An HLA Based Framework for Simulation of Geographically Distributed Data Centers

By
Bukhtawar Elahi
2016-NUST-MS-IT-00000170964

Supervisor
Dr. Muazzam Ali Khan Khattak
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science (SEECS),
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(May 2019)

Thesis Acceptance Certificate

It is certified that final copy of MS/MPhil thesis written by **Ms. Bukhtawar Elahi**, (Registration No. **00000170964**), of School of Electrical Engineering and Computer Science (SEECS) has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Advisor: **Dr. Muazzam Ali Khan Khattak**

Signature: _____
Date: _____

Signature (HOD): _____
Date: _____

Signature (Dean/Principal): _____
Date: _____

Approval

It is certified that the contents and form of the thesis entitled "**An HLA Based Framework for Simulation of Geographically Distributed Data Centers**" submitted by **Bukhtawar Elahi** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Muazzam Ali Khan Khattak**

Signature: _____
Date: _____

Committee Member 1: **Dr. Asad Waqar Malik**

Signature: _____
Date: _____

Committee Member 2: **Dr. Muhammad Shahzad**

Signature: _____
Date: _____

Committee Member 3: **Dr. Safdar Abbas Khan**

Signature: _____
Date: _____

Abstract

Usually, the existing simulators provide limited scalability. Therefore, researchers can simulate their protocols and algorithms to a certain extent. In cloud simulators such as CloudNetSim++ and CloudSim are two most commonly used simulators are designed for geographically distributed data centers; however, the simulators cannot support execution on distributed systems. Thus, scalability is also an issue for such simulators which are designed on distributed scalability model. In this thesis, we have proposed a distributed simulation framework that allows simulator instances to execute on a different system. Thus, the framework provides the scalability across simulators. The proposed framework is based on High-Level Architecture, thus, provide interoperability across simulators, tool, and technologies.

Dedicated to my beloved parents and adored siblings, Their tremendous support and cooperation led me to this wonderful accomplishment.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Bukhtawar Elahi**

Signature: _____

Acknowledgment

All Praises to Almighty ALLAH, Who is the source of all knowledge. Who guides us from darkness to the light and helps us in difficulties. Who enabled me to complete this work and bestowed the wisdom and strength to make material contribution to already existing ocean of knowledge.

I would like to express my heartfelt gratitude to Dr. Asad Waqar Malik for his encouragement, guidance, and confidence on my capability to accomplish this research voyage under his profoundly professional assistance.

I am greatly obliged to my family, for their unconditional love, endless support, encouragement and prayers in all my endeavours.

I would also like to pay special thanks to my committee members for their support and cooperation. Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Table of Contents

List of Tables ix

List of Figures x

List of Abbreviations xi

List of Symbols xii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem Definition . . . 2
- 1.3 Objectives and Research Goals . . . 2
- 1.4 Thesis Organization . . . 3

2 Background Information 5
- 2.1 Cloud Computing . . . 5
- 2.2 CloudSim . . . 6
- 2.3 High-level architecture . . . 7
  - 2.3.1 Run-time Infrastructure . . . 9

3 Literature Review 11
- 3.1 Cloud Simulation Frameworks . . . 11
- 3.2 Distributed Simulation Frameworks Based on HLA . . . 17

4 Methodology 19
- 4.1 System Model . . . 19
- 4.2 Implementation . . . 20
  - 4.2.1 Publish/Subscribe Design . . . 20
  - 4.2.2 Communication Among Federates . . . 23
  - 4.2.3 Framework Functionality . . . 24
  - 4.2.4 Framework Modules . . . 24
    - 4.2.4.1 Broker . . . 25

4.2.4.2 Data center 26
4.2.4.3 Cloud Information Service 27

5 Results **30**
5.1 Service Time 30
5.2 Queuing Delay 32
5.3 Network Delay 33
5.4 Resource Usage 35
5.5 Summary and Discussion 36

6 Future Work & Conclusion **41**
6.1 Future Work 41
6.2 Conclusion 41

References **42**

List of Tables

3.1 Comparison between existing Cloud Simulators 17

5.1 Simulation parameters and system specification 31

List of Figures

1.1 Thesis Organization 3

2.1 Layered CloudSim architecture. 7
2.2 HLA-RTI conceptual diagram 9
2.3 Communication between Federate and RTI. 10

4.1 Architecture Diagram 22
4.2 Publish and Subscribe(P/S) pattern in Framework 23
4.3 Federation simulation flow. 24
4.4 Broker, data center, and CIS interaction through HLA-RTI. . 26

5.1 Average service time of cloudlets 32
5.2 Average queuing delay of cloudlets 33
5.3 Network Delay. 34
5.4 CPU usage at broker federate 35
5.5 CPU usage at data center federate. 36
5.6 CPU usage at CloudSim. 37
5.7 Memory usage at broker federate. 38
5.8 Memory usage at datacenter federate. 39
5.9 Memory usage at CloudSim. 40

List of Abbreviations

BIM	Broker Interaction Mnager
BW	Bandwidth
CIS	Cloud Information Service
CRC	Central RTI Component
DC	Data center
DcIM	Data center Interaction Manager
LRC	Local RTI Component
RTI	Run Time Infrastructure
MI	Million Instructions
MIPS	Million Instructions Per Second
VM	Virtual Machine

List of Symbols

α	Task Computation Time
β	Network Delay
B	Broker
C	Cloudlet Set
γ	Step Size
ϕ	Data center
ρ	Queuing delay
T_{exec}	Execution Time
T_{total}	Total Time

Chapter 1

Introduction

This chapter of thesis elaborates the overview of basic concepts, significance and history of research work. This chapter is directed to be the road-map for our thesis and briefly highlights the further organization and structure of the thesis. The brief history of cloud simulation frameworks is explained here. The chapter also explains the main motivation for carrying out the research work. We have tried to give a compendium idea about the vital contributions, scope of the work and key objectives. Eventually the chapter highlights the goals of each following chapter to represent the overall thesis organization.

1.1 Motivation

Cloud computing has been proved as the best solution for large scale distributed resource management. Cloud service providers and the academia are trying their best to yield algorithms, frameworks, and protocols to utilize the resources in an efficient way [1]. A number of simulation frameworks have been developed particularly for cloud which includes Amazon AWS[1], Google cloud platform[2], Oracle cloud [3], IBM cloud solutions[4] and many more [2].

Cloud Computing provides services which are subscription based services, There are three types of cloud services; Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS). With IaaS, users can rent IT infrastructure servers on a pay-as-you-go basis. IT infrastructure includes storage, networks ,virtual machines (VMs). Clouds can be classified

[1]https://aws.amazon.com/
[2]https://cloud.google.com/
[3]https://cloud.oracle.com
[4]https://www.ibm.com/cloud/

as Public, Community, Private, and Hybrid.

In the most recent years diverse ventures supported by the administration offices addresses the issue of demonstrating and reenactment of complex basic frameworks.Complex basic frameworks can not be considered as independent frameworks, but rather, because of the framework inter dependencies, must be considered as a huge complex framework made out of other connected and dependent frameworks. One of the principle approaches used to mimic complex basic frameworks is to create federation of already present frameworks [3] [4]. So the main thing in the creation of simulated federations is to create a network simulation that is High Level Architecture (HLA) compliant.

HLA is a generalize architecture that aims at providing ease for the development and execution of large scale simulation applications and provides a design through which these simulators can interconnect and intercommunicate with each other thus provides re-usability and interoperability [5].

1.2 Problem Definition

Simulators for geographically distributed data centers do not support execution on distributed systems; scalability is also an issue for such simulators which are designed on distributed scalability model. Simulating a large scale simulation on a single system is difficult and is a limitation. Existing simulators run on a single system and do not provide support connectivity to other simulators.

1.3 Objectives and Research Goals

The goal of this research work is to provide and implement a scalable and interoperable simulation framework. In cloudSim the simulation is run on a single system but in our framework datacenters and broker can run on different machines. Our proposed framework provides extensive environment to simulate cloud applications. This framework allows users to perform simulations on different machines. Thus proposed framework provides scalability and interoperability across simulators. This framework is built on top of CloudSim and HLA has been used for communication among the broker and data center. HLA std 1516 has been used for providing simulation scalability and interoperability. HLA has been used for communicating among these distributed simulator instances. We have also designed a communication layer that receives and sends the HLA run-time infrastructure messages from and

to HLA and translates it into respective request for broker and data center. At the end a simulator is presented/designed that provides scalability across the systems and provides interoperability across different simulators.

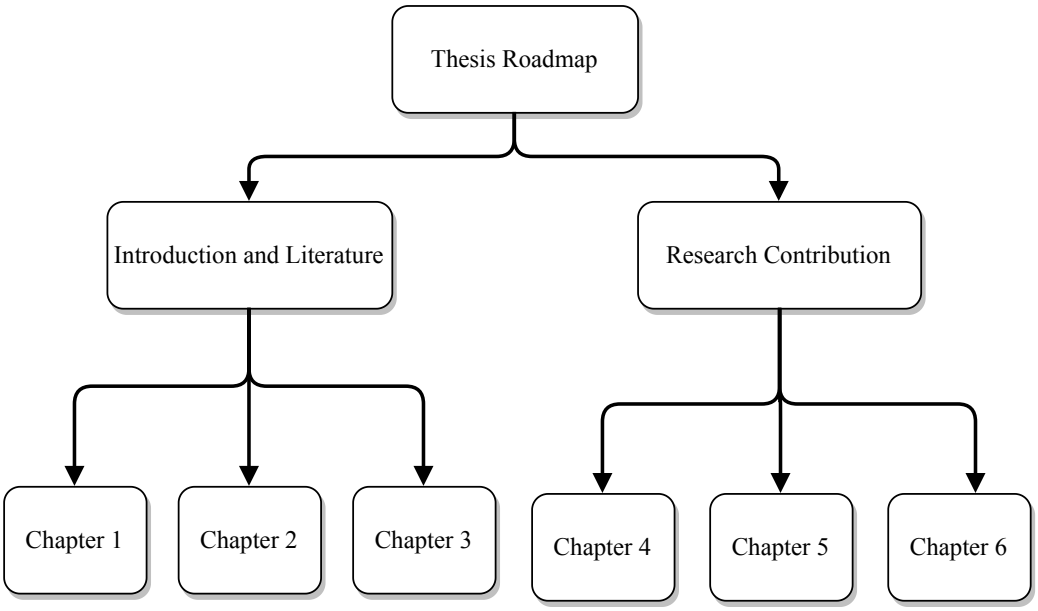

Figure 1.1: Thesis Organization

1.4 Thesis Organization

The presented thesis has been organized into different chapters in which each chapter gives certain aspects of our research, depicted in figure . Following is the brief description of all the chapters that are included in the thesis. **Chapter 1 - Introduction** elaborates the main motivation be- hind the presented thesis. Furthermore, the chapter gives details about aims, scope and major contributions of the thesis.

Chapter 2 - Background Information covers the covers the detailed background information about the CloudSim and High-level architecture (HLA).

Chapter 3- Literature Review gives details of literature survey which has been conducted throughout the research phase.

Chapter 4 - Methodology explains the research approach and methodology followed for the implementation of architecture proposed in this thesis.

Chapter 5 - Results Discussion explains the details of experimental setup, and discussion about the results obtained.

Chapter 6 - Future Work & Conclusion concludes the thesis and highlights the potential future prospects of our research work..

Chapter 2

Background Information

This chapter covers a brief overview of the background knowledge related to this research work. Almost all related theoretical and conceptual points have been covered. First of all, cloud computing has been briefly explained in section 2.1. Which is followed by a brief overview of of CloudSim architecture in section 2.2. And lastly the underneath High-level architecture has been briefly explained in section 2.3.

2.1 Cloud Computing

Cloud computing is a sort of information processing that depends on shared processing resources instead of having localized server machines or individual gadgets to deal with applications. So in cloud computing instead of storing, processing and accessing data on computer's hard drive its over the internet.

Cloud computing is one of the major approach for delivering of delivering reliable, secure, risk-tolerant and scalable computational services. Subsequently convenient, repeatable, and controllable techniques for performance measurement of new cloud applications and strategies are required before their before actual deployment is done. Cloud computing provides many advantages to researchers. Instead of using real test-beds organizations can simulate their large scale applications before the actual deployment, they can also simulate different types of models/devices. Cloud computing has been proved as the best solution for large scale distributed resource management.

There are some constituent elements that are required for cloud computing and are referred to as cloud computing architecture components.These components are basically front-end client, back-end and servers/storage platforms, a cloud based transportation and a network.Cloud Computing provides services to the clients as subscription based that includes infrastructure-

as-a Service (IaaS), platform-as-a service (PaaS), software-as-a-service (SaaS).

2.2 CloudSim

Buyya et al. Proposed CloudSim which is a generalized framework and can beeasily extended. The framework allows modelling and simulation of various application services and cloud computing infrastructures [6]. This was the project of CLOUDS (a cloud computing and simulation lab) at UOM (University of melbourne, Australia). Current release of CloudSim Toolkit 3.0 was released on Jan 13, 2012.

CloudSim is extremely generalized framework that supports several functionalities of cloud computing infrastructures. CloudSim has support for core functionalities as creation of CloudSim entities (Broker, Datacenter, CIS etc), processing and queuing of events in datacenters, correspondence among those entities and the administration and handling of the clock in simulation. CloudSim incorporate essential classes for determining data-cenetrs, hosts, virtual machines, applications, clients, computational assets, and policies and approaches for scheduling and provisioning of these resources. Figure 2.1 shows the layered architecture diagram of CloudSim.

- CloudSim supports modeling and simulation of broker, data centers and multiple virtualized hosts.
- The provisioning policies in CloudSim are generally implemented and can be easily extend as needed by the user and new algorithms can also be implemented and tested easily.
- Provisioning polocy for allocating host resources to VMs can be customized easily.
- Entities can be added dynamically in CloudSim.
- Users can also implement their own host to VM allocation policies for allocation host resources to VM.
- Different data center network topologies can also be implemented in the CloudSim.

CloudSim does not allow dynamic creation of Cloudlets at runtime; Once the simulation is start it does not allow creating new cloudlets and submit it to the broker without requiring creation of Datacenter Brokers at runtime, also it does not allow dynamic creation of virtual machines (creation of VM's) on-demand according to the arrived cloudlets. CloudSim also has no Graphical User Interface (GUI).

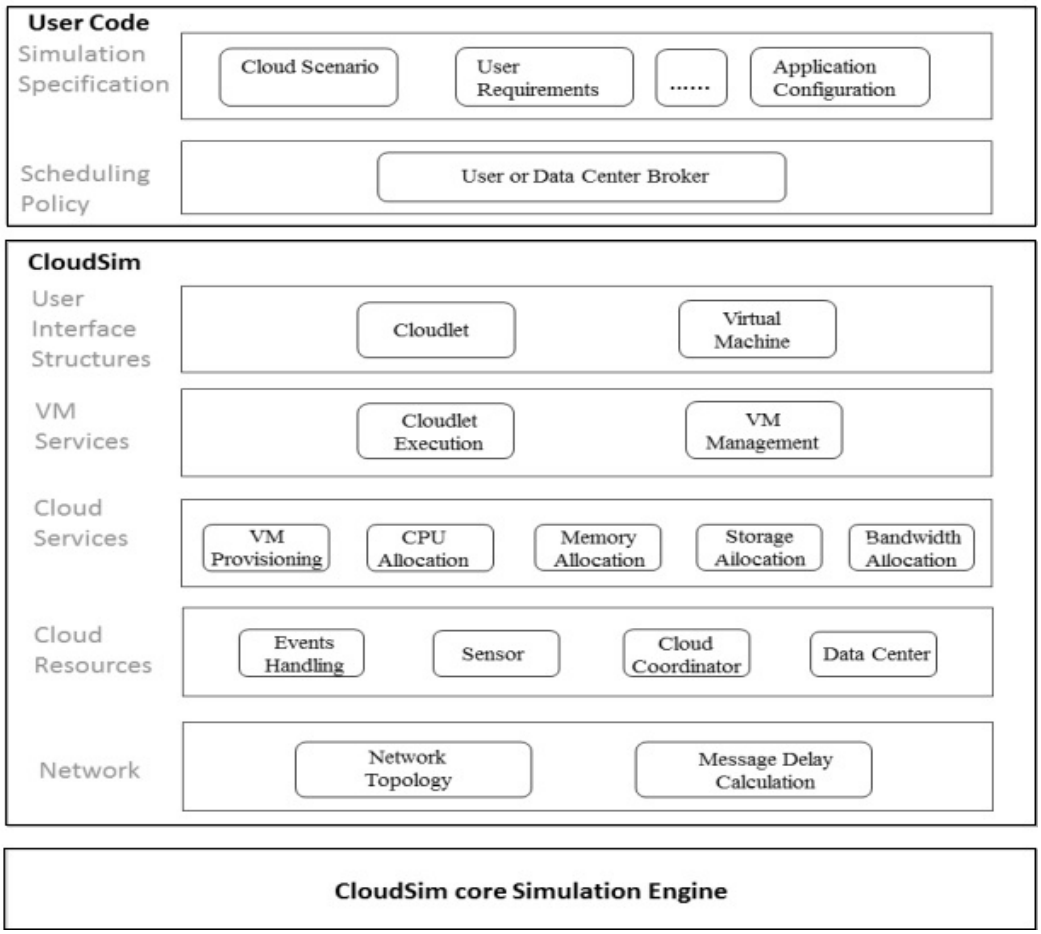

Figure 2.1: Layered CloudSim architecture.

2.3 High-level architecture

High level architecture is a generalize architecture that aims at providing ease for the development and execution of large scale distributed simulation applications and provides a design through which these distributed simulation applications can interconnect and intercommunicate with each other thus provides re-usability and interoperability. HLA 1516 standard - known as HLA Evolved (IEEE 1516-2010) has been used to provide interoperability across simulators.

In simulation, **run-time infrastructure (RTI)** is a middleware that is fundamental component of HLA. Its a software layer provides services to the federates to coordinate and interact with other federates. A **Federate** is a process identified by the single interface that is capable of participating in a

simulation through the runtime infrastructure (RTI). This process can be a library, or an entire simulator. It's identified by the single interface [7] [8] [9]. Multiple federates that are communicating with each other through the RTI comprise the federation.

HLA standard defines federation rules, HLA interface specification (IS), object model template [10]. Federation rules must be followed by simulation federates during federation execution in order to achieve proper functioning and interaction among federates [11]. Object Model Template presents the conventional method for specifying HLA data model information. All the communication among HLA federates and the information that will be received and produced by the federates participating in federation execution will be according to OMT. [11].

OMT consists of federation object model (FOM) and simulation object model documents. FOM outlines all shared data (objects and their attributes and interactions among them) for the whole federation. FOM is one per federation. SOM describes the salient characteristics of federate, presents shared object, attributes and interactions which can be used externally. SOM is used for a single federate (One per federate) and focuses on the federate's internal operation.

How HLA based federates interact with the RTI is defined by Interface Specification. There are six different level service groups in which interface specification is divided. These service groups are : Federation management, Declaration management, Object management, Ownership management, Time management, Data Distribution management, Support services.

Core functionalities such as creation and execution of a federation and its destruction are handled by federation management. It also provides services to federates to to join a specific federation, achieve synchronization among federates and save their states during federation execution and also resign from it. Services for publishing and subscription of object classes, interaction classes and control updates are provided by Declaration management. Ownership management provides services related to transferring object attribute ownership. *Time management* provides services to federates related to time handling during federation execution. To efficiently distribute data and to route data among federates during federation execution is responsibility of *Data distribution management*. *Support Services* is responsible for providing services related to RTI start up and shutdown, setting advisory switches, manipulating regions. It also provides services related to name-to-handle change and handle-to-name change.

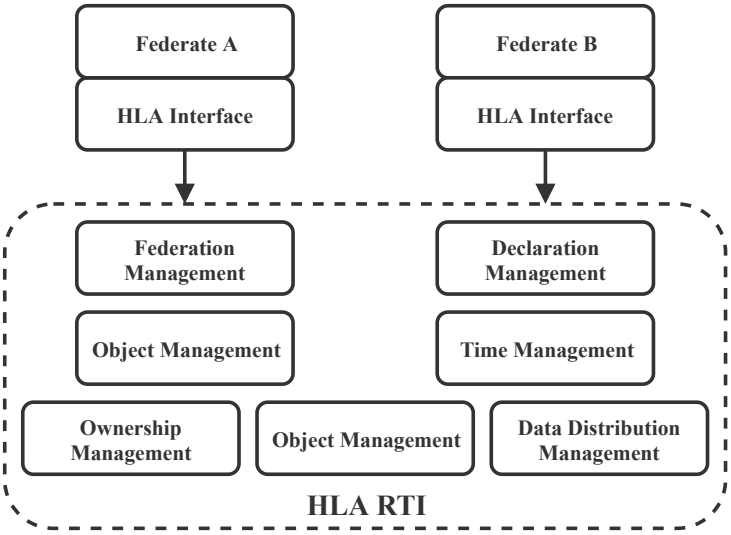

Figure 2.2: HLA-RTI conceptual diagram

2.3.1 Run-time Infrastructure

Run-time Infrastructure (RTI) act as a middle ware between broker, data center and CIS federate. Provides services to the federates for communicating with other federates through the RTI. In our framework broker, data center and CIS federates can not communicate directly. These federates communicate with each other thorough RTI. RTI has two components central RTI (CRC) and local RTI component (LRC). Each CRC and LRC can reside on a same machine and can communicate directly with each other but federates cannot communicate directly with CRC instead the communication is done through the LRC. LRC provides an interface RTI Ambassador to the federate. Similarly, CRC also cannot communicate directly to the federates, communication is done through LRC only. LRC has Federate Ambassador and RTI Ambassador for sending and receiving of interactions and objects to other federates. So, federates communicate with the CRC through the RTI Ambassador interface provided by the LRC and the communication from CRC to federate is handled by LRC and is translated by Federate Ambassador interface on the federate. Figure 2.3 shows the communication between Federate and HLA RTI.

In order to make our Simulation HLA compliant and to make communication messages to pass through RTI we have to implement LRC so that calls from CRC to LRC can be translated and respected action can be performed. Federates send messages through the RTI Ambassador and the calls or messages response that are received from RTI are handled by the federate

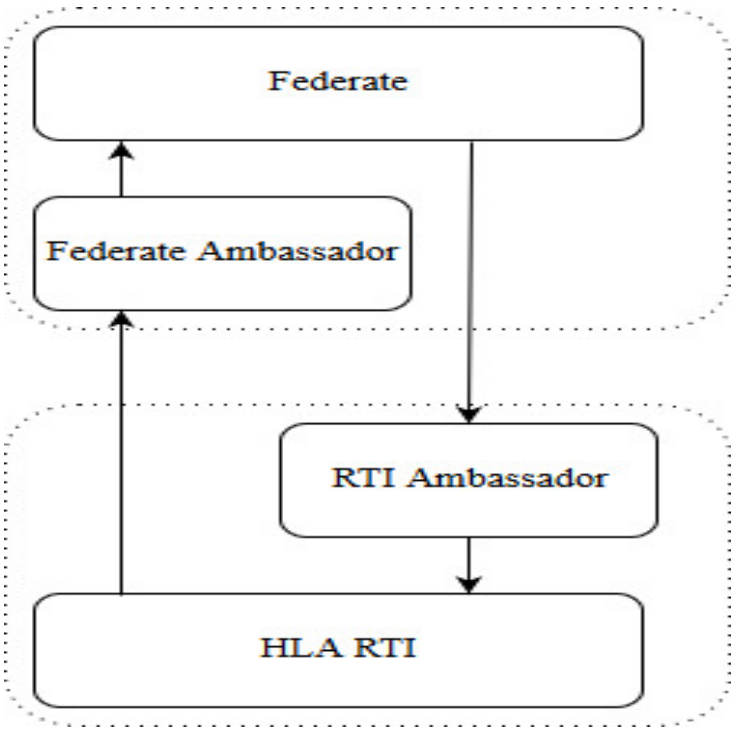

Figure 2.3: Communication between Federate and RTI.

ambassador interface.

Chapter 3

Literature Review

Cloud computing simulation tools provides many advantages to researchers. Researchers can simulate different complex algorithms and procedures before the actual deployment, they can also simulate different types of models/devices. By testing the algorithms on simulation before actual deployment lowers chances of failure and reduces cost, simulation models can be altered and they have a better performance estimation.

This chapter discuss state of art literature review. First of all, current existing cloud simulation frameworks are discussed in section 3.1. Which is followed by an overview of existing cloud simulation frameworks based on HLA in section 3.2.

3.1 Cloud Simulation Frameworks

This section covers an overview of existing cloud simulation frameworks and then comparison between those existing cloud simulators is shown in tabular form.

Buyya et al. [12] proposed CloudAnalyst. CloudAnalyst, is a graphical user interface based simulator extended from CloudSim. CloudAnalyst has extended some properties and abilities of CloudSim. Motivation behind development of CloudAnalyst is to simulate huge scale Cloud applications with the reason for examining the conduct of such applications under different distribution arrangements. This simulator underpins the analysis of informal organization devices in line by the topographical dispersion of clients and server farms. It tends to be connected to decide the conduct of substantial scale Internet applications in the cloud, and furthermore empowers a modeler for circling simulations and to perform a sequence of mockups with minimal varieties in parameters. For the deployment of real-time datacenetrs

CloudAnalyst is viewed as an incredible simulation framework for checking load adjusting, the sending continuous server farms and, cloud group observing and server farm information stream progressively. In CloudAnalyst simulation configuration can be saved as EXtensible Markup Language files and live data results can also be exported in PDF format. It has exceptionally appealing GUI and gigantic adaptability to arrange any land disperse framework, for example, setting equipment parameters (stockpiling, primary memory, transfer speed limit, organize delays and so on.) of a virtual machine or server farm. New broker policy approach can be included effortlessly that manage the clients of any geographical area dependent on tasks accomplished by which Data Center at a specific time. Simulation examination should be possible over and again and the user and system statistics that are generated during the simulation time can be summarizes in tabular and chart form. CloudAnalyst main features are

- Easy-to-use Graphical user interface for setting and reviewing outcomes of a wide range of distributed computing tests.
- CloudAnalyst have modelers that have a high level of command over the analysis by demonstrating elements, for example virtual machines, data-centers, memory, storage capacity and data transfer capacity.
- Simulation scenarios can be saved in the CloudAnalyst and their outcome can be saved as XML documents and even as PDF files.
- Results of simulation can be provided in geographical form as as tables and diagrams, aside from a lot of factual information.

GreenCloud simulator was created by Dzmitry Kliazovich [13]. This test system is utilized to create novel arrangements in observing, , resource allocation, work planning also as communication protocols, improvement and network infrastructure.Energy consumed by the equipment's(processing servers, switches) of data-centers can be finely modeled by the GreenCloud. GreenCloud is a packet-level simulator that has been created by extending NS-2. This simulator distinguishes and recognizes different energy resource utilization components. Simulator is structured with the goal that it can ascertain energy utilization at a specific data-center segment, for example, connect, switch, portal and so on and also correspondence between the packet levels. Further, it offers to know the remaining task at hand circulation in the framework. GreenCloud test system is produced as an augmentation of the Ns2. The outcome of object interaction captured in GreenCloud considerably reduces the simulation time thus improves scalability.

The most recent variant of GreenCloud is 2.1.2. Its main features are as:

- mainly focus on the communication and transmission within a cloud
- It is a packet level simulator
- Main focus of this system is to be monitor the energy consumption in cloud computing networks
- Packet level communication between energy aware cloud data-centers can be computed by GreenCloud
- Simulation of system and network resources within data-centers is supported by the GreenCloud
- Has an easy to understand GUI and is open source

To reduce the resources usage during job selection by considering the task load and correspondence capacity of data-centers [14] [15] scheduler is utilized . One of the disadvantages of this simulator is that it requires a large amount of time for simulation of a model and also consumes huge amount of energy [16]. Because of the large amount of time taken for simulation by this framework it has limited scalability and its adaptability is just limited to small data-centers. This test system is valuable with just the business related to computing energy calculation in the cloud.

Buyya et al. [17] proposed NetworkCloudSim by extending CloudSim. CloudSim and GreenCloud are essentially worked for single server design and wind up deficient for genuine cloud models. In MDCsim no one but applications can impart among one another [16]. To defeat above downside NetworkCloudSim provides correspondence between the application component and different system components. Scheduling is done at two levels in NetworkCloudSim, at Host level and at VM level. Along these lines, VM scheduler is accounts correspondence and computational phases of every application stages. NetworkCloudSim has support for genuine cloud application, for example, web based business, HPC and genuine work process. For simulation of any network protocol withing the cloud architecture NetworkCloudSim can be used. Underlying platform for NetworkCloudSim is CloudSim. Its a open source and implemented in java.It has cost modeling module and support for energy modeling. NetworkCloudSim supports modelling of different network topologies and wide variety application models.

Kimet al. [18] proposed MR-CloudSim an extension of CloudSim that supports processing of work that contain large amount of data that uses MapReduce protocols. Cloud Simulators that are discussed above does not provide support for big data processing techniques. This simulator is suitable

for the processing of the tasks that are related to to big data and use MapReduce Protocol. MR-CloudSim eases the ways to test MapReduce model in a data-centre. MR-CloudSim provides support from energy and cost modeling in distributed cloud computing data-centers.

Rodrigo et al. [19] proposed EMUSIM, an integrated emulation and simulation framework. This framework is a combination of two very important tools CloudSim and Automated Emulation famework (AEF). This was the project of CLOUDS (a cloud computing and simulation lab) at UOM (University of melbourne, Australia). The framework divides the overall work into stages. In first stages is extract application behaviour information, and in second stage it combine the results of first stage to develop a simulation model. This framework is best solution in case when user and researcher has no idea that how it will work in different environment e.g. in parallel and different environments. The deployment cost can be reduced by using EMUSIM because no need to deploy real equipment's. Here network scenario can be simulation and performance can be evaluated prior to equipment deployments. Another main advantage of using this framework is that it is totally open-sourced with latest release 1.3 in Aug 2010. EMUSIM provides not only simulation environment but also emulation environment for cloud computing. The main purpose of EMUSIM framework is to emulate and simulate SaaS (software as a service) in cloud computing by simulating very expensive resources. These resource are very extensive in computing sense and can have very high computing performance. Emulation environment helps to test the performance in a realistic way instead of simply simulating the resources. Users can emulate without renting the cloud resources.

Son et al. [20] designed CloudSimSDN which is a CloudSim expansion for Software Defined Networking (SDN) empowered cloud conditions. This is a lightweight and adaptable simulation framework environment for the purpose of evaluation of network allocation policies. Resource management policies can be evaluated by using CloudSimSDN. It provides support for simulating cloud computing and network entities for measuring the energy consumption in software defined networks and ensuring quality of service. It additionally provides a GUI to to streamlining the simulation configuration. This simulator provides scalability thus different network allocation policies can be analyzed.

Higashino et al. [21] developed CEPSim simulator that is build on top of CloudSim. CEPSim simulator provides support for cloud-based Complex Event Processing (CEP) and Stream Processing (SP) systems related to big data. User queries are transformed into directed acyclic graph.It provides support for modeling of different type of clouds including public, private and hybrid cloud and multi-cloud environments. It also provides support for

simulating client defined queries.

Alves et al. [22] proposed the CMCloudSimulator build on top of CloudSim. This simulator provides support for simulating applications with different deployment configurations. It also provides support for cost modeling and brings about the cost that would be required when it is executed as indicated by the cost model of that service provider. Using extensible Markup Language (XML) these cost models can be defined in the CMCloudSim.

Sá et al. [23] proposed CloudReports for the simulation of distributed cloud computing environments. CloudReports is graphical tool and it provides different aspects for researcher to play role of service providers and users. CloudReports provides simulation for different broker and virtual machine provisioning policies and schedulers. It has been designed for energy aware cloud computing distributed data-centers thus supports power consumption and energy utilization models.

Tian et al. [24] proposed CloudSched emulator. Which is for scheduling of Cloud resources. It can be used for the performance evaluation purposes in order to compare different scheduling procedures.CloudSched takes into consideration different cloud resources such as memory, VM's and network bandwidth for scheduling and provisioning algorithms so CloudSched performs better as compare to CloudAnaylst and CloudSim. CloudSched is a leight weight, high extensible emulator thus provide scalability and its main focus is on IaaS.

CloudExp is a modeling and simulation environment developed by Jararweh et al. This simulator is designed specifically for the simulation of mobile cloud computing environments thus provides different mobile devices mobility scenarios.It has an easy to use GUI , by using this clients can build their own infrastructures. It also provides support for communication cost modeling between clients and cloud [25].

DCSim is developed by Tighe et al. [26] to simulate a virtual data center deployed in Iaas model. DCSim is an event driven simulator that provides support for executing and performing techniques and algorithms related to data-center management [27]. DCSim supports simulation of multiple hosts that can intercommunicate with each other .It also provides support for virtual machine migration between hosts and allows sharing of tasks among VM's. Each host in DCSim has its own resource provisioning policy and own CPU scheduler. Drawback of this simulator is that it ignores the network topology.

iFogSim was proposed by Harshit et al. [28]. It is developed in Java and is an extension to CloudSim. iFogSim proposed the resource management techniques' impact in term of cost, congestion, and latency. There are some limitations in iFogSim. First limitation is that it is java based and core

network characteristics are not supported or ignored.

Ostermann et al. [29] deveoloped GroudSim which is an event-based simulator developed using Java.It was designed to support cloud and grid computing environment. It only requires one simulation thread.It is an event base simulator. GroudSim is mostly used for IaaS but can also be extended for SaaS or PaaS services of cloud.

DynamicCloudSim proposed by Leser et al. [30] is an extension of CloudSim. Due to cloud computing environment heterogeneity some instabilities arise DynamicCloudSim was developed to simulate such kind of instabilities. DynamicCloudSim simulaotr supports cost modeling and communication modeling but it doennot provide support for energy modeling. It can be used to simulate Iaas in cloudComputing.

Nnnez et al. [31] proposed iCanCloud simulation framework that is build on top of OMNeT++ and is designed for large cloud simulations. It also compute the cost of using compute resources. icanCloud supports IaaS cloud service and SaaS(HPC) but it does not support PaaS cloud service, federation policy. It provides modeling of public cloud provider Amazon EC2. iCanCloud supports cost modeling (cost of using compute resources) and communication modeling but it does not provide support for simulating energy modeling, designed for large cloud simulations. It provides support for Amazon EC2 and hypervisor that can be used to compare different policies.

Malik et al. [32] developed CloudNetSim++, a packet-level simulator. This simulator is for distributed data centers. CloudNetSim++ is designed on the top of OMNeT++ and utilized the features of INET framework. It provides support for cost and energy consumption modeling but has limited support for communication modeling. It provides a rich graphic user interface, and communication among different nodes is attained through packets. This framework allows its users to define their own VM migration policies and analyze usage cost.

FogNetSim++ is developed by buid on top of omnet++. This simulator provides a general procedure to perform handover. Users can easily develop and test their resource provisioning and scheduling algorithms using it. It supports measuring of network characteristics like delay, packet loss and also has support for energy modeling [33].

Shiraz et al. [34] developed SmartSim framework used for mobile cloud computing.This is developed to simulate applications for Mobile Cloud Computing running in Smart Mobile Devices (SMDs) .

Sotiriadis et al. [35] proposed SimIC. SimIC is aiming to accomplishing interoperability, adaptability and transporter flexibility while in the meantime presenting the idea of heterogeneity of more than one cloud setups.

Rehman et al. [36] developed secCloudSim that on top of iCanCloud

simulator. Some of the security features (authentication and authorization) which are not supported by iCanCloud are provided by SecCloudsim. However, advanced security features(privacy, integrity and encryption of VMs) are not supported by this simulator.

Li et al. [37] developed DartCSim+ which is an extension of CloudSim. The main feature of DartSim++ is that the developers integrated power and network models in this simulator thus making network and scheduling algorithms power-aware,mechanism for solving the problem of distortion is also added in this. It has easy to use GUI for researchers to test their experiments.

Table 3.1 shows the comparison between existing cloud simulators.

Table 3.1: Comparison between existing Cloud Simulators

Simulation Platform	Underlying Platform	Development Language	Simulator Type	Graphic User Interface	Cloud Service Support
ClouSim	SimJava	Java	Event Based	No	IaaS
CloudAnalyst	CloudSim	Java	Event Based	Yes	Iaas
NetworkCloudSim	CloudSim	Java	Packet Level	No	Iaas
MR-CloudSim	CloudSim	Java	N/A	No	N/A
EMUSIM	CloudSim, Xen	Java	Event Based	Yes	Iaas
CloudSimSDN	CloudSim	Java	Packet Level	Yes	IaaS
CEPSIM	CloudSim	Java	Event Based	Yes	IaaS
CMCloudSimulator	CloudSim	Java	N/A	No	N/A
CloudReports	CloudSim	Java	Event Based	Yes	IaaS, SaaS
CloudSched	-	Java	Event Based	Yes	IaaS
CloudExp	CloudSim	Java	Event Based	Yes	IaaS, SaaS, PaaS
DCSim	-	Java	Event Based	No	IaaS, PaaS
iFogSim	CloudSim	Java	N/A	Yes	N/A
DynamicCloudSim	CloudSim	Java	Event Based	No	IaaS
iCanCloud	Inet, Omnet++	C++	Event Based	Yes	Iaas ,SaaS(HPC)
CloudNetSim++	Inet, Omnet++	C++	Packet Level	Yes	IaaS
FogNetSim++	Inet, Omnet++	C++	Packet Level	Yes	N/A
SmartSim	CloudSim	Java	Event Based	No	IaaS
SimIC	SimJava	Java	Event Based	No	No
secCloudSim	iCanCloud	C++	Event Based	Yes	IaaS, SaaS(HPC)
DartCSim++	CloudSim	Java	Packet Level	Yes	IaaS

3.2 Distributed Simulation Frameworks Based on HLA

This section covers an overview of distributed simulation frameworks based on HLA.

Falcone et al. [38] proposed HLA Development kit Framework that helps in making design and development of HLA federation easier. They proposed a new layer and functionalities are provided through a set of API's. The main benefit of DKF is that it provides portability that allows the researchers to write program once and run anywhere. It is platform independent. It also provides usability. The code generated by using this DKF is easier to main and is more compact. DKF increases the Federate reliability as the core functionalities are managed by the DKF itself. DKF has some

disadvantages.Event Based simulations are not supported by the DKF while it partially supports Synchronization Point mechanism. It also does not provide support to Data Distribution Management (DDM).

Sung et al. [39] proposed a framework for simulation of hybrid systems. Hybrid system is a blend of continuous systems and discrete events. So inter operation between existing hybrid systems simulators is achieved by HLA. They have used adaptor based approach for for inter-operation of frameworks. Framework adaptor comprises of an HLA interface, algorithms applied for synchronization of the different simulators and data conversion. The framework was experimented using water level simulation. This framework has low scalability as it only provides the interoperation for two Models of Computation (MoC) i.e: Discrete Event and Continuous Time and it does not deals with the possibility of expanding the simulation to other MoC, This issue is catered by using Ptolemy instead of Matlab [40].

Guan et al. [41] proposed high level architecture based cloud toolkit. This toolkit enables the modeling and allow simulating the behaviour of Cloud computing infrastructure in which different resource scheduling policies can be implemented and evaluated. They proposed HLA based simulation scheduling scheme in order to enhance simulation efficiency by distributing simulation components. For handling the configuration and maintenance issues resource management model is also proposed .

Gervais et al. [42] proposed an architecture and developed a flight simulator. They have used the HLA for developing and interconnecting flight simulator. This flexible architecture allows to change the version of existing federate by another.

Malik et al. [43] proposed an high level architecture compliant network enabled distributed modeling and simulation infrastructure.Which is designed for simulating large scale distributed systems which in turn can be used to to simulate satellite communication and tracking systems. They used this infrastructure for simulating a radar system consisting of many radars(static radars) acting as federates.

Galli et al. [44]proposed an HLA compliant network simulator that is HLA-Omnet++. This simulator can be used to simulate existing omnet++ network simulation model and new network models can also designed and used.

Lu T et al. [45] proposed the HLA based 3D war-gaming simulation . To support the distributed multi-server domain they have combined the java remote method invocation with HLA run-time infrastructure and also provide multi-player implementation with client-server architecture.

Chapter 4

Methodology

4.1 System Model

The proposed system can support DC federated datacenters, a single federated broker Br. Each data center has h hosts which in turn manage manage the several virtual machines (VMs). Allocation of VM to a host is managed by the VM provisioning controller based on the policy defined by the user. VM provisioning policy that we have implemented for the allocation of VM to a host is on First Com First Serve basis (FCFS). Assigning of CPU core to VM is based on Space-Shared Policy. Policy applied to Cloudlets(tasks) is also Space shared.

By applying the space shared policy, task computation time α of a cloudlet $c \in C$ submitted to a host H and assigned VM, the service time α for the cloudlet can be computed as,

$$\alpha = \frac{s_c}{cap} \tag{4.1}$$

where s_c is the size of the cloudlet (the instructions) that will be needed by the cloudlets to run on a processing machine and *cap* is the computation power of VM. The network delay β_t encountered by the cloudlet from sending end to receiving end is given as,

$$\beta_c = \frac{s_c}{BW} \tag{4.2}$$

where BW is network bandwidth. The average queuing delay ρ of cloudlets is computed as,

$$\rho = \frac{1}{|C|}\left(\frac{\alpha}{2} \cdot \gamma(\gamma - 1) + \sum_{c \in C} \beta_c\right) \tag{4.3}$$

where $|C|$ is size of cloudlet set C and the γ is the customized parameter that represents the steps required to compute the batch of cloudlets, γ is

computed as,

$$\gamma = \frac{\sum_{c \in C} s_c}{\sum_{v \in VMs} cap_v} \tag{4.4}$$

Further, the broker B, CIS and data center Φ communicate with each other to perform the desired functionality. First, B sneds an interacts call to Φ to get the data center features. In response, Φ sends an interaction call in response containing list of hosts and their respective attribures, that is $\Phi = \eta_1 \cup \eta_2 \cup \cdots \eta_k$ where η represents a host. Each host is defined as $\eta_i = (\omega_1, \omega_2, \cdots, \omega_l)$. Here, ω is the characteristics set of each host.

Completion time of all the cloudlets assigned to a Φ_i can be calculated by function $T_{exec}E_f(\Phi)$. $x = |T|$ represents the total number of cloudlets and r represents the VMs per host, it can be written as $1 \leq i \leq x, 1 \leq j \leq k$. Now the objective function mentioned above can be defined as, $\mathrm{T}_{exec}E_f(\Phi_i) = \sum_i \omega_{ik}; E_f(i) = k$ where E_f is the cloudlet mapping function to Φ such that $T_{exec}E_f(i) = j$ represents that the clouldet i is allocated to a VM r at node j.

Total time taken while allocating a task to an available and suitable VM of a host k is represented by T_{total}. T_{total} is computed as, $\mathrm{T}_{total} = \sum_i \sum_k \omega_{ik}; E_f(i) = k$ and $E_f(j) \neq k$ Thus, the total completioncost of all the cloudlets on k hosts located in datacenter Φ_i is computed as, $\mathrm{T}_t = T_{total} + T_{exec}E_f\Phi$

During the cloudlets submission from broker to datacenter the designed framework introduces a realistic network delay which can produce the more realistic results.

4.2 Implementation

This framework is build as an extension to the CloudSim. Our proposed solution aims at integrating homogeneous(CloudSim) simulators connected to the same local area network through RTI. The existing simulators are not scalable and also not able to support interoperability across simulator. Hence, this is the motivation for designing this framework. Due to flexible designed architecture, multiple data centers can be executed on different physical systems. Broker,datacenter and CIS interact with each other through HLA/RTI as shown in Figure 4.1

4.2.1 Publish/Subscribe Design

In the publish and subscribe pattern sending and receiving federates specify to the run-time infrastructure the information they need to receive and

Algorithm 4.1 Federated broker

Input
L_{DCF}: List of data center federates
L_{VM}: List of virtual machines
Q_C: Cloudlet queue
Output
T_c: Cloudlet outcome

```
1: T_c = []                                          ▷ Output buffer
2: while Q_C ≠ {φ} do
3:     c ← dequeue(Q_C)                              ▷ Get cloudlet from queue
4:     ρ ← false
5:     for each x in L_DCF & !ρ do
6:         for each y in L_VM & !ρ do
7:             if vm_xy AVAILABLE then
8:                 send_rti_interaction(c, vm_xy)
9:                 T_c^xy ← recv_rti_interaction()   ▷ Receive cloudlet outcome
10:                ρ ← true
11:            end if
12:        end for
13:    end for
14: end while
15: wait_all()
```

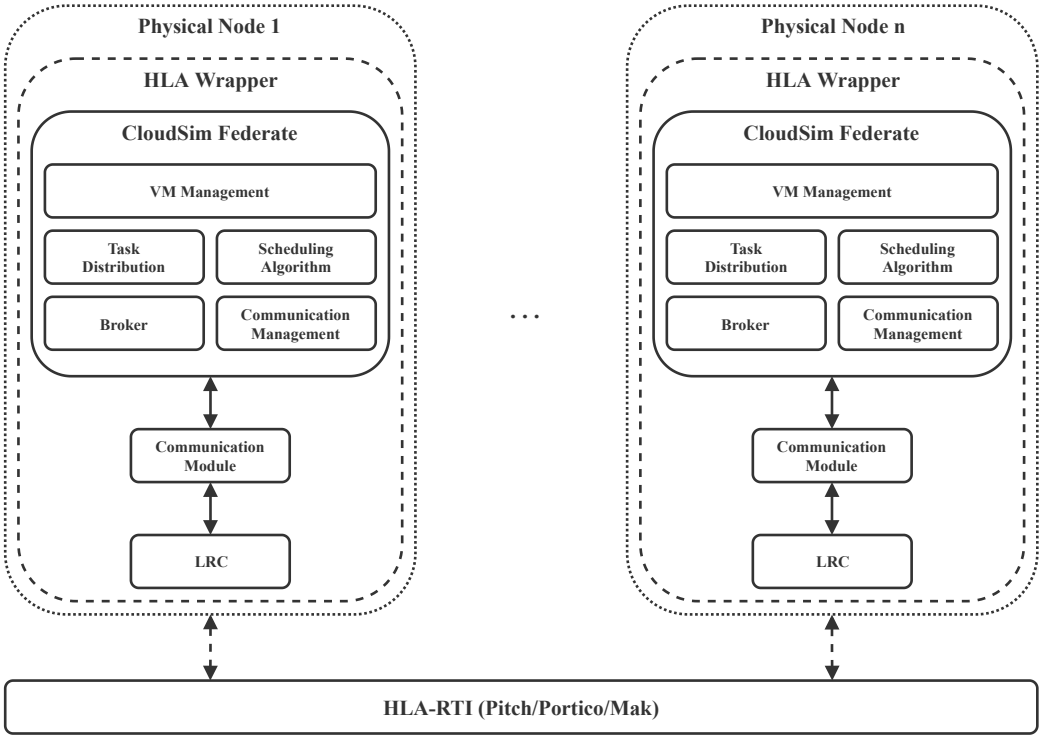

Figure 4.1: Architecture Diagram

what information the are providing to other federates in federation execution. During federation execution process for federates to communicate with other federates, they must Publish and Subscribe their interests which in turn forms a communication model used by the federates. Publishing of an object or interaction implies eagerness to give information that the federate is able send or update. Subscribing certain objects and interactions means the federate declare its interest in receiving certain type of information.

RTI routes data dynamically from the publishing federate to the subscribing federate. At runtime, broker and datacenter federates declare to the RTI the objects and interactions they can ssend (publish) and a set of interaction and data they are willing to receive (subscribe), according to the Federation Execution Details (FED) for HLA 1.3 federations. By following data declaration in FED files, federates can create or register for an object or can send an interaction, which they published. Afterward, the RTI search for the federates who subscribed to that particular information (object or interaction) and then routes the object/interaction to the subscribing federates.

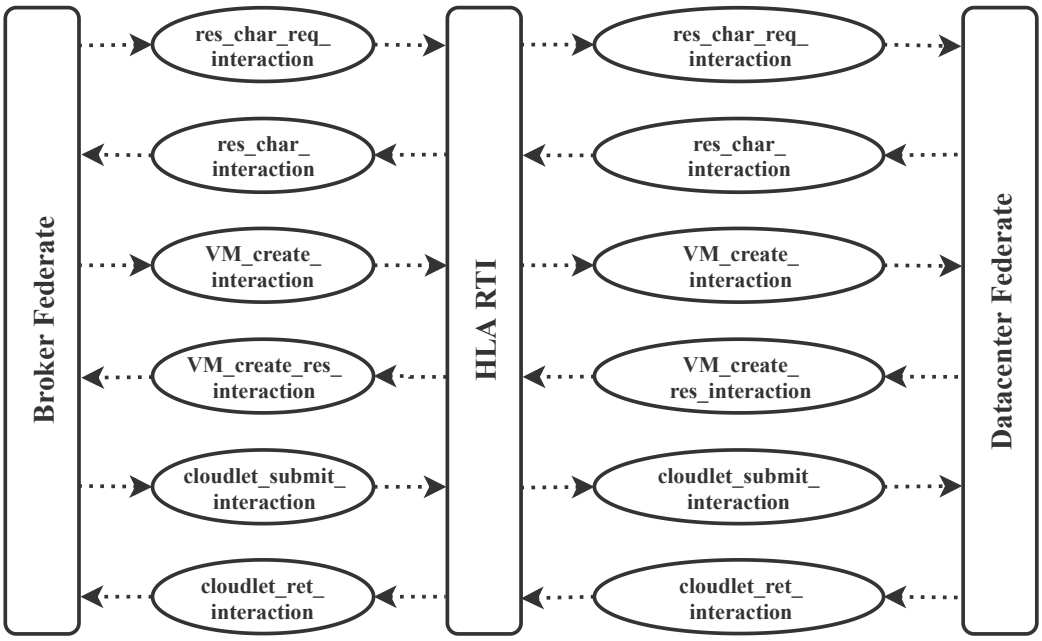

Figure 4.2: Publish and Subscribe(P/S) pattern in Framework

4.2.2 Communication Among Federates

Figure 4.3 shows the flow of communication among federated simulation. At first the broker, data centers and CIS join the federation. After joining of federation; broker, data centers and CIS federates now can communicate with one another through the RTI.

Each DC federate after joining the federation send a request through the RTI to CIS for registering itself to CIS. After registration CIS sends an acknowledgment to the DC; Now the CIS has the list of available DC.

Broker Federate requests to the CIS Federate for the list of available DC. The CIS Federate in response sends a RTI message containing a list of available DC that are registered to the CIS up till now.

Broker sends the request to the DC for its characteristics inquiry. DC sends a response back through RTI to the broker federate containing its characteristics. After knowing the characteristics of the datacenter the broker needs to send the cloudlets (s) to data center by assigning a particular VM to each cloudlet. Data center after processing the cloudlets sends its results back to the broker federate.

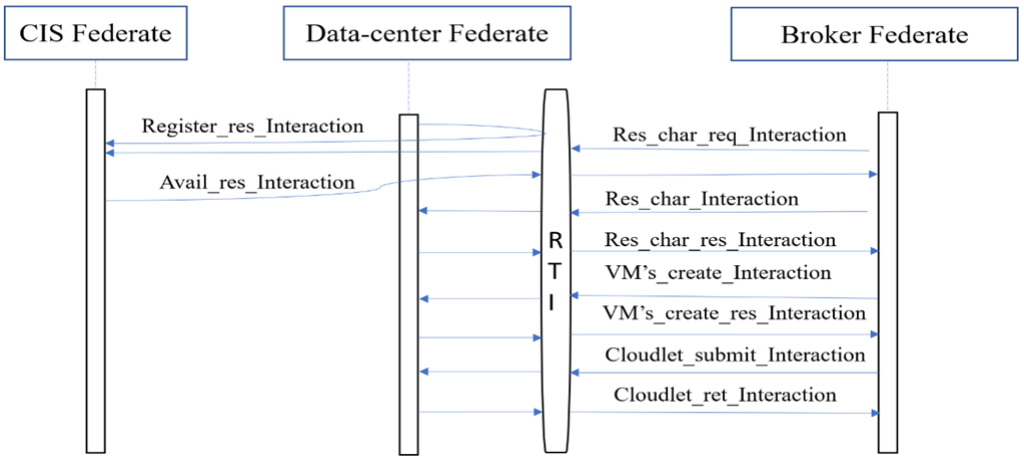

Figure 4.3: Federation simulation flow.

4.2.3 Framework Functionality

Broker, Data center and CIS have a local RTI component associated with them and they are intercommunicating through the central RTI component. Messages and data are sent through the RTI and their response is also received from the RTI. Every resource which is associated for the purpose of execution is residing in the data center. Each data center entity that aims to model a true Data center must encapsulates a list of hosts and a list of storage devices each with defined hardware specifications. Our framework supports server virtualization therefore every host can run multiple VMs. VM allocation policies are defined to assign a VM to a specific host according to that policy. More allocation of resources to virtual machines from host is managed by the policy defined in virtual machine scheduler. The Cloudlets processing is managed by the virtual machines according to the cloudlet schedular policy defined for that particular VM. A Cloudlet represents the task that is submitted by the user to the broker. The Broker act as a middle layer between the user and the data center. The tasks submitted to the broker are then sent to the datacenter by the broker through RTI.

4.2.4 Framework Modules

Main modules in our designed simulator are as follows

4.2.4.1 Broker

Broker is the core module of our simulator. Broker is responsible for communicating with the Cloud Information Service entity. Data center and CIS establish a connection with broker using HLA/RTI, broker is managing a table with all active user nodes. User send task to the broker through publish interaction. Tasks are sent from user to the broker. Broker has broker interaction manager and broker manager module.

Broker Interaction Manager All the messages/interaction that are received and sent from the broker are handled by the broker interaction manager (BIM). Whenever broker needs to send a request or data to data center or CIS it sends the message containing that data to BIM. BIM in turn converts that message to a particular interaction and then sends that interaction to the destination. All the communication of the broker to other modules that are performed through HLA-RTI are handled by this module. The interactions received to the broker are handled by BIM and it sends the received interaction message to the respective handler/module for further processing. BIM Communicate with data centers and handles the interactions published and subscribed by broker manager and VM handler. BIM registers interactions, publishes the interactions that it wants send and subscribe for interaction that interested to receive.

VM handler interacts with the datacenter through BIM to create VM in datacenter. VM handler handles interactions that are related to VM that is create and destroy VM in datacenter. VM hanler also store with it a list of available VMs across all available data centers.

Broker Manager (BM) module sends an interaction to the broker interaction manager (BIM) and interact with a VM manager module of the data center to create available VMs for cloudlets distribution. In the framework data centers are managed by a single broker federate.If needed multiple instances of broker federate can be used. First of all broker manager interacts with the CIS for data centers information and sends a message to the broker interaction manager to get list of available data centers from CIS through resource characteristics request interaction. BIM then sends that interaction to the CIS. After receiving the response interaction from CIS, BIM sends that response interaction containing list of available data center federates to BM. Interaction between different components is shown in Figure 4.4

After that the tasks are submitted to the BM by the task scheduler. Once the cloudlets are submitted to BM, these cloudlets are allocated a VM from a list of available VMs by the VM handler and are forwarded to the respective data center through HLA-RTI by BIM.

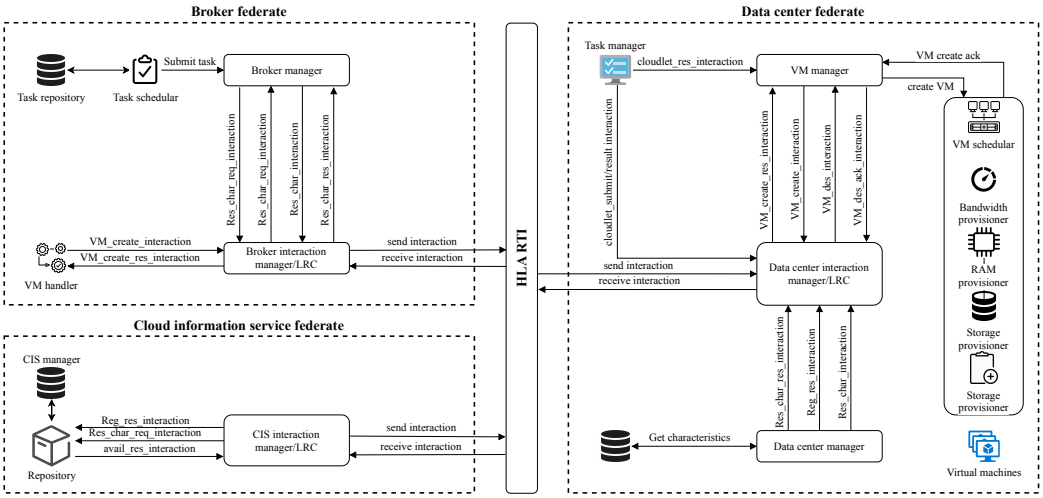

Figure 4.4: Broker, data center, and CIS interaction through HLA-RTI.

4.2.4.2 Data center

Another core module of our simulation framework is data center. Data center comprises of multiple hosts, VMs are allocated to a particular host by data center. Our proposed framework is an extension to CloudSim basic modules and modification have been introduced in them to provide interoperability and scalability across simulators. So now multiple data center instances can be initiated at different physical systems connected through local area network.

Data center interaction Manager All the messages/interaction that are received and sent from the data center to broker or CIS are handled by the data center interaction manager (DcIM). Whenever datacenter needs to send a request/data to broker or CIS it sends the message containing that data to it. Which is then converted into the respective and that interaction is send to the destination. All the communication of the datacenter to other modules is handled by this module. The interactions intended for data center are handled by DcIM and it sends the received interaction message to the respective handler/module for further processing. DcIM Communicate with broker and handles the interactions published and subscribed by data center manager and VM manager. DcIM also registers interactions, publishes the interactions that it wants send and subscribe for interaction that interested to receive.

VM Manager module in datacenter manages the VM related interactions that are to receive create and destroy VM request interaction from data center interaction manager and then create or destroy VM. VM manager as-

sociates a VM to an available host and also handles VMs policy management.

Data center Manager processes the events related to the core datacenter functionalities. It sends the data center characteristics related interaction to the DCIM so that DC characteristic interaction can be sent to BIM from DcIM through HLA RTI. DcM also also define the VM allocation policy for allocating a host a VM.

Task Manager handles the cloudlet related interactions, process the cloudlets that are submitted to the data center, updates the cloudlets that are processed and check for the completion of cloudlets and return their results.

4.2.4.3 Cloud Information Service

Every resource that can be used for the reservation of some particular cloudlet is to be registered with CIS. CIS entity is mainly accountable for resource registration, classification and discovery.

CIS interaction Manger is responsible for managing all the interactions that CIS intends to publish or subscribed for which includes resource registration interaction, request resource interaction and available resources interaction.

CIS manager handles the interactions received by CIS interaction manager and process them, perform the discover services and stores lists of available resources.

Algorithm 4.2 Federated data center

Input
c: cloudlet
Output
T_c: cloudlet outcome

```
1: send_rti_interaction(dc,REG_RESOURCE) ▷ Data center registers with CIS
2: while true do
3:     sender ← recv_rti_interaction()
4:     if sender = φ then break
5:     end if
6:     params ← sender.get_params()          ▷ Get sender information from received interaction
7:     cls ← sender.get_class()          ▷ Get interaction class from received interaction
8:     send_rti_interaction(b, c)
9:     if cls IS CHAR_REQUEST then
10:        char ← itr.get_char()              ▷ Get data center characteristics
11:        send_rti_interaction(itr, REQ_CHAR)          ▷ return data center characteristics
12:    else
13:        if cls IS VM_CREATE then
14:            process_vm_create()    ▷ create VM; allocate PE and memory for new VM on node
15:            send_rti_interaction(VM_CREATE)
16:        else if cls IS CLOUDLET_SUBMIT then
17:            process_cloudlet()
18:            T_c ← send_rti_interaction(CLOUDLET_RET)
19:        else if cls IS DESTROY_VM then
20:            vm_destroy()                    ▷ Process VM destroy request
21:        end if
22:    end if
23: end while
```

Algorithm 4.3 Federated CIS

Input
M_{req}: Request message
Output
L_{CRF}: List of available cloud resources on (data center) federates

1: $L_{CRF} \leftarrow \{\phi\}$
2: $M_{req} \leftarrow \{\phi\}$
3: **while** true **do**
4: sender $\leftarrow$ recv_rti_interaction(M_{req}) ▷ Get sender information from received interaction
5: **if** sender $= \phi$ **then** break
6: **end if**
7: cls $\leftarrow$ sender.get_class() ▷ Get interaction class from received interaction
8: **if** cls **IS** REG_RESOURCE **then**
9: L_{CRF}.add(sender) ▷ Add resource to list of available cloud resources
10: **else**
11: **if** cls **IS** REQ_RESOURCE_LIST **then**
12: $L_{CRF} \leftarrow$ send_rti_interaction(s,L_{CRF}) ▷ Returns list of available cloud resources (data centers)
13: **end if**
14: **end if**
15: **end while**

Chapter 5

Results

This chapter present the simulation results of our proposed framework. The simulation environment consists of a data centre with 1000 hosts. Each host was exhibited to have a CPU core of 1000 MIPS, 2TB of storage and 1 GB of RAM memory. Space-shared provisioning strategy was used for VM's that allow only one VM to be dynamic in a host at particular time. We designed the end-client (through the Broker) to ask for creating and instantiating VMs that had the accompanying limitations: 1000 MB of physical memory, single CPU core. The number of VMs range from 50,100,150 and 200. For our experiment we have taken 1000 to 20,000 task units. Length of tasks/cloudlets is taken randomly from 400 million instructions(MI) to 1200 MI. The simulation parameters and system specifications are mentioned in Table 5.1

5.1 Service Time

Service time is computed as the time the datacenter federate takes to receive the cloudlet and process that cloudlets. After creation of VMs in host by data center, cloudlets are submitted in form of group. In case when we have 50 VMs available the tasks are submitted initially as a group of 50. The number of tasks are varied from 1000 to 30,000 and their average service time is computed. At first, the VMs are kept 50 and then the average service time is calculated for cloudlets from 1000 to 30000. Space shared scheduling policy was followed for VM's so only one VM can run in a host at a given time instance. The number of available virtual machines are then changed from 50 to 100, 150 and 200 and then the average service time is computed for tasks/cloudlets (from 1000 to 30000). It can be clearly noted as the number of available VMs are more the more tasks are executed so average response time

Table 5.1: Simulation parameters and system specification

Parameters	Values
HLA-RTI	Pitch v5.3.2
HLA-RTI simulation model	Conservative approach
CloudSim	v3.0
Simulation time	5hr
Number of time simulation run	10
Number of objects defined in FOM	6
Number of interactions defined in FOM	55
Number of federates	04
Number of physical systems	02
Number of VMs inside data center	200
Number of nodes in a data center	1000
Size of messages	20 bytes
Total number of task generated	30K
CPU	Dual-core 2.3 GHz
Memory	4 GB
Operating System	Linux

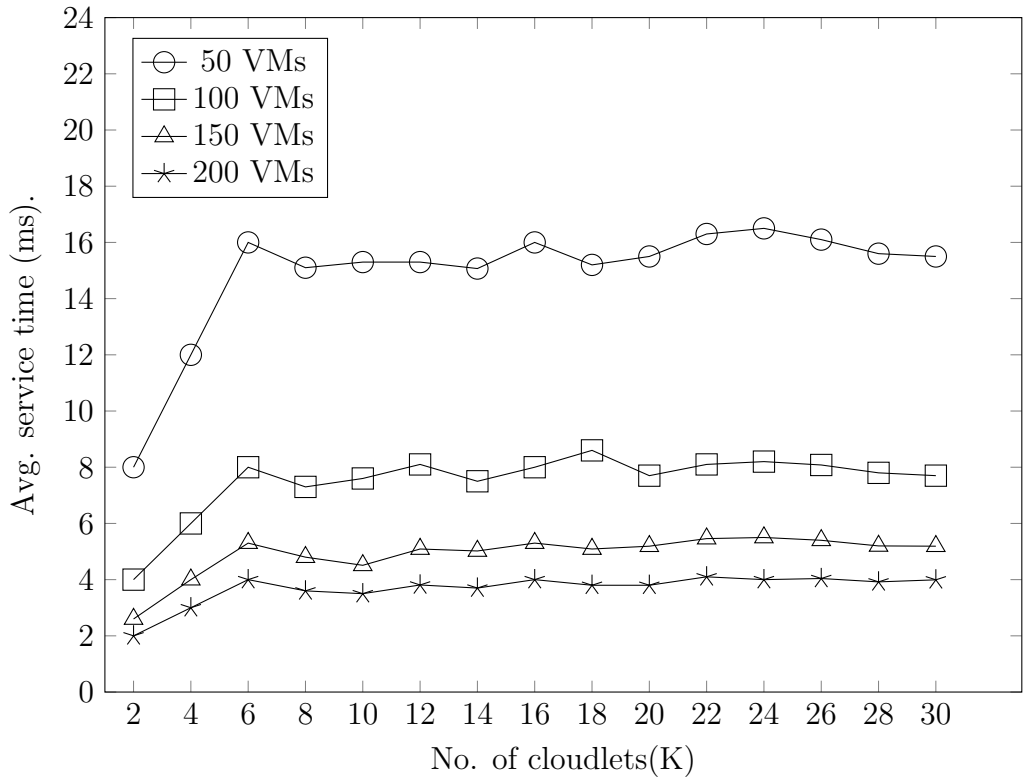

Figure 5.1: Average service time of cloudlets

is improved. As there are more number of VMs available so more cloudlets are executed at once. Figure 5.1 we have calculated the average service time of cloudlets in datacenter while varying the number of virtual machines.

5.2 Queuing Delay

Figure 5.2 shows the average queuing delay of cloudlets at the broker federate side. All the received cloudlets from the task scheduler are kept in a queue by broker. As there are more number of available VMs more tasks/cloudlets can be served in parallel, so the queuing delay of cloudlets decreases, thus decreasing average queuing delay as the number of VMs increases. Average queuing delay is more when we have 50 VMs; the tasks are submitted in a small group of 50 (in case of 50 VMs), the remaining tasks will wait for time until the result of any task/cloudlet is returned the next task in the queue is submitted to the available VM. As we have more VMs the more number of cloudlets can be executed in at same time, so the average queuing delay

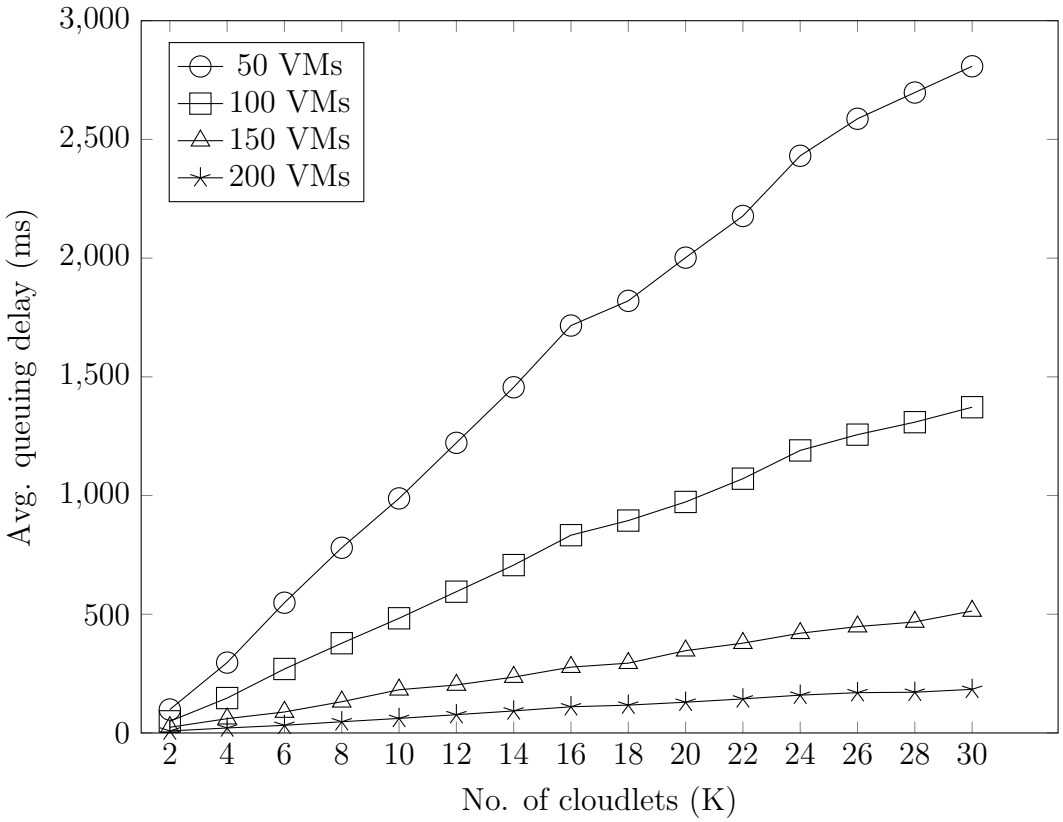

Figure 5.2: Average queuing delay of cloudlets

time decreases as the number of available VMs increases.

5.3 Network Delay

Figure 5.3 shows the average network delay encountered when cloudlets were sent from broker federate to the datacenter federate. It can be clearly observed that the network delay increases as the number of cloudlets that are submitted from broker to datacenter increases. In start when broker have small number of a cloudlets to send there is a minimal network delay. As the same network is used by other applications so the increase in network delay is also observed. Moreover, increasing the batch sizes also incurs an additional delay due to the increase in packet sizes.

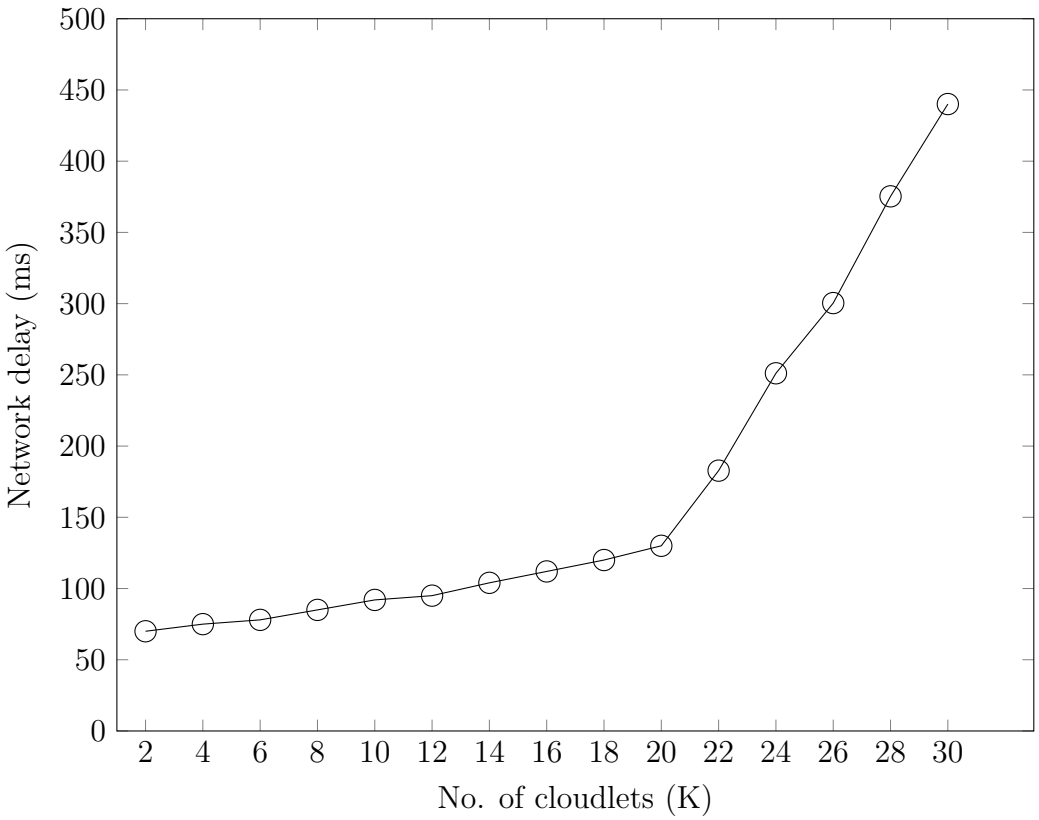

Figure 5.3: Network Delay.

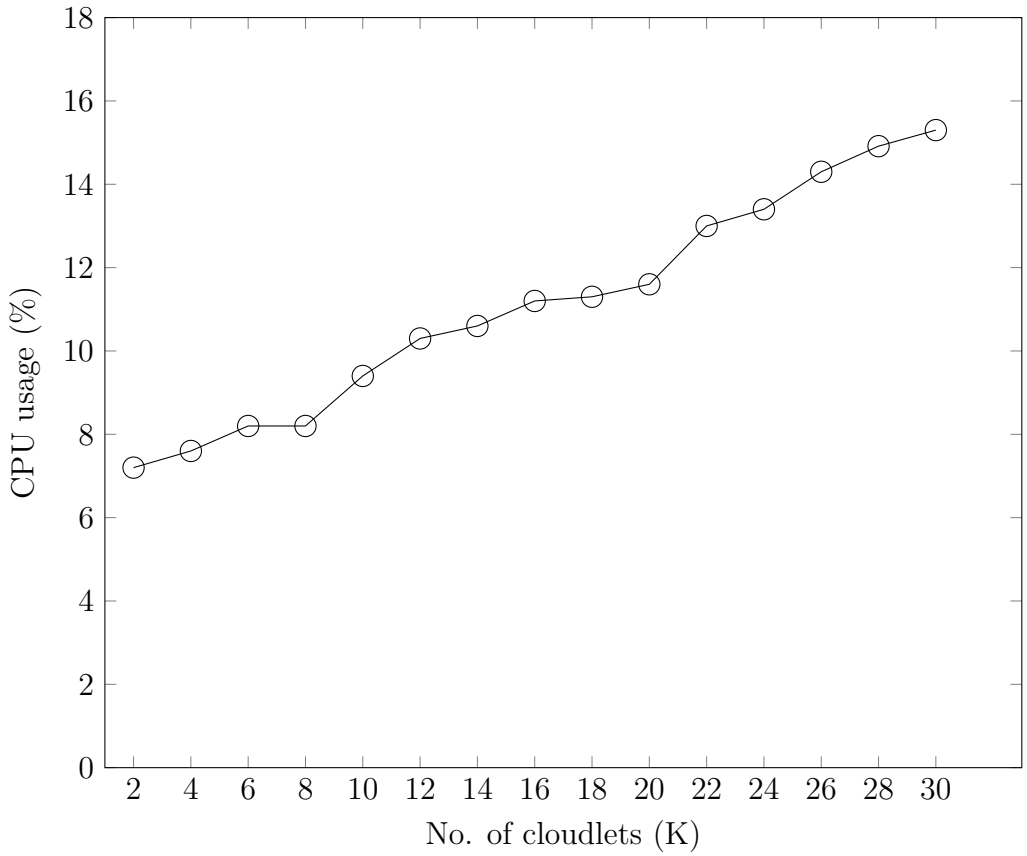

Figure 5.4: CPU usage at broker federate

5.4 Resource Usage

Due to resource limitations of a system its a challenging task to simulate a large number nodes of that system. This limitation is tackled by our proposed framework decoupled design.

Figure 5.4, 5.5, 5.6 shows the percentage of CPU usage at the broker federate, data center federate and ClouSim respectively. In comparison to the CPU usage of traditional CloudSim the CPU usage for the broker federate and data center federate is low. Moreover, the execution of modules on separate systems can lead to lower resource usage, extending scalability of the cloud computing framework. A similar trend is observed for memory consumption in broker federate, data center federate and traditional CloudSim as shown in Figure 5.7, 5.8, 5.9 respectively.

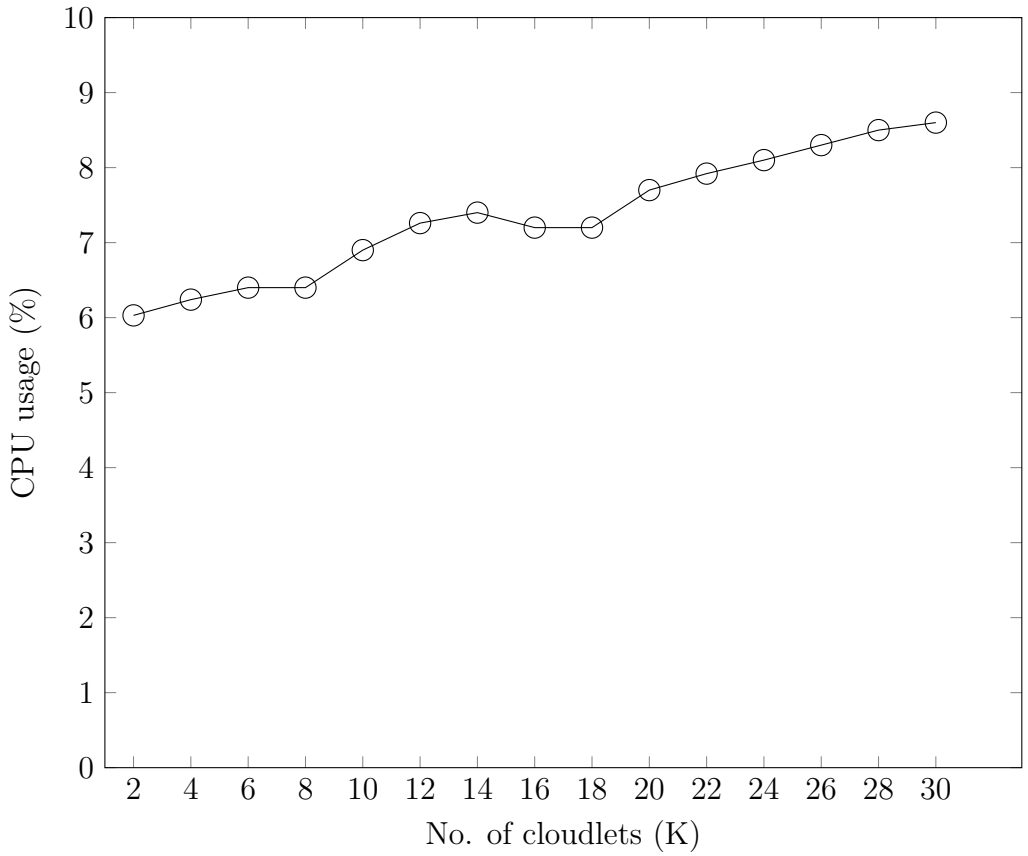

Figure 5.5: CPU usage at data center federate.

5.5 Summary and Discussion

In our proposed framework multiple data centers can be integrated to a broker. Thus, making it possible to simulate a large number of nodes. The service time and queuing delay decreases as increase in number of VMs. We have also compared the CPU and memory usage of proposed framework with traditional CloudSim. As CloudSim has tightly coupled environment, so CPU and memory usage are higher compared to the proposed framework where data center and broker are executing on different physical systems. Thus, resources limitation are no longer the rigid constraint in.

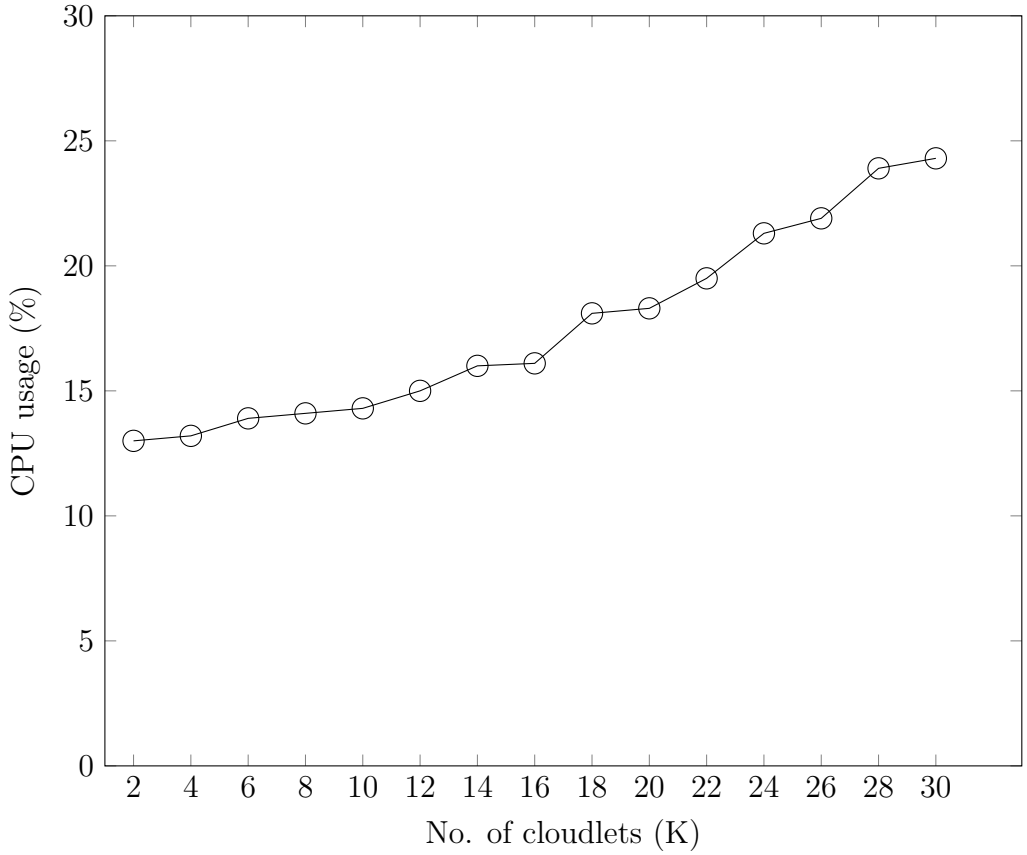

Figure 5.6: CPU usage at CloudSim.

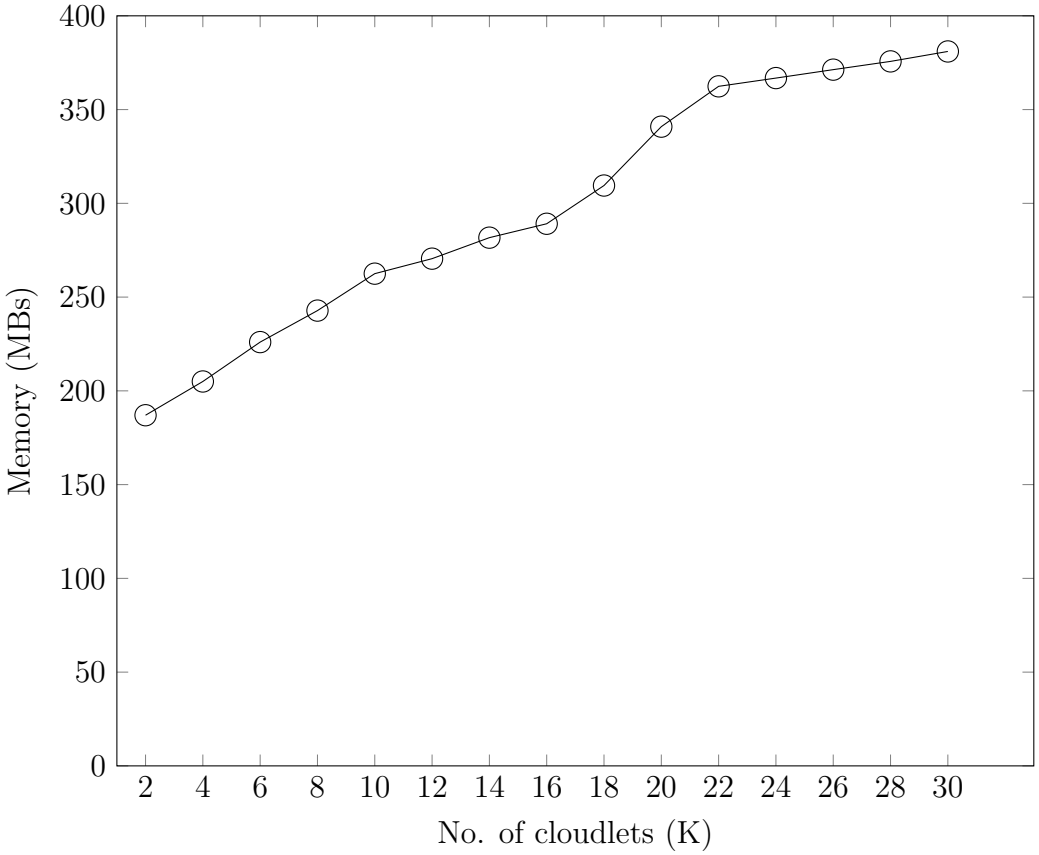

Figure 5.7: Memory usage at broker federate.

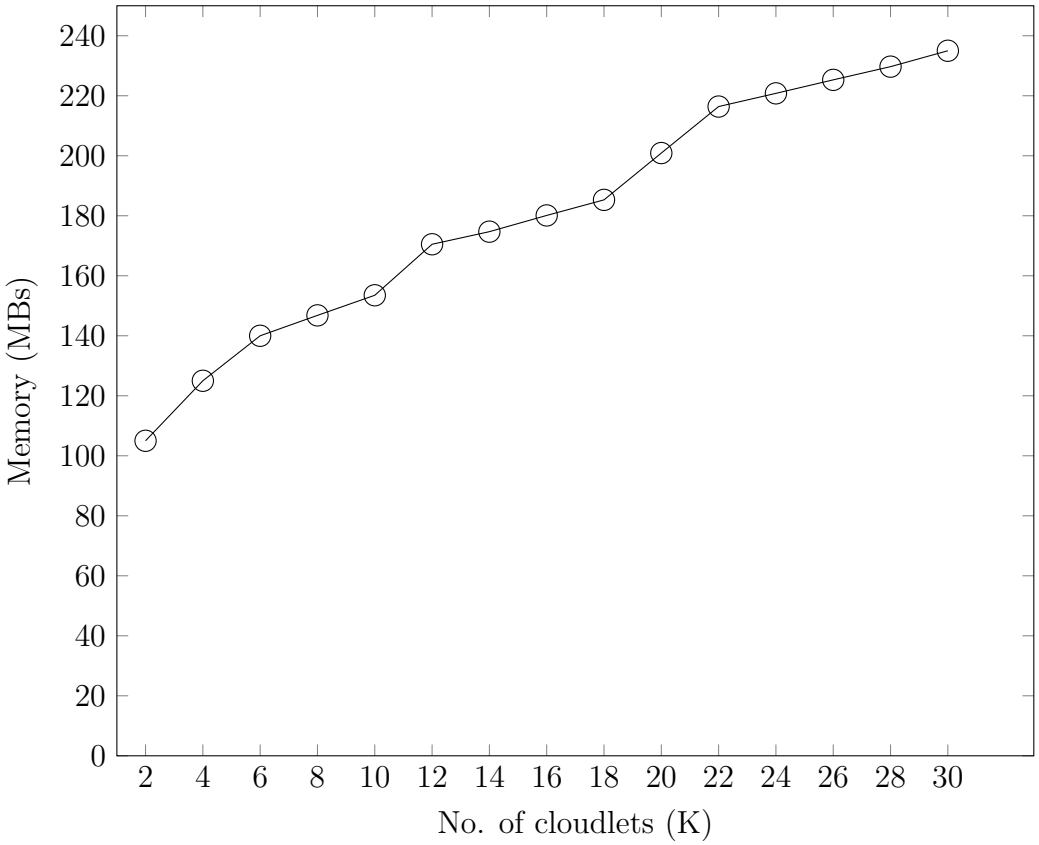

Figure 5.8: Memory usage at datacenter federate.

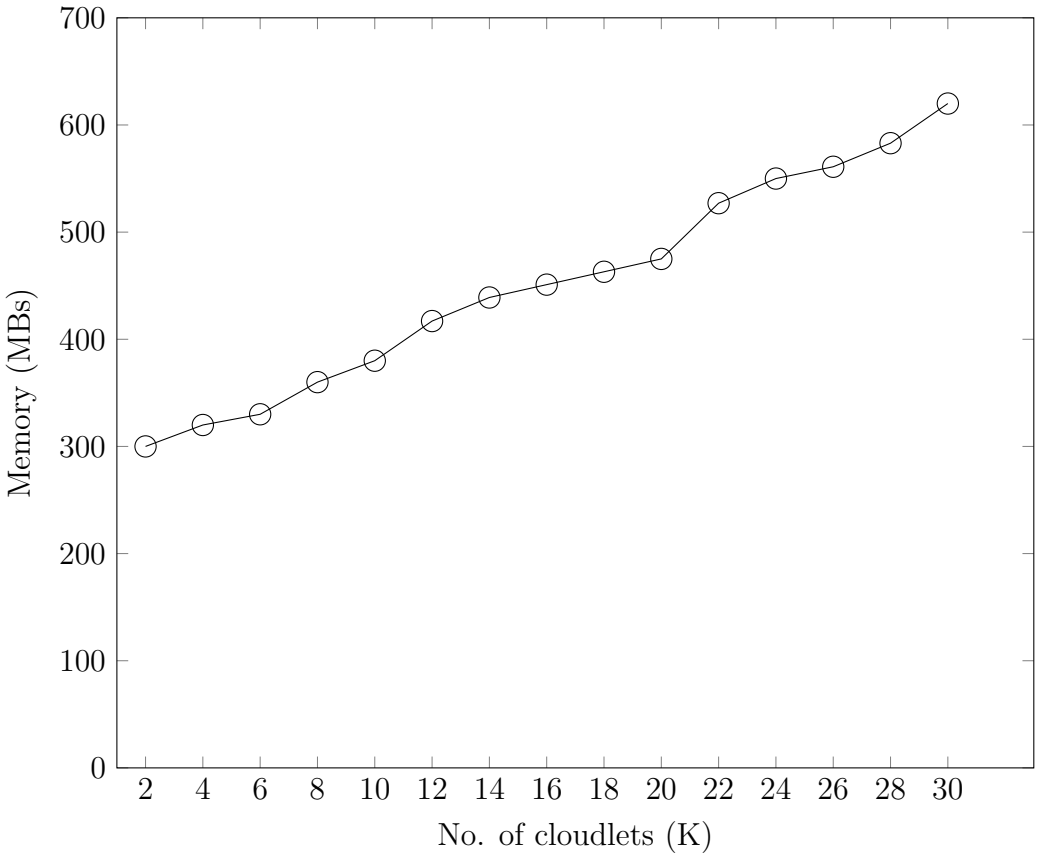

Figure 5.9: Memory usage at CloudSim.

Chapter 6

Future Work & Conclusion

This chapter concludes the presented research work and highlights potential future research directions.The first section of the chapter 6.1 presents the future research work directions, whereas second section section 6.2 gives a brief conclusion of three major research contributions.

6.1 Future Work

Our thesis work can be extended in different directions where further research can be carried out. In the future, we are interested in integrating it with IoT and vehicular network simulators which will open new directions and challenges for researchers.

6.2 Conclusion

Main focus of existing cloud simulation frameworks are on scheduling algorithms, SDN-based networking, VM migrations and towards energy efficiency. However, scalability of existing simulators is field less worked on. Most of existing cloud simulation frameworks supports few thousands of nodes, making it difficult for researches to simulate realistic data center models. We have proposed a CloudSim based simulator that allows simulator instances to execute on a different system. Thus, the framework provides scalability across simulators and to make communication possible among individual modules a separate layer of HLA-RTI is implemented. Furthermore, this framework allows the integration of various simulators for simulating scenarios based on more complexity.

Bibliography

[1] F. Teng, "Ressource allocation and schelduling models for cloud computing," Ph.D. dissertation, Ecole Centrale Paris, 2011.

[2] B. Jerry, *Discrete event system simulation*. Pearson Education India, 2005.

[3] E. Casalicchio and E. Galli, "Metrics for quantifying interdependencies," in *International Conference on Critical Infrastructure Protection*. Springer, 2008, pp. 215–227.

[4] D. D. Dudenhoeffer, M. R. Permann, and M. Manic, "Cims: A framework for infrastructure interdependency modeling and analysis," in *Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 2006, pp. 478–485.

[5] J. S. Dahmann, "The high level architecture and beyond: technology challenges," in *Proceedings of the thirteenth workshop on Parallel and distributed simulation*. IEEE Computer Society, 1999, pp. 64–70.

[6] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[7] I. S. Association *et al.*, "1516–2010-ieee standard for modeling and simulation (m&s) high level architecture (hla)," 2010.

[8] "Ieee standard for modeling and simulation (m amp;amp;s) high level architecture (hla)– object model template (omt) specification - redline," *IEEE Std 1516.2-2010 (Revision of IEEE Std 1516.2-2000) - Redline*, pp. 1–112, Aug 2010.

[9] S. I. S. Committee *et al.*, "of the ieee computer society. ieee standard for modeling and simulation (m&s) high level architecture (hla)-ieee std

1516-2000, 1516.1-2000, 1516.2-2000. new york: Institute of electrical and electronics engineers," *Inc., New York*, 2000.

[10] H. Wang, G. Tang, and Y. Lei, "The framework of distributed interactive simulation for space projects," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2003, p. 5608.

[11] "Ieee standard for modeling and simulation (m amp;amp;s) high level architecture (hla)– federate interface specification - redline," *IEEE Std 1516.1-2010 (Revision of IEEE Std 1516.1-2000) - Redline*, pp. 1–378, Aug 2010.

[12] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. IEEE, 2010, pp. 446–452.

[13] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[14] ——, "Simulating communication processes in energy-efficient cloud computing systems," *CLOUDNET*, pp. 2015–2017, 2012.

[15] ——, "Dens: data center energy-efficient network-aware scheduling," *Cluster computing*, vol. 16, no. 1, pp. 65–75, 2013.

[16] S.-H. Lim, B. Sharma, G. Nam, E.-K. Kim, and C. R. Das, "Mdcsim: A multi-tier data center simulation, platform." in *CLUSTER*, vol. 31, 2009, pp. 1–9.

[17] S. Garg and R. Buyya, "Networkcloudsim: Modelling parallel applications in cloud simulations," 12 2011, pp. 105–113.

[18] J. Jung and H. Kim, "Mr-cloudsim: Designing and implementing mapreduce computing model on cloudsim," in *ICT Convergence (ICTC), 2012 International Conference on*. IEEE, 2012, pp. 504–509.

[19] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013.

[20] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "Cloudsimsdn: Modeling and simulation of software-defined cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on.* IEEE, 2015, pp. 475–484.

[21] W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Cepsim: Modelling and simulation of complex event processing systems in cloud environments," *Future Generation Computer Systems*, vol. 65, pp. 122–139, 2016.

[22] D. C. Alves, B. G. Batista, D. M. Leite Filho, M. L. Peixoto, S. Reiff-Marganiec, and B. T. Kuehne, "Cm cloud simulator: A cost model simulator module for cloudsim," in *Services (SERVICES), 2016 IEEE World Congress on.* IEEE, 2016, pp. 99–102.

[23] T. T. Sá, R. N. Calheiros, and D. G. Gomes, "Cloudreports: an extensible simulation tool for energy-aware cloud computing environments," in *cloud computing.* Springer, 2014, pp. 127–142.

[24] W. Tian, Y. Zhao, M. Xu, Y. Zhong, and X. Sun, "A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 153–161, 2015.

[25] Y. Jararweh, M. Jarrah, Z. Alshara, M. N. Alsaleh, M. Al-Ayyoub *et al.*, "Cloudexp: A comprehensive cloud computing experimental framework," *Simulation Modelling Practice and Theory*, vol. 49, pp. 180–192, 2014.

[26] G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "An analysis of first fit heuristics for the virtual machine relocation problem," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm).* IEEE, 2012, pp. 406–413.

[27] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm).* IEEE, 2012, pp. 385–392.

[28] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management tech-

niques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[29] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "Groudsim: an event-based simulation framework for computational grids and clouds," in *European Conference on Parallel Processing*. Springer, 2010, pp. 305–313.

[30] M. Bux and U. Leser, "Dynamiccloudsim: Simulating heterogeneity in computational clouds," *Future Generation Computer Systems*, vol. 46, pp. 85–99, 2015.

[31] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012.

[32] A. W. Malik, K. Bilal, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan, and R. Buyya, "Cloudnetsim++: A toolkit for data center simulations in omnet++," in *11th IEEE International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET), Charlotte, NC, USA, December 2014.*, 2014.

[33] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan, "Fognetsim++: A toolkit for modeling and simulation of distributed fog environment," *IEEE Access*, vol. 6, pp. 63 570–63 583, 2018.

[34] M. Shiraz, A. Gani, R. H. Khokhar, and E. Ahmed, "An extendable simulation framework for modeling application processing potentials of smart mobile devices for mobile cloud computing," in *Frontiers of Information Technology (FIT), 2012 10th International Conference on.* IEEE, 2012, pp. 331–336.

[35] S. Sotiriadis, N. Bessis, N. Antonopoulos, and A. Anjum, "Simic: Designing a new inter-cloud simulation platform for integrating large-scale resource management," in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on.* IEEE, 2013, pp. 90–97.

[36] U. U. Rehman, A. Ali, and Z. Anwar, "seccloudsim: Secure cloud simulator," in *Frontiers of Information Technology (FIT), 2014 12th International Conference on.* IEEE, 2014, pp. 208–213.

[37] X. Li, X. Jiang, K. Ye, and P. Huang, "Dartcsim+: Enhanced cloudsim with the power and network models integrated," in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 644–651.

[38] A. Falcone, A. Garro, S. J. Taylor, A. Anagnostou, N. R. Chaudhry, and O. Salah, "Experiences in simplifying distributed simulation: The hla development kit framework," *Journal of Simulation*, vol. 11, no. 3, pp. 208–227, 2017.

[39] C. Sung and T. G. Kim, "Framework for simulation of hybrid systems: Interoperation of discrete event and continuous simulators using hla/rti," in *Principles of Advanced and Distributed Simulation (PADS), 2011 IEEE Workshop on*. IEEE, 2011, pp. 1–8.

[40] A. V. Brito, L. F. S. Costa, H. Bucher, O. Sander, J. Becker, H. Oliveira, and E. U. Melcher, "A distributed simulation platform using hla for complex embedded systems design," in *Proceedings of the 19th International Symposium on Distributed Simulation and Real Time Applications*. IEEE Press, 2015, pp. 195–202.

[41] S. Guan, R. E. De Grande, and A. Boukerche, "An hla-based cloud simulator for mobile cloud environments," in *Distributed Simulation and Real Time Applications (DS-RT), 2016 IEEE/ACM 20th International Symposium on*. IEEE, 2016, pp. 128–135.

[42] C. Gervais, J.-B. Chaudron, P. Siron, R. Leconte, and D. Saussie, "Real-time distributed aircraft simulation through hla," in *Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2012, pp. 251–254.

[43] A. W. Malik, A. Basit, and S. A. Khan, "Hla compliant network enabled distributed modeling and simulation infrastructure design," in *Proceedings of the 4th WSEAS International Conference on Software Engineering, Parallel & Distributed Systems*. World Scientific and Engineering Academy and Society (WSEAS), 2005, p. 20.

[44] E. Galli, G. Cavarretta, and S. Tucci, "Hla-omnet++: An hla compliant network simulator," in *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2008, pp. 319–321.

[45] T. Lu and G. Wu, "The war-gaming training system based on hla distributed architecture," in *null*. IEEE, 2002, p. 889.