Multidimensional Workload Consolidation for Enterprise Application Service Providers

Full paper

Hendrik Müller
Otto-von-Guericke-University Magdeburg
hendrik.mueller@ovgu.de

Sascha Bosse
Otto-von-Guericke-University Magdeburg
sascha.bosse@ovgu.de

Abstract

In the domain of enterprise applications, operational costs can be reduced by consolidating orthogonal workloads with the objective of maximizing server utilization levels and minimizing the total amount of required capacity. This is closely related to the well-known bin packing problem which is NP-hard. Related problem formulations often consider varying historical workload traces, but include only one resource dimension, usually the CPU. This implicates a serious risk of overloading other resources that are not related to CPU demands, such as memory. Therefore, we formulate the multidimensional workload consolidation problem and develop eight algorithms to provide solutions. We evaluate their applicability using workload traces gathered from four data centers. A best-fit heuristic that uses a genetic algorithm provides best solution qualities with lowest variance and revealed up to 53.39 percent of unused capacity. In general, multidimensional workload consolidation problems eliminate less server capacity, but effectively reduce the risk of resource overloads.

Keywords

Capacity management, service operation, relocation, heuristics, genetic algorithm.

Introduction

Energy costs have dramatically increased in recent years and are to become a major factor in the total cost of ownership of data centers (Filani et al. 2008; Orgerie et al. 2014). Although energy efficiency of hardware has been improved in recent years, at the same time, the energy consumption of data centers has increased by 56 percent from 2005 to 2010 (Koomey 2011; Splieth et al. 2015). Therefore, achieving energy-efficient IT operations still remains a crucial task for enterprises in order to save running costs.

The greatest energy consumer in servers is known to be the CPU (Barroso et al. 2013; Splieth et al. 2015). A case study that we performed as preparatory work identified low CPU utilization levels on average of 11.68 percent across four enterprise data centers, comprising 216 physical servers and 364 business applications. CPU utilization can be used to approximate the overall system utilization (Pelley et al. 2009), which was estimated by the Gartner Group in 2005 to be less than 20 percent in data centers (Speitkamp and Bichler 2010). According to a newer Gartner report from 2011, servers even run at average utilization levels of less than 15 percent (Gartner 2011). Such low utilization rates adversely affect energy consumption, since servers show high energy consumption of up to 70 percent of their peak power even in idle state (Beloglazov and Buyya 2010). Therefore, energy management has become a key issue for data center operations in order to reduce all energy-related costs, including capital and operating expenses (Barroso and Hölzle 2007). Previous extensive efforts investigated the potential of consolidating application workloads to achieve higher energy savings (Xu and Fortes 2010) by eliminating unused hardware resources. The consolidation of orthogonal workloads on a single server is enabled by the concept of virtualization techniques (Ferreto et al. 2011). For enterprise applications, the workload is often repetitive and, thus, predictable in seasonal patterns (Bichler et al. 2006; Rolia et al. 2005) such as daily or weekly recurrences (Setzer and Stage 2010). In case no workload patterns can be identified, server allocations need to be continuously adjusted as supported by virtualization layers that involve hypervisors and provide live migration capabilities. We will refer to this category of consolidation strategies as online consolidation. On the contrary, offline consolidations benefit from fixed allocations eliminating performance degradations, which result from the virtualization layer overhead and recurring...
migrations. Instead, resource demands of applications can be gathered from historical workload traces. Many offline consolidation strategies are based on the peak demand of these traces, e.g. (Gao et al. 2013; Stillwell et al. 2010; Xu and Fortes 2010). However, a static consolidation in terms of a single workload value that represents the application’s peak demand leads to bad resource utilization (Mi et al. 2010), because workloads can change significantly over time (Petrucci et al. 2011). Therefore, we focus on dynamic offline consolidation problems, which include a time dimension and allow for identifying workload patterns. Using these patterns, enterprise applications can be placed on available physical servers aiming at a maximized average utilization and a minimized amount of required capacity while the performance of IT systems must not be degraded significantly (Beloglazov and Buyya 2010). This describes a combinatorial problem and is closely related to the well-known bin packing problem in which the overall size of used bins is to be minimized. Even the simplest form of bin packing is proven to be NP-hard (Bichler et al. 2006).

Deduced from the CPU’s high contribution to the total server power (Barroso and Hölsle 2007) and from its known role to be the main bottleneck for enterprise applications (Mi et al. 2010; Speikamp and Bichler 2010), most of the related work we studied focuses on the CPU dimension when consolidating application workloads. This leads to a serious risk of overloading other resources, since the demand of an application associated with one system resource may not be related in any way to its demand for other resources. For instance, applications having low CPU consumption may have large memory footprints (Torres et al. 2008). For consolidated workloads, it needs be ensured that they never exceed a server’s capacity regarding all server resources in order to avoid performance problems and associated SLA violations (Setzer and Stage 2010). However, multiple memory intensive applications, placed on the same physical server, can affect each other’s performance severely, e.g. due to interference in their memory bandwidth availability (Nathuji et al. 2010). Such performance degradation produces application slowdowns and can defeat the benefit of consolidation, resulting in violation of customer quality of service (QoS) constraints (Dwyer et al. 2012; Nathuji et al. 2010). Therefore, other resource demands, besides CPU, need to be considered when consolidating enterprise application workloads in order to guarantee response times that are compliant with existing service level agreements. For this purpose, additional dimensions for each considered resource need to be included in the introduced bin packing problem.

Hence, our work is based on the following research questions: How to formulate and solve a dynamic multidimensional workload consolidation problem that avoids resource overloads and is applicable to today’s heterogeneous data centers? Which algorithms are appropriate in terms of their resulting solution qualities and computing times? In this paper, we use the dimensions time, CPU, and main memory for the evaluation of solution algorithms to the formulated multi-dimensional workload consolidation problem. We have chosen main memory, since future system performance is likely to be limited by inadequate memory capacity (Lim et al. 2009), particularly with a view to modern in-memory computing trends. Only if sufficient memory capacity is ensured, CPU resources become a bottleneck to manage in capacity management exercises (Cherkasova and Rolia 2006). In addition, memory usage shows significant temporal variations for different application and workload types (Lim et al. 2009), which indicates grave consequences when being ignored. Our approach was developed according to the design science research paradigm (Hevner et al. 2004). Therefore, the rest of the paper is structured as follows. In the following Section, related work is reviewed and classified regarding their application areas. A mathematical formulation of the multidimensional workload consolidation problem is given in the Section “Problem Formulation”. In the subsequent section, “Solution Algorithms”, optimization approaches including heuristics, metaheuristics, and hybrid algorithms are introduced. Those are applied on workload traces gathered from four productively operated data centers in Section “Evaluation”, which carries out the evaluation of our approach. We conclude in Section “Conclusion and Future Work” and provide an outlook to further research activities.

Related Work

Today’s data centers can be formed by hundreds or thousands of servers (Jennings and Stadler 2015). Therefore, capacity management is a challenging task for shared environments and can involve significant manual effort (Cherkasova and Rolia 2006). Consequently, many approaches have been developed to automate the steps of placing enterprise applications (hereafter referred to as services) on available physical servers in accordance to their demands. In addition, virtualization techniques enable relocations...
of services with little effort. Hence, automatic service placement has attracted increasing interest and is relevant in both practical consolidation projects (Speitkamp and Bichler 2010) and scientific literature. We classify the related work with respect to the considered workload of services to be placed, which can be either static or dynamic (Feller et al. 2011). If, besides the time dimension, multiple resource dimensions, e.g. CPU and memory consumption, are considered within the same optimization run, we refer to multidimensional workload consolidations. In the following, we discuss related work in order of publication year, before providing a classification of these in Table 1.

Rolia et al. (2005) presented an approach for capacity management in enterprise application clusters, which is based on groundwork presented in (Rolia et al. 2003). The considered daily workload patterns lead to a two-dimensional bin packing problem with one resource dimension. A genetic algorithm (GA) was developed to optimize the service allocation and was evaluated by performing a case study with 26 servers. The evaluation was conducted with respect to a particular reference CPU speed, but without considering heterogeneous CPU capacities. Other resource demands, such as main memory, were considered in the problem definition but were not included in the evaluation. Instead, sufficient main memory capacity of the servers was principally assumed.

Stillwell et al. (2010) defined a resource allocation problem with a static, but multidimensional workload. Several approximate solution approaches, such as greedy heuristics and a genetic algorithm, were developed. In the same year, Xu and Fortes (2010) developed a multi-objective grouping GA to minimize resource wastage, power consumption, and temperature hot spots simultaneously. Their approach considers CPU and memory consumption, but is applicable for static workloads only. Its evaluation was performed using simulated data. Speitkamp and Bichler (2010) proposed an LP-relaxation-based heuristic to solve two different optimization problems for server allocation with constant (SSAP) and variable workload (SSAPv) over time. The approach is based on preliminary work provided in (Bichler et al. 2006). Workload and capacity were both measured in SAPS without covering memory demands. In addition to that, only homogeneous data centers were considered.

An ant colony optimization (ACO) algorithm to minimize energy consumption of data centers by solving a multidimensional bin packing problem was presented by Feller et al. (2011). While multidimensional resources and time intervals were considered, homogeneous server capacities were assumed. The workload for each time interval was defined by the respective peak value and a simulation-based experimental evaluation was carried out. Gao et al. (2013) developed a Pareto-based ACO to address the multi-objective optimization of resource wastage and power consumption in virtualized data centers. It uses a multidimensional resource vector containing CPU and memory as well as a threshold for server utilization in order to avoid overload situations. However, only static workloads are considered and resources were normalized, which means the approach is restricted to homogeneous bin sizes.

In order to identify publications that are the most closely related to the introduced optimization problem (see Section “Introduction”) as well as applicable to address the optimization potential, we classify the publications with respect to their application areas. The application area is defined by the considered workload type (dynamic or static), restrictions for the bin composition (e.g. homogeneity), and the variety of considered resource dimensions. Table 1 presents these characteristics for each discussed publication.

<table>
<thead>
<tr>
<th>Publication</th>
<th>Workload</th>
<th>Actual Bin Composition</th>
<th>Actual Resource Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rolia2003</td>
<td>dynamic</td>
<td>Homogeneous CPUs</td>
<td>CPU</td>
</tr>
<tr>
<td>Rolia2005</td>
<td>dynamic</td>
<td>Homogeneous CPUs</td>
<td>CPU</td>
</tr>
<tr>
<td>Bichler2006</td>
<td>dynamic</td>
<td>Homogeneous Servers</td>
<td>CPU</td>
</tr>
<tr>
<td>Cherkasova2006</td>
<td>dynamic</td>
<td>Homogeneous CPUs</td>
<td>CPU</td>
</tr>
<tr>
<td>Stillwell2010</td>
<td>static</td>
<td>Homogeneous Servers</td>
<td>arbitrary</td>
</tr>
<tr>
<td>Xu2010</td>
<td>static</td>
<td>Heterogeneous Servers</td>
<td>CPU and Memory</td>
</tr>
<tr>
<td>Speitkamp2010</td>
<td>dynamic</td>
<td>Homogeneous Servers</td>
<td>CPU</td>
</tr>
<tr>
<td>Feller2011</td>
<td>dynamic</td>
<td>Homogeneous Servers</td>
<td>arbitrary</td>
</tr>
<tr>
<td>Gao2013</td>
<td>static</td>
<td>Homogeneous Servers</td>
<td>CPU and Memory</td>
</tr>
</tbody>
</table>

**Table 1. Classification of related work**

In some publications, additional bin compositions and resource dimensions were discussed but not demonstrated. Thus, we focus on characteristics actually supported by the implemented algorithms and
named the columns of Table 1 respectively. Publications whose workload was classified as static do not use varying resource demands as input for the optimization problem; hence, they are not suitable for business environments since existing workload patterns are not identified. Three of the four publications that consider multiple resource dimensions are suitable for static workloads only. In this manner, the complexity of implemented algorithms was reduced by eliminating the time dimension. The remaining publications consider dynamic workloads, but are limited to homogeneous servers or CPUs. However, in reality, large clusters are formed by many heterogeneous machines with different capacities, number of CPU cores, frequencies, and specific devices (Petrucci et al. 2011). Only one publication, (Xu and Fortes 2010), supports heterogeneous servers while considering multiple resource dimensions. On the contrary, no time dimension was used, leading to a static approach that comprises a total number of two dimensions. Furthermore, some of the mentioned solution approaches are not scalable to arbitrary problem sizes due to long computation times of up to several hours.

To conclude, the vast majority of the related work considers only two dimensions in total. These are either represented by the time dimension and one additional resource dimension or by two resource dimensions without any time consideration (static workload) leading to the risk of overloaded servers and resulting SLA violations. Only one publication, (Feller et al. 2011), describes algorithms that support multiple resource dimensions and dynamic workloads. However, only homogeneous servers are supported, hence, the approach is not applicable to solve the introduced optimization problem of differently sized servers. As a consequence, the identified related work is either not applicable to solve the introduced optimization problem or addresses the optimization potential with significant restrictions or risks. Therefore, if the allocation of services in heterogeneous data centers needs to be optimized based on dynamic workloads and multiple resource dimensions, the development of a new and efficient approach is necessary.

**Problem Formulation**

In order to define a formal optimization problem for workload consolidation, a goal function has to be defined. This function should consider the objective of reducing energy costs by shutting down servers that are not required at the time. For instance, CPU and memory resources can account for more than 50 percent of total server energy consumption (Barroso et al. 2013; Fan et al. 2007). However, considering the capacity minimization of all resource types in an optimization approach leads to challenges of a multi-objective problem formulation, such as the required a priori knowledge for objective weighing (Fonseca and Fleming 1995) or the overwhelming of a decision maker when using a Pareto approach (Kulturel-Konak et al. 2008). In order to avoid these challenges, only a single resource dimension is considered as an objective in the following problem formulation. Since CPU is the component with the highest energy consumption and the greatest dynamic power range, it can be used to estimate the energy costs of a server as done e.g. by Google (Fan et al. 2007). Thus, the optimization problem can be defined as follows. Herein, we refer to services as instances (I) and to servers as hosts (H) in order to distinguish their acronyms and increase readability:

A system consists of n bins or hosts $H = (h_1, ..., h_n)$ with $h_j = (c_1, ..., c_r)$ a capacity vector of length $r \geq 1$. On these hosts, m services or instances $I = (i_1, ..., i_m)$ have to be allocated. For each service, workloads are available for $\tau$ time intervals. Thus, each instance $i_k = (\vec{d}_{i_1}, ..., \vec{d}_{i_{\tau}})$ can be described using $r$ vectors of length $\tau$ which contain the resource demands at a time. An allocation $a: I \rightarrow H$ determines which instance is to be placed on which host. A binary decision variable for each host $b_j$ is true for an allocation if there exists an instance that is placed on the host $h_j$. Thus, the optimization problem is to identify an allocation that minimizes the required capacity of the first resource (i.e. CPU) while not exceeding any resource capacity at a time:

$$\min_a \sum_{j=1}^{n} b_j \cdot c^l_j$$

s.t. $\sum_{d_k^l(t)}(t) \leq c^l_j$ for all $1 \leq k \leq m$ with $a(i_k) = h_i$ and all $1 \leq l \leq r, 1 \leq i \leq n, 1 \leq t \leq \tau$

Therefore, this problem describes a $(r + 1)$-dimensional bin packing with dimensions for the modeled resource types as well as time.

Twenty-second Americas Conference on Information Systems, San Diego, 2016 4
Solution Algorithms

The introduced bin packing problem is known to be NP-hard. Since exact approaches are not efficient enough for real-world use cases, heuristics and metaheuristics are applied to bin packing problems (Jennings and Stadler 2015; Liu et al. 2008). Greedy heuristics and metaheuristics are very efficient and have been applied successfully to workload consolidation problems mentioned in related work. Thus, we adapt these approaches to solve dynamic, multidimensional consolidation problems. In the following, we briefly introduce possible solution algorithms including greedy heuristics, a genetic algorithm, and combinations thereof as well as a grouping genetic algorithm. In Section “Evaluation”, these algorithms will be compared with respect to their efficiency and solution qualities.

Greedy Heuristics

Greedy heuristics have been successfully applied to one-dimensional bin packing problems for years. Although their applicability for multidimensional bin packing has been questioned by (Speitkamp and Bichler 2010), (Stillwell et al. 2010) showed that they can also be successfully applied to such problems. Therefore, we implemented two well-known algorithms in this area, namely the first-fit decreasing (FFD) and the best-fit decreasing (BFD) algorithms. Both are based on sorting the list of services that need to be placed descending by their resource demands. BFD additionally sorts the list of hosts ascending by their remaining capacities. According to (Stillwell et al. 2010), there are two applicable ways of sorting the service list for multidimensional resource demands: based on the maximum demand in each dimension (max) or based on the sum of demands of each dimension (sum). The host sorting for the BFD algorithm is realized by considering the sum of remaining capacity of all dimensions. In this way, we implemented FFDmax, FFDsum, BFDmax, and BFDsum to cover all possible sorting strategies.

Genetic Algorithm

Genetic algorithms are flexible metaheuristics that are based on natural evolution. Comparable to DNA encoding, solution candidates are often encoded in binary or integer strings. In each so-called generation of the algorithm, a number \( \mu \) of solutions are considered in parallel and form the population. In order to generate \( \lambda (\lambda \gg \mu) \) new solution candidates to be considered for the next generation, the algorithm applies recombination and mutation. A selection procedure guides the search process to promising areas of the solution space by preferring solutions with higher quality. Therefore, it chooses \( \mu \) solutions for the next generation (\( \mu, \lambda \)-selection, cf. (Bäck 1996)). This quality is expressed by a fitness function. After conducting \( \text{maxGen} \) generations, the fittest individual of the current population is returned as the result.

The fitness function of the genetic algorithms maps the goal function to a value that has to be maximized. Thus, the negative required capacity of the first resource (cf. Section “Problem Formulation”) is the basis for this value. Since a genetic algorithm can result in allocations which are not feasible, repair mechanisms or penalty functions can be used. The former introduces a bias into the search process which can lead to worse results. Therefore, penalty functions are recommended (Coit and Smith 1996). For this penalty, the maximum difference of required and provided resources at a time (or zero if no overloading occurs) is squared and summed up for each resource type and host. In order to support exploration of the search space in early generations, the penalty increases depending on the generation number \( \text{gen} \):

\[
 f_{\text{gen}}(a) = - \sum_{j=1}^{n} b_j \cdot c_j^1 - \frac{\text{gen}}{	ext{maxGen}} \sum_{l=1}^{r} \max_{t=1}^{n} \left( 0, \max_{a(i_k) = h_j} \left( \sum_{a(i_k) = h_j} d_i^l(t) \right) - c_1^t \right)^2
\]  

(2)

A simple genetic algorithm for bin packing has been used e.g. by (Stillwell et al. 2010) and (Rolia et al. 2003). In this algorithm, allocations are encoded as an integer string of length \( m \) with values 1 to \( n \) determining to which host an instance is allocated. A solution candidate is then instantiated by randomly choosing a host for each instance. Recombination can be realized by a uniform crossover and mutation by swapping host numbers for two random instances. For the selection of \( \mu \) solutions for the next generation, tournaments are used. Therefore, \( t \) individuals are selected randomly from the population and the \( k \)-th fittest solutions of these are selected for the next generation based on a value \( p \) with probability \( p \cdot (1 - p)^{k-1} \). This genetic algorithm is labeled as GA in the following.
**Genetic Algorithm with Greedy Allocation**

Greedy algorithms can always identify an optimal allocation for at least one ordering of items for one-dimensional item sizes (Lewis 2009). Based on this theorem, we use another genetic algorithm to optimize the ordering sequence of services (cf. e.g. (Liu et al. 2008; Stillwell et al. 2010)). A solution candidate is encoded as an integer string containing a permutation of the list of services. The fitness of a solution can be determined by applying a greedy heuristic based on the permutated instance list to generate an allocation. An input sequence is mutated by swapping the position of two instances in the permutation. For the recombination of permutations, edge recombination can be used (cf. (Larrañaga et al. 1999)). In this context, the adjacency matrices of the graphs formed by the edges between instances next to each other in a permutation are recombined. Thus, offspring solutions should contain a high number of edges which are contained in the parent solutions. For selection, tournaments are used. Since two possible greedy heuristics have been implemented, two algorithms can be defined which are labeled GA_FF (with first-fit) and GA_BF (with best-fit).

**Grouping Genetic Algorithm**

In addition, we propose a grouping genetic algorithm (labeled GGA), since classical genetic algorithms are reported to perform poorly on grouping problems (Falkenauer and Delchambre 1992). For multidimensional bin packing, this approach has been used e.g. by (Xu and Fortes 2010). In a GGA, solution candidates are encoded as groups of instances for each host. In order to recombine two solutions, the following operations are performed. Injection: Performs a uniform crossover of groups between two solutions. Deletion: Deletes all duplicate services in each solution. Reinsertion: Adds all services that are not contained in a solution using a first-fit heuristic. For the mutations, services are moved from one group to another, thus, allocating the service to a different server. The mapping from hosts to instances is inverted to compute an allocation; the fitness of which can be computed according to Equation 2. As in the other genetic algorithms, tournament selection is performed.

**Evaluation**

We evaluate the applicability of the eight developed solution algorithms to the multidimensional workload consolidation problem using historical workload traces gathered from four productively operated data centers (hereafter referred to as Case 1 – Case 4). These comprise a total number of 364 services and 216 servers that were monitored for periods of 14-21 days by our industry partner. The services represent running ERP components, like SAP application instances and databases, which can be relocated to any server within the borders of one case. The traces include hourly consumption of CPU and main memory for each running service. Therefore, we use these resource dimensions for evaluation. CPU utilizations were measured in SAPS (SAP Application Performance Standard) whereby 100 SAPS represent 2,000 fully processed order line items per hour (SAP, 2015b). Additionally, we extracted the total capacities for each server in terms of their CPU capacity in SAPS and their available amount of memory in Megabytes. We provide and discuss evaluation results in the following subsections “Findings” and “Discussion”.

**Findings**

As described in Section “Related Work”, related dynamic consolidation problems are usually optimized using consumption traces of only one resource, normally of the CPU. In Section “Introduction”, we state that this approach leads to the risk of overloading other resources, whose usage may not depend on the CPU consumption. To verify this assumption, we analyzed the workload traces of all enterprise services that have been monitored as part of our studied cases with respect to any existing correlation between their CPU usage and their allocated main memory. However, the respective Pearson product-moment correlation coefficient accounted to -0.02127291, hence, no considerable linear correlation was found. In addition, we performed several optimization runs that exclusively consider the CPU demands of all running services (number of resources r=1) and analyzed overloading effects. Across all studied cases, 77.08 percent of available servers in the optimized solutions would show memory insufficiencies at least within one time interval when being deployed for the traced workload. In the worst case, only 1/4 - 1/10 of the requested main memory could actually be allocated. Therefore, we state that allocation solutions that were calculated based on CPU demands tend to eliminate servers that were actually needed in terms of
their main memory capacity. These overloads need to be avoided by applying solution algorithms that consider multiple resource dimensions simultaneously. Therefore, we implemented versions of all algorithms introduced in Section “Solution Algorithms” that are able to solve multidimensional workload consolidation problems. In the following, we analyze both the solution qualities and the computing times of the eight algorithms.

All deterministic algorithms, i.e. the greedy heuristics, needed to be performed only once. For the remaining algorithms, due to their stochastic nature, we performed 100 iterations of optimization runs for each case in order to provide the best and the mean values for the results. For all genetic algorithms, solutions are mutated with 10 percent probability, tournament size has been set to 4, and the best solution is selected with 90 percent probability. In each generation, 100 solutions are processed and 200 offspring are generated by recombination. When comparing original server capacities with the total amount of required capacities after the workload has been consolidated, the actual addressed optimization potential gets unveiled. Therefore, Table 2 shows for each case the percentage of original capacity, based on required SAPS, that was identified by the algorithms to be eliminated in its best iteration.

<table>
<thead>
<tr>
<th>Case</th>
<th>FFDmax</th>
<th>FFDsum</th>
<th>BFDmax</th>
<th>BFDsum</th>
<th>GA</th>
<th>GA_FF</th>
<th>GA_BF</th>
<th>GGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25.54%</td>
<td>25.54%</td>
<td>36.86%</td>
<td>35.05%</td>
<td>27.14%</td>
<td>28.27%</td>
<td>42.75%</td>
<td>35.64%</td>
</tr>
<tr>
<td>2</td>
<td>32.62%</td>
<td>32.62%</td>
<td>47.09%</td>
<td>47.09%</td>
<td>39.30%</td>
<td>32.62%</td>
<td>53.39%</td>
<td>44.12%</td>
</tr>
<tr>
<td>3</td>
<td>31.97%</td>
<td>31.97%</td>
<td>34.02%</td>
<td>29.79%</td>
<td>0.00%</td>
<td>34.36%</td>
<td>35.78%</td>
<td>34.56%</td>
</tr>
<tr>
<td>4</td>
<td>46.82%</td>
<td>46.82%</td>
<td>48.75%</td>
<td>47.81%</td>
<td>35.74%</td>
<td>50.10%</td>
<td>51.66%</td>
<td>44.82%</td>
</tr>
</tbody>
</table>

Table 2. Comparison of solution qualities for each algorithm and case

The mean savings across all cases and algorithms amount to 37.21 percent. The greatest optimization was achieved in Case 2, where up to 53.39 percent of originally needed SAPS can be saved by turning off 9 out of 15 servers that are no longer required. The best results were provided by the GA_BF in all cases. In two cases, the pure GA performed the worst. In case 3, the GA did not actually achieve any optimization in terms of required SAPS, because calculated solutions would have caused resource overloads and were in turn heavily penalized. In two other cases, FFD performed worst. Finally, FFDmax and FFDsum yielded equal results in all cases, whereas BFDmax always achieved better or equal results than BFDsum. The diagram in Figure 1 shows the confidence intervals for the mean relative difference of the fittest, i.e. best values after 100 iterations to the fittest value over all algorithms across the four cases.

![Figure 1. Mean relative differences (confidence intervals) to best objective value](image)

As can be seen in Figure 1, GA_BF always finds the best allocation solution. GGA performs significantly better than GA, but worse than GA_BF. From our observations, GA_BF outperforms GA_FF and is significantly better than heuristics, but also requires significantly more computing time by up to factor 1000. However, for the studied cases, the longest mean computing time for a single iteration accounted to 6.02 seconds (GA) and can therefore be neglected in the targeted domain. Nevertheless, we provide computing times in Table 3 for examining the scalability of the algorithms, which might be relevant in different domains or in the instance of frequent online consolidations.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>FFDmax</th>
<th>FFDsum</th>
<th>BFDmax</th>
<th>BFDsum</th>
<th>GA</th>
<th>GA_FF</th>
<th>GA_BF</th>
<th>GGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCT</td>
<td>0.0284 s</td>
<td>0.0126 s</td>
<td>0.0192 s</td>
<td>0.0152 s</td>
<td>4.016 s</td>
<td>2.272 s</td>
<td>2.826 s</td>
<td>2.83 s</td>
</tr>
</tbody>
</table>

Table 3. Mean computing times (MCT) across all cases
Having analyzed the best results for each algorithm, mean values and standard deviation of the 100 results reveal insights concerning the stability of each genetic algorithm. In this respect, GA_FF and GA_BF are very stable and produce good results after a few iterations. Hence, their mean values are better than the ones of FFD and BFD. On the contrary, GGA has a high fluctuation in result quality. As a consequence, mean values are low in comparison to heuristics due to infeasible solutions in some iterations. Lastly, GA has the highest variance in result quality. Here, a high number of solutions are infeasible due to penalized overload situations, leading to an increased number of necessary iterations and expenditure of time in order to gain satisfying results.

**Discussion**

Our initial experiments showed that optimization runs which address only a single resource dimension tend to eliminate resources that are needed to ensure quality of service. The results clearly indicate dramatic overload situations that must not be neglected. In contrast, the multidimensional workload consolidation problem has been shown to consider multiple resource demands of given services at the same time in order to avoid such overloads. For each allocation of services to a server, the developed algorithms ensured sufficient resource capacities that satisfy the sum of all service demands at any time interval. Hence, multidimensional approaches address less optimization potential but produce more realistic solutions that are applicable to ensure performance requirements of enterprise applications.

Heuristics provide good results in a very short time. However, additional optimization potential can be leveraged if more time-consuming approaches are applied. In particular, GA_BF resulted in optimal results in all cases. However, due to the small number of observed cases and the high variance of GA_FF, further cases would need to be analyzed to confirm a general recommendation of GA_BF for multidimensional workload consolidation problems. Heuristics produce comparably good results, where variations in sorting approaches (max and sum) do not draw significant distinctions. In general, best-fit approaches produce better solution qualities with less variance than first-fit approaches. Besides GA_BF, also GGA can be more effective than heuristics. Since GA_BF has low variance, only a few optimization runs are required to achieve better results than from any heuristic.

To conclude, GA_BF is the preferred algorithm to address the multidimensional workload consolidation problem. Nevertheless, additional experiments using both heuristics and genetic algorithms are required in order to derive a general preference.

**Conclusion and Future Work**

In the domain of enterprise applications, average utilization levels of servers vary between 10 and 50 percent. By consolidating orthogonal workloads of running services, the total required capacity can be reduced significantly, thus, saving operational costs. Since performance must not be degraded significantly, resource usage patterns of historical workload traces need to be considered in order to determine the minimum amount of required capacities. Therefore, existing optimization potential can be addressed by solving a combinatorial problem that is closely related to the well-known bin packing problem which is known to be NP-hard. Related problem definitions preponderantly focus on a single resource dimension, usually represented by CPU demands and capacities. This leads to a significant risk of overloading resources that do not depend on CPU usages. Hence, our analysis of one-dimensional workload consolidation strategies has unveiled alarming resource overloading in terms of memory insufficiencies. As a consequence, we state that multiple resource dimensions must be considered to prevent SLA violations. We formulated the multidimensional workload consolidation problem and adapted four heuristics and four genetic algorithms to provide solutions for such problems. The applicability of the algorithms was evaluated using workload traces of four different data centers that include CPU and memory usages for 364 running services. According to our optimization runs, a best-fit heuristic that uses a genetic algorithm to arrange the input sequence of running services (GA_BF) provides the best solution qualities with the lowest variance in terms of reduced hardware capacities. Therefore, good results are available after a small number of iterations. In the best case, GA_BF revealed that up to 53.39 percent of the originally needed capacity can be saved by turning off servers that are no longer required. Thus, operational costs can be reduced significantly by increasing mean utilization levels. To summarize evaluation results, more server capacity is needed if further resources, such as memory usage, are considered in the workload consolidation problem, but the probability of overloading servers is
significantly reduced. Therefore, our results serve as a contribution to the theoretical field of capacity management by turning related research into more realistic and practically applicable methods. At the same time, our results serve as an appeal to all practitioners of consolidation projects who perform service relocations based on single resource demands. The presented problem is currently limited to offline workload consolidation strategies. However, the short mean computing times of the heuristics would allow the approach to be utilized for online relocations either.

In the future, we plan to assess our results in a broader context by performing a field study across a variety of workload traces gathered from additional cases of different sizes. In this way, the resulting variance of solution qualities will become more meaningful and will indicate the stability of each algorithm solidly. To support practical applications, a web service is planned that can be consumed by field consultants during consolidation projects. In this regard, the support of placement constraints will be added to fulfill requirements resulting from existing infrastructure and operational dependencies.

REFERENCES


