Admission Control for Elastic Cloud Services

Kleopatra Konstanteli* Tommaso Cucinotta§ Konstantinos Psychas* Theodora Varvarigou*

kkonst@mail.ntua.gr tommaso.cucinotta@alcatel-lucent.com el06600@mail.ntua.gr dora@telecom.ntua.gr

*School of Electrical and Computer Engineering, National Technical University of Athens, Greece

§Bell Laboratories, Alcatel Lucent, Dublin, Ireland (formerly at Real-Time Systems Laboratory, Scuola Superiore Sant'Anna, Pisa, Italy)

***Abstract*—This paper presents an admission control test for deciding whether or not it is worth to admit a set of services into a Cloud, and in case of acceptance, obtain the optimum allocation for each of the components that comprise the services. In the proposed model, the focus is on hosting elastic services the resource requirements of which may dynamically grow and shrink, depending on the dynamically varying number of users and patterns of requests. In finding the optimum allocation, the presented admission control test uses an optimization model, which incorporates business rules in terms of trust, eco-efficiency and cost, and also takes into account affinity rules the components that comprise the service may have. The problem is modeled on the General Algebraic Modeling System (GAMS) and solved under realistic provider's settings that demonstrate the efficiency of the proposed method.**

***Index Terms*—admission control; elasticity; cloud computing; optimum allocation;**

I. Introduction

Nowadays, more and more distributed applications are being provided in the Cloud as a composition of virtualized services. For example, in an IaaS, each service is deployed as a set of virtualized components, i.e. Virtual Machines (VMs), that are activated according to their workflow pattern each time a request arrives from the end users. The use of virtualization techniques allows for the seamless allocation of each component of the distributed service inside the Cloud. It also makes easier the process of horizontal elasticity, i.e. adding/removing extra duplicate VMs for each component during runtime to maintain a certain level of performance for the overall service when there are variations in the workload.

At admission control time, and in order to provide strong performance guarantees, the Infrastructure Provider (IP) must consider not only the basic computational and networking requirements but also the extra ones that may be needed to be added at runtime, defined as elastic requirements. In many cases, the elastic requirements may be quite large compared to the basic ones. For example, given a service with a high variation in the number of users, the number of VMs that may be required to be added at runtime may be many times multiple of the number of the basic ones. Therefore the amount of elastic requirements plays a significant role in the total requirements and therefore in the cost of hosting the service, and the IP has a strong interest in investigating the possibility of reducing the resources that need to be booked for elasticity reasons when accepting the service. At the same time, such an approach may increase the possibility of deviating from the agreed quality of service (QoS) level, i.e. the required availability of the resources, and the imposed penalties may as well outgain the advantages of this approach.

Apart from the resource requirements, when allocating a set of services within a Cloud, this problem needs to be solved by optimizing proper metrics that express the goodness of the found allocation. Clearly, from the IP's perspective, such metrics are significant for the cost of each allocation solution. However, apart from the cost, nowadays an IP's business policy may include other factors such as the risk of collaborating with a given Service Provider (SP), as well as other factors that relate to the Cloud resources, such as how eco-efficient a given host is as compared to the others [2]. For example, given a profit-driven IP, in some cases it may be more lucrative to admit a highly profitable service even under the risk of jeopardizing existing less profitable ones. Likewise, given a very conservative IP and two services competing for the same resources, the less profitable service of the two may be accepted in the Cloud, if it is sufficiently less risky than the other one.

The approach proposed in this paper focuses on elasticity and tackles the problem of optimum allocation of distributed services on virtualized resources by incorporating a probabilistic approach in terms of availability guarantees. The proposed optimization model relies on the actual probabilities of requiring extra computational and networking capacity for the services, which are incorporated into the admission control test, allowing to reduce the physical resources that are required for elasticity reasons. The resulting model constitutes a probabilistic admission control test that also allows for proper trade-offs among business level objectives in terms of trust, eco-efficiency and cost, depending on their relative significance as considered by the IP. The output of the optimization problem includes the allocation pattern, i.e. the selected hosts and subnets for hosting the services, and the computational and networking capacity (both basic and elastic) that is allocated on them for each component of the accepted services.

It should be noted that in the preliminary short version of this work [10], the formulated problem considered only the computing requirements of the services, whereas in this paper, the networking requirements are also modeled inside the admission control problem. Furthermore, the model in this paper has been extended to incorporate the level of trust between the IP and the SP as well as the eco-efficiency of the physical hosts inside its multi-objective function and the results have been extended to demonstrate the effect of these factors in the overall admission control process. Lastly, the presented model allows for partial acceptance of the services and possible federation with other Cloud providers, and takes into account the affinity rules the services may have when allocating them onto the available hosts and subnets.

This paper is organised as follows. Section II gives an overview of the related work, followed by the description of the problem under study in Section III. The formulation of the probabilistic admission control problem is described in detail in Section IV, whereas Section V presents an evaluation of the performance of this model and an indicative case study that validates the efficiency of the proposed approach. Finally, conclusions are drawn in Section VI.

II. Related work

Several works that address the problem of optimal allocation of services in Cloud systems have appeared in the recent years [3][17]. In [14], the authors examine this problem in a multi-provider hybrid Cloud setting against deadline-constrained applications. To this direction a mixed integer optimization problem is formulated with the objective to minimize the cost of outsourcing tasks from data centers to external Clouds, while maximizing the internal utilization of the data centers. Mazzucco et al. [12], addressed the problem of optimal allocation for maximizing the revenues of Cloud providers by minimizing the amount of consumed electricity. In [4], a resource allocation problem is formulated in which later tasks can reuse resources released by earlier tasks, and an approximation algorithm that can yield close to optimum solutions in polynomial time is presented. In all these works, the objectives are profit-driven, whereas in this paper, the presented optimization model combines a more complete set of parameters both on the SP and the IP side, such as the trustworthiness of the SP that owns the service and the eco-efficiency of the IP's resources.

A probabilistic aspect to the allocation of distributed services can be found in [15]. In prior work of ours [9], the problem of optimum allocation of real-time workflows with probabilistic deadline and availability guarantees was tackled. In that work the main focus was on the probabilistic framework allowing the provider to overbook resources in the various time frames of each advance reservation request, knowing the probabilities of actual usage/activation of those services by the users. In the present paper, instead, the focus is on the problem of hosting elastic services the workload requirements of which may dynamically grow and shrink, as depending on the dynamically varying patterns of requests submitted by a dynamically varying number of users.

A way to estimate the probability that an end-to-end deadline is respected for a given composition of services is the one to build probabilistic models for the performance achieved by a composition of distributed services. For example, Zheng et al. [16], among others, investigated on mathematical models to compute the probability density function of the response-time of service compositions under various compositional patterns. However, in the present paper we consider elastic services as our primary concern, and the probabilistic composition of services is dealt with by recurring to the probabilistic framework detailed in Section IV-E.

III. Problem Description

In the context of the problem under study, largely inspired by the OPTIMIS [8] and S(o)OS[1] European Projects, an IP is a business entity that owns a set of physical hosts with potentially heterogeneous characteristics in terms of processing speed, architecture, and underlying network capabilities, and establishes SLAs with SPs for hosting distributed services over a period of time. Each service is composed of components that are horizontally scalable, i.e. they are capable of distributing their own work over a number of VMs which can be deployed on different cores, processors, hosts and subnets.

The IP books in advance a set of physical resources for hosting the virtualized components that comprise the service, i.e. the VMs that encapsulate them. Each different type of component is characterized by specific computing and networking requirements, in terms of an abstract single-valued performance metric, as explained in detail in Section III-A. According to the expected usage of the service, the SP is allowed to specify a lower and an upper limit to the computing and networking requirements of each component, that correspond to the basic and elastic resources that may be needed during runtime. At the first activation of the service, only the basic resources are participating in the execution. During runtime and according to workload fluctuations and the policies in place, elastic VMs may be added. However, their capacity requirements cannot exceed the limit defined in the SLA.

A. Performance Model

In this work, it is assumed that the computing and networking capabilities of each physical host, as well as those of the services, may be expressed in terms of a single performance metric. For example, across a set of hosts with similar capabilities in terms of the Instruction-Set Architecture (ISA) of the CPUs, the computing capabilities may be approximated in terms of instructions per second that each host can process, accounting for the different clock speeds and number of CPUs and cores available in each one of them. For example, a host with a single 1 GHz CPU would have a computing capability of 10^9, while a quad-core 3 GHz host would have a computing capability of $12 \cdot 10^9$. Similarly, we would roughly say that a service under given operating performance conditions (i.e. exhibited response-time within the desired range, under a workload following the expected pattern) might require $3 \cdot 10^9$ instructions per second. This would imply that the service should be hosted either replicated over 3 of the mentioned single-CPU machines, or simply as a single instance occupying one of the CPUs of the mentioned quad-core system.

Alternatively, the performance metric might be defined in terms of the performance achieved by a given benchmark (e.g.

[1]More information is available at: http://www.soos-project.eu/ and http://www.optimis-project.eu/.

LINPACK[2] or others) that is relevant for the kind of applications that may potentially be hosted. Actually, a more precise performance model could consider a vector of metrics, e.g., as coming out of a number of heterogeneous benchmarks (e.g. linear algebra, graphics, integer and floating-point operations, etc.). However, in the present work, a single metric is used for the sake of simplicity, and its values are supposed to vary in a range of positive real numbers.

As implied by the just mentioned example, we assume an ideal model of scalability for software, in which:

- each service can be arbitrarily decomposed in a number of possibly imbalanced replicas, running over possibly heterogeneous hosts; this is possible thanks to virtualization technologies by which more and more VMs hosting replicas of a service can be instantiated;
- the whole performance of a service is given by the sum of the performance of the decomposed replicas.

Note that both assumptions may be easily verified in typical Cloud computing services in which the horizontal scalability is needed to deal with a high number of users potentially accessing the services. In such a case, the multitude of replicas each service instantiates serves requests on behalf of different (groups of) users. Each replica can thus operate essentially in isolation from the others, thus the service falls under the model of loosely coupled (or embarrassingly) parallel software. However, as a future extension of this work, the performance model may be extended to consider workload and synchronization overheads by integrating methodologies such as [5]. Furthermore, we assume that the additional potential overheads due to the interference between VMs possibly being hosted on the same hosts are already accounted in the abstract resource requirements of the service. This seems reasonable in this context as we deal with horizontally scalable services that can span across various physical nodes, when needed (as opposed to many small services that can be consolidated on the same host). However, investigations on how to extend the model with a more informed inter-VM interference overhead model are reserved for future work. Note also that the impact of the inter-VM interferences on their temporal behavior can be kept under control using proper soft real-time schedulers at the hypervisor level [7].

B. Resources Topology

The resources of the provider may be generally considered as an interconnection of (potentially heterogeneous) networks that interconnect (potentially heterogeneous) computing nodes. For example, various LANs enclosing multi-processor computing nodes are interconnected by means of one or more WANs. To this direction, the network topology is characterized by the following elements:

- A set of computing nodes, or hosts: $\mathcal{H} = \{1, \ldots, N_{\mathcal{H}}\}$. Each host $j \in \mathcal{H}$ is characterized by an available computing capacity $U_j \in \mathbb{R}^+$, which expresses the value of a given system-wide reference performance metrics (see also Section III-A).
- A set of available subnets: $\mathcal{N} = \{1, \ldots, N_{\mathcal{N}}\}$. Each subnet $n \in \mathcal{N}$ is characterized by a maximum aggregate bandwidth $W_n \in \mathbb{R}^+$, expressed in terms of bytes/s.
- The network topology information, specifying what hosts $\mathcal{H}_n \subset \mathcal{H}$ are connected to each subnet $n \in \mathcal{N}$.

[2]More information is available at: http://www.netlib.org/linpack/.

C. Services Notation

The following notation is used to refer to services:

- Set of service instances (referred to simply as services from here on): $\mathcal{S} = \{1, \ldots, N_S\}$.
- Each service $s \in \mathcal{S}$ is a workflow of $m^{(s)}$ components (encapsulated inside VMs): $\mathcal{S}^s \triangleq \{1, \ldots, m^s\}$, denoted also as $(\xi_1^s, \ldots, \xi_{m^s}^s)$.

Each component $\xi_i^s \in \mathcal{S}^s$ is characterized by the following parameters:

- Minimum basic computing capacity θ_i^s and network capacity b_i^s that ξ_i^s needs to perform its basic functionality on the hosts of a given subnet;
- Maximum extra computing capacity Θ_i^s and network capacity $\mathcal{B}_i^s$, also called elastic requirements, that ξ_i^s may properly exploit;

D. Service Level Agreement Model

As already described in Section III, the IP establishes SLAs with the SPs for hosting their services over a period of time. The SLA for a given service $s \in \mathcal{S}$ carries the following parameters:

- The description of the service workflow $\mathcal{S}^s$, which must be complemented by the computing requirements of each component ξ_i^s: θ_i^s, Θ_i^s, b_i^s, and $\mathcal{B}_i^s$.
- A minimum probability ϕ^s that the required resources are actually available when the request arrives, namely that there are sufficient computing and network resources for the activation of the VMs when needed.
- Gain $\mathcal{G}^s$ for the IP in case the service is accepted.
- Penalty P^s for the IP if the service fails to meet its QoS restrictions.

Furthermore, it is assumed that the IP owns or is able to obtain access to tools that enable him to acquire knowledge about the following two factors that help in the assessment process of admitting or not a service:

- Trust $\mathcal{T}^s$: this factor is a metric of how trustworthy the SP that owns the specific service is.
- Eco-efficiency E_j: this factor is a metric of how eco-efficient a given host j is.

Both these factors have a positive meaning (i.e. the higher their value, the more positive their effect is), and as explained in detail in Section IV-F, they are incorporated in the overall objective of the IP.

IV. Problem formulation

Using the definitions in Section III, the problem under study may now be formalized.

A. Unknown Variables

First of all, let us introduce the unknown variables to be computed. These are:

- The allocated (both basic and elastic) computing capacity for the components on the hosts: $\forall s \in \mathcal{S}$, $\forall i \in S^s$, $\forall j \in \mathcal{H}$, $x^s_{i,\,j} \in \mathbb{R}^+$. If a component ξ^s_i is not given any computing capacity on a given host j, then $x^s_{i,\,j} = 0$. The accepted elastic computing capacity for each component is $\sum_{j \in \mathcal{H}} x^s_{i,\,j} - \theta^s_i$.
- The allocated (both basic and elastic) network bandwidth for the components on the subnets: $\forall s \in \mathcal{S}, \forall i \in S^s, \forall n \in \mathcal{N}$, $y^s_{i,\,n} \in \mathbb{R}^+$, if ξ^s_i is deployed on some host $j \in \mathcal{H}_n$. The accepted elastic network capacity for each component is $\sum_{n \in \mathcal{N}} y^s_{i,\,n} - b^s_i$.

Note that the actual decomposition of each component ξ^s_i into VMs to be deployed on the various hosts where $x^s_{i,\,j} > 0$ is a lower-level detail that is not needed to be addressed in the formulated allocation problem. Once a set of services is accepted to be deployed on a host, then and only then the exact decomposition of the overall host computing capacity U_j will need to be detailed, in terms of the capacity of each processor or core. At that time also the whole computing power allocated to each of the deployed services $x^s_{i,\,j}$ will need to be detailed in terms of computing power to be allocated for the various VMs that, as a whole, will serve users requests on behalf of users from the considered host.

In order to allow the possibility of rejecting one or more services that are being examined at the same time, we introduce into the problem formulation the derivative Boolean variables $\left\{z^s_{i,n}\right\}$ with a value of 1 if the component $i \in \mathcal{S}^s$ is admitted on subnet n and 0 otherwise. These can be put in relationship with the $x^s_{i,\,j}$ variables through the following constraints:

$$\begin{cases} \sum_{j \in \mathcal{H}_n} x^s_{i,j} - \theta^s_i & \geq & K(z^s_{i,\,n} - 1) \\ \sum_{j \in \mathcal{H}_n} x^s_{i,j} & \leq & K z^s_{i,\,n} \\ y^s_{i,\,n} - b^s_i & \geq & K(z^s_{i,\,n} - 1) \\ y^s_{i,\,n} & \leq & K z^s_{i,\,n} \end{cases} \quad \forall i \in S^{(s)},\, \forall n \in \mathcal{N} \qquad (1)$$

where K is a sufficiently large constant. The above inequalities constrain the service component allocation variables $\left\{x^s_{i,\,j},\, y^s_{i,\,n}\right\}$ to give enough computing capacity and network bandwidth on the hosts of subnet n for the basic requirements of the component $\{\theta^s_i,\, b^s_i\}$ with $z^s_{i,\,n} = 1$, or alternatively they force them to be 0 with $z^s_{i,\,n} = 0$. Indeed, when $z^s_{i,\,n} = 1$, the first and the third inequality ensure that $\sum_{j \in \mathcal{H}_n} x^s_{i,j} \geq \theta^s_i$, and $y^s_{i,\,n} \geq b^s_i$, whereas the second and fourth inequalities impose no constraint on the values of $\left\{x^s_{i,\,j},\, y^s_{i,\,n}\right\}$ since K is a sufficiently large constant. On the other hand, when $z^s_{i,\,n} = 0$, the first and the third inequalities impose no constraint, whereas the second and fourth become $\sum_{j \in \mathcal{H}_n} x^s_{i,j} \leq 0$, and $y^s_{i,\,n} \leq 0$. Given that $x^s_{i,\,j}, y^s_{i,\,n} \in \mathbb{R}^+$, this forces $\left\{x^s_{i,\,j},\, y^s_{i,\,n}\right\}$ to become zero, thus the component is not given any computing and networking share on any of the available resources.

B. Partial admittance and federation

In case a service cannot be fully allocated in the underlying Cloud, the IP may want to consider the option of federating with other IPs, before rejecting the service, given that such an option is allowed by the SP. To accommodate this requirement, we introduce a set of variables that derive from the basic unknown variables that were introduced in the previous section, and are used in the formulation of the problem later on. These are:

- The Boolean variable $x^s_i \, : \, x^s_i = \bigvee_{n \in \mathcal{N}} z^s_{i,\,n}$, that becomes 1 when the component is allocated at least on one of the available subnets, or 0 otherwise.
- The Boolean variable $x^s : x^s = \prod_{i \in S^{(s)}} x^s_i$, that signals whether a service as a whole is accepted or not. Thus, if at least one of the components cannot be allocated on any of the available subnets, then $x^s = 0$.
- The partial admittance of a service can be expressed by the variable x'^s: $x'^s = \frac{\sum_{i \in S^s} x^s_i}{m^s}$, where m^s is the number of components that comprise service s. Therefore, $x'^s = 1$ means full admittance of service s, whereas for example $x'^s = 0.5$ means that half of the components that comprise s have been admitted.

C. Allocation Constraints

The allocation constraints for the problem are summarized as follows.

- The overall allocated computing capacity for each component ξ^s_i should not exceed the limit defined by its basic plus elastic computing capacity:

$$\forall s \in \mathcal{S},\, \forall i \in S^s,\, \sum_{j \in \mathcal{H}} x^s_{i,\,j} \leq \theta^s_i + \Theta^s_i. \qquad (2)$$

- The additional load imposed on each host cannot overcome their residual available computing capacity:

$$\forall j \in \mathcal{H}, \sum_{s \in \mathcal{S}} \sum_{i \in S^{(s)}} x^s_{i,\,j} \leq U_j. \qquad (3)$$

- The overall allocated network capacity on the subnets for each component ξ^s_i should not exceed the limit defined by its basic plus elastic network capacity:

$$\forall s \in \mathcal{S},\, \forall i \in S^s,\, \sum_{n \in \mathcal{N}} y^s_{i,\,n} \leq b^s_i + \mathcal{B}^s_i. \qquad (4)$$

- The additional load imposed on each subnet cannot overcome their residual available bandwidth capacity:

$$\forall n \in \mathcal{N}, \quad \sum_{s \,\in\, \mathcal{S}} \sum_{i \in S^s} y^s_{i,\,n} \leq W_n. \qquad (5)$$

D. Affinity rules

Apart from the basic allocation constraints as introduced in the previous section, different types of services and components, may require special treatment when it comes to their deployment. To this direction, an extra basic set of conditional affinity constraints can be added to the allocation problem depending on the specific requirements of the components and/or the services as a whole.

- **A component must be allocated in the same subnet:** the instances of the component must be deployed within the same subnet, which also implies that federation is not allowed. This can be achieved by adding the following constraint :

$$\sum_{n\in\mathcal{N}} z^s_{i,\,n} = x^s_i. \tag{6}$$

- **A component must be allocated in the same physical node:** the instances of the component must be allocated on the same physical node. To achieve this, we add the following constraint:

$$\sum_{j\in\mathcal{H}} h^s_{i,\,j} = x^s_i, \tag{7}$$

  where $h^s_{i,\,j}$ is a Boolean variable that becomes 1 if host j is used in the allocation of component i.
- **The service cannot be federated:** this means none of its components can be federated. To this direction the following constraint is added to the problem:

$$x'^s = x^s. \tag{8}$$

- **The service must be deployed in the same subnet:** all the instances of all the components of the service must be deployed in the same subnet. Works in conjunction with Eq. 6 on component level, with the addition of the following constraint:

$$\sum_{n\in\mathcal{N}} \delta^s_n = x^s, \tag{9}$$

  where δ^s_n is a Boolean variable that becomes 1 if subnet n is used in the allocation of the service s.
- **The service must be deployed in the same physical node:** all the instances of all components of the service must be deployed in the same physical node. Works in conjunction with Eq. 7 on component level, with the addition of the following constraint:

$$\sum_{j\in\mathcal{H}} h^s_j = x^s, \tag{10}$$

  where h^s_j is a Boolean variable that becomes 1 if host j is used in the allocation of service s.

The Boolean variables $h^s_{i,\,j}$, $\delta^s_n$and h^s_j are derivative variables that can be computed by forming logical constraints around the basic variables of the problem. For example, the h^s_j Boolean variable that states whether or not the host $j \in \mathcal{H}$ is used in any possible allocation, can be encoded using the $x^{(s)}_{i,\,j}$ variables as follows:

$$\left\{ \begin{array}{lc} \sum_{i\in\mathcal{S}^s} x^s_{i,\,j} \geq & U_j\left(h^s_j - 1\right) + \epsilon \\ \sum_{i\in\mathcal{S}^s} x^s_{i,\,j} \leq & U_j \cdot h^s_j \end{array} \right. , \forall j \in \mathcal{H} \tag{11}$$

where ϵ is a sufficiently small constant and U_j the available computing capacity of the host j (see Section III-B). This pair of inequalities, given that $x^s_{i,\,j}$ are non-negative variables, constrains the sum of the $x^s_{i,\,j}$ variables pertaining to host j to be all 0 with $h^s_j = 0$, or to hold a strictly positive value with $h^s_j = 1$. The ϵ value may be also used to impose the minimum workload for which it is worth turning on a host, or otherwise it is preferred not to use it (independently of other cost metrics possibly in place). The Boolean variables $h^s_{i,\,j}$, and δ^s_n are calculated in a similar manner.

E. Probabilistic Elasticity

In this section we propose a probabilistic approach to the problem of allocating extra resources for elasticity reasons. The basis of this approach lies in the existence of prediction models that are able to forecast resource usages based on historical monitoring data. Indeed, there are several works to this direction in the literature, such as [11], that produce statistical information in the form of probability distributions of the actual resource requirements experienced at run-time by a service in a virtualized environment by monitoring previous runs of the service. Therefore, given that this statistical knowledge is known, then it can be leveraged inside the problem to tune the allocation in such a way that the service can run flawlessly with at least a minimum probability ϕ^s, which is the minimum probability that there will be sufficient resources for the activation of the VMs when needed as expressed in the SLA (see Section III-D).

To this end, it is assumed that the IP has knowledge about the probability that a given component may use a computing capacity up to x_i and a network capacity up to y_i. This can be formally described by the joint cumulative distribution function $\mathcal{F}^s_i(x_i,\, y_i) = P\left[X^s_i \leq x_i,\, Y^s_i \leq y_i\right]$ of the real-valued random variables X^s_i and Y^s_i representing the computational and network capacity that a component i may require at runtime. In order to deal simultaneously with all the components that comprise the service, this can be generalized for X^s_i, Y^s_i, $\forall i \in \{1, \ldots, m^s\}$: $\mathcal{F}^s(x_1, .., x_{m^s}, y_1, .., y_{m^s})$, where m^s is the number of the components of service s.

Then, in order for a service s to be admitted into the system, instead of reserving resources for the maximum amount of elasticity requirements deterministically, it is sufficient to guarantee that the probability for s to find enough available computing power and network bandwidth when actually required, denoted from this point on as Φ^s, will be higher than or equal to ϕ^s:

$$\Phi^s \geq \phi^s. \tag{12}$$

The insertion of the constraint as shown above allows to reduce the resources that need to be booked for elasticity reasons when accepting the service, by exploiting statistical information, as demonstrated in Section V-B. It should be noted that, if $\phi^s = 1$, then the deterministic case is obtained as a particular case of the probabilistic one, i.e. the model will operate deterministically if the client asks for 100% guarantees in the SLA.

A simplification of Eq. 12 may be obtained if the distributions $\mathcal{F}^s_i$ are independent, i.e.: $\mathcal{F}^s(x_1, .., x_{m^s}, y_1, .., y_{m^s}) = \prod_{i\in\mathcal{S}^s} \mathcal{F}^s_i(x_i,\, y_i)$, and by further assuming that there is a linear dependence between the computing and networking requirements $y_i : y_i = ax_i + b$. In such a case, the probability Φ^s becomes:

$$\begin{array}{rcl} \Phi^s & = & \prod_{i\in\mathcal{S}^s} P\left[X_i < min(x_i,\, \frac{y_i}{a} - b)\right] \\ & = & \prod_{i\in\mathcal{S}^s} \mathcal{F}^s_i(min(x_i,\, \frac{y_i}{a} - b)). \end{array} \tag{13}$$

In order to allow partial admittance of a service and possible federation with another IP, the above can be rewritten as follows:

$$\Phi^s = \prod_{i \in \mathcal{S}^s} 1 - \left[1 - \mathcal{F}_i^s(min(x_i, \frac{y_i}{a} - b))\right] \cdot x_i^s, \tag{14}$$

where x_i^s is the Boolean variable that becomes 1 when the component i is allocated (see Section IV-B).

F. Objective Function

The formalized admission control problem needs to be solved by optimizing proper metrics that express the goodness of the found allocation. As already explained, the maximum required capacity, including the elastic capacity, may be quite large compared to the basic requirements for the service, thus the elastic capacity plays a significant role in the total requirements and the cost for hosting the service. Clearly, from a provider perspective, a metric for the goodness of an allocation solution may be significant, because of the additional costs possibly needed to admit the new services. In order to formalize this, let us introduce the extra cost of ζ_j associated with turning on an unused host $j \in \mathcal{H}_{off} \subset \mathcal{H}$. Then, a simple term to consider in the objective function is the total additional cost $\mathcal{C}$ associated with turning on unused hosts:

$$\mathcal{C} = \sum_{j \in \mathcal{H}_{off}} h_j \cdot \zeta_j. \tag{15}$$

Furthermore, it is in the IP's best interest to extend the optimization goal so that other available host-level information is considered, such as the eco-efficiency of the hosts E_j (see Section III-D). To this direction, the overall objective is complemented by the term $\mathcal{E}$ that expresses the eco-efficiency score of the hosts that are used to form the allocation pattern:

$$\mathcal{E} = \sum_{j \in \mathcal{H}} h_j \cdot E_j. \tag{16}$$

In the overall assessment process, apart from host-specific factors, the IP may want to consider also factors related to the SPs that own the services, like Trust $\mathcal{T}^s$ (see Section III-D). The same reasoning may easily be applied to any other host-level or SP-level information that is available and of importance to the IP.

Additionally, the probabilistic framework as introduced in Section IV-E implies that with a maximum probability of $\overline{\Phi^s} \triangleq 1 - \Phi^s$, an admitted service is not expected to find the needed extra resources available, leading to the necessity to pay the penalty P^s back to the customer. Therefore, for each service that is partially admitted into the system ($x'^s \leq 1$, see Section IV-B), the expected penalty due to SLA violations $\mathcal{P}^s = \overline{\Phi^s} P^s$, should be subtracted from the immediate gain $\mathcal{G}^s$.

By taking into account all the different objectives mentioned above, we finally obtain the following multi-objective function:

$$\max \textstyle\sum_{s \in \mathcal{S}} x'^s (w_\mathcal{G} \mathcal{G}^s - w_\mathcal{P} \mathcal{P}^s + w_\mathcal{T} \mathcal{T}^s) - w_\mathcal{C} \mathcal{C} + w_\mathcal{E} \mathcal{E}, \tag{17}$$

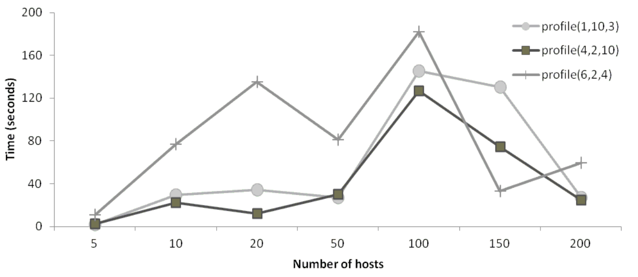

Figure 1. CPU times for computing the optimal solution using BARON.

where $w_\mathcal{G}$, $w_\mathcal{P}$, $w_\mathcal{T}$, $w_\mathcal{C}$, and $w_\mathcal{E}$ are used as weights for configuring the relative importance of the different factors in the multi-objective function and for adapting the heterogeneous quantities in the sum. By changing the values of these weights, the IP can customize the overall service acceptance policy according to its needs. For example, a profit-driven policy would be expressed by setting the values of the weights $w_\mathcal{E}$ and $w_\mathcal{T}$ equal to zero. In this way, the trust and eco-efficiency aspect of the solution will not be taken into account during the optimization process, and the optimal solution in this case will be the cheapest one. Different acceptance policies can be applied by configuring these weights as demonstrated in detail in the results presented in Section V-B.

V. EVALUATION

The problem formalized in the previous section falls within the class of Mixed-Integer Non-Linear Programming (MINLP) optimization problems. It was modeled on the General Algebraic Modeling System (GAMS) [1], a high-level modeling system for mathematical programming of complex, large scale optimization problems. For solving the problem, we used the Branch and Reduce Optimization Navigator (BARON) [13], which is a computational system for solving non-convex MINLP optimization problems to global optimality. All results presented in the analysis that follows have been obtained using BARON with GAMS v23.7 on AMD FX™ Six-Core 3.32 GHz processor with 8 GB of RAM.

A. Performance evaluation

In what follows, we present a performance evaluation of the optimization model discussed in the previous section. To this direction, different model profiles were generated for different number of services, components and subnets. The requirements of the components and the capacity of the subnets were generated so that they have constant values.

The BARON solver that, as mentioned earlier yields the optimal resource allocation, was then used to solve the GAMS models and the running times for each model profile were noted. Several runs were repeated and the average performance was computed across these repeated experiments to increase the accuracy of the results.

Figure 1 plots the CPU time consumed by BARON to find the optimal solution for each of the examined problems of different sizes. The number of hosts to be considered as

Table I
UNUSED HOSTS AND SERVICES CHARACTERISTICS

$j \in \mathcal{H}$	ζ_j	E_j	$n \in \mathcal{N}$
$j_1 - j_{10}$	1	1	n_1
$j_{11} - j_{20}$	10	10	n_2

S	s_1	s_2
$\mathcal{G}^s$	300	500
$\mathcal{P}^s$	30	50
ϕ^s	0.7	0.8
$\mathcal{T}^s$	50	10

Table II
INDICATIVE CASES UNDER EXAMINATION

Case	$w_{\mathcal{G}}$	$w_{\mathcal{C}}$	$w_{\mathcal{P}}$	$w_{\mathcal{E}}$	$w_{\mathcal{T}}$
I	1.667	1.667	1.667	0	0
II	1.5	1.5	2	0	0
III	0.5	0.5	4	0	0
IV	0.5	0.1	4	0.4	0
V	2	1	0	0	2
VI	1	1	0	0	3

candidates for allocation is plotted along the x-axis reaching a maximum of 200, whereas the CPU time consumed is plotted along the y-axis. In order to stress the problem, the services had no conditional deployment constraint and all the hosts were unused and of the same computing capacity. The three lines that are plotted correspond to three different indicative profiles of the model. For example, profile(4,2,10) corresponds to a model that considers 4 services with 2 components each on 10 underlying subnets respectively.

As it can be seen, the number of services has a more negative effect on the solution times as compared to the number of the components that comprise the services. Interestingly, the results also indicate that increasing the number of subnets to which the hosts belong to, reduces the time that is needed to compute the optimal solution. Although the consumed times vary due to the inherent non-convexity of the problem, the results prove that the problem is tractable, and indicate that the cost of an accurate solution may be considered acceptable even for medium-sized problems. For larger-sized problems, a heuristic method for yielding a near-optimal solution at a fraction of the cost of the optimal solution is being developed, following approaches similar to [6].

B. Case study

In this section, we highlight the way the flexibility that is introduced by the probabilistic framework, as described in Section IV-E, is regulated by the weights of the different factors in the multi-objective function, and the way changing these weights results in different solution patterns in terms of the cost, the allocated capacity, the hosts to be used, etc. To this direction, we consider an indicative case study of 2 different subnets $\{n_1, n_2\}$ of the same capacity $(100\, Mb/s)$. Each subnet interconnects 10 unused hosts of the same capacity but with different usage costs ζ_j and eco-efficiency scores E_j, as shown in Table I. For simplicity reasons, the CPU cores of the hosts are considered to be homogeneous, i.e. having the same ISA, speed and capabilities, and a capacity of $U_j = 2.0\, GHz$. Therefore the total computing capacity of the available hosts is $40\, GHz$.

Under these settings, we consider 2 services $\{s_1, s_2\}$ requesting admission with the parameters shown in Table I. Both services consist of 2 components $\{\xi_i^s\}$, that have the same basic and elastic computing and network capacity requirements: $\theta_i^s = 1\, GHz$, $\Theta_i^s = 9\, GHz$, and $b_i^s = 100\, bytes/s$, $\mathcal{B}_i^s = 1\, Mb/s$. Therefore the maximum elastic computing and network capacity that can be accepted for each service is 18 GHz and 2 Mb/s respectively. For simplicity and without loss of generality, we consider that the probability distributions of the components capacity requirements x_i are independent and uniformly distributed in the interval $[\theta_i^s, \theta_i^s + \Theta_i^s]$, and that there is a linear dependence with the network capacity requirements. Then, the probability Φ^s is given by Eq. .13. The values of the parameter ϕ^s, which is the minimum probability that the required elastic capacity will be actually available when needed, are less than 1 (see Table I), and are kept fixed for all cases under examination. Lastly, we consider that the services must be deployed in the same Cloud, i.e. none of their components can be considered as candidates for federation in case of partial acceptance.

Six indicative cases are examined with each of them having different IP acceptance policies as expressed by the different values of the five weights that are involved in the objective function (see Table II. The sum of the five weights is equal to 5 in all cases, and BARON was used to solve the GAMS models that correspond to these cases. The obtained solutions are summarized in Table III.

In Case I, the only non-zero weights are $w_{\mathcal{G}}$, $w_{\mathcal{C}}$, and $w_{\mathcal{P}}$ whereas the rest of the weights are set to 0, denoting a profit-driven IP. According to the BARON output, the optimal solution distributes the instances of two services on 18 hosts (2 remain unoccupied), whereas the elastic requirements that are being accepted, are compressed to the minimum allowed by the probabilistic constraint of Eq. 12, with service s_2 being granted a larger amount since $\phi^{s_2} > \phi^{s_1}$ (see Table III, Case I). Furthermore, the two hosts that remain unoccupied belong to subnet n_2, which interconnects hosts of higher cost compared to the ones that belong to subnet n_1.

For Case II, in which the weight $w_{\mathcal{P}}$ is increased, meaning that the IP wants to degrease the risk of paying penalties to the clients and become more sensitive in terms of the QoS that is offered, the optimal allocation pattern now includes one of the two hosts that were kept unoccupied in the previous case, which is now turned on for hosting more elastic capacity. Further increasing $w_{\mathcal{P}}$ as in Case III, leads to the acceptance of the maximum amount of elastic requirements and all the hosts being turned on (see Table III, Cases II and III).

The following case, Case IV, is similar to the first one with the only difference being that the eco-efficiency weight $w_{\mathcal{E}}$, which was set to zero previously, is now increased compared to the cost weight $w_{\mathcal{C}}$. Therefore, the eco-efficiency of the available hosts is now taken into consideration during the optimization process. Interestingly, the optimal allocation pattern now involves turning on more hosts in subnet n_2 instead of n_1, and the overall eco-score of the allocation pattern is

Table III
COMPARISON OF DIFFERENT CASES

Case	I		II		III		IV		V	VI
Accepted services	s_1	s_2	s_1	s_2	s_1	s_2	s_1	s_2	s_2	s_1
Involved hosts	18		19		20		18		10	10
Elastic capacity	15	16	16	18	18	18	15	16	10	10
Gain-Cost $\mathcal{G} - C$	710		700		690		692		490	290
Eco-efficiency $\mathcal{E}$	90		100		110		108		50	50
Expected penalty $\mathcal{P}$	18.4		5.7		0		18.4		0	0
Trust $\mathcal{T}$	60		60		60		60		10	50
Solution time (secs)	4.656		7.211		6.776		10.456		2.034	2.734

increased, whereas the gain is decreased (see Table III, Case IV). This is due to the fact that the hosts of subnet n_2 although more expensive and equivalent to the ones of n_1 in terms of computing power, they are characterized by a higher eco-efficiency score.

The effect of SP related factors in the optimization process becomes evident in the last 2 cases under examination (Case V and VI) in which the only non-zero weights are the ones related to the gain of admitting a service $w_{\mathcal{G}}$, $w_{\mathcal{C}}$ and the trust $w_{\mathcal{T}}$. In order to create a competing situation between the 2 services that request admittance, the maximum elastic requirements for each component which were set to $\Theta_i^s = 9$ in the previous cases, are increased to 10, and the values of the probability that the resources for elasticity will be available are now set to 1 ($\phi^{s_1} = \phi^{s_2} = 1$). Now, the overall capacity requirements of the services become 42 GHz, exceeding the available capacity of the hosts, which will lead to one of the services being rejected under deterministic admission and given that partial admittance, i.e. federation, is not allowed. For Case V, in which $w_{\mathcal{G}} = w_{\mathcal{T}}$, the service that is being admitted is s_2, whereas s_1 which is the less profitable of the two, is rejected. By setting $w_T = 3$ (Case VI), the trustworthiness of the SPs outweighs the gain of the services, leading to the acceptance of s_1 even though it is the less profitable of the two services. It should be noted that the values of the presented weights were chosen to highlight the shift in the optimization objective and that for the in-between values of the weights, the solutions remained unchanged (not shown due to space constraints).

VI. CONCLUSIONS

This paper presented a probabilistic approach to the problem of optimum allocation of services on virtualized physical resources, when horizontal elasticity is required. The formulated optimization model constitutes a probabilistic admission control test. It exploits statistical knowledge about the elastic workload requirements of horizontally scalable services for reducing the correspondingly booked resources. The proposed model also allows for proper trade-offs among business level objectives such as trust, and eco-efficiency, and also takes into account extra affinity constraints the services might have. In order to provide a strong assessment of the effectiveness of the proposed technique, the problem was modeled on GAMS and solved under realistic provider's settings. Future work is focused on developing a heuristic method for reducing the computation time for solving the presented problem.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7 under grant agreement n.257115 OPTIMIS – Optimized Infrastructure Services and n.248465 S(o)OS – Service-oriented Operating Systems.

REFERENCES

[1] General Algebraic Modeling system (GAMS). GAMS Development Corporation. Available at http://www.gams.com/.

[2] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, May 2011.

[3] Jing Bi, Zhiliang Zhu, Ruixiong Tian, and Qingbo Wang. Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 370 –377, july 2010.

[4] F. Chang, J. Ren, and R. Viswanathan. Optimal resource allocation in clouds. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 418 –425, july 2010.

[5] T. Cucinotta. Optimum scalability point for parallelisable real-time components. In *Proceedings of the International Workshop on Synthesis and Optimization Methods for Real-time and Embedded Systems (SOMRES 2011)*, Vienna, Austria, November 2011.

[6] T. Cucinotta and G. Anastasi. A heuristic for optimum allocation of real-time service workflows. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, Irvine, USA, December 2011.

[7] Cucinotta et al. Virtualised e-learning on the irmos real-time cloud. *Service Oriented Computing and Applications*, pages 1–16. 10.1007/s11761-011-0089-4.

[8] Ferrer et al. OPTIMIS: A holistic approach to cloud service provisioning. *Future Generation Computer Systems*, 28(1):66 – 77, 2012.

[9] K. Konstanteli, T. Cucinotta, and T. Varvarigou. Optimum allocation of distributed service workflows with probabilistic real-time guarantees. *Springer Service Oriented Computing and Applications*, 4(4):229–243, 10 2010.

[10] K. Konstanteli, T. Cucinotta, and T. Varvarigou. Probabilistic admission control for elastic cloud computing. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, Irvine, USA, December 2011.

[11] S. Mallick. Virtualization based cloud capacity prediction. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 849 –852, july 2011.

[12] M. Mazzucco, D. Dyachuk, and R. Deters. Maximizing cloud providers' revenues via energy aware allocation policies. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 131 –138, july 2010.

[13] N. V. Sahinidis. Global optimization and constraint satisfaction: The branch-and-reduce approach. *C. Bliek, C. Jermann, and A. Neumaier (eds.), Lecture Notes in Computer Science, Springer, Berlin*, 2861:1–16, 2003.

[14] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 228 –235, july 2010.

[15] Zhenzhong Zhang, Haiyan Wang, Limin Xiao, and Li Ruan. A statistical based resource allocation scheme in cloud. In *Cloud and Service Computing (CSC), 2011 International Conference on*, pages 266 –273, dec. 2011.

[16] H. Zheng, J. Yang, and W. Zhao. QoS probability distribution estimation for web services and service compositions. In *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 1 –8, December 2010.

[17] Qian Zhu and Gagan Agrawal. Resource provisioning with budget constraints for adaptive applications in cloud environments. *IEEE Transactions on Services Computing*, 99(PrePrints), 2011.