# Survey on Performance Analysis of Virtualized Systems

Han Wang
George Mason University
hwang31@gmu.edu

***Abstract*—This survey goes over the important concepts in virtualization in the overview section, after that we cover the virtualization implementations, performance analysis challenges, and virtualization measurement tool.**

## I. Overview

Virtualization is an important concepts in modern computer system. According to Siblerschatz, Gagne, Galvin, the fundamental idea behind the idea of virtual machine is to create the illusion that the hardware of a single computer's system (CPU, memory, drives) is being running on different environment(s) (guest), and multiple environments can run on that system as if the system is the native OS and guest is in full control [1]. The guest process obtains the virtual copy from the host [1], which means multiple guest can run on a host in a time sharing system. In a nutshell, host is basically the physical hardware that virtual machines, or guest is running on [1].

Currently, virtualization is widely used in data centers and personal computer due to the increase in improvement of CPU and its support of virtualization over time [1]. Cloud computing is a domain that is possible thanks to virtualization, where the server provides the resources such as CPU, memory, I/O to customers via internet (API) [1]. Hence, thanks to the virtualization capabilities, cloud computing technology is able to provide clients software-as-a-service, infrastructure-as-a-service, and platform-as-a-service [5].

The host and guest are completely different in term of functionality. Per creation of virtual machine, it is created and run by the hypervisor, or the virtual machine manager (VMM) [1], and there are different types of hypervisor with different implementation. The types of hypervisor are Type 0, Type 1, and Type 3 [1]. Each different hypervisor is implemented differently and being used with different purpose [1]. Further explanation is provided in the next section.

There are several important requirements for virtualization, those are:

Fidelity: environment must be identical to the original machine [1]

Performance: program shows minor performance decreases when running in virtual environment [1]

Safety: VMM must be in complete control of the environment [1]

## II. Vitualization Implementations

### A. General/Classical Implementation

This section will introduce how virtual machine is implemented generally, where most virtualization option is implemented, which is called virtual CPU [1] (VCPU). VCPU represents the state that the virtual machine is in, but it does not execute any code [1], and it is under directly to VMM, which maintain VCPU for each guest's CPU state and load the right context during context-switched [1].

For safety, virtual machine (guest) could not have the privilege of the kernel level [2]. However, if the virtual system could not access to kernel level, how could the guest's system call be implemented. There must be transfer of control between virtual user mode and virtual kernel mode [1]. Such transfer of control must occurs on physical machine, where physical user mode is transferred into physical kernel mode [1].

This kind of transfer is addressed by the important concept is Trap-and-Emulate (TE) in virtualization architecture.

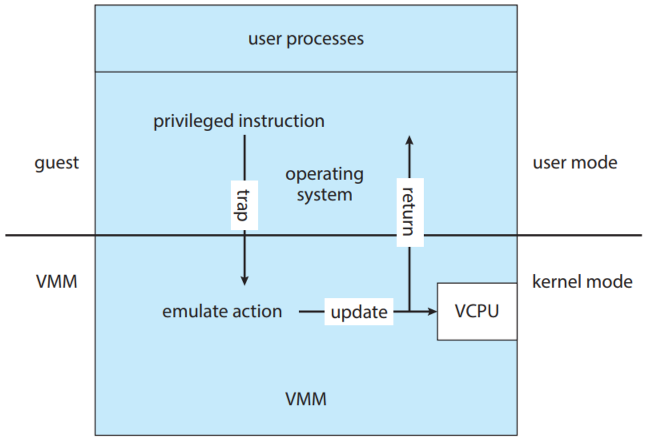


**Figure 1.0** TE implementation [1]

This concept is basically one of the key players in the classical virtualization world. TE is essentially a condition must be met to satisfy the requirement of fidelity and safety.
The functionality of TE is described as whenever kernel tries to run the privilege instruction, it causes a trap to the VMM (in the

real machine), and VMM executes, or also known as emulate [2] and returns control to the virtual machine [1]. Without TE, safety condition could not be met.

## B. *Intel x86 Implementation*

For the virtualization world, Intel x86 was known for lack of hardware support [2]. The reason for this is that the x86 was not built with virtualization considered. The problems of x86 are many, but here are a couple of main reason why it could not virtualize. The visibility of privileged state is not fully implemented where the guest can see that it has been deprivileged [2], this is a problem because the guest must not know that it does not have access to privileged instructions. Another major problem is the lack of traps when privilege instructions run at user-level, this causes the VMM not able to perform TE [2]. Lack of traps is caused by CPU's user mode popf command fail to load all the flag register from the content of the stack [1]. Importantly, user mode's popf command deters any attempt to change the flag "IF", which control interrupt deliveries [2], and this trap could not be generated because of this.

To address this problem, in the early day of x86 virtualization, a solution proposed for such problem would be executed on an interpreter rather than being directly on a physical CPU [2]; however, such solution is disregarded because of the Performance criteria is not guaranteed [2]. The reason for that is because fetch-decode-execute cycle from the interpreter cost hundreds of CPU cycle per one guest instruction [2]. Because it is costly, it is disregarded. The technique calls Binary Translation (BT) to helps virtualization possible by combining interpterion with high performance efficiency [2]. A simple idea technique called binary translation helps virtualization in x86 possible [2].

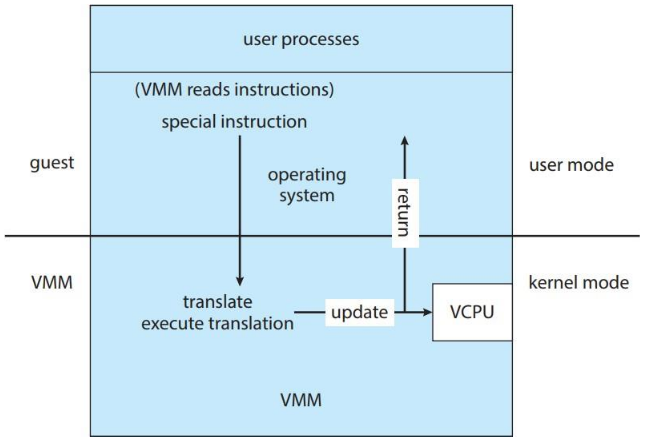


**Figure 1.1** BT implementation [1]

The functionality of BT works like what is described in Figure 1.1, if VCPU is in user mode, guest can run the instruction on the physical CPU [1], if VCPU is in kernel mode, VMM check what the guest VCPU is going to run in the next few instruction by using program counter, and the privileged instructions would be translated to perform the same task [1][2].

## C. *Type 0 Hypervisor Implementation*

Type 0 Hypervisor is a hypervisor type that is close to bare metal implementation [1]. Due to its long existence, it was known under different name, such as "partition" and "domains [1].

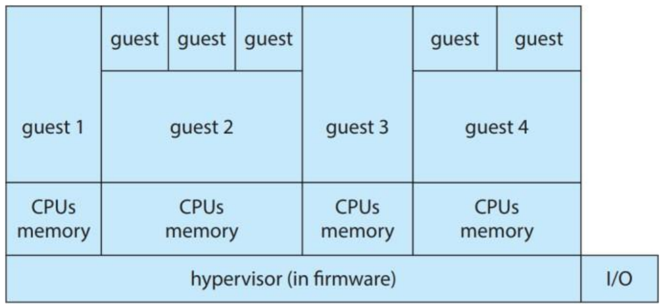


**Figure 1.2** Type-0 Hypervisor general structure [1]

This hypervisor is close to bare metal implementation because VMM is already embedded in firmware and is loaded at boot time [1]. Because the guests do physically have hardware, that simplifies the implementation details [1]. Moreover, because type 0 can run multiple guests (each with different OS), each has their own hardware and allow each to even have their guests hosted [1], hence, virtualization-within-virtualization is possible only on type-0 hypervisor [1].

## D. *Type 1 Hypervisor Implementation*

Type 1 is similar to type 0; it is known for company data centers, and also known as "the data-center operating system" or special purpose operating system [1]

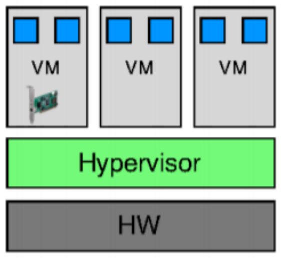


**Figure 1.2** Type 1 Hypervisor [3]

Unlike type 0 hypervisor, where hypervisor is embedded in hardware, type 1 hypervisor is running above the hardware and monitor the guest [4]. The examples of type 1 hypervisor are Xen, VMware ESX, and Hyper-V (Microsoft) [4]. Type 1 hypervisors operates in kernel mode to gain the benefit of hardware protection to satisfy Safety criteria [1]. The main advantage of type 1 hypervisors in comparison to other hypervisors is the implementation of device drivers for other hardware to run and other fundamental OS management (I/O, memory, security) [1].

### E. *Type 2 Hypervisor Implementation*

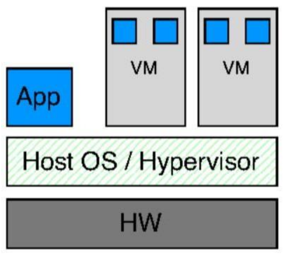


**Figure 1.3** Type 2 Hypervisor [3]

Type 2 hypervisor implementation is simply application run on operating system but provide VMM feature [1], where the guest OS is the software layer above the host OS/Hypervisor [3][4]. Example for those type of hypervisor are KVM, Parallel Desktop, Oracle Virtual Box, and VMware Workstation and Fusion, QEMU [1][4].

## III. PERFOMANCE ANALYSIS CHALLENGES

Performance analysis in virtualization is quite challenging, partly because there are a lot of issue must be under control and there could be many areas to measure; otherwise it is quite hard to effectively measure the performance. There are different metrics and different kind of test. One of the first measurement exist is the measuring live migration of Virtual Machine, conducted by Clark et. al [5]. Live migration is the instance of moving the live OS and all its applications as one unit to another host [5]. The challenge for this surround migrating the memory and local resources [5]. For migrating memory, the challenge is that the migration must minimize the downtime and total migration time [5].

For local resources, the challenge for this part of migration is what to do with the resources that are associated with the physical machine that they originally are virtualized on [5]. Moreover, the researchers in this paper have to worry about measuring network resources and workloads [5]. Moreover the implementation is quite complex as well, it comes with complicated system design with various of step, such as pre-migration, reservation, iterative pre-copy, stop-and copy, commitment, activation [5]. Moreover, the researchers need to take care of measuring the memory management of the guest OS, such as pages in order to have accurate measurement of live migration [5].

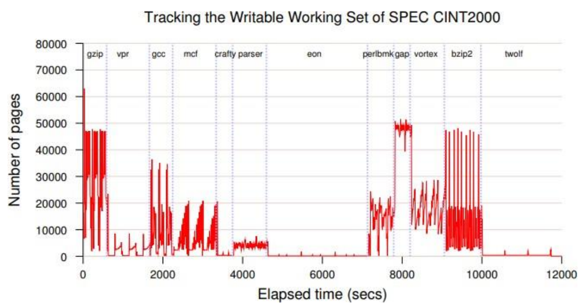


**Figure 2.1** Pages tracking using SPECINT2000 [5]

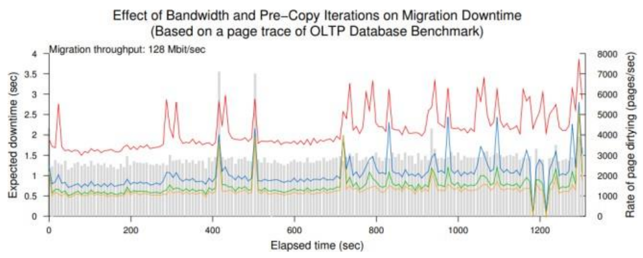


**Figure 2.2** Expected downtime caused by last-round memory copy [5]

Figure 2.1 and 2.2 are result samples from the research paper discussed above. Figure 2.1 shows the measurement of local resource migration and Figure 2.2 shows measurement on memory migration. These two graphs show how complicated measurements of virtual machine migration can be.

Besides live migration performance analysis from Clark et al paper, there are many other form of performance analysis for other criteria in virtualization and they will be discussed below.

In a paper called "A Distributed Control Framework for Performance Management of Virtualized Computing Environments: Some Preliminary Results" by Wang and Kandasamy [6]. In this paper, it addresses approach of how data center could optimize server utilization and energy efficiency by controlling number of machine and, control number of workload, and turning server on and off as needed [6]. The paper measure performance derived from its controller that estimates the incoming request, and then decide to save power of data center automatically by distribute the loads of CPU to handle the request from VM [6].

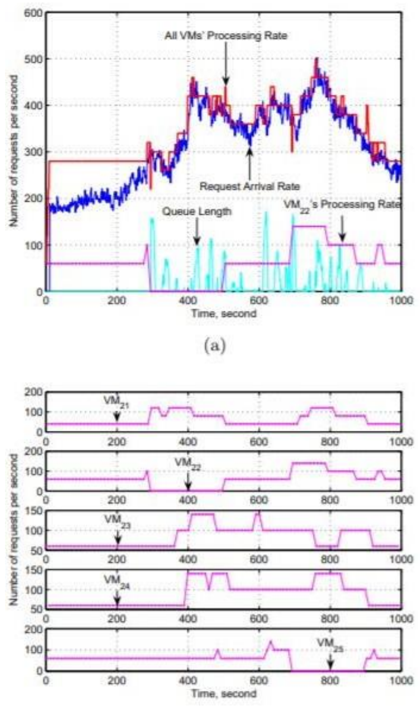


**Figure 2.3** Workload concentrated on one Host and others are turned off [6]

Hence, this saves money of data center because it is able to saves power consumption, because the heavier the workload is placed on 1 CPU, the cost would be more expensive [6]. Figure

2.3 demonstrated the idea that if the workload is reduced, the other hosts would just turned off in order to save power. This form of performance measurement is completely different that what have been mentioned before. Hence, there are different way of measuring performance on virtual machine depending on what the researchers are interested in.

To demonstrate this point further, in the research "Everything You Should Know About Intel SGX Performance on Virtualized Systems" by Ngoc et. al [7], it explains how to measure the Intel SGX (a software to protect data security that is used in virtualization) [7]. This research measures the performance by measuring the main SDK function on both bare-metal and VMs. For this kind of aim, it measures the performance of read and write to encrypted memory, read and write to unencrypted memory, evicting pages, initializing and destroying enclaves (a protective layer of information in SGX) [7]

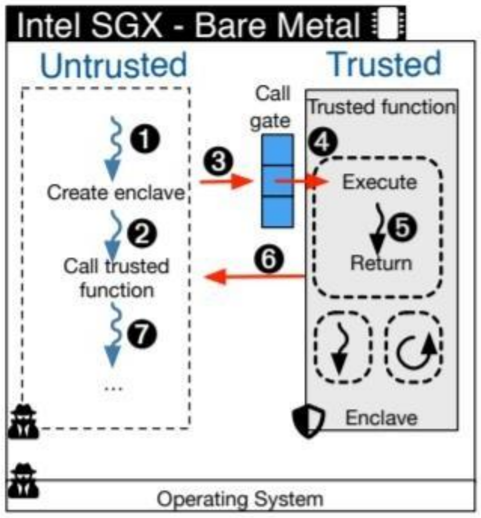


**Figure 2.4a** Intel SGX – Bare Metal Configuration [7]

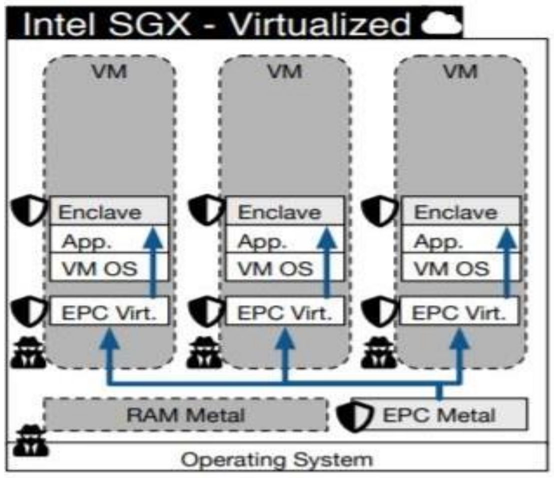


**Figure 2.4b** Intel SGX – Virtualized Configuration [7]

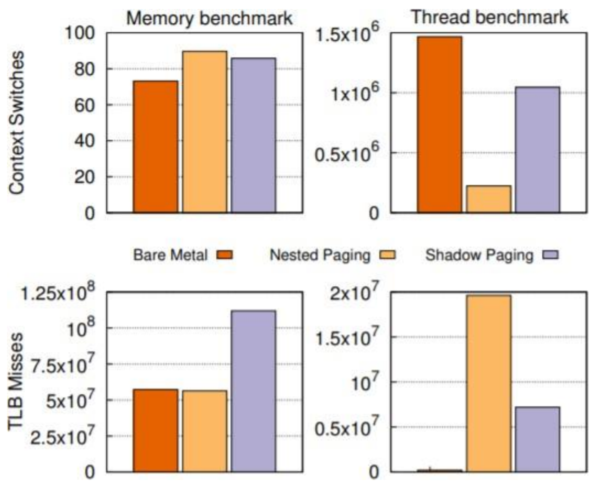


**Figure 2.5** Intel SGX – Virtualized Configuration [7]

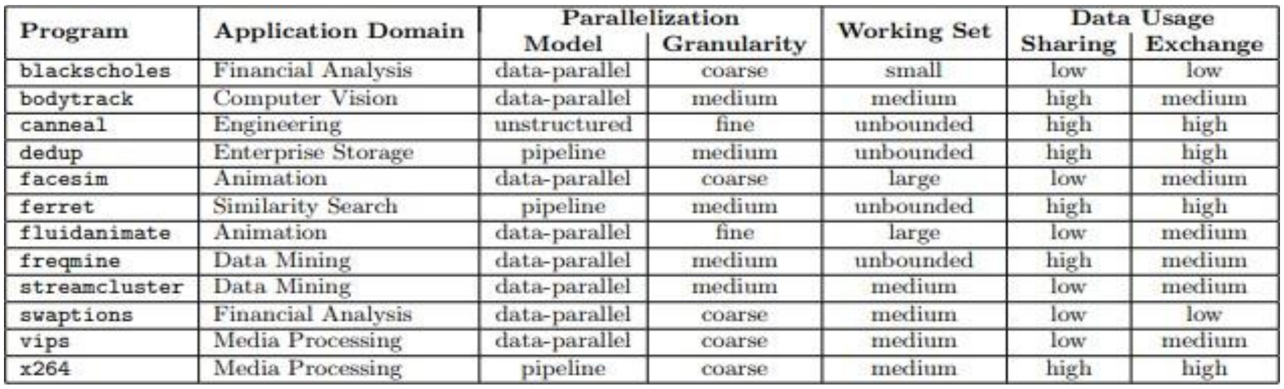
Figure 2.4 a and b is the basic configuration of Intel SGX in bare metal and virtual system. The research in the study use KVM hypervisor as it has SGX support to evaluate performance and to find optimization possibility [7].

Figure 2.5 is about one of the result that the authors find. Apparently, since this study focuses on performance of SGX, it must solely focuses on mostly hardware to measure the performance of SGX. The challenge for measuring performance in virtual machine is quite big because it requires a lot of commitment and research due to the fact that this is heavily relied on operating system concepts.

## IV. Virtualization Measurement tool

Measurement performance of virtualization could be helpful to the user with the right tools. The general ideal tools for user to use is benchmarking. In the study of Ngoc et al, it uses encryption benchmark to measure encryption performance, and it uses HTTP server benchmark to measure networking performance [7]. Hence, benchmark is an ideal start choice to measure performance. To start on this lets visit PARSEC. In the paper "The PARSEC Benchmark Suite: Characterization and Architectural Implications" [8]. It details what benchmarking by stating that if there is no program selection that could provide representative performance of the target application space, performance could be misleading and invalid conclusion could be derived [8]. The paper comes with many requirement such as it must be able to use multi-threaded Application, must cover beyond the capability of previous benchmark, the workload are diverse, and must support research [8].

Each application in the figure 3.0 below has already parallelized and focuses on the emerging workload[8], which means it focuses on potential type of workload that is emerging into the market.



| Program | Application Domain | Parallelization | | Working Set | Data Usage | |
|---|---|---|---|---|---|---|
| | | Model | Granularity | | Sharing | Exchange |
| blackscholes | Financial Analysis | data-parallel | coarse | small | low | low |
| bodytrack | Computer Vision | data-parallel | medium | medium | high | medium |
| canneal | Engineering | unstructured | fine | unbounded | high | high |
| dedup | Enterprise Storage | pipeline | medium | unbounded | high | high |
| facesim | Animation | data-parallel | coarse | large | low | medium |
| ferret | Similarity Search | pipeline | medium | unbounded | high | high |
| fluidanimate | Animation | data-parallel | fine | large | low | medium |
| freqmine | Data Mining | data-parallel | medium | unbounded | high | medium |
| streamcluster | Data Mining | data-parallel | medium | medium | low | medium |
| swaptions | Financial Analysis | data-parallel | coarse | medium | low | low |
| vips | Media Processing | data-parallel | coarse | medium | low | medium |
| x264 | Media Processing | pipeline | coarse | medium | high | high |

**Figure 3.0** PARSEC qualitative workload summary [8]

In a paper called “A characterization of the PARSEC Benchmark Suite for CMP Design”, it evaluates how PARSEC working set affect the tested subject [9]. Specifically, it examines DRAM Latency, throughput, thread scaling, and micro-architecture performance.

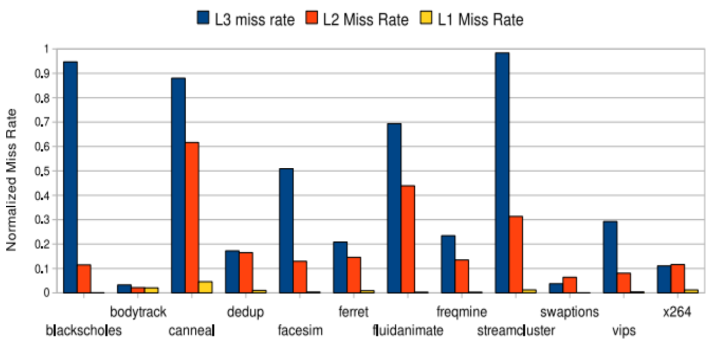


**Figure 3.1** Cache Performance [9]

Figure 3.1 is an example where it shows cache performance being measured with different level with different workload. Moreover, different type of workload affects different degree and level of cache misses [9]. One example of how this can be measured is enough to show that this benchmark suite is capable with testing performance of virtual machine. With its strength, it also have some weaknesses such as it does not have input set for the odd thread counts, such as 5 and 7 in facsim workload [9].

Several works studied the performance of big data applications on modern processors [10, 11, 12, 13, 14, 15]. All of these works use performance counters to measure and monitor the performance and behavior of applications. In [16, 17, 18, 19, 20], authors perform a set of comprehensive experiments to analysis the impact of memory subsystem on the performance of data intensive applications. In [21, 22], author uses compress sensing and hardware accelerators to improve data movement after finding the performance bottleneck using performance counters. Performance counters also can be used to trace the applications behavior in order to find the malicious behavior [23, 24, 25, 26, 27]. Moreover, there are new approaches to improve the performance of modern computing systems such as using machine learning and hardware acceleration [28, 29, 30, 31].

However, performance counters are not enough for measuring the performance of hardware accelerators and we need new mechanism for such goal.

## V. Conclusion

Virtualization is a field that requires in depth understanding in operating system, computer architecture, and good knowledge of how to use benchmarking program to measure performance. The challenges in virtualization arrives from the complexity of virtual machine and how it is designed. Hence, it requires continuously effort of improvement from the academia and industry in order to improve the virtual machine, particularly measuring performance to run diagnostic, and use the result to find improvement or fixing the bug of that particular machine. Moreover, good benchmarking suite is also a good place to start to understand and getting into computer scientific research community.

## Rferences

[1] A. Silberschatz, P. Galvin and G. Gagne, Operating system concepts, 10th ed. WILEY, 2018

[2] K. Adams and O. Agesen, "A Comparison of Software and Hardware Techniques for x86 Virtualization", Vmware.com, 2006.

[3] C. Dall, S. Li, L. Lim and J. Nieh, "ARM Virtualization: Performance and Architectural Implications", cs.columbia.edu, 2019.

[4] ]E. Bauman, G. Ayoade and Z. Lin, "A Survey on Hypervisor-Based Monitoring: Approaches, Applications, and Evolutions", Utdallas.edu, 2019.

[5] C. Clark et al., "Live Migration of Virtual Machines", Usenix.org, 2005.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987.

[7] T. Dinh Ngoc et al., "Everything You Should Know About Intel SGX Performance on Virtualized Systems", Dl.acm.org, 2019.

[8] C. Bienia, S. Kumar, J. Singh and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications", Parsec.cs.princeton.edu, 2008.

[9] M. Bhadauria, V. Weaver and S. McKee, "A Characterization of the PARSEC Benchmark Suite for CMP Design", Parsec.cs.princeton.edu, 2008.

[10] Makrani, Hosein Mohammadi, et al. "Evaluation of software-based fault-tolerant techniques on embedded OS’s components." Proceedings of the International Conference on Dependability (DEPEND’14). 2014.

[11] Makrani, Hosein Mohammadi, et al. "Energy-aware and Machine Learning-based Resource Provisioning of In-Memory Analytics on Cloud." SoCC. 2018.

[12] Sayadi, Hossein, et al. "Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures." 2017 IEEE International Conference on Computer Design (ICCD). IEEE, 2017.

[13] Malik, Maria, Dean M. Tullsen, and Houman Homayoun. "Co-Locating and concurrent fine-tuning MapReduce applications on microservers for energy efficiency." 2017 IEEE International Symposium on Workload Characterization (IISWC). IEEE, 2017.

[14] Malik, Maria, et al. "ECoST: Energy-Efficient Co-Locating and Self-Tuning MapReduce Applications." Proceedings of the 48th International Conference on Parallel Processing. ACM, 2019.

[15] Sayadi, Hossein, et al. "Power conversion efficiency-aware mapping of multithreaded applications on heterogeneous architectures: A comprehensive parameter tuning." 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2018.

[16] Makrani, Hosein Mohammadi, et al. "Understanding the role of memory subsystem on performance and energy-efficiency of Hadoop applications." 2017 Eighth International Green and Sustainable Computing Conference (IGSC). IEEE, 2017.

[17] Makrani, Hosein Mohammadi, and Houman Homayoun. "MeNa: A memory navigator for modern hardware in a scale-out environment." 2017 IEEE International Symposium on Workload Characterization (IISWC). IEEE, 2017.

[18] Makrani, Hosein Mohammadi, and Houman Homayoun. "Memory requirements of hadoop, spark, and MPI based big data applications on commodity server class architectures." 2017 IEEE International Symposium on Workload Characterization (IISWC). IEEE, 2017.

[19] Makrani, Hosein Mohammadi, et al. "A comprehensive memory analysis of data intensive workloads on server class architecture." Proceedings of the International Symposium on Memory Systems. ACM, 2018.

[20] Makrani, Hosein Mohammadi, et al. "Main-memory requirements of big data applications on commodity server platform." 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE, 2018.

[21] Namazi, Mahmoud, et al. "Mitigating the Performance and Quality of Parallelized Compressive Sensing Reconstruction Using Image Stitching." Proceedings of the 2019 on Great Lakes Symposium on VLSI. ACM, 2019.

[22] Makrani, Hosein Mohammadi, et al. "Compressive Sensing on Storage Data: An Effective Solution to Alleviate I/0 Bottleneck in Data-Intensive Workloads." 2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP). IEEE, 2018.

[23] Sayadi, Hossein, et al. "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification." 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC). IEEE, 2018.

[24] Sayadi, Hossein, et al. "Customized machine learning-based hardware-assisted malware detection in embedded devices." 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, 2018.

[25] Sayadi, Hossein, et al. "Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning." Proceedings of the 15th ACM International Conference on Computing Frontiers. ACM, 2018.

[26] Dinakarrao, Sai Manoj Pudukotai, et al. "Lightweight Node-level Malware Detection and Network-level Malware Confinement in IoT Networks." 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2019.

[27] Sayadi, Hossein, et al. "2SMaRT: A Two-Stage Machine Learning-Based Approach for Run-Time Specialized Hardware-Assisted Malware Detection." 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2019.

[28] Neshatpour, Katayoun, et al. "Design Space Exploration for Hardware Acceleration of Machine Learning Applications in MapReduce." 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2018.

[29] Makrani, Hosein Mohammadi, et al. "XPPE: cross-platform performance estimation of hardware accelerators using machine learning." Proceedings of the 24th Asia and South Pacific Design Automation Conference. ACM, 2019.

[30] Neshatpour, Katayoun, et al. "Architectural considerations for FPGA acceleration of Machine Learning Applications in MapReduce." Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation. ACM, 2018.

[31] Makrani, Hosein Mohammadi, et al. "Pyramid: Machine Learning Framework to Estimate the Optimal Timing and Resource Usage of a High-Level Synthesis Design." arXiv preprint arXiv:1907.12952 (2019).