IAP Term Project
Application Aware Placement of Virtual Machines

Ken Kumar 13CS30044
Siva Kesava Reddy 13CS10048
M Siddharth 12EC35028
Mentor: Rohit Dhangar

April 2017

1 Problem Definition

Virtual machine (VM) placement is the process of selecting the most suitable server in large cloud data centers to deploy newly-created VMs. In most scenarios, when end users send their VM requests, some physical server(s) in the cloud data center will be selected to deploy the required VMs. Specifically, the VM requests from different users are diverse and require taking multiple resource constraints into account. For instance, commercial cloud IaaS providers such as Amazon EC2, IBM and Google Compute Engine offer various VM instance types with different types and amounts of resources.

As a consequence, if the owners of cloud data centers cannot effectively deploy different types of VM requests, some resources may become overloaded while the others remain underutilized. Eventually, such unbalanced use of resources may result in the unnecessary activation of physical servers. To this regard, choosing the most appropriate target machine in a large pool of physical servers to create VMs provides a strong motivation for cloud providers to maximize their operational efficiency. Hence, a major concern is to balance the load in terms of not only CPU capacity but also memory, storage and network bandwidth while satisfying all VM requests.

During placement, hosts are rated based on the virtual machine's hardware and resource requirements and the anticipated usage of resources. Host ratings also take into consideration the placement goal: either resource maximization on individual hosts or load balancing among hosts.

In our case, the target is to balance the load among the servers.The administrator then selects a host for the virtual machine based on the host ratings. Following section describes our solution.

2 Solution approach

While deploying a service, we need to consider various factors like what is the expected bandwidth it will be consuming, how many disk reads/writes per unit time it will be performing, what are the expected number of vcpu's it is going to use and so on. Based on our experiments, we fixed the following six parameters which will be taken into consideration while deploying a particular service. These are-

1. Number of disk writes per unit time
2. Memory consumption
3. Network Bandwidth consumption
4. Number of disk reads per unit time
5. Number of vcpu's it is going to need

The intuition behind choosing these parameters was that we can broadly classify our services as being network intensive, cpu intensive, memory intensive, whether it performs more read/writes based on them and thus enable us to make some informed choices while deploying our services.In the next section we present our deployment algorithm in which we take into account the above parameters and then decide on which server to deploy our services. In this section, we briefly describe the main ideology behind our algorithm.

One of our major challenge while formulating our algorithm was to ensure that many similar services do not get deployed on the same server as it could lead to severe performance degradation and adversely affect the quality of service. We had to ensure that we deploy *complimenting* services together which will not only prevent under utilization of the resources allotted to the servers but also maintain the QOS. In our initial approach, we focused more on the service and tried to formulate an optimization function that will help to find the *best* server for deploying our services. The main drawback with our initial approach was that we were greedily selecting the server that best suited our requirements without giving any thought to the future services that we would have to deploy. **Philosophically we were just being bothered about the present and taking care of it without giving much thought about the future needs.** Keeping this in mind, we shifted our focus from the service side to the server side. So, then we tried formulating an optimization function(the final version is presented in the next section), which will try to co-locate *complimenting* services together and at the same time ensure that we have resources available at the servers for future service deployment.

First, we deploy our service on a test server for a fixed time to get a fair idea about the resource consumption pattern of our service, i.e. how much expected bandwidth it is going to consume, how many read/writes to disk per unit time it does and so on. We will also be able to conclude about the type of our service, i.e. whether it is a network intensive service or cpu intensive. Depending on the type of the service, we can give a score to all the above parameters for this service which will be an indication to the type of service and the normalized values of these scores will also act as weights in our algorithm to give them appropriate priorities. And for all servers, we can easily get the information about the resource availability at the servers, like how much bandwidth is available, how many disk reads/write all the services deployed on that server is performing and so on. With these available data, our proposed function computes how much resources we can expect to be available after the service has been deployed on that particular server. We compute this for all the servers and choose the best k servers for which the the resources available after deployment is maximum. This will ensure that we can not overburdening any server and at the same time, since we are taking into consideration the available resources at the servers before and after the service deployment, we are making sure that the future services get there fair share of resources allocated to them.

3 Your Findings

For each service that want to be deployed will have a 5 tuple $(b_1, b_2, b_3, b_4, b_5)$ values:
b_1: How many disk writes the service is performing per unit time on the test server.
b_2: How much primary memory the service is consuming in the test server.
b_3: How much network bandwidth the service is using.
b_4: How many disk reads the service is performing per unit time on the test server.
b_5: How many vcpu's the service has been allocated in the test server.

Similarly each server is represented by a 5 tuple $(S_1^j, S_2^j, S_3^j, S_4^j, S_5^j)$ and there are n servers from $j = 1$ to n:
S_1^j: How many disk writes the server j is performing per unit time.
S_2^j: How much primary memory is left with the server j.
S_3^j: How much network bandwidth is left with the server j.
S_4^j: How many disk reads the server j is performing per unit time.
S_5^j: How many vcpu's are left with the server j.

After the service is deployed initially at a test server we get a 5 tuple of weights $(W_1, W_2, W_3, W_4, W_5)$ and each W_i is from 0 to 10. These weights are generated based on the relative use of the resources by the service.

In the Weight Function Calculation Algorithm described below we integrate all the above values.

Input: *service*: A tuple with 5 values $(b_1, b_2, b_3, b_4, b_5)$
Servers: The set of all servers $S^1, ...S^n$ with each server represented by a 5 tuple $(S_1^j, S_2^j, S_3^j, S_4^j, S_5^j)$
Weights: A tuple with 5 values $(W_1, W_2, W_3, W_4, W_5)$

Variables: MDW is Maximum Disk Writes among all the servers
MDR is Maximum Disk Reads among all the servers
F is an Array that will store the weighted sum.

Output: F:The weighted sum values

Algorithm 1 Weight Function Calculation

```
1: function DEPLOYMENT($service, Servers, Weights$)
2:     $n \leftarrow$ numberof($Servers$)
3:     $MDW \leftarrow max(S_1^j)$
4:     $MDR \leftarrow max(S_4^j)$
5:     $F \leftarrow [F_1, F_2, ....F_n]$
6:     for $i \leftarrow 1$ to $n$ do
7:         $F_i \leftarrow [\frac{W_1(MDW-(S_1^i+b_1))}{MDW} + \frac{W_2(S_2^i-b_2)}{S_2^i} + \frac{W_3(S_3^i-b_3))}{S_3^i} + \frac{W_4(MDR-(S_4^i+b_4))}{MDR} + \frac{W_5(S_5^i-b_5))}{S_5^i}]$
8:     end for
9:     return $F$
10: end function
```

Now as described earlier we will choose first k servers based on these F values in the decreasing order. We can use a post processing heuristic to select a server from these k servers on to which the service will be deployed.

References

[1] Rohit Dhangar Sandip Chakraborty Dhruv Jain, Sandesh C. *Predictable bandwidth allocation for Virtualized Data Center.* Nov 2016.

[2] Rolf Stadler Brendan Jennings. *Resource Management in Clouds: Survey and Research.* Springer Science+Business Media New York, Feb 2014.

[3] Myungjin Lee Lucian Popa Sujata Banerjee Joon-Myung Kang Puneet Sharma Jeongkeun Lee, Yoshio Turner. *Application-Driven Bandwidth Guarantees in Datacenters.* ACM, 2014.

[4] Kirk Beaty Norman Bobroff, Andrzej Kochut. *Dynamic Placement of Virtual Machines for Managing SLA Violations.* IEEE, 2007.