Operating System Noise: Linux vs. Microkernel

Stefan Wächtler
Technische Universität Dresden
Operating-Systems Group
Nöthnitzer Strasse 46, Dresden, Germany
Stefan.Waechtler@mailbox.tu-dresden.de

Michal Sojka
Czech Technical University in Prague
Technická 2, 121 35 Praha 6, Czech Republic
sojkam1@fel.cvut.cz

Abstract

Computers are often used as a tool to measure time in various experiments. Benchmarking and hardware performance evaluation are very common examples. There one typically needs to measure the duration of certain computer operation. The precision of hardware clocks is typically more than satisfying for this purpose but the measured time is also influenced by other factors such as the used operating system. In this paper, we evaluate how does the variance of benchmark results depend on the operating system.

We developed a set of benchmarks to measure the characteristics of memory subsystem, the cost of task preemption and the cost of task migration between CPU cores/sockets. We run the benchmarks on NOVA microkernel as well as on two different configurations of Linux, in all cases on x86 architecture. We captured the data from several tenths of hours of running experiments and compare the obtained results. The mean measured values are in all cases similar. Besides the mean values, we compare the variance of subsequent runs of the benchmark. The results vary depending on the experiment, but there is a clearly visible trend in all experiments – the noise generated by the microkernel is in most cases the lowest, Debian/Linux sits in the middle and the highest noise is produced by a minimal configuration of the Linux kernel.

1 Introduction

Computers running Linux are often used as a tool to measure time in various experiments. Perhaps the most common use case is benchmarking and other performance related measurements. The precision of hardware clocks is typically more than satisfying for this purpose but the measured time is also influenced by other factors such as the used operating system.

The reason is that besides running the measurement application itself, the operating system performs additional activities, that have the impact on the measured time. For example, when one measures the time needed to execute a certain piece of code, the resulting time also includes the time spent by the OS in interrupt handling. Therefore the OS can be seen as a source of noise influencing the measurement.

OS noise is a well known term in high-performance computing (HPC), where it is regarded as a source of scalability problems [1, 2, 3]. HPC applications typically run multiple jobs on dedicated CPUs in parallel. After the jobs are finished, their results are collected. When some jobs are finished later than the others, the other job's CPUs have to wait for the late jobs to complete, wasting their computational power.

In this paper we look at the OS noise from different perspective. We are interested in seeing how it influences the accuracy of time measurements. In the beginning of our effort was a paper by Bastoni et al. [4] where the authors empirically evaluate the cache-related preemption and migration delays. In short, cache-related preemption delay is the time difference in execution times of a task before preemption, when the task's working set is cached in cache memory, and after preemption, during which a part of the working set might have been evicted from the

cache. Cache-related migration delay is defined analogously. The exact meaning of those terms is not important for the purpose of this paper. Here, we treat the various benchmarks for determining the delays under various conditions as black boxes. The focus of this paper is to compare how the resulting numbers differ depending on whether the benchmark is executed on Linux or on microkernel-based OS called NOVA [5].

We ported a set of benchmarks [6] for measuring cache-related preemption and migration delays from NOVA to Linux. Then, we run the same set of benchmarks on the same hardware, but under three different operating systems. One was a minimal configuration of NOVA, the other two were different configurations of Linux. Our expectation was that the noise produced by NOVA will be significantly lower than the noise of Linux. Although this expectation was largely fulfilled, the outcome of our comparison is not so straightforward.

The structure of this paper is as follows. In Section 2, we give a brief overview of NOVA microhypervisor. Section 3 describes our testbed and the benchmark used to measure the noise levels. Section 4 contains the main result of the paper. It explains how we compare the measured noise and summarizes the results from many experiments. We conclude the paper in Section 5.

2 NOVA microhypervisor

NOVA[1] is an open-source hypervisor built on the same principles as microkernel-based operating systems: The kernel contains only the minimal amount of functionality needed for the system to work. In particular, NOVA provides only the following: scheduling, inter-process communication (IPC), memory management and interrupt/exception handling. The rest (such as drivers, virtual machine monitor, etc.) is left to be implemented in the user space. Although NOVA-based system is able to run virtual machines, similarly as Xen[2] or KVM[3], we do not make use of this functionality in this work and instead use NOVA as an ordinary microkernel. The user space used with NOVA in this work is NUL[4]. Currently, NOVA can run on x86 systems only.

NOVA and NUL have several interesting properties that we believe are important for reduction of OS noise. These are:

1. Small memory footprint. The size of uncompressed kernel binary is only 66 KiB. The "hot" code used most often during system execution (IPC) is extremely small – about 1 KiB. This code occupies only a fraction of typical L1 cache memory.
2. Very little kernel code is executed without the user space explicitly asking for it.
3. User space (also very small) can be configured to not run components (drivers, services) that are not needed for the particular application.

The source of OS noise is the situation when the user-space code is preempted by an unrelated activity. Here we list all situations when it can happen under NOVA and how we dealt with that in our benchmark implementation for NOVA:

- Scheduler timer interrupt (local APIC) is used to interrupt the user space task after its time quantum has elapsed. In our benchmark we set the quantum of our tasks to 100 ms.
- Timer service interrupt (HPET). Timer service provides alarm timer functionality. In the whole system, the timers were only used to implement `sleep()` function to simulate the preemption for measurement of cache-related preemption delay.
- Hardware interrupt. In our benchmark, we simply didn't use any hardware besides keyboard and serial line. During the benchmark we neither pressed any keys nor used the serial line, therefore no interrupts were generated.
- Page faults. In case of our benchmark, we allocated and mapped all the needed memory beforehand to prevent page faults from occurring (equivalent of `mlockall()`).
- High priority task. Our benchmark setup did not run any task with higher priority than our measurement task.

When the above mentioned properties are compared with the properties of Linux kernel, Linux is almost opposite of NOVA. It has big memory footprint and lot of activities run in parallel with user applications.

3 Testbed setup & benchmark

All experiments were run on a Dell Precision T7500 system. The system contains two identical Intel

[1] https://github.com/IntelLabs/NOVA, http://hypervisor.org/
[2] http://www.xen.org/
[3] http://www.linux-kvm.org/
[4] http://tudos.org/nul/

Xeon X5650 CPUs comprising of 6 cores per package. One core has two 32 KiB L1 caches, one for instructions and one for data and 256 KiB of unified L2 cache. The L3 cache has the size of 12 MiB and is shared between all cores in the package. The system has 24 GiB of main memory in NUMA configuration. The benchmark only touched the memory local to the first package. However, in some experiments the benchmark was run on a CPU in the "remote" package.

The benchmarking application run in total 46 different experiments. In every experiment, the quantity measured was the time needed for accessing (i.e. reading or writing) a certain amount of memory. In the following we call the size of this memory as *working set size* or WSS in short. Our experiments can be classified into the following groups: cache/memory bandwidth measurement (8 experiments), cache-related preemption delay measurement (32 experiments), cache-related migration delay measurement (6 experiments). Each experiment was run several times in order to calculate the variance of the measured quantity. Most experiments were run 1024 times, migration experiments 512 times and memory bandwidth experiments with modified cache lines 256 times. More detailed description of the experiments can be found in [6].

The benchmark was run under three different operating systems:

1. NOVA: The benchmark was booted over network and the results of the experiments were reported via serial line. No other devices such as disk or network were used.

2. Linux 3.2.13 with minimal configuration. The kernel was configured with minimum features. Some important options were set as follows: `PREEMPT_NONE=y`, `HPET_TIMER=n`, `NO_HZ=y`. The benchmark binaries were put directly into initial ramdisk and the results were sent out via serial console.

3. Standard installation of Debian with Linux kernel 2.6.32-5-686 (as shipped by Debian). The system was booted from disk. Only `sshd` and `openvpn` daemons were run. The results were stored on disk (shell redirection).

4 OS noise comparison

In this section, we compare the noise in measurements from different operating systems.

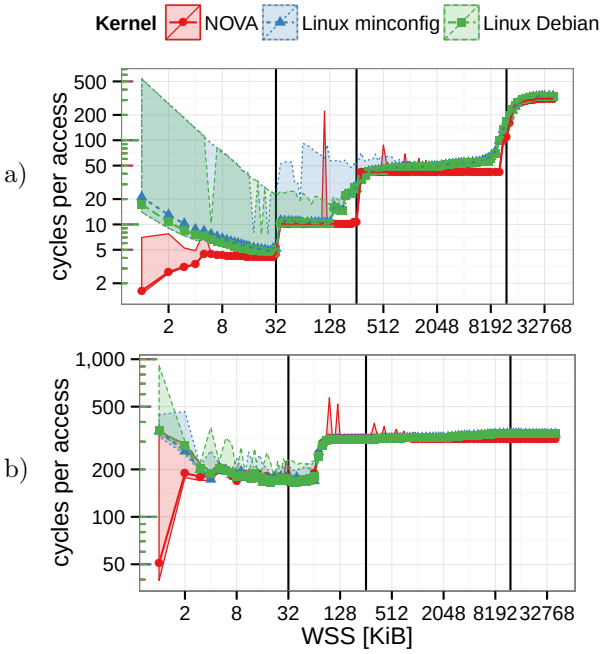

Figure 1: Measured values of CPU cycles per memory access: a) warm cache, b) cold cache. Filled areas represent min/max range. Vertical lines denote the sizes of different cache levels.

The overview of results of two different experiments is shown in Figure 1. The top graph a) shows the number of CPU cycles needed to access the memory in case of warm cache. This is a classical curve familiar to most people that have ever benchmarked memory accesses. The points in the graph represent the mean value of all runs, the shaded areas span between the minimum and maximum measured values. We can observe the following from the graph: (1) Steps on the NOVA curve are much sharper. This is due to smaller cache footprint of NOVA system during the execution of the experiment. (2) The differences between the mean values for the two Linux kernels is very small. (3) The differences at the left end of the graph is caused by inaccuracy of our measurement method. (4) There are big differences in the observed noise (min/max values). NOVA's noise is pretty low with a few exceptions (spikes, see below). Linux noise is bigger and there are differences between the two kernel configurations.

The bottom graph b) in Figure 1 shows again the CPU cycles needed to access the memory but in this case the cache was dirty. The values in the cache had to be written back to the main memory before the cache line could hold the new value requested by the benchmark. Here, we can make similar observations as for the top graph. Only (1) does not apply here, because OS cache footprint is insignificant when the

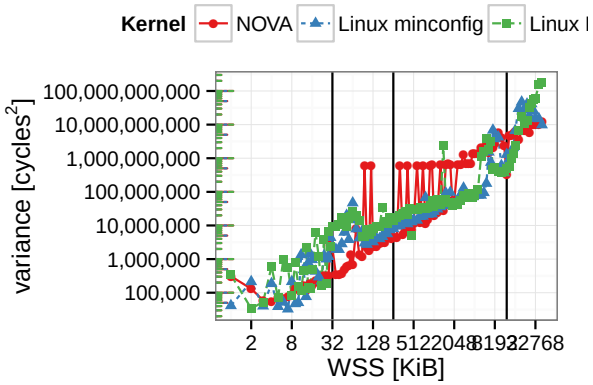

Figure 2: Variance of measured values.

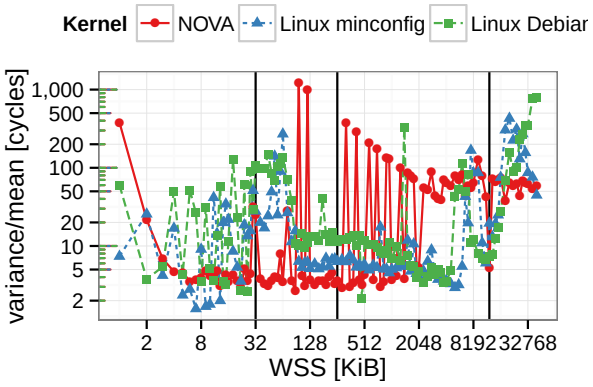

Figure 3: Variance to mean ratio of the same experiment as in Figures 1b and 2.

cache is occupied by something else than the OS.

Both graphs in Figure 1 contain big spikes in maximum values for NOVA. These spikes significantly influence the results at the expense of NOVA. We do not know exactly what is the source of those spikes. We believe that they are caused by system management interrupts (SMI) that does not occur under Linux because Linux initialized some hardware (perhaps USB) that is otherwise handled by BIOS with the help of SMIs. Those spikes does not occur on other systems (AMD Phenom [6]) where we tried to run the benchmark on. Unfortunately, we did not run Linux benchmark on those systems and therefore we cannot use the results from that system in this paper.

To proceed further with our analysis of the noise, we will concentrate on the variance of measured results rather than on the mean value. For most of the experiments, the variance graph looks similarly to the one in Figure 2. It can be seen that variance is more or less proportional to the size of the working set, which is in turn proportional to the measured time. Due the this fact, we use the variance-to-mean

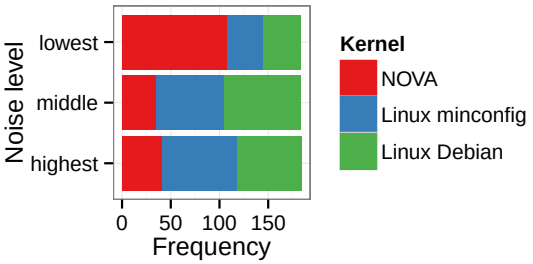

Figure 6: Summary of noise comparison. Frequency values show how many times achieved the particular OS lowest, middle or highest noise level in graphs from Figure 4.

ratio (depicted in Figure 3) rather than the raw values of variance in our comparison. The curves in Figure 3 are more or less flat and it is easier to calculate averages from them.

From the graph in Figure 3, we can see that the character of the curves is different in different WSS regions. For example, the variance-to-mean ratio is much bigger when the working set size is greater than the size of the cache. Due to this fact we process the results as follows. We split the working set sizes to four regions according to the sizes of different caches (in the graph denoted by thick vertical lines). For every such region, we calculate the median value. This process is illustrated in Figure 4, where are raw measured data on the left and the aggregated results (medians) on the right.

We performed the aggregation for all 46 experiments and the results can be seen in Figure 4. This gives us $46 \times 4 = 184$ comparisons, which are summarized in Table 1 and Figure 6.

OS	lowest	middle	highest
NOVA	108 (59%)	35 (19%)	41 (22%)
Linux min.	37 (20%)	70 (38%)	77 (42%)
Linux Debian	39 (21%)	79 (43%)	66 (36%)

Table 1: Summary of noise comparisons.

It can be seen that NOVA offers the lowest noise level (variance-to-mean ratio) in more than half (59%) cases. On the other hand, it exhibits the highest noise level in 22% of cases. Configuration with minimal Linux kernel achieves most often (42%) the highest noise level of the three compared OSes. Debian system sits in the middle between NOVA and minimal Linux kernel; in 43% of cases it achieves the middle noise level.

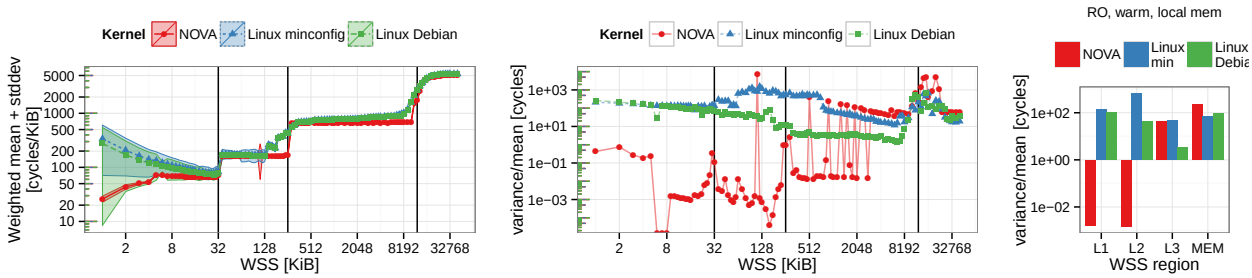

Figure 4: Results of a single experiment. Left: mean values and min/max values, middle: variance-to-mean ratio, right: aggregated (median) variance to mean ratio for different WSS regions.

5 Conclusions

We compared the level of operating system noise of three different systems – NOVA microkernel and two configurations of Linux. The results are not as straightforward as we initially expected. We summarize our observations in the following points:

- In most cases the noise produced by NOVA microkernel is much lower than the noise of Linux. There are, however, the cases where the opposite is true.
- Operating system is not the only dominant source of noise. In some experiments we could see increased level of noise for working set sizes greater than the cache memory. In those cases the noise was approximately the same for all evaluated systems. From this, we conclude that the access to off-CPU resources is also a significant source of noise.
- System management interrupts (provided that the NOVA spikes are caused by them) are really big source of noise.
- Configuration of the Linux kernel has significant impact on the observed noise.

The data used for the analysis in this paper as well as additional pictures can be obtained from http://rtime.felk.cvut.cz/~sojka/papers/os-noise/.

Acknowledgment

This work was supported by the Ministry of Education of the Czech Republic under the Project GACR P103/12/1994.

References

[1] D. Tsafrir, Y. Etsion, D. G. Feitelson, and S. Kirkpatrick, "System noise, OS clock ticks, and fine-grained parallel applications," in *Proceedings of the 19th annual international conference on Supercomputing*, ser. ICS '05. ACM, 2005, pp. 303–312.

[2] K. B. Ferreira, P. Bridges, and R. Brightwell, "Characterizing application sensitivity to os interference using kernel-level noise injection," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, ser. SC '08. IEEE Press, 2008, pp. 19:1–19:12.

[3] T. Hoefler, T. Schneider, and A. Lumsdaine, "Characterizing the Influence of System Noise on Large-Scale Applications by Simulation," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*, Nov. 2010.

[4] A. Bastoni, B. B. Brandenburg, and J. H. Anderson, "Cache-related preemption and migration delays: Empirical approximation and impact on schedulability," in *International Workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT'10)*, 2010.

[5] U. Steinberg and B. Kauer, "NOVA: a microhypervisor-based secure virtualization architecture," in *Proceedings of the 5th European conference on Computer systems*, ser. EuroSys '10. ACM, 2010, pp. 209–222. [Online]. Available: http://os.inf.tu-dresden.de/papers_ps/steinberg_eurosys2010.pdf

[6] S. Wächtler, "Evaluation of migration costs in multi-core CPU scheduling," Beleg Thesis, TU Dresden, Germany, March 2012.

Figure 5: Comparison of noise levels (variances-to-mean ratios) in different experiments. The bars are plot in this order from left: NOVA, Linux minimal, Linux Debian. See also Figure 4 to understand the meaning of the values in the graphs.