Impact of Xen and Virtual Box Virtualization Environments on Timing Precision under Stressful Conditions

Mykola Korniichuk , *Kirill* Karpov, *Irina* Fedotova, *Veronika* Kirova, *Nikolay* Mareev, *Dmytro* Syzov and *Eduard* Siemens

Anhalt University of Applied Sciences - Faculty of Electrical, Mechanical and Industrial Engineering, Bernburger Str. 57, 06366, Köthen, Germany

Abstract. Sharing physical resources among virtual instances introduces time overhead in comparison with direct access to hardware. Such lag is not significant for most of the everyday tasks however it can influence much more in dealing with time-critical applications and especially in case of reliable network services. The purpose of this paper is to compare time overhead on timing operations such as time acquisition and sleep introduced by different virtualization environments: Xen, VMWare ESXi, QEMU. The current research focus to establish possibility of using such platforms in real-time applications with high resource utilization. In terms of present work, there are several load types to be used to simulate real conditions. Measurement performance includes different methods of time measurements and waiting operations. Considering results, the certain recommendations about timing mechanisms using different virtualization environment have been offered.

1 Introduction

Nowadays operating systems are often virtualized due to increasing the hardware utilization and decreasing maintenance costs. However, a lot of different challenges occur due to virtualization. Hypervisor has direct access to hardware and is responsible for sharing all the resources among virtual machines, referred as a guest operating system. This additional software layer, which is influences on the performance and can break time-related functionality.

The precision of time-related operations such as time acquisition and waiting operations are crucial for any networking application. The time overhead on such operations can cause the data rate losses in high performance network applications, such as RMDT transport protocol, where on each sent packet there are 187 time acquisition operations and one sleep procedure.

The goal of this paper is to investigate how precise timing operations behave in different virtual environment under different loads. These results allow to perform an analysis of timing behavior under different conditions and to estimate reliability, provided by TSC and improved sleep methods [4,5]. Such analysis may be helpful for deploying time-related software in a virtual environment.

Measurements are performed on the same hardware setup in so called Enterprise scenario. In this scenario, virtualization is used as a mechanism for fast migration and support of system reliability under conditions of high-performance computation needs. In the experiments presented below, only one guest operating system is being virtualized, and all the measurements performed within the virtualized operating system.

The remainder of this paper is structured as follows: Section 2 reviews related work. The results of experiments with time acquisition and sleep function measurements is presented in Section 4 and 5 respectively. Conclusions and ideas for further work are offered in Section 6.

2 Related work

This work is an extension of our previous paper [1], where QEMU and VMware ESXi hypervisors have been studied. In this paper the same analysis of Xen and Virtual Box virtualization environments will be provided.

Similar analyses for the case of, so called Hosting scenario have been conducted in [2], where impact of different number of virtual machines running in parallel on timing operations inside one of them have been estimated.

S. G. Langer and T. French [3] have conducted research of influence of virtualization on I/O operations. It has been performed on Red Hat, Xen, ESXi, kvm_redhat64 as hypervisors, VMWare player and VirtualBox as hosted-based VMMs.

A detailed performance analysis of Xen Linux was made by P. Barham [4]. The authors made performance measurements in the hosting scenario on the Xen hypervisor-based VMM with different loads to simulate real life servers and compared it with different virtualization software.

The current paper is the extension of paper [1] with two new virtual environments: Citrix XenServer Host 7.0.0-125380c and Virtual Box 4.3.36. All hardware and software equipment, load types and methodologies are the same. With the current paper we intend to finalize the survey of the most valuable virtual environments from the perspective of time-related functionality.

3 Experimental results for time measurements

Time measurements consist of timer data collected with the RDTSC machine instruction, further referred as TSC and timer data collected with clock_gettime() further referred as system call. Experiments is composed of 3 configuration scenarios: Host OS (operating system is installed directly on hardware), Xen hypervisor and Virtual Box hypervisor.

3.1 Idle tested

Figure 1 shows the collected data in an idle state, which further helps to better estimate the results under heavy loads. Bold lines represent the medians, which for Host OS and Xen virtualized OS with TSC Timers are almost the same. However, the results from the Virtual Box significantly differ. As expected, host OS has the lowest time overhead (figure 1 (a)) about 17.2 nsec with TSC timer and 27 nsec with a system call. Xen virtualized OS (figure 1 (b)) shows almost the same median value as for Host OS in case of TSC timer, 18.6 nsec, but has greater overhead for a system call, 119 nsec. Virtual Box (figure 1 (c)) has a greater median overhead in comparison to Xen virtualized OS and host OS – 2.12 µsec for TSC timer and almost the same 2.18 µsec for the system call. Additionally, Virtual Box has the largest spread of values, up to 1 msec for the max value, and 1 nsec for the minimum value. On other platforms (Host OS, Xen, ESXi) minimal difference between time cost values from TSC timer is always 1.4 nsec, while here we have 0.29 nsec.

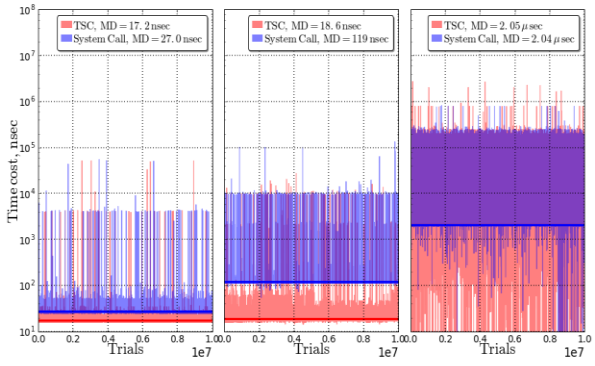

Fig. 1. Measurements of timer cost for the Idle testbed on: (a) Host OS (b) Xen (c) VirtualBox

In Figure 2 the representation of Complementary Cumulative Distribution Function (CCDF) with dashed lines as a means, are showed. Considering the given results, it can be observed that the overhead values which are less than of median values occurs with probability 99%. In idle mode, means and medians have almost the same values. CCDF for Virtual Box has significantly greater tails than for other results, which tells about greater results range in comparison to Host OS and Xen virtualized OS. Considering QEMU CCDF and VMware ESXi, tails are longer on all others test machines, except Virtual Box. Virtual Box tails end on a value up to 10 times greater than all others. These observation allow to assume that Virtual Box uses different way to obtain TSC values.

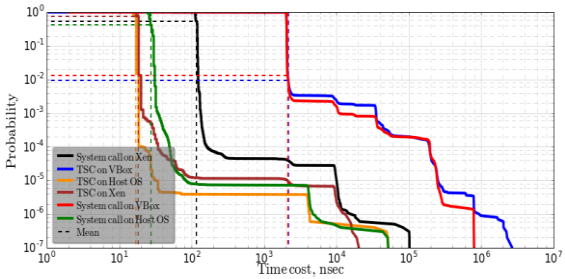

Fig. 2 CCDF representation of measurements of timer cost for the Idle testbed

3.2 CPU – bound testbed

Work lateness can be expected due to limited hardware resources in case of CPU background workload. The median value does not extremely differ from the median values in idle mode. The same can be concluded for QEMU and VMware ESXi [1]. However, this type of load causes long tails on CCDF plot (figure 4), which shows significant influence on mean value. The probability of occurrences of value less then median value and is still 99%. As expected, the best performance achieved on Host OS and the worst one is on Virtual Box virtualized OS. As for CCDF of QEMU and VMware ESXi, their results are mostly repeated results for Host OS, and not greater than System Call for Xen virtualized OS [1]. Virtual Box still shows worst results.

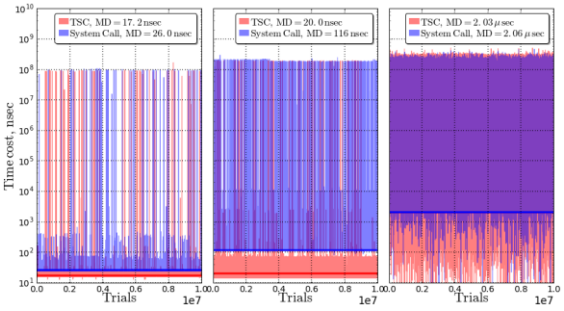

Fig. 3 Measurements of timer cost for the CPU testbed on: (a) Host OS (b) Xen (c) VirtualBox

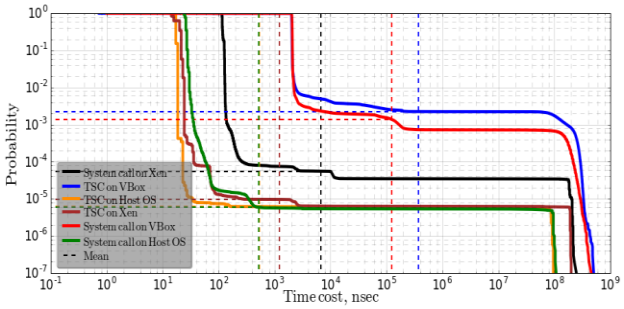

Fig. 4 CCDF representation of measurements of timer cost for the CPU testbed

3.3 I/O – bound testbed

Under a load of interrupt handling caused by IO-bound tasks, the median values are still staying without significant changes.

The range of values is slightly higher than in idle mode. At the same time, median values for TSC on QEMU and on VMware ESXi are 15.8 nsec and 21.5 nsec respectively. For the case of System call on QEMU, the median is 60 nsec, on VMware ESXi it is 34 nsec.

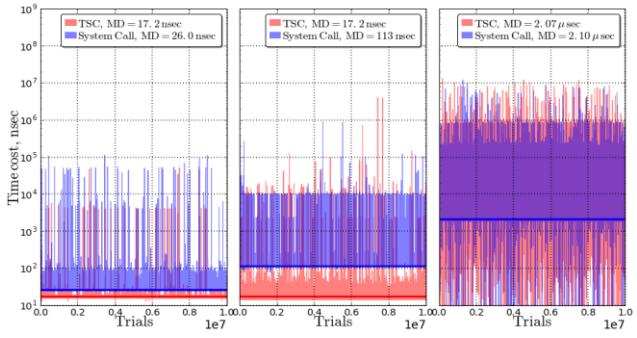

Fig. 5 Measurements of timer cost for the I/O testbed on: (a) Host OS (b) Xen (c) VirtualBox

CCDF plots show longer tails, but the mean value is almost the same as in idle mode, which tells that the probability of obtaining results is greater than median value is extremely small.

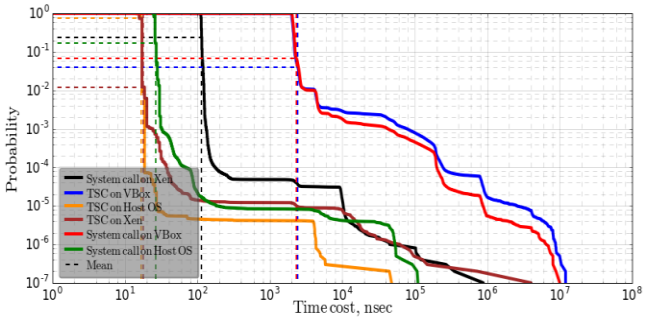

Fig. 6 CCDF representation of measurements of timer cost for the I/O testbed

3.5 Network – bound testbed

Median values for this type of load keeps the same. But the values are changing frequently during the test. That is caused by workflow of load type.

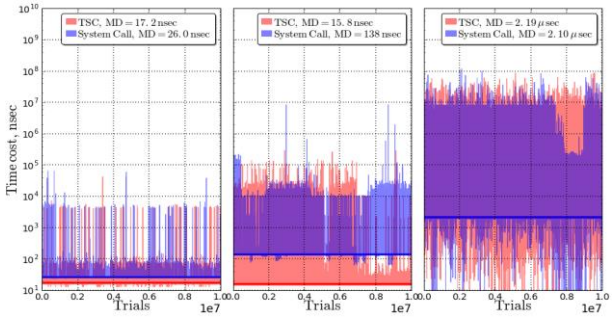

Fig. 7 Measurements of timer cost for the Network (RMDT download) testbed on: (a) Host OS (b) Xen (c) VirtualBox

On CCDF plot, represented on figure 10, the tail length remains almost the same as in idle case only for Host OS. For Xen and Virtual Box tails are much longer. At the same time, probability of obtaining time cost value greater than median value is greater for all cases but still median value occurs with probability around 99%.

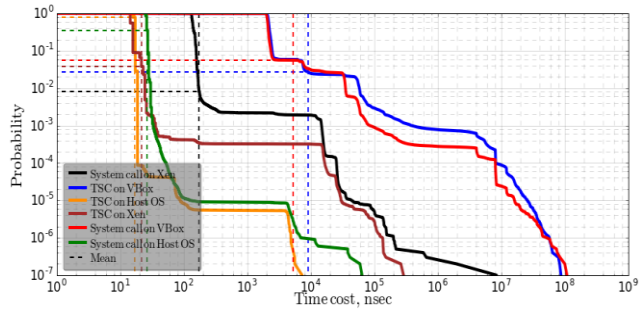

Fig. 8 CCDF representation of measurements of timer cost for the Network (RMDT download) testbed

4 Experimental results for Sleep Measurements

Sleep measurements are based on the comparison of the sleep function of HighPerTimer library [5] and the usleep() function from the standard C library in three scenarios: Host OS, Xen and Virtual Box. Further experiments with the sleep operation from HighPerTimer library are called HPTSleep and experiments with C library are called uSleep.

4.1 Idle testbed

The results for the idle mode with sleep operations are shown on figure 9. Median values for Host OS and Xen virtualized OS for HPTSleep are almost the same, while median of HPTSleep for Virtual Box is significantly greater. In the same time median values for uSleep for all cases are in the same range – between 62 μsec and 131 μsec. The difference of median magnitude between

uSleep and HPTSleep is the greatest for Xen: 94.5 nsec for HPTSleep and 109 µsec for uSleep. For Host OS difference do not significantly differ from Xen: 82 nsec for HPTSleep against 62 µsec for uSleep. For Virtual Box the difference between HPTSleep and uSleep is the least: 5.19 µsec for HPTSleep and 131 µsec for uSleep. The values within the same testbed for QEMU are 97 µsec for uSleep and 116 nsec for HPTSleep, for VMware ESXi – 63.1 µsec for uSleep and 95 nsec for HPTSleep. So Xen results are almost the same as for QEMU. System call for Xen is almost twice greater than same value for VMware ESXi while HPTSleep remains almost the same [1].

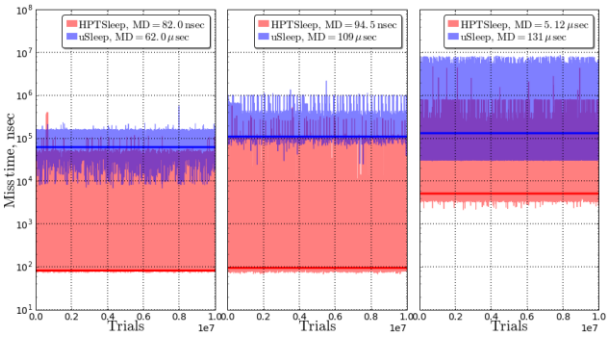

Fig. 9 Measurements of miss time for the idle testbed on: (a) Host OS (b) Xen (c) VirtualBox

Considering the CCDF plots on figure 10, measurements performed with uSleep method are grouped in the right part of the plot. However, HPTSleep for Xen and Host OS are situated in the left part of the plot. HPTSleep for Virtual Box keeps to the right, because of the higher order of values in comparison to Host OS and Xen. QEMU and VMware ESXi behave almost the same as Host OS and Xen.

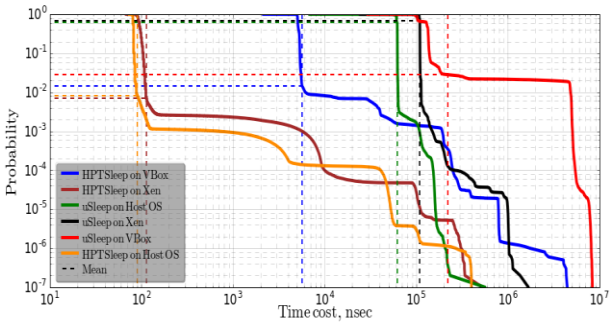

Fig. 10 CCDF representation of measurements of miss time for the idle testbed

4.2 CPU – bound testbed

In the case of CPU load, median (fig. 11) value of uSleep for Virtual Box is greater then in Ideal case: 7.6 msec for CPU load and 131 µsec for Idle case. Median values for all other cases remain the same as for Idle case. Considering QEMU and ESXi [7], System call values are 61.3 µsec and 56 µsec and HPTSleep values are 104 nsec and 114 nsec correspondingly. So, uSleep for Xen shows almost twice greater overhead than for QEMU and VMware ESXi, while HPTSleep value is almost the same.

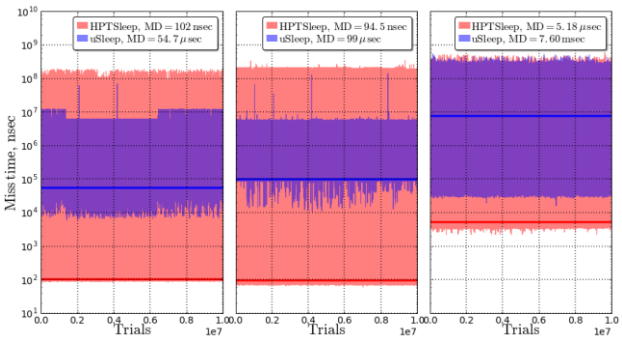

Fig. 11 Measurements of miss time for the CPU-bound testbed on: (a) Host OS (b) Xen (c) VirtualBox

CCDF plot (fig. 12) shows that the tails are significantly increased in comparison to Idle mode. The mean values for HPTSleep of Host OS and Xen are extremely different from mean values of Idle case. Other mean values are increased as well, but not so significantly. System call for Virtual Box differs in comparison to other plots: the plot has a slight structure representation, but not the stepwise. In the same time the probability of obtaining value greater then mean value is much greater then for Ideal case. But still the probability of obtaining value less then median value is about 99%. The behavior of QEMU and VMware ESXi almost repeat the behavior of Host OS [7].

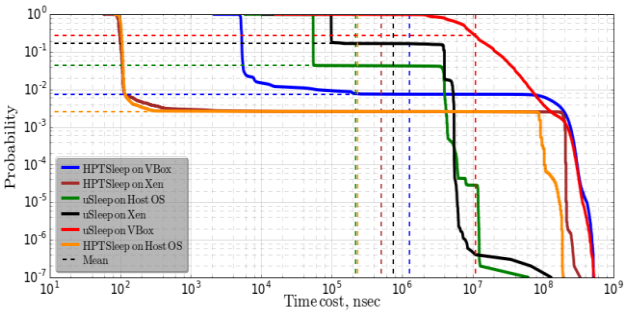

Fig. 12 CCDF representation of measurements of miss time for the CPU-bound testbed

4.3 I/O – bound testbed

In comparison to idle mode, median values remain the same while the range of values is greater. The values of QEMU for HPTSleep and uSleep are 113 nsec and 71.1 µsec and for VMware ESXi are 117 nsec and 62.2 µsec correspondingly. Accordingly, Xen introduces extra time overhead in comparison to QEMU and ESXi.

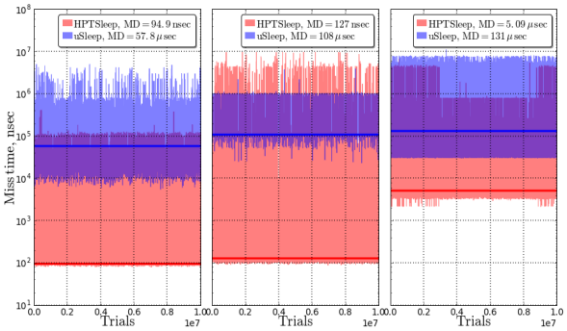

Fig. 13 Measurements of miss time for the IO-bound testbed on: (a) Host OS (b) Xen (c) VirtualBox

CCDF plot shows (fig. 14) that HPTSleep for Host OS is almost the same but probability of getting value greater then minimal is greater then in Ideal case but still not significantly. The mean value a bit greater. The tail of HPTSleep for Xen is much greater than in the idle state. Mean value is greater with less probability of getting mean value then in Ideal case. For all the other cases, mean value remains the same as for Ideal case with not significantly changed tails.

4.4 Network – bound testbed

Observing the influence of network load on sleep operations on figure 16, the median values for this type of load remains approximately the same as for the previous scenario.

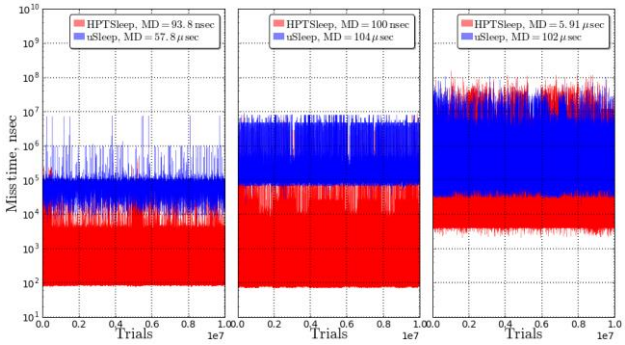

Fig. 14 CCDF representation of measurements of miss time for the Network-bound (RMDT download) testbed

Taking into account CCDF plot (figure 16), for HPTSleep on Xen and Virtual Box, tails are much longer and mean values become significantly greater in comparison to Idle testbed. Probability of obtaining value is greater then mean value is much greater then in Ideal case, but still for values less then median value is around 99%. HPTSleep for Host OS keeps without significant changes. Mean value of uSleep for all cases remains in the same order but tails become significantly longer. The probability for obtaining mean value is greater then in the Idle case. The probability for getting value is less then median value and remains around 99%.

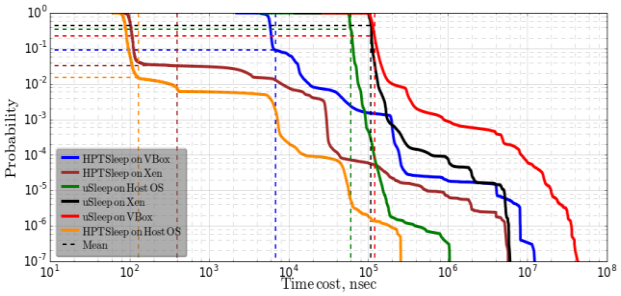

Fig. 15 CCDF representation of measurements of miss time for the Network-bound (RMDT download) testbed

5 Conclusion

Various measurement results for the case, when one virtual instance is located on hardware, allow to conclude the following: (i) While Xen has almost no basis overhead, Virtual Box's access to the hardware adds the highest overhead among all virtaulization platforms (QEMU, Xen, VMware ESXi). (ii) In the case of sleep accuracy, the means of sleeps methods prevail over the virtualization environment effects, except Virtual Box, when miss time median values are 100 times higher then other virtualization in case of HighPerTimer sleep. (iii) Considering the worst-case execution time, the used virtualization plays almost neglecting role, the majority of values are measurably better on host machine then (at least) on QEMU; (iv) hypervisor-based approach brings clearly measurable benefits against OS-based one. (v) Virtual Box has the highest median time overheads and the highest deviation. Also based on minimal overhead values it's make sense to assume that this platform does not provide direct access to hardware. Since that this platform is not recommended for time-critical applications.

Acknowledgement

This work has been funded by Volkswagen Foundation for trilateral partnership between scholars and scientists from Ukraine, Russia and Germany within the project CloudBDT: Algorithms and Methods for Big Data Transport in Cloud Environments.

References

1. K. Karpov, I. Fedotova and E. Siemens, *"Impact of Machine Virtualization on Timing Precision for Performance-critical Tasks"* J. of Physics: Conference Series, **870** (2017)
2. K. Karpov, I. Fedotova, D. Kachan, V. Kirova and E. Siemens, *"Impact of virtualization on timing precision under stressful network conditions,"* Smart Technologies, IEEE EUROCON 2017-17th International Conference on. IEEE, (2017)

3. S. G. Langer, T. French, "*Virtual Machine performance benchmarking*", Journal of digital imaging, **24**, 883 (2011)
4. P. Barham et al. *"Xen and the art of virtualization"*, ACM SIGOPS operating systems review, **37**, 164 (2003)
5. I. Fedotova, E. Siemens, and H. Hu, "*A high-precision time handling library"*, Journal of Communication and Computer, **10**, 1076 (2013)
6. I. Fedotova and E. Siemens, *"High-precision process sleeping aspects in the time handling library HighPerTimer for Linux OS"*, The Forth Scientific and Practical Conference "Information and Measuring Equipment and Technology", (2013)