Chapter 2
Allocation of Virtual Machines

In this chapter, we introduce a solution to the problem of cost-effective VM allocation and reallocation. Unlike traditional solutions, which typically reallocate VMs based on a greedy algorithm such as First Fit (each VM is allocated to the first server in which it will fit), Best Fit (each VM is allocated to the active server with the least residual capacity), or Worse Fit (each VM is allocated to the active server with the most residual capacity), the proposed solution strikes a balance between the effectiveness of packing VMs into few servers and the overhead incurred by VM reallocation.

2.1 Problem Formulation

We consider the case where a system (e.g., a cloud service or a cloud data center) is allocated with a number of servers denoted by H and a number of VMs denoted by V. We assume the number of servers is always sufficient to host the total resource requirement of all VMs in the system. Thus, we focus on the consolidation effectiveness and the migration cost incurred by the server consolidation problem.

Further, we assume that VM migration is performed at discrete times. We define the period of time to perform server consolidation as an epoch. Let $T = \{t_1, t_2, \ldots, t_k\}$ denote the set of epochs to perform server consolidation. The placement sequence for VMs in V in each epoch t is then denoted by $F = \{f_t \mid \forall\ t \in T\}$, where f_t is the VM placement at epoch t and defined as a mapping $f_t : V \rightarrow H$, which specifies that each VM i, $i \in V$, is allocated to server $f_t(i)$. Note that "$f_t(i) = 0$" denotes that VM i is not allocated. To model the dynamic nature of the resource requirement and the migration cost for each VM over time, we let $R_t = \{r_t(i) \mid \forall\ i \in V\}$ and $C_t = \{c_t(i) \mid \forall\ i \in V\}$ denote the sets of the resource requirement and migration cost, respectively, for all VMs in epoch t.

© The Author(s) 2016

L. Tsai and W. Liao, *Virtualized Cloud Data Center Networks: Issues in Resource Management*, SpringerBriefs in Electrical and Computer Engineering, DOI 10.1007/978-3-319-32632-0_2

The capacity of a server is normalized (and simplified) to one, which may correspond to the total resource in terms of CPU, memory, etc. in the server. The resource requirement of each VM varies from 0 to 1. When a VM demands zero resource, it indicates that the VM is temporarily out of the system. Since each server has limited resources, the aggregate resource requirement of VMs on a server must be less than or equal to one. Each server may host multiple VMs with different resource requirements, and each application or service may be distributed to multiple VMs hosted by different servers. A server with zero resource requirements from VMs will not be used to save the hosting cost. We refer to a server that has been allocated VMs as an *active* server.

To jointly consider the consolidation effectiveness and the migration overhead for server consolidation, we define the total cost for VM placement F as the total hosting cost of all active servers plus the total migration cost incurred by VMs. The hosting cost of an active server is simply denoted by a constant E and the total hosting cost for VM placement sequence F is linearly proportional to the number of active servers. To account for revenue loss, we model the downtime caused by migrating a VM as the migration cost for the VM. The downtime could be estimated as in [1] and the revenue loss depends on the contracted service level. Since the downtime is mainly affected by the memory dirty rate (i.e., the rate at which memory pages in the VM are modified) of VM and the network bandwidth [1], the migration cost is considered independent of the resource requirement for each VM.

Let H'_t be a subset of H which is active in epoch t and $|H'_t|$ be the number of servers in H'_t. Let C'_t be the migration cost to consolidate H'_t from epoch t to $t + 1$. H'_t and C'_t are defined as follows, respectively:

$$H'_t = \{f_t(i) | f_t(i) \neq 0, \forall i \in V\}, \forall t \in T$$

$$C'_t = \sum_{i \in V, f_t(i) \neq f_{t+1}(i)} c_t(i), \forall t \in T \backslash \{t_k\}$$

The total cost of F can be expressed as follows:

$$Cost(F) = E \times \sum_{t \in T} |H'_t| + \sum_{t \in T \backslash \{t_k\}} C'_t$$

We study the Total-Cost-Aware Consolidation (TCC) problem. Given $\{H, V, R, C, T, E\}$, a VM placement sequence F is feasible only if the resource constraints for all epochs in T are satisfied. The TCC problem is stated as follows: among all possible feasible VM placements, to find a feasible placement sequence F whose total cost is minimized.

2.2 *Adaptive Fit* Algorithm

The TCC problem is NP-Hard, because it is at least as hard as the server consolidation problem. In this section, we present a polynomial-time solution to the problem. The design objective is to generate VM placement sequences F in polynomial time and minimize $Cost(F)$.

Recall that the migration cost results from changing the hosting servers of VMs during the VM migration process. To reduce the total migration cost for all VMs, we attempt to minimize the number of migrations without degrading the effectiveness of consolidation. To achieve this, we try to allocate each VM i in epoch t to the same server hosting the VM in epoch $t-1$, i.e., $f_t(i) = f_{t-1}(i)$. If $f_{t-1}(i)$ does not have enough capacity in epoch t to satisfy the resource requirement for VM i or is currently not active, we then start the remaining procedure based on "saturation degree" estimation. The rationale behind this is described as follows.

Instead of using a greedy method as in existing works, which typically allocate each migrating VM to an active server with available capacity either based on First Fit, Best Fit, or Worse Fit, we define a total cost metric called saturation degree to strike a balance between the two conflicting factors: consolidation effectiveness and migration overhead. For each iteration of allocation process in epoch t, the saturation degree X_t is defined as follows:

$$X_t = \frac{\sum_{i \in V} r_t(i)}{\left(\left|H'_t\right| + 1\right) \times 1}$$

Since the server capacity is normalized to one in this book, the denominator indicates the total capacity summed over all active servers plus an idle server in epoch t.

During the allocation process, X_t decreases as $|H'_t|$ increases by definition. We define the saturation threshold $u \in [0, 1]$ and say that X_t is low when $X_t \leq u$. If X_t is low, the migrating VMs should be allocated to the set of active servers unless there are no active servers that have sufficient capacity to host them. On the other hand, if X_t is large (i.e., $X_t > u$), the mechanism tends to "lower" the total migration cost as follows. One of the idle servers will be turned on to host a VM which cannot be allocated on its "last hosting server" (i.e., $f_{t-1}(i)$ for VM i), even though some of the active servers still have sufficient residual resource to host the VM. It is expected that the active servers with residual resource in epoch t are likely to be used for hosting other VMs which were hosted by them in epoch $t-1$. As such, the total migration cost is minimized.

The process of allocating all VMs in epoch t is then described as follows. In addition, the details of the mechanism are shown in the Appendix.

1. Sort all VMs in V by decreasing order based on their $r_t(i)$.
2. Select VM i with the highest resource requirement among all VMs not yet allocated, i.e.,

$$\mathrm{i} \leftarrow \arg \max_j \{r_t(j) | f_t(j) = 0, \forall j \in V\}$$

3. Allocate VM i:

   (i) If VM i's hosting server at epoch $t-1$, i.e., $f_{t-1}(i)$, is currently active and has sufficient capacity to host VM i with the requirement $r_t(i)$, VM i is allocated to it, i.e., $f_t(i) \leftarrow f_{t-1}(i)$;

   (ii) If VM i's last hosting server $f_{t-1}(i)$ is idle, and there are no active servers which have sufficient residual resource for allocating VM i or the X_t exceeds the saturation threshold u, then VM i is allocated to its last hosting server, namely, $f_t(i) \leftarrow f_{t-1}(i)$;

   (iii) If Cases i and ii do not hold, and X_t exceeds the saturation threshold u or there are no active servers which have sufficient residual resource to host VM i, VM i will be allocated to an idle server;

   (iv) If Cases i, ii and iii do not hold, VM i is allocated to an active server based on Worse-Fit policy.

4. Update the residual capacity of $f_t(i)$ and repeat the procedure for allocating the next VM until all VMs are allocated.

We now illustrate the operation of *Adaptive Fit* with an example where the system is allocating ten VMs, of which the resource requirements are shown in Table 2.1.

The saturation threshold u is set to one. The step-by-step allocation for epoch t is shown in Table 2.2. The row of epoch $t-1$ indicates the last hosting servers (i.e., $f_{t-1}(i)$) of VMs. The rows for epoch t depict the allocation iterations, with allocation sequence from top to bottom. For each VM, the items underlined denote the actual allocated server while the other items denote the candidate servers with sufficient capacity. The indicators L, X, N, A denote that the allocation decision is based on the policies: (1) Use the last hosting server first; (2) Create new server at high saturation; (3) Create new server because that there is no sufficient capacity in active serves; (4) Use active server by Worse Fit allocation, respectively. Note that the total resource requirement of all VMs is 3.06 and the optimal number of servers to use is 4 in this example. In this example, *Adaptive Fit* can achieve a performance quite close to the optimal.

Table 2.1 Resource requirements for VMs

v_i	$r_t(i)$	v_i	$r_t(i)$
4	0.49	10	0.34
8	0.48	1	0.15
3	0.47	7	0.15
2	0.43	6	0.13
5	0.35	9	0.07

Table 2.2 An example of allocating VMs by *Adaptive Fit*

Epoch	Server 1	Server 2	Server 3	Server 4
$t-1$	v_1, v_9	v_6, v_8	v_5, v_{10}	v_2, v_3
				v_4, v_7
t				$\underline{v_4}$ $\underline{(L)}$
		$\underline{v_8}$ $\underline{(L)}$		v_8
		v_3		$\underline{v_3}$ $\underline{(L)}$
	$\underline{v_2}$ $\underline{(X)}$	v_2		
	$\underline{v_5}$ $\underline{(A)}$	v_5		
		$\underline{v_{10}}$ $\underline{(A)}$		
	$\underline{v_1}$ $\underline{(L)}$	v_1		
		$\underline{v_7}$ $\underline{(A)}$		
			$\underline{v_6}$ $\underline{(N)}$	
	$\underline{v_9}$ $\underline{(L)}$		v_9	

2.3 Time Complexity of *Adaptive Fit*

Theorem 2.1 (Time complexity of Adaptive Fit) *Adaptive Fit is a polynomial-time algorithm with average-case complexity $O(n \log n)$, where n is the number of VMs in the system.*

Proof We examine the time complexity of each part in *Adaptive Fit*. Let m denote the number of active servers in the system. The initial phase requires $O(m \log m)$ and $O(n \log n)$ to initialize A and A' and V', which are implemented as binary search trees. The operations on A and A' can be done in $O(\log m)$. The saturation degree estimation takes $O(1)$ for calculating the denominator based on the counter for the number of servers used while the numerator is static and calculated once per epoch. The rest of the lines in the "for" loop are $O(1)$. Therefore, the main allocation "for" loop can be done in $O(n \log m)$. All together, the *Adaptive Fit* can be done in $O(n \log n + n \log m)$, which is equivalent to $O(n \log n)$. □

Reference

1. S. Akoush et al., in *Proc. IEEE MASCOTS*, Predicting the Performance of Virtual Machine Migration. pp. 37–46 (2010)

http://www.springer.com/978-3-319-32630-6

Virtualized Cloud Data Center Networks: Issues in Resource Management.

Tsai, L.; Liao, W.

2016, VIII, 57 p. 21 illus., Softcover
ISBN: 978-3-319-32630-6