An Energy and Performance Aware Consolidation Technique for Containerized Datacenters

Ayaz Ali Khan, Muhammad Zakarya, Rajkumar Buyya, *Fellow, IEEE*, Rahim Khan, Mukhtaj Khan, and Omer Rana

Abstract—Cloud datacenters have become a backbone for today's business and economy, which are the fastest-growing electricity consumers, globally. Numerous studies suggest that ∼30% of the US datacenters are comatose and the others are grossly less-utilized, which make it possible to save energy through resource consolidation techniques. However, consolidation comprises migrations that are expensive in terms of energy consumption and performance degradation, which is mostly not accounted for in many existing models, and, possibly, it could be more energy and performance efficient not to consolidate. In this paper, we investigate how migration decisions should be taken so that the migration cost is recovered, as only when migration cost has been recovered and performance is guaranteed, will energy start to be saved. We demonstrate through several experiments, using the Google workload data for 12,583 hosts and approximately one million tasks that belong to three different kinds of workload, how different allocation policies, combined with various migration approaches, will impact on datacenter's energy and performance efficiencies. Using several plausible assumptions for containerised datacenter set-up, we suggest, that a combination of the proposed energy-performance-aware allocation (Epc-Fu) and migration (Cper) techniques, and migrating relatively long-running containers only, offers for ideal energy and performance efficiencies.

Index Terms—Clouds, datacenters, containers, energy efficiency, performance, consolidation with migrations

1 Introduction

National energy supply complications, rising fuel costs, and global warming, altogether bring the need for energy efficient computing into sharp focus. Reduction in coal-based power plants, particularly, in the UK, offering a projected energy safety margin of only 0.1 percent in 2017, and the closure of nuclear power plants in France and Germany, carry the very real danger of power outages and load shedding in near future [1]. If we assume comparable consumption rates to the US of approximately 1.8 percent of overall energy consumption [2], a 6 percent growth in datacenter energy efficiency might represent two times increase in such an energy safety margin. Furthermore, [2] also suggests that, due to migration from internal systems to the cloud, datacenter energy consumption will remain constant until 2020. These kinds of problems can be addressed, in part, through approaches such as efficient resource management—resource allocation, scheduling and consolidation [3]. Resource management techniques are dependent on the technology (virtualisation and/or container-based virtualisation—containerization) that the service providers uses to provide resources to customers. Virtualisation brings the notion of a Virtual Machine (VM) and containerization has replaced the VM with a container; both run on virtualised hardware (servers).

Virtualisation is extensively used in cloud infrastructures, particularly, the state-of-the-art in Infrastructure as a Service (IaaS) is largely familiar with the notion of VMs. Cloud service providers such as Amazon EC2, Microsoft Azure and Google make VMs available to users and also run applications (workload/services) inside VMs. Many Software as a Service (SaaS) and Platform as a Service (PaaS) providers are built on top of IaaS with all their applications running inside VMs. Containerization presents an alternative to VMs in the cloud; which are lightweight as they share a single OS kernel. Moreover, containers could be deployed very quickly compared to VMs. Interest in containerization is increasing rapidly, as evidenced by the viral implementation of the Docker engine. In reality, even before the rise of Docker[1] engine, Google [4] has been using containers to run applications for several years [5]. From a customer point of view, the question whether to deploy applications on VMs or containers is application dependant. For example, if it is desirable

- *A.A. Khan, M. Zakarya, R. Khan, and M. Khan are with the Department of Computer Science, Abdul Wali Khan University, Mardan, Khyber Pakhtunkhwa 23200, Pakistan. E-mail: {ayazali, mohd.zakarya, rahimkhan, mukhtaj.khan}@awkum.edu.pk.*
- *R. Buyya and O. Rana are with the University of Melbourne, Parkville, VIC 3010, Australia, and also with the Department of Computer Science, Cardiff University, Cardiff CF10 3AT, United Kingdom. E-mail: rbuyya@unimelb.edu.au, ranaof@cardiff.ac.uk.*

Manuscript received 14 Oct. 2018; revised 6 Apr. 2019; accepted 1 June 2019. Date of publication 5 June 2019; date of current version 6 Dec. 2021. (Corresponding author: Muhammad Zakarya.) Recommended for acceptance by C. De Rose. Digital Object Identifier no. 10.1109/TCC.2019.2920914

1. https://www.docker.com/

to run large number of applications on a single host (virtualised), or the applications are portable, then deploying them on containers is more suitable than VMs.

Once applications are launched, customers are billed based on the runtime of their applications and resource demand (quantity).[2] However, resources are largely under-utilised in datacenters; and significant energy could be saved through resource management techniques [6], [7]. Felter et al. [8] have investigated resource management in containers, VMs and compared the performance of different kinds of workloads (applications) in both platforms to that of natively running the applications on bare-metal (hardware). However, resource management i.e., consolidation of VMs and containers both involves migrations that can be expensive in terms of additional energy consumption, performance loss (hence cost), and this is largely not accounted for in many published models—it can be more energy and performance (cost) efficient not to consolidate [6], [9].

In this paper, we investigate how migration decisions can be made such that the energy and performance costs involved with the migration are recoverable, after which energy is saved and performance is maintained (improved or at least not degraded). We explore the impact on energy and performance efficiencies of allocation heuristics such as Round Robin (RR), Random (R), Best Resource Selection (BRS) [10], Minimum Power Difference (MPD) [11], First Fit (FF), FillUp (Fu) and Epc-Fu when combined with different approaches to migration (no migration (No)—migrate all (All) or Dynamic Consolidation (Dc)—migrate relatively long-running VMs (Cmcr) [6]—migrate relatively long-running containers (Cper)). Key to this exploration is the recovery of costs (in terms of energy and performance) incurred by a migration. This exploration is conducted through extensive simulations that use the Google workload traces for 12,583 hosts and approximately a million tasks (three different application types) [12] in combination with CloudSim [13].

The rest of the paper is organized as follows. Major contributions of the work presented in this paper are stated in Section 2. In Section 3, we discuss server consolidation as a bin-packing problem. In Section 4, we explain container migration, its energy overhead, and how to measure a containerised host energy and performance efficiencies. In Section 5, we propose an Energy and Performance Cost Recovery Consolidation (Cper) technique that avoids migrating containers which would not be able to: (i) recover the energy used in migration; and/or (ii) performs worse on the target host after the migration. Section 6 models resource heterogeneities in Google cluster from a large and real workload dataset perfective. We validate Cper using real workload traces from Google cluster in Section 7 and show that Cper can reduce the migration energy overhead with reduced numbers of migrations, increased or at least maintained performance (expected level), and that the majority of migrated containers now recover their migration cost and continue to perform better, save energy and therefore cost. We offer an overview of the related work in Section 8. Finally, Section 9 concludes the paper with future research directions.

2. https://aws.amazon.com/ec2/pricing/

2 Contributions

Following are the major contributions of our research.

- a new algorithm for container allocation that accounts for host efficiencies (energy and performance), where efficient hosts are utilized first, which decreases datacenter's total energy consumption and increases reliability;
- a mathematical model to estimate the energy consumption of a container, running inside a VM (containerised platform);
- a novel host energy efficiency and a migration approach that migrates containers only if migration cost (in terms of energy and performance) can be recovered, and ideally then save energy and perform better;
- extensive simulations on a real dataset have been performed to demonstrate that migrating relatively long-running containers are more energy and performance efficient; and
- evaluation of the proposed techniques using three different dynamic workload types—where dynamic means varying demand for resources.

3 Problem Description

Resource allocation and consolidation with migration can be considered as multidimensional bin-packing problems where bins are hosts/VMs and items are containers, for container placement. The objective is to minimize the number of hosts needed to accommodate a set of containers [14]. Rich literature has formulated the placement and consolidation problems as bin-packing optimisation problems [14], [15]. However, when hosts are heterogeneous, then the problem becomes more complex due to the number of resource types (CPU, memory) needed to accommodate VMs or containers. In that case, the allocation problem can still be formulated as multi-type bin-packing issue; and the objective would be to minimize the sum of bin costs. Such NP-complete problems are typically solved using Linear Programming (LP) or heuristics. Dynamic consolidation is typically suggested being an improvement on doing nothing, allowing: (a) to switch off the underutilized host if the accommodated containers can be relocated to other hosts; (b) to withdraw hosts from an overloaded state if the sum of accommodated containers becomes larger than its capacity; and (c) migrate the workload to a better performing host to increase service reliability [16]. Besides the trade-off involved between migrating containers among heterogeneous hosts and decreasing the number of hosts to accommodate containers, live container migration can be completed without needing downtime, and ideally without impacting performance (and, specifically, SLAs).

In production clouds (e.g., Amazon EC2 and Google) users are charged for the amount of provisioned resources and the duration of the service (i.e., job execution time (that absolutely depends on the performance level of the container) → container runtime). As a consequence of increased container runtime that may occur either: (i) due to the poor performance levels of the hosts; or (ii) resource contention (i.e., co-located containers compete for similar resources), the energy

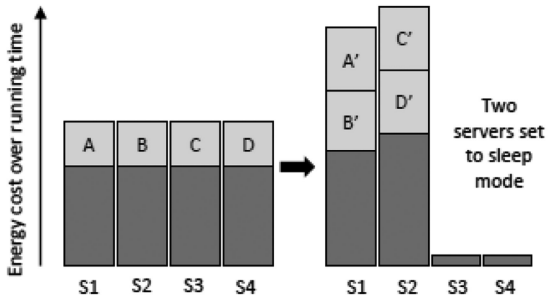


Fig. 1. Energy and performance (runtime) costs change due to consolidation [left - 4 containers are placed on individual servers/VMs; right - containers are consolidated on two servers/VMs]. For each vertical bar (server), the darker area denotes server's idle energy use; the lighted area labelled with container name denotes dynamic energy use [17]. Both shaded areas, as a whole, denote the server's total energy consumption for a specific runtime [bars height].

consumption of the host itself will rise (*albeit the host may be more energy efficient than the others*), as shown in Fig. 1. Greenberg et al. [18] suggest that performance directly impacts providers' revenue. Therefore, it is essential to ensure that a particular container meets its expected level of performance (i.e., achieves a service level), and if not, it might be migrated to a host where it may perform either better or, at least, not reducing its current performance (on the source host) if it is beneficial to do so. Furthermore, the migration decision will be necessary if the recently allocated host is performing better and, also, is more energy efficient than the previous one. Considering these diverse opportunities (i.e., increased runtime, decreased performance and increased energy consumption), our work is aimed at developing a consolidation model to: (i) predict the energy and performance of a container; (ii) derive a correlation among the predicted quantities to decide migration; and (iii) finally migrate the container to achieve better results in terms of the energy consumption and performance. Note that, it is realistic to recoup the performance cost for certain applications such as HPC.

Furthermore, if each container can recover its migration cost first, and then lasts to run until its execution on the energy and performance efficient host, then its migration is more effective for achieving its expected level of performance, energy savings and, therefore, in cost reduction. Dynamic consolidation can be assumed as a multi-objective optimization problem to minimize the amount of energy consumed and increase performance level through avoiding unnecessarily or costly migrations. We describe the problem as CPER—Energy and Performance Cost Recovery in Consolidation, further explained in Section 5.1, and address it through exploration of the impact of container runtimes. However, in an on-demand public cloud platforms, container runtimes are usually unknown, therefore, we consider the container past runtime R_{past} in order to decide its migration.

We can express the container migration as a multi-objective optimization problem which comprises two nominal cost types namely energy consumption cost (ECC) and container performance cost (CPC). The objectives of our bi-objective optimization problem is to minimize both energy (E) and runtime (R) [as R is inversely proportional to performance]. Mathematically, this can be expressed as Eq. (1):

$$f = \begin{cases} \boldsymbol{min}(E) \;\; where\, E = \sum_{i=1}^{hosts} E_i \\ \boldsymbol{min}(R) \;\; where \;\; R = \sum_{j=1}^{containers} Runtime_j. \end{cases} \quad (1)$$

The constraints are: (a) at a time, each container (VM) can only be mapped to a single VM (host); and (b) the total number of containers on a particular host cannot exceed the host's overall resource capacity [14].

Multi-objective optimisation problems could be solved easily if a single objective could be defined. For example, Gupta et al. [19] suggested and used ERP (Energy Response-time Product) in order to capture the existing trade-off between energy and performance, therefore, cost; which is a largely acceptable and suitable metric to capture similar trade-offs [20]. ERP minimization can be seen as maximization of the "performance-per-watt" ratio—where performance is defined as the inverse of response time (mean). In this paper, we determine performance through runtime R that can be considered comparable to response time (based on time factor). Hence, we revise the given name of this metric to the Energy Runtime Product (ERP). The ERP is given by Eq. (2):

$$ERP = E \times R. \quad (2)$$

We assume both E and R are of a comparable magnitude. However, in more complex scenarios, if one dominates the other, then ERP can be expressed as $ERP = \alpha.E \times \beta.R$ (where α and β are the domination factors for E and R respectively) [19], [20]. Theoretically, the objective of the above bi-objective optimization problem is to minimize and evaluate the ERP behaviour for various resource scheduling and consolidation policies, as given by Eq. (3):

$$\boldsymbol{min}(ERP). \quad (3)$$

From the implementation point of view, as predicting R is difficult, hence we use R' i.e., the previous runtime of container, in order to calculate the ERP. Various techniques can be used to predict the execution time of the container, but this is not within the scope of current work.

4 Background

From a cloud service perspective, applications are often encapsulated into pre-configured platforms such as the well-known hypervisor-based VMs, or the relatively new technique, containers; for easy distribution [21]. Virtualisation technology (VMs) create and run several isolated guest OSs on top of a host OS; and containerisation technology (containers) share a single host OS. Moreover, both technologies differ as VMs completely emulate the OS kernel and hardware, whereas containers directly share the hardware and kernel with their host machines. Therefore, containers have lower virtualisation overhead and occupy much less resources than VMs, but are less adaptable; a Linux container (LXC) cannot run on a Windows server. Containers are becoming a compute instance of choice in large-scale cloud platforms [22]. A recent survey of 576 IT leaders [23], which benchmarks the container adoption, conducted by Diamanti,[3]

3. https://diamanti.com/

suggests that approximately 44 percent of IT leaders (respondent in the survey) plan to replace VMs with containers. Moreover, approximately 71 percent of respondents have deployed containers on VMs including both public and private clouds. However, this is not favoured by several organisations as suggested in Diamanti's white paper [24]—instead they suggest running containers on bare-metal. Obviously, this does not suggest VMs disappearance overnight, however, a sudden change in applications deployment can be seen, clearly. VMs and containers both could help to significantly increase the resource utilisation levels in datacenters using consolidation techniques—VMs or containers migration [10].

Like VM migration, container migration may also occur for several reasons within a datacenter, such as host maintenance, user mobility, electricity price and load balancing. Migrations can still be beneficial where renewable energy is used to decrease datacenter energy costs and CO_2 footprint [25]. Furthermore, if a certain workload performs the worst on a specific host due to co-location or resource heterogeneity, then migrating it to another host could be performance and hence cost efficient. During container live migration, the running container is moved from one host to another. This means migrating data on disk, depending on the underlying method to storage, and memory pages. This leads to two kinds of migration: (i) shared file system, where a container image is run from shared storage such as NFS, GlusterFS, and only memory is copied; and (ii) over-Ethernet migrations, where a container image is run from a local drive (disk), and both memory and disk are copied. Since the container image may itself be large, this latter form of migration may take rather longer. A third kind of migration, particularly for container, is record and replay [26]. Certain tools, such as CRIU and P.Haul [as explained in Section 8], are widely used to checkpoint and restore the container on a target host.

In practice, container migration is not for energy savings only. Other factors such as electricity price, renewable energy generation, user mobility and new system such as cloudlets, edge/fog computing also trigger the need for container migration [9]. If we perform live migrations in order to achieve energy and performance efficiencies, then for the duration of migration, there will be an energy cost, as well as loss in performance level, in the additional container running on the source host. The energy cost will depend on the host's power profile $\mathcal{P}$ while the performance cost will be subject to several factors such as available bandwidth. We assume $\mathcal{P}$ a linear function of host's utilisation level—the more it is utilised, the more energy it will consume [27], [28]. The relationship between power consumption and CPU utilisation can be described as given in Eq. (4)

$$\mathcal{P}(u) = \mathcal{P}_{idle} + (\mathcal{P}_{max} - \mathcal{P}_{idle}) \times u, \tag{4}$$

where $\mathcal{P}(u)$ is the estimated power consumption at utilisation level u, $\mathcal{P}_{idle}$ is the static power consumed when the host is 0 percent utilised, and $\mathcal{P}_{max}$ is the power consumed when the host is 100 percent utilised. The part $(\mathcal{P}_{max} - \mathcal{P}_{idle}) \times u$ is known as dynamic power consumption, and is treated as a linear function of u. This simplified model estimates the power consumption of a non-virtualised host with less than 5 percent error, however, it needs modification to account for virtualisation and containerization [28]. Moreover, we assume $\sim$10% performance degradation due to migration [11]. In the first part of this section, we extend this power model to address containerised hosts; in the second part we discuss measuring the migration energy and performance costs.

4.1 Comparing Hosts Efficiencies

In this paper, we investigate migration cost recovery in terms of both energy and performance, which is possible only if two terms are satisfied: (a) the container is migrated to a more energy and performance efficient target host; and (b) after the migration, the container runs on the target host for a sufficient length of time. In this section, we discuss measuring the energy and performance efficiencies of hosts to address conditions (a) and (b).

4.1.1 Energy Efficiency

In non-virtualised systems, if one host performs better and consumes less power than another to execute a particular application/workload, then the former one is more energy and performance efficient than the latter. However, this is possible that certain workloads may run less efficiently on the former host, therefore, efficiency should be addressed across a range of workloads [29]. The well-known TOP500[4] supercomputer benchmarking organization uses the model in Eq. (5) to compare hosts efficiencies, however, performance is not taken into account. Furthermore, in [10], using empirical evaluation of the proposed metric, the authors suggest its inability and non-suitability to decide energy efficient host in a virtualised or containerised platform

$$E_{f_{host}} = \frac{C_{host}}{\mathcal{P}_{max}}. \tag{5}$$

In the above equation, E_f denotes host's energy efficiency and C is the host's capacity in terms of total number of instructions that it can execute in one second (MIPS). In containerised environments, several containers on multiple VMs can be running different workloads on a single host, therefore several factors must be considered in order to compare energy and performance efficiency of the host. To keep it simple, we characterise, first, distribution of the physical host resources to several containers (not VMs) and, hence, the overall energy consumption of a containerised host is denoted by

$$\mathcal{P}_{host} = \mathcal{P}_{idle} + \sum_{i=1}^{n} \mathcal{P}_i^{container}, \tag{6}$$

where $\mathcal{P}$ is the host's total power consumption [measured in Watts (W)], n is the total number of accommodated containers on the host, $\mathcal{P}_{idle}$ is the static or idle power consumption of the host and $\mathcal{P}_i^{container}$ is the dynamic power consumption of the ith container that could be estimated, roughly [28], using the above linear power model, as described in Section 4 [Eq. (4)]

4. http://www.top500.org

$$\mathcal{P}_{container} = \mathcal{W}_{container} \times \mathcal{P}_{dynamic}, \tag{7}$$

where $\mathcal{W}_{container}$ is the fraction of host's resources (e.g., CPU, memory) that are allocated to a particular container. This permits us to streamline concerns by assuming every container as equal to a real physical host; further, like VM sizes, container sizes may also be divided equally, in an IaaS cloud, by the amount of allocated CPU cores (hyper-threaded) out of m cores on a particular host, or by allocation of a specific amount of memory. To keep it simple, we only consider the number of cores (hyper-threaded)

$$\mathcal{W}_{container} = \frac{cores_{container}}{m}. \tag{8}$$

To consider complete heterogeneity and divide the host's energy consumption amongst accommodated containers (n) more fairly, the m (number of cores or vCPUs) is given by Eq. (9)

$$m = \sum_{i=0}^{n} core_{container_i}. \tag{9}$$

Including static power, the total power consumed by a single container will be given by:

$$\mathcal{P}_{container} = \left(\frac{\mathcal{P}_{idle}}{n}\right) + \mathcal{W}_{container} \times (\mathcal{P}_{max} - \mathcal{P}_{idle}) \times u, \tag{10}$$

where n is the total number of containers running on host, u is the utilisation level of $container$. Hence, host's efficiency can be related to the number of containers that are running on it and, more specifically, to their individual efficiencies. Note that, to fairly divide host's $\mathcal{P}_{idle}$ amongst all running containers, $\frac{1}{n}$ should be replaced by $\mathcal{W}_{container}$, as given by Eq. (11), which is the exact amount of resources (cores) allocated to the container

$$\mathcal{P}_{container} = (\mathcal{W}_{container}.\mathcal{P}_{idle}) + \mathcal{W}_{container}.(\mathcal{P}_{max} - \mathcal{P}_{idle}) \times u \tag{11}$$

$$\mathcal{P}_{container} = \mathcal{W}_{container} \times (\mathcal{P}_{idle} + \big((\mathcal{P}_{max} - \mathcal{P}_{idle}) \times u\big)). \tag{12}$$

Eq. (12) describes a simplified model to estimate the energy consumption of a container running on bare-metal (i.e., directly on hardware). However, if containers are running inside VMs, then Eq. (12) can be modified, as Eq. (13), to estimate the energy consumption of a single VM

$$\mathcal{P}_{vm} = \mathcal{W}_{vm} \times (\mathcal{P}_{idle} + \big((\mathcal{P}_{max} - \mathcal{P}_{idle}) \times u\big)). \tag{13}$$

Therefore, the energy consumption of a container running inside a VM can be estimated as given in Eq. (14):

$$\mathcal{P}_{container_{vm}} = \mathcal{W}_{container_{vm}}.\Big[\mathcal{P}_{idle_{vm}} + \big((\mathcal{P}_{max_{vm}} - \mathcal{P}_{idle_{vm}}).u\big)\Big], \tag{14}$$

where $\mathcal{P}_{idle_{vm}}$ and $\mathcal{P}_{max_{vm}}$ are the idle and maximum power consumption of the vm, when it is 0 and 100 percent utilized, respectively. Similarly, $\mathcal{W}_{container_{vm}}$ is the fraction of vm resources allocated to the container and u is the container utilization. The total energy consumption of a containerized host with n number of containers and m VMs is given by Eq. (15):

$$\mathcal{P}_{host} = \mathcal{P}_{idle} + \sum_{i=1}^{m} \mathcal{P}_i^{vm} \left(\sum_{j=1}^{n} \mathcal{P}_j^{container}\right). \tag{15}$$

In our simulations, we use the above model to estimate the container energy consumption before it is being migrated to other hosts. Furthermore, using the above model, the energy consumed by a particular host to run a single container will be at maximum, due to host's $\mathcal{P}_{idle}$. Similarly, the more number of containers run on the host, the more energy efficient each container and, therefore, host will be. Similar to VM density [29], we also define the notion of container density, which is used somewhere else, to address both: (i) the number of containers running on a host; and (ii) the maximum number that possibly will be run whereas evading resource starvation; we merge these to understand container density as the current fraction of the maximum for a host [10].

Limitations. The above model has two shortcomings. (I) Cloud datacenters run heterogeneous applications with diverse resource usage, including not only the CPU, but also the memory, the disk, and the network. Those subsystems apart from the processor have been also reported to make up a noticeable part of the total power consumption depending on the workload [30]. In order to avoid models that are specific for CPU-intensive applications, the impact on the power consumption of the rest of subsystems should be also considered. (II) Moreover, when containers run on Virtual Machines (VMs), the VM CPU utilization is used to characterize the VM workload and to correlate the processor usage with the power consumption. However, the utilization is not the best indicator of the processor usage regarding its correlation with energy consumption, because applications with the same utilization can have different processor energy consumption depending on what instructions they are executing, as reported by Kansal et al. [31].

4.1.2 Performance Efficiency

Theoretically, a host would perform better if it can execute more number of instructions per unit time than the other. For example, in the notion of MIPS or GHz, higher ratios are better than the lowers. However, empirical evaluation in [29] demonstrates that various platforms might perform differently for similar kinds of application workloads, as shown in Fig. 2. Largely, this is possible due to co-location when containers allocated on same host compete for similar resources. It is difficult to predict, in a real cloud, how a containerised application would perform on a specific host. However, using a real dataset from Google's cluster [12], we can derive hosts performance parameters (such as mean μ and standard deviation σ of variations in execution times). For a particular workload, if $\mu_{H_1} < \mu_{H_2}$ then H_1 is more performance efficient that H_2.

As shown in Fig. 2, AMD performs worst for Bzip2 while E5430 performs better. Therefore, migrating the Bzip2 application from E5430 to AMD is not performance efficient. Moreover, these kinds of migrations could also be less energy efficient even if the target host is more energy efficient than the

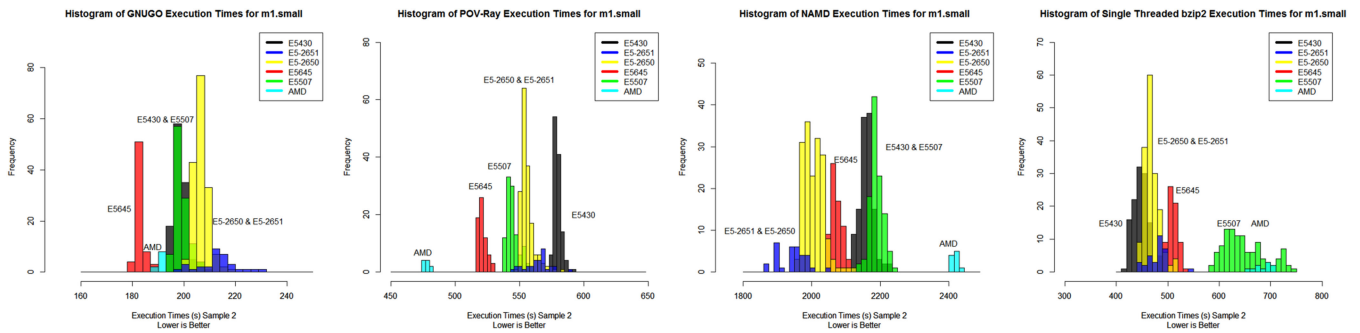


Fig. 2. Performance variations for GNUGO, POV-RAY, NAMD and BZIP2 benchmark workloads on *m1.small* instances [AMD performs 'best' for GNUGO but 'worst' for NAMD, E5-2650 & E5-2651 perform 'best' for NAMD but are 'worst' for GNUGO] [29].

source. Note that, performance and energy are related to each other. For example, if a containerised application is migrated to a lower performance but more energy efficient host, the increase in runtime can still decrease the energy efficiency. Therefore, the performance costs in terms of increased runtime and energy consumption must be taken into account while consolidating the workload onto fewer heterogeneous hosts.

4.1.3 Dilemma

Suppose there are n containers allocated to host H_1 and m containers allocated to host H_2. The performance parameters of H_1 and H_2 are μ_{H_1}, σ_{H_1} and μ_{H_2}, σ_{H_2} respectively. Each container is utilizing 100 percent of its proportional resources allocated. The per container power consumption of each container on H_1 and H_2 are $\mathcal{P}^{H_1}_{container_{i=1:n}}$ and $\mathcal{P}^{H_2}_{container_{j=1:m}}$ respectively according to above equation. The total power consumption of each host H_1 and H_2 is given by $\mathcal{P}_{H_1} = \sum_{i=1}^{n} \mathcal{P}_i^{container}$ and $\mathcal{P}_{H_2} = \sum_{j=1}^{m} \mathcal{P}_j^{container}$ respectively. For a container $container_k$ selected for migration from H_1, with sufficient space to allocate on H_2, and provided that $\mathcal{P}^{H_1}_{container_k} > \mathcal{P}^{H_2}_{container_k}$ and $\mu_{H_2} < \mu_{H_1}$, then H_2 is more power and performance efficient than H_1 with a factor of E_f given by:

$$E_f = \left(\frac{\mathcal{P}_{H_1} \times \mu_{H_1}}{\mathcal{P}_{H_2} \times \mu_{H_2}} \right). \tag{16}$$

E_f could be used to compare two hosts for efficiencies: (i) if $E_f = 1$ then both hosts are similar; (ii) if $E_f < 1$ then H_2 is less energy and performance efficient than H_1; and (iii) if $E_f > 1$ then H_2 is more energy and performance efficient than H_1. In Eq. (16), $\mathcal{P}_{H_1} \times \mu_{H_1}$ can be seen as an equivalent metric to Energy Response time Product (ERP) proposed in [10]; as response time is the reciprocal of runtime. ERP represents the overall energy and performance efficiency of a host; a lower value denotes larger efficiency and vice versa.

Note that, the above dilemma does not take into account the difference between price of electricity at the source and target hosts. When the price of electricity in target host is considerably less than the price of electricity in the source host, it may be economical to migrate containers; even if the source is more energy efficient than the target host. However, performance would again play a role of trade-off with electricity price. The current trends of service providers towards using renewables which may operate intermittently, and therefore necessitate falling back to the energy grid, also implies a need for consolidation policies to be able to effectively switch between the available energy sources. In that case, the electricity price $\mathcal{M}$ can be considered as a third objective in the above formulation [as described in Section 3]; and a multiple of E_f, given by $\mathcal{P}_H \times \mu_H \times \mathcal{M}_H$, for both hosts. However, this would be more beneficial if migrations are performed among datacenters located in different geographical areas [intra-datacenters]. This work focus on migrations inside a single datacenter [inter-datacenters]; where electricity price would be same.

4.2 The Migration Energy and Performance Model

The memory of a container can be copied to the target host in two ways: (i) pre-copy; and (ii) post-copy. In the former case, memory is transferred repetitively first and processor state afterwards, whereas in the latter case, memory is transferred after the processor state is sent to the target host [26]. In both cases, an extra container is created on the target host and is synchronized progressively. After the synchronization, the container is started on the target and its copy is destroyed on the source. Therefore, for the duration of migration, the migration costs roughly double the resources. Moreover, the migration effort could be wasted, if the container terminates during the migration process, or before this resource cost is recouped back. Moreover, [11] suggests approximately 10 percent performance degradation due to migration. The 10 percent performance degradation is analogues to ~10% increase in runtime, and thus increase in energy consumption.

Several works including [8], [11], [14] discuss container consolidation but appear to ignore the cost that is due to the migration energy and performance overheads, and with the notable exception of [32] this is rarely addressed. The migration cost is dictated by the cost of the most expensive container (at source host) running for the duration of migration, plus any associated performance and network costs during migration. The performance cost is suggested to be approximately 10 percent loss, and subsequently increase in runtime [11], [33], [34]. The migration cost also includes some marginal migration cost $\mathcal{MC}_{host}$, in case, the migration procedure needs changes in the power states [on/off] of either one or both hosts [27]. Furthermore, we assume that containers are running inside VMs, so $\mathcal{MC}_{host}$ also include the cost of starting a new VM or terminating a VM if needed during migration. For heterogeneous hosts, the duration (time) needed for a migration is

$$\mathcal{T}_{mig} = \left(\frac{\mathcal{C}_{mem} + \mathcal{C}_{disk}}{\beta}\right), \tag{17}$$

where $\mathcal{T}_{mig}$ is reliant on the memory size $\mathcal{C}_{mem}$ of the container, ephemeral disk image $\mathcal{C}_{disk}$ (for block-live migration) and the network bandwidth β which is available to transmit the migration data $[\mathcal{C}_{mem} + \mathcal{C}_{disk}]$. For (shared-disk) live migration, $\mathcal{C}_{disk}$ is 0 and $\mathcal{C}_{mem}$ can be calculated using the current memory size of the container (active state) and the amount of dirty pages which are copied, continuously, in several rounds n, throughout the migration process/duration $\mathcal{T}_{mig}$. In case of an idle container, the amount of dirty pages (memory) is 0 and, therefore, the network traffic (i.e., data to transfer) is only equal to $\mathcal{C}_{mem}$ (measured in MB), otherwise

$$\mathcal{C}_{mem} = \sum_{i=0}^{n} \mathcal{C}_i \tag{18}$$

$$\mathcal{C}_i = \mathcal{D} \times T_{i-1}, \tag{19}$$

where i represents the round number, $\mathcal{D}$ is the rate at which the container's memory pages are being dirtied (measured in MB/s), T is the duration of the round (in seconds) and $\mathcal{C}$ denotes the size of the dirty pages (measured in MB). We also assume that the container load is dynamic and changes according to the Google's cluster dataset [12], therefore $\mathcal{D}$ is not constant. As suggested in [6]. if $\mathcal{D}$ varies for a container, it may be more realistic to simulate using: (i) a distribution around a mean; or (ii) with reference to historical usage data. Therefore, we estimate the container migratable memory $\mathcal{C}_{mem}$ and migration time $\mathcal{T}_{mig}$ using the container historical usage data—gathered for one hour (at 5 minute intervals). If we assume 10 percent loss in container performance due to migration i.e., $(10\% \times \mathcal{T}_{mig})$, this will become part of the: (i) migration energy overhead; and (ii) remaining runtime (r_2') of the container on target host [as described in Section 5.1]. The migration energy overhead $Cost_{mig}$ is given by:

$$Cost_{mig} = \mathcal{T}_{mig} \times (\mathcal{P}_{source} + \mathcal{P}_{net}) + \left(\frac{\mathcal{T}_{mig}}{10} \times \mathcal{P}_{target}\right) + \mathcal{MC}_{host}, \tag{20}$$

where $\mathcal{MC}_{host} \subset [14.3, 60.0, 110.0]$ denotes the marginal migration cost (in Joules) needed to change the states of hosts [ON-STANDBY|ON-HIBERNATE|ON-OFF] respectively, $\mathcal{P}_{net}$ is the network power consumption, and $\mathcal{P}_{source}$ is the cost of the most expensive container running at source host [10]. Note that the marginal cost $\mathcal{MC}_{host}$ relates to the server' states transitions delays of a typical compute server and the corresponding amount of energy consumed [27], [35]. However, various works report variations in transition delays for various servers and operating systems [36], [37]. Further, due to the complexity involved in measuring the $\mathcal{P}_{net}$, we assume this as zero; because networks are not within scope of our current work. Moreover, this also includes the cost of starting a new VM or switching off a VM; as containers are running inside VMs. Based on the host state (switch on/off, standby, hibernate) that is performed due to the migrated container and the amount of energy consumed correspondingly; the $\mathcal{MC}_{host}$ address as part of the overall energy consumption.

5 Proposed Solution

We deliberate migrations for consolidating the workload onto the fewest hosts to minimize energy consumption whereas ensuring the probable level of performance. The cost of migration (both in terms of energy consumption and performance) together with the marginal migration cost (host status due to migration—switch on/off) should be accounted as part of the actual migration decision. Based on the total number of accommodated containers, if the target host is identical or less energy and performance efficient than the source host, then it is impossible to recoup the migration cost. Otherwise, the migration cost could be earned back, eventually. Using the container runtime and efficiency factor of both source and target hosts, we can estimate a time point t_{off} at the target host where the container will be able to earn back its migration cost $Cost_{mig}$ and would, essentially, be saving energy, performing better if it lasts to run. The longer it will run, the more savings would be achieved. We call this consolidation technique with migration energy and performance costs recovery (Cper), which is, in next section, described in more detail.

5.1 CPER

Consider a container C that runs within a particular VM at source host H_1. A migration of C has been decided to target host H_2 at time t. Let's assume that H_2 is more energy and performance efficient than H_1 with an energy efficiency factor of E_f. In case, if there is no other container running on H_1 and H_2 except C, then the host which either consumes less power in idle state P_{idle} and/or has higher MIPS rating is considered as more energy and performance efficient than the other. However, if there are other containers running on H_1 and H_2, along with C, then the efficiency of each host could be associated to the total number of running containers (e.g., n containers on H_1 and m containers on H_2). The E_f of target host can be computed as given in Eq. (21) below:

$$E_f = \left(\frac{\mathcal{P}_{C_{source}} \times \mu_{C_{source}}}{\mathcal{P}_{C_{target}} \times \mu_{C_{target}}}\right). \tag{21}$$

If E_f is equal to 1, it denotes that the multiple of power profile and performance rating of both hosts are identical and, therefore, we cannot earn back the migration cost. If E_f is less than 1, then the target host is less energy and performance efficient than the source host. The offset of migration cost and additional savings are only possible if E_f is greater than 1.

The cost of migration $Cost_{mig}$ is measured in Watts per hour (Wh). The difference between the power consumption and performance values (efficiencies) of both source and target hosts is given by:

$$\triangle x = \mathcal{P}_{C_{source}} \times \mu_{C_{source}} - \left(\frac{\mathcal{P}_{C_{source}} \times \mu_{C_{source}}}{E_f}\right), \tag{22}$$

or

$$\triangle x = (\mathcal{P}_{C_{source}} \times \mu_{C_{source}}) - (\mathcal{P}_{C_{target}} \times \mu_{C_{target}}), \tag{23}$$

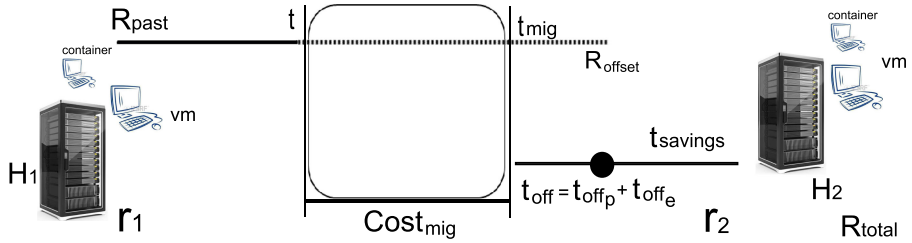


Fig. 3. CPER technique description (an extended version of CMCR [6], [9] that accounts for energy consumption and performance loss) – t_{off_e} and t_{off_p} are the offset points at which the container has recouped back its migration energy and performance costs, respectively.

and, therefore, t_{off} is expressed as:

$$t_{off} = \frac{\mathcal{T}_{mig} \times Cost_{mig}}{\triangle x}. \tag{24}$$

For a particular container C having R_{past} (past runtime) on source host, and its migration was started at time t to the target host and completes in time t_{mig}, as shown in Fig. 3, the total runtime of C on the source host can be expressed as $r_1 = R_{past} + t_{mig}$, and its remaining runtime on the target host can be expressed as $r_2^{'} = R_{total}^{'} - (t - t_{mig}) = R_{total}^{'} - r_1$ [where $R_{total}^{'}$ represents the estimated runtime]. Note that $R_{total}^{'}$ and, hence, $r_2^{'}$ will vary due to differences in performance of both hosts as well as the performance degradation due to migration. Assuming the performance distribution $D_{\mu_{C_{target}}}$ of the target host, the remaining time $r_2^{'}$ can be written as

$$r_2^{'} = D_{\mu_{C_{target}}} \times r_2 + \left(\frac{\mathcal{T}_{mig}}{10}\right), \tag{25}$$

where $r_2 = R_{total} - r_1$ and $\frac{\mathcal{T}_{mig}}{10}$ is the increase in container remaining runtime due to performance degradation during migration. Furthermore, this can also be modelled using the z-score normalization; if the runtimes follows a normal or a log-normal distribution

$$r_2^{'} = exp\left(\sigma_{C_{target}} \times \left\{\frac{log(r_2) - \mu_{C_{source}}}{\sigma_{C_{source}}}\right\} + \mu_{C_{target}}\right) + \left(\frac{\mathcal{T}_{mig}}{10}\right). \tag{26}$$

If $r_2^{'} > t_{off}$, then it means that C has earned back $Cost_{mig}$ and subsequently runs more efficiently to save energy and performs better. The remaining runtime of C on the target host after the t_{off}, is given by:

$$t_s = r_2^{'} - t_{off}. \tag{27}$$

The overall savings $\mathcal{P}_{savings}$ achievable through an energy and performance efficient migration are given by:

$$\mathcal{P}_{savings} = t_s \times \triangle x. \tag{28}$$

Hence, the least value for $r_1 + r_2^{'}$ (as shown in Fig. 3) which is enough to offset $Cost_{mig}$ at time t can be expressed as $R_{offset} = t_{mig} + t_{off}$. Therefore, for every container running for R_{past}, its R_{offset} can be expressed as

$$R_{offset} = R_{past} + \mathcal{T}_{mig} + t_{off}. \tag{29}$$

In case $R_{offset} \geq t_{off}$, it means that the migration is energy and performance efficient. If container C got terminated prior to t_{off}, the cost of migration is not recovered. Moreover, if R_{offset} is not enough to earn back $Cost_{mig}$ then, using Eqs. (30) and (31), t and R_{past} can be estimated to make a migration energy and performance efficient

$$t = t - t_{off} \tag{30}$$

$$R_{past} = R_{past} - t_{off}. \tag{31}$$

In order to trigger appropriate migration decisions for energy efficiency, cost and improved application performance, the above method can be extended, easily, with a runtime prediction technique. For example, as a future work, we might consider runtime prediction methods which are based on historical workload patterns retrieved from a knowledge database [38], [39]. However, the prediction seems depending on the container/physical host; therefore, historical workload patterns on each host are needed, which could be a tedious job, particularly if the datacenter has large number of heterogeneous containers/hosts. Moreover, searching these databases would create significant scheduling delays; however, we bias for allocation speed and implementation simplicity over an absolute optimality.

In the above formulation, both R_{total} and $R_{total}^{'}$ represent the time for which a particular container will run, which might be unknown, particularly, in public clouds. To make the scenario applicable and realistic for such on-demand cloud platforms, we assume the past runtimes of containers (R_{past}) instead of predicting their future runtimes (R_{total}); in order to determine if a particular container is a right candidate for energy-performance efficient migration or not (i.e., a probabilistic approach). Jobs previous runtimes can be used as a way of finding similarities or likelihood of long runs. This assumption is justifiable as described in [39]. Moreover, cloud jobs that run for 30 minutes or longer are likely to be run for hour(s); and those which run for hour(s) are likely to be run for day(s) or even weeks [7], [12]. Therefore, it is reasonable to assume containers' R_{past} as a clue for their longer or shorter runtimes. However, we are aware that this may reduce the applicability of the proposal, as it require the container to be known to the provider; but this is not the case, for instance, with typical IaaS providers (e.g., Google, Amazon EC2). The steps involved in CPER approach are described in Algorithm 1.

Algorithm 1. CPER TECHNIQUE

Input: The list migratable containers L, $Cost_{mig}$, t, t_{mig}, source and target hosts
Output: Return t_{off}, the time point where the $Cost_{mig}$ is recovered

for *each container* $\in$ L **do**
  $E_f \leftarrow (\frac{P_{C_{source}} \times \mu_{C_{source}}}{P_{C_{target}} \times \mu_{C_{target}}})$ [as explained in Section 5.1];
  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**
      $t_{off} \leftarrow currentTime\ (t) + t_{mig} + [\frac{t_{mig} \times Cost_{mig}}{\triangle x}]$;
    else
      $Cost_{mig}$ is not recoverable;
      remove $container$ from L;
    end if
  else
    target is not more energy and performance efficient than the source;
    remove $container$ from L;
  end if
end for
return $t_{off}|L$

6 GOOGLE CLUSTER DATASET

One of the Google's computing clusters workload dataset is publicly available on-line [12]. The dataset traces the activity of a production cluster that consists of 12,583 heterogeneous hosts for a duration of 29 days. Moreover, in [7], these hosts have been classified into three different classes and 10 different architectures, grounded on the number of available CPUs and memory capacity, as described in Table 1. Although, the dataset does not suggest or identify various types of hosts, nevertheless, we are doubtful that these might relate to those three types of hosts that Google describes on-line[5]—Sandy Bridge (2.6 GHz Intel Xeon E5), Ivy Bridge (2.5 GHz Intel Xeon E5 v2), and Haswell (2.3 GHz Intel Xeon E5 v3). The dataset comprises of approximately 672,074 jobs, and each job is created by a single user out of 925 users (total). Every job consists of one or more tasks that combine to approximately 24,281,242 unique tasks. Moreover, the data comprises other information such as task's requirements (CPU and memory) that are helpful to schedule it for execution. The dataset demonstrates that majority of tasks only run for short durations and use a very slight volume of the host's resources that even cannot be assumed as a single unit of CPU or memory. Nevertheless, there are certain long-running tasks that might take week(s) to finish their execution. Existing literature regarding the analysis of the Google's dataset illustrates that, although, there is no identified statistical distribution that fits well the task durations; yet the resources appear to produce a long-tailed distribution [10]. This behaviour might be due to either: (i) human behaviour (irregularity in user's application); and/or (ii) large volume of heterogeneous data (economics of scale).

Moreover, host volumes are normalized to the maximum cores' host within the Google's cluster, whereas disregarding the CPU speed, as described in [12]. For instance, if there are three types of hosts installed with 3.0 GHz with 8 cores, 2.6 GHz with 3 cores and 2.3 GHz with 2 cores, then the normalized volumes for these three hosts will be 1, 0.375 and 0.25 [8:3:2], correspondingly. Google does not deliver precise particulars of their hosts (machines) due to safety and confidentiality reasons. Nevertheless, we can make several plausible and realistic assumptions about host characteristics and architectures from the period when the data was logged in May 2011. Note that if we can extract such details, then the demand for resources (in terms of CPU and memory) will be clearer for simulation purposes.

TABLE 1
Machines Types and Number in Google's Cluster [12]—we Assume Machine Class A, B and C as Equivalent to Sandy Bridge, Ivy Bridge and Haswell Platforms, Respectively [CPU Values are Normalized w.r.t the Highest CPU Server Available in Google's Cluster]

Class	Number	CPU	Memory (GB)	Platform
A (Sandy Bridge)	126	0.25	0.25	*a*
B (Ivy Bridge)	5	0.5	0.03	*b*
	1	0.5	0.06	*c*
	52	0.5	0.12	*d*
	3,863	0.5	0.25	*e*
	6,732	0.5	0.5	*f*
	1,001	0.5	0.75	*g*
	5	0.5	0.97	*h*
C (Haswell)	3	1.0	0.5	*i*
	795	1.0	1.0	*j*

Google does not offer precise information about the cluster usage, however, several researchers suggest from the task usage data that the cluster is only 20–40 percent utilized [7]. Likewise, Sheng et al. [40] projected that the resource demand (in terms of CPU and memory) is sometime larger than the offered peak capacity [80–120 percent utilised—possibly when cluster resources are oversubscribed], however, the cluster is mostly utilized within the range of 20–40 percent.[6] Empirical evaluation of Google workload traces in [41] suggests that resource usage could be accurately modelled simply using the mean of task usage i.e., "*mean usage model*". This shows that resource usage for CPU are relatively stable over time for majority of the tasks. The data itself is anonymized that makes it hard to distinguish if the application were running on a real host (bare metal), in a virtual machine (VM) or inside a container. Several Google's researchers endorse that a container-based virtualisation (containerization) can be supposed. Each job comprises of several tasks (might be more than thousands) that possibly will or will not run on the same host. A task is a Linux program, undoubtedly holding numerous processes, nonetheless, still run on a single host. Thus, it is also practical and realistic to accept each task as a container. Using container runtimes, we further identify and illustrate hosts heterogeneities and their performance parameters inside the Google's cluster.

6.1 Modelling Heterogeneity in Google's Cluster

The tasks information does not reveal to what kind of applications, they belong to. Unfortunately, the dataset

5. https://cloud.google.com/compute/docs/machine-types

6. http://blog.stillwell.me/blog/2013/07/15/first-steps-exploring-the-google-cluster-dataset-with-ipython/

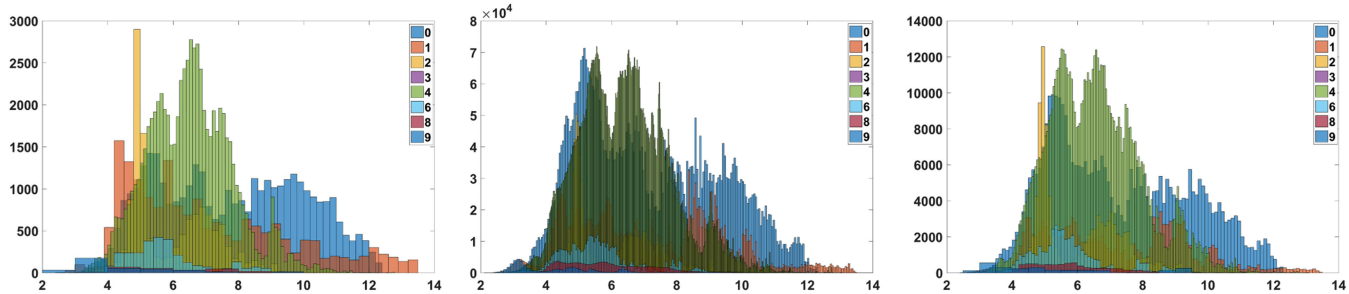


Fig. 4. Performance variations of various applications on machine class A (left), B (middle) and C (right) [priority 0 workload performs better on class B compared to A and C; priority 4 workload performs better on class C compared to A and B – *x-axis* denotes containers runtimes (in log) and *y-axis* denotes the number of containers].

does not deliver precise facts of the host specifications and heterogeneity. Nevertheless, we suppose that such heterogeneity will directly interpret into dissimilarities in energy consumption and performance. We use containers runtime as an evaluation metric to characterize performance variations amongst various host types; lower runtimes represent good performance and vice versa. Still, the dataset does not have what we want (i.e., various kinds of workload); however, every task has a certain priority (0–11) and we can use this as a representation for workload types. As, the dataset providers indicate that priority of every task affects billing [7], therefore, we trust that it will reflect the workload type, accurately.

Each task inside a particular workload type can be assumed a container which runs on one of the aforesaid host types [Table 1]. In Google data, the machine and platform ID's of each task (container) can be used to extract the host's type where resources were provisioned for the container. We further assume the runtime (execution time) of each container as an evaluation metric for performance as it is more beneficial to an IaaS customer. We selected ~21,200,338 tasks after discounting those tasks where machine details were absent. Upon visual inspection, the runtime (log) distribution seems to follow a multi-modal lognormal distribution, essentially. The multi-modality might indicate that analogous applications perform in a different way on various machines. It is advised in [29] that multi-modality narrates to CPU architectures and performance is principally determined by the CPU model and the available amount of memory.As shown in Fig. 4, the priority 0 workload performs better on class B machines compared to class A and class C machines. Similarly, priority 4 workload performs better on class C machines compared to class A and class B machines. Therefore, migration of 0 priority workload from class A and class C machines to class B machines is performance efficient; and may be energy efficient as well. Furthermore, avoiding migrations to less performance efficient machines and those that could not recover their migration cost, may lead to energy and performance efficiencies. To investigate the affect of this, we consider three kinds of workloads (i.e., priority 0, 4 and 9); and calculate machines heterogeneities in terms of performance parameters (mean - μ and standard deviation - σ). "We do not rule out the existence of natural computational variations, however, as the data fits to prior findings on performance, so we can relate these data to heterogeneous infrastructure clouds" [10]. The machines' heterogeneity performance parameters are described in Table 2.

We use these performance parameters to calculate the overall energy and performance efficiency (E_f) of a particular host, as explained in Section 5.1. Note that, σ have a key role and would affect the efficiency level (due to overlapping histograms). However, for simplicity, we assume that with given mean values (μ) for source and target hosts, we can differentiate between their performance levels. Moreover, increase and decrease in containers performance is characterised as increase and decrease in containers runtime, using z-score normalization.

7 Performance Evaluation

Bin-packing problems are usually solved using various heuristics that might not guarantee optimal results, however, they are enough fast to address large-scale problems [14]. It is possible to assume a comparable container packing problem as moving from a given (initial) state of the datacenter to an ideal state, which should run the current demand on fewer hosts. We realise a datacenter state through implementing various scheduling techniques (such as RR, FF, FillUp (Fu), EPC-aware FillUp (Epc-Fu) as initially stated in Section 1), with container packing then needing to guarantee energy and performance efficiencies are assured (as explained in Section 4.1) and the cost of migration (both in terms of energy and performance) can be recouped (earned back). To demonstrate the impact of this, we consider (a) no migration; (b) all possible migrations (dynamic consolidation); and (c) runtime-based migration (Cper). Moreover, we also compare Cper (containers migration) technique to Cmcr technique (VMs migration) [6]. The steps involved in Epc-Fu container allocation and migration policies are described in Algorithms 2 and 3, respectively.

TABLE 2
Parameters for Hosts Performance (Suggesting Different CPU Architectures) in Google Cluster [12] – Lognormal Distribution of Containers Runtime

Machine class	Workload type					
	priority 0		priority 4		priority 9	
	μ	σ	μ	σ	μ	σ
A (Sandy Bridge)	7.79	2.19	6.53	1.31	4.37	1.63
B (Ivy Bridge)	7.16	2.1	6.43	1.4	4.39	1.55
C (Haswell)	7.28	2.11	6.56	1.4	5.19	2.08

Algorithm 2. EPC-AWARE FILLUP (EPC-FU) CONTAINER ALLOCATION

Input: List of hosts (H), wait queue (W), List of container requests (C)
Output: Energy and performance efficient container placement
sort H in decreasing order of ($\mu.P_{container}$) [as described in Section 5.1];
for *each* $container \in C$ **do**
  for *each* $h \in H$ [H *is sorted with respect to the available slots*] **do**
    if h *is active and has enough resources to run the* $container$ **then**
      allocate $container$ to h;
      break the loop and pick the next $container \in C$;
    end if
  end for
  if $container$ *cannot be allocated to any active* $h \in H$ [*in case DCP is assumed*] **then**
    start a new $h' \in H$ and assign $container$ to h';
  else
    "*container* can not be allocated";
    "push the *container* request into W";
  end if
end for

Algorithm 3. EPC-AWARE CONTAINER CONSOLIDATION

Input: List of migratable containers L, Container c (that is to be migrated), H_{source} and H_{target}
Note: [($R_{remainimg} = R'_{total} - R_{past}$)—we use R_{past} instead of $R_{remaining}$, which are known. Algorithm 4 prioritises containers based on their R_{past} for migration.]
Output: Return migration decision d
for *each container* $c \in L$ **do**
  $d \leftarrow$ FALSE;
  estimate $R_{remaining}$ of c [assuming it is possible];
  $\text{ERP}_{source} = \mu_{H_{source}}.P^c_{H_{source}} \times R_{remaining}$ [$E^c_{H_{source}}$ is the power consumed by c on host H_{source}];
  $\text{ERP}_{target} = \mu_{H_{target}}.P^c_{H_{target}} \times R_{remaining}$ [$E^c_{H_{source}}$ and $E^c_{H_{target}}$ are calculated using the power model presented in [6]];
  if $\text{ERP}_{target} < \text{ERP}_{source}$ **then**
    $d \leftarrow$ TRUE [i.e., migrate c using the migration model in [6]] [note that the migration model in [6] accounts for migration costs in terms of energy consumption, migration duration and performance degradation];
  end if
end for
return $d|L$

We assume the consolidation (migration) process as an optimization problem with the objective to decrease the number of hosts in use. Every 5 minutes' interval, the optimization module is run grounded on the present utilization level of all hosts, in 3 steps; (1) *containers selection:* Every host is monitored and if its present utilization level is less than a pre-defined $Threshold_{low}$ (lower threshold value e.g., 20 percent), all accommodated containers on this particular host are chosen for migration. If there are several containers suitable for migration, then the suggested container selection policy [Algorithm 4] gives priority to the container that runs for longer duration i.e., R_{past}—(migrate one container at a time from a single host to minimize performance loss); (2) *hosts selection:* The migration policy chooses the most suitable host from all available hosts that could run these containers. Nevertheless, to decrease the number of active hosts, it avoids allocation to: (i) switched off hosts (if it is possible); and (ii) hosts that intend to go into idle|switched off state (switched on but with no work on them|idle); and (3) *placement:* The list of selected containers is sorted in decreasing order of their R_{past} (past runtimes) that favours to migrate long-running containers first. Finally, a particular container allocation algorithm is used to reallocate all containers, as container placement is a sub problem of the consolidation with migration process.

Algorithm 4. CONTAINER SELECTION POLICY

Input: List of migratable containers (C)
Output: Select a suitable container for migration
$container_{suitable} \leftarrow$ NULL;
for *each container in* C **do**
  access R_{past} [from containers history];
end for
sort C in decreasing order of past runtimes R_{past};
$container_{suitable} \leftarrow C[FIRST]$
  [$FIRST$ denotes the 1st $container$ in $C \rightarrow C[0]$];
return $container_{suitable}$

Metrics. The metrics used to evaluate the energy and performance efficiency of the proposed container allocation and migration policies are: (i) total number of container migrations; (ii) total energy consumed (E) in KWh; (iii) execution time (T) in seconds—as application's performance is inversely proportional to T; (iv) Energy Runtime (performance) Product (ERP), as explained in Section 3; and (v) electricity bill in dollars ($)[10]. For (v), we assume a PUE[7] of 1.10 and energy price of 0.88$ per KWh[8] that mimic a Google datacenter located in Oklahoma, USA.

7.1 Experimental Set-Up

A cluster (simulated using CloudSim) of 12,583 heterogeneous hosts, which comprises of various architecture types (with respect to varying performance) and hardware specifications—as given in Table 3 - is available to run various kinds of benchmark workload. Moreover, the hosts are subdivided by architecture ground on the type of workload they run, as described in Section 4.1. The simulated hosts are configured based on several reasonable assumptions that Google had certain kinds of commonly available servers in their IaaS cloud, when the dataset [12] was traced. The specification (of hardware) and energy consumption values for the hosts were collected from the well-known SPECpower[9] benchmarks.

Our simulation comprises of 6 VM types which resemble to Amazon's instance classes as given in Table 4. The VM types are further ranked (in terms of resource capacities and performance) according to Amazon's description of their VM performance rating—ECU (the EC2 Compute Unit), which is described as: "equivalent CPU capacity of a 1.0—1.2 GHz

7. https://www.google.co.uk/about/datacenters/efficiency/
8. https://www.eia.gov/electricity/monthly/
9. https://www.spec.org/power_ssj2008/

TABLE 3
Host Characteristics for Google's Cloud [the Idle (P_{idle}) and Maximum Power Consumption (P_{max}) of Hosts are Taken from SPECpower Benchmarks]

CPU model	Speed (GHz)	No of cores	No of ECUs	Memory (GB)	P_{idle} (W)	P_{max} (W)	Total amount
Sandy Bridge (Xeon 2670)	2.6	8	20.8	384	55	105	
Ivy Bridge (Xeon 2670)	2.5	10	25	768	65	115	12,583
Haswell (Xeon 2695)	2.3	14	32.2	768	70	120	

2007 Opteron or 2007 Xeon processor"; and its variations in performance is suggested approximately 20 percent (1.0—1.2 GHz) in [42]. Since, the ECU rating is per core, therefore, the total rating of a particular host can be calculated by multiplying its total number of cores and ECU rating [29].

We assume that these hosts are comparable by a single measure that permits for performance ranking, for which we assume the CloudSim's notion of "Million of Instructions Per Second (MIPS)" as a proxy; however, we do not endorse this as a good performance metric for real systems due to several reasons including workload comparability and CPU architecture. One technique to VM sizing is to assign a VM as a single core for the maximum value 1, half a core (hyperthread) for 0.5, and consider that higher VM gearing leads to a quarter of a core for 0.25. But to address allocation more flexibly, along lines with particular IaaS providers, we map CPU frequency for the hosts given to Amazon's ECUs as: 1 GHz CPU, 1.7 GB RAM, giving numerous instance types.[10] Further, the ECU maps MIPS for consistency with the CloudSim simulator (Table 4), and we consider that each instance requires, at least, 1 ECU and 1 vCPU (core) or more, as given in Table 4. The speed of every instance type (MIPS rating) is the multiple of number of ECUs (1 ECU = 1GHz) and vCPUs (cores). For instance, the speed of the *m1.medium* instance, as shown in Table 4, is 2 (ECUs) $\times$ 1 (vCPU) = 2 (GHz).

To address a cloud context such as Amazon Lambda, each Google's task is assigned a single, notional, container, as shown in Table 5, that corresponds to Google machine types with the only exception that all the containers are single core. Similar to VM sizing, containers are also sized (MIPS) and each VM can accommodate more than container. For example, a *m3.medium* instance can accommodate three containers of type A, while two containers of type C and so on. Each task utilizes its allocated resources according to task statistics and usage in Google data [12].

Each container is assumed to run three workload types that belong to Google's cluster dataset. The utilization of each workload type is modelled as a normal distribution function over the mean CPU usage in the trace, at 5 minute intervals. Furthermore, the total execution time of each application is the sum of its each task's execution time.

7.2 Experimental Results

The simulated infrastructure is composed of 12,583 hosts, 3,800 VMs with configuration shown in Table 3, Table 4 and three kinds of workloads that belong to priority 0, 4 and 9, respectively. When a container request is received, a container is created from a list of available flavours as shown in Table 5, and is placed on a suitable VM already running on a particular host. 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 4. Various container types are described in Table 5. We assume that the container workload is heterogeneous and, therefore, changes when a container is migrated from one host to another. After each 5 minute interval, the CPER technique checks for consolidation opportunities, and selects container running for longer times from a list of migration possibilities. Each experiment was performed with five different values for past container runtime given in hours [0, 0.25, 0.5, 0.75, and 1], where 0 means migrate all—dynamic consolidation—and 0.25 means migrate only those containers which are running for 15 minutes or longer, 0.5 means running for 30 minutes or longer, and so on.

7.3 Results Discussion

Fig. 5 presents the results obtained from running three kinds of workloads, with error bars of standard deviations, using various scheduling and consolidation heuristics. Below, in various sections, we describe and analyse our obtained results with respect to various aspects.

7.3.1 Energy Consumption

We observed that the energy consumption of various applications varies significantly from 1.75 to 43.31 percent, across various policies and hardware platforms. The results show that EPC-aware scheduling techniques would be more economical than consolidation techniques (particularly PRIORITY 9 workload). For example, without migration a

TABLE 4
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 5
Container Types and their Characteristics [33]

Container type	Speed (MHz)	Cores	ECU's	Memory (MB)
A	1,000	1	1	128
B	1,225	1	1.23	256
C	1,500	1	1.5	512

10. http://www.ec2instances.info

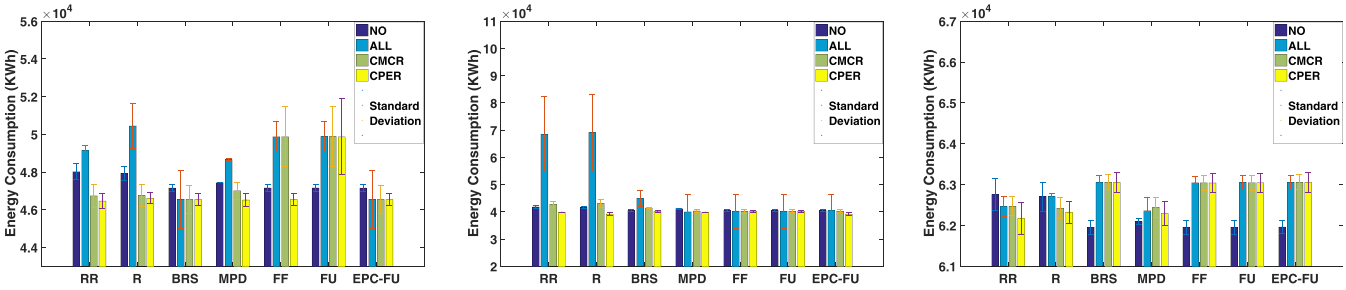


Fig. 5. Energy consumption in KWh for three kinds of workloads [priority 0, 4 and 9 from left to right – minimum values are best; the error bars show standard deviations with respect to mean values].

1.27 percent decrease in energy consumption was achieved using EPC-FU instead of RR. But using EPC-FU, only 0.93 percent decrease in energy consumption was achieved with dynamic consolidation. The results show an average decrease of 1.19 percent in energy consumption for EPC-FU compared to R scheduler. Similarly, for CPER, EPC-FU is on average 0.71 percent more cost efficient as compared to FF scheduler. We also note that no migration can be more economical than the dynamic consolidation, CMCR and CPER (FU) for certain kinds of workload, if an EPC-aware scheduling approach is used. CPER beats both consolidation techniques (ALL and CMCR) as it allocate containers to the most energy and performance efficient hosts first in order to save energy and maintains the expected performance level, minimizes the total number of migrations (runtime-based migration) and increases the probability that a container recovers its migration costs in terms of energy and performance. For PRIORITY 4 workload, the FU scheduling combined with CMCR approach leads to minimum ERP; however, a combination of EPC-FU and CPER beats this. Moreover, CPER reduces the total number of container migrations as described in Fig. 6.

7.3.2 Performance

Table 6 shows the workload performance, mean number of hosts in use, ERP, execution time and datacenter utilization, measured in 5 minute intervals. Note that, values in Table 6 are only shown for best approaches i.e., FU and EPC-FU. The results show that in terms of scheduling approach, EPC-FU is effective in using minimum number of most efficient hosts: EPC-FU did not allocate containers to host type A which has larger idle power consumption and is less energy and performance efficient compared to type B and type C hosts. If we migrate only for energy efficiency (CMCR), then performance of the workload might be affected. However, migration both for energy and performance efficiency (CPER) always ensure energy savings with expected, improved or at-least no performance degradation.

The runtime of migrated containers depends on the scheduling heuristics and workload type. The RR scheduler distributes containers equally amongst the available VMs and, therefore, hosts, keeping all the VMs|hosts running but least utilized most of the time—creating large migration (consolidation) opportunities. Likewise, the R scheduler typically chooses a different host for container allocation through randomisation, which might result in high energy consumption and increased migration efforts—since all hosts are active but least utilized. Therefore, the optimal value for both, RR and R, techniques is always achieved with CPER, with past runtime $\leq$ 15 minutes. Moreover, the most efficient heuristics such as FF, FU and EPC-FU, produces optimal results by migrating containers with past runtime $\leq$ 15 minutes.

7.3.3 Migration Costs Recovery

The data and migration statistics produced in Table 7, show that combining CPER and EPC-FU means only 1.9 percent of containers are migratable and 61.89 percent of these were able to recover their migration cost. For EPC-FU with dynamic consolidation, 2.73 percent containers were migrated with 11.2 percent of recovering migration cost. Our investigation suggests that CMCR migration technique [6] is largely unable to recover containers migration cost; as heterogeneities of the resources and applications were not taken into account. Our proposed technique CPER increases the probability of recovering the migration cost in a containerized platform. For three different kinds of Google trace workloads [12], with the same simulation, we see that no-migration technique would be more economical than dynamic consolidation if efficient container and VM scheduling heuristics are

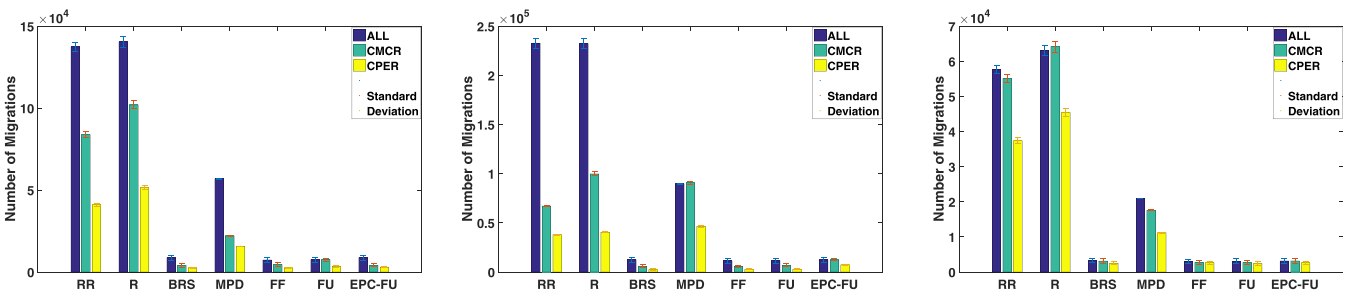


Fig. 6. Number of migrations for three kinds of workloads [priority 0, 4 and 9 from left to right – minimum values are best; the error bars show standard deviations with respect to mean values].

TABLE 6
Experimental Mean Results for Different Approaches (5 min. Interval) – the Host in use are the Maximum Number of Host which were Used During the Entire Experiment [Best Values are Shown in Boldface]

Workload type	Scheduling approach	Consolidation technique	Hosts in use A	Hosts in use B	Hosts in use C	Avg. hosts	VMs created	Dcenter util (%)	Ex. time (minutes)	ERP	Bill $
Priority-0	Fu	No	0	4,194	3,456	3,189	10,312	33.94	315.48	3.444	45,645.7
		All	0	4,194	3,659	1,282	6,467	76.58	314.56	3.632	48,282.3
		Cmcr	0	4,194	3,111	1,291	5,989	74.0	314.67	3.632	48,282.3
		Cper	0	4,194	3,051	1,282	5,981	76.6	**313.87**	**3.39**	48,268.8
	Epc-Fu	No	0	4,194	2,532	3,179	9,991	34.02	315.48	3.443	45,642.0
		All	0	4,194	3,456	1,375	7,012	77.8	314.38	3.388	45,066.1
		Cmcr	0	4,194	2,123	1,381	6,988	77.22	314.58	3.389	45,049.0
		Cper	0	4,194	2,121	1,373	6,999	77.8	**313.91**	**3.387**	**45,046.9**
Priority-4	Fu	No	0	4,194	1,181	1,484	8,564	38.92	84.76	0.794	39,187.2
		All	0	4,194	798	893	5,783	63.17	84.33	0.786	38,972.8
		Cmcr	0	4,194	512	903	5,893	61.41	84.42	**0.755**	38,960.9
		Cper	0	4,194	610	894	5,821	63.04	**83.99**	0.76	38,926.1
	Epc-Fu	No	0	4,194	1,392	1,503	8,766	37.67	84.76	0.794	39,190.5
		All	0	4,194	845	925	7,970	59.94	84.3	0.786	39,173.1
		Cmcr	0	4,194	451	756	7,897	59.95	84.46	0.785	38,959.0
		Cper	0	4,194	451	555	7,871	60.35	**84.07**	**0.743**	**37,919.4**
Priority-9	Fu	No	1,211	4,194	4,194	1,766	8,926	44.91	1,107.42	15.883	**59,975.9**
		All	873	4,194	4,194	939	7,122	81.06	1,085.6	15.84	61,032.9
		Cmcr	971	4,194	4,194	944	6,762	80.46	1,085.98	15.194	61,029.1
		Cper	777	4,194	4,194	939	6,769	81.03	**1,085.38**	**14.912**	61,027.3
	Epc-Fu	No	1,300	4,194	4,194	1,765	8,799	44.94	1,107.42	15.883	59,977.6
		All	1,233	4,194	4,194	939	8,110	80.84	1,086.8	15.862	61,038.7
		Cmcr	742	4,194	4,194	944	8,082	80.37	1,086.74	15.859	61,038.7
		Cper	878	4,194	4,194	938	8,062	80.85	**1,085.93**	**15.858**	61,036.0

used. Our second finding is that migrating relatively long running containers to more energy and performance efficient hosts to recover their migration cost, are more economical and energy, performance efficient.

Furthermore, we observed that the cost recovery approach is more beneficial to VMs migration [6] than containers migration. Possibly, this is due to: (i) large number of container migrations triggered during each consolidation round (5 minute intervals); and/or (ii) containers were not running for enough time on the target hosts to recover their migration costs. We observed an increase in cost recovery and decrease in number of migrations for long consolidation intervals. Moreover, we also observed that repeatable migration of a particular container degrades the workload performance. Techniques like Cmcr and Cper could minimise the total number of migrations, however, they are not able to avoid these repeatable migrations. A control mechanism is needed to avoid such efforts for migrations. For example, we can migrate a particular container only once in a pre-defined duration such as an hour; or migrate those containers which are being migrated less frequently, in the past. This would certainly increase the workload performance and infrastructure energy efficiency.

7.3.4 Generalization of Findings

As shown in Table 6, Cper produces good results for three various workloads that correspond to three various real datasets. Moreover, we have also checked the applicability of Cper technique on three other workload datasets i.e., Priority 1, Priority 2 and Priority 5. In order to show that Cper is applicable and would also work in a real cloud test-bed, we run these experiments with different datacenter set-up, assumptions and workload sizes. In other words, we validated Cper technique to check whether generalization of our findings and results is correct or not. The experimental details and results are shown in Table 8. The host types,

TABLE 7
Cost Recovery with Dynamic Consolidation (Dc - All), Cmcr and Cper

Scheduling approach	Priority-0						Priority-4						Priority-9					
	Fu			Epc-Fu			Fu			Epc-Fu			Fu			Epc-Fu		
	Dc	Cmcr	Cper	Dc	Cmcr	Cper	Dc	Cmcr	Cper	Dc	Cmcr	Cper	Dc	Cmcr	Cper	Dc	Cmcr	Cper
(%) migratable containers	2.1	1.98	1.61	2.21	2.09	1.89	3.78	3.04	2.38	3.73	2.99	2.9	2.74	2.08	2.0	3.01	2.87	2.55
(%) containers recovered	10.89	41.08	53.87	19.56	39.87	61.89	9.8	36.76	49.88	11.2	34.1	50.2	11.56	37.45	47.2	12.9	41.8	59.87
$Cost_m$																		

TABLE 8
Experimental Results Across Various Datacenter's Set-Up, Assumptions and Workload Sizes – the "Best" Values are Shown in Bold Face [Readers are Advised to see [10] for Hosts Characteristics and Performance Parameters]

Datacenter set-up							Results			
Workload type	Scheduling approach	Consolidation technique	Hosts	Types of hosts	VMs	Containers	Energy (KWh)	Ex. time (minutes)	ERP	Bill $
Priority-1	Epc-FU	No	1,500	E5430	3,200	11,390	163.87	31.34	1.43	158.6
		All		E5507			146.8	32.67	1.33	142.1
		Cper		E5-2650			**134.9**	**30.44**	**1.14**	**130.6**
Priority-2	Epc-FU	No	3,200	E5507	7,600	26,300	487.75	59.89	8.11	472.1
		All		E5645			**439.4**	61.99	7.57	**425.3**
		Cper		E5-2651			440.1	**59.1**	**7.22**	426.0
Priority-5	Epc-FU	No	400	E5430	1,200	1,600	25.66	19.2	0.14	24.8
		All		E5645			26.21	20.26	0.15	25.4
		Cper		E5-2670			**19.01**	**18.69**	**0.1**	**18.4**

their energy consumption and performance parameters were taken from our previous work [10], as it is. Moreover, the number of containers correspond to the total number of tasks in a particular workload type. A consistency of our results and major findings, in terms of least ERP values, can be seen across all the experiments. Since, our objectives are minimizing energy consumption and improving performance; however, the datacenter bills are totally estimated on energy use only. Therefore, datacenter bills are not guaranteed to be optimal, for our techniques in Table 6 - Priority-9.

7.3.5 Statistical Significance of Results

In order to demonstrate that there are, significant, statistical differences among the means of the produced results from our approaches and others, we performed the *t-test* (post-hoc) statistical analysis, repeatedly [10]. This can be achieved through calculating the probability of error (p value) by the *t-critical* ratio. The difference between two approaches (datasets) is said to be statistically significant, if and only if $p \leq 0.05$. When $p = 0.05$ (i.e., confidence interval 95 percent), then the differences between means of the two datasets have only a 5 percent probability of appearing by chance [10]. As shown in Fig. 7, the least values for p (*t-critical* = 2.447) show a clear efficiency of the proposed Epc-FU and Cper techniques over the combinations of No, All, Cmcr and various allocation approaches.

We observed no-overlaps for the Epc-FU allocation policy and others [Fig. 7 – left]; however, the existing overlaps among All, Cmcr and Cper [Fig. 7 – right] verifies our previous discussion and, therefore, results. However, the mean value of Cper [Fig. 7 – the red line in each bar] shows that, on average, it outperforms the other consolidation techniques.

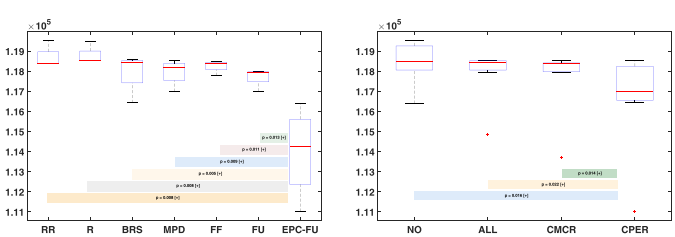


Fig. 7. T-test analysis for various allocation [left] and migration [right] policies; *y-axis* denotes energy use in KWh – [+] denotes significant difference between the means of datasets.

In this Section 7.3, we demonstrated the impact of workload runtimes on resource allocation and migration decisions, workload performance, energy efficiency, costs and the recovery of migration costs when heterogeneous resources and applications are taken into account. Our empirical evaluation demonstrates that energy efficiency and workload performance vary across different hardware platforms. As a result, potential energy, performance and cost benefits can be achieved, in datacenters, through energy and performance efficient allocation and migration techniques such as Epc-FU and Cper. We believe that energy, performance and costs recovery, in combinations, are relatively unexplored in the existing literature, as described in Section 8.

8 Related Work

Consolidation of VMs have been largely studied in the literature, however, containers (running on bare metal), containers running inside VMs and their consolidation is not investigated yet. Sareh et al. [43] have described containers and resource management in a containerized datacenter. Various container allocation and consolidation techniques are empirically demonstrated in [33] using an event driven cloud simulator—ContainerCloudSim [44]. In their work, although energy efficiency of the datacenter is explored, however, various applications and performance of the workload are not discussed. Furthermore, the work presented in [33], [43], consolidate either container or VMs; the resource manager is not able to decide itself whether a container or a VM should be migrated.

Pongsakor et al. [34] presented a container re-balancing technique in order to increase the container schedule rate and cluster utilization. Container migration is discussed in the context of a large real dataset from Google. Moreover, a comparison of VMs and containers is presented. However, their approach is based on the assumptions that containers run on the bare metal. Also, the model used to capture the container migration time is rebuttal. Nider et al. [45] have investigated the migration of containerized applications between servers inside a datacenter (heterogeneous), in order to improve power efficiency. A post-copy container migration technique is implemented based on the CRIU technology. Their results demonstrate that the post-copy

migration approach significantly reduces container's downtime, and potentially reduces network traffic as well.

Felter et al. [8] studied resource management of containers (Docker) and VMs (KVM), and compared the achievable performance level of various applications (CPU, memory, storage and network intensive) w.r.t bare-metal. The authors have considered workload metrics such as latency and throughput to determine virtualisation and containerization overheads. The interesting point in their experiments is that the execution times for VMs and containers overlap for certain kinds of workload. Moreover, their findings show that "containers and VMs impose almost no overhead on CPU and memory usage; they only impact I/O and OS interaction". Based on their research finding, the authors reject the idea that "IaaS should be implemented on VMs and PaaS on containers"—as there is no technical reason [8]. Unfortunately, migrations are not taken into account.

Scheepers et al. [46] suggest resource isolation as one of the major issues in container-based virtualisation. The authors have investigated the performance of hypervisors (Xen) and containers (LXC) for various application types. For a script written in PHP and which inserts randomly generated data into a database, the authors concluded that the same script executes in 16 seconds on a Xen platform, while it took 335 seconds on a LXC platform. This shows that LXC are unable to isolate resources successfully as compared to Xen.

Chenying et al. [47] proposed a container live migration technique i.e., longing and replay—which is similar to pre-copy VM live migration. In the first step, a new container is started at target host and a log file is created at the source host to store the source container activity. Iteratively, the log file is replayed on the target host, until the log file is small enough. Finally, the container is started on the target host, and its copy on the source host is terminated. Their approach reduces the application downtime and container migration time, significantly. Nadgowda [48] have discussed containers live migration in more details; and proposed Voyager—which combines CRIU-based memory migration with data federation capabilities of union mounts to minimize migration downtime. "With a union view of data between the source and target hosts, Voyager containers can resume operation instantly on the target host, while performing disk state transfer lazily in the background" [48]. All of these techniques, with notable exception of [8], have focused on live migration of VMs and/or containers both, hence server consolidation, but cost recovery in migration, heterogeneity of datacenter resources and applications are not addressed.

This work is different from CMCR presented in [6] in two different ways: (i) the host efficiency quantification approach in [6] does not care for host performance factor; and (ii) if we consider VM migration for other factors such as electricity price, renewable energy generation, user mobility and new system such as cloudlets, edge or fog computing, then the proposed scheme in [6] may not work in that case. CPER addresses these issues and hence could be beneficial to migrate containerized applications. In production clouds, containers are used instead of VMs that might be running on either: (i) bare metal (Google); or (ii) inside VMs (AWS EC2 container service). In respect to (i), VMs migration could be replaced with containers migration. However, in respect to (ii), there are two possibilities: (a) containers can be migrated to other VMs; and/or (b) VMs can be migrated to other hosts.

9 Conclusions and Future Work

Continual investigation of energy and performance efficient resource management will be indispensable to support in addressing the most important issues of national energy supply, global warming, and rising fuel costs and, particularly, to evade the need for datacenter outages. Improvements in energy-performance efficiency must decrease energy consumption, and therefore energy bills, and associated costs to energy infrastructures, along with concomitant environmental benefits. Rich literature is devoted to container allocation and migration; however, they appear to ignore: (i) resource plus workload heterogeneities; (ii) migration cost in terms of energy, performance and its recovery; and (iii) the impact of runtimes on migration decisions. In this paper, we considered combinations of several scheduling methods and consolidation with migration approaches, with information of the container past runtimes, to investigate energy saving potentials and performance of various workload types in containerised datacenters. In particular circumstances, we found that not migrating containers could be more energy-performance and, therefore, cost efficient than using dynamic consolidation. For a million containers, the best approach overall limits migrations to approximately 1.9 percent of containers, of which 61.89 percent recover their migration cost.

IaaS providers such as AWS EC2, possibly, will take benefits from our suggested consolidation with migration technique to save energy (therefore money), and improve customer experience; that might be translated to: (a) reinvestment in buying more infrastructure; (b) pass into provisioning costs—reduce user monetary costs; and (c) get reputation for the business - minimize CO_2 emissions. Our evaluation suggests that migrating comparatively long-running containers to more energy and performance efficient hosts can be more economical and cost efficient. Nevertheless, we did not address how container runtimes should be estimated. In public IaaS clouds, this will be a challenging task as the service providers, usually, do not have knowledge of the user workload. The use of machine learning based prediction approaches and their precision is an exciting area for further investigation and research. Moreover, there are certain limitations, as described in Section 4.1.1, with the mathematical models which we used in our evaluation. We have keen interest in the implementation of the suggested model using accurate and validated mathematical models. Moreover, our next step would be to implement and validate the presented model on a real cloud platform. From the applicability point of view, [10] describes how the proposed framework will be put in practice.

Emerging systems such as cloudlets, edge/fog computing and mobile edge clouds (MECs) also trigger the need for migration techniques, particularly, to run user's application at the edge of the network. One major issue that

comes with proximity is how to ensure that customers always receive good or expected level of performance as they move across different locations. In future, we intend to extend this work for such scenarios [21]. Moreover, containers and VMs may perform quite differently for various application types. Similarly, electricity prices may vary in various geographical locations or energy sources; that would certainly affect cloud economics when migrations are performed among datacenters. Further research is needed to investigate various applications, their performance and resource management inside containers, VMs and when containers are placed over VMs, in various locations.

ACKNOWLEDGMENTS

This work is supported by Abdul Wali Khan University, Pakistan. This is substantially extended version of a preliminary version presented at the 13th International Conference on Economics of Grids, Clouds, Systems, and Services.

REFERENCES

[1] [Online]. Available: http://www.telegraph.co.uk/finance/newsbysector/energy/11923465/Blackout-risk-rises-as-UK-energy-crisis-deepens.html, Accessed on: Jul. 21, 2016.
[2] A. Shehabi, S. Smith, N. Horner, I. Azevedo, R. Brown, J. Koomey, E. Masanet, D. Sartor, M. Herrlin, and W. Lintner, "United states data center energy usage report," Lawrence Berkeley Nat. Laboratory, Berkeley, CA, USA, Tech. Rep. LBNL-1005775, vol. 4, 2016.
[3] P. Delforge, "America's data centers are wasting huge amounts of energy," *Natural Resources Defense Council (NRDC)*, pp. 1–5, 2014.
[4] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with borg," in *Proc. 10th Eur. Conf. Comput. Syst.*, 2015, Art. no. 18.
[5] Y. Tay, K. Gaurav, and P. Karkun, "A performance comparison of containers and virtual machines in workload migration context," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. Workshops*, 2017, pp. 61–66.
[6] M. Zakarya and L. Gillam, "An energy aware cost recovery approach for virtual machine migration," in *Proc. Int. Conf. Econ. Grids Clouds Syst. Serv.*, 2016, pp. 175–190.
[7] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. 3rd ACM Symp. Cloud Comput.*, 2012, Art. no. 7.
[8] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2015, pp. 171–172.
[9] M. Zakarya, "An extended energy-aware cost recovery approach for virtual machine migration," *IEEE Syst. J.*, vol. 13, no. 2, pp. 1466–1477, Jun. 2019.
[10] M. Zakarya and L. Gillam, "Energy and performance aware resource management in heterogeneous cloud datacenters," PhD dissertation, Dept. Comput. Sci., Univ. Surrey, Guildford, Surrey, U.K., 2017.
[11] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Comput.: Practice Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.
[12] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format+ schema," Google Inc., White Paper, pp. 1–14, 2011.
[13] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw.: Practice Experience*, vol. 41, no. 1, pp. 23–50, 2011.
[14] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Comput. Syst.*, vol. 27, no. 8, pp. 1027–1034, 2011.
[15] M. Mishra, "Towards a unified theory of VM placement," PhD dissertation, Indian Inst. Technol., Mumbai, India, 2015. [Online]. Available: https://www.cse.iitb.ac.in/, Accessed on: Jul. 2, 2017.
[16] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *Proc. IEEEIFIP Netw. Operations Manage. Symp.*, 2006, pp. 373–381. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1687567
[17] A. Roytman, A. Kansal, S. Govindan, J. Liu, and S. Nath, "Algorithm design for performance aware VM consolidation," Microsoft Res., Redmond, WA, USA, Tech. Rep.: MSR-TR-2013-28, 2013.
[18] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2008.
[19] V. Gupta, "Stochastic models and analysis for resource management in server farms," PhD dissertation, Intel Corporation, 2011. [Online]. Available: http://ra.adm.cs.cmu.edu/anon/usr/ftp/usr0/anon/2011/CMU-CS-11-114.pdf, Accessed on: Mar. 18, 2018.
[20] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. Kozuch, "Energy-efficient dynamic capacity provisioning in server farms," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-10–108, 2010.
[21] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.
[22] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: State of the art and research challenges," *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 430–447, Mar./Apr. 2018.
[23] Diamanti, "2018 container adoption benchmark survey," 2018. [Online]. Available: https://diamanti.com/wp-content/uploads/2018/07/WP_Diamanti_End-User_Survey_072818.pdf
[24] Diamanti, "Five reasons you should run containers on bare metal, not VMs," 2018. [Online]. Available: https://diamanti.com/wp-content/uploads/2018/07/Diamanti_WP_Five_Reasons_You_Should_Run_Containers_on_Bare_Metal_071918.pdf
[25] J. Shuja, A. Gani, S. Shamshirband, R. W. Ahmad, and K. Bilal, "Sustainable cloud data centers: A survey of enabling techniques and technologies," *Renewable Sustainable Energy Rev.*, vol. 62, pp. 195–214, 2016.
[26] P. Niroj, "Live container migration: Opportunities and challenges," Aalto University, 2017. [Online]. Available: https://wiki.aalto.fi/download/attachments/116662239/live-container-migration%20%281%29.pdf?version=1&modificationDate=1481801946073&api=v2, Accessed on: Mar. 18, 2018.
[27] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Comput.*, vol. 40, no. 12, pp. 33–37, 2007.
[28] M. Callau-Zori, L. Samoila, A.-C. Orgerie, and G. Pierre, "An experiment-driven energy consumption model for virtual machine management systems," *Sustainable Comput.: Informat. Syst.*, vol. 18, pp. 163–174, 2018.
[29] J. O'Loughlin and L. Gillam, "Performance evaluation for cost-efficient public infrastructure cloud use," in *Proc. Int. Conf. Grid Econ. Bus. Models*, 2014, pp. 133–145.
[30] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *IEEE Trans. Comput.*, vol. 61, no. 4, pp. 563–577, Apr. 2012.
[31] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proc. 1st ACM Symp. Cloud Comput.*, 2010, pp. 39–50.
[32] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster Comput.*, vol. 16, pp. 249–264, 2013.
[33] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "A framework and algorithm for energy efficient container consolidation in cloud data centers," in *Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst.*, 2015, pp. 368–375.
[34] U. Pongsakorn, Y. Watashiba, K. Ichikawa, S. Date, H. Iida, et al., "Container rebalancing: Towards proactive Linux containers placement optimization in a data center," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf.*, 2017, pp. 788–795.
[35] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "An energy-efficient VM prediction and migration framework for overcommitted clouds," *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 955–966, Oct.–Dec. 2018.
[36] D. Meisner, B. T. Gold, and T. F. Wenisch, "The PowerNap server architecture," *ACM Trans. Comput. Syst.*, vol. 29, no. 1, 2011, Art. no. 3.

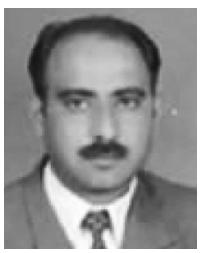
[37] E. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in *Proc. Int. Workshop Power-Aware Comput. Syst.*, 2002, pp. 179–197.
[38] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proc. 26th Symp. Operating Syst. Principles*, 2017, pp. 153–167.
[39] W. Smith, I. Foster, and V. Taylor, "Predicting application run times with historical information," *J. Parallel Distrib. Comput.*, vol. 64, no. 9, pp. 1007–1016, 2004.
[40] S. Di, D. Kondo, and W. Cirne, "Characterization and comparison of cloud versus grid workloads," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2012, pp. 230–238.
[41] Q. Zhang, J. L. Hellerstein, R. Boutaba, et al., "Characterizing task usage shapes in Google's compute clusters," in *Proc. 5th Int. Workshop Large Scale Distrib. Syst. Middleware*, 2011, pp. 1–6.
[42] H. Zhuang, X. Liu, Z. Ou, and K. Aberer, "Impact of instance seeking strategies on resource allocation in cloud data centers," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, 2013, pp. 27–34.
[43] S. F. Piraghaj, "Energy-efficient management of resources in container-based clouds," PhD dissertation, Dept. Comput. Inf. Syst., Univ. Melbourne, Parkville, Australia, 2016.
[44] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers," *Softw.: Practice Experience*, vol. 47, no. 4, pp. 505–521, 2017.
[45] J. Nider and M. Rapoport, "Cross-ISA container migration," in *Proc. 9th ACM Int. Syst. Storage Conf.*, 2016, Art. no. 24.
[46] M. J. Scheepers, "Virtualization and containerization of application infrastructure: A comparison," in *Proc. 21st Twente Student Conf. IT*, 2014, pp. 1–7.
[47] C. Yu and F. Huan, "Live migration of docker containers through logging and replay," in *Proc. Int. Conf. Mechatronics Ind. Informat.*, 2015, pp. 623–626.
[48] S. Nadgowda, S. Suneja, N. Bila, and C. Isci, "Voyager: Complete container state migration," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 2137–2142.

Ayaz Ali Khan received the MPhil (MS) degree in computer science from the COMSATS Institute of Information Technology (CIIT), Islamabad, Pakistan. He is currently working toward the PhD degree in the Department of Computer Science, Abdul Wali Khan University Mardan, 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.

Muhammad Zakarya received the PhD degree in computer science from the University of Surrey, Guildford, United Kingdom. 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.

Rajkumar Buyya received the PhD degree in computer science from Monash University. He is a 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, University of Melbourne, Australia. His research interests include cloud, grid, distributed, and parallel computing. He is a fellow of the IEEE.

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 interests include 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.



Mukhtaj Khan received the PhD degree in computer science from the Department of Electronic and Computer Engineering, Brunel University, United Kingdom. He is currently an assistant professor with the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. His research interests include big data analytics, smart grids, cloud computing, and distributed systems. Moreover, he owns deep understanding over the performance modelling of Hadoop systems.

Omer Rana received the PhD degree from Imperial College. He is a professor of performance engineering in the School of Computer Science & Informatics, Cardiff University and deputy director of the Welsh e-Science Centre. His research interests extend to three main areas within computer science: Problem solving environments, high performance agent systems and novel algorithms for data analysis and management. Moreover, he leads the Complex Systems Research Group, School of Computer Science & Informatics and is director of the "Internet of Things" Lab, Cardiff University.