Validating the CPU usage statistics provided by Linux KVM hypervisor.

Roopa Govind , Mamatha T
Dept. of Computer Science and Engineering,Cambridge Institute of Technology, Bangalore, India

Abstract

The x86 based virtualization technologies has been an important innovation that has fuelled the realization of cloud computing using commodity hardware. Resource usage statistics of the virtual machines is one of the important factors for the cloud providers to help them take critical and valuable decisions varying from resource provisioning to QoS requirements, identifying bottlenecks in the system and performance profiling of VMs. Hence it is important to ensure that the resource usage information provided by the hypervisor, in our case, Linux KVM, is accurate and reliable. In this work we have come up with a technique to validate the CPU usage statistics provided by Linux KVM hypervisor for the running virtual machines is indeed reliable and accurate.

Keywords—**CPU statistics, Monitoring, CPU workloads, Virtual machine monitors, System performance, Libvirt, KVM hypervisor.**

I. INTRODUCTION

Most of the production environments, today, have workloads that are either memory intensive, I/O intensive or CPU intensive. Above this, the workload also exhibits dynamic behavior which results in underutilized resources. Some of the study undertaken reveals that the resource utilization could be as low as 15%, which doesn't justify the ROI that enterprises spend towards IT. In such situations, to improve resource utilization, many enterprises are shifting towards cloud computing solutions where elastic resources can be availed on a pay-by-use mode.

Virtualization is a key enabler in these cloud based solutions where multiple workloads are consolidated on a single machine. On such virtualized servers, physical hardware is shared among multiple virtual machines by a layer called hypervisor or virtual machine monitor (VMM). Workloads in turn, are hosted in the virtual machines and access the underlying shared hardware through hypervisor. Virtualization helps provide isolation, security, encapsulation in such shared environments as well as ensures better utilization of server resources.

Many virtual machine monitors have emerged in such a scenario, varying from VMware ESX to Linux KVM hypervisor. In recent years, to make development of virtual machine monitors easy, hardware vendors like AMD and Intel have added virtualization extensions to x86 processors & as well as memory (EPT from Intel and RVI from AMD) which were initially difficult to virtualize[1].

The Kernel based Virtual Machine (KVM) is a relatively new VMM which utilizes these hardware extensions and has found its way in Linux kernel. It is a full virtualization solution, which requires no changes in guest operating system [2]. While virtualization provides a simple mechanism to share resources by isolating the application's software environment, most of the applications incur some kind of virtualization overhead. This overhead varies depending upon the type of application, type of virtualization and virtual machine monitor used. Particularly, for I/O applications, the overhead of CPU used by VMM on behalf of VMs is considerable and affects performance characteristics of applications [3][4].

To account for such overhead, it is necessary to monitor resources used by the hypervisor on behalf of the VM. Most of the resource monitoring tools available in the market today will depend largely on the hypervisor to provide an accurate and reliable resource usage statistics. In Linux KVM, the VM-specific CPU usage is mainly provided through Cgroups accounting. In this paper, we present a method for validating the CPU resource statistics provided by Linux KVM hypervisor is indeed accurate and reliable.

The rest of the paper is organized as follows. Section II describes the KVM architecture with a focus on how the Guest VMs utilize the underlying hardware provided by KVM. Section III describes the existing method to obtain CPU statistics of a VM from the KVM hypervisor and how to interpret the statistics to make critical and valuable decisions varying from resource provisioning to QoS requirements. Section IV describes the technique to validate the reliability and accuracy of the CPU statistics provided by KVM hypervisor and section V presents the analysis made during the validation phase. Finally, Section VI concludes by discussing the actions taken based on the observation made during the validation phase.

II KVM ARCHITECTURE

KVM is a more recent hypervisor which embeds virtualization capabilities in Linux kernel using x86 hardware virtualization extensions [2][7]. It is a full virtualization solution, where Guests are run unmodified in VMs. It consists of two modules, namely, kvm.ko and an architecture dependent kvm-amd.ko or kvm-intel.ko module. Under KVM, each VM is spawned as a regular linux process named KVM and scheduled by the default linux scheduler. For KVM the hardware has to support three processor modes, namely user, kernel and guest mode. The guest mode is added to support hardware assisted virtualization. The virtual machine executes in this guest mode which in turn has user and kernel mode in itself [5][6].

For using shared I/O hardware, these VMs interact with QEMU emulator in host user space which provides emulated I/O devices for virtual machines. For instance, in the case of network related applications, QEMU provides emulated Network Interface Card (NIC) to VMs and interacts with tun-tap device on the other side. The tap device is connected to physical NIC through a software bridge.

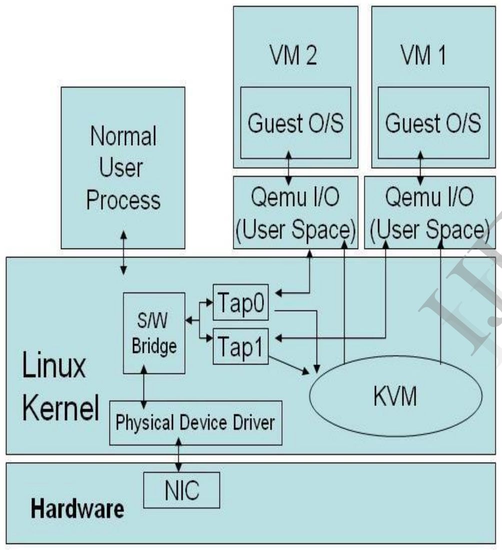


Fig 1 shows the typical KVM architecture

III OBTAINING & INTERPRETING THE DOMAIN CPU STATISTICS FROM KVM HYPERVISOR

Libvirt is a hypervisor-independent virtualization API that is able to interact with the virtualization capabilities of a range of operating systems. Domain CPU statistics from KVM hypervisor can be obtained through libvirt, the management API layer of Linux KVM. Virsh is a command line interface tool for managing guests and the hypervisor. The virsh tool is built on the libvirt management API and operates as an alternative to the xml tool and the graphical guest Manager (virt-manager). The CPU statistics for a running VM can be obtained using the virsh cpu-stats command, which displays the per-CPU and total statistics about the domain's CPUs

```
[roopa@064904 ~]$ virsh list --all
 Id    Name                     State
----------------------------------------------------
 1     ubuntu-openstack             running
```

To get the current domain CPU stats for this running VM, “Ubuntu-Openstack”, we can run the cpu-stats command as follows:

```
[roopa@064904 ~]$ virsh cpu-stats  ubuntu-openstack
CPU0:
        cpu_time           2.935644229 seconds
CPU1:
        cpu_time           0.656418975 seconds
CPU2:
        cpu_time           0.504604680 seconds
CPU3:
        cpu_time           0.435697162 seconds
Total:
        cpu_time           4.532365046 seconds
        user_time          0.210000000 seconds
        system_time          0.610000000 seconds
```

The above output shows the per-CPU and total statistics about the domain's CPUs. As we can see above, there are 4 physical CPU's on the host and the time this VM has run on each CPU is displayed in seconds. The Total stat shows the total time the VM has run on all the available physical CPU's and the user_time corresponds to the time the VM has run in "User Mode" (running applications) & system_time corresponds to the time it has run in "Root mode & Guest mode" (running kernel code).

The main aim of this paper is to demonstrate a technique to validate the accuracy and the reliability of this CPU statistics provided by KVM.

IV TECHNIQUE TO VALIDATE THE CPU STATISTICS PROVIDED BY KVM HYPERVISOR

The proposed technique developed to validate the reliability and accuracy of the CPU statistics is by increasing the CPU time of the VM by running a CPU intensive application in User Mode inside the VM and analyse the statistics generated as a result of the CPU load. To ensure that our approach is accurate and precise, the test environment was setup using the following steps.
1. Run only 1 VM on the host.
2. Number of VCPUs in VM is equal to the number of host CPUs and each VCPU of the VM is pinned to the host CPU.

The virsh vcpuinfo command shows this information.

```
[roopa@06490 ~]$virsh vcpuinfo ubuntu-openstack

        VCPU:          0
        CPU:           0
        State:         running
        CPU time:      5.1s
        CPU Affinity: y---


        VCPU:          1
        CPU:           1
        State:         running
        CPU time:      5.9s
        CPU Affinity: -y--
```

```
VCPU:          2
CPU:           2
State:         running
CPU time:       3.5s
CPU Affinity: --y-

VCPU:          3
CPU:           3
State:         running
CPU time:      3.2s
CPU Affinity: ---y
```

3. Increase the CPU shares of the Libvirt Cgroup using Control Groups mechanism.
4. Increase the CPU scheduler prioritization for the guest under investigation.

The Libvirt API, and the virsh command line tool provide mechanisms to set tunables on guests. For the QEMU drivers in Libvirt, the virsh schedinfo command provides access to CPU scheduler prioritization for a guest. The “cpu_shares” value is a relative priorization. All guests start out with a cpu_shares of 1024. If you double its “cpu_shares” value, it will get twice the CPU time as compared to other guests. This is applied to the guest as a whole, regardless of how many virtual CPUs it has.

```
[roopa@064904 ~]$ sudo virsh schedinfo --set cpu_shares=4096 ubuntu-openstack

        Scheduler      : posix
        cpu_shares     : 4096
        vcpu_period    : 100000
        vcpu_quota     : -1
```

Steps 1-4 is an approach to ensure that the VM under investigation will utilize the host CPU's in accordance with the CPU load that is generated inside the VM. To be more precise we are expected to see in SAR output, the “guest %" time on the host will increase in accordance with the CPU load inside the VM.

5. Gather the VM CPU statistics without running a CPU load inside the VM. This is taken to have a baseline to compare with during our analysis.

6. Generate a 75% CPU load inside the VM for 3 mins and subsequently generate a 90% load inside the VM for another 3 mins using an open-source cpu-load generator tool.

7. After the CPU load is generated, gather the VM CPU statistics for further analysis.

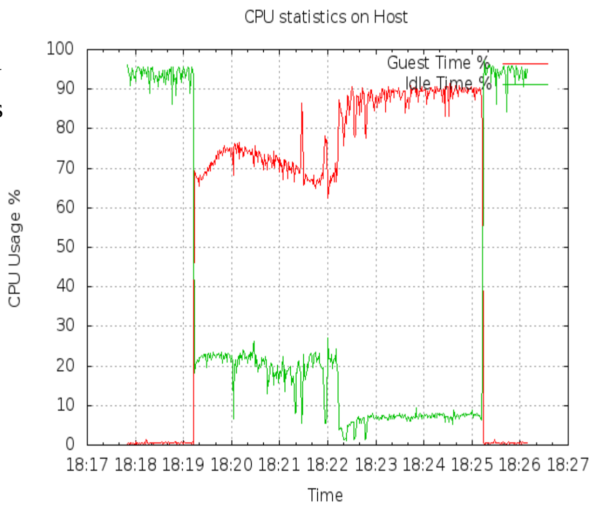
V Analysis Of Statistics Obtained

We start the process of analysis by first gathering the VM CPU statistics before generating the CPU load by using the virsh cpu-stats command as follows:

```
[roopa@064904 ~]$ virsh cpu-stats ubuntu-openstack


CPU0:
        cpu_time          9.587765257 seconds

CPU1:
        cpu_time          7.645363434 seconds
```

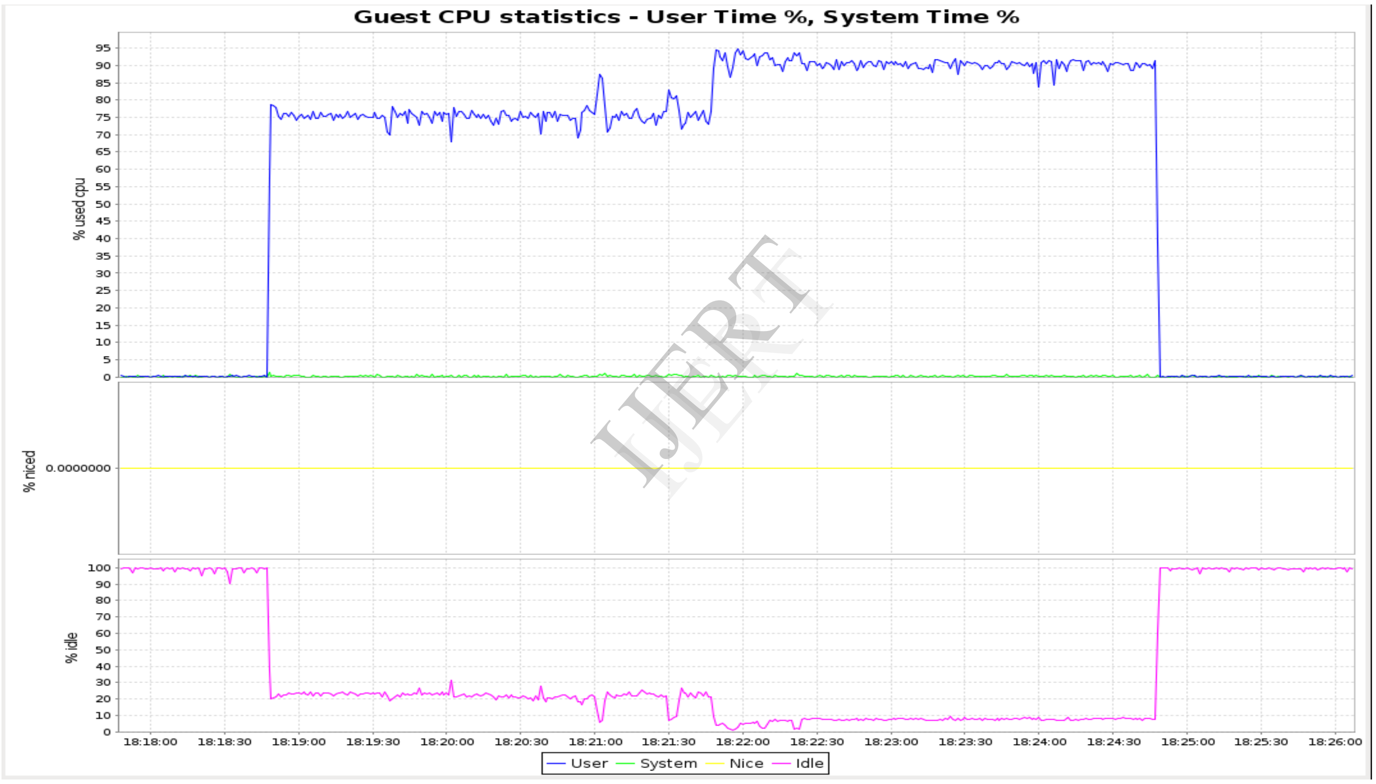


```
CPU2:
        cpu_time        7.583843416 seconds

CPU3:
        cpu_time        8.768952257 seconds

Total:
        cpu_time        33.585924364 seconds
        user_time        1.570000000 seconds
        system_time       4.100000000 seconds
```

We then generate the workloads with the running VM and observe the CPU usage on the Guest as well as the Host using the SAR tool [8]. The graph below shows the CPU usage inside the VM before and after generating the CPU load. The graph shows that we generated a CPU load of 75% for 3 mins and 90% for 3 mins, where the "User Time" shot up to above 75% and 90% subsequently. During this time "System Time" was almost 0% & Idle Time is below 10%.



On the host, during this time, the CPU utilization of the VM matched up to the load we generated inside the VM.

Following is the CPU statistics taken after generating a CPU load of 75% for 3 mins and 90% for subsequent 3 mins inside the VM.

```
[roopa@064904 ~]$ virsh cpu-stats ubuntu-openstack

CPU0:
        cpu_time        324.015818392 seconds

CPU1:

        cpu_time        324.397892025 seconds
```

```
CPU2:

          cpu_time        338.334924222 seconds

CPU3:
            cpu_time        329.628496451 seconds

Total:
          cpu_time        1316.377131090 seconds
          user_time         12.520000000 seconds
          system_time       79.510000000 seconds
```

From the statistics above the observation is that the user_time provided by KVM hypervisor is less than the system_time which does not correlate with the CPU utilization of the VM. Also the user_time + system_time is way too less than the total cpu_time provided by KVM hypervisor. Thus we have identified that the CPU usage information provided by Linux KVM is not accurate.

VI CONCLUSION AND FUTURE WORK

The domain CPU statistics for a VM provided by KVM hypervisor is not accurate and reliable. The issue is reported to Linux Kernel community and this bug is currently under investigation.

https://bugzilla.kernel.org/show_bug.cgi?id=55061

In this work we validate the CPU usage statistics provided by the Linux KVM hypervisor.

This research can be extended to validating the usage of other resources by VM's such as memory, Network and storage.

VII References

[1] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," Commun. ACM, vol. 17, no. 7, pp.412–421, Jul. 1974. [Online]. Available: http://doi.acm.org/10.1145/361011.361073

[2] (2012) Main page-kvm. [Online]. Available: http://www.linux-kvm.
org/page/Main Page

[3] J. Lakshmi, "System virtualization in the multi-core era, a qos perspective," Dissertation, Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore, 2010.

[4] M. Dhingra, J. Lakshmi, and S. Nandy, "Resource usage monitoring in clouds," in Grid Computing (GRID), 2012 ACM/IEEE 13th InternationalConference on, sept. 2012, pp. 184 –191.

[5] S. Zeng and Q. Hao, "Network i/o path analysis in the kernel-based virtual machine environment through tracing," in Information Science and Engineering (ICISE), 2009 1st International Conference on, dec.2009, pp. 2658 –2661.

[6] J. Zhang, K. Chen, B. Zuo, R. Ma, Y. Dong, and H. Guan, "Performance analysis towards a kvm-based embedded real-time virtualization architecture," in Computer Sciences and Convergence Information Technology(ICCIT), 2010 5th International Conference on, 30 2010-dec. 22010, pp. 421 –426.

[7] A. Kivity, "kvm: the Linux virtual machine monitor," in OLS '07: The 2007 Ottawa Linux Symposium, Jul. 2007, pp. 225–230

[8]System Monitoring With sar And ksar http://www.howtoforge.com/system-monitoring-with-sar-and-ksar