PerficientCloudSim: a tool to simulate large-scale computation in heterogeneous clouds

Muhammad Zakarya[a,b,*], Lee Gillam[a,*], Ayaz Ali Khan[b], Izaz Ur Rahman[b]

[a]Department of Computer Science, University of Surrey, UK
[b]Department of Computer Science, Abdul Wali Khan University, Pakistan
{mohd.zakarya,izaz,ayazali}@awkum.edu.pk,l.gillam@surrey.ac.uk

Abstract. The major reason for using a simulator, instead of a real test-bed, is to enable repeatable evaluation of large-scale cloud systems. CloudSim, the most widely used simulator, enables users to implement resource provisioning, and management policies. However, CloudSim does not provide support for: (i) interactive on-line services; (ii) platform heterogeneities; (iii) virtual machine (VM) migration modelling; and (iv) other essential models to abstract a real datacenter. This paper describes modifications needed in the classical CloudSim to support realistic experimentations that closely match experimental outcomes in a real system. We extend, and partially re-factor CloudSim to "PerficientCloudSim" in order to provide support for large-scale computation over heterogeneous resources. In the classical CloudSim, we add several classes for workload performance variations due to: (a) CPU heterogeneities; (b) resource contention; and (c) service migration. Through plausible assumptions, our empirical evaluation, using real workload traces from Google and Microsoft Azure clusters, demonstrates that "PerficientCloudSim" can reasonably simulate large-scale heterogeneous datacenters in respect of resource allocation and migration policies, resource contention, and platform heterogeneities. We discuss statistical methods to measure the accuracy of the simulated outcomes.

Keywords: Clouds, Datacenters, Simulations, Modelling, Performance, Heterogeneity

1 Introduction

Quantifying the performance of resource provisioning policies in a real cloud platform for different workload models under transient conditions is challenging due to, at least, three reasons: (i) clouds exhibit varying demands, system sizes and hardware resources; (ii) cloud users have heterogeneous and competing QoS requirements; and (iii) workloads have varying performance needs [1]. Furthermore, the use of real IaaS (Infrastructure as a Service) clouds to benchmark the workload performance and infrastructure energy consumption under these variable conditions is constrained by real test-bed availability. Consequently, it is difficult to reproduce verified results and findings that can be trusted. In addition, it would be time-consuming and, therefore, costly (in terms of efforts) to re-configure benchmarking parameters across a large-scale real IaaS cloud for multiple runs and experimentation. Therefore, in large-scale real IaaS clouds, it is not reasonable to conduct benchmarking experiments in a repeatable and scalable manner, and other evaluation methods should be adopted.

A more feasible alternative is the use of simulations. Simulation tools make it possible to evaluate new research hypothesis, proposals, algorithms and/or ideas (here resource/CPU heterogeneity, modelling workload co-location, migration performance and infrastructure energy consumption, benchmarking the cloud workloads) in a controlled platform which helps to reproduce the results easily. Simulations might also offer benefits by allowing to: (i) test services in a repeatable platform; and (ii) tune system bottlenecks before deploying on to real clouds. Furthermore, to develop and test adaptive application provisioning techniques in the context of cloud computing, simulations enable evaluation of heterogeneous workloads, applications and datacenter resources. Note that, due to the abstraction of the real system, it is not essential that the simulator must be as precise as a complete real-world cloud platform in producing results and findings. However, comparable

results could be produced very cheaply and quickly with small programming efforts. Precision and accuracy of various cloud simulators including CloudSim is further described in [2], [3]. Furthermore, as workload performance varies with respect to resource heterogeneities – CPU architectures, as shown in Fig. 1 – therefore, it is essential to model such variations in workload performance during simulations. Similarly, energy consumption of a virtualised server is strongly correlated with the number of VMs running on it but not server energy efficiency. VMs migrations cost energy and degrade workload performance, therefore, users experience higher monetary costs. All these factors have negative impact on the service provisioning and consolidation policies, cloud economics and energy related costs. Therefore, it is important to choose appropriate models while simulating a particular system.

There are a number of cloud simulators suggested in the literature. These include but are not limited to DISSECT-CF [4], CloudSim [1], GreenCloud [5] and DCSim [6]. Albeit, none of these simulators can abstract a complete real platform; however, each one can, at least, model a particular aspect of a real system. For example, CloudSim is more focused on modelling resource allocation policies while GreenCloud is more focused on datacenter networks. CloudSim is an extension of the GridSim [7] simulator that uses SimJava[1] library as a framework for passing messages among various entities and event handling. It is also capable to carry out simulations of heterogeneous IaaS clouds where different applications (workflows) are being executed using numerous experimental parameters. CloudSim and its variants enable programmers to deliberate different characteristics of datacenters, including the number and specification (CPU models) of hosts, memory, storage, network topology, bandwidth and design of datacenter usage. Besides these, turning on/off hosts, VM consolidation through migrations, and integration of energy models (SPECpower[2] benchmarks) are the notable features of CloudSim to model energy, performance and cost efficient datacenters [8]. DCSim provides a very attractive and systematic rack layering approach for modelling a datacenter architecture. In this paper, we make use of CloudSim due to two reasons: (a) its popularity and wide use within the cloud research community; and (b) the theme of work i.e. resource allocation. However, we believe that still CloudSim cannot model a realistic heterogeneous cloud platform. For example, it does not provide a function to add or remove VMs while the simulation is running, which is necessary for the implementation of a real datacenter that runs interactive, on-line, services. Normally, all the simulation entities are instantiated at the beginning and terminated at the end of the simulations. The main distribution of CloudSim expect users to specify the arrival times and service demands of all VMs in advance, which is not possible for on-line services e.g. computer games – most likely the resource demand of interactive services varies significantly over time. Moreover, VMs performance is strongly dependent on the CPU platform or architecture running them, as demonstrated in [9] – similar workloads may experience quite different execution times. Similarly, co-located VMs with similar workloads, particularly if competing for same resources, could also negatively affect workload performance, execution times and, therefore, costs [10], [11]. For example, Fig. 1 describes variations in workload runtimes when run over different CPU architecture or when co-located with other VMs on a particular CPU architecture i.e. E5430 performs best while E5645 performs worst for Bzip2 application. Similarly, energy efficiency of a virtualised host can be related to the total number of VMs just like the fuel consumption, fare and number of people in a car and a bus. A bus consumes more fuel but still cheaper than a car when available seats are full. These performance models and variation in workload performance, due to CPU architecture, co-location and service migration, would certainly affect infrastructure energy efficiency, service revenues (energy bills) and user's monetary costs. Therefore, it is essential to account for all these factors when simulating a particular cloud system. We believe that these models, in particular, the CPU heterogeneity model is not considered in available cloud simulators.

In this paper, we offer an extension to the classical CloudSim simulator through adding several important features and characteristics of large-scale heterogeneous platforms, as described above. We call this "PerficientCloudSim" – where perficient is both a portmanteau term for performance and energy efficient, and also as a word in its own right relating "effective" or "something that

[1] http://www.dcs.ed.ac.uk/home/simjava/tutorial/

[2] https://www.spec.org/power_ssj2008/

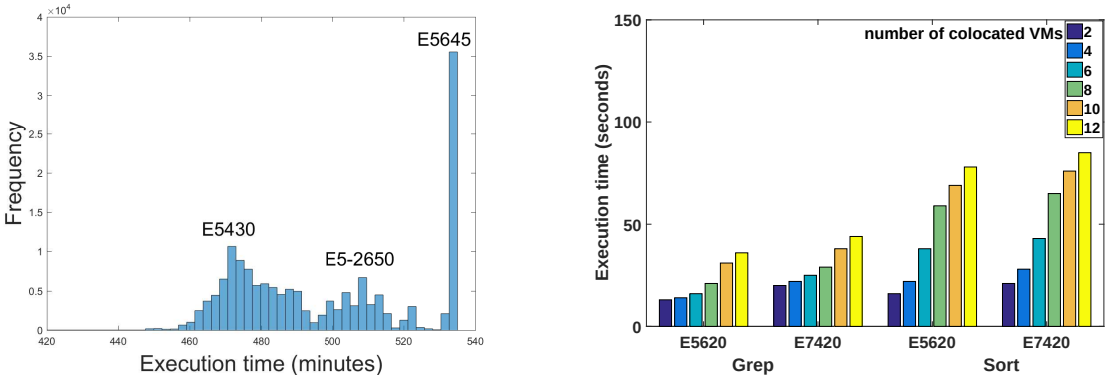

Fig. 1: Variations in workload performance due to platform heterogeneities and interference [11], [12], [left: Bzip2 runtimes over three different CPU platforms – right: two applications (Grep, Sort) runtimes over two different CPU platforms with various number of co-located VMs] – a particular host's co-location interference is highly correlated with the number of VMs running on it

accomplishes or completes a task". The proposed simulator can be used to simulate and evaluate energy and performance aware resource provisioning, consolidation and management policies in infrastructure clouds (IaaS). Our package consists of several important classes to model: (i) CPU heterogeneity - as instances of similar types may perform quite differently on various architectures [9]; (ii) migration energy consumption - migration also consumes energy because there are two VMs running for the duration of the migration (one on the source host and another one, newly created that should be synchronised, on the target host) [3]; (iii) virtualised power model to measure VM energy consumption [12]; and (iv) resource contention due to co-located VMs [11]. We evaluate PerficientCloudSim for energy, performance and cost aware resource management policies (resource allocation and consolidation) using several realistic, plausible assumptions, datacenter set-up and real workload traces from the Google's cluster [13]. Furthermore, we are not aware of the existence of a similar simulator, which could model resource/platform (architecture) heterogeneities, in the cloud research community.
The rest of the paper is organized as follows. A discussion on the major shortcomings of the classical CloudSim simulator can be found in Sec. 2. Major contributions of the work presented in this paper are illustrated in Sec. 3. We offer an overview of the related work and simulators in Sec. 4. In Appendix A, we introduce the classic version of the CloudSim simulator; and a brief discussion of its main and core classes. The extended version of the simulator i.e. PerficientCloudSim is described in Sec. 5. The empirical evaluation study of the proposed PerficientCloudSim simulator, using real workload traces, is described in Sec. 6. We discuss various validation and verification techniques and the accuracy of the proposed simulator (results) in Sec. 7. Finally, Sec. 8 concludes this paper along with several future research directions. A detail discussion of the CloudSim simulator can be found in App. A.

2 Problem Description

As described earlier, cloud workloads may run quite differently and variations in runtimes can be related to CPU heterogeneities and total number of co-located VMs on a particular host that compete for same resources (i.e. they run similar workloads) [9], [11]. Moreover, energy efficiency of resources (hosts) could be translated to energy consumption of individual VMs and their total counts, accommodated on a host [14]. Similarly, migration of VMs may create resource contention due to co-location that subsequently affects workload performance and energy consumption. Prior studies [9], [11] have shown that, in public clouds, similar workloads run quite differently on same VMs (instance classes) which are hosted on different or even similar servers (CPU architectures).

Furthermore, a particular CPU model might run a specific application faster; but, same CPU model may not run other applications with required quality of service. Therefore, performance measurement is an important feature for cloud, in particular, real-time workloads. Subsequently, performance affects users' monetary costs; while energy consumption affects our environment and infrastructure energy bills, therefore, revenue. Therefore, it is essential to account for all these factors, particularly, when simulating public cloud scenarios where real world experimentation are too difficult to carry out. The goal of a particular simulated environment would be to represent a realistic platform in order to guarantee accurate and verifiable results. Most publicly available IaaS cloud simulators do not account for such performance variations. Albeit, CloudSim is the most widely used (and cited) simulator in the cloud research community, however, it has the following major limitations:

- in IaaS cloud, VMs performance (runtime) is log-normally distributed with respect to CPU architecture; and CloudSim does not account for this;
- on a particular host, co-located VMs with similar workloads competing for same resources suffer from significant performance degradation (performance interference), and classical CloudSim does not account for that;
- there is no migration technique implemented nor any abstract class or model is available that could offer an affective approach to VM migration, such as pre-copy, post-copy etc. – albeit migrations are performed;
- it does not have any model to account for VM migration energy consumption and performance degradation; and
- since event-based simulations are, naturally, memory hungry; therefore, CloudSim abnormally terminates when reading large files.

In order to accurately evaluate resource provisioning and management policies, particularly performance and cost aware, the above limitations of the classical CloudSim package must be addressed. In this paper, we propose "PerficientCloudSim" that accounts for the above limitations, as described in Sec. 3. In particular, we offer several statistical models to capture a real heterogeneous IaaS as close as possible. Furthermore, performance degradation due to CPU models and application heterogeneities are integrated. Similarly, resource contention and virtualised host energy models are abstracted. Researchers who are interested in gaining energy, performance, and cost benefits, in infrastructure clouds, can use PerficientCloudSim simulator to evaluate appropriate resource allocation, placement and consolidation with migrations techniques. Besides class naming that starts from the word "Google", as shown in Fig. 7, the proposed simulator is a general purpose and can be used to replay real workload traces from other service providers such as Microsoft Azure cloud [15], HPC [16] and PlanetLab [8].

3 Major Contributions

This paper suggests a simulator that can be used to model heterogeneous cloud datacenters with varying demand and applications. The major contributions of the research conducted in this paper are:

- an extended version of the well-known CloudSim simulator "PerficientCloudSim" is developed that could abstract a real heterogeneous datacenter more closely (statistically validated and verified);
- several modifications are presented, in terms of VM migration costs (such as energy consumption and performance loss - 10% [8]), application heterogeneity [11], and architectural or design level performance churns;
- an approach to model resource heterogeneity is presented – that captures the workload performance loss or improvement due to: (a) service migrations among various servers having different CPU architectures; and (b) resource contention due to co-located VMs that compete for same resources;

– a power model to accurately account for a single VM energy consumption – as energy efficiencies of virtualised hosts can be strongly correlated to its power consumption and the total number of VMs they are running; and
– we use real workload datasets from large-scale cloud service providers such as Google and Microsoft Azure cloud in order to evaluate various resource allocation and consolidation policies using the proposed "PerficientCloudSim" simulator.

4 Related Work

To perform larger-scale experiments and feasibility studies, cloud simulators could be used. A wide variety of simulators (both open source and commercial) is available with different features, capabilities and characteristics. They are designed with one (ad-hoc) or more objectives (generic) [17]. For example, few of them may simulate resource allocation or provisioning policies, but, networks and corresponding congestion models may not be taken into account. Similarly, some may model datacenter networks, workloads and/or cloud architecture, but, they lack simulating management policies and datacenter resource heterogeneity. Albeit, none of them can be used to simulate a whole cloud that resembles a real heterogeneous platform. However, few of them could, possibly, be integrated in order to perform particular tasks or desired levels of simulations/experimentations. Sharkh et al. [18], has presented a systematic review of several cloud simulators which are often used by the cloud research community. The generic simulators can simulate a reasonable cloud, but still with a particular dedicated feature – e.g. VM placement, network, service migrations. A very limited taxonomy of various cloud simulators, which are most widely-used and cited in the cloud research community, is presented in Fig. 2. For a detailed discussion and various surveys of cloud/fog simulators, interested readers must read [19], [20]. Perez et al. [19] have discussed various simulators for simulating cloud, fog and internet of thing (IoT) environments. These simulators are largely discrete event-based, designed in Java. Fog and IoT simulators are not within the scope of this work. The cloud simulators can be classified with respect to cloud back-end technologies i.e. virtualisation, containerisation. Unfortunately, we are not aware of any simulator that could offer support for running various kinds of sand-boxing technologies in a hybrid datacenter e.g. the Intel's CIAO platform[3] [21].
Among these, CloudSim [1] is one of the most popular and widely used Simulator. CloudSim is also highly cited in the cloud research community. By using CloudSim, researchers and industrial developers can focus on specific system design issues that they want to investigate, without being involved into the low level details related to cloud infrastructures and services. Largely, CloudSim is used to evaluate resource scheduling, allocation and consolidation techniques. CloudSim can simulate large-scale datacenters and virtualised hosts, with customizable policies for provisioning host resources to VMs [8] and containers [22]. It provides an easy way to implement scheduling and provisioning policies at both VM and host level. Some of its components or classes have been already validated (but not as a whole) in its support for modelling resource consolidation and VM migration [23]. Moreover, it provides several classes to model workloads that can be easily extended to other real workloads such as the Google's cluster dataset [13]. CloudSim already provides a way to simulate energy and performance efficient clouds and datacenters through the use of DVFS (Dynamic Voltage and Frequency Scaling) and resource consolidation with migration policies. Due to its widespread use and high citations, there are several proposed extensions that add extra capabilities to the classical CloudSim framework[4].
CloudSim is based on Java programming language that provides portability across different platforms and operating systems. Greencloud [5] is a sophisticated packet-level simulator for energy-aware datacenters with a focus on cloud networks and communications. It offers a detailed fine-grained modelling of the energy consumed by the datacenter IT equipment such as hosts, network switches, and communication links. iCanCloud [24] is a simulation platform with a view to model and simulate clouds. The key focus of iCanCloud is to predict the trade-offs between cost and

[3] https://ciao-project.github.io/
[4] http://www.cloudbus.org/cloudsim/

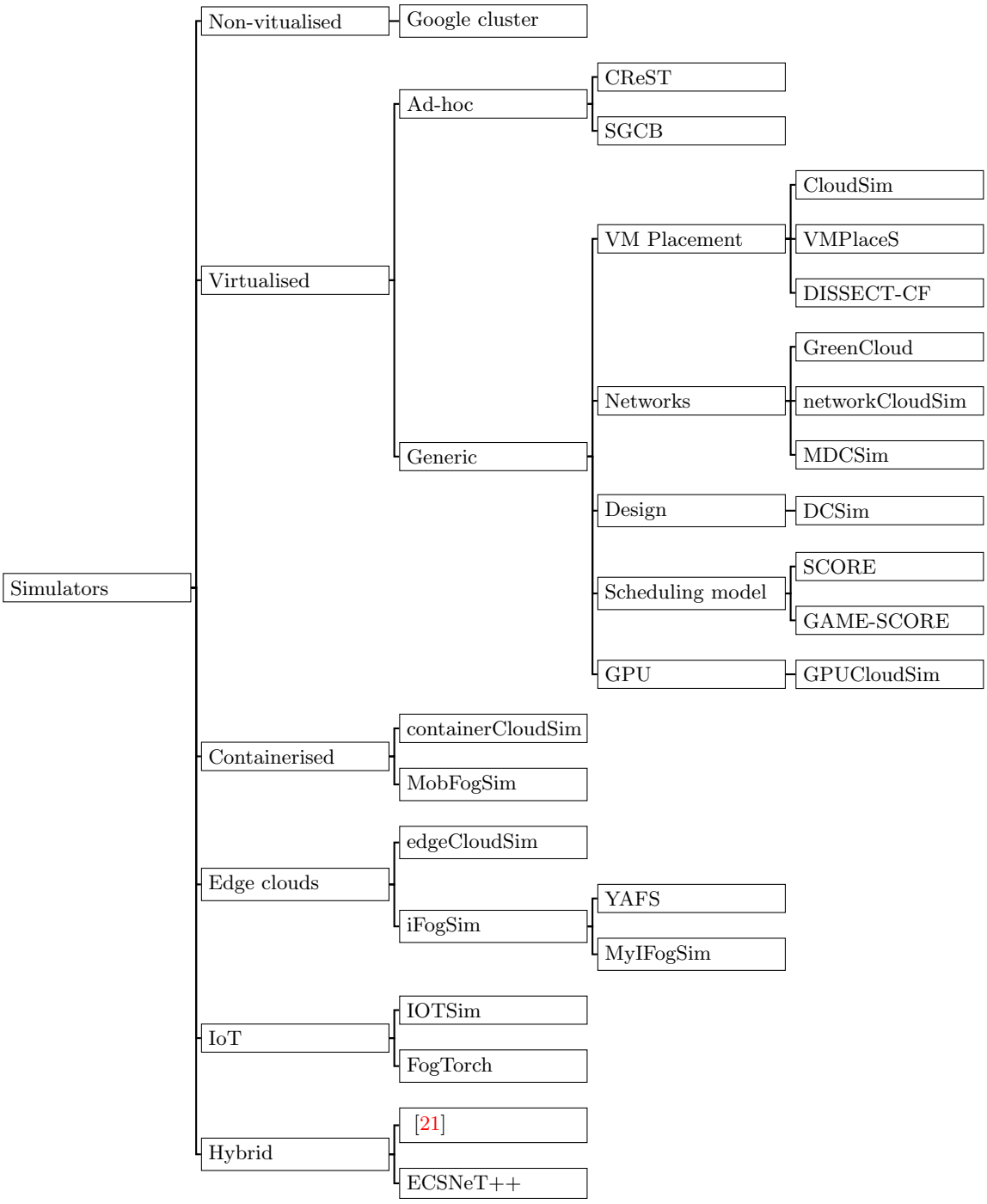

Fig. 2: Taxonomy of various most widely-used cloud simulators [ad-hoc simulators address a particular concern while generic address a majority of cloud computing challenges] – YAFS is developed in Python that stands for Yet Another Fog Simulator

performance of heterogeneous workloads executed on specific hardware, and then provide useful information about resource provisioning costs. ContainerCloudSim [22] offers support for modelling and simulating containers that may run either on bare-metal or inside VMs. EdgeCloudSim [25] can

simulate edge computing platform that considers both computational and networking resources. Moreover, iFogSim [26] is based on the CloudSim, which enables modelling and simulation of IoT devices in the context of fog computing. MobFogSim [27] extends iFogSim to enable service migration and modelling of device mobility in fog infrastructure. MobFogSim is verified through relating the results obtained in simulations with those achieved in a real testbed where services were assumed running in containers. Furthermore, both cold and post-copy container migration methods are outlined and subsequently simulated. A list of various extensions to the classical CloudSim can be found on the link in footnote[5]. Note that, neither of these versions offers support for running various kinds of sand-boxing technologies, such as virtualisation, containerisation, in a hybrid datacenter with the notable exception of our work in [21].

DCSim [6] differs from GreenCloud in that it is more focused on virtualised datacenters which provide IaaS platform to multiple tenants, similar to CloudSim. However, it differs from CloudSim in that it focuses on transactional and continuous workloads[6]. As such, DCSim provides the additional capability of modelling replicated VMs sharing incoming workload as well as dependencies between VMs that are part of a multi-tiered application. In addition, DCSim has a more layered and realistic cloud architecture (hosts inside a rack, racks inside a cluster and clusters inside a datacenter) in comparison to CloudSim. Moreover, DCSim provides support for inter-racks VM migrations (migrating VMs among various hosts in a particular rack) and inter-clusters VMs migration (migrating VMs among hosts across several racks). The former one could potentially increase rack utilisation, while the later one increases cluster utilisation.

DISSECT-CF [4] is a compact, highly customizable open source cloud simulator with a focus on the internal organization and behaviour of IaaS systems. This simulator provides more in depth energy estimation techniques both at host and VM level and is largely validated with real world experiments. The high level components (e.g., IaaS level VM and host schedulers) of the simulator were not validated with real life measurements but by comparing its results with two other simulators: CloudSim [1] and GroudSim [28]. The experiments are demonstrated in [4], and the relative error of DISSECT-CF compared to other simulators was revealed to be as low as 0.29%. It is important to note that the offered high level schedulers are not necessarily following the implementation details of any scheduler of real life IaaS systems, although they are having similar behaviour, and are offered only to show simple example implementations. Gabor Kecskemeti describes about the accuracy of DISSECT-CF that "similar to the CPU models, for highly accurate results, we must suggest a custom VM scheduler that closely matches the one used by our modelled real life IaaS"[7] [4].

GreenCloud is an extension of NS2 (network simulator) to evaluate energy-aware cloud datacenters. The main strength of GreenCloud is the detailed modelling of communication in a datacenter network. MDCSim [29] is a commercial discrete event simulator that models specific hardware characteristics of different datacenter components such as hosts, communication links and switches. Nunez et al. [24] proposed iCanCloud, which is a hypervisor-based simulator specifically with a focus on simulating instance types provided by Amazon EC2. Some tools (simulators) that could simulate an entire cloud stack include CloudSim [1] and DISSECT-CF [4]. However, CloudSim provides limited or no support for more realistic and complex applications composed of communicating tasks and workflows, and has no cross-layer interaction. Furthermore, CloudSim does not account for CPU model/platform/architecture heterogeneities. The DISSECT-CF simulator allows access to internal cloud information (such as VMs and workloads) and accurately models energy consumption of IaaS clouds at two different levels: (i) hosts (coarse-grained); and (ii) VMs (fine-grained). Moreover, energy consumption of a VM migration is also modelled and implemented in the DISSECT-CF simulator [30].

Another simulator "VMPlaceS" [17], [31], built on top of SimGrid, offers support for evaluating various VM placement and consolidation policies. In addition, node failures (addition and removal) and dynamic workload fluctuations are considered in large-scale clusters. Three well-known VM

[5] http://www.cloudbus.org/cloudsim/

[6] http://www.dummies.com/programming/cloud-computing/hybrid-cloud/types-of-workloads-in-a-hybrid-cloud-environment/

[7] personal correspondence with Gabor Kecskemeti, a Research Fellow in Laboratory of Parallel and Distributed Systems, MTA SZTAKI – www.lpds.sztaki.hu

placement and consolidation approaches, i.e. Entropy, Snooze, and Dvms (that correspond to centralised, hierarchical and distributed scheduling, respectively), have been investigated for various metrics such as performance, computation and reconfiguration durations, violation, and energy efficiency. However, CPU heterogeneity, resource contention due to co-located VMs and migration performance are not taken into account. Albeit, VMPlaceS can simulate large-scale IaaS clouds that consist up to 8,000 hosts and 80,000 VMs; using real workload traces in a reasonable time. The Google simulator[8] offers support for simulating jobs/tasks scheduling; however, the notion of VMs and related capabilities such as migration are not considered. Therefore, it can not be used to simulate cloud computing platforms. CReST [32] simulator is a stand-alone application that offers simulation at numerous levels of abstraction: from physical resources, energy consumption and thermal flows within a datacenter, to infrastructure networking and application services' virtualisation with additional capabilities of variable user demand. In [32], CReST has been evaluated for various communication protocols, thermal management, and network topologies. Unfortunately, performance degradation models with respect to migrations, CPU heterogeneity, resource contention, and co-located VMs is not discussed.

SimGrid Cloud Broker (SGCB) simulator, as designed in [33], offers support for simulating multiple Amazon IaaS (EC2) like clouds along with different availability zones (AZs) and regions. Furthermore, cloud storage (S3, EBS) simulations, various instance types and integrating various price models (accounting and billing service) are also possible with the proposed SGCB simulator. The simulator has been tested and verified for large-scale clusters running up to 30,000 instances in several regions. In addition, SGCB can simulate PaaS (platform as a service) and SaaS (software as a service) cloud models. To some extent, performance of the resources (instances) and disk drives are considered; however, resource contention, CPU heterogeneity, and infrastructure energy consumption are not taken into account. SCORE [34] can simulate heterogeneous, both real and synthetic workloads, in terms of power-efficient monolithic, as well as, parallel scheduling models. Empirical tests confirm that SCORE is a reliable tool for testing energy efficiency, security, and scheduling strategies in cloud-computing environments while offering: (i) the design of an energy consumption model; (ii) extension of VM placement policies; (iii) shutting on and off servers; (iv) heterogeneity resources; and (v) security profiles. GAME-SCORE [35] is an extension of SCORE that offers support to balance between two conflicting requirements of datacenters i.e. high throughput and low energy consumption. GAME-SCORE is based on a non-zero sum game (Stackelberg) with a resource manager that maximises performance via placement decisions while the energy-efficiency manager asks the resource manager to migrate applications for minimising energy consumption via shutting down idle servers.

To meet the computational needs of cloud users, IaaS providers offer GPU-enabled services [12]. However, owing to the complications of the GPU devices, conventional virtualisation methods are not applicable directly. Therefore, numerous virtualisation approaches including, full, para and hardware-assisted virtualisation are implemented to share GPU among several VMs. To ease up GPU-enabled experimentations, GPUCloudSim [36] is built on top of CloudSim that enables modelling and simulation of GPU-enabled VMs in IaaS datacenters. The package also comprises various models to simulate interference among co-running workloads, virtualisation overhead and energy consumption of the GPU devices. Furthermore, provisioning and scheduling policies are integrated to enable experimentations. Nutshell [37] is a recently proposed simulator that makes it easy to model and simulate various cloud solutions. Prominent features presented by the Nutshell include: provision of IaaS resources; creating new datacenter architectures; communication protocols; and VM schedulers. Similarly, ECSNeT++ [38] is an extension of the most widely used OMNeT++ toolkit that could simulate distributed stream processing applications on edge and IaaS clouds (hybrid platforms i.e. mobile edge clouds). This validated simulator can measure network, processing delays, and the average energy consumption of the edge devices. Similarly, 5GPy [39] is SimPy-based simulator that can simulate a hybrid cloud, fog, RAN (radio access network) environment. However, CPU heterogeneity, performance impacts due to resource contention and co-location are not taken into account.

[8] https://github.com/google/cluster-scheduler-simulator

Compared to other cloud simulators, CloudSim offers an easy way to design and validate energy efficient computation. The results produced using CloudSim are currently being validated (in terms of energy consumption, VM provision/allocation and migration policies) in several real cloud platforms such as OpenStackNeat [23], as described in [40]. However, all of its components and classes are not validated, yet. With the only exception of the resource contention model, the accuracy of "PerficientCloudSim" is demonstrated in [2]. However, due to abstraction and mathematical (statistical) models, the results produced in a simulation environment may not be still achievable on a real platform. Therefore, it is essential to use various statistical techniques in order to validate and verify simulated results and outcomes. To summarise these simulators, each one has a capability to model certain aspects of an IaaS cloud and/or fog infrastructure. However, CPU heterogeneity and performance impacts due to resource contention are relatively ignored. For example, GPU-CloudSim can model performance impacts of resource contentions (w.r.t co-located workloads); however, CPU heterogeneity, migration performance impacts and application heterogeneity ae not taken into account. Furthermore, neither of the existing simulator has the capability to account for resource contention (w.r.t hardware) and CPU heterogeneity. Table 1 describes summary of the related work. We believe, information in this table will help our readers to quickly identify gaps for further research, investigation and improvements.

Table 1: Summary of the related work and simulators with respect to various evaluation criteria [Mig - migration, RC - resource contention, App - application]

Simulator	Parameters								
	Scheduling models	Energy aware	Shutdown policies	Scheduling strategies	Performance models			Network models	Resource heterogeneity
					Mig	RC	App		
CloudSim		×	×	×	×		×	×	
CReST		×		×				×	
DISSECT-CF		×	×	×			×		
VMPlaceS	×		×	×					
GreenCloud		×	×	×				×	
DCSim			×	×			×		
SCORE	×	×	×	×	×				
GPUCloudSim		×		×		×			
PerficientCloudSim		×	×	×	×	×	×	×	×

5 PerficientCloudSim

CloudSim [1], its layered architecture, and the features provided, are briefly explained in App. A. It is essential that readers should go through that discussion first in order to fully understand our proposed simulator "PerficientCloudSim" and the list of additional capabilities offered through abstract classes. Fig. 3 shows the block diagram of the CloudSim and its various components, offering a view of different capabilities for which CloudSim can be used such as scheduling (CPU level), VM placement and consolidation with migration techniques (host level). CPU level scheduling is not within the scope of current work: the theme of this paper is VM allocation and migration which is demonstrated with respect to the well-known CloudSim simulator. However, we found several limitations that need to be addressed before implementing, evaluating energy, performance and, therefore, cost aware scheduling and consolidation with migration techniques that closely resemble a real heterogeneous datacenter. To tackle these issues, we extended existing or added additional classes into the classical CloudSim package. Below we briefly describe various extended classes and their implementation.

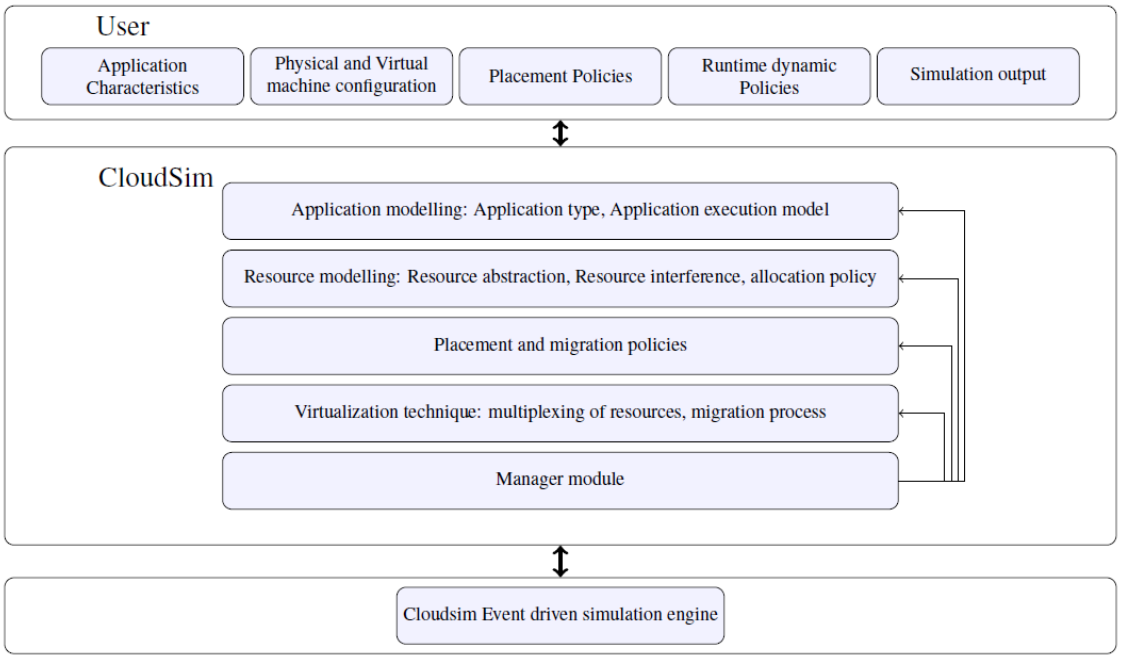

Fig. 3: CloudSim high level block diagram that shows a list of CloudSim features such as virtualisation, resource allocation and migration of VMs – users request VMs for their application (workload) and can see the desired output [1] – see App. A for further details regarding CloudSim's architecture, various class and sequence diagrams

5.1 Modelling Interactive Services and Large-scale Dataset

For example, the available version 3.0.3 of the CloudSim simulator does not support dynamic creation of VMs at runtime. Moreover, reading large data, particularly, from Google cluster is not feasible due to its huge size. To make it possible, we extend the *DatacenterBroker* class, with additional capabilities to read the Google data and monitor the submission time of each task to create VMs at runtime. If a task cannot be allocated to a VM, the *DatacenterBroker* puts the task in its waiting queue (W) to handle it later when enough resources are available. The *DatacenterBroker* also implements an instance type selection algorithm [12] to choose a suitable VM type, based on the amount of requested resources (such as CPU, memory) and performance levels, for each task and charge the user according to Google custom machine[9] prices. This will enable the service providers to evaluate various instance selection policies in order to optimise their datacenters and revenues. We also extend the *PowerDatacenter* class to implement certain migration policies, in particular, to take appropriate (energy, performance and cost aware) migration decisions. For example, migrate those VMs that could recover their migration costs [3]; and in other words, avoid migrating short running costly VMs.
We face two major issues while dealing with the above classes. For example, an existing problem in the *PowerDatacenter* class was that several VMs were suddenly disappeared when they were in migration process. In the extended class, we resolved this issue. In essence, there was no check on the VM status during its migration and some of the VMs finished their execution while they were in the migration process. Another issue was the heap memory, particularly, when dealing with large-scale simulations that run for longer periods. As discussed earlier, CloudSim deals with each entity as an object, and it is important to clear the corresponding memory states and references when some objects are destroyed or not referenced. We noted that the Java garbage collector was also unable to free memory space, as the objects were referenced even when they have been already

[9] https://cloud.google.com/custom-machine-types/

destroyed. We do not claim this as problem with the JVM platform, however, this usually happens due to bugs in programming and/or creating/referring to unnecessary events or objects in the simulations. To resolve this issue, we used two techniques: (i) modified the *DatacenterBroker* class to explicitly destroy VM objects when they finish their execution; and (ii) force the Java garbage collector to periodically clear the memory.

The classical CloudSim package consists of a *WorkloadFileReader* class that reads traces from the given text files; and create VMs, accordingly. However, reading large trace files for long durations, and creating VMs dynamically at runtime, are not supported. Therefore, the extended *GoogleWorkloadFileReader* class reads every task and requests the broker to create a VM at runtime. We assume each Google cluster task is a VM, and extract its characteristics (required resources, duration etc.) from the trace [13]. The required resources (CPU, memory), submission (request) time and runtime (execution time) of every task are known prior to VM creation. Note that, the execution time is calculated from the workload's required MIPS on a particular CPU (GHz). It is also possible that a user's task has a deadline, and it might be useful to know before launching a VM whether it is able to finish its execution in time or not – performance is guaranteed or not. The VM various requirements which are described in Table 2, were incorporated in the *GoogleJob* class that subsequently extends the classical *Cloudlet* class. In this case, every task request also goes through an instance selection algorithm [3], and a suitable (that ensures the requested levels of workload performance), cheap (the lowest price) instance type (VM) is selected from the available pool of instance (VM) classes.

The Google cluster dataset is briefly explained in [13]. We wrote a Python script to read task durations and start times from the dataset (available for free at the GitHub repository)[10]. Using task runtime details from Google data, we wrote the dataset as a text file (*.txt*) which contains various fields as shown in Table 2. The start times can be modelled as poison distribution, starting from 0 and continue until simulation duration; while the finish time of each VM is computed as its start time plus its execution duration. Furthermore, execution time of a VM is calculated through dividing total MIPS needed (allocated task size) by CPU speed. Note that, the CPU needed for a particular VM is converted into the notion of MIPS (Millions of Instruction Per Second) in order to create consistency with the CloudSim. The *GoogleWorkloadFileReader* class reads the file and sends an event for VM creation at runtime (start time). When a VM is created, the *GoogleDatacenterBroker* class sets its termination time (finish time) and sends a *VmDestroy* event when that particular time is reached and observed.

As described earlier that reading large files containing millions of entries can create problems with the JVM memory (heap). In such circumstances, the simulations on our server were significantly slow and, later on, terminated abnormally. Perhaps, this is due to less amount of memory (RAM). To resolve this issue, we used two approaches: (i) feed the data in chunks or multiple files instead of a single file and explicitly clear the heap memory; and (ii) use statistical models instead of real datasets. In respect of (i), the *GoogleWorkloadFileReader* class could now read the trace file(s) in chunks; and clears up the heap memory at regular time. In respect of (ii), we observed that using mathematical models instead of a real trace could significantly reduce the simulations times. However, this may not be of significant worth as statistical models may not essentially represent real data due to abstraction. Albeit, performance boost can be achieved through loading chunks of data from disk, thus reducing JVM memory usage and employing statistical models instead of real data sets. However, these statements are rather contradictory to earlier findings [41]. It is well-understood that reading from a disk is always slower than RAM - discrete event simulation are, in nature, memory hungry due to the amount of events it has to handle in large systems. We also use another dataset from the PlanetLab (data related to CoMon project) [42] to validate the simulations performed in the PerficientCloudSim. The PlanetLab data consists of CPU utilisation values for more than a thousand of VMs with an interval of five minutes. It is also possible to statistically model the workload variations and then use the derived model in simulations. Note that, the classical CloudSim simulator already offers various classes for this purpose, as described in App. A. Moreover, we only offer a brief discussion of the most important CloudSim classes which

[10] https://github.com/google/cluster-data

are essential to understand in order to incorporate the proposed classes of the PerficientCloudSim simulator. For complete discussion of the the CloudSim simulator, interested readers are suggested to read [1].

Table 2: Fields in the dataset [CPU requirements are described in MIPS to make them consistent with the CloudSim – for example if a VM, which has 1 GHz CPU, runs for one hour then it needs $1000 \times 60 \times 60 = 3,600,000$ MIPS [1], [3]]

Field name	Type	Description
Start time	(*integer*)	start time of the VM in seconds
CPU	(*integer*)	total number of MIPS that the VM needs to execute
Memory	(*integer*)	memory needed for the VM in MB
Network	(*integer*)	network resources needed for the VM in MB
Finish time	(*integer*)	finish time of the VM in seconds Start time+Ex. time [computed via dividing MIPS by speed]

5.2 Modelling Energy Consumption

A slight modification was made to the *PowerHost* class in order to ensure that an idle host still consumes idle power (P_{idle}). In the original class, the idle hosts were consuming no energy. To implement realistic simulations (which resemble closely to a real test-bed), we extend the *Host* class for host reconfiguration costs (in terms of energy consumption and delay), as shown in Table. 3 [43]. In addition, hosts could be kept in sleep mode, hibernated or standby - thus offering quick preparation if workload suddenly rises. These costs matter when switching on/off the resources (hosts) using dynamic capacity planning (DCP). These changes were incorporated in the *GooglePowerHost* class. To create a history of the VMs past runtimes, the *Vm* class was extended. The extended *PowerVmList* then ensures that VMs selected for migration are in descending order of their past runtimes (R_{past}). These changes were incorporated in the *GooglePowerVm* class providing a way to maintain the VMs runtimes history. All VM allocation policies were implemented through extending the *VmAllocationPolicy* class. The new class *GooglePowerVmAllocationPolicyAbstract* now offers support for evaluating various energy, performance and cost efficient resource placement and migration policies. However, there is no model implemented for the energy consumption of VM migration in the classical CloudSim. We add a *MigrationPowerModel* class to the CloudSim package, which can be extended to any power model of the migration process. Initially, the migration power model demonstrated in [44] was added to perform the simulations. We also add a new class to the CloudSim, i.e. *VmLevelHostPowerModel*, to estimate the energy consumption of VMs on a virtualised host. We extend this class with the VM power model as explained in [14] and which is also demonstrated on a real cloud test-bed in [45], [46]. The VM power model is dependent on the linear power model of the CPU energy consumption E i.e. CPU energy consumption is proportional to its utilisation level (the more it is utilised the more it will consume and vice versa), given by Eq. 1:

$$E = P_{idle} + (P_{max} - P_{idle}) \times U \tag{1}$$

where U is the current utilisation level of the CPU, while P_{idle} and P_{max} denote the CPU energy consumption when idle and 100% utilised, respectively. This model can be translated to VM level power model, as described later in Sec. 5.8. Furthermore, considering virtualised servers, alone the VM energy usage is directly related with the number of VMs executing over that specific server. Thus, it is realistically sensible to forecast/estimate the VM energy usage in context with the linear

CPU energy usage model using Eq. 2:

$$E_{VM} = \left(\frac{P_{idle}}{N_{VMs}}\right) + \mathcal{F}_{VM} \times (P_{max} - P_{idle}) \times U_{VM} \tag{2}$$

where aggregated VMs housed on a server is given by N_{VMs}, the extent of server's resources, like number of cores allotted to the VM, is given by $\mathcal{F}_{VM}$, and usage level of VM is shown by U_{VM}. The aggregated energy consumption of a virtualised server E being executing N_{VMs} number of VMs is given by Eq. 3:

$$E = \sum_{k=1}^{N_{VMs}} P_{VM_k} \tag{3}$$

Simply saying, we take up to consider resource like $\mathcal{F}_{VM}$ as the number of cores. Though, additional resources like CPU and memory may possibly be decided in resource allocation. The usage of U_{VM} for a VM mentions from the actual (predefined) workload datasets. By using $\mathcal{F}_{VM}$, it empowers us to make it simple in consideration of an individual VM as an actual server. In IaaS clouds, sizes of VMs are equally divided by the number of assigned cores (hyper-threaded) obtainable through C cores on the server, or simply by allotting on amount of memory. To keep it simple, we use it equivalent to only number of cores, given by Eq. 4.

$$\mathcal{F}_{VM} = \frac{cores_{VM}}{C} \tag{4}$$

The aforementioned model can be used to forecast energy consumption of a particular VM on a specific server at suitable usage levels. Similarly, energy consumption of a migrated VM is the amount of energy consumed at source or target host - since during live migration, there are two VMs running for the duration of the migration. This will also include the energy consumed in network and marginal migration cost [14].

Table 3: Host reconfiguration costs (energy consumption and set-up delay) [3], [43]

Old state	New state	Energy consumption (joules)	delay (seconds)
Off	On	60.0	30
Standby	On	14.3	5
Hibernate	On	60.0	30
On	Off	110.0	30
On	Standby	14.3	5
On	Hibernate	60.0	30

5.3 Modelling CPU Heterogeneity

CloudSim does not model the resource heterogeneity and performance variations due to CPU models and/or workload contention. However, the performance degradation due to migration is already taken into account [8]. To model resource heterogeneity and performance variations, we use several performance benchmarks from Amazon EC2, relate them to the Google workload to extract performance parameters and feed them into the PerficientCloudSim. We extended the *Host* class with various performance parameters (such as mean, standard deviation) to model variations in workload runtimes. The new *GooglePowerHost* class accounts for these parameters when running workloads; and the *GooglePowerDatacenter* class accounts for performance adjustments when migrating workloads among various hosts. The former one computes performance degradation/improvement in the executable workload (MIPS) given the type of host's performance distribution, mean, standard deviation; and subtract/add these MIPS from/to the executable workload – thus impacting

the workload total runtime (performance). The later one uses the z-score normalisation technique to compute the remaining runtime which relates to a particular distribution (target host) for another distribution (source host) [12]. For example, if a particular real host executes a given workload in 42 – 57 minutes (log-normally distributed with given mean and standard deviation); then the *GoogleVmScheduler* class would appropriately adjust the ratio of MIPS processed to produce similar runtimes for the workload. Similarly, for migrations the expected remaining runtime (z-score) is set for the VM on the target host using the *GooglePowerVm* class. From an implementation point of view, when a particular VM is being migrated to a particular target host, its remaining runtime on the source host is translated to a remaining runtime (expected) on the target host using the z-score statistical technique [12]. For example, if we assume a source and a target host with performance parameters, μ_s, σ_s, μ_t and σ_t, in terms of means and standard deviations (of execution times), respectively. Then the remaining runtime (expected) T_t of a migrated VM on the target host (after the migration) for its remaining runtime (T_s) on source host (before the migration) can be computed as:

$$T_t = exp\left[\sigma_t \times \left(\frac{log(T_s) - \mu_s}{\sigma_s}\right) + \mu_t\right] \tag{5}$$

Note that, Eq. 5 has been derived using the standard score (a.k.a z-score normalisation) technique, as given by Eq. 6. The standard score is used to calculate the probability of a particular score (T) which occurs in the interior of a dataset (normally distributed) with given mean μ and standard deviation σ. Furthermore, standard score provides an easy approach to relate two or more than two different datasets which are normally distributed.

$$z = \frac{T - \mu}{\sigma} \tag{6}$$

Eq. 7 is used to compare the runtimes of the VM on the source and target hosts (T_s, T_t), respectively; given the statistical means (μ_s, μ_t) and standard deviations (σ_s, σ_t) of both source and target hosts having normal distributions.

$$\frac{T_s - \mu_s}{\sigma_s} = \frac{T_t - \mu_t}{\sigma_t} \tag{7}$$

Since, prior studies [3], [9] have shown that VMs runtimes are usually lognormal; therefore, the exp in Eq. 5 is used to translate the normal distribution to an equivalent lognormal distribution. In order to model performance interference, we use applications runtimes, as described in [11], that were obtained on a real platform (see right of Fig. 1). The performance interference is strongly correlated to the number of VMs running on a particular host. Based on experimental evaluation in [11], we can use various statistical techniques such as linear regression to fit a line, for each particular host. For example, for CPU model "E5620" and application type "Grep", we get $R^2 = 0.9104$ with the following regression line.

$$y = 2.4429(x) + 4.7333 \tag{8}$$

where y is the runtime and x is the number of co-located VMs. From an implementation point of view, when a VM is launched (new), terminated or migration happens, we compute the total number of running VMs on a particular host. Using the old number of VMs (x_1) and new number of VMs (x_2), we compute the increase or decrease in the performance interference i.e. $y = y_2 - y_1$. Finally, the increase or decrease in the performance interference (y) is added or subtracted to each VM runtime, accommodated on that particular host. We are aware that the runtime will certainly be affected by the VM sizes as well. For example, a host that runs two VMs of large type will suffer from lower contention as compared to same host when running 10 VMs of small type. This behaviour can be modelled with respect to heterogeneous VMs. Moreover, the contention may be higher if co-located VMs are highly utilised than when underutilised. Therefore, requiring to profile each function for different data sizes and aspects seems infeasible. We will consider such complex models in the future. We believe, a generalised model should be adapted in order to account for these various options. This discussion in not within the scope of our current work. Further details on a generalised heterogeneity model can be found in [47].

5.4 Modelling Workload Heterogeneity

As described earlier, using the CloudSim notion of MIPS as a single measure implies homogeneous workloads. Moreover, we assume MIPS as a single proxy to denote both resource and application heterogeneities; since there is no other way to simulate the differences among various workloads e.g. a disk-bound and a CPU bound task, of integer/floating point operations and etc. It is also possible that utilisation levels of workloads may vary; however, this only provides a way to differentiate between CPU intensive and non-intensive workloads. Moreover, utilisation levels can translate to variations in runtimes. Therefore, we assume that workloads can be classified through their requests for larger or smaller MIPS i.e. runtimes. Due to non-availability of a performance oriented real workload dataset within the cloud research community, we use tasks' priorities from the Google's cluster dataset as proxy to represent workload type. We believe this assumption can accurately represent application types since tasks' priorities affect billing (users monetary costs) [13]. Moreover, tasks' runtimes that belong to a particular priority are assumed as variations in workload performance. To make our assumptions valid, the workload type and characterization could be achieved through mapping or comparing Google tasks' runtimes to runtimes of real benchmark workloads i.e. distributions of runtimes normalised over the same scale; and the closer similarity is assumed as a particular type of workload [12].
Note that, the assumption that each instruction requires at most one CPU cycle in order to be executed constitutes a very optimistic approach. There are substantial differences in IPC (instructions per cycle) values among cores (in particular those which share various components) with different pipelines, speculative mechanisms, hyper-threading, memory and cache hierarchy, which is the gist of CPU or workload heterogeneity and variations in execution times. For example, hyper-threading allows cores to hold the states of two instructions in execution (threads), simultaneously. In case, both threads compete for same resources, contention will definitely happen [9]. Therefore, we believe that validation through actual workload execution on microprocessors can be misleading when they leverage the same or very similar microarchitecture. Besides these architectural heterogeneities, workload performance can also be affected by the speed at which data is transferred from memory to the CPU. Moreover, memory space measures can also affect workload performance if is close to full. Finally, the shared storage systems, along with network devices, may create performance bottlenecks for various workloads. Note that, our simulated model for workload heterogeneity is based on monte-carlo simulations using a log-normal distributions of data given in Table 4.

Table 4: Runtimes of different workloads across various CPU models (seconds) [48]

Workload type	CPU model	Runtimes
povray	E5507	544s
	E5430	579s
bzip2	E5507	641s
	E5430	447s

5.5 Modelling Resource Contention

Performance of workloads may be affected with resource contention which potentially occurs due to co-located VMs i.e. VMs running on a particular host compete for same or similar resources as demonstrated in [11]. Co-located VMs may suffer from approximately 6.96% to 37.5% performance loss [9], [11]; possibly due to the: (i) mismatch between the available host' resources; and (ii) aggregated resource utilisation levels of co-located VMs. Moreover, these variations in performance vary for various applications; some applications are impacted slightly while others may experience

heavy losses in performance which are directly proportional to the total numbers of co-located VMs and strongly dependent over CPU platforms. In other words, the larger number of co-located VMs on a host, the more losses can be seen across all VMs. The former relationship, i.e. performance loss due to number of co-located VMs, can be modelled using either regression/trend lines or other models suitable for positive linear relationships. The later one, i.e. performance loss across various applications over various CPU platforms, can be modelled through statistical distributions (multi-modal) – where multi-modality denotes runtimes' variations of a particular application over several CPU architectures. The overall performance loss (or improvement) due to resource contention is, therefore, given by:

$$T_{rc} = D_p + L \tag{9}$$

where D_p and L represent the difference between runtimes (distributions of variations in runtimes due to CPU architectures) and linear model (regression equation for number of co-located VMs), for a particular application, respectively. In our case, D_p is computed using the z-score normalisation technique, as described above. Therefore, $D_p = T_t$ as given in Eq. 5. Moreover, L is estimated through plotting the trend/regression line over the real data gathered in [11], as shown in Fig. 4. Note that, L varies with respect to various applications and their resource demand or usage. From implementation point of view, for a particular application, the T_{rc} is converted into equivalent MIPS; and, then, added or subtracted to denote either performance loss/degradation or gain/improvement, respectively. A generic (workload independent) model is still needed which can represent performance churns for all workloads (applications), which are col-located across heterogeneous datacenters' resources. For example, we can normalise the four regression models, as shown in Fig. 4, into an integrated generic model (probably averaged) i.e. a single equation; which can be used for various numbers of co-located VMs, with respect to two different workload types and CPU models. We believe, similar models would certainly help in simulating real IaaS clouds, closely and accurately. Note that, our simulated model for resource contention of different hosts is based on several monte-carlo simulations using the log-normal distributions of data given in Table 5.

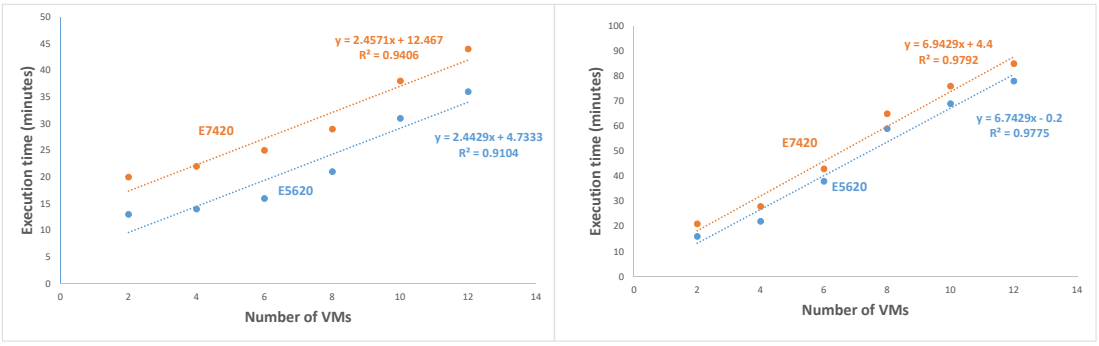

Fig. 4: Resource contention due to co-located VMs for two different applications i.e. Grep (left) and Sort (right) running over two different CPU platforms [11] – ∼2% to 9% errors are expected

5.6 Modelling Resource Migrations

Note that, the VM migration process is very straightforward in the classical CloudSim simulator. From implementation point of view, when a VM is triggered for migration, it is terminated on the source host and after the migration duration, another one is created on the target host. This procedure does not really represent a valid migration process such as pre-copy (where migrations

Table 5: Runtimes of different workloads on co-located VMs (seconds) [11]

Workload type	CPU model	Number of VMs 2	4	6	8	10	12
		Runtimes					
Sort	E7420	21	28	43	65	76	85
	E5620	16	22	38	59	69	78
Grep	E7420	20	22	25	29	38	44
	E5620	13	14	16	21	31	36

happen in various rounds); particularly, in dynamic network scenarios where bandwidth and congestion vary significantly. Moreover, during live migration, as VM's memory is often transferred iteratively which may consume considerable network I/O resources and could, therefore, severely disrupt VM services. The performance of a migration approach is reliant on the VM workload, available resources on source and target hosts, and co-located services on both hosts [49]. Furthermore, in real scenarios, different approaches to migrations would have different performance impacts on VMs running various kinds of workloads. For example, as shown in Fig. 5, the migration durations largely overlap for two applications (Bzip2 and Dacapo) when migrated with two different migration algorithms (pre-copy and post-copy); however, the down times are completely different. Therefore, it is essential to add migration models that can simulate various approaches to migration, given different experimental parameters such as: (a) the dynamic behaviour of network bandwidth; (b) workloads priorities for migration; (c) time limitations for completing migrations; and (d) serial or parallel migrations[11]. The VmigSim[12] simulator offers support and extensions (additional classes) to the classical CloudSim for implementing several approaches to VM migrations such as the well-known pre-copy technique. However, instead of inter-datacenter migrations, VmigSim offers intra-datacenters migrations. In the current version of PerficientCloudSim, we use the default migration technique offered in classical CloudSim. However, with trivial programming efforts, VmigSim can be integrated into PerficientCloudSim.

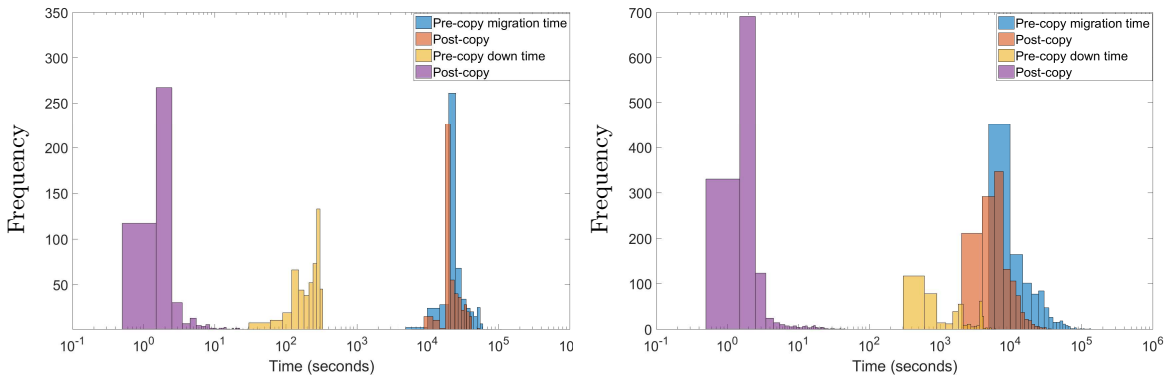

Fig. 5: Migration durations and down times for two different benchmark applications using two different approaches to migrations i.e. pre-copy and post-copy [left: Bzip2 – right: Dacapo] – the migrations data was collected on a real cluster [49]

Moreover, a VM may experience severe performance loss during its migration duration. In the

[11] http://www.uni.net.th/wunca_regis/wunca32_doc/21/014_WUNCA32_Presentation_Thammasat.pdf
[12] https://github.com/poa28451/VmigSim

classical CloudSim, a 10% decrease in workload performance is assumed, as described earlier [1]. However, this may hold only for certain kinds of workloads, e.g. CPU intensive; but, other workloads may experience different levels of degradation. Moreover, a VM (in migration) may also affect the performance of other workloads running both on source and destination hosts [50]. Therefore, the available resource capacities both on source and destination hosts have impacts on VM migration performance. Similarly, different approaches to migration, such as pre-copy, post-copy, may have different impacts on the workload performance. Due to non-availability of a representative VM migration workload[13], with the only exception of [49], migration performance modelling is relatively un-explored. Fig. 6 shows that performance loss for various applications and migration policies can be modelled as log-normally distributed. In [49], a machine learning technique has been used to model VM migration performance and durations. Moreover, the performance degradation is noted having a relationship with the VM utilisation levels i.e. approximately 1–3% (up to 80% utilisation) and approximately 5% (up to 97% utilisation). In order to account for the implementation of such models, we added an abstract class *MigrationPerformanceModel*; which can further be extended with the desired migration performance model.

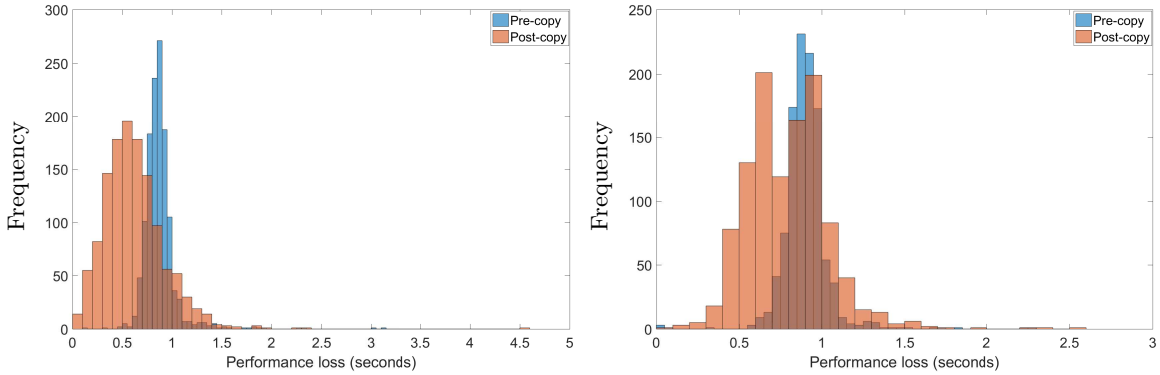

Fig. 6: Performance degradation during migrations for two different benchmark applications using two different approaches to migrations i.e. pre-copy and post-copy [left: DACAPO – right: PARSEC] – the migrations data was collected on a real cluster [49]

As described in [14], during a live VM migration, there are exactly two VMs running that cost extra energy – the original one on the source host and another one on the target host. When migrating for energy efficiency, this cost must be taken into account, particularly, when workloads run for short durations. For short lived VMs, migration efforts might be wasted [3]. Largely, the energy is consumed while moving the data (memory and ephemeral storage) of the migrated VM [51]. Beside this, energy consumption of both source and destination hosts (therefore, running VMs – co-located) is also affected [52]. The abstract *MigrationPowerModel* class adds the capability to measure the VM migration energy consumption. This class can be further extended with the desired migration energy model.

5.7 Putting All Efforts Together

To build PerficientCloudSim, the above models were integrated into the classical CloudSim simulator. Fig. 7 shows the UML (Unified Modelling Language) class diagram of several extended classes in the proposed PerficientCloudSim simulator. The classes in the *google* package extend the CloudSim functionality for various VM allocation and migration policies. Similarly, the classes defined in the *googlecluster* package read Google data and create VMs according to their arrival rate

[13] https://csap.snu.ac.kr/software/lmdataset

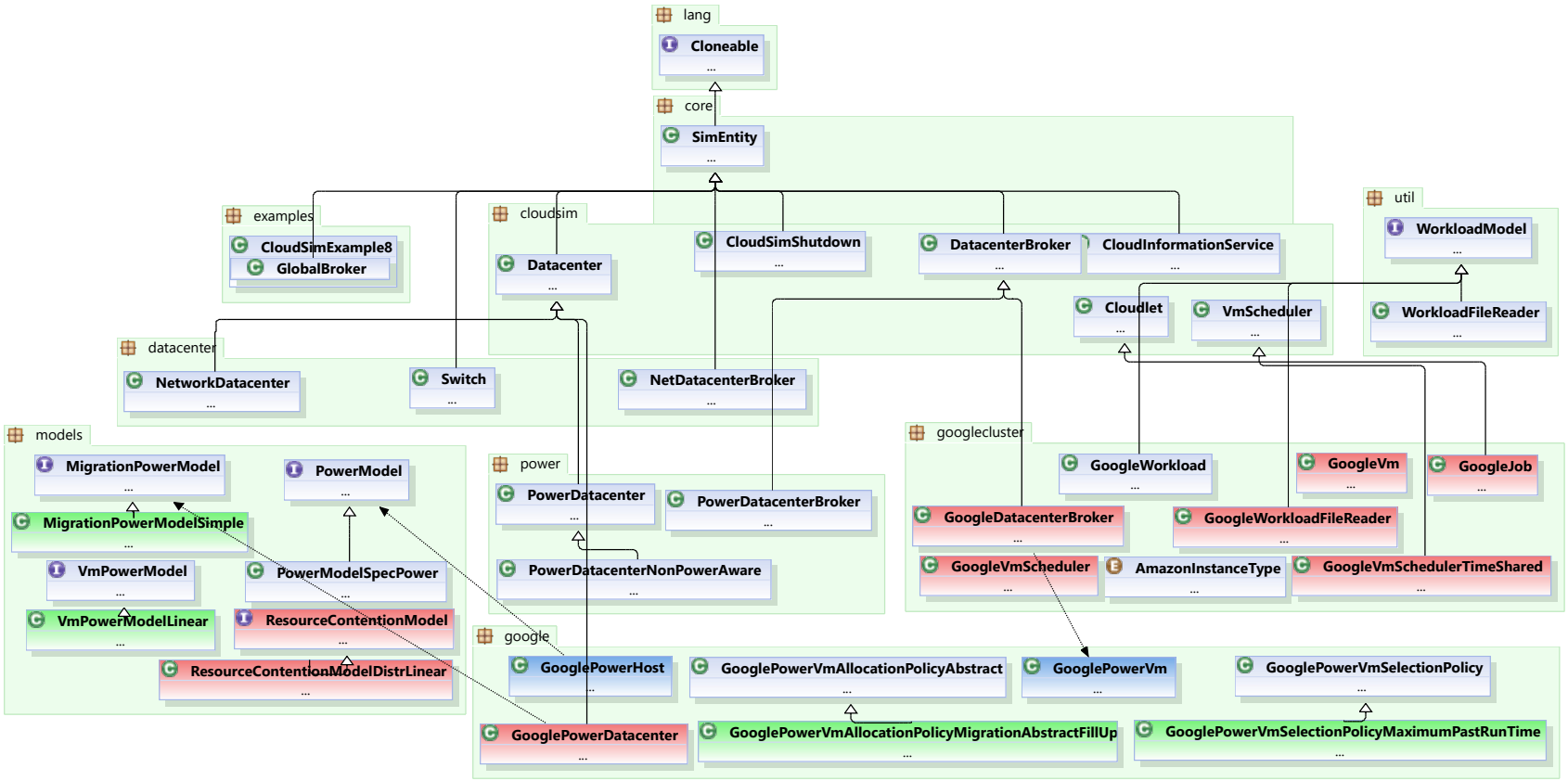

Fig. 7: PerficientCloudSim class diagram [The classes shown in red, green or blue inside *google* and *googlecluster* packages are extensions to the classical CloudSim default classes. In *util* package, we extend the **WorkloadModel** with **GoogleWorkload** to read the Google cluster data [13], the green classes in *google* package are the proposed allocation and migration policies while the green class in *models* package is the migration energy consumption model – the **VmLevelHostPowerModel** is implemented inside the *PowerModel* class – the blue *GooglePowerVm* class extends the *PowerVm* class to account for VMs past runtime and the blue *GooglePowerHost* class extends the *PowerHost* class to account for host reconfiguration costs. Similarly, the *ResourceContentionModel* class implements the performance loss in workloads due to co-located VMs – this class can further be extended to model the desired levels of degradation in workload performance] – the word "Google" cannot make PerficientCloudSim a specific purpose simulator, it can be used to replay other traces

(at runtime). Readers who are unaware of the basics of CloudSim, should read [1] first to appreciate how these classes relate to each other and what they are supposed to do. Sec. 5.8 summarises various additional and extended classes of the PerficientCloudSim package.

5.8 PerficientCloudSim Extra Classes

Following are the major classes that were extended to add additional capabilities in the classical CloudSim:

AmazonInstanceType: offers several types of instances that relate to various Amazon EC2 instances[14]. This class can be easily modified to add and delete additional instances along with their prices.

GoogleDatacenterBroker: models and provides support for users - resources interaction, reads VMs details from text files, selects a suitable and cheaper instance, creates VMs at runtime, and optimises the datacenter state through offering support for resource management and workload consolidation.

GooglePowerDatacenter: offers several options to control migrations, for example, migrate relatively long running VMs, only; and adjust VMs/workload performance when they are being migrated among various heterogeneous hosts.

GooglePowerHost: provides support for modelling platform heterogeneities (variations in runtimes due to CPU architectures and co-location). The heterogeneity parameters in terms of runtimes (i.e. mean and standard deviation) must be specified for each host. This class also models the resource interference (performance) or contention – VMs with similar workloads that compete for similar resources experiences significant performance loss (the co-location interference is highly correlated with the total number of VMs running on a particular host) [11]. Note that, the performance interference model (in terms of regression line equation) is directly integrated into the *GooglePowerHost* class.

GooglePowerVm: provides a way to store previous runtimes of VMs in order to trigger appropriate scheduling and consolidation decisions.

GoogleVmScheduler: is responsible for the implementation of the host heterogeneity through allocating and deallocating CPU (MIPS) to run a particular workload. Moreover, this class also implements the migration performance model [51]. Similarly, the resource contention model also feed loss or improvement in performance to this class in order to update processing of the co-located VMs.

GooglePowerVmAllocationPolicyAbstract: provides support for energy, performance and cost efficient resource management through VM allocation, datacenter optimisation with consolidation. The optimisation module is responsible to trigger effective workload placement and migrations. Various allocation policies may be used to extend this class.

GooglePowerVmSelectionPolicy: offers support for implementing various VM selection policies that are used to select appropriate VMs for migrations from under-utilised and over-utilised hosts.

GoogleJob: denotes a task (*cloudlet*) that belongs to a particular workload. Several additional job characteristics, such as execution time, required CPU and memory capacity, were added into this class.

[14] https://aws.amazon.com/ec2/instance-types/

GoogleWorkloadFileReader: reads tasks information from a text file, either in chunks or as a whole. This class can read very large files through offering a way to clear the heap space at regular intervals.

GoogleWorkload: extends the *WorkloadModel* class in order to account for tasks details in the Google trace [13].

MigrationPowerModel: is used to model the energy consumption of a VM migration process. Moreover, this class can be extended with other appropriate models.

MigrationPowerModelSimple: this class implements a very simple, but, reasonably accurate model to measure the energy consumption of a migrated VM, as suggested in [51]. In the proposed model, energy consumption ($Cost_{mig}$) is strongly dependant and directly proportional to the size or amount of the VM data (measured in MBs) being copied over the available network bandwidth (MB/s), given by Eq. 10.

$$Cost_{mig} = 0.512 \times VM_{data} + 20.165 \tag{10}$$

where the VM_{data} denotes the VM memory (in case of live VM migration) plus ephemeral storage (in case of non-shared block VM migration). Note that, $Cost_{mig}$ and VM_{data} are measured in Wh (Watt hour) and MBs (Megabytes), respectively. More accurate VM migration energy consumption models, as demonstrated in [30] can also be integrated into this class. Maio et al. [30] suggest that the energy consumption of both source and destination hosts is affected during a VM migration. Therefore, it is essential to account for churn in energy consumption at both source and destination hosts.

MigrationPerformanceModel: is used to model the workload performance loss, due to a VM migration, on: (i) migrated VM; (ii) source host; and (iii) destination host. Moreover, this class can be extended with other appropriate models.

MigrationPerformanceModelSimple: extends the *MigrationPerformanceModel* class with a simple technique. Note that, during a VM migration, workloads running both at source and destination hosts may experience trivial performance loss. We believe, such loss in performance is considered in the resource contention model. This class implements a simple performance model from [50], [51], in respect of migrated VM, source and destination hosts.

ResourceContentionModel: this class provides support for adding a particular resource contention model that denotes performance loss or gains due to co-located VMs. The increase or decrease in MIPS are fetched to the *GoogleVmScheduler* class in order to update VMs processing, accordingly. This parent class can further be extended to implement the desired levels of performance loss in VMs workloads.

ResourceContentionModelDistrLinear: This class implements the resource contention model from [11], in terms of performance degradation due to: (i) CPU heterogeneity (through runtimes distributions mapping); and (ii) number of co-located VMs (linear relationship), as described earlier in Sec. 5. Moreover, for each application L is taken from Fig. 4.

$$T_{rc} = T_t + L \tag{11}$$

where T_t denotes distributions of variations in runtimes due to resource contention (co-located VMs) - a part of Eq. 9.

VmLevelHostPowerModel: models and estimates the energy consumption of a single VM (E^h_{vm}) using the linear relationship between CPU usage, fraction of hosts resources allocated to the VM, and energy. The VM with larger resources consumes more energy as compared to a VM with

smaller resources [3], [12], [52].

$$E^h_{vm} = \frac{P^h_{idle}}{N} + W^h_{vm} \times (P^h_{busy} - P^h_{idle}) \times U^h_{vm} \tag{12}$$

where N is the total number of VMs on a particular host h, P^h_{idle} and P^h_{busy} are the energy consumed when h is idle (0% utilised) and fully utilised, respectively. Further, W^h_{vm} are the host resources (cores) allocated to the VM and U^h_{vm} is the VM utilisation level. Note that, W^h_{vm} ensures that each VM is accounted for energy consumption in the amount of resources they have provisioned - the more resources (or CPU cores) they provision the more energy they consume and vice versa. A more accurate model will also distribute the idle energy consumption among co-located VMs according to their W^h_{vm} and not N.

6 Performance Evaluation

In order to evaluate the performance and working mechanism of the PerficientCloudSim, we use a simple experiment. The experiment runs a particular workload on various heterogeneous servers and VMs using several VM allocation and consolidation with migration policies. The VM allocation and server consolidation can be assumed as a kind of bin-packing problem. Such problems can be solved using various heuristic algorithms. Albeit, heuristics may not ensure optimal results, however, they are fast enough for achieving the task i.e. VM placement and, therefore, more realistic in large-scale systems [53]. It is reasonable to assume an analogous VM packing problem as switching from an initial datacenter state to an ideal state, which should be one using the fewest hosts [14]. We achieve a datacenter state through the implementation of various scheduling heuristics, with VM packing then needing to guarantee energy efficiency and performance. To evaluate the effect of this, we consider dynamic consolidation using migrations. As described in [54], dynamic consolidation can be achieved either using a single threshold or double threshold values. In this paper, we use two threshold values i.e. a lower one and an upper one. If the utilisation level of a particular host decreases below 20% (lower threshold) or increases above 80% (upper threshold), then all the VMs running on this particular host are being migrated to other hosts.

Evaluation Metrics To evaluate the performance of the proposed policies, we consider several, essential, performance evaluation metrics. Two most important of these metrics are the total energy consumption (measured in KWh) of physical hosts and the workload total runtime (i.e. total execution time measured in minutes). The energy consumption was calculated according to values taken from the SPECpower benchmarks and the workload runtimes. Another metric is the total number of migrations started during the datacenter state optimisation. Other metrics include the SLA Violation (SLAV), which denotes how much SLA was violated in response to the VM migration, performance degradation and server down time. An SLA violation occurs if a particular VM cannot receive the requested amount of MIPS. This might happen if several VMs share the same host and demand for performance that cannot be ensured due to resource consolidation or migration. The metric SLAV denotes the level by which QoS requirements are being violated, due to management policies, that were determined between the provider and users. We assume that service providers pay penalties to users if SLAs are being violated [55]. The key metrics are the energy consumption and SLAV; but, these metrics are usually adversely interrelated as energy can typically be decreased by the increased level of SLA violations. The main objective of resource management is to reduce the energy consumption with minimum SLA violations. Therefore, a combined metric i.e. Energy and SLA Violations (ESV), suggested in [8], is used to capture both energy consumption and the level of SLAV [Table 9].

6.1 Experimental Setup

The simulated datacenter comprises 800 heterogeneous physical hosts, which consists of 50% HP ProLiant ML110 G4 servers, and 50% HP ProLiant ML110 G5 servers. Various characteristics of

these servers are described in Table 6. The energy consumption of these servers were considered according to SPECpower[15] benchmarks. Note that, these values for energy consumption relate to results of a particular benchmark test [i.e. SPECpower_ssj2008, published in 2011]. Furthermore, as these benchmark results concern a very specific type of workload [Server Side Java Operations Per Second – SSJ-OPS] – this may not be suitable to represent a more general scenario. However, in this paper, we use these results just for comparison purposes. Note that, simulation of less powerful CPUs is more beneficial as lighter workload can easily overload the hosts; and, therefore, creating more chances for VM consolidation. That's why servers with more cores were not preferred. Note that, similar to the classical CloudSim, millions of instructions per second (MIPS) is the core performance metric used in the PerficientCloudSim. Note that, for resource heterogeneity and co-located VMs contention we consider only the CPU and workloads; whereas, other hardware resources can also affect VM performance as described in Sec. 5.4. Therefore, using MIPS ratings as a single measure for various performance metrics should not be the only scale of measurement under different situations.
The hosts' CPU frequencies were mapped onto MIPS ratings: 1,860 MIPS for each core of the HP ProLiant ML110 G5 host, and 2,660 MIPS for each core of the HP ProLiant ML110 G5 host. The performance parameters of these two hosts were assumed as described in Table 7. Moreover, each host has 1 GB/s network bandwidth, half is used for VMs communication and the other half is available for migration purposes [8]. All hosts in the simulated datacenter were fully connected [using the CloudSim's default network topology – *topology.brite*] and, therefore, migrations could occur between any pair of hosts. We are aware that data of each VM transferred over a communication link should also cause some latency. However, in this paper, this latency was not directly considered during the simulation experiments; that would certainly affect simultaneous migrations occurring over the same communication link. We assume that the performance model [49] which we have used in our experiments has already considered this latency; and, therefore, we have translated these delays into the execution times of the migrated VMs.

Table 6: Host types and their characteristics – P_{idle} and P_{max} denote the power consumption of server when they are idle (0% utilised) and maximum utilised, respectively; this linear power model is suggested to be more than 90% accurate for certain workload types [2]

Host type	CPU Model	Speed (MHz)	Num. of cores	RAM (GB)	P_{idle} (Wh)	P_{max} (Wh)
HP ProLiant ML110 G4	Intel Xeon 3,075	1,860	2	4	86	117
HP ProLiant ML110 G5	Intel Xeon 3,075	2,660	2	4	93.7	135

The features and characteristics of the various VM types [as shown in Table 8] resembles to Amazon EC2 instance types with the only exception that all the VMs were single-core, as the workload data come from single-core VMs [8]. For the same reason, the amount of RAM was divided by the number of cores for each VM type, as shown in Table 8. Moreover, the computational requirements of each VM, which should be expressed in MIPS, were also mapped to CPU frequencies e.g. 1000 MIPS means 1GHz of host's CPU. We further assume that each instruction require at most one CPU cycle in order to be executed. Primarily, the VMs were assigned agreeing to the resource desires defined by the VM types. But, throughout the lifespan, VMs consumed fewer resources agreeing to the input workload data, making chances for dynamic VM consolidation. In our simulations, every VM was assigned a single workload trace randomly with the objective to stress the consolidation algorithms, having no memory limits. The energy consumption of a VM is computed

[15] https://www.spec.org/power/

Table 7: Hosts performance parameters (runtimes in hours) for Bzip2 workload (mapped to Google's cluster tasks) that were calculated from simple visual inspection, using log-normal distributions, as described in [56]

Host type	performance parameters			
	min	max	mean (μ)	std. deviation (σ)
HP ProLiant ML110 G4	15.70	19.99	17.42	1.145
HP ProLiant ML110 G5	19.04	24.5	20.78	1.267

Table 8: VM/instance types and their characteristics

VM type	CPU (MHz)	RAM (GB)
High-CPU Medium G4	2,500	0.85
Extra Large	2,000	3.75
Small	1,000	1.7
Micro	500	0.613

using the model in Eq. 12. The performance interference models were computed using the real data, as described in [11], with regression line equations.

6.2 Experimental Results

The VM allocation policy proposed in [56] was implemented as an abstract allocation policy in the PerficientCloudSim simulator, and the results of three various VM allocation techniques, i.e., Mbfd [8], Epobf [57], and Cbfd [56], and three migration policies i.e. MMT [8], Cmur, and Cmur+Mcp [56], were being evaluated using several plausible assumptions and workloads. The allocation policies are based on various heuristic approaches that select appropriate VMs, in terms of various objectives such as energy efficiency, and performance, to run a particular workload. For example, the Mbfd allocation scheme that extends the best fit decreasing heuristic, runs the workload on those VMs that consume less energy. The migration policies look for consolidation opportunities through examining the hosts' utilisation levels; and if certain hosts utilisation levels are approaching to some pre-defined threshold values, then suitable VMs are being migrated from under-utilised/over-utilised hosts to other average utilised hosts. We use two statistical techniques to identify under-utilised/over-utilised hosts i.e. Local Regression (LR) and Thresholds (THR) [8]. The later one uses a lower and an upper static threshold values; and the former one uses a predictive mechanism to estimate the future workload utilisation. The workload consists of several thousands tasks (approximately 18k), where each task was allocated to a single VM. The results are shown in Table. 9. The focus of the experiment is not on various resource management policies, their energy consumption and workload performance; but, to show that PerficientCloudSim successfully accounts for workload performance degradation and improvements which are essential while moving workloads around heterogeneous resources [3].
From the above experiments, we can see that the allocation policy, as suggested in [56], could save up-to 9.38% more energy than the "Mbfd" policy for different host overload detection techniques such as LR, and THR [8]. Furthermore, we also observed that a combination VM allocation "Cbfd" and migration policy "Cmur" could save more energy, particularly, in large-scale systems. The mean results (energy consumption, average number of migrations, SLAV and ESV) for various workload traces, various allocation and migration policies are shown in Table 9. The VM allocation technique (Cbfd) combined with the migration technique (Cmur) has significantly improved (∼41.25%) the ESV (product of energy and SLAV metrics) i.e. 2,495.14 is reduced to 1,465.86 – approximately 41.24% overall improvement. This clearly shows that these methods outperform

Table 9: Total energy consumption, workload runtimes and number of migrations [THR: uses an upper threshold (0.8) for host overload, MBFD, default VM allocation policy in CloudSim, MMT: minimum migration time, CBFD: COMBINED BFD, CMUR: COMBINED MUR, MCP: MIGRATION CONTROL POLICY – 'best' approaches are shown in bold]

Policy			Avg. energy	Avg. exe.	Avg. number	SLAV	ESV
Host overload	ALLOCATION	MIGRATION	E (kWh)	time	of migrations	%	(E×SLAV)
No migrations							
MBFD [8]			2,318.6	419.2	0	0	-
EPOBF [57]			1,503.7	411.3	0	0	-
CBFD [56]			1,493.2	409.8	0	0	-
CBFD+			1,145.8	398.6	0	0	-
With migrations							
THR	MBFD	MMT	1,246.069	431.8	26,779	10.14	2,495.14
	EPOBF	MMT	1,201.9	428.9	25,543	10.23	2,062.44
	CBFD	MMT	1,192.483	427.9	37,100	10.135	1,950.82
	CBFD+	CMUR	**1,157.89**	397.8	**19,367**	**10.07**	**1,589.95**
	CBFD+	CMUR+MCP	**1,157.91**	398.6	**11,187**	**10.01**	**1,580.68**
LR	MBFD	MMT	1,233.638	417.5	28,389	9.6	2,242.92
	EPOBF	MMT	1,189.01	417.9	29,452	10.1	1,909.0
	CBFD	MMT	1,170.816	399.9	31,202	9.5	1,622.75
	CBFD+	CMUR	**1,154.79**	400.1	**18,530**	**9.47**	**1,465.86**
	CBFD+	CMUR+MCP	**1,152.21**	402.3	**10,091**	**9.36**	**1,424.69**

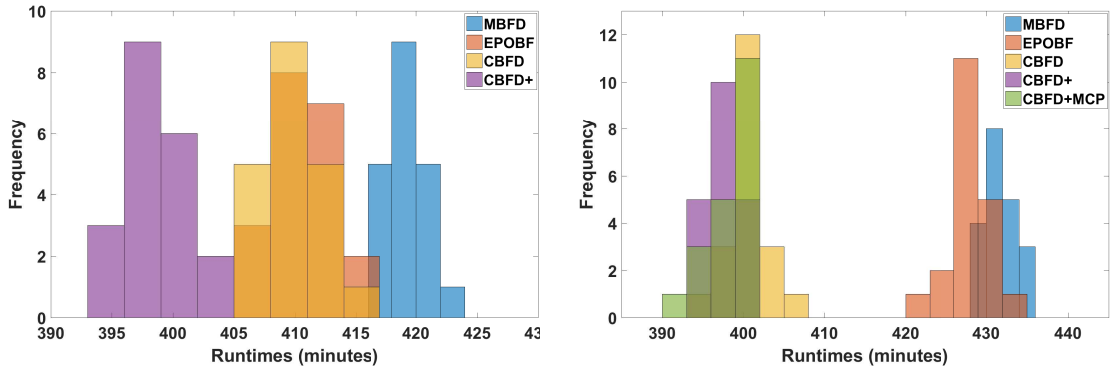

Fig. 8: Variations in workload performance due to platform heterogeneities, performance interference and migration downtime [left: no migration – right: with migration] – lower values denote the 'best' performance

the well-known Best Fit Decreasing (BFD) heuristic. Table 9 also shows that the workload performance could be, possibly, improved (1.16%) at slight increase (0.013%) in energy consumption using the migration control policy "MCP". Moreover, the number of migrations could be significantly decreased that could be approximately 45.54% less than using the CBFD policy combined with CMUR technique. Lastly, the energy savings achievable through efficient allocation policies are higher than using consolidation with migration techniques. A simple intuitive reason for this could be resource heterogeneities and that migrations are costly.

These experimental results, as described in [11], [12], demonstrate significant performance variations among similar VMs that are largely caused by the hardware (CPU) heterogeneities and

performance interferences among various VMs – co-located on same host that compete for same resources. These variations in workload performance affect runtimes and, therefore, costs. Resultantly, potential performance and, therefore, cost benefits can be achieved in infrastructure clouds through: (i) performance aware resource allocation – launching appropriate VMs over suitable CPU platforms; and (ii) performance aware consolidation – minimising the performance interference among VMs through migrations. In respect of (ii), if performance of a particular workload is not satisfactory on certain resources; it might also be migrated to other machines that offer better performance. In respect of (i), if users always look for efficient resources (where their workloads perform better) this might result in the well-known *instance seeking problem* [9]. Instance seeking problem may affect infrastructure utilisation levels, energy efficiency, and service providers' revenues. To further explain these variations in workload performance (runtimes), we describe other experiments and empirical evaluation in Sec. 6.3.
Next, we investigate total cost savings in a heterogeneous datacenter through energy, performance and cost "EPC-AWARE" allocation and migration i.e. if energy, performance efficiencies are assured then allocation and/or migrations decisions are taken. The total electricity bill, user monetary costs and costs savings (in US dollars - $) are described in Table 10. For these analyses, we assume a PUE[16] of 1.10 and energy price of 0.88$ per KWh[17] that mimic a Google datacenter located in the Oklahoma state, USA. Moreover, we assume that the users bills are computed at 0.0017$ per second[18]. In case, an SLA violation occurs for a particular VM, the providers pay a penalty of $1.0 - which is subtracted from users' bills and total costs savings. The service providers could save up to 20.14% costs using the "EPC-AWARE" allocation and consolidation technique instead of using the first fit policy.

Table 10: Costs savings [energy and users monetary costs are described in US dollars]

Allocation policy	Energy costs ($)	Performance (minutes)	Users monetary costs ($)	Total costs savings (%)
NO MIGRATIONS				
FIRST FIT	2090.54	418.88	42.73	2133.25
EPC-AWARE	1600.21	400.09	40.81	1641.02
MIGRATE ALL				
FIRST FIT	1929.43	422.87	43.13	7.53
EPC-AWARE	1633.56	415.73	42.4	-2.13
EPC-AWARE MIGRATIONS				
FIRST FIT	1454.38	409.78	41.8	29.86
EPC-AWARE	1269.92	398.84	40.68	20.14

6.3 Results Discussion

How hosts are addressed or physically ordered in a datacenter has an impact on the scheduling and migration approaches. For example, if available hosts are in increasing order of their energy efficiencies and a first fit (FF) algorithm is used for allocation, then the total energy consumption would be different if hosts are arranged in decreasing order of their energy efficiencies. Similarly, if hosts are ordered based on their performance efficiencies (CPU models), then the existing trade-off between energy and performance would also differ for various scheduling policies (VM allocation and migration) and different workloads. Therefore, we discuss the impact of host ordering and

[16] https://www.google.co.uk/about/datacenters/efficiency/
[17] https://www.eia.gov/electricity/monthly/
[18] https://aws.amazon.com/ec2/pricing/

scheduling approaches on energy efficiency and workload performance. Each allocation and migration policy selects a host to run a VM and the starting point for such decisions could produce variations in energy, performance and hence cost. For example, if the initial ordering were reversed or changed (physically or logically through the scheduling approach), this may potentially alter our results and outcomes.

To determine the impact this may have on infrastructure energy consumption and workload performance, we run the experiments from our previous work [12] three times with three different initial physical orders for hosts: (i) no order (NR) – random; (ii) increasing order based on E_f (INC); and (iii) decreasing order based of E_f (DEC). The E_f for each host is calculated by dividing its maximum power consumption by the number of slots (cores or GCEUs) it has. For example, if we have three hosts (h_1, h_2 and h_3) having $E_f^{h_1} = 4$, $E_f^{h_2} = 1$ and $E_f^{h_3} = 2$ where smaller E_f represents lower energy efficiency. Then the orders would be: NR – [h_1, h_2, h_3], INC – [h_2, h_3, h_1] and DEC – [h_1, h_3, h_2]. Note that, E_f denotes the overall energy consumption of a host, but, not for a particular VM. For WORKLOAD (3), unfortunately we were unable (due to stranded resources)[19] to schedule all VMs for NR and DEC, therefore we increased the number of hosts from 12,583 to 18,583. This demonstrates that the order of hosts does matter and could affect energy efficiency and workload performance (hence cost).

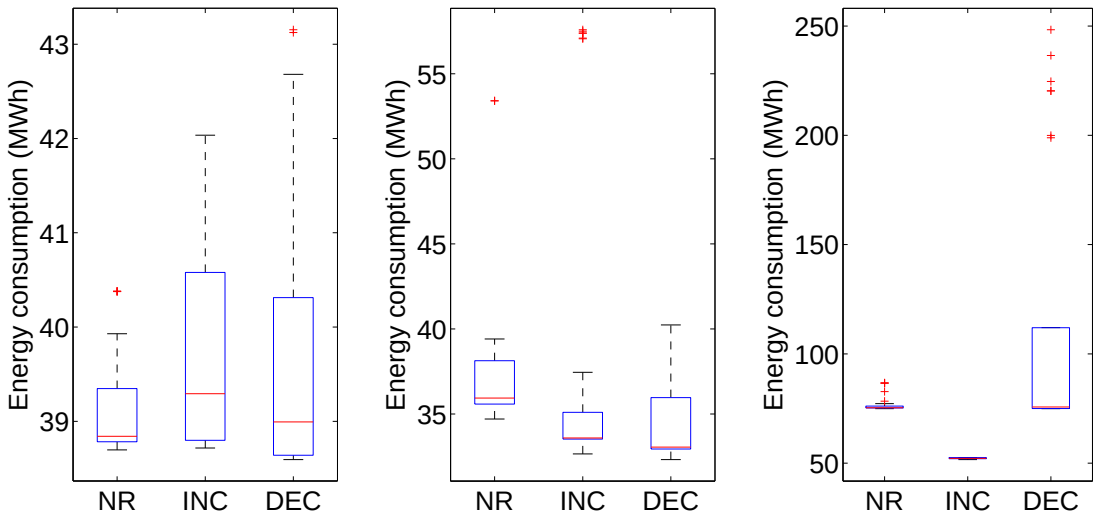

Fig. 9: Variations in total energy consumption when hosts are physically ordered based on their levels of energy consumption – efficiency factor (E_f) [left - WORKLOAD (1), middle - WORKLOAD (2), right - WORKLOAD (3)] – note that, these experiments were performed using the simulation set-up as described in [12] [NR: no order, INC: increasing order of E_f, DEC: decreasing order of E_f]

Fig. 9 and Fig. 10 both demonstrate variations in energy consumption and performance (runtime) for different kinds of workload and hosts orders (based on their energy efficiency – E_f). These experiments suggest that the physical order of hosts could significantly affect the IaaS energy consumption and application performance depending on the workload type. For example, no ordering may be energy efficient for WORKLOAD (1), but, not much affective for WORKLOAD (2) and WORKLOAD (3). Similarly, the increasing order is energy efficient for WORKLOAD (2) and WORKLOAD (3), but, worst for WORKLOAD (1). Surprisingly, the no ordering is, on average, more

[19] the host that cannot be allocated due to a single resource unavailability - enough CPU is available but there is insufficient memory [58]

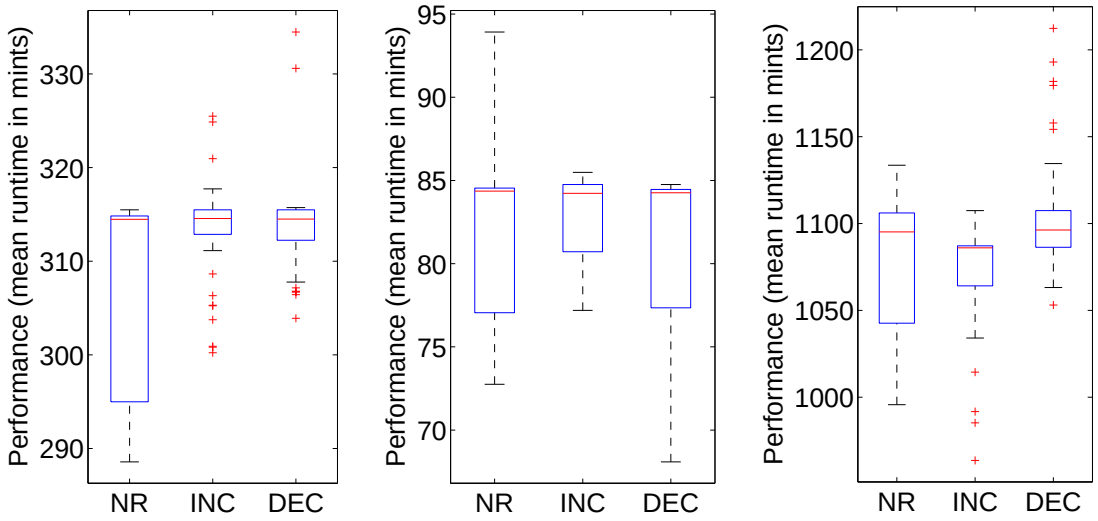

Fig. 10: Variations in runtimes (performance) when hosts are physically ordered based on their levels of energy consumption – efficiency factor (E_f) [left - WORKLOAD (1), middle - WORKLOAD (2), right - WORKLOAD (3)] – note that these experiments were performed using the simulation set-up as described in [12] [NR: no order, INC: increasing order of E_f, DEC: decreasing order of E_f]

performance efficient that other orders. This is due to the fact that cloud workloads are distributed over various resources; and the no ordering approach creates more opportunities to run workloads on more performance efficient hosts. Moreover, different scheduling policies offer different levels of energy consumption and workload performance for various combinations of hosts ordering and workloads. Here, ordering is discussed in terms of allocation policies that means logical addressing and is not a physical shift. However, this might be extended to: (i) putting hosts in different racks; and/or (ii) physically shifting hosts inside a rack [3]. Note that, the proposed cloud simulator "PerficientCloudSim" is used in our previous works such as [2], [12], [56], [59].

6.4 Generalisation of Outcomes

In order to demonstrate that PerficientCloudSim simulator can replay real workload traces from various cloud providers, we use the data provided in [15] from the Microsoft Azure cloud. Various properties and characteristics of the dataset are further described in [15]. The experimental parameters, datacenter set-up, hosts and VMs characteristics (in terms of energy consumption and resource contention) were kept the same, as described earlier in Sec. 6.1. However, the number of hosts, VMs, workload execution times and their utilisation patterns were kept varying over different experimental scenarios. Moreover, various combinations of VM allocation techniques (Round Robin - RR, First Fit - FF, Fill Up - FU) and consolidation with migration policies (no migration at all - NO, migrate all - ALL, migrate VMs if their migration costs are recoverable - CMCR) were assumed [14], [60]. Note that, the CMCR (i.e. Consolidation with Migration Cost Recovery) approach migrates relatively long-running VMs; using the theory of probability – VMs that have run for longer durations (before migrations) have higher chances to continue their execution even after their migrations to other hosts [14].
The results obtained, in terms of energy consumption and execution times (performance), are shown in Table 11. These results are averaged over ten different runs. The standard deviations (values followed by ±), when overlapped, mean that both approaches perform similar. Our evaluation, as

shown in Fig. 11, suggests that these outcomes and findings are largely consistent with our previous outcomes. When no migrations are considered, then the FU approach offers the best performance - as more energy and performance efficient hosts are utilised first. The ALL migration approach can be expensive than the NO migration approach; which favours no-consolidation. Consolidation of workloads could be affective and could save energy if unnecessary or costly migrations can be avoided. For example, the CMCR approach ensures significant energy savings and performance gains.

Table 11: Energy consumption and performance (averaged over ten runs) – the value after ± denotes standard deviation [FU performs better than RR]

Policy		Energy consumption	Performance
Allocation	Migration	(KWh)	(minutes)
RR	NO	3,042.67±211.35	574.10±3.47
	ALL	3,134.99±244.69	562.52±2.53
	CMCR	2,478.45±188.34	549.59±4.02
FU	NO	2,967.45±129.34	546.63±3.47
	ALL	2,898.02±231.44	531.41±4.01
	CMCR	2,100.23±101.56	528.04±4.29

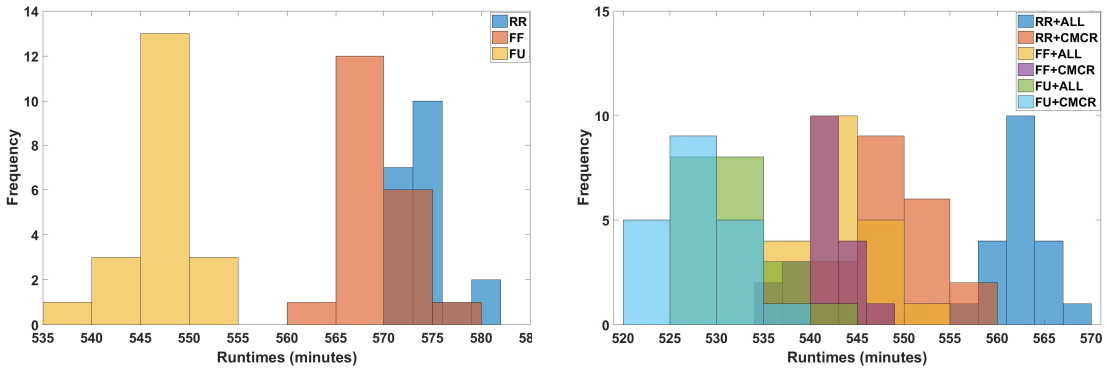

Fig. 11: Variations in workload performance due to platform heterogeneities, performance interference and migration downtime [left: no migration – right: with migration] – lower values denote the 'best' performance

To further generalise our outcomes and validate that the proposed simulator can simulate large-scale heterogeneous IaaS, we run several experiments with different datacenter sizes, VMs, and workload types (with various combinations). The three workload types i.e. W_1, W_2, and W_3 represent variations in CPU utilisation levels and their execution times, as described in Sec. 6.5. These experiments were performance using the FU VM allocation and migrate ALL approaches. Table 12 shows our observed results in terms of number of migrations, energy consumption and workload performance with respect to minimum (Min) and maximum (Max) across three runs. The outcomes suggest when a mix of different workloads are taken into account, larger variations may happen in the workload performance and, subsequently, in energy consumption. These findings are, largely, inline with our previous discussion that suggest and justify the scalability of the proposed simulator.

Table 12: Different workloads and their combination on various datacenter sizes

Workload type	Datacenter size (servers)	Number of VMs	Number of migrations		Energy usage (KWh)		Performance (Minutes)	
			Min	Max	Min	Max	Min	Max
W_1	3,000	50k	528	672	71.22	71.26	19.34	19.56
W_2	6,000	70k	500	563	167.13	167.78	88.2	90.01
W_3	9,000	0.1m	487	501	295.73	311.69	171.9	201.56
W_1, W_2	6,000	0.12m	1,001	1,098	171.81	175.51	101.23	111.56
W_2, W_3	9,000	0.17m	1,792	1,891	203.25	352.75	277.89	279.01
W_1, W_3	6,000	0.15m	934	1,056	203.49	207.75	189.56	201.44
W_1, W_2, W_3	12,000	0.22m	2875	3221	578.79	584.83	391.67	399.7

6.5 Impact of Workload Types on Performance

In this section, we investigate the impacts of various workload types, such as CPU, memory and disk intensive, on resource provisioning, migration and simulation behaviour. Furthermore, we also study the behaviour of threshold values used to optimise the state of the datacenter to reduce energy consumption and improve performance. These workloads were classified according to their demand for resources. For example, the CPU utilisation of the CPU intensive workloads (W_1) varies between 80% to 95%; while other resource demands were kept lower than 30%. Similarly, for memory intensive workloads (W_2) their CPU utilisation levels were kept 50% to 75% while memory usage was 80% to 95%. For disk intensive workloads (W_3), the CPU demand varies between 20% to 45% while disk utilisation were kept higher than 80%. Besides these, workloads may be classified as batch and production services. These synthesized workloads were modelled using a normal distribution function based on findings from real cloud workload traces such as Google [12] and Microsoft Azure clusters [15]. As, we only account for CPU heterogeneity; therefore, we discuss our results in terms of CPU intensive workloads. A first fit (FF) VM allocation policy combined with different approaches to migrations (such as NO, ALL and CMCR) was used while other experimental setup was the same, as described previously in Sec. 6.1.

Table 13: Performance of three different types of workloads (averaged over ten runs) – the value after ± denotes standard deviation in workload runtimes

Policy		Workload type	CPU utilisation (normalised)	Performance (minutes)
Allocation	Migration			
FF	NO	CPU intensive W_1	0.8 - 0.95	935.45±11.68
	ALL			912.35±16.32
	CMCR			901.21±17.56
	NO	Memory intensive W_2	0.5 - 0.75	1,327.21±6.21
	ALL			1,298.32±11.89
	CMCR			1,213.67±12.73
	NO	Disk intensive W_3	0.2 - 0.45	1,599.31±4.99
	ALL			1,556.12±10.23
	CMCR			1,500.98±8.45

Table 13 summarises the results which we achieved for three different types of workloads. We observed larger variations for CPU intensive application while their level of performance were optimal as compared to other two application types. The performance improvements are essentially due to the workload demands - if CPU resources (more formally MIPS in the notion of CloudSim) are allocated in high amount; then workload will finish quickly. Furthermore, the variations in runtime are significant due to the fact that workloads compete for CPU resources (contention) or co-located. Workloads' performance could be more severely affected with migrations due to platform heterogeneities. When CPU demand is lessened (in case of memory and disk intensive workloads), then besides their increase runtimes (i.e. lower performance) smaller variations can be observed. Unfortunately, the memory and disk usage models and heterogeneities are currently not supported in the proposed simulator. Another important factor is the predefined threshold values ($U_{threshold}$ - upper threshold, $L_{threshold}$ - lower threshold) which are setted to optimise the state of the datacenter. The $U_{threshold}$ can avoid overloading hosts while $L_{threshold}$ can help in switching off idle hosts to increase performance and energy efficiencies, respectively. However, it is a tedious job to set appropriate threshold values. Moreover, static threshold may be appropriate in situations where the workload is stationary; however, for dynamic workloads (as usually happens in clouds) adaptive thresholds are more affective [8].

Table 14: Impact of CPU utilisation thresholds (averaged) on energy consumption – the value after ± denotes standard deviation

Allocation Migration	Workload type	(Normalised) $L_{threshold}$	(Normalised) $U_{threshold}$	Number of migrations	Hosts switched-off	Energy (KWh)	Performance (minutes)
FF + ALL	CPU	0.1	0.9	101	12	467.34±5.2	909.43±15.84
		0.2	0.8	167	34	424.98±4.8	912.35±16.32
		0.3	0.7	122	19	441.46±6.1	907.01±14.67
	Memory	0.1	0.9	95	13	723.09±11.3	1,291.67±12.01
		0.2	0.8	189	37	699.34±9.9	1,298.32±11.89
		0.3	0.7	211	55	584.24±9.7	1,304.11±10.5
	Disk	0.1	0.9	454	73	927.99±1.8	1,558.6±10.1
		0.2	0.8	489	72	923.77±2.1	1,556.12±10.23
		0.3	0.7	532	73	924.69±2.0	1,555.78±9.4

For example, setting a higher value for $U_{threshold}$ would essentially affect the workload performance; while a lower value would increase the unneeded migration opportunities. Similarly, a high value for $L_{threshold}$ will not guarantee appropriate migrations; while a lower value would unnecessary switch off hosts. Similar to datacenter temperature level, which are usually kept as low as 55°F (threshold), a one degree increase could save significant amount of energy[20]; hosts' utilisation thresholds have significant impacts on energy efficiency and workload performance. In our case, an upper utilisation threshold during the allocation process ensures that hosts are not over-utilised – since a VM can only be placed if there are enough resources and the utilisation threshold is not exceeded. We carried out several experiments while adjusting the lower and upper utilisation thresholds during the consolidation. Table 14 describes variations in energy consumption and workload performance along with total number of migrations and hosts switched off to save energy when different thresholds and workload types are taken into account. We observed that for CPU intensive workloads, the impact of thresholds might be severe and, therefore, difficult to guess the most appropriate ones.

[20] https://blog.se.com/datacenter/2013/02/01/4-ways-to-reduce-energy-consumption-in-any-data-center/

In such situations, adaptive thresholds may be more affective. However, workloads that do not demand for CPU frequently, the impact could be trivial or even negligible. The lower one affects the number of switched off hosts (energy efficiency) while the upper one has great impacts over the workload performance (possibly due to resource contention). However, this needs further research and investigation.

7 PerficientCloudSim Accuracy

Numerical results which are computed using simulators may not be guaranteed. Furthermore, it is essential to ensure that the simulator produces accurate results that are similar to those computed in a real test-bed. Unfortunately, a real test-bed is not available to us. Therefore, two different techniques are most widely used in order to ensure that the simulator is just like a real test-bed and can produce accurate results: (a) map simulated models to analytical, simplified, models; and/or (b) map the simulated models with other simulated, but verified, models [2]. The fastest method to ensure the accuracy of our results is to map them with simplified, analytical results which were achieved in a real test-bed. Nevertheless, this is not ensured that a simplified model must be always available. Moreover, this method can only be used in simple scenarios, because locating analytical models for real problems could be difficult or, at least, impossible; and perhaps, this might be a reason for using numerical simulations [1].
Fortunately, various modules of our simulator, *the VM level host efficiency model, performance interference, migration energy cost* have already considered in real test-beds. For example, Ibrahim et al. [45] proposed a similar power consumption model and produced their outcomes at Leeds cloud test-bed at the University of Leeds, UK[21]. Recently, the authors extended their model in [45] to account for heterogeneous workloads [46]. Furthermore, Xu et. al [11] experimentally evaluated performance interference due to co-located workloads. Since, we use the *VM level host efficiency, performance interference, migration energy cost* as baseline models for various resource placement and migration algorithms in the PerficientCloudSim simulator. Therefore, if we can map results obtained over these baseline models to analytical results, as demonstrated in [45], [46], [11]; then, it is possible to ends up with a method to approximate the simulated results and PerficientCloudSim's accuracy to a real test-bed.

Real Test-bed In [45], [46], a cluster comprising four homogeneous hosts is considered for evaluation of the power model. Every host comprises a 4-cores X3430 Intel Xeon CPU, that runs at 2.40GHz clock speed, and 8GB of memory (RAM). All hosts are inter-connected through a Gigabit/s Ethernet link. To measure the energy consumption, a watt meter is attached to every host. According to authors, same hosts are part of the Leeds test-bed that comprises a cluster of commodity servers (Dell), as described in [45]. If we knew the idle (P_{idle}) and peak (P_{max}) energy consumption of various hosts in the Leeds test-best, then, it is possible to use a linear power model (to measure energy usage) which is assumed as more than 90% accurate [8]. Unfortunately, with the exception of CPU, memory and disk capacities, we do not know further information of these machines. Therefore, to simulate the energy consumption of hosts and VMs, we use SPECpower benchmarked values (in terms of real energy consumption, for particular hosts, measured in Wh at certain level of CPU utilisation e.g. 0%, 10%, 20% ... and 100%)[22]. Moreover, we assume that the VM workload demand for CPU cores is random (stochastic) – utilises the requested resources exactly, as described in [1].
In another experiment, the authors evaluated that increasing the total number of VMs in a particular host (co-located) affects its overall energy consumption that can be accurately modelled through a positive linear relationship. This is in-line with [11], which describes a linear relationship between the number of co-located VMs, on a particular host, and performance interference. The linear growth in energy consumption (with respect to the number of co-located VMs) and the total

[21] https://institutes.engineering.leeds.ac.uk/computing/research/distsys/facilities.shtml
[22] https://www.spec.org/power_ssj2008/results/res2010q2/power_ssj2008-20100420-00252.html

energy consumption of the host can be described, is given by Eq. 13:

$$\mathcal{Y} = 10.127\mathcal{X} + 72.587 \ (idle\ power) \qquad [\mathcal{R}^2 = 0.9996] \tag{13}$$

where $\mathcal{X}$ is the total number of co-located VMs that collectively run on a particular host h and $\mathcal{Y}$ is the power consumption of h. Using the host power model (virtualised) in [14], the VM-level power consumption of all 4 VMs collectively is almost similar to host's total energy consumption (similar utilisation levels) [45]. Moreover, Xu et al. [11] verified a linear relationship between the number of co-located VMs and performance interference, as described in Sec. 5.5. Various linear equations and parameters for the performance interference model are shown in Fig. 4. For example, resource contention of "Sort" application on "E7420" CPU platform is given by:

$$\mathcal{Y} = 6.9429\mathcal{X} + 4.4 \qquad [\mathcal{R}^2 = 0.9792] \tag{14}$$

Since, we also use similar VM-level host power and performance models (virtualised) as a baseline for various resource allocation and management policies, in the proposed PerficientCloudSim simulator. Therefore, if the results obtained in PerficientCloudSim for the power and performance models have similarities to outcomes obtained on a real test-bed in [45], and [11], then PerficientCloudSim could be considered as accurate and verified.

Simulated Model We simulated a simplified model of the above numerical experiment in a real test-bed. We assume three baseline models i.e. (a) the host level energy consumption; and (b) the linear relationship between energy consumption and number of VMs on a particular host; and (c) the linear relation ship between the number of co-located VMs on a particular host and level of performance interference. Note that, the exact idle and peak power consumption of the real test-bed resources is unknown. In our experiments that use SPECpower benchmarks, the idle power consumption P_{idle} of every host is approximately 61.3 Watts per hour (Wh). For simplicity, we ignore the peak power consumption as it is useless here. If we assume the $\mathcal{Y}$ intercept, in the linear model i.e. Eq. 13, as the host' idle power consumption (72.587 Wh), then the regression line' slope ($m_1 = 10.238$) of the simulated model (i.e. Eq. 15) is very near to the slope ($m_2 = 10.127$) of the linear model in real test-bed experiments (Eq. 13). These slopes of regression lines describe a similar rate of change in the energy consumption with respect to the total number of co-located VMs on a particular host. Unfortunately, the type of workload running in VMs within the real test-bed. Therefore, we simulate stochastic workloads that have random behaviour in utilising the host' CPU resources. Note that, the stochastic utilisation model is already available in the CloudSim class file "*UtilizationModelStochastic()*". To validate the model over this random behaviour, we ran the same experiment ten times and, then, mapped the average values to those of the real test-bed results [2].

$$\mathcal{Y} = 10.238\mathcal{X} + 61.3 \ (idle\ power) \qquad [\mathcal{R}^2 = 0.9315] \tag{15}$$

In Eq. 15, $\mathcal{X}$ is the total number of VMs running on a particular host and $\mathcal{Y}$ is the total power consumption of the host. Furthermore, the simulated performance interference for the "E7420" CPU model was modelled as (where the intercept was explicitly defined as 4.4 to match with real model in Eq. 14):

$$\mathcal{Y} = 7.009\mathcal{X} + 4.4 \qquad [\mathcal{R}^2 = 0.9792] \tag{16}$$

The simulated results (host level energy consumption with respect to the number of co-located VMs) closely match the real-life values obtained in a real test-bed. Similarly, the performance interference model also mapped closely to that of the real test-bed. The relative error ($\mathcal{RE}$) which is the difference between the slopes of both linear models (m_1, m_2 - real and simulated) is less than 1.096. For the performance interference models, the $\mathcal{RE}$ is equal to approximately 1.01. The $\mathcal{RE}$ is computed as:

$$\mathcal{RE} = \pm\left[100 - \left(\frac{m_1}{m_2} \times 100\right)\right] \tag{17}$$

Therefore, compared to the real test-bed outcomes, our simulations of the simplified model (in PerficientCloudSim) and results can be assumed approximately 98.904% (100 - 1.096) – 98.99%

(100 - 1.01) accurate. With this accuracy, we can easily compute the expected errors in energy or performance efficiencies of various policies and, therefore, simulator. For example, the resource management policy "CMCR" [43], which is suggested approximately 3.66% more energy and 1.87% more performance efficient than the no migration technique, could potentially save approximately 3.66±0.04% more energy and is ∼1.87±0.019% more performance efficient than the no migration approach. Note that, the ±0.04 and ±0.019 are the $\mathcal{RE}s$ which are approximately 1.096% of 3.66 and 1.01% of 1.87 – computed as $\frac{1.096}{100} \times 3.66$ and $\frac{1.01}{100} \times 1.87$, respectively. However, with this it is understood that *"a model which works for simple scenarios is not assured to work for complex scenarios; however, a model which does not work for simple scenarios will absolutely not work for complex ones"* [2].

8 Conclusions and Future Work

To support and accelerate the research related to clouds, applications and services, it is essential that accurate software tools (simulators) are designed and developed to aid researchers and industrial developers. In order to examine and evaluate clouds, datacenters and various applications behaviours, simulation-based approaches also offer significant benefits. For example, they allow cloud researchers to: (i) determine the performance of their provisioning and service delivery policies in a repeatable and controllable environment (largely free of cost); and (ii) tune the performance bottlenecks before real-world deployment on commercial clouds. In this paper, we discussed the design and development of the "PerficientCloudSim" simulator that can reasonably simulate large-scale heterogeneous datacenters in terms of performance variations due to hardware (CPU) heterogeneities and co-located VMs. Our experimental evaluations suggest that significant performance and, therefore, cost benefits could be achieved, in IaaS clouds, through scheduling the demand on available resources using appropriate resource allocation and consolidation with migration policies.

In this paper, the heterogeneity of resources was considered through relating the Google data to real benchmark workloads [12]. Resource contention due to co-located VMs and migration was statistically modelled and implemented. We also used the performance parameters of hosts in a real cloud (Amazon EC2) as demonstrated in [9]. Using real workload data from cloud service providers, we proposed "PerficientCloudSim" which accounts for resource heterogeneities in terms of CPU architecture, energy consumption, workload migration and resource contention due to co-location. Our evaluation suggests that "PerficientCloudSim" can reasonably simulate performance degradation (variations in runtimes), caused by CPU heterogeneity and resource contention, in cloud workloads. Moreover, energy consumption and performance of migrated workloads was considered. We believe, modelling such a heterogeneous datacenter was not possible in available cloud simulators, before. Furthermore, cloud researchers can use "PerficientCloudSim" to test and validate various resource allocation and management policies, before they are implemented over real heterogeneous infrastructure clouds. In addition, we suggest that other cloud simulators should also be adapted to represent resource and workload heterogeneities in real clouds. Based on our findings, we can take this study further to suggest generalised models for CPU architecture heterogeneity, resource contention and migration. The model would be able to demonstrate performance variations (runtimes) for various workloads based on the CPU model (architecture) and migration opportunities or decisions. Moreover, we are working to integrate various migration approaches (pre-copy) into the proposed simulator.

Further research is needed to determine what kinds of workload are not suitable for migration and can run more energy and performance efficiently without being migrated. Similarly, investigation of workload runtimes is required for aggregation based VM placement techniques which put similar workloads/VMs onto the same heterogeneous hosts. Such resource provisioning policies are currently used in many production clouds but not in Google's cluster [58]. Similar VM placement policies are demonstrated in [61], however, it is assumed that runtime of workloads is known in advance. There is a need for the investigation of similar runtime based VM placement and their impacts on infrastructure energy consumption and workload performance, particularly,

when co-located VMs compete for similar resources. Moreover, besides CPU heterogeneity, resource contention and consolidation through migrations, workloads (VMs) performance can also be substantially affected by the heterogeneities of other on-board hardware resources such as RAM, GPU card; and that needs further investigation and research. Similarly, investigation of the energy consumption and performance of the migrated VMs, using real workload dataset [49], is essential in order to study their impact with respect to various migration policies, co-location and workload types. These parameters can further be used to predict energy, performance and cost efficient migrations, migration durations and possibilities of performance losses or improvements; in order to automate energy, performance and cost efficient resource management across hyper-scale cloud datacenters [15]. Note that, for resource heterogeneity and co-located VMs contention we considered only the CPU and workloads aspects; whereas, other hardware resources can also affect VM performance as described in Sec. 5.4. Therefore, using MIPS ratings as a single measure for various performance metrics should not be the only scale of measurement under different situations. In the future, we will consider incorporating other hardware heterogeneity features into the PerficientCloudSim.

Acknowledgement

This work was financially supported by the Abdul Wali Khan University, Pakistan and the University of Surrey, UK. The work was produced as part of a PhD program under the supervision of Dr. Lee Gillam at the University of Surrey. Interested cloud researchers can contact the corresponding author for using "PerficientCloudSim". We are really thankful to Changyeon Jo (CSAP, Seoul National University, Korea) for providing an impressive real workload dataset that consists of VM migration statistics.

References

1. Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.
2. Muhammad Zakarya and Lee Gillam. Modelling resource heterogeneities in cloud simulations and quantifying their accuracy. *Simulation Modelling Practice and Theory*, 94:43–65, 2019.
3. Muhammad Zakarya and Lee Gillam. *Energy and performance aware resource management in heterogeneous cloud datacenters.* PhD thesis, University of Surrey, 2017.
4. Gabor Kecskemeti. Dissect-cf: a simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, 58:188–218, 2015.
5. Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283, 2012.
6. Michael Tighe, Gaston Keller, Michael Bauer, and Hanan Lutfiyya. Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*, pages 385–392. IEEE, 2012.
7. Rajkumar Buyya and Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience*, 14(13-15):1175–1220, 2002.
8. Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.
9. John O'Loughlin. *A workload-specific performance brokerage for infrastructure clouds.* PhD thesis, University of Surrey, 2018.
10. John O'Loughlin and Lee Gillam. Sibling virtual machine co-location confirmation and avoidance tactics for public infrastructure clouds. *The Journal of Supercomputing*, 72(3):961–984, 2016.
11. Fei Xu, Fangming Liu, and Hai Jin. Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud. *IEEE Transactions on Computers*, 65(8):2470–2483, 2016.

12. Muhammad Zakarya and Lee Gillam. Managing energy, performance and cost in large scale heterogeneous datacenters using migrations. *Future Generation Computer Systems*, 93:529–547, 2019.
13. Charles Reiss, John Wilkes, and Joseph L Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., Mountain View, CA, USA, Technical Report*, 2011.
14. Muhammad Zakarya and Lee Gillam. An energy aware cost recovery approach for virtual machine migration. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 175–190. Springer, 2016.
15. Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167. ACM, 2017.
16. Ohad Shai, Edi Shmueli, and Dror G Feitelson. Heuristics for resource matching in intel's compute farm. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 116–135. Springer, 2013.
17. Adrien Lebre, Jonathan Pastor, Anthony Simonet, and Mario Südholt. Putting the next 500 vm placement algorithms to the acid test: The infrastructure provider viewpoint. *IEEE Transactions on Parallel and Distributed Systems*, 30(1):204–217, 2019.
18. Mohamed Abu Sharkh, Ali Kanso, Abdallah Shami, and Peter Öhlén. Building a cloud on earth: A study of cloud computing data center simulators. *Computer Networks*, 108:78–96, 2016.
19. David Abreu Perez, Karima Velasquez, Marilia Curado, and Edmundo Monteiro. A comparative analysis of simulators for the cloud to fog continuum. *Simulation Modelling Practice and Theory*, page 102029, 2019.
20. Fairouz Fakhfakh, Hatem Hadj Kacem, and Ahmed Hadj Kacem. Simulation tools for cloud computing: A survey and comparative study. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 221–226. IEEE, 2017.
21. Ayaz Ali Khan, Muhammad Zakarya, and Rahim Khan. H^2–a hybrid heterogeneity aware resource orchestrator for cloud platforms. *IEEE Systems Journal*, 2019.
22. Sareh Fotuhi Piraghaj, Amir Vahid Dastjerdi, Rodrigo N Calheiros, and Rajkumar Buyya. Containercloudsim: An environment for modeling and simulation of containers in cloud data centers. *Software: Practice and Experience*, 47(4):505–521, 2017.
23. Anton Beloglazov and Rajkumar Buyya. Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds. *Concurrency and Computation: Practice and Experience*, 27(5):1310–1333, 2015.
24. Alberto Núñez, Jose L Vázquez-Poletti, Agustin C Caminero, Gabriel G Castañé, Jesus Carretero, and Ignacio M Llorente. icancloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10(1):185–209, 2012.
25. Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493, 2018.
26. Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.
27. Carlo Puliafito, Diogo M Gonçalves, Márcio M Lopes, Leonardo L Martins, Edmundo Madeira, Enzo Mingozzi, Omer Rana, and Luiz F Bittencourt. Mobfogsim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory*, 101:102062, 2020.
28. Simon Ostermann, Kassian Plankensteiner, Radu Prodan, and Thomas Fahringer. Groudsim: an event-based simulation framework for computational grids and clouds. In *European Conference on Parallel Processing*, pages 305–313. Springer, 2010.
29. Seung-Hwan Lim, Bikash Sharma, Gunwoo Nam, Eun Kyoung Kim, and Chita R Das. Mdcsim: A multi-tier data center simulation, platform. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–9. IEEE, 2009.
30. Vincenzo De Maio, Gabor Kecskemeti, and Radu Prodan. A workload-aware energy model for virtual machine migration. In *2015 IEEE International Conference on Cluster Computing*, pages 274–283. IEEE, 2015.
31. Adrien Lebre, Jonathan Pastor, and Mario Südholt. Vmplaces: A generic tool to investigate and compare vm placement algorithms. In *European Conference on Parallel Processing*, pages 317–329. Springer, 2015.
32. John Cartlidge and Dave Cliff. Comparison of cloud middleware protocols and subscription network topologies using crest, the cloud research simulation toolkit-the three truths of cloud computing are: Hardware fails, software has bugs, and people make mistakes. In *CLOSER*, pages 58–68, 2013.

33. Frédéric Desprez and Jonathan Rouzaud-Cornabas. Simgrid cloud broker: Simulating the amazon aws cloud. 2013.
34. Damián Fernández-Cerero, Alejandro Fernández-Montes, Agnieszka Jakobik, Joanna Kołodziej, and Miguel Toro. Score: Simulator for cloud optimization of resources and energy consumption. *Simulation Modelling Practice and Theory*, 82:160–173, 2018.
35. Damian Fernández-Cerero, Agnieszka Jakóbik, Alejandro Fernández-Montes, and Joanna Kołodziej. Game-score: Game-based energy-aware cloud scheduler and simulator for computational clouds. *Simulation Modelling Practice and Theory*, 93:3–20, 2019.
36. Ahmad Siavashi and Mahmoud Momtazpour. Gpucloudsim: an extension of cloudsim for modeling and simulation of gpus in cloud data centers. *The Journal of Supercomputing*, 75(5):2535–2561, 2019.
37. Ubaid Ur Rahman, Kashif Bilal, Aiman Erbad, Osman Khalid, and Samee U Khan. Nutshell—simulation toolkit for modeling data center networks and cloud computing. *IEEE Access*, 7:19922–19942, 2019.
38. Gayashan Amarasinghe, Marcos D de Assunção, Aaron Harwood, and Shanika Karunasekera. Ecsnet++: A simulator for distributed stream processing on edge and cloud environments. *Future Generation Computer Systems*, 111:401–418, 2020.
39. Rodrigo Izidoro Tinini, Matias Romário Pinheiro dos Santos, Gustavo Bittencourt Figueiredo, and Daniel Macêdo Batista. 5gpy: A simpy-based simulator for performance evaluations in 5g hybrid cloud-fog ran architectures. *Simulation Modelling Practice and Theory*, 101:102030, 2020.
40. Krishnadhan Das. *Cloud computing simulation*. PhD thesis, Indian Institute of Technology, Bombay Mumbai, 2011.
41. Sheng Di and Franck Cappello. Gloudsim: Google trace based cloud simulator with virtual machines. *Software: Practice and Experience*, 45(11):1571–1590, 2015.
42. KyoungSoo Park and Vivek S Pai. Comon: a mostly-scalable monitoring system for planetlab. *ACM SIGOPS Operating Systems Review*, 40(1):65–74, 2006.
43. Muhammad Zakarya. An extended energy-aware cost recovery approach for virtual machine migration. *IEEE Systems Journal*, 13(2):1466–1477, 2018.
44. Qiang Huang, Fengqian Gao, Rui Wang, and Zhengwei Qi. Power consumption of virtual machine live migration in clouds. In *Proceedings - 2011 3rd International Conference on Communications and Mobile Computing, CMC 2011*, pages 122–125, 2011.
45. Ibrahim Alzamil, Karim Djemame, Django Armstrong, and Richard Kavanagh. Energy-aware profiling for cloud computing environments. *Electronic Notes in Theoretical Computer Science*, 318:91–108, 2015.
46. Ibrahim Alzamil and Karim Djemame. Energy prediction for cloud workload patterns. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 160–174. Springer, 2016.
47. Ayaz Ali Khan, Muhammad Zakarya, Rahim Khan, Izaz ur Rahman, Mukhtaj Khan, et al. An energy, performance efficient resource consolidation scheme for heterogeneous cloud datacenters. *Journal of Network and Computer Applications*, page 102497, 2019.
48. John O'Loughlin and Lee Gillam. Performance evaluation for cost-efficient public infrastructure cloud use. In *International Conference on Grid Economics and Business Models*, pages 133–145. Springer, 2014.
49. Changyeon Jo, Youngsu Cho, and Bernhard Egger. A machine learning approach to live migration modeling. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 351–364. ACM, 2017.
50. Yangyang Wu and Ming Zhao. Performance modeling of virtual machine live migration. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 492–499. IEEE, 2011.
51. Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. *Cluster computing*, 16(2):249–264, 2013.
52. Mar Callau-Zori, Lavinia Samoila, Anne-Cécile Orgerie, and Guillaume Pierre. An experiment-driven energy consumption model for virtual machine management systems. *Sustainable Computing: Informatics and Systems*, 18:163–174, 2018.
53. Tiago C. Ferreto, Marco A S Netto, Rodrigo N. Calheiros, and César A F De Rose. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, 27(8):1027–1034, 2011.
54. Muhammad Zakarya. Energy, performance and cost efficient datacenters: A survey. *Renewable and Sustainable Energy Reviews*, 94:363–385, 2018.
55. Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

56. Ayaz Ali Khan, Muhammad Zakarya, and Rahim Khan. Energy-aware dynamic resource management in elastic cloud datacenters. *Simulation Modelling Practice and Theory*, 92:82–99, 2019.
57. Nguyen Quang-Hung, Nam Thoai, and Nguyen Thanh Son. Epobf: energy efficient allocation of virtual machines in high performance computing cloud. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XVI*, pages 71–86. Springer, 2014.
58. Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at Google with Borg. *Proceedings of the Tenth European Conference on Computer Systems - EuroSys '15*, pages 1–17, 2015.
59. Ayaz Ali Khan, Muhammad Zakarya, Rajkumar Buyya, Rahim Khan, Mukhtaj Khan, and Omer Rana. An energy and performance aware consolidation technique for containerized datacenters. *IEEE Transactions on Cloud Computing*, 2019.
60. Muhammad Zakarya and Lee Gillam. Energy efficient computing, clusters, grids and clouds: a taxonomy and survey. *Sustainable Computing: Informatics and Systems*, 14:13–33, 2017.
61. Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Release-time aware vm placement. In *Globecom Workshops (GC Wkshps), 2014*, pages 122–126. IEEE, 2014.
62. Tom Guérout, Thierry Monteil, Georges Da Costa, Rodrigo Neves Calheiros, Rajkumar Buyya, and Mihai Alexandru. Energy-aware simulation with dvfs. *Simulation Modelling Practice and Theory*, 39:76–91, 2013.
63. Saurabh Kumar Garg and Rajkumar Buyya. Networkcloudsim: Modelling parallel applications in cloud simulations. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 105–113. IEEE, 2011.
64. Rodrigo N Calheiros, Marco AS Netto, César AF De Rose, and Rajkumar Buyya. Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. *Software: Practice and Experience*, 43(5):595–612, 2013.
65. Sören Frey and Wilhelm Hasselbring. The cloudmig approach: Model-based migration of software systems to cloud-optimized applications. *International Journal on Advances in Software*, 4(3 and 4):342–353, 2011.
66. Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, Albert Zomaya, et al. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in computers*, 82(2):47–111, 2011.
67. Vincenzo De Maio, Gabor Kecskemeti, and Radu Prodan. An improved model for live migration in data centre simulators. In *Proceedings of the 9th International Conference on Utility and Cloud Computing*, pages 108–117. ACM, 2016.

[Appendix A: CloudSim]

CloudSim [1] is a tool-kit to model and simulate IaaS clouds. CloudSim is an extension of the GridSim [7] simulator that uses SimJava[23] library as a framework for event handling and passing messages between entities. Due to several limitations of SimJava, CloudSim is designed using a new discrete event management framework as demonstrated in [1]. CloudSim is widely used to simulate resource provisioning techniques. It has the ability to perform simulations of IaaS clouds where heterogeneous workload is assigned and executed with different experimental conditions. It enables users to express datacenter characteristics, including the number and specification of hosts, storage, network topology and design of datacenter usage. It enables the design of VM placement policies, allocation of host cores (PEs) to VMs and division of CPU time between users' workloads. Switching on/off hosts, VM consolidation with migration, and integration of energy models (SPECpower[24] benchmarks) are the notable techniques to model energy and performance efficient datacenters in CloudSim [8].

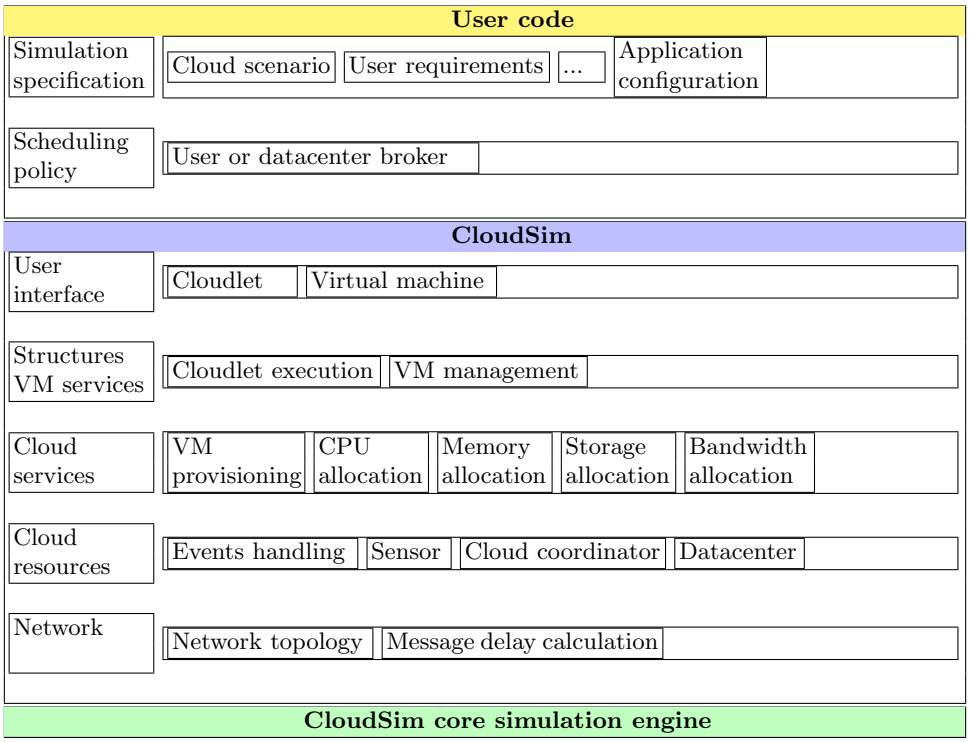

Fig. 12: CloudSim basic architecture [1]

Within CloudSim, the application layer is managed by a broker which requests VMs creation. A broker can concurrently own one or more VMs, that are kept running on hosts until their explicit de-allocation by the broker i.e tasks are finished or some local requests are waiting (high priority) according to the SLA lease. VMs and host capacities are defined in Million of Instructions Per Second (MIPS). For example, a CPU of 2.0GHz is defined 2,000 million instructions per second, which means that it can execute 2,000 instructions per second. Tasks (cloudlets) are allocated to VMs

[23] http://www.dcs.ed.ac.uk/home/simjava/tutorial/

[24] https://www.spec.org/power_ssj2008/

and defined in terms of the number of CPU instructions necessary for their completion (MIPS). Note that, if MIPS is used as a single measure, it means that workloads are basically homogeneous. Therefore, we assume larger or smaller MIPS as proxy for various workloads, represented by their CPU architecture-specific runtimes, since there is no other way to simulate the distinction between various workloads (e.g. a disk-bound and a CPU bound task). This provides a motivation for the discussion of CPU architectures and workloads at this point, which seems relevant. From CPU heterogeneity point of view, smaller or larger MIPS would certainly translate to variations in runtimes even for similar workloads. From application heterogeneity point of view, applications' (workloads) runtimes (therefore, smaller or larger MIPS) are mapped statistically to runtimes of several benchmark workloads, such as Bzip2, Povray [9]. Closer similarities in runtimes (i.e. smaller or larger MIPS) are assumed as different types of applications. Fig. 12 shows the multi-layered design of the CloudSim simulator framework and its various architectural components [1].

The basic requirements to run CloudSim include: (i) Sun's Java version 8 or newer; and (ii) Apache ant[25] or Maven[26] to compile CloudSim. Both Ant and Maven "simplifies Java project management by providing various tools and plugins for project building, testing, and packaging, dependency management, etc." CloudSim is not compatible with older versions of Java and may not be compatible with non-Sun Java version, such as GCJ or J++. A numerical library such as Flanagan's[27] or Apache Math[28] is needed to run several built in examples inside the CloudSim. CloudSim can be installed on any desktop machine that supports Java 8 or newer version. However, machines with large memories are preferable, particularly, when dealing with large-scale datacenter simulations. Each CloudSim entity is an object, created at the start of the simulation, and this may cause heap memory issues when the Java garbage collector is not able to find un-referenced objects for deletion.

CloudSim has been widely used to: (i) measure the effects of energy-aware VM allocation and migration algorithms on datacenter OpEx (operational expenditures); and (ii) evaluate scheduling mechanisms to allocate tasks to VMs [62]. Because CloudSim is an event driven simulation toolkit, its components maintain a message queue and generate messages, which they pass to other entities. A CloudSim simulation can instantiate several datacenters, each of which comprises several hosts (heterogeneous), which in turn host/accommodate multiple VMs executing one or more tasks. A datacenter is then characterized by its policy of placing requested VMs onto hosts (with the default strategy being to select the host with the least CPU cores in use) [8]. Moreover, each datacenter can be configured to charge users with different prices for storage, VM usage, and data transfer (i.e. pricing models).

Each host has its own policy, which defines how its compute resources are to be divided among accommodated VMs, i.e. whether VMs operate on shared (time shared) or distinctly separated (space shared) resources and whether over subscription of resources is allowed or not. In a similar fashion, each VM also comes with a scheduling policy which specifies how its compute resources are to be divided among tasks. On top of this architecture, an application-specific datacenter broker supervises the whole simulation. The broker is responsible to: (a) make requests for allocation and deallocation of VMs inside the datacenter, and (b) assign tasks to VMs for execution. Fig. 13 shows the UML class diagram of CloudSim and its various components.

As a completely customizable tool, CloudSim allows extension of policies in all its components, which makes it a suitable research tool that can handle the complexities arising from simulated platforms [1]. Several extensions to CloudSim have been presented in the literature. For example: (i) NetworkCloudSim [63], which introduces sophisticated network modelling and inter-task communication; (ii) EMUSIM [64], which uses emulation to identify the performance requirements and runtime workload characteristics and feeds this information to CloudSim for more accurate simulation; and (iii) CloudMIG [65], which "facilitates the migration of software systems to the cloud by contrasting different cloud deployment options based on the simulation of a code model in CloudSim". The internal processing of CloudSim in the form of a sequence diagram is shown

[25] http://ant.apache.org/

[26] https://maven.apache.org/

[27] https://www.ee.ucl.ac.uk/ mflanaga/java/

[28] http://commons.apache.org/proper/commons-math/

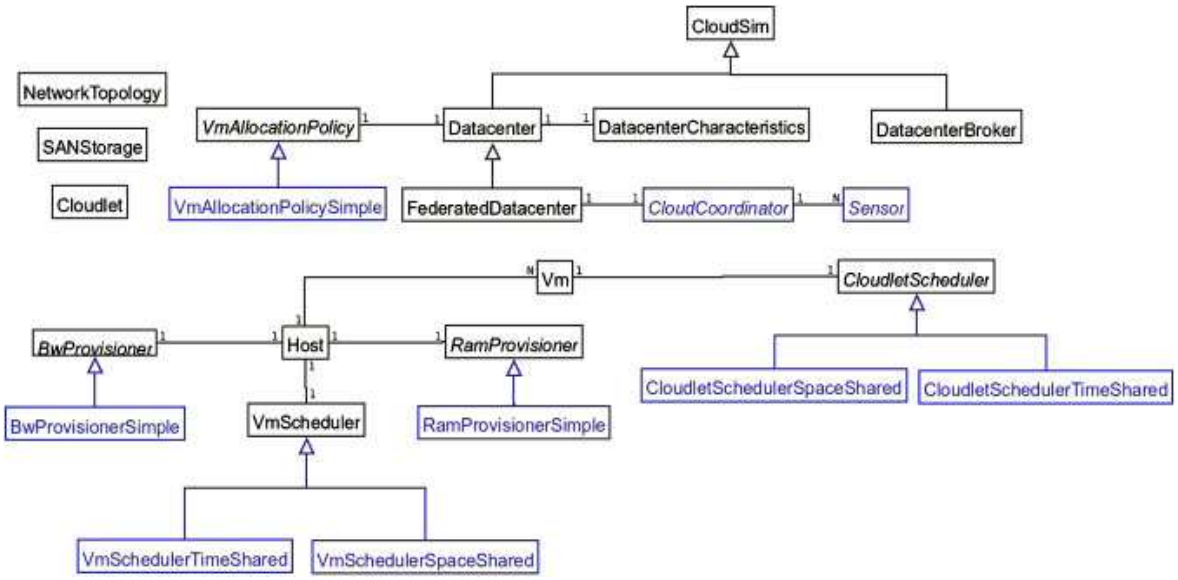

Fig. 13: CloudSim class design diagram [1] [an explanation of these classes is given in Sec. 8.1]

in Fig. 14. The two methods *updateVMProcessing()* and *updateCloudletsProcessing()* are the core processes of the simulation internal processing. At datacenter level, at each simulation step, the former method updates VMs processing inside the hosts. Similarly, at host level, the former method invokes the latter one to update tasks execution which are currently running inside VMs. A comprehensive discussion of the simulation internal processing is available in [1].

Despite its popularity and number of citations, CloudSim is not validated and verified yet, although there are several models inside CloudSim which have been validated in the real world. For example, the linear power model and migration performance model are validated as accurate (∼10% degradation) [66]. However, the linear power model assumes that energy consumption exclusively depends on CPU utilisation by ignoring other components such as memory and network. Moreover, CloudSim does not take into account several important parameters in its VM migration model such as over commitment and memory dirtying rate [67]. Similarly, there is no model to capture the overhead involved in the virtualisation and migration technologies[29]. Das et al. [40] have extended CloudSim with these models and have validated it through comparing with real world experiments with an approximate error of ±2%.

CloudSim simulator is designed with using two different kinds of classes: (i) the main classes which interact with each other to simulate a cloud [Sec. 8.1]; and (ii) the core classes which make simulations possible [Sec. 8.2]. We briefly describe these classes, here as we believe it would help readers in understanding CloudSim and our extensions; a detailed discussion of these classes can be found in [1].

8.1 Main Classes

The class diagram of the CloudSim main classes is shown in Fig. 13 [1].

BwProvisioner: (abstract class) models the provisioning policy of network bandwidth to VMs.

CloudCoordinator: periodically monitors the internal state of datacenter (resources) and undertakes dynamic load-shredding decisions.

Cloudlet: models the cloud-based application services (each Cloudlet refers to a user's job/task).

[29] http://www.cloudbus.org/cloudsim/

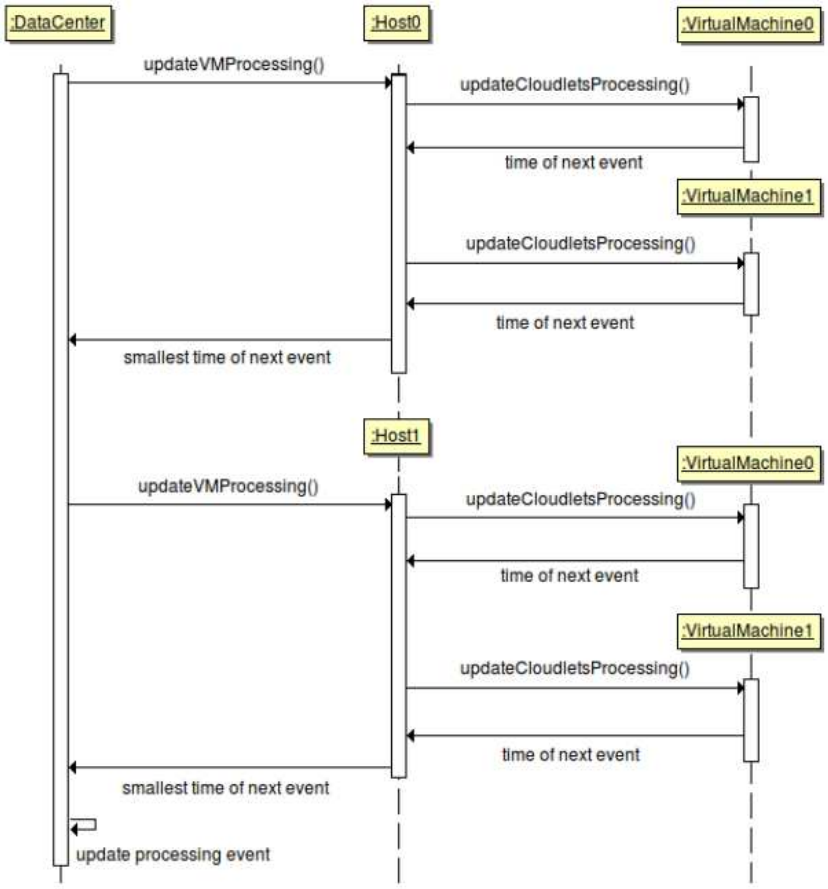

Fig. 14: CloudSim sequence diagram [40] [the CloudSim core classes are described in Sec. 8.2]

CloudletScheduler: (abstract class) can be extended by the implementation of different policies to determine the share of processing power (CPU) among Cloudlets that are running inside a VM. Two types of provisioning policies are offered in CloudSim: (i) space-shared; and (ii) time-shared [1].

CloudletSchedulerTimeShared: represents a provisioning policy which allows resources of a single VM to be shared among various cloudlets or tasks - allocate VMs resources to cloudlets or tasks.

CloudletSchedulerSpaceShared: represents a provisioning policy which does not allow resources of a single VM to be shared among various cloudlets or tasks - allocate whole VM to a single and exactly one cloudlet or task.

Datacenter: models the core infrastructure-level services (hardware) that are offered by cloud providers (such as Google).

DatacenterBroker: discovers suitable cloud service providers by querying the CIS and undertakes on-line negotiations for allocation of resources that can meet the application requirements.

DatacenterCharacteristics: contains configuration information of datacenter resources.

Host: models a physical resource – host/server and its characteristics.

NetworkTopology: contains the information for inducing network behaviour (latencies) in the simulation.

RamProvisioner: represents the provisioning policy for memory (RAM) allocation to VMs.

SanStorage: models a Storage Area Network (SAN) that is commonly ambient in datacenters for storing data (such as Amazon S3).

Vm: models a VM and its characteristics, which is managed and hosted by Host class.

VmAllocationPolicy: represents a provisioning policy to place VMs on hosts.

VmScheduler: models the policies (space-shared, time-shared) to allocate processor cores (PEs) to VMs.

VmSchedulerTimeShared: represents a provisioning policy which allows resources of a single host to be shared among various VMs - allocate processor cores (PEs) to VMs.

VmSchedulerSpaceShared: represents a provisioning policy which does not allow resources of a single host to be shared among various VMs - allocate whole processor core (PE) to a single and exactly one VM.

8.2 Core Classes

The class diagram of the CloudSim core classes is shown in Fig. 15 [1].

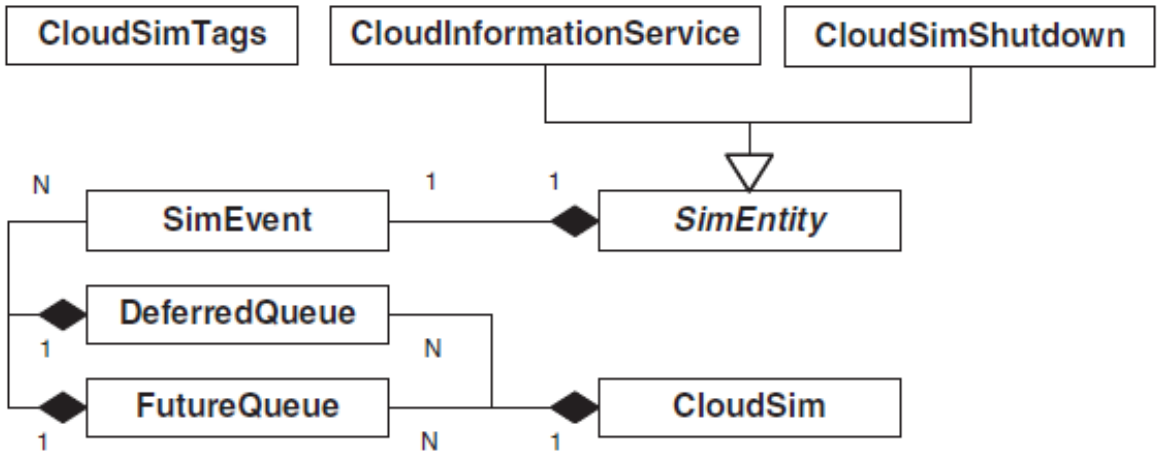

Fig. 15: Class diagram of CloudSim core classes [1]

CloudSim: responsible to manage event queues and control execution of the simulation events. Every event generated by the CloudSim entity at runtime is stored in a queue (future events). The events are sorted based on their creation time. Each event that is scheduled at the simulation step is removed from the queue (future events) and is added to another queue (deferred events).

CloudInformationService: provides resource registration, indexing, and discovering capabilities [1].

CloudSimShutdown: waits for the termination of all end-user and broker entities, and then signals the end of simulation to CIS.

CloudSimTags: consists of various static events/commands which indicate the action taken by CloudSim entities when they receive or send events.

DeferredQueue: implements the deferred event queue.

FutureQueue: implements the future event queue.

SimEntity: (abstract class) represents a simulation entity that is able to send messages to other entities. Furthermore, CIS is responsible to process received messages, fire and handle events. All entities must override the three core methods in this class: (i) *startEntity()* – entity initialization; (ii) *processEvent()* – processing of events; and (iii) *ShutdownEntity()* – entity destruction.

SimEvent: represents a simulation event that is passed between two or more entities.

The above classes when linked with each other can model a simple, but, complete datacenter. Besides these classes, other classes might exist in CloudSim simulator and its various variants, such as *FederatedDataCenter*, *containerDataCenter*, *VmScheduler*, and *CloudletScheduler*. All these classes are inherited either from the above core classes or main classes in some or other way. Therefore, we do not include them in this paper. A detailed discussion, functionalities/capabilities and explanation of these inherited classes can be found in [1], [22]. Further materials, documentations, on-line courses, publications and several variants of the CloudSim simulator are available at the portal of CLOUDS Laboratory[30].

[30] http://www.cloudbus.org/cloudsim/