Summer 6-27-2016

OPTIMIZING SERVER CONSOLIDATION FOR ENTERPRISE APPLICATION SERVICE PROVIDERS

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

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

Klaus Turowski
Otto-von-Guericke-University, klaus.turowski@ovgu.de

OPTIMIZING SERVER CONSOLIDATION FOR ENTERPRISE APPLICATION SERVICE PROVIDERS

Hendrik Müller, Faculty of Computer Science, Otto-von-Guericke-University, Magdeburg, Germany, hendrik.mueller@ovgu.de

Sascha Bosse, Faculty of Computer Science, Otto-von-Guericke-University, Magdeburg, Germany, sascha.bosse@ovgu.de

Klaus Turowski, Faculty of Computer Science, Otto-von-Guericke-University, Magdeburg, Germany, klaus.turowski@ovgu.de

Abstract

In enterprise application environments, hardware resources show averagely low utilization rates due to a provisioning practice that is based on peak demands. Therefore, the consolidation of orthogonal workloads can improve energy efficiency and reduce total cost of ownership. In this paper, we address existing workload consolidation potential by solving a bin packing problem, where the number of servers is to be minimized. Since dynamic workloads, gathered from historical traces, and priorities of running services are considered, we formulate the Dynamic Priority-based Workload Consolidation Problem (DPWCP) and develop solution algorithms using heuristics and metaheuristics. Relevance is pointed out by an analysis of service resource demands and server capacities across four studied cases from productively operating enterprise application service providers. After a classification of related work, seven algorithms were developed and evaluated regarding their exploited optimization potential and computing time. Best results were achieved by a best-fit approach that uses a genetic algorithm to optimize its input sequence (GA_BF). When applying the GA_BF onto the four studied cases, average utilization rates could be increased from 23 to 63 percent within an average computing time of 22.5 seconds. Therefore, the overall server capacity was reduced significantly by up to 83%.

Keywords: Dynamic workload consolidation, Enterprise applications, Heuristic, Service priority.

1 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). Therefore, achieving energy efficiency in IT operations is a crucial task for enterprises in order to save running costs. In some cases, 40-50% of the total data center operational budget is spent on energy costs for IT components (Filani et al., 2008). 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% from 2005 to 2010 (Koomey, 2011; Splieth et al., 2015). This is mainly caused by low average utilization levels of hardware resources in enterprise application environments (Beloglazov and Buyya, 2010; Mi et al., 2010), since existing consolidation potential is not exploited for the continuously growing number of servers. This growth, in turn, is caused by low utilization rates due to a provisioning practice that is based on peak demands (Rolia et al., 2003). The average utilization of data center servers was estimated by the Gartner Group in 2005 to be less than 20% (Speitkamp and Bichler, 2010). According to a newer Gartner report from 2011, servers even run at average utilization levels less than 15% (Gartner, 2011). An analysis of data collected from more than 5,000 production servers by (Beloglazov and Buyya, 2010) showed a capacity usage of usually 10-50% for a six-month measurement period (Beloglazov and Buyya, 2010). A case study that we performed preparatory (see Section 2) to the work presented in this paper confirmed these observations and showed an average CPU utilization of 33% across four enterprise data centers, comprising 206 physical servers and 311 business applications. Such low utilization rates adversely affect energy consumption, since even idle servers, depending on their type and architecture, can consume up to 70% of their peak power (Beloglazov and Buyya, 2010).

Thus, shutting down idle servers is beneficial for the total energy efficiency, aiming at load concentration instead of load balancing (Petrucci et al., 2011). In addition to that, cooling infrastructure is more effective with fewer servers, because effects of overheating can be avoided (Beloglazov and Buyya, 2010). Thus, the consolidation of workloads can achieve higher energy savings (Xu and Fortes, 2010) by eliminating unused hardware resources, enabling IT service providers to produce services more efficiently. However, the performance of IT systems must not be degraded significantly (Beloglazov and Buyya, 2010) in order to support business processes effectively. Therefore, IT Service Management (ITSM) frameworks such as ITIL and ISO 20000 embed the task of balancing performance and operational costs into the capacity management process.

Load concentration is more effectively if consolidated workloads are orthogonal. In this case, the mean utilization can be maximized. The consolidation of orthogonal workloads on a single server is enabled by the concept of virtualization techniques (Ferreto et al., 2011) which can be distinguished regarding their level of operational abstraction. Common virtualization techniques in the domain of enterprise applications are server virtualization and application virtualization.

Server virtualization decouples the operating system and all application processes from physical resources by creating virtual machines (VM). The virtualization layer, built up by a hypervisor, controls the allocation of physical resources to VMs and can initiate migrations to different physical servers in a running state of the VM. On the other hand, the necessity of a hypervisor and operating systems for each set of application causes an overhead that leads to performance degradations (Sahoo et al., 2010). Since low response times are still the major requirement of both ERP users and administrators (IT-Onlinemagazin, 2015), application virtualization is another common technique in the domain of enterprise applications. It decouples applications from their operating system and underlying hardware without the need for hypervisors and additional operating systems, thus, leading to less performance degradation. In fact, application landscapes that are distributed across multiple physical servers, can share the same operating system through the use of centralized storage, which ensures flexibility and decreases maintenance costs. This concept is particularly used for standard business software such as SAP ERP systems which are based on the same technology platform and have similar or equal requirements on operating system level. Therefore, one implementation of application virtualization was introduced by SAP as adaptive computing (AC) (SAP, 2015a). While this concept enables applications to be started on any physical server, on the downside, relocations require downtime during the migration phase and need to be well-planned due to ERP’s exceptional significance for business continuity (Müller and Turowski, 2015). Therefore, in the domain of enterprise applications, dynamic online migrations are no option and applications need to be allocated to hardware resources based on their expected peak demands in order to guarantee performance operational level agreements for a certain time period (Rolia et al., 2003).

However, a static consolidation in terms of a single workload value that represents the applications peak demand leads to bad resource utilization (Mi et al., 2010), because workloads can change significantly over time (Petrucci et al., 2011). For enterprise applications, the workload is often repetitive and, thus, predictable in seasonal patterns (Rolia et al., 2005; Bichler et al., 2006) such as daily or weekly recurrences (Setzer and Stage, 2010). An analysis of the data collected as part of our case study confirms this statement, since the average CPU utilization across all applications changes in accordance to typical working hours and days. Thus, orthogonality of workloads can be identified from historical data. Changes in identified patterns happen slowly, allowing allocations to be adapted over time (Cherkasova and Rolia, 2006). Larger data centers may save 30-35% of servers by considering complementary workloads as part of their capacity management (Setzer and Stage, 2010). In addition to that, different types of applications need to be considered, since dedicated development and quality assurance applications’ performance is not as crucial as those of productive ERP applications. Thus, the optimization of server consolidation while guaranteeing sufficient application performance can improve energy efficiency of enterprise data centers.

A server's CPU is the main bottleneck for enterprise applications (Mi et al., 2010; Speitkamp and Bichler, 2010). At the same time, CPU is known to be the greatest energy consumer in servers (Barroso et al., 2013; Splieth et al., 2015). Therefore, it can even be used to estimate server energy consumption as being done in data centers operated by Google (Fan et al., 2007; Splieth et al., 2015). Consequently, the CPU’s utilization can be used in order to trade-off performance and energy consumption. CPU utilization as well as capacity can be measured in SAPS (Speitkamp and Bichler, 2010) and gathered from historical workload traces in order to identify seasonal patterns. Based on this information, enterprise applications can be placed on available physical servers aiming at a maximized average utilization and a minimized amount of required capacity while ensuring operational level agreements depending on the type of application. This is a combinatorial problem and closely related to the well-known bin packing problem which is NP-hard. Figure 1 visualizes the application virtualization concept and an exemplary solution of the optimization problem at application layer. The size of a server represents its capacity and the individual workloads of ERP systems and databases are represented by their shapes leading to certain applications that are more compatible to each other than different ones. In the visualized example, the applications on server 4 could be moved to servers 2 and 3. Consequently, the average utilization increases and the number of running servers (bins) can be reduced.

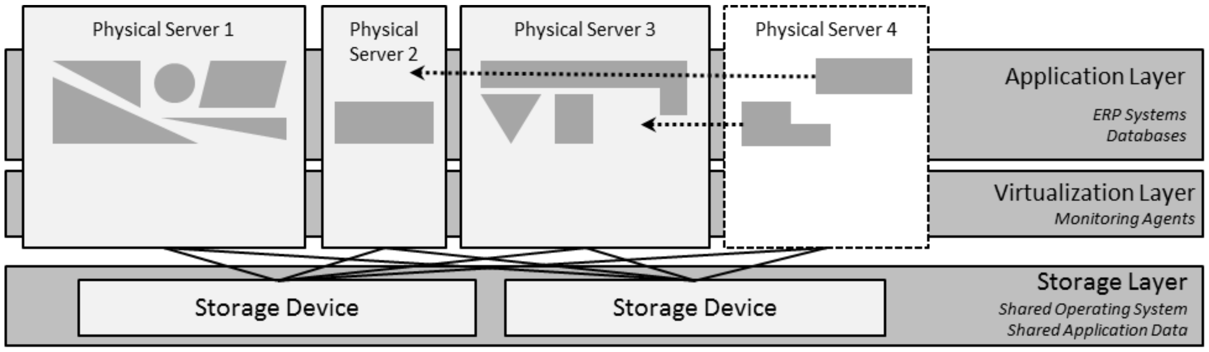


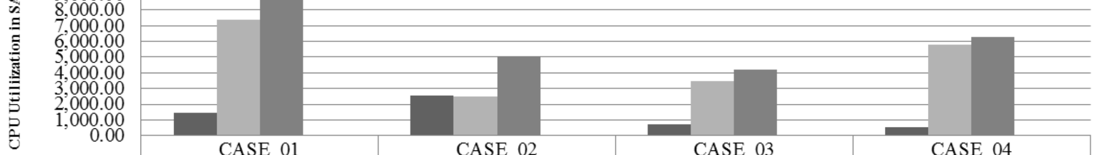
Figure 1. Workload consolidation on application virtualization infrastructure

Many approaches for solving such bin packing problems exist, but those are either not scalable even to moderate problem sizes (Mi et al., 2010) or not applicable for enterprise application consolidations.

In this paper, we describe an approach for the efficient optimization of resource allocation for enterprise applications that complies with existing service level agreements by taking service priorities into account. The approach was developed according to the design science research paradigm (Hevner et al., 2004; Peffers et al., 2007) and is based on foundations that we formulated in Mueller et al. (2016). The relevance of the research is pointed out in Section 2 by analyzing load and capacity data of servers running at four data centers as part of preparatory case studies. In Section 3, existing approaches that may be applicable to solve the problem are discussed. A mathematical formulation of the optimization problem is given in Section 4. In section 5, a number of solution algorithms is proposed including heuristics such as first-fit-decreasing (FFD), meta-heuristics such as a genetic algorithm (GA) and hybrid approaches such as a genetic algorithm using first-fit (GA_FF). Based on the workload traces of the three data centers, the evaluation of the developed approach is carried out in Section 6 by comparing the solution algorithms in terms of possible savings and execution times. Section 7 summarizes the findings of the paper and provides an outlook to further research activities.

2 OPTIMIZATION POTENTIAL ACROSS FOUR STUDIED CASES

According to the introduced literature (see Section 1), average utilization levels of servers in enterprise data centers vary between 10 and 50%. In addition to that, we studied 206 servers of four cases, each representing a productive data center that was monitored for periods of 14-21 days by our industry partner. In all cases, the servers host exclusively SAP software within an environment of virtualized applications as shown in Figure 1. These kinds of ERP systems follow a three-tier-architecture of presentation, application and database layers. The presentation layer is located on client-side, so the logical entities that are to be placed on available servers include SAP application instances and databases (In the following, we refer to both of them as *services*). The application layer of an ERP system, in many cases, is distributed across multiple SAP application instances that can be placed on individual physical servers. In such standardized environments, the total CPU capacity of each physical server and the CPU demands for each service can be measured in SAPS. SAPS stands for "SAP Application Performance Standard" whereby 100 SAPS represent either 2,400 SAP transactions or 2,000 fully processed order line items per hour (SAP, 2015b). Therefore, a service that serves 150,000 SAP transactions per hour must not to be placed on a server with a capacity lower than 6,250 SAPS. As preparatory work, we imported the monitored data from our industry partner into a relational database schema so that we were able to group the consumed SAPS per service and time interval and to query for servers and their respective SAPS capacity. Appropriate time intervals can be hours and days. In all cases, the existing servers are heterogeneous with varying CPU types, amounts and frequencies. In order to evaluate the scalability of our approach, we selected cases of different sizes, thus, the amount of services varies between 15 and 103 and the amount of servers varies between 4 and 88. We define the average optimization potential as the average amount of unused capacity for each server across all observed time intervals in terms of idle SAPS.



	CASE_01	CASE_02	CASE_03	CASE_04
■ AVG_USED_SAPS	1,460.32	2,573.45	689.24	519.78
■ AVG_IDLE_SAPS	7,386	2,472.56	3,481.77	5,779.06
■ AVG_MAX_SAPS	8,846.32	5,046	4,171.01	6,298.84
AVG_UTILIZATION	16.51%	51%	16.52%	8.25%

Figure 2. Capacity and utilization overview of the four studied cases

Figure 2 shows this value as well as the average used SAPS across all services, the average maximum capacities across all servers and the average server utilization for each case. The optimization potential, represented by AVG_IDLE_SAPS in Figure 2, was calculated using average demands across all services. For placement decisions, average demands cannot be considered due to ERP's exceptional significance for business continuity (Müller and Turowski, 2015). Instead, peak or near-peak demands for each time interval need to be considered (Rolia et al., 2003; Speitkamp and Bichler, 2010). Therefore, the de facto optimization potential depends on the definition of peak demands which can be represented by a particular quantile of all observed demand values (Speitkamp and Bichler, 2010). Since different types of systems have different performance requirements resulting from their operational level agreements (OLA), the quantile definition must depend on the system type. For development systems, e.g., the 0.95-quantile of all values for a particular hour might be an acceptable sizing bases instead of using their peak demands, since overload situations will result in higher response times (Speitkamp and Bichler, 2010), which might be acceptable for development systems but is not desired for productive systems. As preparatory work, we classified the services of all studied cases regarding their system types, which include production (PROD), development (DEV), quality assurance (QA), database (DB) and other (OTHER). We will refer to these cases again, when using them for the evaluation of our optimization problem and implemented algorithms in Section 6.

3 RELATED WORK

Capacity management is a challenging task for shared environments that can involve significant manual effort (Cherkasova and Rolia, 2006), particularly as today's data centers can be formed by hundreds or thousands of servers (Jennings and Stadler, 2015). Therefore, approaches have been developed to automate the steps of placing services on available physical servers in accordance to their demands. Virtualization techniques such as server virtualization and application virtualization enable services to be decoupled from physical resources. These techniques are increasingly used in business environments (Vogels, 2008). Thus, relocations of enterprise applications have become possible with little effort and allocation adaptions can be made more frequently. As a consequence, automatic service placement has attracted increasing interest and is relevant in both practical consolidation projects (Speitkamp and Bichler, 2010) and scientific literature.

The optimization potential, identified in Section 2, can be addressed by solving a bin packing problem where the overall size of used bins is to be minimized. Since these problems are proved to be NP-hard, heuristics and metaheuristics are applied to solve these types of problems (Bichler et al., 2006; Poli et al., 2007; Liu et al., 2008; Xu and Fortes, 2010; Jennings and Stadler, 2015). Although an optimal solution cannot be guaranteed, those are much more efficient than mathematical programming approaches (Ferreto et al., 2011). Heuristics can be very effective for a certain type of problem (Poli et al., 2007) but tend to waste resources in general (Feller et al., 2011) and depend strongly on the input sequence (Liu et al., 2008), e.g. first-fit-decreasing (FFD) or best-fit-decreasing (BFD). FFD is proved to be never suboptimal by more than 22% for a one-dimensional bin packing (Bichler et al., 2006). In contrast, metaheuristics provide a more flexible way to deal with varying problems (Mi et al., 2010). The problem's item characteristics depend on the type of considered workload for the service that is to be placed. Therefore, we distinguish the related work regarding the considered workload, which can be either static or dynamic (Feller et al., 2011). For a given service, a static workload refers to the maximum or a particular quantile of demands gathered from historical traces, thus, the optimization problem has no time dimension. Dynamic workloads include all demands for multiple time intervals, e.g. the peak SAPS demand for each hour or week of that service, leading to at least two-dimensional item sizes. Therefore, an optimization algorithm uses multiple varying values and is able to consider typical workload patterns of different services; thus, orthogonality in time can be identified. In the following, we discuss related work in order of publication year, before providing a classification of these (see Table 1).

Rolia et al. (2003) developed a genetic algorithm (GA) and a linear integer programming model for optimal resource management in data centers. In summary, they concluded that the GA performed better and is more robust to definition extensions. In order to avoid overloading, servers and thresholds for unused capacity are considered. While minimizing the number of servers used, a service is defined by all processes running on a server at the time of workload monitoring. In real data centers, this number of processes (*n*) can represent 1 to *n* different applications. Therefore, a set of applications, running on one monitored server, will always run together on the newly allocated server after solving the optimization problem, which leads to the risk of wasted optimization potential. In addition to that, application types cannot be distinguished regarding their particular performance requirements. For evaluation, 41 servers needed to be grouped in categories of similar hardware, including the restriction of homogeneous CPUs, which is not realistic for today's heterogeneous data center environments (Heath et al., 2005).

Rolia et al. (2005) presented an approach for capacity management in enterprise application clusters. The considered daily patterns lead to a two-dimensional bin packing problem. Productive and non-productive services are distinguished and potential placement constraints for certain applications are considered. A genetic algorithm was developed to optimize the current allocation and it was evaluated using a case study from 26 servers. The evaluation was not conducted using the heterogeneous CPUs of the enterprise application environment, but with respect to a particular reference CPU speed.

Bichler et al. (2006) defined two different problems for server allocation with constant (SSAP) and variable workload (SSAPv). As workload, they consider CPU utilizations for 5-minute averages during a month and aggregated these to hourly demands for a typical day. The approach is applicable only for homogeneous data centers with servers having identical capacities and cost. The used algorithm requires a solving time of 5 minutes to several hours depending on the problem size. For evaluation, data of 30 servers from a data center provider was used. In the same year, Cherkasova and Rolia (2006) presented a shared resource pool management with respect to quality of services, thus, different priorities of demands were considered. The performance requirements must be defined manually in terms of limits for degraded and acceptable performance. Therefore, these requirements do not depend on the system type gathered from historical traces. A genetic algorithm was developed to optimize existing allocations and evaluation was performed using data from 26 applications. To apply the approach, the server's CPUs need to be homogeneous but their number can vary for each server.

In 2010, three publications cover the resource allocation problem and its optimization. Stillwell et al. (2010) defined a resource allocation problem with static, but multi-dimensional workload as a mixed-integer linear program. The approach enables to identify optimal allocations in exponential time. While distinguishing between fixed and proportional resource demands, the optimization goal is to provide each request with a maximum on needed resources. Greedy heuristics and a genetic algorithm were developed and the evaluation was performed based on generated data. Xu and Fortes (2010) developed a multi-objective grouping genetic algorithm to minimize resource wastage, power consumption and temperature hot spots at the same time. The approach is applicable for static workloads only and the operations are highly problem-specific. Evaluation was performed using simulated data.

Speitkamp and Bichler (2010) developed an LP-relaxion-based heuristic to solve two types of bin packing problems (SSAP and SSAPv). In a first step, an LP-relaxed solution is found and fractional assignments are combined greedily as a second step. The approach was evaluated using real workload data gathered from traces of 259 ERP servers and 160 web servers that were monitored for three months in 5-minute intervals. Within the workload, daily patterns were found and for each hour, the 0.95-quantile was used as the sizing bases. Thus, peak demands have been excluded because overload situations were identified to be unlikely and will only result in increasing response times. Both the workload and capacity were measured in SAPS, representing the servers CPU utilization and capacity. The computation time seems to be worse than exponential and depends heavily on the problem size and the number of placement constraints. Therefore, the approach is limited to a maximum of 700 servers and, thus, not applicable for large data centers that can be formed often by hundreds of thousands of servers (Jennings and Stadler, 2015). In addition to that, only homogeneous data centers were considered.

Feller et al. (2011) developed an ant colony optimization (ACO) algorithm to minimize energy consumption of data centers by solving a multi-dimensional bin packing problem. Different resources and time intervals but homogeneous bins were considered. The workload for each time interval was defined by the respective peak value and a simulation-based experimental evaluation was performed.

Another Pareto-based ACO was developed by Gao et al. (2013) to solve a multi-objective optimization of resource wastage and power consumption. It uses a multi-dimensional resource vector and a threshold for server utilization. VM placement decisions are made based on the two dimensions CPU and memory. On the other hand, only static workloads are considered and resources were normalized, thus, the approach is restricted to homogeneous bins. The approach was evaluated in comparison with other meta-heuristics in terms of spacing and the number of solutions in the resulting Pareto front.

In order to identify publications that are most closely related to the introduced optimization problem (see Section 1) as well as applicable to address the optimization potential (see Section 2), we classify the publications regarding their application areas. The application area is formed by the considered workload type, restrictions for the bin composition and potential operational level agreement (OLA) distinctions. Table 1 lists these characteristics for each publication.

Publication	**Workload**	**Bin Composition**	**Service Distinction**
Rolia2003	dynamic	Similar server groups with homogeneous CPUs	no
Rolia2005	dynamic	Homogeneous CPUs	yes
Bichler2006	dynamic	Homogeneous Servers	no
Cherkasova2006	dynamic	Homogeneous CPUs	yes
Stillwell2010	static	Homogeneous Servers	no
Xu2010	static	Heterogeneous servers	no
Speitkamp2010	dynamic	Homogeneous Servers	no
Feller2011	dynamic	Homogeneous Servers	no
Gao2013	static	Homogeneous Servers	no

Table 1. Classification of related work

Those publications that do not use varying workloads as input for the optimization problem are not suitable for business environments since existing workload patterns are not identified. Thus, only one (static) time interval is considered by taking its peak demand, which leads to the risk of wasted optimization potential. 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). Finally, some of the mentioned approaches are not scalable to arbitrary problem sizes due to long computation times up to several hours. As a conclusion, the identified related work is either not applicable to solve the introduced optimization problem or addresses the optimization potential with significant restrictions. Therefore, if the allocation of services in heterogeneous data centers need to be optimized based on dynamic workloads, the development of a new and efficient approach is necessary. Furthermore, the approach requires a mechanism that distinguishes between different service types, since, e.g., productive services usually have higher performance requirements than development services.

4 OPTIMIZATION PROBLEM FORMULATION

As stated above, an allocation is sought in which the overall server operational costs are to be minimized while performance of the services has to be guaranteed. The workload of the services to be placed is dynamic. Therefore, the optimization problem described here is closely related to the Static

Server Allocation Problem with variable workload (SSAPv) from (Speitkamp and Bichler, 2010) whereby static does not refer to the type of workload but to the consolidation mode (sometimes called offline-approach).

In contrast to their work, the optimization problem considers different types of services that have different performance priorities. Enterprise applications that are used as productive business process support must not suffer from severe performance degradation. Other services, for instance, development or quality assurance services have a lower performance priority. In this work, the different performance priorities of services are encoded into the process of mapping several data points into a single demand value for a fixed sequence of time intervals $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_K)$. In the following problem formulation, we refer to services as instances (*I*) and to servers as hosts (*H*) in order to distinguish their acronyms and increase readability.

A **Dynamic Priority-Based Workload Consolidation Problem (DPWCP)** is a tuple (H, I, q) with $H = (c_1, \ldots, c_n)$ a list of host capacities, $I = (I_1, \ldots, I_m)$ a list of instances and $q: \mathbb{N} \rightarrow [0,1]$ a function.

An instance $I_j = \left(prio, (\boldsymbol{d_1}, \ldots, \boldsymbol{d_K})\right)$ is formed by its priority $prio \in \mathbb{N}$ and a vector of data points $\boldsymbol{d_k} = (d_{k_1}, \ldots, d_{k_l})$ for each time interval. Using the function q, a single, priority-dependent demand value d_{jk} can be computed for each time interval as the $q(prio)$-quantile of the data points $\boldsymbol{d_k}$.

An allocation of instances to hosts is a mapping $a: I \rightarrow \{1, \ldots, n\}$. A binary variable b_i encodes whether a host is used or not in an allocation:

$$b_i = 0 \Leftrightarrow \neg \exists I_j : a(I_j) = i$$

Since the allocated capacity is to be minimized, the optimization problem is

$$\min \sum_{i=1}^{n} b_i \cdot c_i$$

$$\text{s.t.} \sum d_{jk} \leq c_i \text{ for all } 1 \leq j \leq m \text{ with } a(I_j) = i \text{ and all } 1 \leq k \leq K.$$

Thus, the DPWCP can be seen as a two-dimensional bin packing problem (demand and time) with variable bin sizes and unique items.

In order to consider existing placement requirements or restrictions resulting from security, business or high availability considerations, our developed solution approach allows service placement constraints to be defined prior to the optimization run. Therefore, pre-allocated services and their needed resources will be excluded from the optimization by decreasing the respective server's capacities as also done in other approaches (Rolia et al., 2005; Bichler et al., 2006). In that manner, e.g., productive services can be placed on dedicated servers. This leads to a reduced number of needed spare servers, since HA-capabilities are usually only needed for servers that serve productive services.

For the configuration of $prio$, we distinguish five service types, namely production (PROD), development (DEV), quality assurance (QA), database (DB) and any other type (OTHER). The decision of a particular quantile for each service type needs to be made by system architects, since it influences the probability of overload situations. In the case that service placement constraints allow productive services to be placed together with any other service type, processes can be prioritized on operating system level (Ohlin and Kjær, 2007) to avoid performance impairments due to concurrently occurring peak demands. However, the probability of such overload situations to occur due to low quantiles was identified to be very low. An analysis performed by (Speitkamp and Bichler, 2010) resulted in 0.025 percent capacity violations when choosing the 0.8-quantile and 0.12 percent when using the 0.4-quantile. For evaluation (see Section 6), we conservatively configure no quantile to be less than 0.9, leading to reduced optimization potential in favour of a negligible risk of capacity violations.

5 SOLUTION ALGORITHMS

In order to solve the DPWCP, exact and approximate approaches can be applied. However, since the original bin packing problem is NP-hard (Bichler et al., 2006), exact approaches are not efficient enough for real-world use cases. Therefore, heuristics and metaheuristics are applied to bin packing problems (Poli et al., 2007; Liu et al., 2008; Jennings and Stadler, 2015).

For solving the SSAPv, (Speitkamp and Bichler, 2010) propose an LP-relaxion-based heuristic. The scalability of this approach is, however, questionable (cf. Section 3). Greedy heuristics or metaheuristics are very efficient and have been applied successfully to the SSAP and similar problems. Thus, these approaches could be adapted for efficiently solving the DPWCP. In the following, three different types of solution algorithms are discussed: greedy heuristics, a genetic algorithm and a hybrid algorithm. This results in seven possible solution algorithms.

5.1 Greedy Heuristics

Greedy heuristics have been successfully applied to bin packing problems for years. Two of the most popular algorithms in this area are the first-fit decreasing (FFD) and best-fit decreasing (BFD) algorithms. Both are proved to achieve results with less than 22 % more bins than an optimal solution for a one-dimensional bin packing (Bichler et al., 2006). However, their applicability for multi-dimensional bin packing has been questioned by (Speitkamp and Bichler, 2010). Nevertheless, (Stillwell et al., 2010) showed that greedy heuristics are also successfully applicable to multi-dimensional bin packing.

Therefore, both algorithms have been implemented for the DPWCP. In both algorithms, the list of services is sorted descending by demand. According to (Stillwell et al., 2010), there are two applicable ways of sorting the service list: based on the maximum demand in time or based on the sum of demands per time interval. This classification leads to the four greedy heuristics used in the following, labeled as **FFDmax**, **FFDsum**, **BFDmax** and **BFDsum**.

In the first-fit algorithm, the services are processed in the sorted order and are allocated to the first server which has enough capacity. The best-fit algorithm, however, allocates a service to the server with lowest remaining capacity that has enough capacity for the service to be allocated.

In order to implement the best-fit algorithm, the list of servers is sorted after each allocation of a service. Therefore, the remaining capacity at the time point of maximum loading is considered. When the list of servers is sorted ascending by remaining capacity, a first-fit allocation can be applied.

5.2 Genetic Algorithm

Genetic algorithms are metaheuristics that have previously been adapted for solving resource allocation problems, e.g. in (Rolia et al., 2003) or (Stillwell et al., 2010). This type of algorithms bases on evolutionary principles such as natural selection. Therefore, solution candidates are encoded in a gene sequence and form a population. We implemented a genetic algorithm with comma-selection and 1-elitism which behaves as follows. First, a random population is generated and evaluated. In each generation, offspring are created that replace the original population. These new candidates can be subject to mutation. After the population is evaluated, only a fraction of the candidates is considered for the next generation based on a selection strategy. The best individual is saved from the genetic operators in order to ensure a stable solution quality.

According to the genetic algorithms for resource allocation found in the literature, a solution candidate is encoded as an integer sequence of length m for the services. Each gene is a number between 1 and n, determining the server to which the service is allocated. An example for this encoding with $m = 6$ and $n = 4$ could be:

$$(1\ 1\ 3\ 2\ 4\ 1)$$

The population can be instantiated by choosing a random server for each service. For recombination of different solution candidates, a uniform crossover is applied. Thus, two individuals create two offspring by exchanging genes randomly. Mutation is implemented as a swap of servers for two services with a certain probability (cf. (Stillwell et al., 2010)). A tournament selection is used for choosing individuals for the next generation (Bäck, 1996). Therefore, t individuals are chosen randomly from the population and sorted by their objective function value (fitness). The individual with the kth best fitness is selected with probability $p_t(1-p_t)^{k-1}$. This procedure is repeated until $\mu - 1$ individuals are selected.

Using these genetic operators, there is the possibility that an allocation is infeasible due to overloaded servers. Repair mechanisms can be used to transform infeasible solution candidates to feasible ones (cf. (Stillwell et al., 2010)). However, the introduction of repair mechanisms introduces a bias in the search process. Therefore, penalty functions are recommended (Coit and Smith, 1996). For this problem, the following penalty function is used:

$$p(a) = \sum_{i=1}^{n} \max\left(0, \max_{1\le k\le K}\left(\left(\sum_{a(j)=i} I_j.d_k\right) - c_i\right)^2\right)$$

To calculate the penalty, the maximum quadratic overloading in a time interval is summed up for all servers. The penalty is then subtracted from the objective function value.

For the experiments conducted with the genetic algorithm, the label **GA** is used.

5.3 Genetic Algorithm with Greedy Allocation

The third class of algorithms is a hybrid of the greedy heuristics and a genetic algorithm. Because greedy algorithms such as first-fit or best-fit can always identify an optimal allocation for at least one ordering of items for one-dimensional item sizes (Lewis, 2009), the idea is to use a genetic algorithm to optimize this ordering sequence (cf. e.g. (Liu et al., 2008)). Therefore, a solution candidate is encoded as a permutation of the services of length m. Random permutations are used to initialize the population. The mutation operator swaps the position of two services with a certain mutation probability. For this genetic algorithm, tournament selection was applied, too. Recombination of solution candidates encoded as permutations provides some challenges since the result of a recombination should also be a permutation. Additionally, the offspring should contain parent information in order to guide the search process to promising solutions. Therefore, edge recombination was chosen (cf. (Larrañaga et al., 1999)). This recombination process bases on the adjacency matrices of the graphs formed by two solution candidates. In the end, the offspring should contain edges which have been contained in the parent individuals.

In order to evaluate the fitness of a solution candidate, a heuristic is applied to allocate resources for the ordered services. Because no overloading is possible, no penalty must be defined. According to the two implemented heuristics, the two hybrid algorithms are labeled as **GA_FF** and **GA_BF**. Since the three implemented metaheuristics are stochastic in their nature, 100 iterations of each algorithm are carried out to achieve meaningful results.

6 EVALUATION

In Section 5, we introduced seven algorithms for solving the optimization problem. For each algorithm and studied case (cf. Section 2), we performed optimization runs using both daily and weekly workload time intervals, resulting in a total number of 56 optimization runs. For each algorithm, we calculated mean values of their computing time and solution quality indicators. The solution quality can be defined by the percentage of reduced server capacity for the new server landscape in terms of SAPS.

Table 2 shows the mean performance of the algorithms in the eight optimization runs (two for each case), whereby *mdb* refers to the mean relative difference to the best objective value achieved across all algorithms. Therefore, the lowest number refers to the algorithm with the best mean solution quality. The frequency of achieving the best objective value is represented by *fb*. Thus, FFDmax achieved the best solution quality in 25% of all cases (2 out of 8 cases). In some cases, the same and best solution was achieved by multiple algorithms. The mean runtime varies between 0.0114 seconds and 22.460 seconds and the runtime of heuristics is only 1/1000 of the genetic algorithms.

Algorithm label	***mdb***	***fb***	**Mean runtime in s**
FFDmax	0.2030	0.250	0.0125
FFDsum	0.1780	0.250	0.0114
BFDmax	0.0029	0.375	0.0217
BFDsum	0.0056	0.250	0.0235
GA	0.4153	0.125	20.294
GA_FF	0.1593	0.250	18.046
GA_BF	0.0009	0.875	22.460

Table 2. Comparison of the implemented algorithms

Figure 3 presents the 0.9-confidence intervals of *mdb* for the seven algorithms. It can be seen that the pure GA performs poorly while greedy heuristics that use a GA to order the input sequence perform better. For all cases, algorithms based on a best-fit allocation perform better than first-fit ones and BFDmax achieved barely better solution qualities than BFDsum. While GA_FF does not perform significantly better than any FFD heuristic, the GA_BF achieves better solution qualities in seven out of eight optimization runs and, therefore, shows the lowest *mdb* by a clear margin.

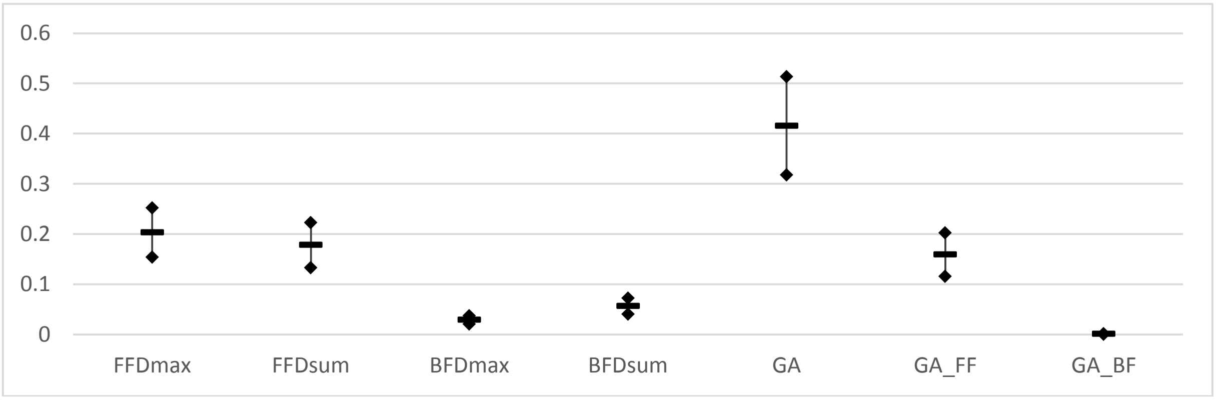


Figure 3. Mean relative differences (confidence intervals) to best objective value

The *mdb* was used as a relative value in order to compare the developed algorithms to each other and evaluate, which of them achieves best solution quality results. The actual quality, of course, reflects the exploited optimization potential for each studied case. As the GA_BF performed best for all cases, Table 3 shows the savings in terms of SAPS capacity when comparing the studied server landscape with the server landscape that was optimized using the GA_BF. In general, high savings up to 83% were possible and the mean utilization could be increased massively by up to factor 6. In 3 out of 4 cases, using daily patterns based on workloads for each hour leads to more savings compared to weekly patterns that are based on workloads for each weekday. The optimization potential highly depends on the workload of all services. Therefore, in case 2, only 25% of the servers (one out of four) can be switched off since these already show a comparatively high mean utilization. In contrast, the workload patterns of case 1, 3 and 4 enabled the optimization run to exclude more than 50% of the running servers.

	Case 1	**Case 2**	**Case 3**	**Case 4**
Number of services	96	15	103	97
Number of servers (before optimization)	31	4	89	82
SAPS capacity (before optimization)	274,000	20,184	374,000	516,500
Mean utilization (before optimization)	16.51 %	51 %	16.52%	8.25%
Number of servers (after optimization)	14	3	22	11
SAPS capacity (after optimization)	83,112	13,875	95,905	87,554
Mean utilization (after optimization)	72.2 %	74.2 %	54.4 %	51.5 %
Maximum SAPS reduction	69.67 %	31.26 %	74.36 %	83.05 %
Best pattern	weekly	both	daily	daily

Table 3. Exploited optimization potential for each studied case

Whereas Table 3 shows aggregated results, we also studied solution qualities for particular servers in order to evaluate service placement decisions of the algorithms. Therefore, Figure 4 exemplifies details on the CPU utilization levels for two servers which we have randomly taken from Case 1. In order to comply with privacy regulations, actual names of ERP instances were renamed to Service 1-n. As can be seen in Figure 4, existing low utilization levels were raised by grouping an increased number of services on the same server. The services were chosen with respect to their workload orthogonality, thus, capacity limits are never exceeded when deploying the resulting allocation solutions for the monitored workload.

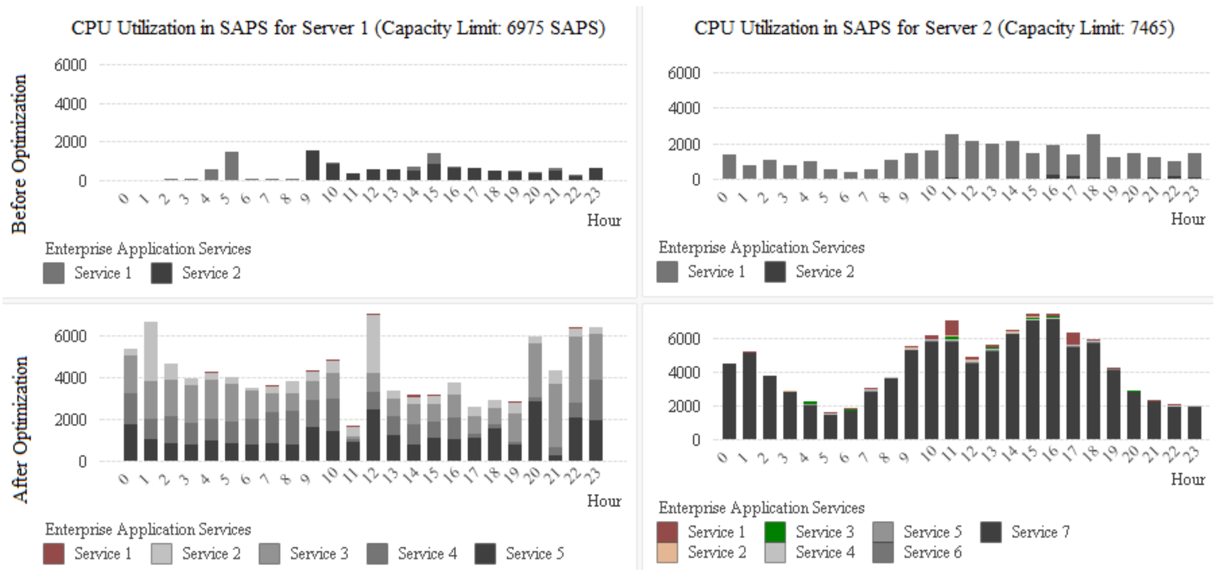


Figure 4. Comparison of server utilization levels before and after optimization run

The savings, listed in Table 3, were achieved using a mixed quantile configuration for the extraction of workload values from the historical traces. Our approach allows the configuration of a quantile for each service type. In the mixed configuration, we used the 0.98-quantile for productive and database services. For workloads of development services, we used the 0.90 quantile and for quality assurance services the 0.95-quantile was taken as a basis. Any other services that could not be classified using these types were treated as productive services, thus, using the 0.98-quantile. Since the quantile configuration influences the probability of overload situations, we compared three different configurations to each other, presented in Table 4. In the peak configuration, each data vector is mapped to its maximum as a reference for the other scenarios. In the secondly listed configuration, the 0.95-quantile is used for each service with no respect to its priority as conducted in Speitkamp2010. In our experiments, using the 0.95-quantile achieves better savings, since most services are productive and fewer resources are allocated to productive systems. However, the mean performance that can be guaranteed for productive systems is less than 84 % while in the mixed configuration almost 97 % can be achieved. Hence, the effect of different quantiles for different priorities can be significant.

	Mean savings compared to peak value	**Mean performance of productive systems**
1.0-quantile (peak)	0.00 %	100.00 %
0.95-quantile	23.85 %	83.99 %
Mixed quantiles	15.84 %	96.96 %

Table 4. Comparison of different quantile functions

Overall, efficient solutions for solving the *DPWCP* were conducted using the mixed quantile configuration. Heuristics provide good results within a fraction of a second while a GA consumes significantly more time for worse results. In contrast, the best-fit approach with GA-optimized input sequence provides best results. To compare the algorithm's scalability, we created a fifth, theoretical case of over 1000 services that represents all services and servers of case 03 multiplied by 12. In this case, GA_BF computed best results within 40 minutes, which indicates a polynomial complexity. In comparison, (Speitkamp and Bichler, 2010) mentions that such cases cannot be solved within 24 hours using their developed LP-relaxion-based heuristic.

7 CONCLUSION AND FUTURE WORK

Average utilization rates of servers in enterprise environments have been identified to be very low. Using workload consolidation, utilization rates can be increased and idle servers can be switched off. This is beneficial for the data center's energy efficiency and, therefore, reduces the TCO. In order to distribute all workloads of services across available physical resources in an optimal way, a bin packing problem was formulated. To ensure that individual performance requirements of particular service types are met, a priority function was embedded into the problem formulation. Since bin packing is known to be NP-hard, seven different heuristics and metaheuristics were applied. All developed algorithms were able to optimize existing allocation scenarios that were gathered from four productively running data centers. However, the solution quality and computation time differs. While a genetic algorithm performed poorly, greedy heuristics such as first-fit-decreasing (FFD) and best-fit-decreasing (BFD) calculated useful results after fractions of seconds. Since greedy heuristics can always identify an optimal allocation for at least one ordering of items for one-dimensional item sizes, we developed a best-fit approach that uses a genetic algorithm to order the input sequence. This GA_BF achieved the best solution qualities for all studied cases within up to 22.5 seconds. Furthermore, we extracted both daily (hourly time interval) and weekly (daily time interval) workload patterns from the historical traces and compared their respective solution qualities to each other. In most cases, using hourly time intervals leaded to better results in terms of capacity savings. The GA_BF shows a polynomial complexity, thus, a theoretical case that comprises more than 1000 services could be solved within 40 minutes which indicates a significant improvement compared to the most closely related work found in the literature. Equally important, optimization algorithms need to consider service priorities in order to guarantee SLA-compliant performance. Therefore, identified optimization potential was efficiently and practically addressed by solving the DPWCP. For the future, several extensions to the work presented in this paper are planned. First, a field study will be conducted in order to compare the developed algorithms and their solution qualities for a greater number of different data centers. Therefore, we plan to add a variety of additional cases whereas developed cleaning and processing logic can be leveraged. In order to be able to include also memory-intensive applications like in-memory databases, we plan to add memory consumption as another dimension to the bin packing problem. Furthermore, together with our industry partner, we plan to automate the developed approach in a way that consultants can consume a web service that processes workload data and server capacities of any customer data center.

References

Bäck, T. (1996). Evolutionary Algorithms in Theory and Practice. Oxford University Press.
Barroso, L.A., Clidaras, J. and Hölzle, U. (2013). The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines Synthesis Lectures on Computer Architecture, 8 (3), 1–154.
Beloglazov, A. and Buyya, R. (2010). Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers. In Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science ACM.
Bichler, M., Setzer, T. and Speitkamp, B. (2006). Capacity planning for virtualized servers. In Workshop on Information Technologies and Systems (WITS) Milwaukee, WI, USA.
Cherkasova, L. and Rolia, J. (2006). R-Opus: A composite framework for application performability and QoS in shared resource pools. In International Conference on Dependable Systems and Networks IEEE.
Coit, D.W. and Smith, A.E. (1996). Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm IEEE Transactions on Reliability, 45, 254–266.
Fan, X., Barroso, L.A. and Weber, W.-D. (2007). Power Provisioning for a Warehouse-sized Computer. In ACM International Symposium on Computer Architecture pp. 13–23.
Feller, E., Rilling, L. and Morin, C. (2011). Energy-aware ant colony based workload placement in clouds. In Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing. IEEE.
Ferreto, T.C., Netto, M.A.S., Calheiros, R.N. and De Rose, C.A.F. (2011). Server consolidation with migration control for virtualized data centers Future Generation of Computer Systems, 27, 1027–1034.
Filani, D., He, J., Gao, S., Rajappa, M., Kumar, A., Shah, P. and Nagappan, R. (2008). Dynamic Data Center Power Management: Trends, Issues, and Solutions. Intel Technology Journal, 12 (1).
Gao, Y., Guan, H., Qi, Z., Hou, Y. and Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing Journal of Computer and System Sciences, 79, 1230–1242.
Gartner (2011). Ten Key Actions to Reduce IT Infrastructure and Operations Costs. URL: http://www.gartner.com/newsroom/id/1807615 .
Heath, T., Diniz, B., Carrera, E.V., Meira, W., Jr and Bianchini, R. (2005). Energy Conservation in Heterogeneous Server Clusters. In 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming pp. 186–195.
Hevner, A.R., March, S.T., Park, J. and Ram, S. (2004). Design Science in Information Systems Research Management Information Systems Quarterly, 28 (1), 75–105.
IT-Onlinemagazin (2015). Users Rate SAP-Innovations: Survey Result 2015. URL: http://it-onlinemagazin.de/anwender-bewerten-sap-innovationen-umfrageergebnisse-2015/ .
Jennings, B. and Stadler, R. (2015). Resource Management in Clouds: Survey and Research Challenges Journal of Network and Systems Management, 23, 567–619.
Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. Technical Report. Analytics Press, Oakland, CA.
Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I. and Dizdarevic, S. (1999). Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators Artificial Intelligence Review, 13 (2), 129–170.
Lewis, R. (2009). A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing Computers & Operations Research, 36, 2295–2310.
Liu, D.S., Tan, K.C., Huang, S.Y., Goh, C.K. and Ho, W.K. (2008). On solving multiobjective bin packing problems using evolutionary particle swarm optimization European Journal of Operational Research, 190, 357–382.

Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D. and Yuan, L. (2010). Online Self-Reconfiguration with Performance Guarantee for Energy-Efficient Large-Scale Cloud Computing Data Centers. In IEEE International Conference on Services Computing (SCC) pp. 514–521, IEEE, Miami, FL, USA.

Müller, H. and Bosse, S. (2016). Multidimensional Workload Consolidation for Enterprise Application Service Providers. In AMCIS 2016 Proceedings.

Müller, H. and Turowski, K. (2015). Big Data on Performance Logs-A Collaborative Monitoring Cloud for ERP Systems. In Proceedings on the International Conference on Internet Computing (ICOMP) p. 75, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Ohlin, M. and Kjær, M.A. (2007). Nice resource reservations in Linux. In Proceedings, Second IEEE International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBID07), Munich, Germany.

Orgerie, A.-C., De Assuncao, M.D. and Lefevre, L. (2014). A Survey on Techniques for Improving the Energy Efficiency of Large Scale Distributed Systems ACM Computing Surveys, 46 (4), 1–35.

Peffers, K., Tuunanen, T., Rothenberger, M.A. and Chatterjee, S. (2007). A design science research methodology for information systems research Journal of Management Information Systems, 24 (3), 45–77.

Petrucci, V., Carrera, E.V., Loques, O., Leite, J.C.B. and Mossé, D. (2011). Optimized Management of Power and Performance for Virtualized Heterogeneous Server Clusters. In 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) pp. 23–32, IEEE, Newport Beach, CA, USA.

Poli, R., Woodward, J. and Burke, E.K. (2007). A Histogram-matching Approach to the Evolution of Bin-packing Strategies. In IEEE Congress on Evolutionary Computing (CEC) pp. 3500–3507, IEEE, Singapore.

Rolia, J., Andrzejak, A. and Arlitt, M. (2003). Automating Enterprise Application Placement in Resource Utilities. In 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM), LNCS (Brunner, M. and Keller, A. Ed.), pp. 118–129, Springer.

Rolia, J., Cherkasova, L., Arlitt, M. and Andrzejak, A. (2005). A capacity management service for resource pools. In 5th International Workshop on Software and Performance (WOSP) pp. 229–237, ACM.

Sahoo, J., Mohapatra, S. and Lath, R. (2010). Virtualization: A survey on concepts, taxonomy and associated security issues. In 2010 Second International Conference on Computer and Network Technology (ICCNT) pp. 222–226, IEEE.

SAP (2015a). Adaptive Computing. URL: http://scn.sap.com/docs/DOC-8646 .

SAP (2015b). SAP Benchmark Glossary. URL: http://global.sap.com/campaigns/benchmark/bob_glossary.epx#s .

Setzer, T. and Stage, A. (2010). Decision support for virtual machine reassignments in enterprise data centers. In IEEE/IFIP Network Operations and Management Symposium (NOMS) Workshops IEEE, Osaka, Japan.

Speitkamp, B. and Bichler, M. (2010). A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers IEEE Transactions on Services Computing, 3, 266–278.

Splieth, M., Bosse, S., Schulz, C. and Turowski, K. (2015). Analyzing the Effects of Load Distribution Algorithms on Energy Consumption of Servers in Cloud Data Centers. In Proceedings of the 12th International Conference on Wirtschaftsinformatik (WI).

Stillwell, M., Schanzenbach, D., Vivien, F. and Casanova, H. (2010). Resource Allocation Algorithms for Virtualized Service Hosting Platforms Journal of Parallel and Distributed Computing, 70, 962–974.

Vogels, W. (2008). Beyond server consolidation Queue, 6 (1), 20–26.

Xu, J. and Fortes, J.A.B. (2010). Multi-objective Virtual Machine Placement in Virtualized Data Center Environments. In IEEE/ACM International Conferenc on Cyber, Physical and Social Computing (CPSCom) IEEE, Hangzhou, China.