Service Placement for Hybrid Clouds Environments based on Realistic Network Measurements

Walter Cerroni*, Luca Foschini†, Genady Ya. Grabarnik‡, Larisa Shwartz§, Mauro Tortonesi¶
* Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Italy
walter.cerroni@unibo.it
† Department of Computer Science and Engineering, University of Bologna, Bologna, Italy
luca.foschini@unibo.it
‡ Department of Math & CS, St. John's University, Queens, NY, USA
grabarng@stjohns.edu
§ Operational Innovation, IBM TJ Watson Research Center, NY, USA
lshwart@us.ibm.com
¶ Department of Engineering, University of Ferrara, Ferrara, Italy
mauro.tortonesi@unife.it

***Abstract*—The ever increasing sophistication in IT services demands to consider dynamic and large-scale deployments in distributed environments built on top of federated public and private Clouds. Unfortunately, in these complex hybrid Cloud environments, it is extremely difficult to estimate the impact of even simple reconfigurations of the IT service architecture before enacting them. There is the need for new service management tools that are capable of exploring alternative IT service architectures, of accurately evaluating them to find out the most convenient one, and of reconfiguring the IT service accordingly. This paper presents Business-Driven Management as a Service Plus (BDMaaS+), that significantly evolves our previous work to consider the business-driven evaluation of IT services in hybrid Cloud environment, advanced IT service models that complex multi-tier workflows, and realistic network models. The experimental evaluation of BDMaaS+ to optimize a realistic large-scale IT service demonstrates that our tool is capable of finding a more convenient configuration that significantly reduce the total daily costs while granting at the same time agreed service levels.**

***Index Terms*—Cloud Computing, Optimization, Simulation.**

I. Introduction

Cloud computing technological advances and aggressive commercial offerings are pushing an ever growing number companies to migrate a 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].

This is particularly complex because in the new scenario there is still a wide lack of proper tools to monitor the behavior of the distributed infrastructure, such as to quantify and model delays among datacenters and/or simulate possible dynamic re-adjustments of the service deployment according to both system-level conditions as well as business-level drivers, such as the current pricing situation. These trends suggest the opportunity to investigate *highly dynamic and adaptive Cloud based IT services*, that can dynamically realign their configuration to match the ever changing characteristics of modern Cloud environments. This ambitious objective calls for the development of new and sophisticated service management tools, that are capable of exploring alternative IT service architectures, of finding out the most convenient one, and of reconfiguring the IT service accordingly.

To overcome all those open issues, we propose a novel support called Business-Driven Management as a Service Plus (BDMaaS+). The complex nature of modern hybrid Cloud IT services, with a large number of workflows on top of a plethora of software components of different types and deployed in different data centers, makes it difficult to estimate the impact of (even simple) reconfigurations of the IT service architecture through the adoption of (tractable) analytic models. BDMaaS+ adopts the simulative approach instead: it reenacts IT services under different configurations, using realistic service execution and network communication models, to accurately capture peculiar behavior of real-life IT services. It then calculates the performance of each IT service configuration using sophisticated business level evaluations.

BDMaaS+ significantly evolves our previous support BDMaaS proposal [2] by showing several new elements of novelty. First, BDMaaS+ allows to optimize multi-tier services consisting of complex workflows composing multiple application components of different types, also considering realistic deployment constraints. Second, leveraging our experience in inter-data center network delay modeling, it considers realistic characterizations to be accounted for the dynamic re-adaptation of component deployment at runtime [3]. Third, it proposes an extended cost model that is capable of calculate expenses for running application components in private Cloud data center. Finally, it adopts a significantly improved optimization solution based on a memetic algorithm to enable robust and resilient exploration of the large and challenging search space.

We released BDMaaS+ as open source to make it available to the community working in the field. For more information, we refer the reader to the project home page: https://de.unife.it/dsg/research-projects/BDMaaS.

II. Modeling Web Services in Hybrid Clouds

In a hybrid Cloud environment IT service architectures must be assumed as made of service components that are distributed over different public and private Cloud data centers. In particular, 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 [4]. We assume that service requests are routed through a number of software components, or activities in WS-BPEL parlance. While in our experiments we consider the case of sequential workflows with the help of occasional synchronization elements, our model is indeed more general and can be applied to different service dynamics.

We also consider deployment constraints at the software component level, that limit or outright prohibit the reallocation of specific components to other data centers. In fact, complex hybrid Cloud IT services often leverage 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.

Modeling workflows as the set of service components that must serve a given request in the corresponding business process allows us to consider complex services and to characterize the relationships between the involved service components. In addition, it enables 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.

A. Service Execution Model

We define components as software entities that run inside a VM. Software components will typically have different resource requests. Some components might run only on a subset of the VM types available in the Cloud - the most powerful ones. In addition, the performance of the service component will depend from the size of the VM it is allocated to.

We adopted a service component execution model based on $G/G/s_i$ FCFS queues. In this model, every service component is modeled as a queue with customizable service times, that could be defined either through a parametric probability distribution (e.g., Poisson, Gaussian, lognormal, etc.), through an empirical probability distribution (e.g., collected from service logs of), or even through a trace measured from a component implementation run in a real-life environment.

In addition to be a conceptual framework that is easy to understand and to work with, $G/G/s_i$ FCFS queues have several other advantages. In fact, they allow to define a service discipline that depends from the VM type on which the component is instantiated, thus enabling to easily capture the performance improvement brought by the adoption of a more powerful VM type without excessively complicating the model.

B. A Latency Model for Inter-Cloud Data Center Communications

In the evaluation of a component placement configuration for a globally deployed IT service, the accurate reenactment of network latencies becomes an essential modeling aspect. Unfortunately, realistically modeling inter-datacenter communication delays is still a relevant research problem. In fact, communication latency between different data centers depends from several (uncontrollable and highly varying) factors. Its accurate modeling calls for the adoption of robust and comprehensive solutions, both from the empirical and from the theoretical perspectives, based on data collected from real-life measurements.

With regards to the communication latency model, we adopt the solution that we presented in [3]. More specifically, we developed an evolved 2-parameter model that improves existing approaches by considering both delays, i.e., ping Round-Trip-Times (RTTs), and the Time-To-Live (TTL) parameter reported by IP packets, to consider multiple communication paths. We then developed a Gaussian mixture model on each (RTT, TTL) cluster, using the original Relaxed Boxed Approximation algorithm for high accuracy identification of model parameters.

III. Cost Evaluation

To fully estimate the total cost TC for a service provider to run a Web service in a hybrid (public/private) Cloud with a specific configuration component CC, we need to consider several contributions:

$$TC(CC) = SP(CC) + PC(CC) + PRC(CC) \tag{1}$$

where SP is a function that calculates the costs caused by SLO violation penalties and PC and PRC are functions that calculate the IT costs incurred for the public and private Cloud data centers respectively. In turn, CC represents an instantiation list of $\#vms_{ijk}$ VMs of type j, where j is an index in the *(small, medium, large, xlarge, xxlarge, xxxlarge)* tuple, run in data center DC_i hosting a service component of type C_k.

To evaluate $SP(CC)$, BDMaaS+ reenacts the IT service with component configuration CC and analyzes the corresponding simulation logs. First, it calculates the observed value for the metrics of relevance for SLO violation purposes, and then it confront those values with target ones to evaluate whether SLO violations occured. If so, BDMaaS+ the corresponding amount of penalties that running the IT service in configuration CC would caused the service provider to incur.

The following subsections respectively detail how the PC and PRC functions are evaluated.

Finally, let us note that, when evaluating the performance at the business level of an IT service, BDMaaS+ also considers in addition to $TC(CC)$ a "performance penalty" component $PP(CC)$ that calculates the *intangible costs* related to performance regressions, risk management, and reconfigurations for operating the IT service in configuration CC with respect to the current configuration CC_0.

A. 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 vms 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:

$$PC(CC) = \sum_{i=1}^{N}\sum_{j=1}^{M} vmhu(vm_{i,j}) * vmhc(vm_{i,j}) \tag{2}$$

An additional resource that have 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.

B. A Cost Model for Private Clouds (or Data Centers)

Typical expense categories for creating and maintaining private Clouds are:

1) (FC) Facilities and related expenses (buildings, additional cooling hardware, networking cabling, etc.)
2) (HW) Hardware (computing, storage, networking, etc. devices)
3) (SW) Software (licensing of Virtualization Layer, OSes, server and application software)
4) (LBR) Labor (maintenance of facilities, hardware, software, etc)
5) (EN) Energy consumption (powering computing, storage and other devices, air conditioning, etc.)
6) (CS) Cloud Services (consumer's maintenance, internet provisioning, data transfers in and out, storage, etc.)

Organizations that use private Clouds typically cover some expenses, while other expenses are covered by subdivisions consuming private Cloud services. As a rule, expenses in categories 1, 2, 4, 5 are shared by all internal consumers or covered by organization as a whole (base expenses/costs, BC). Shared (base) type of expenses if charged are charged either on fixed rate (independently from usage) or inverse proportional to total number of consumers.

Expenses in categories 3, 6 are usually covered by consuming subdivisions on per usage basis (we use the term variable cost or VC). We can assume VC to be a linear function of costs per VM.

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

$$BC = \sum BC(DC_i, C_j) \tag{3}$$

To simplify formula 3 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 equation 2. In addition, since typically private Clouds have restricted resources, we have constrains on total number of possible VMs deployed ($TNVM(DC_i)$).

To ensure proper instantiation we need VMs with resources at least satisfying the minimal requirements for the software component considered and sufficient performance to handle typical component load. For that we have consider resource types as different dimensions: CPU, GPU, I/O, network, etc. Licenses are also another type of resource. For instance, team licenses might have a maximum number of instances of a given component type at any time. For example a component running deep learning calculations will require significant GPU, a DBMS will require significant RAM and disk storage, an FTP server will require significant disk space, etc. We express that as m-th resource required for component C_k does not exceed m-th resource for j-th VM: $r_m(VM_j) \geq r_m(C_k)$ (*Deployability Constraint*). Another constraint is that a component is deployed to one VM and there are no empty VMs (*Component per VM Constraints*).

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

$$\begin{aligned} PRC(CC) = &\sum_{i=1}^{N}[FXC(DC_i) \\ &+ VC(\sum_{j=1}^{M}\sum_{k=1}^{T} vmhu_{i,j,k} * vmhc_{i,j,k})] \\ &\texttt{Subject to:} \\ &\textit{Deployability Constraints} \\ &\textit{Component per VM Constraints} \\ &\sum_{j=1}^{M}\sum_{k=1}^{T} \#vm_{i,j,k} \leq TNVM(DC_i), \quad i = 1, ..., N \end{aligned} \tag{4}$$

IV. Memetic Algorithm Optimization

For the service component placement optimization, we adopted a *memetic algorithm* that explores the space of possible configurations to find the best performing one. Memetic algorithms are efficient hybridizations of population-based optimization heuristics with refinement techniques such as smart local search algorithms [5].

More specifically, we have chosen to use a memetic algorithm based on the combination of an outer search phase based on *Quantum Particle Swarm Optimization* [6] 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.

This construction represents a common form of decomposition of 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 [7]. In our case, the decomposition allows to treat separately the outer and inner search phases, which have different characteristics and operate on significantly different search spaces.

A. Outer search

The global search part of the metaheuristics takes care of assigning service component to each data center. More specifically, it aims at finding $\#c_{ik}$, the number of software component instances of type C_k to allocate at data center i.

This is an apparently rather simple search space of $\mathbb{N}^{I*K}$ size, where I is the number of data centers and K is the number of software component types that we consider, which can be independently explored. However, not only this represents a very large vector space, but the hybrid Cloud application scenario forces us to consider constraints on the possible deployments of components in public and/or private Cloud data centers. For instance, for security or management concerns customers might want to enforce the deployment of DBMS (or other data layer) components in their private data Centers. As a result, the allocation technique should enable to define and enforce these constraints.

In addition to the large search space, the optimization problem has to deal with a complex objective function, which is likely to be “jagged” and not differentiable. This means that we cannot adopt traditional techniques, such as gradient-descent based ones, that are not well suited for this task. Instead, there is the need to consider metaheuristics.

Quantum Particle Swarm Optimization (QPSO) is a variant of Particle Swarm Optimization (PSO), a swarm intelligence technique inspired to the behaviour of bird flocks [6]. Traditional PSO is a relatively simple to implement optimization algorithm that, however, unlike GAs presents a few critical aspects, such as lower resilience to early convergence [6] and a more difficult parameter tuning process [8].

Improved versions of the algorithm such as QPSO and variants of PSO based on multiple swarms have later emerged to addressed 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 [9]. Finally, being a population-based metaheuristics, PSO variants are (relatively) easily parallelizable, thus enabling (and suggesting) the adoption of in Cloud architectures for their implementation.

B. Inner search

The local search procedure takes care of deciding which VM types to use for the instantiation of service components within each Cloud data center.

When attempting to map a service component to a VM type, the first thing that the procedure does is to select the VM types that satisfy the deployment constraints expressed in the previous Section. To this end, it analyzes the metadata that describes the requisites of each service component.

Once the the allowed VMs are known, we can start the local search to find which VM types are the best suited to host service component. This is effectively a search over a combinatorial space of size:

$$\prod_{c=1}^{C} a(c)^{\sum_{d=1}^{D} b(c,d)} \tag{5}$$

where $a(c)$ is the number of VM types allowed for the instantiation of component c and $b(c, d)$ 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, following the approach originally introduced in [10], we have chosen to adopt a simplified VM allocation algorithm. More specifically, we decided to enforce a (truncated) discrete exponential distribution for the set of VM types to instantiate for each component - an assumption that effectively transforms the combinatorial search to a significantly more convenient search within a continuous and lower dimension space.

V. Experimental Results

Let us consider, as a case study, a realistic enterprise-class IT service deployed on a large scale for customers with global presence, and try to optimization its architectural configuration using BDMaaS+. More specifically, we will consider a Money Management and Transfer System (MMTS) IT service, that allows users to manage their bank accounts and to submit money transfer request.

MMTS is composed by 3 applications (*A*, *B*, and *C*), whose architecture mostly follows the classic 3-tier paradigm, with a Web Server, an Application Server and a DataBase Management System. The presence of additional components such as a Financial Transaction System and a Queue Manager[1] extends the aforementioned paradigm with an additional tier. In addition, applications B and C share the same replicated DataBase.

For the IT service deployment, we consider 6 public Cloud data centers: Amazon EC2’s US East, US West, Ireland, Sydney, Brazil, and Singapore; and 2 private Cloud data centers, respectively located in northwestern USA and in Japan.

[1]The Queue Manager represents the entry point to a reporting function, which represents an external support system for MMTS, and as a result we do not consider the related software components in the optimization of MMTS.

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

While most MMTS software components can be deployed to any Cloud data center, we consider a couple of deployment constraints. The RDBMS C component must reside in Private Cloud 1 (US) for security reasons, and the Financial Transaction System component, implemented by a legacy system, must reside in private Cloud data center 2. Thus, neither of these components can be either replicated or redeployed to any other data center.

We consider a global customer with 5 divisions distributed across the World that account for a different share of requests (1/9 from the East Coast USA division, 1/6 from West Coast USA, 5/18 from South America, 1/3 from Asia, and 1/9 from Europe). We assume a constant intensity for the aggregated flow of requests, which we model with a Pareto distribution with location 1.2E-4 and shape 5, corresponding to 400,000 requests per minute. We also assume that the requests emanating from each division will be automatically forwarded to the closest Cloud data center.

To model the communication latencies between the different locations, we assume that the times to transfer a request or response message from a location to the other are modeled according to the Gaussian mixture model we developed in [3]. Since the private Cloud data centers considered in this experiment are close to Amazon EC2's Oregon and Tokyo data centers, we assume that we can use the network communication model we built for those public Cloud data centers without loss of generality. Finally, we assume that the latencies for message transfers within a single location are significantly smaller and can thus be safely ignored.

We consider 12 different workflows for the MMTS IT service, as shown in Table I. For each *(workflow, customer location)* couple, we set an SLO objective defined as a step function of the average response time metric and a corresponding violation penalty.

Finally, we used an arctg-based model to define the PP component for MMTS. Defining the metric:

$$m_{CC} = \frac{expected\_requests - served\_requests}{expected\_requests}$$

as the share of requests that were not fully served in the simulation-based evaluation of the IT service with configuration CC we then have:

$$PP(CC) = \begin{cases} max_p * \frac{2}{\pi} * arctg(k * m_{CC}) & \text{if } m_{CC} > 0 \\ 0 & \text{otherwise} \end{cases}$$

with $max_p = 100,000$ USD/day and $k = 200$ (we experimentally verified these values to be well suited for the purposes of optimizing the MMTS service).

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.

Given the problem size and complexity, we configured the QPSO algorithm to use 40 particles and a contraction-expansion coefficient $\alpha = 0.75$ and adopted an 8-sample random search for the simplified VM allocation algorithm.

The results we obtained from applying BDMaaS+ to the case study described above are presented in Fig. 1 and 2. Fig. 1 depicts the sum of the TC and PP components discussed in Section III, thus showing both tangible and intangible operational costs. Fig. 2 depicts only the TC component, thus showing only tangible costs.

As it can be seen, in only 11 iterations of the QPSO algorithm, each one corresponding to the evaluation of 320 different IT service configurations, BDMaaS+ is capable of finding the optimal configuration for the system, reducing the total daily costs down to 45,048 \$. This remarkable convergence speed, significantly higher than the one achieved by the optimization solution based on genetic algorithms that we adopted in an earlier version of BDMaaS+, was constantly exhibited in all the experiments we conducted.

VI. RELATED WORK

Early efforts in service placement 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.) [11]. Several papers investigated load balancing in Cloud data

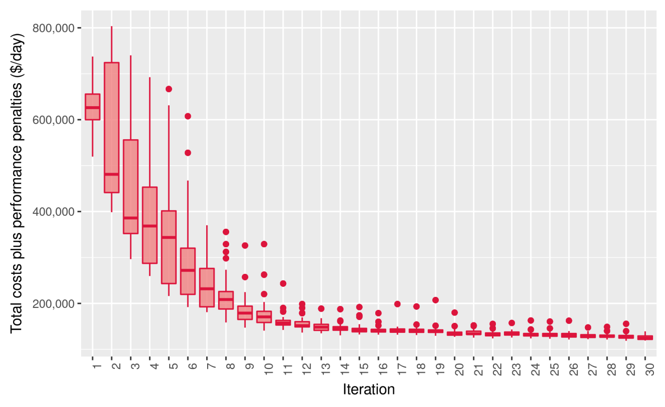

Fig. 1. Distribution of total costs and performance penalties for the evaluated configurations of MMTS at each iteration of the optimization algorithm.

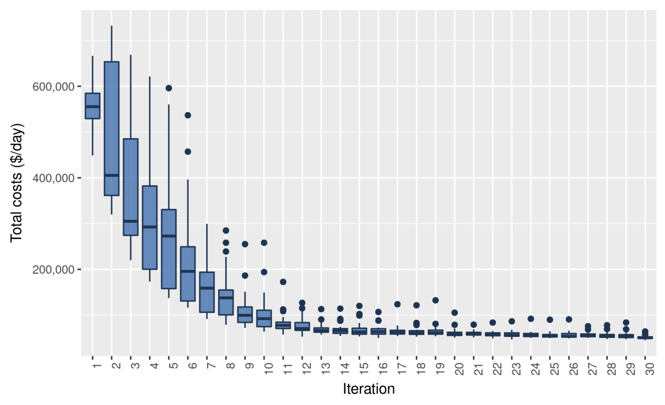

Fig. 2. Distribution of total costs for the evaluated configurations of MMTS at each iteration of the optimization algorithm.

centers while considering multiple layers and proposed a solution based on multi-dimensional knapsack [12], network-aware VM placement for aggregate traffic reduction purposes [13], and VM placement solutions resilient to dynamic traffic variations [14].

With regards to the optimization of large scale IT services, the literature can be roughly divided in two complementary research avenues. The first on considers an "internal" Cloud provider-centric perspective, e.g., investigating the management of IT service and infrastructure changes to achieve SLOs and minimize business disruptions [15], or proposing a migration framework to simultaneously minimizes VM migration costs and optimize the success rate of virtual resource mapping requests [16]. The second research avenue, which includes the present work, considers an "external" and service provider-centric perspective, e.g., by comparatively analyzing the economic models for Cloud computing and traditional in-house IT service delivery [17].

From the perspective of simulating IT services in Cloud environments, let us note that while a number of studies and theoretical models have been proposed in the literature to effectively address the service placement problem [16] [18] [19], only few simulation tools are available [20]. To the best of our knowledge, we couldn't find in the literature other solutions that, like BDMaaS+, at the same time allows to consider complex multi-tier application workflows, a realistic model for inter-datacenter delays in a hybrid Cloud scenario, and sophisticated business-level evaluation of IT service configurations.

The adoption of computational intelligence methods for the optimization of Cloud based applications has recently started receiving an increasing attention in network and system management research area. Moens et al. [21] compare a traditional ILP optimization method with genetic algorithms (GAs) and particle swarm optimization (PSO) based ones, showing that for large problems the significantly better scalability of GAs and PSO offsets the guarantee of finding the best solution of the problem provided by ILP. Li et al. [19] consider dynamic VM prices, adopting for experimental evaluation purposes a relatively simple scheme that modulates the cost of VM allocations according to the currently resource availability at Cloud data centers, and compare the performance of several optimization algorithms. In previous work, we attempted to optimize the placement of software components in federated Cloud environments using an adaptive genetic algorithm [2] and a memetic algorithm composed by an outer search based on a genetic algorithm and an inner random search [10] respectively. Compared those works, the Quantum Particle Swarm Optimization algorithm proposed in this work allows for a much faster convergence.

Finally, to the best of our knowledge, our solution adopts a realistic latency model (that we recently presented in [3]) that improves existing similar works in the literature, such as [22], [23], [24], by considering both Round-Trip Time (RTT) and Time-To-Live (TTL) parameters to obtain more realistic values.

VII. Conclusions

The paper proposes Business-Driven Management as a Service Plus (BDMaaS+), a novel support for the placement of complex workflows for hybrid Cloud environments. Collected results show that BDMaaS+ is able to significantly reduce the total daily costs by granting at the same time agreed service levels under realistic network, application, and cost models. Moreover, BDMaaS+ is available for practitioners to experiment advanced business-driven placement strategies in an easy way.

Encouraged by these results, we are now working on various future research directions: 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 Net-*

work Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE, 2013, pp. 320–327.

[2] M. Tortonesi and L. Foschini, "Business-driven service placement for highly dynamic and distributed cloud systems," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2016.

[3] 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.

[4] 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642307000500

[5] 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.

[6] J. Sun, C.-H. Lai, and X.-J. Wu, *Particle Swarm Optimisation: Classical and Quantum Perspectives*. CRC Press, 2011.

[7] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[8] 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.

[9] O. Kramer, *A Brief Introduction to Continuous Evolutionary Optimization*. Springer, 2013.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] K. Sripanidkulchai and S. Sujichantararat, "A business-driven framework for evaluating cloud computing," in *Network Operations and Management Symposium (NOMS 2012), 2012 IEEE/IFIP*, 2012, pp. 1335–1342.

[18] 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.

[19] 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.

[20] 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. [Online]. Available: http://dx.doi.org/10.1002/spe.995

[21] H. Moens, B. Hanssens, B. Dhoedt, and F. D. Turck, "Hierarchical network-aware placement of service oriented applications in clouds," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–8.

[22] 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.

[23] 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.

[24] 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.