On the Consolidation of Data-centers with Performance Constraints

Jonatha Anselmi, Paolo Cremonesi, and Edoardo Amaldi

Politecnico di Milano, DEI
Via Ponzio 34/5, I-20133 Milan, Italy
jonatha.anselmi@polimi.it

Abstract. We address the *data-center consolidation* problem: given a working data-center, the goal of the problem is to choose which software applications must be deployed on which servers in order to minimize the number of servers to use while avoiding the overloading of system resources and satisfying availability constraints. This in order to tradeoff between quality of service issues and data-center costs. The problem is approached through a robust model of the data-center which exploits queueing networks theory. Then, we propose two mixed integer linear programming formulations of the problem able to capture novel aspects such as workload partitioning (load-balancing) and availability issues. A simple heuristic is proposed to compute solutions in a short time. Experimental results illustrate the impact of our approach with respect to a real-world consolidation project.

1 Introduction

As the complexity of information technology (IT) infrastructures increases due to mergers or acquisitions, new challenging problems arise in the design and management of the resulting computer systems. In fact, they are often costly, non-flexible, yielding under-utilized servers and energy wastings. To reduce conflicts among the offered services, many enterprise data-centers host most of their services on dedicated servers without taking into account the possibility of deploying multiple services on a single server. Therefore, many servers are not used at their maximum capabilities and, in turn, expensive hardware investments are often required. Nowadays companies search for IT solutions able to significantly drop data-centers costs, e.g., energy consumption, space costs, and obtain a flexible system satisfying customer demand.

In this framework, the *consolidation* of data-center resources is a current solution adopted by many industries. The objective of consolidation problems is to reduce the complexity of computer systems while guaranteeing some performance and availability constraints. This is usually achieved by searching for the best mapping between software applications and servers which minimizes data-center costs. The main issues taken into account by a data-center consolidation solution are i) to reduce the complexity of the IT infrastructure, ii) to

increase system performance, iii) to obtain a flexible system, iv) to reduce data-center costs, and v) to improve operational efficiencies of business processes. The current challenge characterizing a consolidation is finding an optimal allocation of services to target servers able to meet the above issues while satisfying performance and availability requirements. This results in new capacity planning problems which we address in this work. An important example of performance index is given by the response time, i.e., the time interval between the submission of a job into a system and its receipt. Such variable is strictly related to the servers utilizations, i.e., the proportion of time in which a server is used. In fact, the higher the servers utilizations, the higher the resulting response time. Therefore, constraints on the maximum utilization of each server are important to

1. avoid the saturation of physical resources letting the system handle unexpected workload peaks,
2. guarantee a low sensitivity of data-center response time in front of small workload variations (it is well-known, e.g., [6], that the response time curve grows to infinity according to a hyperbole when the utilization of a server approaches unity),
3. increase the data-center reliability because if a failure occurs on a server, then the associated applications can be moved on different servers preventing a drastic growth of data-center response time.

Constraints on the maximum response time are very often used in many applications for ensuring some quality of service (see, e.g., [4],[1]). The solution of such problem is aimed to yield a data-center *configuration* able to satisfy the above issues while minimizing costs.

1.1 Related Work

During the last decades, the resource management problem has been analyzed in depth by many researchers in many frameworks and several works are available in the literature. However, little appeared in the literature for the recent problem of data-center consolidation with performance constraints even though it attracted the attention of many IT companies.

Rolia et al. [12] analyze the consolidation problem with a dynamic approach taking into account the workloads seasonal behavior to estimate the server demands. Their consolidation problem limits the overall utilization of each server and it assumed that each application is deployed on a single server. An integer linear program formulation and a genetic algorithm are proposed to solve the problem and a case study with 41 servers is presented. Bichler et al. [3] present a similar dynamic approach tailored for virtualized systems. The main difference of their approach is that the optimization problem is solved exploiting multi-dimensional bin-packing approximate algorithms [8]. In the context of virtualized servers, an example of how to consolidate servers is also shown in [9],[10]. Our previous work [1] tackles the data-center consolidation problem

exploiting queueing networks theory. Linear and non-linear optimization problems are provided as well as accurate and efficient heuristics. However, such work is essentially based on the assumption that one software application must be deployed on exactly one server. Furthermore, it is assumed that the *service demands* of the queueing networks model are known.

1.2 Our Contribution

In the present work, we again tackle the data-center consolidation through an optimization problem extending our previous work [1]. The proposed formulation now takes into account the capability to handle the workload partitioning (or load-balancing) of applications, i.e., the fact that one application can be deployed on many servers. This is clearly related to availability issues. The estimates of performance indices are again obtained by exploiting queueing networks (QN) theory (see, e.g., [6]) because it provides versatile and robust models for predicting performance. However, this is achieved through a new, innovative approach. In fact, the standard theory underlying QN models assumes that a number of input parameters, e.g., arrival rates and service demands, must be known in advance to obtain performance estimates. Unfortunately, in practice these parameters can be very difficult to obtain for a variety of reasons (see, e.g., [7]). Therefore, we adopt a new, robust methodology to estimate performance which is only based on the observable variables which are usually easy to measure. As a matter of fact, in real-world scenarios a working infrastructure exists before starting a performance evaluation and a measurement phase can be carried out. In the context of IT systems, common experience reveals that server utilizations and *speed-up*s possess such requirements. Our analysis assumes the knowledge of only these two latter parameters, i.e., standard input parameters such as arrival rates and service times are not part of our approach. We then propose a number of linear optimization models related to the data-center consolidation. Given that the computational effort needed by standard exact solution algorithm is expensive, an heuristic is shown to efficiently solve the optimization problems in an approximate manner. The computational effort and the accuracy of such heuristic is evaluated with respect to a real-world consolidation project with 38 servers and 311 web applications. We then present several minor extensions of practical interest to the above issues, e.g., the case in which applications require storage.

This work is organized as follows. In Section 2, we discuss the parameters characterizing the data center and define the associated QN model. In Section 3, we present our main formulation of the consolidation problem proposing a heuristic for its efficient solution. Section 4 is devoted to experimental results on a real-world consolidation project. Finally, Section 5 draws the conclusions of our work and outlines further research.

2 Data-center Queueing Network Model

2.1 Data-center Description

The data center is composed of M heterogeneous servers. The cost of using server j, which comprises energy consumption, maintainability costs, etc., is denoted by c_j, $j = 1, \ldots, M$.

The *speed-up* of server j is denoted by ρ_j and it is understood as its relative processing capacity obtained by the execution of suitable benchmarks with respect to a reference server (say server 1), i.e., the ratio between the processing speeds of server j and 1.

The data center hosts R different applications (or services) and each application is deployed on multiple tiers (e.g., web-server tier, application-server tier, etc.). Application r sequentially spans L_r tiers, $r = 1, \ldots, R$, and when an application r job (or client) joins the data center, it initially executes tier 1 on some server, then it proceeds to tier 2 and so on till the L_r-th. For application r jobs, when the L_r-th tier is reached, the request is forwarded back to the $(L_r - 1)$-th for some further processing and so on till the first one. It is well-known that this behavior agrees with standard multi-tiered architectures. We denote by

$$L = \sum_{r=1}^{R} L_r \tag{1}$$

the total number of application tiers. More than one application tier can be deployed on a given server and each tier of each application can be deployed on multiple servers.

The deployment of a given application on multiple tiers is usually referred to as *vertical scalability* and it is important to provide a better performance handling larger workloads and to solve possible conflicts among different layers (different application tiers may use different technologies). On the other hand, the deployment of a given application tier on multiple servers is usually referred to as *horizontal scalability* and lets us deal with load-balancing issues. The horizontal scalability is also important to guarantee availability constraints: in fact, if a given application tier is deployed on multiple servers, then a failure on a single server does not prevent the availability of the application because the workload can be rearranged among the available servers.

To reduce management costs and to increase the data-center availability, we assume that each application tier must be deployed on a number of servers ranging between two fixed values. Therefore, we denote by $m_{r,l}$ and $n_{r,l}$, respectively, the maximum and the minimum number of servers in which tier l of application r must be deployed.

Another source of lack of data-center availability is the deployment of several tiers on a same server. Therefore, we assume that a maximum number of v_j application tiers can be deployed on server j. This assumption is also meant to avoid the modeling of non-negligible overheads in service times estimates (usually referred to as *virtualization overhead*) which would be introduced by

the middleware management if the number of virtual machines running on a single server is large.

In agreement with the notation of basic queueing networks theory [6], we denote by $D_{j,r,l}$ the mean *service demand* (time units) required by a job executing tier l of application r on server j when the network contains no other job.

If not specified, indices j, r and l will implicitly range, respectively, in sets $\{1,\ldots,M\}$, $\{1,\ldots,R\}$ and $\{1,\ldots,L_r\}$ indexing servers, applications and tiers.

2.2 QN Model

QN models are a popular tool for evaluating the performance of computer systems and, in the mathematical formulation of the data-center consolidation problem, they let us deal with simple analytical expressions of performance indices. The class of queueing network models we consider goes beyond the popular class of product-form (or separable) queueing networks [2],[6],[11]. In fact, we consider those queueing networks satisfying the utilization law, e.g., [6] (it is well-known that this is a much larger class). This is simply due to the fact that the performance indices we consider are server utilizations only. Therefore, this lets our approach rely on wide assumptions and be widely applicable and robust.

Since the data center hosts different applications (characterized by different service demands) and an arriving job can execute only one of them, the model we build is *multiclass*. For convenience, a job requesting the execution of application r is referred to as a class-r job. Since the number of jobs populating the data center is not constant, the model we build is *open* and we denote by λ_r the mean workload (arrival rate) of class-r jobs, $r = 1,\ldots,R$. Jobs circulate in the network visiting a number of stations and eventually leave the network. The stations of the QN model the data center servers and, in the following, we use the term *station* when we refer to the QN and the term *server* when we refer to the data-center.

Let $D_{j,r}$ be the mean *service demand* [6] of class-r jobs at station j, i.e., the total average time required by a class-r job to station j during the execution of all its tiers and when the network contains no other job. Within this standard definition, we underline that the service demands include the processing times of jobs at servers when they visit stations passing from the first tier to the last one and returning back from the last tier to the first one. This notion of service demand also takes into account that it is possible to deploy more tiers of a given application on the same server. For instance, assuming that only tiers from 1 to $l_r \leq L_r$ of application r are deployed on server j, we have

$$D_{j,r} = \sum_{l=1}^{l_r} D_{j,r,l}. \tag{2}$$

The time interval needed by a server to transfer a job to an other server is assumed to be negligible.

Within this queueing network model of the data center, we recall that the average utilization of station j due to class-r jobs, i.e., the *busy* time proportion

of server j due to class-r jobs, is given by

$$U_{j,r} = U_{j,r}(\lambda_r) = \lambda_r D_{j,r}. \tag{3}$$

Formula (3) is known as the *utilization law* [6]. Clearly, the total average utilization of server j is given by

$$U_j = U_j(\lambda_1, \cdots, \lambda_R) = \sum_{r=1}^{R} U_{j,r} < 1. \tag{4}$$

Since we deal with the *averages* of utilizations, when referring to an index we will drop the word average.

We now show a simple example to illustrate the queueing network model underlying the data center. Let us consider the case of two applications, i.e., $R = 2$, having both three tiers, i.e., $L_1 = L_2 = 3$ and $M = 5$ available servers, and let us also suppose that the application tiers are deployed on the servers as indicated in Table I. For instance, we have that tier 2 of application 2 is deployed on server 3. We notice that server 5 is not used. Since each tier of each application is deployed on exactly one server, all service demands are given by the sum of service times as in (2). The QN model underlying the deployment scheme of

Tier	Class 1	Class 2
1	1	2
2	1	3
3	2	4

Table 1. Deployment scheme of the example.

Table I is such that the stations service demands are given by Table II. Within

Station	Class 1	Class 2
1	$D_{1,1,1} + D_{1,1,2}$	0
2	$D_{2,1,3}$	$D_{2,2,1}$
3	0	$D_{3,2,2}$
4	0	$D_{4,2,3}$
5	0	0

Table 2. Service demands of the deployment scheme in Table I.

this example, each application tier is deployed on a single server and server 5 is not used. We also note that the QN model of the data-center does not explicitly take into account the notion of tier which is embedded in the notion of service demands.

Within the definition of speed-up given in Section 2.1, the following relation must hold

$$\frac{D_{i,r,l}}{\rho_i} = \frac{D_{j,r,l}}{\rho_j} \tag{5}$$

for all i, j, r, l, which implies

$$\frac{D_{i,r}}{\rho_i} = \frac{D_{j,r}}{\rho_j}. \tag{6}$$

3 Formulation and Algorithm

The objective of the data-center consolidation problem is to exploit the available servers in order to obtain a *configuration* able to satisfy, *in the average*, performance constraints on utilizations and data-center response times while minimizing the sum of servers costs.

The decision variables we include in our optimization models are

$$x_{j,r,l} = \begin{cases} 1 \text{ if tier } l \text{ of application } r \text{ is deployed} \\ \quad \text{on server } \ j, \\ 0 \text{ otherwise,} \end{cases} \tag{7}$$

$$y_j = \begin{cases} 1 \text{ if server } j \text{ is used} \\ 0 \text{ otherwise,} \end{cases} \tag{8}$$

and

$$z_{j,r,l} \geq 0 \tag{9}$$

denoting the *proportion* of application r and tier l workload assigned to server j.

Let a *configuration* be a possible assignment of variables $x_{j,r,l}$ satisfying the issues discussed in Section 2.1, i.e., a *feasible* deployment scheme. A configuration can be interpreted as a function f mapping tier l of application r to a subset of $\mathbf{M}$, i.e., a subset of the set of stations. The goal of the optimization problem is to find the configuration of minimum cost which satisfies constraints on server utilizations and constraints on data-center *structural* properties such as the fact that each application tier must be deployed on at least $n_{r,l}$ and at most $m_{r,l}$ servers. We refer to this latter property as *workload partitioning* and it is an innovative aspect of our formulation. The deployment of an application tier on multiple servers is known to increase its availability.

We assume that the data-center has an initial configuration f and that the per-class utilizations of such configuration are known. This reflects a common real-world scenario because in practice a data-center consolidation is performed on a working infrastructure and, within this framework, server utilizations are usually easy to measure and robust. Therefore, we assume the knowledge of per-class utilizations $U_{f(r,l),r,l}$ for all r and l. Considering (3) and (5), we have

$$U_{j,r,l} = \frac{\rho_j}{\rho_{f(r,l)}} U_{f(r,l),r,l}, \ \forall j, r, l \tag{10}$$

which expresses, in a robust manner, the per-class utilization of tier l of application r if it would be deployed on server j as a function of measured data and known parameters.

3.1 Formulation of the Consolidation Problem

Let $\hat{U}_j$ denote the value of the maximum utilization that server j is allowed to have. We formulate the consolidation problem through the following ILP problem

$$\mathcal{P} : \min \sum_{j=1}^{M} c_j y_j \tag{11}$$

subject to:

$$\sum_{j=1}^{M} z_{j,r,l} = 1, \qquad \forall r, l \tag{12}$$

$$\sum_{r=1}^{R} \sum_{l=1}^{L_r} U_{f(r,l),r,l} \frac{\rho_j}{\rho_{f(r,l)}} z_{j,r,l} \leq \hat{U}_j y_j, \quad \forall j \tag{13}$$

$$z_{j,r,l} \geq \frac{x_{j,r,l}}{m_{r,l}}, \qquad \forall j, r, l \tag{14}$$

$$z_{j,r,l} \leq \frac{x_{j,r,l}}{n_{r,l}}, \qquad \forall j, r, l \tag{15}$$

$$\sum_{r=1}^{R} \sum_{l=1}^{L_r} x_{j,r,l} \leq v_j, \qquad \forall j \tag{16}$$

$$z_{j,r,l} \geq 0, \qquad \forall j, r, l \tag{17}$$

$$x_{j,r,l} \in \{0, 1\}, \qquad \forall j, r, l \tag{18}$$

$$y_j \in \{0, 1\}, \qquad \forall j \tag{19}$$

Clearly, the objective function (11) minimizes the weighted sum of server costs.

Constraints (12) ensure that variable $z_{j,r,l}$ represents proportions of the workload of tier l of application r to forward to server j.

Constraints (13) limit the overall utilization of j by means of relation (10).

Constraints (14) and (15) model, respectively, the fact that the workload of tier l of application r must be allocated on at most $m_{r,l}$ and at least $n_{r,l}$ servers. These constraints ensure the avoidance of very *unbalanced* workloads which may yield situations where most of the workload of an application tier is assigned to a particular server (this is ensured by (14)), and the avoidance of splitting the workload among a very large number of servers which may result in maintainability cost and inefficiencies (this is ensured by (15)). Both constraints (14) and (15) imply that $x_{j,r,l} = 1$ if and only if $z_{j,r,l} > 0$. This can be easily seen if we rewrite (14) and (15) as follows

$$\frac{x_{j,r,l}}{m_{r,l}} \leq z_{j,r,l} \leq \frac{x_{j,r,l}}{n_{r,l}}, \ \forall j, r, l \tag{20}$$

where we see that if $x_{j,r,l} = 0$ (respectively $x_{j,r,l} = 1$) then $z_{j,r,l}$ is forced to be zero (strictly positive).

Finally, constraints (16) limit the number of application tiers to deploy on j.

3.2 Heuristic Solution

The number of binary variables adopted by $\mathcal{P}$ is $M + ML$. Since large-scale data-centers are composed of hundreds of servers and applications, i.e., several thousands of variables $x_{j,r,l}$, the exact solution of $\mathcal{P}$ through standard techniques (e.g., branch and cut) requires a strong computational effort. Therefore, we now provide a simple heuristic aiming to find a good solution in a shorter time.

The heuristic we propose initially guesses the set of servers which yields the configuration of minimum cost and, with respect to this set only, checks whether or not a feasible configuration exists. If such configuration does not exist, then the guess is iteratively refined by adding the *best* server until a feasible solution is found.

Algorithm 1 is the heuristic we propose for the efficient solution of $\mathcal{P}$.

We initially solve $\mathcal{P}$ assuming that variables $x_{j,r,l}$ are continuous. Therefore, the number of binary variables drops from $M + ML$ to M. The optimum of this problem requires a significantly smaller computational effort and the optimal configuration found must be a lower bound on the configuration of minimum cost. We note that a feasible solution of this problem always exists because we assumed that the data-center initially has a working configuration. Then, we define set Y as the set of servers chosen by the optimal configuration of the relaxed problem and $\mathcal{P}'$ which takes into account the servers belonging to Y only. $\mathcal{P}'$ is thus composed of much fewer variables and constraints than $\mathcal{P}$. We then search for a feasible solution of problem $\mathcal{P}'$ (Line 4). If this problem is feasible then a solution is found and the algorithm ends. Otherwise, through problem $\mathcal{P}''$ we augment the space of feasible solutions by adding to Y a server not included in the configuration computed by the relaxed problem in Line 1. Then, we iteratively check for the feasibility of $\mathcal{P}'$ until a feasible configuration exists. We remark that such configuration eventually exists because we initially assume a working configuration.

Given that the optimum of the relaxed problem defined in Line 1 is a lower bound on the solution of $\mathcal{P}$, if the condition in the loop holds at its first evaluation, then Algorithm 1 provides the optimum. In general, if n is the number of iterations performed by the algorithm, then n is an upper bound on the difference between the number of servers identified by the optimal solution of $\mathcal{P}$ and by the proposed heuristic. This holds because we add a server to Y at each iteration and because the objective function value corresponding to the optimal configuration of $\mathcal{P}$ cannot be less than the one obtained in the relaxation of Line 1.

Algorithm 1 Heuristic solution for $\mathcal{P}$

Solve the relaxation of $\mathcal{P}$ when $x_{j,r,l}$ are continuous between 0 and 1, and y_j are binary;
$Y := \{j \in \{1, \dots, M\} : y_j = 1\}$;
$\tilde{Y} := Y$;
for $k = 1, \dots, M - |Y|$ **do**
    Let $\mathcal{P}'$ be problem $\mathcal{P}$ where

- the objective function (11) is removed,
- variables y_j are fixed to 1 for all $j \in \tilde{Y}$, and
- variables y_j and $x_{j,r,l}$, for all $r, l, j \notin \tilde{Y}$ are removed;

    Solve $\mathcal{P}'$;
    if a feasible solution of $\mathcal{P}'$ exists **then**
        break;
    end if
    Let $\mathcal{P}''$ be problem $\mathcal{P}$ where variables

- y_j are fixed to 1 for all $j \in Y$, and
- $x_{j,r,l}$ are binary if $j \in Y$, otherwise continuous between 0 and 1,

    and the following constraint is included

$$\sum_{j=1}^{M} y_j \leq |Y| + k; \tag{21}$$

    Solve $\mathcal{P}''$;
    $\tilde{Y} := \{j \in \{1, \dots, M\} : y_j = 1\}$;
end for
return variables $x_{j,r,l}$;

3.3 Minor Extensions

We now propose minor extensions of practical interest related to the formulation above.

1. Consider the case in which tiers $l_1, \ldots, l_K$ of application r_1 must be deployed on single but different servers, which implies $n_{r_1,l_1} = m_{r_1,l_1} = n_{r_1,l_2} = \ldots = m_{r_1,l_K} = 1$. This need can be due to operating systems incompatibilities, e.g., Windows software on Linux servers. In this case, the constraint is given by
$$\sum_{k=1}^{K} z_{j,r_1,l_k} \leq 1, \ \forall j. \tag{22}$$
We note that (22) is expressed in terms of continuous variables z_{j,r_1,l_k} (instead of x_{j,r_1,l_k}). It is known that this yields a more efficient formulation. Analogously, we can avoid the deployment of particular application tiers on some servers by simply imposing $z_{j,r,l} = 0$ for some j, r and l.
2. Consider the opposite case where tiers $l_1, \ldots, l_K$ of application r_1 must be deployed on the *same* (single) server, which implies $n_{r_1,l_1} = m_{r_1,l_1} = n_{r_1,l_2} = \ldots = m_{r_1,l_K} = 1$. In this case, we add the constraints
$$\begin{array}{rl} z_{j,r_1,l_1} = z_{j,r_1,l_2}, & \forall j \\ z_{j,r_1,l_2} = z_{j,r_1,l_3}, & \forall j \\ \ldots & \\ z_{j,r_1,l_{K-1}} = z_{j,r_1,l_K}, & \forall j. \end{array} \tag{23}$$
3. In many practical cases, some applications must be deployed only on a given subset of servers. This situation can arise for security issues where some critical applications must be deployed in virtual private networks. Let S denote the subset of set $\{1, \ldots, M\}$ containing the indices of the data-center servers which are able to execute the tiers of application r_1. In this case, the constraints are given by
$$x_{j,r_1,l} = 0, \ \forall j \notin S, t, \tag{24}$$
which reduce the size of the problem because many binary variables become constants.

4 Experimental Results

In this section, we present experimental results in order to evaluate the accuracy and the computational requirements of our approach. Experimental analyses have been performed by running the Ilog Cplex v10.0.0 optimization solver on a 2.80GHz Intel Xeon CPU with hyperthreading technology. Algorithm 1 has been implemented in the AMPL language [5].

We apply Algorithm 1 to a real consolidation project within the data-center of one of the largest European telecommunication companies. The portion of

the data-center involved in the consolidation project consists of 311 single-tier applications running on 311 dedicated servers. The applications were originally consolidated with a manual mapping between the applications and 38 brand-new systems. For each system, the mapping required to keep overall utilization below a 70% threshold. Figure 1 shows the CPU utilizations for the manually consolidated servers. The applications were consolidated using VMWare ESX Server

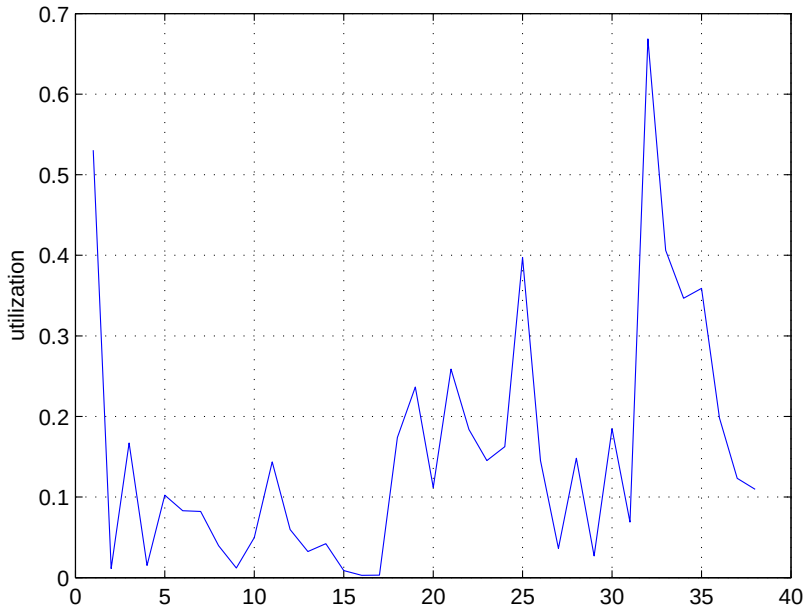


Fig. 1. Utilizations of the initial configuration.

3.5. The systems used for the consolidation were HP ProLiant BL680c G5 blade servers and ProLiant DL58x servers. Most of the servers have 8 CPUs, but the blade systems have up to 80 processors (see Figure 2). The total computational power of the selected systems exceeds 1.6 THz.

We applied Algorithm 1 to the above data-center configuration in order to find a better consolidation strategy. Before running the algorithm, systems and applications have been monitored for a one-month period in order to measure, for each application, the average CPU utilization. Moreover, for each server, configuration information have been collected, describing processing power (MHz) and number of CPUs. Such metrics have been used to derive the relative speed-ups between systems. Algorithm 1 has been applied by varying the utilization thresholds in the range between 0.3 and 0.7, with step 0.1. In Figure 3.a, we

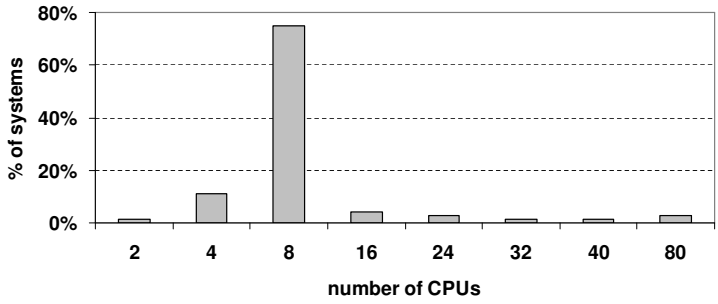
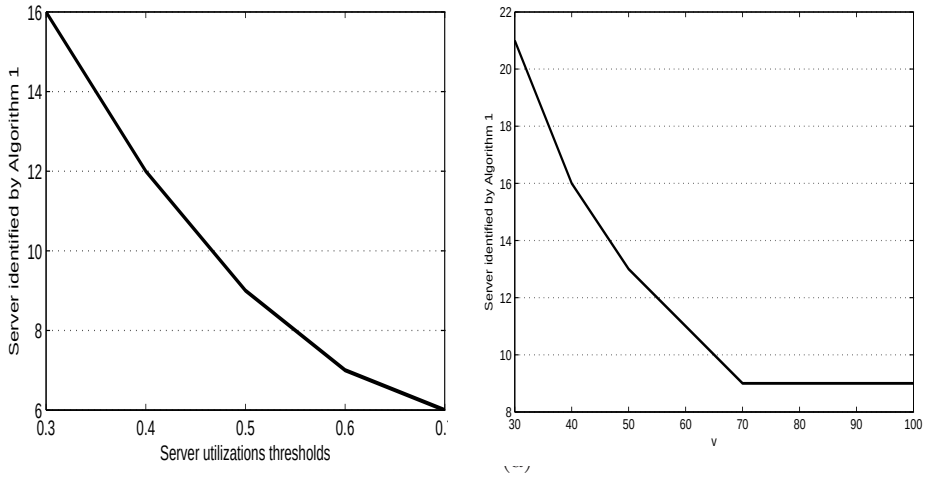


Fig. 2. Distribution of the number of CPUs of the servers.

show the number of servers identified by our approach. When the target maxi-



(b)

Fig. 3. Number of servers identified by the proposed heuristic by varying the utilization thresholds (on the left). Number of servers identified by the proposed heuristic assuming $n_{r,t} = 2$ and $m_{r,t} = 4$ and varying v (on the right).

mum server utilization of 0.7 is considered, we show that it is possible to obtain a configuration which adopts only 6 servers. With respect to the 38 servers chosen by the initial configuration, this has a drastic impact on data-center costs. We notice that the number of servers identified by our approach decreases as the maximum server utilization increases. This is obviously due to the fact that more applications can be deployed on a single server as its maximum utilization increases. In all cases, the number of iterations performed in the loop of the

algorithm was zero. This implies that an optimal configuration has been always found. With our heuristic, all experiments terminated within 3 seconds. This because the relaxation in Line 1 of Algorithm 1 identifies a very small set of servers which significantly yields to reduce the total number of binary variables $x_{j,r,l}$.

We now consider the case where each application must be deployed on at least 2 and at most 4 servers. Assuming 0.7 as maximum utilization thresholds, we vary the maximum number of applications to deploy on a given server from 30 to 100 with step 10 and show the number of servers identified by Algorithm 1 (see Figure 3.b). Even in this case, the number of iterations performed in the loop of the algorithm was zero. In the figure, we see the price we have to pay for load-balancing applications among multiple servers. In fact, in this case the optimal configuration is composed of 9 servers.

5 Conclusions

In this paper, we addressed the problem of finding an *optimal* data-center configuration able to satisfy performance and availability constraints. Recently, this problem received a lot of attention by industries. We built a queueing network model of the data-center and imposed constraints on server utilizations, a critical parameter strictly related to data-center stability. Then, we tackled the problem as an optimization problem and proposed new mixed integer linear programming formulations able to take into account innovative aspects. These include the possibility of deploying a given software applications on a number of servers between two given thresholds in a controlled, load-balanced manner. Given that the computational effort needed by standard exact solution algorithm is expensive, an heuristic is proposed to efficiently solve the optimization problem in an approximate manner. The approach is robust because servers utilizations are derived without taking into account the standard input parameters characterizing queueing models, e.g., arrival rates and service demands. In fact, the expressions of server utilizations have been obtained within the observable variables which, in data-centers, are usually easy to measure and robust. Experimental results on a real consolidation project revealed that the heuristic is able to compute optimal configuration in very short time. We leave as future work the extension of our formulation which takes into account resources *profiles*, i.e., the possibility of having different workload demands at different time intervals.

References

1. J. Anselmi, E. Amaldi, and P. Cremonesi. Service consolidation with end-to-end response time constraints. In *Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference 3-5 Sept. 2008 Page(s):345 - 352.*
2. F. Baskett, K. Chandy, R. Muntz, and F. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J.ACM*, 22(2):248–260, 1975.

3. M. Bichler, T. Setzer, and B. Speitkamp. Capacity planning for virtualized servers. In *In Proceedings of the Workshop on Information Technologies and Systems, Milwaukee, Wisconsin, USA, 2006*.
4. V. Cardellini, E. Casalicchio, V. Grassi, and R. Mirandola. A framework for optimal service selection in broker-based architectures with multiple qos classes. In *SCW '06: Proceedings of the IEEE Services Computing Workshops*, pages 105–112, Washington, DC, USA, 2006. IEEE Computer Society.
5. R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, November 2002.
6. E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1984.
7. Z. Liu, L. Wynter, C. H. Xia, and F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *Perform. Eval.*, 63(1):36–60, 2006.
8. S. Martello, D. Pisinger, and P. Toth. New trends in exact algorithms for the 0-1 knapsack problem. 1997.
9. D. A. Menasce. Virtualization: Concepts, applications, and performance modeling. the volgenau school of information technology and engineering, 2005.
10. D. A. Menasce, V. A. F. Almeida, and L. W. Dowdy. *Performance by Design: Computer Capacity Planning by Example: Computer Capacity Planning*. Prentice Hall International.
11. D. A. Menasce, L. W. Dowdy, and V. A. F. Almeida. *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
12. J. Rolia, A. Andrzejak, and M. F. Arlitt. Automating enterprise application placement in resource utilities. In *DSOM*, pages 118–129, 2003.