Durham Research Online

Deposited in DRO:

28 April 2016

Version of attached file:

Accepted Version

Peer-review status of attached file:

Peer-reviewed

Citation for published item:

Abar, S. and Lemarinier, P. and Theodoropoulos, G. and OHare, G. (2014) 'Automated dynamic resource provisioning and monitoring in virtualized large-scale datacenter.', in 2014 IEEE 28th International Conference on Advanced Information Networking and Applications (AINA) : 13-16 May 2014, University of Victoria, Victoria, Canada. Piscataway: IEEE, pp. 961-970.

Further information on publisher's website:

Publisher's copyright statement:

Additional information:

Automated Dynamic Resource Provisioning and Monitoring in Virtualized Large-scale Datacenter

Sameera Abar
School of Computer Science and Informatics, UCD
IBM Dublin Research Laboratory, Ireland
sameera.abar@ucd.ie

Pierre Lemarinier
IBM Dublin Research Laboratory
Mulhuddart, Dublin 15, Ireland
pierrele@ie.ibm.com

Georgios K. Theodoropoulos
Institute of Advanced Research Computing
Durham University, United Kingdom
georgios.theodoropoulos@durham.ac.uk

Gregory M. P. O'Hare
School of Computer Science and Informatics
University College Dublin, Dublin 4, Ireland
gregory.ohare@ucd.ie

Abstract—**Infrastructure as a Service (IaaS) is a pay-as-you-go based cloud provision model which on demand outsources the physical servers, guest virtual machine (VM) instances, storage resources, and networking connections. This article reports the design and development of our proposed innovative symbiotic simulation based system to support the automated management of IaaS-based distributed virtualized datacenter. To make the ideas work in practice, we have implemented an OpenStack based open source cloud computing platform. A smart benchmarking application "Cloud Rapid Experimentation and Analysis Tool (aka CBTool)" is utilized to mark the resource allocation potential of our test cloud system. The real-time benchmarking metrics of cloud are fed to a distributed multi-agent based intelligence middleware layer. To optimally control the dynamic operation of prototype datacenter, we predefine some custom policies for VM provisioning and application performance profiling within a versatile cloud modeling and simulation toolkit "CloudSim". Both tools for our prototypes' implementation can scale up to thousands of VMs, therefore, our devised mechanism is highly scalable and flexibly be interpolated at large-scale level. Autonomic characteristics of agents aid in streamlining symbiosis among the simulation system and IaaS cloud in a closed feedback control loop. The practical worth and applicability of the multi-agent-based technology lies in the fact that this technique is inherently scalable hence can efficiently be implemented within the complex cloud computing environment. To demonstrate the efficacy of our approach, we have deployed an intelligible light-weight representative scenario in the context of monitoring and provisioning virtual machines within the test-bed. Experimental results indicate notable improvement in the resource provision profile of virtualized datacenter on incorporating our proposed strategy.**

Keywords: **Autonomic Computing, Symbiotic Simulation, Virtualized Datacenter, Resource Provisioning, Cloud Benchmarking, Multi-agent Technology**

I. RESEARCH OVERVIEW

Cloud computing technologies are gaining an increase in the acceptance from IT companies and institutions for use in the deployment of efficient, flexible, and scalable virtualized datacenters. Virtualization plays an important role as an enabling technology for datacenter implementation by abstracting compute, network, and storage service platforms from the underlying physical hardware. Virtualized infrastructures support server consolidation and on demand provisioning capabilities, which results in high server utilization rates and significant cost and energy savings [1]. The typical approach to datacenter infrastructure for cloud computing is to build a collection of general purpose commercial servers connected by standard local-area network switching technologies. On top of this, an Infrastructure-as-a-Service (IaaS) multi-tenancy usage model is constructed where users are given access to the systems capacity in the form of virtual servers and networks [2]. Virtualized servers run a diverse set of virtual machines (VMs). Virtual machine provisioning includes instantiation of one or more virtual machines each with different options of processors, memory, and I/O performances; resource provisioning deals with mapping and scheduling of VMs onto physical servers within the cloud; and application provisioning is the deployment of specialized applications within VMs and mapping of end user's requests to application instances. Machine provisioning is a central function in cloud-oriented environments, that is, the physical machines are brought online and new virtual machines are created on a very regular basis. It often happens that runtime images (the entire content of the disks) are cloned from an existing machine's templates repository. On demand nature of the IaaS clouds, enables to deploy and terminate hundreds of thousands of virtual machines (VMs) in a single large-scale cloud datacenter.

Secure, efficient, and scalable management of these virtualized infrastructures is essential to guarantee optimal functioning of a datacenter. Cost of cloud usage varies significantly depending on the dynamic resource configuration [3]. In this context, benchmarking tests serve as a standard or reference basis for the cloud systems' evaluation or

comparison purposes. With the increasing adoption of cloud computing infrastructures, there is a need of proper benchmarking techniques in order to assess the performance of cloud infrastructures and software stacks, to assist the users with provisioning decisions and to compare various available cloud offerings. Virtualized cloud services are inherently time-shared and may over-commit their resources. Moreover, ever-changing customer load may affect the performance of cloud infrastructures. With new benchmarking tools at their disposal, enterprises at least have a way to validate claims through experiment automation before cloud infrastructures launch [4].

IBM's initiative of autonomic computing refers to a computing environment with the ability to manage itself and dynamically adapt to change in accordance with embedded policies in runtime control loops [5]. While the software agent research area is very active, and it offers an appropriate tool to tackle the enormous challenges involved with the development of next-generation expert management systems, its concerns towards this domain have been so far not well covered. Currently, the multi-agent based applications incorporating the resource knowledge of the domain under consideration stay focused on the low intensity system design level. Motivated by these considerations to prove that these novel concepts are not just hype, the prime focus of this research proposal is to report, explore, and discuss the real-world practical implementation of a prototype system employing multi-agent technology for datacenter virtualization. Symbiotic simulation system is defined as one that interacts with a physical system in a mutually beneficial manner. The simulation is driven by the real-time data collected from a physical system. Basically, the simulation model needs to accurately imitate the characteristics of the physical system under consideration to enrich the real-system operation. In turn, the results of the conceptual reasoning or what-if analysis performed by the simulator are used to control the dynamic behavior of the actual system within various time constraints.

Key focus of our work is to integrate the autonomic computing and symbiotic simulation paradigms within the cloud computing domain. Based on this strategy, we design and develop an innovative symbiotic simulation based scalable prototype to support the autonomic resource allocation and management of IaaS-based distributed virtualized datacenter. The rest of paper is organized as follows. Section 2 highlights the comparison of our approach with other related works. Section 3 illustrates the conceptual design and functional architecture of our proposed approach. The details regarding prototype implementation and some use-cases within the experimental system are presented in Section 4. Finally, Section 5 concludes our proposed research methodology followed by the future work directions in Section 6.

II. COMPARISON TO OTHER APPROACHES

Scope of our research project revolves around the autonomic symbiotic simulation paradigm. The term symbiotic simulation was coined at the Dagstuhl Seminar on Grand Challenges for Modeling and Simulation by Fujimoto et al. in 2002 [6]. However, the concepts behind symbiotic simulation have been around for longer. In symbiotic simulation paradigm, the simulation model benefits from the continuous supply of the latest data and the automatic validation of its simulation outputs, whereas the physical system benefits from the improved performance obtained from the analysis of simulation experiments. In this context, the proposals [7], [8] have inspired us the most and closely related to our research direction. More specifically, what we are trying to bring about is the advanced practical manifestation of the very basic idea envisioned in [7] by Qi Liu et al. Primarily in this work, the authors put forward a multi-agent based symbiotic scheme for autonomic cloud management purposes. The system architecture consists of a low-level execution (physical) layer and a high-level management (simulation) layer sharing together the same virtual infrastructure. The daemons or agents associated with these layers, work in a seamless self-governing integrated mode to monitor and manage the resources of virtualized datacenter system. However, this work lacks the actualization or implementation details of the idea. In addition to this, [8] presents the hardware-in-the-loop simulation technique used in cloud benchmarking tool, which integrates an efficient discrete-event simulation with the cloud infrastructure under test in a closed feedback control loop. The experiments demonstrated that the technique can synthesize complex resource usage patterns for effective cloud performance benchmarking. In contrast our work aims to construct a practical software agent-based support mechanism for the dynamic resource provisioning and monitoring in virtualized large-scale datacenter systems and this effort can be considered as a first-ever step towards this direction. In addition to this, our literature survey explores some other recent related researches. In [9] Heiko Aydt et al. provide a general overview of the state of the art symbiotic simulation applications and discussed some of the most important research issues to be addressed in this regard. Symbiotic simulation is closely related to dynamic data-driven application systems (DDDAS), that entails the applications's ability to dynamically inject data into the execution system and steer the measurement process. In particular, while DDDAS is more generally concerned with data-driven applications, symbiotic simulation refers more specifically to real-time data-driven simulations about a physical system. Although these paradigms are not identical, they overlap in some respects. As a result, some symbiotic simulation systems can also be considered as a dynamic data-driven application systems and vice versa. Here, we mention a DDDAS considered as

Prototype System's Implementation Tools

Cloud Environment:
OpenStack Cloud Computing Software
Cloud Benchmarking Tool:
IBM's Cloud Rapid Experimentation and Analysis Tool (aka CBTool)
Cloud Modeling and Simulation Tool:
CloudSim-3.0.3
Agent Development Tool:
IDEA-1.3.1
Agent Base-processes:
Java-Development-Kit (JDK-7)
Bourne-Again-Shell (bash) scripts
Operating System:
Ubuntu Server 12.04.2 LTS

Open source IaaS cloud computing platform:
OpenStack deployed on Shamrock machine

Nova Compute Management API

Benchmarking Daemons

Cloud Benchmarking Tool
(CBTool — Python Scripts)

Benchmarking agents or daemon processes get attached with physical resources and virtual machine instances at bootstrap time, and interact with the real cloud network, collect system's performance metrics, and generate data into the real-time storage repository
Agents/daemons for benchmarking VM provision profiling, resource allocation and application performance monitoring

Extension of CloudBench's Daemons
Client-side scripting:
(Python, Java, Ruby, C++)
- Enhancing the functionality of existing benchmarking daemons
- Creating new daemons/agents
Daemons for detecting network traffic congestion, workload surges, storage availability, hardware failures, ...

Repository for run-time data metrics collection

Symbiotic Simulation Feedback Control Loop

Cloud Simulation Tool
(CloudSim — Java Programs)

Set of what-if rules as internal knowledge of simulation agents, specify the optimal behavior scenarios within the cloud infrastructure, thereby, replicating various services provided by the virtual layers of module-based management software
Admission control, Resource allocation, VM life cycle management, VM scheduling/provisioning, Load balancing, Monitoring billing/costs, Server utilization & power optimization, Quality-of-service guarantee, Intrusion detection, Performance-driven memory overcommitment, Failure/attack recovery, ...

Simulation Agents

Management processes conduct what-if simulations
[1] To supervise the autonomic actions carried out by benchmarking agents
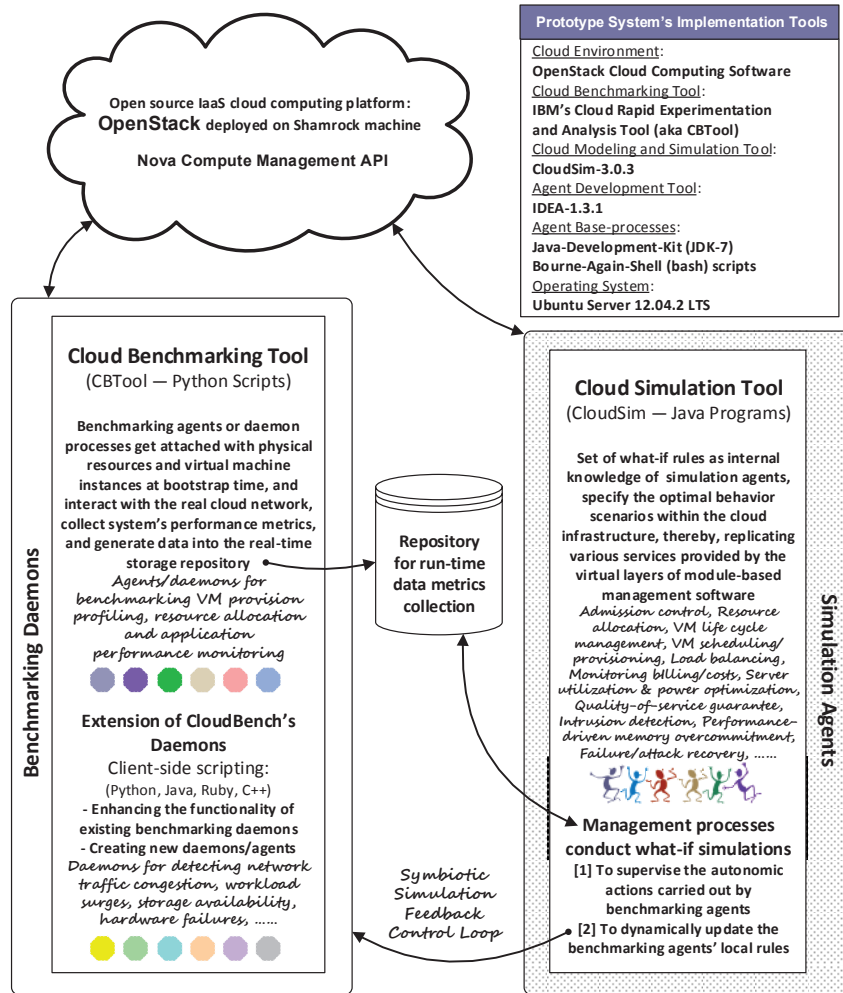[2] To dynamically update the benchmarking agents' local rules

Figure 1: Conceptual model representing our autonomic management support strategy for cloud computing based datacenter infrastructure.

a symbiotic simulation system having the ability to improve the reliability of decision support based simulations as described by Kennedy and Theodoropoulos [10]. Researchers and developers have extensively been investigating various application domains of symbiotic simulation paradigm. [11] demonstrates via experiments that agent technology in a symbiotic simulation loop can be used effectively to carry out the monitoring, optimization and control functions of the semiconductor assembly and test operation. Furthermore, [12] implements an advanced symbiotic simulation-based automation system in decision making and problem solving regarding the operations of the various tools of an entire semiconductor manufacturing plant. The simulation system uses what-if experiments to investigate alternative decision making scenarios and the best decision is determined by analysing the simulation results. [13] shed light on the use of what-if reasoning strategy in conceptual simulation during data analysis processes. The applications areas of symbiotic simulation to assist in real-time decision support mechanism

are wide-spread; for instance: business process optimization and control, emergency response installations, medical and health-care systems, symbiosis phenomena in Biology, unmanned aerial vehicles planning, model validation for weather forecasting and radiation tracking, scheduling and supply chain domain, etc.

III. HOW DOES THE SYSTEM WORK?

Fig. 1 illustrates the conceptual model of our autonomic management support strategy for cloud computing based datacenter infrastructure. As shown, the proposed model consists of cloud benchmarking layer and multi-agent based simulation zone. To make the ideas work in practice, we have implemented an OpenStack [14] based open source cloud computing platform which controls large pools of compute, storage, and networking resources throughout a virtualized datacenter. OpenStack cloud for public and private is simple to implement, and is massively scalable and feature rich. OpenStack Management Software "Nova" comprises of a cluster of modules: API daemons; compute daemons for
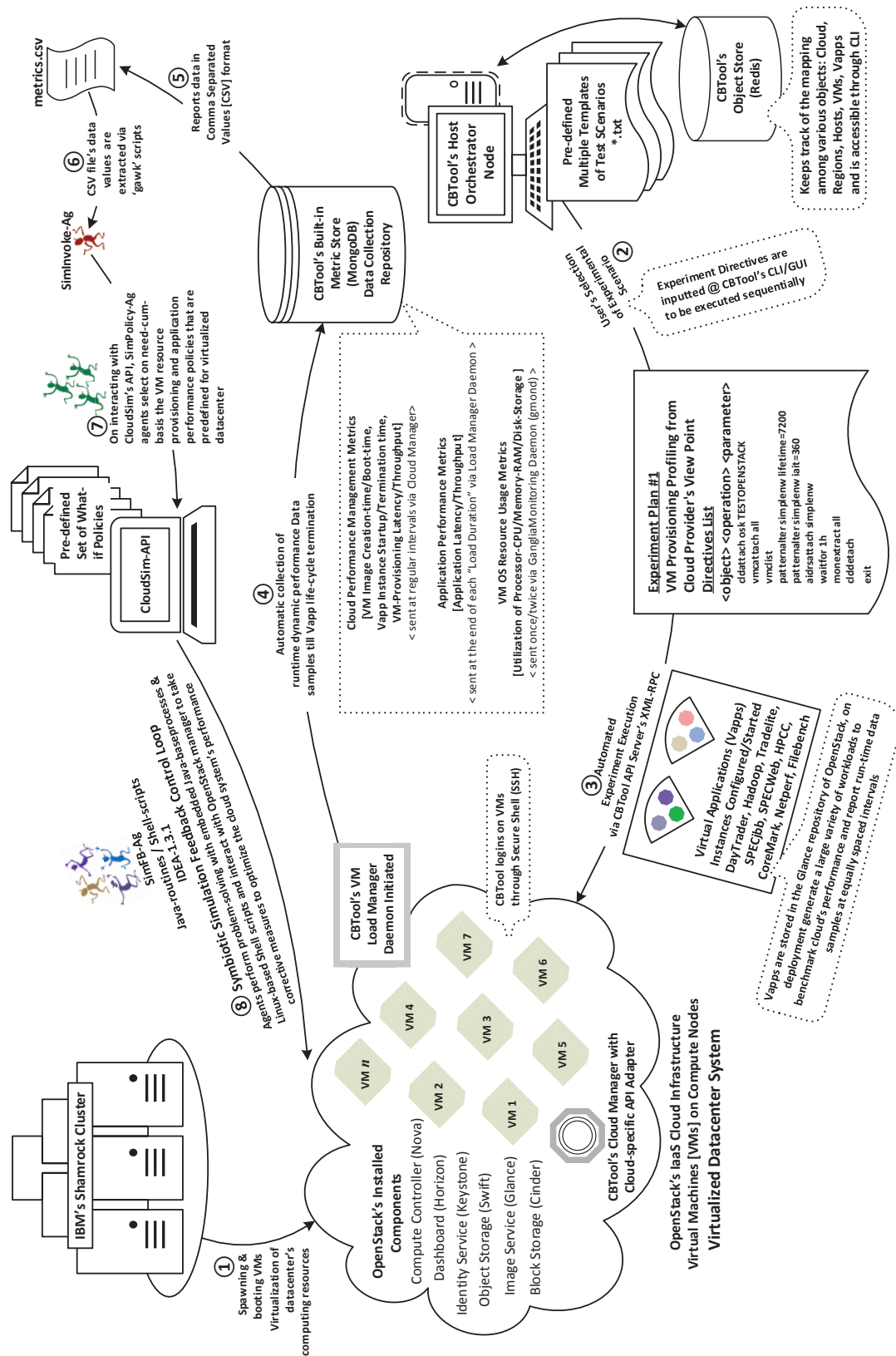
Figure 2: Functional architecture of the proposed automated dynamic resource provisioning and monitoring mechanism for virtualized large-scale datacenter.

creating and terminating VMs; volume daemons for creating, attaching, and detaching persistent storages to VMs; and network daemons for managing network connections. Next, we discuss the tools utilized in implementing our prototype system.

A. Cloud Benchmarking

It is required to automate the evaluation and benchmarking of almost every observable interaction of the IaaS cloud and applications, load generation, resource usage, and failure behavior. The benchmarking layer is constructed in "Cloud Rapid Experimentation and Analysis Tool (aka CBTool)" [15] designed and developed at The IBM Thomas J. Watson Research Center. This is an open-source framework that automates cloud-scale evaluation and benchmarking through the running of controlled experiments, where multiple multi-tier applications are automatically deployed without any human intervention. Experiments can be executed in multiple clouds using a single orchestrator interface. CBTool is capable of managing experiments spread across multiple regions and for prolonged periods of time. This application has plugin adapters (for instance, Simulated Cloud, OpenStack, Amazon Elastic Compute Cloud, VMware vCloud Director, CloudStack, IBM Smart Cloud Provisioning, Parallel Libvirt Manager) available, with support to add multiple cloud models incrementally. A built-in data collection system collects, aggregates and stores metrics for cloud management activities, for instance: VM provisioning and application runtime information. The benchmarking daemon processes (Python-scripts for VM provision profiling, resource allocation and application performance monitoring) are attached to physical resources (compute servers, network switches, storage servers) and virtual applications. Daemons interact with the real cloud network, collect runtime application and system (OS) performance data from hosts and guests, and inject the metrics into the real-time storage repository. Specialized fresh daemons can be constructed and/or the functionality of existing benchmarking daemons can readily be enhanced for detecting network traffic congestion, workload surges, storage availability, and hardware failures [16]. Basically, a benchmarking application marks the resource provisioning and application performance profile of the cloud system. In our research project, we have utilized a benchmarking tool to obtain runtime metrics for injecting to a simulation system via multi-agent based middleware.

B. Cloud Simulation

The recent efforts to design and develop cloud technologies focus on defining novel methods, policies and mechanisms for efficiently managing cloud infrastructures. To test these newly developed methods and policies, researchers need tools that allow them to evaluate the hypothesis prior to a real deployment in an environment, where one can reproduce tests. Simulation-based approaches in evaluating cloud computing systems and application behaviors offer significant benefits, as they allow Cloud developers: (i) to test performance of their provisioning and service delivery policies in a repeatable and controllable environment free of cost, and (ii) to tune the performance bottlenecks before real-world deployment on commercial clouds. To meet these requirements, a research group at The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne has developed "CloudSim" [17], a Java-based toolkit for modeling and simulating extensible clouds. As a completely customizable tool, it allows extension and definition of policies in all the components of the software stack thereby making it a suitable research tool that can handle the complexities arising from simulated environments [18]. Our intent is to exploit some core features of CloudSim system, that is, the support for modeling and simulating virtualized computing nodes in datacenters and policies for service brokers, VM and application provisioning and resource allocations. CloudSim supports large-scale simulation environment with little or no initialization overhead and memory consumption. This novel tool also presents requirements of elastic applications that need to scale across multiple, geographically distributed data centers that are owned by one or more cloud service providers. We need to formulate a set of a-priori optimal behavior custom policies or what-if pilot-cases (such as admission control, resource allocation, VM life cycle management, VM scheduling/provisioning, load balancing, monitoring billing/costs, server utilization & power optimization, quality-of-service guarantee, intrusion detection, performance-driven memory overcommitment, failure/attack recovery) for desirable systems' behavior. The CloudSim's API (Application Programming Interface API) is configured within the agent construction tool IDEA-1.3.1 [19]. We have successfully been using this tool for developing various practical multi-agent based prototypes. Fig. 2 represents the functional architecture of the proposed mechanism for automated dynamic resource provisioning and monitoring in virtualized large-scale datacenter. Here, a detailed flow of VM provisioning based experiment execution depicts real-time benchmarking metrics of cloud being fed to a distributed multi-agent based intelligence middleware layer, in conjunction with associated Java-routines and Bourne-Again-Shell scripts. The practical worth and applicability of the proposed multi-agent-based approach lies in the fact that this technique is inherently scalable hence can efficiently be implemented within the complex cloud computing environment. Autonomic characteristics of agents aid in streamlining symbiosis among the simulation system and IaaS cloud in a closed feedback control loop. A unique characteristic of the multi-agent system is the flexible adaptation of its services to the dynamically changing cloud computing environment. Once the agents interact and cooperate with each other during the complex problem solving, reasoning, and

Everything | Application Performance | HOST Performance | VM Performance | Provisioning Performance

name	size	role	type	cloud ip	age	state	vmc name	host name	ai name	aidrs name	Step 1: provisioning request originated	Step 2: provisioning request sent	Step 3: provisioning request completed	Step 4: network acessible	Step 5: file transfer	Step 6: application start	Step 999: provisioning request failed
											epoch	secs	secs	secs	secs	secs	secs
vm 1	m1.micro	tinyvm	nullworkload	undefined	3880	None	RegionOne	--	ai 1	aidrs 1	1382917336	--	--	--	--	--	0
vm 8	m1.micro	tinyvm	nullworkload	10.0.0.8	1354	None	RegionOne	unknown	ai 8	aidrs 1	1382919862	0	26	--	--	--	320
vm 9	m1.micro	tinyvm	nullworkload	10.0.0.8	994	None	RegionOne	unknown	ai 9	aidrs 1	1382920222	1	26	--	--	--	321
vm 10	m1.micro	tinyvm	nullworkload	10.0.0.8	633	None	RegionOne	unknown	ai 10	aidrs 1	1382920583	1	26	--	--	--	321
vm 2	m1.micro	tinyvm	nullworkload	10.0.0.2	3519	attached	RegionOne	unknown	ai 2	aidrs 1	1382917697	0	21	48	40	6	--
vm 3	m1.micro	tinyvm	nullworkload	10.0.0.3	3158	attached	RegionOne	unknown	ai 3	aidrs 1	1382918058	0	21	71	41	6	--
vm 4	m1.micro	tinyvm	nullworkload	10.0.0.4	2798	attached	RegionOne	unknown	ai 4	aidrs 1	1382918418	1	21	101	69	7	--
vm 5	m1.micro	tinyvm	nullworkload	10.0.0.5	2437	attached	RegionOne	unknown	ai 5	aidrs 1	1382918779	1	21	165	107	8	--
vm 6	m1.micro	tinyvm	nullworkload	10.0.0.6	2076	attached	RegionOne	unknown	ai 6	aidrs 1	1382919140	1	26	223	180	9	--
vm 7	m1.micro	tinyvm	nullworkload	10.0.0.7	1715	attached	RegionOne	unknown	ai 7	aidrs 1	1382919501	0	26	288	260	11	--

Figure 3: Provisioning status of virtual machines on the Web-based dashboard of CBTool.
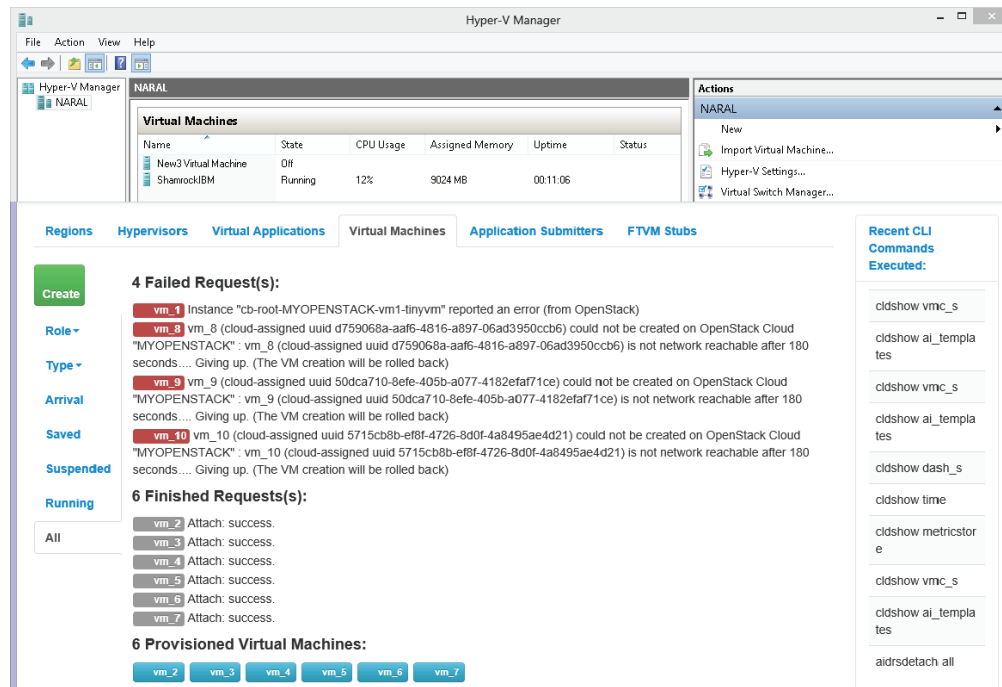
Figure 4: Error information of virtual machines failed to provision.

decision making sessions to resolve a particular management issue within the computing system; the integrated response is reported to the systems interface and a part of the embedded control knowledge modifies/updates dynamically in their working memory area according to the runtime conditions. This demonstrates how we have utilized a simulation tool to control the dynamic behavior of a test cloud system.

IV. EXPERIMENTAL TESTBED

Our test set-up comprises of a core virtual environment configured within Hyper-V (Microsoft Windows client virtualization Hypervisor technology) which multiplexes access to physical resources. Hyper-V lets run multiple 32-bit or 64-bit x86 guest operating systems (OS) at the same time on a single computing node inside a virtual machine, thereby maintaining isolation between VMs at all times. Each active virtual machine instance requires its own memory and storage. This nested virtualization support creates a new abstraction for cloud users where the VM format becomes irrelevant to the cloud. The hardware specifications of our host machine are: Processor: Intel (R) Core (TM) i7-3537U 86-64 CPU@2.00GHz with Second Level Address Translation (SLAT) support and BIOS-level

Table I: Provisioning status of virtual machines versus preset arrival-rates.

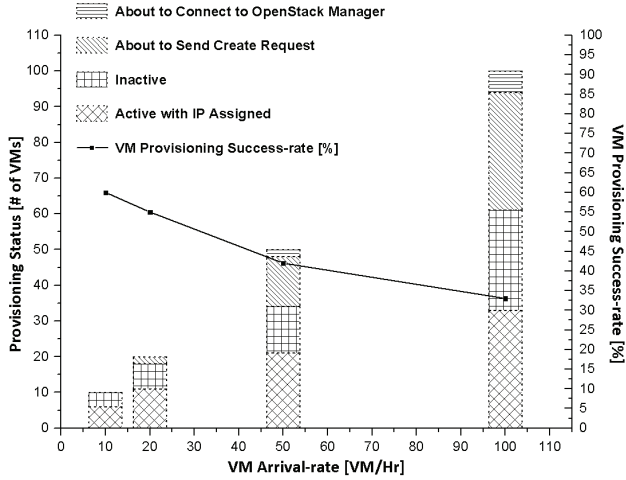| VM Arrival-rate [VM/Hr]	VM Provisioning Status [# of VMS]				VM Provisioning Success-rate [%]
	Active with IP Assigned	Inactive	About to Send Create Request	About to Connect to OpenStack Manager	
10	6	4	0	0	60
20	11	7	2	0	55
50	21	13	14	2	42
100	33	28	33	6	33

Figure 5: Provisioning status of virtual machines versus preset arrival-rates.

hardware-assisted virtualization option enabled for Intel VT or AMD-Virtualization; virtual disk storage: 200GB; and main virtual memory (RAM): 9GB maximum plus 9GB for the Hyper-V's parent or host OS Windows8.0 Pro (64-bit). The guest operating systems: 64-bit Ubuntu12.04 LTS with the simulation engine of CloudSim-3.0.3 & agent system IDEA-1.3.1 configured on the node. We have deployed an OpenStack cloud environment having Keystone identity service and Horizon dashboard on the controller node with 2 vCPUs and compute services Nova API, Scheduler, Glance VM-image registry, Swift object store, Neutron networking, and Cinder volume service.

Scenario-1: Benchmarking VM Provision Profile
:: From Cloud Provider's View Point ::
In this case, a cloud services provider wants to profile the provisioning baseline performance/scalability status of a virtualized infrastructure. He needs to strategically plan the critical management challenge about how efficiently the virtual machines are provisioned as the virtualized environment grows. The intent is to measure the clouds capability to handle excessive number of simultaneous provisioning requests and provide high VM arrival-rates. For this scenario execution, Vapps under variable inter-arrival times are continuously spawned on the test cloud. A special Vapp "SimpleNullWorkLoad" [simplenw] is deployed for assessing clouds provisioning capacity. Here, 'NullWorkload' implies no load generation after a VM has been provisioned. For

testing cloud scalability, cloud's provider extracts provisioning latency and throughput as a function of VMs arrival-rate [i.e., # of VMs/Hour]. Provisioning latency indicates how much time elapses between the arrival of a request to create a new instance-VM and the availability of such VM for use. Whereas provisioning throughput indicates how many simultaneous provisioning requests can the cloud handle. We configured CBTool's orchestrator node on our cloud and used a standard KVM test image "tinyvm" [size: 2GB, memory: 192MB, OS: Ubuntu11.10 64-bit] in "qcow2" format for instantiation. CBTool can orchestrate different arrival rates and lifetimes for VMs using probability distributions. Initially, we conducted a very simple experiment with a fixed arrival-rate of 10 VM/Hr. Spawning a new instance involves complex interaction between multiple components inside OpenStack. Due to the storage and memory limitations of our OpenStack cloud, we select the m1.micro flavor for cloning VM instances. In this particular scenario, the use of a small VM image proved to be convenient. Our ultimate goal is to be able to collect performance data in a timely and reliable manner. **Inputting the experiment's directives:**[1]

- cldattach osk MYOPENSTACK (Attaches OpenStack cloud to CBTool's orchestrator node)
- vmcattach all (Registers a new Virtual Machine Container for a given cloud)
- vmclist (List all VMCs or regions currently attached to CBTool's experiment)
- patternalter simplenw lifetime=7200
- patternalter simplenw iait=360 (Inter-arrival parameter "iait" of "simplenw" deploys one instance of virtual application "nullworkload" every 360 seconds, i.e., 10 VMs/Hr. For consecutive VMs arrive at constant interval, the throughput (VMs/Hr) is just the inverse of the inter-ai-time)
- aidrsattach simplenw (Attaches Vapp Submitter "simplenw" which keeps on deploying null-workload virtual applications)
- waitfor 1h (simplenw runs for 1 hour; the 'waitfor' operation pauses the execution of new directives for a specified period of time)
- aidrsdetach all (All previously created application daemons are terminated to prevent run-up costs i.e., cloud usage billing by unit of time)
- waituntil AI ARRIVING=0 (Wait until the # of approaching applications becomes zero)
- aidetach all (Disconnects application instances)
- monextract all (A set of provisioning runtime performance data is extracted)
- clddetach
- exit (Cloud disconnects and cleans up all VM objects and daemons)

On executing the experiment, the provisioning trend of

---

[1]Provisioning template designed by the creators of CBTool.

CBTool

Experiment Plan #1
VM Provision Profiling from
Cloud Provider's View Point
Directives List
<object> <operation>

    cldattach osk TESTOPENSTACK
    vmcattach all
    vmclist
    patternalter simplenw lifetime=7200
    patternalter simplenw iait=360
    aidrsattach simplenw
    waitfor 1h
    monextract all
    Clddetach
    exit

metrics.csv

VMs reprovisioned/migrated on another available host or region
IBM@shamrock-Virtual-Machine:~/OpenStack$ nova list

```
+--------------------------------------+-------------+--------+------------+-------------+------------------+
| ID                                   | Name        | Status | Task State | Power State | Networks         |
+--------------------------------------+-------------+--------+------------+-------------+------------------+
| 7d6cdbd1-00cc-46e8-850e-c2d150b86585 | vm_1-tinyvm | ACTIVE | None       | Running     | private=10.0.0.2 |
| 3d3dd8e7-25ad-485b-8181-d6ba76df91f0 | vm_8-tinyvm | ACTIVE | None       | Running     | private=10.0.0.5 |
| 4e21dada-8475-45b6-837a-b10e9645e8e6 | vm_9-tinyvm | ACTIVE | None       | Running     | private=10.0.0.8 |
| c2a51822-ab1b-413f-b25f-a512e1bb5b3d | vm_10-tinyvm| ACTIVE | None       | Running     | private=10.0.0.9 |
+--------------------------------------+-------------+--------+------------+-------------+------------------+
```

Extracting a selected set of metrics (via 'gawk' utility) for VMs which haven't been provisioned

ProvisioningMetrics-Errors.csv

uuid	vm_name	last_known_state	vmc_name
d5112260-9001-5497-b43e-464bcabd542a	vm_1	Inactive	RegionOne
d759068a-aaf6-4816-a897-06ad3950ccb6	vm_8	About to Send Create Request	RegionOne
50dca710-8efe-405b-a077-4182efaf71ce	vm_9	About to Send Create Request	RegionOne
5715cb8b-ef8f-4726-8d0f-4a8495ae4d21	vm_10	About to Connect to Openstack Manager	RegionOne

OpenStack Manager
Nova API's
Management Stack

Cluster of components: API daemons; compute daemons for creating and terminating virtual machines; volume daemons for creating, attaching, and detaching persistent storages to VMs; network daemons for managing network connections

Compute Node

Runtime error information is matched with a predefined string or precondition. and an event is generated

SimInvoke-Ag

Integrating CloudSim-API with IDEA: agent design system

Simulation tool controls the dynamic behavior of test cloud system

CloudSim

Simulation Model for VM Provision Profiling
// CloudSimVMProvisionPolicies.java
// A Java-baseprosses created in a cloudSim system
// for improving the cloud's provisioning operation
1. import packages (org.cloudbus.cloudsim.*)
2. Initialize the CloudSim library
3. Input:
        uuid   //VM's universally unique identifier
        vm_name   //VM's id
        last_known_state   //VM's provisioning status
        vmc_name   //Openstack's region
4. Create classes
        Datacenter   //Encapsulates a set of compute virtualized hosts which provides resources
        DatacenterBroker //Rrepresents a broker acting on behalf of a user
        Host   //Models a physical resource such as server
        Vm   //Virtual machine running inside a host
5. for (each vm_name) do
        if (last_known_state = last_known_state) then
            vmDestroy (vm vm_name); //Destroys VM running in the host
        end if
6. Create virtual machines
        Vm vm_n = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared()); //Creates VM objects specific characteristics
        vmlist.add(vm_n); //Adding VMs to vmList
        broker.submitVmList(vmlist); //Submitting vmList to broker
7. Create custom user broker-policies to submit vms according to the specific rules of our provisioning scenario
        VmAllocationPolicy(hostList) //Provisioning policy of virtual machines to hosts in
                                      //a datacenter
        allocateHostForVm(Vm vm, Host host) //Allocates a specified host for a given VM
8. end for

Computational Baseprocess such as CloudSim's Java program is wrapped by the knowledge contained in the agent

Reprovision missed VMs on another host or region or compute node

Clearing cache of inactive and delayed VMs

SimPolicy-Ag with CloudSim's Java compiled class-files

SimFB-Ag Java-routines & Shell-scripts

```
#! /bin/sh

# Launch the VM instances

#$ nova boot --image <image_id> --flavor <flavor_id> --key_name
<keypair_name> --security_group <sec_group> <server_name>

nova boot --image cloudbench_tinyvm --flavor m1.tiny --key_name
openstackkey --security_group default vm_n-tinyvm

echo "VMs reprovisioned successfully !"

# ssh  -i klab_id_rsa -l klabuser vm_n-tinyvm@10.0.0.n
```

Linux-based Shell script
NovaBoot.sh

A part of the embedded control knowledge of agents modifies/updates dynamically in the systems' working memory according to runtime conditions
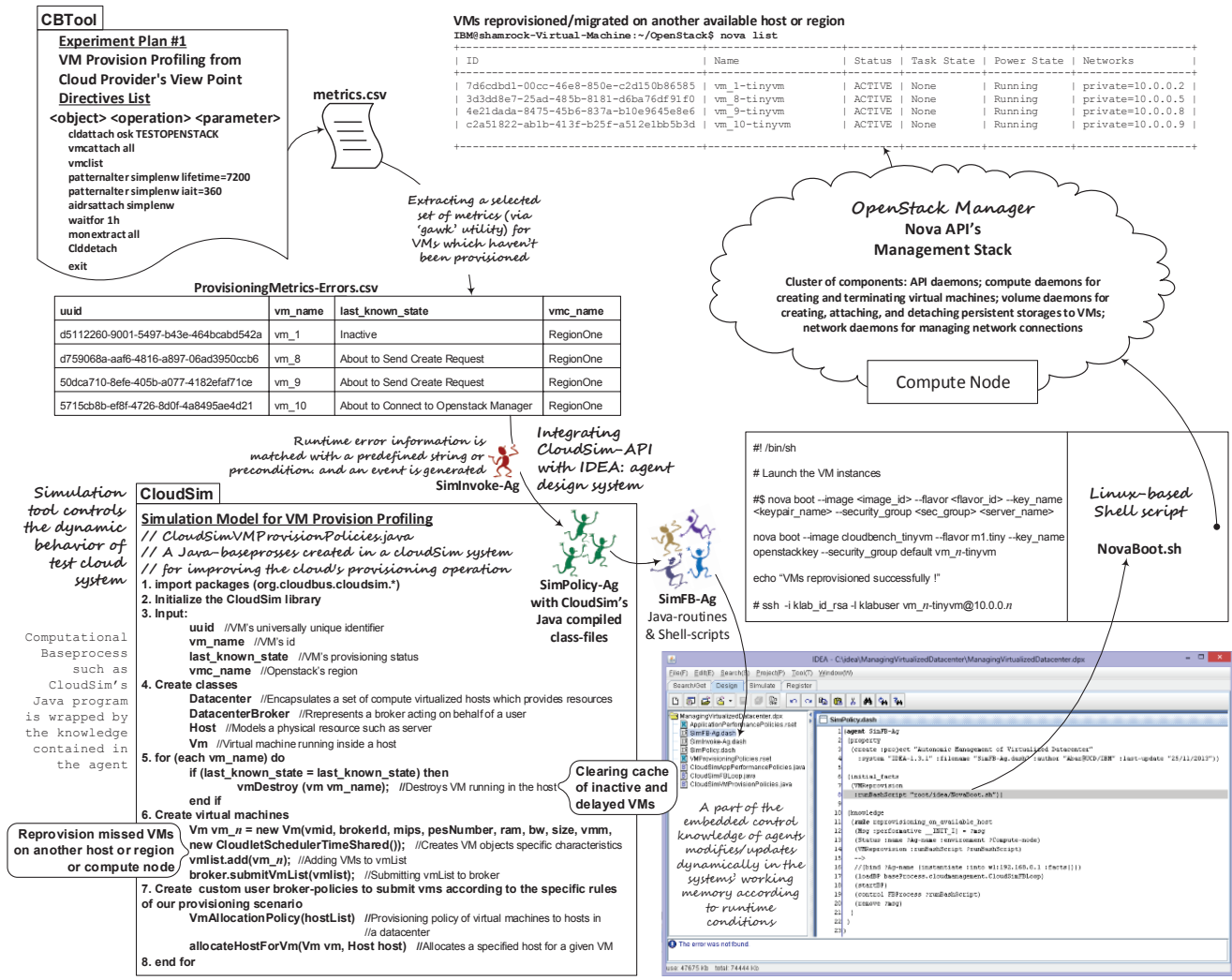
The error was not found

Figure 6: Execution of Scenario-1 to reprovision inactive/delayed VMs on another host

virtual machines on the graphical interface of cloud benchmarking tool is reflected in Fig. 3. Here, the problem areas in provisioning actually indicate resource bottlenecks. Fig. 4 pinpoints the detailed provisioning failure information of the virtual machines within the 10VM/Hr arrival-rate slot. As reported, vm_1 crashes due to direct boot error from OpenStack. While at times, the rollback or revert in VM creation occurs due to OpenStacks' network unreachable condition and difficulty in establishing SSH connection, while other errors in VM provisioning result due to overlap in processing excessive concurrent requests of VM instantiation. It has been observed that the management metrics vary largely with the increases in number of active/running VMs thereby dropping provisioning throughput on higher resource utilization. Moreover, VM image types and size affects the

provision latency or response-time. We repeated the whole process with specified inter-arrival-time "iait" values of 180, 72, 36 corresponding to throughput 20, 50, 100 VMs/Hr respectively. The results indicate that VM provisioning efficiency or success-rate of our cloud decreases dramatically on narrowing the inter-arrival interval between two successively approaching VMs. Table 1 and Fig. 5 display the provisioning status (primarily active/inactive) of virtual machines versus fixed arrival-rates (10, 20, 50, 100 VMs per hour). The performance log is exported in comma-separated-value (CSV) file-format and parsed through "gawk" which is a splendid Linux utility to slice logs and extract the selected records to analyze. Fig. 6 demonstrates the high-level abstracted execution of provisioning scenario-1. As shown, a selected data sample (containing error information of VMs

which could not be provisioned on demand mainly due to host being overloaded) is extracted from the CSV metrics file. Once the runtime information is matched with a precondition set in the "SimInvoke-Ag" agent, simulation is initialized inputting data values to the simulation model (a Java-based routine created in CloudSim system) for optimizing VM provisioning operation and maximizing the performance of virtualized datacenter. Corresponding to each input value, an appropriate set of managed what-if policies (according to the specific rules of our provisioning activity) is selected and invoked by the "SimPolicy-Ag" agents. This interaction has been made possible by embedding the CloudSim simulation engine's API within the agent design system. SimFB-Ag agent interacts directly with the OpenStack's API via Java-baseprocesses & Linux-based Bash (Shell) scripts to take corrective measures to optimize the cloud systems performance. Agents rule invocation along with the embedded problem-solving computational algorithms and runtime metrics, perform the automated dynamic resource management support. In this particular scenario, "VmAllocationPolicy()" instantiates the same number of inactive and delayed VMs on another resource rich host or region (compute-node). This simulation model can flexibly be extended with policies for migrating VMs, allocating bandwidth/memory/storage, and optimizing cloudlets (applications) performance.

Scenario-2: Benchmarking the Application Performance Profile

:: From Cloud Customer's View Point ::

This a case where a customer searches for a provider to outsource the execution of its own applications of interest and wants to assess the application performance profile of a cloud. To this end, the prospective client might opt to launch on a cloud a mix set of virtual benchmarking applications (DayTrader, Tradelite, Hadoop, SPECjbb, LOST, NETPERF, Coremark, Filebench) versus complex workloads to emulate the actual response in a simple and convenient manner. Each Vapp has specific uniform distribution for load level and a fixed value for load duration. We actualized this scenario by configuring a CBTool's orchestrator node on our default OpenStack cloud and used a standard KVM test image "cloudbench_hibench_daytrader" [size: 29GB, memory: 2GB, OS: RHEL6.1 64-bit] in "qcow2" format for instantiation. Initially, we conducted an experiment plan (designed by the CBTool's development group) by deploying a single DayTrader VApp on OpenStack, varied the load level with a uniform distribution of load values between 1 and 20 (simultaneous clients), and all applications are left to run for 5 hours while the load level continues to vary on each 90 second intervals. Finally the applications are removed from the test.

Inputting the experiment's directives:

- cldattach osk MYOPENSTACK
- vmcattach all
- vmclist
- typealter daytrader load_level=uniformIXIXI1I20
- typealter daytrader load_duration=90
- aiattach daytrader
- waitfor 5h
- aidetach all
- monextract VM app
- monextract VM os
- clddetach,
- exit

At the end of test execution, the DayTrader benchmarks produce a set of runtime virtual application performance evaluation metrics (throughput and latency) per VM as a function of the load level and tags it with a load identifier. It is seen that application metrics vary with increases in the number of running VMs and load intensity submitted to the applications.

V. Concluding Remarks

Virtualized datacenters are massively distributed cloud computing infrastructures. The customers acquire scalable compute and storage resources from diverse cloud offerings in the form of virtual machines to run their own applications inside VMs with pricing amenable to the requisite workload execution. Adequate monitoring and provisioning of dynamic resources within the complex cloud computing environments poses many distinctive challenges for cloud service providers. In this paper, we have argued that virtualization-based environments are an important application area for autonomic computing, but still in its infancy at the moment. There is a need of a workable autonomic management support solution to cope with the issues of cloud provisioning and application performance profiling. In this regard, our key contribution is the development of an innovative symbiotic simulation based strategy to support the automated management of IaaS-based next-generation virtualized datacenter. For the proof of concept, we have implemented an OpenStack based practical cloud computing prototype with integrated automation support of performance benchmarking. A set of a-priori custom policies for desirable virtual machines provisioning and application performance behavior are formulated within a cloud modeling and simulation toolkit. The real-time metrics of running cloud are injected in to the simulation system via multi-agent based middleware in a closed feedback control loop. Owing to the self-governing nature of software agents, the active utilization of the dynamic status information of the large-scale, heterogeneous, and complex environments has become a reality. The research outcome is an enriched symbiotic environment in which real and simulated cloud computing entities are seamlessly interwoven in an autonomic manner. The tools we utilized for the implementation of prototype system can scale up to thousands of virtual machines thus making our devised mechanism highly scalable and parallel. To demonstrate the effectiveness of our approach, we have

deployed a simple and light-weight yet clear representative scenario for monitoring and provisioning virtual machine assets within our test-bed. On executing experiment, at times VM creation reverts or rollbacks due to simultaneous multiple provisions causing an increase in the provision response time, however, the test results confirm significant improvement in the resource provision profile of virtualized datacenter on incorporating our proposed scheme.

VI. OUTLOOK

At present, our pragmatic analysis of VM application performance measurement with compute intensive workloads is underway. Future work involves experimenting with a few more working scenarios to confirm the feasibility of proposed approach. In this regard, we intend to implement clouds' power management scenario using Host/VM operating system related performance data samples. Additionally, customizable datacenter broker policies need to be defined in-line with these specific scenarios. Another essential aspect is the creation of associated shell scripts. Moreover, we aspire to craft an energy-conscious green computing policy scenario. The design of proposed prototype integrates various software modules hence our current efforts include ensuring firm coupling among the system's components. Cloud applications and services for large-scale data centers can be deployed on the supercomputing systems. We strive to evaluate the scalability of the proposed concept towards managing multi-dimensional resources of the virtualized datacanters that are built on exceptionally high-end machines. We also aim to look into some advanced machine learning technique to efficiently extract values of interest from runtime benchmarking metrics for more realistic application scenarios at larger scale. So far, we have utilized only the available daemon processes for experimentation. Currently we are working to upgrade the functionality of existing ones as well as programming Java/Python-based daemons from scratch for implementing new use-cases. Finally we anticipate to explore the customer/provider's service level agreement oriented resource management solutions.

ACKNOWLEDGMENT

REFERENCES

[1] R. Moreno-Vozmediano, R. S. Montero and I. M. Llorente, "IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures," *IEEE Computer*, vol. 45, no. 12, pp.65-72, December 2012.

[2] J. Appavoo, A. Waterland, D. Da Silva, V. Uhlig, B. Rosenburg, E. Van Hensbergen, J. Stoess, R. Wisniewski and U. Steinberg, "Providing a cloud network infrastructure on a supercomputer," *In proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC)*, ACM, NY, USA, pp.385-394, 2010.

[3] P. Goncalves, S. Roy, T. Begin and P. Loiseau, "Dynamic Resource Management in Clouds: A Probabilistic Approach," *IEICE Transactions on Communications: Special Section on Networking Technologies for Cloud Services*, vol.E95-B, no.8, pp.2522-2529, August 2012.

[4] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl and C. Tosun, "Benchmarking in the Cloud: What It Should," Can, and Cannot Be, *Selected Topics in Performance Evaluation and Benchmarking: Lecture Notes in Computer Science*, Springer-Verlag, Berlin, vol. 7755, pp.173-188, 2013.

[5] R. Murch, "Autonomic Computing," *On Demand Series: IBM Press*, 2004.

[6] R. Fujimoto, D. Lunceford and E. Page, "A.M.U.: Grand challenges for modeling and simulation," *Dagstuhl Technical Report 350*, Schloss Dagstuhl. Seminar 02351, August 2002.

[7] Q. Liu, D. Da Silva, G. K. Theodoropoulos and E. S. Liu, "Towards an agent-based symbiotic architecture for autonomic management of virtualized data centers," *In proceedings of the Winter Simulation Conference (WSC)*, Berlin, Germany, pp.1-13, December 2012.

[8] Q. Liu, M.A. Silva, M.R. Hines and D. Da Silva, "Hardware-in-the-loop simulation for automated benchmarking of cloud infrastructures," *Proceedings of the Winter Simulation Conference (WSC)*, pp.1-12, December 2012.

[9] H. Aydt, S. J. Turner, W. Cai, M. Y. Hean Low, "Research issues in symbiotic simulation," *In proceedings of the Winter Simulation Conference (WSC)*, pp.1213-1222, 2009.

[10] C. Kennedy and G. K. Theodoropoulos, "Intelligent management of data driven simulations to support model building in the social sciences," *In proceedings of the International Conference on Computational Science*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 3993, pp.562-569, 2006.

[11] M. Y. Hean Low, L. K. Wei, P. Lendermann, S. J. Turner, S. H. Leo and R. Chim, "An Agent-based Approach for Managing Symbiotic Simulation of Semiconductor Assembly and Test Operations," *In proceedings of the International Conference on Autonomous Agent and Multiagent Systems (AAMAS)*, Utrecht, The Netherland, pp.85-92, July 2005.

[12] H. Aydt, S. J. Turner, W. Cai and B. Ping Gan, "Symbiotic simulation for optimisation of tool operations in semiconductor manufacturing," *In proceedings of the Winter Simulation Conference (WSC)*, pp.2088-2099, 2011.

[13] S. B. Trickett and J. G. Trafton, What if: The Use of Conceptual Simulations in Scientific Reasoning," *Journal of Cognitive Science*, John Wiley and Sons, vol. 31, no. 5, pp.843-875, 2007.

[14] http://www.openstack.org/

[15] https://github.com/ibmcb/cbtool/

[16] M. Silva, M. R. Hines, D. Gallo, Q. Liu, K. D. Ryu, and D. da Silva, "CloudBench: Experiment Automation for Cloud Environments," *In proceedings of the IEEE International Conference on Cloud Engineering*, pp.302-311, March 2013.

[17] https://code.google.com/p/cloudsim/

[18] R. N. Calheiros, R. Ranjan, A. Beloglazov, Cesar A. F. 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 (SPE)*, Wiley Press, New York, USA, vol. 41, no. 1, pp.23-50, January 2011.

[19] http://www.k.riec.tohoku.ac.jp/s/idea/index.html