Journal of Network and Systems Management (2024) 32:2
https://doi.org/10.1007/s10922-023-09775-8

A Distributed Virtual-Machine Placement and Migration Approach Based on Modern Portfolio Theory

Manoel C. Silva Filho[1,2] · Claudio C. Monteiro[1] · Pedro Ricardo M. Inácio[2] · Mário M. Freire[2]

Received: 4 May 2023 / Revised: 4 May 2023 / Accepted: 29 August 2023 /
Published online: 25 October 2023

Abstract
Virtual machine placement and migration (VMPM) are key operations for managing cloud resources. Considering the large scale of cloud infrastructures, several proposals still fail to provide a comprehensive and scalable solution. A variety of approaches have been used to address this issue, e.g., the modern portfolio theory (MPT). Originally formulated for financial markets, MPT enables the construction of a portfolio of financial assets in order to maximize profit and reduce risk. This paper presents a novel VMPM approach applying MPT and incremental statistics computation for VMPM decision-making so as to maximize resource usage while minimizing under and overload. Extensive simulation experiments were conducted using CloudSim Plus, relying on synthetic data, PlanetLab and Google Cluster traces. Results show that the proposal is highly scalable and largely reduces computational complexity and memory footprint, making it suitable for large-scale cloud service providers.

Keywords VM placement · VM migration · Modern Portfolio Theory · MPT · Cloud computing · CloudSim Plus

1 Introduction

Virtual Machine Placement and Migration (VMPM) involves processes that need to deal with a large number of different aspects in managing cloud resources. Some factors that play an important role in decision making are resource balancing, overload, costs and overall gains after performing such processes. The workload for cloud-hosted services is usually dynamic and unpredictable. Despite the apparently infinite cloud computing resources, they have to be properly managed

Claudio C. Monteiro, Pedro Ricardo M. Inácio and Mário M. Freire have contributed equally to this work.

Extended author information available on the last page of the article

to ensure that the demand and supply relation is met. Considering the scale of cloud infrastructure and the NP-hardness of VMPM, several works are limited to providing a comprehensive and scalable solution for these concerns such as those surveyed in our previous work [1] and many other [2–8].

In that regard, this paper proposes a new distributed approach for VM placement and migration based on the modern portfolio theory (MPT) developed by Markowitz (1959) [9]. The theory is applied for selection and management of investment portfolios, which are investment wallets containing a list of financial assets. These assets, such as stocks, represent the portfolio as a whole and each one is accountable for a specific percentage of the overall portfolio value. Values of assets are dynamically determined and difficult to predict, as it is the resource demand in cloud computing. That requires the shares of each asset in a portfolio to be rebalanced regularly, to ensure that the percentage of each asset remains the same as expected by the investor, according to his/her risk preferences. This is known as the risk profile [10].

A good portfolio is based on multiple assets that are aligned with investor goals. Assets can be selected strategically or tactically, respectively based on: (i) past information, which might give an overview of asset performance over time; (ii) and/or forecasts, beliefs regarding future asset performance, market information or trends [9].

The main MPT foundation is the portfolio diversification effect to reduce risk, while either yielding the (i) highest expected return for a given level of risk or (ii) lowest risk for a given level of return. It applies the mean-variance analysis to select assets, usually seeking to maximize the return mean (profit) and minimize return variance (risk). This analysis strongly relies on asset covariance (correlation) to determine how different assets fit together, so that the goals just mentioned are met.

Asset selection takes into account the imperfect correlation between pairs of assets, computed based on historical data. When there is a positive correlation between two assets, their expected values tend to change in the same direction. A less risky portfolio can be compounded by pairs of negatively correlated assets, so that they move in opposite ways; when one loses value, the other one gains, reducing risk and balancing returns. In practice, the correlation is not perfect, since assets change values at different scales (speeds).

The risk associated with assets is defined as the volatility of their prices, formally the variance of the returns. The more volatile an asset is, the higher its risk, although higher returns are usually earned. Due to the volatility and the often unpredictable nature of financial assets, it is challenging to create a portfolio that ensures a certain level of risk and return. That is why portfolios usually have to be re-evaluated and rebalanced periodically, even to add or remove some assets.

Regarding cloud computing, cloud resources are physical or logical assets (RAM, CPU, bandwidth, storage, etc.) that have different returns for cloud providers and customers, respectively, e. g., (i) profit and (ii) quality of service (QoS). Management of cloud assets by means of VM placement and migration also involves the same issues faced by financial assets, such as the volatility of returns. One impacting factor for volatility is the unpredictable demand that affects both cloud and financial markets.

Modern portfolio theory has applications in different fields such as project portfolio management by government and industry. It has been applied to create a portfolio of imperfectly correlated projects that minimize risks and maximize returns for a company [11]. Since MPT is widely used to optimize allocation of assets in a portfolio and to manage risk and return, this paper presents an approach that applies the theory to create cloud computing portfolios, each one representing a suboptimal allocation of a set of VMs to a given cluster of hosts (PMs). Thus, the proposal seeks to maximize resource utilization (return), while minimizing underload, overload and SLA violations (risk).

The VMPM problem has a well-known NP-hard complexity [1], considering the large scale of cloud infrastructures. Accordingly, the contributions of this paper are: (i) a novel way to use the MPT for VMPM that largely reduces computational complexity and memory footprint by applying the Welford algorithm to incrementally compute statistics; (ii) the definition of a multi-dimensional resource metric that considers variance as a resource unbalacing factor; (iii) use of the Gossip protocol to exchange data for VMPM; (iv) a distributed and decentralized proposal for VMPM decision-making that enables scalability, load balancing and fault-tolerance; and (v) our proposal was extensively tested using CloudSim Plus, a well-known and de facto simulator for Cloud Computing nowadays. To the best of our knowledge, this is the first proposal broadly applying the MPT that is computationally feasible for large-scale cloud providers.

The paper is organized as follows. Section 2 presents a parallel between financial markets and cloud computing resource allocation, their familiar points and their differences; Sect. 3 presents the foundations of portfolio analysis using modern portfolio theory (MPT) and showing how that theory can be applied for VM placement and migration; Sect. 4 reviews the related studies; Sect. 5 contains the MPT-based VM placement and migration proposal; Sect. 6 includes the proposal evaluation; and Sect. 7 finally reveals the conclusion and suggestions for future work.

2 Parallelism Between Financial Markets and Cloud Computing Resource Allocation

Considering the aspects surrounding MPT, a parallel between financial and cloud computing markets can be clearly drawn. First, players and other elements involved in both markets must be identified, as proposed in Table 1.

Financial assets, such as stocks, are bought by investors expecting some return. Virtual computing resources (such as VMs, containers or storage services) are rented by customers for some time to run services that benefit both the customers and their end-users. The generic term “asset” can be used to represent a financial or computing asset. An investor can buy and hold an asset for a certain time, usually while it is meeting the goals of that investor. In the same way, a cloud customer can rent a resource for some time, while it is useful or valuable for him/her. Investors and cloud customers are both customers in their markets.

Financial brokers trade assets belonging to someone else on behalf of customers. In cloud computing, providers mainly trade their own computing resources

Table 1 Parallelism between financial markets and cloud computing resource allocation

Financial markets	Cloud computing resource allocation
Financial assets or simply assets	Virtual resources (VMs, containers, etc)
Investors	Cloud customers
Financial brokers	Cloud providers
Investment contract	Service level agreement (SLA)
Demand and supply of assets	Demand and supply of computing resources
Taxes, brokerage and management costs	Co-located VM interference, VM placement and migration overhead, underlying infrastructure and platform overhead
Portfolio of assets	Portfolio of virtual resources
Portfolio balancing	Load balancing
Portfolio monetary return for investors	(i) Portfolio monetary return and costs reduction for providers and (ii) quality of service (QoS) for customers

with customers, although they can also trade resources owned by other providers in a hybrid or federated cloud. With Amazon Web Services (AWS) e.g., the Spot Instances service [12] trades computing resources according to demand and supply, and defines prices dynamically.

Investment contracts and service level agreements (SLAs) are both contracts defining rights and duties for the parties involved, namely: (i) brokers and investors in financial markets and (ii) cloud providers and customers in cloud markets. These contracts also define penalties for infringing parties. Demand/supply for financial assets is a main driver of asset prices in both markets. The demand for computing resources is determined by customers and the workload of their running services. This demand is highly dynamic and may be even harder to predict than in the financial market.

Buying and selling operations in the financial market usually have brokerage fees and taxes that investors must pay in addition to the asset price. Usually, the longer an investor holds an asset and waits to realize profits, the fewer fees and taxes he/she will be charged. Such charges impact investor net profit, so that the time in which these operations are performed has to be carefully decided. Similarly, VM placement and migration has to be performed in a timely manner. The interference and inherent overhead of such operations can be viewed as the fees and taxes to be paid. The collateral effect of these operations may reduce the performance of customer services and also impact other customers using the same physical infrastructure.

A cloud provider owns and may borrow computing resources from other providers. Cloud customers rent such resources from the provider on a pay-per-use basis. In the financial market on the other hand, investors buy assets from a broker, which are owned by a third-party such as a public company operating on the stock market. As has been shown, there are numerous similarities between cloud and financial markets that enable application of the Modern Portfolio Theory to the VMPM problem.

There are numerous similarities between financial and cloud computing markets, as shown in Table 1. The foundations of the MPT are based only on asset return and risk, whereas the latter is based on the variance of returns. This way, returns on assets play a major role in the theory, explaining why it has been applied in different fields. Certainly, there are differences between the financial market and other markets such as cloud computing, e.g., regulatory issues. For instance, cloud providers such as Amazon Web Services (AWS) in China have a dedicated infrastructure that does not communicate with the global AWS infrastructure [13]. Despite such regulatory concerns, we were not able to identify other aspects that interfere with the application of the MPT for VMPM problem.

3 Portfolio Analysis and VM Placement

3.1 Basic Concepts

A portfolio is a collection of assets, each one in a given proportion (also called allocation, share or weight). Correlation is one of the MPT foundations for computing portfolio risk, the latter represented as the volatility of the overall portfolio return. Correlation between two assets changes over time and must be estimated periodically to balance the portfolio and maintain risk at desired levels. A graph between risk and return can be plotted as a set of compounded portfolios of different assets, as presented in Fig. 1. A point in this graph represents a portfolio with a specific level of risk and return, according to weights assigned to each asset. The graph for a set of different portfolios is usually a bullet-shaped curve.

For a given portfolio, any other one outperforming it in terms of better return or lower risk is north-west headed. The green line starting at the upper half of the curve, is known as the efficient frontier. It is formed by optimal portfolios where one cannot achieve either (i) higher return without increasing risk or (ii) lower risk without decreasing return. The efficient frontier is drawn from all points along the risk (X) axis that have the maximum return for a specific level of risk.

A portfolio p is defined as efficient if and only if there exists no other portfolio q such that:

$$\mu_{t_q} >= \mu_{t_p} \wedge k_q < k_p \tag{1}$$

where μ_t is the mean return and k the risk for a given portfolio such as p or q [14].

The yellow horizontal line isolates the inefficient portfolios from the efficient ones, respectively at the lower and upper half of the curve. For a given level of risk there are at least two portfolios, as far as return is concerned: the least and the most profitable ones. The least profitable portfolio for a level of risk is a point at the edge of the lower half of the curve. On the other hand, for the same risk, one can realize the highest return by choosing the portfolio on the green line. As an example, in Fig. 1, for a risk of 10%, one can get a portfolio with the lowest return of about 1.7%. However, for the same risk there is a portfolio with the highest return of about 17% (10 times more).

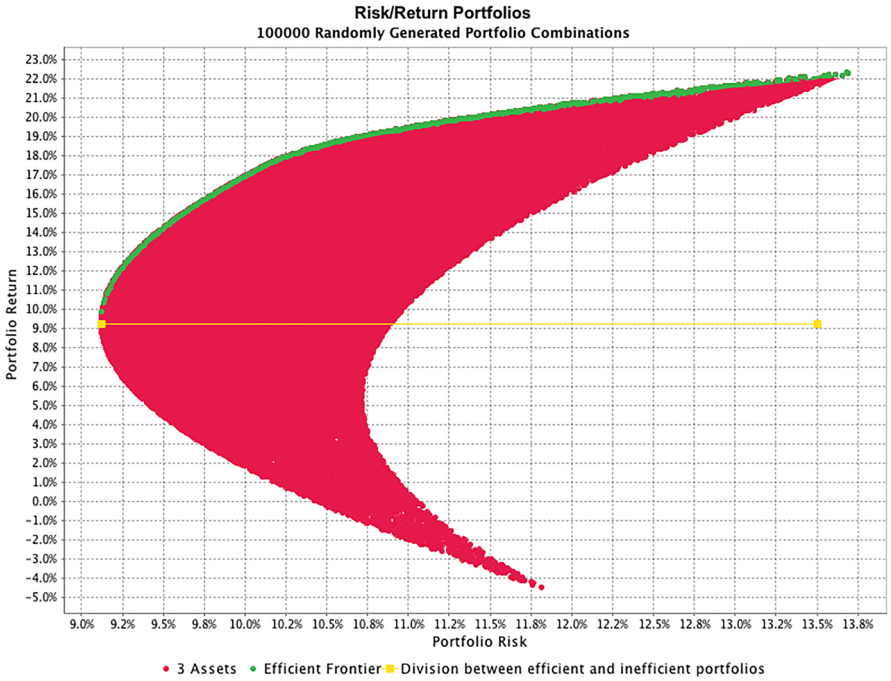


Fig. 1 Portfolio analysis: a set of randomly generated portfolios and the efficient frontier

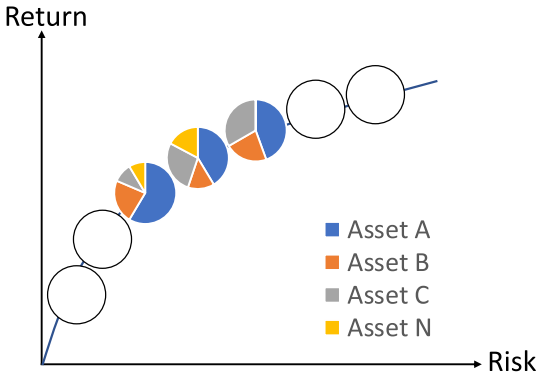


Fig. 2 Efficient frontier: each circle is the most profitable portfolio for a given level of risk (Adapted from [15])

In fact, the inefficient portfolios are not only contained inside the lower half of the curve. For a given level of risk, any portfolio not belonging to the efficient frontier has a lower return. The closer to that frontier, more efficient a portfolio is [11]. Portfolios along the efficient frontier can also be viewed as in Fig. 2. Each point represents an efficient portfolio compounded from different allocations of assets.

Consider that (i) only strategic asset allocation (which is based only on historical data) is being used to create portfolios and (ii) the goal is to minimize risk and maximize return. The only reasonable choice is a portfolio belonging to the efficient

frontier. Any other one should be chosen only if a tactical allocation is being performed, based on market trends or beliefs. The efficient frontier (*EF*) is therefore a set of portfolios that can be formally defined as in (2):

$$EF = \{\forall p \in P | k_p = min(k | \mu_t = \mu_{t_p}) \\ \wedge \ \mu_{t_p} = max(\mu_t | k = k_p)\} \tag{2}$$

Consider μ_t the mean return and k the risk of a portfolio p; P the set of all generated portfolios. The efficient frontier contains every portfolio p where its risk is the minimum for that level of return ($min(k|\mu_t = \mu_{t_p})$) and its return is the maximum for that level of risk ($max(\mu_t|k = k_p)$). It also only contains portfolios whose risk and return are both higher than those of the least risky portfolio. This is a curve starting at the minimum risk and ending at the maximum return [16], as shown in Figs. 1 and 2. The return μ_{t_p} of a portfolio p is the sum of a weighted average of its asset returns means [16], as defined in (3):

$$\mu_{t_p} = \sum_{a=1}^{N} t_a * w_a \tag{3}$$

where N is the number of assets, a represents a portfolio asset, t_a the return mean of that individual asset and w_a its weight. The return of a portfolio is straightforwardly computed using that equation. Conversely, computing the portfolio risk k_p is more complex. It is not based on the averaged variance of each asset, since the risk tends to decrease as more low-correlated assets are included.

The portfolio risk is represented by the variance of its assets returns, the sum of a so called variance-covariance $N \times N$ matrix [10], as (4):

$$k_p = Var(t_p) = \sum_{a=1}^{N} \sum_{b=1}^{N} w_a * w_b * \sigma_{ab} \tag{4}$$

where σ_{ab} is the covariance of returns between assets a and b. Considering that only a sample of the entire return history is being used, the sample covariance [10] is in turn computed as defined in (5):

$$Cov(a, b) = \sigma_{ab} = \frac{\sum_{i=1}^{n} (t_{a_i} - \mu_{t_a}) * (t_{b_i} - \mu_{t_b})}{n - 1} \tag{5}$$

where t_{a_i} is an individual sample of the asset return, μ_{t_a} is the asset mean return (also denoted as the expected return $E[t]$), and n is the number of samples along the time period.

3.2 Problem Formulation

The Markovitz optimization problem defined by the MPT [14] can be formulated as in (6):

$$\begin{aligned} &\max \sum_{a=1}^{n} w_a * \mu_{t_a} - k_a \\ &\text{s.t. } w_1 + ... + w_n = 1 \end{aligned} \tag{6}$$

aiming to maximize the sum of the weighted (w_a) mean return of each asset (μ_{t_a}), minus its risk (k_a), subject to the sum of weights adds up to 1. Such a restriction ensures that all available wealth for that portfolio is invested. That formulation aims to maximize returns while minimizing risk.

Portfolio analysis in MPT shows how to select assets weights so that the most profitable portfolio for a given level of risk can be found. On the other hand, each cloud customer explicitly sets the desired amount of resources, manually weighting them. Those amounts cannot be arbitrarily changed by the provider, at the risk of degrading customer services performance and violating SLA. Alternatively, the provider may dynamically allocate resources to customers, up to the contracted amount, so that SLA violations and resource under/over utilization is minimized. However, this scenario is not considered here.

A cloud portfolio could therefore be defined as a group of VMs to be placed inside a given cluster of PMs, in order to maximize return and minimize risk. This set of VMs is going to be called a cluster portfolio, where return is the mean percentage of resources utilisation and risk is its variation.

VMs are abstractions for the computing resources being used from a PM [17, 18]. The benefits of diversification, presented earlier, can be achieved in the cloud by selecting distinct VMs from different customers to compose a portfolio. Diversification relies on the actual imperfect correlation between two assets. Inverse (anti/negative) correlation is the most impacting factor for risk reduction. That is, as an asset price goes down, the other one goes up. This way, losses on one asset are compensated by returns on another one. There are several proposals regarding correlation when placing and migrating VMs. Some of them have shown correlation between different computing resources such as CPU and RAM [3, 4, 6, 8, 19–24].

Some authors state that when there is a higher correlation between computing resources usages, there is a higher probability of the hosting PM becoming overloaded [25, 26]. The reason is that as the usage of a resource rises, the directly correlated ones also tend to rise, increasing the overall resource usage.

Regarding the VMPM problem, the selected strategy has to ensure that resources required for all VMs be placed into a given PM/cluster do not exceed the PM/cluster capacity, as defined in (6). Exceeding it causes oversubscription, which usually should be avoided to meet SLA.

While in MPT the allocation of all financial assets must add up to 100%, commonly in cloud infrastructures that percentage should be lower than a given threshold for any type of resource. That ensures there will be available resources for PMs internal operations, possible VM vertical scaling, workload bursts and even for overhead expected during VM migrations.

Before formulating the VMPM problem, weights have to be addressed. These define the fraction of each asset selected for a portfolio and are used to adjust risk. Different portfolios usually share the same assets in equal or different proportions.

Reducing weight of riskier assets tends to reduce overall portfolio risk. Considering a VM as an indivisible cloud asset, the provider cannot assign just a fraction of it to a given cluster. This way, VM weights are always 1, which can be removed from the formulation, as presented in (7).

$$\begin{aligned} &\max \sum_{a=1}^{n} \mu_{t_a} - k_a \\ &\text{s.t. } \mu_{t_1} + ... + \mu_{t_n} <= threshold \end{aligned} \tag{7}$$

Considering μ_{t_a} as the overall return for VM a and k_a its risk, the sum of returns for all VMs inside a portfolio is not expected to exceed a defined threshold of the total PM/cluster capacity. VM return is the utilization mean of its resources, namely CPU, RAM and network traffic for this proposal, as defined in (8):

$$\mu_{t_a} = average(\mu_{cpu}, \mu_{ram}, \mu_{inv\_net}) \tag{8}$$

where μ variables are the VM mean utilisation for CPU, RAM and network traffic (normalized between 0 and 1). For network traffic, the utilization mean is inverted since the lower the traffic the better. That is a specific metric which will be detailed in Equation (9) of Sect. 5. There are other variables that could be included in the equation, such as storage, I/O throughput, and network QoS metrics. Storage space is commonly not an issue for current cloud providers. This is an abundant and cheap resource [27], mainly after the availability of Storage Area Networks (SANs). Therefore, the most critical resources in a cloud data center were considered, while including more variables than some related works presented in Sect. 4.

Although the resource utilization average is a unidimensional metric for a multidimensional problem (as surveyed in [1]), the formulation in (7) subtracts the risk from the maximization function. Considering the MPT foundation, risk is a volatility metric based on the variance of returns. If a VM resources variance is high, that means the utilization of individual resources is unbalanced. This way it reduces the VM utility by considering resource utilization volatility, and consequently reduces unbalancing.

Although weights are removed from the formulation, they are implicitly represented as the number of VMs assigned to each cluster or host. However, to reduce the number of decisions to be made and the computational complexity of the proposal, those weights are not input variables, but final results from a given VM placement. Those weights could be adjusted after the proposed placement, but this kind of tuning is not addressed here.

4 Related Work

There is a plethora of proposals for VM placement and migration available in the literature, as surveyed in [1]. The majority use traditional methods to compute statistics, based on resource utilization history collected over long time periods. That leads to overhead on CPU and memory in order to make VMPM decisions, due to

the volume of collected data. Other proposals apply heuristics such as Ant Colony Systems or Simulated Annealing, which are known to lack scalability for large scale infrastructures. This section presents some of those proposals that apply correlation between VMs and/or the modern portfolio theory.

4.1 Dynamic Correlative VM Placement

An MPT-based VM placement and migration proposal is presented in [20]. It considers correlation between multiple VMs when generating a placement scheme for a given PM, so that time-varying (anti-correlated) resource demands are multiplexed. This means that VMs with different resource peak times are placed together to maximize resource usage.

The proposed scheme is based on VM demand prediction and volatility, using the auto regressive integrated moving average (ARIMA) model to predict k-step-ahead utilization for a VM resource. The generalized auto regressive conditional heteroscedasticity (GARCH) model is used to predict resource usage standard deviation, so as to reduce that the risk of prediction-based VM placement.

Although the proposal uses correlation among a group of VMs, it is not clear how MPT is applied as a whole. The only MPT foundations the proposal relies on are risk and VMs correlation. MPT relies on the creation of multiple portfolios with different allocations for the available assets. This means that they can be plotted together to select the best one that minimizes risk and maximizes return.

The work presents a first-fit greedy algorithm to place the first VM into the first PM, so that the expected PM load is not higher than its capacity. Finally, experiment results are assessed based only on computational simulation using synthetic data. Datacenter traces or testbeds are not applied to validate such results, which are presented without any simulation parameters and confidence intervals.

4.2 MPT-Based Static Single Resource Management

Portfolio theory-based algorithms are proposed in [28] for QoS-aware resource assignment in cloud environments, in order to maximize resource usage and minimize energy consumption. The proposal considers resource requirement as a random value x, instead of using historical data. However, since the cumulative distribution function (CDF) of x is usually not known, the amount of required resource is estimated using the Cantelli inequality [28]. The paper shows that estimation tends to allocate more resources than required. If the distribution of the random variable is known, the estimation can be more precise.

The proposal simply applies a subset of the MPT to reduce the standard deviation (σ) of VM resource requirements, mapping a static set of VMs to PMs. The work generates and selects the VM portfolio for a given PM that minimizes σ. However, only CPU usage is considered as resource. A hierarchical resource management solution is presented in which (i) a cloud-level manager maps VMs to clusters of PMs and (ii) cluster-level managers map each VM to a specific PM. Cluster managers work independently in parallel.

Although a distributed solution is proposed, it uses a central cloud manager accountable to allocate VMs across the clusters, which is not suitable for current large-scale cloud infrastructures. That manager may not be able to generate a high number of different portfolios for selecting optimized sets of VMs to distribute among clusters. Clusters may not receive a set of VMs that enables them to obtain suboptimal-enough portfolios. This issue is even stated in a further publication from the same authors [29]. Collaboration between clusters would give better results, such as presented in [30, 31].

The proposal is assessed only by simulation and only using synthetic data. It outperforms the well-known simulated annealing (SA), first fit decreasing (FFD) and best fit decreasing (BFD) heuristics.

4.3 MPT-Based Dynamic Multiple Resource Management

The aforementioned work is extended in [29] by enabling resource allocation for dynamic-arrived VMs and multiple resources. The previous work considered only CPU requirements of VMs as random values to be estimated. Nevertheless, this extended proposal only considers VM workload intensity, a more general metric, as random values to be estimated. Workload intensity X for a VM n is defined as the average number of jobs running inside that VM. The requirement R for a resource k of a VM n is estimated using a linear regression in terms of such a workload intensity [29]. The authors state that the model is reasonable since there is a correlation between workload intensity and resource requirements.

The work presents a first fit decreasing (FFD) algorithm for mapping VMs to a cluster of PMs. It computes the cost of a VM n as a risk factor σ^2/μ, where σ^2 is the variance of resource requirements and μ is the mean.[1] It then assigns VMs with the highest cost to a cluster of PMs owning the highest capacity.

Since the hierarchical solution may not produce such optimized results, as discussed in the previous section, the authors propose a joint FFD algorithm which iterates over PMs from all clusters and selects VMs according to their workload intensity.

The major issue with this joint solution is the size of the problem for a large amount of PMs across all clusters. This approach may provide better placements, compared to the hierarchical proposal, since it has complete information about all VMs and PMs. However, trying to solve such a complex problem using all that information in a centralized way is impractical for large-scale data centers. That is one of the reasons why the papers presented in the literature attempt suboptimal solutions based on partial information view. Although the authors state that the hierarchical algorithm is highly scalable, the global manager is a single point of failure and is likely to be a bottleneck.

A local manager also performs VM migration when over or underutilized PMs are detected. It allows VMs from overloaded PMs to be migrated intra-cluster,

[1] That is merely a simplification of the actual equation.

when there is any PM able to offload the overloaded one. If no PM in the cluster has enough capacity to host VMs from an overloaded PM, inter-cluster migration is requested.

5 MPT-Based VM Placement Proposal

5.1 Proposal Overview

Correlation among assets plays a leading role in MPT, so that the MPT-based VM placement proposal (henceforth, MPT VMPM) considers it for placing VMs into clusters of PMs. Portfolio creation depends on the return of each asset. It can be computed based on different metrics, such as: (i) VM utilization level, i.e. the resource demand (as closer to an upper utilization threshold the better); (ii) the network traffic and distance (the lower the better); (iii) task completion time (the lower the better); (iv) other metrics or a composition of them.

The work of Wei et al. [20] presented in Sect. 4.1 considers the resource demand of a VM as the return and its volatility as the risk. The present work proposes an average resource utilization metric to define the return of a VM, as already presented in (7) and (8).

The foundation of relying on the VM resource usage to compute the return is the principle that the most resources used the better. Since underloaded VMs may lead to underloaded PMs, this causes wastage of resources. Furthermore, several works have shown that an idle PM is accountable for up to 70% of the energy consumption compared to the PM in full utilization [1]. Although energy efficiency is improving with new technologies, idle allocated resources are a waste, since they could be used by other customers.

According to the VMPM problem formulation in (7), the proposal aims at maximizing returns. However, for resources such as network traffic, minimizing their use provides more benefits, meaning maximization of returns. Mimimized traffic reduces: (i) task wait time, which leads to reduced task completion time; (ii) communication delay and network congestion. In order to achieve the mentioned goal, a weighted network traffic metric is defined in (9):

$$\mu_{inv\_net} = (vm_{bw} - bw)/hops \tag{9}$$

where vm_{bw} is the VM bandwidth capacity, bw is the VM bandwidth consumption and $hops$ is the number of network hops between inter-communicating VMs. That metric is used as a way to invert bandwidth consumption, while ensuring the resulting value lying between [0..1]. In order to minimize traffic, inter-communicating VMs should be placed as close as possible by applying that metric. The application of bandwidth and hops in network metrics is very common and the literature shows their utilization in different ways [32, 33].

5.2 Portfolios Generation

The reasoning behind MPT relies on the generation of multiple portfolios with random weights for each asset, as depicted in Fig. 1. That sample chart was drawn from 100,000 random portfolios. From these portfolios, it is supposed to select one, which will represent the actual allocation of a subset of VMs (assets) into some datacenter or cluster. Here, only clusters are taken into account. However, consider a number of 1000 candidate clusters where VMs from a selected portfolio may be placed into. That would require a total of 100 million portfolios generated for all clusters, just to select one for each cluster. That imposes a huge memory and processing overhead, as will be presented in Sect. 5.4. Furthermore, considering that allocation of VMs into data centers/clusters is re-evaluated periodically, that overhead may impact the regular cluster operation.

Alternatively, in order to provide a low-overhead solution that is suitable for large-scale cloud providers, the current proposal does not generate that huge an amount of portfolios for a single cluster. When a new VM arrives and does not yet have a utilization history, it is placed inside some cluster, applying a well-known algorithm such as First-Fit (FF). While utilization metrics are computed over time, some clusters are randomly and periodically selected to have their portfolio re-evaluated. That is performed by recomputing portfolio risk and return along with the arrival of VMs. Arrivals occur when new VMs are submitted to the cloud or finished ones are re-started.

Since cluster controllers can receive requests to place arrived VMs and there may be multiple controllers for each cluster, numerous potential portfolios can be computed so as to evaluate the placement of those VMs. Arrived VMs are placed into the cluster with higher return and/or lower risk. Accordingly, the current proposal presents a very particular utility function as defined in Algorithm 1. It computes the portfolio utility (which has to be maximized) for placing a set of VMs into a target cluster. The function compares the utility of a cloned portfolio containing the new VMs and the original cluster portfolio. The function is detailed as follows:

- Line 2—Checks if the risk of the other potential portfolio has been decreased at least by the defined threshold (*MIN_RISK_DEC*).
- Line 3—Checks if the return has been increased at least by the defined threshold (*MIN_RETURN_INC*).
- Line 4 to 7—If there is a minimum risk decrease or return increase compared to the original cluster portfolio, it computes the new portfolio utility to enable ranking that portfolio among other possible placements into different clusters; otherwise, the utility is considered zero, so that such a target cluster is ignored for that VM arrival.

Table 2 Comparing accuracy of results for computing statistics using the Welford algorithm (1st line) and traditional method using the entire samples history (2nd line), for 100000 samples uniformly distributed between 0 and 10

Average	Std. Deviation	Variance
5.02078838047343	2.88597411191550	8.32884657464645
5.02078838047353	2.88597411191553	8.32884657464665

5.3 Mean and Variance for MPT

The MPT formulation relies on the covariance to compute risk (5), which requires a history of sample data. Considering that these samples represent VM resource utilization, as higher the frequency and longer the collection period, the VM return (mean utilisation) and risk (volatility) are more accurate. However, for large-scale cloud data centers with thousands machines and hundreds of thousands VMs, keeping such a utilization history imposes a heavy memory footprint and largely increases computational complexity in the long run, as the sample number grows. Proposals such as [28, 29], which rely on the regular MPT formulation, have this memory and CPU overhead, which may not be feasible for actual large-scale cloud providers. The overhead of a proposed solution should not compromise SLA for current customers.

Alternatively, the Welford online algorithm is applied [34] to incrementally compute portfolio return and risk. This is a non-interactive approach which updates the mean and variance for each new sample, without storing each sample. The algorithm was chosen because it is well-known and produced precise results in our tests, compared to the regular computation of mean and variance. It is defined in (10):

$$\begin{aligned} \mu_i &= \mu_{i-1} + \frac{\delta_{i-1}}{N} \\ \delta_i &= x_i - \mu_i \\ \sigma_i^2 &= \sigma_{i-1}^2 + (\delta_i * \delta_{i-1}) \end{aligned} \tag{10}$$

where N is the number of samples, x_i and δ_i are a sample and its deviation from the current mean μ for time i; σ_i^2 is the variance for time i.

In order to ensure the accuracy of Welford's algorithm compared to the traditional way of computing statistics, we have created a simple experiment using a pseudo-random number generator (PRNG) to generate 100000 samples and compute the mean, variance and standard deviation for both methods. Samples were uniformly distributed between 0 and 10. The experiment was executed multiple times using different seeds, showing that the Welford algorithm has a high precision up to the 12th decimal place for these 3 metrics. Results are presented in Table 2 using the seed provided in Table 3.

Table 3 General simulation parameters

Parameter	Description	Value(s)
Base seed	Seed for the 1st experiment run (incremented by 1 for each run)	1611254387276*
Simulation runs	Number of times the experiment is executed	30†
$\#DCs$	Number of data centers	1†
$\#Hosts$	Number of hosts by datacenter, defining DC size and capacity	1000†
Cluster size	Number of hosts inside each cluster	200†
τ_d	CPU lower utilisation threshold	30%‡
τ_u	CPU upper utilisation threshold	85%‡
$\#CPU_h$	Possible number of CPUs (cores) for available hosts. For each created host, the number of CPUs is selected from this set in a round-robin way	[8, 12]*
$MIPS_h$	Host processing capacity in million instructions per second	10000*
RAM_h	RAM capacity for each host	[64, 128] GB*
STO_h	Storage capacity for each host, linear to the number of CPUs	$\#CPU_h * 12.5$ TB*
BW_h	Bandwidth capacity for each host	10 Gbps‡
Host idle shutdown deadline	Time to wait before shutting down and idle host (according to τ_l)	60 min†
#VMs	Number of statically created VMs at the start of the simulation (40% of the number of hosts)	400†
Dynamic VM arrival time interval	The time interval in which VMs will dynamically arrive during simulation runtime, following a poisson distribution	1.5 min*
VM requirements	Defined based on AWS EC2 instances. Each VM is created from a EC2 template randomly chosen in a uniform way	Check Table 4
CPU_m	CPU utilization model: defines how applications running inside VMs use the CPU	PlanetLab trace files, Gaussian and Uniformly Distributed PRNGs‡§
RAM_m and BW_m	RAM and BW utilization model: define how applications running inside VMs use the RAM and BW	Uniformly Distributed PRNG§
i_c	Time interval in which resource utilisation data is collected for further decision making	10 min*

Values choosen: * arbitrarily, † to limit simulation scale and reduce execution time, ‡ based on common values for such parameters, § to vary VM resource utilization and better placement opportunities

5.4 Computational Complexity & Memory Footprint Analysis

The online algorithm (10) computes mean and variance in a single pass, updating those statistics as new samples arrive. That provides a constant computational complexity $O(1)$ instead of linear $O(n)$ for the worst case, where n is the number of samples. Similar approaches for rapid computation of such statistics were previously presented by [34].

Consider that (i) the cloud provider has 10 k hosts and 100 k VMs; (ii) CPU, RAM and network utilization metrics are collected every 30 min throughout 1 year; and (iii) a metric is a double value requiring 8 bytes of memory. The total number of samples collected for a single metric by a proposal that stores those values, for just one VM, will be 17520, defined as:

$$n = 2 \text{ samples/hour} * \ 24 \text{ hours} * 365 \text{ days}$$

The total memory footprint of a proposal that stores those samples, taking all metrics and VMs, will be:

$$n * 8 \text{ bytes} * 3 \text{ metrics} * 100k \text{ VMs}$$

That is almost 40 GB of historical data per year. The memory footprint is initially negligible, considering the entire cloud provider infrastructure, but it may become an issue in the long run if the entire sample history is stored. Despite that footprint is irrelevant for an entire cloud infrastructure, the computational complexity $O(n)$ to compute metrics for a high volume of data is not. It may pose a huge overhead as the sample history grows. Furthermore, that complexity is infeasible for simulation environments, restricting the scale of experiments.

Applying the Welford algorithm for this same scenario and considering the 6 variables in (10), the memory footprint is reduced to: 6 variables $*$ 8 bytes $*$ 3 metrics $*$ 100k VMs. The result is a constant value around 14 MB of memory for the given number of VMs, or a negligible 144 bytes per VM.

The followed approach to computing statistics improves the scalability of the proposal for both large-scale simulations and actual cloud infrastructures. Those improvements reduced the time for running such simulations, with the parameters presented in Table 3, from hours to minutes (using a 2,8 GHz dual-core hyper-threading Intel i7-4558U PC with 8 GB 1600 MHz DDR3 RAM).

5.5 Proposed Architecture

MPT VMPM is a distributed proposal with a decentralized management, as presented in Fig. 3. Hosts inside cloud data centers are grouped into clusters. Each cluster has at least one controller node that assesses the placement of VMs inside that cluster. A controller can be either a specific PM used exclusively for this task or a regular PM hosting customers VMs. A cluster controller can be selected either (i) manually (at least for the first time), according to its processing capacity; or (ii) randomly (the approach followed for simplicity), mainly in case of a controller failure.

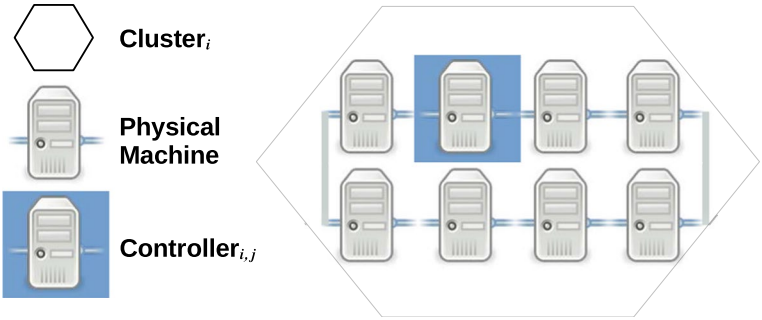
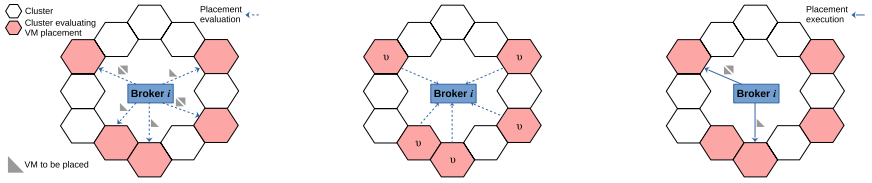


Fig. 3 Overview of the proposal architecture: physical machines inside a cluster with at least 1 controller. Datacenter machines are grouped into different clusters, where each one can have multiple controllers



(a) Stage 1: A broker requesting VM placement evaluation to cluster controllers.

(b) Stage 2: Controllers compute the cluster utility Function 1 (υ), for placing requested VMs and reply to the broker.

(c) Stage 3: Broker ranks placement possibilities according to μ_t/r ratio and decides the actual placement.

Fig. 4 MPT VMPM architecture using the gossip protocol: stages of the VM placement assessment across multiple clusters in a given datacenter

PMs are grouped into clusters of equal size, as presented in Table 3. Clusters are created by splitting the set of all existing PMs into subsets. Network topology usually determines how PMs are grouped into clusters, but the proposal is agnostic on that point. Different sizes and heterogeneity of clusters will certainly impact performance and results. However, due to the current complexity, scale of the simulations, and limited infrastructure, it was not possible to include these variables in the experiments, which are presented as future work.

The proposal considers datacenter brokers available in current cloud provider infrastructure in order to randomly select cluster controllers. Brokers are accountable for responding to customer requests to manage VMs (create, clone, destroy, etc) and dispatch those requests to controllers for evaluation of VM placement/migration. This way, an existing component in the cloud infrastructure can be extended to provide new capabilities, instead of adding a new one. Considering the scale of global cloud infrastructures, such brokers are already distributed so that they can handle a large number of concurrent requests from multiple customers.

Since the number of Hosts and VMs inside a cluster is usually large and imposes a decision making overhead which may affect customer SLAs, multiple nodes inside a cluster can be chosen as controllers. Besides distributing that overhead across multiple clusters, this approach enables load balancing inside the cluster, while providing fault-tolerance for the controller. In such a multi-controller by cluster scenario, there is no leader defined and each controller has the same responsibilities. That is

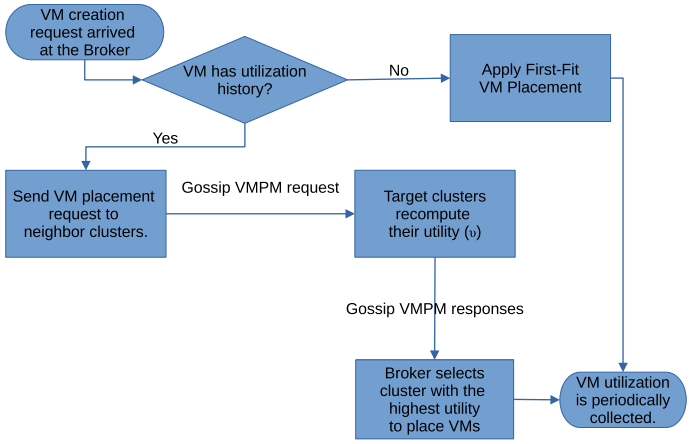


Fig. 5 MPT VM placement general algorithm (MPT VMPM)

an important feature for distributed systems, since in case of a controller failure, any available node can replace it.

Figure 4 gives a detailed view of the architecture. The set of VMs placed into Hosts of some cluster represents the cluster portfolio, having a given risk (k) and return (μ_t). When the broker receives a request to place VMs that have a utilization history, it sends such a history and VMs resource requirements to randomly selected clusters. Target clusters recompute their risk and return to place arriving VMs and reply to the broker informing the new risk and return values, according to the utility Function 1.

The aforementioned function aims to increase the return/risk ratio, which is a simplified version of the Sharpe Ratio [10], adapted for the VMPM problem. The simplification was required since the original equation includes some financial market metrics which do not have a correspondence in the cloud computing market. The *returnRiskRatio* variable defines the amount of return by each unit of risk. This is one of the variables to be maximized. The higher the return for lower risk, the better. In such a scenario, adding up that ratio provide more utility for the proposed placement.

After receiving the replies from target clusters, the broker ranks them by the return of the utility function in a decreasing order. The cluster providing the highest utility is selected to place arrived VMs. In case an arriving VM has no utilization history, a First-Fit algorithm is applied to find the first suitable Host for that VM.

In order to enable such requests between brokers and cluster controllers, a distributed communication protocol must be introduced. The proposal uses the Gossip epidemic protocol to perform a distributed search, sending VM placement requests across neighboring clusters. The MPT VMPM algorithm that performs such a flow is presented in Fig. 5. Considering the high-scale of cloud infrastructures and low response times required by VMPM proposals in order to promptly select target PMs for VMs, the Gossip protocol was chosen due to its: (i) high scalability, (ii) decentralized management, (iii) peer-to-peer propagation of data across a network

of nodes and (iv) fault-tolerance [30, 31, 35]. Those are crucial characteristics for distributed systems such as the proposed one.

Consider that we have multiple brokers and controllers communicating using the Gossip protocol. Each broker that receives a request to place VMs can randomly select one of the available controllers in a target cluster to send a request for placement evaluation. This way, multiple controllers in the same cluster will receive a different request, balancing the load. However, we have not simulated multiple brokers in the experiments.

Although the presented architecture relies on network communication, Figs. 3 and 4 are not intended to represent any network topology, since the proposal is agnostic in that regard.

6 Proposal Evaluation Using CloudSim Plus

6.1 CloudSim Plus Overview

The MPT VMPM proposal is evaluated by extensive and large-scale simulations using CloudSim Plus, our open source cloud computing simulation framework [36], available at https://cloudsimplus.org. CloudSim Plus currently represents the state-of-the-art in general cloud computing simulation. Several other simulation frameworks have been built using it as the underlying simulation engine, including RECAP-DES [37], LEAF [38], PureEdgeSim [39], SatEdgeSim [40] and others.

CloudSim Plus is a completely re-engineered project that largely improves simulation scalability and fixes critical bugs that jeopardize simulation accuracy, correctness and result reliability. Furthermore, the framework has been actively maintained since 2016 and this proposal is based on its version 7.2.0. Some of the exclusive features of the framework include:

- Accurate power consumption models; joint power- and network-aware simulations; parallel execution of simulations;
- Host fault injection; virtual memory and reduced bandwidth allocation due to oversubscription; closest datacenter VM placement; grouped VM placement;
- Horizontal and vertical VM scaling; etc.

6.2 Methodology

This research work follows a Design Science Research (DSR) approach [41, 42]. Starting from the VMPM problem, different proposals were studied, as presented in [1]. The Modern Portfolio Theory (MPT) was selected and adapted, considering: (i) that we could not find any work comprehensively applying the theory for the VMPM problem; (ii) the suitability of the MPT for that problem, as it has been shown throughout this work and (iii) the possibility of applying an incremental approach for computing the fundamental statistics that MPT relies on (mean, variance and covariance) to reduce computational complexity and memory footprint, which has

Table 4 AWS EC2 instances configuration used for creating VMs: prices and configuration for Linux instances in the US Eastern (Ohio) time zone

Name	CPU cores	RAM (GBs)	Name	CPU cores	RAM (GBs)
a1.md	1	2.0	t2.s	1	2.0
a1.l	2	4.0	t2.md	2	4.0
a1.xl	4	8.0	t2.l	2	8.0
t3.n	2	0.5	t2.xl	4	16.0
t3.mc	2	1.0	m5.l	4	8.0
t3.s	2	2.0	m5a.l	2	8.0
t3.md	2	4.0	m5a.xl	4	16.0
t3.l	2	8.0	m5ad.l	2	8.0
t3.xl	4	16.0	m5ad.xl	4	16.0
t3a.n	2	0.5	m5d.l	2	8.0
t3a.μ	2	1.0	m5d.xl	4	16.0
t3a.s	2	2.0	m5dn.l	2	8.0
t3a.md	2	4.0	m5dn.xl	4	16.0
t3a.l	2	8.0	m5n.l	2	8.0
t3a.xl	4	16.0	m5n.xl	4	16.0
t2.n	1	0.5	m4.l	2	8.0
t2.μ	1	1.0	m4.xl	4	16.0

Total of 34 VM templates. Some of them have the same base configuration, but different processing capacity, which is not being considered here.

Source: http://aws.amazon.com/ec2/pricing/on-demand (n = nano, μ = micro, s = small, md = medium, l = large)

Table 5 Gossip/MPT parameters to enable clusters to trade VMs using the MPT VMPM proposal

Param	Description	Value (%)
Max target clusters	Max % of random source clusters that can be requested to evaluate VM placements	60†
τ_{kd}	Minimum acceptable risk (k) decrease threshold	2‡
τ_{tu}	Minimum acceptable return (t) increase threshold	2‡

Values chosen to: † (i) reduce the number of potential target clusters and network traffic and (ii) limit simulation scale and reduce execution time; ‡ provide a minimum placement improvement to be worth the VM migration overhead

not been applied yet. The proposal was designed as presented in Sect. 5.5 and an experimental approach was followed by creating simulation scenarios using the CloudSim Plus framework, which improves simulation correctness. Finally, experiment results were collected following a scientific approach, to compute confidence intervals (CI's) for data from multiple simulation runs.

The simulation experiments to be presented here were created from both synthetic data (using pseudo-random number generators, PRNGs) and trace files of

VMs running into PlanetLab [43] (a former global research network) and Google Cluster. These three different sources were used together in all experiments, providing heterogeneous workloads as happens with actual cloud providers. The simulation is designed as described below, while its parameters are presented in Tables 3, 4, 5.

The experiments run in a simulated cloud infrastructure with one datacenter (the number of data centers is restricted in order to reduce simulation time). A fixed number of statically created VMs are submitted to the cloud infrastructure when the simulation starts, defining an initial workload. During simulation execution, new VMs arrive following a Gaussian distribution with a given mean. Those VMs represent the dynamic requests and workload the cloud provider receives all the time. Created VMs have some defined capacity, according to actual VM instances from AWS Elastic Cloud Computing (EC2) service, as presented in Table 4. Such instance configurations are based on AWS since it is a worldwide cloud provider and other providers follow similar configurations when setting the capacity for their VM instances.

VMs are managed as black boxes, where the cloud provider does not know the applications running inside it. Next, applications (cloudlets [36]) are created merely to simulate some kind of workload inside each VM. The VM workload is based on the CPU, RAM and bandwidth (BW) utilization of those applications. Each application applies some model to define how those resources will be used along the time. The RAM and BW utilization is based on a uniform PRNG. The CPU utilization is based on Gaussian and uniform PRNGs and also PlanetLab trace files. This way, different kinds of workloads are simulated, as expected in public cloud providers.

The evaluation of the MPT VM placement proposal is conducted by executing multiple runs of different simulation experiments to assess and compare results, as presented below:

(E1) Random VM placement: places VMs into suitable hosts randomly selected following a uniform distribution, using a PRNG. That is a naive approach simply to show how the FF and MPT proposals outperform it.
(E2) MPT Random VM placement: it works the same way as the experiment E1 above, using the initial random VM placement. Then it applies the MPT proposal to improve the initial placement.
(E3) First-Fit (FF) VM placement: a low-complexity round-robin algorithm that tries to place each VM on the first suitable host found inside any available datacenter cluster. That host is used to place any consecutive VM. When such a host is not suitable for any VM anymore, the next one is selected for current and following VMs.
(E4) MPT FF VM placement: the proposed trading MPT algorithm that periodically negotiates random exchanges of VMs between neighboring clusters, recomputing the cluster risk and return. If such metrics are improved, according to the utility Function 1, VMs are actually migrated between clusters. The proposal uses the previous FF algorithm for initial VM placement.

6.3 Result Analysis

6.3.1 Metrics and Experiment Types

Considering the experiments just enumerated, this section presents simulation results for the following evaluated metrics:

(M1) Active Hosts Avg (%): average percentage of active hosts during simulation execution, considering the total number of hosts on the datacenter across all clusters. The lower this metric is, the better, since the VM placement algorithm is consolidating the maximum number of VMs into the minimum number of hosts. However, that can lead to overload.
(M2) Active Hosts Max (%): maximum percentage of active hosts during simulation execution, following the same reasoning of the previous metric. However, an increased value may indicate that the server consolidation process is not sufficiently effective, which will require the activation of a higher number of hosts at some point in time.
(M3) Used CPU cores Avg (%): average percentage of used CPU cores in active hosts. This is used as a way to indicate how loaded a server is, according to the number of cores used. If the server is active, the highest number of cores used that do not overload the hosts is desirable. An increased value improves the cost/benefit relation of the host, since it reduces resource waste and balances power consumption [1].
(M4) Used CPU cores Max (%): maximum percentage of used CPU cores for active hosts, following the same reasoning of M3.
(M5) CPU overload mean (%): average percentage of CPU load for times hosts are overused, considering a defined threshold. This indicates how overloaded hosts are on average. Overload is well known to cause issues for companies, cloud provider infrastructure and customers, as broadly discussed in the literature surveyed in [1].
(M6) CPU overload time (%) from uptime: percentage of the time, on average, hosts are overused (considering their uptime) above the threshold discussed in the previous metric. The longer this overload time is, the higher the negative impacts will be. An increased value may indicate that the VM placement algorithm is not sufficiently effective in distributing VMs, since overload is the major issue to address.
(M7) CPU underload mean (%): average percentage of CPU load for times hosts are underused, considering a defined threshold. In contrast to metrics such as M5 and M6, this one indicates resources are being wasted, while fewer hosts could be activated to meet demand and reduce power consumption.
(M8) CPU underload time (%) from uptime: percentage of the time, on average, hosts are underused (considering their uptime) below the threshold discussed in the previous metric. An increased value may indicate the VM placement algorithm is scattering VMs in more hosts than required, which may increase power consumption and resource waste.

(M9) Hosts CPU mean load for non under/overload times (%): percentage of Host CPU capacity is being used during regular times. The closer the value is to the CPU upper utilization threshold the better, since it maximizes resource utilization while trying to avoid under and overload.
(M10) Hosts uptime (%) mean: percentage of time (according to the total simulation time) hosts are active on average. The higher this value for non under or overloaded hosts the better, following the same reasoning for the previous metric.
(M11) Total overload hosts (%): percentage of overused hosts along the total simulation time.
(M12) Total underload hosts (%): percentage of underused hosts along the total simulation time.
(M13) Hosts startup/shutdown power consumption (MW): the total power consumed by all activated hosts during startups and shutdowns.
(M14) Hosts total power consumption (GW): the total power consumed by all activated hosts during entire simulation execution.
(M15) Hosts startups mean: mean number of times hosts are powered on (since an idle host may be shutdown). The lower this value is, the less frequently hosts are switched on and off, which may reduce the time VMs have to wait for a host to be activated.
(M16) Delayed VM mean wait time (seconds): mean time VMs have to wait for a host being powered on (due to host boot time), excluding VMs that were promptly assigned to an already active host.
(M17) Delayed VMs (%): percentage of VMs that had to wait for a host activation before the placement could be performed.

Experiments are classified in two categories:

- Random: E1—places all VMs randomly; E2—applies the MPT placement for VMs having some utilisation history and the random placement otherwise.
- First-Fit (FF): E3—places all VMs using a FF algorithm; E4—applies the MPT placement for VMs having some utilisation history and the FF placement otherwise.

Results for different VM placement strategies, considering the previous metrics, are presented and discussed in the next subsection. In order to understand the results, a brief overview of how each experiment works is presented as follows.

6.3.1.1 E1 vs E2: Random vs Random + MPT VMPM E1 does not apply any analysis to place VMs and it is expected to degrade some important metrics. E2 improves most of them by applying the MPT algorithm after the initial random placement.

6.3.1.2 E3 vs E4: FF vs FF + MPT VMPM E3 uses a FF algorithm to place VMs into the first suitable host available, prioritizing those that are already active. The algorithm performs server consolidation by packing the maximum number of VMs into the same host. Despite having a low computational complexity, it does not compare VM requirements or assess any other metrics other than available resource capacity.

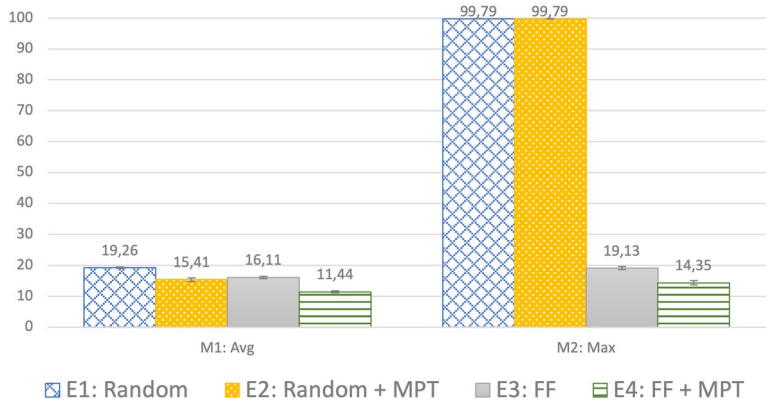


Fig. 6 Active hosts (%)

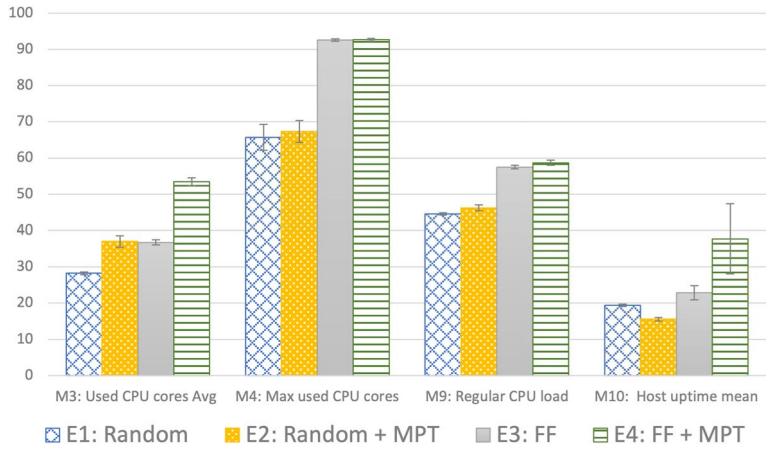


Fig. 7 General CPU usage (%)

That may leave unallocated residual capacity for some resources due to a mismatch in resource allocation for placed VMs. That means some kind of resources may run out, while other ones are plentiful.

The MPT proposal in E4 performs a more complex analysis of VMs resource demand to create a portfolio for a given cluster, trying to balance risk and return. That improves resource utilisation and other metrics discussed as follows.

6.3.2 Evaluating Experiment Results

Figures 6, 7, 8, 9, 10, 11 present charts grouping some related metrics for all experiments, including the confidence interval (CI) error margin. This way, it is easier to compare them and analyze how they change from one experiment to another.

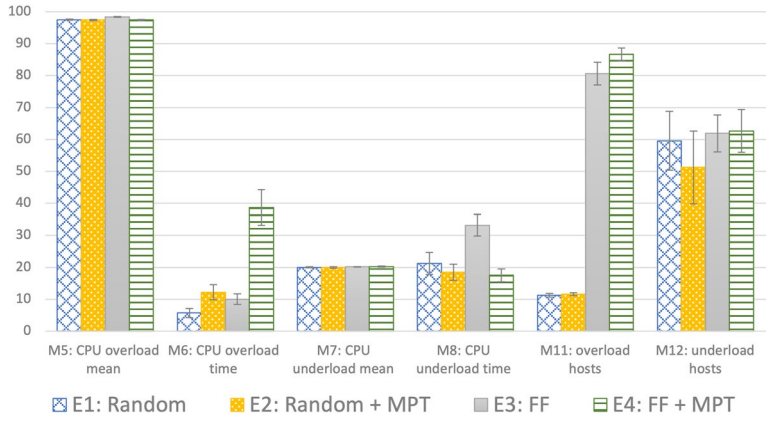


Fig. 8 CPU usage for non-regular times (%)

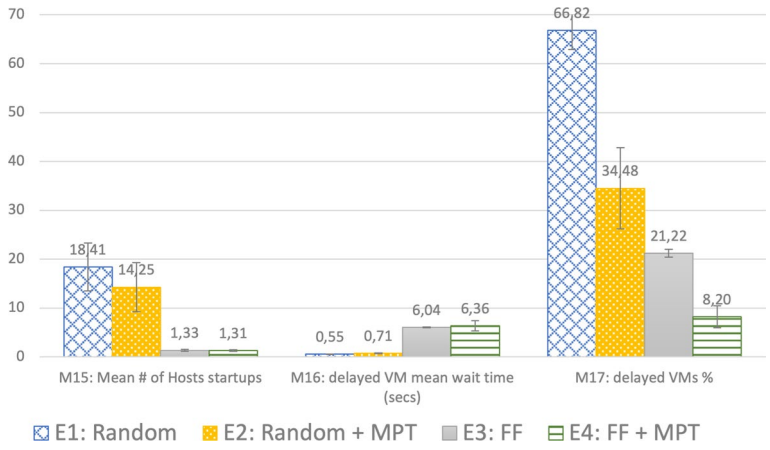


Fig. 9 Number of times hosts are started up and VM wait time

Figure 6 shows the average (M1) and max (M2) percentage of hosts used throughout the entire simulation runs. The reduction is gradual and MPT VMPM experiments (E2 and E4) achieve the best results, when one compares each one to the corresponding experiment without MPT. The differences in the average number of active hosts are not so large because idle hosts in any proposal are turned off after a specific time limit, in order to reduce power consumption. There is a significant difference in the max % of hosts compared to experiments using initial random placement (E1 and E2) and initial FF placement (E3 and E4). That clearly happens because random placement will spread VMs across a larger number of hosts, starting up many more hosts that become idle over time. MPT VMPM (E4) outperforms all experiments for both metrics presented.

Figure 7 shows hosts % uptime mean (according to total simulation time) and CPU utilization metrics. Those are metrics to be maximized. As long as resource

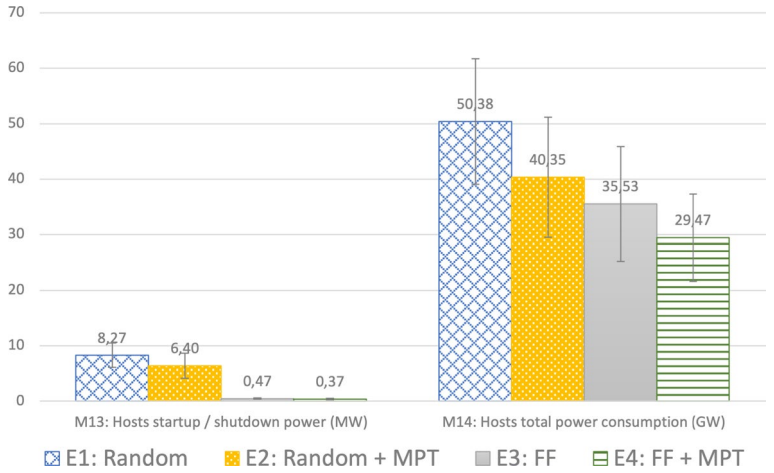


Fig. 10 Power consumption (MW and GW)

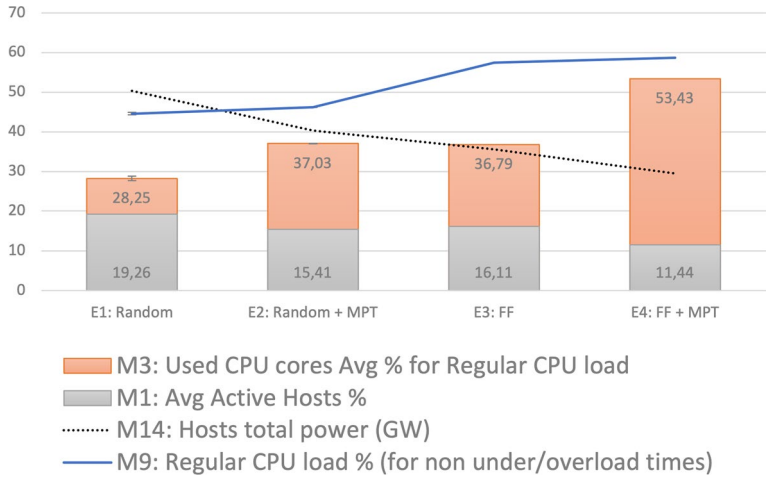


Fig. 11 Active hosts to CPU usage relation

utilization is kept high, hosts are expected to be active longer (M10), thus reducing the number of startups/shutdowns. A mostly increasing trend may be seen from E1 and E4 usually, with a small error margin. E4 has the maximum values for different CPU utilization metrics, because the proposal performs server consolidation to maximize resource utilization. Furthermore, CPU utilization average is kept the closest to the upper utilization threshold. The server consolidation performed by E4 shows that the host uptime mean is the highest among all experiments. That means fewer hosts are kept active longer. The figure shows that E4 maximizes the number of CPUs used (M3 and M4) and due to the minimization of active Hosts in Fig. 6, that explains the highest CPU load (M9).

Figure 8 shows CPU utilization for times when hosts are either under or overloaded. M5 is the average CPU utilization when hosts are overloaded. The difference in results for this metric between experiments is negligible. When hosts become overloaded, they tend to use almost the maximum capacity available. That is because an upper utilization threshold is defined merely to compare results. However, no limit in resource utilization is actually set, showing why every experiment will sometimes use all the available CPU if other resources are enough for each VM. Such metrics provide the best results when minimized, but the chart makes it clear that they are high and usually increasing from one experiment to another. That is the side effect of server consolidation approaches, as already widely discussed.

Despite the similar results for M5, the mentioned side effect causes CPU overload time to be much higher for E4. That is why setting a limit for resource utilization is critical. Without an actual limit, the MPT proposal performs a very aggressive server consolidation, which can be seen in M6 and M11.

Underload metrics are not significantly improved by any experiment because they are not attempting to address this issue. Random placement does not make any placement decision-making; it simply gets the first random host that is suitable for a VM. FF placement is concerned with performing server consolidation as the MPT proposal. In financial markets, one way to perform portfolio balancing is by buying assets that have depreciated in order to increase their portfolio share. That approach can also be used to reduce underload, but that was not directly addressed here. At any rate, considering that E4 activates fewer hosts on average, the absolute number of underload hosts will be smaller, since the percentage of underloaded hosts is around 60% for all experiments (M12).

Figure 9 shows the mean number of times hosts are activated (M15) and the consequent delay for VMs to be placed into such selected hosts (M16). E3 and E4 show each Host is started less than 2 times on average, because they are kept running longer (as Fig. 7 has shown). The reduction in M15 is clear and impressive; even the first two experiments have a large error margin. Although the reduction in M15 would indicate smaller VM wait times, it does not always work like that. Random experiments do activate a larger number of hosts, as Fig. 6 has shown. In the long run, those experiments will have a higher number of already active hosts, reducing VM placement wait time (M16), at the cost of more power consumed (as discussed below). FF and MPT experiments have a higher decision-making computational complexity. That means the random placement can be faster in finding a suitable host (at the cost of degrading several metrics), which explains the higher wait time for FF and MPT. Finally, since random experiments activate more hosts, more VMs have to wait (M17), despite waiting less (M16).

Figure 10 shows power consumption metrics, which are decreasing from E1 to E4. This happens since host utilization is maximized and they are kept active longer as we move from one experiment to another (as can also be seen in Fig. 7). The number of active hosts in Fig. 6 also corroborates this result. Although the power consumed in startups and shutdowns (M13) is negligible (compared to total power consumption in GW), those results are aligned with the number of host startups presented in Fig. 9. There is a reduction of 17% in

total power consumption from E3 to E4, which is mainly due to the reduced number of activated hosts and maximization of CPU utilization.

Figure 11 groups some of the results presented, making it easier to evaluate them. It can be seen that as the number of activated hosts (M1) decreases, the average number of CPU cores used (M3) increases. However, simply assessing the number of cores used does not provide a complete view of how CPU is being used. Besides the maximization in number of used cores, the load of those cores (M9) is also increased. That maximizes resource utilization, reducing power consumption (M14) and resource wastage. As can be seen, experiment E4 applying the MPT proposal yields the best results for all those metrics.

There is a large number of metrics that can be categorized as metrics to be either maximized or minimized. It is not easy to grasp the overall performance of the experiments by considering each individual metric. In order to address that, Eq. 11 presents $Pscore_e$, a normalized average representing a score used to analyze the performance of an experiment e, for a given category of metrics:

$$Pscore_e = \frac{\sum_{i=1}^{M} m_i / max(m_i)}{M} \tag{11}$$

where M is the number of metrics considered for computing the score, m_i is the metric value for experiment e, and $max(m_i)$ is the maximum value for the same metric among experiments E1 to E4.

Finally, there is a *Pscore* for every experiment, which takes into account each metric category. Figure 12 presents the (i) $Pscore_{max}$ considering metrics M3, M4, M7, M9, M10 to be maximized and (ii) $Pscore_{min}$ considering the remaining metrics to be minimized. These results clearly show that the proposed MPT+FF approach (E4) has the best performance for both categories of metrics. Metrics to be maximized have the highest normalized average, while the ones to be minimized have the lowest. $Pscore_{max}$ equal to 100% for E4 means that the experiment has the highest results for the metrics to be maximized.

Looking from a different perspective to the 17 metrics computed, the proposal: (i) improves M1..M4, M8..M10, M13..M15 and M17 (11 metrics); (ii) achieves small differences between all experiments for M5, M7, M12 and (iii) degrades only M6, M11, M16. It is an improvement in 64.7% of the metrics, having 17.6% of them with small differences in results and degrading 17.6% of the remaining ones. Considering (i) and (ii), that means 82.3% of the metrics improved or unchanged (14 out of 17). Finally, the performance of the MPT-based proposal leads to the best results, compared to the isolated utilization of the Random and FF approaches. These results show that our proposal is suitable for large-scale cloud computing infrastructures.

Considering the scale of the experiments carried out, a huge amount of data was generated for the extensive set of metrics presented. In order to avoid a lengthy and overwhelming discussion, additional experiment results and insights are provided in the appendix.

6.3.3 Related Work Comparison

This section discusses some results of the present proposal with the related ones in Sect. 4. The related work, and some others found in the literature, provide a limited presentation and analysis of results. They tend to: (i) suppress experiment parameters, (ii) collect a reduced number of metrics, (iii) neglect confidence intervals and (iv) lack computational complexity and memory footprint analysis. Unfortunately, that makes it unfeasible or impossible to perform a wide, accurate and meaningful comparison of results. The workload is usually not clearly described in such publications. Evaluating any proposal against them may lead to an inaccurate comparison of experiments that possibly use uneven or disproportionate workloads. Therefore, the following analysis is restricted to such limitations.

The work of Wei et al. [20] presents a MPT-based VM placement proposal that does not fully apply the theory, disregarding the generation of multiple portfolios for selection. Despite outperforming some traditional algorithms, it minimizes the number of hosts used and reduces under/overload by maximizing resource utilization. Results are very limited and present only a few metrics. For instance, numbers of active hosts are presented only in absolute values. Those numbers are not representative unless they provide the total number of hosts available to enable computing percentages.

Results show that the proposal uses the maximum of 118 hosts. However, if the simulation scenario was created, for instance, with 150 hosts, that means the proposal is either (i) inefficient in minimizing the number of active hosts (although it outperforms traditional and more simple algorithms) or (ii) the workload is very high, justifying that result. However, it is not possible to know which one is the case. Figure 6 shows that our proposal largely reduces the number of hosts used (less than 15% on average), but there is no way to compare results with [20].

The work of Wei et al. [20] shows that resource utilization follows an almost uniform distribution, trying to keep it between an under and overload threshold. However, several works surveyed in [1] show that the utilization of resources may follow different distributions, according to the kind of application, such as: batch processes, scientific applications, databases and web servers. That is why our proposal uses different distributions for synthetic data and distinct trace files to simulate varying workloads, as occurs in actual cloud infrastructures. Since [20] only provides the cumulative distribution function of resource utilization, such results are not comparable to the metrics we collected (such as average/max used CPU cores and average CPU load).

The work by Wei et al. tries to keep resource utilization close to the upper utilization threshold. However, our proposal goes beyond, and also minimizes network traffic and the time for VMPM decision-making (due to the reduced amount of data to process, already discussed in Sect. 5.3).

The results presented by Hwang and Pedram (2012 and 2018) [28, 29] are focused on evaluating the execution time of their proposed algorithms and number of VM

migrations.[2] However, more important than the absolute execution times is the computational complexity analysis those works fail to provide. The execution time of an algorithm may vary drastically, according to the power of the computer on which the experiments are run. Alternatively, presenting (i) running times in percentage values or (ii) a computational complexity analysis is more meaningful and comparable. Furthermore, as already discussed in Sect. 5.4, such works perform a traditional and complex computation of statistics that our proposal outperforms. Thus, our results cannot be comparable with theirs.

Table 6 presents a summary comparing the related work to this proposal.

7 Conclusion

NP-hard optimization problems such as VM placement are a challenge for large-scale scenarios such as the ones built through simulation in this proposal. The literature offers a vast array of proposed solutions for that issue. Due to the complexity of the problem, it is difficult to achieve breakthrough results.

Nonetheless, our current research and previous studies [1] provide a broad analysis of the literature and propose a solution that uses the Modern Portfolio Theory (MPT) to balance risk and return of VM portfolios for clusters of hosts. Using the Welford online algorithm, it defines a simplified way to compute return/risk for portfolios, which, to the best of our knowledge, has not been used for large-scale VM placement scenarios.

The proposal combines a distributed architecture, decentralized management, partial neighborhood view and return/risk balance. The Welford algorithm drastically reduced computational complexity and memory footprint, moving from linear to constant time. Considering 100 k VMs in the scenario presented in Sect. 5.4, it moves from 40 GB of utilization history to a negligible 14 MB for one year of operation. That makes our proposal suitable for large-scale cloud computing infrastructures.

Furthermore, simulation experiments used only just synthetic data, but real traces from cloud data centers. Large-scale experiments were performed and a high number of metrics were collected and broadly discussed. Many of them were improved, such as number of active hosts, CPU utilization, host uptime and power consumption, although they did degrade overload metrics.

The work collected a total of 17 metrics, applied a 3-dimensional resource analysis (enabling other resources to be included), and on average, it improves all metrics to be maximized and all those to be minimized. For 82.3% of them (14 out of 17) the proposal either improves or provides the same results compared to the other experiments. Even for some metrics which are degraded (such as CPU overload), the difference is not excessive if compared to similar approaches such as FF.

[2] VM migrations metric are not being collected since the proposal was more focused on assessing resource utilization and power consumption (unlike the related work). That is left for future research, since there is a high number of metrics evaluated already.

Table 6 Related work comparison

Work	Approach	Main characteristics	Performance	Limitations
Hwang and Pedram [28]	MPT-based static single resource management	Distributed architecture. Considers VM correlation for placement	The proposal outperforms 4 different algoritms at reducing the VM placement cost (CPU utilization standard deviation)	Centralized, fault-intolerant management with a single cloud-level resource manager. The work only considers CPU requirements, ignoring all other resources. The experiments are based only on synthetic data
Hwang and Pedram [29]	MPT-based dynamic multiple resource management	Distributed proposal assessing multiple VM resource requrements	It also reduces cost as the previuos work, compared to 4 different algorithms. The proposal running time is in the middle between other algoritms	It uses complete information view for decision-making, which is impractical for large-scale datacenters. Estimate individual resource requirements based on linear-regression, which does not apply for all kind of worloads. The experiments are based only on synthetic data
Wei et al. [20]	Dynamic correlative VM placement	Applies VM demand and volatility prediction	The proposal out-performs 3 different resource allocation algoritms in minimizing resource overload	Centralized, unscalable and fault-intolerant. No computational complexity analysis performed
This Work (2023)	Distributed VMPM approach based on modern portfolio theory	Multi-dimensional resource management. Distributed, scalable and fault-tolerant architecture with decentralized management	It has a low memory footprint and reduced computational complexity. It outperforms 3 different resources allocation algoritms, with 82.3% of the metrics improved or unchanged (14 out of 17)	It overcomes the limitations of previous proposals. Nevertheless, experiments using clusters with different sizes were not performed, despite the proposal is ready to be applied to such environments. Some resources such as storage space were not considered, just to simplify the simulations

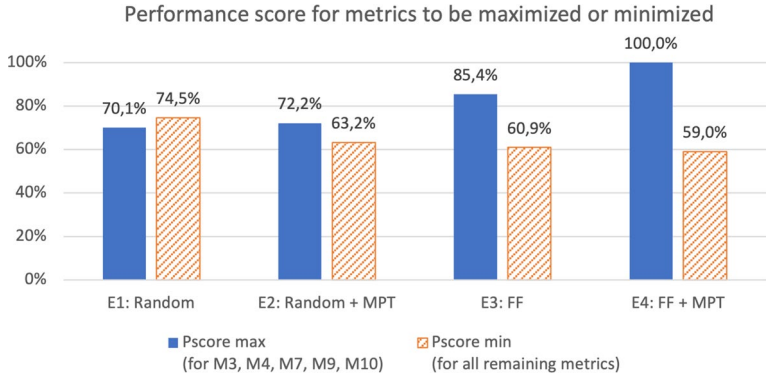


Fig. 12 *Pscore*: measuring the performance of experiments E1 to E4 by computing the normalized average for metrics to be maximized or minimized

Such an increase in overload is compensated by the large reduction of 17% in total power consumption. Although the proposal in general is more complex than FF, the decision-making overhead is negligible compared to the latter. Considering the VM average wait time as an overhead metric, the proposal increases that by only 0.32 s, while generally outperforming all the other experiments as previously discussed.

Experiments followed a strict scientific approach, performing multiple simulation runs and accordingly presenting parameters and confidence intervals. They use CloudSim Plus, which is currently a very accurate and reliable simulation framework, recognized by academia and used in many of the most relevant projects in the last years. All of that increases the reliability of experiment results.

Although this work has considered only VMs as virtual assets to be placed/migrated into a cloud infrastructure, the research can be applied to other assets such as containers, and deployment of:

(i) Functions in function as a service (FaaS) or serverless architectures,
(ii) Or distributed and parallel applications.

Finally, there are some directions for future work, which can improve the proposal and result analysis in different ways, such as for evaluating:

(i) The interference of running VMs or migration operation on co-located VMs as an additional measure of risk,
(ii) Combined reactive VM migration strategy with MPT placement to reduce under and overload,
(iii) Setting firm resource utilization limits to minimize overload,
(iv) Placement of correlated/anti-correlated VMs, regarding their mean execution time. If VMs with a similar execution time span are placed together, the host may be expected to be underloaded only when a major amount of VMs are finished. However, these correlated VMs may lead to overload. On the other hand, if VMs with anti-correlated execution time span are placed together,

that tends to reduce host overload bursts. However, that may lead to leaving the host underloaded longer, as VMs may finish at a different pace,

(v) Unallocated residual resources due to VM placement mismatch: while some kinds of resources run out, others remain plentiful, increasing resource waste,

(vi) Performance and results for multiple cluster sizes; VM migration metrics,

(vii) Placement considering availability requirements, when VMs from the same customer or application need to be placed in different availability zones,

(viii) Monetary cost evaluation for providers and customers, looking for cost-effective ($) placement for both parties.

Appendix 1 Extra Results

This section presents some additional results and discussion from the simulation experiments. The charts presented in Figs. 6, 7, 8, 9, 10 were produced based on the data from Tables 7, 8, 9, 10. The next subsections compare the results for pairs of VM Placement (VMPM) experiments.

Table 7 E1: results for random VMPM

Metric	95% CI	*Std. Dev.
M1 Active hosts Avg (%)	19.26 ± 0.31	0.84
M2 Active hosts Max (%)	99.79 ± 0.06	0.15
M3 Used CPU cores Avg (%)	28.25 ± 0.34	0.91
M4 Used CPU cores Max (%)	65.73 ± 3.60	9.65
M5 CPU overload mean (%): threshold 85.0%	97.52 ± 0.15	0.41
M6 CPU overload time (%) from uptime	5.69 ± 1.43	3.82
M7 CPU underload mean (%): threshold 30.0%	20.01 ± 0.18	0.49
M8 CPU underload time (%) from uptime	21.19 ± 3.43	9.19
M9 Hosts CPU utilization Mean for non under/overload times (%)	44.58 ± 0.29	0.78
M10 Host uptime (%) mean	19.41 ± 0.37	1.00
M11 Overload Hosts (%)	11.23 ± 0.60	1.61
M12 Underload Hosts (%)	59.58 ± 9.20	24.64
M13 Hosts startup / shutdown power (MW)	8.27 ± 2.19	5.86
M14 Hosts total power (GW)	50.38 ± 11.34	30.36
M15 Hosts startups mean	18.41 ± 4.87	13.06
M16 delayed VM mean wait time (seconds)	0.55 ± 0.02	0.06
M17 delayed VMs (%)	66.82 ± 3.94	10.56

Table 8 E3: results for FF VMPM

Metric	95% CI	*Std. Dev.
M1 Active hosts Avg (%)	16.11 ± 0.35	0.95
M2 Active hosts Max (%)	19.13 ± 0.46	1.24
M3 Used CPU cores Avg (%)	36.79 ± 0.71	1.89
M4 Used CPU cores Max (%)	92.62 ± 0.35	0.95
M5 CPU overload mean (%): threshold 85.0%	98.37 ± 0.13	0.34
M6 CPU overload time (%) from uptime	10.03 ± 1.63	4.37
M7 CPU underload mean (%): threshold 30.0%	20.14 ± 0.10	0.26
M8 CPU underload time (%) from uptime	33.18 ± 3.41	9.12
M9 Hosts CPU utilization Mean for non under/overload times (%)	57.50 ± 0.52	1.39
M10 Hosts uptime (%) mean	22.87 ± 1.94	5.21
M11 Overload Hosts (%)	80.63 ± 3.60	9.63
M12 Underload Hosts (%)	61.88 ± 5.79	15.50
M13 Hosts startup / shutdown power (MW)	0.47 ± 0.11	0.31
M14 Hosts total power (GW)	35.53 ± 10.31	27.62
M15 Hosts startups mean	1.33 ± 0.20	0.54
M16 delayed VM mean wait time (seconds)	6.04 ± 0.05	0.14
M17 delayed VMs (%)	21.22 ± 0.82	2.18

Table 9 E2: results for MPT random VMPM

Metric	95% CI	*Std. Dev.
M1 Active hosts Avg (%)	15.41 ± 0.59	1.58
M2 Active hosts Max (%)	99.79 ± 0.06	0.15
M3 Used CPU cores Avg (%)	37.03 ± 1.55	4.16
M4 Used CPU cores Max (%)	67.31 ± 3.06	8.20
M5 CPU overload mean (%): threshold 85.0%	97.44 ± 0.18	0.49
M6 CPU overload time (%) from uptime	12.20 ± 2.36	6.32
M7 CPU underload mean (%): threshold 30.0%	19.95 ± 0.19	0.52
M8 CPU underload time (%) from uptime	18.42 ± 2.53	6.78
M9 Hosts CPU utilization Mean for non under/overload times (%)	46.23 ± 0.82	2.19
M10 Hosts uptime (%) mean	15.56 ± 0.53	1.42
M11 Overload Hosts (%)	11.59 ± 0.50	1.35
M12 Underload Hosts (%)	51.23 ± 11.42	30.59
M13 Hosts startup / shutdown power (MW)	6.40 ± 2.25	6.03
M14 Hosts total power (GW)	40.35 ± 10.82	28.97
M15 Hosts startups mean	14.25 ± 5.01	13.41
M16 delayed VM mean wait time (seconds)	0.71 ± 0.08	0.21
M17 delayed VMs (%)	34.48 ± 8.31	22.26

Table 10 E4: results for MPT FF VMPM

Metric	95% CI	*Std. Dev.
M1 Active hosts Avg (%)	11.44 ± 0.35	0.94
M2 Active hosts Max (%)	14.35 ± 0.71	1.89
M3 Used CPU cores Avg (%)	53.43 ± 1.10	2.94
M4 Used CPU cores Max (%)	92.67 ± 0.35	0.95
M5 CPU overload mean (%): threshold 85.0%	97.46 ± 0.20	0.53
M6 CPU overload time (%) from uptime	38.68 ± 5.55	14.85
M7 CPU underload mean (%): threshold 30.0%	20.19 ± 0.17	0.46
M8 CPU underload time (%) from uptime	17.54 ± 2.00	5.35
M9 Hosts CPU utilization Mean for non under/overload times (%)	58.69 ± 0.72	1.92
M10 Hosts uptime (%) mean	37.73 ± 9.68	25.92
M11 Overload Hosts (%)	86.70 ± 2.01	5.37
M12 Underload Hosts (%)	62.64 ± 6.72	18.00
M13 Hosts startup / shutdown power (MW)	0.37 ± 0.12	0.33
M14 Hosts total power (GW)	29.47 ± 7.84	20.98
M15 Hosts startups mean	1.31 ± 0.13	0.35
M16 delayed VM mean wait time (seconds)	6.36 ± 1.02	2.72
M17 delayed VMs (%)	8.20 ± 2.23	5.97

1.1 VMPM Experiments: E1 vs E2

Random vs Random + MPT

(M1) Active hosts avg %: since VMs are randomly distributed across Hosts in E1, it tends to have more active hosts on average. As VMs are dynamically arriving, activated Hosts receive more workload and remain active longer. This experiment has the worst results for that metric.

E2 reduces that metric by around 4 percentage points (pp). Since VMs are better distributed between available hosts in order to maximize resource utilization, fewer hosts are active on average.

(M2) Maximum active hosts %: E1 selects hosts using a uniform PRNG. For a large-scale experiment, as the number of arrived VMs increases in the long run, the algorithm tends to maximize the number of active hosts by time. For such experiments, at some point in time, almost all available Hosts were active. Compared to the E2, this metric remains the same, since it applies the initial random placement while VMs build a utilization history. The static number of VMs, used to startup all experiments, imposes a higher resource demand at the beginning. Due to the placement randomness, more hosts are activated over time. An increase in this and the previous metric leads to higher power consumption (M13, M14).

(M3) Used CPU cores Avg % is also correlated to the previous metrics. As more Hosts are randomly activated without considering optimization of available

resources in E1, they tend to be underloaded, using a small percentage of CPUs and other resources.

Such a metric is around 37% in E2, an increase of almost 9pp. Although there is a considerable improvement in order to reduce resource waste, the value is still too low. Since the initial random placement leads to activation of a higher number of hosts (compared to other algorithms), the MPT algorithm tries to place arrived VMs with some utilization history into one of the already active hosts. It does not try migrating VMs from those under or overloaded hosts. This way, the MPT proposal simply reduces the number of hosts in such situations by trying to place more VMs into underloaded hosts and avoiding overloaded ones. If there are too many underloaded hosts, a high VM arrival rate is needed to improve that and consequently M3. Hence, the quality of the initial placement is crucial for the MPT proposal to achieve higher improvements.

(M4) Maximum percentage of used CPUs corroborates the small value for the previous metric for E1, which is the lowest among all experiments. Since most of the hosts are underloaded, the maximum utilization of CPUs is not as high as in other experiments, showing there is a large margin for improvement. While CPU utilisation has the greatest impact on power consumption, an underused host consumes the largest share of that power.

(M5) CPU overload mean % considers the host CPU as overloaded when it exceeds an 85% threshold. After a host becomes overloaded, that metric represents the average overload CPU utilization. There is no significant difference between E1 and E2 and this is an area where the MPT algorithm may be improved. Since it tries to maximize the cluster portfolio return (resource utilization), it sometimes packs too many VMs into hosts.

(M6) CPU overload time % from host uptime: results show an increase to 12% in E2, around 6pp longer overload time. That confirms findings from previous metrics and follows the same reasoning.

(M7) CPU underload mean % considers the host CPU as underloaded when it goes below a 30% threshold. There is no significant improvement in E2 either, as already discussed in M3.

(M8) CPU underload time % from Host uptime: considering the discussion in M3, the MPT proposal was not able to achieve any considerable improvement in E2. It is able to improve VM placement over time, but that is highly dependent on the initial placement.

(M9) CPU utilization mean for times hosts are not under/overloaded (%) is below 50%, showing a large margin for improvement. Conversely, E2 was only able to improve this metric by about 2pp, as it was affected by the large number of activated hosts.

(M10) hosts uptime % mean from total simulation time is more difficult to assess, since low or high values depend on the load of each Host. For instance, if a host is underloaded, four different scenarios may happen:

- (i) wait for current VMs to finish or (ii) migrate them out in order to shut down the host (reducing uptime);
- (iii) migrate VMs in or (iv) place new ones into the host to reduce underload (increasing uptime).

Since the MPT proposal is not currently accountable to perform automatic VM migration (despite it can be incorporated), only scenarios (i) and (iv) are possible. This way, depending on the scenarios that happen more frequently, an improvement in M10 may be either an increase or decrease on that metric. For E2, results show hosts were active 15% of the time, which is 4pp below from E1. If M15 is analysed together, it can be seen that the number of times hosts are switched on and off is 4pp less than E1, as already discussed. Therefore, the MPT proposal for E2 activates a host and tries to keep it active as shorter as possible, while reducing the number of host switches.

Considering that the initial random placement inevitably activates more hosts, scenario (i) presented before may predominate. When the MPT algorithm is applied, it tries to maximise resource utilization. That leads to place more VMs into hosts with a higher resource utilization, up to a defined threshold. Therefore, those hosts will be active longer. If the entire datacenter is not overloaded, there may be a larger number of hosts running a smaller number of VMs. Those hosts become idle sooner and are shutdown. Such a scenario would explain the smaller hosts uptime mean for E2.

(M11) Overload Hosts %: even with a smaller number of hosts (M1) used in E2, the average percentage of overloaded hosts is almost the same. The placement considering risk and return of co-located VMs plays a role in such results, since the Modern Portfolio Theory enables a balance between such metrics. Overloded hosts lead to extend execution time of (i) batch processes (jobs) that eventually finish; and (ii) requests for web and application servers, increasing queued requests wait time. That imposes a higher power pressure. Reducing overload also reduces power consumption (M14).

(M12) Underload Hosts % also confirms the scattered placement performed by the random algorithm in E1, since the percentage of underloaded hosts is high. That causes a huge resource waste, as already discussed broadly in previous metrics such as M4. That initial number of activated hosts is too high, due to lack of a server consolidation approach to maximize resource utilization. Conversely, E2 reduces M12 by around 9pp, after applying the MPT placement. That directly impacts power consumption (M14), which is reduced by 10pp.

(M13) Hosts startup/shutdown power (MW) is largely affected by the scattered VM placement performed by the random algorithm in E1. Hosts are set up to shut down after becoming idle for some time. Since startup and shutdown operations demand extra power, they increase the total power consumption (M14). Although the former (in MW) is negligible compared to the later (in GW), M13 has the highest value among all experiments. Compared to E4 using the FF + MPT placement, it is about 21 times more.

(M14) Hosts total power consumption (GW) for the entire simulation execution is measured in gigawatts. E2 decreases it by 10pp due to reduced: (i) number of activated hosts resulting from server consolidation (M1) and (ii) resource waste (such as M3, M4, M7 and M8).

(M15) Nnumber of times Hosts are started up on average: the initial random placement in E1 makes hosts receive a smaller number of VMs. This way, such hosts tend to become completely idle quicker, as their VMs finish, then shut down. Since that placement does not prioritize active Hosts, the number of host startups and shutdowns increases, as can be seen in this metric. All other experiments reduce that: (i) E2 reduces it 4pp; (ii) E3 and E4 achieve great results for that metric, switching hosts on and off less than 2 times on average, while maximizing resource utilization. That represents a reduction of more than 90%.

(M16) Ddelayed VM mean wait time (seconds) is correlated to the previous metric. The placement algorithm has a leading role in determining the time a VM has to wait until a suitable host is found. Such an algorithm impacts the number of Host startups. That process takes some time, which adds to the VM wait time. However, due to the low complexity of algorithms in E1 and E2, results show a very small VM wait time. E2 slightly increases it. Despite the high number of host switches, after a host is started up, arriving VMs placed into that host will not have to wait for its activation. Since those proposals activate a higher number of hosts, that leads do reduced average wait time.

(M17) Delayed VMs % shows that 66% of VMs in E1 had to wait for a host be activated. As the random algorithm in E1 performs a more scattered placement of VMs across hosts, leading to more host activation, there are more VMs being placed into a host that was previously inactive. E2 reduces it by almost half, since more VMs are placed into the first suitable active host before activating a new one. However, since the FF has to perform a search for a suitable Host, that slightly adds to the VM wait time. E1 may select, in the first try, a random host that is suitable for a VM simply by chance, reducing the mean wait time.

2.2 VMPM Experiments: E3 vs E4

FF vs FF + MPT

(M1) Active hosts avg % was expected to be lower than in previous experiments when more suitable placement policies are applied. E3 achieves about 3pp reduction compared to E1, but it is higher than random + MPT E2. Even in such an experiment that starts from a worse placement policy, the MPT algorithm applied over that initial random placement can still achieve better results than FF alone. The FF + MPT in E4 has the best results, reducing almost 5pp more from E3.

(M2) Maximum active hosts % has a huge difference in results compared to previous experiments. Since the initial FF placement in E3 and E4 performs server consolidation, that drastically reduces the maximum number of activated hosts.

While almost all hosts were activated in the first experiments, for E3 at most 19% of them were active at the same time. That is about 5pp lower in E4.

(M3) Used CPU cores Avg % is increasing for each experiment, due to an improved server consolidation process (lower number of active hosts as just discussed). E2 is even 1pp better than E3, although the former starts from an improper random placement. E4 has the most remarkable results, increasing that metric by almost 17pp compared to E3. Although 54% of used CPUs is still low, that is highly dependent on the imposed workload. Large-scale experiments were performed, taking dozens of hours to finish. In order to avoid extending that simulation time, the workload was not increased to ensure a higher average CPU utilization.

(M4) Maximum percentage of used CPUs is higher (as expected) considering the increase in the previous metric. Results are technically the same for E3 and E4 (92%). The similar results for each pair of experiments are due to the utilization of the same placement policy at the beginning, before VMs build a utilization history. In designed experiments, that is when the highest workload occurs. In comparison to previous experiments, it is an increase of around 25 pp.

(M5) CPU overload mean % is practically the same for all experiments. It can be seen that on average, when a host becomes overloaded (above the defined threshold), it becomes highly overloaded. Here is a scenario where the proposal can be improved by introducing reactive VM migration, moving the host as quickly as possible from this overload state.

(M6) CPU overload time % from host uptime is more important than the previous metric. The shorter the time hosts stay in overload state the better. By applying a server consolidation algorithm such as FF or MPT VMPM, a higher probability of overload is expected. Using algorithms such as the latter, we attempt to find a balance between overload/SLA violations and resource optimization/power consumption/costs. Accordingly, a decreasing trend in power consumption can be seen (M14).

(M7) CPU underload mean % is almost the same for all experiments. There is no clear reason for such results, which indicates the need for further research. It shows that when a Host becomes underloaded (considering the defined threshold), its CPU utilization is around 20% on average.

(M8) CPU underload time % from Host uptime is the highest in E3. Considering that E1 performs random placement, it uniformly distributes all VMs across available hosts, trying to balance the load in the same way. Consequently, there will be fewer under and overloaded hosts on average, as the results already show.

Server consolidation such as FF algorithm tends to concentrate more VMs into fewer hosts. As more VMs arrive, a new host needs to be activated. That host may remain underused and be idle longer, until enough VMs have arrived to maximize its utilization. Since reactive VM migration is not enabled, the underload time increases.

The random placement leads to activation of more hosts with fewer VMs. Hosts end up being shut down sooner, as their small number of VMs finish

quicker. M15 confirms that the number of times hosts are switched on and off can be 13 times smaller, compared to E1.

Despite the highest underload time in E3, power consumption is linearly reduced (M14) due to a better initial placement. In summary, underload time, as many metrics, cannot be assessed by itself.

(M9) CPU utilization mean for times hosts are not under/overloaded (%) have only a small improvement of E4, 1pp, compared to E3. However, compared to E2 it is around 12pp higher. Compared to M3, this experiment shows that the time hosts are under or overloaded had a small impact on the average CPU utilization.

(M10) Hosts uptime % mean from total simulation time shows that hosts stay active longer than in previous experiments. E4 increases this metric almost twofold. It reduces the power consumed by switching hosts on and off (M13), keeping hosts operating longer. Despite the small reduction in power consumption for M13, there is an increase in the time VMs have to wait for a host to be activated. Since hosts are only activated when needed (different from random placement), as already active hosts are operating at their maximum capacity, more arriving VMs have to wait for the host activation.

(M11) Overload Hosts % is expected to be higher, compared to previous experiments. That is a side effect of server consolidation approaches, which must be balanced to meet service level agreements (SLAs). However, such adjustments were not addressed in this proposal, although the MPT VMPM algorithm is able to deal with it. The percentage of hosts that were overloaded sometimes reaches 80% for E3 and 86% E4. However, that metric is better evaluated with M6. The bottom line is that the MPT proposal is increasing the overload time and overloaded hosts.

(M12) Underload Hosts % is high in both E3 and E4. The reasoning is the same as already explained for M8. E4 has practically the same percentage of underloaded hosts, although for a shorter time period (M8).

(M13) Hosts startup/shutdown power (MW) consumption is about 20% smaller in E4, due to the reduced number of host switches (M15). Although it adds little to the total power consumption (M14), there is a great reduction compared to previous experiments. Results from E1 to E3 and E2 to E4 are almost 18 times smaller.

(M14) Hosts total power consumption (GW) shows a clear decreasing trend, as already discussed in previous metrics. E3 reduces it by 15pp compared to E1 and 5pp compared to E2. E4 reduces it by 20pp compared to E1 and 6pp compared to E3.

(M15) Number of times Hosts are started up on average has been initially discussed in M8. Results from E3 and E4 are technically the same and show that hosts are switched on/off fewer than 2 times on average.

(M16) Delayed VM mean wait time (seconds) is impacted by the previous metric and again have similar results. There is up to a tenfold increase from E1 and E2. However, that is still small considering the time unit is seconds. The reasoning behind such results also was discussed in M8. But considering the higher com-

plexity of FF and MPT VMPM compared to random placement, that increase in wait time is expected.

(M17) Delayed VMs % is correlated to the previous metric. Results show that each experiment largely reduces that metric. E4 reduces it by almost 3 times, compared to E3. Since the former tries to maximize resource utilization for already active hosts and reduces the number of hosts used on average, that leads to less VMs waiting for a host activation.

Acknowledgements We would like to thank the *Universidade da Beira Interior*, the *Instituto de Telecomunicações* of Portugal, the Portuguese *FCT/MCTES*, the *Centro de Competências em Cloud Computing (C4)*, and the *Instituto Federal de Educação, Ciência e Tecnologia do Tocantins*, which are partners in this work. We also would like to thank Google for providing access to the Google Cloud Platform Research Credits program that enabled execution of large-scale simulations in the cloud.

Author Contributions MCSF: PhD candidate, main paper author, who designed, implemented and tested the presented proposal. MMF: PhD supervisor, who guided the writing and implementation process, reviewing the paper along the way. CCM: PhD co-supervisor, who guided the writing and implementation process, reviewing the paper along the way and helped to set up the infrastructure. PRMI: research group member, who guided simulation experiments, proposed ways to present results and ensured results validation.

Funding Open access funding provided by FCT|FCCN (b-on). This work is funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—CAPES (Manoel C. Silva Filho Scholarship—Process No 13585/13-4), the Portuguese Fundação para a Ciência e a Tecnologia (FCT/MCTES) through national funds and, when applicable, co-funded by EU funds under the project UIDB/50008/2020 and by FCT/COMPETE/FEDER under the project SECURIoTESIGN with reference number POCI-01-0145-FEDER-030657, and funded by operation Centro-01-0145-FEDER-000019—C4—Centro de Competências em Cloud Computing, co-funded by the European Regional Development Fund (ERDF/FEDER) through the Programa Operacional Regional do Centro (Centro 2020). It is also based upon work supported by the Google Cloud Research Credits program with the award GCP19980904.

Data Availability The workload trace files used are available at https://github.com/cloudsimplus/planetlab-workload-traces and https://github.com/google/cluster-data.

Declarations

Conflict of interest The authors declare that they have no competing interests.

Ethical Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication All authors consent in publishing this research paper in the Journal of Network and Systems Management.

References

1. Silva Filho, M.C., Monteiro, C.C., Inácio, P.R.M., Freire, M.M.: Approaches for optimizing virtual machine placement and migration in cloud environments: a survey. J. Parallel Distrib. Comput. **111**, 222–250 (2018). https://doi.org/10.1016/j.jpdc.2017.08.010
2. Ahmad, R.W., Gani, A., Hamid, S.H.A., Shiraz, M., Yousafzai, A., Xia, F.: A survey on virtual machine migration and server consolidation frameworks for cloud data centers. J. Netw. Comput. Appl. **52**, 11–25 (2015). https://doi.org/10.1016/j.jnca.2015.02.002
3. Masdari, M., Nabavi, S.S., Ahmadi, V., Ahmad, R.W., Gani, A., Hamid, S.H.A., Shiraz, M., Yousafzai, A., Xia, F., Masdari, M., Nabavi, S.S., Ahmadi, V.: An overview of virtual machine placement schemes in cloud computing. J. Netw. Comput. Appl. **66**, 106–127 (2016). https://doi.org/10.1016/j.jnca.2016.01.011
4. Mann, Z.Á.: Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. ACM Comput. Surv. **48**(1), 1–34 (2015). https://doi.org/10.1145/2797211
5. Usmani, Z., Singh, S.: A survey of virtual machine placement techniques in a cloud data center. Int. Conf. Inform. Secur. Priv. **78**, 491–498 (2015). https://doi.org/10.1016/j.procs.2016.02.093
6. Nhapi, L., Yadav, A.K., Rao, R.S.: Virtual machine provisioning for cloud scenarios. In: Proceedings of the second international conference on information and communication technology for competitive strategies, pp. 1–6. ACM Press, New York (2016)
7. Pietri, I., Sakellariou, R.: Mapping virtual machines onto physical machines in cloud computing. ACM Comput. Surv. **49**(3), 1–30 (2016). https://doi.org/10.1145/2983575
8. Zhang, J., Huang, H., Wang, X.: Resource provision algorithms in cloud computing: a survey. J. Netw. Comput. Appl. **64**, 23–42 (2016). https://doi.org/10.1016/j.jnca.2015.12.018
9. Markowitz, H.M.: Portfolio selection: efficient diversification of investments, p. 356. Springer, Berlin (1959)
10. Brealy, R.A., Myers, S.C., Allen, F.: Principles of corporate finance, pp. 1–889. Elsevier, Amsterdam (2014)
11. Mckenna, P.: Modern portfolio theory: driving project portfolio management with investment techniques (2005). https://web.archive.org/web/20140719100623/http://www.ibm.com/developerworks/rational/library/aug05/mckenna/index.html. Accessed 09 Oct 2021
12. AWS: Amazon EC2 Spot Instances (2016). https://aws.amazon.com/ec2/spot/. Accessed 23 Aug 2021
13. Services, A.W.: Amazon Web Services in China. https://www.amazonaws.cn/en/about-aws/china/. Accessed 9 Nov 2022
14. Evstigneev, I.V., Hens, T., Schenk-Hoppé, K.R.: Mean-variance portfolio analysis: the markowitz model. In: Mathematical financial economics: a basic introduction, pp. 11–19. Springer, Berlin (2015)
15. Berrada, T., Chaieb, I., Demaurex, J., Brandon, R.G., Girardin, M., Krueger, P., Preuschoff, K., Scaillet, O.: Portfolio and risk management (2018). https://www.coursera.org/learn/portfolio-risk-management/. Accessed 08 Aug 2021
16. Markowitz, H.: Portfolio selection. J. Finance **7**(1), 77–91 (1952)
17. Buyya, R., Garg, S.K., Calheiros, R.N.: SLA-oriented resource provisioning for cloud computing: challenges, architecture, and solutions. In: Proceedings of international conference on cloud and service computing, pp. 1–10. IEEE, Hong Kong (2011)
18. Kapil, D., Pilli, E.S., Joshi, R.C.: Live virtual machine migration techniques : survey and research challenges. In: IEEE 3rd international advance computing conference (IACC), pp. 963–969. IEEE, Ghaziabad (2012)
19. Sohrabi, S., Moser, I.: The Effects of hotspot detection and virtual machine migration policies on energy consumption and service levels in the cloud. Procedia Comput. Sci. **51**, 2794–2798 (2015). https://doi.org/10.1016/j.procs.2015.05.436
20. Wei, W., Wei, X., Chen, T., Gao, X., Chen, G.: Dynamic correlative VM placement for quality-assured cloud service. In: IEEE international conference on communications (ICC), pp. 2573–2577. IEEE (2013)
21. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J. Comput. Syst. Sci. **79**(8), 1230–1242 (2013). https://doi.org/10.1016/j.jcss.2013.02.004

22. Ranjana, R., Raja, J.: A survey on power aware virtual machine placement strategies in a cloud data center. In: International conference on green computing, communication and conservation of energy (ICGCE 2013), pp. 747–752. IEEE, Chennai (2013)
23. Xu, F., Liu, F., Liu, L., Jin, H., Li, B.: iAware: making live migration of virtual machines interference-aware in the cloud. IEEE transactions on computers **63**(12), 3012–3025 (2013). https://doi.org/10.1109/TC.2013.185
24. Torres, G., Ave, C., Liu, C., Ave, C.: The impact on the performance of co-running virtual machines in a virtualized environment. In: Florin Pop, R.P. (ed.) Proceedings of the third international workshop on adaptive resource management and scheduling for cloud computing, pp. 21–27. ACM, New York (2016)
25. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurr. Comput. Pract. Exp. **24**, 1397–1420 (2012)
26. Masoumzadeh, S.S., Hlavacs, H.: Integrating VM selection criteria in distributed dynamic VM consolidation using Fuzzy Q-learning. In: 9th international conference on network and service management (CNSM 2013) and its three collocated workshops, pp. 332–338. IEEE, Zurich (2013)
27. Services, A.W.: Amazon S3 Pricing. https://aws.amazon.com/s3/pricing/. Accessed 05 Nov 2022
28. Hwang, I., Pedram, M.: Portfolio theory-based resource assignment in a cloud computing system. In: Proceedings of IEEE 5th international conference on cloud computing, pp. 582–589. IEEE (2012)
29. Hwang, I., Pedram, M.: Hierarchical, portfolio theory-based virtual machine consolidation in a compute cloud. IEEE Trans. Serv. Comput. **11**(1), 63–77 (2018). https://doi.org/10.1109/TSC.2016.2531672
30. Masoumzadeh, S.S., Hlavacs, H.: A gossip-based dynamic virtual machine consolidation strategy for large-scale cloud data centers. Proceedings of the third international workshop on adaptive resource management and scheduling for cloud computing (ARMS-CC'16), pp. 28–34 (2016). https://doi.org/10.1145/2962564.2962565
31. Marzolla, M., Babaoglu, O., Panzieri, F., Zamboni, M.A.: Bologna, I.-: server consolidation in clouds through gossiping. In: IEEE international symposium on a world of wireless, mobile and multimedia networks, pp. 1–6. IEEE, Lucca (2011)
32. Zhao, Y., Huang, Y., Chen, K., Yu, M., Wang, S., Li, D.: Joint VM placement and topology optimization for traffic scalability in dynamic datacenter networks. Comput. Netw. **80**, 109–123 (2015). https://doi.org/10.1016/j.comnet.2014.12.014
33. Kanagavelu, R., Lee, B.-S., Le, N.T.D., Mingjie, L.N., Aung, K.M.M.: Virtual machine placement with two-path traffic routing for reduced congestion in data center networks. Comput. Commun. **53**, 1–12 (2014). https://doi.org/10.1016/j.comcom.2014.07.009
34. Welford, B.P.: Note on a method for calculating corrected sums of squares and products. Technometrics **4**(3), 419–420 (2012). https://doi.org/10.2307/1266577
35. Filho, M.C.D.S.: Java Gossip Simulator: An API to enable simulating the dissemination of data across nodes in a network using the gossip epidemic protocol (2020). https://github.com/manoelcampos/java-gossip-simulator. Accessed 09 Aug 2021
36. Silva Filho, M.C., Oliveira, R.L., Monteiro, C.C., Inácio, P.R.M., Freire, M.M.: CloudSim Plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In: IFIP/IEEE international symposium on integrated network management. Lisbon, Portugal, p. 7. (2017). https://doi.org/10.23919/INM.2017.7987304
37. Bendechache, M., Svorobej, S., Endo, P.T., Mario, M.N., Ares, M.E., Byrne, J., Lynn, T.: Modelling and simulation of ElasticSearch using CloudSim. In: 2019 IEEE/ACM 23rd International symposium on distributed simulation and real time applications (DS-RT), pp. 1–8 (2019). https://doi.org/10.1109/DS-RT47707.2019.8958653
38. Wiesner, P., Thamsen, L.: LEAF: simulating large energy-aware fog computing environments. arXiv preprint http://arxiv.org/abs/2103.01170 (2021)
39. Mechalikh, C., Taktak, H., Moussa, F.: PureEdgeSim: a simulation framework for performance evaluation of cloud, edge and mist computing environments. Comput. Sci. Inform. Syst. **18**, 43 (2020)
40. Wei, J., Cao, S., Pan, S., Han, J., Yan, L., Zhang, L.: SatEdgeSim: a toolkit for modeling and simulation of performance evaluation in satellite edge computing environments. In: 2020 12th

International conference on communication software and networks (ICCSN), pp. 307–313. IEEE (2020)
41. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inform. Syst. **24**(3), 45–77 (2007)
42. Vaishnavi V., K.W., Petter, S.: Design science research in information systems. Association for Information systems (2019). http://www.desrist.org/design-research-in-information-systems/. Accessed 09 Oct 2021
43. University, P.: PlanetLab: An open platform for developing, deploying and accessing planetary-scale services (2021). https://planetlab.cs.princeton.edu. Accessed 09 Oct 2021

Manoel C. Silva Filho He is a professor in the Computer Department at IFTO, working on professional, higher education and specialization courses. He holds a Master degree in Electrical Engineering from Universidade de Brasília (Brazil), where he developed applications and frameworks for the Brazilian Digital Television System. Currently he is a Computer Engineering Ph.D. student at UBI, working with computer-based simulation and resource optimization in Cloud Computing environments. He is the main developer and maintainer of CloudSim Plus, the state-of-the-art and general-purpose cloud computing simulation framework available at https://cloudsimplus.org.

Claudio C. Monteiro He holds an Electrical Engineering Ph.D. from UNB, where he proposed a framework to reduce hadover latency between heterogeneous wireless networks. He is a professor in the Computer Department at IFTO, having experience with computer science, wireless networks and protocols, QoS/QoE and operating systems. He is the regional secretary for SBC in the Tocantins state (Brazil).

Pedro Ricardo M. Inácio He is an Associate Professor of Computer Science at UBI, which he joined in 2010. He holds a Computer Science and Engineering Ph.D. from UBI. The Ph.D. work was performed in the enterprise environment of Nokia Siemens Networks Portugal. He is an IEEE senior member and a researcher of the Instituto de Telecomunicações. His main research topics are information assurance and security, computer based simulation, and network traffic monitoring, analysis and classification.

Mário M. Freire He received the 5-year BS degree in Electrical Engineering and the two-year MS degree in Systems and Automation in 1992 and 1994, respectively, from the University of Coimbra, Portugal. He received the PhD degree in Electrical Engineering in 2000 and the Habilitation title in Computer Science in 2007 from the University of Beira Interior (UBI), Portugal. He is a full professor of Computer Science at UBI, which he joined in the Fall of 1994. In April 1993, he did one-month internship at the Research Centre of Alcatel-SEL (now Nokia Networks) in Stuttgart, Germany. His main research interests fall within the area of computer systems and networks, including network and systems virtualization, cloud and edge computing and security and privacy in computer systems and networks. He is the co-author of seven international patents, co-editor of eight books published in the Springer LNCS book series, and co-author of about 130 papers in international journals and conferences. He serves as a member of the editorial board of the ACM SIGAPP Applied Computing Review, serves as associate editor of the Wiley Security and Privacy journal and of the Wiley International Journal of Communication Systems, and served as editor of IEEE Communications Surveys and Tutorials in 2007–2011. He served as a technical program committee member for several IEEE international conferences and is co-chair of the track on Networking of ACM SAC 2022. Dr. Mário Freire is a chartered engineer by the Portuguese Order of Engineers and he is a member of the IEEE Computer Society and of the Association for Computing Machinery.

Authors and Affiliations

Manoel C. Silva Filho[1,2] · Claudio C. Monteiro[1] · Pedro Ricardo M. Inácio[2] · Mário M. Freire[2]

✉ Manoel C. Silva Filho
mcampos@ifto.edu.br

Claudio C. Monteiro
ccm@ifto.edu.br

Pedro Ricardo M. Inácio
inacio@ubi.pt

Mário M. Freire
mario@ubi.pt

[1] Departamento de Informática, Instituto Federal de Educação, Ciência e Tecnologia do Tocantins (IFTO), Palmas, Tocantins 77021-090, Brazil

[2] Instituto de Telecomunicações (IT) e Departamento de Informática, Universidade da Beira Interior (UBI), Covilhã, Castelo Branco 6201-001, Portugal