Benjamin Byholm | Ivan Porres

Dynamic Horizontal and Vertical Scaling of Multiple Cloud Services in Soft Real-Time

Turku Centre *for* Computer Science

TUCS Technical Report
No 1182, May 2017

Dynamic Horizontal and Vertical Scaling of Multiple Cloud Services in Soft Real-Time

Benjamin Byholm
Department of Information Technologies
Åbo Akademi University
Ivan Porres
Department of Information Technologies
Åbo Akademi University

TUCS Technical Report
No 1182, May 2017

Abstract

We approximately solve the problem of dynamic horizontal and vertical scaling of multiple cloud services in soft real-time. We recognize that this is a generalized bin packing problem combined with a linear assignment problem. We formalize the problem domain and develop an autonomous resource management system with soft real-time constraints. The system incorporates a genetic algorithm with quadratic expected time complexity for solving the packing problem, providing a service deployment plan with an optimal number of servers, and a successive shortest path algorithm with quadratic expected time complexity for finding an optimal way of realizing the obtained deployment plan.

Keywords: Cloud computing, Consolidation, Elasticity, Packing

TUCS Laboratory
Software Engineering Laboratory

1 Introduction

We approximately solve the problem of dynamic horizontal and vertical scaling of multiple cloud services in soft real-time. We wish to find an optimal assignment of services to servers, minimizing the number of servers and service instances, since service instances incur overhead through memory use and more complex management, while operational servers incur monetary and energy costs. This requires three types of changes in the configuration of the data center: *scaling horizontally* by increasing or decreasing the number of servers, *scaling vertically* by increasing or decreasing the resource allocation for a service in a given server, as well as service and server *consolidation*, where services move between servers, leaving empty servers which can terminate.

Let us assume as an example, that a data center hosts three services (a, b and c) in three servers. Let us assume that each server can execute 500 million instructions per second (MIPS), while the current demand is 600 MIPS for service a, 100 MIPS for service b and 300 MIPS for service c. If the resource demand for a service exceeds the capacity of a server, the service provider deploys the service in multiple servers (horizontal scaling), using a load-balancing mechanism to spread the requests evenly between the servers. Figure 1 shows that each server is underutilized, due to the policy that the load associated to service a is distributed evenly along its two instances. But, if we discard this policy, we can consolidate the services in two servers, as shown in the right side of the figure.

Due to the high volatility in the load of many Internet-scale applications, the algorithms involved must be fast and produce solutions before they become irrelevant due to changed premises, i.e. operate under soft real-time constraints. We need to cope with large clusters consisting of millions of services and servers without spending too much resources, since that is expensive.

In practice, server consolidation entails service migration. Whether services are stateful or stateless, migration comes at a cost. For a stateful service, session state must be migrated. For a so-called stateless service, its state is usually stored in a remote database. To reduce latency and bandwidth, it is sensible to employ a service-local cache of session state. Here, migration comes at the opportunity cost of losing the cache. For a genuinely stateless service, migration is still not free: Migrating a service entails downtime and network costs. Hence, we should migrate as little as possible.

In this article, we present a formal definition of this problem as well as algorithms that work under soft real-time constraints. To solve the problem, we divide it in two parts: obtaining an optimal assignment of sessions to servers and realizing said assignment with minimal effort. To obtain a nearly optimal assignment, we solve a packing problem based on the sets of servers and services. To realize the assignment, we solve an assignment problem between the existing and desired assignments. This minimizes the amount of

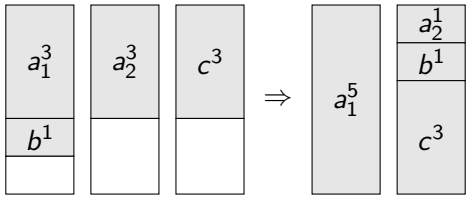

Figure 1: Example of consolidation with horizontal and vertical scaling.

session state we must move. We thus attain near-optimal, dynamic horizontal and vertical scaling of multiple cloud services in soft real-time.

2 Preliminaries

In this section we describe the problem domain of service scaling and server consolidation, as well our model for cloud servers and computing resources.

2.1 Problem Domain

Without loss of generality, consider a set of services $\boldsymbol{\mathcal{S}}$ and a set of servers $\boldsymbol{\mathcal{M}}$. A server $m \in \boldsymbol{\mathcal{M}}$ provides concurrent computing resources to a set of containers $\boldsymbol{\mathcal{C}}_m \subseteq \boldsymbol{\mathcal{C}}$, which implement services $s \in \boldsymbol{\mathcal{S}}$. A container isolates a service instance and provides access to a guaranteed amount of its server's resources. At least $k_s \geq 0$ containers implement any given service $s \in \boldsymbol{\mathcal{S}}$. A container can grow and shrink on demand, limited by the resources of its server.

We consider two types of services: stateless, transaction-oriented services and stateful, session-oriented services. Transaction-oriented services are stateless and fit into the fully open queueing model. Transactions can be processed indistinctly by any container implementing the service. Schroeder et al. (2006) note that session-oriented services fit into the half-open queueing model. Sessions can move between containers at a cost, because session state must be moved. This also applies to so-called stateless services that store session state in a database. By caching state in the server, latency is reduced. Moving a session incurs the opportunity cost of losing the cache, having to retrieve the state from the database. Thus, session-oriented services require that we also keep track of which container each session is associated with.

Based on this discussion, we assume that each service has a set of associated sessions $\boldsymbol{\mathcal{D}}_s \subseteq \boldsymbol{\mathcal{D}}$, which determine what containers they must deploy to servers. Figure 2 depicts the main concepts of our problem domain as a unified modeling language (UML) class diagram. Sessions can move between containers implementing a service at a cost. Each session belongs to a specific, fixed service and has an affinity for a specific container that varies over time.

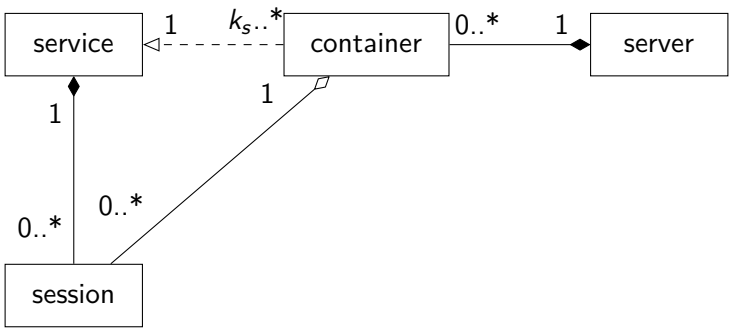

Figure 2: Services are associated with sessions, determining how many containers they must deploy to servers. A session has affinity for its container.

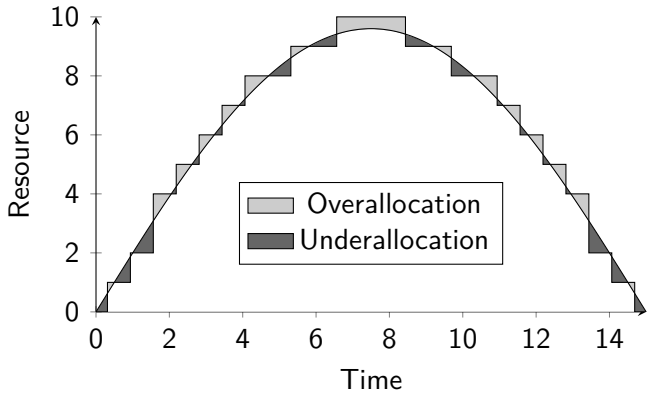

Figure 3: Alllocation must be quantized with a given resolution. Overallocation is an opportunity cost, while underallocation leads to lost revenue.

2.2 Sessions, Services and Computing Resources

Every service $s \in \boldsymbol{\mathcal{S}}$ has a set of associated sessions $\boldsymbol{\mathcal{D}}_s \subseteq \boldsymbol{\mathcal{D}}$. Each session requires a nonzero amount of the bottleneck resource. We assume that there is only one bottleneck resource in the entire system. Given a performance model of each service, which we can derive at runtime, we know how many sessions any server can handle for any service.

Resources are discrete quantities. To make the problem tractable, we must quantize resource supply and demand at some resolution, as shown in Figure 3. While we might waste capacity when quantizing resources, there exists a quantum ε_r for which all smaller quanta drown in noise and overhead dominates. Hence, we do not consider this a problem in practice, particularly since cloud systems have high volatility and we can pick an arbitrarily small quantization error. Smaller error requires increased computation time for the algorithms described in this paper. The operator determines this parameter.

We allow for the sessions $d \in \boldsymbol{\mathcal{D}}_{sc} \subseteq \boldsymbol{\mathcal{D}}$ to have an affinity for a particular

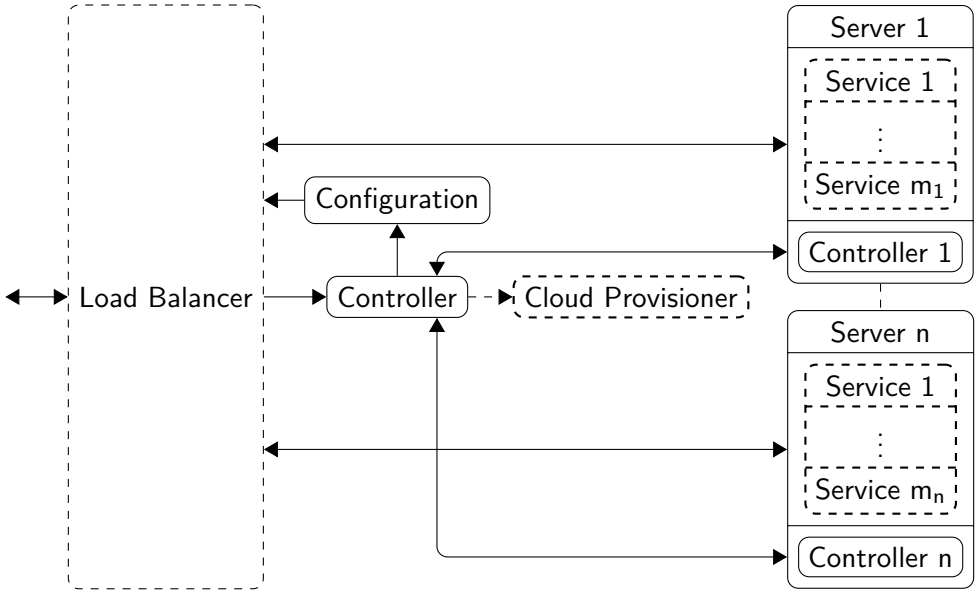

Figure 4: Architecture of a cloud system supporting dynamic scaling.

container $c \in \boldsymbol{\mathcal{C}}$. Usually this is the container that has handled the previous requests in the session. This captures session state stored in containers, whether soft or hard state. After quantization, we are left with a set of services requiring various amounts of the bottleneck resource and a set of servers capable of accommodating various container sizes. We wish to find an allocation of services to servers with the fewest containers.

2.3 Services and Containers

Our approach is mostly agnostic with respect to the underlying technology used for containers $c \in \boldsymbol{\mathcal{C}}$, it works for virtual machines (VMs), Docker containers, sandboxes, policy groups, etc. The only two requirements that we impose for such technology is that it is possible to deploy and concurrently run multiple containers in the same server and that it is possible to limit the server resources allocated to each container.

3 Solving the Problem

The dynamic scaler takes action whenever it sees that a server can be eliminated, when a container cannot grow unobstructed or when a new service cannot fit anywhere. It reorganizes the container assignment in an efficient manner. The system is depicted in Figure 4.

3.1 Main algorithm

The main algorithm executes whenever the sets of sessions for each service changes. Its logic is depicted in Figure 5. If a new session arrives, there are two possibilities: Either there is sufficient capacity in some container, and we assign the session to that container, or no container can accommodate the session. If no container has sufficient capacity, at least one container must grow by at least one unit. If there is a suitable container on a server with unused capacity, we simply grow the container by a unit and assign the session to this container. If no container has room to grow, we add a server, make a new assignment plan and migrate sessions to realize the new plan with as little effort as possible. If a session leaves, we shrink the container that lost a session, if possible. If this allows us to use at least one server fewer, we make a new assignment plan, migrate sessions and terminate unused servers.

3.2 Pack: Assigning Services to Containers to Servers

We represent the problem of assigning services to containers and containers to servers as an instance of the minimum fragmentable items bin packing (MIN-FIBP) problem, as presented by Byholm and Porres (2017).

Bin packing with fragmentable items is a variant of the classic bin packing problem, which permits cutting items into smaller fragments. We wish to minimize the number of fragments for given sets of items and bins. Figure 6 shows an example problem.

This optimization problem comprises an assignment $\boldsymbol{X}$ of services $s \in \boldsymbol{\mathcal{S}}$ to servers $m \in \boldsymbol{\mathcal{M}}$. The objective (1a) is to minimize the number of fragments, i.e. integer entries χ_{sm} greater than zero. The first constraint (1b) is that each service $s \in \boldsymbol{\mathcal{S}}$ must be fully assigned to some servers $m \in \boldsymbol{\mathcal{M}}$, so the sizes of all containers of service s must add up to its quanta $\psi(s)$. The second constraint (1c), states that each server $m \in \boldsymbol{\mathcal{M}}$ has a capacity $\Psi(m)$ we cannot exceed. The third constraint (1d) restricts containers to multiples of unit-sized parts, so the size of each container must be a natural number:

$$\min_{\boldsymbol{X}} \quad \sum 1_{\boldsymbol{X}>0} \tag{1a}$$

$$\text{subject to} \quad \sum_{m \in \boldsymbol{\mathcal{M}}} \chi_{sm} = \psi(s),\ \forall s \in \boldsymbol{\mathcal{S}}, \tag{1b}$$

$$\sum_{s \in \boldsymbol{\mathcal{S}}} \chi_{sm} \leq \Psi(m),\ \forall m \in \boldsymbol{\mathcal{M}}, \tag{1c}$$

$$\chi_{sm} \in \mathbb{N},\ \forall (s, m) \in \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{M}}. \tag{1d}$$

The MIN-FIBP problem is a generalization of the classic bin packing problem which permits cutting items into smaller fragments. We solve the MIN-FIBP problem in $\mathcal{O}(|\boldsymbol{\mathcal{S}}|^2)$, using the method of Byholm and Porres (2017). This results in a guaranteed 5/4-approximation in the worst case.

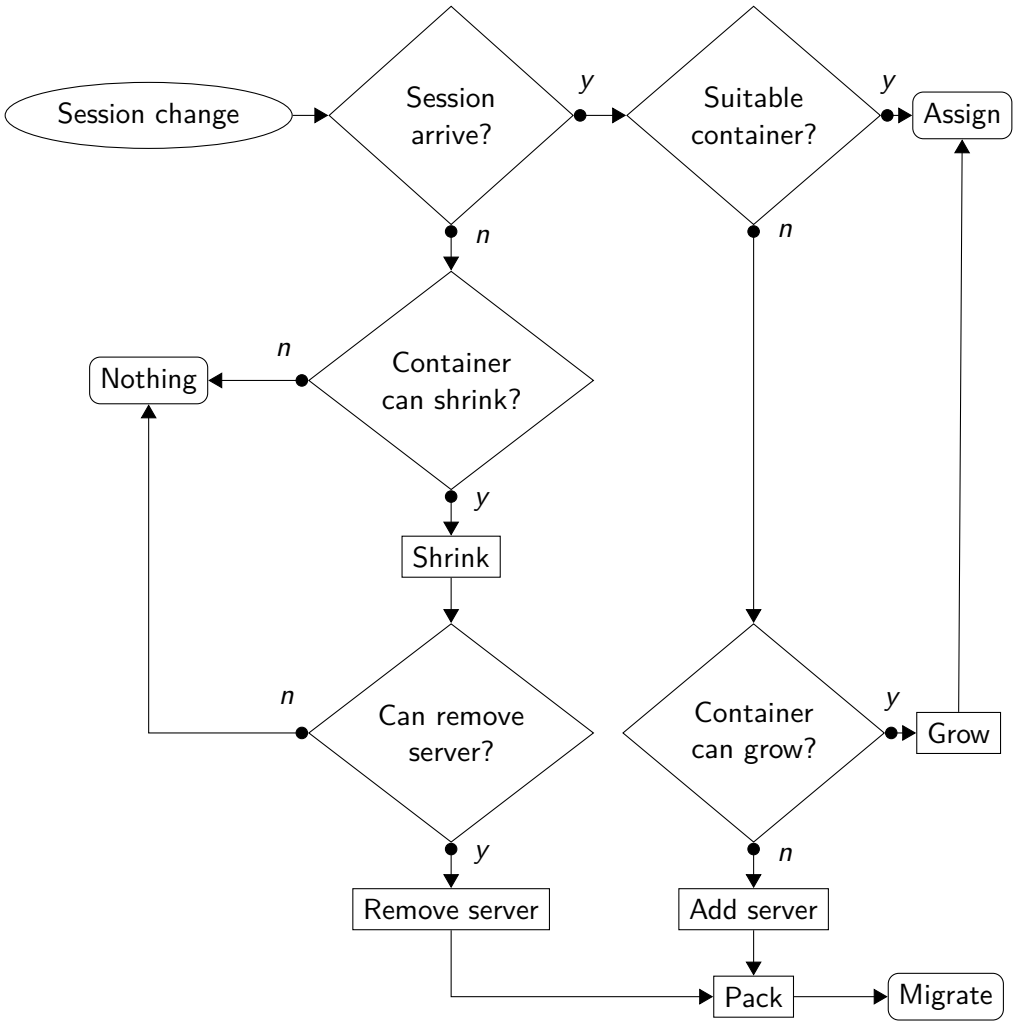

Figure 5: The primary algorithm performs horizontal and vertical scaling combined with consolidation in a manner which avoids unnecessary effort.

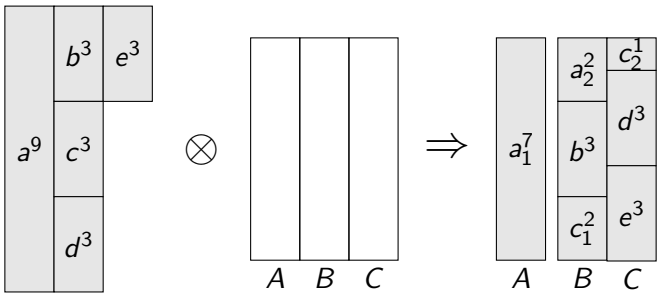

Figure 6: Fragmentable items bin packing. Five items are packed into three containers of size seven. In the process, two items are cut into smaller fragments, resulting in two blocks.

3.3 Migrating Sessions among Containers

We reduce the problem of migrating sessions among containers to a linear assignment problem. We solve the assignment problem as follows: Given an existing assignment $\boldsymbol{\mathcal{U}}$, a desired assignment $\boldsymbol{\mathcal{V}}$ of equal size and a cost matrix $\boldsymbol{\Phi}$, find an assignment $\boldsymbol{X} : \boldsymbol{\mathcal{U}} \to \boldsymbol{\mathcal{V}}$ with minimum cost.

To compute the cost of an assignment, we form the cost matrix $\boldsymbol{\Phi}$ as the amount of session state we must move to make $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ equivalent:

$$\boldsymbol{\Phi} = [\textstyle\sum\{f(z) \colon z \in \bigcup u \mathbin{\triangle} v\} \colon (u, v) \in \boldsymbol{\mathcal{U}} \times \boldsymbol{\mathcal{V}}], \tag{2}$$

where $u \mathbin{\triangle} v$ denotes the symmetric difference between sets u and v, while $f : \boldsymbol{\mathcal{D}} \to \mathbb{N}$ gives the size of session $d \in \boldsymbol{\mathcal{D}}$. We then solve the ensuing problem:

$$\begin{aligned}
\min_{\boldsymbol{X}} \quad & \frac{1}{2}\langle \boldsymbol{\Phi}, \boldsymbol{X} \rangle_{\mathrm{F}} && \text{(3a)} \\
\text{subject to} \quad & \sum_{u \in \boldsymbol{\mathcal{U}}} \chi_{uv} = 1, \ \forall v \in \boldsymbol{\mathcal{V}}, && \text{(3b)} \\
& \sum_{v \in \boldsymbol{\mathcal{V}}} \chi_{uv} = 1, \ \forall u \in \boldsymbol{\mathcal{U}}, && \text{(3c)} \\
& \chi_{uv} \in \mathbb{B}, \ \forall (u, v) \in \boldsymbol{\mathcal{U}} \times \boldsymbol{\mathcal{V}}, && \text{(3d)}
\end{aligned}$$

where $\langle \boldsymbol{\Phi}, \boldsymbol{X} \rangle_{\mathrm{F}}$ denotes the Frobenius inner product of matrices $\boldsymbol{\Phi}$ and $\boldsymbol{X}$.

According to Burkard et al. (2012), LAPMOD by Volgenant (1996) is the best algorithm for this purpose. Hence, we use it to solve the problem.

3.4 Example

Figure 7 illustrates an example problem. We want to find an optimal configuration for four different stateful services: a, b, c and d. Each service requires central processing unit (CPU) and random access memory (RAM) proportional to the number of sessions and CPU is the bottleneck resource.

Service a requires 700 MIPS, service b requires 100 MIPS, while services c, d and e require 200 MIPS each. In the current configuration, service a is deployed in server A, services b and d are deployed in server B, while services c and e reside in server C. Each server has a capacity of 700 MIPS.

Solving the MIN-FIBP problem provides an assignment of containers to servers, but containers of a given size are still interchangeable. In our example, one server will have a container requiring 700 MIPS, while another server will have three containers requiring 200 MIPS each and one requiring 100 MIPS. The third server is unused and can therefore shut down.

We now wish to minimize the amount of data migration required to realize the new assignment. In this example, we use a simple migration cost model based on the memory requirements of each service. The output of

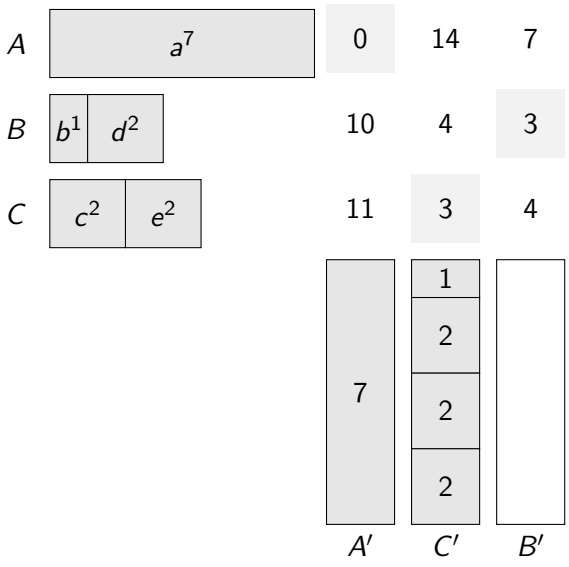

Figure 7: An assignment problem between the current and desired servers.

the LAPMOD algorithm tells us to place service a in server A and the other services in server C, since this minimizes the migration cost. Doing so leaves server B unused, so it can shut down.

4 Related Work

The ROADEF/EURO 2012 challenge, presented by Murat Afsar et al. (2016), was a competition organized by the French operational research and decision aid (ROADEF) and European operational research (EURO) societies. Google proposed the problem of machine reassignment for the competition. It is similar to our problem of dynamic horizontal and vertical scaling with consolidation, but with more restrictions which we have not addressed. According to Murat Afsar et al. (2016), the approaches used by qualifying teams were: lower bound methods, local search, large neighborhood search, hyper-heuristics, greedy randomized adaptive search procedures, tabu search, simulated annealing, variable neighborhood search, late acceptance heuristics, path-relinking, constraint programming, (mixed-integer) linear programming, dynamic programming and parallel search. We use a grouping genetic algorithm, which is not mentioned in the list.

Bin packing is an intuitive model not only for the machine reassignment problem, but also for related problems, such as ours. Vector bin packing is a generalization of bin packing for multiple dimensions. In the ROADEF/EURO 2012 challenge, Gabay and Zaourar (2016) introduced a further generalization of the vector bin packing problem where the bins have variable sizes. We use an alternative generalization of the classic bin packing problem called minimum

fragmentable items bin packing with equal capacities (MIN-FIBP-EQ), where items may be cut into smaller fragments. This naturally allows for denser packings. As noted by Gabay and Zaourar (2016), problems with heterogeneous bins can be reduced to problems with uniform bin sizes. Similarly, instances of MIN-FIBP can be reduced to equivalent instances of MIN-FIBP-EQ. However, Gabay and Zaourar (2016) make a fair assessment that this may lead to complications when dealing with multiple resources.

Bin packing and other optimization algorithms can be classified as offline or online. Offline algorithms take the entire problem into account, while strict online algorithms only account for one item at a time. Song et al. (2014) presented an online algorithm for adaptive resource provisioning in the cloud and argued that offline algorithms have a drawback in that they "may incur a large number of VM movements when the load of VMs changes dynamically because an offline algorithm by its nature does not consider the existing VM layout when packing the VMs into the set of PMs". We do not consider this a fair assessment of the offline approach. An offline algorithm can and should account for the existing layout and try to minimize migration. Our solution is an offline algorithm which does precisely that. Moreover, as noted by Song et al. (2014), offline algorithms can generally achieve an approximation ratio far better than online algorithms.

Vertical scaling seems to be an often neglected aspect of resource management in the cloud. Many works focus solely on horizontal scaling. Turowski and Lenk (2015) studied the vertical scaling capability of OpenStack and performed a comparison of several guest operating systems and hypervisors. It appears that Ubuntu under Xen supports vertical scaling in all relevant aspects. Further extending the vertical scaling capabilities of popular systems should not be a major obstacle.

Sandpiper, developed by Wood et al. (2009), is a resource management approach which does account for vertical scaling, in addition to horizontal scaling and consolidation. We use similar techniques for monitoring, profiling and other implementation details of the larger system. Like Sandpiper, our system identifies hotspots and takes appropriate action. However, Sandpiper seems to use a simple greedy algorithm for its migration phase, while we solve a fragmentable bin packing problem giving a good deployment with minimum servers, followed by an assignment problem to minimize migration.

Wolke et al. (2015) compared VM allocation strategies for cloud environments and found that aggressive VM placement combined with reallocation achieved the highest efficiency. None of the studied approaches used fragmentable bin packing. Our approach supports fully dynamic allocation using fewer servers than any comparable model based on classic bin packing.

Beloglazov et al. (2012) used a variable size bin packing model to manage data centers in an energy-aware and efficient manner. We use a fragmentable bin packing model, which requires fewer servers, by definition.

Ashraf et al. (2016) presented a cloud system involving prediction-based VM provisioning and admission control for multi-tier web applications. The system supports horizontal scaling, but, unlike our approach, does not incorporate vertical scaling or consolidation.

Ant colony systems are another family of popular metaheuristics not mentioned in the report on the ROADEF/EURO 2012 challenge. Farahnakian et al. (2015) used an ant colony system to consolidate VMs for green cloud computing. In addition to consolidation, our approach also deals with vertical and horizontal scaling while accounting for both monetary and energy costs.

Ashraf et al. (2015) used an ant colony system for the multiobjective cloud-based software component deployment problem and compared it with a simple, Holland-style genetic algorithm: NSGA-II. This related problem provides an initial deployment of component-based software. Our problem instead deals with iterated redeployment according to variable demand.

Ashraf and Porres (2017) used an ant colony system for multi-objective, dynamic VM consolidation in the cloud. In addition to consolidation, our approach also performs horizontal and vertical scaling. Moreover, we also operate under soft real-time constraints, which is important due to the high volatility in Internet-scale cloud systems.

5 Conclusion and Future Work

We presented a new take on the problem of dynamic horizontal and vertical scaling of multiple cloud services in soft real-time. We recognized that the problem can be reduced to a MIN-FIBP problem combined with a linear assignment problem. This is the first work based on fragmentable items bin packing (FIBP), allowing services to be cut into smaller instances which allows for denser packings with less wasted capacity.

Future work entails a practical generalization of FIBP to multiple dimensions, allowing us to efficiently handle multiple bottleneck resources. Federated clouds pose a different challenge, since they change the nature of the underlying assignment problem: If the cost of migration also depends on where a container moves, in addition to its contents, a linear assignment problem is no longer sufficient. It would require solving a generalized assignment problem, which is $\mathcal{APX}$-hard according to Chekuri and Khanna (2005).

Acknowledgments

Benjamin Byholm received scholarships from the Nokia Foundation and the Finnish Foundation for Technology Promotion (TES). This work was part of the DIMECC Need for Speed program `http://n4s.fi/`.

References

Adnan Ashraf and Ivan Porres. Multi-Objective Dynamic Virtual Machine Consolidation in the Cloud Using Ant Colony System. *International Journal of Parallel, Emergent and Distributed Systems*, pages 1–18, 2017. ISSN 1744-5779.

Adnan Ashraf, Benjamin Byholm, and Ivan Porres. A Multi-Objective ACS Algorithm to Optimize Cost, Performance, and Reliability in the Cloud. In Omer Rana and Manish Parashar, editors, *8th IEEE/ACM International Conference on Utility and Cloud Computing*, pages 341–347. IEEE, 2015.

Adnan Ashraf, Benjamin Byholm, and Ivan Porres. Prediction-Based VM Provisioning and Admission Control for Multi-Tier Web Applications. *Journal of Cloud Computing*, 5(1):15, 2016. ISSN 2192-113X. doi: 10.1186/s13677-016-0065-9.

Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012. ISSN 0167-739X. doi: 10.1016/j.future.2011.04.017.

Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems: Revised Reprint*. SIAM, 2012. ISBN 978-1611972221.

Benjamin Byholm and Ivan Porres. Fast Algorithms for Fragmentable Items Bin Packing. Technical Report 1181, TUCS, 2017. ISBN 978-952-12-3560-3.

Chandra Chekuri and Sanjeev Khanna. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. *SIAM Journal on Computing*, 35(3):713–728, 2005. doi: 10.1137/S0097539700382820.

Fahimeh Farahnakian, Adnan Ashraf, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Ivan Porres, and Hannu Tenhunen. Using Ant Colony System to Consolidate VMs for Green Cloud Computing. *IEEE Transactions on Services Computing*, 8(2):187–198, March 2015. ISSN 1939-1374. doi: 10.1109/TSC.2014.2382555.

Michaël Gabay and Sofia Zaourar. Vector bin packing with heterogeneous bins: application to the machine reassignment problem. *Annals of Operations Research*, 242(1):161–194, 2016. ISSN 1572-9338. doi: 10.1007/s10479-015-1973-7.

Hasan Murat Afsar, Christian Artigues, Eric Bourreau, and Safia Kedad-Sidhoum. Machine reassignment problem: the ROADEF/EURO challenge 2012. *Annals of Operations Research*, 242(1):1–17, 2016. ISSN 1572-9338. doi: 10.1007/s10479-016-2203-7.

Bianca Schroeder, Adam Wierman, and Mor Harchol-Balter. Open Versus Closed: A Cautionary Tale. In *3rd Symposium on Networked Systems Design and Implementation (NSDI 2006)*, pages 239–252, Berkeley, CA, USA, 2006. USENIX.

Weijia Song, Zhen Xiao, Qi Chen, and Haipeng Luo. Adaptive Resource Provisioning for the Cloud Using Online Bin Packing. *IEEE Transactions on Computers*, 63(11):2647–2660, Nov 2014. ISSN 0018-9340. doi: 10.1109/TC.2013.148.

Marian Turowski and Alexander Lenk. *Vertical Scaling Capability of OpenStack*, pages 351–362. Springer International Publishing, Cham, Switzerland, 2015. ISBN 978-3-319-22885-3. doi: 10.1007/978-3-319-22885-3_30.

Anton Volgenant. Linear and semi-assignment problems: A core oriented approach. *Computers & Operations Research*, 23(10):917–932, 1996. ISSN 0305-0548. doi: 10.1016/0305-0548(96)00010-X.

Andreas Wolke, Boldbaatar Tsend-Ayush, Carl Pfeiffer, and Martin Bichler. More than bin packing: Dynamic resource allocation strategies in cloud data centers. *Information Systems*, 52:83–95, 2015. ISSN 0306-4379. doi: 10.1016/j.is.2015.03.003.

Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53(17):2923–2938, 2009. ISSN 1389-1286. doi: 10.1016/j.comnet.2009.04.014.

Turku
Centre *for*
Computer
Science

Vesilinnantie 3, 20500 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Computer Science
- Computer Engineering

ISBN 978-952-12-3560-3
ISSN 1239-1891