KVM Virtualization Impact on Active Round-Trip Time Measurements

Ramide Dantas
DASE/CSIN
IFPE, Recife
ramide@gprt.ufpe.br

Djamel Sadok
CIn/GPRT
UFPE, Recife

Christofer Flinta, Andreas Johnsson
Research Area Cloud Technologies
Ericsson Research, Stockholm

***Abstract*—Active measurements tools transmit probe packets between a sender and a receiver to estimate performance metrics such as round-trip time, jitter and loss. In this paper we evaluate how KVM virtualization affects measurements of performance metrics, specifically the round-trip time (RTT). To understand the impact we investigate the interplay of various environment and measurement parameters with virtualization. A number of experiments are performed in order to investigate which parameters had major impact on the RTT. The paper shows that the measurements are affected by CPU load in the host as well as network load while I/O load seemed to have limited impact.**

Keywords— Active network measurements; virtualization; cloud management; performance metrics; KVM

I. INTRODUCTION

Active measurements (aka active probing) have long been an accepted method for determining performance parameters such as delay, round-trip time, loss and jitter of packet-switched networks. The basic concept is to transmit probe packets from a sender towards a receiver. Each packet is time stamped on both sides which enables performance parameter estimation. Common protocols for active measurements include IETF TWAMP [1] [2] and IETF ICMP [3].

As a majority of the computing, storage and networking capabilities is migrating towards cloud platforms there is a need to study how these tools are affected by the new environment. Active measurement tools are necessary for monitoring network performance between the guest machines (from the customer perspective) and to keep the cloud platforms working smoothly (for cloud operators).

Active measurement tools are designed with network elements in mind. Buffering at intermediate nodes is a major factor that is embraced when measuring the network performance and computing the estimates. Virtualization poses a series of new challenges since it introduces additional software layers, hence introducing additional delay, that the probe packets need to cross.

In this paper we utilize a testbed to evaluate how KVM virtualization affects active measurements of round-trip time (RTT). We investigate the effects of various environment parameters such as the type of load (CPU, I/O and network) and CPU reservation for VMs. The results show that the RTT is vulnerable to the CPU and network load for co-located VMs, especially using short time interval between measurement packets. In case of CPU load, reserving a CPU core to the measurement VM proves to be a good practice. Experiments with network load, on the other hand, point to the virtual network driver of the host as responsible for off-the-chart measurements. The paper also provides an initial evaluation of which part of the communication path (apart from the physical network) that has the main impact on the RTT measurements.

II. RELATED WORK

We recognize that there is plenty of related work in the field. However, due to the page limitation we restrict ourselves to only mention a few contributions.

The work in [4] investigates the accuracy in time stamping of network packets and proposes a method for estimating the time stamping error. Further, the results in [10] shows that the time stamping accuracy is reduced when operating in a virtual environment. This paper differentiates itself by evaluate how de-facto standard measurement tools perform in this setting.

The impact of virtualization (Linux-VServer and Xen) on packet transmission and reception is investigated in [5]. The results shows that heavy network usage from competing VMs can introduce delays as high as 100 ms to the RTT. This is further investigated in the Amazon EC2 environment [6]. The results show unstable network performance caused by virtualization. This paper shows similar results but extending the work to the KVM environment.

The work in [7] focuses on UDP jitter. The authors identify the software layers responsible for adding extra delay in the Xen and VMware virtualization platforms. Again this paper shows similar results but extended to the KVM setting.

Network performance of three virtualization platforms: KVM, Xen and OpenVZ are compared in [8]. They measured maximum throughput, network performance isolation, fairness and jitter. The experiments were conducted with various CPU or network workloads. In most of the analysis Xen performed equally or better than KVM and OpenVZ.

III. EXPERIMENT SETUP

In order to assess the way virtualization affects the RTT metric, we constructed a testbed comprised of two directly connected physical host machines, see Figure 1. Each machine runs one measurement VM and a set of background VMs. The measurement VMs send and receive measurement packets using ICMP while the background VMs generate different load types in order to assess the isolation capabilities (or lack of) of

the virtualization platform. Measurement and background VM's are connected to an internal bridge emulated by the host Linux kernel. A secondary physical network bridge is used for experiment scheduling (not shown in figure). Hence, no control traffic is going through the experiment setup.

The host machines are equipped with Intel i7 processors, 16 GB RAM and Ubuntu Server 12.04 (kernel v3.2.0). The guest machines are Debian logical boxes running kernel 2.6.32.

The virtualization platform used in the experiments is the Linux Kernel-based Virtual Machine (KVM) [11]. KVM takes advantage of the kernel's facilities for sharing host resources and avoiding duplication of functionalities. KVM uses QEMU [12] for emulating guest machine virtual hardware devices and uses processor virtualization extensions (Intel VT-x or AMD-V) for running each VM directly on the hardware. The network driver in the guest machines is either emulated (e1000) or para-virtualized (VirtIO [13]).

We perform Round Trip Time (RTT) experiments using the standard ICMP *ping* command. In each experiment 100 ICMP packets are transmitted with two different packet intervals; 1 second ("1 Hz Ping") or 1 ms ("1000 Hz Ping").

The aim of the study is to quantify the effect of loading various physical resources on the machines, including CPU, network and disk I/O, on the RTT measurement estimates. The load is generated either on the host containing the measurement VM that initiated the ICMP measurement (sender side) or on the host running the target measurement VM (receiver side).

CPU and I/O loads are generated using the *stress* tool [14] which executes in the background VMs. The stress tool can generate load on for example the CPU, memory and disk. Network load (TCP) is generated by iperf [15]. All loads are controlled and configured through the secondary bridge.

The number of background VMs are set to one of the following values: 0, 1, 2, 4, 8, 16, 32, 64 or 128. These VMs are executed on either the sender or the receiver side. The measurement VM either executes with or without CPU core reservation. In the case without CPU core reservation the VM has to compete for resources with background VMs.

To study the impact of virtualization on the RTT estimates we also capture the ICMP packets at various points of the communication path using the *tcpdump* tool. The capture points are depicted and enumerated (1-6) in Figure 1.

They are: 1) Sender-VM virtual interface, 2) Sender-Bride which listens to the virtual bridge, 3) Sender-Host, which is the physical NIC). Equivalent capture points are located on the receiver side and are labeled Receiver-VM, Receiver-Bridge and Receiver-Host. By analyzing the generated packet traces we estimate RTT at the intermediary measurement points of the path to aid isolation of effects on the RTT.

We have evaluated all above parameter combinations (864 scenarios), for each we collected 7 different RTT estimates (6 capture points and the Ping output). We limit the presentation of the results to the most relevant findings. Table 1 summarizes the complete set of parameter types and their values.

TABLE I. EXPERIMENT PARAMETERS.

Parameter	Values
VM CPU Reservation	Yes, No
BG Load Type	CPU, Network, I/O
BG Load Location	Sender, Receiver
Number of BG VMs	0, 1, 2, 4, 8, 16, 32, 64, 128
VM Network Driver	e1000, VirtIO
Ping Frequency	1 Hz, 1000 Hz

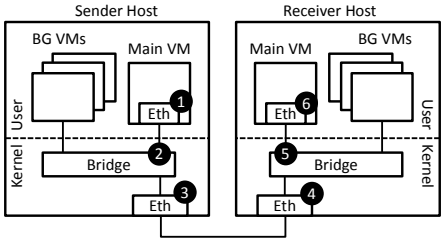

Fig. 1. Experiment setup with two host machines with KVM and VMs.

IV. MEASUREMENT RESULTS

Since we have a considerable parameter space at hand, the results are divided into subsections corresponding to the following different load scenarios: (A) measurements with no load, (B) the effect of CPU load, (C) the effect of network load and finally (D) the effect of Input/Output (I/O) load. The section ends with a discussion about outliers (E).

A. Measurements with no load

Table 2 summarizes the results obtained without execution of background VMs. The main finding is measurements are sensitive to the time interval between the probe packets. It can be seen that high frequency measurements give lower RTT values throughout the experiments. For example, the median RTT estimates are close to 0.8ms with a measurement frequency of 1Hz. For the 1000Hz experiments the RTT values are in most cases cut in half. It is also interesting that CPU reservation has no impact at all.

One possible explanation to the response of RTT with respect to frequency is that the hypervisor tries to increase the overall network performance. Hence, high-intensity traffic benefits from lower waiting times in the hypervisor network I/O module, therefore affecting the RTT results. We refer to [9] for a discussion on KVM/QEMU network performance improvement techniques.

B. Effect of CPU load

This section presents results from CPU load scenarios; what is the impact of CPU reservation and does the measurement frequency and load location impact the RTT measurements?

Figure 2 (left) shows a boxplot of RTT measurements. The results originate from a specific scenario with CPU reservation, load on the sender side and 1Hz measurement frequency.

One conclusion is that the number of background VMs does not affect the measurement results. Measurement results from scenarios without CPU reservation showed similar results as in this figure and is hence omitted.

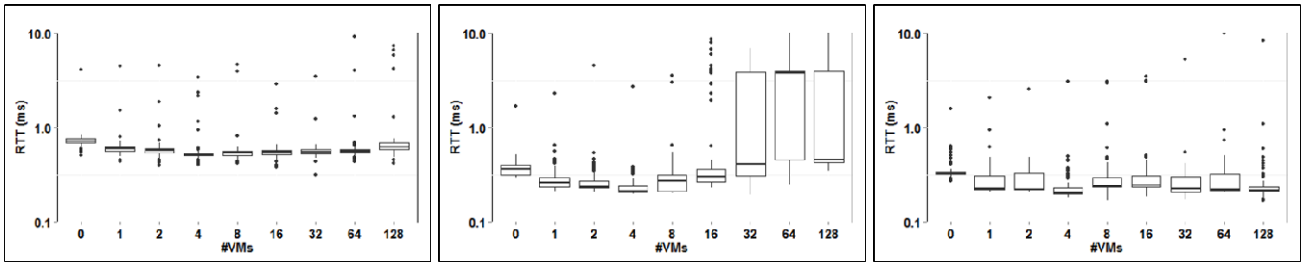

Fig. 2. RTT (ms) vs #VMs. For all plots the e1000 driver is used. (Left) CPU load at the sender side, 1Hz Ping, with CPU reservation. (Center) CPU load at the sender side, 1000Hz Ping, without CPU reservation. (Right) CPU load at the sender side, 1000Hz Ping, with CPU reservation.

TABLE II. RTT MEASUREMENTS FOR THE CASES WITHOUT LOAD IN THE HOST MACHINES.

Ping Freq.	Driver	CPU Reserv.	RTT [ms]				
			Min.	Med.	Max.	Avg.	SD
1 Hz	e1000	On	0.510	0.721	4.100	0.743	0.345
		Off	0.511	0.788	2.110	0.788	0.144
	VirtIO	On	0.613	0.825	4.150	0.847	0.340
		Off	0.599	0.833	3.640	0.849	0.287
1000 Hz	e1000	On	0.271	0.323	1.570	0.355	0.139
		Off	0.294	0.360	1.670	0.375	0.141
	VirtIO	On	0.231	0.299	3.820	0.346	0.356
		Off	0.233	0.304	5.290	0.376	0.503

The center and right graph in Figure 2 shows results from scenarios with 1000Hz measurement frequency. It is obvious that CPU reservation of the measurement VMs impacts the RTT estimates. In the case without CPU reservation the RTT has an increasing trend from 8 simultaneous background VMs. This is not the case with CPU reservation; rather the RTT remains unaffected by the number of background VMs.

One possible explanation to the results from high frequency measurements is the kernel prioritization of tasks. Since the measurement VM is I/O bound it is given higher priority by the Linux kernel than the CPU load. This higher priority is not sufficient for isolating the measurement VM (even with CPU reservation) from the other VMs. That is, the measurements must compete for CPU time with background VMs, which affects packet sending and time stamping.

The above results originate from scenarios with load on the sender side. As indicated in Section II the CPU load can also be executed at the receiver host. The general finding for these scenarios is that the RTT is less sensitive to CPU load. The RTT results are very similar to Figure 4 regardless if the measurement VM executes on a reserved CPU core or not.

This can be explained by the fact that the ICMP request packet requires less support from the receiver VM to be processed since the packet is replied to by the VM kernel and not by a user space application. This in turn requires a lower number of context switches to process the packet and hence the impact on RTT estimates remain low.

We also note, while omitting the results in the paper, that the VirtIO network driver gives results with similar tolerance to VM load as those using the e1000 driver.

C. *Effect of network load*

This section investigates the impact of network load on the RTT measurements. The discussed results are shown in Figure 3 (left and center). The figures shows the average RTT for different capture points (see Section II). A time stamp is taken when an ICMP packet crosses the capture point on the forward and reverse path. The capture point time corresponds to the time for a specific ICMP packet to cross the capture point twice. The major contribution to the RTT comes from the Sender-VM and Sender-Bridge capture points. Hence, to increase clarity we only show results from these capture points in the figures.

The RTT is significantly affected both with and without CPU reservation for most configurations, making contention at the network stack the major cause of RTT increase. In fact, network load is the type of load with major impact on RTT. For most of the experiment configurations the measured RTT increases significantly, going from less than 1ms to more than 100ms in some cases.

The behavior, however, should not be seen as exclusive to the software network stack implemented in Linux. Hardware network switches and routers being loaded with high network traffic on several simultaneous ports may present similar, although less significant, results. Our scenario shows an extreme case where up to 128 machines are sending traffic into the same network element at full speed, and therefore the RTT measures will reflect the amount of traffic in the network, no matter the type of switch used (virtual or real).

Comparing the left and center graphs of Figure 3 one can observe a striking difference in estimated RTT for the Sender-Bridge. The emulated driver (e1000 in left) provides an average RTT below 20ms throughout the graph. Using the para-virtualized driver (VirtIO in center) the RTT increased at a higher pace; ending up at 120ms. We assume the difference is due to implementation decisions in the two drivers.

The RTT measured at capture point Sender-Host shows little or no increase (see figure 3, left and center), even for a large number of executing background VMs.

This suggests that the increased RTT originates from the virtual NIC drivers. The current experiment does unfortunately not allow us to identify whether the RTT increase comes from contention in the forward or reverse direction. This is part of future research.

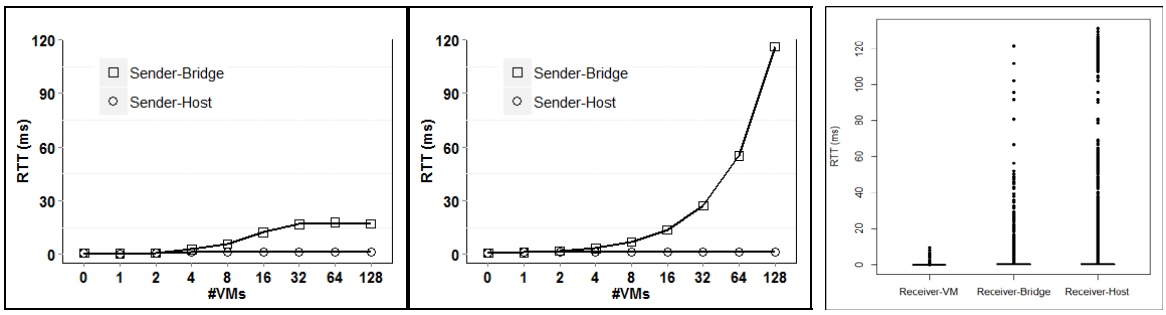

Fig. 3. (Left) Average RTT (ms) vs. #VMs using network load, e1000 driver, without CPU reservation, 1 Hz Ping. (Center) Average RTT (ms) vs. #VMs using network load, VirtIO driver, without CPU reservation, 1 Hz Ping. (Right) Boxplots of RTT measurements (ms) for three capture point.

D. *Effect of Input/Output load*

The experiments with input and output load in the background VMs showed that the RTT in general is insensitive to the number of background VMs and that the results do not improve by CPU core reservation. This finding however can be strongly dependent on the type of I/O load generated by the *stress* tool (disk writes and deletes). Further investigation is necessary for a conclusive position on the impact of I/O load.

E. *Analysis of Outliers*

One common effect we noticed in all experiments was the presence of samples with RTT values above 100ms even if the median was below 1ms. This type of outlier behavior has also been discussed in previous work such as [5]. Here we narrow down the origin using capture point information. To do this, all sample data were grouped by capture point. Figure 3 (right) shows the boxplots obtained for three interesting capture points: Receiver-VM, Receiver-Bridge and Receiver-Host.

The delay added by the Receiver-VM capture point is limited to approximately 10ms. The capture point Receiver-Bridge, that covers the passage between the virtual and real domains, introduces more extreme RTT values. The Receiver-Host capture point, now including the whole stack, seems to push the RTT above 100ms. This suggests that the major contribution in terms of extreme RTT comes from network virtualization at the host level and the network access. In-depth investigations reveal that the most extreme outliers originate from experiments with network load (mainly with 64 and 128 background VMs). Future work will isolate the causes these effects in the Linux bridge.

V. Discussion and Conclusion

This paper presents results and conclusions from evaluating the impact of KVM virtualization on active measurements conducted with ICMP ping. The goal of the evaluation was to identify which type of system, load and virtualization parameters had most influence on the measurement results.

The results suggest that a cloud operator needs to ensure a high degree of isolation between virtual machines in order to support high accuracy measurements to cloud customers. Having a reserved core for the customer's virtual machine is an intuitive option, and as our results showed, a good one. But depending on the load type at the host machine it might not be enough or even necessary. The I/O profile in the experiments showed little impact on measurements, with or without CPU reservation. High network load on the other hand may not benefit from CPU reservation since the contention is in the hypervisor. We also illustrated the problem with outliers, which is not entirely resolved using CPU reservation.

Acknowledgements

Work supported by the Research and Development Centre, Ericsson Telecomunicações S.A., Brazil.

References

[1] K. Hedayat et al. "A Two-Way Active Measurement Protocol (TWAMP)". IETF RFC 5357, October 2008.

[2] S. Baillargeon, C. Flinta, A. Johnsson, "Ericsson Two-Way Active Measurement Protocol (TWAMP) Value-Added Octets". IETF RFC 6802, November 2012.

[3] J. Postel. "Internet Control Message Protocol". IETF RFC 792, September 2001.

[4] P. Arlos, M. Fiedler. "A method to estimate the timestamp accuracy of measurement hardware and software tools." Passive and Active Network Measurement, 2007.

[5] J. Whiteaker, F. Schneider, R. Teixeira. "Explaining packet delays under virtualization." ACM SIGCOMM Computer Communications Review 41.1, 2011.

[6] G. Wang, TS E. Ng. "The impact of virtualization on network performance of amazon ec2 data center." IEEE INFOCOM, 2010.

[7] A. Arsene, D. Lopez-Pacheco, G. Urvoy-Keller. "Understanding the network level performance of virtualization solutions." IEEE Cloud Networking (CLOUDNET), 2012.

[8] A. Soule, Fabio. Picconi. "Measuring the impact of virtualization on network applications." Thomson Technical Report Number: CR-PRL-2009-03-0001, 2009.

[9] L. Rizzo, G. Lettieri, V. Maffione. "Speeding up packet I/O in virtual machines." ACM/IEEE Architectures for networking and communications systems, 2013.

[10] R. Mutia, N. Sadeque, J. Andersson, A. Johnsson. "Time-stamping accuracy in virtualized environments". Advanced Communication Technology (ICACT), 2011.

[11] Kernel Based Virtual Machine (KVM): http://www.linux-kvm.org/

[12] QEMU: http://www.qemu.org/

[13] VirtIO: http://www.linux-kvm.org/page/Virtio

[14] STRESS tool: http://people.seas.harvard.edu/~apw/stress/

[15] IPERF tool: https://iperf.fr/