BDMaaS+: Business-driven and Simulation-based Optimization of IT Services in the Hybrid Cloud

Walter Cerroni, *Senior Member, IEEE,* Luca Foschini, *Senior Member, IEEE,* Genady Ya. Grabarnik, *Senior Member, IEEE,* Filippo Poltronieri, *Student Member, IEEE,* Larisa Shwartz, *Senior Member, IEEE,* Cesare Stefanelli, *Member, IEEE,* and Mauro Tortonesi, *Member, IEEE*

***Abstract*—The maturity of heterogeneous and hybrid public Cloud environments enables service providers to deploy there their complex IT services trusting these large and complex infrastructures. At the same time, evaluating the impact of changes at service configuration before and at the runtime is still a very challenging and difficult task. Moreover, a comprehensive performance evaluation of IT service configurations should not be limited just to costs for IT resource acquisition, but also include risk related elements such as Service Level Agreement (SLA) violation penalties and other intangibles. To support IT service providers in this difficult task, we developed Business-Driven Management as a Service Plus (BDMaaS+), a novel decision support tool that can evaluate IT service configuration through simulation with realistic service and network models. By allowing service providers to define expanded operational parameters, BDMaaS+ also enables what-if scenario analysis, thereby opening interesting possibilities at the planning level. Experimental results, collected from our thorough evaluations, demonstrate how a service provider can leverage BDMaaS+ to explore the potential of high-level business SLA changes and data center additions.**

***Index Terms*—Cloud Computing, Business-Driven IT Management (BDIM), Optimization, Simulation.**

I. Introduction

Technological advances and aggressive commercial offerings in Cloud computing are pushing an ever growing number of companies to migrate part of the IT services hosted in their private data centers to the Cloud. Along this direction, *Cloud bursting*, namely, the practice of temporarily leveraging Cloud-based virtual resources to deal with computationally very expensive tasks or significant spikes in request loads, is becoming more and more commonly adopted in privately hosted IT services. These phenomena are opening brand new research issues to govern in a dynamic fashion the management of complex services deployed in highly intermixed and distributed public-private virtualized Cloud environments, namely, *hybrid Cloud* scenarios [1].

Hybrid Cloud scenarios present service providers with compelling opportunities to optimize their IT architecture, e.g., by moving some components to different Cloud data centers or switching to Service Level Agreements (SLAs) that are more convenient from both the service provider's and the end customer's perspectives. However, the high sophistication of modern IT services and the complexity of hybrid Cloud environments make it extremely difficult, at the business level, to evaluate the impact of changes to an IT service architecture before deploying them [2]. The performance assessment of possible alternative deployments calls for new and sophisticated service management tools that provide *what-if scenario analysis* functions. Those tools should be capable of exploring alternative IT service architectures and of evaluating their performance through a comprehensive business level behavioral analysis, with the purpose of identifying the most convenient one.

However, the realization of similar management tools presents significant challenges in IT service modeling and evaluation. First, modern hybrid Cloud IT services implement a large number of workflows on top of many software components of different types deployed in heterogeneous environments. The complexity of those deployments makes it difficult to estimate the impact of (even simple) re-configurations of the IT service architecture. Second, at the current stage, there is still no widely recognized standard-de-facto observatory of the performances provided by main Cloud players. That is true for the several dimensions characterizing a Cloud offering, such as unit costs and provided service levels (e.g., response time and network delay) both within the same or between different data centers (intra-/inter-data center latency). Third, while a number of studies and theoretical models have been proposed that effectively address the service placement problem [3] [4] [5], only few simulation tools are available [6]. To the best of our knowledge none of them is able, at the same time, to: i) provide a realistic model for the Internet and inter-data center delays in a hybrid Cloud scenario; ii) take into account complex business-level specifications; and iii) provide realistic templates of complex multi-tier application workflows.

To overcome all the open issues listed above we propose a novel solution called Business-Driven Management as a Service Plus (BDMaaS+) that shows several new elements of technical novelty. First, it supports the placement of realistic multi-tier services consisting of complex workflows made by multiple application components of different types (e.g., Web Server, App Server, Relational Databases, Transaction Servers and Queue Managers). Such a placement is based on real network measurements and monetary costs for a large-scale Cloud computing environment, implemented on top of 6 dif-

W. Cerroni and L. Foschini are with University of Bologna, Bologna, Italy e-mail: {walter.cerroni, luca.foschini}@unibo.it.

G. Ya. Grabarnik is with St. John's University, Queens, NY, USA e-mail: grabarng@stjohns.edu.

F. Poltronieri, C. Stefanelli and M. Tortonesi are with University of Ferrara, Ferrara, Italy e-mail: {filippo.poltronieri, cesare.stefanelli, mauro.tortonesi}@unife.it.

L. Shwartz is with IBM TJ Watson Research Center, NY, USA e-mail: lshwart@us.ibm.com.

ferent Amazon EC2 data centers and 2 private Clouds. Second, BDMaaS+ leverages our experience in both service/system modeling and inter-data center network delay modeling to feed our novel simulator with realistic characterizations, to be accounted for by the dynamic re-adaptation of component deployment at runtime [7]. Third, it adopts a simulative approach to reenact IT services under different configurations to accurately capture peculiar behavior of real-life IT services. In particular, it realizes an innovative optimization solution based on a memetic algorithm to enable robust and resilient exploration of the large and challenging search space, thus realizing an effective what-if scenario analysis tool [8]. Fourth, BDMaaS+ has been implemented and used to collect a wide set of experimental results that show the benefits and original aspects of our proposal demonstrating the effectiveness of our solution. We also make BDMaaS+ available to the community working in the field[1].

This manuscript is an extension of previous works [9], [8] and [10] published, respectively, at the TCC journal, and at the CNSM 2018 and the IM 2019 conferences. Those papers illustrated the models adopted by BDMaaS+ for business-level performance assessment of IT services in hybrid Cloud environments and showcased the effectiveness of our solution for what-if scenario analysis purposes. The present contribution significantly extends and enhances our previous efforts along several directions. First, we enhanced the background and related works part (see Section II) through a thorough reorganization of surveyed existing efforts in the literature. Second, we detail the optimization solution at the core of BDMaaS+, based on a memetic algorithm leveraging Quantum-inspired Particle Swarm Optimization that was never presented before and significantly improves the previous optimization solution based on genetic algorithms, by devoting to it the whole Section VI. Third, to further validate and show the distinctive advantages offered by BDMaaS+, we present in Subsection VII-B a brand new comparison of our business-driven optimization approach with two others categories that summarize a wide part of the existing efforts in the literature, namely those based on IT costs only and those including also service level objective violation penalties.

II. Background and Related Work

Optimal placement of VMs on Cloud platforms is a well-studied topic in literature. There is, in fact, the need to find suitable resource allocation schemes capable of minimizing monetary costs for both service and Cloud providers[2]. In the scope of this paper, we focus on the service providers perspective to deal with the challenges they face in delivering a service to a multitude of costumers located across the globe.

Among the overall works, we divide those efforts into different categories based on the primary objective these works try to achieve in formalizing an optimal allocation of VMs on Cloud providers, corresponding to different philosophies on the estimation of costs involved at the business level for running IT systems. Therefore, we distinguish between three categories: *IT costs (IC)*, *IT costs and Service Level Objective (SLO) violation penalties (IC+SLO)*, and *business-driven (BD)* optimization. We provide a background discussion of those approaches and a survey of articles in scientific literature addressing the optimal placement of VMs considering the above classification in the following subsections.

A. Background

In the first category, IC represents the infrastructural costs to deploy a service/application on Cloud Computing providers. More specifically, IC expresses the prices of VM units on either different Cloud Computing providers or different data-centers belonging to the same provider. Let us note that these costs constitute the main part of the whole monetary costs to optimize. By focusing on this element, the majority of works investigate how to find an optimal allocation capable of minimizing the overall IT costs [11]–[16].

Within the IC+SLO category, we consider those works that still take into account the infrastructural costs as a primary objective for optimization, but also introduce and deal with SLAs. SLAs define certain characteristics that a service should provide to its users/customers. This happens when a service provider has to guarantee to its clients that an application would respect certain criteria/standard, e.g. the 95th percentile of requests must be satisfied within 200 ms. Consequently, if these requirements are not met, service providers are in violation of the agreement and might have to pay in a monetary penalty depending on the SLAs.

Therefore, the IC+SLO approach aims at finding a minimal cost allocation that respects the defined SLAs. Let us also specify that some work [17] define SLAs to specify the requirements a VM allocation should provide/guarantee. However, such solutions do not associate a monetary cost to SLO violations but only invalidate the allocations that do not satisfy the defined SLAs [17]–[21].

Finally, works in the *business-driven (BD)* category aim to optimize service performance from a more comprehensive business-level perspective [22]. In addition to IT costs and SLO violations, they consider additional KPIs, allowing to consider in the evaluation also criteria pertaining to business-level domains such as human resource management and risk management, and even intangibles. For instance, aggressively looking for the minimum set of resources to run an IT service might lead to an excessively brittle system with poor scalability and redundancy and/or to customer dissatisfaction and consequent monetary losses. Additionally, some system configuration might be impossible to manage with the existing personnel and would thus call for further hirings. BD approaches allow to capture all these KPIs through a comprehensive and uniform framework that measures all aspects of system performance in monetary terms, including both real (IT costs, expected SLO violations) and virtual (potential fallback of risk posture, required extra salaries, other intangibles) costs.

While the tool is perfectly capable of implementing the IC and IC+SLO optimization approaches, we specifically

[1]For more information about the BDMaaS+ software, installation, configuration language, experimental results, etc., we refer the reader to the project home page: https://ds.unife.it/research/bdmaas.

[2]In the reminder of the paper, without loss of generality, we focus on the Infrastructure as a Service (IaaS) level and we consider Virtual Machines (VMs) as the resource units to allocate.

developed BDMaaS+ to enable a BD optimization approach. It is needless to say that this model represents a significant difference from other works in the literature such as [11]–[16], which mainly consider IC approaches for optimizing the allocation of VMs within multiple data centers (private and public).

B. Related Work

Early efforts in service placement in Cloud environments focused on load-balancing related objectives. Significant research addressed this topic at the infrastructure level by considering mainly internal IT objectives such as SLAs for service providers and technical requirements (e.g., physical host's CPU, memory, etc.). A number of good surveys are also available in this field [23] [24]. In [25], the authors investigate load balancing in Cloud data centers with server and storage virtualization facilities while considering multiple layers like servers, storage, and switches solving it using multi-dimensional knapsack. Network-aware VM placement while reducing the aggregate traffic into the data center (e.g., by co-locating VMs that communicate much with each other) is considered in [26]. VM placement that is resilient to dynamic traffic time-variations, i.e., minimizes the number of VM relocations, is considered in [27]. An interesting article looking into the VM reassignment/relocation problem in hybrid Cloud infrastructure is the work in [16]. In this manuscript, Saber et al. focus their efforts on large business IT organizations to propose a suitable metaheuristic for the evaluation of VM reassignments.

Another avenue of research considered the management of large scale Cloud services with the objective of maximizing either service performance, Cloud provider revenues, or both [28]. In a seminal work [29], Hagen and Kemper investigated the management of IT service and infrastructure changes to achieve SLOs and minimize costly business disruptions, focusing on a compelling real case study. Rekik et al. propose a linear programming model to provide an optimal business process model under Quality of Service (QoS) requirements in Cloud federations [30]. Other works investigated the offering of resources and services in the form of Virtual Data Centers (VDCs). One of the most interesting is VDC Planner, which implements a migration framework to simultaneously minimize VM migration costs and optimize the success rate of VDC mapping requests [3]. The minimization of migration costs is also discussed in a recent work focusing on network function virtualization in [31]. Other works considered service component placement optimization using many different objectives and/or criteria such as dimensioning [32], job completion time [33], VM consolidation [34] [35], and energy saving [36] [37]. More recently, sophisticated auto-scaling solutions emerged [38], [39]. To this end, there is an increasing interest in the adoption of container orchestration systems as Kubernetes and Docker Swarm [40], [41]. However, even if these solutions offer useful insights for orchestration and auto-scaling, they do not provide the support for deadline-based policies by themselves [42]–[44].

With regards to the optimization methodologies, a substantial part of the related literature focuses its efforts on the minimization of the VM allocation costs, such as the works in [11], [13], [14], [18], [45]. More specifically, in [14] Stefanello et al. address the problem of minimizing the VM placement problem across separated data centers by proposing a biased random-key genetic algorithm (GA). This work is then extended with the formulation of an improved linear mathematical model in [45]. Also dealing with geographically distributed data centers is the work in [15], in which the authors propose a mixed linear integer mathematical model in order to find a VM allocation that minimizes communication and bandwidth costs.

In [11], the authors address the service composition problem by proposing a solution to minimize and prevent SLA violations by means of a prediction and prevention monitoring system. Focusing instead on machine learning class-specific services, Zhang et al. propose a framework called MArK in [19]. In this scope, MArK aims at finding a minimum cost and SLO compliant VM allocation for machine learning applications running on AWS. In [13], the authors investigate the task planning problem in hybrid Cloud infrastructure and propose a mixed integer linear programming model to optimize total costs under deadline constraints. In this avenue of research, another interesting work focusing on workflow scheduling under deadline-constraint across multiple Clouds is [46]. More specifically, Guo et al. propose a strategy based on Particle Swarm Optimization (PSO) and GA operators to solve a cost minimization problem that considers both VM costs and inter/intra data centers transfer time.

From a pricing perspective, it is usual for Cloud Computing optimization solutions to deal with hourly-based prices. However, different pricing scheme may exist such as the one discussed by Dubois and Casale in [18]. In particular, the authors present a spot instance model in which a part of resources is granted to those making the highest bidder. To deal with this interesting peculiarity, this work proposes a novel heuristic called OptiSpot, which is capable of minimizing the spot price bid for VMs. In [47], Cardellini et al. propose a resource pricing and provisioning strategies for maximizing the revenues of Cloud providers. The problem of spot pricing model in Cloud Computing is also considered in [48]–[51].

Compared to those works, the present manuscript belongs to a complementary area of research, which adopts a service provider-centric perspective. More specifically, BDMaaS+ adopts a comprehensive service cost model that, beyond virtual resource acquisition, also considers SLO violations and risk related aspects. In addition, our solution adopts a realistic latency model (that we recently presented in [7]) that improves existing similar works in the literature, such as [52]–[54], by considering both Round-Trip Time (RTT) and Time-To-Live (TTL) parameters to obtain more realistic values.

III. The BDMaaS+ solution

BDMaaS+ currently focuses on Web Services (WSs) as the basic building blocks for the realization of complex IT services as workflows of WSs composed according to the WS Business Process Execution Language (WS-BPEL) standard. In other words, BDMaaS+ conceptually operates at the Platform-as-a-Service (PaaS) level with the main goal of finding the best

placement configuration of the WSs in the distributed Cloud environment.

A. Problem Statement

From a mathematical formulation perspective, BDMaaS+ attempts to solve the following optimization problem:

$$\begin{aligned} \arg\min_{x \in \mathbb{S}} \quad & BI(x) \\ \text{s.t.} \quad & r_m(VM_j) \geq r_m(C_k), \\ & \sum_{j=1}^{M} \sum_{k=1}^{T} \#vm_{i,j,k} \leq TNVM(DC_i) \end{aligned} \tag{1}$$

where BI is the Business Impact function, x represents the IT service configuration, i.e., an instantiation list of $\#vms_{ijk}$ VMs of type j run in data center DC_i hosting a service component of type C_k, and $\mathbb{S}$ is the space of all possible configurations.

BDMaaS+ adopters are expected to provide their own definition for the BI function. As mentioned in Section II-A, BDMaaS+ technically allows to adopt BI function that follow any of the IC, IC+SLO, and BD approaches. However, we expect that most users will want to adopt the BD approach, which - as we will demonstrate in Section VII-B - allows a more accurate and comprehensive evaluation of IT service configurations / VMs optimal placement. Finally, let us note that by purposely enlarging the perimeter of $\mathbb{S}$, BDMaaS+ users can seamlessly add what-if scenarios in the evaluation, as we will demonstrate later in Section VIII.

To ensure proper instantiation BDMaaS+ needs to consider only VMs with resources at least satisfying the minimal requirements for the software component considered and sufficient performance to handle typical component load. We express that as m-th resource required for component C_k does not exceed m-th resource for j-th VM: $r_m(VM_j) \geq r_m(C_k)$. In addition, since typically private Clouds have restricted resources, we need to consider the limit on the total number of possible VMs deployed in a data center i ($TNVM(DC_i)$). Finally, the user could insert additional constraints, e.g., limiting the deployment within a specific data center or forcing a given number of replicas for some service components.

B. Architecture

As regards the internal architecture of the BDMaaS+ framework (illustrated in Fig. 1), we organized it in several components addressing each needed management aspect. The API component, namely, BDMaaS+ REST-based APIs, allows service providers to interact with the BDMaaS+ engine and includes two subcomponents, namely, Configuration Management and Policy Management. These subcomponents allow service providers to enter a configuration of the Cloud computing environment (e.g., number of data centers, service model, etc.), to select the optimization policies to apply (e.g., business objectives, parameters for the optimization algorithm, etc.), and to express in a declarative way through the definition of high-level business goals the what-if objectives to explore.

Focusing on the BDMaaS+ engine, it consists of three main stages that work in a pipeline, namely: modeling, optimization, and decision making. At the modeling stage, Demand Model, Network Model, and Service Model are the three main components. They provide, respectively, the functions for: i) building the models of the service user service request arrival process (e.g., customers' locations, distributions of service request inter-arrival times, etc.); ii) gathering network measurements collected on-the-field by local measurement agents deployed at all private/public Cloud data centers to draw realistic network models; iii) and emulating IT service execution and deployment (e.g., service time distribution, service component placement, etc.). These three modules are fed by their respective monitoring agent twins on the leftmost part of Fig. 1 that integrate with existing Cloud platforms to gather monitoring information about the infrastructure (virtual resources) and applications component levels by updating the parameters of the service execution model as better detailed in the next section.

The optimization stage consists of the Optimization macro-component and represents the core part of BDMaaS+. It is in charge of reenacting the Cloud computing IT service and of evaluating possible alternative service placement configurations over the hybrid Cloud environment. First, the Service Placement Simulation component mimics possible service placements among those generated by the modeling stage, leveraging the `sisfc` simulator that we realized and made available to the community working in the field[3]. Then, the Business Impact Analysis component implements the performance analysis of the simulated configurations using the comprehensive business level evaluation techniques described in Section V. More specifically, the component assigns an overall cost (namely, business impact) to each of these possible configurations by exploring them according to the current modeled/monitored context.

At the third stage, the Decision Making component selects the best IT service placement configuration, namely, the one minimizing the business impact, according to the user preferences, current network conditions, and the output data provided by the Optimization component. Finally, BDMaaS+ was designed to be easily integrated with existing Cloud-based IT services through lightweight BDMaaS agents installed at each data center. Each agent includes three relatively simple and implementation-specific "connector" components: Demand Monitoring, Service Monitoring, and Actuator, depicted in light-blue in Fig. 1. Finally, the Actuator component is capable of automatically putting the new service configuration in place as required by the Decision Making component.

IV. IT Service and System Modeling

BDMaaS+ builds on top of modeling concepts specifically designed to accurately reenact the behavior of IT services in hybrid Cloud environments for what-if scenario analysis purposes. More specifically, those concepts enable BDMaaS+

[3] `sisfc` is a discrete event simulator that we specifically designed to reenact large scale Cloud services at the single service request granularity. `sisfc` is open source and can be downloaded at https://github.com/mtortonesi/sisfc.

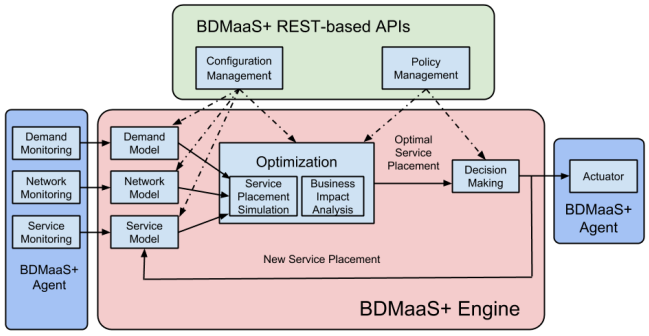


Fig. 1. The internal architecture of the BDMaaS+ support.

to measure how the whole service performance is affected by component reallocation to a different data center. We believe that this model allows to capture the behavior of a large part of real-life IT service architectures and to easily challenge them through the definition of high-level what-if assertions.

A. Service Request/Response and Execution Models

We consider an IT service as a collection of (distributed) service entry points. Each entry point represents a simple Web service available to the customer, implementing a business process according to the WS-BPEL workflow semantics [55].

Since in our model the basic unit of load on the data center is a request for a workflow, it is essential to accurately model the request generation process and the service component execution. To enable a request generation model that is capable of accurately reenacting dynamic request loads considering the different service customers, we separately model the requests arriving from each customer (or from different customer divisions in case of large customers with global presence).

We assume that service requests are routed through a number of software components, according to the workflow definition specified in BDMaaS+'s configuration file. Software components are entities that are instantiated in a VM. BDMaaS+ considers different sizes of VMs, with a corresponding amount of virtual (CPU, RAM, etc.) resources, and modulates the performance of a software component performance according to the size of the VM it is instantiated on.

In turn, service component execution is modeled using $G/G/s_i$ FCFS queues. BDMaaS+ allows to specify either parametric (e.g., Poisson, Gaussian, lognormal, etc.) or empirical (e.g., built from the analysis of service logs of real life software components) distributions for the modeling of each *(software component, VM size)* tuple, thus enabling the accurate modeling of a wide range of software components.

B. Latency Model

One of the critical aspects that any hybrid Cloud service placement technique should take into account is the impact of the underlying network infrastructure, interconnecting the relevant private and public Cloud data centers. In particular, given the importance of modeling interactions among service components geographically distributed on remote data centers, we decided to focus our attention on network latency as a significant component of service responsiveness.

In order to come up with a realistic model of inter-data center latency, we adopted a statistical estimation methodology based on real-life delay measurements [7]. More specifically, our model builds on a data set of round-trip time (RTT) values measured by an extensive "ping" campaign performed between each pair of Amazon Elastic Compute Cloud (EC2) data center locations [56], [57]. A thorough analysis of the data set revealed that, besides the RTT measured in the campaign, another relevant parameter reported by the ping application is the value of the time-to-live (TTL) field in the IP header of the ping reply. In fact, for a given data center pair, we found some kind of correlation between significant variations of the RTT and the corresponding reported TTL value, which could probably mean that different network paths were taken by packets during the measurement interval. Such a path variability can have a significant impact on latency, because most of the RTT is due to processing and queuing time at intermediate nodes.

According to the aforementioned considerations, we devised an approximation model of the inter-data center latency based on the following Gaussian mixture formula, which estimates the RTT probability density function:

$$f_{RTT}(t) = \sum_{i=1}^{n} a_i e^{-\frac{(t-b_i)^2}{2c_i^2}} \tag{2}$$

where t is the RTT expressed in milliseconds; $a_i = w_i/\sqrt{2\pi c_i^2}$ is the amplitude coefficient of the i-th Gaussian component, incorporating the weight w_i and variance c_i^2 coefficients; b_i is the mean (or shift) coefficient; c_i is the standard deviation coefficient.

Analyzing the collected data, we were able to identify the mixture model parameters with high accuracy (adjusted $R^2 \geq 0.945$) of all data center pairs. More specifically, we built accurate latency models for 35% of the data center pairs using common fitting tools and 4 Gaussian components. Raising the number of Gaussian components to 8 allowed us to capture an additional 15% data center pairs. For the remaining data center pairs, we had to consider 2-dimension models: computing separate Gaussian mixtures for different TTL values we applied a weighted sum based on the proportion of ping replies measured with a given TTL value, capturing another 38% of data center pairs. For the remaining 12% of data center pairs we devised and applied the *Relaxed Boxed Approximation* (RBA) algorithm [7], which corrects the tendency towards either overfitting or underfitting in the Gaussian mixture models produced by common fitting tools by introducing bounding box constraints for each of the model parameters, according to domain-specific heuristics, while allowing the fitting algorithm some room for local optimization.

The approximation model described above allows us to generate very realistic instances of inter-data center latency to be used in the simulation of the BDMaaS+ framework, by also enabling the definition of advanced what-if analysis.

C. SLA Model

BDMaaS+ adopts a simple but flexible SLA model. First, we define a SLO as a tuple of workflow ID, metric measure, metric function, and target objective:

$$SLO_i = (WF_i, \mu_i, f_i, \theta_i)$$

where the metric measure μ_i and target objective θ_i are either univariate or multivariate quantities. If the value $f_i(\mu_i)$ obtained for the metric does not fall within the target objective value θ_i, an SLO violation occurs. We then define an SLA SLA_C stipulated with a customer C as a set of *SLA components* SC_j, each one represented by an SLO, violation penalty, and time interval tuple:

$$SLA_C = \{SC_j | (SLO_j, penalty_j, time_j)\}.$$

BDMaaS+ allows a significant flexibility in the definition of SLOs, as well of custom metric measures and functions, of SLA components, and of SLAs through a dedicated domain specific language.

We believe that this model is generic enough to represent a wide range of SLAs used in real-life situations, as it enables to define SLO violation conditions that go beyond the trivial semantics of threshold comparison for metric and SLAs that consider multiple SLOs, even in non-trivial combinations.

V. Business Impact Analysis of IT Service Configurations

BDMaaS+ performs the business-driven evaluation of a given configuration x for an IT service through a model that considers 3 subcomponents: IT spending cost evaluation, SLA violation penalties estimation, and business (mis)alignment penalties [9]. Formally, it is:

$$BI(x) := BD(x) = IC(x) + SLO(x) + BAP(x). \quad (3)$$

As discussed in Section II, $IC(x)$ calculates the operational costs caused by running the system with configuration x and SLO is a function that calculates the costs caused by SLO violation penalties. Finally, $BAP(x)$ considers all costs caused by the adoption of a service configuration which is not aligned to the business objectives. The following subsections respectively detail how BDMaaS+ evaluates the IC, SLO, and BAP functions.

A. IT Related Cost Evaluation

In hybrid Cloud environments, we need to consider both the pay-per-use pricing offering typically proposed by public Cloud platforms as well as dedicated cost models that quantify IT related spending for virtual resource acquisition in private Cloud data centers. We then have:

$$IC(x) = PC(x) + PRC(x) \quad (4)$$

where and PC and PRC are functions that calculate the IT costs incurred by running the service with component configuration x for the public and private Cloud data centers respectively.

1) A Cost Model for Public Clouds: Since public Clouds operate strictly on the pay-per-use, utility computing paradigm, the calculations of costs that need to be sustained for running (a portion of the) IT service components in public Clouds is straightforward.

To calculate the costs for computational resource consumption, we consider the hours used by specific j-th type *vm*s in i-th datacenter ($vmhu(vm_{ij})$) metric as returned by the SISFC simulator, that tracks the number of hours consumed by the IT service components at each public Cloud data center, divided per VM type. We then sum the metric over the different data centers and VM types considered for the IT service deployment scenario, weighted for the corresponding VM type hourly cost ($vmhc(vm_{ij})$), to calculate the total cost for computational consumption on public Cloud:

$$PC(x) = \sum_{i=1}^{N} \sum_{j=1}^{M} vmhu(vm_{i,j}) * vmhc(vm_{i,j}) \quad (5)$$

An additional resource that has impact on costs is bandwidth consumption. However, given the rather convenient bandwidth pricing strategies adopted by most Cloud providers and the fact that we focus on IT services based on Web technologies that do not exhibit a significant bandwidth requirement, we decided to ignore bandwidth consumption related costs.

2) A Cost Model for Private Clouds (or Data Centers): Typical expense categories for creating and maintaining private Clouds include: facilities and related expenses (buildings, additional cooling hardware, networking cabling, etc.), hardware (computing, storage, networking, etc.), software (licensing of Virtualization Layer, OSes, server and application software), labor (maintenance of facilities, hardware, software, etc.), energy consumption (powering computing, storage and other devices, air conditioning, etc.), and Cloud Services (consumer's maintenance, internet provisioning, data transfers in and out, storage, etc.).

We summarize the costs mentioned above in notations of this section in the following: Base costs (BC) is a function of data center (DC_i), consumer (C_j)

$$BC = \sum BC(DC_i, C_j) \quad (6)$$

To simplify formula 6 based on typical cost model we assume that there is a fixed cost $FXC(DC_i)$ for usage of private data center DC_i, and cost associated with use of specific number of of VMs per usage cost similar to eq. 5.

Thus the cost of the private Cloud component configuration PRC is a sum of fixed costs FXC and variable costs VC:

$$PRC(x) = \sum_{i=1}^{N} [FXC(DC_i) + VC(\sum_{j=1}^{M} \sum_{k=1}^{T} vmhu_{i,j,k} * vmhc_{i,j,k})] \quad (7)$$

B. SLO Related Cost Evaluation

To calculate $SLO(x)$, BDMaaS+ evaluates the amount of SLO violation penalties that running the IT service in con-

figuration x would cause the service provider to incur. More specifically, BDMaaS+ reenacts the IT service with component configuration x and analyzes the corresponding simulation logs.

Referring to the SLA model discussed in Section IV-C, for each SLA component BDMaaS+ calculates the observed value for the metrics and time interval of relevance for SLO violation purposes, and then it confronts those values with target ones to evaluate whether SLO violations occurred.

C. Business Alignment Penalty Evaluation

When evaluating the performance at the business level of an IT service, BDMaaS+ also considers in addition to $IC(x)$ and $SLO(x)$ a "business (mis)alignment penalty" component $BAP(x)$ that calculates a wide range of costs related to personnel management, risk management, performance regressions, reconfigurations and intangibles for operating the IT service in configuration x with respect to the current configuration x_0. These effects are not fully captured by the previous 2 components alone, so we define a function to specifically address them. $BAP(x)$ represents a fundamental element of novelty in BDMaaS+ service evaluation with respect to other proposals.

The definition of $BAP(x)$ will typically be service specific, so any effort to present a generically applicable reference model for that component here would represent a futile exercise. However, we expect that adopters will commonly want to define a model of $BAP(x)$ that consider penalties for brittle configurations, e.g., with low redundancy and built on a barely adequate set of resource that does not provide any leeway to address workload spikes of unexpected faults. While apparently convenient from an IC approach perspectives, those configurations might indeed lead to poor results at the risk management and/or customer satisfaction levels.

Note that this model also allows to define virtual penalties that evaluate intangible business aspects which do not contribute to the monetary expenses that a service provider needs to pay for running an IT service, but that provide an indication – from a monetary perspective – of how much poor performance can affect the business.

VI. Optimization Algorithm

The optimization component represents a crucial part of BDMaaS+. In fact, let us point out that $\mathbb{S}$ in eq. 1 is a combinatorial space of size:

$$\prod_{i=1}^{\#DC} \sum_{n=0}^{TNVM(DC_i)} \binom{n + \left[\sum_{c \in C} a(c)\right] - 1}{n} \tag{8}$$

where $\#DC$ is the number of data centers, C is the set of software component types that we consider, and $a(c)$ is the number of VM types allowed for the instantiation of component c.

For practical applications, this space becomes so large that it cannot be explored exhaustively - especially since each evaluation of the objective function requires a relatively computationally expensive simulation. This rules out the adoption - even at the experimental evaluation level for benchmarking purposes - of a large part (if not all) of COTS solvers, as they would require too much time to explore the search space, and calls for heuristic approaches instead.

To address this issue, we specifically devised a *memetic algorithm* built on the combination of an outer search phase based on *Quantum-inspired Particle Swarm Optimization (QPSO)* [58], that takes care of assigning service components to Cloud data centers, and an inner search phased based on a purposely developed *simplified VM allocation algorithm*, that takes care of choosing which VM types should be used to host each of the service components. Memetic algorithms are efficient hybridizations of population-based optimization heuristics with refinement techniques such as smart local search algorithms [59]. Their compound structure allows to decompose the search procedure in a global (or explorative) and a local (or exploitative) part, and on the adoption of different strategies for each of them [60]. In our case, the decomposition is particularly convenient as it allows to treat separately the outer and inner search phases, which have fundamentally different characteristics and operate on significantly different search spaces.

Alg. 1 depicts the outer phase of the memetic algorithm, which is essentially an implementation of QPSO Type II in a search space that is a subset of $\mathbb{N}^{\#DC+\#C}$, where each coordinate represents a specific matrix of software component instantiations in the Cloud data centers. QPSO is a variant of Particle Swarm Optimization (PSO), a swarm intelligence technique inspired to the behaviour of bird flocks [58]. Traditional PSO is a relatively simple to implement optimization algorithm that, however, presents a few critical aspects, such as lower resilience to early convergence [58] and a more difficult parameter tuning process [61]. Improved versions of the algorithm such as QPSO and variants of PSO based on multiple swarms have later emerged to address these issues. In particular, we chose QPSO because it is particularly effective for dynamic optimization problems and also integrates rather well within a continuous optimization framework [62]. Finally, like PSO, QPSO is easily parallelizable, thus enabling to take full advantage of modern multicore CPUs.

QPSO is a remarkably simple but elegant algorithm. It keeps track of the best location visited by each particle (`p.bestpos` in Alg. 1) and by the entire swarm (`swarm_bestpos`), and for each particle it defines a stochastic attractor (`attr`) that lies on the hyperplane between `p.bestpos` and `swarm_bestpos`. It then defines a wave function around the stochastic attractor, modulating its size according to the current search performance, i.e., using the distance between the current particle position and the centroid between the best locations visited by each particle (`mean_bestpos`) as a weight. Finally, it calculates the next position of the particle as a (vectorial) sum of the stochastic attractor and a displacement sampled from the wave function.

Alg. 1 does not directly evaluate the BI function, but calls instead an inner search procedure (line 8). The latter is in charge of mapping service components to VM types, and implements a search over a combinatorial space of size:

$$\prod_{c=1}^{\#C} a(c)^{\sum_{d=1}^{\#DC} b(c,d)} \quad (9)$$

where $b(c,d)$ is the function returning the number of VMs to instantiate for a given component c and data center d couple.

To reduce the problem complexity, we adopt a working assumption that significantly simplifies the inner search procedure. More specifically, following the approach originally introduced in [63], we decided to enforce a (truncated) discrete exponential distribution for the set of VM types to instantiate for each component. This effectively transforms the combinatorial search into a significantly more convenient search within the continuous interval $(0, 1)$.

Alg. 2 depicts our implementation of the memetic algorithm inner phase, based on 2 procedures. The `inner_search` procedure implements a random local search, sampling the aggressiveness parameter β from the (0, 1) interval and passing it to the `allocate_vms` function. For every data center, the latter is in charge of selecting the VM types that satisfy the deployment constraints by analyzing the description of each service component (line 16) and of calculating the number of VMs of each type to allocate according to a discrete exponential distribution shaped by β (line 18).

Algorithm 1 Outer search of memetic algorithm

```
1: procedure OUTER_SEARCH(conf)
2:     particles ← random_swarm(conf)
3:     iteration ← 0; best ← null
4:     repeat
5:         iteration ← iteration + 1
6:         mean_bestpos ← (1/M) Σ_{i=1}^{M} particles[i].bestpos
7:         for all p in particles do
8:             val ← inner_search(p)
9:             if val > p.bestval then
10:                p.bestpos ← p.pos
11:                p.bestval ← val
12:            end if
13:            if val > swarm_bestval then
14:                swarm_bestpos ← p.pos
15:                swarm_bestval ← val
16:            end if
17:        end for
18:        φ⃗ ← rand(); u⃗ ← rand(); s ← rand()
19:        attr ← φ⃗ * p.bestpos + (1 − φ⃗) * swarm_bestpos
20:        δ ← α * |p.pos − mean_bestpos| * ln(1/u⃗)
21:        p.pos ← attr + sign(rand() − 0.5) * δ
22:    until iteration ≥ conf.max_iterations
23:    return swarm_bestpos, swarm_bestval
24: end procedure
```

Algorithm 2 Inner search of memetic algorithm

```
1: procedure INNER_SEARCH(p)
2:     attempt ← 0; p.value ← −∞
3:     repeat
4:         attempt ← attempt + 1; q ← p
5:         q.vms ← allocate_vms(q.pos, rand())
6:         q.value ← evaluate(q)
7:         if q.value > p.value then
8:             p.vms ← q.vms
9:         end if
10:    until attempt ≥ max_attempts
11: end procedure
12:
13: function ALLOCATE_VMS(comp_per_dc_matrix, β)
14:    vms ← []
15:    for all c, dc in comp_per_dc_matrix do
16:        v ← []; avts ← allowed_vm_types(c, dc)
17:        for i ← 1, ..., size(avts) do
18:            v ← [v, (c * β^{i−1}, avts_i, dc)]
19:        end for
20:        vms ← [vms, normalize(v)]
21:    end for
22:    return vms
23: end function
```

VII. Reference Use Case and Baseline Validation

To evaluate our techniques and tools, we devised a realistic case study capturing the behavior of an enterprise-class IT service deployed on a large scale for customers with global presence. More specifically, we considered a Money Management and Transfer System IT service (MMTS in the following) that allows users to manage their bank accounts and to submit money transfer requests. We believe MMTS represents an interesting case study that raises non-trivial challenges from the optimal software component placement perspective. In fact, this type of IT service often leverages legacy software components that cannot be easily migrated and other software components that implement sensitive functions and whose deployment thus has to withstand security constraints; in both cases, they must be deployed in a local private Cloud.

A. Description of Case Study

MMTS consists of 3 applications, which we label *A*, *B*, and *C*. A is an ASP.NET application running on Microsoft Windows, B is a Web application based on the LAR (Linux, Apache, Ruby on Rails) stack, and C is a LEMP (Linux, Nginx, MySQL, PHP) application. The architecture of the applications, depicted in Fig. 2 along to the 12 different workflows implemented by MMTS, mostly follows the classic 3-tier paradigm, with a Web Server, an Application Server and a DataBase Management System (DBMS). However, the 3-tier paradigm is extended by some components, such as the Financial Transaction System and the Queue Manager that represents the interface to a reporting system based, e.g., on PDF document generation and submission through an e-mail Server. Let us briefly note that the reporting function represents an external support system for MMTS, and thus we do not consider it in the software component placement. In addition, applications B and C share the same replicated DataBase: a MySQL DBMS configured with master/slave replication. A full description of the workflows is provided in Table I.

For MMTS deployment, we consider a federation of 8 different Cloud facility locations. More specifically, we con-

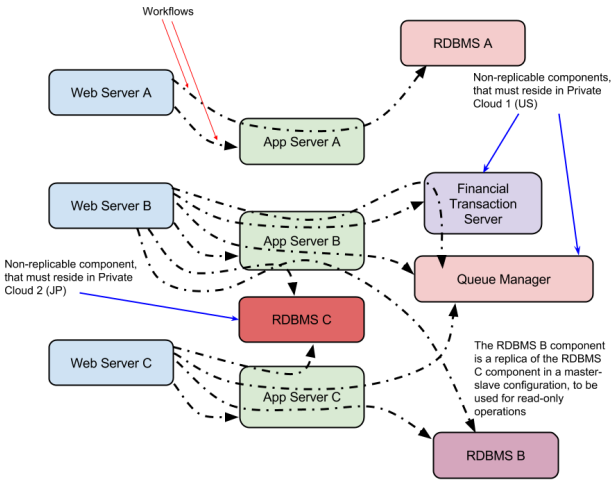


Fig. 2. Architecture of distributed, multi-tier service case study.

sider 6 public Cloud data centers, namely Amazon EC2's *us-west-1*, *us-east-1*, *eu-west-1*, *ap-southeast-2*, *sa-east-1*, and *ap-southeast-1* data centers, and 2 private Cloud data centers, respectively located in northwestern USA and in Japan.

We also consider a few deployment constraints. The MySQL master component, identified as RDBMS C in Fig. 2, must reside in private Cloud data center 1 (US) for security reasons and cannot be migrated to any other data center. The Financial Transaction System component is implemented by a legacy system, residing in private Cloud data center 2, which cannot be migrated to another data center. All the other software components can be allocated to any data center.

We consider a customer with global presence, with one division in each of the following locations: East Coast USA, West Coast USA, South America, Asia, Europe. The divisions are of varying sizes and account for a different share of requests: 11.1%, 16.6%, 27.8%, 33.3% and 11.1% respectively.

We believe that this use case is representative of real-life situations in which a global scale service provider offers IT services to many enterprise level customers. In fact, this use case allows considering interesting possibilities such as a service provider offering MMTS to its end customers might want to explore alternative configurations through what-if scenario analysis. For instance, it might want to understand what would happen if the customer moved one (or a part) of its divisions somewhere else. Or, it might want to understand if stipulating a different SLA with its customers could lead to significant savings that enable to propose particularly convenient pricing for its IT service, or which kind of pricing to offer in case a customer asked for a new SLA. Our contribution is a tool that enables service providers to simulate different configurations within what-if scenarios to identify the most convenient one from the business perspective.

B. Baseline Experiment and Approach Validation

To validate the tool and the business-driven optimization approach it enables, we used BDMaaS+ to reenact the realistic case study introduced in Section VII and optimize it according to different approaches identified in Section II-A: BD, IC, and IC+SLO. We believe that these two categories are representative of the majority of related work on Cloud Computing optimization/optimal VMs placement.

In all the experiments we assume that the 2 private Cloud data centers have the same location as Amazon EC2's *us-west-2* and *ap-northeast-1* data centers. The latency between their different locations is modeled according to the Gaussian mixture approximation presented in Section IV-B. We also assume that the latency for message transfers within a single location is significantly smaller than the inter-data center latency, and can thus be safely ignored.

We assume that the aggregated flow of request has a constant intensity - and whose interarrival times can be modeled with a Pareto distribution with location 1.2E-4 and shape 5, corresponding to 6,666.66 requests per second. The requests emanating from each division will be automatically forwarded to the closest Cloud data center. (This is a common practice, as for instance Amazon allows to do with its Route 53 system.)

We configured BDMaaS+ to reenact the MMTS IT service in different configurations for 60 seconds of simulated time, plus 10 seconds of simulation warmup time, roughly corresponding to the processing of 400,000 service requests, and evaluate the performance of each configuration.

Following the approach discussed in Section V-C, and assuming the MMTS business management would want to run a system capable of withstanding impromptu workload spikes, we defined a BAP component that penalizes brittle IT service configurations that lead to unsatisfied requests:

$$BAP(x) = \begin{cases} max_p * \frac{2}{\pi} * arctg(k * m_x) & \text{if } m_x > 0 \\ 0 & \text{otherwise} \end{cases}$$

where:

$$m_x = \frac{expected\_requests - served\_requests}{expected\_requests}$$

is the share of requests that were not fully served in the simulation-based evaluation of the IT service with configuration x. More specifically, we set $max_p = 100,000$ USD/day and $k = 200$, as we experimentally verified these parameter values to be well suited for the purposes of optimizing the MMTS service.

Finally, we configured BDMaaS+'s memetic optimization algorithm [8] to use a 40-particle swarm and a contraction-expansion coefficient $\alpha = 0.75$ in the outer QPSO algorithm and 10 attempts for the inner search. We experimentally verified these parameters to be well suited to optimization problems of this size and complexity.

For this *baseline* and validation experiment, we adopted a simple SLO penalty model based on considering a predefined penalty for each *(customer division, workflow)* couple in case the corresponding measured Mean Time To Resolution (MTTR) for service requests exceeded a predefined threshold. The penalty amounts we used are identical for each customer division, and are shown in Table II.

Firstly, let us present in Table III the results achieved using the three optimization approaches. More specifically, Table III shows the business impact evaluation (in $/day) for the best IT service configuration x^* found by BDMaaS+ using the IC,

TABLE I
WORKFLOWS IMPLEMENTED BY THE MMTS IT SERVICE.

Reqs %	Name	Description	Component sequence
15 %	WF01	Home page	Web Server A - App Server A
20 %	WF02	Login	Web Server A - App Server A - RDBMS A
15 %	WF03	Financial services page	Web Server B - App Server B
3 %	WF04	Request loan	Web Server B - App Server B - RDBMS C
3 %	WF05	Money transfer	Web Server B - App Server B - Financial Transaction Server
6 %	WF06	Reporting for transfers	Web Server B - App Server B - Queue Manager
3 %	WF07	Delayed money transfer	Web Server B - App Server B - Financial Transaction Server - Queue Manager
5 %	WF08	Last month money transfers	Web Server B - App Server B - RDBMS B
15 %	WF09	Account management page	Web Server C - App Server C
6 %	WF10	Last month statement	Web Server C - App Server C - RDBMS B
3 %	WF11	Update credentials	Web Server C - App Server C - RDBMS C
6 %	WF12	Reporting for account	Web Server C - App Server C - Queue Manager

TABLE II
SLO PENALTY MODEL USED IN THE BASELINE EXPERIMENT.

Workflows	Trigger condition	Penalty amount
WF01-02	$MTTR > 200ms$	200 $/day
WF03	$MTTR > 200ms$	300 $/day
WF04	$MTTR > 220ms$	300 $/day
WF05	$MTTR > 300ms$	300 $/day
WF06	$MTTR > 350ms$	300 $/day
WF07	$MTTR > 180ms$	400 $/day
WF08	$MTTR > 210ms$	400 $/day
WF09	$MTTR > 330ms$	400 $/day
WF10-12	$MTTR > 350ms$	400 $/day

TABLE III
BUSINESS IMPACT EVALUATION (IN $/DAY) FOR BEST IT SERVICE CONFIGURATIONS FOR THE 3 APPROACHES IN BASELINE EXPERIMENT.

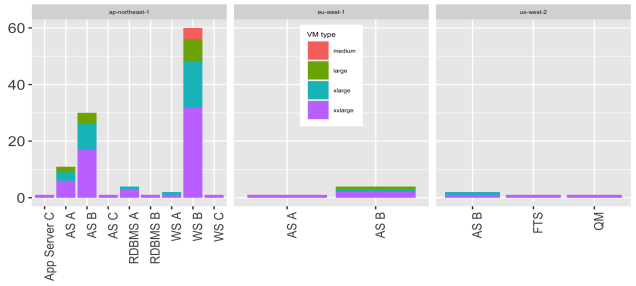
Approach	$BI(x^*)$	$IC(x^*)$	$SLO(x^*)$	$BAP(x^*)$
IC	27,572	27,572	15,600	38,527
IC+SLO	38,598	26,798	11,800	46,364
BD	70,478	27,005	11,500	31,973

IC+SLO, and BD approaches. These approaches respectively define $BI(x^*)$ as the amount of IT costs $IC(x^*)$, as the sum of IT costs and SLO violation penalties ($IC(x^*)+SLO(x^*)$), and as the sum of IT costs, SLO violation penalties and business alignment penalties ($IC(x^*)+SLO(x^*)+BAP(x^*)$).

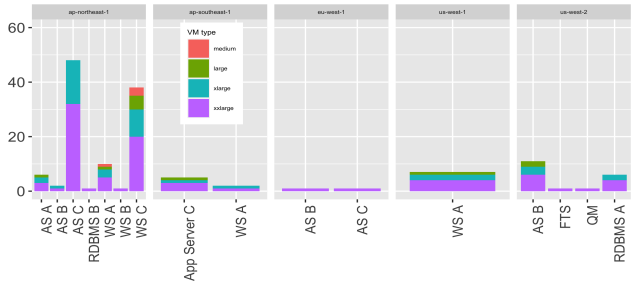
Let us note how, while producing a $BI(x^*)$ which is apparently higher because it considers all the 3 IC, SLO, and BAP components in the business impact evaluation, BD actually outperforms the other approaches and is in fact capable of finding a solution with a lower $IC(x^*)+SLO(x^*)$ value (38,505 $/day instead of 43,127 and 38,598 $/day respectively for the IC and IC+SLO approaches).

The BD approach allowed to reach this goal by enabling BDMaaS+ to explore IT service configurations that were more aggressive in consolidating VMs in a smaller set of data centers. This is immediately apparent when examining the best IT service configurations generated by the three optimization approaches illustrated in Fig. 3, which depicts the size, type, and number of VMs distributed among the data centers.

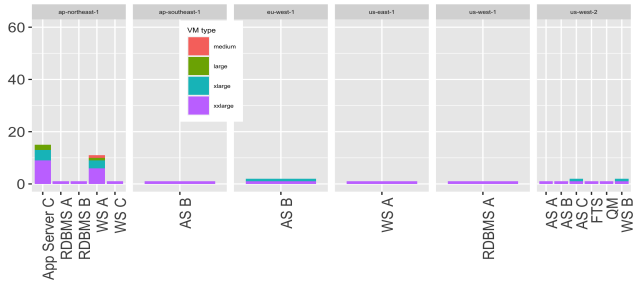
To further understand the performance differences between the approaches, let us examine Fig. 4a, 4b, and 4c, which depict the number of closed requests for the BD, IC, and IC+SLO approaches during the iterations of the memetic



(a) BD



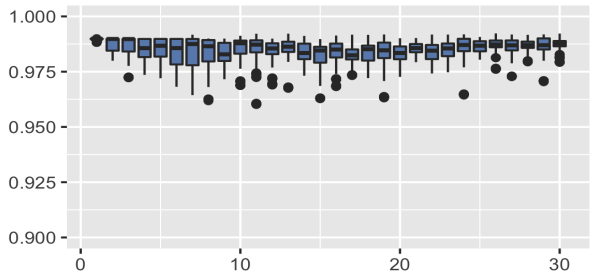
(b) IC



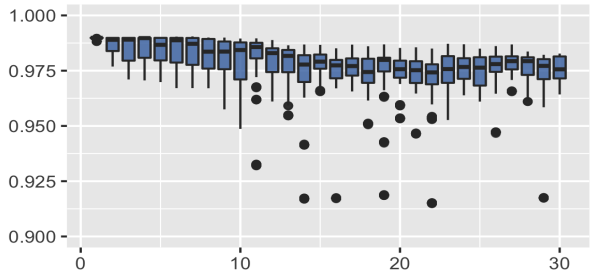
(c) IC + SLO

Fig. 3. Distribution of VMs (type, size, and numbers) among data centers found by the optimization algorithm for the three optimization approaches.

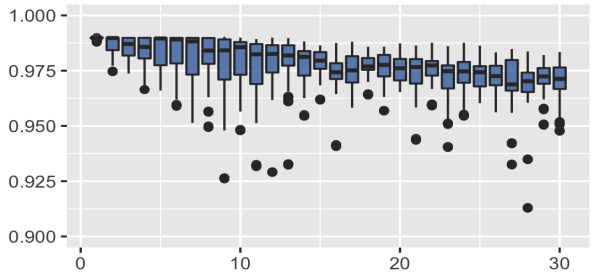
algorithm (BDMaaS+ considers a request closed if the IT system provides a response to the requester). The figure shows how BD finds IT service configurations capable of serving a higher number of requests when compared to IC and IC+SLO. This is mainly due to the BD formulation which, by explicitly considering BAP in the optimization criteria, effectively prevents the optimizer from aggressively looking for configurations with low IT costs but poor performance



(a) BD



(b) IC



(c) IC + SLO

Fig. 4. Distribution of the percentage of the closed requests during the iterations of the memetic algorithm.

and scalability.

Finally, Fig. 5 depicts the distribution of the values of the BAP component observed during the iterations of the optimization algorithms for the 3 approaches. The figure shows that, by not explicitly considering the BAP component in the optimization criteria, the IC and IC+SLO approaches steer towards a portion of the search space which corresponds to configurations with high BAP values and thus suboptimal.

We believe that all these results confirm that only a more comprehensive evaluation, such as that enabled by BD approach, is capable of i) minimizing total costs and ii) maximizing the service performance.

VIII. Experimental Results

Let us now demonstrate the effectiveness of our techniques and tools for what-if scenario analysis purposes. We will roleplay as the MMTS business management and use BDMaaS+ to formulate hypothetical scenarios and evaluate from the monetary perspective what the optimal IT service configurations in those conditions would be. First, let us introduce the Total Cost metric: $TC(x) = IC(x) + SLO(x)$. While we will keep instructing BDMaaS+ to adopt the BD approach in the experiments, we will consistently use $TC(x)$ (instead of $BI(x) = IC(x) + SLO(x) + BAP(x)$) as a reference metric in the presentation of results as it is more relevant for what-if scenario analysis purposes.

A. Validation of simplified latency model

To support the definition of deployment scenarios in BDMaaS+ that consider data center locations for which no RTT information is available and hence no accurate latency model can be built, we devised a simplified version of the latency model in Section IV-B. More specifically, starting from the model based on real locations we used in the baseline experiment, we identified 3 levels of distance between data centers from the network latency perspective: *near (N)*, *intermediate (I)*, and *far (F)*. Then we modeled the respective latency according to the Gaussian mixture obtained for the two Amazon EC2 data center pairs with the smallest (N case) and largest (F case) average latency, and then chose a third pair with an intermediate value between them (I case). The inner portion of Table IV, i.e., excluding the cells with a gray background, shows the distances between the data centers considered in this scenario according to the simplified model.

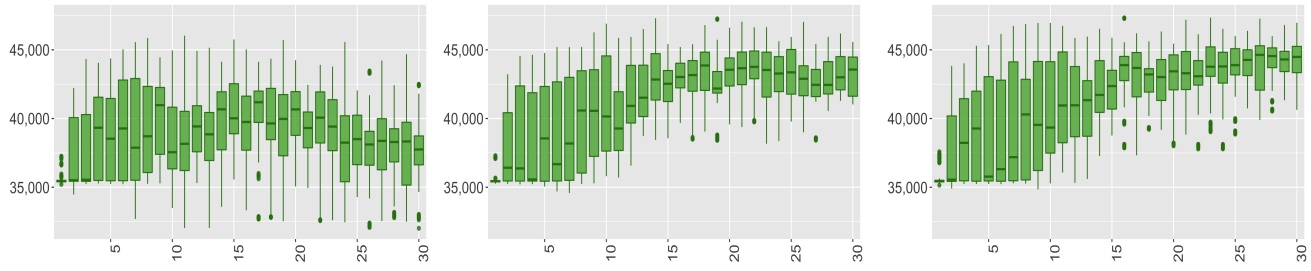
To validate this simplified latency model, we ran an experiment in which we considered the same 5 customer division and 8 data center locations of the baseline experiment but used the simplified latency model instead of the full one. The total costs for running the MMTS IT service in this case study are presented in Fig. 6. As one can see, the optimal configuration for the IT service corresponds to a total cost of 39,696 $/day. This means that the adoption of the simplified latency model in place of the accurate one introduces an 8.3% inaccuracy, a small but significant difference that demonstrates the importance of using a latency model that is as realistic as possible for the evaluation of IT services in hybrid Cloud scenarios.

TABLE IV
Simplified latency model for Cloud data centers based on the 3 identified distance levels (N: near, I: intermediate, F: far).

	us-west-2	us-east-1	ap-northeast-1	eu-west-1	ap-southeast-2	sa-east-1	ap-southeast-1	ap-south-1
us-west-1	N	N	I	I	F	F	F	F
us-west-2		N	I	I	F	F	F	F
us-east-1			F	I	F	I	F	F
ap-northeast-1				F	I	F	N	I
eu-west-1					F	F	F	F
ap-southeast-2						F	F	I
sa-east-1							F	F
ap-southeast-1								N

B. Addition of a customer and data center location

We then ran a third experiment in which we consider the "what-if" scenario in which 1/3 of the requests from the Asia division of the customer located in Tokyo (next to private Cloud data center 2) moved to Mumbai, India. To support the changed needs of the customer's Asia division, we extend the



(a) BD (b) IC (c) IC+SLO

Fig. 5. Distribution of BAP(x) (in $/day) using the different approaches during the iteration of the optimization algorithm.

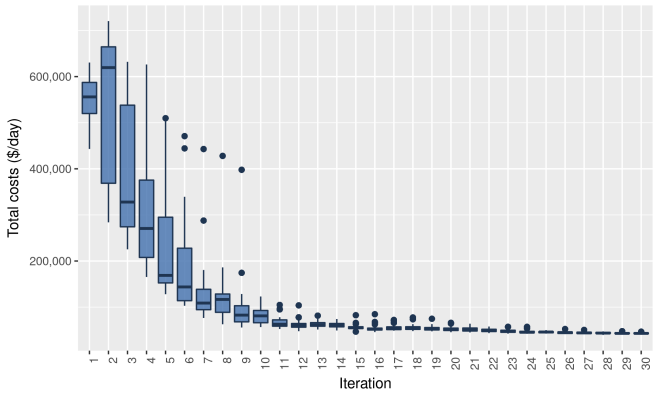


Fig. 6. Distribution of total costs for the evaluated configurations of MMTS at each iteration of the optimization algorithm in the simplified latency model validation experiment.

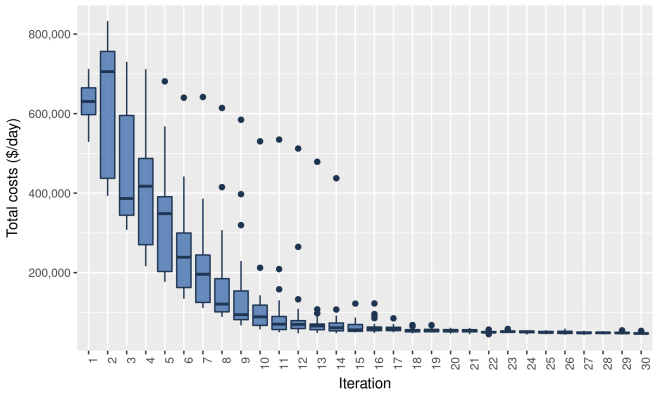


Fig. 7. Distribution of total costs for the evaluated configurations of MMTS at each iteration of the optimization algorithm in the 9 data centers and 6 customer divisions experiment.

deployment scenario considered in the previous experiment to include Amazon EC2's *ap-south-1* data center.

Since we have no RTT data available for *ap-southeast-1*, we extrapolate an extended version of the simplified model that also considers the portion of Table IV with gray background (bottom line and rightmost column).

The total costs for running the MMTS IT service in this case study are presented in Fig. 7. As one can see, BDMaaS+ is capable of optimizing the MMTS IT service for this extended scenario as well, reducing the costs to 43,378 $/day. This value is 9.3% higher than the one obtained from the previous experiment. This means that switching to a configuration of the MMTS IT service that considers a customer division in India would be expensive for the service provider, which should consider contracting a higher service price with its customer.

C. Modified SLA model validation

As a second example of "what-if" experiment, we consider the adoption of a different SLO penalty model, using the accurate latency model as in the baseline experiment. For this experiment, we defined a SLA in which the fundamental metric to consider for SLO violations is not the observed mean (MTTR) but the observed *99-percentile (99p)* of request processing times. To this end, we adopted the SLO objectives and penalties defined in Table V, also applied in an identical fashion to each customer division as the previously adopted SLO penalty model.

TABLE V
SLO PENALTY MODEL USED IN THE MODIFIED SLA EXPERIMENT.

Workflows	Trigger condition	Penalty amount
WF01-02	$99p > 500ms$	300 $/day
WF03-04	$99p > 500ms$	500 $/day
WF05-06	$99p > 750ms$	500 $/day
WF07-08	$99p > 500ms$	800 $/day
WF09-12	$99p > 750ms$	800 $/day

TABLE VI
EVALUATION (IN $/DAY) OF BEST IT SERVICE CONFIGURATIONS FOR THE 3 APPROACHES IN MODIFIED SLA MODEL EXPERIMENT.

Approach	$TC(x^*)$	$IC(x^*)$	$SLO(x^*)$	$BAP(x^*)$
IC	30,074	27,174	2,900	38,367
IC + SLO	30,108	27,208	2,900	40,724
BD	30,046	27,446	2,600	36,837

We use the modified SLA model to provide a final validation of the BD approach, by comparing it with the IC and IC+SLO approaches. As a result of the optimization process, Table VI shows total costs, IT costs, SLO violation penalties, and business alignment penalties obtained using the different approaches. As before, these results show that all three approaches can find minimal solutions from the total cost perspective. Moreover, as for the previous validation reported in Section VII-B, the gap between those solutions is still negligible. This gives another proof of their capabilities in reducing monetary costs.

However, let us note that if the optimization approach does

not take into account the performance of the system, e.g number of closed requests, the optimized IT service configuration may result in poor service performance. In fact, it is possible that such configurations would be of minimum cost and not incurring in higher SLO violation penalties because they satisfy the defined SLAs for a reduced number of requests.

To support this claim, Fig. 8 illustrates the distribution of the percentage of closed requests for the BD, IC, and IC+SLO approaches during the iterations of the memetic algorithm. It is clear that the former exhibit a better trend compared to the others. Therefore, even in this modified experiment, these results confirm that BD is still capable of providing optimized performance and minimal cost allocation.

All results confirm the validity and the effectiveness of the BD approach in minimizing total costs when compared to the IC and IC+SLO approaches. Furthermore, this final validation demonstrates that even in a challenging scenario subjected to a binary SLA agreement the BD approach is the only capable to find the optimal IT service configuration both from a cost and from a performance perspective.

D. Final remarks

In light of the experiments presented above, we can formulate a few interesting conclusions. First, the memetic optimization algorithm adopted by BDMaaS+ consistently exhibited a very good performance in terms of convergence speed, roughly reaching the optimum after just 11 iterations. We speculate that this is due to the adoption of the QPSO algorithm, which significantly outperforms the solution based on genetic algorithms that we had adopted in earlier versions of BDMaaS+ [9].

Secondly, the experiments demonstrate the importance of evaluating an IT service using an accurate latency model as opposed to an approximated one. However, the adoption of a simplified latency model represents a quick way to extend a known deployment scenario to consider other data center / customer locations for which no RTT data is available, and to draw a first round of important lessons before investing resources to prepare a more realistic latency model.

In any case, the results obtained with a simplified latency model should be confirmed by using an accurate latency model before putting in practice any corrective action on real world systems.

In addition, the significant cost differences obtained in the first and last experiments seem to indicate that there is a large space for exploration for SLA definition with customers. What-if solutions such as BDMaaS+ can be instrumental in finding out more convenient configurations from the service provider perspective, that can allow to formulate particularly convenient pricing for end customers.

Finally, a consideration about the execution times of BDMaaS+. Each of the experiments presented above took roughly one day to run in a workstation equipped with an 8-core Intel i7-3770 CPU and 16 GB RAM and running Arch Linux (kernel version 4.18.5), Java 10.0.2 and JRuby 9.2. In each experiment, 9600 different configurations for the IT services and 3.8 billion service requests were evaluated.

Given the consistent fast convergence exhibited by BDMaaS+'s memetic algorithm and its capability to run unmodified with a much larger parallelism, we speculate that a more recent computer (or VM) equipped with a 24-core or 32-core CPU would be fully capable to run those results in a few hours - thus enabling the continuous re-evaluation of complex IT services overnight. Let us also note that the simultaneous use of multiple computers to further increase the parallelism in "what-if" experiments would require only minor changes to the current version of BDMaaS+ and could significantly bring down the execution times - to the order of magnitude of one hour.

IX. Conclusions and Future Works

The performance optimization of large IT service deployments over hybrid Cloud environments can be extremely complex and calls for what-if-analysis-ready Cloud management tools. This paper presented BDMaaS+, a robust and comprehensive business-driven evaluation solution for service providers able to reproduce the behavior of real-life IT support organizations with a very high degree of accuracy. The experimental evaluation we conducted considering a realistic use case and several what-if scenarios demonstrates the BDMaaS+ effectiveness in exploring alternative IT service configurations that maximize service provider overall revenues. Then, we also demonstrated the efficacy of the BDMaaS+'s BD approach by comparing it with other optimization philosophies presented in the related literature, which mainly focus on IT cost minimization. The comparison proved that the BD approach outperforms the other approaches as it enables to identify more convenient and robust IT service configurations.

Encouraged by these results, we are now working on various future research directions. On the one hand, we are further refining the core modeling parts to obtain accurate evaluations integrating BDMaaS+ within state-of-the-art container orchestration technologies, to ease the deployment of workflow across different Cloud platforms; further evolving our metaheuristics to be able to exploit different types of computing instances, including also on-demand and spot instances; implementing a permanent observatory for inter-/intra-datacenter network delays for all main public Cloud providers.

References

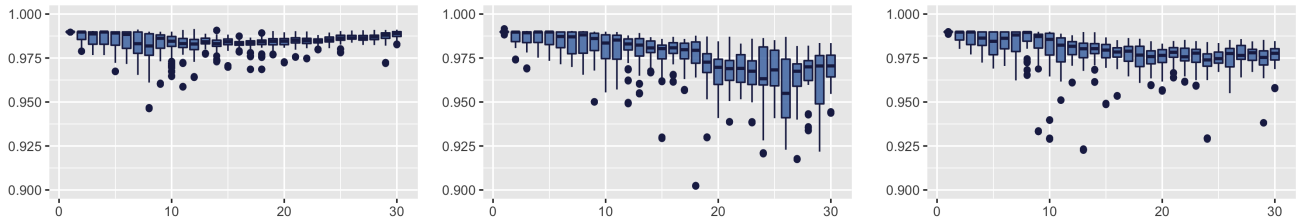
[1] K. Bai, N. Ge, H. Jamjoom, E. Ea-Jan, L. Renganarayana, and X. Zhang, "What to Discover Before Migrating to the Cloud," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 320–327.

[2] A. Keller, "Challenges and directions in service management automation," *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 884–901, Oct 2017. [Online]. Available: https://doi.org/10.1007/s10922-017-9437-9

[3] M. F. Zhani, Q. Zhang, G. Simona, and R. Boutaba, "VDC Planner: Dynamic migration-aware Virtual Data Center embedding for Clouds," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pp. 18–25.

[4] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic Service Placement in Geographically Distributed Clouds," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 99, p. pp, 2013.

[5] W. Li, P. Svärd, J. Tordsson, and E. Elmroth, "Cost-optimal cloud service placement under dynamic pricing schemes," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, Dec 2013, pp. 187–194.



(a) BD (b) IC (c) IC+SLO

Fig. 8. Distribution of percentage of closed requests for the configurations explored by the memetic algorithm for the 3 optimization approaches.

[6] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[7] W. Cerroni, L. Foschini, G. Y. Grabarnik, L. Shwartz, and M. Tortonesi, "Estimating delay times between cloud datacenters: A pragmatic modeling approach," *IEEE Communications Letters*, vol. 22, no. 3, pp. 526–529, March 2018.

[8] W. Cerroni, L. Foschini, et al. M. Tortonesi, "Service Placement for Hybrid Clouds Environments based on Realistic Network Measurements," in *2018 International Conference on Network and Service Management (CNSM 2018) - miniconference track*, November 2018.

[9] M. Tortonesi and L. Foschini, "Business-driven service placement for highly dynamic and distributed cloud systems," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 977–990, 2018.

[10] W. Cerroni, L. Foschini, G. Y. Grabarnik, L. Shwartz, and M. Tortonesi, "What-if scenario analysis for it services in hybrid cloud environments with BDMaaS+," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 62–70.

[11] P. Leitner, W. Hummer, and S. Dustdar, "Cost-based optimization of service compositions," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 239–251, April 2013.

[12] S. jun Xue and W. Wu, "Scheduling workflow in cloud computing based on hybrid particle swarm algorithm," 2012.

[13] M. Malawski, K. Figiela, and J. Nabrzyski, "Cost minimization for computational applications on hybrid cloud infrastructures," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1786 – 1794, 2013.

[14] F. Stefanello, "A new linear model for placement of virtual machines across geo-separated data centers," 2015.

[15] M.-H. Lin, J.-F. Tsai, Y.-C. Hu, and T.-H. Su, "Optimal allocation of virtual machines in cloud computing," *Symmetry*, vol. 10, no. 12, 2018.

[16] T. Saber, J. Thorburn, L. Murphy, and A. Ventresque, "Vm reassignment in hybrid clouds for large decentralised companies: A multi-objective challenge," *Future Generation Computer Systems*, vol. 79, pp. 751 – 764, 2018.

[17] A. A. Youssef and D. Krishnamurthy, "Burstiness-aware service level planning for enterprise application clouds," *Journal of Cloud Computing*, vol. 6, no. 1, p. 17, Aug 2017.

[18] D. J. Dubois and G. Casale, "Optispot: minimizing application deployment cost using spot cloud resources," *Cluster Computing*, vol. 19, no. 2, pp. 893–909, Jun 2016.

[19] C. Zhang, M. Yu, W. Wang, and F. Yan, "Mark: Exploiting cloud services for cost-effective, slo-aware machine learning inference serving," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. Renton, WA: USENIX Association, Jul. 2019, pp. 1049–1062. [Online]. Available: https://www.usenix.org/conference/atc19/presentation/zhang-chengliang

[20] R. A. K. Mehdi and M. Nachouki, "Cloud capacity planning based on simulation and genetic algorithms," in *Intelligent Systems and Applications*, Y. Bi, R. Bhatia, and S. Kapoor, Eds. Cham: Springer International Publishing, 2020, pp. 160–174.

[21] A. Kulkarni and B. Annappa, "Context aware vm placement optimization technique for heterogeneous iaas cloud," *IEEE Access*, vol. 7, pp. 89 702–89 713, 2019, cited By 1.

[22] A. Moura, J. Sauve, and C. Bartolini, "Business-driven IT management - upping the ante of IT : exploring the linkage between IT and business to improve both IT and business results," *Communications Magazine, IEEE*, vol. 46, no. 10, pp. 148–153, October 2008.

[23] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo, "Dynamic resource management using virtual machine migrations," *IEEE Communications Magazine*, vol. 50, no. 9, 2012.

[24] A. Thakur and M. S. Goraya, "A taxonomic survey on load balancing in cloud," *Journal of Network and Computer Applications*, vol. 98, pp. 43 – 57, 2017.

[25] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 53.

[26] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[27] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware vm placement for cloud systems," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*. IEEE, 2012, pp. 498–506.

[28] A. Hammoud, A. Mourad, H. Otrok, O. A. Wahab, and H. Harmanani, "Cloud federation formation using genetic and evolutionary game theoretical models," *Future Generation Computer Systems*, vol. 104, pp. 92–104, 2020.

[29] S. Hagen and A. Kemper, "Facing the unpredictable: Automated adaption of it change plans for unpredictable management domains," in *2010 International Conference on Network and Service Management*, Oct 2010, pp. 33–40.

[30] M. Rekik, K. Boukadi, N. Assy, W. Gaaloul, and H. Ben-Abdallah, "Optimal deployment of configurable business processes in cloud federations," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1692–1705, Dec 2018.

[31] H. Yu, J. Yang, and C. Fung, "Fine-grained cloud resource provisioning for virtual network function," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1363–1376, 2020.

[32] K. Salah, K. Elbadawi, and R. Boutaba, "An analytical model for estimating cloud resources of elastic services," *Journal of Network and Systems Management*, vol. 24, no. 2, pp. 285–308, Apr 2016.

[33] A. Ponraj, "Optimistic virtual machine placement in cloud data centers using queuing approach," *Future Generation Computer Systems*, vol. 93, pp. 338 – 344, 2019.

[34] A. Fatima, N. Javaid, T. Sultana, W. Hussain, M. Bilal, S. Shabbir, Y. Asim, M. Akbar, and M. Ilahi, "Virtual machine placement via bin packing in cloud data centers," *Electronics*, vol. 7, no. 12, 2018. [Online]. Available: http://www.mdpi.com/2079-9292/7/12/389

[35] T. Mahdhi and H. Mezni, "A prediction-Based VM consolidation approach in IaaS Cloud Data Centers," *Journal of Systems and Software*, vol. 146, pp. 263 – 285, 2018.

[36] M. A. Kaaouache and S. Bouamama, "An energy-efficient vm placement method for cloud data centers using a hybrid genetic algorithm," *Journal of Systems and Information Technology*, vol. 20, no. 4, pp. 430–445, 2018. [Online]. Available: https://doi.org/10.1108/JSIT-10-2017-0089

[37] F. Alharbi, Y.-C. Tian, M. Tang, W.-Z. Zhang, C. Peng, and M. Fei, "An ant colony system for energy-efficient dynamic virtual machine placement in data centers," *Expert Systems with Applications*, vol. 120, pp. 228 – 238, 2019.

[38] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling web applications in clouds: A taxonomy and survey," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 73:1–73:33, Jul. 2018. [Online]. Available: http://doi.acm.org/10.1145/3148149

[39] K. Salah, P. Calyam, and R. Boutaba, "Analytical model for elastic scaling of cloud-based firewalls," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 136–146, March 2017.

[40] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," 2016.
[41] D. Inc., "Docker Swarm," http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/, [Online; accessed 29-November-2019].
[42] T. Kiss, J. DesLauriers, G. Gesmier, G. Terstyanszky, G. Pierantoni, O. A. Oun, S. J. Taylor, A. Anagnostou, and J. Kovacs, "A cloud-agnostic queuing system to support the implementation of deadline-based application execution policies," *Future Generation Computer Systems*, vol. 101, pp. 99 – 111, 2019.
[43] S. Qi, S. G. Kulkarni, and K. K. Ramakrishnan, "Assessing container network interface plugins: Functionality, performance, and scalability," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 656–671, 2021.
[44] L. Toka, G. Dobreff, B. Fodor, and B. Sonkoly, "Machine learning-based scaling management for kubernetes edge clusters," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 958–972, 2021.
[45] F. Stefanello, V. Aggarwal, L. S. Buriol, J. F. Gonçalves, and M. G. Resende, "A biased random-key genetic algorithm for placement of virtual machines across geo-separated data centers," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 919–926.
[46] W. Guo, B. Lin, G. Chen, Y. Chen, and F. Liang, "Cost-driven scheduling for deadline-based workflow across multiple clouds," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1571–1585, Dec 2018.
[47] V. Cardellini, V. D. Valerio, and F. L. Presti, "Game-theoretic resource pricing and provisioning strategies in cloud systems," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 86–98, 2020.
[48] J. Wan, R. Zhang, X. Gui, and B. Xu, "Reactive pricing: An adaptive pricing policy for cloud providers to maximize profit," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 941–953, Dec 2016.
[49] S. Mireslami, L. Rakai, B. H. Far, and M. Wang, "Simultaneous cost and qos optimization for cloud resource allocation," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 676–689, Sep. 2017.
[50] H. Cheng, Z. Li, and A. Naranjo, "Cloud computing spot pricing dynamics: Latency and limits to arbitrage," *Information Systems Research*, vol. 27, no. 1, pp. 145–165, 2016, cited By 9.
[51] S. Karunakaran and R. Sundarraj, "Bidding strategies for spot instances in cloud computing markets," *IEEE Internet Computing*, vol. 19, no. 3, pp. 32–40, 2015, cited By 9.
[52] S. Secci, P. Raad, and P. Gallard, "Linking virtual machine mobility to user mobility," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 927–940, Dec 2016.
[53] L. M. Vaquero, S. S. Lor, D. Audsin, P. Murray, and N. Wainwright, "Sampling ISP Backbone Topologies," *IEEE Communications Letters*, vol. 16, no. 2, pp. 272–274, February 2012.
[54] T. Mizrahi and Y. Moses, "On the behavior of network delay in the cloud," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2016, pp. 875–876.
[55] C. Ouyang, E. Verbeek, W. M. van der Aalst, S. Breutel, M. Dumas, and A. H. ter Hofstede, "Formal semantics and analysis of control flow in WS-BPEL," *Science of Computer Programming*, vol. 67, no. 2–3, pp. 162 – 198, 2007.
[56] P. Bailis, A. Davidson, A. Fekete, A. Ghodsi, J. M. Hellerstein, and I. Stoica, "Highly available transactions: Virtues and limitations," in *Proc. of Very Large Data Base Endowment*, vol. 7, no. 3, Nov. 2013, pp. 181–192.
[57] "AWS ping traces repository," https://github.com/pbailis/aws-ping-traces, [Online; retrieved on April 5, 2017].
[58] J. Sun, C.-H. Lai, and X.-J. Wu, *Particle Swarm Optimisation: Classical and Quantum Perspectives*. CRC Press, 2011.
[59] X. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A Multi-Facet Survey on Memetic Computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591–607, Oct 2011.
[60] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, available for free at http://cs.gmu.edu/~sean/book/metaheuristics/.
[61] A. Rezaee Jordehi and J. Jasni, "Parameter selection in particle swarm optimisation: a survey," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 4, pp. 527–542, 2013.
[62] O. Kramer, *A Brief Introduction to Continuous Evolutionary Optimization*. Springer, 2013.
[63] G. Grabarnik, L. Shwartz, and M. Tortonesi, "Business-driven optimization of component placement for complex services in federated Clouds," in *Network Operations and Management Symposium (NOMS 2014), 2014 IEEE/IFIP*. IEEE, 2014, pp. 1–9.

Walter Cerroni (Senior Member, IEEE) is an Associate Professor of communication networks with the University of Bologna, Italy. He coauthored more than 130 articles published in the most renowned international journals, magazines, and conference proceedings. His recent research interests focused on multiple aspects of control, management and orchestration of communication network infrastructures, including software-defined networking, network function virtualization, multi-access edge computing, fog computing, service function chaining, intent-based networking systems. He serves/served as Series Editor for the IEEE Communications Magazine, Associate Editor for the IEEE Communications Letters, and Technical Program Co-Chair for IEEE-sponsored international workshops and conferences.

Luca Foschini, IEEE (Senior Member) received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 2007. He is currently an Associate Professor of computer engineering with the University of Bologna. His interests span from integrated management of distributed systems and services to mobile crowd-sourcing/-sensing, from infrastructures for the deployment of Industry 4.0 solutions to fog/edge cloud systems. Finally, he is serving as secretary of the ComSoc CSIM TC, and as voting member and awards committee chair for the IEEE ComSoc EMEA board.

Genady Ya. Grabarnik currently teaches at Math and CS Department, St John's University. He is a trained mathematician and authored over 80 papers. He spent 10 years at IBM T.J.Watson Research Center where his work was celebrated with a number of awards including Outstanding Technical Achievement Award and Research Achievement Awards. He is a prolific inventor with over 65 US patents. His interests include research in functional analysis, inventions, and research in computer science and artificial intelligence.

Filippo Poltronieri (Student Member, IEEE) received the Ph.D. degree from the University of Ferrara, Italy, in 2021. He joined the Distributed System Research Group, led by Prof. Cesare Stefanelli in 2017. His research interests include Distributed Systems, Edge/Fog Computing, Cloud Computing, and tactical networks. He has been visiting the Florida Institute for Human & Machine Cognition (IHMC) in Pensacola, FL (USA) in 2016-2017 and 2018.

Larisa Shwartz (Member, IEEE) received the PhD degree in mathematics from UNISA University. She is currently a researcher at the IBM T.J. Watson Research Center, Yorktown Heights, NY. She has research experience in mathematics and computer science, but is now focusing on IT service management technologies for service delivery. She has more than 55 publications and 52 patents.

Cesare Stefanelli (Member, IEEE) received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 1996. He is currently a Full Professor of distributed systems with the Engineering Department, University of Ferrara, Italy. At the University of Ferrara he coordinates a Technopole Laboratory dealing with industrial research and technology transfer. He holds several patents, and coordinates industrial research projects carried on in collaboration with several companies. His research interests include distributed and mobile computing in wireless and ad hoc networks, network and systems management, and network security.

Mauro Tortonesi (Member, IEEE) received the Ph.D. degree in computer engineering from the University of Ferrara, Italy, in 2006. He was a Visiting Scientist with the Florida Institute for Human & Machine Cognition (IHMC), Pensacola, FL, USA, from 2004 to 2005 and with the United States Army Research Laboratory, Adelphi, MD, USA, in 2015. He is currently an Associate Professor with the Department of Mathematics and Computer Science, University of Ferrara. He holds two international patents and participates on the editorial board of four international scholarly journals. He has co-authored over 80 articles published in international venues in the distributed systems research area, with particular reference to IoT solutions in industrial and military environments, Cloud and Fog computing, wireless middleware, and IT service management.