GI-Fachgruppentreffen 2018 - Proposed Talk:

Fast and Accurate Functional Simulation for Dynamic Full System Analysis

Marc Rittinghaus Frank Bellosa
Operating Systems Group
Karlsruhe Institute of Technology (KIT)
{firstName}.{lastName}@kit.edu

A common technique to identify bugs and discover vulnerabilities in applications is to run the software of interest in a functional, that is instruction-level, simulator such as Valgrind [10] or Pin [6] and employ dynamic analysis. For operating system centric research a functional full system simulation (FFSS) is required which includes the operating system in the simulation. FFSS gives developers and security researchers the means to inspect every operation carried out, even in privileged kernel-mode components. This way Google engineers identified over 20 kernel security vulnerabilities in Windows 8 by analyzing the memory access patterns at the system call interface [4].

While functional full system simulation has thus generally proven to be a very powerful tool, a well-known limitation is its immense slowdown. Depending on the required level of detail and the degree of instrumentation, running a workload with FFSS is up to multiple orders of magnitude slower compared to native execution. We have measured a slowdown of 30x with QEMU [1] and up to 1000x with the more accurate Simics [7] simulator. Similar numbers have also been reported by other researchers [5, 8]. In practice, the slowdown creates severe obstacles for a comprehensive use of functional full system simulation:

Interactivity Scenarios that should capture interactivity with a human user or an external network device are not feasible. A single key stroke can quickly take from multiple seconds up to minutes until being fully processed. Network protocols such as TCP in turn react to the situation with throttling and timeouts.

Accuracy of Results Since the simulation considerably slows down the guest, activities dependent on external events such as I/O operations appear to complete faster - a phenomenon called *time dilation* by Chen et al [9]. This distorts measurements and produces unrealistic behavior.

Coverage Evaluating a test scenario in full length can take considerable time, forcing researchers to reduce coverage. The authors of the Google study summarize that the slowdown was the primary restrictive factor, which limited the coverage of their analysis to the system boot phase and short desktop usage [4]–potentially missing further vulnerabilities.

SimuBoost [11] strives to drastically reduce the slowdown of functional full system simulation, thereby allowing researchers to accurately analyze interactive, network-centric,

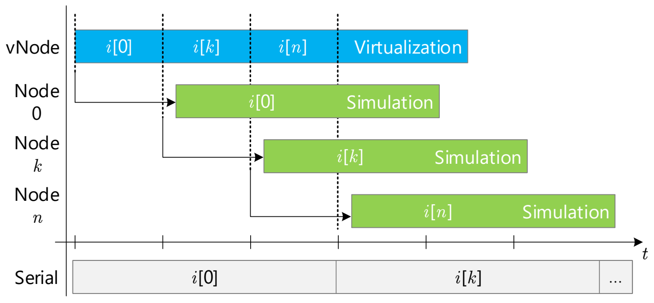

Figure 1: The workload is executed with fast virtualization. Checkpoints at the interval boundaries serve as starting points for parallel simulations.

and long-running workloads. The core idea is to run the workload in a virtual machine (VM) using fast hardware-assisted virtualization. At regular intervals[1] the hypervisor takes a snapshot of the VM state (i.e., memory content, device states, etc.). The checkpoints then serve as starting points for simulations, enabling to simulate and analyze each interval simultaneously in one job per interval. By using multiple nodes (i.e., CPU cores or hosts) a parallelized simulation of the target workload can be achieved (Figure 1).

Functional full system simulation can be build to always produce identical runs. Hardware-assisted virtualization, however, is subject to non-deterministic input such as erratic I/O completion timing. SimuBoost records this non-determinism and uses deterministic replay [2, 3, 12] to accurately reproduce the execution in the simulations, including realistic timing behavior.

Both checkpointing and recording non-determinism need to be geared toward low run time overhead to (1) retain the execution speed difference that drives the parallelization, (2) keep perturbations on the examined workload as small as possible, and (3) preserve seamless interactivity. The simulation nodes on the other side need to quickly receive and load the checkpoints. Furthermore, the resource consumption (e.g., memory) of simulations should be kept low so as to permit a maximum degree of parallelism on each host.

The proposed talk will discuss technical details on the mechanisms we have developed to solve the challenges in each area and present results from the prototype we have built.

[1]Between hundreds of milliseconds to multiple seconds.

1. REFERENCES

[1] F. Bellard. Qemu, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, pages 41–46, 2005.

[2] T. C. Bressoud and F. B. Schneider. Hypervisor-based fault tolerance. *ACM Trans. Comput. Syst.*, 14(1):80–107, 1996.

[3] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. Revirt: Enabling intrusion analysis through virtual-machine logging and replay. *SIGOPS Oper. Syst. Rev.*, 36(SI):211–224, Dec. 2002.

[4] M. Jurczyk and G. Coldwind. Identifying and exploiting windows kernel race conditions via memory access patterns. *Presented as Black Hat*, 2013.

[5] S. Kim, F. Liu, Y. Solihin, R. Iyer, L. Zhao, and W. Cohen. Accelerating full-system simulation through characterizing and predicting operating system performance. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*, pages 1–11. IEEE, 2007.

[6] C. Luk et al. Pin: building customized program analysis tools with dynamic instrumentation. volume 40. ACM, 2005.

[7] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, 2002.

[8] P. S. Magnusson, F. Larsson, A. Moestedt, B. Werner, J. Nilsson, P. Stenström, F. Lundholm, M. Karlsson, F. Dahlgren, and H. Grahn. Simics/sun4m: A virtual workstation. In *Usenix Annual Technical Conference*, pages 119–130, 1998.

[9] A. M. G. Maynard, C. M. Donnelly, and B. R. Olszewski. Contrasting characteristics and cache performance of technical and multi-user commercial workloads. In *ACM SIGPLAN Notices*, volume 29, pages 145–156. ACM, 1994.

[10] N. Nethercote et al. Valgrind: A framework for heavyweight dynamic binary instrumentation. *SIGPLAN*, 42(6), 2007.

[11] M. Rittinghaus, K. Miller, M. Hillenbrand, and F. Bellosa. Simuboost: Scalable parallelization of functional system simulation. In *Proceedings of the 11th International Workshop on Dynamic Analysis (WODA 2013)*, Houston, Texas, Mar. 16 2013.

[12] L.-K. Yan, M. Jayachandra, M. Zhang, and H. Yin. V2e: Combining hardware virtualization and softwareemulation for transparent and extensible malware analysis. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments*, VEE '12, pages 227–238. ACM, 2012.