Maximum Migration Time Guarantees in Dynamic Server Consolidation for Virtualized Data Centers

Tiago Ferreto[1], César A.F. De Rose[1], and Hans-Ulrich Heiss[2]

[1] Faculty of Informatics, PUCRS, Brazil
tiago.ferreto@pucrs.br, cesar.derose@pucrs.br
[2] Technische Universitaet Berlin, Germany
heiss@cs-tu-berlin.de

Abstract. Server consolidation is a vital mechanism in modern data centers in order to minimize expenses with infrastructure. In most cases, server consolidation may require migrating virtual machines between different physical servers. Although the downtime of live-migration is negligible, the amount of time to migrate all virtual machines can be substantial, delaying the completion of the consolidation process. This paper proposes a new server consolidation algorithm, which guarantees that migrations are completed in a given maximum time. The migration time is estimated using the max-min fairness model, in order to consider the competition of migration flows for the network infrastructure. The algorithm was simulated using a real workload and shows a good consolidation ratio in comparison to other algorithms, while also guaranteeing a maximum migration time.

1 Introduction

Server consolidation is a key feature in current virtualized data centers. It focuses on minimizing the amount of resources required to handle the data center workload, and therefore, it has a direct impact on the costs of the data center infrastructure. Due to the variations in demand of the applications executed in the data center, virtual machines capacities should be periodically revisited in order to provide good performance to the applications and minimize overprovisioning. A server consolidation algorithm evaluates these capacity changes and derives a new mapping of virtual machines to the available resources. The new mapping may require migrating virtual machines among physical servers, which can be performed using live-migration techniques with negligible downtime.

Although the migration of virtual machines is imperceptible for the users, migrating several virtual machines concurrently can require a considerable amount of time, which results in a larger delay in the server consolidation process. Therefore, estimating the total migration time is essential when planning the server consolidation of virtual machines. Even so, current server consolidation algorithms disregard this matter.

E. Jeannot, R. Namyst, and J. Roman (Eds.): Euro-Par 2011, LNCS 6852, Part I, pp. 443–454, 2011.

This paper proposes a new server consolidation algorithm which minimizes the number of required physical servers to handle a set of virtual machines, while also guaranteeing that the migrations performed in the transition to the new mapping be completed in a specified maximum time. The algorithm was evaluated using real workloads from TU-Berlin and compared with common implementations of the server consolidation problem, such as heuristics and linear programming. The results obtained show that the algorithm is able to guarantee a maximum migration time in most cases and also minimizes considerably the number of migrations performed, while requiring a small amount of additional physical servers.

2 Related Work

The server consolidation problem consists in mapping a set of virtual machines with different capacities to a set of physical servers in order to minimize the number of physical servers required. This problem is analogous to the classic bin-packing problem, which is classified as an NP-hard problem [6], and aims at mapping a set of items with different capacities into a minimal set of bins. There are several approaches in the literature to solve this problem, in which the most common use heuristics and linear programming. Some of the most common heuristics for the bin-packing problem are [6,11]: first-fit, best-fit, worst-fit and almost worst-fit. Each heuristic uses a different policy to select which bin should receive an item. A common optimization is to order the items in decreasing order before starting the mapping process, resulting in the algorithms: first-fit decreasing, best-fit decreasing, worst-fit decreasing and almost worst-fit decreasing. The main goal of heuristics is to find good solutions at a reasonable computational cost, however it does not guarantee optimality. Another common approach is to use linear programming to find an optimal solution. The drawback of this approach is that it usually has higher requirements in terms of computing power and time.

Applications running in data centers usually present periods of high and low utilization. In order to minimize the amount of active resources, the capacity required in each virtual machine is periodically evaluated, and a new mapping is produced using server consolidation. Server consolidation techniques have widespread adoption in virtualized data centers. However, the process of mapping virtual machines to physical servers is not trivial. Depending on the application requirements and the resource provider goals, different strategies can be applied. Several works have been published in the last years proposing different approaches in the server consolidation process.

Andrzejak *et al.* [2] proposed static and dynamic server consolidation algorithms based on integer programming and genetic algorithm techniques. The algorithms were evaluated using a production workload containing traces from enterprise applications. The results present the benefits of using server consolidation, showing that the same workload could be allocated in a much smaller number of physical servers. The genetic algorithm resulted in solutions as good

as with integer programming, with the benefit of reaching the solution much faster. In this work, migrations are considered to happen instantaneously, i.e., there is no migration cost included in the algorithms.

Speitkamp and Bichler [4,15] described linear programming formulations for the static and dynamic server consolidation problems. They also designed extension constraints for limiting the number of virtual machines in a physical server, guaranteeing that some virtual machines are assigned to different physical servers, mapping virtual machines to a specific set of physical servers that contain some unique attribute, and limiting the total number of migrations for dynamic consolidation. In addition, they proposed an LP-relaxation based heuristic for minimizing the cost of solving the linear programming formulations.

Bobroff *et al.* [5] proposed a dynamic server consolidation algorithm, which focus on minimizing the cost of running the data center. The cost is measured using a penalty over underutilized and overloaded physical servers, and over service level agreements (SLA) violations, defined as CPU capacity guarantees. The algorithm uses historical data to forecast future demand and relies on periodic executions to minimize the number of physical servers to support the virtual machines.

Khanna *et al.* [9] proposed a dynamic management algorithm, which is triggered when a physical server becomes overloaded or underloaded. The main goals of their algorithm are to: i) guarantee that SLAs are not violated (SLAs are specified considering mainly response time and throughput); ii) minimize migration cost; iii) optimize the residual capacity of the system; and iv) minimize the number of physical servers used. Migration cost is defined as the amount of resources used by each virtual machine.

Wood *et al.* [17] developed the Sandpiper system for monitoring and detecting hotspots, and remapping/reconfiguring virtual machines whenever necessary. In order to choose which virtual machines to migrate, Sandpiper sorts them using a volume-to-size ratio (VSR), which is a metric based on CPU, network, and memory loads. Sandpiper tries to migrate the most loaded virtual machine from an overloaded physical server to one with sufficient spare capacity.

Mehta and Neogi [13] introduced the ReCon tool, which aims at recommending dynamic server consolidation in multi-cluster data centers. ReCon considers static and dynamic costs of physical servers, the costs of virtual machine migration, and the historical resource consumption data from the existing environment in order to provide an optimal dynamic plan of virtual machines to physical server mapping over time. Virtual machine migration costs are defined as directly related to amount of resources used by the VM, such as CPU and memory. Similarly, Verma *et al.* [16] developed the pMapper architecture and a set of server consolidation algorithms for heterogeneous virtualized resources. The algorithms take into account power and migration costs and the performance benefit when consolidating applications into physical servers. In the pMapper architecture, the migration cost is analyzed as the impact of the migration during the application execution. However, the migration cost model only considers the impact when migrating a single virtual machine, i.e. the migration cost does not change when several migrations occur concurrently.

Despite the several approaches investigated in server consolidation, none of them have already studied the real impact of virtual machines migration in the server consolidation process. Most of the works that deal with virtual machine migration only take into account the number of migrations, or the amount of memory transferred (related to the amount of resources used by the VM), but the impact of these transfers in the completion of the consolidation process have never been explored. It is necessary to estimate how long does it take to migrate each virtual machine considering that they compete for the network infrastructure.

3 Server Consolidation Algorithm

The server consolidation problem focus on minimizing the number of physical servers required to map a list of virtual machines, respecting the capacity of physical servers and demands of virtual machines. In the algorithm proposed here, besides minimizing the number of physical servers, it also aims at establishing a maximum migration time during the server consolidation process. The migration time directly reflects the amount of time required to complete the consolidation.

One of the main challenges is to accurately estimate migration time, taking into consideration that migrations are performed in a shared network infrastructure, and it may affect the available bandwidth for each migration. In the proposed algorithm, the evaluation of the available bandwidth for each migration is performed using the max-min fairness (MMF) model [8,14]. This model is often considered in the context of IP networks carrying elastic traffic. It presents the following properties: i) all transfers have the same priority over the available bandwidth, ii) link bandwidths are fairly shared among transfers being allocated in order of increasing demand, iii) no transfer gets a capacity larger than its demand, and iv) transfers with unsatisfied demands get an equal share of the link bandwidth.

Given a set of network links with respective bandwidths and the links used by each migration, it is possible to obtain the available bandwidth for each migration using a progressive filling algorithm which respects the MMF model properties [3]. The algorithm initializes the bandwidth available for each transfer with 0. It increases the bandwidth for all transfers equally, until one link becomes saturated. The saturated links serve as a bottleneck for all transfers using them. The bandwidths for all transfers not using these saturated links are incremented equally until one or more new links become saturated. The algorithm continues, always equally incrementing all transfer bandwidths not passing through any saturated link. When all transfers pass through at least one saturated link, the algorithm stops.

The migration time is measured as the time it takes to transfer the current memory allocation of each virtual machine, with the available bandwidth obtained using the MMF model. However, it is not possible to simply divide one by the other. Considering that some migrations finish before others, the available

bandwidth for each migration can change, and the remaining amount of memory to be transferred should take into account the new available bandwidth. Therefore, migration time is measured in incremental steps.

After each migration finishes, the amount of time passed is added to the migration time of all virtual machines, and the amount of memory transferred during this time is decreased from the total amount of memory to transfer. If there is no more memory to transfer, the migration is removed from the set of running migrations. The algorithm continues until there are no more running migrations. In the end, we have an estimation of the migration time of each migration.

The proposed algorithm is divided in two distinct phases. The first phase aims at finding a feasible mapping of virtual machines to physical servers that minimizes the maximum migration time of all virtual machines. In the second phase, the feasible mapping is iteratively modified in order to produce solutions using a smaller number of physical servers, but also respecting a maximum migration time threshold. In cases where it is not possible to guarantee migration times smaller than a specified threshold, the algorithm finds a solution that minimizes the number of virtual machines that have their migration times higher than the specified threshold.

3.1 First Phase: Minimizing Migration Time

The first phase is based on the traditional descent method for local neighborhood search. Based on an initial solution, a set of small modifications to this solution is derived. The result obtained with each modification is analyzed and the best one is chosen. If this modification optimizes the current solution, then it is applied and the process repeats, otherwise the algorithm stops.

Algorithm 1 presents the algorithm for the first phase. It starts using a copy of the current mapping as the current solution. The repetition of the current mapping results in zero migrations, however the physical servers can become overloaded, i.e., the physical server capacity can not be able to handle the changes of the virtual machines demands mapped to it. The strategy is to remove each overloaded physical server from this overload state, migrating some of its virtual machines to other physical servers, choosing every time the alternative that results in minimal migration times. The algorithm generates migration alternatives for each overloaded physical server. The physical server that performs migrations which results in minimal migration time is chosen. The migrations are included in the current solution, and the process repeats, until there are no more overloaded physical servers.

The migration alternatives for each overloaded physical server are generated as the combinations of virtual machines that remove the physical server from the overload state. For example, given an overloaded physical server with capacity 100, packing virtual machines: v_1 with demand 50, v_2 with demand 40, v_3 with demand 30, and v_4 with demand 20, the algorithm generates the following combinations of virtual machines: (v_1), (v_2), and (v_3, v_4). Each combination of virtual machines is applied in the current solution using the best-fit decreasing

heuristic and its cost is evaluated. The cost function considers the maximum migration time and also the sum of the migration times of all migrations. The goal is to find a feasible mapping, respecting physical servers capacities and virtual machines demands, which minimizes this cost function, i.e., results in the lowest maximum migration time.

The algorithm repeats until there are no more overloaded physical servers. In the end, the result generated will contain at least the same amount of physical servers as the present mapping. The second phase is used to decrease the number of physical servers, while guaranteeing that the maximum migration time stays under a given threshold.

Algorithm 1. First phase of the server consolidation algorithm

```
current_solution ← getCurrentMapping()
while there are overloaded physical servers in current_solution do
  for all p in overloaded physical servers do
    alternatives ← generateMigrationAlternatives(p)
    for all alt in alternatives do
      sol ← bestFitDecreasing(alt)
      cost ← calculateCost(sol)
    end for
  end for
  current_solution ← getAlternativeWithLowestCost()
end while
```

3.2 Second Phase: Minimizing the Number of Physical Servers

The second phase is implemented using the tabu search metaheuristic [7]. The main idea of tabu search is to maintain a memory about previous local searches, in order to avoid performing repeatedly the same moves, returning to the same solution and staying confined into a local optima. The tabu search method is based on a repetition of steps that explores the possible solutions for the problem.

The second phase is presented in Algorithm 2, and it starts by using as initial solution the mapping resulted from the first phase. The strategy is to select in each iteration one physical server to empty, reassigning its virtual machines to other physical servers. The selection of the physical server is based on a filling function, proposed by [12], which gives a measure of easiness to empty a physical server. The function gives higher priority to physical servers with low occupied capacity and more virtual machines packed on it. The physical server with the lowest filling index, according to the filling function, and that is not in the tabu list is chosen. The tabu list stores a list of previously chosen physical servers, and the goal of the tabu list is to avoid choosing repeatedly the same physical servers to empty.

After choosing the physical server using the filling function, the virtual machines mapped to it are retrieved and the physical server is set as unavailable during this iteration, in order to avoid remapping all virtual machines to it again. The list of virtual machines is used as input to a permutation function, which returns lists with these virtual machines in all possible orderings. Each alternative is evaluated, applying the worst-fit heuristic to map the virtual machines to the physical servers. The alternative that provides a solution with best cost, according to the cost function, is selected and its solution is defined as the current solution. The cost function that should be minimized combines the number of physical servers and the number of breaks of maximum migration time. This last term refers to the number of migrations with migration time higher than the specified threshold.

The physical server chosen at first is included in the tabu list and set as available again to pack virtual machines in the next iterations. The tabu list size has a fixed capacity, and when this capacity is exceeded, the oldest entry is removed. The tabu list size should be smaller than the number of physical servers. If the cost of the current solution is smaller than the cost of the best solution, then the current solution is defined as the best solution. This process repeats until the best solution does not present any enhancements in a pre-specified number of iterations. The final solution can result in a situation that guaranteeing the maximum migration time is not possible, however it will minimize the number of breaks of maximum migration time.

4 Evaluation

The evaluation of the server consolidation algorithm was performed using workloads composed of traces from servers of the Technical University of Berlin (TU-Berlin), which are normally used by researchers and students to execute computational experiments. Each workload contains samples of CPU and memory utilization per hour during a week, totalizing 168 samples per trace. The workloads present different characteristics, such as: number of traces, average CPU utilization, average memory utilization, and average variability. The variability in a trace indicates the percentage of consecutive samples that present a variation in CPU or memory values. A variability of 0% indicates that the trace keeps with same CPU and memory utilization during the whole trace duration, whereas 100% indicates that each consecutive sample presents a different CPU or memory utilization. The higher the variability, higher also is the probability of changes in the mapping of virtual machines to physical machines, and hence a higher number of migrations can be performed. Table 1 presents the workloads and its characteristics.

The physical infrastructure simulated in the experiments is a data center composed of 100 homogeneous physical servers with CPU and memory capacities equal to 100. Each server is connected to a single crossbar switch using bidirectional links forming a star network topology. Each link has a capacity of 100 per unit of time. It means that it takes 1 unit of time to transfer all memory from one physical server to another one using full link capacity.

Algorithm 2. Second phase of the server consolidation algorithm

```
current_solution ← getFirstPhaseMapping()
best_solution ← current_solution
repeat
  for all p in physical_servers not in tabu_list do
    p_index ← getFillingIndex()
  end for
  p ← getPhysicalServerWithLowestIndex()
  vms ← getVirtualMachinesFrom(p)
  set p as unavailable
  moves ← getPermutations(vms)
  for all m in moves do
    sol ← worstFit(m)
    cost ← calculateCost(sol)
  end for
  current_solution ← getAlternativeWithLowestCost()
  insertIntoTabuList(p)
  set p as available
  if calculateCost(current_solution) < calculateCost(best_solution) then
    best_solution ← current_solution
  end if
until termination condition
```

Table 1. Details of TU-Berlin workload groups

	Number of traces	Avg. CPU utilization (%)	Avg. memory utilization (%)	Avg. variability (%)
Workload 1	43	25.2	28.36	17
Workload 2	61	32.37	39.39	41
Workload 3	36	47.28	48.67	63

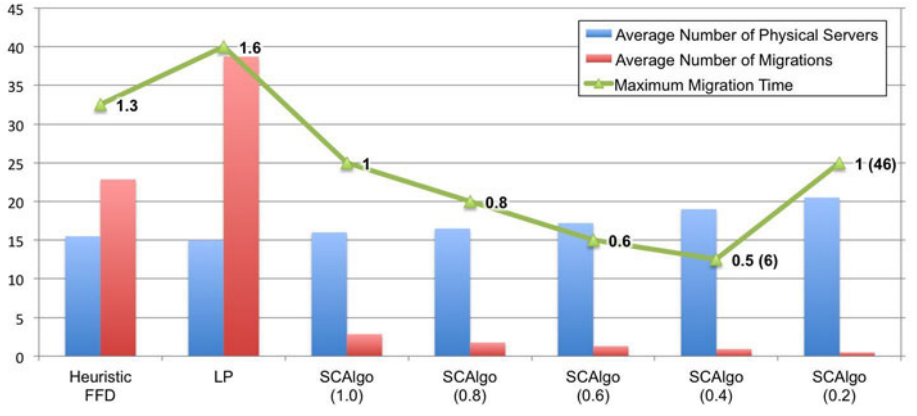
The server consolidation algorithm proposed was compared with typical implementations of the server consolidation problem using heuristics and linear programming. The heuristics implemented were: first-fit decreasing (FFD), best-fit decreasing (BFD), worst-fit decreasing (WFD) and almost worst-fit decreasing (AWFD). They were all implemented using the Python language. The linear programming (LP) solution was implemented using Zimpl [10] and solved using the SCIP [1] solver. The solver was configured with a timeout of 5 minutes, i.e., if the solver can not find the optimal result in 5 minutes, it returns the best result found so far. This approach is usually used since linear programming problems can take a long time to find an optimal solution. All experiments were performed on a Intel Core 2 Duo processor with 2.4 GHz and 4 GBytes of memory. The server consolidation algorithm proposed was implemented using the Python language. The second phase (using Tabu Search) was configured with tabu list size

equal to 5 and to terminate when the solution does not present any changes in the last 10 iterations (termination condition). The first mapping is performed using the first-fit heuristic since there is no previous mapping to be used by the algorithm. The server consolidation algorithm was executed using five different thresholds of maximum migration time. The thresholds are: 1.0, 0.8, 0.6, 0.4 and 0.2 units of time.

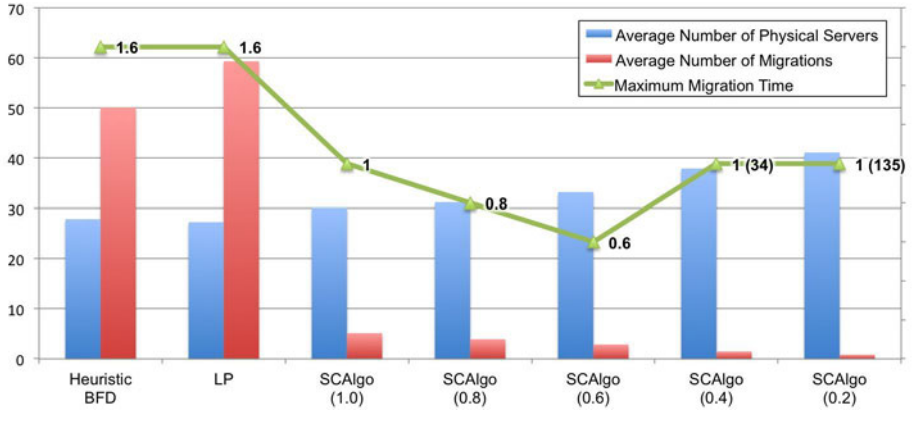
For each experiment combining algorithm and workload, the metrics measured are: the average number of physical servers required to process the workload, the average number of migrations required, and the maximum migration time. Figures 1a, 1b and 1c present the results obtained with the algorithms for each workload. The heuristics bar only presents the heuristic that presented the best solution.

LP presented the lowest average number of physical servers in all workloads, but it was closely followed by the best heuristic. However, LP also presented the highest average number of migrations, requiring migrating almost all virtual machines each consolidation step in all workloads. The average percentage of virtual machines migrated each consolidation step using LP are: 90% for workload 1, 97.2% for workload 2 and 95.8% for workload 3 using LP. These high values are due to the aggressive methods applied by linear programming in order to find an optimal solution. Despite the lowest number of physical servers, the high number of migrations is a huge obstacle for its utilization in a real environment. The heuristics required a lower number of migrations, but with a considerable increase when using workloads with higher variability. Heuristics tend to keep the mapping of virtual machines in the same physical servers when there is a low variation in the virtual machines capacities. Besides presenting a high number of migrations, LP and heuristics also presented, as expected, a high maximum migration time, considering that these algorithms only try to optimize the number of physical servers used by the workload.

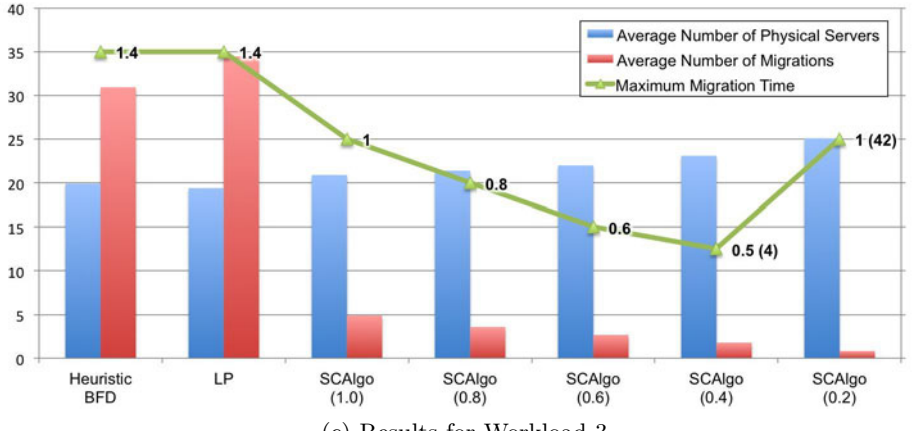
The proposed server consolidation algorithm (named as SCAlgo in the charts) was able to maintain the maximum migration time for the first three thresholds: 1.0, 0.8 and 0.6. The reduction in the number of required migrations is considerable, requiring only a small increase in the number of physical servers. In workload 1, with the time threshold of 1.0, the algorithm migrates an average of 6.5% of the virtual machines against 90% using LP, requiring only an increase of a single machine in average. Workloads 2 and 3 migrate an average of 8.3% (against 97.2% in LP) and 13.6% (against 95.8% in LP) of virtual machines, while requiring an increase of approximate 2.9 and 1.5 machines in average in comparison to the results obtained using LP. In the cases where the proposed server consolidation could not guarantee the maximum migration time (thresholds of 0.4 and 0.2), the algorithm minimized the number of migration breaks, i.e., the number of migrations that presented migration times higher than the specified threshold. In the charts, the number of migration breaks is presented in parentheses besides the maximum migration time for thresholds 0.4 and 0.2. This limitation is directly related to the characteristics of the workload, specially regarding the amount of memory utilization. Even when using



(a) Results for Workload 1



(b) Results for Workload 2



(c) Results for Workload 3

full network capacity, virtual machines with high memory demands cannot be transferred during the maximum migration time specified.

5 Conclusion and Future Work

This paper presented a new server consolidation algorithm to be used in virtualized data centers that, besides minimizing the number of physical servers used, also guarantees that all necessary migrations occur during a specified maximum migration time. The maximum migration time has direct relation to the completion of the consolidation process and, therefore, should be taken into account in the server consolidation algorithm. Several algorithms have been proposed for the server consolidation problem, but none of them have focused on ensuring a maximum migration time in order to minimize the delay in the consolidation process. The server consolidation algorithm proposed has been evaluated using real workloads and typical solutions for server consolidation using linear programming and heuristics. The results obtained indicate that the proposed algorithm can efficiently provide guarantees using common thresholds of time, with a huge decrease in the number of migrations performed and a slight increase in the number of additional physical servers required. As future work, we intend to perform more experiments using different workloads and optimize the algorithm in order to provide guarantees of maximum migration time in more complex scenarios. We also intend to analyze how resource providers can use the maximum migration time in SLAs in order to offer a more controlled environment for its users.

References

1. Achterberg, T.: SCIP - a framework to integrate constraint and mixed integer programming. Tech. Rep. 04-19, Zuse Institute Berlin (2004)
2. Andrzejak, A., Arlitt, M., Rolia, J.: Bounding the resource savings of utility computing models. Technical report hpl-2002-339, Hewlett Packard Laboratories (2002)
3. Bertsekas, D., Gallager, R.: Data Networks. Prentice-Hall, Englewood Cliffs (1992)
4. Bichler, M., Setzer, T., Speitkamp, B.: Capacity planning for virtualized servers. In: Proceedings of the 16th Annual Workshop on Information Technologies and Systems, WITS 2006 (2006)
5. Bobroff, N., Kochut, A., Beaty, K.: Dynamic placement of virtual machines for managing sla violations. In: Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (2007)
6. Coffman Jr., E., Garey, M., Johnson, D.: Approximation Algorithms for Bin Packing - A Survey. In: Approximation algorithms for NP-hard problems, PWS Publishing Co. (1996)
7. Floudas, C.C.A., Pardalos, P.M.: Encyclopedia of Optimization. Springer-Verlag New York, Inc., Secaucus (2006)
8. Ioannis, D.: New Algorithm for the Generalized Max-Min Fairness Policy based on Linear Programming. IEICE Transactions on Communications (2005)

9. Khanna, G., Beaty, K., Kar, G., Kochut, A.: Application performance management in virtualized server environments. In: Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium, NOMS 2006 (2006)
10. Koch, T.: Rapid Mathematical Programming. Ph.D. thesis, Technische Universitaet Berlin (2004)
11. Kou, L., Markowsky, G.: Multidimensional bin packing algorithms. IBM Journal of Research and development 21(5) (1977)
12. Lodi, A., Martello, S., Vigo, D.: TSpack: A Unified Tabu Search Code for Multi-Dimensional Bin Packing Problems. Annals of Operations Research 131(1-4) (2004)
13. Mehta, S., Neogi, A.: ReCon: A Tool to Recommend Dynamic Server Consolidation in Multi-Cluster Data Centers. In: Proceedings of the IEEE Network Operations and Management Symposium, NOMS 2008 (2008)
14. Nace, D., Nhatdoan, L., Klopfenstein, O., Bashllari, a.: Max-min fairness in multi-commodity flows. Computers & Operations Research 35(2) (2008)
15. Speitkamp, B., Bichler, M.: A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers. IEEE Transactions on Services Computing (2010)
16. Verma, A., Ahuja, P., Neogi, A.: pMapper: Power and migration cost aware application placement in virtualized systems. In: Proceedings of the ACM/IFIP/USENIX 9th International Middleware Conference (2008)
17. Wood, T., Shenoy, P.J., Venkataramani, A., Yousif, M.S.: Sandpiper: Black-box and gray-box resource management for virtual machines. Computer Networks 53(17) (2009)