Report for the research scholarship "Adaptive evaluation of cloud system performance"

Zoltán Ádám Mann

Department of Computer Science and Information Theory
Budapest University of Technology and Economics
zoltan.mann@gmail.com

1 Introduction

As cloud data centers (DCs) serve an ever-growing demand for computation, storage, and networking capacity, their operation is becoming a crucial issue. The energy consumption of DCs is of special importance because of both its environmental impact and its contribution to operational costs. According to a recent study, DC electricity consumption in the USA alone will increase to 140 billion kWh per year by 2020, costing US businesses 13 billion USD annually in electricity bills and emitting nearly 100 million tons of CO_2 per year [7].

In order to reduce energy consumption, DC operators use a combination of several techniques. Virtualization technology enables the safe co-existence of multiple applications packaged as virtual machines (VMs) on a single physical machine (PM), thus allowing high utilization of physical resources. Live migration makes it possible to move a working VM from one PM to another without noticeable downtime. Since the load of VMs fluctuates over time, this enables DC operators to flexibly react to such changes. In times of low demand, VMs can be consolidated to a low number of PMs, and the remaining PMs can be switched to a low-power state, leading to considerable energy savings. When load starts to rise, some PMs must be switched back to normal mode again so that VMs can be spread across a higher number of PMs.

Finding the best VM placement for the current load level is a tough optimization problem. First of all, multiple resource types must be taken into account, e.g., CPU, memory, disk, and network bandwidth. PMs have given capacity and VMs have given load along these dimensions, and this must be taken into account in VM placement. Moreover, in order to accommodate critical applications, a good placement must also respect special requirements of the VMs concerning performance and fault-tolerance. This is necessary for example for moving telecommunication functionalities into the cloud, as such applications are typically subject to real-time performance constraints and strict dependability requirements. Therefore, VM placement optimization must provide possibilities for isolation (e.g., dedicated processor cores) and anti-affinity constraints (e.g., VMs of a given type must be placed on different PMs in order to limit the impact of single-PM faults).

In the past couple of years, several different approaches have been proposed for the VM placement problem; for a recent survey, see [5]. From the problem formulation point of view, most approaches focus on minimizing energy consumption and operational costs by consolidating the workload on few PMs and switch off unused PMs, subject to capacity constraints. Some works take also the costs of migrations into account [11, 2, 9, 12] or try to minimize the number of overloaded PMs [1, 10, 3].

From an algorithmic point of view, the proposed approaches can be mostly grouped into two categories: (i) heuristics without any performance guarantees or theoretical underpinning and (ii) exact algorithms using off-the-shelf mathematic programming – mostly integer linear programming (ILP) – solvers [5]. It is dangerous to rely solely on heuristics because in some cases they can lead to extremely high costs or dramatic performance degradation of the involved applications [6]. On the other hand, the exact algorithms suggested so far all suffer from serious scalability issues, limiting their applicability to small problem instances.

2 Optimized deployment of critical applications in IaaS

The starting point of the present research activities was the joint work with Imre Kocsis and Dávid Zilahi from the Department of Measurement and Information Systems [4]. The aim of that work was to allow tenants to specify special requirements on the placement of critical VMs, such as

- the need for dedicated processor cores
- prohibition of migration
- anti-affinity constraints (at most k of a given set of VMs can be on the same PM)

In our problem formulation, we considered multiple resource dimensions (CPU cores, memory, egress packet rate, ingress packet rate, IOPS) that are particularly well suited to describe the resource needs of VMs running telco applications. We considered three distinct optimization objectives:

- Minimizing the number of active PMs in order to save energy
- Maximizing the number of future VMs for which capacity is available, in order to be prepared to quickly accommodate future VM requests
- Minimizing the impact of the failure of a PM on a tenant

These constraints and objectives together define a problem in which the goals of the tenants and the provider can be optimized jointly, leading to an allocation that represents the best trade-off between the conflicting goals.

In order to solve this highly non-trivial optimization problem, we transformed it to an integer linear programming formulation. Binary variables were used to encode which PMs are active and which VM is mapped to which PM. A further integer variable was used to capture the highest number of VMs of the same tenant allocated to the same PM (which is the worst-case impact of a PM failure on a tenant in terms of the number of affected VMs). Using these variables, all constraints could be expressed as linear inequalities. The objective function is the weighted sum of the three particular objectives mentioned above; the weights can be tuned to best reflect their relative importance.

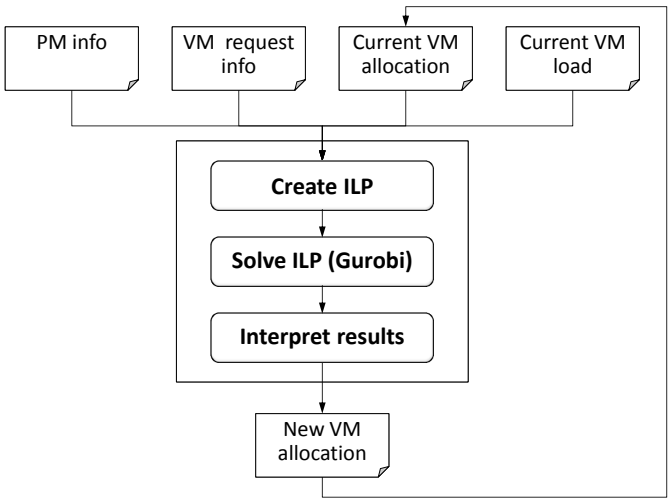

Figure 1: Deployment optimization framework

I implemented a Perl program that gathers the necessary information about the available PMs and the requested VMs, creates the integer program, runs a commercial off-the-shelf solver (Gurobi) to solve the integer program, and then determines the new VM-to-PM mapping based on the solver's output. This mechanism is depicted in Figure 1.

As a case study, we used our approach to optimize the placement of an open-source Voice-over-IP system (Clearwater), consisting of several components packaged in different VMs. Applying different values to the weights of the optimization objectives, we demonstrated that our approach is able to consolidate the workload to the minimum number of necessary PMs, to provide reserves for future VM requests, and/or to balance fault impact.

We also performed a scalability assessment. Using a time limit of 30 seconds on the optimization, the solver managed to deliver optimal results for up to 60 PMs. For bigger problem instances, the solver also gives solutions, although not necessarily optimal, and a lower bound on the optimum. This way, we could determine that the results are very close to the optimum for up to 130 PMs. Even for 200 PMs, the cost of the found solution is known to be less than twice the optimum.

Since our preliminary results were presented at the 3rd International IBM Cloud Academy Conference in May 2015, we also wrote a journal paper version containing more details, which is currently under consideration at International Journal of Cloud Computing.

Besides, the proposed approach has also been used to optimize the placement of the components of the OpenStack cloud management system, which is itself a distributed system with many components, some of which require special placement constraints.

In this context, an interesting problem is how to achieve robust partitioning of sets of replicated service instances. If two critical services A and B both have several instances to guarantee high availability, then these instances should be placed on PMs in such a way that there is at least one instance of A that is not co-located with an instance of B (and vice versa). Otherwise, an error of or attack on service B may impact all instances of A simultaneously. We also managed to incorporate this kind of constraint into our optimization framework by capturing it through appropriate linear inequalities and auxiliary variables.

3 Performance improvements through a custom optimization algorithm

Integer programming is a powerful tool for solving difficult optimization problems. Although state-of-the-art ILP solvers are highly optimized to achieve good performance, they still fall victim to combinatorial explosion and hence struggle to solve big problem instances.

An alternative to using general-purpose ILP solvers is to create a problem-specific solver. By incorporating problem-specific knowledge, better performance may be achieved.

To realize this idea in the context of the VM placement problem, we have been working on a custom branch-and-bound algorithm together with Dávid Bartók, a student working on his BSc thesis at our department. We are attacking a version of the VM placement problem in which PM capacities and VM sizes are represented as d-dimensional vectors (where d is an arbitrary positive integer), and the aim is to minimize the weighted sum of the number of active PMs and the number of migrations. In addition to capacity constraints, we also constrain the number of migrations because it is technically not feasible to perform too many migrations at the same time [8].

Our algorithm is based on a branch-and-bound framework, in which VMs are assigned to PMs one after the other in a tree-like manner, until a solution is found, or the current branch turns out to contain no solution at all or no solution that would be better than the best solution found so far; in these cases, the search backtracks.

We implemented several techniques to increase the efficiency of our algorithm:

- Incremental computation of several metrics (e.g., cost of the current partial solution, number of migrations so far etc.) in order to not have to re-compute them in every node of the search tree
- Different policies for selecting the next VM to branch on
- Different policies for sorting the PM candidates for the selected VM
- A lower bound on the cost of any solution resulting from a partial solution, based on optimally solving a relaxed problem

- Symmetry breaking
- Discarding small improvement possibilities
- Explicit limit on the runtime of the algorithm

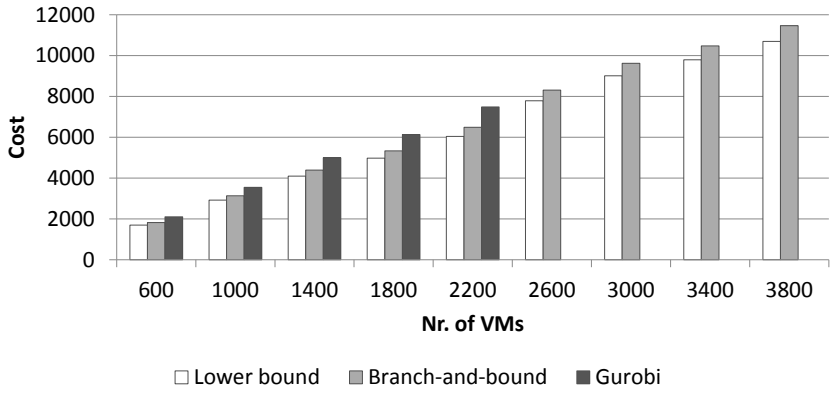

Figure 2: Scalability comparison

Using these techniques, our branch-and-bound algorithm clearly outperforms the used commercial off-the-shelf solver (Gurobi) for big problem instances. As shown in Figure 2, Gurobi does not find any valid solution for problems with more than 2200 VMs within the applied time limit of 60 seconds, whereas our algorithm produces good results even for much larger problem instances. Moreover, the results of the branch-and-bound algorithm are in most cases within 10% of the lower bound, and therefore, also within 10% of the optimum.

4 Publications

The results are described in the following papers:

- Imre Kocsis, Zoltán Ádám Mann, and Dávid Zilahi: Optimal deployment for critical applications in Infrastructure as a Service. Under consideration at International Journal of Cloud Computing.
- Dávid Bartók and Zoltán Ádám Mann: A branch-and-bound approach to virtual machine placement. HPI Cloud Symposium “Operating the Cloud”, 2015.

From the latter paper, also a journal version is planned.

5 Future work

Both of the above topics deserve further research. Relating to the deployment of critical VMs in the cloud, the used problem model can be further extended to make it more realistic, for example considering inter-VM interaction, such as dependencies, communication, and load correlation. Our branch-and-bound algorithm could be further improved by several further techniques, such as using randomization and restarts or parallelization. Especially promising is the combination of the two topics by applying the branch-and-bound approach to the problem of optimizing the deployment of critical VMs in the cloud.

References

[1] Anton Beloglazov and Rajkumar Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366–1379, 2013.

[2] D. Breitgand and A. Epstein. SLA-aware placement of multi-virtual machine elastic services in compute clouds. In *12th IFIP/IEEE International Symposium on Integrated Network Management*, pages 161–168, 2011.

[3] Rahul Ghosh and Vijay K. Naik. Biting off safely more than you can chew: Predictive analytics for resource over-commit in IaaS cloud. In *5th International Conference on Cloud Computing*, pages 25–32. IEEE, 2012.

[4] Imre Kocsis, Zoltán Ádám Mann, and Dávid Zilahi. Optimal deployment for critical applications in infrastructure as a service. In *3rd International IBM Cloud Academy Conference (ICACON 2015)*, 2015.

[5] Z. Á. Mann. Allocation of virtual machines in cloud data centers – a survey of problem models and optimization algorithms. *ACM Computing Surveys*, 48(1), 2015.

[6] Z. Á. Mann. Rigorous results on the effectiveness of some heuristics for the consolidation of virtual machines in a cloud data center. *Future Generation Computer Systems*, 51:1–6, 2015.

[7] Natural Resources Defense Council. Scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers. `http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf`, 2014.

[8] W. Song, Z. Xiao, Q. Chen, and H. Luo. Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, 63(11):2647–2660, 2014.

[9] P. Svärd, W. Li, E. Wadbro, J. Tordsson, and E. Elmroth. Continuous datacenter consolidation. Technical report, Umea University, 2014.

[10] L. Tomás and J. Tordsson. An autonomic approach to risk-aware data center overbooking. *IEEE Transactions on Cloud Computing*, 2(3):292–305, 2014.

[11] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53(17):2923–2938, 2009.

[12] Zhen Xiao, Weijia Song, and Qi Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1107–1117, 2013.