Journal of Network and Computer Applications 173 (2021) 102869

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

HeporCloud: An energy and performance efficient resource orchestrator for hybrid heterogeneous cloud computing environments

Ayaz Ali Khan [a], Muhammad Zakarya [a,*], Izaz Ur Rahman [a], Rahim Khan [a], Rajkumar Buyya [b]

[a] Department of Computer Science, Abdul Wali Khan University, Mardan, Pakistan
[b] Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, University of Melbourne, Australia

ARTICLE INFO

Keywords:
Datacenters
Virtualisation
Containerisation
Resource management
Server consolidation
Workload migration
Energy efficiency
Performance

ABSTRACT

In major Information Technology (IT) companies such as Google, Rackspace and Amazon Web Services (AWS), virtualisation and containerisation technologies are usually used to execute customers' workloads and applications. The computational resources are provided through large-scale datacenters, which consume substantial amount of energy and have, therefore, ecological impacts. Since long, Google runs users' applications in containers, Rackspace offers bare-metal hardware, whereas AWS runs them either in VMs (EC2), containers (ECS) and/or containers inside VMs (Lambda); therefore, making resource management a tedious activity. The role of a resource management system is of the greatest importance, principally, if IT companies practice various kinds of sand-boxing technologies, for instance, bare-metal, VMs, containers, and/or nested containers in their datacenters (hybrid platforms). The absence of centralised, workload-aware resource managers and consolidation policies produces questions on datacenters energy efficiency, workloads performance, and users' costs. In this paper, we demonstrate, through several experiments, using the Google workload data for 12,583 hosts and approximately one million tasks that belong to four different kinds of workload, the likelihood of: (i) using workload-aware resource managers in hybrid clouds; (ii) achieving energy and cost savings, in heterogeneous hybrid datacenters such that the workload performance is not affected, negatively; and (iii) how various allocation policies, combined with different migration approaches, will impact on datacenter's energy and performance efficiencies. Using plausible assumptions for hybrid datacenters set-up, our empirical evaluation suggests that, for no migration, a single scheduler is at most 16.86% more energy efficient than distributed schedulers. Moreover, when migrations are considered, our resource manager can save up to 45.61% energy and can improve up to 17.9% workload performance.

1. Introduction

[1]Problems such as global warming, national and international energy supply, water complications, growing fuel costs, and computational business economics entirely bring the necessity for energy and performance, consequently, cost-efficient computation into sharp focus. Depletion in power plants that operate using coals, specifically, in the UK, offering an estimated safety margin for energy [i.e. capacity and demand ratio] of just 0.29% in 2017 (Shehabi et al., 2016), and the termination of several nuclear power plants in Germany and France, bring the actual risk of power outages and load-shedding in the very near future. Due to growth in renewables, a minor upsurge in energy safety margin of the UK can be realized in 2018 (i.e. an uptake from ~29.0% to ~36.0%). If we presume similar rates of consumption to the world of about 3.0% of total energy usage, then ~9.6% rise in datacenter energy efficiency will transform to approximately two times growth in the UK's energy safety margin (Shehabi et al., 2016), (Zakarya, 2018a). Similarly (Shehabi et al., 2016), also indicates that, until 2020, datacenters energy efficiency will remain unchanged, since industrial private workloads will migrate from internal private clouds to the public clouds. However, due to increase in mobile services and number of users, Internet of Things (IoT), and computing at scale, an increasing trend in energy consumption of the current datacenters can still be seen. Such an increase in energy consumption and the expected level of service

* Corresponding author.
E-mail addresses: ayazali@awkum.edu.pk (A.A. Khan), mohd.zakarya@awkum.edu.pk (M. Zakarya), izaz@awkum.edu.pk (I.U. Rahman), rahimkhan@awkum.edu.pk (R. Khan), rbuyya@unimelb.edu.au (R. Buyya).

[1] VM, EC2, ECS stand for virtual machine, elastic compute cloud and elastic container service, respectively.

https://doi.org/10.1016/j.jnca.2020.102869
Received 8 March 2020; Received in revised form 21 August 2020; Accepted 13 October 2020
Available online 28 October 2020

performance would certainly affect the environmental sustainability (3% Greenhouse gases), user's monetary costs and cloud economics [\euro 183.98billions in 2016 to \euro 217.05billions in 2017 - ~18% increase]. For example, AWS experienced approximately 1% reduction in their sales due to only 100 ms loss in performance (Zakarya, 2018a). Therefore, it is essential to look deeply into the problem and identify possible causes, opportunities and appropriate solutions for energy savings and performance improvements (as agreed in Service Level Agreement - SLA document) (Zakarya, 2018a), (Zakarya and Gillam, 2017).

The above issues advise the necessity to investigate for the sources and reasons of growing energy consumption in IaaS (Infrastructure as a Service) clouds and try to get rid of the reasons and/or manage them using conceivable solutions under workload performance constraints. The growing quantity and practice of ICT (Information & Communication Technology) equipment in IaaS cloud datacenters takes a consequential influence on the workload performance and IaaS energy consumption levels. Similarly, the falling practice of non-renewable energy sources, like coal, power plants, rises the necessity to design solutions to manage IaaS clouds resources in order to diminish the rising levels of energy usage, worldwide (Zakarya, 2018a). In respect of the former statement, datacenter's resources are usually under-utilised and idle; thus, making it possible to use methods like virtualisation and containerisation to save energy. In respect of the later statement, workloads might be moved, across resources powered by various energy production methods such as coal and renewables, when it is essential (as renewables are intermittent) or more beneficial (cost-efficient) to do so.

Virtualisation and containerisation enable same hardware for sharing among different users that: (i) increases resource utilisation; and (ii) creates opportunities for energy savings using resource consolidation. Besides these gains, virtualisation, containerisation and consolidation technologies could create performance-related problems due to migration and co-location (workloads compete for resources) leading to higher users' monetary costs, VM runtimes, and energy consumption. Moreover, public clouds may also achieve IaaS energy and performance efficiencies through appropriate resource management, allocation, and placement policies (Zakarya, 2018b). In hyper-scale cloud environments such as Intel, Google and AWS, containers have nearly replaced VMs as computational instance of choice. Compared to traditional VMs, containers have lower overheads of deployment and can, therefore, offer the best performance for certain workload types, as demonstrated in (Felter et al., 2015), (Kozhirbayev and Sinnott, 2017), (Kominos et al., 2017), (Mondesire et al., 2019), (Chae et al., 2019). Various applications have dissimilar business goals; few of them might run proficiently within VMs whereas few would perform best within containers or over bare-metal resources. Additionally, through running containers in VMs, supreme levels of resource utilisation are guaranteed through consolidation. Nevertheless, this may produce performance problems, in particular, when containers and VMs are being migrated collectively crosswise heterogeneous resources. Moreover, if workloads are running over various platforms in a datacenter, then there would be various migratable entities such as containers, VMs, hybrid (containers—VMs) and bare-metal workloads. Some of them would be more effective than the others and vice versa. For example, inter-platform migrations may occur in a particular platform; and intra-platform migrations may occur within platforms.

VMs and containers have allowed the quick adoption of the cloud computing environment, and the necessities, in terms of utility computing, moved to incorporate various kinds of sand-boxing technologies including virtualisation, containerisation, bare-metal and virtualised containerisation (Kominos et al., 2017). Generally, HPC (high performance computing) workers will favour provisioning the raw hardware (bare-metal) in order to deploy and run their workloads which decreases the hazards of performance degradation due to virtualisation. This is evidenced through the recent introduction of the bare-metal instances in the AWS cloud; which allows users to have full control over their provisioned resources. Moreover, certain workloads would perform better on containers than VMs and vice versa. For example, bank applications would run more securely in VMs than containers (isolation). In such circumstances, as shown in Fig. 1, variations in applications runtimes would create questions on datacenter energy consumption, workload performance and cloud economics i.e. users monetary costs and energy bills. For example, Fig. 1 demonstrates that the performance of Bzip2 workload over E5-2630 (CPU model) significantly varies across various platforms i.e. 300–500 min on bare-metal, 400–500 min on VMs, 300–500 min on containers, 550–700 min on containers over VMs. This suggests that the containerised applications' performance is comparable to the bare-metal infrastructure. The CPU models correspond to processor families with different clock speed, instruction set architecture (ISA), cache size, type, and performance variations during the fabrication process. Interested readers should refer (O'Loughlin, 2018), for further discussion of various CPU models and workload performance.

Big data and IaaS providers, e.g. Intel, Google, Microsoft, Rackspace, and AWS, examine and explore leading-edge solutions made over the VMs and/or containers technologies. The desires of these progresses partake a crucial influence on the IaaS management systems that are utilised to design dedicated services to handle with heterogeneities of resources and users' workloads. In consort with the difficulty of resource management system, up till then, such evolutions had been accomplished independently, deprived of demonstrating whether accurate abstractions will let the supervision of any type of sand-boxing technologies in a combined way (centralised) or in a distributed style. Furthermore, how various combinations of resource allocation and migration policies would affect IaaS energy consumption and workload performance. These sand-boxing technologies provide possibilities of affective resource scheduling, placement and consolidation. For example, workload could be scheduled or migrated to resources where its performance, high resource utilisation and energy efficiency are guaranteed. However, consolidation requires migrations that could be expensive in regard to energy consumption and performance loss (Zakarya, 2017). Moreover, similar workloads may perform quite differently on various platforms as described above. Similarly, certain cloud users may need full access to bare-metal resources in order to get total control of their provisioned hardware. This will, probably, soon force IaaS providers to rethink of using various platforms in their datacenters. Therefore, management complexities would further grow, when cloud providers will use a mixture of these technologies – in order to maximise their resource usage and reduce their operational costs. Thus, certain workloads may execute faster over the containers, or bare-metal hardware (non-virtualised) platforms; but, might perform the worst over VMs (technical white paper, 2016). The lowest execution times might mean the highest energy efficiency, and the least users' costs. Moreover, IaaS energy efficiency might also relate to these sand-boxing technologies, workload types and energy profiles of hardware (Zakarya, 2017).

This brings possibilities for hybrid datacenters which concurrently implement all sand-boxing technologies, e.g. the Intel's CIAO (Cloud Integrated Advances Orchestrator),[2] Magnum,[3] Kolla[4] and, subsequently, higher opportunities for efficient workload placement, consolidation and migration decisions across various technologies. This could be achieved through clustering the IaaS resources such that each cluster corresponds to a particular sand-boxing technology. Furthermore, each cluster may have either its own scheduler or share a centralised scheduler. Using individual schedulers for each sand-boxing technology such as containerisation, virtualisation, containers—VMs, bare-metal might not be suitable regarding energy and performance

[2] https://ciao-project.github.io/.
[3] https://wiki.openstack.org/wiki/Magnum.
[4] http://docs.openstack.org/developer/kolla/.

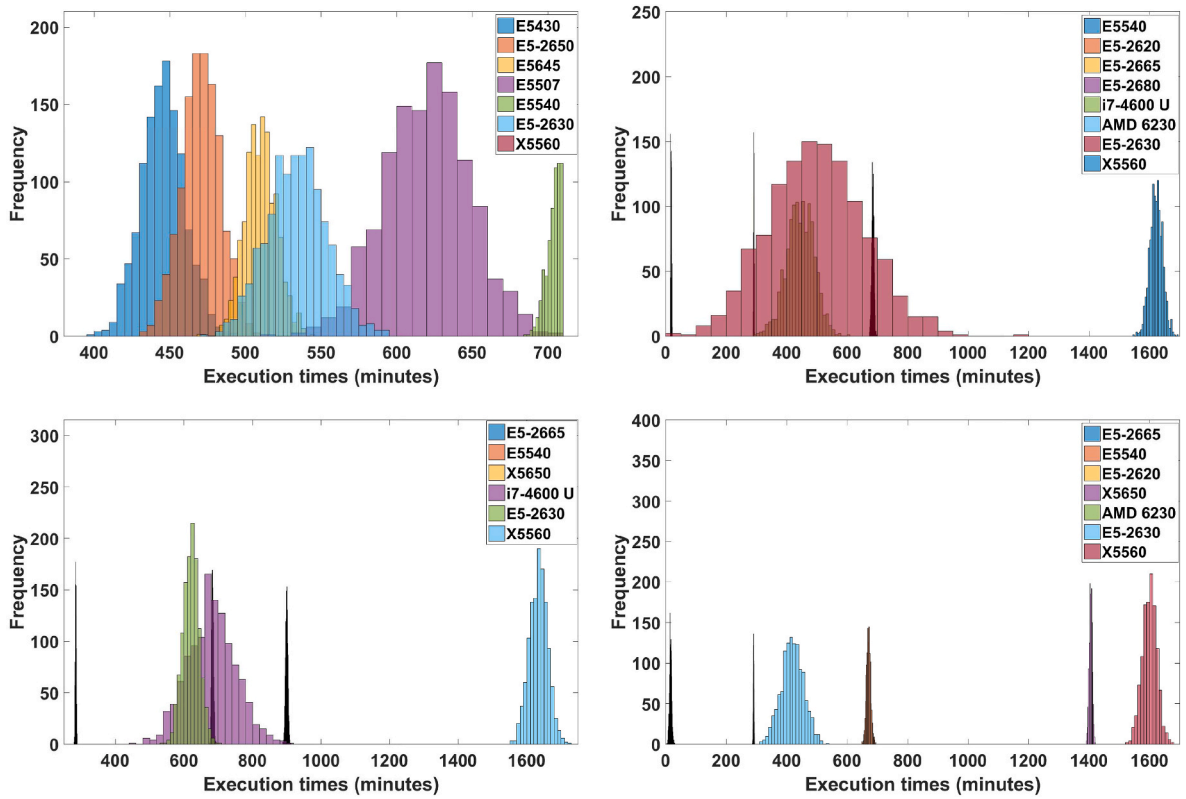


Fig. 1. Variations in applications performance when running over various sand-boxing technologies and CPU models [from left to right and top to bottom: VMs, containers, containers—VMs and bare-metal] – performance of Bzip2 workload over E5-2630 significantly varies across various platforms.

efficiencies; due to the absence of entire datacenter' state and resource usage details at each scheduling (platform) level. If these schedulers, can communicate and share entire datacenter state with each other (i.e. centralised scheduler); appropriate energy and performance efficient management decisions could be triggered (Zakarya, 2018a). Moreover, this will provide support for inter-platform and intra-platforms migrations; which are, to the best of our knowledge, unexplored in the existing literature of cloud computing. The former one occurs among the hosts of a particular platform e.g. VMs that could increase resource contention. The latter one would be more appropriate if certain workloads are misplaced during allocation. When both approaches are concurrently assumed, high levels of resource utilisation could be achieved.

This research aims to identify additional probable savings through efficient resource placement, allocation and consolidation with migrations (i.e. resource management) to reduce datacenters energy usage so that the workload performance is not affected undesirably due to resources and workloads heterogeneities. Furthermore, we investigate the impact of inter-platform and intra-platforms migrations on IaaS energy efficiency and workload performance, therefore, costs. The objective is to deal with these challenges through suggesting an architecture (reference) and a single, platform-independent, resource manager. The key contest would be, possibly, to decide the right set of abstractions for the development of a combined-style service which leverages the key approach/methodology i.e. deprived of implementing a specific, dedicated service, e.g. individual schedulers (distributed approach) and platform-specific monitoring, for each sandboxing technology. We, then, propose a centralised, workload-aware, scheduler and a consolidation technique which reduces the datacenter's energy consumption, and increases workload performance. In public clouds, reasonable best efforts would mean no loss in performance, as this will certainly affect the SLA's; and violation to SLA's would require a penalty to service providers. Whereas, in private clouds, increase in performance would be essential for certain workloads types such as HPC and database applications. We perform expansive simulations of the suggested framework using real workloads from large-scale IaaS providers such as Intel (Shai et al., 2013), Microsoft Azure (Cortez et al., 2017) and Google (Reiss et al., 2011) clusters that correspond to HPC (bare-metal), virtualisation and containerisation workloads, respectively.

The major contributions of our research are:

- a reference architecture and a single, platform-independent, resource manager is advised;
- an energy, performance and cost (EPC-aware) resource scheduler is presented that could effectively manage hybrid IaaS cloud infrastructures that run different kinds of sand-boxing technologies;
- an EPC-aware orchestrator is proposed that migrates various workloads energy, performance and, therefore, cost-efficiently;
- in order to concurrently simulate and evaluate hybrid clouds with various sand-boxing technologies, a cloud simulator is developed; and
- investigate the impact of datacenter resource configuration (physical order of hosts) on energy consumption and workload performance.

The rest of the paper is organised as follows. In Sec. 2, we discuss the resource allocation, placement and consolidation issue. In Sec. 3, we propose HeporCloud – a heterogeneity-aware hybrid approach that places and migrates workload appropriately. Sec. 4 describes various models to demonstrate energy and performance heterogeneities of various cloud platforms. We evaluate and validate HeporCloud through real workload datasets from Google, Intel and Azure clusters in Sec. 5 and demonstrate its efficiency in terms of energy, performance and, therefore, cost with respect to existing methods. In Sec. 6, we offer an overview of the related work. Finally, Sec. 7 summarises the paper along with several future research directions.

2. Problem description

In this section, we transform our multi-objective optimisation

problem into a single-objective problem. Multi-objective optimisation problems can be solved in two ways: (i) concurrently solve all objectives at once; and (ii) solve one objective first, and then make it a constraint on the next one. Moreover, various objectives can be combined into a single metric, and then solved as a single objective problem (Zakarya and Gillam, 2019). The three parties involved within the optimisation problem are: IaaS - service providers; SaaS - users or applications; and workloads. Furthermore, workloads can be assumed as SaaS. The aim of IaaS providers is to minimise energy consumption, SaaS wants to improve or, at least, maintain workload performance, and users wants to reduce their costs for the resources. The last two objectives are redundant across SaaS – as improved performance in terms of runtimes (reduced) will achieve the users objective (reduced costs). In the rest of this section, we mathematically formulate our objective optimisation problem.

2.1. Mathematical formulation

Assuming the above diverse requirements and circumstances (i.e. decreased runtimes, decreased costs, increased consumption of energy and reduced performance), our aim is to develop an allocation and consolidation model which: (i) predicts the energy consumption and levels of workload performance; (ii) correlates the predicted quantities (containers—VMs, performance and energy) to decide affective migrations; and (iii) lastly migrate best migratable entities to obtain optimal or approximate outcomes in terms of energy consumption and workload performance. The proposed technique is an effort to reduce infrastructure energy consumption (IaaS) without negatively affecting the workload performance (SaaS), even if migrated. We can describe the workload allocation or migration as a multi-objective optimisation (MOO) issue that consists of three nominal types of costs i.e. cost of energy consumption (Ec), users' monetary cost (Uc), and workload performance cost (Wpc). These costs are related to two different parties i.e. service providers (IaaS, SaaS) which are engaged in the whole progression; and regarding their properties and characteristics, each cost is exactly mapped to a particular goal/objective as given underneath:

I. IaaS → reduce the quantity of consumed energy during workload execution – Ec;
II. Workloads (SaaS) → improve or, at least, maintain the probable level of performance at the settled costs (in order to avoid penalties and meet SLAs - service level agreement) – in terms of execution time (R), where performance is expressed as the opposite of R and our aim is to reduce or, at least, maintain R – Wpc; and
III. Customers (SaaS) → are billed appropriately i.e. reduce cost or, at least, maintain the cost as per SLA agreement – Uc.

This could be understood spontaneously that Uc is proportional to R (i.e. each user is billed subject to his/her submitted workload runtime), and thus, if objective (II) is achieved then objective (III) is also achieved, intuitively. Therefore, objective (III) is not taken into account, as a separate objective, in this paper. As a result, we transform and express the multi-objective optimisation problem into an equivalent bi-objective optimisation problem. The two objectives of the transformed bi-objective optimisation problem are: reduce energy consumption (E); and reduce workload runtime (R). Mathematically, these objectives can be expressed as an objective function f, given by Eq. (1):

$$f = \begin{cases} \textbf{\textit{minimise}}(E) & where\ E = \sum_{i=1}^{platforms} P_i\ and\ P = \sum_{j=1}^{hosts} E_j \\ \textbf{\textit{minimise}}(R) & where\ \ R = \sum_{j=1}^{w} Runtime_j \end{cases} \tag{1}$$

subject to:

$$\begin{cases} H, V\ and\ \mathcal{S} & set\ of\ hosts,\ VMs|containers\ and\ resources \\ & e.g.\ CPU,\ memory,\ disk \\ (i)\sum_{h\in H} x_{hv} = 1 & where\ x_{hv} = \{0,1\} \Longrightarrow x_{hv} = 1 \\ & if\ VM\ v\ is\ mapped\ to\ host\ h \\ (ii)\sum_{v\in V} u_{vr}x_{hv} \le c_{hr}y_h & where\ u_{vr}\ denotes\ VM|container\ resources \\ & \forall\, v \in V\ and\ h \in H \\ & c_{hr}\ denotes\ amount\ of\ host'\ resources\ \forall\, r \in \mathcal{S} \\ & y_h = \{0,1\} \Longrightarrow y_h = 1 \\ & if\ host\ h\ is\ used\ otherwise\ y_h = 0 \end{cases} \tag{2}$$

where P denotes the total energy consumed by all hosts in a particular platform and E_j is the energy consumption of a specific host j that can be assumed as a linear function of the host' CPU utilisation level or number of running VMs (virtualised hosts). Moreover, the datacenter' total energy consumption is denoted by E. Furthermore, $Runtime_j$ denotes the execution time of a particular VM—container—task that belongs to workload w in a specific platform. The sum of all tasks' execution times in a workload is represented by R. Lower values for R mean higher performance and, subsequently, lower users' monetary costs Uc – VMs are billed according to execution times (Pay As You Go i.e. PAYG model). The constraints of container—VM placement problem are: (i) each container—VM is exactly allocated to a single VM—host at a time; (ii) the sum of all containers—VMs accommodated on a particular VM—host should not exceed the VMs—hosts individual capacities; and (iii) user's monetary cost remains as per SLA, as shown in Eq. (2) (Ferreto et al., 2011). Besides various constraints of the optimisation problem, Eq. (2) also illustrates various parameters and variables. For example, the variables x_{hv} corresponds to the mapping factor when a VM—container is allocated to a host. In consolidation scenarios, energy consumption could also be minimised through minimising the number of used hosts i.e. $\min(\sum_{h\in H} y_h)$.

In order to transform the above bi-objective optimisation problem (*min-min*) into a single objective minimisation problem, we can combine these objectives in different ways. For instance, Gupta et al. (Gupta, 2011) proposed the *ERP* metric (i.e. Energy Response time Product) which captures the trade-off that exists between energy, performance and, therefore, cost. Moreover, *ERP* is a widely used and appropriate evaluation metric to represent comparable trade-offs in the cloud community (Zakarya and Gillam, 2019), (Gandhi et al., 2010). Note that, reducing *ERP* can be assumed as maximising the "Performance-Per-Watt" ratio (*PPW* - performance achieved when one Watt-hour energy is consumed)[5] – where performance of the workload is expressed as reciprocal of the response time. In our formulation, workload performance is computed through runtime R which is assumed equivalent to the response time, grounded on the factor of *time*. Consequently, we reread the specified name of the *ERP* metric as the Product of Energy and Runtime (*ERP*). The *ERP* evaluation metric is given by Eq. (3):

$$ERP = E \times R \tag{3}$$

Hypothetically, the single objective of our bi-objective optimisation issue is to reduce, investigate and assess the behaviour of the *ERP* for numerous allocation and consolidation with migration policies, as specified by Eq. (4):

$$\min(ERP) \tag{4}$$

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

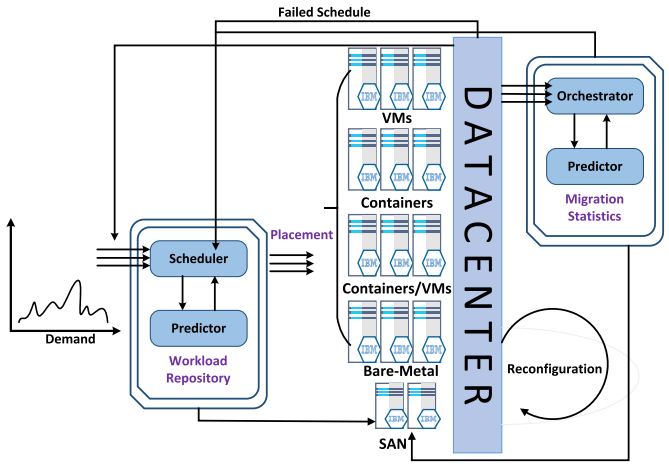


Fig. 2. The proposed HeporCloud architecture for hybrid clouds [SAN - Storage Area Network].

and orchestrator to manage heterogeneous hybrid datacenters resources, energy and performance, therefore, cost efficiently. The HeporCloud architecture is shown in Fig. 2. The scheduler and orchestrator both uses certain types of predictors to take effective scheduling decisions. Note that, the orchestrator looks/collects opportunities for consolidations, and, subsequently, which VMs and/or containers to migrate to where (hosts); and informs the scheduler to complete the operation. The proposed architecture consists of four IaaS resource types (VMs, containers, containers over VMs, bare-metal) and a storage module. The storage module is responsible to hold workload details and previous placement and migration actions which are used by the predictors to take appropriate decisions. Next sections, describes this in more detail.

3.1. The HeporCloud framework

We suggest a resource manager/architecture called "HeporCloud" which empowers the management of numerous, dissimilar, sand-boxing technological resolutions; and assess the performance of the projected

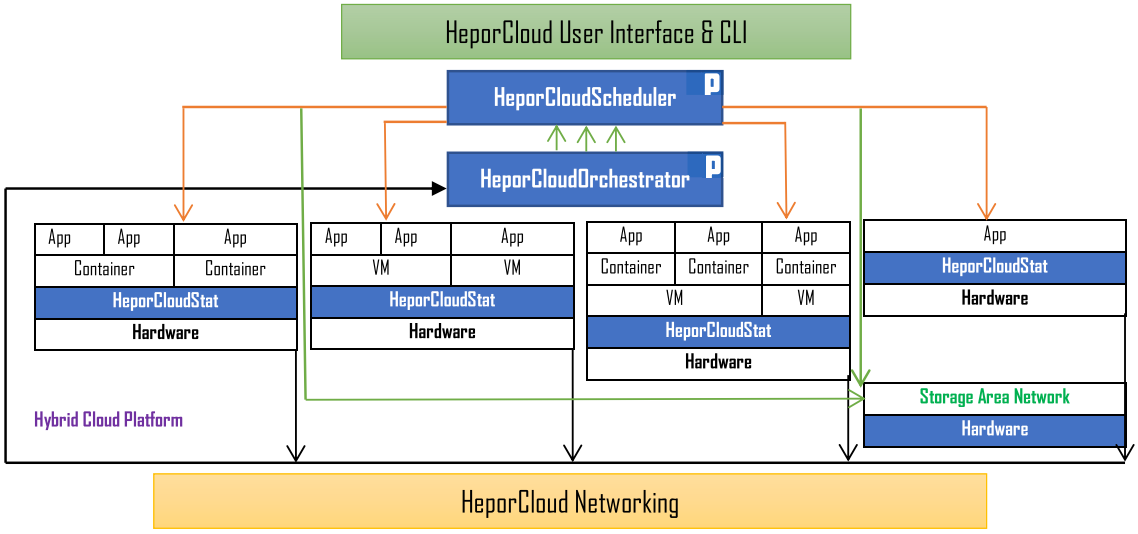


Fig. 3. The proposed HeporCloud framework [from an implementation point of view].

This transforms the above bi-objective optimisation problem into a single objective. The reason behind this simplification and using simple heuristic approach is to bias for dispatching speed and implementation simplicity over an absolute optimality. Albeit, multi-objective minimisation and meta-heuristic techniques can offer optimal results for off-line problems (Adhikari and Narayana Srirama, 2019), (Kaur et al., 2019); however, for on-line problems they are not preferable since the workload is not known and it will take long time to reach a placement decision (Zakarya, 2017), (Hu et al., 2019). From an experimental point of view, *ERP* of every host is estimated, for each placement and/or migration decision, using energy and runtime prediction techniques, as discussed in Sec. 3. The incoming workload is assigned and/or migrated to the host having the least *ERP*. Through realistic and plausible assumptions in a significantly modified version of an event driven cloud simulator "CloudSim" (Calheiros et al., 2011), we investigate how various resource placement, and consolidation with migration policies in a heterogeneous cloud, may affect the energy consumption of the IaaS cloud, performance of the SaaS workloads, and users' monetary cost when various types of heterogeneous workloads are taken into account.

3. HeporCloud - system architecture and resource management algorithms

In this section, we propose "HeporCloud" that uses a single scheduler resource manager through simulations along with plausible assumptions and real workload datasets. The proposed resource manager, as shown in Fig. 3, comprises three major modules: (i) a single scheduler [HeporCloudScheduler]; (ii) an orchestrator [HeporCloudOrchestrator]; and (iii) HeporCloudStat which is responsible to collect node level statistics such as resource utilisation levels, workload runtimes, placement and migration statistics and etc. The HeporCloudStat is an agent very similar to the cluster monitoring systems such as DataDog[6] and Ganglia.[7] The collected data is stored over a shared server, preferably, a Storage Area Network (SAN) that is accessible to the scheduler and orchestrator over a network. The HeporCloudScheduler and HeporCloudOrchestrator are installed on a separate host while HeporCloudStat is installed on every host of the datacenter.

The HeporCloudScheduler and HeporCloudOrchestrator are aware of the whole infrastructure (multiple platforms); and use predictors for workload-aware resource allocation and migration decisions. Since, the proposed HeporCloud framework uses a centralised approach rather than a distributed one (Khan et al., 2019a); therefore, when more and more VMs and/or containers interact with the HeporCloudScheduler and/or HeporCloudOrchestrator then due to either: (i) delay in

[6] https://www.datadoghq.com/.
[7] http://ganglia.sourceforge.net/.

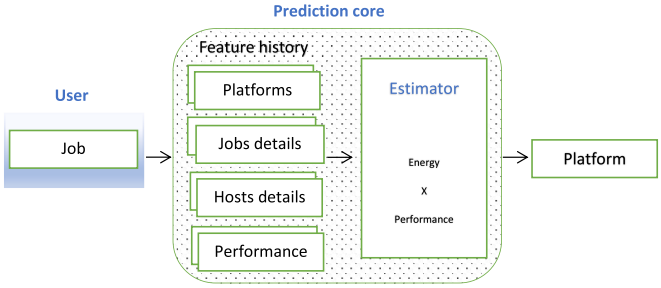


Fig. 4. Prediction of workload platform – in the first phase, similar jobs are being collected from the feature history; in the second phase, a best platform is being chosen (George et al., 2017).

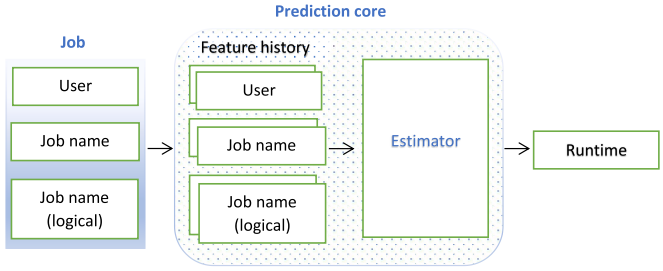


Fig. 5. Prediction of workload runtimes – in the first phase, similar jobs are being collected from the feature history; in the second phase, a particular statistical method is used to estimate runtimes (Tumanov et al., 2016).

communication; or (ii) some other reason (e.g. huge number of migrations, network congestion), the response may become slow. Subsequently, it will affect the system performance with respect to time, placement and migration decisions and will therefore, further, impacts on users monetary costs and energy consumption. This issue is more likely to arise with increase in number of VMs, containers or both. Fortunately, datacenters have their own dedicated networks and, we believe, this might be tolerable.

3.1.1. The HeporCloud scheduler

From an implementation point of view, the HeporCloudScheduler can be assumed as a centralised scheduler that interconnects various hardware technologies such as virtualised, containerised, virtual containerised (i.e. containers run inside VMs) and bare-metal, in the cloud platform. In order to optimise resource allocation and management, the scheduler maintains a history ($\mathcal{HY}$) of resource utilisation, various applications, their energy consumption and performance. Since, history or historical information is generated with the passage of time; which means that history may not be available when the model initially starts working. In such cases, the scheduler uses the well-known First Fit (FF) technique to place workload on available resources. Later on, they might be migrated to appropriate resources in next consolidation round. The pseudocode for HeporCloudScheduler is described in Alg. 1.

The scheduler predicts the future workload type and categorize it according to its energy consumption and performance. The prediction module, as shown in Fig. 4, looks for a platform where the application's performance is best at minimum energy cost. In the first phase, the runtime for the submitted job is predicted, as shown in Fig. 5. Due to high level of correlation between job runtimes, and three other job parameters i.e. its submitting user, job name and job logical name, our model predicts runtime using these parameters. Furthermore, along with runtimes, other job characteristics such as its submission time, job priority, and resource (CPU, memory, disk) actual usage and resource requirements can also be taken into account. However, it will increase the algorithmic complexity. In the second phase, the prediction module searches the features' history of previous executed jobs on various platforms (hosts) and their performance or runtimes. The submitted job is compared to other jobs, using various parameters such as submitting user, name, and closer, most similar, jobs along with platforms are noted. The idea of these two phases stems, basically, from (George et al., 2017), (Tumanov et al., 2016). In the third phase, the estimator chooses the best platform (host) that could run the workload with the minimum product of energy consumption and runtime (*ERP*). For example, if job X *ERP* is Y when run in a container and Z when run in a VM, and Y > Z, then the containerised platform is selected. If there are more than one platforms—hosts, then all platforms—hosts within a particular cluster—platform are sorted out in decreasing order of their *ERPs* (objective

Algorithm 1
The HeporCloudScheduler.

Input: workload w_i, history ($\mathcal{HY}$)
Output: schedule w_i
if $\mathcal{HY} \neq NULL$ **then**
  predict w_i from the history $\mathcal{HY}$;
  categorize w_i from the history $\mathcal{HY}$ with respect to energy and performance;
  if $w_i \in \{$ *HPC, VM, Container, Container*$|$*VM* $\}$ **then**
    $p \leftarrow$ select the best platform to run w_i;
    for *each host* $\in p$ **do**
      select the best host $h \in p$;
      if h *can accomodate* w_i **then**
        schedule w_i accordingly;
      end if
    end for
  end if
else
  schedule w_i using FF technique;
end if
return *schedule*

function). Thus, if the predicted workload falls within the category of HPC (bare-metal applications), VM, container—VM, or container, then the scheduler runs it, as appropriate, either on the most energy-performance efficient bare-metal, VM, virtualised container, or container, respectively. Unfortunately, if there is no suitable platform or host, then the job is packed in a VM and allocated to a host on the FF policy. Furthermore, if all the platforms are equally good for a certain workload, then the job is allocated with random (RND) allocation policy. Note that, the predictor is a history-based, and the HeporCloudStat (as explained later in this section) collects node level statistics and updates the previous history, at a separate host (probably attached to a Network Area Storage - NAS), at regular intervals of time e.g. 5 min. Once a VM terminates, its runtime, submitting user, hosts, resource usage, placement, and migration details are saved to an NAS. As shown in Fig. 2, the predictor associated with the HeporCloudScheduler is responsible to read workload details from a NAS server. In (Tsafrir et al., 2007), the authors have used a very simple predictor for runtimes which only averages the runtimes of the last two tasks submitted by a particular user. Their results ascertain that, counter-intuitively, using the most recent data is more significant than taking out longer history for comparable tasks. Thus, we believe, that maintaining a long history $\mathcal{HY}$ would have, only, a negative impact on the algorithm complexity.

3.1.2. The HeporCloud orchestrator

The HeporCloudOrchestrator executes sporadically or as soon as there are consolidation occasions, possibly, because of lesser demand for resources S. From implementational viewpoint, as soon as the resource (CPU, memory, disk) utilisation levels of some particular hosts either upsurges or declines from some pre-defined threshold values (U_{low} and U_{up}), then the HeporCloudOrchestrator is triggered to optimise the current state of the datacenter resources through consolidation with migration technique. We assume $U_{low} = 0.2$ and $U_{up} = 0.8$, as described later in Sec. 5.1, which mark hosts which are less than 20% utilised as under-utilised and those which are more than 80% utilised as over-utilised. The orchestrator forecasts whether an application ought to be migrated across various platforms with the intention to lessen energy consumption and, also, performance degradation. The steps involved in the optimisation are described in Alg. 2. Firstly, all migratable entities (workload, VM, container) are searched for [step 1]. The *optimise*(s_i) module searches across all hosts (H) for under-utilised ($H_{underutilised}$) and over-utilised ($H_{overutilised}$) hosts using the pre-defined threshold values (Zakarya and Gillam, 2016). Secondly, for every migratable entity, a suitable platform is estimated as its target platform using a predictor and the features' history [step 2–6]. This could be achieved using the HeporCloutScheduler, as shown in Fig. 4. Thirdly, the HeporCloudScheduler is directed to take appropriate decision i.e. migrate or do not migrate [step 7–14]. Moreover, various characteristics of the migratable entities on both source and target hosts, such as energy consumption, remaining runtime, and performance requirements, are considered during the migration decisions, as shown in Alg. 3.

Note that, Alg. 3 computes the total savings, in terms of energy and performance, which are achievable through migrating a particular VM—container to an appropriate host or platform. In the first phase, optimisation and migratable entities are estimated [step 1–2]. In the second phase, energy and performance of the target hosts $h' \notin \{H_{underutilised}, H_{overutilised}\}$ is computed [step 3–7]. The energy consumption of a particular migratable entity is computed using the model proposed in (Liu et al., 2011) and given by Eq. (5) - where energy is largely consumed during transferring the VM data from source to destination.

$$E_{mig} = 0.512 \times VM_{data} + 20.165 \tag{5}$$

where VM_{data} is the amount of data transferred (in MBs) during the migration process. This model is suggested more than 90% accurate. Furthermore, we use the transformed linear power model, as given by Eq. (12), to predict the container—VM energy consumption. Similarly, the performance of the migratable entity is estimated using the benchmark data (in terms of statistical distributions, means and standard deviations) and Eq. (8), as described later in Sec. 3.2 and Sec. 4.2. In third phase, expected remaining runtimes of VMs—containers are estimated using a particular prediction technique [step 8]. In fourth phase, the expected energy savings and performance gains are computed [step 9]. If migrations are affective i.e. savings are possible, then it is added to the migration list, otherwise next migration entity is considered for the above steps [step 10–17]. Finally, the migration list is sorted in decreasing order of VMs—containers savings (to prioritise entity for higher savings); and passed to the HeportCloudOrchestrator to take appropriate decisions [step 18–20].

Furthermore, if there are several migration opportunities, then we prioritise that migratable entity that could: (a) spend less energy on migration; and (b) save more energy or money (perform better) after the migration. The savings are calculated using Alg. 1. Note that, savings are possible only if migrations are performed to more energy and performance efficient target hosts than the source host. We use the host

Algorithm 2
The HeporCloudOrchestrator.

Input: current state of the datacenter s_i, migration map ($migMap$)
Output: migration decision md_i

```
1  migMap ← optimize(s_i);
2  i = 0;
3  while migMap.isNotEmpty = TRUE do
4  |  entity ← migMap.[i];
5  |  if entity ∈ { HPC, VM, Container, Container|VM } then
6  |  |  wc ← predict the most EPC efficient migratable enetity using Alg. 3 and Alg. 4;
7  |  |  scheduler.informToMigrate(wc);
8  |  |  migMap.[wc] ← null;
9  |  |  increment i;
10 |  else
11 |  |  entity already terminated;
12 |  end if
13 end while
14 return migMap
```

Table 1
Different workloads' runtimes over various CPU architectures and VM instances [696 MB input file to Bzip2 - Ubuntu 10.04 AMD desktop ISO file], container two types, virtualised containers (container—VMs), and bare-metal hardware, CoV is computed through dividing σ over the μ (Felter et al., 2015), (Kominos et al., 2017), (Zakarya, 2017), (O'Loughlin and Gillam, 2014), (Vaucher, 2015).

Sandboxing technology	Workload type	CPU model	m1.small					m1.medium				
			(μ)	(σ)	Min	Max	CoV	(μ)	(σ)	Min	Max	CoV
Virtualisation	Bzip2	**E5-2665**	241.3	1.18	237.97	245.2	0.005	–	–	–	–	–
	Pxz	**E5540**	709.6	7.8	680.4	733.6	0.011	393.6	3.4	381.9	403.9	0.009
		E5-2630	535	20	470.4	606.6	0.037	–	–	–	–	–
		X5560	1680	32.5	1625	1755	0.019	–	–	–	–	–
	–											
	Stream	**E5-2665**	59.2	1.88	52.16	65.0	0.032	–	–	–	–	–
	Povray	**E5540**	623.9	3.2	612.5	636.8	0.005	241.1	2.9	231.9	250.7	0.012
		E5-2630	128	2	120.5	134.2	0.016	–	–	–	–	–
		X5560	525.5	0.6	524.4	526.8	0.001	–	–	–	–	–
Containerisation	Pxz	**E5540**	685.2	3.9	670.68	698.17	0.006	387	7.6	362.55	410.53	0.02
		E5-2665	290.9	0.98	287.4	293.9	0.003	–	–	–	–	–
	Bzip2	**E5-2630**	495	159	42.1	1048.8	0.321	–	–	–	–	–
		X5560	1622	21.75	1580	1667	0.013	–	–	–	–	–
	–											
	Stream	**E5540**	211.7	2.5	204	219.9	0.012	131.2	1.4	126.9	136	0.011
		E5-2665	73.5	0.64	71.7	75.3	0.009	–	–	–	–	–
	Povray	**E5-2630**	118	32	30.9	221.9	0.27	–	–	–	–	–
		X5560	524.5	1.05	521.2	525.4	0.002	–	–	–	–	–
Containers—VMs	Pxz	**E5-2665**	284.2	1.45	279.8	288.7	0.005	–	–	–	–	–
		E5540	683.8	2.8	674.7	695.4	0.004	388.9	3.8	375.8	402.1	0.01
	Bzip2	**E5-2630**	621	23	535.1	687.9	0.037	–	–	–	–	–
		X5560	1634	26	1584	1688	0.016	–	–	–	–	–
	–											
	Stream	**E5-2665**	62.2	1.33	58.3	65.9	0.022	–	–	–	–	–
		E5540	211.3	2.1	205.3	219.9	0.01	131.4	2.1	124.7	138	0.016
	Povray	**E5-2630**	149	2	142.6	155.8	0.013	–	–	–	–	–
		X5560	527	0.5	526	528	0.000	–	–	–	–	–
Bare-metal	Pxz	**E5-2665**	290.8	1.13	287.6	294.4	0.004	–	–	–	–	–
		E5540	670.8	6.9	647.7	694.4	0.010	360.4	4.3	343.4	376.4	0.013
	Bzip2	**E5-2630**	418	37	307.8	518.2	0.088	–	–	–	–	–
		X5560	1600	23.75	1575	1670	0.015	–	–	–	–	–
	–											
	Stream	**E5-2665**	76.2	0.93	73.2	79.1	0.012	–	–	–	–	–
		E5540	192.0	0.8	189.6	194.3	0.004	109.5	1.0	106.4	112.8	0.009
	Povray	**E5-2630**	100	10	61.9	134.8	0.101	–	–	–	–	–
		X5560	521.6	0.625	520.4	522.9	0.001	–	–	–	–	–

efficiency factor E_f which is described as the product of host' energy consumption and performance (runtime) for a particular workload (Zakarya, 2017). For every migratable entity, we use E_f to compare its source and target in terms of energy consumption and expected level of performance [step 1–2]. Note that, $P_{C_{source}}$ and $\mu_{C_{source}}$ denote the energy consumption and expected level of performance of a particular workload on source host. If $E_f > 1$, this means that the target is more energy and/or performance efficient that the source; therefore, savings are possible. The savings are computed with our previous models i.e. CMCR - Consolidation with Migration Cost Recovery (Zakarya, 2018b), (Zakarya and Gillam, 2016) and Consolidation with migration Performance, Energy costs Recovery - CPER (Khan et al., 2019b) [step 3–9]. Furthermore, CMCR ensures to offset the migration cost with possible savings which runs in three steps: (i) compute the level of differences between source and target i.e. δx [step 4]; (ii) compute the time point where migration cost has been recouped back i.e. t_{off} [step 6]; and (iii) predict workload runtime in order to compute its remaining runtime (t_s), after which it has recovered its migration cost, for possible savings $\mathcal{P}_{savings}$ [step 7–9]. Note that, $\mathcal{P}_{savings}$ can be computed through multiplying the task remaining runtime with the difference between efficiencies of source and target hosts. If $E_f \leq 1$, then savings are not guaranteed, therefore, such migratable entities are removed from the migration list [step 10–17]. Finally, migratable entity along with saving is returned to Alg. 3 [using Alg. 4]. In order to calculate the remaining runtime of a migratable entity r_{time} on the target platform, its total runtime is predicted with the *predictRuntime*() routine, using the features' history, as shown in Fig. 5. There is already rich literature which has focused on application runtimes prediction (Tumanov et al., 2016), (Smith et al., 2004). An application's runtime can be predicted in two steps: (i) find similar applications that were executed in the past; and (ii) use a statistical approach to estimate its runtime. In respect of (i), application characteristics like submitting user, resource requirement, past runtime can be used to gather similar applications and their data from a database. The database is to be maintained either on each host or over a centralised SAN server. The accuracy of the prediction is strongly related to the accuracy of the similarity measures. In respect of (ii), techniques like mean, moving average, and regression analysis could be used to reach an estimation. Further details on runtime prediction can be found in (Tumanov et al., 2016), (Tsafrir et al., 2007), (Smith et al., 2004).

Algorithm 3
Feasible migration technique.

Input: hybrid platform, migration map ($migMap$), temporary migration map ($migTempMap$)
Output: the most effective migratable entity (e)

$platform.optimize()$ [call optimisation module for consolidation opportunities - Alg. 1];
$migTempMap \leftarrow getMigratableWorkloads()$ [again input from Alg. 1];
$migMap \leftarrow NULL$;
for *every host h in the hybrid platform* **do**
  for *every item i in migTempMap* **do**
    $rate \leftarrow h.energy.predict(i) \times h.performance.predict(i)$;
    [$rate \rightarrow$ minimise the product of energy and performance];
    $r_{time} \leftarrow i.predictRuntime() - (getCurrentTime() - i.getSubmitTime)$;
    $savings \leftarrow callAlgo(rate, r_{time})$ [call Alg. 4 to compute possible savings];
    if $savings \geq 0$ **then**
      $migMap.add(i, savings)$;
      remove i from $migTempMap$;
    else
      do nothing;
    end if
  end for
end for
$migMap.sortDesc(savings)$;
$e \leftarrow migMap.pop()$;
return e

Algorithm 4
Calculate power savings.

Input: The list migratable entities L, $Cost_{mig}$, t, t_{mig}, source and target hosts
Output: Return $\mathcal{P}_{savings}$, the amount of energy that could be saved

for *each migratable entity* $e \in L$ **do**
  $E_f \leftarrow (\frac{P_{C_{source}} \times \mu_{C_{source}}}{P_{C_{target}} \times \mu_{C_{target}}})$ [compare hosts with respect to energy and performancee efficiecies using the CMCR [4] or the CPER [30] approach];
  if $E_f > 1$ **then**
    $\triangle x \leftarrow P_{C_{source}} \times \mu_{C_{source}} - (\frac{P_{C_{source}} \times \mu_{C_{source}}}{E_f})$;
    if $\triangle x > 0$ **then**
      $t_{off} \leftarrow currentTime\ (t) + t_{mig} + \left\lceil \frac{t_{mig} \times Cost_{mig}}{\triangle x} \right\rceil$;
      $r_{total} \leftarrow predictRuntime(e)$;
      $t_s = r_{total} - t_{off}$;
      $\mathcal{P}_{savings} = t_s \times \triangle x$;
    else
      $Cost_{mig}$ is not recoverable;
      remove migratable entity e from L;
    end if
  else
    target is not more energy and performance efficient than the source;
    remove migratable entity e from L;
  end if
end for
return $\mathcal{P}_{savings}$

3.1.3. The HeporCloudStat

The HeporCloudStat module is running on every cluster node; and is responsible to collect node level statistic, periodically (e.g. regularly at 5 min intervals) or when needed (e.g. when migrations are performed). These statistics, including resource (CPU, memory, disk) consumption, utilisation levels, platform type, energy consumption, workload runtimes (performance), submitting users, and allocation along with migration details are stored on the same or on a separate host; preferably, a NAS server within the datacenter. Moreover, every agent of the HeporCloudStat on each node is connected to the master HeporCloudStat module that runs on the NAS server - which is responsible to collect and store node statistics on NAS. The HeporCloudScheduler and HeporCloudOrchestrator use these details for various purposes such as workload-aware resource allocation, platform/host selection and prediction of effective migrations (either VMs, or containers), respectively. Furthermore, certain APIs and built-in functionalities of the hypervisor

can be used to gather these node level details (e.g. *dstat*, *ps*). Moreover, third party cluster monitoring tools, either centralised or distributed that runs like an agent or daemon on each cluster node, such as DataDog and Ganglia, can also be used to gather node statistics (Massie et al., 2004).

We are aware that the HeporCloudStat is a burden on the cluster node that: (i) keeps on maintaining and/or calculating statistical details of hosts regarding VMs—containers in addition to performing its essential task of job execution; (ii) update its information with NAS server periodically or on-demand; and (iii) for hundreds or thousands of cluster nodes updating their information on NAS server will itself generate a lot of network traffic and, subsequently, burden on the datacenter network that may result in performance degradation in terms of longer latencies. In respect of (ii) and (iii), it might be tolerable due to datacenters internal dedicated networks and multi-cast channels (Massie et al., 2004). However, in respect of (i) this could degrade the cluster node performance and, most importantly, the available capacity for virtualised or containerised workloads. However, for a single node this overhead is very small, probably, less than 0.1% (CPU) and approximately 0.9 MB i.e. 0.09% (memory), as demonstrated in (Massie et al., 2004), for Ganglia monitoring tool on PlanetLab cluster. For other system, these values could be up to 0.025% (CPU) and 1.3 MB i.e. 0.25%. We further believe, a centralised HeporCloudStat, which runs on a dedicated powerful server, probably closer to the NAS server will resolve these issues to some extent. We prefer the distributed HeporCloudStat over the centralised one due to: no single point of failure; and quick communication with system other modules.

3.2. Resource predictions

Application's runtime prediction techniques have at least two major benefits: (i) efficient resource scheduling (placement) decisions can be made – e.g., VMs—containers with similar applications (or runtimes) can be placed on the same host (Dabbagh et al., 2014); (ii) if a host needs maintenance, the resource manager can estimate runtimes of all VMs—containers located on it – to determine when maintenance can be scheduled, and whether VMs—containers need to be migrated or not (Cortez et al., 2017); and (iii) cost effective migration decisions can be triggered – e.g. if certain co-located workloads perform worst, they can be migrated to other hosts (Khan et al., 2019a). In (Cortez et al., 2017), the authors have demonstrated that most runtimes of VMs in Azure cloud are relatively short i.e. more than 90% of runtimes are shorter. The curves show a knee around a day and then almost flatten out; suggesting that, if a particular VM runs for a day, it will very likely run much longer. Moreover, the relatively small percentage of long-running VMs actually account for more than 95% of the total resource usage. These findings are very similar to Google cloud, where applications run in containers (Zakarya, 2017), (Reiss et al., 2011).

As noted in Sec. 3.1, the HeporCloud framework consists of three prediction techniques: (i) predict the workload type; (ii) predict the workload runtime; and (iii) use workload runtimes and previous runs to predict an appropriate platform/host. In respect of (i) and (iii), appropriate resources and technology could be selected to run these workloads. In respect of (ii), appropriate migration decisions could be made. Cortez et al. (2017) suggests that same subscriptions have almost similar workloads, largely, with similar CPU utilisations. The author have used Fast Fourier Transform (FFT) to find periodicity in various workloads and categorized them as either potentially interactive or delay-insensitive (batch workloads). Moreover, the literature is significantly vast that describe various techniques, such as resource utilisation levels, priorities and submitting users, to profile and predict cloud workloads. The providers can use this knowledge in packing VMs and containers on hosts, as appropriate. For example, delay-insensitive workloads can be packed more tightly; while interactive workloads can be loosely packed onto hosts. Furthermore, the provider can avoid over-subscription of resources that run interactive workloads while allowing over-subscription for other workloads. Lastly, it is also possible to choose an appropriate and most affective sand-boxing technology (VM, container, container—VM, bare-metal) to run these workloads in an energy and performance efficient way.

To simplify the implementation, we use the workload priority as a representation of its type. This is in-line with previous assumptions as the task's priorities affect billing in Google cloud (Reiss et al., 2012). In our dataset, each job is submitted along with its priority. In order to predict the workload runtimes, we use the two most important features i.e. submitting user and workload type (priority). This is also in-line with previous work as demonstrated in (Cortez et al., 2017), (Tsafrir et al., 2007) – users, largely, submit similar jobs in Azure cloud. We are aware that there would be other efficient ways, such as machine learning techniques (gradient boost trees, performance monitoring), to estimate runtimes (Masdari and Khoshnevis, 2019); however, they might be impractical in large environments - requiring storage to log and maintain records and then searching the log for predictions. In (Tsafrir et al., 2007), the authors demonstrated that simple averaging the recent durations of jobs by the same user can result in good prediction outcomes. Moreover, their outcomes demonstrate that good predictability may not be always inline with good performance. Therefore, we believe complex predictors might be expensive in large-scale systems. Once the workload type and its estimated runtimes are known, we can use the *ERP* of every host to select the most energy, performance and cost-efficient host. The process also involves transforming the remaining runtime of a particular workload on a source host to equivalent remaining runtime on a target host. This can be achieved using the z-score normalisation (Zakarya and Gillam, 2019). The z-score (also known as standard score), as specified by Eq. (6), is usually used for computing the likelihood or probability of a particular score (x) that happens in the interior of a normally distributed dataset. Besides this, the concept of z-score also offers a technique to associate two or more than two scores that belong to different datasets having normal distributions.

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

Eq. (7) can be used to find runtime of the migrated workload (estimated) on the target host with given statistical means (μ, μ_1) and standard deviations (σ, σ_1) of source and target hosts (for normal distributions).

$$\frac{x - \mu}{\sigma} = \frac{x_1 - \mu_1}{\sigma_1} \tag{7}$$

where x and x_1 denote the probable execution times of the migrated workload on the source and target hosts, correspondingly. Moreover, the right and left sides of Eq. (7) narrate to the z-scores of the target and source hosts, correspondingly. This formulation permits us to analyse and estimate the likelihood of a score (i.e. the anticipated growth or reduction in execution time of the workload on the target host) happening inside a normally distributed dataset. Note that, the dataset consists of the performance (runtime) dissimilarities due to resource, workload and/or platform heterogeneities. The above Eq. (7) can be redrafted, to find the anticipated runtime (x_1) of the migrated workload on the target host given its remaining runtime (x) on the source host, as Eq. (8). For lognormal distributions, both x and x_1 are replaced with log (x), log(x_1) according to the definitions of normal and lognormal distributions (Zakarya and Gillam, 2019).

$$x_1 = exp\left(\sigma_1 \times \left\{\frac{log(x) - \mu}{\sigma}\right\} + \mu_1\right) \tag{8}$$

Note that, the remaining runtime of the workload running inside a VM—container can be computed through subtracting current time from the workload total runtime - predicted using the model, as shown in Fig. 5.

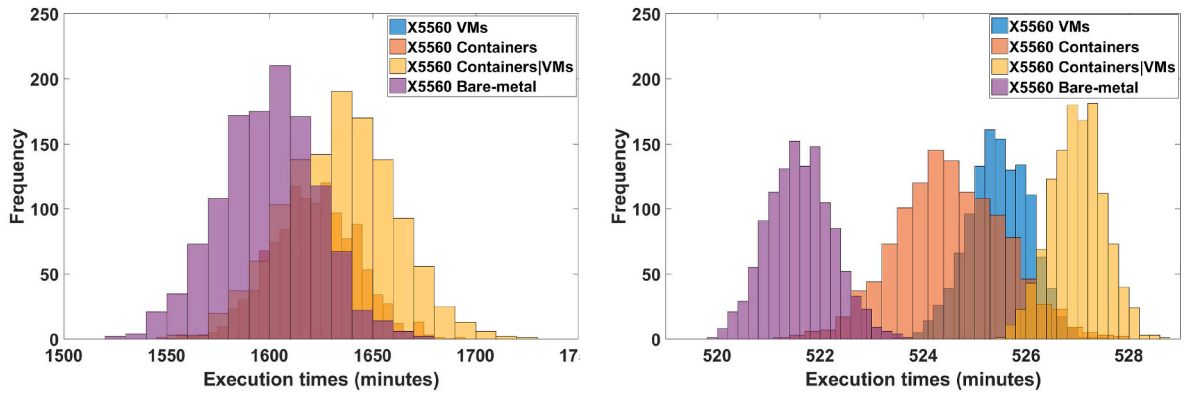


Fig. 6. Variation in runtimes of two different applications (left: Bzip2 – right: Povray), when run over different sand-boxing technologies, on X5560 CPU platform [Bzip2 in containers and bare-metal has comparable runtimes; Povray in containers performs better than VMs and virtualised containers].

4. Modelling energy consumption and platforms heterogeneities

In Sec. 4.1, we discuss how the energy consumption of a container—VM or physical host should be measured in simulations. In Sec. 4.2, we describe the performance of various benchmarks workloads when run over various platforms such as virtualisation, containerisation, virtualised containers and bare-metal hardware.

4.1. Modelling energy consumption

We use actual data for energy consumption of various servers that was collected by SPECpower[8] standards; so that, subsequently, the total energy consumed by datacenter is calculated [as described in Sec. 5.1]. Though, there is no defined method to calculate energy usage by a VM as well as container straight away; consequently, we use numerous mathematical models to obtain their estimated energy usage. Alternative method is division of server's total energy usage by total number of VMs/containers executing on that particular server. The aggregated energy usage (P) for a non-virtualised host is assessed by using linear power model, obtained by Eq. (9); where the extent of energy used is directly proportional to consumption level of the CPU (Zakarya, 2017).

$$P = P_{trivial} + (P_{maximum} - P_{trivial}) \times U \tag{9}$$

where CPU usage level is given by U, P_{maximum} and $P_{trivial}$ give energy usage once CPU usage is 100% and trivial i.e. 0%, respectively. Fan et al. (2007) experimentally evaluated the accuracy of this model as high as 95% over a few hundred servers. The authors also suggested a non-linear power model, given by Eq. (10); and demonstrated its accuracy as high as 99% (Dayarathna et al., 2015).

$$P = P_{trivial} + (P_{maximum} - P_{trivial}) \times (2U - U^{r}) \tag{10}$$

where r is the calibration parameter to minimise the square error and needs to be computed experimentally. Largely, it is used as equal to 1.4 in existing works (Dayarathna et al., 2015). Furthermore, considering containerised/virtualised servers, alone the VM—container energy usage is directly related with the number of VMs—containers executing over that specific server (Zakarya and Gillam, 2016), (Alzamil and Djemame, 2016). Thus, it is realistically sensible to forecast/estimate the VM—container energy usage in context with the linear CPU energy usage model (Zakarya, 2017) by having use of Eq. (11):

$$\mathcal{P}_{V\,M|container} = \left(\frac{P_{trivial}}{N}\right) + \mathcal{W}_{V\,M|container} \times (P_{maximum} - P_{trivial}) \times U_{V\,M|container} \tag{11}$$

where aggregated VMs—containers housed on a server is given by N, the extent of server's resources, like number of cores allotted to VM—container, is given by $\mathcal{W}_{V\,M|container}$, and usage level of VM—container is shown by $U_{VM|container}$ (Khan et al., 2020). In a real cluster setup, authors in (Alzamil and Djemame, 2016) demonstrated that this model has mean error as low as 1.75. Additionally, $P_{trivial}$ and P_{maximum} show the server's energy usage, calculated in Watt-hours, when the server is 0% (trivial) and 100% utilised, respectively (Zakarya and Gillam, 2016). The aforementioned model can be used to forecast energy consumption of a particular VM on a specific server at suitable usage levels (Alzamil and Djemame, 2016). Similarly, if a VM is executing M containers, then each container's energy consumption $\mathcal{P}_{container}$ can be computed through transforming Eq. (11) to Eq. (12):

$$\mathcal{P}_{container} = \left(\frac{\mathcal{P}_{vm_{trivial}}}{M}\right) + \mathcal{V}_{container} \times (\mathcal{P}_{vm_{maximum}} - \mathcal{P}_{vm_{trivial}}) \times U_{container} \tag{12}$$

where the usage level of container is given by $U_{container}$ and the fraction of resources for VM allotted to a container is given by $\mathcal{V}_{container}$. Furthermore, $\mathcal{P}_{vm_{trivial}}$ and $\mathcal{P}_{vm_{maximum}}$ denote the VM energy consumption when idle and fully utilised, respectively. We are aware of the fact that there may be several other precise models which may predict the energy usage of a server, VM, and container more accurately (Dayarathna et al., 2015), (Colmant et al., 2015), (Callau-Zori et al., 2018), (Lebre et al., 2019). However, the energy usage for a VM—container have been seen by these models as fraction of resources allocated from the physical server along with its usage levels. In the perspective of our efforts in this work, we use SPECpower standards for host's energy usage that settles for CPU, memory and/or disk. Thus, we believe that the datacenter total energy usage will not be affected using any other energy consumption model. As, in this paper, we focus on datacenter's energy efficiency, but, not a complete cloud environment, therefore, we assume that the above model is enough accurate and reasonable to measure the energy consumption of a datacenter. To have desperate cloud scenario, the energy usage within communication networks and/or other parts of the environment should be taken in consideration (Jiang and Chen, 2018). Further, energy used by containers—VMs during communication which are placed at various hosts should be considered when predicting total energy use. For further details, interested readers should read (Dayarathna et al., 2015), (Khan et al., 2020).

4.2. Modelling performance

Due to non-availability of a representative cloud workload (performance-specific) and CPU performance benchmarks; we model resource and application heterogeneities using statistical distributions. First, we collect data and simulate them using monte-carlo simulations. Using mapping techniques, we, then, relate this data to available real dataset

[8] https://www.spec.org/.

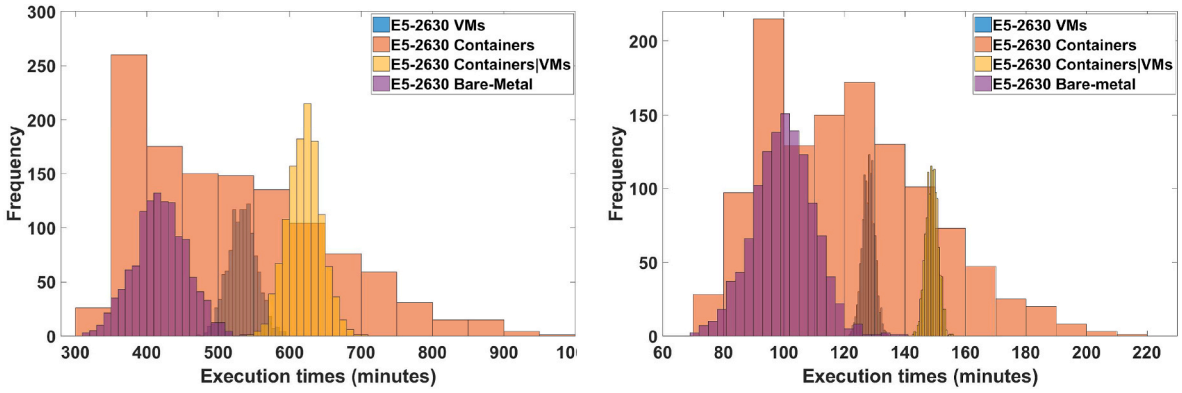


Fig. 7. Variation in runtimes of two different applications (left: Bzip2 – right: Povray) when run over different sand-boxing technologies, on E5-2630 CPU platform [both applications, when run in virtualised containers, may perform better than VMs and containers; containers could be as bad as best in certain scenarios].

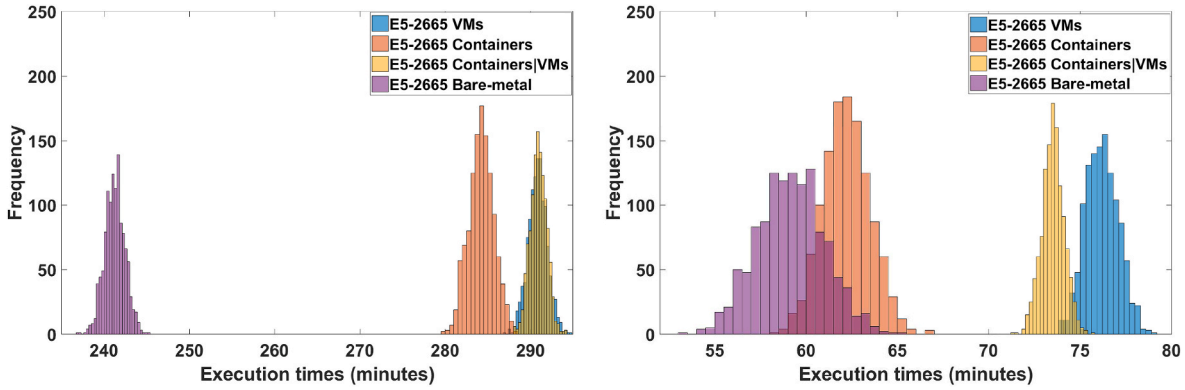


Fig. 8. Variation in runtimes of two different applications (left: Bzip2 – right: Povray) when run over different sand-boxing technologies, on E5-2665 CPU platform [both applications, when run in virtualised containers, may perform better than VMs and containers; containers could be as bad as best in certain scenarios].

(runtimes) in order to extract resource and application performance parameters (Zakarya and Gillam, 2019). In this section, we discuss heterogeneities of various applications when run on four different sand-boxing technologies. Morabito et al. (2015) and Felter et al. (2015) have discussed virtualisation, containerisation and virtualised containers. Their investigation suggests that virtualisation performs worse than the other techniques (~45.76%); however, virtualised containers performs better than virtualisation (~4%) and worse than containers (~26.46%). These variations in performance can be related to CPU models (O'Loughlin and Gillam, 2014); and affect user monetary costs and providers revenues.

4.2.1. Virtualisation

Virtualisation can increase the utilisation levels of datacenter resources (such as CPU, memory, disk), however, they suffer from performance degradation due to resource contention or interference – particularly when VMs with similar workloads compete for same resources (Xu et al., 2016). Moreover, similar workloads (running inside similar instance types) perform quite differently on various CPU architectures (O'Loughlin and Gillam, 2014). The distribution of runtimes of a particular workload on a specific CPU platform can be modelled as log-normal. Moreover, certain platforms could offer the best performance for certain workloads; however, their performance is questionable for other kinds of workloads. As shown in Table 1, these variations in runtimes, given in terms of minimum (Min), maximum (Max), mean (μ), standard deviation (σ), and coefficient of variance (CoV), could be significant – thus have notable impact on infrastructure energy efficiency, workload performance and, therefore, users monetary costs. Note that, Pxz is also a compression tool, similar to Bzip2; which is widely available in Ubunto and Fedora operating systems. Furthermore, we assume the Stream workload throughput (i.e. data copied in MB/s) as a proxy of VMs—containers runtimes; and the order of hosts performance is adjusted to MB/s. As lower MB/s means longer runtime to transfer data, therefore the graph in Fig. 6, Fig. 7, Fig. 8 would be interpreted with the best performance from right to left (Zakarya, 2017).

4.2.2. Containerisation

Containerisation is an alternative technique to virtualisation; that have been largely used in public datacenters such as Google. Similar to VMs, containers suffer from performance variations for similar workloads (Ruan et al., 2016). These variations can also be related to various CPU platforms, as shown in Table 1. Management platforms, for example the Google's Kubernetes, impose affinity constraints (for co-location) in order to guarantee that numerous workloads and applications (with resemblances in resource demands and usage) could be crowded and accommodated over same hosts (Medel et al., 2016). In Kubernetes, this is achieved though groups of containers known as pods. Sharma et al. (2016) experimentally proved that compared to VMs, co-located containers suffer from large performance degradation and interference. Consequently, container placement and migration policies desire to be more improved in order to pick the precise set of co-located and neighbouring containers on a particular host. Kozhirbayev et al. (Kozhirbayev and Sinnott, 2017) have compared bare-metal and two containerised platforms i.e. LXC and Docker, using various benchmark workloads. Their evaluation, on a real cluster, suggests that the performance of both containerised platforms vary for various workloads. Since, containers share the operating system, and other binaries of the physical host; therefore, it is possible to fit three times more containers

on the host as VMs.

4.2.3. Containerisation over virtualisation

The lack of isolation and efficient resource sharing with resource over-subscription (soft limits) makes running containers inside VMs a more feasible architecture [as happens in AWS EC2 container service, Lambda that uses Dockers and Google container engine]. Soft limits enable applications to use resources beyond their allocated limits if those resources are not in use (over-subscription). For VMs, resource limits are usually hard which means that VMs are not allowed to utilise more resources than their provisioned resources even if the resources are idle. Soft limits and over-subscribing resources may provide efficient resource utilisation and management (Dabbagh et al., 2016). Merging containers with VMs could provide both the benefits of containers and VMs. For example, VMs are securer than containers and containers could provide better resource utilisation levels than VMs. Furthermore, Sharma et al. (2016) stated that containers in VMs provide performance benefits as well. The authors suggest that neighbouring containers within a VM can be trusted because containers from a single tenant may be allowed to run in a particular VM. Mondesire et al. (2019) suggest that running containers inside VMs instead on bare-metal have at least two advantages: (i) if a particular container needs restarting, only the VM, which accommodate this container, will be restarted; and (ii) snapshots of VMs could be migrated to other hosts, if needed.

Table 1 show variations in runtimes when various workloads or functions are being executed in virtualised containers (containers run inside VMs) over different CPU architectures. These statistics were collected from various published works (Felter et al., 2015), (Kominos et al., 2017), (Vaucher, 2015). Using monte-carlo simulations for these statistics (with respect to mean and standard deviation), we observed that containers running inside VMs could increase resource utilisation; however, the workload performance is negatively affected, probably, due to large number of co-located containers (O'Loughlin, 2018). Moreover, for certain workloads, virtualised containers may provide for comparable performance to bare-metal; however, for other workloads, the reverse might be true as their performance is lower than VMs. This is illustrated visually in Fig. 8.

4.2.4. Bare-metal

The business requirements of organisation may include full and privileged access to raw hardware which they provision for their services. This may also happen for certain workloads types, such as HPC and real-time applications, that need the best performance. Therefore, it would be essential to run user's application on bare-metal hardware instead of packing it in a container or a VM. Amazon AWS has recently launched bare-metal instance class which offers complete access to the provisioned resources (the recent M5 instances). Similarly, the Rackspace cloud also offers bare-metal hardware known as "OnMetal". Moreover, bare-metal offers several advantages over containers and VMs, e.g.: (i) security - multiple containers—VMs on a single VM—host create chances for network attacks such as denial of service; and (ii) performance is affected - the "noisy neighbour" or co-location problem. Few works have characterized the performance of containers, VMs, virtualised containers and bare-metal; and largely, the later one performs better than the former ones (Felter et al., 2015), (Morabito et al., 2015), (Sharma et al., 2016). As shown in Table 1, variations in runtimes can be seen across various CPU platforms (architectures) for various kinds of applications and hardware platforms (VMs, containers, container over VMs, bare-metal). Several reasons, such as resource or CPU contention, cache design, for these kinds of variations over bare-metal hardware are described in (O'Loughlin, 2018).

In Table 1, some statistics such as μ, Min and Max were being calculated manually; as they were not available in the literature. For example, where Min and Max values were not found but μ and σ were present, we used monte-carlo simulations and statistical distributions (log-normal for VMs and normal for others) to calculate them from the μ and σ. Moreover, where μ and σ were not found, we used range equation to calculate them from the Min and Max. Figs. 6–8 show the performance (runtimes) of various applications, when run over various sandboxing technologies. Large variations can be seen for Povray benchmark on X5560 as compared to Bzip2. Moreover, containers could be of comparable performance to bare-metal; however, in certain scenarios their performance may be even worse than VMs and containers—VMs both. We speculate this situation might happen due to numerous co-located containers on a host, with similar workloads competing for same resources. Fig. 8 illustrates that containers could offer comparable performance to bare-metal; however, bare-metal consume more energy due to lower resource utilisation levels. Moreover, for certain workloads, the performance of virtualised containers may be worse than VMs.

5. Performance evaluation

Bin-packing issues are NP hard and are, therefore, solved using heuristics approaches, usually. Albeit, heuristics may not ensure optimal outcomes, however, they can quickly reach to an approximate (or near to optimal) solution for large-scale problems (Ferreto et al., 2011). We consider the resource placement and consolidation problem as migrating from a particular datacenter state (current) to an ideal state. The ideal state is the one that uses the fewest hosts in order to run all workloads, possibly, those which are energy, performance, therefore, cost efficient. We obtain a datacenter state through the implementation of the proposed HeporCloud framework, along with consolidation policy that ensures both energy and performance efficiencies. To demonstrate the impact of this on IaaS resources and workloads, we presume: (i) no migration; (ii) migrate all; (iii) migrate workloads individually either from VMs, containers, bare-metal, or virtualised containers (containers over VMs); and (iv) HeporCloud which predicts an effective migration among various migrations possibilities. Moreover, we consider various scheduling policies. Furthermore, we also investigate migrations performed within a particular platform (inter-platform) and those triggered across various platforms (intra-platforms).

We presume the process of consolidation with migration as an optimisation issue with the aim to reduce the number of hosts needed to run a particular workload. The optimisation module runs after each 5 min, based on the current levels of utilisation of all switched on hosts. The whole process happens in three steps; (a) *VMs—containers selection:* All hosts are regularly monitored and when their current utilisation levels are lower than some pre-defined low threshold value U_{low} (e.g. 25%), then all accommodated VMs and containers on these particular hosts are marked for migration. In case, there are several VMs and/or containers appropriate for migrations, then the proposed VMs—containers selection policy [Alg. 2] prioritises those VMs or containers which may offer higher margins for costs savings ($\mathcal{P}_{savings}$) – (we prefer to migrate one VM or container from a particular host, at a time, to reduce performance loss); (b) *hosts selection:* The migration approach looks for the most appropriate (i.e. EPC aware) host from all available hosts that might run these migratable VMs and/or containers energy and performance, therefore, cost efficiently. Nevertheless, in order to further reduce the total number of active hosts, the allocation and migration policies circumvent allocations to: (i) switched off hosts (if possible); and (ii) hosts that might change their status to idle or switched off states (i.e. hosts which are switched on but are idle); and (c) *placement:* The list of all VMs and containers which are appropriate for migrations is sorted in decreasing order of their estimated future runtimes ($R_{predicted}$). This further ensures that long-running VMs—containers will be migrated first – since they guarantee effectiveness of their migration efforts. In last, a particular VM and/or a container allocation algorithm(s) (heuristics) is (are) used to re-allocate all migratable VMs and containers. Note that, placement of migratable entities is a sub-problem of the overall consolidation with migration procedure (Zakarya, 2017).

Evaluation metrics: We consider total number of migrations (inter-platform and intra-platforms), energy consumption (KWh) and

Table 2
Various characteristics of Intel, Microsoft Azure and Google datasets.

Dataset	Size (Zip) GB	Number of instances	Number of users	Period (traced)	Important fields	Runtime distribution
Intel	2.46	48,821,850	4727	Oct 2016 Nov 2016	life time, user, cores, memory	*lognormal*
Azure	18.8	2,013,767	5958	Nov 16, 2016 Feb 16, 2017	life time, average CPU, memory, disk, VM size, category	*lognormal*
Google	41.0	24,281,242	922	May 2011	runtime, CPU, memory, disk, priority, user	*lognormal*

Table 3
Various characteristics and properties of different datasets (sampled) used in simulations; variations in these statistics might produce different outcomes.

Dataset	Size (no of tasks)	Runtimes (seconds)		CPU demands (utilisation %)		Memory demands (MBs)		Instance type
		μ	σ	μ	σ	μ	σ	
Intel	6.12k	25.42	11.89	51.09	5.56	764.32	541.89	server
Microsoft Azure	6.25k	49.92	16.65	44.32	5.32	1002.45	994.87	vCPUs
Google	6.01k	49.36	9.83	67.34	10.34	865.45	654.66	cores
Synthesized	6.31k	102.58	37.06	63.76	11.78	982.87	867.99	cores—vCPUs

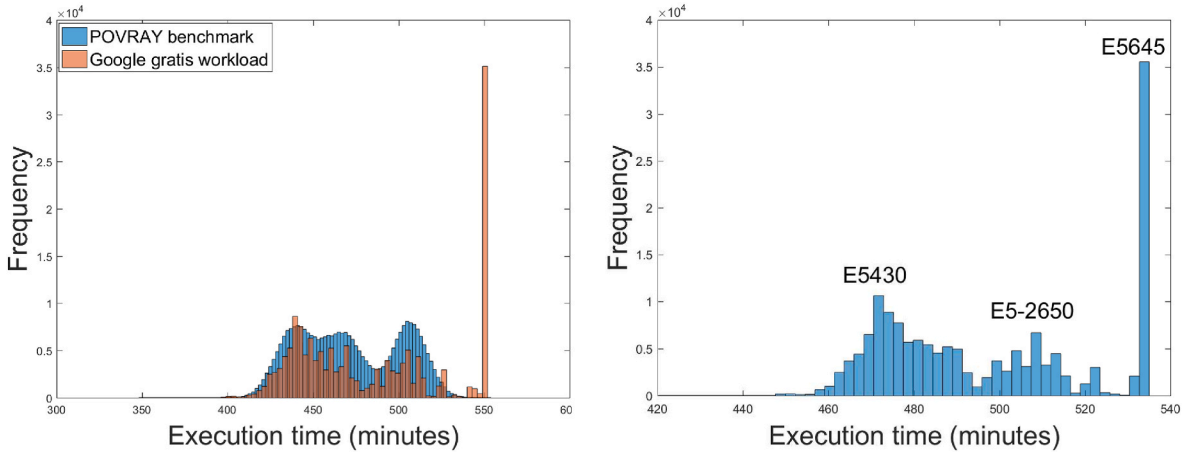


Fig. 9. Mapping Google data to real benchmarks (left) and plausible assumptions for choosing appropriate hosts (right) (Zakarya and Gillam, 2019) – performance parameters for various hosts are shown in Table 4 [Povray workload performs best on E5430 and worst on E5645].

Table 4
Different benchmarks runtime parameters for lognormal distribution (Zakarya and Gillam, 2019), (O'Loughlin and Gillam, 2014).

Workload	Benchmark	CPU Model	Real benchmarks				Google data				$\mathcal{F}$
			(μ)	(σ)	Min	Max	(μ)	(σ)	Min	Max	
Gratis	Povray	E5430	439	11	421	467	438.06	9.42	421	467	$\mathcal{F} < 7.65$
		E5-2650	468	12	451	500	473.87	11.93	451	500	$9.75 > \mathcal{F} \geq 7.65$
		E5645	507	10	490	535	498.55	10.44	490	535	$\mathcal{F} \geq 9.75$

Table 5
Hosts various characteristics for Amazon's cloud (simulated).

CPU MODEL	SPEED (MHz)	NO OF CORES	NO OF ECUs	MEMORY (GB)	P_{IDLE} (Wh)	P_{MAX} (Wh)	AMOUNT
E5-2630	2300	12	27.6	128	99.6	325	12,583
E5430	2830	8	22.4	16	166	265	
E5507	2533	8	20	8	67	218	
E5-2620	2000	12	24	32	70	300	
E5645	2400	12	28.8	16	63.1	200	
E5-2650	2000	16	32	24	52.9	215	
E5-2651	1800	12	21.6	32	57.5	178	
E5-2670	2600	16	41.6	24	54.1	243	
E5540	2500	4	10	72	151	312	
X5560	2800	8	22.4	128	133	288	
E5-2665	3000	8	24	256	117	314	
X5650	2666	12	31.2	64	80.1	258	

Table 6
Amazon different instance types and their characteristics – Mem means memory (RAM).

Instance type	No of vCPUs	No of ECUs	Speed (GHz) MIPS	Mem (GB)	Storage (GB)
t2.nano	1	1	1.0	0.5	1
t1.micro	1	1	1.0	0.613	1
t2.micro	1	1	1.0	1	1
m1.small	1	1	1.0	1.7	160
m1. medium	1	2	2.0	3.75	410
m3. medium	1	3	3.0	3.75	4

Table 7
Container types and their characteristics.

Container type	Speed (MHz)	Cores	ECU's	Memory (MB)
A	1000	1	1	128
B	1225	1	1.23	256
C	1500	1	1.5	512

workload performance (execution time measured in minutes) as the performance evaluation metrics.

Datasets: The datasets used in this paper relate to three different workloads i.e. HPC - Intel (Shai et al., 2013), Virtualised - Microsoft Azure (Cortez et al., 2017), and containerised - Google's cluster (Reiss et al., 2011). Due to non-availability of a virtualised containers workloads, we used synthesized workloads, extracted from benchmark results which are described in Table 1. Each dataset consists of various task' parameters such as submit & finish time, resource (CPU, memory, disk) demand & usage, task priorities, submitting user, VM type & category, scheduling or workload class, and etc. The type of each workload corresponds to tasks' priorities (containers in Google) and VM category (virtual machines in Microsoft Azure). In Azure cloud, the runtimes curves show a knee around 24 h (i.e. more than 90% VM life times are short), and then the curve is almost flattened out. This demonstrates that, VMs that run for 24 h are likely to run for longer. Moreover, a small amount of long-running VMs account for CPU resources. The same observations can also be seen for Google's cluster data (Zakarya, 2017) – where 90% tasks run for less than an hour. In Intel's clusters, largely jobs require a single core, and, largely, jobs runs for short durations [80% jobs run for less than 10 min and 90% run for less than an hour] (Shai et al., 2013).

This discussion is relevant to aggregation-based placement and migration since: (i) aggregating shorter and longer tasks might be affective(w.r.t energy consumption and performance); and (ii) migrating shorter tasks may not be affective as their migration efforts might be wasted (Zakarya and Gillam, 2019). Moreover, VM workloads runtimes are strongly correlated to workload types (Cortez et al., 2017); while container workloads runtimes can be highly related to submitting users and resource i.e. CPU, memory, disk usage patterns. Several important characteristics and properties of these three datasets are shown in Table 2.

Since, it will take significant time and, most importantly, a cluster of resources to simulate and replay all these workloads. Due to non-availability of a cluster, we consider random samples where each sample comprises approximately six thousands tasks, extracted from each of these datasets as representative workload. The tasks runtimes, inside a particular workload type, are assumed as proxies to represent variations in performance. The histogram of each sampled dataset can be seen as a multi-modal distribution, where modality relates to different CPU architectures and the statistics of each model denote the CPU performance. Various characteristics and properties of these sampled datasets, in terms of tasks level statistics, are shown in Table 3. Note that, the sums of execution times (in hours), of all tasks in different workloads are approximately 43.22, 68.67, 82.41, and 179.8, respectively. These execution times are, then, used to compare the performance of various workloads (evaluation metric).

Unfortunately, the above datasets does not have neither resource contention, nor workload and machine performance details - in terms of VM placement and migration techniques. Therefore, in order to extract parameters for resource contention for each workload on every host, we mapped runtimes' histograms of these workloads and real benchmarks. In order to carry out that, we plot these histograms on the same scale (Zakarya and Gillam, 2019). To find closer similarity among the mapped histograms, we used their statical values i.e. means (μ, μ_1) and standard deviations (σ, σ_1) - where μ, σ and μ_1, σ_1 denote the means and standard deviations of above workloads (Table 2) and real benchmarks (Table 1). The closer the means and standard deviations, the more accurate is the mapping. This method enable us to find the best CPU models for our workloads, and therefore, appropriate performance parameters. For example, the right-hand side of Fig. 9 shows the distributions of runtimes over the same scale for both Google data (priority 0) (Zakarya and Gillam, 2019) and real benchmarks (O'Loughlin and Gillam, 2014). The left-hand side of Fig. 9 shows various hosts or CPU models i.e. E5430, E5-2650, E5645; which are chosen using a distribution factor ($\mathcal{F}$), as described in Table 4. The closest statistical means and standard deviations without overlaps are assumed as the mapping criteria and extract performance parameters. Further details on the statistical mapping of the Google data to benchmark workloads are discussed in (Zakarya, 2017).

5.1. Experimental setup

We performed all the experiments in an event-driven simulation environment through integrating two well-known cloud simulators i.e. CloudSim (Calheiros et al., 2011) and ContainerCloudSim (Piraghaj et al., 2017). The former one offers support for virtualisation technology (VMs) and the latter one is recognized for containerisation technology (containers). Nevertheless, ContainerCloudSim does not provide the ability of running nested/virtualised containers (containers that run inside VMs). Furthermore, its presently existing version does not care for VMs migration, nevertheless, partial operation of the optimisation segment can be realized in the current code. We prolonged its numerous classes, like the optimisation segment and the broker, in order to accomplish our evaluation and experimentations. The extended version of the broker class has the competence to boot up a VM—container, during simulations, as soon as a request is received. The request arrival matches task arrival durations from the Google cluster workload trace. A cluster (simulated using extended CloudSim) of 12,583 heterogeneous hosts/servers, which consists of various architecture types (regarding fluctuating performance) and CPU hardware specifications – as shown in Table 5 - is ready to execute numerous types of benchmarked workloads. The simulated servers are configured based on several reasonable assumptions as described in Sec. 4. The hardware specification (CPU) and energy usage details for various servers were taken from the well-known SPECpower[9] benchmarks. The servers' energy consumption, VMs—containers migration energy costs and performance of VMs—containers were supposed as illustrated in (Zakarya, 2017). Additionally, we used four different types of workloads, i.e. Hpc (bare-metal), W_1 (VMs), W_2 (containers) and W_3 (virtualised containers), that belong to real data from major cloud providers i.e. Intel's compute cloud (Shai et al., 2013), Google (Reiss et al., 2011) and Microsoft Azure (Cortez et al., 2017) clusters, respectively.

Our simulation environment consists of six types of VMs and three types of containers which relate to Amazon's instance classes as given in Table 6 and Table 7, respectively. Further, the VMs and containers types are ranked (regarding their performance levels and resource capacities)

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

Table 8
Virtualisation and containerisation total migration-time and down-time (in seconds) at various workloads with different utilisation levels (Kotikalapudi, 2017).

Workload	66% utilisation				100% utilisation			
	mean	std. dev.	min	max	mean	std. dev.	min	max
				KVM				
Downtime	14	1.886	12	17	16	2.404	13	21.5
Migration time	119.1	3.281	115	125	129.8	3.553	125	135
				LXC				
Downtime	8	1.155	6	10	10	1.886	7	12.5
Migration time	82	3.266	77	90	95.7	1.703	92	98

Table 9
Configuration times in seconds [ProLiant DL580 Gen8 with Ubuntu Server 14.04] – the host configuration energy consumption is computed based on its maximum power consumption and the transition times from one state to another (Zakarya, 2017), (Dabbagh et al., 2016).

	Container	VM	Container—VM	Host
Start	1.623	3.005	4.628	600 (Kominos et al., 2017)
Off	2.493	64.422	66.915	
Restart (soft)	4.209	125.463	129.672	
Restart (hard)	4.338	6.043	10.381	
Delete	2.473	3.767	6.24	

according to the terminology and measurement of Amazon's for gauging performance rating of their offered instances – ECU (EC2 Compute Unit), which is defined as: "equivalent CPU capacity of a 1.0 GHz–1.2 GHz 2007 Opteron or 2007 Xeon processor". Moreover, for certain workload types, variations in their performance levels over various instances in the AWS cloud are suggested as equal to ~20% (1.0 GHz–1.2 GHz) of the workloads original runtimes (O'Loughlin and Gillam, 2014). Note that, the above ECU rating is described as per core, thus, the total rating of a particular host or VM (with multiple cores or vCPUs) can be computed through multiplying their ECU rating and total number of cores or vCPUs (O'Loughlin and Gillam, 2014).

We presume that these hosts and VMs are equitable by just a single measure which allows for performance ranking, and for which we consider the CloudSim's "Million of Instructions Per Second (MIPS)" terminology as a proxy. One possible approach for VM sizing is to assign every VM a single core (hyper-threaded) for the maximum value of 1, half a core for 0.5, and assume that higher gearing of a VM leads to a quarter of a hyper-threaded core for 0.25. However, along lines with specific service providers (IaaS), and to more flexibly address the resource allocation, we map frequencies of hosts' CPUs to that of Amazon's notion of ECUs as: 1 GHz CPU, 1.7 GB RAM, giving various instance (container—VM) types[10] (Zakarya, 2017). Moreover, to keep consistency with the "CloudSim" simulator each ECU maps the notion of MIPS (Table 6), and we assume that every container—VM, at least, needs 1 ECU and 1 vCPU (hyper-threaded core) or more, as shown in Table 6. Therefore, the speed of every container—VM type (also called the MIPS rating) is the multiplication of the total number of ECUs (1 ECU = 1 GHz) and vCPUs (hyper-threaded cores). For example, the CPU speed of a *m3.medium* VM type, as shown in Table 6, is 3 (ECUs) × 1 (vCPU) = 3 (GHz).

To address a public cloud terminology more closely, for example Amazon Lambda, each Azure's, Google's task is allocated to a single, notional, VM or container, that relates to Google machine types with the only exception that all VMs and containers are exactly single core, as given in Table 7. Further, very similar to sizing of VMs, containers are also sized (according to MIPS) and every VM can host, at least, one or more than one container. For example, an instance *m1.medium* can host exactly one container of type C (no resource over-subscription), while two containers of type A, and so on. Moreover, every task utilises its allocated resources (CPU, memory, disk) according to resource usage and tasks' statistics in Intel's cloud (Shai et al., 2013), Google (Reiss et al., 2011) and Microsoft Azure datasets (Cortez et al., 2017). Note that, in our experiments we do not account for resource over-subscription of containers and/or VMs (Dabbagh et al., 2016).

Each container is assumed to run four workload types that belong to either Intel's cloud (Shai et al., 2013), Google's cluster (Reiss et al., 2011) or Microsoft Azure (Cortez et al., 2017) datasets. These datasets denote four various kinds of workloads i.e. HPC (bare-metal), containers, VMs, containers—VMs, respectively. Due to the unavailability of the fourth dataset i.e. virtualised containers (when containers run inside VMs), we used a synthesized workload which was derived from various statistics such as mean (μ) and standard deviation (σ), as shown in Table 1. Further, the utilisation of each workload type is modelled as a normal distribution function over the mean CPU usage, as calculated from the original traces, at 5 min intervals. Moreover, the total execution time of each application is the sum of its every tasks' execution times. Each dataset consists of approximately 6000 tasks whose sum of execution times are 43.22, 68.67, 82.41 and 179.8, in hours.

To begin with, all containers and VMs were assigned resources conferring to: (a) the resource necessities well-defined by the type of container—VM; (b) container placement strategy; and (iii) VM placement strategy. To ensure that each workload executes on a cheap (cost-effective) and suitable container—VM, the instance-type selection algorithm, as proposed in (Zakarya, 2018b), was implemented. This ensures reduction in stranded resources. It is notable that entirely containers—VMs were assigned to VMs—hosts by means of the workload-aware FF placement strategy and their coming match to task coming epochs and proportion in Google and Azure clusters datasets (Zakarya, 2017). Nevertheless, by means of the workload trace, if containers—VMs exploit their provisioned IaaS resources fewer, this produce chances for resource consolidation. Furthermore, every container—VM is arbitrarily given a workload data trace from one of the instance types (Intel + Azure + Google + Synthesized datasets) which executes till its computation is finished. Throughout the consolidation

Table 10
Energy consumption in kWh [the values followed by ± symbol denote standard deviations] - the overlap for no migration and migrate all approaches under HeporCloud field represents the trade-off between efficient allocation and migration.

Policy	Bare-metal	VMs	Containers	Containers—VMs	HeporCloud
No migrations	2563	2678	2792	2934	2412 [±121]
Migrations				–	
Migrate all	2982	2856	2921	2785	2353 [±437]
HeporCloud				–	
inter-platform	2456	2087	2873	3173	1782 [±57]
intra-platforms	2389	2129	2666	3198	1759 [±98]

[10] http://www.ec2instances.info.

Table 11
Performance of various workloads using different techniques to migrations and virtualisation (in minutes) [the workload prediction accuracy represents scheduling and resource prediction accuracy corresponds to the efficiency of hosts].

Migration policy	Workload type	Sand-boxing technology					Prediction accuracy	
		Bare-metal	VMs	Containers	Containers—VMs	HeporCloud	Workload	Resource
No	Hpc	43.22	67.32	51.52	65.83	43.22	–	–
	W_1	66.89	72.77	69.8	73.1	68.67	–	–
	W_2	71.43	98.19	87.71	91.88	82.41	–	–
	W_3	173.2	211.8	198.2	191.9	179.8	–	–
	HeporCloud	170.91	207.29	199.11	190.87	172.77	59.8%	78.8%
		inter-platform migrations						
Hpc	Hpc	44.98	67.32	51.52	66.32	43.22	–	–
	W_1	69.12	72.77	69.8	74.9	68.67	–	–
	W_2	76.89	98.19	87.71	94.76	82.41	–	–
	W_3	147.1	172.3	168.9	167.9	155.6	–	–
W_1	Hpc	43.22	78.86	51.52	55.56	43.22	–	–
	W_1	66.89	79.74	69.8	80.8	68.67	–	–
	W_2	71.43	99.14	87.71	99.12	82.41	–	–
	W_3	147.1	172.3	168.9	167.9	155.6	–	–
W_2	Hpc	43.22	67.32	52.67	69.54	43.22	–	–
	W_1	66.89	72.77	70.1	69.98	68.67	–	–
	W_2	71.43	98.19	87.93	88.34	82.41	–	–
	W_3	147.1	172.3	168.9	167.9	155.6	–	–
W_3	Hpc	43.22	67.32	52.67	69.54	43.22	–	–
	W_1	66.89	72.77	70.1	69.98	68.67	–	–
	W_2	71.43	98.19	87.93	88.34	82.41	–	–
	W_3	141.9	170.1	166.9	167.0	151.4	–	–
Migrate all	Hpc	44.99	68.89	52.56	65.99	44.67	56.1%	43.23%
	W_1	69.78	72.01	68.88	75.31	68.12	60.4%	44.8%
	W_2	77.17	98.79	88.34	93.97	84.09	58.9%	46.2%
	W_3	149.87	177.11	169.88	177.55	159.11	59.7%	47.8%
HeporCloud	Hpc	43.29	68.21	53.78	56.67	**45.22**	60.1%	79.9%
	W_1	67.34	72.89	70.3	68.12	**68.97**	62.2%	80.2%
	W_2	71.67	99.7	88.01	81.07	**72.99**	58.9%	80.3%
	W_3	141.9	170.1	166.9	167.0	**151.4**	59.5%	79.5%
		intra-platforms migrations						
Hpc		51.91	69.23	55.89	60.78	**49.32**	–	–
W_1		66.99	71.56	73.67	69.32	**65.21**	–	–
W_2		74.32	94.12	89.45	82.78	**74.1**	–	–
W_3		139.23	169.89	165.54	164.87	**153.23**	–	–
Migrate all		141.76	168.12	164.32	165.89	**155.99**	61.45%	48.8%
HeporCloud		138.89	166.23	165.23	165.98	**148.23**	62.98%	84.8%

Table 12
Energy consumption (in kWh) using single and distributed schedulers [Improvement means using single (centralised) scheduler instead of distributed (individual) for all platforms].

Policy	Distributed scheduler	Single scheduler	Improvement (%)
No migrations	2742 [±159]	2412 [±121]	7.62–16.86
	Migrations		
Migrate all	2886 [±85]	2353 [±437]	0.39–20.8
inter-platform	2647 [±475]	1782 [±57]	15.33–44.75
intra-platforms	2596 [±458]	1759 [±98]	13.14–45.61

phase, the optimisation part governs the under-utilised and over-utilised servers by means of two procedures: (a) the one threshold strategy – that practices a static higher utilisation threshold (e.g. 80%) for every server; and (b) two threshold's strategy – that practices two thresholds values i.e. a lower threshold (e.g. 20%) and an upper threshold (e.g. 80%). Regarding the outcomes offered in this manuscript, we practice the later method i.e. double threshold policy. In case, there exist numerous migratable containers—VMs, at that moment the consolidation strategy practices their *rating* [computed as the product of energy usage and estimated execution time – as described in Sec. 3] to rank first those containers—VMs which might save large energy whereas supreme levels of workload performance is preserved ($\mathcal{P}_{savings}$). Additionally, the destination hosts—VMs for all migratable VMs—containers are recognized by means of a modified version of the default host selection policy in ContainerCloudSim (Piraghaj et al., 2017). The modified host selection policy accounts for host's *ERP* in order to pick an energy and performance efficient one for the migratable entity.

In order to account for migration costs (energy usage and performance loss), we made minor amendments to the above experimentations. The migration duration of each VM is computed from its size, type and network capacity (bandwidth), along with the period wanted to boot up the fresh VM. Previous works (Calheiros et al., 2011), (Piraghaj et al., 2017) have supposed that half of the entire network capacity is accessible for the migration of VMs—containers, whereas the remaining half is kept for communication between VMs—containers. Other works (Felter et al., 2015), (Amaral et al., 2015) adopt that the migration period of a container is the duration desirable to boot up a fresh container at the destination server. In case, a previously existing VM cannot run a particular container, at that point a fresh VM is booted first – thus, the migration period of the container is the sum of booting up a VM and a container along with the memory pages dirtied by the container during the migration. To simplify concerns, these works further accept that the container is stateless, subsequently leading to no dirtied pages that should to be migrated. In our experiments, the migration total time and down time are taken from (Kotikalapudi, 2017), which are experimentally evaluated and account for network costs in terms of communication and delay, as shown in Table 8. Furthermore, datacenters are usually equipped with dedicated networks; and these costs would largely impact migrations across several datacenters (geographically distributed). This study is limited to various types of migration, however, within a single datacenter. Moreover, consolidation can bring several hosts to idle states, thus making it possible to switch off (bring to sleep states) them in order to save energy. Switching on/off and states transitions of hosts can bring scheduling

Table 13
Energy consumption and workload performance using single and distributed schedulers for various datacenter set-up, workload types [the results are averaged over different platforms].

Policy	Datacenter size	Workload type	Single scheduler		Distributed scheduler	
			Energy (KWh)	Performance (hours)	Energy (KWh)	Performance (hours)
No migrations	12,583	$W_1 + W_2$	2993	142.67	3003	143.88
Migrate all			2283	126.56	2487	129.43
HeporCloud			2109	128.33	2349	128.82
No migrations	25,166	$W_1 + W_3$	3789	213.67	3237	200.78
Migrate all			3822	218.43	3101	193.56
HeporCloud			3203	199.32	3002	185.08
No migrations	12,583	$W_1 + W_2 + W_3$	2984	294.79	2903	291.34
Migrate all			2845	301.68	2699	290.56
HeporCloud			2601	281.2	2576	274.93

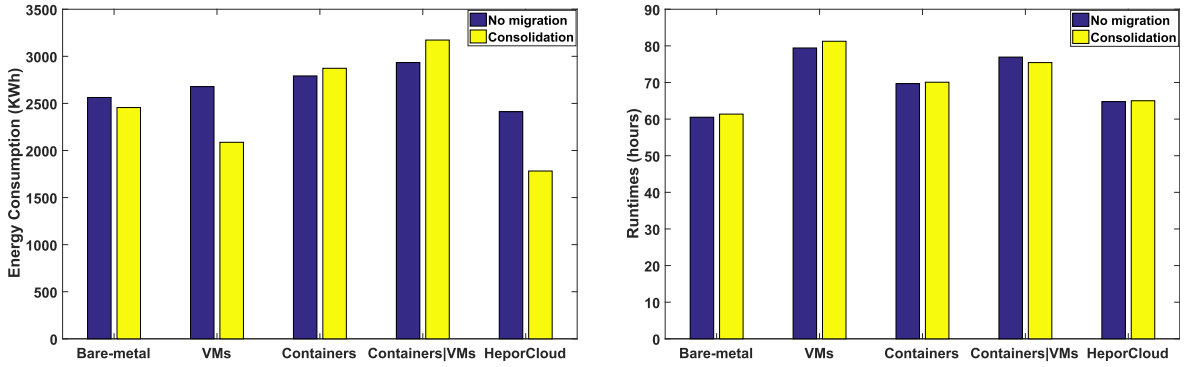


Fig. 10. Energy consumption (KWh) [left] and overall performance (execution times in hours) [right].

delays. We assume container—VM start, reboot, off times as shown in Table 9. These costs (delays) are very important, in the context of cloud computing, as users are charged for the total duration of their service [workloads runtimes + instances launch times]. These values are also used inside ContainerCloudSim (Piraghaj et al., 2017), by default. Furthermore, containers—VMs performance is affected as described in Sec. 4. Transitions among various states of a host also incur non-negligible energy overheads (Zakarya, 2017).

5.2. Experimental results

The simulated cloud environment is composed of 12,583 hosts, 3800 containers—VMs with configuration shown in Table 5, Table 6 and four kinds of workloads that belong to bare-metal, virtualised, containerised and virualised containers, respectively. When a container—VM request is received, a container—VM is created from a list of available flavours as shown in Table 7, Table 6 and is placed on a suitable host or a VM which is already running on a particular host. The scheduler is workload aware which predicts the type of workload; then the platform which might offer performance benefits; and then a most energy and performance efficient host to run the workload. Moreover, the scheduler uses a FF heuristic to place similar workloads in a particular platform. If there is no suitable VM to accommodate the container, then a new VM is created from a list of available instance types as shown in Table 6. Various container types are described in Table 7. We assume that the container workload is heterogeneous and, therefore, changes when a container is being migrated from one host to another. For bare-metal, the container is assigned solely to a particular host; and no other allocation is possible to this host until the container is being terminated and the host resources are free. The allocation policy is workload-aware that allocate containers to appropriate platform. After each 5 min interval, the HeporCloud technique checks for consolidation opportunities, and selects effective migrations from a list of possible migratable entities.

5.3. Results discussion

The obtained results of numerous scheduling and consolidation with migration policies are described in Table 10 (overall energy usage) and Table 11 (workload runtimes). Besides trivial decrease in performance approximately from 2.14% to 3.02%, the suggested HeporCloud framework can considerably reduce the overall energy consumption of the IaaS cloud, that might be as high as 30.47%, and the total number of migrations. Large variations in energy consumption for the migrate all approach demonstrate that uncontrolled migrations could lead to datacenter inefficiency. In our experiments, we found that efficient allocation could be approximately 10.18% more energy efficient than the no migration technique. Similarly, an overlap between inter-platform and intra-platforms migrations shows an existing trade-off that is, largely, dependent on the workload type and datacenter configuration. From performance perspective, intra-platforms migrations are more effective than the inter-platform migrations; if and only if migrations are triggered to more performance efficient hosts. Otherwise, if migrations are uncontrolled then intra-platforms migration are worse

Table 14
Percentages of migratable entities, cost recovery and prediction accuracy [the sign ± denotes standard deviation over various runs].

Migration policy		Migratable %	Cost recovered %	Runtime accuracy
NO		–	–	–
Migrate all	inter-platform	4.01 (±2.1)	24.87 (±2.8)	78.45% (±1.01)
HeporCloud		3.17 (±1.4)	79.02 (±4.2)	79.33% (±1.11)
NO		–	–	–
Migrate all	intra-platforms	6.87 (±2.3)	41.76 (±3.8)	69.41% (±1.88)
HeporCloud		5.01 (±1.2)	75.89 (±2.6)	70.34% (±2.26)

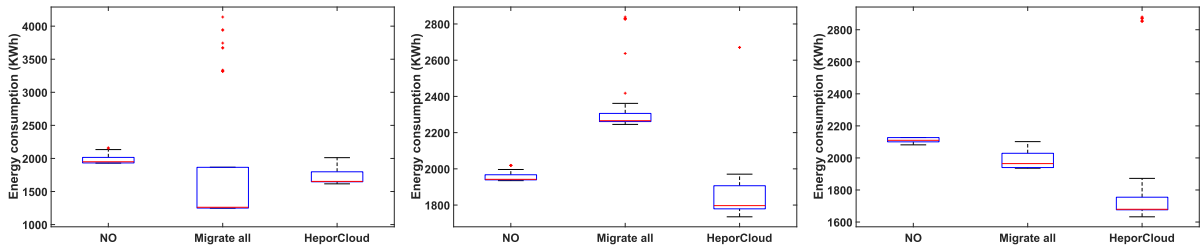


Fig. 11. Variations in energy consumption (KWh) with respect to various migration policies and datacenter configurations [left: INC – middle: NR – right: DEC].

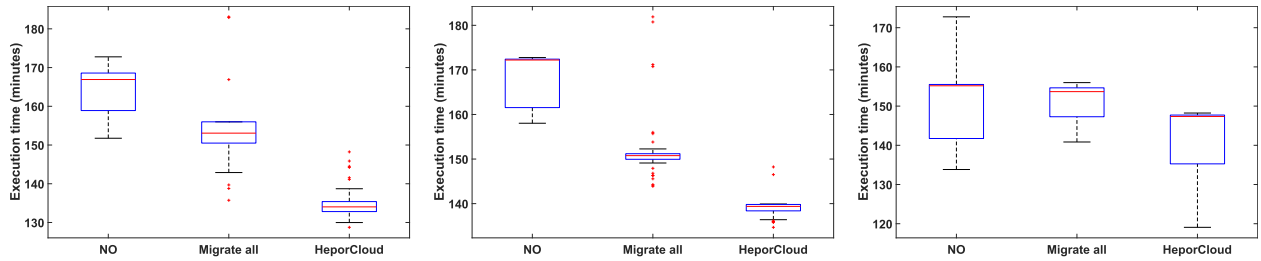


Fig. 12. Variations in workload performance (minutes) with respect to various migration policies and datacenter configurations [left: INC – middle: NR – right: DEC].

Table 15
Variations in energy consumptions (KWh) for various models and workloads.

Policy	Energy model		Workload utilisation	
	mean	st. dev.	mean	st. dev.
No migrations	2114.87	8.89	1934.2	297.93
Migrate all	2467.54	29.87	21.52	892.83
HeporCloud	1741.49	21.83	1877.81	501.1

than inter-platform migration.

An interesting behaviour can be observed when using a single (centralised) scheduler and distributed (individual) schedulers, as shown in Table 12. For example, when migrations are not taken into account, single scheduler is approximately 7.62%–16.86% more energy efficient than the distributed schedulers. However, when migrations are considered, then: (i) for uncontrolled migrations – the single scheduler produces large variations in energy consumption; and (ii) for controlled migrations – the distributed schedulers produces large variations in energy consumption. In both cases, using a single (centralised) scheduler is more energy efficient than using the distributed, but, individual schedulers for each sand-boxing platform. Moreover, the total number of triggered migrations and, therefore, performance of various applications is affected when a single scheduler or several distributed schedulers are used. Since, the single scheduler has knowledge of all platforms and resources; therefore, more performance aware migration decisions can be triggered as compared to distributed schedulers. Moreover, single scheduler can reduce the resource contention and interference that could occur due to co-location i.e. when similar workloads co-located on same hosts and compete for similar resources. However, distributed schedulers can put similar workloads on same resources that subsequently degrade workload performance and, therefore, increases energy consumption. Similarly, centralised scheduler have better control over the available resources, therefore, resource placement and migration can result in well-utilised state for the datacenter (minimum resources switched on). In contrast, distributed schedulers will keep more resources switched on.

Unfortunately, the centralised scheduler suffers from single point of failure. Moreover, when more number of VMs—containers interact with the centralised scheduler, its response could be degraded due to the delay involved in communication with available resources. Moreover, the scalability of the single scheduler is affected with increase in datacenter size and workload resource consumptions. Unfortunately, networks and communication delays are not within the scope of this paper. Therefore, in order to investigate the scalability of the single scheduler, we run several experiments over different datacenter set-ups (sizes) and various kinds of workloads - using same simulation set-up, energy and performance parameters, as initially described in Sec. 5.1. Table 13 summarises our outcomes. Our evaluation shows that for large-scale datacenters, distributed schedulers are performance efficient than the single scheduler [6.03%–11.39%]. However, energy consumption overlaps for both schedulers. Moreover, certain workloads and, therefore, resource placement and migration decisions also create large variations in the performance and energy efficiencies of the single resource scheduler. Therefore, various requirements, datacenter characteristics and workload information must be considered, essentially, before deciding to use either a single or a distributed resource scheduler.

As shown in Fig. 10, the HeporCloud technique is more energy efficient (approximately from 14.61% to 37.97%) and more performance efficient (from 7.23% to 20.0%) than the only virtualisation and containerisation technologies, respectively. When migrations are disabled, our approach is ~6.3% more energy efficient than the bare-metal. However, the savings can be up to 37.8%, if migrations are taken into account to consolidate workloads on fewer hosts. These savings for virtualised containers were observed as 17.8% and 43.8%, respectively. Moreover, our evaluation demonstrates that bare-metal hardware offers optimal performance; however, the energy consumption is reliant, largely, on the workload type and patterns for job arrival (% of job requests in a particular time window). Perhaps, the resource utilisation pattern of a workload (i.e. CPU usage) has higher impacts on overall infrastructure energy consumption. Although, a single job can be run much fast and quickly that could, subsequently, decrease the energy consumption of that particular job; however, the provisioned resources can not be allocated to those jobs which are waiting in the *waiting queue*. Resultantly, increased wait times for jobs increases latencies and runtimes. Furthermore, bare-metal resources are largely under-utilised. These issues may potentially increase job wait times and, therefore, negatively affects the workload performance, user cost and datacenter energy efficiency.

Similarly, VMs might increase IaaS energy consumption due to longer runtimes (poor performance of workloads), in particular, due to co-location which might happen very frequently if VMs workloads compete for same resources. In case of virtualised containers i.e. containers run over VMs, the performance of the containerised workload is limited to the actual resource usage and available capacities of VMs that

Table 16
Variations in average energy consumptions (KWh) and performance for various prediction models.

Policy	Model Eq. (8) (distributions mapping)				Model Average (recent job runtimes)			
	E (KWh)	P (minutes)	Migrations	Accuracy	E (KWh)	P (minutes)	Migrations	Accuracy
Migrate all	2467.54	156.83	856	68.04%	2398.66	157.93	803	83.01%
HeporCloud	1741.49	137.21	589	69.72%	1801.03	139.07	644	82.99%

Table 17
Costs savings [Energy and users monetary costs are described in US dollars].

Policy	Energy costs ($)	Users monetary costs ($)	Total costs savings (%)
No migrations	2654.26	1057.35	–
Migrate all	2793.65	973.75	7.91
HeporCloud	1702.7	907.17	14.2

results in longer runtimes and, therefore, lower energy efficiency. The approach "bare-metal i.e. containers run directly on bare-metal" offers for the highest energy, performance and cost efficiencies. Moreover, if the placement and/or migration policies have knowledge of the type of workloads, and infrastructure, then it is possible to obtain optimal energy, performance, and, therefore, cost efficiencies.

When resource migrations are enabled, then our proposed technique "HeporCloud" and algorithms offer for higher energy efficiency at equitable cost of performance to that of the "bare-metal" approach. Furthermore, since the HeporCloudOrchestrator is aware of all four environments (sand-boxing technologies), therefore, it is possible to decide appropriate, energy, performance and cost efficient (EPC-aware) migrations (Zakarya, 2018b). Unlike several demonstrated outcomes (Vaucher, 2015), our empirical evaluation shows that bare-metal hardware are expensive regarding energy usage, possibly because of the lowest levels of resource utilisation (i.e. whole resources are provided to a single VM—container) (Kominos et al., 2017). Virtualised containers, those run over VMs offer the highest and peak resource utilisation levels; but, unfortunately their performance is limited to VMs performance that hosts them – therefore, consuming more energy than VMs and containers, as shown in Fig. 10. Another reason for this low performance of virtualised containers is, probably, the large ration of co-location and neighbouring containers that might compete for same resources. This suggests a trade-off between VMs and containers (when run inside VMs) with respect to energy consumption and workload performance.

Table 14 shows the migration statistics such as the percentage of migratable entities, those which were able to recover their migration costs, and the accuracy of the prediction technique. The data show that HeporCloud has significantly reduced the total number of migrations and, largely, migrations have recovered their costs. We observed more number of triggered migrations across platforms (intra-platforms) as compared to with a particular platform (inter-platform). The prediction accuracy is computed through analysing the data we gathered during numerical simulations. For example, in case of inter-platform migrations, approximately 79.33% of the migrated entities were continued to run until they recovered their migration costs t_{off}. In other words, the prediction accuracy does not reveal that how accurately runtimes were estimated; instead, it reveals that how many runtimes were lengthier than the t_{off}; and the migratable entities were able to recover their migration costs. In next section, we briefly evaluate the impact of datacenter configurations and energy consumption models on overall infrastructure energy efficiency.

Table 18
Energy consumption of the CIAO and HeporCloud architectures in KWh [minimum values are "best"]; these improvements can be translated to providers energy bills.

Policy	CIAO	HeporCloud	Improvement
No migrations	2662	**2412**	9.4%
Migrations	–		
inter-platform	2076	**1782**	14.20%
intra-platforms	2076	**1759**	15.27%

Table 19
Performance comparison of the CIAO and HeporCloud architectures – performance means workload execution time in minutes [the "best" results are shown in boldface]; these improvements can be translated to users monetary costs.

Workload type	Performance		Improvement
	CIAO	HeporCloud	
HPC	55.09	**45.22**	17.9%
W_1	70.18	**68.97**	1.7%
W_2	86.46	**72.99**	15.6%
W_3	170.91	**151.4**	11.42%

5.4. Datacenter configuration impact on energy consumption and performance

How hosts are arranged and addressed in an IaaS datacenter (configuration) has a possible impact on the resource allocation approach. For instance, if hosts are kept and arranged in an increasing order of their energy consumption (efficiency factor – E_f) and a particular algorithm is used to allocate these resources; then, the total energy consumption would be different in circumstances if same hosts are ordered in decreasing order of their energy efficiencies. Moreover, if hosts are kept physically or ordered logically based on their expected levels of performance (CPU models), or both energy consumption and performance (*ERP*), then the trade-off between energy consumption and performance would vary for various resource allocation, consolidation (migration) policies and workloads. We demonstrate the potential impact of host ordering and allocation approaches (how hosts are addressed logically) on infrastructure energy efficiency and workload performance, therefore, costs. Each resource allocation and migration policy chooses a particular host to execute the given workload; where the starting point for such re-allocation decisions could, possibly, produce variations in energy consumption, performance and, therefore, users monetary costs. For instance, if the initial ordering were reversed, this may change the experimental outcomes in terms of energy consumption and workload performance (Zakarya, 2017).

Table 20
Energy Runtime Product (*ERP*)[14] to demonstrate the trade-off between energy consumption and performance using CIAO and HeporCloud frameworks [minimum values are "best"].

Policy	Energy (E)	Performance (P)	*ERP*
			E × P
No migrations	2662	44.89	**119.5**
HPC	2221	55.09	**122.4**
W_1	2189	70.18	**153.6**
W_2	2299	86.46	**198.8**
W_3	2311	151.4	**349.9**
CIAO	2076	55.09	**114.4**
HeporCloud	1782	45.22	**80.6**

In order to determine how datacenter configuration affects IaaS energy consumption and workload performance, we run the experiments from Sec. 5.1 ten times with three different initial orders for hosts: (a) INC – increasing order based on E_f; (b) NR – no order i.e. random; and (c) DEC – decreasing order based of E_f. For every host, the E_f is computed through dividing the host peak power consumption (P_{max} – energy consumed at 100% utilisation level) by the total number of available slots (cores, vCPUs or GCEUs). For instance, if we have four hosts (H_1, H_2, H_3 and H_4) having $E_f^{H_1} = 4$, $E_f^{H_2} = 1$, $E_f^{H_3} = 2$ and $E_f^{H_4} = 3$; where larger E_f denotes higher energy efficiency. Then the corresponding orders would be: INC – [H_2, H_3, H_4, H_1]; NR – [H_1, H_2, H_3, H_4]; and DEC – [H_1, H_4, H_3, H_2]. We can also use other metrics such as *ERP* to compute order for hosts. Various orders of hosts create various levels of energy and performance efficiencies. Fig. 11 and Fig. 12 demonstrate variations in workload performance (runtime) and energy consumption for various kinds of workloads and orders of hosts. Note that, ordering here is discussed in terms of logical addressing which means through allocation policies and is, therefore, not a physical shift. As a future work, this might be transformed to: (a) physically shifting hosts within a particular rack or across several racks; and/or (b) putting hosts in different racks. These empirical evaluations demonstrate that the physical order of hosts is a major concern for service providers that could affect IaaS energy consumption and performance, therefore, cost of running workloads in large-scale datacenters. In order to show how energy consumption models for VM—container would affect the IaaS energy efficiency, we repeatedly performed the above experiments with four different models – two models from Sec. 4.1 and two from (Alzamil and Djemame, 2016) were selected. Moreover, we changed the workload utilisation levels to stress and release the CPU demand to see its impact. Surprisingly, our evaluation suggests that there are very small variations in IaaS energy; due to the fact that the entire energy relates to physical hosts and their benchmarked values at various utilisation levels. Since, every VM—container energy is computed as part of the SPECpower benchmarks at particular utilisation level; therefore, the impact is lower. These variations were observed larger when workload types or utilisation levels are changed, as shown in Table 15.

For the above experiments, we changed the runtime prediction model in order to see the impact of the accuracy of a prediction approach on IaaS energy efficiency (E) and workload performance (P). As shown in Table 16, the accuracy of the prediction approach may significantly affect the scheduling decisions, particularly, the total number of migrations. However, the energy efficiency and performance of the workloads are not affected, severely. For example, in the case of 'migrate all' approach, number of migrations are reduced; however, in the case of 'HeporCloud' the total number of migrations were increased. Albeit, the accuracy of the averaging-based approach is higher than the distribution-based approach; but, the impacts over energy consumption and performance is not non-trivial. However, these findings are not inline with previous work (Tsafrir et al., 2007); and this might be related or specific to our abstraction and simulation of a real system. Furthermore, these impacts would vary to the use of prediction e.g. (i) whether runtimes are predicted in the initial scheduling phase or in the optimisation phase; and/or (ii) whether workload types, utilisation patterns, or runtimes are predicted. In the future, we will do further investigation on how various machine or deep learning based prediction approaches will affect the resource management of hybrid IaaS clouds.

5.5. Costs savings

The total electricity bill, user monetary costs and costs savings (in US dollars - $) are described in Table 17. For these analyses, we assume a PUE[11] i.e. power usage effectiveness of 1.10 and energy price of 0.88$ per KWh[12] that mimic a Google datacenter located in the Oklahoma state, USA. The PUE is a simple ratio of power consumed in computation and power used in other parts of the datacenters e.g. cooling, offices etc. A minimum value for PUE of 1 means that all the power consumed in datacenter is for computational purposes. Moreover, we assume that the users bills are computed at 0.0017$ per second.[13] In practice, various providers offer various pricing for their instances. The providers could save up to 35.9% energy costs using the proposed HeporCloud technique rather than using simple allocation and migration heuristic approaches. Moreover, the users costs could also be reduced up to approximately 7.91%–14.2% as compared to simple management policies. We believe, these savings would translate to a million dollars savings per year for big service providers, such as AWS, Google, that operate and manage thousands of machines in their computational clusters.

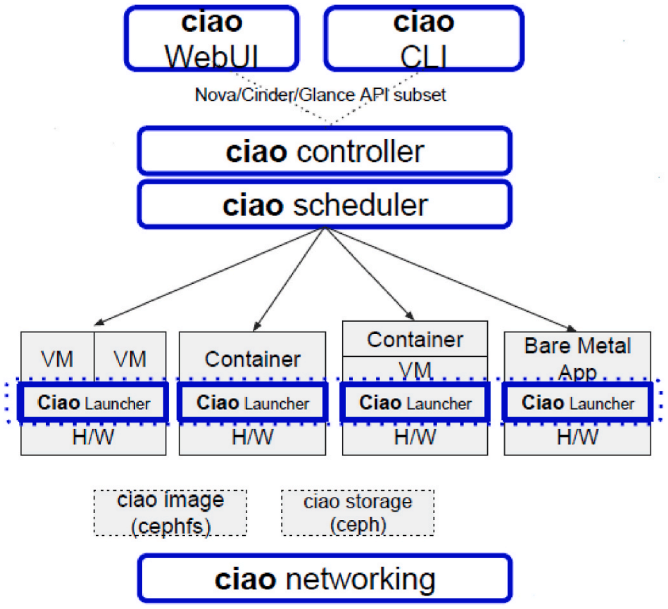


Fig. 13. The CIAO architecture.[16].

5.6. CIAO vs. HeporCloud framework

The results in terms of energy consumption and workload performance are shown in Table 18 and Table 19, respectively. The HeporCloud orchestrator is approximately 9.4%–14.2% more energy and 1.7%–17.9% more performance efficient than the Intel's CIAO framework when several combinations of workloads and migration policies are taken into account. We also observed a similar behaviour of the CIAO framework; due to the existing trade-off between energy consumption and workload performance (Khan et al., 2019a). Using several reasonable assumptions, our evaluation suggests that approximately 13.57% energy could be saved at cost of 3.88% loss in performance. These savings seem to be in line with possible savings using the HeporCloud framework i.e. ~30.47% energy savings at cost of 2.14% loss in performance. When migrations are considered, the HeporCloud framework could be as high as 14.2% cost-effective than the CIAO architecture. Similarly, if costly migrations are avoided, then approximately 15.27% performance improvements are achievable. The performance improvements for various workloads are shown in Table 19.

These results demonstrate the efficiency of our proposed HeporCloud framework. Due to the existing trade-off between energy consumption

[11] https://www.google.co.uk/about/datacenters/efficiency/.

[12] https://www.eia.gov/electricity/monthly/.

[13] https://aws.amazon.com/ec2/pricing/.

Table 21
Various methods and their pros and cons with respect to other approaches [for example Bare-metal offer the lowest utilisation levels and the highest energy consumption compared to VMs, Container, and Containers—VMs and so on].

Work	Approach	Pros and cons					
		resource utilisation	workload performance	energy consumption	resource contention	opportunities	
						placement	consolidation
Technology	Bare-metal	lowest	optimal	highest	low	low	–
	VMs	moderate	low	low	high	high	high
	Containers	high	high	low	very high	very high	very high
	Containers—VMs	very high	moderate	high	very high	highest	highest
	Hybrid	–	–	–	–	–	–
Scheduler	Centralised	very high	high	lowest	low	high	lowest
	Hierarchical	high	low	low	moderate	moderate	high
	Distributed	low	low	high	high	low	very high
Migrations	No migrations	lowest	high	high	moderate	–	–
	Inter-platform	high	low	moderate	high	–	–
	Intra-platform	low	high	high	low	–	–
	Inter + Intra	–	–	–	–	–	–

and workload performance, it is impossible to improve both; however, the best approach would improve one factor with a slight decrease in another factor. The trade-off can be observed accurately using a single metric i.e. Energy Runtime Product (*ERP*)[14] – the product of energy consumption (P) and performance (R); where runtime is the inverse of performance (as lower workload run-times mean good performance and vice versa). A detail discussion of the *ERP* metric is given in Sec. 2. Table 20 describes *ERP* values for several migration policies along with the CIAO and HeporCloud frameworks. In the case of "no migration" technique, albeit performance is optimal (minimal); however, energy consumption is maximum. For uncontrolled migrations (HPC, W_1, W_2, W_3), variations in energy consumption and workload performance can be seen; along with significant overlaps, which represent that migrations could be more expensive than the "no migration" approach. In such scenarios, workload-specific resource allocation (for example HeporCloud) could offer energy savings and performance guarantees. The proposed HeporCloud approach has an *ERP* of **80.6** which is far better than the CIAO's *ERP* of **114.4**.

5.7. Computational complexity

The computational complexity of the HeporCloudScheduler is based on the time taken during the prediction module. If T_{pred} is the amount of time needed to predict and categorize a particular workload w, then the worst case computational complexity is given by Eq. (13):

$$\mathcal{O}\left(T_{pred} + \alpha.\frac{\beta}{platforms}\right) \tag{13}$$

where α, β and *platforms* denote the number of VMs—container, hosts and platforms, respectively. In other words, $\frac{\beta}{platforms}$ denotes the number of hosts in a platform. The best case $\mathcal{O}(T_{pred})$ occurs when the VM—container is placed in the first attempt. Furthermore, T_{pred} is dependent on the prediction method i.e. linear regression, support vectors, boosted tree, amount of data and available resources of the NAS server.

The computational time of Alg. 2 is largely dependent on the time needed to perform datacenter' (or a particular platform) state optimisation i.e. *optimise*(s_i). The best case computational time for the optimisation module can be described as $\mathcal{O}(\alpha.\beta)$. Furthermore, the worst case complexity of Alg. 3 can be computed from the number of migratable entities $\alpha' \in \alpha$ and hosts identified as under-loaded and/or over-loaded $\beta' \in \beta$, given by $\mathcal{O}(\alpha'.\beta')$. The computation cost incurred in computing the power savings for all migratable entities can be described as $\sum_{i=0}^{\alpha'-1}\gamma_i$, where γ can be assumed as constant time. Therefore, the total computational time for Alg. 3 and Alg. 1 is given by Eq. (14).

$$\mathcal{O}(\alpha'.\beta') + \sum_{i=0}^{\alpha'-1}\gamma_i \tag{14}$$

Integrating Eq. (14) into the cost incurred by Alg. 1, the total worst case computational complexity of the HeportCloudOrchestrator module can be computed using Eq. (15). Note that, complexity of the prediction approach should be also taken into account.

$$\mathcal{O}(\alpha.\beta)^2 + \gamma \tag{15}$$

6. Related work

The Cloud Integrated Advanced Orchestrator (CIAO), proposed by Intel. Runs VMs and containers within the same system. Since, there are no such details or publications being offered by the Intel other than the CIAO's project code which is freely available online; thus, certain details are still unknown. For instance, how: (i) explicit placement decisions are made by the scheduler for various workloads; (ii) migration of VMs and containers takes place; and (iii) no relevant publication showing evaluation of the workload performance and IaaS energy efficiency. The architecture proposed by Intel's CIAO platform is shown in Fig. 13. It is composed of: (a) a controller to implement how users communicate with the system; (b) a scheduler which tells how user workloads are allocated for resources; and (c) a launcher which is responsible for how each compute server's processing stats are collected. At each system, controller and scheduler are executed individually whereas the launcher executes on all compute servers within the IaaS cluster. Moreover, the well-known FF heuristic is used for allocating resources instead of best fit (BF) method. It is also evident that Intel's researchers are biased for the implementation simplicity and job dispatching speed rather than absolute optimality. Still, BF or other workload-specific scheduler and heuristic methods may be more beneficial for placement in the future, particularly, for unknown workloads. Moreover, the scheduling choice focuses, primarily, on memory, disk and CPU availability of various compute nodes relative to the start of requested workload[15].

The CIAO architecture does not support live migrations; however, through check-pointing i.e. stopping and restating, a particular instance could be migrated from one host to another host, seamlessly. Moreover, several instances running on a particular host can also be migrated,

[14] http://epubs.surrey.ac.uk/841959/.

[15] https://ciao-project.github.io/.

[16] http://events17.linuxfoundation.org/sites/events/files/slides/Linuxcon%20NA%202016.pdf.

concurrently, through a single command. This might be useful if a host needs to be temporarily removed from the cluster for maintenance and up-gradation purposes. The CIAO project documentation also suggests that the scheduler currently implements a trivial approach which prefers not using the Most-Recently-Used (MRU) compute host for workload placement. Although, this could be inexpensive and may lead to enough spread of new workloads across the cluster; however, this may not be essentially energy efficient. Unfortunately, we are not aware of any work, in the literature, which addresses the cluster's energy consumption and performance of various workloads in a similar hybrid orchestration platform; when various approaches to resource scheduling and migration are taken into account.[16] The study of bare metal, VMs and individual containers in terms of workload performance, network usage, disk I/O operations, memory use and boot time is discussed by Kominos et al. (2017). It is still felt that there is no exploration of resource consolidation in the context of resource management, IaaS energy efficiency, performance of workloads, and applications as well as resource heterogeneities. Lubomski et al. (2016) have compared various virtualisation techniques such as VMs, containers; and suggested non-significant performance loss when containers run within VMs. However, Mavridis et al. (Mavridis and Karatza, 2017) suggest that the performance loss could be significant. The pros and cons of various existing works, regarding schedulers, and methods, are summarised in Table 21. This will help readers to quickly identify gaps for further consideration.

The HeporCloud framework uses the FillUp allocation policy to place user's workloads onto available resources in a workload-specific way (Zakarya, 2018b). It means that most energy and performance efficient hosts are utilised first. However, the CIAO uses a FF policy to place the workload onto resources without any prioritisation with respect to energy and/or performance. Moreover, the HeporCloud framework selects the best candidate for migration using the consolidation with migration cost recovery (Cmcr) (Zakarya, 2018b) and consolidation with migration performance, energy costs recovery (Cper) (Khan et al., 2019b) policies. Furthermore, HeporCloud bias towards migrating containers first instead of VMs; as containers are lightweight that would finish their migration quickly. However, CIAO implements a simple migration policy which selects the top most instance, that could be either a container or a VM. Moreover, migration occurs when the node' utilisation decreases certain threshold e.g. 20%. We assume that over utilisation will not happen; as the HeporCloud policies will not place workloads on nodes exceeding certain threshold e.g. 90%. Several studies (Kominos et al., 2017),(Tay and GauravPavan, 2017) discuss the performance efficiency of containers, VMs, and bare-metal, individually; however, hybrid resource management with respect to energy efficiency is relatively ignored, and this is rarely addressed with the notable exception of (Vaucher, 2015). In (Vaucher, 2015), the authors suggest that bare-metal hardware might offer the highest levels of performance at the lowest energy cost. Nevertheless, the evaluated outcomes take single application (job) into account; and, thus, there outcomes are not ensured as optimal or near to optimal, particularly, if a dynamic system is considered.

Moreover, the authors in (Sharma et al., 2016), (Tay and GauravPavan, 2017) have discussed VMs, containers and virtualised containers (containers—VMs), however, bare-metal and energy efficiency are not investigated. Tay et al. (Tay and GauravPavan, 2017) suggests exploring different technologies including VMs and containers, in the context of workload consolidation and migration policies. Sharma et al. (2016) evaluated that containers running inside VMs offer performance benefits; and neighbouring containers inside a VM could be trusted, as well. Felter et al. (2015) explored resource management of VMs (KVM) and containers (Docker), and associated the obtainable performance of certain workloads and applications regarding bare-metal hardware. Their examination and evaluation shows that for certain types of workloads container's and VM's performance overlaps. Moreover, the authors reject the finding that "IaaS should be implemented on VMs and PaaS on containers" – since there is no technical cause. Unfortunately, service migrations are not taken into account. The investigation of performance for bare metal, VMs, containers and virtual/nested containers, for running interactive game-based simulations, is discussed in Mondesire et al. (2019). The authors suggest that the container's performance is comparable with the performance of bare metal hardware; and mingling VMs with containers gives performance advantages over the use of containers and VMs, individually. Additionally, there is no such investigation over scheduling and placement for hybrid platforms in terms of energy usage and performance impacts due to consolidation of workloads and resource heterogeneities.

Several researchers noted the under-utilisation nature of VMs in public IaaS clouds (Tchana et al., 2015), (Tchana et al., 2016); and anticipated a solution known as software consolidation. It accommodates multiple applications, at runtime, over the same VMs to reduce the number of utilised VMs. Furthermore, VM consolidation can be combined with this technique to reduce the number of servers needed to run particular workloads. Software consolidation might decrease: (i) the energy usage; (ii) the total number of VMs needed; and (iii) users' costs, both in private and public IaaS clouds. The investigation also prevails that ~40% energy could be saved within the authors' private IaaS cloud. Additionally, user's budgetary cost is saved by about 40.5% for AWS EC2 public IaaS cloud. The said work is relatively close to HeporCloud; however, the algorithms presented in (Tchana et al., 2015), (Tchana et al., 2016) are simple in deciding how running applications are to be: hosted in VMs, containers, container—VM, and bare-metal; or migrated in terms of energy consumption and possible performance impacts over the workloads. In (Khan et al., 2019a), we neither considered nested containers nor inter-platform and intra-platforms migrations. Furthermore, performance of workloads across four platforms was not benchmarked. Similarly, we modelled performance of application consolidation that runs either in VMs or containers in (Khan et al., 2020); however, resource management in hybrid clouds were not taken into account. Moreover, centralised schedulers were not investigated. Besides these differences, workload predictions were also ignored in (Khan et al., 2019a), (Khan et al., 2020). In (Mavridis and Karatza, 2019), the authors have studied the combination of virtualisation and containerisation technologies through running containers on top of VMs. Their aim is to improve containers' key problem i.e. isolation (since containers share the same kernel) and, to incorporate benefits of containers into VMs. Therefore, containers were run on bare-metal, and inside VMs (using KVM and Xen); and the authors suggest its possibility subject to trivial performance overhead. Besides high resource utilisation, their evaluation suggests that running containers on KVM is more energy-performance efficient than running them on Xen. Unfortunately, their work has ignored resource management aspects such as scheduling, consolidating workloads, in hybrid clouds. Moreover, migrations are not examined.

Besides VM placement (Zakarya, 2017), container placement is also largely studied in the literature. For example, in (Hu et al., 2020) authors have presented ECSched, a graph-based scheduler to handle concurrent container requests in heterogeneous clusters subject to multi-resource constraints. The ECSched scheduler assumes a batch of requests, at the same time, to find a condensed placement. The authors suggest that ECSched produces good results, in terms of low completion times, and improved resource utilisation. However, the proposed scheduler is impractical for online problems when tasks do not arrive in batches; or, the online problem should be converted to an offline problem in order to fetch requests at the same time. Moreover, VMs, nested containers, hybrid platforms and their energy efficiency is not explored. Similarly, KEIDS (Kaur et al., 2020) incorporate a container scheduler/management system on top of Kubernetes to account for interference and energy consumption of IoT applications in distributed clouds (operated by different energy sources). The KEIDS scheduler is approximately 14.42%, and 31.83%, better that the FCFS scheduler in terms of improved energy utilisation, and minimal interference. In (Lv

Table 22
Summary of the related work [the scheduler architecture applies to works in which multiple sandboxing technologies are being used; and the scheduler plus management policy are used to evaluate the platforms for various metrics].

Parameters		Related Work								HeporCloud
		Sharma et al. (2016)	Mondesire et al. (2019)	Mavridis and Karatza (2017)	Tay and GauravPavan (2017)	Tchana et al. (2016)	Khan et al. (2019a)	Chae et al. (2019)	Mavridis and Karatza (2019)	CIAO
Platforms	VMs	×	×	×	×	×	×		×	×
	Containers	×	×	×	×	×	×		×	×
	Containers-VMs	×	×	×	×	×	×	×	×	×
	Bare-metal	×	×	×	×	×	×	×	×	×
Metrics	Energy consumption	×				×	×	×		×
	Workload performance		×	×	×		×	×	×	×
Scheduler architecture	Single						×			×
	Distributed		×	×	×	×	×	×	×	×
Management policy	Allocation	×	×	×				×	×	×
	Migration	×			×	×				×
Runtime	Prediction									×
	Probabilistic	×					×			

et al., 2019), a communication-aware worst fit decreasing heuristic algorithm is proposed for container placement. Moreover, a container reassignment strategy is presented to balance the containers distribution across various servers and optimise application performance and throughput. Albeit, renewable, and distributed clusters are taken into account; however, except containers, other scenarios such as VMs, virtualised containers, hybrid platforms and migrations are not explored.

ProCon (Fu et al., 2019) schedules containers subject to: (i) the instant resource utilisation of hosts; and (ii) estimation of future resource usage. The ProCon scheduler balances the resource contentions across the cluster and reduces task runtimes through monitoring their execution progress. ProCon reduces completion time by up to 53.3% and improves performance by 23.0% against the default scheduler available in Kubernetes. Various approaches to virtualisation (full - KVM, para - Xen and OS level - Docker) are discussed in (Chae et al., 2019). The performance of KVM and Docker was compared in three different ways: (a) the CPU and memory usage of the host, (b) Idleness of CPU, memory usage and I/O performance through migrating a large file, and (c) performance of the Web server through JMeter. These comparisons show that Docker is faster than KVM. The authors have only compared KVM and Docker which were configured on a single host. Moreover, the authors demonstrate that placement algorithms affect the performance of VMs and containers. The PIVOT task scheduler (Jiang et al., 2020) supports cross-cloud, cross-region execution of data-intensive applications while hiding the complexity of the underlying heterogeneous systems and respecting cost and performance requirements of the containerised application. PIVOT has two capabilities: (i) an application scheduler schedules various tasks of an application; and (ii) the global scheduler has a task queue that put tasks for final dispatching and placement onto hosts. Furthermore, the scheduling problem is modelled as a vector bin-packing problem and solved effectively using greedy approximation algorithms such as first fit heuristic.

In (Rocha et al., 2019), a task-oriented and energy-aware scheduler "HEATS" is suggested for containerised workloads that allows customers to trade performance vs. energy needs and exploits the resource heterogeneity. In the first phase (probing), HEATS learns the energy and performance characteristics of hosts. In the second phase (monitoring), it monitors tasks execution on hosts. In the third phase (scheduling), HEATS speculatively migrates workload across various hosts to match customers' demands. Their evaluation suggests that, depending on the workload type, HEATS can save up to 8.5% energy while marginally affect the overall task runtimes (by at most 7%). Renewables along with appropriate resource allocation and consolidation approaches can mitigate the energy related issues in cloud environment. In (Kumar et al., 2018), containerised workloads are placed on those clusters which has enough renewable energy. Moreover, a container consolidation scheme is designed to minimise the energy consumption of hosts. In (Hu et al., 2019), authors have discussed bin-packing, approximate and meta-heuristic algorithms. Moreover, a container scheduling approach is suggested to account for various objectives such as load-balancing and multi-resource guarantee. Other works have also suggested meta-heuristic based approaches to solve the workload placement problem (Adhikari and Narayana Srirama, 2019), (Kaur et al., 2019), (Gill et al., 2019). However (Hu et al., 2019), suggests that meta-heuristic approaches can take hours to reach a solution, and, are not suitable for container scheduling. In the literature (Zakarya, 2018b), (Kominos et al., 2017), (Mondesire et al., 2019), (Sharma et al., 2016), (Tchana et al., 2016), (Nadgowda et al., 2017), various allocation and consolidation with migration methods have been investigated for VMs and containers, but, individually. Nevertheless, we are not aware of any work that investigates the impact of energy savings and workload performance degradation when migrations of VMs, containers, containers over VMs and bare-metal applications are taken into account, at the same time. Similarly, no study describes scheduling for hybrid platforms in terms of using a centralised scheduler and/or distributed schedulers for various resource platforms. Moreover, inter-platform and

intra-platforms migrations, which are possible in hybrid datacenters, are also not discussed anywhere else. The summary of the comparison between our proposed HeporCloud and other closely related works is given in Table 22. We believe the information in Table 22 would also help the readers to quickly identify gaps for further research and investigation.

7. Conclusions and future work

In this paper, we proposed a framework HeporCloud and, an integrated, workload-aware single resource scheduler and orchestrator for hybrid cloud platforms. The proposed resource manager is able to allocate and predict effective workloads placement and migration decisions. Using reasonable assumptions, our empirical evaluation suggests that HeporCloud can schedule and consolidate various kinds of workloads energy, performance and, therefore, cost efficiently. Our investigation suggests that: (i) using a, centralised, single scheduler is more energy and performance efficient than using individual schedulers in hybrid platforms; (ii) under certain configurations, it might be more energy and performance efficient not to migrate workloads; and (iii) inter-platform migrations are more energy efficient than intra-platforms migrations, however, performance of the workload varies significantly. Moreover, containers are more energy, performance efficient than VMs; and energy efficient than bare-metal hardware due to high level of resource utilisation. Furthermore, for certain kinds of workloads, virtualised containers may be as bad as good, as compared to VMs and containers.

We identified few issues in the HeporCloud framework that needs to be addressed. First, when more and more VMs—containers interact with the HeporCloudScheduler and/or the HeporCloudOrchestrator then, due to delay in communication or network congestion, the system response might become slow. Secondly, the HeporCloudStat is a burden on the cluster node that maintains and calculates statistical information regarding resource consumption, in addition, to its necessary task of job execution. Furthermore, it needs to update its information with the NAS server, periodically. Further research is needed to account for these important issues. A distributed-type implementation for the HeporCloudStat module would be an alternative solution. In that case, the HeporCloudStat can be installed as an agent on every cluster host and which are essentially connected to a master HeporCloudStat agent that runs on a dedicated powerful server. In the future, we will consider multi-objective minimisation and meta-heuristics to solve the placement problem. We are keen to validate the proposed HeporCloud framework on a real cloud test-bed; through importing it in the OpenStack (Kominos et al., 2017). More technically, our aim would be to advise an architecture or, more specifically, a hybrid resource manager to the OpenStack community in order to integrate the Ironic, Nova, Magnum and kolla services operating over raw bare-metal (hardware), VMs, system containers and virtualised containers i.e. containers run inside VMs, respectively. Moreover, migrations could affect the workload performance severely, particularly, if a single VM—container is migrated several times (repeatable migrations) during a consolidation round (Khan et al., 2019b). In the future, along with more accurate energy consumption and migration models (Dayarathna et al., 2015), we would add a migration control mechanism (Khan et al., 2019c), to HeporCloud framework, to avoid repeatable migrations.

CRediT authorship contribution statement

Ayaz Ali Khan: Conceptualization, Methodology, Investigation. **Muhammad Zakarya:** Supervision, Writing - original draft, Software. **Izaz Ur Rahman:** Visualization, Data curation. **Rahim Khan:** Supervision, Visualization, Formal analysis. **Rajkumar Buyya:** Validation, Project administration, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported in part by the Abdul Wali Khan University Mardan (AWKUM), Pakistan and in part by the Higher Education Commission (HEC), Pakistan. This work is partially supported by an Australian Research Council (ARC) Discovery Project.

References

Adhikari, Mainak, Narayana Srirama, Satish, 2019. Multi-objective accelerated particle swarm optimization with a container-based scheduling for internet-of-things in cloud environment. J. Netw. Comput. Appl. 137, 35–61.

Alzamil, Ibrahim, Djemame, Karim, 2016. Energy prediction for cloud workload patterns. In: International Conference on the Economics of Grids, Clouds, Systems, and Services. Springer, pp. 160–174.

Amaral, Marcelo, Polo, Jorda, Carrera, David, Mohomed, Iqbal, Unuvar, Merve, Steinder, Malgorzata, 2015. Performance evaluation of microservices architectures using containers. In: 2015 IEEE 14th International Symposium on Network Computing and Applications. IEEE, pp. 27–34.

Calheiros, Rodrigo N., Ranjan, Rajiv, Beloglazov, Anton, De Rose, Csar AF., Buyya, Rajkumar, 2011. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software Pract. Ex. 41 (1), 23–50.

Callau-Zori, Mar, Samoila, Lavinia, Orgerie, Anne-Ccile, Pierre, Guillaume, 2018. An experiment-driven energy consumption model for virtual machine management systems. Sustain. Comput.: Inform. Syst. 18, 163–174.

Chae, MinSu, Lee, HwaMin, Lee, Kiyeol, 2019. A performance comparison of linux containers and virtual machines using docker and kvm. Cluster Comput. 22 (1), 1765–1775.

Colmant, Maxime, Kurpicz, Mascha, Pascal, Felber, Huertas, Loc, Rouvoy, Romain, Sobe, Anita, 2015. Process-level power estimation in vm-based systems. In: Proceedings of the Tenth European Conference on Computer Systems. ACM, p. 14.

Cortez, Eli, Bonde, Anand, Muzio, Alexandre, Russinovich, Mark, Fontoura, Marcus, Bianchini, Ricardo, 2017. Resource central: understanding and predicting workloads for improved resource management in large cloud platforms. In: Proceedings of the 26th Symposium on Operating Systems Principles. ACM, pp. 153–167.

Dabbagh, Mehiar, Hamdaoui, Bechir, Guizani, Mohsen, Rayes, Ammar, 2014. Release-time aware vm placement. In: Globecom Workshops (GC Wkshps), 2014. IEEE, pp. 122–126.

Dabbagh, Mehiar, Hamdaoui, Bechir, Guizani, Mohsen, Rayes, Ammar, 2016. An energy-efficient vm prediction and migration framework for overcommitted clouds. IEEE Trans. Cloud Comput.

Dayarathna, Miyuru, Wen, Yonggang, Fan, Rui, 2015. Data center energy consumption modeling: a survey. IEEE Commun. Surv. Tutor. 18 (1), 732–794.

Fan, Xiaobo, Weber, Wolf-Dietrich, Andre Barroso, Luiz, 2007. Power provisioning for a warehouse-sized computer. In: ACM SIGARCH Computer Architecture News, ume 35. ACM, pp. 13–23.

Felter, Wes, Ferreira, Alexandre, Rajamony, Ram, Rubio, Juan, 2015. An updated performance comparison of virtual machines and linux containers. In: Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on. IEEE, pp. 171–172.

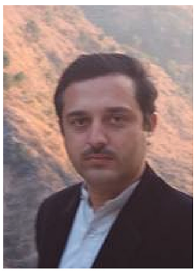
Ferreto, Tiago C., Netto, Marco A.S., Calheiros, Rodrigo N., De Rose, Csar A.F., 2011. Server consolidation with migration control for virtualized data centers. Future Generat. Comput. Syst. 27 (8), 1027–1034.

Fu, Yuqi, Zhang, Shaolun, Terrero, Jose, Mao, Ying, Liu, Guangya, Li, Sheng, Tao, Dingwen, 2019. Progress-based container scheduling for short-lived applications in a kubernetes cluster. In: 2019 IEEE International Conference on Big Data (Big Data). IEEE, pp. 278–287.

Gandhi, Anshul, Gupta, Varun, Harchol-Balter, Mor, Kozuch, Michael, 2010. Energy-efficient Dynamic Capacity Provisioning in Server Farms. School of Computer Science, Carnegie Mellon University. Tech. Rep. CMU-CS-10-108.

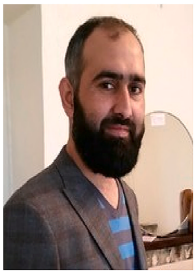
George, Amvrosiadis, Park, Jun Woo, Ganger, Gregory R., Gibson, Garth A., Baseman, Elisabeth, DeBardeleben, Nathan, 2017. Bigger, Longer, Fewer: what Do Cluster Jobs Look like outside Google? Technical report, Technical Report CMU-PDL-17-104 Carnegie Mellon Univedrsity Parallel Data.

Gill, Sukhpal Singh, Garraghan, Peter, Stankovski, Vlado, Casale, Giuliano, Thulasiram, Ruppa K., Ghosh, Soumya K., Ramamohanarao, Kotagiri, Buyya, Rajkumar, 2019. Holistic resource management for sustainable and reliable cloud computing: an innovative solution to global challenge. J. Syst. Software 155, 104–129.
Gupta, Varun, 2011. Stochastic Models and Analysis for Resource Management in Server Farms. PhD thesis. Intel Corporation.
Hu, Yang, De Laat, Cees, Zhao, Zhiming, et al., 2019. Multi-objective container deployment on heterogeneous clusters. In: Proc. 19th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.(CCGRID), pp. 592–599.
Hu, Yang, Zhou, Huan, de Laat, Cees, Zhao, Zhiming, 2020. Concurrent container scheduling on heterogeneous clusters with multi-resource constraints. Future Generat. Comput. Syst. 102, 562–573.
Jiang, Han-Peng, Chen, Wei-Mei, 2018. Self-adaptive resource allocation for energy-aware virtual machine placement in dynamic computing cloud. J. Netw. Comput. Appl. 120, 119–129.
Jiang, Fan, Ferriter, Kyle, Castillo, Claris, 2020. A cloud-agnostic framework to enable cost-aware scheduling of applications in a multi-cloud environment. In: NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, April 20-24, 2020. IEEE, pp. 1–9.
Kaur, Kuljeet, Garg, Sahil, Aujla, Gagangeet Singh, Kumar, Neeraj, Zomaya, Albert, 2019. A multi-objective optimization scheme for job scheduling in sustainable cloud data centers. IEEE Trans. Cloud Comput.
Kaur, K., Garg, S., Kaddoum, G., Ahmed, S.H., Keids, M. Atiquzzaman, 2020. Kubernetes-based energy and interference driven scheduler for industrial iot in edge-cloud ecosystem. IEEE Internet of Things J. 7 (5), 4228–4237.
Khan, Ayaz Ali, Zakarya, Muhammad, Khan, Rahim, 2019a. H^2 a hybrid heterogeneity aware resource orchestrator for cloud platforms. IEEE Syst. J. 13 (4), 3873–3876.
Khan, Ayaz Ali, Zakarya, Muhammad, Buyya, Rajkumar, Khan, Rahim, Khan, Mukhtaj, Rana, Omer, 2019b. An energy and performance aware consolidation technique for containerized datacenters. IEEE Trans. Cloud Comput.
Khan, Ayaz Ali, Zakarya, Muhammad, Khan, Rahim, 2019c. Energy-aware dynamic resource management in elastic cloud datacenters. Simulat. Model. Pract. Theor. 92, 82–99.
Khan, Ayaz Ali, Zakarya, Muhammad, Khan, Rahim, Izaz Ur Rahman, Khan, Mukhtaj, et al., 2020. An energy, performance efficient resource consolidation scheme for heterogeneous cloud datacenters. J. Netw. Comput. Appl. 150, 102497.
Kominos, Charalampos Gavriil, Seyvet, Nicolas, Vandikas, Konstantinos, 2017. Bare-metal, virtual machines and containers in openstack. In: Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on. IEEE, pp. 36–43.
Kotikalapudi, Sai Venkat Naresh, February 2017. Comparing Live Migration between Linux Containers and Kernel Virtual Machine: Investigation Study in Terms of Parameters.
Kozhirbayev, Zhanibek, Sinnott, Richard O., 2017. A performance comparison of container-based technologies for the cloud. Future Generat. Comput. Syst. 68, 175–182.
Kumar, Neeraj, Singh Aujla, Gagangeet, Garg, Sahil, Kaur, Kuljeet, Ranjan, Rajiv, Garg, Saurabh Kumar, 2018. Renewable energy-based multi-indexed job classification and container management scheme for sustainability of cloud data centers. IEEE Trans. Industr. Inform. 15 (5), 2947–2957.
Lebre, Adrien, Pastor, Jonathan, Simonet, Anthony, Sdholt, Mario, 2019. Putting the next 500 vm placement algorithms to the acid test: the infrastructure provider viewpoint. IEEE Trans. Parallel Distr. Syst. 30 (1), 204–217.
Liu, Haikun, Jin, Hai, Xu, Cheng-Zhong, Liao, Xiaofei, 2011. Performance and energy modeling for live migration of virtual machines. Cluster Comput. 249–264.
Lubomski, Pawe, Kalinowski, Andrzej, Krawczyk, Henryk, 2016. Multi-level virtualization and its impact on system performance in cloud computing. In: International Conference on Computer Networks. Springer, pp. 247–259.
Lv, Liang, Zhang, Yuchao, Li, Yusen, Xu, Ke, Wang, Dan, Wang, Wendong, Li, Minghui, Cao, Xuan, Liang, Qingqing, 2019. Communication-aware container placement and reassignment in large-scale internet data centers. IEEE J. Sel. Area. Commun. 37 (3), 540–555.
Masdari, Mohammad, Khoshnevis, Afsane, 2019. A survey and classification of the workload forecasting methods in cloud computing. Cluster Comput. 1–26.
Massie, Matthew L., Chun, Brent N., Culler, David E., 2004. The ganglia distributed monitoring system: design, implementation, and experience. Parallel Comput. 30 (7), 817–840.
Mavridis, Ilias, Karatza, Helen, 2017. Performance and overhead study of containers running on top of virtual machines. In: 2017 IEEE 19th Conference on Business Informatics (CBI), ume 2. IEEE, pp. 32–38.
Mavridis, Ilias, Karatza, Helen, 2019. Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing. Future Generat. Comput. Syst. 94, 674–696.
Medel, Vctor, Rana, Omer, Baares, Jos ngel, Arronategui, Unai, 2016. Modelling performance & resource management in kubernetes. In: 2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC). IEEE, pp. 257–262.
Mondesire, Sean C., Angelopoulou, Anastasia, Sirigampola, Shehan, Goldiez, Brian, 2019. Combining virtualization and containerization to support interactive games and simulations on the cloud. Simulat. Model. Pract. Theor. 93, 233–244.
Morabito, Roberto, Kjllman, Jimmy, Komu, Miika, 2015. Hypervisors vs. lightweight virtualization: a performance comparison. In: 2015 IEEE International Conference on Cloud Engineering. IEEE, pp. 386–393.
Nadgowda, Shripad, Suneja, Sahil, Bila, Nilton, Isci, Canturk, 2017. Voyager: complete container state migration. In: Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on. IEEE, pp. 2137–2142.
O'Loughlin, John, 2018. A Workload-specific Performance Brokerage for Infrastructure Clouds. PhD thesis. University of Surrey.
O'Loughlin, John, Gillam, Lee, 2014. Performance evaluation for cost-efficient public infrastructure cloud use. In: International Conference on Grid Economics and Business Models. Springer, pp. 133–145.
Piraghaj, Sareh Fotuhi, Dastjerdi, Amir Vahid, Calheiros, Rodrigo N., Buyya, Rajkumar, 2017. Containercloudsim: an environment for modeling and simulation of containers in cloud data centers. Software Pract. Ex. 47 (4), 505–521.
Reiss, Charles, Wilkes, John, Hellerstein, Joseph L., 2011. Google Cluster-Usage Traces: Format+ Schema. Google Inc., Mountain View, CA, USA. Technical Report.
Reiss, Charles, Tumanov, Alexey, Ganger, Gregory R., Katz, Randy H., Kozuch, Michael a., 2012. Heterogeneity and dynamicity of clouds at scale. In: Proceedings of the Third ACM Symposium on Cloud Computing - SoCC 12, pp. 1–13.
Rocha, Isabelly, Gttel, Christian, Felber, Pascal, Pasin, Marcelo, Rouvoy, Romain, Schiavoni, Valerio, 2019. Heats: heterogeneity-and energy-aware task-based scheduling. In: 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). IEEE, pp. 400–405.
Ruan, Bowen, Huang, Hang, Wu, Song, Jin, Hai, 2016. A performance study of containers in cloud environment. In: Asia-Pacific Services Computing Conference. Springer, pp. 343–356.
Shai, Ohad, Shmueli, Edi, Feitelson, Dror G., 2013. Heuristics for resource matching in intel's compute farm. In: Workshop on Job Scheduling Strategies for Parallel Processing. Springer, pp. 116–135.
Sharma, Prateek, Chaufournier, Lucas, J Shenoy, Prashant, Tay, Y.C., 2016. Containers and virtual machines at scale: a comparative study. In: Middleware, p. 1.
Shehabi, A., Smith, S.J., Horner, N., Azevedo, I., Brown, R., Koomey, J., Masanet, E., Sartor, D., Herrlin, M., Lintner, W., 2016. United states Data Center Energy Usage Report, vol. 4. Lawrence Berkeley National Laboratory, Berkeley, California. LBNL-1005775.
Smith, Warren, Foster, Ian, Taylor, Valerie, 2004. Predicting application run times with historical information. J. Parallel Distr. Comput. 64 (9), 1007–1016.
Tay, Y.C., Gaurav, Kumar, Pavan, Karkun, 2017. A performance comparison of containers and virtual machines in workload migration context. In: Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on. IEEE, pp. 61–66.
Tchana, Alain, De Palma, Noel, Safieddine, Ibrahim, Hagimont, Daniel, Diot, Bruno, Vuillerme, Nicolas, 2015. Software consolidation as an efficient energy and cost saving solution for a saas/paas cloud model. In: European Conference on Parallel Processing. Springer, pp. 305–316.
Tchana, Alain, De Palma, Noel, Safieddine, Ibrahim, Hagimont, Daniel, 2016. Software consolidation as an efficient energy and cost saving solution. Future Generat. Comput. Syst. 58, 1–12.
technical white paper, H.P., 2016. Linux Container Performance on Hpe Proliant Servers: Understanding Performance Differences between Containers and Virtual Machines.
Tsafrir, Dan, Etsion, Yoav, Feitelson, Dror G., 2007. Backfilling using system-generated predictions rather than user runtime estimates. IEEE Trans. Parallel Distr. Syst. 18 (6), 789–803.
Tumanov, Alexey, Jiang, Angela, Park, Jun Woo, Kozuch, Michael A., Ganger, Gregory R., 2016. Jamaisvu: Robust Scheduling with Auto-Estimated Job Runtimes. Technical report, Technical Report CMU-PDL-16-104. Carnegie Mellon University.
Vaucher, Sbastien, 2015. Comparing Virtual Machines and Linux Containers.
Xu, Fei, Liu, Fangming, Jin, Hai, 2016. Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud. IEEE Trans. Comput. 65 (8), 2470–2483.
Zakarya, Muhammad, 2017. Energy and Performance Aware Resource Management in Heterogeneous Cloud Datacenters. PhD thesis. University of Surrey.
Zakarya, Muhammad, 2018a. Energy, performance and cost efficient datacenters: a survey. Renew. Sustain. Energy Rev. 94, 363–385.
Zakarya, Muhammad, 2018b. An extended energy-aware cost recovery approach for virtual machine migration. IEEE Syst. J. 13 (2), 1466–1477.
Zakarya, Muhammad, Gillam, Lee, 2016. An energy aware cost recovery approach for virtual machine migration. In: International Conference on the Economics of Grids, Clouds, Systems, and Services. Springer, pp. 175–190.
Zakarya, Muhammad, Gillam, Lee, 2017. Energy efficient computing, clusters, grids and clouds: a taxonomy and survey. Sustain. Comput.: Inform. Syst. 14, 13–33.
Zakarya, Muhammad, Gillam, Lee, 2019. Managing energy, performance and cost in large scale heterogeneous datacenters using migrations. Future Generat. Comput. Syst. 93, 529–547.



Ayaz Ali Khan is currently a PhD student in the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. He completed his M.Phil (MS) in computer science from COMSATS Institute of Information Technology (CIIT), Islamabad, Pakistan. His area of research includes energy-aware and performance-efficient scheduling, resource allocation, placement and management, at datacenter level. Moreover, he has enough knowledge of distributed systems, optimisation, game theory and computer programming. His work has been published in reputed internal journals.

Muhammad Zakarya received the PhD degree in computer science from the University of Surrey, Guildford, U.K. He is currently a Lecturer with the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. His research interests include cloud computing, mobile edge clouds, performance, energy efficiency, algorithms, and resource management. He has deep understanding of the theoretical computer science and data analysis. Furthermore, he also owns deep understanding of various statistical techniques which are, largely, used in applied research. His research has been appeared in several international journals of repute and conferences.



Izaz Ur Rahman received the PhD degree in computer science from the Department of Electronic and Computer Engineering, Brunel University, UK. He is currently an Assistant Professor with the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. His research interest includes power systems, optimisation algorithms, internet of things and artificial intelligence.

Rahim Khan received the PhD degree in computer science from the Ghulam Ishaq Khan Institute (GIKI), Swabi, Pakistan. He is currently an Assistant Professor with the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. His research interest includes the Wireless Sensor Networks (WSNs) deployment, Internet of Thing (IoT), routing protocols, outliers' detection, techniques for congestion control, Decision Support System (DSS), vehicular ad-hoc networks, data analysis and similarity measures.

Rajkumar Buyya is a Fellow of IEEE and Life Member of ACM, Professor of Computer Science and Software Engineering, Future Fellow of the Australian Research Council, and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, at the University of Melbourne, Australia. His research interests include cloud, grid, distributed, and parallel computing. R. Buyya has a PhD in computer science from Monash University. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the University, commercialising its innovations in Cloud Computing. He has co-founded five IEEE/ACM international conferences: CCGrid, Cluster, Grid, e-Science, and UCC (Utility and Cloud Computing) and served as the Chair of their inaugural meetings. He has presented over 400 invited talks (keynotes, tutorials, and seminars) on his vision on IT Futures and advanced computing technologies at international conferences and institutions in Asia, Australia, Europe, North America, and South America. For further information on Dr. Buyya, please visit: http://www.buyya.com.