simulation system over various Cloud platforms.

According to proposed architecture, a prototype of ISSC, is implemented in this thesis, to prove the feasibility of ISSC concept and analyze the performance of ISSC. A series of experiments are designed and carried out on this prototype of simulation system. From the performed experiments, we can obtain a conclusion ISSC shows obvious advantages on the interoperability of Cloud environment. It can be considered as one reliable and efficient solution on interoperable simulation system over a diverse Cloud environment.

6.2 Future work

Based on the work of this thesis, some work can be done in future time. Firstly, at present, only two Cloud platforms are supported by this simulation system. It can be extended to support more Cloud platforms in the future, for example, Microsoft Windows Azure. It is apparent that the wide support to various Cloud platforms will greatly expand scenarios where ISSC is applied. Secondly, at present, only basic functionalities have been designed and implemented in ISSC, which can be improved in the future. For example, at present, OpenVPN is the only option which used to connect virtual machines in ISSC. It is can be extended to other network solutions to increase the adaptability of ISSC to various network situation in the future. Moreover, ther performance of ISSC can also be improve in the future, including the time cost of setting up simulation environment and executing various simulation tasks. Through

optimizing the architecture of the simulation system and network, considerable improvement on performance of ISSC can be expected.

Bibliography

[1] Richard M Fujimoto. *Parallel and distributed simulation systems*, volume 300. Wiley New York, 2000.

[2] Richard Fujimoto. Parallel and distributed simulation. In *Proceedings of the 2015 Winter Simulation Conference*, pages 45–59. IEEE Press, 2015.

[3] Richard M Fujimoto. Distributed simulation systems. In *Simulation Conference, 2003. Proceedings of the 2003 Winter*, volume 1, pages 124–134. IEEE, 2003.

[4] Andreas Tolk et al. *Engineering principles of combat modeling and distributed simulation*. Wiley Online Library, 2012.

[5] Simulation Interoperability Standards Committee et al. of the ieee computer society. ieee standard for modeling and simulation (m&s) high level architecture (hla)-ieee std 1516-2000. new york: Institute of electrical and electronics engineers. *Inc., New York*, 2000.

[6] Simulation Interoperability Standards Committee et al. of the ieee computer society. ieee standard for modeling and simulation (m&s) high level architecture (hla)-ieee std 1516.1-2000. new york: Institute of electrical and electronics engineers. *Inc., New York*, 2000.

[7] Simulation Interoperability Standards Committee et al. of the ieee computer society. ieee standard for modeling and simulation (m&s) high level architecture (hla)-ieee std 1516.2-2000. new york: Institute of electrical and electronics engineers. *Inc., New York*, 2000.

[8] Frederick Kuhl, Richard Weatherly, and Judith Dahmann. *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR, 1999.

[9] Topu Okan, Durak Umut, and Ouztzn Halit. *High Level Architecture*. Springer International Publishing, 2016.

[10] James O Calvin and Richard Weatherly. An introduction to the high level architecture (hla) runtime infrastructure (rti). In *proceedings of the 14th workshop on standards for the interoperability of defence simulations, Orlando, FL*, pages 705–715, 1996.

[11] Azzedine Boukerche and Mirela Sechi M Annoni Notare. Behavior-based intrusion detection in mobile phone systems. *Journal of Parallel and Distributed Computing*, 62(9):1476–1490, 2002.

[12] Azzedine Boukerche and Carl Tropper. A static partitioning and mapping algorithm for conservative parallel simulations. In *ACM PADS and SIGSIM Simulation Digest*, volume 24, pages 164–172. ACM, 1994.

[13] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. Analysis of a randomized congestion control scheme with dsdv routing in ad hoc wireless networks. *Journal of Parallel and Distributed Computing*, 61(7):967–995, 2001.

[14] Azzedine Boukerche and Samer Samarah. A novel algorithm for mining association rules in wireless ad hoc sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):865–877, 2008.

[15] Osama Abumansoor and Azzedine Boukerche. A secure cooperative approach for nonline-of-sight location verification in vanet. *IEEE Transactions on Vehicular Technology*, 61(1):275–285, 2012.

[16] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. *INC, IMS and IDC*, pages 44–51, 2009.

[17] Ling Qian, Zhiguo Luo, Yujian Du, and Leitao Guo. Cloud computing: an overview. In *IEEE International Conference on Cloud Computing*, pages 626–631. Springer, 2009.

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

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

[20] Peter Mell and Tim Grance. The nist definition of cloud computing. *Communications of the ACM*, 53(6):50, 2010.

[21] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. Swimnet: a scalable parallel simulation testbed for wireless and mobile networks. *Wireless Networks*, 7(5):467–486, 2001.

[22] Azzedine Boukerche and Xin Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad hoc networks*, 5(8):1303–1316, 2007.

[23] Horacio Antonio Braga Fernandes De Oliveira, Azzedine Boukerche, Eduardo Freire Nakamura, and Antonio Alfredo Ferreira Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Transactions on Computers*, 58(5):677–691, 2009.

[24] Azzedine Boukerche and Amber Roy. Dynamic grid-based approach to data distribution management. *Journal of Parallel and Distributed Computing*, 62(3):366–392, 2002.

[25] Rodolfo Bezerra Batista, Azzedine Boukerche, and Alba Cristina Magalhaes Alves de Melo. A parallel strategy for biological sequence alignment

in restricted memory space. *Journal of Parallel and Distributed Computing*, 68(4):548–561, 2008.

[26] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[27] Thijs Metsch, Andy Edmonds, et al. Open cloud computing interface-infrastructure. In *Standards Track, no. GFD-R in The Open Grid Forum Document Series, Open Cloud Computing Interface (OCCI) Working Group, Muncie (IN)*, 2010.

[28] Thijs Metsch, Andy Edmonds, R Nyrén, and A Papaspyrou. Open cloud computing interface–core. In *Open Grid Forum, OCCI-WG, Specification Document. Available at: http://forge. gridforum. org/sf/go/doc16161*, 2010.

[29] Guilherme Sperb Machado, David Hausheer, and Burkhard Stiller. Considerations on the interoperability of and between cloud computing standards. In *27th Open Grid Forum (OGF27), G2C-Net Workshop: From Grid to Cloud Networks*, 2009.

[30] Claus Pahl, Li Zhang, and Frank Fowley. Interoperability standards for cloud architecture. 2013.

[31] Patrícia Takako Endo, Glauco Estácio Gonçalves, Judith Kelner, and Djamel Sadok. A survey on open-source cloud computing solutions. In *Brazilian Symposium on Computer Networks and Distributed Systems*, pages 3–16, 2010.

[32] Andy Edmonds, Thijs Metsch, Alexander Papaspyrou, and Alexis Richardson. Open cloud computing interface: Open community leading cloud standards. *ERCIM News*, 2010(83):23–24, 2010.

[33] Víctor Méndez Muñoz, Víctor Fernández Albor, Ricardo Graciani Diaz, Adriàn Casajús Ramo, Tomás Fernández Pena, Gonzalo Merino Arévalo, and Juan José Saborido Silva. The integration of cloudstack and occi/opennebula with dirac. In *Journal of Physics: Conference Series*, volume 396, page 032075. IOP Publishing, 2012.

[34] Andy Edmonds, Thijs Metsch, Alexander Papaspyrou, and Alexis Richardson. Toward an open cloud standard. *IEEE Internet Computing*, 16(4):15–25, 2012.

[35] Dana Petcu, Ciprian Craciun, and Massimiliano Rak. Towards a cross platform cloud api. In *1st International Conference on Cloud Computing and Services Science*, pages 166–169, 2011.

[36] Azzedine Boukerche and Luciano Bononi. Simulation and modeling of wireless, mobile, and ad hoc networks. *Mobile ad hoc networking*, pages 373–409, 2004.

[37] Azzedine Boukerche, Sajal K Das, Alessandro Fabbri, and Oktay Yildiz. Exploiting model independence for parallel pcs network simulation. In *Parallel and Distributed Simulation, 1999. Proceedings. Thirteenth Workshop on*, pages 166–173. IEEE, 1999.

[38] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Transactions on Vehicular Technology*, 60(2):656–663, 2011.

[39] Azzedine Boukerche, Anahit Martirosyan, and Richard Pazzi. An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks. *Mobile Networks and Applications*, 13(6):614–626, 2008.

[40] Leandro A Villas, Azzedine Boukerche, Horacio ABF De Oliveira, Regina B De Araujo, and Antonio AF Loureiro. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, 12:69–85, 2014.

[41] Steffen Strassburger, Thomas Schulze, and Richard Fujimoto. Future trends in distributed simulation and distributed virtual environments: results of a peer study. In *Proceedings of the 40th Conference on Winter Simulation*, pages 777–785. Winter Simulation Conference, 2008.

[42] Bo Hu Li, Xudong Chai, Lin Zhang, Baocun Hou, Ting Yu Lin, Chen Yang, Yingying Xiao, Chi Xing, Zhihui Zhang, Yabin Zhang, et al. New advances

of the research on cloud simulation. In *Advanced methods, techniques, and applications in modeling and simulation*, pages 144–163. Springer, 2012.

[43] Amine Ghrab, Sabri Skhiri, Hervé Kœner, and Guy Leduc. Towards a standards-based cloud service manager. 2013.

[44] Thijs Metsch, Andy Edmonds, and Victor Bayon. Using cloud standards for interoperability of cloud frameworks. *A technical RESERVOIR report*, 2010.

[45] S Crosby, R Doyle, M Gering, M Gionfriddo, S Grarup, S Hand, M Hapner, D Hiltgen, et al. Open virtualization format specification. *vol. DSP0243*, 1(0), 2010.

[46] Athanasios Bamis, Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks. *Computer Networks*, 52(1):130–154, 2008.

[47] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A distributed algorithm for dynamic channel allocation. *Mobile Networks and Applications*, 7(2):115–126, 2002.

[48] Zhenxia Zhang, Azzedine Boukerche, and Richard Pazzi. A novel multi-hop clustering scheme for vehicular ad-hoc networks. In *Proceedings of the 9th ACM international symposium on Mobility management and wireless access*, pages 19–26. ACM, 2011.

[49] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *IEEE transactions on Parallel and Distributed Systems*, 13(9):911–923, 2002.

[50] Azzedine Boukerche, Xin Fei, and Regina B Araujo. An optimal coverage-preserving scheme for wireless sensor networks based on local information exchange. *Computer Communications*, 30(14):2708–2720, 2007.

[51] Wenying Zeng, Yuelong Zhao, Kairi Ou, and Wei Song. Research on cloud storage architecture and key technologies. In *Proceedings of the 2nd International*

Conference on Interaction Sciences: Information Technology, Culture and Human, pages 1044–1048. ACM, 2009.

[52] Wolfgang Theilmann, Ramin Yahyapour, and Joe Butler. Multi-level sla management for service-oriented infrastructures. In *European Conference on a Service-Based Internet*, pages 324–335. Springer, 2008.

[53] Lizhe Wang, Gregor Von Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud computing: a perspective study. *New Generation Computing*, 28(2):137–146, 2010.

[54] Asad Waqar Malik, Alfred J Park, and Richard M Fujimoto. An optimistic parallel simulation protocol for cloud computing environments. *SCS M&S Magazine*, 4:1–9, 2010.

[55] Bo Hu Li, Xudong Chai, Baocun Hou, Wenhai Zhu, Yabin Zhang, Zhen Tang, Changfeng Song, and Suchuan Mu. Cloud simulation platform. In *Proceedings of the 2009 Grand Challenges in Modeling & Simulation Conference*, pages 303–307. Society for Modeling & Simulation International, 2009.

[56] Elie El Ajaltouni, Azzedine Boukerche, and Ming Zhang. An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure. In *Distributed Simulation and Real-Time Applications, 2008. DS-RT 2008. 12th IEEE/ACM International Symposium on*, pages 61–68. IEEE, 2008.

[57] Horacio ABF Oliveira, Eduardo F Nakamura, Antonio AF Loureiro, and Azzedine Boukerche. Error analysis of localization systems for sensor networks. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 71–78. ACM, 2005.

[58] Azzedine Boukerche and Robson Eduardo De Grande. Dynamic load balancing using grid services for hla-based simulations on large-scale distributed systems. In *Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, pages 175–183. IEEE Computer Society, 2009.

[59] Azzedine Boukerche, Anis Zarrad, and Regina Araujo. A cross-layer approach-based gnutella for collaborative virtual environments over mobile ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(7):911–924, 2010.

[60] Azzedine Boukerche and Damla Turgut. Secure time synchronization protocols for wireless sensor networks. *IEEE Wireless Communications*, 14(5):64–69, 2007.

[61] Xiaocheng Liu, Xiaogang Qiu, Bin Chen, and Kedi Huang. Cloud-based simulation: The state-of-the-art computer simulation paradigm. In *Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation*, pages 71–74. IEEE Computer Society, 2012.

[62] Xiaocheng Liu, Qiang He, Xiaogang Qiu, Bin Chen, and Kedi Huang. Cloud-based computer simulation: Towards planting existing simulation software into the cloud. *Simulation Modelling Practice and Theory*, 26:135–150, 2012.

[63] Kurt Vanmechelen, Silas De Munck, and Jan Broeckhove. Conservative distributed discrete-event simulation on the amazon ec2 cloud: An evaluation of time synchronization protocol performance and cost efficiency. *Simulation Modelling Practice and Theory*, 34:126–143, 2013.

[64] Steffen Strassburger, Thomas Schulze, and Richard Fujimoto. Future trends in distributed simulation and distributed virtual environments: results of a peer study. In *Proceedings of the 40th Conference on Winter Simulation*, pages 777–785. Winter Simulation Conference, 2008.

[65] Bo Hu Li, Xudong Chai, Lin Zhang, Baocun Hou, Ting Yu Lin, Chen Yang, Yingying Xiao, Chi Xing, Zhihui Zhang, Yabin Zhang, et al. New advances of the research on cloud simulation. In *Advanced methods, techniques, and applications in modeling and simulation*, pages 144–163. Springer, 2012.

[66] Azzedine Boukerche, Nathan J McGraw, Caron Dzermajko, and Kaiyuan Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *38th Annual Simulation Symposium*, pages 259–266. IEEE, 2005.

[67] Kaouther Abrougui, Azzedine Boukerche, and Richard Werner Nelem Pazzi. Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):717–735, 2011.

[68] Leandro Villas, Azzedine Boukerche, Regina Borges De Araujo, and Antonio AF Loureiro. Highly dynamic routing protocol for data aggregation in sensor networks. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 496–502. IEEE, 2010.

[69] Azzedine Boukerche and Yonglin Ren. A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In *Proceedings of the 5th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 88–95. ACM, 2008.

[70] Azzedine Boukerche and Kaouther Abrougui. An efficient leader election protocol for mobile networks. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 1129–1134. ACM, 2006.

[71] Gabriele DAngelo and Moreno Marzolla. New trends in parallel and distributed simulation: From many-cores to cloud computing. *Simulation Modelling Practice and Theory*, 49:320–335, 2014.

[72] Bo Hu Li, Xudong Chai, Lin Zhang, Baocun Hou, Ting Yu Lin, Chen Yang, Yingying Xiao, Chi Xing, Zhihui Zhang, Yabin Zhang, et al. New advances of the research on cloud simulation. In *Advanced methods, techniques, and applications in modeling and simulation*, pages 144–163. Springer, 2012.

[73] Gabriele D'Angelo. Parallel and distributed simulation from many cores to the public cloud. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 14–23. IEEE, 2011.

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

[75] Shaochong Feng, Yanqiang Di, Zhu Xianguo Meng, et al. Remodeling traditional rti software to be with paas architecture. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 1, pages 511–515. IEEE, 2010.

[76] Azzedine Boukerche, Yan Du, Jing Feng, and Richard Pazzi. A reliable synchronous transport protocol for wireless image sensor networks. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 1083–1089. IEEE, 2008.

[77] Azzedine Boukerche and Carl Tropper. A distributed graph algorithm for the detection of local cycles and knots. *Parallel and Distributed Systems, IEEE Transactions on*, 9(8):748–757, 1998.

[78] Azzedine Boukerche and Sajal K Das. Dynamic load balancing strategies for conservative parallel simulations. In *Parallel and Distributed Simulation, 1997., Proceedings., 11th Workshop on*, pages 20–28. IEEE, 1997.

[79] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. Sdar: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 618–624. IEEE, 2004.

[80] Leandro Aparecido Villas, Azzedine Boukerche, Heitor Soares Ramos, Horacio AB Fernandes de Oliveira, Regina Borges de Araujo, and Antonio Alfredo Ferreira Loureiro. Drina: a lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. *IEEE Transactions on Computers*, 62(4):676–689, 2013.

[81] Silas De Munck, Kurt Vanmechelen, and Jan Broeckhove. Revisiting conservative time synchronization protocols in parallel and distributed simulation. *Concurrency and Computation: Practice and Experience*, 26(2):468–490, 2014.

[82] Bo Hu Li, Xudong Chai, Baocun Hou, Haiyan Yu, Zhihui Du, Yanqiang Di, Jijie Huang, and Changfeng Song. Research and application on cosim (collaborative simulation) grid. *The proceeding of MS-MTSA*, 6, 2006.

[83] Heng He, Ruixuan Li, Xinhua Dong, Zhi Zhang, and Hongmu Han. An efficient and secure cloud-based distributed simulation system. *Appl. Math*, 6(3):729–736, 2012.

[84] Heng He, Li Chen, Pingpeng Yuan, Xin Xu, and Xiaofeng Wang. A security architecture for grid-based distributed simulation platform. In *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*, volume 1, pages 207–212. IEEE, 2008.

[85] D Davis and G Pilz. Cloud infrastructure management interface (cimi) model and rest interface over http. *vol. DSP-0263, May*, 2012.

[86] Yonglin Ren, Richard Werner, Nelem Pazzi, and Azzedine Boukerche. Monitoring patients via a secure and mobile healthcare system. *IEEE Wireless Communications*, 17(1):59–65, 2010.

[87] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. Secure localization algorithms for wireless sensor networks. *IEEE Communications Magazine*, 46(4):96–101, 2008.

[88] Robson E De Grande and Azzedine Boukerche. Dynamic balancing of communication and computation load for hla-based simulations on large-scale distributed systems. *Journal of Parallel and Distributed Computing*, 71(1):40–52, 2011.

[89] Azzedine Boukerche, Jan M Correa, Alba Cristina Magalhaes Melo, and Ricardo P Jacobi. A hardware accelerator for the fast retrieval of dialign biological sequence alignments in linear space. *IEEE Transactions on Computers*, 59(6):808–821, 2010.

[90] Azzedine Boukerche and Robson Eduardo De Grande. Optimized federate migration for large-scale hla-based simulations. In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, pages 227–235. IEEE Computer Society, 2008.

[91] Doug Davis and Gilbert Pilz. Cloud infrastructure management interface (cimi) model and restful http-based protocol: An interface for managing cloud infrastructure. Technical report, Technical Report DSP0263, Distributed Management Task Force, May 2012. URL dmtf. org/sites/default/files/standards/documents/DSP0263_1. 0.0. pdf. Last accessed in July 2014.(Cited on page 53).

[92] Doug Davis and Gilbert Pilz. Cloud infrastructure management interface (cimi) model and restful http-based protocol. Technical report, Technical report, Distributed Management Work Force (DMTF), 2012.

[93] MN Rajaprabha. Libcloud: A private digital library resource cloud. *Networking and Communication Engineering*, 5(2):90–94, 2013.

[94] W Beary. The ruby cloud services library.

[95] Tobias Kurze, Markus Klems, David Bermbach, Alexander Lenk, Stefan Tai, and Marcel Kunze. Cloud federation. *CLOUD COMPUTING*, 2011:32–38, 2011.

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

[97] Robson Eduardo De Grande and Azzedine Boukerche. Dynamic partitioning of distributed virtual simulations for reducing communication load. In *Haptic Audio visual Environments and Games, 2009. HAVE 2009. IEEE International Workshop on*, pages 176–181. IEEE, 2009.

[98] Azzedine Boukerche and Ming Zhang. Towards peer-to-peer based distributed simulations on a grid infrastructure. In *41st Annual Simulation Symposium (ANSS-41 2008)*, pages 212–219. IEEE, 2008.

[99] E Robson and Azzedine Boukerche. Dynamic balancing of communication and computation load for hla-based simulations on large-scale distributed systems. *Journal of Parallel and Distributed Computing*, 71(1):40–52, 2011.

[100] Nicolas Ferry, Alessandro Rossini, Franck Chauvel, Brice Morin, and Arnor Solberg. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In *2013 IEEE Sixth International Conference on Cloud Computing*, pages 887–894. IEEE, 2013.

[101] Dana Petcu. Multi-cloud: expectations and current approaches. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 1–6. ACM, 2013.

[102] R Slobojan. What is next for jclouds? *InfoQ. com, April*, 11, 2011.

[103] W Beary. What is jclouds? *InfoQ. com, April*, 11, 2016.

[104] Steven Thomas Graham and Xiaodong Liu. Critical evaluation on jclouds and cloudify abstract apis against ec2, azure and hp-cloud. In *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pages 510–515. IEEE, 2014.

[105] Delta Cloud. Delta cloud-many clouds. one api. no problem, 2013.

[106] R Slobojan. What is next for jclouds? *InfoQ. com, April*, 11, 2011.

[107] Boris Parák, Zdenek Šustr, Florian Feldhaus, Piotr Kasprzakc, and Maik Srbac. The rocci project–providing cloud interoperability with occi 1.1. In *International Symposium on Grids and Clouds (ISGC)*, volume 23, 2014.

[108] Piyush Harsh, Florian Dudouet, Roberto G Cascella, Yvon Jégou, and Christine Morin. Using open standards for interoperability issues, solutions, and

challenges facing cloud computing. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*, pages 435–440. IEEE, 2012.

[109] Kalyan S Perumalla. Parallel and distributed simulation: traditional techniques and recent advances. In *Proceedings of the 38th conference on Winter simulation*, pages 84–95. Winter Simulation Conference, 2006.

[110] PF Reynolds and Darren Drewry. The grandest challenge in distributed simulation. In *Distributed Simulation and Real-Time Applications, Fifth IEEE International Workshop on, DS-RT 2001.*, pages 36–36. IEEE, 2001.

[111] Pamella Knight, Aaron Corder, Ron Liedel, Jessica Giddens, Ray Drake, Carol Jenkins, and Paul Agarwal. Evaluation of run time infrastructure (rti) implementations. In *Huntsville Simulation Conference*, 2002.

[112] Peter Ryan and Lucien Zalcman. The dis vs. hla debate: What's in it for australia. In *Proceedings of SimTect 2003*. Citeseer, 2003.

[113] Anders Persson. Migration and load balancing in hla based distributed simulations. *Master of Science Thesis Stockholm*, 2005.

[114] Frederick Kuhl, Richard Weatherly, and Judith Dahmann. *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR, 1999.

[115] Moon Gi Seok, Dae Jin Park, Geun Rae Cho, and Tag Gon Kim. Framework for simulation of the verilog/spice mixed model: Interoperation of verilog and spice simulators using hla/rti for model reusability. In *2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6. IEEE, 2014.

[116] Alfredo Garro and Alberto Falcone. On the integration of hla and fmi for supporting interoperability and reusability in distributed simulation. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pages 9–16. Society for Computer Simulation International, 2015.

[117] B Möller. The hla tutorial v1. 0. *Pitch Technologies, Sweden*, 2013.

[118] BP Zeigler, H Sarjoughian, and T Lake. Creating simulations in hla/rti using the devs modeling framework. *DEVS/HLA Tutorial*, 1, 1998.

[119] Benoit Bréholée and Pierre Siron. Certi: Evolutions of the onera rti prototype. In *Fall Simulation Interoperability Workshop*, 2002.

[120] Bruno d'Ausbourg, J-L Bussenot, and Pierre Siron. Perfosim: A performance evaluation tool for hla distributed simulations. In *Distributed Simulation and Real-Time Applications, 2002. Proceedings. Sixth IEEE International Workshop on*, pages 23–30. IEEE, 2002.

[121] Eric Noulard, Jean-Yves Rousselot, and Pierre Siron. Certi, an open source rti, why and how. In *Spring Simulation Interoperability Workshop*, pages 23–27, 2009.

[122] Jean-Baptiste Chaudron, Martin Adelantado, Eric Noulard, and Pierre Siron. Hla high performance and real-time simulation studies with certi. 2011.

[123] Bruno d'Ausbourg, Pierre Siron, Eric Noulard, and Pierre Siron. Running real time distributed simulations under linux and certi. In *Proceedings of the 2008 Summer Computer Simulation Conference*, page 43. Society for Modeling & Simulation International, 2008.