Noname manuscript No.
(will be inserted by the editor)

SOC: Satisfaction-oriented Virtual Machine Consolidation in Enterprise Data Centers

Xi Li · Anthony Ventresque · John Murphy · James Thorburn

Received: date / Accepted: date

Abstract Server sprawl is a problem faced by data centers, which causes unnecessary waste of hardware resources, collateral costs of space, power and cooling systems, and administration. This is usually combated by virtualization based consolidation, and both industry and academia have put many efforts into solving the underlying virtual machine (VM) placement problem. However, IT managers' preferences are seldom considered when making VM placement decisions. This paper proposes a satisfaction-oriented VM consolidation mechanism (SOC) to plan VM consolidation while taking IT managers' preferences into consideration. In the mechanism, we propose: i) an XML-based description language to express managers' preferences and metrics to evaluate the satisfaction degree; ii) to apply matchmaking to locate entities (i.e., VMs and physical machines (PMs)) that best match each other's preferences; iii) to employ the VM placement algorithm proposed in our previous work to minimize the number of hosts required and the resource wastage on allocated hosts. SOC is compared with two baselines: placement-only and matchmaking-only. The simulation results show that most of the VM-to-PM mappings output from placement-only violate given preferences, while SOC has a satisfaction degree close to matchmaking-only, without requiring too many PMs as matchmaking-only does, but only an amount close to placement-only. In brief,

Xi Li · John Murphy
Lero@UCD, School of Computer Science and Informatics, University College Dublin, Dublin, Ireland
E-mail: xi.li@ucdconnect.ie; j.murphy@ucd.ie

Anthony Ventresque
Lero@UCD, School of Computer Science and Informatics, University College Dublin, and IBM Research, Smarter Cities Technology Centre, Damastown Industrial Estate, Dublin, Ireland
E-mail: anthony.ventresque@ucd.ie

James Thorburn
IBM Software Group, Toronto, Canada
E-mail: jthorbur@ca.ibm.com

SOC is effective in minimizing the number of hosts required to support a certain set of VMs, while maximizing the satisfaction degree of both managers from the provider and requester side.

Keywords VM Placement · Server Consolidation · Matchmaking · Satisfaction · Preference

1 Introduction

Enterprise data centers possess massive IT infrastructures. As corporations grow, new computing resources are purchased regularly to meet the increasing demand, but the pace of the retirement of the old machines rarely keeps pace. Merging and acquiring new companies also brings additional servers that can be difficult and time-consuming to integrate efficiently. Moreover, the practice of dedicating one or more physical machines (PMs) to one application to realize application isolation before virtualization technologies were widely adopted results in increased number of servers and server underutilization. All these factors contribute to the problem which is often referred to as server sprawl. It causes not only unnecessary waste of hardware resources, but also collateral costs (e.g., space, power, cooling, administration). The virtualization-based consolidation technology [25] is a common solution adopted in data centers to solve this.

Consolidation is to place multiple virtual machines (VMs) on fewer, powerful PMs to enable the decommissioning of the unneeded PMs. However, it is challenging to decide which VM should be placed on which PM in large-scale heterogeneous environments. This underlying VM placement problem is usually formulated as the NP-hard bin-packing problem (BPP). Due to the significance of this problem and the potential savings that could be obtained, both industry and academia have put many research efforts into solving it. However, the preferences of the managers (i.e., a person in charge of the administration of a group of VMs or PMs) are seldom considered when making VM placement decisions. We claim that they are important and propose to address this problem in this paper.

Managers are to some extent autonomous, especially in multi-national, multi-divisional, and multi-geographic enterprises. In such cases, they have specific permissions, obligations and special preferences that they want to see fulfilled in placements. For instance: i) in the case that physical servers spread across data centers at multiple sites, the manager could have preferences on site/city/building for the following reasons:

- interested in reducing costs related to electricity, human labour or physical space, which may vary with locations;
- interested in performing site consolidation;
- has high availability requirement.

ii) the manager may also have preferences about the configuration of hardware and system. For example, some legacy applications must continue to run on

the platforms for which they were developed due to the lack of cross-platform ability. As any new placement plan is subject to various local IT managers' approval, placement recommendations made without taking managers' preferences into account may be inefficient due to managers' non-cooperation. Moreover, satisfying managers' preferences as much as possible can give them more incentive to migrate or take more workload, which helps the enterprise achieve optimal resource utilization state. Therefore, the objectives of consolidations in such enterprise environments are not only to minimize the number of physical servers required to operate, but also to maximize the managers' satisfaction of their preferences. In this paper, we propose a *satisfaction-oriented VM consolidation mechanism (SOC)* to achieve these objectives. In this mechanism, we call the party who wants to find capacity or workload the requester, and the other party that supplies capacity or workload the provider. The *SOC* system performs two key tasks. The first one is to allow managers to express their preferences and use matchmaking to find entities (VMs or PMs) that satisfy both parties (providers and requesters) the most. To the best of our knowledge, this is the first application of matchmaking in VM consolidation. Different from the conventional matchmaking, the input of *SOC*'s matchmaking process is the manager's preferences instead of demand or job descriptions. The second task is to apply *Resource Balancing Placement (RBP)*, a VM placement algorithm proposed in our previous work [14], to allocate VMs to PMs.

The main contributions of this paper are:

- An XML-based description language to specify IT managers' preferences, applying matchmaking to find entities that match the given preferences, and metrics to evaluate the satisfaction degree of a VM placement solution;
- A satisfaction-oriented VM consolidation mechanism that maximizes the managers' satisfaction while providing good placement of VMs on PMs;
- The implementation of a prototype of the proposed *SOC* system. Because there is no existing consolidation approach which considers managers' preferences, *SOC* is compared with two baseline strategies: matchmaking-only (MM), which plans placement based on preferences but does not try to consolidate, and placement-only (PL), which consolidates VMs but does not consider preferences. We demonstrate through experiments that *SOC* has similar satisfaction degree as MM, while not requiring too many hosts as MM does, but only an amount close to PL; most of the mappings (up to 97.6%) output from PL, however, violate given preferences.

The rest of the paper is organized as follows: Section 2 presents some related work, followed by an architectural description of the *SOC* system in Section 3. Section 4 discusses the matchmaking process of *SOC* and the definition of the satisfaction degree metrics. Section 5 presents the evaluation of *SOC*, and analysis of the impact of integrating preferences. Section 6 concludes this paper and discusses some possible future work.

2 Related Work

Research on workload management focuses on different aspects. In particular, the energy consumption aspect has received much attention. Verma et al. proposed an architecture, pMapper, and application placement algorithms [23] that minimize the power and migration costs while meeting a fixed performance Service Level Agreement (SLA). The authors later created a power model and studied the virtualization issues due to consolidation, and used the insights from those studies to design a power-aware application placement controller [24] built on pMapper. Feller et al. [10] proposed an energy-aware workload consolidation algorithm based on the Ant Colony Optimization. Beloglazov and Buyya proposed a technique for dynamic consolidation of VMs that can reduce energy consumption and maintain SLAs violation at a certain level [4], and developed algorithms for energy-efficient resource provisioning to VMs without violating the negotiated SLAs [3]. Apart from energy consumption, the data traffic between VMs (i.e., network) is another aspect being considered. An algorithm [17] was proposed to address the scalability issue of data center networks using network-aware VM placement. Algorithms that minimize communication costs and latency were also proposed for resource allocation in distributed clouds [2]. Biran et al. [5] formulated the network-aware VM placement problem as a Min Cut Ratio-aware VM Placement problem that considers not only local physical resources but also network resources, and introduced heuristics to solve it. In addition to these, there are also studies focusing on the SLA aspect [6,8,22], the contention between VMs [11,13], and the temperature in the data center [18]. However, the IT managers' preferences are not considered in the literature.

On the other hand, matchmaking has a long history in the area of demands and supplies, jobs and resources matching. Raman et al. [19] proposed the ClassAd matchmaking framework for the high throughput computing system Condor in 1998. The authors later proposed Gangmatch [20], a multilateral extension to ClassAd. Liu et al. [16] also extended ClassAd to support both single-resource and multiple-resource selection. Matchmaking was reinterpreted as a constraint problem in [15] where constraint-solving technologies were exploited to implement matching operations. Sycara et al. [21] defined a language and used it in a matchmaking process among software agents on the Internet. However, consolidation is not one of the objectives of these approaches, and matchmaking has not been applied in the area of VM placement to the best of our knowledge. Moreover, these conventional matchmaking approaches focus on request and resources matching, and are not applied to matching users' preferences. Recently, a study [7] proposed to extend the matchmaking process to consider users' preferences by using Conditional Preference Networks (CP-Nets) to describe, structure and reason users' preferences. However, this work focuses only on the impact of introducing CP-Nets to matchmaking, but not the schedule process of grid resources. This research is orthogonal to our approach, and can be leveraged to enhance the preference description to handle more complicated preferences.

3 SOC System Architecture

In this section, we describe the *SOC* system from a high level view as shown in Figure 1. The input of the system is the description of a manager's request. The final output is the mapping between VMs and PMs (placement solution) indicating which VM should be placed on which PM. There are two types of requests: VM requests, which seek capacity provided by PMs, and PM requests, which seek more workloads (VMs). As the processing of these two types of requests are similar, we describe only VM requests as examples in this paper. Each request contains three main elements: a list of VM identities, common characteristics of the listed VMs (e.g., site, city, hardware model), and the manager's preferences.

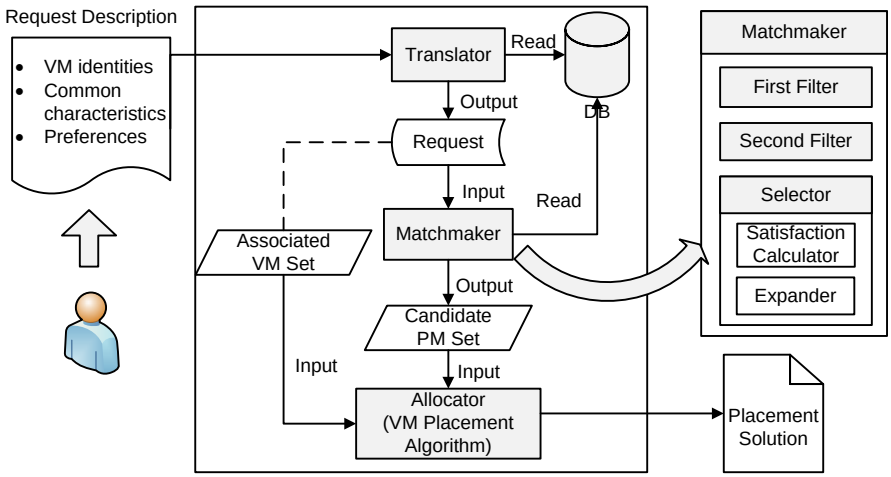


Fig. 1 Satisfaction-oriented VM Consolidation System Architecture

The system is composed of four components, namely the *database (DB)*, the *Translator*, the *Matchmaker*, and the *Allocator*. The DB contains information associated with all VMs and PMs in all data centers of an enterprise. In our prototype, DB includes hardware configurations and resource capacities of PMs, resource requirements of VMs, and locations and organizations of VMs and PMs. Managers' preferences are also recorded and linked to individual VM or PM in the DB once they are expressed. There are existing tools for monitoring and collecting hardware configuration information, and VM resources usage traces, such as IBM Tivoli Monitoring (ITM) [1], which is currently used by our industrial partner. VMs' resource demands can also be forecasted by several techniques such as the ones presented in [9, 12]. Filling the DB is, however, out of the scope of this paper.

Once a request is submitted, the Translator translates it from human language description to a request object in memory. The VMs specified in the

request are obtained from the DB. Then, the Matchmaker takes the request and the information in DB as inputs, and makes bidirectional matches between VMs and PMs considering both parties' preferences (VMs' preferences are specified in the request, PMs' preferences are specified in DB if there are any). A set of candidate PMs, which is the output of the Matchmaker, together with the VM set associated with the request, are then given to the Allocator which finds the mappings between VMs and PMs that optimize the PMs' resource utilization.

The Allocator focuses on the VM placement problem. It employs a VM placement algorithm to place the VMs specified in a request on PMs in the candidate PM set output from the Matchmaker. *SOC* is flexible to integrate any VM placement algorithm that takes a set of PMs and a set of VMs as inputs. In this paper, we employ an effective algorithm *RBP* [14], which minimizes the number of PMs required, and the resource wastage on each allocated PM by placing VMs that are complementary to PMs' residual capacity.

After the placement solution is approved by the managers of both parties, the actual migrations can be executed.

4 Matchmaking and Satisfaction Computation

The matchmaking process identifies the entities that satisfy both parties' preferences the most. In this section, we first show how requests and preferences are expressed, then we explain how the Matchmaker works, and how the satisfaction degree is computed. For simplicity, we consider only 2 dimensions (i.e., CPU and RAM) in the demonstration and the experiments. However, our approach is able to deal with multiple dimensions of resources.

4.1 Request and Preferences

4.1.1 Request

A manager triggers the system by submitting a request expressing its preferences. As shown in Figure 2, a request has a type and consists of three main elements: a list of VM identities, common characteristics of the listed VMs, and managers' preferences. In the example given in Figure 2, we use attributes such as site, city, hardware model that are common in data centers. However, the quantity and definition of the attributes is just a matter of agreement in the enterprise where the system is used, and our approach is general enough to accept any attributes as long as they are included in the DB.

The matchmaker in *SOC* performs bidirectional matching between requesters and providers (i.e., matches both parties' preferences). A request's preferences are matched with individual providers' characteristics. However, providers' preferences are matched with the request as a whole (instead of individual entities listed in the request) using the common characteristics specified in the request.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
  <Type>VM</Type>
  <IDs>
    <id>26</id>
    <id>75</id>
  </IDs>
  <Common_Characteristics>
    <site>S1</site>
    <city>C2</city>
    <min_cpu unit="MHz">627</min_cpu>
    <max_cpu unit="MHz">29549</max_cpu>
    <min_ram unit="MB">873</min_ram>
    <max_ram unit="MB">31442</max_ram>
  </Common_Characteristics>
  <Preferences>
    <Inflexible_Prefs>
      <site>S1</site>
      <cpu_qty>&gt;=4</cpu_qty>
    </Inflexible_Prefs>
    <Flexible_Prefs>
      <city weight="0.4">C3</city>
      <building weight="0.3">B2</building>
      <organization weight="0.2">Org5</organization>
      <hardware_model weight="0.1">Model10</hardware_model>
    </Flexible_Prefs>
  </Preferences>
</Request>
```

Fig. 2 Example of A Request in XML

4.1.2 Preferences

We define two types of preferences: *inflexible* and *flexible*. Inflexible preferences are those that must be satisfied, while flexible preferences are those that are preferred but acceptable if not matched. As exact match does not exist in many cases, the use of flexible preferences gives both parties more chances to find a close to ideal match. Each flexible preference is assigned a weight, so that one violation will only lower the satisfaction degree but not reject a match immediately. The higher the weight, the more important satisfying this preference is.

Flexible preferences have two types of attributes, *normal* and *categorized*. We discuss in Section 4.4 the different ways of computing their satisfaction degrees. Normal attributes are matched simply based on syntactic matching. For example, value "Org3" for organization name will match preference `<organization> Org3 </organization>`, and value "2" for quantity of CPUs will match preference `<cpu_qty> &gt;=2 </cpu_qty>` (`&gt;` stands for the greater than symbol "`>`"). Categorized attributes are categorized based on certain rules applied to their values. For instance, *site* can be a categorized attribute based on its expense level (e.g., power rate, cooling rate, salary rate and raised floor rate). Similarly, buildings can be categorized based on the

facilities they provide or their energy efficiency such as building energy rating. The matching of this type of attributes is to check if two values belong to the same category or how close their categories are.

4.2 Syntax and rules

We define a set of operators in Table 1 to express preferences and common characteristics more easily and comprehensively depending on managers' needs. Note that attributes can be of two types: *textual* (e.g., S1) and *numerical*. The last four operators in Table 1 are applicable to numerical values only. Numerical common characteristics can only be specific values or ranges (e.g., >2&<=8). Textual common characteristics can only be specific values (i.e., no operators), otherwise, their categories can not be identified and matched with providers' categorized preferences. Rules for using the operators are as follows:

- "|" can only be used alone (e.g., S1 | S2);
- "!" can be used alone or with "&" (e.g., !S1, !S1&!S2);
- "&" must be used with "!" or " > ", " >= ", " < ", " <= ".

Table 1 Operators for Expressing Preferences and Common Characteristics

Operators	Description	Preferences		Common Characteristics	
		Textual	Numerical	Textual	Numerical
|	or	✔	✔		
&	and	✔	✔		✔
!	not	✔	✔		
>	greater than		✔		✔
>=	greater than or equal to		✔		✔
<	less than		✔		✔
<=	less than or equal to		✔		✔

In a request, managers are free to choose the attributes to specify, and decide if they are inflexible or flexible preferences. However, they must provide the minimum and maximum resources requirement or capacity (i.e., min_cpu, max_cpu, min_ram, max_ram).

Furthermore, we set the following rule to clarify ambiguities and deal with missing information:

- a preference is considered as not satisfied if the attribute is not included in a request's common characteristics or an entity's characteristics.

If the dissatisfied preference is inflexible, the corresponding entities are considered as not matched. Therefore, the more specific the common characteristics are, the more candidate entities the request may find. Managers also need to

decide how firm they should insist on their preferences, as for a set of preferences, the more preferences specified as flexible, the bigger the chance to find a close to ideal match.

4.3 Matchmaker

The Matchmaker (see Figure 1) consists of two *Filters* and one *Selector*. It processes all VMs specified in one request as a whole as they share the request's preferences and common characteristics. Thus, the matching is between the request and each individual PM. The *First Filter* filters PMs that violate the request's inflexible preferences. The *Second Filter* filters PMs with inflexible preferences not satisfied by the request's common characteristics. The *Second Filter* also filters out PMs with a capacity smaller than the minimum resource requirement stated in the request, as no VM in the current request can be placed on them. The *Selector* then computes the satisfaction degree of each request-to-PM mapping (RPM), and selects the set of PMs with satisfaction degrees within a certain range. This is discussed in more details in Section 4.4 and Section 4.5 respectively.

4.4 Satisfaction Computation

Table 2 Key Notations and Description

Symbol	Description
K	Number of provider's preferences, $p = 1..K$
K'	Number of requester's preferences, $p' = 1..K'$
C	Number of possible categories of a categorized attribute
p_c	Category of a provider's characteristic c
$r_{c'}$	Category of a requester's characteristic c'
p_p	Category of a provider's flexible preference p
$r_{p'}$	Category of a requester's flexible preference p'
w_p	Weight of provider's flexible preference p, $\sum w_p = 1$
$w_{p'}$	Weight of requester's flexible preference p', $\sum w_{p'} = 1$
$P(p, c')$	A provider's satisfaction degree of flexible preference p
$R(p', c)$	A requester's satisfaction degree of flexible preference p'
Sp	A provider's total satisfaction degree of all preferences
Sr	A requester's total satisfaction degree of all preferences
S	Satisfaction degree of a RPM

As inflexible preferences are either satisfied or violated, the satisfaction computation applies only to flexible preferences. Table 2 shows the notations used in this paper. The satisfaction degree of one single preference is computed as presented in Equation (1) and (2), where c' is a requester's characteristic corresponding to preference p, and c corresponds to p'. If the corresponding c/c' does not exist, the satisfaction degree of preference p'/p is 0 according

to our rule in Section 4.2. For normal attributes, the satisfaction degree is a binary variable indicating matched or mismatched (1 or 0). For categorized attributes, it measures how far two categories are from each other. The closer they are, the higher the satisfaction degree of this preference is. For instance, if *site* is a categorized attribute based on its expense level and a manager prefers a site with the lowest expense, the second lowest site will be the best choice if the exact match does not exist. When the characteristic belongs to the same category the preference belongs to, the satisfaction degree is 1.

$$P(p, c') = \begin{cases} 1 & \textit{if } p \text{ is normal and satisfied} \\ 1 - (|p_p - r_{c'}|/C) & \textit{if } p \text{ is categorized} \\ 0 & \textit{other} \end{cases} \tag{1}$$

$$R(p', c) = \begin{cases} 1 & \textit{if } p' \text{ is normal and satisfied} \\ 1 - (|r_{p'} - p_c|/C) & \textit{if } p' \text{ is categorized} \\ 0 & \textit{other} \end{cases} \tag{2}$$

The satisfaction degree of a provider (Sp) or a requester (Sr) is computed as the weighted sum of all of its flexible preferences' satisfaction degrees. However, some PMs or VMs may not have flexible preferences or any preferences assigned at all. In that case, considering the satisfaction degree as 0 is unfair as there is no violation. On the other hand, it should not be 1 as exactly matched entities should have priority over entities with no preference specified. Therefore, if there is no flexible preference specified, Sp and Sr are defined to be less than an exact match (value 1) and greater than the closest categorized preference match. Sp and Sr are expressed as below:

$$Sp = \begin{cases} \sum_{p=1}^{K} w_p P(p, c') & \textit{if flexible preference } p \textit{ exists} \\ (1 - 1/C) + \epsilon & \textit{if no flexible preference is specified} \end{cases} \tag{3}$$

$$Sr = \begin{cases} \sum_{p'=1}^{K'} w_{p'} R(p', c) & \textit{if flexible preference } p' \textit{ exists} \\ (1 - 1/C) + \epsilon & \textit{if no flexible preference is specified} \end{cases} \tag{4}$$

The satisfaction degree for the closest categorized preference match is $(1-1/C)$ computed using Equation 1 or 2 (with $|p_p - r_{c'}| = 1$ or $|r_{p'} - p_c| = 1$). ϵ is a small positive value that keeps Sp or Sr greater than the closest match but less than 1. We define ϵ as:

$$\epsilon = \frac{(1 - (1 - 1/C))}{2} = \frac{1}{2C} \tag{5}$$

Finally, the satisfaction degree of a RPM (S) is the sum of the satisfaction degrees of the provider and the requester:

$$S = Sp + Sr \tag{6}$$

4.5 Expander and Further Placement

After computing the satisfaction degree S for all PMs, they are grouped into different levels ($Sp, Sr \in [0, 1]$, hence $S \in [0, 2]$): level 1 contains PMs having satisfaction degrees within range $[2, 1.6)$, level 2 is $[1.6, 1.2)$, level 3 is $[1.2, 0.8)$, level 4 is $[0.8, 0.4)$ and level 5 is $[0.4, 0]$. Section 5.5.2 presents further discussion on the separation of satisfaction levels. The Expander examines the total resource demands of all the VMs in the request, and the total capacity of all the PMs in level 1. If the total capacity is not enough to support all VMs, it expands to the next level and repeats the examination and expansion if necessary until the total capacity is larger than or equal to the total demands. The PM set output from the Matchmaker then satisfies all inflexible preferences of the request and satisfies the flexible preferences the most.

As the expansion depends only on the total capacity and total requirements, it may happen, though not often, that the PMs selected by the Expander can not accommodate all VMs eventually. Because it is very unlikely that the VMs can fit each PM perfectly, so that a small amount of resources on each PM might be wasted. To handle this situation, *SOC* examines the solution output from the Allocator whether all VMs have been placed. If not, *SOC* repeats all processes, from the Matchmaker to the Allocator, using the remaining PMs and VMs.

5 Experimental Results

In this section, we first introduce the evaluation metrics and experimental setup, and then present two sets of experiments to evaluate the performance of *SOC* and analyze the impact of varying managers' preferences.

5.1 Metrics

Because there is no existing consolidation approach considering managers' preferences, we compare *SOC* with two baseline strategies: placement-only (PL) and matchmaking-only (MM). PL does not consider preferences but use *RBP* to place VMs. MM selects for each VM the PM that maximizes the satisfaction degree (S') of a VM-to-PM mapping (VPM), which is computed similarly as the satisfaction degree (S) of a RPM. The difference is that S considers a request's common characteristics, and S' considers an individual VM's characteristics. MM uses S' instead of S because it checks each VM-PM pair, hence the individual VM's characteristics are available. *SOC* uses S as it considers all VMs in a request as a whole. However, S' is applicable in measuring *SOC's* solution, since the mappings between VMs and PMs are already determined in the solution. Therefore, we use:

- the execution time, to measure *SOC's* efficiency,
- the number of hosts required, to measure the quality of the placements,

– and the average satisfaction degree (S'_{avg}) of all VPMs in a solution, to measure the satisfaction degree of a solution, which is expressed as follows:

$$S'_{avg} = \frac{\sum_{i=0}^{N-1} S'_i}{N-1}$$

N being the number of VMs in the solution, which is equal to the number of VMs specified in the request.

5.2 Dataset and Parameters

In our previous work [14], we defined and generated a benchmark which covers a wide range of possible data centers with different demographics. In that benchmark, we use *shape* and *size* to represent VMs or PMs, and identified 5 distributions (i.e., normal, u-shaped, uniform, j-shaped and reverse j-shaped) for *shape* and *size* respectively. Based on these distributions, we generated 25 VM sets and 25 PM sets. In this paper, we generated a mixed VM set (2,000 VMs) by randomly selecting 80 VMs from each of the 25 VM sets in the benchmark, and a mixed PM set (1,000 PMs) by randomly selecting 40 PMs from each of the 25 PM sets in the benchmark, in order to simulate the heterogeneity of VMs and PMs.

The request used in the experiments is similar to the example shown in Figure 2. The difference lies in the number of VMs listed, and the minimum and maximum values of CPU and RAM of them. There are only two `<id>` elements between tag `<IDs>` and `<\IDs>` in the example, however, the request used in the experiments lists 500 VMs (i.e., 500 `<id>` elements with randomly selected ids) to be consolidated, which have common site (i.e., S1) and city (i.e., C2) as the example request. The minimum and maximum values of CPU and RAM in the common characteristics are also different and accordingly correspond to the 500 listed VMs.

Table 3 Attributes Used in the Experiment

Attribute	Value	Type
site	$S1, \ldots, S7$	Categorized
city	$C1, \ldots, C9$	Categorized
building	$B1, \ldots, B10$	Categorized
organization	$Org1, \ldots, Org8$	Normal
hardware_model	$M1, \ldots, M9$	Normal (PM only)
cpu_qty	$2, 4, 8$	Normal (PM only)

Table 3 shows the attributes used in the characteristics and preferences in the experiments. We set the number of categories to 5, hence $C = 5, \epsilon = 0.1$. PMs in the mixed PM set have all listed attributes with values randomly assigned. VMs in the mixed VM set have the first four attributes. As the request lists 500 VMs having common site and city to be consolidated, 500

VMs of the mixed VM set (selected according to the ids listed in the request) have $site = S1$ and $city = C2$, and values randomly assigned to the other attributes; the remaining $1,500$ VMs have values randomly assigned to all attributes.

5.3 Experimental Setup

We conducted two sets of experiments: i) simulating real-world scenarios to evaluate the performance of the proposed *SOC* system in terms of the number of PMs required, the satisfaction degree, and the execution time (see Section 5.4); ii) varying the number of preference sets and the number of PMs with preferences assigned to analyze their impacts on the solutions (see Section 5.5). In these experiments, the VM set, the PM set and the request are fixed as described above. However, the preferences of PMs are alterable to simulate scenarios with different preferences. We created five preference sets for PMs to simulate the preferences of five managers, each managing a subset of the PM set. Each preference set consists of a group of attributes, which are fixed, but with randomly assigned values.

We ran ten experiments for the performance evaluation with the five preference sets randomly assigned to 800 PMs in each run. One group of values from Table 4 are used for each experiment. We did not assign preference sets to all PMs because in the real world it happens that some PMs have no preference specified. In the second set of experiments, we use the first group of values in Table 4, and change the number of preference sets (i.e., the number of managers who specify preferences) assigned (C_{ps}) (e.g., assign pref 1; assign pref 1 and 2; assign pref 1, 2 and 3 etc.), and the number of PMs to which those preference sets are assigned (C_{pm}) (i.e., 200, 400, 600, 800, $1,000$). The PMs with preferences assigned were picked randomly and differentiated five times for each combination of C_{ps} and C_{pm} to compute the average results. All experiments are carried out on a laptop with 8GB RAM and an Intel Core i7-2760QM 2.4GHz CPU running Windows 7.

5.4 Performance Evaluation

5.4.1 Quality

Figure 3a shows the quality of placement (i.e., number of hosts required) of the three approaches. The x-axis indicates the indexes of the ten experiments with different preference sets and values assigned. As PL does not consider preferences, the number of hosts required in each experiment is the same (i.e., 15). *SOC* requires 54.1% fewer hosts than MM on average, while being very close to PL except for the last experiment (i.e., using preference values of group 10). In that experiment, MM also requires much more PMs. As the PMs are heterogeneous in terms of capacity and are assigned different preferences, one

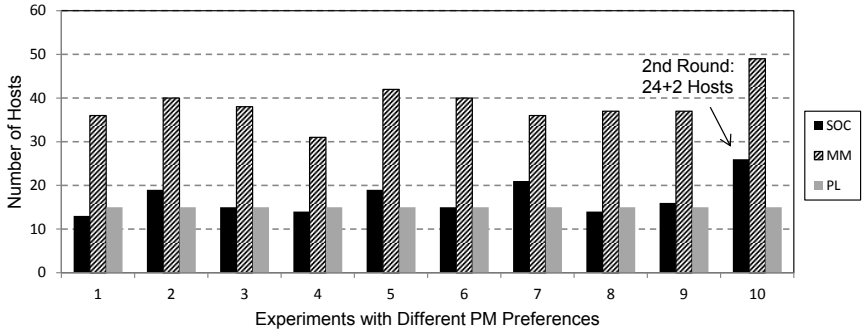
Table 4 Preference Sets for PMs Used in The Experiments with Ten Groups of Values

Prefs	Types/Attributes	Weights	Values				
			1	2	3	4	5
Pref 1	Inflexible: Site		!S4	S7	S6	S7	S3
	Flexible: City	0.5	C8	C2	C8	C3	C6
	Flexible: Building	0.3	B10	B2	B5	B9	B10
	Flexible: Organization	0.2	Org1	Org2	Org3	Org7	Org3
Pref 2	Inflexible: Site		S6	!S3&!S4	!S1&!S2	S2	S7
	Inflexible: Organization		Org8|Org5	Org5	Org4	Org3	Org6
	Flexible: City	0.7	C1	C8	C4	C4	C5
	Flexible: Building	0.3	B5	B1	B8	B7	B1
Pref 3	Flexible: Site	0.4	S1	S1|S4	S1	S6	S5
	Flexible: City	0.3	C4	C4	C2	C3	C3
	Flexible: Building	0.2	B2	B6	B5	B6	B4
	Flexible: Organization	0.1	Org5	Org7	Org4	Org6	Org3
Pref 4	Inflexible: Organization		Org6	Org2	Org1	Org1	Org2
	Flexible: Site	0.8	S1	S4	S1	S2	S1
	Flexible: City	0.2	C2	C8	C1	C1	C6
Pref 5	Flexible: Site	0.6	S2	S4	S6	S1	S1
	Flexible: City	0.1	C9	C5	C4	C8	C2
	Flexible: Organization	0.3	Org4|Org1	Org3	Org7	Org7	Org5

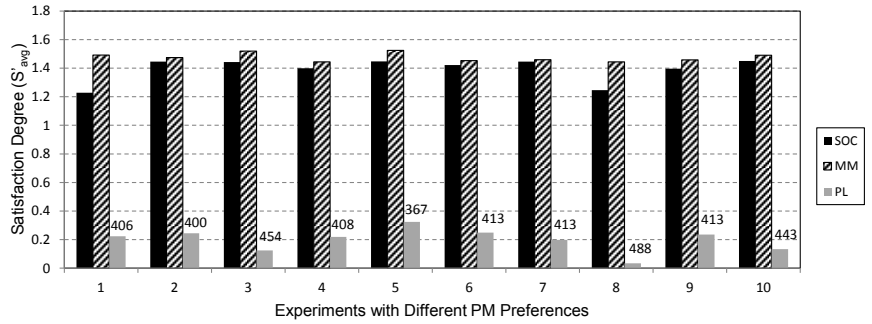
Prefs	Types/Attributes	Weights	Values				
			6	7	8	9	10
Pref 1	Inflexible: Site		S7	S1	S2	S2	S1
	Flexible: City	0.5	C3	C4	C3	C1	C5
	Flexible: Building	0.3	B10	B4	B6	B10	B2
	Flexible: Organization	0.2	Org2	Org2	Org5	Org1	Org8
Pref 2	Inflexible: Site		S3	S6	S3	S7	S1
	Inflexible: Organization		Org2	Org3	Org6	Org8	Org5
	Flexible: City	0.7	C9	C9	C9	C9	C6
	Flexible: Building	0.3	B2	B8	B2	B2	B3
Pref 3	Flexible: Site	0.4	S7	S4	S2	S2	S4
	Flexible: City	0.3	C5	C5	C1	C1	C2
	Flexible: Building	0.2	B6	B1	B9	B8	B5
	Flexible: Organization	0.1	Org5	Org2	Org4	Org1	Org4
Pref 4	Inflexible: Organization		Org8	Org4	Org5	Org1	Org7
	Flexible: Site	0.8	S4	S5	S3	S7	S3
	Flexible: City	0.2	C2	C2	C9	C8	C8
Pref 5	Flexible: Site	0.6	S2	S5	S5	S3	S6
	Flexible: City	0.1	C2	C9	C7	C2	C3
	Flexible: Organization	0.3	Org8	Org8	Org8	Org3	Org4

possible reason for this higher number of PMs required is that the qualified candidate PMs have small capacity. In order to investigate if this is the reason causing more hosts required, we performed an experiment using preference values of group 10 and homogeneous PMs, which have CPU and RAM set to the average value of the original PM set. The result (see Figure 4) validates our assumption that *SOC* requires the same amount of PMs as PL when the PMs are homogeneous, while the other experimental parameters remain the same. Therefore, the big amount of hosts required in the 10th experiment is

inevitable due to the constraint of the inflexible preferences, which filtered out PMs with big capacity. Moreover, it is worth noting that the 10th experiment is the only one that needs a second round of placement (described in Section 4.5). In this experiment, *SOC* first selects the PMs in the first two satisfaction levels (i.e., 3 PMs in Level 1 and 24 PMs in Level 2) as candidate PMs, because the total capacity of them is enough for the VMs' total demands. However, it turns out that 3 of the selected PMs have too small CPU capacity, though big enough RAM capacity, thus, no VM can fit into them. It results in 461 VMs placed on 24 PMs after the first round of placement. Consequently, the system performs the second round of placement, taking the remaining VMs and PMs as inputs. The second round of placement uses 2 PMs for the remaining 39 VMs, and leads to 26 PMs in total as presented in Figure 3a. It is noteworthy that the number of hosts required by *SOC* can even be fewer than PL in a few cases. The VM placement algorithm employed in *SOC* and PL is a heuristic algorithm. It has been proven [14] to be able to produce near optimal solutions much faster than an optimization solver, however, it does not guarantee to find the optimal solution. Therefore, there is a chance that a solution using fewer hosts can be found, especially when the hosts are heterogeneous.

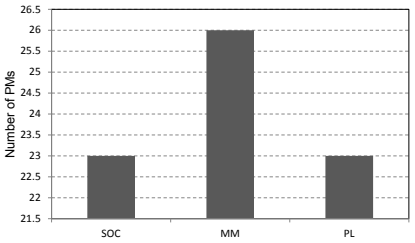


(a) Number of Hosts (The lower the better)

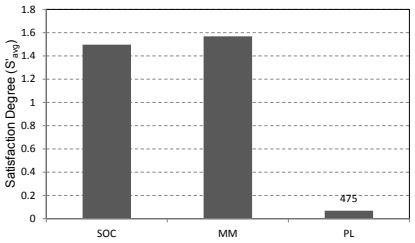


(b) Satisfaction Degree (The higher the better; numbers above PL bars are the number of mapping violations)

Fig. 3 Performance Evaluation Results of SOC, MM and PL



(a) Number of PMs



(b) Satisfaction Degree (Numbers above PL bars are the number of mapping violations)

Fig. 4 Experimental Results with Homogeneous PMs (using preference values of group 10)

Figure 3b shows how well the preferences of both the requester and the providers are satisfied. The numbers above the PL bars are the number of mapping violations. We count a mapping violation if at least one of the given inflexible preferences of the request or the PM in a mapping is violated. The satisfaction degrees of *SOC* are very close to that of MM. As $Sp, Sr \in [0, 1]$, hence $S, S' \in [0, 2]$, and $S'_{avg} \in [0, 2]$. Given the diversity of the attributes and the requirement of meeting both parties' preferences instead of only one's, it is very unlikely for S'_{avg} to reach the maximum value. The results also reveal that planning VM consolidation without considering the managers' preferences has a big chance to violate the managers' preferences. Within the ten experiments we ran, the lowest mapping violation is as high as 400 (80%), and the highest one reaches 488 (97.6%).

5.4.2 Efficiency

We present the execution time of *SOC*, PL and MM in Figure 5, which shows that the execution time of *SOC* for a problem size of $2,000$ VMs and $1,000$ PMs is slightly more than 1 second. The 10th experiment takes a bit extra time of about 0.1 second as it performs a second round of placement. There is an overhead of approximately 0.2 second for the matchmaking process in *SOC* compared to PL. However, this overhead is much less than the time taken by MM, which performs pure matchmaking. Although *SOC* takes more time than PL and MM, 1 second is very short time, thereby we consider *SOC* efficient.

5.5 Analysis of the Impact of Preferences

In this section, we demonstrate further why it is important to consider managers' preferences, and how it affects the results when the number of preference sets and the number of PMs that have preferences assigned changes.

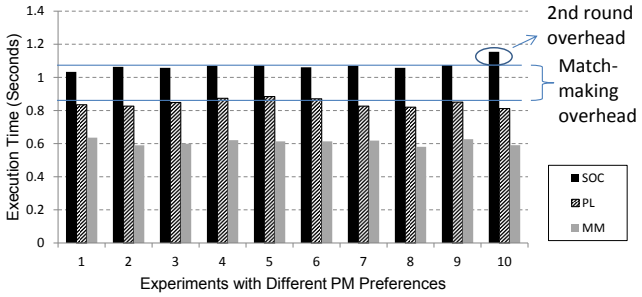


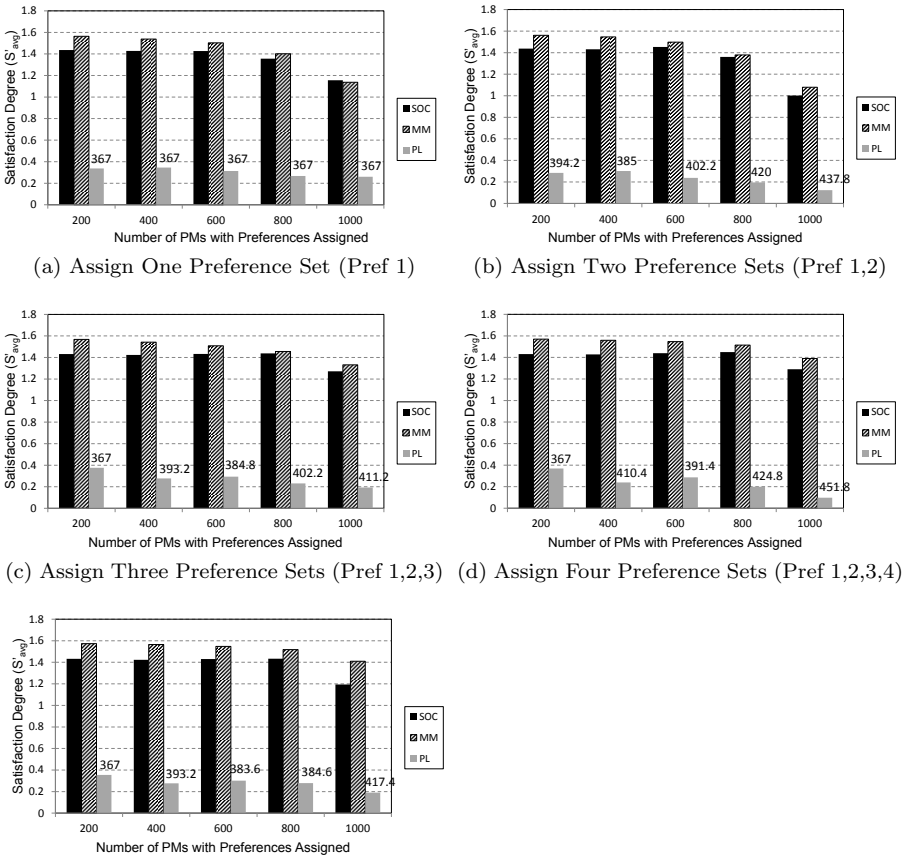
Fig. 5 Execution Time of SOC, MM and PL

5.5.1 Impact on Satisfaction Degree and Violation

We present in Figure 6 the average satisfaction degree for the five runs of each experiment. The results are grouped by the number of preference sets assigned. As shown in Figure 6a, the number of mapping violations of PL reaches 367 even when only one preference set is assigned to 200 PMs. As this specific number of violations appears multiple times in the results and does not change in Figure 6a, we believe that these violations are caused by PMs violating the VM request's inflexible preferences, and any increment comes from VMs violating PMs' inflexible preferences. As the placement solutions of PL are the same in every experiment since it does not consider preferences, changing the preferences of those PMs that are not selected as target PMs in PL, or adding preferences that are not violated by the VMs will not cause changes in the number of mapping violations. For example, the VMs in the request have $site = S1$ which never violate the inflexible preference of Pref 1 with value in group 1(i.e., $!S4$), thus, the number of mapping violations in Figure 6a remain the same when the number of PMs with preferences assigned increases. However, when any of the selected PMs are assigned preferences that the VMs violate, the count of violations increases.

The results in Figure 6 also show that: i) increasing the number of preference sets has only a slight impact on the satisfaction degree of SOC and MM; ii) increasing the number of PMs with preferences assigned decreases the S'_{avg} of MM, which implies that it is more difficult to find a mapping satisfying both parties' interests perfectly; iii) increasing the number of PMs with preferences assigned does not always decrease the S'_{avg} of SOC, which implies that SOC is not very vulnerable to the change of the number of PMs with preferences assigned. However, SOC's S'_{avg} is inevitably reduced when all PMs are assigned preferences.

Although MM selects for each VM the PM that maximizes the satisfaction, it focuses on one VM at a time without seeing the satisfaction degree of other VPM. Thus, SOC can have higher S'_{avg} than MM as in Figure 6a ($S'_{avg}(SOC) = 1.156$, $S'_{avg}(MM) = 1.136$ for $1,000$ PMs). Figure 7 uses two hosts (H1, H2) and two VMs (VM1, VM2) to illustrate the case in which SOC has higher S'_{avg} than MM. As shown in Figure 7a and 7b, the satisfaction



(a) Assign One Preference Set (Pref 1)

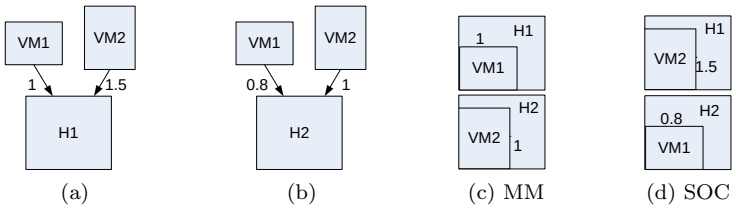
(b) Assign Two Preference Sets (Pref 1,2)

(c) Assign Three Preference Sets (Pref 1,2,3)

(d) Assign Four Preference Sets (Pref 1,2,3,4)

(e) Assign Five Preference Sets (Pref 1,2,3,4,5)

Fig. 6 Impact of Preferences on PMs' Satisfaction Degree (Numbers above PL bars are the average number of mapping violations)



(a) (b) (c) MM (d) SOC

Fig. 7 Example of SOC Having Higher S'_{avg} Than MM

degree of VPM between VM1 and H1 $S'_{VM1-to-H1} = 1$, and the satisfaction degree of VPM between VM1 and H2 $S'_{VM1-to-H2} = 0.8$. MM processes one VM at a time. When processing VM1, it determines that H1 is a better host

than H2 for VM1. After VM1 is placed, there is no space on H1 for VM2. Thus, VM2 is placed on H2 as shown in Figure 7c, generating $S'_{avg} = 1$. However, if VM2 is placed on H1 and VM1 is placed on H2 (Figure 7d), the average satisfaction degree S'_{avg} would be 1.15.

5.5.2 Discussion on Satisfaction Levels

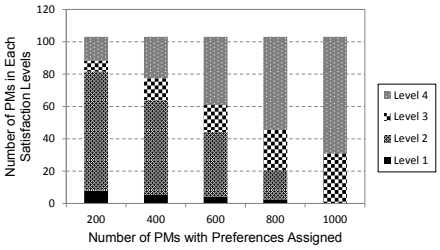
We also investigate the impact on satisfaction levels and discover that as the number of PMs with preferences assigned increases, the number of PMs having high satisfaction degree (i.e., Level 1 and 2) drops and those with low satisfaction degree (i.e., Level 3 and 4) grows as shown in Figure 8. However, the proportion of the number of PMs in each satisfaction level follows a similar trend when the number of preference sets increases.

From the results, we can see that it is tricky to define a fixed border and number of satisfaction levels for all cases. Because this depends not only on the PMs' capacity and VMs' demands, which determines how many PMs are enough to support a certain set of VMs, but also the number of PMs with preferences assigned, which affects the distribution of satisfaction degrees. Having too many levels may result in too many expansions and lower the efficiency of the system to some extent. More important, each level has fewer PMs when there are more levels, meaning smaller expansion steps. Thus, the expansion is more likely to stop when the PMs have the exact amount of capacity the VMs require, with no spare capacity. In this case, it is more likely to require a second round of placement as discussed in Section 5.4.1. On the other hand, too few levels will increase the search space and will not help in focusing on PMs with high satisfaction degrees.

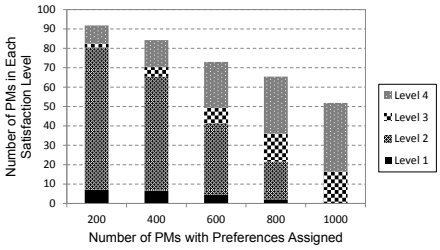
We explore this one possible separation of the levels in our prototype as described in Section 4.5. Among the first set of experiments we ran, 77.8% of them need to expand to Level 2, and 22.2% need Level 3. In the second set of experiments 76% of them need Level 2, and 20% need Level 3. These results show that this separation is reasonable, though there is space for improving it. However, as implied by the results, the levels do not need to be adjusted when new managers add preference sets into the system.

6 Conclusion and Future Work

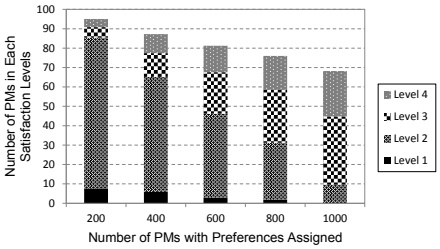
This paper proposes *SOC*, a satisfaction-oriented VM consolidation system, which uses matchmaking and a VM placement algorithm (*RBP*) to provide consolidation solutions that satisfy IT managers' preferences and optimize resource utilization in enterprises. To the best of our knowledge, this is the first application of matchmaking techniques to the VM placement problem to match managers' preferences. *SOC* is a flexible mechanism which allows a wide range of expression of references, as long as the corresponding information is available in the DB. Our experiments show two important findings: i) a placement solution not taking into account preferences risk hampering



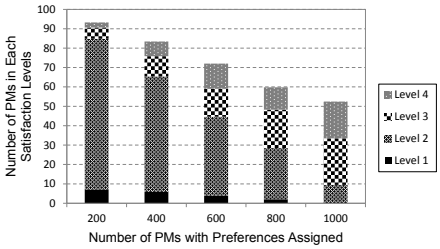
(a) Assign One Preference Set (Pref 1)



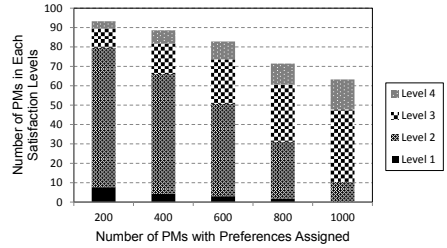
(b) Assign Two Preferences Sets (Pref 1,2)



(c) Assign Three Preference Sets (Pref 1,2,3)



(d) Assign Four Preferences Sets (Pref 1,2,3,4)



(e) Assign Five Preferences Sets (Pref 1,2,3,4,5)

Fig. 8 Impact of Preferences on The Number of PMs in Each Satisfaction Level (the number of PMs in Level 5 is 0)

managers' satisfaction, which is a big issue in decentralized organizations such as enterprises; ii) *SOC* produces near optimal placements that also give the most satisfaction to the managers.

It would be interesting to investigate in the future the separation of the satisfaction levels further and define an adaptive separation of the levels. Moreover, the proposed mechanism can be extended by considering more real world constraints such as the interference between co-located VMs (e.g., performance degradation).

Acknowledgements This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie) and by Enterprise Ireland Innovation Partnership in cooperation with IBM and University

College Dublin under grant IP/2010/0061. Anthony Ventresque was supported, in part, by Science Foundation Ireland Industry Fellowship grant 12/IF/12789.

References

1. Best practices for identifying and resolving infrastructure problems. URL `http://www-03.ibm.com/software/products/us/en/tivomoni/`. Accessed: 2014-03-16
2. Alicherry, M., Lakshman, T.: Network aware resource allocation in distributed clouds. In: Proceedings of the 31st International Conference on Computer Communications, INFOCOM '12, pp. 963–971. IEEE (2012)
3. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. **28**(5), 755–768 (2012)
4. Beloglazov, A., Buyya, R.: Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In: Proceedings of the International Workshop on Middleware for Grids, Clouds and e-Science, MGC '10, pp. 4:1–4:6. ACM (2010)
5. Biran, O., Corradi, A., Fanelli, M., Foschini, L., Nus, A., Raz, D., Silvera, E.: A stable network-aware vm placement for cloud systems. In: Proceedings of the International Symposium on Cluster, Cloud and Grid Computing, CCGRID '12, pp. 498–506. IEEE (2012)
6. Breitgand, D., Epstein, A.: Sla-aware placement of multi-virtual machine elastic services in compute clouds. In: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, IM '11, pp. 161–168. IEEE (2011)
7. Cafaro, M., Mirto, M., Aloisio, G.: Preference-based matchmaking of grid resources with cp–nets. J. Grid Comput. **11**(2), 211–237 (2013)
8. Cardellini, V., Casalicchio, E., Lo Presti, F., Silvestri, L.: Sla-aware resource management for application service providers in the cloud. In: Proceedings of the International Symposium on Network Cloud Computing and Applications, NCCA '11, pp. 20–27 (2011)
9. Chen, M., Zhang, H., Su, Y.Y., Wang, X., Jiang, G., Yoshihira, K.: Effective vm sizing in virtualized data centers. In: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, IM '11, pp. 594–601. IEEE (2011)
10. Feller, E., Rilling, L., Morin, C.: Energy-aware ant colony based workload placement in clouds. In: Proceedings of the IEEE/ACM 12th International Conference on Grid Computing, GRID '11, pp. 26–33 (2011)
11. Govindan, S., Liu, J., Kansal, A., Sivasubramaniam, A.: Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC '11, pp. 22:1–22:14 (2011)
12. Isci, C., Hanson, J., Whalley, I., Steinder, M., O.Kephart, J.: Runtime demand estimation for effective dynamic resource management. In: Proceeding of the IEEE Network Operations and Management Symposium, NOMS '10, pp. 381–388
13. Jin, H., Qin, H., Wu, S., Guo, X.: Ccap: A cache contention-aware virtual machine placement approach for hpc cloud. International Journal of Parallel Programming pp. 1–18 (2013)
14. Li, X., Ventresque, A., Murphy, J., Thorburn, J.: A fair comparison of vm placement heuristics and a more effective solution. In: Proceedings of the 13th International Symposium on Parallel and Distributed Computing, ISPDC '14. IEEE (2014)
15. Liu, C., Foster, I.: A constraint language approach to matchmaking. In: Proceedings of the 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications, RIDE '04, pp. 7–14. IEEE (2004)
16. Liu, C., Yang, L., Foster, I., Angulo, D.: Design and evaluation of a resource selection framework for grid applications. In: Proceedings of the 11th International Symposium on High Performance Distributed Computing, HPDC '02, pp. 63–72. IEEE (2002)
17. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of the 29th Conference on Information Communications, INFOCOM '10, pp. 1154–1162 (2010)

18. Moore, J., Chase, J., Ranganathan, P., Sharma, R.: Making scheduling "cool": Temperature-aware workload placement in data centers. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '05, pp. 5–5 (2005)
19. Raman, R., Livny, M., Solomon, M.: Matchmaking: Distributed resource management for high throughput computing. In: Proceedings of the 7th International Symposium on High Performance Distributed Computing, HPDC '98, pp. 140–146. IEEE (1998)
20. Raman, R., Livny, M., Solomon, M.: Policy driven heterogeneous resource co-allocation with gangmatching. In: Proceedings of the 12th International Symposium on High Performance Distributed Computing, HPDC '03, pp. 80–89. IEEE (2003)
21. Sycara, K., Widoff, S., Klusch, M., Lu, J.: Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. Autonomous Agents and Multi-Agent Systems **5**(2), 173–203 (2002)
22. Van, H.N., Tran, F.D., Menaud, J.M.: Sla-aware virtual resource management for cloud infrastructures. In: Proceedings of the 9th International Conference on Computer and Information Technology - Volume 02, CIT '09, pp. 357–362. IEEE (2009)
23. Verma, A., Ahuja, P., Neogi, A.: pmapper: Power and migration cost aware application placement in virtualized systems. In: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, Middleware '08, pp. 243–264 (2008)
24. Verma, A., Ahuja, P., Neogi, A.: Power-aware dynamic placement of hpc applications. In: Proceedings of the 22nd Annual International Conference on Supercomputing, ICS '08, pp. 175–184. ACM (2008)
25. Vogels, W.: Beyond server consolidation. Queue **6**(1), 20–26 (2008)