Graphical Abstract

A Dynamic Evolutionary Multi-Objective Virtual Machine Placement Heuristic for Cloud Data Centers

Ennio Torre, Juan J. Durillo, Vincenzo de Maio, Prateek Agrawal, Shajulin Benedict, Nishant Saurabh, Radu Prodan

Highlights

A Dynamic Evolutionary Multi-Objective Virtual Machine Placement Heuristic for Cloud Data Centers

Ennio Torre, Juan J. Durillo, Vincenzo de Maio, Prateek Agrawal, Shajulin Benedict, Nishant Saurabh, Radu Prodan

- Multi-objective algorithm for virtual machine (VM) placement in Cloud data centers;
- Approximation of Pareto optimal set of VM placements;
- Resource overcommitment, resource wastage and live migration energy tradeoff;
- Island-based optimisation heuristic based on genetic NSGA-II algorithm;
- Improved evaluation results compared to state-of-the-art heuristics.

A Dynamic Evolutionary Multi-Objective Virtual Machine Placement Heuristic for Cloud Data Centers

Ennio Torre, Juan J. Durillo[a], Vincenzo de Maio[b], Prateek Agrawal[c], Shajulin Benedict[d], Nishant Saurabh, Radu Prodan[e]

[a] *Leibniz Supercomputing Center, Munich, Germany*
[b] *Vienna University of Technology, Vienna, Austria*
[c] *Lovely Professional University, Punjab, India*
[d] *Indian Institute of Information Technology, Kottayam, India*
[e] *University of Klagenfurt, Austria*

Abstract

Minimizing the resource wastage reduces the energy cost of operating a data center, but may also lead to a considerably high resource overcommitment affecting the Quality of Service (QoS) of the running applications. The effective tradeoff between resource wastage and overcommitment is a challenging task in virtualized Clouds and depends on the allocation of virtual machines (VMs) to physical resources. We propose in this paper a multi-objective method for dynamic VM placement, which exploits live migration mechanisms to simultaneously optimize the resource wastage, overcommitment ratio and migration energy. Our optimization algorithm uses a novel evolutionary meta-heuristic based on an island population model to approximate the Pareto optimal set of VM placements with good accuracy and diversity. Simulation results using traces collected from a real Google cluster demonstrate that our method outperforms related approaches by reducing the migration energy by up to 57% with a QoS increase below 6%.

Keywords: VM placement, multi-objective optimisation, resource

overcommitment, resource wastage, live migration, energy consumption, Pareto optimal set, genetic algorithm, data center simulation

---

1. Introduction

Virtualized data centers are the backbone of many Cloud providers like Amazon and Google that rent their physical Infrastructure-as-a-Service (IaaS) to their customers. In a virtualized data center, *Virtual Machines (VMs)* wrap customer applications, representing their execution environment hosted onto data center *Physical Machines (PMs)*. The *VM placement* problem aims to find an allocation or mapping of a set of VMs onto a subset of available PMs that optimizes one or more objectives relevant to the IaaS provider. More specifically, if the aim is to minimize the size of this PM subset, we refer to this problem as the *VM consolidation*. This problem gathered attention in the last decade thanks to its ability to minimize not only the *resource wastage*, but also the *energy consumption* and the overall electricity cost by turning unused PMs to a lower power state.

1.1. Motivation 1: VM overcommitment

Despite these benefits, data center operators take a cautious attitude towards consolidation. One approach allocates the VMs according to their resource requests (i.e. CPU, memory, disk) such that the cumulative demand is lower than the PMs' resource capacity. However, this suffers from *over-provisioning* [9], as IaaS customers tend to overestimate their VM resource requests to ensure fulfillment of their application requirements at all times. This results in a low consolidated data center with underutilized PMs.

A solution to overprovisioning is to optimize the VM placement according to the VMs resource demands independently of their requests. One technique called *resource overcommitment* [10] allows placing (or consolidating) more VMs onto the same PM by sharing hardware resources exceeding its physical capacity. Unfortunately, overcommitment can have a detrimental impact on the performance of applications [6] by congesting limited PM resources with significant Quality of Service (QoS) violations and penalties.

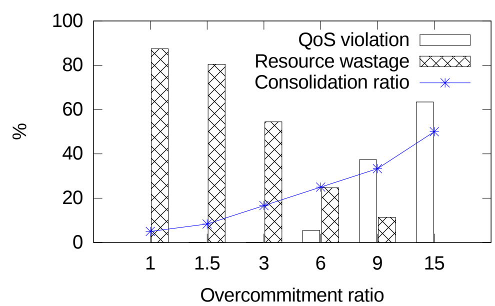


Figure 1: Impact of overcommitment on resource wastage and QoS.

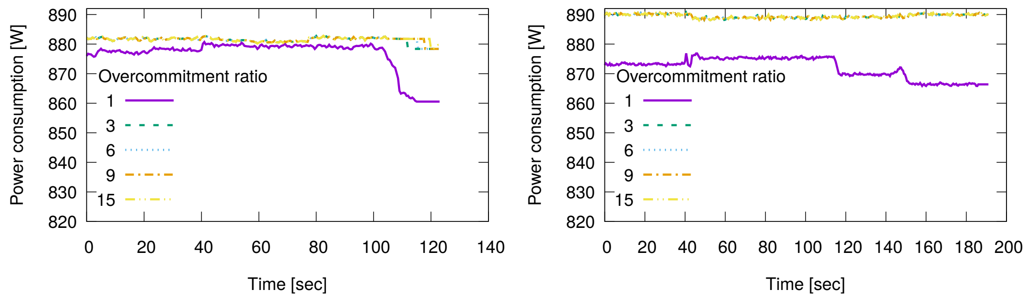
To verify our motivation, we simulated 100 VMs (using the homogeneous experimental scenario described in Section 7.1) with application workloads generated by randomly sampling the Google cluster traces [35] using a Poisson distribution. We studied the effects of the overcommittment ratio (i.e. the ratio between the requested and the available resources on each PM) to three important metrics: resource wastage, consolidation ratio (i.e. the ratio between the number of VMs and the allocated PMs) and QoS violation (i.e. the percentage of VMs that do not receive sufficient PM resources). Fig. 1 shows that the benefits of the overcommitment to the resource wastage decreases from 87% in case of no overcommittment, to 0% for the highest overcommitment ratio. Contrarily, the consolidation ratio increases from 5% to 50%. On the negative side, overcommitment comes with an increase in QoS violations which, although negligible for a low overcommitment ratio (from 1 to 6), reach a serious 60% value for the highest overcommitment. It is therefore clear that the relation between the resource wastage and the overcommitment ratio is crucial for an energy-efficient resource management under QoS constraints, Moreover, understanding this relation is not immediate due to the conflicting nature of the two objectives leading to a wide spectrum of different possible tradeoffs.

1.2. Motivation 2: live VM migration

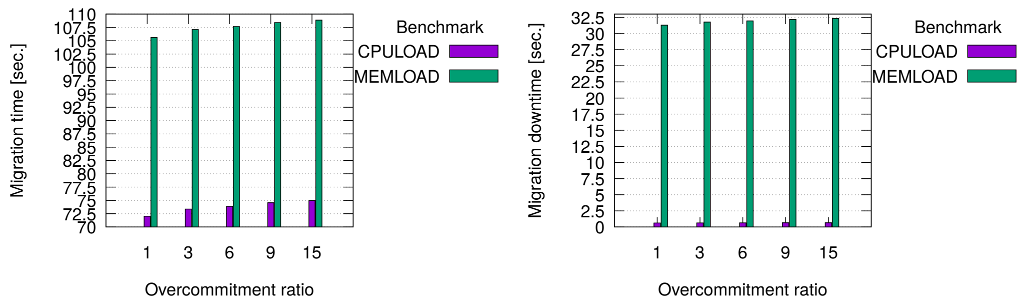
Dynamic real-world VM workloads require continuous on-the-fly modifications of the VM placements, supported through *VM migrations* and enabling different potential resource overcommitment and wastage tradeoffs. To minimize the impact on the application QoS, *live migration* transparently moves a running VM between different PMs without disconnecting it from the client and exposing no (or minimum) interruptions.

The cost of VM migration has two main dimensions [18]: the *downtime* (i.e. VM unavailability) and the additional power consumption on the source and target PMs during this process. To understand the impact of overcommitment on these parameters, we conducted an experiment that measures the VM energy consumption and its migration time on an overcommitted PM using the same VM workload as in Section 1.1 on the machines presented in Table 1. We further generated two VM-internal CPU and memory-intensive workloads that influence its live migration, as described in [19]:

- `CPULOAD` uses OpenMP implementation of a matrix multiplication algorithm running on all cores allocated to the VM.
- `MEMLOAD` continuously updates the VM memory pages using a high 95%



(a) Instantaneous `CPULOAD` power. (b) Instantaneous `MEMLOAD` power.



(c) Migration time versus overcommitment. (d) Downtime versus overcommitment.

Figure 2: Impact of overcommittment on VM power consumption, migration time and downtime for CPU and memory-intensive workloads.

dirtying rate (i.e. percentage of memory pages that become dirty in a given interval), considered as the most impacting factor on downtime [3].

Fig. 2a and Fig. 2b show that the overcommitment ratio (especially above three) does not affect the power consumption of the source PM. The drop in power in the absence of overcommitment after around 100 s in Fig. 2a and 120 s in Fig. 2b is due freeing the source PM resources. However, Fig. 2c shows that the overcommitment has an impact on the VM migration time and affects its energy consumption. For this reason, we separately consider the VM migration energy from server energy consumption.

Fig. 2d shows the `CPULOAD` downtime ranges between 300 ms to 600 ms, which is negligible according to [3]. The `MEMLOAD` downtime is up to 30 in the case of an extreme 95% dirtying rate. Since our experimental workloads exhibit a dirtying rate below 15% for all the tasks with a downtime below 1 s, we define the VM migration cost in term of its energy consumption only.

Finally according to [31], the energy overhead may be up to 2.5 kJ for each VM migration, accounting for 10% of the idle energy consumption of an average PM. Furthermore, the energy overhead of VM migration can be as high as 5.8% of total energy consumption of a data center [2]. Therefore, understanding the impact of migrations on the other metrics is important, since they influence other aspects of data centre operation.

1.3. Problem statement

To address these motivations, we propose a *multi-objective method* and algorithm for dynamic placement of VMs in response to their fluctuating resource demands. Our goal is to minimise the energy consumption in data centres faced with dynamic workloads by dynamically allocating VMs to the minimum number of PMs using a three-fold strategy: 1) reduce the number of PMs by increasing the overcommitment; 2) analyse the effects of the overcommitment and overly reduced number of PMs on the QoS; 3) analyse the effects on live migration, ignored in related work.

We model the VM placement as a multi-objective *Vector Bin Packing (VBP)* [15] problem considering three conflicting objectives: resource wastage, overcommitment ratio, and migration cost. We introduce a novel *island-based evolutionary meta-heuristic* that dynamically provides a set of tradeoff VMs placements that accommodate the workload demand. Contrary to single solution approaches, we demonstrate the advantage of approximating the entire set of "optimal" tradeoff placements through simulation results on real-world traces obtained from a Google cluster. Such a multi-objective approach is an asset that reveals tradeoff placements from the search space not covered by single-objective approaches and impossible to see otherwise.

1.4. Article structure

The next section discusses the related work, followed by a multi-objective optimization background in Section 3. We introduce and formalize the VM placement problem in Section 4, implemented in practice within the architecture of a dynamic Cloud computing environment for real-world scientific and industrial workloads, presented in Section 5. We present an island-based evolutionary meta-heuristic for solving the VM placement problem in Section 6. Section 7 evaluates our method compared to related approaches on real data center simulation traces. Section 8 concludes the paper.

2. Related Work

We classify the studies on dynamic VM placement in two categories: single and multi-objective approaches.

2.1. Single-objective placement

In contrast to our multi-objective approach, these works are limited to approximating a single optimized placement solution.

First Fit Decreasing (FFD) is a fundamental algorithm used in the community for benchmarking VM placement algorithms. FFD sorts the VMs according to their CPU and memory size in descending order, and sequentially places them on the first PM with sufficient resources. MM_MBFD [7] is a two phase algorithm that combines a minimisation of migration (MM) algorithm that selects the VMs to migrate based on a double-CPU threshold policy with a modified best-fit decreasing (MBFD) VM placement heuristic to keep the total CPU demand of the placed VMs between the two thresholds.

Murtazaev et al. [25] proposed a method that reduces the number of active PMs by iteratively migrating the VMs from the least loaded to the more loaded ones. Verma et al. [34] introduced an algorithm that minimizes the number of migrations and the energy cost by first computing the placement that minimizes the power consumption, then calculating the migrations required to modify the current placement, and finally migrating the VMs with the minimum ratio between the estimated power consumption and the migration cost. Van et al. [33] implemented a method for power and performance-efficient provisioning of VMs and PMs by using a utility function for the optimal tradeoff between energy and performance. Beloglazov et al. [7] proposed a method that initially determines the minimum number of migrating VMs for keeping the PMs' utilization within a certain interval, followed by a modified FFD placement algorithm. Takahashi et al. [32] presented a greedy heuristic to minimize the total power consumption and the performance degradation due to VM consolidation. Mi et al. [22] proposed a proactive VM placement reconfiguration method based on predicted application demand using a modified genetic algorithm that minimizes the overall PM power consumption and maximizes the utilization. Ferdaus et al. [14] implemented a colony optimization meta-heuristic for finding a dynamic VM placement that balances the load among the the minimum number of PMs and minimizes the power consumption. All these works provided QoS guarantees by constraining the PM utilization achievable by a VM placement.

Other approaches look at the tradeoff between power consumption and performance degradation, but transform the problem into a single objective one by using a utility function or a weighted sum of the objectives. However, weighting and combining incompatible metrics in a single arithmetic function, despite being artificial and unrealistic, is an a-priori method with unclear impact on the solution. Li et al. [17] modeled the VM placement as a mixed integer linear programming problem that optimizes application performance, license cost and power consumption. Xu et al. [36] built a controller that places the VMs with optimized temperature, performance and power.

2.2. Multi-objective placement

Similar to our approach, several works focused on the simultaneous optimization of multiple objectives.

Lama et al. [16] built an analytic queuing model simulated as a multi-objective problem that minimizes the response time and the number of PMs and VMs of an n-tier application using the NSGA-II algorithm [11] and a stress-strain decision making strategy. Contrary to [16], our approach does not focus on a particular workload type (i.e. three-tier applications), but considers heterogeneous workloads including the migration cost incurred by placement modifications (ignored in [16]).

Sallam at al. [30] proposed a migration policy that selects a VM from a given set based on the migration cost, resources wastage, power consumption, thermal dissipation, and PM load. Our approach also accounts for migration cost and resource wastage, however, our goal is to dynamically optimise the VM placements, while [30] focuses on selecting the "optimal" VM to migrate.

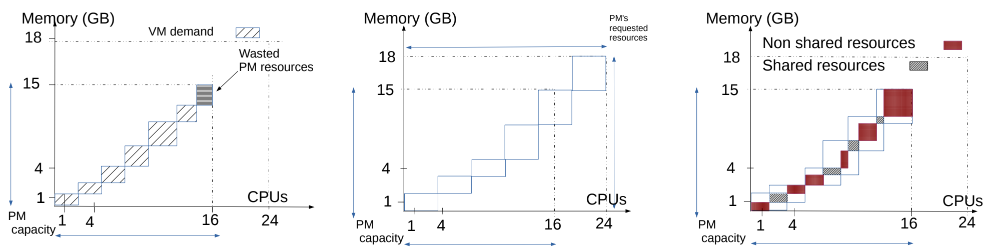
To the best of our knowledge, our work is the first to exploit the conflict between resource overcomittment and wastage represented as a tradeoff between energy consumption and performance metrics.

3. Background

3.1. Vector Bin Packing

We theoretically model the VM placement as a *Vector Bin Packing (VBP)* problem, which maps a set of VMs with known resource demands onto a set of PMs with known resource capacities [24]. The demands of the VMs and the capacities of the PMs are v-dimensional vectors, whose components represent v resource types, such as CPU number, memory size, disk space, or network bandwidth. VBP's goal is to place the VMs onto the minimum number of PMs, such that the cumulative demand of the VMs sharing the same PM is smaller than or equal to its capacity in each dimension.

Fig. 3a gives a VM placement example across two resources: CPU and memory. The inner dotted line represents the resource capacity of a single PM, while the dimensions of the small rectangles indicate the two resource demand dimensions of six VMs. After placing the VMs in a vector fashion, their aggregated resource demand determines the wastage in each dimension. Fig. 3b shows an overcommitment example due to load fluctuations of the VMs allocated in Fig. 3a and requiring more physical resources than the total PM capacity. The virtualization technology copes with the overcommitment by multiplexing the PM resources shared among VMs, as shown



(a) Resource wastage. (b) Overcommitment. (c) Resource sharing.

Figure 3: VBP allocation of six VMs onto one PM with 16 CPUs and 15 GB of memory.

by the red-shaded rectangles in Fig. 3c. Resource sharing increases with the overcommitment ratio and degrades the performance.

Our method generalizes the original VBP by dealing with overcommitment and treating VM placement as a dynamic problem. In our model, VMs have a static size and a variable resource demand described by two v-dimensional vectors. PMs have an associated resource wastage, an overcommitment ratio and a VM migration cost. While the resource wastage and the overcommitment ratio depend on the demand and on the size of the placed VMs, the migration cost depends on the energy consumption required to modify a placement. Our goal is to simultaneously optimize the PMs' resources wastage, overcommitment ratio, and migration cost, which are conflicting objectives that require multi-objective optimization.

3.2. Multi-objective optimization

A *multi-objective optimization* problem consists of a vector of k objective functions $\overrightarrow{f(\overrightarrow{x})} = (f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x}))$. These objectives are usually in conflict with each other, meaning that optimizing one implies worsening at least another one. Without loss of generality, we consider the minimization of all functions. The next section describes objectives considered in this work.

The components of a solution vector (or simply solution) $\overrightarrow{x} = (x_1, \ldots, x_n)$ are *decision variables*. The set of all possible solutions is called the *search space* S. In our case, each decision variable corresponds to a VM. The i^{th} decision variable represents the identifier of the PM used for mapping the i^{th} VM. Therefore, a *solution* represents a mapping of all the VMs placed onto the available PMs. The set of all possible mappings is the search space S.

A solution $\overrightarrow{x_1}$ *dominates* another solution $\overrightarrow{x_2}$ (or mathematically $\overrightarrow{x_1} \preccurlyeq \overrightarrow{x_2}$) if it is better in at least one objective and not worse in the rest: $f_i\left(\overrightarrow{x_1}\right) \leq f_i\left(\overrightarrow{x_2}\right), \forall i \in [1, n]$, and $\exists j \in [1, n]$ such that $f_j\left(\overrightarrow{x_1}\right) < f_j\left(\overrightarrow{x_2}\right)$.

The solution of a multi-objective optimization problem is a set of non-dominated solutions representing a tradeoff among the objective functions. The set of tradeoff solutions non-dominated by any other solution in S is called *Pareto optimal set*, and the objective values of the solutions in the Pareto optimal set defines the *Pareto frontier*. A Pareto frontier usually consists of an infinite set of points whose computation is an NP-complete problem. The goal of a multi-objective optimization is to approximate the Pareto optimal set by maximizing two properties: (1) *convergence* by being as close as possible to the Pareto frontier, and (2) *diversity* by uniformly covering the range of tradeoff solutions in the Pareto frontier.

3.3. NSGA-II

Evolutionary algorithms are popular techniques to approximate the Pareto frontier of a multi-objective optimization problem. Among them, NSGA-II [11] is the most popular and well-known in the literature, presented in Algorithm 1. NSGA-II works with a population T of N candidate solutions, initialized in line 1 and improved in convergence and diversity in an iterative process (lines 3 – 14). In each iteration, the algorithm generates a new set Q of solutions (line 9) by means of two operations: crossover (line 7) and mutation (line 8). These operations have their inspiration in the theory of species evolution, and try to exploit the content of T to seek higher quality solutions. The set $R = T \cup Q$ (line 11) generates the population T for the next iteration and uses the `rankingAndCrowding` function (line 12) to arranges the population in different fronts. The first front contains non-dominated solutions, the second front contains only solutions dominated by the first front, and so on. Each front sorts the solutions according to a density measurement metric called *crowding distance* [11] to ensure diversity. The `selectBestIndividuals` function finally selects the best N individuals from R (line 13) aiming to converge towards the Pareto frontier. The algorithm repeats these steps until reaching a termination condition (line 3).

4. Model and Problem Statement

We present the resource model and the objective functions of our method.

4.1. Physical machines (PMs)

We consider a data center composed of m PMs $P = \{p_1, \ldots, p_m\}$. A *resource capacity vector* $\overrightarrow{CV}(p) = (c_1, \ldots, c_v)$ describes each PM $p \in P$, where every dimension $k \in [1, v]$ indicates the capacity of each PM physical resource r_k in the set $R = \{r_1, \ldots, r_v\}$. In a typical Cloud scenario, $R = \{CPU, memory, disk, network\}$, abstracted by the virtualization tech-

Algorithm 1: NSGA-II algorithm

$T \leftarrow$ `initializePopulation()` ; // Initial population.
$R \leftarrow \emptyset$; // Auxiliary population.
while `terminationCondition()` **do**
  $Q \leftarrow \emptyset$; // Offspring population.
  for $i \leftarrow 1$ **to** $\frac{\text{populationSize}}{2}$ **do**
    parents $\leftarrow$ `selection(`T`)`
    offspring $\leftarrow$ `crossover(`parents`)`
    offspring $\leftarrow$ `mutation(`offspring`)`
    $Q \leftarrow$ offspring
  end
  $R \leftarrow T \cup Q$
  `rankingAndCrowding(`R`)` ; // Population sorting.
  $T \leftarrow$ `selectBestIndividuals(`R`)`
end
Result: Non-dominated solutions from T.

nology [5]. Our study focuses on CPU and memory, as the most overcommited resources in data centers and strongly affect the VM migration [19].

4.2. Virtual machines (VMs)

We identify two sets of VMs that participate in the placement process. The *incoming VMs* are new VMs that scale up applications or create new application deployments. The *hosted VMs* are the currently running ones. All together, they define a set $VM = \{vm_1, \ldots, vm_n\}$ placed on an optimized subset of PMs $P_{used} \subseteq P$. Each $vm \in VM$ has two v-dimensional vectors.

Resource size vector. $\overrightarrow{SV}(vm) = (s_1, \ldots, s_v)$ indicates the amount s_k of the resource r_k requested by the VM vm, with $k \in [1, v]$;

Resource demand vector. $\overrightarrow{DV}(vm, t) = (d_1(t), \ldots, d_v(t))$ defines the vm's workload demand $d_k(t)$ for each resource r_k at time instance t, with $k \in [1, v]$.

4.3. VM placement objectives

We defined in Section 3.2 a placement of n VMs as $\overrightarrow{x} = (x_1, \ldots, x_n)$, where each decision variable x_i maps the i^{th} VM $vm_i \in VM$ to a PM $p \in P$. Given $\overrightarrow{x}$, we further define the set of VMs allocated on the same PM $p \in P$ as follows: $\delta(p, \overrightarrow{x}) = \{vm_i \in VM \mid x_i == p\}$.

To facilitate the VBP problem formulation, we normalize (to values in the space $[0, 1]^v$) the vector $\overrightarrow{CV}(p)$ with respect to the resource capacity of the PM p, and the vectors $\overrightarrow{SV}(vm)$ and $\overrightarrow{DV}(vm, t)$ with respect to the capacity of the PM hosting the VM vm. We define in the following the three objective functions targeted by our optimization process.

4.3.1. Objective 1: resource wastage

The first objective function f_1 quantifies the resource wastage over each v resource dimension entailed by a placement $\overrightarrow{x}$. We define the *cumulative demand vector* for a machine p at instant of time t as:

$$\overrightarrow{CDV}(p, t, \overrightarrow{x}) = \sum_{vm \in \delta(p, \overrightarrow{x})} \overrightarrow{DV}(vm, t).$$

This vector aggregates the resource demands of all VMs placed on a PM p at time instance t. If the components of $\overrightarrow{CDV}(p, t, \overrightarrow{x})$ are larger than the ones in the resource capacity vector $\overrightarrow{CV}(p)$ at any time instance t, the cumulative demand exceeds the PM capacity and the VMs contend for PM resources.

We define the *resource wastage vector* for a machine p at instance t as:

$$\overrightarrow{WV}(p,t,\overrightarrow{x}) = \overrightarrow{CV}(p) \ominus \overrightarrow{CDV}(p,t,\overrightarrow{x}),$$

where the operation $\ominus$ has the following definition:

$$\overrightarrow{A} \ominus \overrightarrow{B} = \left(\max\left(A_1 - B_1, 0\right), \ldots, \max\left(A_v - B_v, 0\right)\right).$$

This vector indicates the amount of unused resources in each dimension. When $\overrightarrow{CDV}(p,t,\overrightarrow{x})$ is larger than $\overrightarrow{CV}(p)$ in one dimension, we set $\overrightarrow{WV}(p,t,\overrightarrow{x})$ to 0 in that dimension, as the resource has 100% utilisation and no wastage.

We define the *total wastage vector* at time instance t as the sum of the resource wastage vectors of across all used PMs:

$$\overrightarrow{TWV}\left(\overrightarrow{x}, t\right) = \sum_{p \in P_{used}} \overrightarrow{WV}(p,t).$$

We finally define the resource wastage f_1 as the magnitude of the total wastage vector:

$$f_1\left(\overrightarrow{x}\right) = \left\|\overrightarrow{TWV}\left(\overrightarrow{x}, t\right)\right\|.$$

4.3.2. Objective 2: overcommitment ratio

The second objective function f_2 measures the *overcommitment ratio*, as the percentage of PM resources requested by VMs in excess to their capacity. Given a placement $\overrightarrow{x}$, the overcommitment ratio f_2 adds all the normalized resources size vectors $\overrightarrow{SV}(vm) = (s_1, \ldots, s_v)$ of all the VMs divided by the number of PMs in use:

$$f_2\left(\overrightarrow{x}\right) = \frac{\sum_{p \in P_{used}} \sum_{vm \in \delta(p,\overrightarrow{x})} \sum_{i=1}^{v} s_i}{|P_{used}|},$$

where $P_{used} = \{p \in P \mid \delta(p,\overrightarrow{x}) \neq \emptyset\}$.

4.3.3. Objective 3: migration cost

The third objective f_3 estimates the *migration cost* triggered by a placement $\overrightarrow{x'}$. Following the experimental motivation from Section 1.2, we calculate the VM migration cost as its energy consumption considering the page dirtying rate and the load of the source p_{src} and target p_{trg} PMs [18, 19]:

$$P\left(vm, p_{src}, p_{trg}, t\right) = P\left(vm, p_{src}, t\right) + P\left(vm, p_{trg}, t\right).$$

Therefore, the energy consumption of migrating a VM vm on a PM p is:

$$E(vm, p) = \int_{t_{start}}^{t_{stop}} P(vm, p, t)\,\mathrm{d}t,$$

where $t_{stop} - t_{start}$ is the duration of the migration, as defined in [18]. Given a current placement of n VMs $\overrightarrow{x} = (x_1, \ldots, x_n)$, the cost of migrating them to a new placement $\overrightarrow{x}' = (x'_1, \ldots, x'_n)$ is:

$$f_3\left(\overrightarrow{x'}\right) = \sum_{\substack{i \in [1,n] \wedge \\ x_i \neq x'_i}} E\left(vm_i, x_i\right) + E\left(vm_i, x'_i\right).$$

5. Dynamic VM Placement in Cloud Data Centers

In this section, we first describe the overall software architecture hosting our dynamic VM placement method, implemented in practice as part of the ASKALON Cloud computing environment [13]. Afterwards, we give illustrative examples of typical VM workloads under its operation that benefit from our approach. Finally, we describe the dynamic VM placement algorithm.

5.1. ASKALON Cloud Computing Environment for Real-World Scientific and Industrial Workloads

We carry out this research as part of the ASKALON [13] application development and computing environment for scientific and industrial applications on distributed high-performance Cloud infrastructures. ASKALON supports the scientists and engineers in designing applications as independent tasks or workflows through a number of services that transparently execute them as VMs onto the underlying heterogeneous PMs, as follows:

1. *Execution Engine* is responsible for processing the incoming tasks and prepares them for scheduling, deployment and execution;

2. *Monitoring* service observes the infrastructure workload and provides useful utilization metrics to the other services;

3. *Multi-objective optimisation* applies techniques like the Island NSGA-II proposed in this paper (see Section 6) and identifies the "best" VM to PM mappings based on the user objectives (see Section 4.3);

4. *Decision making* is a manual or automatic procedure that selects the preferred mapping from the set of Pareto optimal mappings;

5. *Dynamic VM placement* uses a simple iterative algorithm to deploy the tasks wrapped in optimised VMs onto the PMs selected by the decision making service (see Section 5.3);

6. *Resource management* optionally allocates or releases resources to minimize the number of active PMs.

We successfully applied ASKALON over the the last two decades together with various domain scientists for running computational intensive workloads on a number of applications, including computational chemistry, meteorology, astrophysics, graphics rendering, and online games.

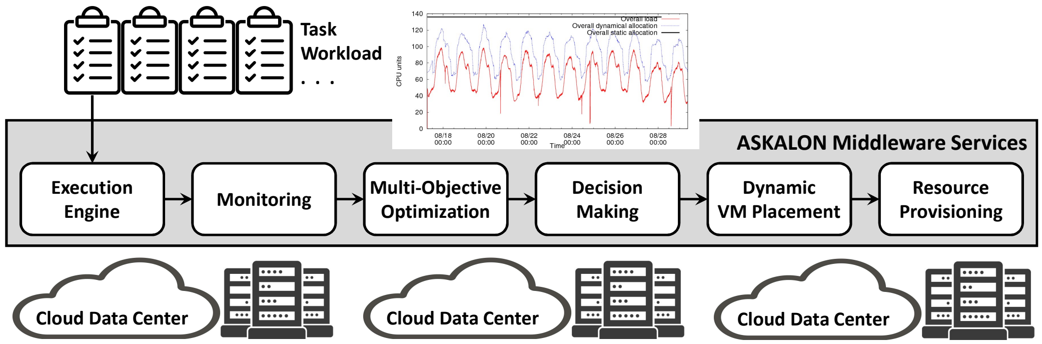


Figure 4: ASKALON Cloud computing architecture.

5.2. *Illustrative Target Workload Examples*

We target elastic predictable workloads running on data centers, such as periodic workloads. Periodic workloads are common in real-world, for example in business applications performing (monthly, yearly) auditing or balance computations, in transportation systems experiencing typical rush hours and idle times (during night), and in massively multiplayer online game (MMOG) servers with high player activity during afternoon and low numbers of connected player after midnight up (to early morning) [26].

Our design follows a successful preliminary work on dynamic resource provisioning and VM placement for MMOGs, which achieved a 250% improvement in resource provisioning for RuneScape [26]. To guarantee a seamless interaction to all players at all times, we triggered in this work a simple dynamic VM placement algorithm (based on resource matchmaking) with a high frequency of two minutes using a low overhead live migration functionality with a low impact on the QoS below 3%. In such scenarios, the infrastructure operators typically perform dynamic resource management at regular intervals (e.g. hourly) to optimize their utilization, depending on

Algorithm 2: Dynamic VM placement algorithm.

Input : $P = \{p_1, \ldots, p_m\}$; `// Set of PMs`

Input : $VM_0 = \{vm_1, \ldots, vm_n\}$; `// Initial set of VMs`

Input : $\overrightarrow{x_0}$; `// Initial placement`

1 $t \leftarrow 0$

2 $\overrightarrow{x} \leftarrow \overrightarrow{x_0}$

3 **while** $t \leq t_{end}$ **do**

4     $VM_{t+1} \leftarrow VM_t + VM_{new}$; `// New VM set`

5     $S \leftarrow$ `Island-NSGAII`$(P, VM, \overrightarrow{x})$; `// Pareto optimal set`

6     $\overrightarrow{x} \leftarrow$ `decisionMaking`(S); `// Preferred solution`

7     $t \leftarrow t + 1$; `// Next time instance`

8 **end**

historical variable resource use at specific times of the day, week, month or year. Within a certain provisioning interval, the operators perform occasional adaptive VM placements depending on the dynamic CPU and memory load exhibited by individual VMs. The appropriate VM placement interval is workload dependent and can range from minutes to hours, days or longer.

5.3. Dynamic VM Placement Algorithm

Algorithm 2 describes our dynamic VM placement method with three input parameters: 1) the set P of PMs in the data center described by their capacity vector $\overrightarrow{CV}$, 2) the initial set of VMs VM described by their resource size $\overrightarrow{SV}$ and demand After approximating the Pareto frontier, a `decisionMaking` function selects in line 6 a preferred tradeoff VM placement. This procedure can be either manual or automatic based on environment-specific rules or constraints defined by the resource provider, such as minimizing the wastage without QoS violations or keeping VM migration cost bellow a server energy fraction. $\overrightarrow{DV}$ vectors, and 3) an initial placement $\overrightarrow{x_0}$ representing the initial state of the data center. The algorithm iterates over a series of timestamps to periodically optimize the VM placements according to the time-varying VM resource demands (lines 3–8). At each timestamp, it merges the set of incoming VMs VM_{new} with the already hosted ones VM_t into a new set VM_{t+1} (line 4). Afterwards, the main `IslandNSGA-II` function (line 5) implemented as an evolutionary multi-objective optimization heuristic periodically computes an approximation to the Pareto optimal set of possible VM placements onto the available PMs. This approximation contains "optimal" tradeoffs among the three objectives described in Section 4.3.

6. Island NSGA-II Algorithm

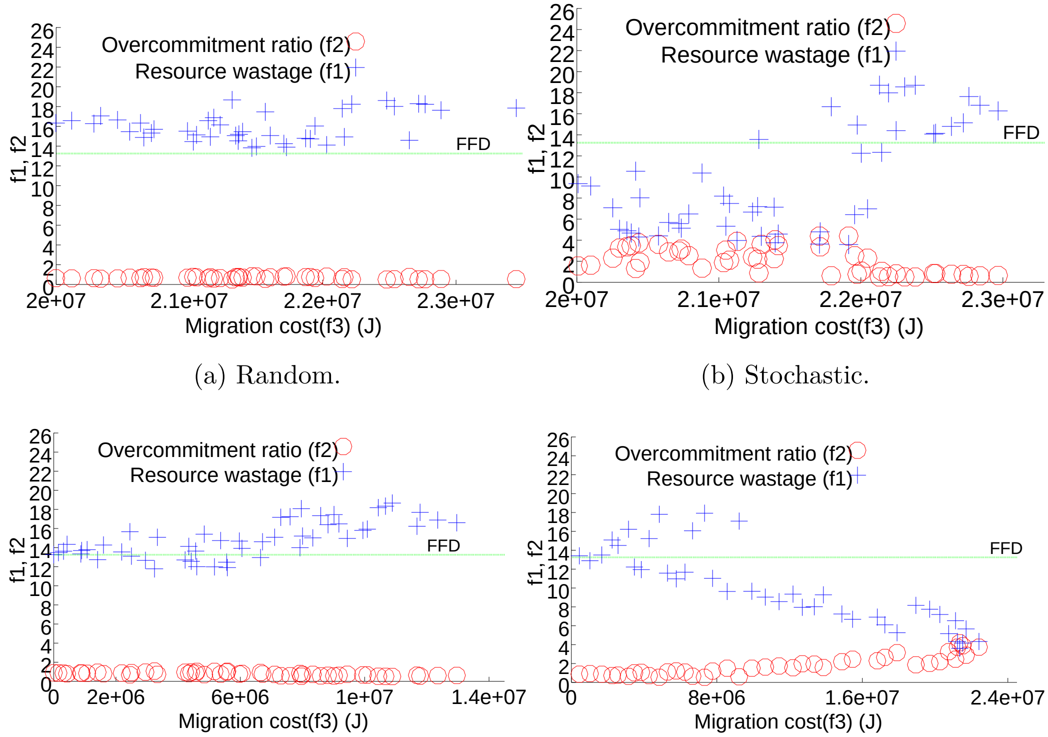
We research in this section an evolutionary algorithm that improves the convergence and diversity of NSGA-II [11] presented in Section 3.3 to determine the Pareto optimal set of tradeoff placements, modelled as a generalization of the NP-complete VBP problem [15] (see Section 3.1). Every VM placement is a vector $\overrightarrow{x} = (x_1, \dots, x_n)$, as defined in Section 4. NSGA-II works with a population T of candidate VM placements, randomly initialised by selecting a random PM $p \in P$ for each decision variable x_i of each placement $\overrightarrow{x} \in T$. We employ a single-point crossover operator that selects two placements $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$ from the population T, and generates a new placement $\overrightarrow{x_3}$ by combining the first half of $\overrightarrow{x_1}$ with the second half of $\overrightarrow{x_2}$. The mutation operator takes the new placement created by crossover, randomly selects a

VM, and changes its placement to a random PM.

6.1. *Pareto analysis*

The generated Pareto frontier has three dimensions corresponding to the three objective functions. To facilitate the analysis, we use two-dimensional representations of the Pareto frontiers mapping the resource wastage (f_1) and the overcommitment ratio (f_2) on one unit-less y-axis and the VM migration energy (in J) on the x-axis. For example, Fig. 5 represents the overcommitment ratio (blue cross) and resource wastage (red circle) as comparable Pareto frontiers. A placement $\overrightarrow{x}$ representing a tradeoff between a resource wastage $f_1(\overrightarrow{x})$, an overcommitment ratio $f_2(\overrightarrow{x})$ and a migration energy $f_3(\overrightarrow{x})$ is a (blue) cross at coordinates $(f_3(\overrightarrow{x}), f_1(\overrightarrow{x}))$ and a red circle at coordinates $(f_3(\overrightarrow{x}), f_2(\overrightarrow{x}))$. We can therefore conclude that the Pareto frontier in Fig. 5b provides a lower resource wastage than the one in Fig. 5a. In addition, a horizontal (green) line represents the solution computed by the FFD [25] introduced in Section 2.

Fig. 5a displays the outcome of the NSGA-II algorithm for the homogeneous scenario described in Section 7.1 at the first time instance. We compute the initial VM placement $\overrightarrow{x_0}$ using the FFD baseline method (see Section 2.1), which leads to a high resource wastage. Moreover, Fig. 5a shows that NSGA-II produces solutions that do not improve much the resource wastage, even for placements involving a lot of migrations (indicated by high energy consumption towards the right part of the x-axis.)



(a) Random.

(b) Stochastic.

(c) Biased stochastic.

(d) Island.

Figure 5: Pareto optimal sets for different generation methods of the initial population.

6.2. *NSGA-II with stochastic initial population*

To lower the resource wastage, we aim to improve the results of NSGA-II by initializing the population T with placements with minimum resource wastage that map all VMs onto one single PM from the set P. The idea is to start from optimal placements respect to the first objective f_1 and explore the solution space for finding better solutions with respect to resource overcommitment f_2 and migration cost f_3. For generating the initial population, we first calculated the rate $\eta_p = \mathcal{F}\left(|\delta(p, \overrightarrow{x_0})|, \epsilon\right)$ of expected placements on each PM $p \in P$, where $\eta_p, \epsilon \in (0, 1]$, $|\delta(p, \overrightarrow{x_0})|$ is the number of VMs currently placed on p, and $\sum_{p \in P} \eta_p = 1$. The parameter ϵ guarantees proportional placements on a PM p, which are not present in the current placement (i.e., $|\delta(p, \overrightarrow{x_0})| = 0$). Afterwards, we simulated a roulette wheel with $m = |P|$ slots of size proportional to $\eta_p, \forall p \in P$, spun the wheel $|T|$ times, and created each time a VM placement onto the winning PM p.

The Pareto frontier in Fig. 5b improves on the original NSGA-II algorithm with several low resource wastage placements coming at higher overcommitment and migration costs (crowded towards the right of the x-axis.)

6.3. NSGA-II with biased stochastic initial population

To lower the migration costs, we insert the current placement in T, which introduces a bias towards the region of the Pareto optimal set with a low migration cost. Fig. 5c shows that, although the algorithm provides solutions similar to the current placement (i.e. low migration cost), they are far from the stochastic approach in resource wastage (or overcommitment).

6.4. Island NSGA-II

To increase the diversity of the population and converge to wider variety of better solutions, we employed the island model, which conceptually consists of several populations (the islands) evolving independently of each other, potentially using different algorithms and occasionally exchanging individuals. Algorithm 5 considers two islands corresponding to the stochastic and biased stochastic generation of the initial population, initialised in lines 6 and 7. At every iteration (lines 5 – 14), we gather the populations of each island, merge them (line 8), extract the best individuals according to ranking and crowding metrics [11] (line 9), and update the current population of each

Algorithm 3: Island NSGA-II algorithm.

Input : $P = \{p_1, \ldots, p_m\}$; `// PM set`

Input : $VM = \{vm_1, \ldots, vm_n\}$; `// Initial VM set`

Input : $\overrightarrow{x}$; `// Current placement`

$T1 \leftarrow \emptyset$; `// First island`

$T2 \leftarrow \emptyset$; `// Second island`

$i \leftarrow 0$

while $i \leq i_{\max}$; `// Iterate for` $i_{\max}$ `generations`

do

  $T1 \leftarrow$ `stochasticGeneration(`$\overrightarrow{x}, P, VM, T1$`)`

  $T2 \leftarrow$ `biasedGeneration(`$\overrightarrow{x}, P, VM, T,$`)`

  $Q \leftarrow T1 \cup T2$

  `rankingAndCrowding(`Q`)`

  $T \leftarrow$ `selectBestIndividuals(`Q`)`

  $T1 \leftarrow T$

  $T2 \leftarrow T$

  $i \leftarrow i + 1$

end

island accordingly. This method forces individual islands to further explore solutions on the entire Pareto frontier and avoids focus on local areas only.

Fig. 5d shows that the island algorithm finds better tradeoff placements, ranging from few migrations and similar resource wastage to many migrations and substantially different wastage (or overcommitment).

6.5. Complexity analysis

MM_MBDF, MM_MBDF_2 and FFD have an $O(n \cdot m)$ complexity [7], where m is the number of PMs and n is the number of VMs. The Island NSGA-II algorithms consist of two phases. The first phase uses FFD to compute an initial solution with $O(m \cdot n)$ complexity. The second phase is a classical NSGA-II algorithm with a complexity of $O\left(o \cdot p^2\right)$ [11], where o is the number of objectives and p is the population size. Since our problem has three objectives ($m = 3$), this results in an overall complexity of $O(m \cdot n + p^2)$. Related work [12] demonstrated that the population size does not need to scale in the same magnitude as the decision variable vector (i.e. $p \ll n$) leaving our Island NSGA-II algorithm with quadratic complexity of $O(m \cdot n)$.

7. Experimental Evaluation

We first evaluate our method as a decision making tool in Section 7.3. Secondly, we compare it with other VM placement solutions in Section 7.4.

7.1. Experimental setup

We conducted the experiments using the GroudSim [28] discrete event-based simulator for Cloud environments, extended to consider CPU and memory overcommitment through mechanisms such as memory reclamation and CPU-proportional fair scheduling [5, 27].

We simulated a data center with 200 PMs in two configurations displayed in Table 1: (1) *homogeneous* with PMs of type `M2` only, and (2) *heterogeneous* four PM types (`M1`, `M2`, `M3` and `M4`) of 50 PMs each. We set the initial number of VMs to 500 and computed the initial VM placement $\overrightarrow{x_0}$ using the

Table 1: Experimental data center.

PM type	*Virtual CPUs*	*RAM*	*Power [idle]*	*Power [100%]*
`M1`	32 (16×Opteron 8356)	32 GB	501 W	840 W
`M2`	40 (10×Xeon E5-2690v2	128 GB	164.2 W	382 W
`M3`	32 (8×Xeon E5-2660	32 GB	90 W	310 W
`M4`	32 (8×Xeon E5-2660	32 GB	105 W	340 W

basic FFD algorithm (see Section 2.1). Afterwards, we added and removed a random number of VMs at different time instances to simulate a real Cloud environment where users deploy and stop VMs in an unpredictable manner, as researched in [8].

We simulated one full day of data center operation in each experiment. We set the interval between two periodic Pareto frontier computations to 30 min, as Google cluster traces exhibit a long-term resource demand variability [29] and a stable task resource demand within an hourly time period. We selected the size and the workload of each VM using the Google cluster traces [35] containing the resource use of 25,462,157 tasks over a period of 29 days. Every task run on a separate VM and requested a maximum percentage of the PM resources. After deploying a simulated VM, we randomly selected a task and determined its VM size and workload demand by the maximum requested resources and by its resource use. The number of VMs migrated at each time instance depends on the amount of CPU load [21]. Our workload is typical to Web applications, whose load ranges from $10.7 - 87.6\%$ with 62.8% median [1] following a diurnal pattern (i.e. the load reaches its peak during daytime hours), as shown in Figure 4.

We collected the VM resource demand at a five second sampling rate. Upon each VM placement, we determined the resource demand vector $\overrightarrow{DV}$ by averaging the collected resource demand over the considered period using an exponential moving average, which makes the computation of the resource wastage vector $\overrightarrow{WV}$ robust to workload demand oscillations.

The simulation aims to evaluate our dynamic VM placement algorithm using a large number of parameters and situations that are not easily reproducible in real life. Considering that the Google traces account for 29 days real execution, performing the same Pareto analysis using real experimentation requires several years of execution. Apart from the workload injection, our algorithm uses precise resource information with no stochastic variables involved, indicating that the simulation matches the real execution.

7.2. Evaluation metrics

We use five metrics in our experimental evaluation:

- *Total energy consumption* accounts for the CPU power consumption P_p consumed by all PMs $p \in P_{used}$, considering CPU as the most energy consuming resource according to [20], proportional to its utilization [23]:

$$E = \sum_{p \in P_{used}} \int_0^{t_s} P_p(t) \cdot \mathrm{d}t,$$

  where: $P_p(t) = \left(P_p^{\text{max}} - P_p^{\text{idle}}\right) + U_p(t) \cdot P_p^{\text{idle}}$, t_s is the simulation time, P_p^{max} and P_p^{idle} are the power consumptions of the PM p at maximum and idle utilization levels (see Table 1), and $U_p(t)$ is p's CPU utilization at instance t (i.e. complement of CPU component of wastage vector $\overrightarrow{WV}(p, t)$);

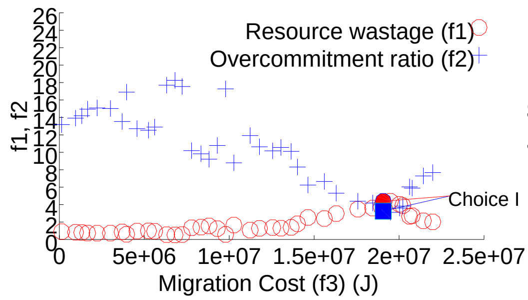
- *QoS violations* is the percentage of VMs that receive less resources than their current demand relative to the entire simulation time;
- *Average number of active PMs* during the complete simulation;
- *Energy consumption of VM migration* is the energy consumed due to live migrations (i.e. objective f_3 in Section 4.3). as a separate objective of our study motivated in Section 1.2;
- *Average overcommitment ratio* is the average amount of resources allocated in excess to the overall PM capacities (i.e. objective f_2 in Section 4.3).

7.3. Dynamic VM placement

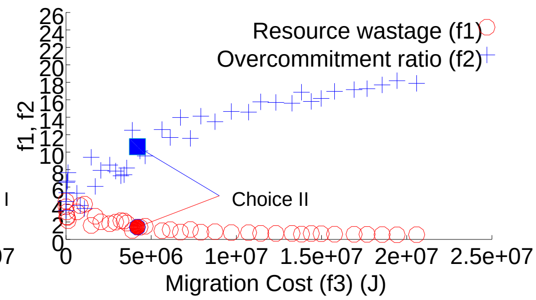
We theoretically evaluated the adaptation of the dynamic VM placement method by considering a decision making procedure triggered at the second, ninth, and fourteenth hour of simulation, labeled as *Choice I*, *II*, and *III*. Fig. 6 displays the Pareto frontiers obtained before and after each choice using a two-dimensional graphical representation explained in Section 3.3.

Choice I (Fig. 6b) taken after the first hour of simulation incurs a higher cost of migration leading to a low resource wastage. This placement results in a reduction of the power consumption by up to 80% by turning 92 PMs to a low power state (Fig. 7a). The consolidation of the VMs on the remaining 32 active PMs brings a 9% increase in QoS violations (Fig. 7c), which may result in a high penalty for the Cloud provider under strict QoS requirements As a consequence, the decision maker has the option to select a more energy costly placement offering a better QoS.

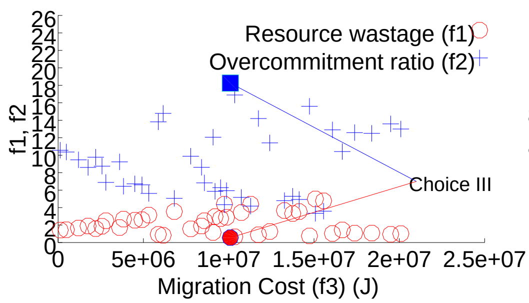
Choice II (Fig. 6c) taken after the fourth hour of simulation incurs a higher resources wastage and a lower overcommitment than the previous



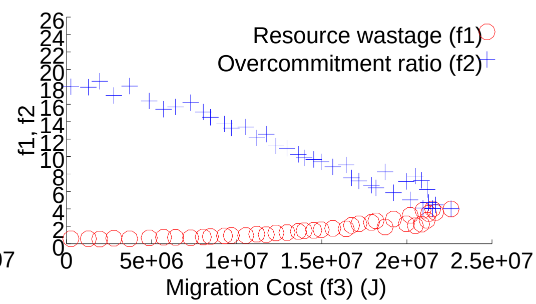
(a) Initial Pareto front.



(b) After *Choice I*.



(c) After *Choice II*.



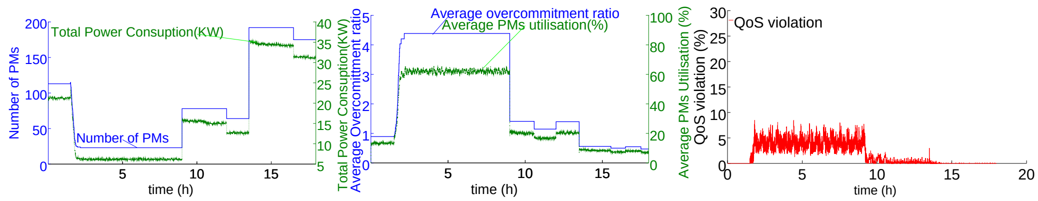
(d) After *Choice III*.

Figure 6: Pareto frontier generated by different VM tradeoff placements.

placement. Fig. 7c shows that the lower overcommitment ratio reduces the QoS violations to 1.2%, however, the utilization drops to 20% due to fewer consolidated VMs onto the same PM (Fig. 7b).

Choice III (Fig. 6d) taken after the ninth hour of simulation further increases the resources wastage. All 200 PMs present in the data center become active (Fig. 7a) and the power consumption increases by 130%. Fig. 7c shows the benefit on QoS of this rather expensive choice.

Following our experiments, we observe that for an overall resource load below 80%, at most 10% of maximum number of VMs are migrated, with peaks of 40% when system is fully loaded, as typical in other setups which



(a) Power consumption versus active PMs. (b) Overcommitment ratio versus resource wastage. (c) QoS violations.

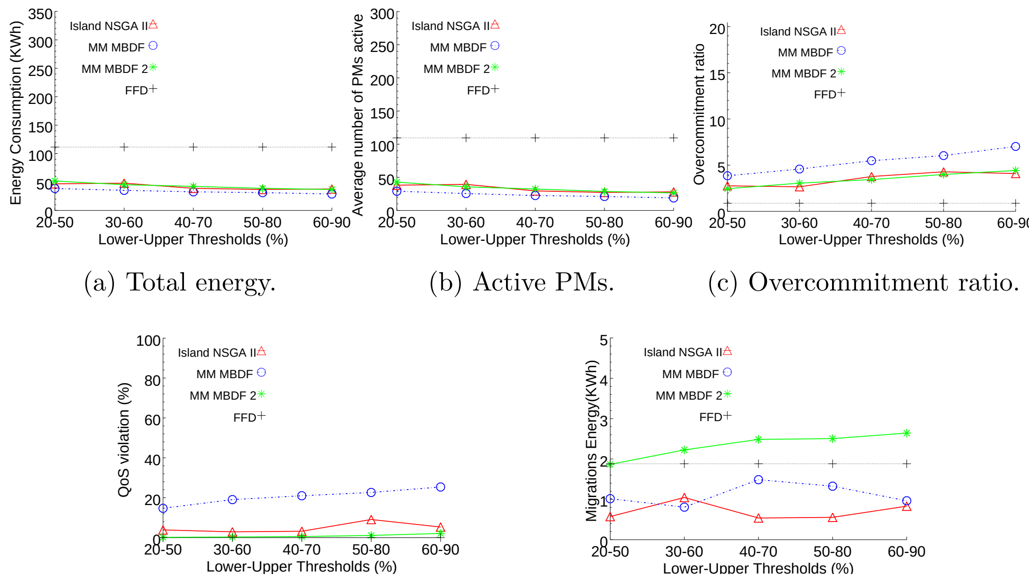
Figure 7: VM provisioning metrics after each tradeoff placement from Fig. 6.

exhibit similar load patterns [21]. In most cases, however, number of VM migrations is between $3 - 10\%$ of the maximum number of VMs, which has a limited impact on performance (i.e. QoS violations).

7.4. *Island NSGA-II*

We compare in this section the Island NSGA-II algorithm against two state-of-the-art VM placement algorithms presented in Section 2.1:

- FFD as a baseline comparison method used by nearly all related works in evaluating their VM consolidation algorithms. We employ FFD by placing VMs according to their resource demand rather than request.
- MM_MBFD as one of the most cited and most recent dynamic placement algorithms in the literature that considers a trade-off between QoS and energy consumption metrics, similar to us;
- MM_MBDF_2 as our own extension to MM_MBFD that includes memory utilization for a fair comparison, not considered in the original version [7]. We experimented with different MM_MBDF and MM_MBDF_2 upper and lower thresholds with a 30% difference.



(a) Total energy. (b) Active PMs. (c) Overcommitment ratio.

(d) QoS violations. (e) VM migration.

Figure 8: Simulation results for homogeneous data center.

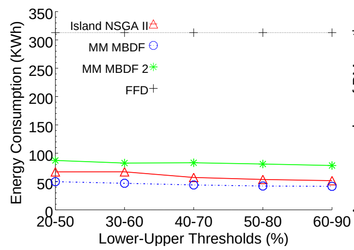
As these algorithms generate a single placement instead of a Pareto frontier, we cannot consider multi-objective metrics such as the hypervolume [4] in their comparison. For a fair comparison of a single tradeoff placement, we select the solution on the Pareto frontier closest to MM_MBDF_2 that, similar to us, considers both memory and CPU utilization in its optimization.

Among the most recent related works (see Section 2), we do not consider the ACO-based approach in [14] because it focuses only on reducing power consumption, rather than on finding trade-off solutions. Similarly, we do not compare our method with [30], as it optimizes the offline VM placement.

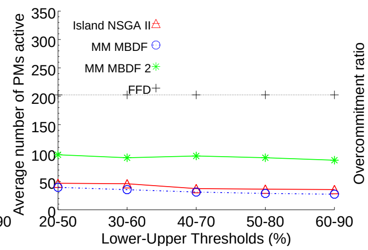
7.4.1. Homogeneous data center (200 PMs of type `M2`)

MM_MBDF reduced the total energy consumption by 21% in average compared to Island NSGA-II (see Fig. 8a) by exclusively placing VMs according to their CPU demand and overcommiting the memory, which lowering the number of active PMs by 30% (see Fig. 8b). Fig. 8c shows that MM_MBDF achieved an overcommitment ratio 48% higher than MM_MBDF_2 in average, and 45% higher than Island NSGA-II. The side effect is an increase in QoS violations, as displayed in Fig. 8d. MM_MBDF_2 reduced the QoS violations by 20% in average compared to MM_MBDF by jointly considering both CPU and memory demands. Island NSGA-II performed close to MM_MBDF_2 with respect energy consumption (see Fig. 8a), average active PMs (see Fig. 8b), and overcommitment ratio (see Fig. 8c). With respect to QoS, Fig. 8d shows that Island NSGA-II exhibited the same level of violations as MM_MBDF_2 in low-consolidated scenarios, and deviated by no more than 6% in high-consolidated ones. In addition, it reduced the migration cost by 70% compared to MM_MBDF_2, by 40.6% compared to MM_MBDF, and by 64% compared to FFD (see Fig. 8e). Finally, FFD produced energy-inefficient placements consuming 110 kWh regardless of the lower and upper thresholds, since it allocated VMs according to their static resource requests that suffer from overprovisioning.

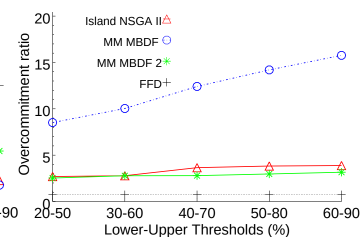
We conclude that MM_MBDF reduces the energy consumption while incurring higher QoS violations than its memory-aware version. On the other hand, Island NSGA-II computes VM placements close in performance to MM_MBDF_2, while decreasing the migration energy by up to 70%.



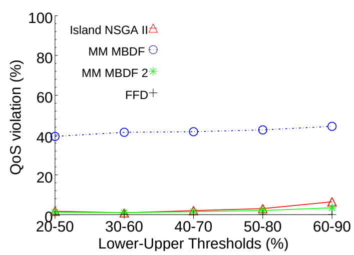
(a) Energy consumption.



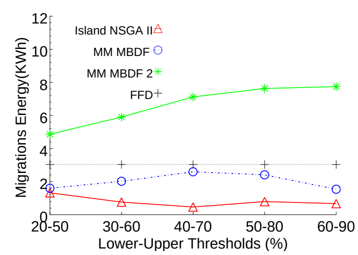
(b) Number of active PMs.



(c) Overcommitment ratio.



(d) QoS violations.



(e) VM migration.

Figure 9: Simulation results for heterogeneous data center.

7.4.2. Heterogeneous data center (50 PMs of each type `M1` – `M4`)

In the heterogeneous data center Island NSGA-II decreased the energy consumption by 41% compared to MM_MBDF_2 and by 63% compared to FFD (see Fig. 9a) by reducing the number of active PMs (see Fig. 9b), while keeping the QoS violations below 6% (see Fig. 9d). Fig. 9b shows that Island NSGA-II was able to use up to 50% less PMs than MM_MBDF_2, and up to 75% less than FFD (Fig. 9b). Interestingly, Island NSGA-II achieved an energy consumption close to MM_MBDF by overcommitting 71% less resources in average (see Fig. 9c) and bringing 40% less QoS violations (see Fig. 9d). Finally, Island NSGA-II significantly reduced the migration energy by 76% compared to MM_MBDF_2, by 38% compared to MM_MBDF, and by 62% compared to FFD, similar to the homogeneous experiment.

8. Conclusions and Future Work

The effective tradeoff between resource wastage and overcommitment is a challenging task, and essential for reducing the energy cost of operating a data center while guaranteeing the QoS. A Cloud data center approaches this challenge by placing VMs to the available PMs. This paper addresses this complex research problem by bringing the following scientific contributions: 1) a multi-objective formulation of the dynamic VM placement problem that uses resource wastage, overcommitment ratio, and migration cost to represent the energy and QoS tradeoffs; 2) an island-based evolutionary meta-heuristic to approximate the set of Pareto tradeoff VM placements; 3) a dynamic VM placement algorithm which supports decision making operators in optimizing VM placements according to energy and QoS constraints.

We demonstrated using real traces from a Google data center cluster that single solution VM placement approaches, although very appealing to formulate, understand and apply, are not effective compared to our multi-objective approach that approximates the Pareto optimal set that reveals uncovered regions of resource wastage, overcomittment, and live migration tradeoffs. Our algorithm is linear in complexity with the number of VMs and PMs and does not necessarily require human intervention for selecting prefered VM; placement tradeoffs from the Pareto frontier. A basic decision making with constant complexity can simply consider "energy budgets" or QoS constraints projected onto the Pareto optimal set of tradeoff placements. Moreover, multi-objective optimization literature showed that approximating the complete set of tradeoff solutions is in many cases cheaper than computing a single solution due to different navigaton of the search space. The Island

NSGA-II heuristic demonstrates performance close to other approaches and exhibits less than 6% more QoS violations, while significantly reducing the migration energy consumption by 55% in a homogeneous data center, and by 57% in a heterogeneous data center.

In future work, we intend to extend the dimensions of our problem by taking into account network and disk resources. We also plan to research automated decision making strategies to automate the selection of the "best" Pareto solution during the dynamic VM placement process. Another interesting work is to understand the interplay between the placement optimization interval and the data center overall dynamics, such as migration time and PM transition latency to a different (i.e. normal, low) power state.

Acknowledgements

This work received funding from:

- European Union's Horizon 2020 research and innovation programme, grant agreement 825134, "Smart Social Media Ecosytstem in a Blockchain Federated Environment (ARTICONF)";
- Austrian Science Fund (FWF), grant agreement Y 904 START-Programm 2015, "Runtime Control in Multi Clouds (RUCON)";
- Austrian Agency for International Cooperation in Education and Research (OeAD-GmbH) and Indian Department of Science and Technology (DST), project number, IN 20/2018, "Energy Aware Workflow Compiler for Future Heterogeneous Systems".

References

[1] D. Aikema, C. Kiddle, and R. Simmonds. Energy-cost-aware scheduling of HPC workloads. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–7. IEEE, 2011.

[2] S. Akiyama, T. Hirofuchi, and S. Honiden. Evaluating impact of live migration on data center energy saving. In *6th International Conference on Cloud Computing Technology and Science*, pages 759–762. IEEE, 2014.

[3] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper. Predicting the performance of virtual machine migration. In *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 37–46, 2010.

[4] C. L. Bajaj, V. Pascucci, G. Rabbiolo, and D. R. Schikore. Hypervolume visualization: a challenge in simplicity. In *IEEE Symposium on Volume Visualization*, pages 95–102. IEEE, 1998.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5e):164–177, 2003.

[6] S. A. Baset, L. Wang, and C. Tang. Towards an understanding of oversubscription in cloud. In *2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, page 7. USENIX Association, 2012.

[7] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012.

[8] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti. VM placement strategies for cloud scenarios. In *IEEE 5th International Conference on Cloud Computing*, pages 852–859. IEEE, 2012.

[9] R. N. Calheiros, R. Ranjan, and R. Buyya. Virtual machine provisioning based on analytical performance and QoS in cloud computing environments. In *2011 International Conference on Parallel Processing*, pages 295–304. IEEE, 2011.

[10] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes. Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment. *Network, IEEE*, 29(2):56–61, 2015.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[12] J. J. Durillo, A. J. Nebro, C. A. C. Coello, F. Luna, and E. Alba. A comparative study of the effect of parameter scalability in multi-objective metaheuristics. pages 1893–1900, April 2008.

[13] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazón, and M. Wieczorek. ASKALON: A development and grid computing environment for scientific workflows. In I. J. Taylor, E. Deelman, D. B.

Gannon, and M. Shields, editors, *Workflows for e-Science*, pages 450–471. Springer, 2007.

[14] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya. Virtual machine consolidation in cloud data centers using aco metaheuristic. In *Euro-Par 2014 Parallel Processing*, pages 306–317. Springer, 2014.

[15] D. S. Johnson. *Encyclopedia of Algorithms*, chapter Vector Bin Packing, pages 1–6. Springer, 2008.

[16] P. Lama and X. Zhou. aMoss: Automated multi-objective server provisioning with stress-strain curving. In *2011 International Conference on Parallel Processing*, pages 345–354. IEEE, 2011.

[17] J. Z. Li, M. Woodside, J. Chinneck, and M. Litoiu. CloudOpt: multi-goal optimization of application deployments across a cloud. In *7th International Conference on Network and Services Management*, pages 162–170. IFIP, 2011.

[18] H. Liu, H. Jin, C.-Z. Xu, and X. Liao. Performance and energy modeling for live migration of virtual machines. *Cluster Computing*, 16(2):249–264, 2013.

[19] V. D. Maio, G. Kecskemeti, and R. Prodan. A workload-aware energy model for virtual machine migration. In *IEEE International Conference on Cluster Computing*, pages 274–283. IEEE, 2015.

[20] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz. Towards energy-proportional datacenter memory with

mobile dram. In *2012 39th Annual International Symposium on Computer Architecture*, pages 37–48. IEEE, 2012.

[21] C. Mastroianni, M. Meo, and G. Papuzzo. Self-economy in cloud data centers: Statistical assignment and migration of virtual machines. In *Euro-Par 2011 Parallel Processing*, volume 6852, pages 407–418. Springer, 2011.

[22] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *2010 IEEE International Conference on Services Computing*, pages 514–521. IEEE, 2010.

[23] L. Minas and B. Ellison. *Energy efficiency for information technology: How to reduce power consumption in servers and data centers*, volume pre. Intel Press, 2009.

[24] M. Mishra and A. Sahoo. On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *2011 IEEE International Conference on Cloud Computing*, pages 275–282. IEEE, IEEE Computer Society, 2011.

[25] A. Murtazaev and S. Oh. Sercon: Server consolidation algorithm using live migration of virtual machines for green computing. *IETE Technical Review*, 28(3):212–231, 2011.

[26] V. Nae, A. Iosup, and R. Prodan. Dynamic resource provisioning in massively multiplayer online games. *IEEE Transactions on Parallel and Distributed Systems*, 22(3):380–395, 2011.

[27] J. Nieh, C. Vaill, and H. Zhong. Virtual-time round-robin: An O(1) proportional share scheduler. In *USENIX Annual Technical Conference*, pages 245–259. USENIX Association, 2001.

[28] S. Ostermann, K. Plankensteiner, and R. Prodan. Using a new event-based simulation framework for investigating resource provisioning in clouds. *Scientific Programming*, 19(2-3):161–178, 2011.

[29] 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 *Third ACM Symposium on Cloud Computing*, page 7. ACM, 2012.

[30] A. Sallam and K. Li. A multi-objective virtual machine migration policy in cloud systems. *The Computer Journal*, 57(2):195–204, 2014.

[31] A. Strunk and W. Dargie. Does live migration of virtual machines cost energy? In *27th International Conference on Advanced Information Networking and Applications*, pages 514–521. IEEE, March 2013.

[32] S. Takahashi, A. Takefusa, M. Shigeno, H. Nakada, T. Kudoh, and A. Yoshise. Virtual machine packing algorithms for lower power consumption. In *4th International Conference on Cloud Computing Technology and Science*, pages 161–168. IEEE Computer Society, 2012.

[33] H. N. Van, F. D. Tran, and J.-M. Menaud. Performance and power management for cloud infrastructures. In *3rd International Conference on Cloud Computing*, pages 329–336. IEEE, 2010.

[34] A. Verma, P. Ahuja, and A. Neogi. pMapper: power and migration

cost aware application placement in virtualized systems. In *Middleware 2008*, pages 243–264. Springer, 2008.

[35] J. Wilkes and C. Reiss. ClusterData2011. `https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md`.

[36] M. Xu, L. Cui, H. Wang, and Y. Bi. A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. In *Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on*, pages 629–634. IEEE, 2009.