Seminar Report
on
Techniques for Virtual Machine Placement
and
Mirgration in Data Centers

Dhaval Manvar
Roll No : 10305021

Under Guidance of
Prof. Anirudha Sahoo
Computer Science and Engineering Department
Indian Institute of Technology Bombay

April 3, 2011

Abstract

1 Indroduction

2 Literature Survey

In this section, we will go through some of the useful techniques for VM sizing (From section 2.1 to 2.5) and VM palcement on PM (From section 2.1 to 2.5) in the Data Center.

2.1 Profiling and Modeling Resource Usage of Virtualized Applications

At the time of server placement into virtualized environment, we should have the initial resource requirements of the server. In virtualized environment, Virtual Machine Monitors(VMM) provides the isolation among VMs on a shared hardware platform. In Xen and VMware ESX server, VMM(hypervisor) or Driver Domain(Dom0 in Xen) contains device driver so that all I/O activities are taken care by hypervisor as shown in $Figure1$.

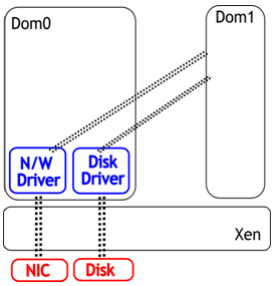

Figure 1: Guest-Domain I/O activity

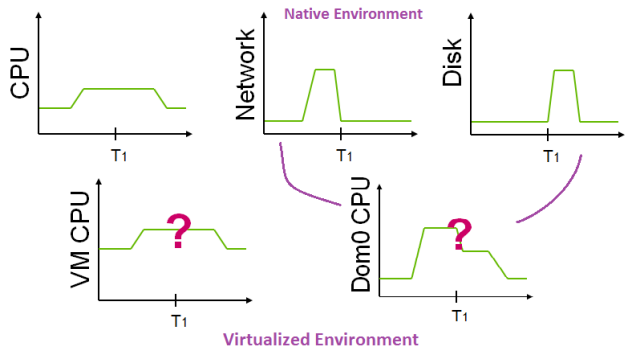

Figure 2: Mapping of Resources Usage in Native to CPU usage in Virtual Environment

This implies that hypervisor uses CPU whenever any VM does I/O activity. So we have to consider all these virtualized overhead in server placement on VMM. Given resource utilization traces of the server running on native environment, this technique estimates what its resource(CPU) requirements would be if the server were transitioned to a virtual environment(Xen) on the same hardware platfrom as shown in $Figure1$.
The given technique[2] estimates the CPU requirements of the following two components as shown in $Figure1$:

- VM (User/Guest Domain)
- Dom-0 which perform I/O processing on behalf of the VM.

Virtualization layer does not add overhead on other metrics (Disk and Network request rates) since it will not happen on virtualized environment that server will generate *extra* network packets or disk request. The overall approach is to develop a model which maps the resource requirements of server in native to virtual platform. This model can be developed by running the series of same microbenchmarks on both the platforms with same hardware configuration and trace the statistics of resouce utilizations of the server. This technique is logically divided into two components :

- Plateform Profiling
- Model Generation

There are two criteria to select Microbenchmark.

- Microbenchmark must be able to apply a range of workload intensities to measure the virtualization overhead due to different workload (Network I/O, Disk I/O etc.).
- Microbenchmark should run nearly-identical in both native and virtual environments to minize the error in model generation.

In platform profiling phase, it gatheres the statistics of following 11 different metrics :

CPU	**Network**	**Disk**
User Space %	Rx packetssec	Read reqsec
Kernel %	Tx packetssec	Write reqsec
IO wait %	Rx bytessec	Read blockssec
	Tx bytessec	Write blockssec

Using the gathered data of different metrics, the given technique generates model by following three components :

- A robust linear regression algorithm that is used to reduce the impact of outliers.
- A stepwise regression approach that is used to include only the most statistically significant metrics in the final model
- A model refinement algorithm to build accurate model by eliminating or rerunning erroneous bechmarks.

To develop a model two sets of equations, one for Dom-0 CPU utilization and one for Guest Domain CPU utilization, to be solved as a linear combination of the different 11 metrics :

$$U^1_{dom0} = c_0 + c_1 * M^1_1 + c_2 * M^1_2 + ... + c_{11} * M^1_{11}$$
$$U^2_{dom0} = c_0 + c_1 * M^2_1 + c_2 * M^2_2 + ... + c_{11} * M^2_{11}$$

where

- M^i_j is a value metric M_i collected during the time interval j for a benchmark executed in the native environment.
- U^j_{dom0} is a measured CPU utilization for a benchmark executed in virtualized environment with the corresponding time interval j.

The approximated utilization can be calculated as :

$$\hat{U}^j_{dom0} = c_0^{dom0} + \sum_{i=1}^{11} M^j_i . c_i^{dom0}$$

where
$c_0^{dom0}, c_1^{dom0}, ..., c_1^{dom0}1$ denote the approximated solution for the first equation set. Least Squares Regression method is used to solve this solution. The same approach is used to model the CPU utilization of Guest Domain.

2.2 Dynamic Placement of Virtual Machines for Managing SLA Violations

The paper[1] has described algorimth for prediction of resource requirement of the Virtual Machine(VM) based on the recent history of resource demands. **The main goal** of the algorithm is to get gain by dynamically allocating minimum resources to VM with respect to static allocation such that the overall SLA violations should not be more than p-percentile. The overall **approach of the paper** is to predict future resource requirements based on recent time series resource demand distribution of the VM. Based on the future resource demand, algorithm remap the VM to PM, to consolidate VMs to reduce the number of PMs or to reduce the load of VMs, such that target PM's capacity should not less than VM's requirement. By live migration, VM will be placed on the corresponding PM. Algorithm selects VMs for remapping by calculating Gain that it would get by dynamic resource management.

A VM can be candidate for dynamic resource management if

- the workload of that VM vary with time
- the workload within a particular time duration should have high autocorrelation.

The first condition indicates the requirement of dynamic resource management. Without variablity in workload(resource demand), there is no meaning of dynamic resource provisioning. The second condition is about the prediction error. If the workload remains approximatelly same in predictable time interval, there will be less prediction error. The algorithm finds out the VMs for dynamic resource management which are satisfing the above two conditions. Now algorithm calculate the gain that it would get by dynamic resource management of the selected VMs by using following equation.

T : Reallocation interval (Time interval between two remapping of VM to PM)
$u(x)$: the demand probability density
L_p : p-percentile of distribution u (Static resource allocation w.r.t. p-percentile SLA violations)
$u_T(x)$: distribution of predicted time series
$E_p(T)$: p-percentile of distribution of prediction error
$G(T)$: The ratio of the time-averaged dynamically adusted capicity to the statically allocated capacity with the same target overflow percentile p

$$Gain, G(T) = \frac{\int_{\infty}^{0} [L_p - (x + E_p(T))] * u_T(x)\, dx}{L_p} \tag{1}$$

Based on the Gain $G(T)$, the algorithm decides how much a given VM can gain from dynamic management and categorize VM as predictable or unpredictable.
Algorithm forcasts future resource demand of VM based on CPU demand time series which contains one or more periodic components. It then divids the demand time series into a sum of periodic components. Once we find out the candidate VMs for dynamic management and their future resource demand(for next time interval T), then Management Algorithm tries to remap each candidate VM to PM such that migrating VM's resource requirement should be less than target PM's capacity.

Management algorithm first **sorts the candidate VMs in Descending order** of their future resource demand and then it attemps to place the first VM in the ordered list to PM as first fit bin packing problem. Algorithm ensures for each target PM that the p-percentile of the distribution of sum of resource demands of all VMs (i.e. the VMs those are already placed on this PM plus the selected candidate VM) allocated to this PM should not greater than PM's capacity. If the list of PMs is exhausted without satisfying this condition the VM is assigned to the PM that has the smallest difference between allocated demands and its capacity. The alorithm iterates for all candidate VMs.

3 Conclusion

4 References

References

[1] Kirk Beaty Norman Bobroff, Andrzej Kochut. Dynamic placement of virtual machines for managing sla violations. 2007.

[2] Kivanc Ozonat Prashant Shenoy Timothy Wood, Ludmila Cherkasova. Predicting application resource requirements in virtual environments. 2008.