Ulrich Lampe, Markus Kieselmann, André Miede, Sebastian Zöller, Ralf Steinmetz: **A Tale of Millis and Nanos: Time Measurements in Virtual and Physical Machines.** In: Kung-Kiu Lau, Winfried Lamersdorf, Ernesto Pimentel: Proceedings of the 2nd European Conference on Service-Oriented and Cloud Computing (ESOCC 2013), vol. LNCS 8135, p. 172-179, Springer-Verlag, September 2013. ISBN 978-3-642-40650-8.

A Tale of Millis and Nanos: Time Measurements in Virtual and Physical Machines

Ulrich Lampe[1], Markus Kieselmann[1], André Miede[2], Sebastian Zöller[1] and Ralf Steinmetz[1]

[1] Multimedia Communications Lab (KOM), TU Darmstadt, Germany
{firstName.lastName}@KOM.tu-darmstadt.de
http://www.kom.tu-darmstadt.de/

[2] Fakultät für Ingenieurwissenschaften, HTW des Saarlandes, Saarbrücken, Germany
andre.miede@htw-saarland.de
http://www.htw-saarland.de/

Abstract. Cloud computing makes large infrastructure capacities available to users in a flexible and affordable fashion, which is of specific interest to scientists for conducting experiments. Unfortunately, our past research has provided first indications that virtual machines – the most popular type of cloud-based infrastructure – have substantial deficits with respect to time measurements, which are an important tool for researchers. In this paper, we provide a detailed analysis on the accuracy of time measurements based on various machine configurations. They cover influence factors such as machine type, virtualization solution, and programming language. The results indicate that not the use of virtualization as such, but the potentially uncontrollable utilization of the physical host is a decisive factor for the accuracy of time measurements. Different virtualization solutions and programming languages play an inferior role. Our findings, along with the publicly released tool *TiMeAcE.KOM*, can provide a valuable decision support for researchers in the selection and configuration of cloud-based experimental infrastructures.

Keywords: cloud computing; infrastructure; virtual machine; experiment; time measurement; accuracy; timeace

1 Introduction

A key feature of cloud computing is elasticity, i. e., the ability to access Information Technology (IT) resources in a flexible and affordable fashion [1]. Apart from small and medium enterprises, this characteristic is specifically relevant for researchers, who frequently require large capacities on short term in order to conduct scientific experiments. In this context, Infrastructure as a Service (IaaS) offers are of specific interest. They provide flexible environments, i. e., Virtual Machine (VM) instances, which permit the execution of practically any existing software without major adaptation [2].

Unfortunately, our past work has provided initial indications that VMs suffer from deficits with respect to the accuracy of time measurements [3]. This is

problematic in so far as time measurements are an important tool in scientific research, e. g., in the comparative evaluation of exact and heuristic optimization approaches [4]. In this paper, we substantially extend our past research through the consideration of additional influence factors, e. g., different virtualization solutions and programming languages. This work also extends a recently published work-in-progress paper [5], which is based on similar results, through the inclusion of statistical test results and an overview of related work.

The remainder of this paper is structured as follows: Section 2 describes our experimental design and setup. The results and practical conclusions are described in Section 3. Subsequently, Section 4 provides an overview of related work. Section 5 concludes the paper with a summary and outlook.

2 Experimental Setup

2.1 Measurement Tool

In this work, we pursue the same principal experimental approach as in our past research [3]: We repeatedly measure the computation time of a deterministic function in order to quantify potential inaccuracies in time measurement. Deterministic, in this context, means that the function exhibits the same *computational* complexity for a given input parameter. Thus, the observed computation time for each execution should also be identical under ideal conditions. Accordingly, variations in the computation times can directly be related to measurement inaccuracies.

For our experiments, we have implemented a measurement tool, which features a simple counter function as its core component. The function accepts a single integer a as argument and returns the required computation time as result. The tool can be configured to conduct a series of $b \in \mathbb{N}$ batches. Each batch comprises $c \in \mathbb{N}$ calls of the aforementioned counter function, using the arguments $a \in A = \{2^0, 2^1, \ldots, 2^m\}$, where $m \in \mathbb{N}$. The tool automatically adapts the given argument through multiplication by a so-called *machine speed index*. This index is initially determined by the program and ensures that for a given argument a, the computation time is approximately $a \times 10$ ms, regardless of the underlying processor. This guarantees that the observed runtimes feature roughly the same *absolute* values for identical arguments.

2.2 Experimental Configurations

The aim of our work is to quantify the impact of different potential influence factors on the accuracy of time measurements, which constitutes the *dependent variable* in our experiments. Thus, we employ a multitude of different *machine configurations* in our experiments, where the influence factors are modeled through five *independent variables*.

The first independent variable of interest is the *machine type*, with respect to which we distinguish two options. As previously outlined, *VMs* are the most

common form of IaaS, and are commercially offered based on flexible pay-as-you-go pricing models today. In contrast, Physical Machines (*PMs*) represent traditional, dedicated experimental infrastructure. The *deployment model* is the second independent variable in our experiments. Specifically, we consider VMs from a *public cloud* (Amazon EC2) and a *private cloud* that is operated on the basis of multiple IBM Blade servers at our research lab (KOM). In addition, we made VMs available using a *local host* computer. As third independent variable, we regard the *virtualization software*. Concerning this factor, we distinguish between *ESXi*, a solution that is commercially marketed by VMware, and *Xen*, an open-source software that forms the basis for Amazon's Elastic Compute Cloud (EC2). As fourth independent variable, we consider the *host utilization*, i. e., computational load that the PM or host system for the VMs is subjected to. Concerning this factor, we distinguish between three options. In the case of *low load*, the PM exclusively hosts one instance of the measurement tool or VM. In the case of *high load*, the system runs multiple tool instances or VMs in parallel. Lastly, in the case of *random load*, the host utilization is out of our control sphere, and potentially fluctuates during the experiments. As fifth and final independent variable, we regard the *programming language*. For that purpose, we have implemented the measurement tool in similar form in *C* and *Java*. This choice was made because Java as such uses a form of virtualization, the so-called Java VM, which may potentially influence the measurement accuracy independent of the underlying infrastructure. In contrast, C does not feature such a concept.

2.3 Measurement Procedure

In principal, we follow a *full-factorial approach* in our experiments. That is, we examine each possible combination of values for the five independent variables, i. e., influence factors, that were introduced in the previous section. However, as can easily be reasoned, some combinations are mutually exclusive: For example, Amazon does not provide a choice between different virtualization systems, but uses Xen as standard solution. Nevertheless, our experiments encompass a total of 16 different machine configurations, which should provide a comprehensive overview of different influence factors.

As PM and local host for the VMs, we used a desktop computer, equipped with an Intel Core 2 Duo processor at 2.0 GHz and 2 GB of memory. Given that our previous research showed no major differences between Linux and Windows concerning the accuracy of time measurements [3], we exclusively employed the former as guest operating system in our experiments. Specifically, we chose Ubuntu Server 12.04.1 LTS, which was booted into the default text-based shell to minimize the influence of background services. In order to generate high load for the corresponding configurations, we either launched three parallel VM instances or measurement tool instances on the physical host.

For every configuration, we conducted 20 experimental batches with 100 method calls each (i. e., $b = 20$, $c = 100$). The set of applicable arguments was specified as $A = \{2^0 = 1, ..., 2^9 = 512\}$, i. e., $m = 9$. Thus, we obtained a

total *sample* of 20,000 runtime observations per configuration, with *subsamples* of 2,000 observations per argument and machine configuration. In total, across all 16 configurations, 320,000 individual observations were collected.

3 Experimental Results and Practical Recommendations

In accordance with our previous work [3], we use the normalized standard deviation, i. e., the *Coefficient of Variation* (CV), as measure of accuracy. It is given by the ratio between the standard deviation (commonly denoted as σ) and the mean value of the observations (μ) in a sample [6]. The CV numerically represents the dependent variable in our experiments. Due to the definition of the CV, *higher* values indicate *lower* accuracy and vice versa; hence, in the case of ideal accuracy, the observed CV would correspond to zero.

A comprehensive overview of all machine configurations, along with the CVs that were observed of each argument of the counter function, is given in Table 1 in the appendix. The table also provides the relative rank for each configuration with respect to the observed accuracy per argument.

Given the findings of our previous work [3], which indicated general deficits of VMs with respect to time measurements, our new experiments provide some surprises. Specifically, the VMs from the *private* cloud at our institute provide the best accuracy for small arguments, i. e., $a \leq 2$, among all tested configurations (cf. #13 and 14 in Table 1). For increasing arguments, the VMs lose some ground to the PMs, specifically when the Java-based implementation of the measurement tool is used (cf. #2 in Table 1). Nevertheless, a *Friedman test* at the common confidence level of 95% shows no significant difference between the PM-based configurations and the configurations that used VMs from our cloud ($p = 0.5034$).

Yet, the results confirm the deficits of *public* clouds with respect to time measurements. Notably, the VM instances from Amazon EC2 exhibit the highest CVs, i. e., lowest accuracy, for most arguments, specifically those in the subsecond range (cf. #15 and 16 in Table 1). Correspondingly, the Friedman tests show that VMs from the public cloud perform significantly worse compared to PMs with low utilization ($p = 0.0000$). However, in comparison to a PM under high load, the Friedman test shows no significant difference ($p = 0.2632$).

Concerning the two virtualization solutions, ESXi and Xen, we obtained mixed results. On the basis of the locally hosted VMs and low utilization, the observed CVs indicate some advantages for ESXi with respect to small arguments (i. e., $a \leq 2$), while the relative performance of Xen improves with growing arguments (cf. #5, 6, 9, and 10 in Table 1). In addition, Xen achieves more favorable accuracy once high host utilization comes into play; in this case, ESXi generally appears to perform very poorly (cf. #7 and 8 in Table 1). Correspondingly, a Friedman test indicates a significant difference between both solutions and confirms a superior measurement accuracy for Xen across the considered programming languages and host utilization ($p = 0.0038$).

From the above discussion, one may conclude that the host utilization plays a key role in the accuracy of measurements, and this is strikingly confirmed in our experiments. Regardless of the machine type and virtualization software, imposing additional load on the physical host results in sharp increases in the observed CVs (cf. #7, 8, 11, and 12 in Table 1). The same applies for the PM (cf. #3 and 4 in Table 1). Accordingly, the Friedman tests confirm the role of the host utilization as decisive factor in measurement accuracy, both for PMs and VMs ($p = 0.0000$ in both cases).

Concerning the impact of the programming language, we observe very mixed results. Neither C nor Java consistently achieves higher accuracy across all considered machine configurations (cf., for example, #1, 2, 15, and 16 in Table 1). In accordance, a Friedman test indicates no significant differences between both programming languages at a 95% confidence level ($p = 0.8701$).

In conclusion, the experimental results in this paper – to some extent – relativize the preliminary findings of our previous work: Most notably, we have found that contemporary virtualization technology as such does *not* necessarily imply deficits with respect to the accuracy of time measurements. In fact, the lowest CVs, i. e., best accuracies, among all machine configurations in our experiments were observed on VM instances from a private cloud. Likewise, those VMs that were hosted on a single physical host performed very similarly to a "raw" PM.

Our experiments have shown that a different influence factor, namely host utilization, is the key determinant for time measurement inaccuracies. Unfortunately, this is the very factor that commonly lies out of the control sphere of the end user when leasing resources from a public cloud; in fact, from the viewpoint of the cloud provider, the consolidation of multiple VMs on a single physical host is highly desirable in order to reduce operational cost. The same also applies to a private cloud in principal, even though the level of control may be higher for the end user in such deployment model. To state it more explicitly, virtualization does not hurt the accuracy time measurement, but high host utilization – which is a key benefit of virtualization – does.

Hence, our results confirm the most important recommendation of our previous work: If accurate time measurements, specifically in the sub-second range, are required in scientific experiments, dedicated PMs should be preferred over VMs. Yet, if host utilization as the key influence factor can be effectively controlled by the end user, VMs may also provide acceptable accuracy. In this context, dedicated VM instances with performance guarantees – which have recently appeared in the public cloud market – could be of interest as well.

In order to help scientists in the assessment of experimental infrastructures, we have created a lightweight tool called *Time Measurement Accuracy Estimation* (TiMeAcE.KOM). This tool, which is available through our Web site[3], automatically conducts a small set of measurements using a simple counter function, and provides a textual assessment of measurement accuracies.

[3] `http://www.kom.tu-darmstadt.de/timeace/`

4 Related Work

To the best of our knowledge, our previous work [3] and the present paper is the only research that specifically examines the accuracy of time measurements in physical and virtual environments. However, with the renewed interest in virtualization technology and the hype around cloud computing, various research efforts have been undertaken in related fields recently.

In this context, timekeeping on VMs is the first major area of interest. A comprehensive overview of this topic has been provided in a whitepaper by VMware [7]. The authors provide an extensive background on timekeeping mechanisms on PMs. Based on this, they outline different options for timekeeping in virtualized environments and also provide hints for improving timekeeping accuracy. A specific proposal for improving the timekeeping VMs in Xen has been made by Chen et al. [8]. Their approach, called XenHVMAcct, aims to provide the same accuracy in hardware-assisted VMs, which use an unmodified guest operating system, as in para-virtualized VMs that rely on modified systems. Broomhead et al. [9] also introduce an improved timekeeping mechanism in the context of Xen. Their work specifically targets clock inaccuracies that are introduced by live migration operations.

The second notable area of related research concerns performance evaluations in cloud computing environments. El-Khamra et al. [10], for example, have examined the runtime fluctuations of a scientific workflow in FutureGrid, a scientific Grid testbed, and the commercial Amazon EC2 cloud. The authors also find relatively large variations in runtime, but attribute them to performance fluctuations, rather than timekeeping deficits. Schad et al. [11] have conducted a longitudinal study of performance variations in Amazon EC2 using a suite of benchmarks. They find substantial fluctuations in the performance of different system components, such as processor and network, and conclude that the conduction of performance experiments on leased VMs can be problematic. However, Schad et al. do not take the potential systematic weaknesses of time measurements in virtualized environments into account either.

5 Summary and Outlook

Commercial cloud providers make large pools of compute capacity available to end users based on a pay-as-you-go scheme. This is of specific interest to researchers, who can exploit VM instances to conduct scientific experiments. However, past research has indicated that VMs suffer from inaccuracy when it comes to time measurements, which are a common instrument in science, e. g., in the assessment of heuristic optimization approaches. Based on this notion, this work provided an extensive analysis concerning the accuracy of time measurements depending on different influence factors, namely machine type, deployment model, virtualization software, host utilization, and programming language.

We found that the machine type, i. e., the use of virtualization as such, is *not* a key determinant of time measurement inaccuracies; instead, the utilization

of the physical host plays a decisive role. According to our observations, a high degree of load on the physical host – as it can likely be expected in cloud data centers due to the use of consolidation techniques – results in dramatic loss of accuracy. Furthermore, we concluded that the virtualization software Xen has small advantages over ESXi. For the two considered programming languages, C and Java, we observed no statistically significant results with respect to time measurement accuracy. Based on our findings, we recommend scientists to either use PMs or VMs from a controlled environment if accurate time measurements, specifically in the sub-second range, are required.

For the future, we plan to extend our existing work through a longitudinal (i. e., long-term) study with different commercial cloud providers. With such design, we expect to identify the impact of potential performance fluctuations on the accuracy of time measurements. Furthermore, these additional experiments may permit fellow scientists to make a more educated decision among competing cloud offers.

Acknowledgments

This work has been sponsored in part by the E-Finance Lab e. V., Frankfurt a. M., Germany (`www.efinancelab.de`).

References

1. Owens, D.: Securing Elasticity in the Cloud. Comm. of the ACM **53**(6) (2010) 46–51
2. Briscoe, G., Marinos, A.: Digital Ecosystems in the Clouds: Towards Community Cloud Computing. In: Proc. of DEST 2009. (2009)
3. Lampe, U., Miede, A., Richerzhagen, N., Schuller, D., Steinmetz, R.: The Virtual Margin of Error – On the Limits of Virtual Machines in Scientific Research. In: Proc. of CLOSER 2012. (2012)
4. Silver, E.: An Overview of Heuristic Solution Methods. J. of the Operational Research Society **55**(9) (2004) 936–956
5. Lampe, U., Kieselmann, M., Miede, A., Zöller, S., Steinmetz, R.: On the Accuracy of Time Measurements in Virtual Machines. In: Proc. of CLOUD 2013. (2013)
6. Jain, R.K.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley (1991)
7. VMware, Inc.: Timekeeping in VMware Virtual Machines. `http://www.vmware.com/files/pdf/techpaper/Timekeeping-In-VirtualMachines.pdf` (2011)
8. Chen, H., Jin, H., Hu, K.: XenHVMAcct: Accurate CPU Time Accounting for Hardware-Assisted Virtual Machine. In: Proc. of PDCAT 2010. (2010)
9. Broomhead, T., Cremean, L., Ridoux, J., Veitch, D.: Virtualize Everything But Time. In: Proc. of OSDI 2010. (2010)
10. El-Khamra, Y., Kim, H., Jha, S., Parashar, M.: Exploring the Performance Fluctuations of HPC Workloads on Clouds. In: Proc. of CloudCom 2010. (2010)
11. Schad, J., Dittrich, J., Quiané-Ruiz, J.: Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. Proc. of the VLDB Endowment **3**(1-2) (2010) 460–471

All online references in this paper were last accessed and validated in June 2013.

Table 1. Observed time measurement accuracies, i.e., *coefficients of variation*, by machine configuration. Values in parentheses denote the rank among all configurations for the given argument, ordered from most accurate (1) to least accurate (16). Absolute runtimes for the counter function approximately correspond to $a \times 10$ ms. Abbreviations: M/T (Machine Type), D/M (Deployment Model), V/S (Virtualization Software), H/U (Host Utilization), P/L (Programming Language).

#	Machine Configuration					Function Argument									
	M/T	D/M	V/S	H/U	P/L	$a = 1$	$a = 2$	$a = 4$	$a = 8$	$a = 16$	$a = 32$	$a = 64$	$a = 128$	$a = 256$	$a = 512$
1	PM	n/a	n/a	Low	C	0.0123 (3)	0.0090 (3)	0.0065 (5)	0.0048 (5)	0.0034 (5)	0.0025 (5)	0.0018 (5)	0.0014 (5)	0.0011 (3)	0.0008 (5)
2	PM	n/a	n/a	Low	Java	0.0328 (6)	0.0403 (6)	0.0005 (1)	0.0003 (1)	0.0002 (1)	0.0003 (2)	0.0001 (1)	0.0001 (2)	0.0000 (1)	0.0000 (1)
3	PM	n/a	n/a	High	C	0.3990 (9)	0.2213 (9)	0.1304 (9)	0.1205 (9)	0.1352 (12)	0.1306 (13)	0.1205 (13)	0.1061 (15)	0.0965 (15)	0.0823 (15)
4	PM	n/a	n/a	High	Java	0.5527 (10)	0.3876 (10)	0.3152 (10)	0.2814 (12)	0.2501 (14)	0.2290 (14)	0.2078 (16)	0.1795 (16)	0.1667 (16)	0.1496 (16)
5	VM	Local	ESXi	Low	C	0.0172 (4)	0.0096 (4)	0.0078 (6)	0.0065 (6)	0.0105 (7)	0.0067 (7)	0.0063 (6)	0.0127 (8)	0.0040 (6)	0.0044 (7)
6	VM	Local	ESXi	Low	Java	0.0290 (5)	0.0203 (5)	0.0226 (7)	0.0085 (7)	0.0057 (6)	0.0045 (6)	0.0195 (8)	0.0086 (6)	0.0048 (7)	0.0033 (6)
7	VM	Local	ESXi	High	C	1.5958 (14)	1.1553 (15)	0.8151 (15)	0.6014 (16)	0.4440 (15)	0.3063 (15)	0.1725 (14)	0.0789 (13)	0.0406 (13)	0.0211 (13)
8	VM	Local	ESXi	High	Java	1.6590 (15)	1.2490 (16)	0.8356 (16)	0.5877 (15)	0.4783 (16)	0.3420 (16)	0.1934 (15)	0.0822 (14)	0.0424 (14)	0.0234 (14)
9	VM	Local	Xen	Low	C	0.0892 (7)	0.0677 (7)	0.0494 (8)	0.0346 (8)	0.0259 (8)	0.0181 (8)	0.0132 (7)	0.0104 (7)	0.0081 (8)	0.0066 (10)
10	VM	Local	Xen	Low	Java	0.2463 (8)	0.0733 (8)	0.0029 (3)	0.0005 (2)	0.0002 (2)	0.0001 (1)	0.0001 (2)	0.0001 (1)	0.0002 (2)	0.0002 (2)
11	VM	Local	Xen	High	C	1.1925 (13)	0.7790 (13)	0.5021 (13)	0.2873 (13)	0.1266 (11)	0.0681 (11)	0.0361 (11)	0.0198 (12)	0.0115 (11)	0.0077 (12)
12	VM	Local	Xen	High	Java	1.0830 (11)	0.7112 (12)	0.4449 (12)	0.2489 (11)	0.1135 (10)	0.0597 (10)	0.0296 (10)	0.0157 (9)	0.0090 (9)	0.0055 (8)
13	VM	Private	ESXi	Low	C	0.0028 (1)	0.0023 (1)	0.0027 (2)	0.0021 (3)	0.0018 (3)	0.0014 (3)	0.0011 (3)	0.0008 (3)	0.0013 (5)	0.0005 (3)
14	VM	Private	ESXi	Low	Java	0.0118 (2)	0.0074 (2)	0.0029 (4)	0.0022 (4)	0.0022 (4)	0.0017 (4)	0.0017 (4)	0.0010 (4)	0.0013 (4)	0.0006 (4)
15	VM	Public	Xen	Random	C	1.7477 (16)	1.0945 (14)	0.6565 (14)	0.3150 (14)	0.1503 (13)	0.0886 (12)	0.0405 (12)	0.0196 (11)	0.0118 (12)	0.0061 (9)
16	VM	Public	Xen	Random	Java	1.1398 (12)	0.6293 (11)	0.3576 (11)	0.1783 (10)	0.0974 (9)	0.0499 (9)	0.0271 (9)	0.0159 (10)	0.0108 (10)	0.0075 (11)