BACKFILLME: An Energy and Performance Efficient Virtual Machine Scheduler for IaaS Datacenters

Muhammad Zakarya, Lee Gillam, Mohammad Reza Chalak Qazani, Ayaz Ali Khan, Khaled Salah, Omer Rana

Abstract—Backfilling refers to the practice of allowing small jobs to be completed ahead of schedule as long as they do not cause the first job in the line to wait. Users are expected to offer estimates of how long jobs will take to complete in order to make these decisions possible, and these projections are often based on historical data. However, predictions are very hard and may not be accurate, particularly in cloud computing scenarios where jobs or applications run on Virtual Machines (VMs). In addition, scheduling and consolidation techniques can improve the energy efficiency and performance of applications. Consolidation involves VM migrations that can have a negative impact on workload performance and users' costs. Backfilling can be used as an alternative technique for consolidation (short-term) and/or can be used along with consolidation (long-term). Backfilling methods are well-utilised in single computing systems, but are relatively unexplored in cloud resource allocation. A backfilling-based resource allocation and consolidation technique is proposed. Using real workloads from the Google cluster traces, we investigate the impact of backfilling on infrastructure energy efficiency and performance. For 12583 heterogeneous servers and approximately three million jobs that belong to three different applications, we observed that approximately 19% energy savings and 6% workload performance improvements are achievable using the backfilling approach. Furthermore, our evaluation suggests that using VM runtime as a criterion for the backfilling approach is approximately 3.56% – 7.78% more energy and 1.91% – 3.38% more performance efficient than using priority as a backfilling criterion.

Index Terms—Clouds, resource allocation, backfilling, service migration, energy efficiency, performance

1 INTRODUCTION

The energy crisis has led to a need for solutions to address increased energy usage, particularly due to the growing number of ICT devices and the depletion of non-renewable energy sources like coal and fossil fuels. Cloud computing has shown potential as an energy-efficient solution, providing on-demand, expanded network access, and quick elastic services to businesses and IT sectors [1], [2]. Experts have predicted that cloud computing could reduce global GHG emissions by 5.5% by 2025, with the Lawrence Berkeley National Laboratory's model suggesting an 87% reduction in energy use [3]. Google's recent report suggests that cloud computing could contribute to energy savings of 68% to 87% [4]. However, the energy consumption of cloud datacenters is still increasing, with an expected increase from 460TWh in 2022 to 1050TWh in 2026 due to the increase in mobile devices and the demand for the Internet. The Internet of Things (IoT), big data, artificial intelligence (AI), cryptocurrency, and the global pandemic have significantly increased data production and processing requirements [5]. Large energy consumption also reduces the profit that can be achieved from various datacenter services, especially with increasing energy prices. Techniques like resource allocation, scheduling, and consolidation can help overcome these problems, but consolidation of virtual machines (VMs) may be costly due to extra energy consumption, decreased workload performance, and increased user-related costs. Despite numerous VM placement techniques suggested in the literature, there is still a gap to further optimize datacenters in terms of energy consumption, workload performance, and users' costs [6].

- *M. Zakarya and M. Qazani are with the Faculty of Computing and Information Technology, Sohar University, Sultanate of Oman. M. Zakarya is with the Department of Computer Science, Abdul Wali Khan University, Pakistan. L. Gillam is with the Department of Computer Science, University of Surrey, UK. A.A. Khan is with the Department of Computer Science, University of Lakki Marwat, Pakistan. K. Salah is with Kahlifa University, UAE. O. Rana is with Cardiff University, UK.*
E-mail(s): MZakarya@su.edu.om, mohd.zakarya@awkum.edu.pk, l.gillam@surrey.ac.uk, MQazani@su.edu.om, ayazkhan@ulm.edu.pk, khaled.salah@ku.ac.ae, ranaof@cardiff.ac.uk

Corresponding authors: M. Zakarya

Backfilling is the methodology for running jobs out of order, even if sufficient resources are not available for some particular jobs [7], [8]. The scheduler continues monitoring the queue and selects certain jobs that can be accommodated with the available resources. However, in essence, these jobs must not affect the running jobs on the servers. Such decisions are taken while accounting for the jobs' execution time (either provided by the users or predicted from historical data). There are three major types of the backfilling approach: (i) EASY-backfilling; (ii) Conservative backfilling; and (iii) fattened backfilling [9], [10]. Conservative backfilling and EASY-backfilling vary primarily in how they handle resource limits, choose jobs, use preemption and suspension tactics, and prioritize tasks. Conservative backfilling adopts a reservation-based strategy, prioritizes higher-priority tasks, and seeks to cause the least amount of interruption to longer-running projects. The EASY-backfilling method, on the other hand, promotes early task execution, permits resource overbooking (resource over-subscription), and is more tolerant of work priorities, even if they are lower [11]. If short tasks do not cause the first job in the queue to wait longer than the average wait time for the jobs that have previously been completed, the fattened backfilling permits them to advance. The VM allocation system's unique objectives, needs, and restrictions determine which of these methods should be used [12]. At a system level, the backfilling approach has been evaluated, but in terms of VM resource allocation and clouds, its effectiveness has not yet been evaluated. Moreover, besides runtimes, other aspects of the VMs and applications, such as priorities, should be considered in VM placement decisions.

Despite the fact that there are algorithms that, in theory, provide the best performance, in practice, these algorithms are too complicated to be used in actual production systems. Most of these algorithms, as discussed in Sec. 6, are focused on VM placement and consolidation that involve migrations. From simple heuristics to more advanced placement algorithms, using evolutionary methods and predictive models, migrations affect the performance of workloads [4], [5], [13], [14]. Backfilling is the most direct and practical method for achieving efficient and equitable scheduling without VM migrations. In fact, a lot of actual production systems employ backfilling [15]. Due to the performance-driven nature of

the majority of scheduling algorithms, scheduling rules have been extensively examined in the literature (Sec. 6), but there is still more study to be done in this area to improve energy efficiency and workload performance. In this paper, we propose a backfilling-based VM allocation and consolidation technique that accounts for the immediate placement of certain types of VMs based on their runtimes and priorities, such that energy consumption is reduced while the performance of other running applications is not affected. We observed an existing trade-off between energy consumption and performance; however, for certain types of VMs (applications), the backfilling approach can reduce energy consumption and improve performance. Furthermore, we also noted that backfilling can significantly reduce the total number of migrations and, therefore, the performance of the applications is not negatively affected. The major contributions and findings of our research are:

- we propose an energy and performance aware VM placement algorithm that accounts for VMs or applications' previous runtimes and priorities;
- an energy and performance aware consolidation technique is suggested that uses the notion of backfilling scheduler for certain types of VMs or workloads; and
- a backfilling strategy is suggested that chooses the most appropriate VMs for placement.

The rest of the paper is organized as follows: In Sec. 2, we model the problem. In Sec. 3, we discuss the methodology and propose several algorithms for VM allocation and consolidation. Furthermore, several statistical models are discussed to model the energy and performance of cloud resources. In Sec. 4, we deliberate the proposed backfilling technique that allocates VMs when there are enough resources available to run them. We validate the proposed scheme using real workload traces from Azure clusters in Sec. 5. We offer an overview of the related work in Sec. 6. Finally, Sec. 7 concludes the paper and describes several directions for future research.

2 Problem Description

Today's batch work schedulers for parallel supercomputers all employ default algorithms that are somewhat interchangeable. Basically, they run each task until it is finished and choose which ones to execute in first come, first served (FCFS) order. Because workloads do not pack optimally and processors are not being used, this too-simplified technique results in substantial fragmentation [16], [17]. Furthermore, the rate at which applications' requests arrive α and the rate at which the application execution finishes π are strongly correlated with each other and affect the allocation. Most schedulers employ backfilling, which requires monitoring the queue for smaller jobs that can use the available resources if the next queued job cannot be executed owing to an absence of available processors [18]. The backfilling strategy was tested for task scheduling while taking runtimes into consideration. However, its efficacy in VM allocation has not yet been investigated. Furthermore, its efficacy is measured in terms of execution times, but other factors, such as priorities, and criteria to backfill, remain mostly ignored.

The optimization problem of packing VMs to minimum servers can be seen as a bin-packing issue, where VMs are items and servers are bins. It is possible to conceptualise VM allocation and consolidation as bin-packing problems that are NP-hard [19]. For NP-hard problems, there are no optimal solutions and different heuristic approaches such as first fit, worst fit, etc. are considered suitable [20]. Therefore, we also assume the VM allocation as an NP-hard optimization problem. Bin-packing, as used in the context of virtualisation, describes the effective distribution of VMs among physical hosts while taking into account resource limitations. The objective is to reduce the number of physical hosts needed while making sure that the allocated VMs do not use more resources than the hosts can provide. In general, resources such as CPU, memory, storage, and network bandwidth are taken into account. Assume that we have m VMs and n physical hosts. To express whether a VM is assigned to a given physical host, we can use binary decision variables. Suppose x_{ij} is a binary variable such that Eq. 1 satisfies:

$$x_{ij} = \begin{cases} 1 & \text{if virtual machine } i \text{ is allocated to physical host } j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The aforementioned allocation problem is constrained in that the overall resource utilisation of the allotted VMs on a particular host must not be greater than the host's capacity. In this paper, we account for two resources i.e. CPU and memory (M). Other host's resources such as storage and network may also be considered. The following Eq. 2 describes the capacity restrictions for each physical host denoted by j:

$$\sum_i (x_{ij} \cdot \mathrm{CPU}_i) \leq \mathrm{CPU}_j \;\; and \;\; \sum_i (x_{ij} \cdot \mathrm{M}_i) \leq \mathrm{M}_j \tag{2}$$

where, CPU_i and $Memory_i$ represent the respective resource requirements of VM_i. Furthermore, there must be precisely one physical host assigned to each VM. This allocation restriction can be written mathematically as follows in Eq. 3:

$$\sum_j x_{ij} = 1 \quad \text{for all } i \tag{3}$$

Additional limitations can be imposed to guarantee compatibility if some VMs have particular needs or dependencies that must be met by compatible hosts. The goal is to employ the fewest possible physical hosts for VM allocation. Therefore, the objective function is what we may define as given in Eq. 4:

$$\text{Minimize} \quad \sum_j y_j \tag{4}$$

In this case, the binary variable y_j represents the usage of the physical host j, or the allocation of at least one VM to it. The objective function sums up all physical hosts' y_j variables and ensures reduced energy consumption by running all VMs on minimum resources. This is important to remember that the bin packing issue is NP-hard, which means that finding the best and optimal solution for complicated situations may be computationally very difficult or even impossible. Heuristics and approximation algorithms are therefore frequently employed to efficiently identify close-to-optimum solutions. By defining VM allocation as a bin packing issue and using proper algorithms, we may maximise the use of physical resources or reduce the total number of hosts, which subsequently increases the energy efficiency of virtualised environments.

The above mathematical model can be extended for accounting the backfilling approach in resource allocation. The goal is to allocate limited resources to different VMs in such a way that the total cost (in terms of energy consumption, performance, users' monetary costs) or number of resources is minimized while considering the possibility of periodically accounting for resource over-usage through the concept of backfilling approach. Let us suppose that we have N number of VMs and a set of available resources denoted by R. Each resource $r \in R$ has a certain user's cost from the provider which is denoted by C_r. Furthermore, C_i denotes the user's cost (in terms of all offered resources) associated with VM_i and U_{ir} represents the amount of resource r that are used by VM_i without considering the backfilling approach. Similarly, O_{ir} characterizes the over-usage of resource r by VM_i subject to the backfilling approach and B_{ir} denotes the backfilling cost (in terms of performance lost) associated with the over-usage of resource r by VM_i . Then, the amount of resources r that are allocated to a particular VM_i are denoted by x_{ir}. Therefore, the objective function

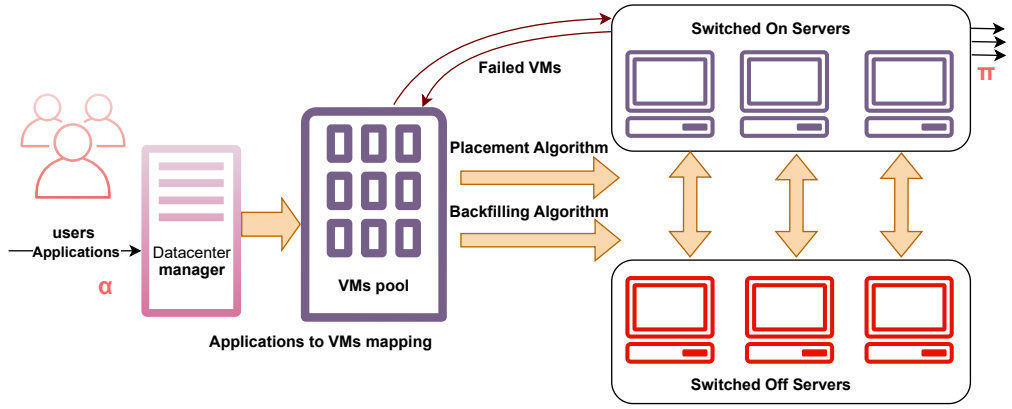

Fig. 1: Allocation and Migration of VMs – The rate at which applications requests arrive is denoted with α and the rate at which the application execution finish is denoted by π (Backfilling increases utilisation, migrations, and number of switched on and off hosts)

of the allocation problem can be mathematically expressed as given in Eq. 5.

$$\text{Minimize} \quad \sum_{i=1}^{N} \left(C_i + \sum_{r \in R} (x_{ir} \cdot B_{ir}) \right) \tag{5}$$

subject to three different constraints: (i) each task must receive a certain amount of required resources as given by Eq. 6, where U_i is the required amount of resources for VM_i ; (ii) resource allocation must not exceed the availability of resources as given by Eq. 7; and (iii) resource allocation and over-usage must be non-negative as given by Eq. 8.

$$\sum_{r \in R} x_{ir} = U_i \quad \forall i = 1, \ldots, N \tag{6}$$

$$x_{ir} \leq U_{ir} + O_{ir} \quad \forall i = 1, \ldots, N, \forall r \in R \tag{7}$$

$$x_{ir}, O_{ir} \geq 0 \quad \forall i = 1, \ldots, N, \forall r \in R \tag{8}$$

These constraints can be updated to make advanced VM placement accounting for colocation, performance, and workloads [13].

3 Methodology

In conventional cloud datacenters, VM consolidation is often carried out using static policies or heuristics that have been specified. These methods, however, frequently fall short in their ability to adjust to the dynamic nature of cloud workloads, leading to inefficient resource allocation and utilisation. In [21], VM consolidation is approached probabilistically, allowing for more informed and flexible VM placement choices. This section describes our adapted research methodology from [21] in detail. In Sec. 3.2, we discuss the VM placement, migration, and consolidation techniques. In Appendix. A, we deliberate statistical models to measure the energy consumption of resources and the performance of the workloads. The proposed backfilling technique is discussed in Sec. 4.1.

3.1 Implementation

In this paper, we assume that the scheduler is responsible for both initial placement and backfilling decisions and is working in a centralised way. For initial placement, when there is a VM request the scheduler finds out a suitable host. In the case of backfilling, the scheduler runs periodically to optimize the current status of the cluster. To overcome the limited scalability and single point of failure, this implementation methodology can easily be extended to a decentralised approach where the entire cluster of resources is divided into subclusters where each cluster has its local scheduler [22], [4]. The initial placement can be done on a local scheduler (for each cluster) while backfilling can be implemented by a global scheduler (working on top of all local schedulers) as it has the information of all clusters and their resources. Independent optimization of VM placement in different clusters, without coordination among local and global schedulers, may lead to missed synergies and therefore to sub-optimal outcomes [22]. Usually, decentralised approaches have additional costs in terms of communication overhead, inefficient resource utilisation, increased delay in decision-making, and complexity of resource management. Furthermore, service providers prefer quickness in decision-making over optimal results [23]. Therefore, we use a centralised approach for the implementation of the proposed algorithms.

3.2 Allocation and Migration of VMs

The statistical methods used to assign VMs to hosts in the datacenters and the respective dynamic migration are discussed in this section. The scenario of VM assignment and migration is shown in Fig. 1. In the figure, initially, the application request is submitted by the client to the datacenter manager [24], [25]. Afterward, a VM is selected relevant to the application in terms of properties like resources required, i.e. memory, CPU, storage, etc., and the nature of the operating system as per the client's request. Once this is done, the assignment procedure assigns the VM to the host available in the pool of multi-core hosts (that are usually virtualised through hypervisors). The nature of the workload is dynamic for the application, i.e., resource requirements keep changing over time, therefore, keeping the maximum capacity of the VM intact [26], [27]. It is evident from the example where we have Web servers. The CPU requirement is dependent on the generation of workloads from Web users. On a periodic update, CPU usage at each host (server) is analyzed to remain within the upper and lower threshold values. If it diffracts from the given threshold values, the migration procedure is invoked, where the current VM is assigned to a new host. The parameters α and π, as provided in Fig. 1, define a request for application and rate of service for the host's resources, respectively. They are further used in Sec. 3 to analyze the performance of the allocation techniques.

3.2.1 *The Allocation Algorithm*

As soon as the client application is connected to a relevant VM, it is forwarded for execution on a host. The following points must be taken into consideration: (i) assignment of the VM is preferred to a host with the highest CPU usage so that consolidation may be achieved of VMs to switch off the idle hosts; (ii) the maximum host capacity may not reach demanded CPU usage where SLA violations will occur due to increased workload; and (iii) if switching a host back on is mandatory for a VM, it may be allowed under strict requirements, but it eventually reduces host consolidation and will increase power usage [21].

Based on the objectives defined above, the VM assignment procedure is described as follows: A broadcast by the datacenter manager

is beaconed for requests to be assigned to hosts. The active host in the pool performs the Bernoulli trial where the probability of success is dependent on current CPU usage, π (ranging from 0 to 1), and maximum use allowed T_n [18]. The function assignment probabilistic function, $fassign(\pi)$, will have a null value if $\pi > T_n$, on the contrary, it will be determined as given in Eq. 9:

$$f_{assign}(\pi) = \frac{1}{M_p} \cdot u^p \cdot (T_a - \pi) \tag{9}$$

The above Eq. 9 illustrates specific values of integer parameter q using function graph and T_a = 0.9. The maximum value of 1 is achieved by normalization using factor $1/Mp$. The CPU usage is restricted using the defined function. It is made that threshold T_a value is not exceeding (as no further VMs are assigned if the value of π is reaching the threshold value) and favor consolidation, preferably VMs assignment is on a high loaded host. The function touches maximum using the value of π, i.e. assignment attempt is successful at this value with higher probability – as given by $q/(q+1)$. The T_a value gradually increases with an increase in the q value. So, the q value modulates the function shape, and as a result, the consolidation process is tuned. Further details of this method can be found in [18], [21].

If Bernoulli trials succeed for a particular host, it replies to the beaconed broadcast message by showing its willingness to accommodate the VM. Then, the datacenter manager performs a selection of a host from the available hosts. This selection is based on the energy efficiency of the host and runtimes of the VMs running on it [23]. The host with minimum relative runtimes of VMs is selected for allocation as described in Alg. 3. On the other hand, if hosts are unavailable due to their utilisation reaching out the threshold value T_a, or simply Bernoulli trials are not successful, then idle hosts are switched on and will accommodate all required VMs. Suppose the option of the idle host is not available, which means all the hosts are active. In that case, VMs are accommodated, using resource over-subscription, forcibly to any of the hosts with a fraction of CPU available (this host is selected in the second round of broadcast requests), contrarily, will be placed in a waiting queue: gives a clue that host numbers are too low to hold the load. However, over-subscription can degrade the host performance for all running VMS, therefore these VMs are stored in a waiting queue for backfilling.

The significant advantage of the methodology is realized by the fact that it is self-organizing with decentralised in nature as the decisions are primarily taken locally. To accommodate incoming VMs randomly, the datacenter manager should know about active hosts and/or switch them off. However, it is not required by the manager to decide how VMs are placed onto the host or keep updated state information of the host.

Algorithm 1: The VM Allocation Algorithm

```
Input: VM list, Host list, Failed VMs Queue Q
X ← Range from 0 – 1 relative resource usage ;
T ← Maximum allowed usage ;
P ← Shape parameter ;
M_p ← Factors to normalize maximum value to 0 - 1 ;
Output: Allocate VMs
if Q ≠ NULL then
    Use Alg. 3 to backfill VMs ∈ Q ;
end if
for every VM ∈ VM list do
    for every host ∈ Host list do
        if host utilisation ∈ X and can accommodate the VM
         then
            Perform Bernouli Trial using Eq. 9 ;
            if Bernouli Trial ← Success then
                host ← VM ;
                else
                    add VM to Q ;
                end if
            end if
        end if
        if host capacity > max(X) then
            host_idle ← VM ;
            else
                add VM to Q ;
            end if
        end if
    end for
    Use Alg. 3 to backfill VMs ∈ Q ;
end for
```

3.2.2 The Migration Algorithm

The assignment process helps VMs to be placed over a smaller possible number of hosts, as given in the performance analysis section. On the other hand, though, there is a possibility that a few hosts may be underutilised and could be turned off to save energy [23]. Therefore, if VMs are allocated efficiently to hosts i.e. they finish their execution on a particular host or possibly their request for the host resources is reduced. Additionally, the host may become overloaded. Truthfully, VM allocation to a host is based on the requested need of CPU, but an increase in workload is seen for the same host for other VMs [4]. The scenario creates SLA violations where the extent of dependability for a datacenter along with QoS offered to end-users is affected. In both cases, to gain profits, VMs are migrated to other hosts to either switch off a host or distribute its workload.

Live migration is carried out by migration procedure for VMs [28]. On the contrary, various other methods for VM migration (as discussed in Sec. 6) compared to [21], the technique in [21] is self-organized and guarantees a steady and continual migration process. Randomly, at every interval, all the hosts look for under-utilisation and over-utilisation. If it takes place, then it computes the respective migration probability functions, as given in Eq. 10 and Eq. 11, and denoted by $f^{low}_{migrate}(\pi)$ and/or $f^{up}_{migrate}(\pi)$, respectively [18], [21]:

$$f^{low}_{migrate}(\pi) = (1 - \pi/T_{low})^{\gamma} \tag{10}$$

$$f^{up}_{migrate}(\pi) = (1 + \frac{\pi - 1}{1 - T_{up}})^{\delta} \tag{11}$$

In both cases, the host executes a Bernoulli trial, and a migration decision is taken on local VMs. Fig. 1 demonstrates the function's definition, where VMs migration initiation is done if CPU utilisation lies below the lower threshold T_{low} or it is exceeding the upper threshold T_{up}. The migration process is withdrawn if the CPU utilisation lies between both thresholds.

The function shape is modulated using parameters γ and δ that are latterly used to decide whether to perform the migration or not [18], [21]. A complete demonstration is detailed in Sec. 3.2 for the successful migration process of VMs through the assignment procedure. If a host is overloaded (90%), the threshold value of T_a for the assignment function is 0.9 times the CPU usage of the current host. It makes sure that migration of VMs is done to hosts having less load and restricts VMs from being migrated from overloaded hosts to new hosts. The updated value of T_a is broadcasted to all hosts with a potential migration request, and the VM is allocated to an available host. If there is more than one host, then the most energy-efficient host with minimum relative runtimes of VMs is selected, as described in Alg. 3. In case there are no hosts available, then the VM is retained at the originating host.

Algorithm 2: The VM Consolidation Algorithm

Input: Allocated VM list, Host list
$T_{low} \leftarrow$ lower threshold value ;
$T_{up} \leftarrow$ upper threshold value ;
$List^{VM}_{host} \leftarrow$ List of VMs running on host ;
$Migration_{map} \leftarrow$ List of suitable VMs to migrate ;
Output: $Migration_{map}$
for *each optimization round* **do**
  for *every host* $\in$ *Host list* **do**
    Use Eq. 10 and Eq. 11 to analyze host ;
    if *host* $\leftarrow$ *available* **then**
      Evaluate α and β ;
    end if
    if $T_a > T_{up}$ *or* $T_a < T_{low}$ **then**
      for *each VM* $\in List^{VM}_{host}$ **do**
        Compute the relative resources of VM using Eq. 12 ;
        $VM \leftarrow$ Choose VM with high relative resources ;
        $Migration_{map} \leftarrow VM$;
      end for
    end if
  end for
  Use Alg. 1 to allocate VMs $\in$ migration map ;
end for

4 Proposed Solution

4.1 The Backfilling Algorithm

The backfilling mechanism is illustrated in Fig. 2. The traditional FCFS algorithm assigns VMs resources in a first come, first served fashion; therefore, creating stranded resources [29]. Furthermore, the first queued VM in FCFS may starve as a result of succeeding VMs continuously jumping over it. The workaround is to reserve this VM and only allow it to be executed again if the other VMs respect it [7]. The key distinction between backfilling and consolidation strategies is that backfilling seeks for VMs in the queue, whereas consolidation operates on a regular basis to balance the load across all servers depending on predetermined utilisation levels. The backfilling approach is an effective method for allocating VMs to hosts that helps to maximise resource utilisation and overall effectiveness of virtualised environments. Furthermore, backfilling enables the placement algorithm to make use of idle resources that are present in the system when a high-priority VM request is received, as opposed to having to wait for the full allocation of resources. It locates underused resources or partially populated nodes and assigns the high-priority VM to them in order to prevent any resources from sitting idle for an extended length of time. With this method, the system's total throughput is increased and the waiting time for VM allocation is significantly decreased.

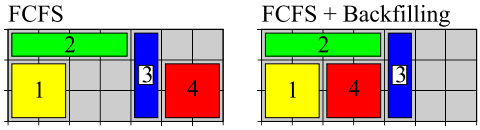

Fig. 2: The Backfilling Approach using FCFS – the shaded area denotes host resources such as CPU (cores) and memory (size) and the colored area with numbers denotes VMs and the required host's resources – If VM 4 had needed more than 2 CPU cores, the reservation for VM 3 would have been breached, making it impossible to backfill VM 4 [7]

The proposed backfilling mechanism is illustrated in Alg. 3. The technique will backfill all the VMs based on either their runtimes or priorities. We assume that task priorities or application types are defined by users. We can also use various statistical methods to guess the type of applications from VMs usage activity [6]. Similarly, the runtimes of the applications are based on a probabilistic technique rather than actual prediction or user-defined values. Google trace data shows that long-running jobs are those that run for more than an hour [29]. In other words, the investigation shows that the jobs run for more than an hour (past runtime) and continue to run for a day (future runtime). Therefore, we assume that VMS with longer previous runtimes would be more likely to run for a longer duration in the future. In this way, we will be able to allocate long-running VMs to hosts that already have VMs running with similar runtimes.

Algorithm 3: The VM Backfilling Algorithm

Input: $Q \leftarrow$ Alg. 1, Host list
Estimate priorities and runtimes of all VMs $\in Q$;
Sort Q in increasing order of their runtimes ;
Sort Q in decreasing order of their priorities ;
for *each VM* $\in Q$ **do**
  Compute the VM previous runtime R_{past} or R;
  for *each host* $\in$ *active hosts* **do**
    if *host has enough capacity to accommodate VM* **then**
      $L \leftarrow$ List of all VMs on *host* ;
      $H \leftarrow$ Compute relative runtimes for all VMs $\in L$ using Eq. 12 ;
    else
      exit and pick the next VM $\in Q$;
    end if
  end for
  host $\leftarrow min(H)$;
  host $\leftarrow$ VM ;
end for
Output: Allocate VMs

In fact, the past runtimes of VMs, denoted by R_{past}, on a particular host are known to us, therefore, we can statistically derive relative runtimes, denoted by R_{rel} that can guide us to a most suitable host for the placement. For example, suppose we have n VMs running on a certain host (H), and we designate their individual runtimes as $H = \{R^{VM_1}_{past}, R^{VM_2}_{past}, ..., R^{VM_n}_{past}\}$. We determine the relative execution timings for each VM in order to compare the runtimes of various VMs running on different hosts. To do this, one method is to divide the execution time (R_{past}) of each VM by the minimum runtime of the VM among all VMs, denoted by set (H). There may be alternative statistical ways to calculate the relative runtime duration, however, we believe this method is the easiest. The following Eq. 12 is used to determine the relative runtime duration for each VM:

$$R^{VM_i}_{rel} = \frac{R^{VM_i}_{past}}{min(H)} \tag{12}$$

where $min(H)$ denotes the VM that has the minimum R_{past}. We do the above computation for all available hosts and then select the one that has a VM with the shortest duration (for short-running workloads) or a VM with the longest duration (for long-running workloads). This process ensures that resources of an appropriate host are provisioned to the VM or application, therefore, leading to the possibility of switching off hosts to reduce energy consumption. For example, if there are four VMs out of which two are short-running (suppose 2 hours) and the other two are long-running (suppose 10 hours). If short-running VMs are placed on a separate host and long-running VMs are placed on another host, it means that after two hours, one host can be switched off. However, if these VMs are placed randomly, then both hosts will run for at least ten hours consuming large energy. To see whether there are backfilling chances, the backfilling strategy may either be performed on a regular basis or once a work is finished or withdrawn from the system.

4.2 Computational Complexity

The computational complexity of the VM allocation algorithm (Alg. 1) depends on the total number of VMs (m) and total number of servers (m). The worst case occurs when a VM is not allocated to all available hosts. The worst-case complexity is, therefore, $O(mn)$. For the consolidation technique (Alg. 2), the worst case occurs when all migratable VMs (m') are not allocated to the available resources (n'). The worst-case complexity of Alg. 2 is, therefore, given by $O(m'n')$. Since, m' and n' belong to subsets of m and n; therefore, the overall complexity can be described as $O(mn)$. The backfilling approach (Alg. 3) comprises the sorting steps, which generally increases its worst-case time complexity to $O(mn.log(mn))$.

5 Performance Evaluation

Our simulated datacenter comprises 12583 heterogeneous servers that belong to five different CPU architectures in an equal amount. The CPU architecture represents heterogeneity in terms of performance and energy consumption, as shown in Table 1. Moreover, we implemented five different scheduling policies, i.e., FCFS, FF (first fit), BRS (best resource selection) [30], ecoCloud [21], and FillUp, and then extended the FCFS and FillUp policies with the backfilling mechanism leading to two other scheduling policies, i.e., FCFS+Backfill and FillUp+Backfill. We also provide a comparison with recent techniques: (i) SAF+Backfill technique which sorts all jobs according to their "area" or "geometry" [31]; and (ii) Shortest Gap - Priority-Based Fair Scheduling (SG-PBFS) which is a backfilling technique that attempts to manipulate the gaps in the schedule of cloud jobs [32]. We assume that migrations are enabled in the datacenter, and the consolidation policy migrates VMs when certain servers either exceed or fall behind predefined threshold values for their resource (CPU) utilisation. Moreover, we assume five different types of VMs that mimic instances of Amazon Web Services (AWS), as shown in Table 2. We also assume three different types of application or workload, i.e., W1, W2, and W3. Note that each workload comprises more than a million tasks, each of which has specific needs for hosts' resources such as CPU cores, memory, and runtime, among other things. Additionally, we assume that every task makes use of the stochastic utilisation model that is part of the classical CloudSim simulation toolkit [33]. We use PerficientCloudSim [34] which is an extended version of CloudSim [33] to run all the experiments. PerficientCloudSim provides support to model the performance degradation due to co-located VMs on a particular, as well as, resource contention. These tasks and their characteristics, such as arrival time, schedule time, and run times, are related to the Google cluster dataset [29], [35]. We treat a workload as a whole, treating all tasks as a single application whose total execution duration equals the workload runtime. The execution time is then used to compare the performance of various scheduling policies. The price of each application is then measured based on the execution time of the application and the type of VMs that were allocated to run the workload.

TABLE 1: Various characteristics of simulated servers

CPU model	Speed (MHz)	No of cores	No of ECUs	Memory (GB)	P_{idle} (Wh)	P_{max} (Wh)
E5430	2,830	8	22.4	16	166	265
E5-2620	2,000	12	24	32	70	300
E5645	2,400	12	28.8	16	63.1	200
E5-2650	2,000	16	32	24	52.9	215
E5-2670	2,600	16	41.6	24	54.1	243

5.1 Evaluation Metrics

We consider the total number of migrations, energy consumption (KWh), and workload execution time (in minutes) as performance metrics to evaluate various resource allocation, consolidation, and management techniques. In addition to this, we also study the VM waiting time and the overall throughput of the system. The waiting time is the duration between the submission time and schedule time of a VM request, while throughput is the total number of VMs that completed their workloads' execution within a specific time. In order to present the trade-off between several metrics, such as energy consumption, performance, and the number of migrations, we use a composite measure, denoted by N_T, which is computed using the weighted sum approach, that integrates these data in a way that makes the trade-off easy to assess. A lower value for N_T indicates better performance in terms of the combined criteria. The formula is given in Eq. 13:

$$N_{EP} = E_w \times E_n + R_w \times R_n + M_w \times M_n \tag{13}$$

where E_w, R_w, and M_w denote the weights while E_n, R_n, and M_n characterises the normalized values for energy consumption, performance (in terms of runtimes), and the number of migrations, respectively. The weights can be adjusted by the IaaS to give priority to one objective over others and vice versa.

5.2 Experimental Results

Table 3 shows the outcomes that were obtained for various resource allocation, consolidation, and backfilling techniques. For W1 and W2 workloads, the FCFS and FillUp techniques have a trade-off for workload performance and energy consumption that has been balanced with the backfilling policy. For the W1 workload, the FCFS consumed 3290.04KWh and performance was 449.68; however, FCFS+Backfill consumed 2978.32KWh and performance was 452.73. Similarly, the FillUp consumed 3253.3KWh and the performance was 450.78; however, FillUp+Backfill consumed 2824.89KWh and performance was 450.87. The FillUp approach performs better than the classical FCFS, FF, ecoCloud, SAF+Backfill, SG-PBFS, and BRS techniques. The backfilling ensures that hosts are always within their utilisation levels, which significantly reduces the total number of migration occurrences. The BRS approach always tries to migrate and results in high energy consumption and performance loss. This means that consolidation is expensive in terms of energy consumption and application performance. The severity of these impacts and trade-offs varied for various workloads and scheduling practices used. For long-running workloads (W3), approximately 19.23% energy consumption and 2.97% performance gains were achieved using the backfilling policy. However, the backfilling did not produce any significant reductions in energy consumption and workload performance when the datacenter was kept highly utilised. For example, when utilisation levels were between 70% and 80%, the FillUp and FillUp+backfill produced almost similar outcomes. Furthermore, ecoCloud, SAF+Backfill, SG-PBFS, and FillUp+backfill produced comparable outcomes when the datacenter resources were utilised between 30% – 45%. Despite these outcomes, due to random resource utilisation of VMs, FCFS, ecoCloud, SAF+Backfill, BRS, and FF algorithms were also observed to overlap their achieved energy consumption savings, migrations, and performance outcomes. As shown in Table 3 (**bolded**), the trade-off between energy consumption, performance, and number of migrations is well balanced using the proposed FillUp + Fill algorithm.

In Table 4, experimental outcomes are shown for various workloads when they are assumed to run simultaneously. The results are mostly in line with the previous outcomes in terms of the existing trade-off between energy consumption and performance; however, mixed workloads create large migration opportunities. The backfilling strategy achieves the appropriate balance based on real-time workload needs by dynamically modifying resource allocation to optimize application performance and energy consumption. We also observed that the criteria used to backfill VMs have a significant impact on the energy savings and performance

TABLE 2: Amazon different instance types and their characteristics

Instance type	No of vCPUs	No of ECUs	Speed (MHz) MIPS	MEMORY (GB)	Storage (GB)	Reserved price (1 year) ($/hour) US East - N. Virginia
t2.nano	1	1	1,000	0.5	1	0.006
t1.micro	1	1	1,000	0.613	1	0.02
m1.small	1	1	1,000	1.7	160	0.044
m1.medium	1	2	2,000	3.75	410	0.087
m3.medium	1	3	3,000	3.75	4	0.067

TABLE 3: Experimental results in terms of energy consumption (E), performance (P), and total number of migrations (M) for three different types of workloads [Energy is measured in KWh and lower values are better than high, Performance is the inverse of the execution time expressed in minutes and lower values are better than high] – approaches with the lowest values for N_T metric are best in balancing the trade-off between various objectives

Scheduling technique	W1				W2				W3			
	E	P	M	N_T	E	P	M	N_T	E	P	M	N_T
FCFS	3290.04	449.68	437	0.94	5308.47	779.2	231	0.967	7116.79	993.56	201	0.985
FF	3253.82	451.33	431	0.933	5274.01	768.1	245	0.964	7008.96	1002.56	193	0.975
BRS	3341.01	491.32	512	0.989	5327.54	799.8	300	1	7078.72	1012.43	221	0.997
FillUp	3253.3	450.78	401	0.942	4919.37	781.6	191	0.904	6991.88	927.9	167	0.94
ecoCloud [21]	3176.65	451.46	502	0.941	4998.6	792.48	203	0.933	6893.6	977.59	109	0.92
SG-PBFS [32]	3338.55	476.1	477	0.987	5002.7	789.26	188	0.927	6903.66	986.79	179	0.956
SAF+Backfill [31]	3301.69	461.33	399	0.948	5281.06	777.21	289	0.981	7101.7	985.79	219	0.988
FCFS+Backfill	2978.32	452.73	223	0.828	4992.45	780.3	125	0.897	6810.2	952.57	88	0.896
FillUp+Backfill	2824.89	450.87	202	**0.792**	4627.07	781.8	99	**0.84**	6772.63	911.89	67	**0.872**

of the workloads. For example, when the runtime of VMs is used as a criterion to backfill VMs; then the energy savings and performance values are more than those which are achievable when VMs priorities are considered to backfill VMs. For example, the FillUp+Backfill approach using priority consumed 5722.12KWh and the performance was approximately 499.45. The same approach with runtime backfilling criteria consumed 5401.96KWh and the performance was approximately 438.67. Similarly, for mixed workloads the SG-PBFS approach using priority consumed 8236.7KWh and the performance is approximately 1107.32. The same approach with runtime backfilling criteria consumed 6967.88KWh and the performance is approximately 1018.99. Other methods including ecoCloud, In fact, workloads with the same priorities have the same demands for CPU and other resources, which lead to resource contention and performance degradation. This should be noted that we modelled resource contention and its performance impact in our simulator PerficientCloudSim. Further details on various statistical models for servers and workloads performance impacts can be found in [34]. This means that the order of VMs in which they are scheduled affects the infrastructure energy efficiency, total number of migrations, as well as, the workloads' performance. As shown in Table 4 (**bolded**), for both priority and runtime-based backfilling, the trade-off between energy consumption, performance, and the number of migrations is balanced well using the proposed FillUp+Backfill algorithm (having the least values for N_T).

Migrations are expensive, and they have costs in terms of energy consumption and performance loss, in particular, when VMs have similar workloads and compete for similar resources i.e. co-located on the same physical hosts. Therefore, resource allocation policies should be used to replace the consolidation of VMs. To study this scenario, we compared FF and FillUp scheduling techniques while accounting for consolidation, no-consolidation, and backfilling mechanisms as shown in Fig. 3 and Fig. 4. For short-running workloads, consolidation degrades performance, leading to high energy consumption, and increases users' costs. For short-running workloads, we believe that the number of migrations is significantly high due to frequent termination of VMs and making opportunities for consolidation. However, for long-running workloads, consolidation saves a significant amount of energy consumption with trivial performance impacts. However, in a heterogeneous environment, if longer workloads are assigned to servers that consume high energy; then, the energy savings may not be high. Our investigation suggests that the backfilling approach has trivial performance impacts, and a significant amount of energy can also be saved. Therefore, the decision between consolidation and allocation rules ultimately comes down to the workload characteristics, performance needs, resource utilisation goals, cost considerations, and system or infrastructure goals at large. This should be noted that prior to choosing a choice, it is essential to thoroughly evaluate these variables and weigh the trade-offs. The backfilling mechanism may improve system performance and lower VMs' waiting times by dynamically changing the scheduling order and effectively using available resources for resource provisioning.

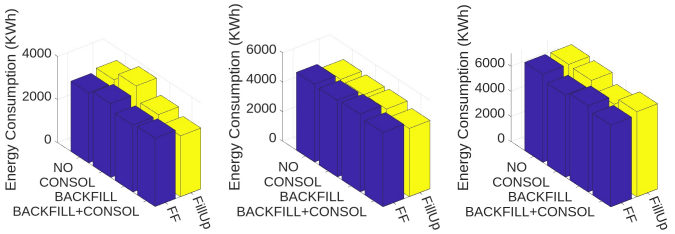

Fig. 3: Energy consumption for various scheduling and consolidation techniques using three different workload types [W1, W2, W3 – from left to right]

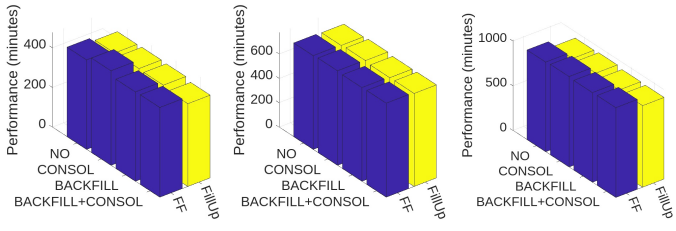

Fig. 4: Performance of various scheduling and consolidation techniques using three different workload types [W1, W2, W3 – from left to right]

As shown in Fig. 5 (left), backfilling improves the datacenter's throughput and reduces the VMs' waiting times. In our experiments, we used the submission and scheduling times of VMs as

TABLE 4: Experimental results in terms of energy consumption (E), performance (P), and the total number of migrations (M) for different workloads running simultaneously using various scheduling and backfilling methods [Energy is measured in KWh and lower values are better than high, Performance is the inverse of the execution time expressed in minutes and lower values are better than high] – approaches with the lowest values for N_T metric are best in balancing the trade-off between various objectives

Scheduling technique	Backfill method	W1+W2				W2+W3				W1+W2+W3			
		E	P	M	N_T	E	P	M	N_T	E	P	M	N_T
FCFS	Runtime	6598.51	528.88	568	0.987	6302.03	726.98	332	0.99	8593.17	1076.36	787	0.986
FF		6527.83	519.43	576	0.978	6276.57	713.43	338	0.984	8531.33	1064.71	769	0.976
BRS		6668.55	521.12	612	0.997	6339.97	739.12	321	0.995	8680.78	1081.44	833	1
FillUp		6322.67	495.38	492	0.938	5847.27	708.28	258	0.917	8261.17	1056.06	659	0.939
ecoCloud [21]		6269.75	472.56	338	0.901	5821.42	708.55	180	0.892	7445.7	1038.56	574	0.939
SG-PBFS [32]		5697.52	473.21	202	0.828	5594.4	725.1	108	0.856	6967.88	1018.99	343	0.868
SAF+Backfill [31]		5910.9	508.55	287	0.872	5813.01	708.05	312	0.93	6482.41	1061.06	368	0.845
FCFS+Backfill		5970.77	505.03	248	0.87	5945.02	727.06	113	0.891	7925.31	1050.79	336	0.873
FillUp+Backfill		5401.96	438.67	201	**0.79**	5504.96	707.47	66	**0.825**	7163.45	1013.34	268	**0.795**
FCFS	Priority	6399.99	536.34	608	0.941	6867.22	741.76	378	0.984	8897.87	1102.26	814	0.984
FF		6347.47	525.87	599	0.929	6654.76	737.77	349	0.956	8869.54	1089.34	803	0.978
BRS		6983.6	539.33	655	1	6983.56	738.98	402	0.998	9042.53	1111.86	843	1
FillUp		6635.32	502.76	503	0.927	6239.21	742.32	321	0.916	8778.33	1087.67	798	0.971
ecoCloud [21]		6702.2	507.31	291	0.902	6481.78	743.43	374	0.95	8642.89	1092.7	518	0.93
SG-PBFS [32]		6470.56	511.09	559	0.926	6229.92	738.55	382	0.928	8236.7	1107.32	353	0.887
SAF+Backfill [31]		5841.33	516.53	338	0.841	6507.43	742.78	342	0.944	8795.06	1093.92	529	0.941
FCFS+Backfill		6274.93	537.53	297	0.883	6492.86	741.65	276	0.926	8267.79	1103.74	566	0.914
FillUp+Backfill		5722.12	499.45	245	**0.807**	6154.75	743.02	149	**0.866**	7988.87	1086.12	308	**0.86**

they are given in the Google traces [29], [36]. In our datasets, 23.41% of VMs wait for 1 second before the requested resources are provisioned, according to the distribution of workloads wait times. In 44.6% of cases, the waiting time is less than 30 seconds. Unexpectedly, 27 VMs (0.03% of the VMs) wait for more than 2 hours. These were observed using the FillUp approach with no consolidation or backfilling. With FillUp+Backfill approach using priority as a criterion, the percentage of VMs that were waiting for 1 second was reduced to 2.88% while the percentage of VMs that were waiting for 30 seconds was dropped to 13.82%. Furthermore, 22 VMs (0.026% of the VMs) were observed to be waiting for more than 2 hours. With the FillUp+Backfill approach using runtime as a criterion, the percentage of VMs that were waiting for 1 second was reduced to 2.23% while the percentage of VMs that were waiting for 30 seconds was dropped to 8.94%. Furthermore, 7 VMs (0.01% of the VMs) were observed to have waited more than 2 hours. If customers pay per second for their provisioned resources, then these improvements in scheduling delays can significantly decrease users' monetary costs. Subsequently, the throughput of the system, in terms of the number of VMs that completed their workloads' execution, is increased as shown in Fig. 5 (right). Most of the VMs were running for less than an hour, and, with consolidation, there was a significant increase in throughput per hour. However, consolidation decreases the performance of VMs and, therefore, increases users' costs. The backfill approach has resulted in an increase of the overall system throughput.

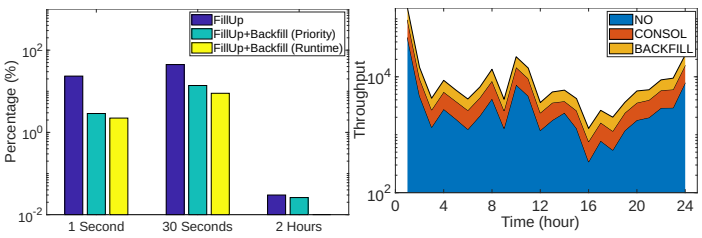

Fig. 5: Ratio of VMs wait times using Backfilling techniques (left) and Number of VMs per hour that completed their workloads' execution i.e. throughput (right)

Over-subscription is when consumers are given access to more resources than the datacenter can physically handle. Over-subscription may increase energy usage owing to an increase in workload and cooling needs, even while it permits greater resource utilisation and expanded user capacity. Moreover, depending on the degree of resource overuse, it may affect datacenter performance. For certain kinds of workloads, performance may be maintained or even increased when resources are oversubscribed in a responsible manner. However, extreme over-subscription can cause resource conflict, increased latency, and poor performance. Therefore, it is essential to account for workload types when over-subscription is needed. To prevent performance degradation due to over-subscription levels, it is essential to properly monitor and control resource allocation. Finally, we studied different criteria of VMs that are used to backfill particular VMs such as priority, workload type, runtimes, and their impacts on the obtained findings. The results are shown in Fig. 6 when priority and runtimes are assumed to backfill VMs. Our evaluation suggests that using VMs runtimes as a criterion for backfilling is approximately 3.56% – 7.78% more energy and 1.91% – 3.38% more performance efficient than using priority as a backfilling criterion.

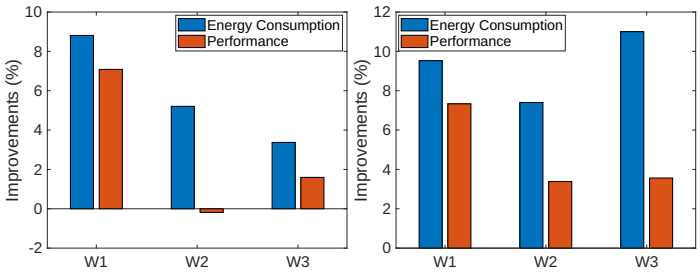

Fig. 6: Overall improvements in energy consumption and performance when using FCFS+Backfill+Runtime (left) and FillUp+Backfill+Runtime (right) instead of FCFS+Backfill+Priority (left) and FillUp+Backfill+Priority (right)

5.3 Results Discussion

We observed a trade-off between increasing system usage and completing tasks by the deadline. Backfilling increases resource use, however it has the potential to slow down some specific workloads' response times. Furthermore, which type of workloads (VMs) should be considered for backfilling that has a great impact on energy consumption and performance. For example, our evaluation suggests that using VMs runtimes as a criterion for backfilling is approximately 3.56% – 7.78% more energy and 1.91% – 3.38% more performance efficient than using priority as a backfilling criteria.

Priority-based backfilling approaches may be used to prioritize high-priority tasks and guarantee that they get scheduled quickly, even if doing so implies preempting lower-priority tasks, in order to overcome this issue. Priority-based backfilling creates a balance between resource utilisation and completing work deadlines by dynamically altering job priorities depending on their urgency and relevance. The system may make ineffective scheduling decisions and perform worse overall if it is unable to predict when resources will become available.

While consolidation may help with resource optimization, certain workloads, such as those that are latency-sensitive, resource-intensive, or have bursty or unpredictable resource needs, may not be good candidates for consolidation. These workloads often have particular needs or traits that make it difficult for them to share physical resources. To maximise efficiency while catering to the various demands of various workloads, it is crucial to strike the ideal balance between consolidation and the provision of specialized resources. Backfilling decreases the wait time for these VMs by enabling shorter or failed VMs to start during the backfill window. This decrease in turnaround time for VMs' workloads improves system performance as a whole and tenant satisfaction. Furthermore, backfilling allows for the faster execution of additional VMs, thereby increasing the total system throughput. Subsequently, utilising resources effectively allows for the processing of more workloads, increasing the productivity of the datacenter. Backfilling can increase the effectiveness and throughput of the cloud datacenter by efficiently using the resources that are already available and decreasing idle time.

Backfilling might cause problems with resource contention if it is not done appropriately. Performance problems or conflicts can arise when VMs with workloads using varying priorities use the same resources. The scheduling method becomes more sophisticated and complex during backfilling, since it needs to estimate resource availability and take different priorities and workloads into account. Therefore, it can be quite difficult to implement an effective backfilling method, in particular, which accounts for various objectives. To avoid resource conflict, it could occasionally be essential to set aside and reserve resources for VMs running high-priority workloads. This makes sure that enough resources are there when the high-priority workloads need them. Therefore, effective backfilling is possible without endangering the availability or accomplishment of high-priority VMs by setting aside resources. The qualities of the workload and the accessibility of idle resources determine how successful backfilling is. Backfilling can be especially useful in effectively utilising the available resources during such idle intervals if the workload displays bursts of activity with sporadic slack periods.

The scheduler keeps track of a "backfill window" that denotes the window of time during which VMs can be backfilled into the voids created by short-running VMs. The datacenter load, workload priority, and any particular scheduler-implemented policies all affect the backfill window's size, subsequently, affecting the outcomes. The scheduler uses an algorithm to find appropriate backfill VMs during the backfill window. This algorithm looks for gaps in the schedule caused by VMs that have been assigned resources but have not yet begun their workloads' execution. To identify the top backfill candidates, it assesses VM attributes, including runtime, resource needs, and priority. These attributes have significant impacts on the obtained outcomes [37].

5.4 Comparison with State-of-the-art Schedulers

We compared the proposed backfilling approach with six other backfilling approaches i.e. Easy, conservative, fattened backfilling, ecoCloud, SG-PBFS, and SAF+Backfill [38]. All these mechanisms were implemented with the same VM allocation and migration policies on the same hardware resources. The EASY backfilling technique just verifies that the subsequent VMs in the queue do not cause a delay for the first VM in the queue. Estimates of work runtimes serve as the foundation for this idea of backfilling. We use the previous runtimes of VMs to estimate their future runtimes through a probabilistic approach [39]. Additionally, the conservative strategy, in which short-running VMs are only advanced if they do not cause any other VMs in the queue to be delayed. short-running VMs are permitted to advance by the fattened backfilling algorithm as long as they do not cause the first VM in the queue to wait longer than the average runtime for previously completed VMs [10]. The experimental results are shown in Table 5. The results demonstrate the EASY backfilling approach is more performance efficient, for short-running workloads, but that comes at the cost of high energy consumption. Similarly, for long-running workloads, conservative backfilling is more energy-efficient (7034.08), but it has affected the workload performance significantly (1101.67). Other methods, including ecoCloud, SG-PBFS, and SAF+Backfill were also observed to have similar behaviour. As shown in Table 5 (**bolded**), the trade-off between energy consumption and performance is balanced well using the proposed FillUp+Backfill+Runtime algorithm (having the least values for N_T).

The proposed runtime-based backfill method outperforms the EASY approach and for various workloads, the energy consumption is reduced with a ratio of 4.8% – 12.78%. However, reduced performance was observed for short-running workloads, while approximately 7.17% performance improvements were noted for long-running workloads. Furthermore, our approach is approximately 6.8% – 9.77% more performance efficient than the conservative backfilling approach. However, we noticed an overlap in energy consumption for long-running workloads. Despite the fact that comparable outcomes were obtained for the proposed and fattened backfilling, in terms of energy consumption (0.4% – 1.7%) and performance (0.26% – 0.6%), for short-running workloads; however, for long-running workloads our approach outperformed the fattened mechanism. The energy consumption is approximately 2.8% more, while the performance can be up to 5.8% more than the fattened method. The improvement or deterioration in energy consumption and performance of the proposed FillUp+Backfill+Runtime approach with respect to the closest rivals are shown in Fig. 7.

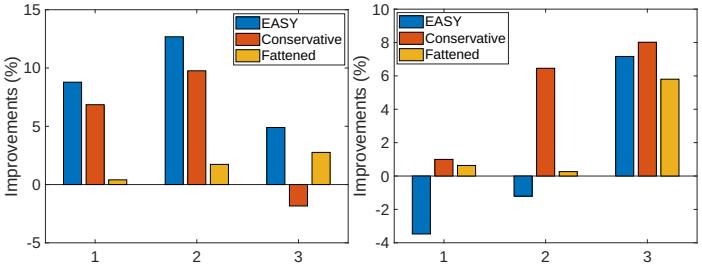

Fig. 7: Comparison of the proposed FillUp+Backfill+Runtime approach to the closest rivals in terms of energy consumption (left) and performance (right) – a negative value denotes dis-improvement [on x-axis 1 denotes workload W1, 2 denotes W2, and 3 represents W3]

6 Related Work

In [10], the authors propose a technique (fattened backfilling) that provides more backfilling options and is, thus, more efficient. Specifically, the proposed algorithm permits short tasks to continue, provided that they do not result in the first job in the queue waiting no longer than the average waiting time for tasks that have already been finished. The results show that fattened backfilling improves waiting and response times more than conservative and EASY backfilling. Backfilling may result in the loss of optimal packing opportunities since it considers each task in the queue independently. The popular backfilling techniques—EASY, conservative, and fattened—have been extensively studied for systems with a

TABLE 5: Comparison with other closest rivals in terms of energy consumption (E) and performance (P) for three different types of workloads [Energy is measured in KWh and lower values are better than high, Performance is the inverse of the execution time expressed in minutes and lower values are better than high] – approaches with the lowest values for N_T metric are best in balancing the trade-off between various objectives

Scheduling technique	W1			W2			W3		
	E	P	N_T	E	P	N_T	E	P	N_T
EASY backfilling [7]	5922.1	479.03	0.983	6304.54	699.04	0.97	7532.11	1091.55	0.962
Conservative backfilling [9]	5799.35	500.66	0.988	6100.45	756.3	0.981	7034.08	1101.67	0.928
Fattened backfilling [10]	5423.91	498.83	0.948	5601.9	709.33	0.908	7366.7	1075.76	0.944
ecoCloud [21]	5409.6	497.02	0.945	5542.98	701.5	0.899	7212.9	1024.94	0.914
SG-PBFS [32]	5800.3	500.1	0.987	5990.43	719.3	0.951	7088.32	1018.99	0.902
SAF+Backfill [31]	5501.98	491.05	0.95	5883.94	701.67	0.931	7432.87	1064.86	0.945
FillUp+Backfill+Runtime	5401.96	495.67	**0.943**	5504.96	707.47	**0.898**	7163.45	1013.34	**0.906**
FillUp+Backfill+Priority	5722.12	499.45	0.979	6154.75	743.02	0.979	7988.87	1086.12	0.994

single processor, but the cloud research community has not given them much attention. In [9], the authors suggest that dynamic programming may be used to maximise utilisation and choose the best packing based on the current queue state. Although consolidation and backfilling appear to be comparable strategies, they are very distinct from an execution standpoint. Migrations required for consolidation have a major effect on workload performance. Consolidation is not recommended in actual cloud platforms for this reason, among others [23]. Although backfilling does not require migrations, other constraints should be considered, such as the co-location of VMs and the behavior of the applications that are currently running on a specific host [13].

The approach in [7] groups all jobs based on their expected speed of completion. Then it determines whether the jobs can be completed within the allotted time and whether the available hosts can meet their demands. Short jobs are given precedence when using the backfill approach, which demonstrates a noticeable performance improvement. Numerous actions are carried out simultaneously through the pipeline system used in the execution technique. The results indicate that in conventional large systems, backfill results in a 20% increase in resource usage and a longer turnaround time [7]. About 90% of the small tasks are backfilled. The findings indicate that, while large tasks may not yield the same return, smaller tasks have more possibilities to backfill. Moreover, the backfill method does not demonstrate that carrying out large jobs frequently has a high priority. The work described in [7] is less concerned with user costs, performance, and energy usage and more with runtime forecasts.

In [30], authors proposed methods for VM migrations from overloaded servers to underloaded servers where the location of VMs and users is considered for minimal delay. To achieve energy efficiency, the algorithm runs in four phases, where two initial phases determine overloaded and underloaded hosts. In the third phase, VMs are migrated from overloaded hosts to underloaded servers with minimal delays (VMs closer to servers). The last phase determines the remaining overloaded and underloaded servers to accomplish more VM migrations. The results demonstrate approximately 13% reduction in energy consumption and 15% reduction in SLA violations. Furthermore, migrations are reduced by 19% and resource usage is reduced by approximately 20% in comparison to existing algorithms. Unfortunately, backfill methods and workload performance have not been investigated. Considering large-scale datacenters with diverse resources, the method described in [40] employs a heuristic technique "MinPR" to minimise energy and resource usage. This is accomplished by using power-efficient hosts to their full potential and minimising the number of servers that are actively in use. In addition, improving utilisation through load balancing reduces resource waste (migrating workloads and switching off underutilsed hosts). The placement of VMs under reward or penalty systems is related to a resource utilisation property. By ranking them according to their power efficiency, the property facilitates the provision of power-efficient hosts [41]. Energy efficiency is discussed; however, workload performance and backfill techniques are not examined.

In [42], [43], task-oriented methods for VM placement employing VMs, heterogeneous tasks, and servers inside cloud infrastructure are presented. The objective is to assign jobs to VMs, which are then energy-efficiently assigned to servers. This enables lower task rejection rates, makespan, and energy consumption. When a dynamic variance is seen in service demands for the systems' resource requirements, the task-based VM-placement algorithm (ETVMC) is preferred over the round-robin and FCFS approaches [43]. Unfortunately, these techniques have not employed a backfilling strategy, and their research is limited to the energy consumption of tasks or VMs. In resource allocation, a higher number of provisioned hosts results in lower SLA violation rates, which raises the user's price. Therefore, in [44] a method is presented to strike a balance between SLA violations and user costs to maximise server resource utilisation. Additionally, cloud workloads need a lot of disk operations and are data-intensive. As a result, the server's performance, disk bandwidth, and CPU consumption all suffer. Performance is not assured; however, the MMEVMP technique reduces SLAV and energy consumption [44]. With an emphasis on load balancing and energy consumption using various constraints, [13] assesses six VM placement strategies in heterogeneous clusters. The obtained outcomes demonstrate that load-balancing algorithms like spot migration could produce a lower CPU utilisation variability, while energy-saving algorithms like activeness-aware placement could dramatically decrease the number of active hosts. Furthermore, system performance and migration overhead are significantly influenced by constraints like VM-to-VM colocation.

In [45], a multi-criterion decision-making migration algorithm "TOPSIS" is presented that uses six criteria to perform VM migrations. To reduce energy usage and SLA violations, the algorithm ranks all VMs available for migration and picks the VM with the highest ranking score. The VM migration focuses on improving resource utilisation by consolidating VMs onto fewer hosts while backfilling focuses on maximising resource utilisation by allocating low-priority/failed VMs before high-priority VMs. In [32], a priority-based fair scheduling method "PBFS" is presented to schedule jobs that allow them to access essential resources at the best possible times. Furthermore, a backfilling technique "Shortest Gap PBFS" is presented to control durations in tasks' schedules. Performance evaluation demonstrates that, in terms of schedule, makespan, and overall delay, SG-PBFS performs better than its closest rivals. Unlike our research, the works in [45] and [32] have not investigated the impacts of the backfilling on workload performance, energy consumption, and users' costs. Furthermore, the criteria used to backfill particular jobs and their impacts on the findings are also not taken into account.

In [14], authors suggest a hybrid "Interactive PSO-GA" (Particle Swarm Optimization and Genetic Algorithm) method for allocating resources in data centers in an energy-efficient manner. To enhance convergence and optimization, the method executes PSO and GA

concurrently, exchanging information. The experimental evaluation shows that it can reduce the convergence time by 50% and energy usage by up to 34% compared to other methods. Its significant speedup and great parallel efficiency (increase to 97.5%) confirm its suitability for use in actual cloud systems. For diverse cloud data centers, [46] suggests an enhanced paradigm for resource provisioning and VM migration. It presents a random perturbation-enhanced JAYA (RP-JAYA) technique for VM placement and a host overload detection algorithm based on Z-Score. These techniques minimize performance deterioration to 0.08%, reduce VM migrations by 32%, and reduce energy usage by up to 49%. Simulations show notable efficiency gains over more conventional methods like PSO, GA, and MBFD. The summary of the comparison between our proposed technique "BackFillMe" and other closely related works is given in Table 6. We believe the comparison would help readers to quickly identify gaps for further research.

7 Conclusions and Future Work

In this paper, we design an algorithm that uses the concept of backfilling to solve the resource allocation problem efficiently, both in terms of energy and performance. We experimentally evaluated the proposed backfilling approach, which guarantees effective use of the underlying infrastructure, maximises the number of VMs that can be accommodated, and boosts the performance of the virtualised environment by utilising the idle resources that are already accessible. To maximise the advantages of backfilling, precise resource availability estimates, machine learning methods, and priority-based scheduling algorithms must be taken into account. Advancements in backfilling techniques will be essential for establishing effective job scheduling and improving overall performance as computer systems keep developing and handling more complicated workloads. If it is anticipated that an overloaded host won't be overloaded or underloaded in the near future, then in that case migrating VMs from these hosts won't be effective and essential. Therefore, in order to determine if the migration is really required, we must consider how the host's resources will be used in the future. The probabilistic Bernoulli trial strategy makes the assumption that every server or VM has the same capacity and capabilities, therefore, treating each server as equally likely to be selected for allocation. Because of the homogeneity assumption, lack of resource awareness, and inefficient use of resources, this may result in suboptimal placement decisions. In the future, we will work toward adjusting the probability of selecting a server based on its capacity, current load, and other factors. When dealing with large cluster sizes and VMs, centralised scheduling might become a bottleneck and a single point of failure. In the future, we will implement the proposed method in a distributed and decentralised manner.

Acknowledgments

This work is partially supported by Adul Wali Khan University, Mardan, Pakistan and, in parts, by the Sohar University, Sultanate of Oman.

References

[1] A. Shehabi, S. J. Smith, E. Masanet, and J. Koomey, "Data center growth in the united states: decoupling the demand for services from electricity use," *Environmental Research Letters*, vol. 13, no. 12, p. 124030, 2018.

[2] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020.

[3] R. Buyya, S. Ilager, and P. Arroba, "Energy-efficiency and sustainability in new generation cloud computing: A vision and directions for integrated management of data centre resources and workloads," *Software: Practice and Experience*, vol. 54, no. 1, pp. 24–38, 2024.

[4] M. Zakarya, L. Gillam, K. Salah, O. F. Rana, S. Tirunagari, and R. Buyya, "Colocateme: Aggregation-based, energy, performance and cost aware VM placement and consolidation in heterogeneous iaas clouds," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1023–1038, 2023. [Online]. Available: https://doi.org/10.1109/TSC.2022.3181375

[5] A. A. Khan and M. Zakarya, "Energy, performance and cost efficient cloud datacentres: A survey," *Computer Science Review*, vol. 40, p. 100390, 2021.

[6] A. A. Khan, M. Zakarya, R. Buyya, R. Khan, M. Khan, and O. Rana, "An energy and performance aware consolidation technique for containerized datacenters," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1305–1322, 2019.

[7] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Backfilling using system-generated predictions rather than user runtime estimates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 789–803, 2007.

[8] H. Elshazly, J. Ejarque, and R. M. Badia, "Storage-heterogeneity aware task-based programming models to optimize i/o intensive applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3589–3599, 2022.

[9] E. Shmueli and D. G. Feitelson, "Backfilling with lookahead to optimize the packing of parallel jobs," *Journal of parallel and distributed computing*, vol. 65, no. 9, pp. 1090–1107, 2005.

[10] C. Gómez-Martín, M. A. Vega-Rodríguez, and J.-L. González-Sánchez, "Fattened backfilling: An improved strategy for job scheduling in parallel systems," *Journal of Parallel and Distributed Computing*, vol. 97, pp. 69–77, 2016.

[11] D. G. Feitelson, "Experimental analysis of the root causes of performance evaluation results: a backfilling case study," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 175–182, 2005.

[12] D. Talby and D. G. Feitelson, "Improving and stabilizing parallel computer performance using adaptive backfilling," in *19th IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2005, pp. 10–pp.

[13] S. Kim and Y.-r. Choi, "Constraint-aware vm placement in heterogeneous computing clusters," *Cluster Computing*, vol. 23, no. 1, pp. 71–85, 2020.

[14] V. D. Reddy, G. Gangadharan, G. Rao, and M. Aiello, "Energy-efficient resource allocation in data centers using a hybrid evolutionary algorithm," *Machine learning for intelligent decision science*, pp. 71–92, 2020.

[15] Q. Fang, J. Wang, Q. Gong, and M. Song, "Thermal-aware energy management of an hpc data center via two-time-scale control," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2260–2269, 2017.

[16] Y. Etsion and D. Tsafrir, "A short survey of commercial cluster batch schedulers," *School of Computer Science and Engineering, The Hebrew University of Jerusalem*, vol. 44221, no. 2005, p. 13, 2005.

[17] M. Zakarya, A. A. Khan, M. R. C. Qazani, H. Ali, M. Al-Bahri, A. U. R. Khan, A. Ali, and R. Khan, "Sustainable computing across datacenters: A review of enabling models and techniques," *Computer Science Review*, vol. 52, p. 100620, 2024.

[18] C. Mastroianni, M. Meo, and G. Papuzzo, "Self-economy in cloud data centers: Statistical assignment and migration of virtual machines," in *European Conference on Parallel Processing*. Springer, 2011, pp. 407–418.

[19] M. Zakarya and L. Gillam, "An energy aware cost recovery approach for virtual machine migration," in *International Conference on Economics of Grids, Clouds, Systems and Services*. Springer, 2016.

[20] 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 Computer Systems*, vol. 27, no. 8, pp. 1027–1034, 2011.

[21] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic consolidation of virtual machines in self-organizing cloud data centers," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 215–228, 2013.

[22] Z. A. Mann, "Decentralized application placement in fog computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3262–3273, 2022.

[23] M. Zakarya and L. Gillam, "Energy and performance aware resource management in heterogeneous cloud datacenters." Ph.D. dissertation, University of Surrey, 2017.

[24] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[25] K. Kaur, S. Garg, G. Kaddoum, E. Bou-Harb, and K.-K. R. Choo, "A big data-enabled consolidated framework for energy efficient software defined data centers in iot setups," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2687–2697, 2019.

[26] J. O'Loughlin and L. Gillam, "Sibling virtual machine co-location confirmation and avoidance tactics for public infrastructure clouds," *The Journal of Supercomputing*, vol. 72, no. 3, pp. 961–984, 2016.

TABLE 6: Summary of the related works [VMP - VM Placement, BF - BackFill, S - Single system, LB - Load balancing, and C - Cloud]

Matching criteria	related work											BackFillMe
	[7]	[30]	[40]	[42]	[43]	[13]	[14]	[46]	[10]	[32]	[45]	
Placement	BF-S	VMP+C	VMP	VMP	VMP	VMP+LB	VMP	VMP+C	BF-S	VMP+BF	VMP+C	VMP+BF-C
Migrations		✓				✓		✓			✓	✓
Performance				✓	✓	✓	✓	✓	✓	✓	✓	✓
Energy efficiency		✓	✓	✓	✓	✓	✓	✓			✓	✓
Previous runtimes												✓
Priority	✓									✓		✓
Predictions	✓											
Costs												✓

[27] R. Kavanagh, K. Djemame, J. Ejarque, R. M. Badia, and D. Garcia-Perez, "Energy-aware self-adaptation for application execution on heterogeneous parallel architectures," *IEEE Transactions on Sustainable Computing*, vol. 5, no. 1, pp. 81–94, 2019.

[28] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Profit-aware application placement for integrated fog–cloud computing environments," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 177–190, 2020.

[29] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., Mountain View, CA, USA, Technical Report*, 2011.

[30] A. Beloglazov and R. Buyya, "OpenStack Neat: A framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds," *Concurrency Computation*, vol. 27, no. 5, pp. 1310–1333, 2015.

[31] D. Carastan-Santos, R. Y. De Camargo, D. Trystram, and S. Zrigui, "One can only gain by replacing easy backfilling: A simple scheduling policies case study," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2019, pp. 1–10.

[32] S. A. Murad, Z. R. M. Azmi, A. J. M. Muzahid, M. K. B. Bhuiyan, M. Saib, N. Rahimi, N. J. Prottasha, and A. K. Bairagi, "Sg-pbfs: Shortest gap-priority based fair scheduling technique for job scheduling in cloud environment," *Future Generation Computer Systems*, vol. 150, pp. 232–242, 2024.

[33] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[34] M. Zakarya, L. Gillam, A. A. Khan, and I. U. Rahman, "Perficient-cloudsim: a tool to simulate large-scale computation in heterogeneous clouds," *The Journal of Supercomputing*, vol. 77, pp. 3959–4013, 2021.

[35] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *Proceedings of the fifteenth European conference on computer systems*, 2020, pp. 1–14.

[36] P. Minet, E. Renault, I. Khoufi, and S. Boumerdassi, "Analyzing traces from a google data center," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2018, pp. 1167–1172.

[37] H. Ali, M. S. Qureshi, M. B. Qureshi, A. A. Khan, M. Zakarya, and M. Fayaz, "An energy and performance aware scheduler for real-time tasks in cloud datacentres," *IEEE Access*, vol. 8, pp. 161 288–161 303, 2020.

[38] A. W. Mu'alem and D. G. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling," *IEEE transactions on parallel and distributed systems*, vol. 12, no. 6, pp. 529–543, 2001.

[39] M. Zakarya, "An extended energy-aware cost recovery approach for virtual machine migration," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1466–1477, 2018.

[40] S. Azizi, M. Zandsalimi, and D. Li, "An energy-efficient algorithm for virtual machine placement optimization in cloud data centers," *Cluster Computing*, vol. 23, no. 4, pp. 3421–3434, 2020.

[41] Y. Kumar, S. Kaul, and Y.-C. Hu, "Machine learning for energy-resource allocation, workflow scheduling and live migration in cloud computing: State-of-the-art survey," *Sustainable Computing: Informatics and Systems*, vol. 36, p. 100780, 2022.

[42] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168–180, 2014.

[43] S. K. Mishra, D. Puthal, B. Sahoo, P. P. Jayaraman, S. Jun, A. Y. Zomaya, and R. Ranjan, "Energy-efficient vm-placement in cloud data center," *Sustainable computing: informatics and systems*, vol. 20, pp. 48–55, 2018.

[44] M.-H. Kim, J.-Y. Lee, S. A. R. Shah, T.-H. Kim, and S.-Y. Noh, "Min-max exclusive virtual machine placement in cloud computing for scientific data environment," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–17, 2021.

[45] K. K. Chakravarthi and L. Shyamala, "Topsis inspired budget and deadline aware multi-workflow scheduling for cloud computing," *Journal of Systems Architecture*, vol. 114, p. 101916, 2021.

[46] D. R. Vemula, M. K. Morampudi, S. Maurya, A. Abdul, M. M. Hussain, and I. Kavati, "Enhanced resource provisioning and migrating virtual machines in heterogeneous cloud data center," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 9, pp. 12 825–12 836, 2023.

[47] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, "Performance and energy modeling for live migration of virtual machines," *Cluster Computing*, pp. 249–264, 2011.

[48] C. C. Lin, P. Liu, and J. J. Wu, "Energy-aware virtual machine dynamic provision and scheduling for cloud computing," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 736–737.

[49] B. Shi and H. Shen, "Memory/disk operation aware lightweight vm live migration across data-centers with low performance impact," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 334–342.

APPENDIX A
MODELLING MIGRATION ENERGY AND PERFORMANCE

In this appendix, we deliberate several statistical models that are used in simulations to measure the energy consumption and performance of resources and workloads during migrations.

A.1 Modelling Migration Energy Consumption

In datacenters, migrations are common, and their power consumption overhead must not be ignored. The power consumption of a migration process is a dynamic portion of the power model that contains the power usage of numerous hardware, including CPU, memory, data transmission rate, i.e., R, and network interface card. At source, the power consumption due to migration rises as R is increased, while the migration time decreases. In [47], the author shows that at different R, the average consumption leads to fewer than six percentiles of disparity, which verifies that the migration power consumption overhead is mostly free of R. The authors claim that migration is an I/O-intensive process where the power is mostly spent on data sending and receiving. The authors have modelled the power consumption costs, due to migration, based on the capacity of network traffic. The VM migration contains a source, a destination host, and a network switch. As switching fabric is difficult, therefore power is tough to compute. In [48], the authors stated that live migration does not affect the execution time of a VM, but it increases the power consumption of source and destination hosts. On migration starts, the destination host creates an extra process to copy VM memory contents that will raise the loading, leading to a rise in total power consumption. The authors claim that a live migration rises 20% loading on the destination host, while there is no increase at the source host. Their model for migration energy overhead is given by Eq. 14:

$$E_{mig} = E_{source} + E_{dest} \tag{14}$$

Considering the power model where P_p is the peak power consumption, the idle power consumption is assumed as half of the peak power denoted by α, c is the number of cores used, and C denotes the total number of cores on a host. Then, the total power consumption of the host is given by Eq. 15:

$$P = (\frac{c}{C}.(1-\alpha)+\alpha) \times P_p \tag{15}$$

We get the source and destination energy consumption overhead as given in Eq. 16 and Eq. 17, respectively:

$$E_{source} = (\frac{c}{C}.(1-\alpha)+\alpha) \times P_p.T_{mig} \tag{16}$$

$$E_{dest} = (\frac{c^{\circ}}{+}L)(C.(1-\alpha)+\alpha) \times P_p.T_{mig} \tag{17}$$

where T_{mig} is the migration total time, L is the increase in load at the destination host, c° is the number of cores required at the destination for the VM to be migrated, and $\frac{c}{C}$ denotes the load on the host. For offline migration, as the VM is suspended, there is a strong decrease in power consumption of the source server. For live migration, the source server will touch a new peak for power consumption due to the groundwork for sending a VM to the destination. The destination server will experience a peak in its power demand due to resource provisioning in a migration approach independent behavior. The network cost cannot be ignored, especially for the live migration, where the VM is continuously writing to memory, and it takes much time to transfer the VM to the destination. Some studies have ignored this cost, with the assumption that it will only affect the migration when running at full utilisation. The authors assume that a VM migration will never be issued when the bandwidth between the two servers is fully utilised. Other studies consider this cost as constant, with the assumption that the power consumption of the network switches is not proportional to the amount of traffic being moved over some time.

A.2 Modelling Migration Performance

Modelling the performance of migration includes numerous factors, including the VM memory size, network transmission rate, the migration algorithm, and the workload features, i.e., memory dirtying rate. The key parameters are VM size (V_{mem}), network traffic (V_{mig}), total migration time (T_{mig}), downtime (T_{down}), memory transmission rate (R), memory dirty rate (D), threshold for the last round (V_{th}), and writable working set (W) to transfer hot pages. To minimize T_{down}, live migration copies the dirty pages at the previous round of transmission iteratively. Consider that there are n rounds, which completes the pre-copy algorithm; then, the volume of data at round i is V_i and the elapsed time is T_i for $0 \leq i \leq n$. Therefore, the data transmitted and the time during each round are given by Eq. 18 and Eq. 19, respectively:

$$V_i = \begin{cases} V_{mem} & \text{if i = 0} \\ D.T_{i-1} & \text{if i > 0} \end{cases} \tag{18}$$

$$T_i = \frac{D.T_{i-1}}{R} = \frac{V_{men}.D^i}{R^{i+1}} \tag{19}$$

Consider that $D < R$ on average and ω denotes the ratio of D to R as given by Eq. 20:

$$\omega = \frac{D}{R} \tag{20}$$

Combining Eq. 18, Eq. 19, and Eq. 20, we get Eq. 21:

$$V_i = V_{mem}.\omega^i \tag{21}$$

The total network traffic is given by Eq. 22:

$$V_{mig} = \sum_{i=0}^{n} V_i = V_{mem}.\frac{1-\omega^{n+1}}{1-\omega} \tag{22}$$

The total migration time is given by Eq. 23:

$$T_{mig} = \sum_{i=0}^{n} T_i = \frac{V_{mem}}{R}.\frac{1-\omega^{n+1}}{1-\omega} \tag{23}$$

The migration downtime contains two different parts: (1) the time to transfer lasting dirty pages in the stop-and-copy period i.e. T_n; and (2) the time to resume the VM at the destination host i.e. $Tresume$ which has slight variation and can be characterized as a constant value of 20ms [4]. The migration downtime is given by Eq. 24:

$$T_{down} = T_n + T_{resume} \tag{24}$$

The inequality $V_n \leq V_{th}$ can be written as $V_{mem}.\omega^n \leq V_{th}$ to calculate the total number of rounds for algorithm convergence, which is given by Eq. 25:

$$n = log_{\omega}.\frac{V_{th}}{V_{mem}} \tag{25}$$

From the above studies, we determine that a VM having a small memory image and trivial ω would cause a smaller amount of network traffic leading to shorter T_{mig}, therefore is a better nominee for migration. Note that if ω is smaller, then the pre-copy technique will converge faster.

If the D is even larger than the R then the amount of data transmitted in each round i will beat the VM size, which will increase the total migration time even if the migration will not be accomplished. We do not consider such situations in our modelling, but the Xen hypervisor has solved this issue using the writable working set technique [23]. The pages that are rottenly dirtied i.e. hot pages are ignored to transfer till the last round of migration. More details on such type of study can be found in [49].