Business-Driven Optimization of Component Placement for Complex Services in Federated Clouds

Genady Ya. Grabarnik
Dept. of Math & CS
St. John's University
Queens, NY, USA
Email: *grabarng@stjohns.edu*

Larisa Shwartz
Operational Innovation
IBM TJ Watson Research Center
NY, USA
Email: *lshwart@us.ibm.com*

Mauro Tortonesi
Dept. of Engineering
University of Ferrara
Ferrara, Italy
Email: *mauro.tortonesi@unife.it*

***Abstract*—With the advent of connected services ecosystems, new generations of services and systems are being conceived, responding to the ever growing demands of the market place. In parallel, the effective adoption of the Cloud computing paradigm is becoming an essential enabler for business enterprises. With such importance placed on the services ecosystem, the design and management of services becomes a key issue both for the providers and the users. One of the main challenges for service providers lies in the complexity of the services, comprising of a multiplicity of technologies and competing and cooperating providers, which is difficult to address through current technology-centric service design approaches, in particular for the deployment infrastructure. The work described in this paper lays a foundation for business driven service design by proposing a business goals driven model of resource allocation in the Cloud. We define a goal/loss/processing cost function for resource allocation that we optimize, while taking into account the dynamic and varying nature of requests load.**

***Index Terms*—Resource allocation, Cloud, business processes**

I. Introduction

The technical foundations behind connected services, such as the adoption of standards and a paradigm shift to virtualized systems, are merely an instantiation, a point solution, to satisfy the requirements laid out by overwhelming forces in the business world to satisfy the competitive pressures that come from a global economy with much faster business cycles and broader outreach than in past years. Chief among these technical forces are the heterogeneity and the distribution of services, which are in many ways directly related to economic and business environment factors (mergers and acquisitions, divestitures, enterprise quickly moving into new business areas). The days of a single vendor and a centralized environment as the predominant computing paradigm are numbered. In a connected services paradigm the services are exposed and consumed dynamically in accordance to agreements between service participants, which involve business-focused performance constraints.

The access to relatively inexpensive virtualized resources and the availability of highly and seamlessly reconfigurable options provide a compelling economic motivation for redeploying existing services on the Cloud to better suit consumer needs. However, to take full advantage of the Cloud, service providers need to consider redesigning services to benefit from *dynamic service placement*. One of the most compelling reasons for pursuing the transition to the Cloud is the desire to create a competitive business advantage for an organization. The rapid pace of new business needs, the needs of making informed decisions more rapidly require an ability to quantify and maximize a business value that is unattainable in the current environment.

Deploying or migrating complex services to the Cloud is always a very challenging task. Service providers have to deal with complex IT architectures implementing multiple business processes on top of a plethora of software components with non-trivial dependencies and interactions [1].

Federated Cloud environments or, more in general, Cloud-based solutions that span across several Cloud data centers further exhacerbate these issues. In these environments, service providers have the opportunity to deploy parts of their IT infrastructure to different Cloud data centers, in order to benefit from competitive pricing or to reduce communication latencies by deploying software components in (physical) proximity to the customer's location. However, while exploiting several data centers, either from a single Cloud provider or from different Cloud providers, can present interesting opportunities for the service providers, it also significantly complicates the problem of finding the most efficient architecture for Cloud services.

This paper presents a business driven model and a simulation-based tool for the dynamic placement of complex services in federated Clouds. The general approach is to get a formal description of the resources necessary to satisfy the stream of service requests and to apply resource cost and possible penalties for the SLO breaches to get goal/loss/processing cost function, which we optimize to find the best resource allocation. By considering the dynamic and periodic nature of requests loads, our tool enables the predictive allocation of service components and resources in federated Cloud environments.

978-1-4799-0913-1/14/$31.00

The rest of the paper is organized as follows: Section II provides a motivating example of a business scenario. Section III describes the problem settings and introduces the main notations used in the paper. Section IV presents our algorithmic work. Section V presents our developed framework and system for business driven Cloud based service design. In Section VI, we present our empirical studies. Section VII summarizes the related work. Section VIII offers concluding remarks and future work considerations.

II. Component Placement in Federated Clouds

Let us consider a Cloud IT service implementing the online purchasing function for a very large enterprise customer with a global presence. The service implements 3 different business processes, respectively for purchasing an item, for browsing all the items available for purchasing, and for searching for a specific item. The business processes are depicted in Fig. 1. In the purchase business process, a customer request goes through several software components: a Web server and an Application Server that implement the Web interface and the business logic of the service, a Financial Transaction Server that deals with payment requests, and finally a Persistence Storage that records purchasing information for accounting and billing purposes. The other two business processes are simpler, and both dealing with a single step after the request visits the Web Server and the Application Server. [1]

Even this relatively simple example is capable of highlighting the difficulties that emerge when deploying an IT service in federated Cloud environments. First, each of the software components that implement the IT service usually has minimum performance requirements that result in constraints on the types of Virtual Machines (VMs) that can be adopted for their instantiation. For instance, Search Servers might require generously sized VMs for their instantiation while Web Servers might be able to run even on less powerful VMs. Also, several less resource-hungry components could be co-instantiated on a single VM. In addition, each component usually needs multiple instantiations on a number of VMs according to the request load of the corresponding business processes. In fact, the type and number of service components allocated to each data center, as well as the type of VMs used for their instantiation, is likely to have a considerable impact on the whole IT service performance. This is especially true in case of shared software components, that are used for the implementation of several business processes and that could have a severe impact at the entire system level if not correctly sized.

The elastic nature of Cloud computing also enables and suggests to tailor the number of service components instantiated according to the request workload generated by customers. A dynamic service resizing function would allow service providers to instantiate additional components to respond to higher loads and to de-instantiate them when they are not needed anymore in order to reduce VM bills.

Finally, the realization of complex Cloud services calls for an accurate evaluation of IT costs. For the Cloud approach to be cost efficient, there is the need to consider not only IT performance metrics and VM bills, but also to evaluate business-related Key Performance Indicators (KPIs) and their full impact on the service provider's business. In fact, while IT service configuration that instantiate a large number of redundant components might lead to high VM bills, configurations with an excessively low number of components or with components instantiated in underpowered VMs might lead to Service Level Objective (SLO) violations that could result in costly penalties for the service provider.

Service providers would significantly benefit from service component placement optimization tools capable of exploring the space of possible IT service configurations in federated Cloud environments to find the best possible one, thus eliminating the need for difficult and time-consuming trial-and-error optimization. The development of these tools, however, is indeed very challenging, as it calls for a service model that is general enough to consider all the main players in Cloud computing providing and for advanced simulation-based *what-if scenario analysis* approaches that are capable of dealing with the the significant complexity of the problem domain. Finally, since the customers request workload often follows daily and weekly patterns that are relatively easy to predict and to synthetically reproduce [2], there is the opportunity to explore predictive approaches that are capable of anticipating load changes and of suggesting new IT service configurations that could better respond to the new demand levels.

III. Model description

We decided to focus on the IaaS model for Cloud services, and thus to consider the basic unit of resource allocation as a VM. Cloud providers enable to instantiate several types of VMs, that will typically differ for resource consumption (CPU, storage, etc.) and cost. The model we are using covers a number of the aspects connecting business processes, IT services, IT resource consumption and IT resource costing description.

First, 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 also assume that the basic unit of load on the data center is a request for a workflow. Modeling workflows as the sequence of service components that a request has to be processed by in the corresponding business process enables to consider complex services, to fully capture the relationships between service components, and to measure the impact of a component reallocation to a different data center on the whole service performance.

[1] In order not to unnecessarily complicate the illustrative example, we did not consider caching, authentication, or syncronization between, e.g., different persistence storage components in the business processes.

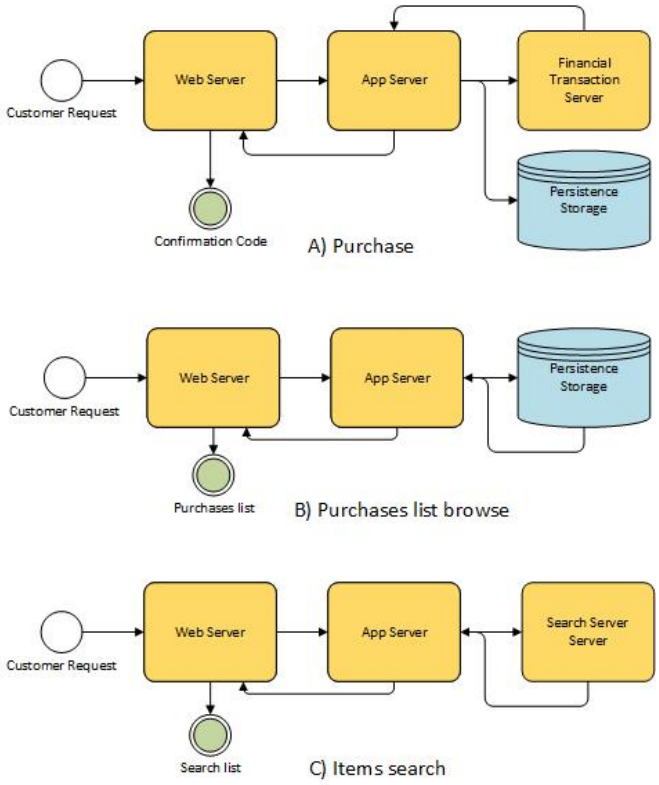

Fig. 1. Purchase and Lookup BP

In order to develop a generically applicable model of IT resources costing we analyzed the pricing schemes of several Cloud providers and created a model able to handle most of them. For instance, a portion of the pricing schemes for Google and Amazon EC2 is shown in tables I II, and III [3] [4] [5].

We do not consider, instead, traffic related costs, as we assume that service providers leveraging federated Cloud architectures will configure their IT infrastructure to always serve a customer's requests from the closest data center. This makes traffic costs an issue of lesser importance, that we plan to address in future works.

TABLE I
GOOGLE VM INSTANCE TYPES AND PRICING.

Instance type	Virtual Cores	Memory	Local disk	Price $/ Hour US	Price $/Hour Europe
standard-4-d	4	15GB	1770GB	$0.53	$0.58
standard-8-d	8	30GB	2x1770GB	$1.06	$1.16
highmem-4-d	4	26GB	1770GB	$0.61	$0.69
highmem-8-d	8	52GB	2x1770GB	$1.22	$1.38

We assume that each customer's request is fulfilled by a business process, instantiated as an IT process. A (simplified) business process is a Directed Acyclic Graph (DAG) consisting of nodes corresponding to the components of the process, and directed edges corresponding to either passing parameters or data or messages. Note that although actual call graphs (see Fig. 1) are not DAGs, we can always assume that each call

TABLE II
AMAZON VM INSTANCE TYPES.

Optimized	Type	Processor	v CPU	ECU	RAM (GB)	Storage (GB)	Net-work
General	m1.large	64-bit	2	4	7.5	2 x 420	Medium
General	m1.xlarge	64-bit	4	8	15	4 x 420	High
Compute	c1.medium	32/64	2	5	1.7	1 x 350	Medium
Compute	c1.xlarge	64-bit	8	20	7	4 x 420	High
Memory	m2.xlarge	64-bit	2	6.5	17.1	1 x 420	Medium
Memory	m2.2xlarge	64-bit	4	13	34.2	1 x 850	Medium
Memory	cr1.8xlarge	64-bit	32	88	244	2x120SSD	10 Gb/s
Storage	hi1.4xlarge	64-bit	16	35	60.5	2xTB SSD	10 Gb/s

TABLE III
AMAZON VM PRICING.

Standard On-Demand Instances			
	US East (N. Virginia)	**EU (Ireland)**	**Asia Pacific (Singapur)**
Small (Default)	$0.060 per Hour	$0.065 per Hour	$0.080 per Hour
Medium	$0.120 per Hour	$0.130 per Hour	$0.160 per Hour
Large	$0.240 per Hour	$0.260 per Hour	$0.320 per Hour
Extra Large	$0.480 per Hour	$0.520 per Hour	$0.640 per Hour
Second Generation Standard On-Demand Instances			
Extra Large	$0.500 per Hour	$0.550 per Hour	$0.700 per Hour
Double Extra Large	$1.000 per Hour	$1.100 per Hour	$1.400 per Hour

returns result of the call and some error code, what make call graph close by properties to DAGs.

One or many edges v_{ij} may start from a node n_i and end in the node n_j. That means that data produced by the node is multiplied and split between nodes n_j accepting edges. One or many edges v_{ij} may end in a node n_j. We assume in this case that data is either joined (all edges should provide data) or merged (at least one edge must provide data). More advanced business process models are described by the BPEL [6] industry standard, but for our purpose this simplified model represents a good approximation of a business process.

A. IT costing model

We suppose that the main available resources are in the form of VMs that have CPUs, RAM, storage. An additional resource that have impact on the Business value is networking. VMs have predefined types that depend from the specific Cloud provider, say *standard-8-d* in Table I. Each VM has a location (possibly distributed over multiple datacenters all over the world) and price of the VM may depend on it.

Processing begins from some initial state-placement of the components. This allows us to run application in cyclic or online manner. Load monitor monitors current load and time since last recalculation, and, when predicted or actual level require, load monitor initiates recalculation of VM placement. This in turn may initiate modification of placement.

We assume that copies of all components will be preloaded in a few versions of VMs. So, in order to be placed, VM sometimes should be copied in appropriate location and started. Note that boot time of a VM is typically low, thus allowing us fast change of a placement.

The goal (or loss) function that we consider is a sum of the required resources costs and cost associated with SLO breaches. The main components of the goal function are:

- Cost of the placement (or deployment) of the required components,
- Cost of the relocating some of the components,
- Cost of the SLO breaches.

Overall load on the model is created by randomly generating a few possible customer requests that are emulated by different workflows. For examples of the workflows see Fig. 1.

A number of papers considered statistical models for load generation, we give a short overview of the load generation modeling papers in section VII.

Each type component has a threshold for a number of requests, possibly dependent on type of VM. If number of requests to the component exceeds threshold then respond time for those requests exceeds time specified by SLO. If number of such requests is significant (say exceeds 2% of total number of requests), then we observe SLO breach with consequent penalty from customer. In order to prevent SLO breaches, new components are allocated upon demand.

We now consider each part of the goal function separately.

B. IT resource model and consumption

We consider a number of different types of load in mathematical model; for the experimentation purpose we restrict ourselves to 3 types of load. Load is regulated by requests, with number of requests varying leading to different levels of loads. The high CPU and memory, low storage and networking type could model calculation components. The low CPU and memory, low storage and high networking type could model Web server components. The high CPU and memory, high storage and networking type could model database components.

We assume that each VM is characterized by n_0 types of resources r_{kn}, and there are n_1 types of VMs V_k with $k = 1, .., n_1$, each type of VM has its size of available resources.

Example 1: The Amazon pricing table III shows 4 types of resources, CPU, RAM, Storage and Network capabilities. Based on different resource configurations the table shows 8 different types of VM instances.

A number of constraints in the original optimization problem are related to IT resources. Each workflow component should be executed on one VM, and this VM instance should have sufficient amount of resources. If T_{ji} is the i-th component of the j-th type of service request, and it requires r_{jin} amount or size of the n-th resource, then for the k-th type of VM instance to be able to instantiate the component it requires the following constrain to be satisfied

$$r_{kn} \geq r_{ijn}. \tag{1}$$

C. SLO model

We consider an SLO model which is well integrated with the rest of the framework. When the number of service requests that are not fulfilled in the predefined time exceeds certain threshold, service provider pays a fine.

Example 2: If percent of service requests executed in more that 10 seconds exceeds 1 percent, Then Service Provider pays a fine (or reduces service charge) of 5000$

Thus SLO loss may be expressed as a characteristic function of the set $ratio \geq t_{ijk}\ \chi_{[r,\infty)}$(ratio of requests over threshold), where the threshold t_{ijk} depends on component i, request j and VM type k.

D. Formulation of the optimization problem

The problem of finding the optimal next placement may be formulated as follows:

$$\begin{aligned} y^* &= \arg\min_{y \in \mathcal{P}} C(y, y_0, t_t) \\ &\text{subject to } r_{kn} \geq r_{ijn}, \end{aligned} \tag{2}$$

for all $i = 1 \ldots m, k = 1 \ldots, n = 1 \ldots$, where $\{\mathcal{P}\}$ is a set of all possible placements, y_0 is given initial placement t_t is the parameter describing requests (including predicted) with possible SLO breach for the next period.

The cost function $C(y, y_0, t_t) = C_1(y) + C_2(y, y_0) + C_3(y, t_t)$ is represented as a sum of cost of placement y, cost $C_2(y, y_0)$ of re-placement from placement y_0 to placement y, and expected SLO cost $C_3(y, t_t)$.

IV. Algorithms

For the service component placement optimization, we have adopted a 2-phase metaheuristics that explores the space of possible configurations to find the best performing one.

More specifically, the metaheuristics implements an outer phase that takes care of assigning service components to Cloud data centers, using a *genetic algorithm*, and an inner phase that takes care of choosing which VM types should be used to host each of the service components, using a *random search* strategy.

The design of the metaheuristics that we adopted is inspired to *memetic algorithms*, a common form of metaheuristics, based on the 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].

A. Assigning service components to Cloud data centers

The global search part of the metaheuristics takes care of assigning service component to each data center. This represents a search on a very large vector space, i.e., a subset of D^C, where D is the number of different data centers and C the number of components of the different service types that we consider.

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.

In particular, we have chosen to use a genetic algorithm for the global search procedure. Genetic algorithms are a family of population-based metaheuristics that iteratively refine a pool of candidate solutions to find the best suited one, borrowing concepts and techniques from natural evolution theory, such as fitness to the environment, selection, and inheritance, mutation, and recombination of genetic material [7].

Genetic algorithms have several advantages that make them particularly well suited for this kind of problems. In fact, they are a robust metaheuristics that can be adopted for the optimization of challenging objective functions. In addition, genetic algorithms lend themselves well for the integration with other metaheuristics to realize memetic algorithms, and can be tuned to favor exploration or exploitation parts of the search process [8]. Finally, population-based metaheuristics are (relatively) easily parallelizable, thus enabling (and suggesting) the adoption of in Cloud architectures for their implementation.

Algorithm 1 presents the global search procedure based on genetic algorithms that we have adopted. Our particular genetic algorithm implementation is based on a integer vector representation for the genotype (the position in the search space that in the genetic algorithms metaphor represents the genetic material of a specific individual), on binary tournament selection, intermediate recombination, and random geometrically-distributed mutations. Notice that the actual fitness evaluation, performed in line 7, is demanded to the local search procedure.

Algorithm 1 Find optimal service placement

```
1:  procedure OPTIM(conf)
2:      P ← random_population(conf)
3:      generation ← 0
4:      repeat
5:          generation ← generation + 1
6:          for all p in P do
7:              p ← find_best_vm_mapping(p)
8:          end for
9:          P_next ← ∅
10:         for i ← 1, population_size/2 do
11:             p_1 ← binary_tournament(P)
12:             p_2 ← binary_tournament(P)
13:             c_1, c_2 ← integer_recombination(p_1, p_2)
14:             P_next ← P_next ∪ geometric_mutation(c_1)
15:             P_next ← P_next ∪ geometric_mutation(c_2)
16:         end for
17:         P ← P_next
18:     until generation ≥ max_generations
19:     return fittest_element(P)
20: end procedure
```

B. Instantiating service components

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 resource constraints expressed in equation 1. 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:

$$\prod_{c=1}^{C} a(c)^{\sum_{d=1}^{D} b(c,d)} \tag{3}$$

where $a(c)$ is the function returning the types of VMs 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, 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. The allocation of VMs is performed according to the exponent parameter α_t (where $\alpha_t \in \mathbb{R}_{>0}$), which represents the "aggressiveness" in allocating service components of type t to VMs of larger size. This means that, to instantiate components of type t, the algorithm will allocate α_t more VMs of size s than those of size $s-1$. In fact, the lower α_t is, the more the distribution will be skewed towards larger size VMs. This assumption effectively transforms the search space from a combinatorial space to the more convenient $\mathbb{R}^C_{>0}$.

We decided to adopt a random search procedure in the space of possible VMs, as presented in algorithm 2. While random search is a relatively unsophisticated heuristics, we chose it for its robustness in dealing with "jagged" objective functions.

Algorithm 3 presents the pseudo-code for our VM allocation procedure, and figure 2 shows an example allocation of 100 VMs of a given component type with different values for the aggressiveness parameter α_t.

Algorithm 2 Find best VM mapping

```
1:  procedure FIND_BEST_VM_MAPPING(p)
2:      attempt ← 0
3:      p.vms ← allocate_vms(p.components)
4:      p.fitness ← evaluate_fitness(p)
5:      repeat
6:          attempt ← attempt + 1
7:          q ← p
8:          for all c in q.components do
9:              c.α ← random(0.0, 1.0)
10:         end for
11:         q.vms ← allocate_vms(q.components)
12:         q.fitness ← evaluate_fitness(q)
13:         if q.fitness > p.fitness then
14:             p.vms ← q.vms
15:         end if
16:     until attempt ≥ max_attempts
17:     return p
18: end procedure
```

V. Architecture and Implementation

We realized a prototype implementation of the framework discussed in the previous Sections. Our prototype enables

Algorithm 3 Allocate VMs

```
1: procedure ALLOCATE_VMS(components)
2:     vms ← []
3:     for all c in components do
4:         ▷ avt and v are ordered by decreasing VM size
5:         avts ← allowed_vm_types(c.type)
6:         v ← []
7:         for i ← 1, size(avts) do
8:             v[i] ← c.α^(i−1)
9:         end for
10:        v ∗ c.to_allocate/sum(v)          ▷ normalize v
11:        vms.append(v)
12:    end for
13:    return vms
14: end procedure
```

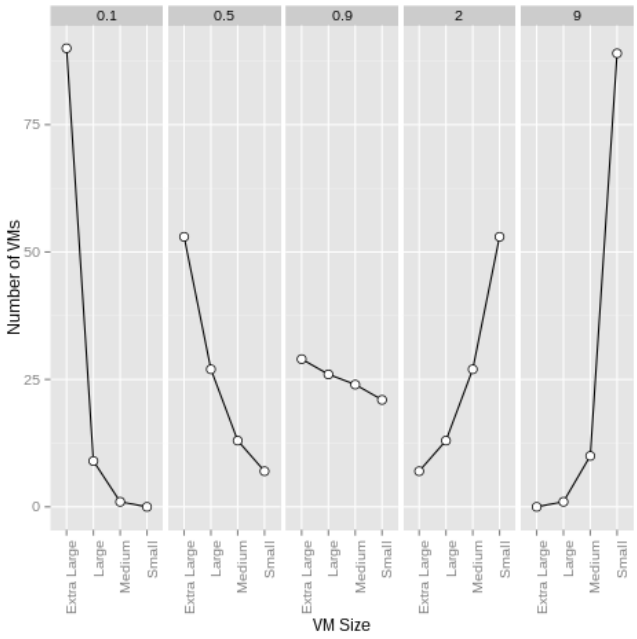

Fig. 2. Allocation of 100 VMs with different values for the aggressiveness parameter α_t

business manager to provide a full description of the Cloud service that they want to optimize. It then reenacts the Cloud service with different configurations through a series of what-if scenario simulations, returning the configuration with the lowest cost.

The prototype architecture is depicted in Fig. 3. The Cloud Service Configuration component is in charge of processing the description of the IT service provided by business managers and of using this information to coordinate the other components of the framework in the reenactment and optimization processes.

The configuration provided by the business manager includes all the information needed to reenact the IT service and evaluate the business performance of many possible configurations. More specifically, the configuration describes the business processes that the Cloud service implements (defined through a BPEL-inspired language), the service components used for the business processes' implementation (with their respective minimum hardware requirements and average service times), the data centers that can be used for the placement of service components (with their VM pricing model), the customer request generation model, and finally the business impact model to consider for the evaluation of each IT service configuration (i.e., the target function to optimize). This configuration model was designed to enable the prototype to consider a very large class of Cloud services offered on top of major Cloud providers, such as Google and Amazon.

The Optimization component is in charge of exploring the space of possible configurations for the IT service. To this end, the Optimization component leverages the algorithms described in the the previous Section.

The Simulator component is in charge of reenacting the IT service with the particular configuration selected by the Optimization component. This component implements a discrete-event simulator that reenacts each VM as a queue. Incoming requests are added at the end of the queue and handled according to the queue management policy (FIFO, priority-based, etc.). The use of multiple servers and/or of non-trivial request management strategies enables this model to reenact a wide range of real-life service components with reasonable accuracy.

Finally, the Business Impact Analysis component is in charge of evaluating the cost, i.e., the business impact, of each configuration. To this end, it calculates the costs related to VM pricing and analyzes the KPIs provided by the Simulator component to evaluate whether SLO violation penalties should be applied. The result is returned to the Optimization component.

We realized the prototype in the Ruby programming language. More specifically, we adopted the JRuby platform (http://jruby.org), a Ruby virtual machine implemented in Java, for its excellent support for concurrent programming and its capability to interface with high quality Java-based scientific libraries, such as Apache Commons Math (http://commons.apache.org/math/).

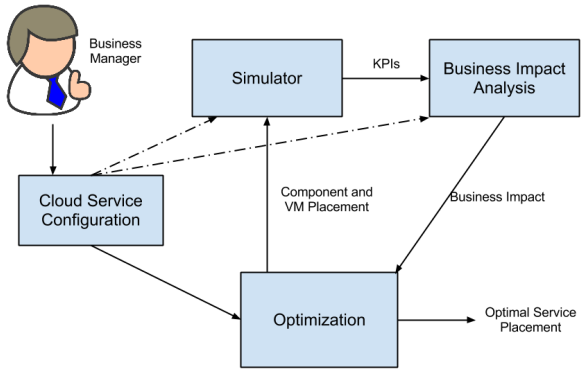

Fig. 3. Architecture of the prototype

VI. Experimentation and Case Studies

We experimentally tested our prototype by reenacting the deployment of the Cloud service example illustrated in Section II. More specifically, we considered the optimization of an enterprise-class installation of the service, serving up to 10 million requests per hour to a single enterprise customer with presence in 3 different continents: Europe, (North) America, and Asia. In order to minimize communications-related latencies in service delivery, we assumed that the requests from each division would be served from a Cloud data center in the same continent. To this end, we consider the deployment of components in the Amazon EC Cloud data centers in Ireland, (Northern) Virginia, and Singapore.

We assumed that each division in the customer organization makes a different use of the Cloud service, with the US division being responsible for 50%, the EU division for 30%, and the Asia division for 20% of the total number of requests. We also assumed different usage profiles for each division, with the EU division's request distribution for workflows A-C being 30%, 60%, and 10%, the US division's being 45%, 45%, and 10%, and the Asia division's being 50%, 30%, and 20%.

In addition, in the attempt to reproduce realistic customer service requests dynamics, we considered a workload that changes dynamically according to daily pattern. Fig. 4 shows the daily trend in the request loads from the different divisions as well as the aggregated one.

We adopted a stochastic model based on non-homogeneous Poisson processes to reenact the customer service requests arrival process. While some real-life service requests patterns might require more sophisticated models [9], non-homogeneous Poisson processes represents a good tradeoff between model accuracy and model complexity [2].

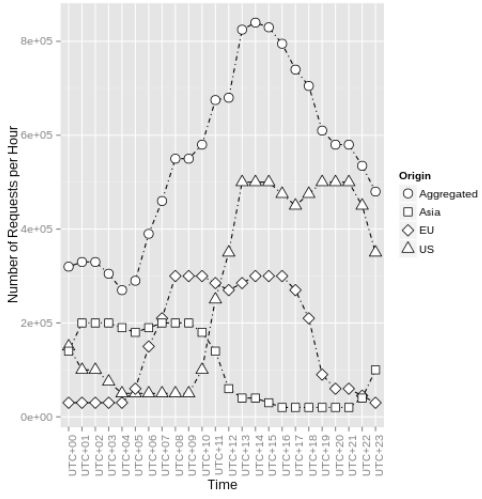

Fig. 4. Load of requests

We also limited the number of VM types considered to General purpose VMs from *m1.small* to *m1.xlarge* [4]. We assumed no requirement for the Web Service component (thus enabling it to run on any type of VM), a 2GB RAM minimum requirement for the Application Server and Search Server components (thus preventing them from running on m1.small VMs), and a minimum requirement of 2 CPUs for the Persistence Storage and Financial Transaction Server components (thus enabling them to run only on m1.large or m1.xlarge VMs). For the reenactment of each VM we used G/M/1 FIFO queues, with exponentially distributed service times with a mean that depended from the specific type of VM type and the software component that it instantiates.

In the experiments, we considered only shared components. This means that, after each step in a workflow, the next VM to which the request will be forwarded will be randomly selected from all the VMs instantiating the software component type specified in the next step of the workflow.

Finally, to keep the IT costing model relatively simple, we only considered on-demand VMs, that have a fixed and per-use pricing. For the penalties model we considered a 500 $ penalty for every percentage point of requests with a time to resolution higher than 2 seconds.

In order to test the prototype effectiveness for predictive optimization, we decided to test how it responds to the greatest difference in request load, the one between UTC+12 and UTC+13 (+1,450,000 requests per hour). We ran a first experiment at the UTC+12 time. The best configuration returned by the prototype had an associated total cost of 12,220.92 $/day and is shown in Fig. 5. We then ran a second experiment at the UTC+13 time. The best configuration we achieved for UTC+12 was not very well suited to handle the increased load, returning a cost of 17,220.92 $/day, significantly higher than the one exhibited at UTC+12 because of heavy SLO violation penalties. Instead, the best configuration returned by the prototype, shown in Fig. 6, was capable of finding a significantly better configuration with a cost of 15,656.52 $/day, which represent a saving of 9.1%.

For each experiment, we used a population of 64 individuals for the genetic algorithm and 4 iterations for the local search procedure, terminating the evaluation after 5 generations. This effectively means that 1280 different configurations were evaluated for each experiment. We chose these values because, after several experiments, we empirically determined that they represented a reasonable trade off between exploration of the search space and computational resource utilization. Using those parameters we were capable of running each of the experiments in roughly 3 hours on a 2011 laptop equipped with 8GB RAM and a quad-core second generation Intel i7 processor and running a 64-bit version of Arch Linux (kernel 3.10.5).

VII. Related Work

It is possible to envisage a large number of applications for the optimal placement of IT service provider processes in the Cloud from the business point of view. It may be used to identify possible values of the SLA penalty for different levels of the quality of service agreement breaches. It may be also used to identify the minimum possible cost of hosting for the Cloud provider. In this paper we consider application of

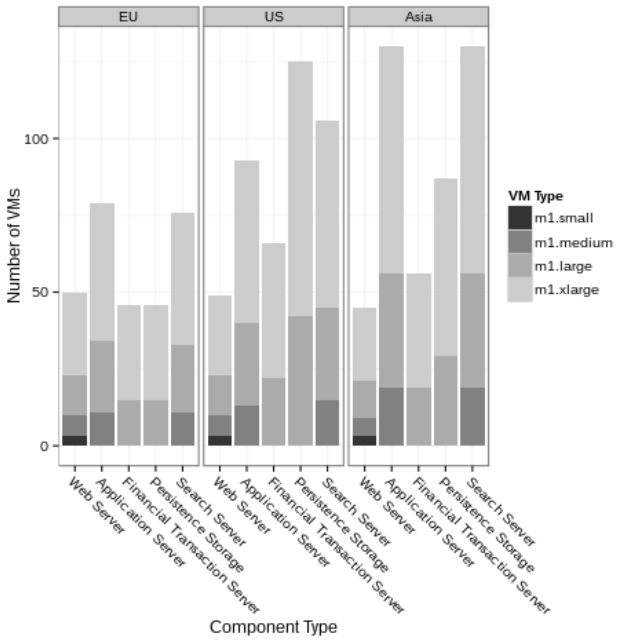

Fig. 5. Optimal component and VM allocation at UTC+12.

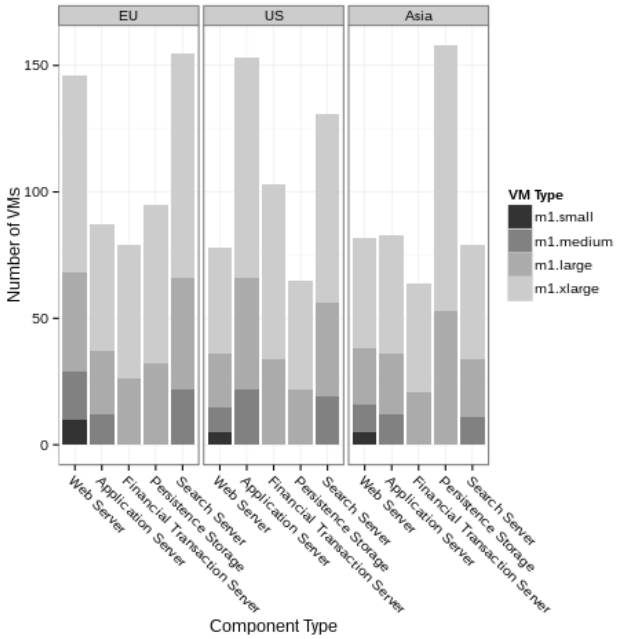

Fig. 6. Optimal component and VM allocation at UTC+13.

optimal placement to the deployment of IT service provider processes, a problem which is known to be NP hard [10].

One of the first application of the business optimal resource placement is due to Menasce et al. [11], who used simulation to identify possible QoS levels and SLA penalties costs for the distributed resource allocations. A nice approach was suggested in [10] where queuing theory was used to obtain exact an mathematical solution for the gain optimization problem under restrictions on incoming service requests, simplified IT process, etc. The restrictions of the above work were partially lifted in [12], where in addition placement of the multi-tiered system were analyzed.

Some researchers, including two authors of this paper, adopted discrete utility functions and heuristics to solve the placement problem. A continuous utility function was considered in [13] and an iterative approach to finding optimal placement was suggested. An approach based on approximation with good theoretical guaranties of closeness to optimal solution is suggested in [14]. An interesting approach based on fuzzy logic and control theory to the multi-tier placement problem was suggested in [15]. An approach to optimization based on minimizing energy usage is suggested in [16], see also [17], [18].

Cloud computing advantages from multiple (business, technical, maintenance) point of views are considered in [19] and [20]. Additional details may be found in the monograph [21].

Statistical modeling of the Cloud load is considered in [22] and [23]. We use non-homogeneous Poisson processes to model burstiness of the Cloud load, extending [2].

Another interesting avenue of research recently emerging involves the optimization of component placement in Clouds from the perspective of [24] [25]. These studies optimize component placement from the point of view of Cloud providers instead of the service provider one.

Finally, business driven service component optimization in federated Clouds through genetic algorithm optimization was originally proposed in [26]. This paper significantly extends that work to consider a resource definition and usage model, complex workflows, a more realistic workload model, and a 2-phase optimization metaheuristics that splits the optimization process in the service component placement and VM allocation subproblems.

VIII. Conclusions and Future Work

This paper presents a framework for business driven approach to a placement of complex services in federated Clouds. For modeling complex services, we assume that the basic unit of load on the data center is a request for a workflow and covers a number of the aspects connecting business processes, IT services, IT resource consumption and IT resource costing description. Modeling IT resource costing is demonstrated by considering a number of Cloud providers and including cost of the placement (or deployment) of the required components, cost of the relocating some of the components and the cost of the SLO breaches.

We are planning to extend our model to consider also dynamic changes in service configuration, e.g., to take advantage of *Cloud bursting* to respond to high peeks in the request load. In addition we are planning to experiment with more sophisticated metaheuristics for the optimization of service components and VMs and to investigate a cost model for VM migrations, both within a single data center and across different data centers.

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] C. Bartolini, C. Stefanelli, and M. Tortonesi, "Synthetic incident generation in the reenactment of it support organization behavior," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 126–133.

[3] "Google Pricing," https://cloud.google.com/pricing/compute-engine, [Online; retrieved on July 3, 2013].

[4] "Amazon EC2 Instances," http://aws.amazon.com/ec2/instance-types/, [Online; retrieved on September 1, 2013].

[5] "Amazon Pricing," http://aws.amazon.com/ec2/pricing/, [Online; retrieved on July 3, 2013].

[6] "BPEL 2.0," http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, [Online; retrieved on July 26, 2013].

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

[8] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 35:1–35:33, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2480741.2480752

[9] C. Bartolini, C. Stefanelli, and M. Tortonesi, "Bmodeling it support organizations from transactional logs," in *Network Operations and Management Symposium (NOMS 2010), 2010 IEEE/IFIP*. IEEE, 2010, pp. 256–263.

[10] Z. Liu, M. S. Squillante, and J. L. Wolf, "On maximizing service-level-agreement profits," in *Proceedings of the 3rd ACM conference on Electronic Commerce*. ACM, 2001, pp. 213–223.

[11] D. A. Menascé, V. A. Almeida, R. Fonseca, and M. A. Mendes, "Business-oriented resource management policies for e-commerce servers," *Performance Evaluation*, vol. 42, no. 2, pp. 223–239, 2000.

[12] D. Ardagna, M. Trubian, and L. Zhang, "Sla based profit optimization in multi-tier systems," in *Network Computing and Applications, Fourth IEEE International Symposium on*. IEEE, 2005, pp. 263–266.

[13] ——, "Sla based resource allocation policies in autonomic environments," *Journal of Parallel and Distributed Computing*, vol. 67, no. 3, pp. 259–270, 2007.

[14] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Inf. Process. Lett.*, vol. 100, no. 4, pp. 162–166, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.ipl.2006.06.003

[15] Y. Diao, J. L. Hellerstein, and S. Parekh, "Using fuzzy control to maximize profits in service level management," *IBM Systems Journal*, vol. 41, no. 3, pp. 403–420, 2002.

[16] 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*. IEEE, 2010, pp. 131–138.

[17] P. Dube, G. Grabarnik, and L. Shwartz, "Suits: How to make a global it service provider sustainable?" in *Network Operations and Management Symposium (NOMS 2012), 2012 IEEE/IFIP*. IEEE, 2012, pp. 1352–1359.

[18] G. Y. Grabarnik and L. Shwartz, "Managing service requests with common renewable resources," *Journal of the Chinese Institute of Industrial Engineers*, vol. 28, no. 2, pp. 102–110, 2011.

[19] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[20] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[21] J. Rosenberg and A. Mateos, *The cloud at your service*. Manning Publications Co., 2010.

[22] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*. IEEE, 2010, pp. 87–92.

[23] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *System of Systems Engineering (SoSE), 2011 6th International Conference on*. IEEE, 2011, pp. 276–281.

[24] 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.

[25] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "HARMONY: Dynamic heterogeneity-aware resource provisioning in the cloud," *The 33rd International Conference on Distributed Computing Systems (ICDCS)*, July 2013.

[26] L. Foschini and M. Tortonesi, "Adaptive and business-driven service placement in federated cloud computing environments," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 1245–1251.