aDICIOn: A PREDICTIVE AND AUTO SCALABLE MODEL FOR PRIVATE CLOUD

Anuja Hurgat[1], Sahil Parmar[2], Ankit Patil[3], Gaurav Shaha[4]

Computer Science, Pune Institute of Computer Technology,
Pune, Maharashtra, India

[1]anuja.1011@gmail.com

[2]sahilpparmar@gmail.com

[3]ankitpatil2010@gmail.com

[4]grvshaha@gmail.com

***Abstract*— One of the essential features of cloud computing is on-demand provision of resources. This would allow applications to scale-up or scale-down on the basis of dynamic workloads. We present a self-learning prototype that exploits adaptive allocation of memory to scale gracefully in the presence of rapid variations in workload. The memory usage of each virtual machine will be monitored from user space and a profile of each virtual machine is built. Past records of a virtual machine will help to manage allocation of memory thus reducing the decreased performance in case of under-allocated guest or leading to efficient management of memory in case of over-allocated guest. In order to accomplish this, a predictive model for provision of memory is used. Cluster analysis is performed to track the trends in memory usage of the virtual machine. Memory provisions are made as per the suggestions made by the model. Feedback about the current decision regarding the provisioning is given to the predicting system which will evaluate its own performance and accordingly change the provisioning schema. The feasibility of this approach has been experimentally evaluated. Thus, cloud computing with adaptive resource allocation has the potential to increase the usability, stability and performance of large-scale applications.**

Keywords: Self-learning, predictive, auto-scalability, cluster analysis.

I. INTRODUCTION

A. Scenario

KVM (Kernel-based Virtual Machine):

Kernel-based Virtual Machine (KVM), a hypervisor, represents the latest generation of open source virtualization. The goal of this hypervisor is to create a modern hypervisor that builds on the experience of previous generations of technologies and leverages the modern hardware available today. KVM is implemented as a loadable kernel module that converts the Linux kernel into a bare metal hypervisor. It is a lightweight hypervisor module that comes with the mainline Linux kernel.

B. Challenges Posed By The Current Systems:

Instances with static memory are available in Amazon and Eucalyptus. Memory requirements of instances are subject to change. These trends in requirement are very difficult to predict for any user. Also all the memory allocated for any instance may not be utilized which may lead to wastage of memory. Maintaining sufficient resources just to meet peak requirements can be quite costly.

On-demand resource provision is one of the key features of cloud computing, allowing applications to grow or shrink on the basis of dynamic workloads. Elasticity can become a very important feature if VMs can dynamically scale-up or scale-down on the basis of workloads [1]. It will increase efficiency and reduction in power consumption.

C. Current Optimizations And Their Limitations:

Although Xen and KVM provide a ballooning driver they cannot tell when to relocate and how much memory should be allocated or de-allocated without affecting the performance of the applications running on it. The VMware ESX server uses a share-based allocation system. The share of a VM is determined by its memory utilization. VMware uses a sampling method to determine memory utilization. The utilization is the ration of pages actively in use to a random set of pages samples periodically. A VM which has low memory utilization is more likely to get its memory de-allocated. But this approach can only predict memory requirement when there is free memory.

Xen and KVM are two open-source virtualization solutions. The Xen hypervisor acts as a thin layer between the hardware and the operating system, allowing multiple virtual servers to run simultaneously on a single physical server. We chose to work with KVM and implement features like auto-scalability in it because of the following reasons:

- Main disadvantage of Xen is that porting of OS is required to virtualize it. While it is possible in Open Source Operating Systems but proprietary OS like Windows have not opened up to the concept and are not likely to in near future.
- Several major hosting providers are switching their platforms from Xen to KVM. Several Linux vendors including Red Hat and Canonical, which makes Ubuntu have thrown their weight behind the KVM.

- It is difficult to simply drop a Linux guest on top of Xen. You need to make extensive modifications to the guest to make it work properly.
- KVM, meanwhile, shines in many areas where Xen does not. Unlike Xen, which is its own separate microkernel, KVM is part of Linux, and thus does not require an additional layer to install and maintain. Furthermore, Linux guests run on KVM without any hassle.
- Open cloud platform XCP (Xen Cloud Platform) [2] has these features to some extent, uses Xen hypervisor.

Thus, KVM technology has rapidly emerged as the next-generation virtualization technology, following on from the highly successful Xen implementation.

II. RESOURCE MONITORING

Resource Monitoring is an important aspect of Cloud Computing industry. Monitoring enables us to anticipate the issues and hence resolve them before they start appearing as problems to users. It gives us more control and confidence on the virtual machines and hence helps us in meeting the service quality requirements. Optimization of performance can be achieved by dynamically managing the resources, which minimizes this cost by providing the desired elasticity to handle workload spikes without the overheads. Decision of scaling up/down the infrastructure is based on the monitoring data which tells the current state of resources.

Metrics present at machine level [3] provide us the status of the machine. Poor performance could be due to an issue at machine level like CPU over-utilization or memory over-usage. Different metrics like CPU utilization, load, free memory, number of processes running, disk space, swap memory usage, etc. can be used to monitor each VM periodically. The proc filesystem is a pseudo-filesystem which is used as an interface to kernel data structures. Proc which contains system information such as cpuinfo, meminfo, loadavg, swap space, etc. is used by monitoring commands. By accessing the filesystem we can obtain various metrics which can be used for monitoring different virtual machines.

Following are some files which are available under /proc :

- /proc/cpuinfo – information about CPU
- /proc/meminfo – information about memory
- /proc/vmstat – virtual memory statistics

III. WORKLOAD BALANCING

A. *Prediction of memory usage*

Prediction of memory usage in a cloud system is crucial since it may help to reduce frequent migrations and re-sizing of VM. We use a statistical pattern-based recognition approach to predict memory usage [7]. A statistical clustering algorithm is then employed to identify high-density clusters in this space. The clusters are represented by their centroids. They are defined to be the states representing process memory usage. The identified state model represents the memory usage of a program in its past executions and is used to predict the virtual machine's memory requirements in its future executions.

1) *Cluster Analysis*

Memory usage patterns are obtained by clustering [4] and high density clusters are detected. Thus points belonging to a particular cluster will have identical behavior in their memory usage.

Each point of the cluster is weighted. The data point that represents most-recently recorded metrics is weighted greater that the data points that represents the older data. This ensures that the model adapts according to the changing behavior of the virtual machine memory utilization. The weighting will decrease exponentially with time. The degree of exponentiation will be determined by the stability factor β as explained in Feedback subsystem.

$$W(t, \beta)_m = \frac{1}{t^{\beta}}$$

, where $0 \leq \beta < \beta_{threshold}$

'W' is weight associated with each metric

't' is time since the metric has been recorded

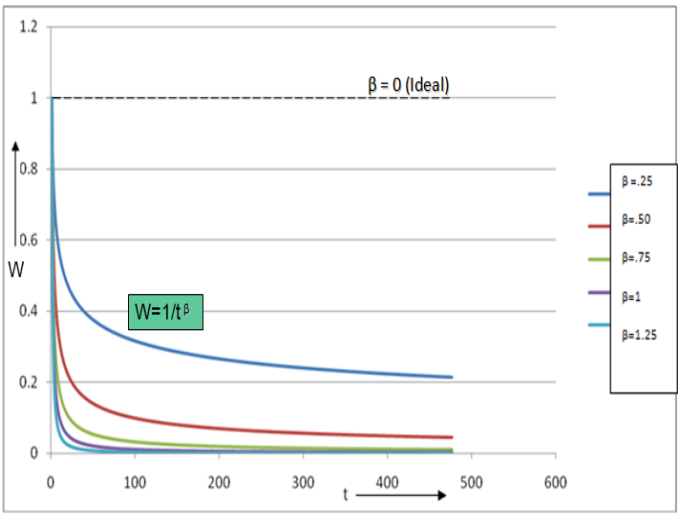


Fig. 1 Variation of W with t and β.

A high instability in memory usage by virtual machine will result in an increase in β. The threshold value of β ($\beta_{threshold}$) is that value after which the prediction algorithm drastically fails. It is initialized by the cloud administrator. For values of β above $\beta_{threshold}$ the prediction subsystems recommendations are not followed by provisioning and automatic memory resizing subsystems.

B. Provisioning subsystem

A virtual machine need to be placed on a host with specifications that best matches its needs. For a virtual machine with $\beta < \beta_{threshold}$ i.e. a stable virtual machine, it is placed on a host with enough available memory as determined by the predicting algorithm. For a new or unstable virtual machine it will be placed on a physical host with maximum memory available.

```
Provision (virtual machine vm)
        If β < β_threshold then
                clust = highest memory utilizing cluster
                Allocated memory for new vm = max (memory needed by clust)
        Else
                Allocate fixed predetermined memory
End
```

C. Automatic memory resizing

The memory required by a virtual machine is allocated dynamically and adaptatively as determined by the prediction algorithm or using the standard predetermined values.

```
Automatic_memory_resizing ()
L1:     After every monitoring time interval t
        Monitor (vm)
        Cluster_analysis (vm)
        If swap usage > acceptable value then
                Increase the memory allocated by 10% of total allocated memory
                Goto L1
        Endif
        If β < β_threshold then
                clust = highest memory utilizing cluster
                Allocated memory= max (memory needed by clust)
        Elseif memory usage < 50% for consecutively for predetermined time then
                Decrease memory allocated to 80% of allocated memory
        Endif
End
```

D. Feedback subsystem

The difference in the predicted memory usage and the actual memory usage will determine the performance of the prediction algorithm. The memory utilization stability of a virtual machine is determined by a stability factor β. Stability factor of the most-recently monitored instance is β'.

1) Determination of β' for a recent monitoring activity

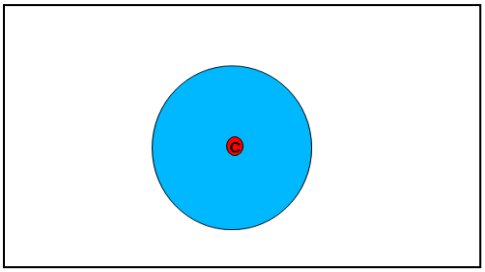
β' depends upon two factors:

- Euclidean distance of the new point in the space from the mean of clusters.
- Whether the new point is or is not a part of an already existing cluster.

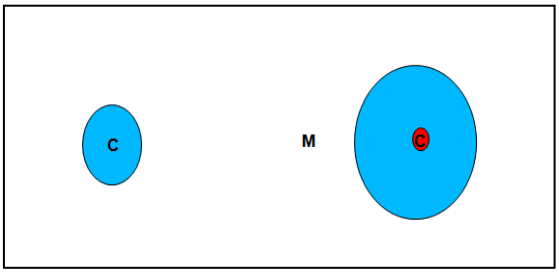
The value of β' is stored for further calculation of β.

Following are few cases in decreasing order of the accuracy of predictor. The value of β' will decrease in the following order. 'C' represents center of cluster, 'M' represents weighted mean. The red dot represents the most-recent point.

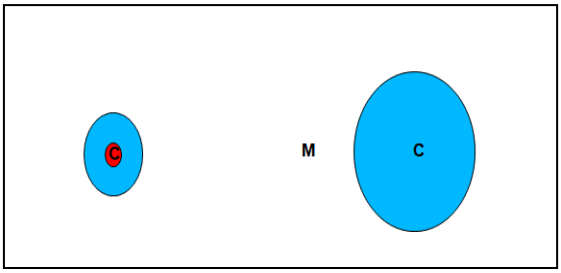
Case 1: Most-recent point at the mean of the only detected cluster.



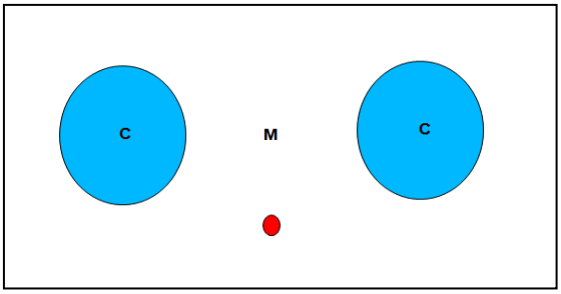
Case 2: Most-recent point at the mean of the largest detected cluster



Case 3: Most-recent point at the mean of the smallest detected cluster



Case 4: Most-recent point not in any of the detected clusters



2) Determination of β

Stability factor β is determined as the weighted average of all β' which is the result of all the recent monitoring activities. The value of n is chosen as a sufficiently large integer.

$$\beta = \frac{2[(n-1)\beta' + (n-2)\beta_1{}' + (n-3)\beta_2{}' + \cdots \beta']}{(n-1)(n-2)}$$

E. Memory re-allocation on a congested host and live migration

When swap usage of a VM on a congested host is utilized consistently on a large scale for some predetermined time it becomes necessary that memory from another virtual machine is reclaimed and allocated to it. For the sake of simplicity consider two virtual machines VM1 and VM2. The algorithm tentatively reduces the memory allocation of VM1 by *S*, increases the allocation of VM2 by *S*, and determines their swap usage. We continue this step with increment/decrement strides of 2*S*, 3*S*, and so on, until the swap usage reaches a local minimum. The algorithm repeats the above process but now it reduces allocation of VM1 while increasing allocation for VM2. It stops when it detects the other local minimum. The algorithm returns the allocation plan based on the lower of the two minima. This algorithm can run recursively when there are more than two VMs. For a very close difference in swap usage the VM with greater memory allocation according to current prediction is allocated. If the swap usage of all the VMs goes beyond the predetermined level the most affected VM is reallocated on another host according to its placement strategy.

IV. CONCLUSION

The immense development in the field of virtualization combined with the wide usage horizon and inadequacy of the current solutions has led to various approaches to virtualization being explored. Cloud computing has evolved with advance techniques in virtualization. As virtualization and cloud computing technology gain more ground in both- academics and industry, resource management in a virtualized system remains a key problem. Under-allocation of memory will lead to a significant drop in performance of the system while over-allocation will lead to wastage of memory. In this paper, we present a memory balancer that predicts the memory usage of a VM and automatically adjusts the memory allocation to improve memory resource utilization. An evaluation of our implementation shows that it is capable of boosting the overall performance of the system by sizeable proportions through automatic memory balancing. In future work, we plan to extend this model to include automatic CPU and disk allocation/de-allocation.

ACKNOWLEDGEMENT

We wish to express our thanks to the following people for their contributions to the system or the paper. Mr. Chirag Jog, our guide whose knowledgeable guidance, support and timely criticism played an important part in shaping this project. Mrs. Rekha Kulkarni's reviews which helped us to improve this project and the paper. Last but not the least we wish to thank all the anonymous reviewers who patiently read our paper and gave us valuable inputs which helped us in making significant improvements in the paper.

REFERENCES

[1] Trieu C. Chieu, Ajay Mohindra, Alexei A. Karve and Alla Segal," Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment ", 2009 IEEE International Conference on e-Business Engineering.

[2] Xen Cloud Platform Administrator's Guide Release 0.1, Published October 2009, 1.0 Edition.

[3] Sathiya Moorthy, "*Explore Linux /proc file system",* http://www.thegeekstuff.com/2010/11/linux-proc-file-system/, November 15, 2010.

[4] Martin Ester, Hans-peter Kriegel, Jörg S, Xiaowei X, "A density-based algorithm for discovering clusters in large spatial databases with noise",1996.

[5] Sauravjyoti Sarmah, Rosy Das, and Dhruba Kr. Bhattacharyya, "A Distributed Algorithm for Intrinsic Cluster Detection over Large Spatial Data", World Academy of Science, Engineering and Technology 45 2008.

[6] Waheed Iqbal, Matthew N. Dailey, Imran Ali, and Paul Janecek, "Adaptive Resource Allocation for Back-end Mashup Applications on a Heterogeneous Private Cloud", Computer Science and Information Management Asian Institute of Technology, Thailand.

[7] Murthy v. Devarakonda and Ravishankar k. Iyer, "Predictability of Process Resource Usage: **A** Measurement-Based Study on UNIX", IEEE transactions on software engineering, vol. 15, no. 12, December 1989.