An HLA-based Cloud Simulator for Mobile Cloud Environments

Shichao Guan
PARADISE Research Laboratory
EECS - University of Ottawa
Email: sguan049@uottawa.ca

Robson Eduardo De Grande
PARADISE Research Laboratory
EECS - University of Ottawa
Email: rdgrande@site.uottawa.ca

Azzedine Boukerche
PARADISE Research Laboratory
EECS - University of Ottawa
Email: boukerch@site.uottawa.ca

***Abstract*—Mobile Cloud Computing is a concept infrastructure wherein Cloud Computing resources are utilized to offload tasks from mobile elements. The combination of Cloud Computing and Mobile Computing increases the complexity of modeling and performance evaluation in terms of task scheduling policies, energy consumption model, the mobility of mobile devices over a layered architecture of both local Clouds and remote Cloud data centers. In this paper, a distributed HLA-based Cloud toolkit is proposed, enabling the modeling and simulation of Mobile Cloud Computing environment. The proposed toolkit simulates the behaviors of the Mobile system and the Cloud infrastructure, within which different resource scheduling policies can be evaluated in a repeatable manner. In addition, an HLA-based simulation scheduling scheme is proposed, trying to improve simulation execution efficiency by automatically parallelizing and distributing simulation components. A Cloud-based simulation resource management paradigm is also implemented, handling the configuration and maintenance issues regarding underlying system resources and simulation data. Based on the experiments, the proposed toolkit can achieve better simulation execution efficiency, with consideration of both Cloud and Mobile behaviors, compared to current Cloud Computing environment simulators.**

***Index Terms*—Mobile Cloud Computing; Modeling and Simulation; Performance Evaluation; Distributed Simulation; Scalability.**

I. Introduction

As with the development of computing, storage, and network hardware, significant attention is directed towards mobile devices, which has significantly changed habits and life routines of most people. Equipped with modern CPU, flash storage, and high-speed network, one single smartphone now can accomplish tasks and jobs that may require several workstations in the past. As a result, new features, more functionality, and enhanced sensors can be designed, implemented, and integrated together into a miniaturized scale to fit in pockets.

To overcome the fundamental challenges, such as unstable network, relatively resource-poor computing and storage capacity, as suggested in [1], Cloud Computing, defined in [2] and [3], is widely utilized to support computational-intensive tasks and to prolong battery life for mobile elements. Due to the flexible XaaS service provisioning model(IaaS, PaaS, SaaS) defined in Cloud Computing, mobile applications can be handled in different granularities from hardware to software in a self-managed manner. In addition to offloading computational load, integrating Cloud with Mobile Computing can easily introduce mature technologies in Cloud Computing to enhance security monitoring, data backup, load balance so that higher QoS can be achieved to enrich the performance of applications.

Due to the advantages of Mobile Cloud Computing, an increasing number of researches from both academia and industry have focused on Mobile Cloud Computing; this interest is based on the statistics shown in [4], [5] and [6]. Recent works are targeting mainly the following components:

- *Offloading Model*: it defines leveraging resources to form a Cloud for resource-poor mobile devices, concerning VM overlay [7]; task partitioning, profiling and scheduling [8], [9]; and energy consumption with overhead evaluation [10], [11], [12], [13], [14].
- *Mobile Element Management*: it describes functionalities on data and environment management, which include building localization systems [15], [16]; connection issues [17], [18]; and context-aware optimizations [19], [20], [21].

In order to evaluate and validate new solutions in a cost-effective manner, simulations are typically used for repeatable and extensive evaluation, compared to real experiments and prototype construction. Many simulators are proposed and developed in terms of network, mobility, and Cloud infrastructure, with different emphases. Network-based simulators are designed for measuring algorithms used on different layers of the network; mobility simulators mainly target behaviors in the specific location; Cloud-based simulators focus on the issues within Cloud data centers, such as resource scheduling policies, energy consumption reduction, and network topology optimization. However, due to the novelty of Mobile Cloud Computing, no official definition, standards or simulators, are currently available for this novel technological field.

In this paper, an HLA-based simulator is proposed, targeting the Mobile Cloud environment, especially for the offloading process, concerning behaviors of both mobile elements and cloud resources. In the simulator, a scheduling scheme is designed to improve simulation execution efficiency by parallelizing and distributing simulation tasks. In addition, a simulation management paradigm is implemented to ease the configuration and maintenance of simulation resources.

This paper is organized as the following. Section II presents the state of the art, highlighting the challenging issues. Section

III introduces the proposed simulator by describing its architecture, key components, and functioning. Section IV shows the experimental results and the respective discussion. Finally, the last section summarizes the paper and presents future work directions.

II. Related Work

As with the maturing of the concept of Mobile Computing and Cloud Computing, many simulation solutions have been proposed in terms of mobility, network, and cloud infrastructure, targeting different requirements and aspects.

Network Simulator 2 [22] and its successor Network Simulator 3 [23] are discrete event-triggered, object oriented simulators for R&D of wired and wireless networking. By integrating C++ hierarchy and object-oriented extension of Tcl (Tool Command Language), simulation execution efficiency can be achieved, and flexible user-defined network topology can be properly supported. At their latest running version, software packages are updated with more open-source components integrated and supported; original simulation parameters are updated with support for virtualization settings. However, additional work or plugins are needed to simulate, or emulate, the stacked layers of Cloud services in them.

Similarly, OMNeT++ [24] introduces a simulation framework based on discrete events written by C++ mainly built for networking purposes. The proposed framework supports hierarchical modules that can communicate through messages via connections. To enable flexible structure of the model, description language NED is employed to parameterize network topologies and graphical presentation. OMNet++ provides a more flexible, dynamic, and modular simulation platform where mobility can be incorporated from specialized simulators. However, this simulation framework requires plugins and additional components to enable Cloud aspects into the simulation environment.

Opportunistic Networking Environment (ONE) simulator [25] specifically considers the DTN (Delay-Tolerant Networking) with mobility patterns, which includes both DTN routing protocols and mobility models based on trace outputs. In addition, energy consumption models and visualization components are also provided in a modular manner, being available for extending functions with supported interfaces. Nevertheless, this simulator restricts itself in the scope of DTNs, hindering the possibilities in implementing sophisticated mobile Cloud models.

GridSim [26] is a simulation toolkit that has been developed to support the modeling of grid elements, users, machines, and networks and to enable the evaluation of resource brokers and scheduling algorithms, based on SimJava [27]. This simulator is the precursor of CloudSim [28], which has been proposed to focus specifically on Cloud computing environment. CloudSim is a Java-based toolkit for simulation and modeling the Cloud Computing environment. The simulator supports analysis based on different granularities from soft-layer to hard-layer, within which behaviors of tasks, VMs, hosts, and clusters can be defined, observed, and evaluated. In addition, the proposed framework also supports features regarding resource scheduling policies, load balancing algorithms, energy consumption evaluation, and network protocol design in different Cloud environments. CloudAnalyst [29] is a cloud simulator built on top of CloudSim. This simulator specifies on the modeling of the behavior of applications; it considers the different load among geographically distributed Cloud infrastructures against the execution of applications.

GreenCloud [30] is a Ns2-based [22] simulator for Cloud infrastructures, specially for energy consumption of network elements in data centres. The GreenCloud framework mainly considers the energy consumption on servers, switches, and links. The energy consumption can be observed over the performance of applications, or processes, that present distinct execution characteristics, which can exercise computation-intensive load, communication-intensive load, and balanced load. In this case, power management schemes can be compared and evaluated on servers, as well as networking components, depending on the volume of task interactions.

iCanCloud [31] is a simulator based on SIMCAN simulation framework [32] for Cloud Computing. This simulator offers a model that covers the full set of elements of a Cloud; it focuses on the system functioning from hardware models to protocols and applications. The proposed simulator introduces a modular perspective of a Cloud through a model that naturally divides the Cloud infrastructure into different layers. The simulator also provides a GUI-based user interface to ease parameter configuration. In addition, this simulator emphasizes on the execution efficiency of the Cloud simulations by suggesting and attempting to utilize Message Passing Interface (MPI) [33] as a design feature to achieve distribution and parallelism in the simulations.

SPECI [34] is a simulation tool particularly designed for Cloud data centers; it mainly focuses on analyzing the performance issues present when scaling up Cloud systems. In this work, the Simkit [35] is employed to provide discrete event graph models that support component-based modeling, so this toolkit enables to trigger system behaviors according to a central clock and event list. Although this work introduced Discrete event simulations (DES) [36] components, it restricts itself on the aspect designs and patterns, not fully discussing simulation execution efficiency.

Based on the discussion regarding aforementioned previous works and summary of their characteristics shown above, there are no simulators that are specially tailored for Mobile Cloud Computing. On one hand, network- and mobile-based simulators cannot evaluate or are drastically limited for modeling the performance of Cloud infrastructure in terms of CPU, memory, and network environment in large-scale data centers. On the other hand, Cloud-based simulation toolkits lack the consideration of resource-poor mobile elements, as well as the mobility of these elements. In this paper, we propose a simulator specifically developed for Mobile Cloud Computing, targeting the offload of these devices into the Cloud. In the simulator, mobility issues, network conditions, and Cloud resources are taken into consideration. To improve simulation

execution efficiency, a distributed simulation framework is employed: HLA-based distributed execution mechanisms are introduced to parallelize and distribute simulation components. The configuration and maintenance of underlying systems are handled by resource management components of the simulator.

III. Simulator Architecture and Key Components

The main focus of the proposed system is to observe and evaluate the offloading process in the Mobile Cloud environment. To decouple the simulation components and to enable parallel and distributed execution, HLA-based middleware [37] is introduced to support the simulation of system behaviours. To manage simulation resources and underlying system resources, a resource management system is designed and implemented to ease the handling of simulation preparation, configuration and to support large-scale distributed multi-user simulations, based on the concept of Cloud Computing [2] and [3]. In this section, the entire architecture of the simulator will be discussed with details of key features and functioning.

A. Simulator Architecture

As shown in Fig. 1, the proposed system is mainly divided into two parts – Simulation Function Layer and Resource Management Layer:

- Simulation Function Layer: this layer directly involves simulation related setting and functions. It defines the properties of the simulation objects of both Cloud objects and Mobile objects, and formalizes the simulation entities collaboration execution procedure during task partitioning, profiling and scheduling processes.
- Resource Management Layer: this layer mainly handles supporting services for simulation execution. The Simulation Resource Manager handles the configuration, maintenance and monitoring services of the soft-layer simulation resources. The Virtual Resource Manager handles the construction of simulation execution-ready environment, targeting the hard-layer computing and networking resources. The Raw Resource Manager is responsible for executing resource management policies on the raw resources.

B. Simulation Function Layer

In this layer, the key functions are designed and implemented, targeting the simulation execution process. In this proposed simulation model, object attributes and interactions are defined based on poRTIco HLA RTI [38].

1) Offloading Process Modeling: Based on the researches regarding task remote execution, shown in [8] and [9], the offload process in Mobile Cloud Computing is mainly divided into three stages: task partitioning, profiling and task scheduling:

- Task partitioning: this stage is to divide the application tasks into unit tasks which cannot be divided further and preliminarily defines the tasks that that require remote execution. The tasks and programs can be partitioned statically based on the annotation of programmers or automatically based on pre-defined constraints. The constraints involves tasks that are requiring specific mobile components which are not available in remote Cloud resources and re-execution components which may introduce nested migrations. Typically in this stage, tasks are partitioned into functions or classes, waiting for profiling based on the dynamic execution environment.
- Profiling: the profiling process decides whether a unit task should be executed remotely on the Cloud resource or locally on the mobile element itself, based on the consideration of energy consumption. The task evaluation is according to the energy consumption comparison between native execution and remote execution. In the native execution scenario, the energy consumption of the task is decided by the task size and the computational power of the corresponding native device. As for the remote execution scenario, the energy consumption of the task becomes more complex. The entire energy cost includes the cost during uploading task, remote execution and download execution results. In this case, the calculation of energy cost is determined by not only task size and local computational power, but network situation between mobile devices and remote resource, computational power of remote resources, task code size and result format. Due to the mobility of the mobile devices, the network environment such as Wi-Fi or 3G, 4G Cellular network may change during the uploading and downloading subprocess. Overhead may be introduced if network is not available or stable. In this stage, each unit task is evaluated in local execution scenario and remote scenario, ready for global optimization and final scheduling decision. TO improve the speed of evaluation, location-based historical evaluation data may be introduced and referenced to reduce the rounds of evaluation and to improve offloading efficiency.
- Scheduling: the final step of offloading is to decide the list of remote executable tasks. Although each task is evaluated as shown in the previous stage, remotely and locally, the final partitioning decision requires to consider the global optimization of the entire application. In the extreme case, even each task's local execution cost is smaller than its remote execution cost, offloading all tasks to remote execution may achieve a better result than executing all tasks locally. To optimize the problem from the global point of view, the behavior of this process can be defined as finding the shortest path in the network. To simplify the problem, given a unit task that contains two choices to be executed, two nodes can be used to present these difference choices, as node local and node remote. For each node, the weight of edge links to it is the energy consumption cost to select this scheduling choice. In this case, the global optimization of the application is to find the path through the nodes with the least cost. In reality, the weight of the edges may differ due to different concerns of the scheduling process. On one hand, time restrict applications may introduce scheduling deadline-

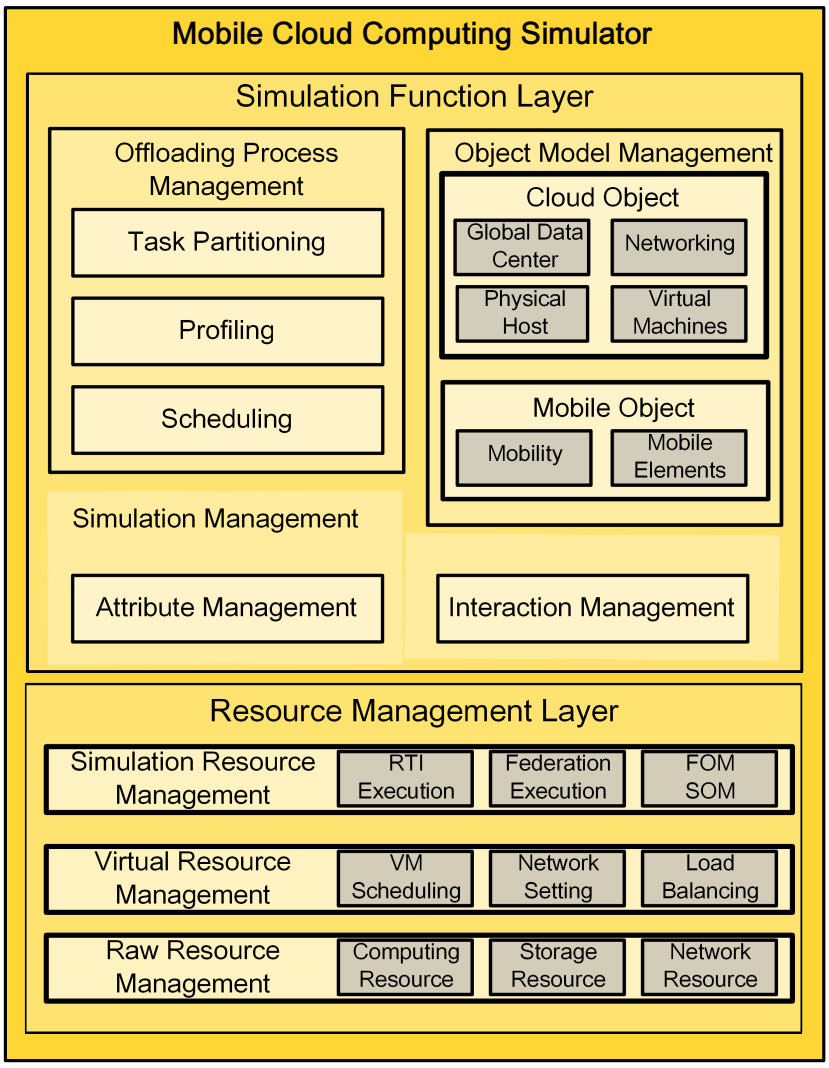

Fig. 1. Mobile Cloud Computing Simulator

based weights in addition to energy consumption. On the other hand, energy exhausted devices may introduce more weights on the energy consumption.

2) Object Setting: To fulfill the simulation and modeling of the proposed offloading model, objects are divided into Cloud resource and mobile elements respectively, with different attribute setting and interactions, as shown in Fig. 2:

- Remote Resource Object: the objects include remote resources that are responsible for executing tasks from mobile elements. Based on the scale of the remote resources, the type of which can be divided into native machine objects for a single remote server and cluster objects for a bunch of machines stored in data centres. For the latter cluster object, it can be further classified by network object, physical object and virtual object. The network object defines the network environment for the cluster physical object, which involves network topology and network parameters such as bandwidth, latency. Similar to the native machine object, the physical object inside cluster object identifies the properties of the computing host unit and storage host unit. The virtual object contains the setting of virtual machines that are virtualized based on physical machines. These virtual object instances are responsible for the handling of each one of offloading tasks on behalf of the cluster object.
- Mobile Element Object: the mobile objects include the mobile devices and mobile tasks. Due to the characteristics of the mobile device such as mobility and resource capability, it utilize a different set of parameters compared to static machine objects.

3) Simulation Execution: the HLA framework is a formal framework to enable interoperability and reusability of simulations. The typical simulation execution process involves RTIExec, FedExec and Federates. Federates are simulation entities which defines the behaviours of the target based on Federate Rules and Federation Rules. Simulation data is defined by OMT (Object Model Template) while data communication is handled by RTI (Run Time Infrastructure) components,

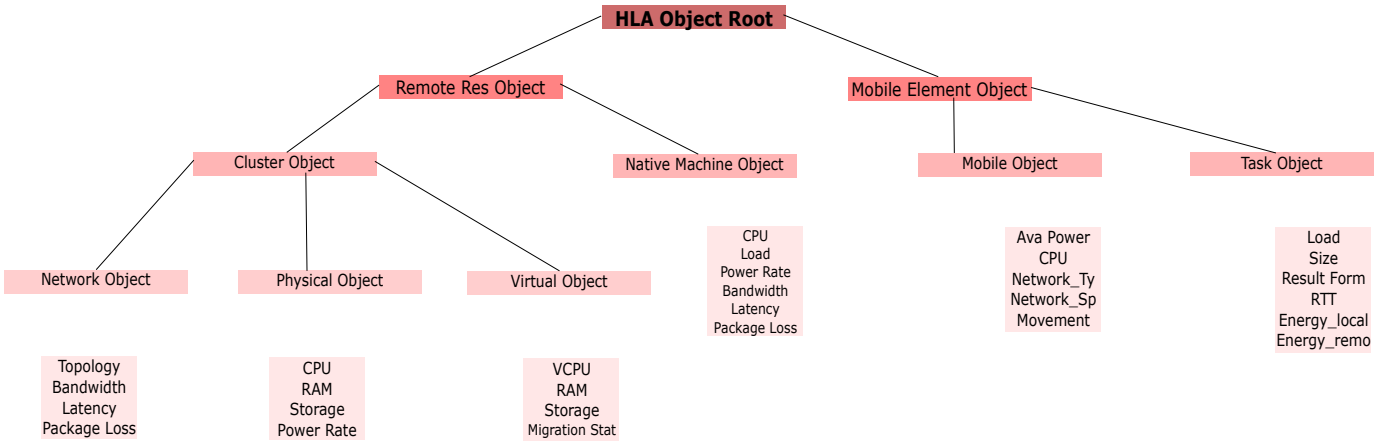

Fig. 2. Object Architecture

detailed in HLA Interface Specification. The simulation is initiated by RTIExec, which delegates a FedExec on the RTI Ambassador of the first joined federate. Other federates can be involved in the target federation through FedExec invoking functions. Then FexExec starts and terminates the execution of the corresponding federation.

To create aforementioned objects, the federate is to publish and register the objects and instances. Through this declaration, the objects and instances can be subscribed and discovered. Similarly, interactions also require the federate to subscribe first so that the intended interaction classes can be properly produced. The Callback functions gather the results from different federates then trigger updates accordingly.

C. Resource Management Layer

Although the HLA standard provides the framework for deploying distributed simulations on distributed computing resources, the handling of executing the simulations is not specified. In this case, the underlying system requires a pre-configuration process for each physical element before the simulation can start properly.

To cater the resource management details, a three-layered Cloud-based Resource Management System is proposed, to ease the configuration, management, and maintenance of the system. The Cloud architecture is chosen due to several reasons. It supports scalability and can easily scale up with on the modular simulation components. In addition, it handles functions in a XaaS manner, based on which the management of resources can be accomplished from soft-layer to hard-layer, shown in Fig. 3.

1) Simulation Resource Management: The simulation Resource Management is mainly focusing on the SaaS layer simulation application management within the virtual machine:

- Simulation configuration: a GUI-based user interface is provided to set and manage simulation-related parameter values, in the FOM and SOM; and computing resource values from the underlying layer. With a graphic view of the available resources, users can personalize the parallel level of the simulation.

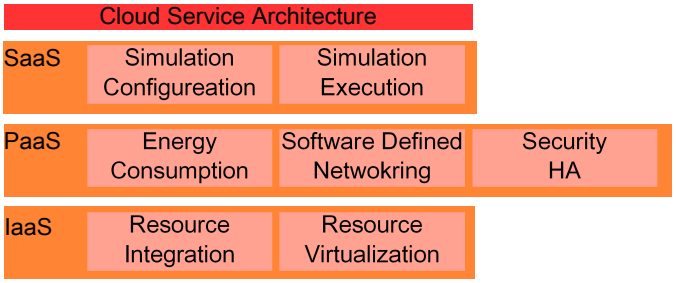

Fig. 3. Cloud-based Resource Management Architecture

- Simulation execution: in order to enable parallel execution, the Federation needs to wait until all Federates join it. The Simulation Resource Manager ensures the correct execution by analysing execution log files from the simulation application and system report. The execution final results are saved as log files, available from the web-based GUI.

This management layer provides an entry to the simulator system, with a global view of the entire underlying resources. Although the configuration and management can be accomplished manually in the command line, the graphic tools can ease the code details for users that are not familiar with system administration management.

2) Virtual Resource Management: The virtual resource management is built on PaaS. In this layer, the focus of the task involves the scheduling policies and network settings among virtual instances.

- Scheduling: the scheduling of the virtual instances are mainly based on the execution efficiency of the user-defined simulations. For computing-intensive simulations, large VMs with multiple VCPU and large RAM are booted while for communicating-intensive simulations VMs are relatively small but with less communication distances. Considering the energy consumption, migration is utilized to centralize the tasks in physical hosts and halt idle resources.

TABLE I
SINGLE MACHINE ENVIRONMENT

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

TABLE II
VIRTUAL RESOURCE LIST

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

- Network Setting: in order to utilize arps to broadcast messages for communication, federates are required to be set in the same network. To overcome the physical network limitations between physical resources, a SDN middleware [39] is utilized to build a virtual local network over the physical links.

In case of execution failures and data damage, replicas and migrations are utilized to handle hardware faults. The users simulation resources are backup at idle time periodically as Cloud images, which are utilized to reboot instances. VMs on the broken physical host are migrated to idle ones so that the simulation process can restart with no interruptions.

3) Raw Resource Management: The Raw Resource Manager is used to coordinate raw computing resources, storage resources and network resources and implementing VT (virtualization technology). This layer handles the real execution of system setting from above GUI and maintains the Cloud middleware.

IV. PERFORMANCE EVALUATION AND DISCUSSION

The prototype of the proposed simulator has been implemented and deployed to test and compare with CloudSim, in terms of the performance of execution efficiency and resource provisioning efficiency. The simulator is built based on one single machine with a Quad Core 2.40GHz Intel CPU (4700MQ) and 16 gigabytes of RAM. VMware Workstation [40] was used as the virtualization tool, which created one management node and two computing nodes for implementing a simple cluster environment. The details of the nodes are shown in Table I.

In the first experiment, the simulation environment preparation process is evaluated on the performance on Resource Management Layer. In the experiment scenario, the environment configuration involves hard resource virtualization & integration and soft resource deployment. The simulation application is tested based on a general poRTIco RTI [38] example application with default VM setting in OpenStack [41], as shown in Table II. The Virtual Resource Manager utilize Round Robin as the resource scheduling policy with Linux system as virtual instance OS – CentOS 6 [42] and Fedora 17 [43].

TABLE III
SIMULATION ENVIRONMENT PREPARATION

VM Type	T_sche(s)	T_f17(s)	T_cen6(s)	T_avg(s)
Mini	2.064	3.445	3.945	
Small	2.278	3.705	3.998	189.845
Medium	2.162	3.753	4.011	

The experiment result is shown in Table III. T_sche is the time consumed by the Virtual Resource Manager during resource scheduling while T_f17 and T_cen6 is the time utilized by Raw Resource Manager to establish the corresponding Linux OS. T_avg is the average soft-layer resource provisioning time from Simulation Resource Manager. The result indicates that the simulation environment initialization can be finished in 200 seconds, with full configuration of underlying resources and simulation requirements. Due to the utilization of Virtualization technologies, the performance of the proposed environment preparation process is affected. The efficiency can be further improved by using native hosts that the Resource Manager deployed on.

To evaluate the simulation execution efficiency of the proposed simulator, the second experiment is built to compare its performance to CloudSim. Due to the target of CloudSim is the Cloud data centre, which does not involve mobile elements, the experiment is implemented to only evaluate the scheduling process of offloading, assuming that the task partition and resource profiling is finished. In the proposed offloading scenario, Cloud resources are built in two data centres (one federate per data centre). The simulation first initiates the two data centres and then selects the most suitable destination computing resource for task offloading. To make the comparison fairly, this experiment is built on the single machine environment as shown in the global setting for both simulators.

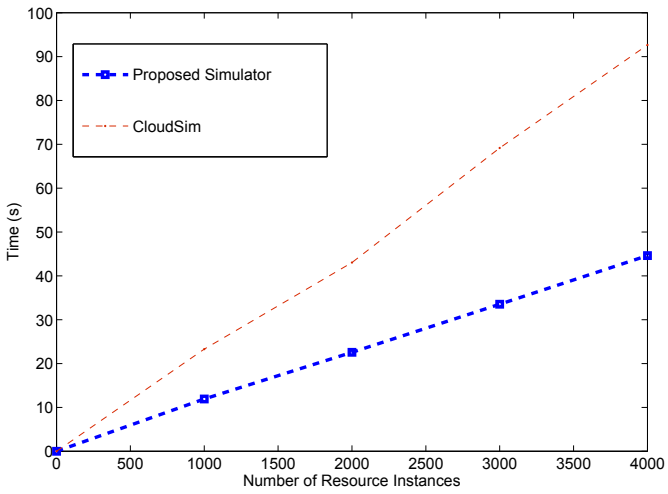

Fig. 4. Simulation Execution Efficiency on Number of Resources

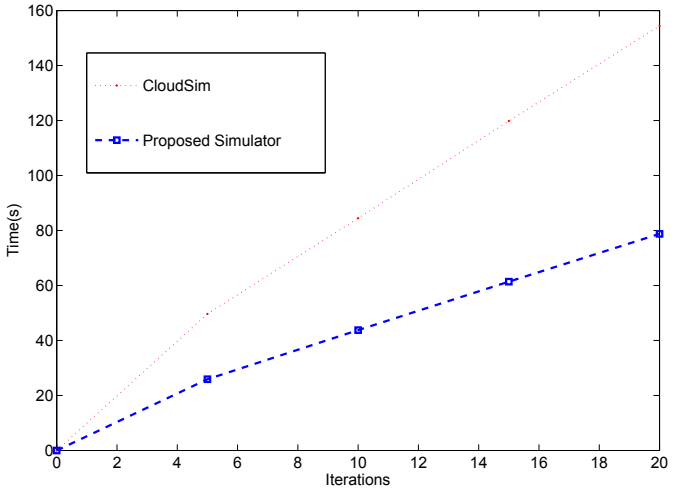

Fig. 5. Simulation Execution Efficiency on Number of Federates

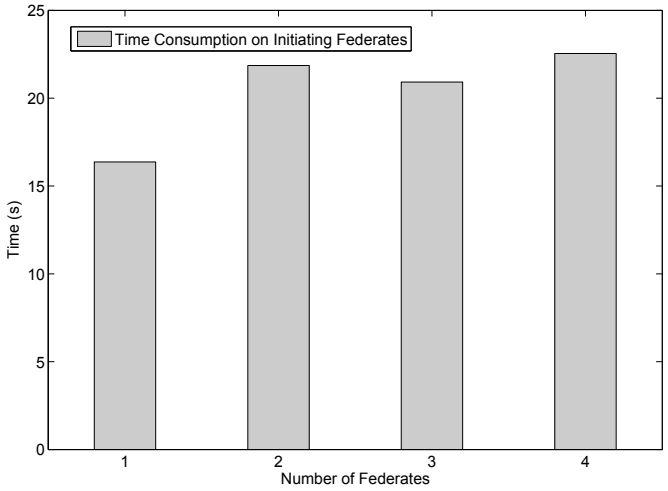

Fig. 6. Simulation Execution Efficiency on Federates

The result is illustrated in Fig. 4. As it shows, even in the single machine scenario, the proposed simulator can largely reduce the simulation time almost by half as with the increasing of the number of resource instances. The improvement of the simulation execution efficiency is due to better utilization of multi-cores. In traditional simulators, typically only one thread is utilized for a single simulation task while the optimization of the simulation execution is totally relying on the modeler's own effort. In the proposed simulator, paralleled tasks can be executed simultaneously by default, by delegating each federate per thread.

Similar results are presented in Fig. 5, which concerns the simulation execution performance when iteration increases. For each five iterations, the simulation execution time is recorded for both simulators. As it can be seen that it consumes more time to execute the first five iterations than the others. This is because the simulation initiation overhead, which includes the instantiation of object instances, loading and reading parameter setting list, deploying scheduling policy. In addition to the simulation initiation overhead, the proposed simulator also introduces extra overhead due to the distributed architecture. In order to avoid the deadlock that may be caused by the lack of global event list in the federates, time policies are needed to be deployed during initiation process so that the correct event execution order can be ensured. Despite this, the proposed simulator's overhead is still smaller, as the gradient shows in the iterations between zero to five. This is because the simulation object is delegated to distributed federates so that the instantiation of objects is also distributed, which largely reduced the time utilized in the initiation.

In the third experiment, the overhead of federation initiation during simulation environment preparation is specifically evaluated, based on the single machine mode the same as the second experiment scenario. In the experiment, the time used for creating and joining the federation by the HLA middleware is recorded with different numbers of federates involved in the federation.

The result is shown in Fig. 6. During the creation process, the federate ambassador is firstly registered then connected to the RTI. After this, the FOM modules are referenced to create the federation context. As discussed in the Simulation Execution section, the first linked federate will create the federation and the rest federates will join this federation which is just created. As presented in the results, the federation creating process takes around 16 seconds while the federate joining process takes another approximately 5 seconds in total. The fluctuation is caused by background system load while holding these virtual servers. The time consumed by joining the federation is not necessarily to increase linearly with the number of involved federates because the first federate broadcasts the federation creation information in the local network and other federates can receive it almost simultaneously. As a result, each federates can handle the response respectively.

As shown in the experiments, the paralleled nature of the proposed simulator can efficiently reduce the simulation execution time and initiation overhead. Also, the simulation environment can be easily built with full configuration of underlying resources.

V. Conclusion and Future Work

In this paper, an HLA-based distributed simulator is presented and evaluated. The proposed simulator is designed and implemented specifically for the offloading process on the Mobile Cloud Computing environment. To improve simulation execution efficiency, HLA-based federated architecture is utilized to distribute and parallel simulation objects and events. To ease the management of underlying systems, a Cloud-based resource management scheme is proposed. A simulation execution-ready environment can be automatically configured and built with the supporting services from different layers in the Cloud architecture. In the future work, the simulation execution performance of the proposed system will be evaluated based on the task partitioning and profiling process, compared with simulators that focus on the mobility and network issues.

REFERENCES

[1] M. Satyanarayanan, "Fundamental challenges in mobile computing," in *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*. ACM, 1996, pp. 1–7.

[2] P. Mell and T. Grance, "The nist definition of cloud computing," 2011.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[5] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[6] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 133–143, 2014.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[8] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[9] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[10] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[11] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2716–2720.

[12] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 190–194.

[13] M. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 1285–1293.

[14] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing." *HotCloud*, vol. 10, pp. 4–4, 2010.

[15] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did you see bob?: human localization using mobile phones," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 149–160.

[16] N. Banerjee, S. Agarwal, P. Bahl, R. Chandra, A. Wolman, and M. Corner, "Virtual compass: relative positioning to sense mobile social interactions," in *Pervasive computing*. Springer, 2010, pp. 1–21.

[17] E. E. Marinelli, "Hyrax: cloud computing on mobile devices using mapreduce," DTIC Document, Tech. Rep., 2009.

[18] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*. ACM, 2010, p. 6.

[19] H. J. La and S. D. Kim, "A conceptual framework for provisioning context-aware mobile cloud services," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 466–473.

[20] P. Papakos, L. Capra, and D. S. Rosenblum, "Volare: context-aware adaptive cloud service discovery for mobile systems," in *Proceedings of the 9th International Workshop on Adaptive and Reflective Middleware*. ACM, 2010, pp. 32–38.

[21] A. Klein, C. Mannweiler, J. Schneider, and H. D. Schotten, "Access schemes for mobile cloud computing," in *Mobile Data Management (MDM), 2010 Eleventh International Conference on*. IEEE, 2010, pp. 387–392.

[22] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2*. Springer Science & Business Media, 2011.

[23] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, 2008.

[24] A. Varga *et al.*, "The omnet++ discrete event simulation system," in *Proceedings of the European simulation multiconference (ESM2001)*, vol. 9, no. S 185. sn, 2001, p. 65.

[25] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 55.

[26] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[27] F. Howell and R. McNab, "Simjava: A discrete event simulation library for java," *Simulation Series*, vol. 30, pp. 51–56, 1998.

[28] 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.

[29] 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.

[30] 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.

[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. Núñez, J. Fernández, J. D. Garcia, F. Garcia, and J. Carretero, "New techniques for simulating high performance mpi applications on large storage networks," *The Journal of Supercomputing*, vol. 51, no. 1, pp. 40–57, 2010.

[33] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message-passing interface*. MIT press, 1999, vol. 1.

[34] I. Sriram, "Speci, a simulation tool exploring cloud-scale data centres," in *Cloud Computing*. Springer, 2009, pp. 381–392.

[35] A. Buss, "Simkit: component based simulation modeling with simkit," in *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*. Winter Simulation Conference, 2002, pp. 243–249.

[36] A. Ferscha and S. K. Tripathi, "Parallel and distributed simulation of discrete event systems," 1998.

[37] J. S. Dahmann, R. M. Fujimoto, and R. M. Weatherly, "The dod high level architecture: an update," in *Simulation Conference Proceedings, 1998. Winter*, vol. 1. IEEE, 1998, pp. 797–804.

[38] P. Tim and F. Michael. The poRTIco Project. Accessed: 2016-05-22. [Online]. Available: http://www.porticoproject.org/

[39] Y. Pu, Y. Deng, and A. Nakao, "Cloud rack: Enhanced virtual topology migration approach with open vswitch," in *Information Networking (ICOIN), 2011 International Conference on*. IEEE, 2011, pp. 160–164.

[40] M. Rosenblum, "Vmwares virtual platform," in *Proceedings of hot chips*, vol. 1999, 1999, pp. 185–196.

[41] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[42] Centos image. Community ENTerprise Operating System. 2015. URL: http://openstack.redhat.com/.

[43] Fedora image. Fedora. 2015. URL: http://berrange.fedorapeople.org/images/.