RESEARCH INTERNSHIP FINAL REPORT

Performance and security implications of 17 years of hardware-assisted virtualization on hypervisors

WIDE TEAM

By:

Wilson Emmanuel Waha Lindjeck

Supervised by :

David Bromberg (Prof., IRISA, Rennes) & Djob Mvondo(Dr., IRISA, Rennes)

Contents

List of Figures

List of Tables

Abbreviations

Acronym	**Meaning**
CPU	Central Processing Unit
NIC	Netwrok Interface Card
RAM	Random Access Memory
VMM	Virtual Memory Monitor
TLB	Translation Lookaside Buffer
PCIe	Peripheral Component Interconnect Express
VF	Virtual Function
LLC	Last Level Cache
vSUB	Virtual Sub System
LOC	Lines of code

Introduction

Cloud computing offers a wide range of services, including database access, server access, application deployment, storage services and access to remote machines. Cloud providers such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP) and others are using virtualization to make the most of their resources. The aim of virtualization is to create virtual environments, i.e. software simulations of physical systems (CPU, memory, NIC, GPU, disk). This makes it possible to run multiple operating systems, applications or virtual machines on a single physical host. Virtualization is made possible by virtual machine management software or hypervisors.

To enhance virtualization functionality, hardware manufacturers such as Intel and AMD have been integrating new features directly into their hardware (CPUs, NICs, etc.) since 2007. Once these features have been added, developers in the hypervisor community must update their code to take account of the new functionality. This update often requires major modifications both to the hypervisor and to the operating systems to be supported, as in the case of paravirtualization (e.g. Xen[3] or Hyper-V). These modifications generally lead to bugs, security breaches and performance degradation…

The objective of this work is to evaluate the impact of the addition of new hardware features on Intel equipment since 2007 on the performance of a hypervisor widely used on the market, namely Xen. Concretely, we evaluate the impact on the performance of applications running in virtual machines caused by each Intel hardware feature, the impact on the administrative tasks of the hypervisor such as the time of creation of VMs, deletion and snapshot. The impact on the ecosystem of the hypervisor by analyzing the modifications made such as the number of lines of code added, deleted, the number of files updated, the number of patches released for each hardware feature, the bugs and security breaches created.

1

Intel virtualization technology and xen hypervisor

1.1 Introduction

In 2007, Intel began integrating features into their hardware (CPUs, NICs, etc.) to enhance virtualization functionality. Once a hardware feature has been made available, developers in the various virtual machine or hypervisor management software communities, such as Hyper-V, VMware, Xen. . . , integrate this new hardware feature into their products. In this chapter, we first present the features released by Intel since 2007 to enhance virtualization, and then talk about the Xen hypervisor.

1.2 Intel Hardware features for virtualization

Intel has integrated a range of hardware features into their equipment, enabling better management of hardware resources between the different VMs running on a server. These features have been added to better manage resources such as memory bandwidth, L3 cache and networking. Here we present each hardware feature and the virtualization sub-systems concerned.

1.2.1 EPT: Extended Page Tables

Second Level Address Translation (SLAT) or nested paging is an extension of the paging mechanism that enables physical addresses of operating systems running in virtual environments to be mapped to main memory addresses, thus reducing the overhead of shadow page tables. To improve memory management mechanisms during virtualization, hardware vendors such as Intel and AMD have added features to their hardware. AMD's implementation of SLAT has been called Nested Page Table (NPT) since the introduction of their third-generation Optetron processor called Barcelona. Intel has also implemented SLAT through Intel VT-x technology in their Nehalem micro-architecture and called it Extendes Page Tables (EPT)[4]. SLAT can be implemented in 02 different ways: the first is a software implementation called Shadow page table and the second is a hardware implementation[5]:

Software Assited Paging (Shadow Page Table)

The addresses generated by a program are virtual addresses which must be mapped to physical addresses in memory. In a native operating system, virtual addresses are mapped to physical addresses using page tables. In a virtual environment such as a VM, guest operating systems try to map the same mechanism to the programs they run. However, programs running in VMs do not have direct access to physical memory, so VMMs must take care of mapping each physical address in a VM to its RAM equivalent. In shadow paging, the hypervisor maintains a correspondence between physical page numbers and machine page numbers, and saves a correspondence between logical page numbers and machine page numbers in a shadow page table. The hypervisor synchronizes the shadow page table with the guest OS page table. This synchronization generates an overhead when the guest OS updates its page table. Figure 1.1 shows the shadow page table mechanism.

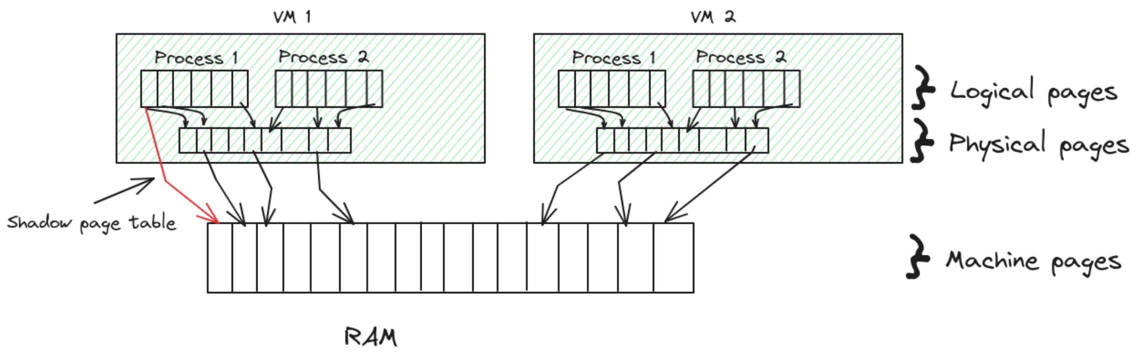

Figure 1.1: Shadow page table mechanism

Hardware Assited Paging (Extended page table)

EPT implements an additional page table that maps the physical pages of the guest OS to the pages in main memory. With EPT, a page table is always maintained by the guest OS to map logical pages to physical pages in the VM, while the hypervisor maps the guest OS's physical pages to physical pages in RAM via additional page tables. So for each memory access operation, EPT performs a mapping between the physical address of the VM's page table and the address on the host provided by the hypervisor.

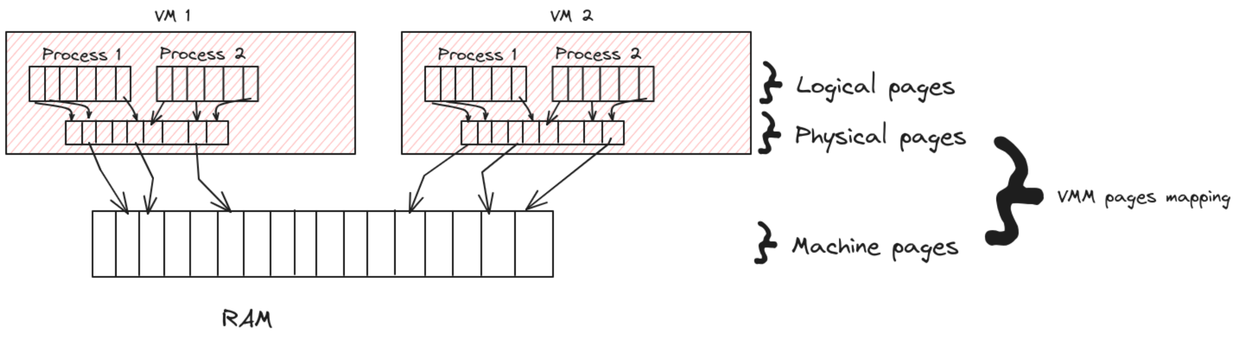

Figure 1.2: EPT mechanism

1.2.2 VPID: Virtual Processor Identifier

One of the main reasons for extending Intel's virtualization technology was to reduce the performance loss caused by VM-exits. In 2008, Intel introduced VPID to its Nehalem processor range. In a native system, addresses generated by programs are considered to be virtual addresses, and must therefore have a corresponding physical address in memory. To improve performance, the memory management unit uses a cache space called TLB to store addresses already in use, thus avoiding the extra cost of searching for an address in main memory. However, when the TLB is full and the virtual address searched for has no correspondent among all its entries, a set of page tables must be scanned to access the corresponding physical address in memory, freeing up space in the TLB to store the new pairing (virtual address, physical address). All these operations considerably reduce performance.

In a virtualization system, the complexity is even higher, as each guest OS must map its virtual addresses to its physical addresses, and then the physical addresses of the guest OSs must be converted to machine addresses. To tackle this problem, Intel has created a Master TLB which contains the TLB entries of both the guest OS and the host. Each virtual processor is assigned an identifier, and mapping operations between the virtual address and the physical address of the guest OS are carried out by the guest, while the correspondence between physical and machine addresses is left to the hypervisor. Note that with VPID, there is no TLB flush during VM-entries and VM-exits, which considerably increases performance[6].

1.2.3 VMDq: Virtual Machine Device Queues

In virtual environments, applications running in VMs generally need to access the Internet, or even communicate with each other over a local network. Hypervisors must therefore be able to manage network packets to and from each VM. This network traffic management increases CPU cycles, reducing performance for other tasks. In 2008, Intel integrated a new virtualization feature into their network cards: VMDq[1]. VMDq is a technology that relieves the hypervisor of some of its network packet management work. With the introduction of VMDq, each VM has a queue containing all the network packets for that VM. At the data link layer, VMDq has a packet classifier which, based on the VM's MAC address, assigns the network packets to the corresponding queue. Once the packets have been placed on the various queues, the hypervisor's virtual switch routes the packets to the corresponding VMs, thus lightening the hypervisor's workload. Figure 1.3 shows the VMDq packet management mechanism.

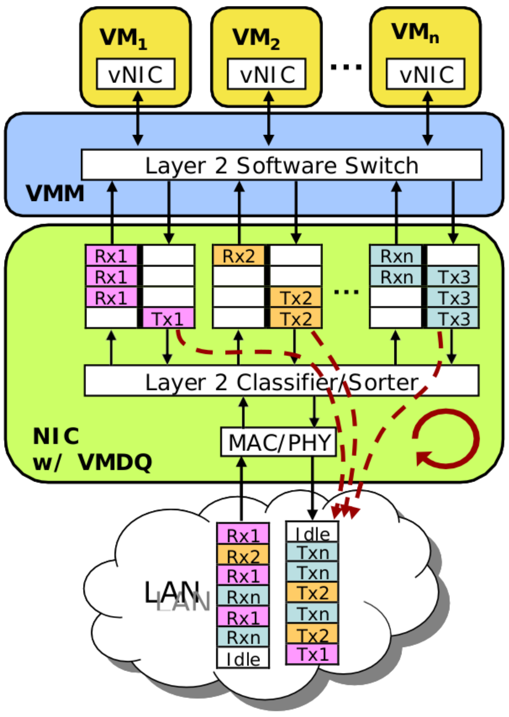

Figure 1.3: VMDq packets sorting

source [1]

1.2.4 SR-IOV: Single Root I/O Virtualization

Device emulation consumes a lot of CPU resources. In the case of virtualization, network device emulation consumes even more CPU cycles. With a view to reducing this loss, Intel has introduced the SR-IOV concept, a specification and hardware technology that enables a PCIe device to be shared between several virtual machines, while retaining the performance and security of traditional hardware. The SR-IOV specification describes the creation of virtual functions (VFs) for PCIe devices, each VF being directly assignable to a virtual machine, enabling each of them to access unique hardware resources. Each VM can then access one of these virtual functions as if it were a dedicated physical device.

SR-IOV functionality is made possible by the introduction of the following PCIe functions:

- **Physical functions (PFs)** : A PCIe function that provides the functionality of its device (e.g. networking, GPU sharing) to the host, but can also create and manage a set of PFs. Each SR-IOV-compatible device has one or more PFs.
- **Virtual functions (VFs)** : Lightweight PCIe functions that behave as independent devices. Each VF is derived from a PF. The maximum number of VFs a device can have depends on the device hardware. Each VF can only be assigned to one VM at a time, but several VFs can be assigned to one VM.

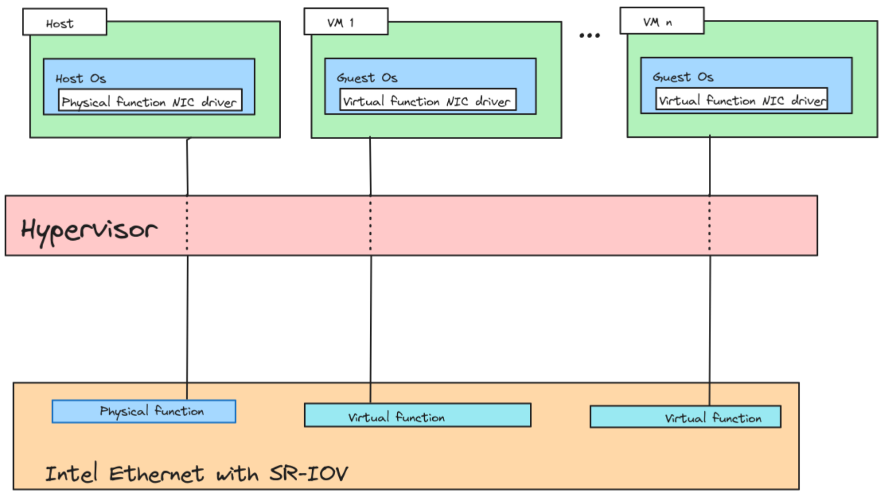
Host
Host Os
Physical function NIC driver
VM 1
Guest Os
Virtual function NIC driver
...
VM n
Guest Os
Virtual function NIC driver
Hypervisor
Physical function
Virtual function
Virtual function
Intel Ethernet with SR-IOV

Figure 1.4: SR-IOV architecture

1.2.5 PML: Page Modification Logging

In 2015, Intel in collaboration with VMware released a new range of processors equipped with a new hardware feature to enhance virtualization called Page-Modification Logging (PML). This is a hardware technique for tracking changes to a virtual machine's memory pages. Normally, tracking memory modifications involves a process called “dirty logging”. The system protects memory pages from being written to. When a write operation occurs, it triggers an error and the system marks the page concerned as “dirty”. This leads to additional workload due to the extra error handling. PML offers a more efficient way of achieving the same objective. It exploits hardware capabilities to automatically record memory addresses (physical guest addresses) that have been modified by the VM each time write protection is activated. This eliminates the need for additional write errors. By avoiding write errors for logging, PML improves performance particularly in scenarios where write operations are frequent within the VM. PML provides valuable information for virtual machine management tasks like checkpointing, live migration (moving a running VM to another physical machine), and working set size estimation (determining the actively used memory by the VM) [7, 8].

1.2.6 CMT: Cache Monitoring Technology

Ensuring efficient resource sharing when running multiple VMs on the same server is crucial to guaranteeing Quality of Service (QOS). Managing shared resources on a server is a challenge, and one of the most critical resources is the last-level cache (or L3 cache). Over the years, researchers have proposed solutions to improve LLC[9, 10] management, but most of these solutions were based on simulations and therefore not applicable in a cloud environment. To remedy this problem, Intel has integrated a new technology called Cache Monitoring Technology(CMT) into its Haswell processor range. CMT means that the hardware provides the operating system or hypervisor with information on LLC utilization across different processor cores, so that it can make better resource scheduling decisions, facilitate detection of “noisy neighbors” and improve performance debugging. Figure 1.5 shows an example of LLC management using CMT technology.

1.2.7 CAT: Cache Allocation Technology

Applications running in datacenters have heterogeneous workloads, and some applications such as streaming and transcoding applications consume more space on shared resources such as the LLC.

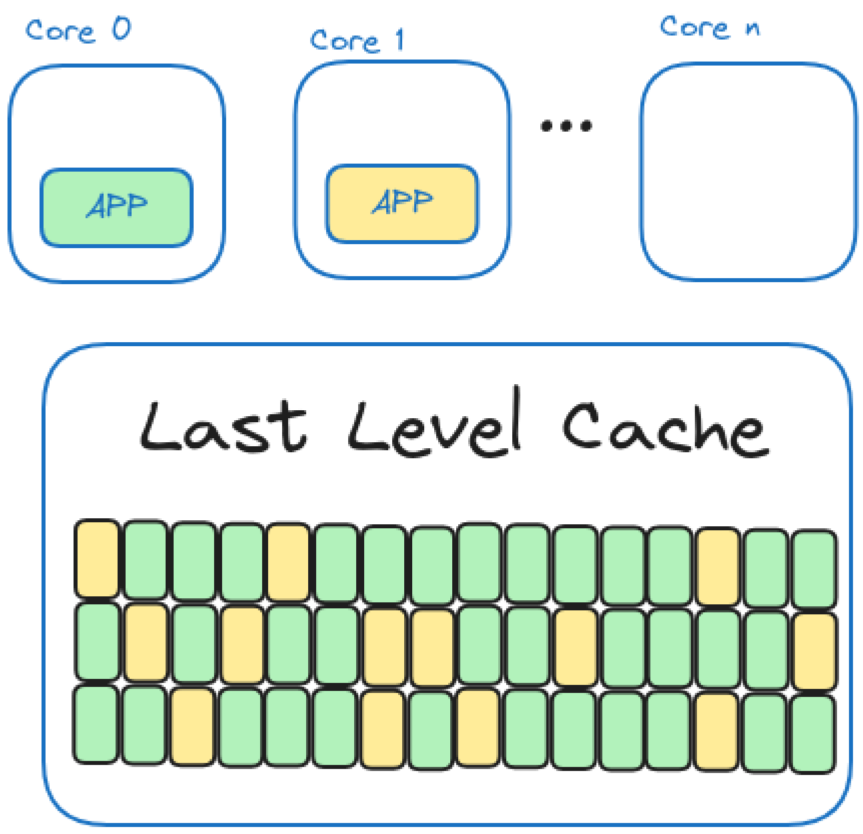

Figure 1.5: CMT example

To deal with this problem, Intel has integrated a new hardware feature called Cache Allocation Technology into their Intel® Xeon® processor E5 v4 family. Cache Allocation Technology (CAT) enables software-programmable control of the amount of cache space that can be consumed by a given thread, application, virtual machine or container. This allows, for example, OSs to protect important processes, or hypervisors to prioritize important VMs even in a noisy datacenter environment.

CAT offer:

- the OS or hypervisor the ability to group applications into classes of service (CLOS), indicating the amount of level 3 cache available for each CLOS.
- The ability to enumerate CAT capacity and support the associated LLC allocation via CPUID.

The notion of class of service (CLOS) behaves like a resource label into which a set of applications/threads/VMs/containers can be grouped, and the CLOS in turn has resource capacity bitmasks (CBMs) indicating the amount of cache that can be used by a given CLOS. Figure 1.6 shows the

CLOS mechanism introduced by CAT.

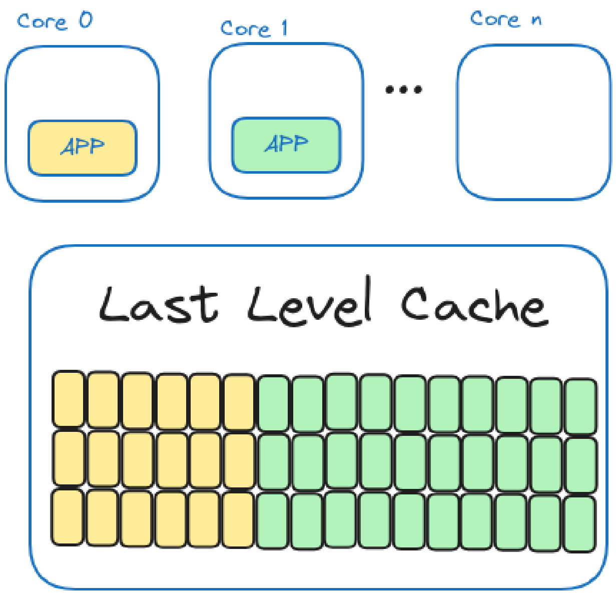

Figure 1.6: CAT example

1.2.8 CDP: Code Data and Prioritization

With the aim of constantly improving L3 cache management, Intel has introduced CDP technology in their Intel® Xeon® processor E5 v4 family, which is an extension of CAT that enables code and data to be separated at the LLC level[11]. Key use cases include protecting the code of certain applications on the L3 cache, particularly those with large code and data footprints, which might otherwise compete for LLC space.

1.2.9 MBM: Memory Bandwidth Monitoring

To better manage application workloads on a server, level 3 cache management is not the only element required when it comes to shared resources. Some applications have a large memory footprint and therefore cannot fit into the LLC. Memory bandwidth management is an essential element for managing thread/app/VMs/container workloads in a virtualized environment. After releasing CMT to improve LLC management, Intel introduced a series of technologies called Intel Resource Director Technology, including MBM, which enables memory bandwidth to be managed. MBM uses the same basic infrastructure as CMT, with new feature enumeration (via CPUID) and new event codes to retrieve memory bandwidth [12].

CMT and MBM have a few things in common:

- Mechanism enabling the operating system or hypervisor to specify a software-defined identifier for each software thread (applications, virtual machines, etc.) scheduled to run on a core. These identifiers are known as RMIDs (Resource Monitoring IDs).
- Hardware mechanisms for monitoring cache occupancy and bandwidth statistics for a given product generation, by software identifier.
- Mechanisms for the OS or hypervisor to read back the collected metrics such as L3 occupancy or Memory Bandwidth for a given software ID at any point during runtime.
- A mechanism to enumerate the presence of the RDT Monitoring capabilities within the platform (via a CPUID feature bit).

1.2.10 MBA: Memory Bandwidth Allocation

With the aim of improving the performance of applications running in VMs, Intel has added a new hardware feauture to improve memory bandwidth management, called MBA. MBA enables memory bandwidth to be distributed between the different cores of a server. MBA is part of the Intel Resource Director Technology (RTD) suite, and complements CAT technology, which manages the L3 cache, while MBA allocates memory bandwidth[13, 14].

1.2.11 SPP: Sub-Page Protection

EPT-Based Sub-page Write Protection, also known as SPP, is a new virtualization technology introduced by Intel in its Ice Lake processor range, enabling a hypervisor to control GPA (Guest physical address) write access permissions at a finer level of granularity than the usual sub-pages (128 bits). Typically, memory access permissions are set on pages of variable size between 4KB and 16KB, depending on hardware specifications. With SPP, access permissions are set at sub-page level, giving the hypervisor fine-grained control over write access permissions by applications to VM memory [15].

EPT-Based Sub-page Write Protection allows :

- enhance security by controlling write access to VM memory

- improve device virtualization and memory check-point.

SPP is active when the "sub-page write protection" VM-execution control is 1. A new 4-level paging structure named SPP page table(SPPT) is introduced, SPPT will look up the guest physical addresses to derive a 64 bit "sub-page permission" value containing sub-page write permissions. The lookup from guest-physical addresses to the sub-page region permissions is determined by a set of this SPPT paging structures.

All the technologies released by Intel to improve virtualization functionalities should be taken into account by developers of virtual machine management software or hypervisors such as VMware, Linux/KVM, Xen, etc.

1.3 Overview of Xen hypervisor

Xen is a type 1 hypervisor, or virtual machine management software[3, 16]. It enables multiple operating systems to run on a single host, while maintaining performance close to that of native systems. Xen uses paravirtualization, enabling guest systems to access the underlying hardware. Rather than emulating the hardware, Xen performs modifications on guest OSes, letting them know they're in a virtualized environment. It is the basis for a number of commercial and open source applications, such as server virtualization, infrastructure as a service (IaaS), desktop virtualization and security applications.

1.3.1 Xen Architecture

Xen is a baremetal hypervisor, i.e. it runs directly on top of the hardware and is therefore responsible for managing resources such as CPU, memory, interrupts, clock, I/O...In conventional operating systems, there are 02 execution modes: privileged mode and user mode. The operating system, which is the priority software, runs in privileged mode (ring 0), while user applications run in user mode (ring 3) figurereffig:os-ring.

Based on this architecture, Xen is installed in Ring 0 instead of the host system, and above it (Ring 1) all the OSes instances. In Xen terminology, running VMs are still called Domains, and there's a special domain called Domain 0, which is the system from which Xen was installed.

Figure 1.8 shows the architecture of the Xen hypervisor.

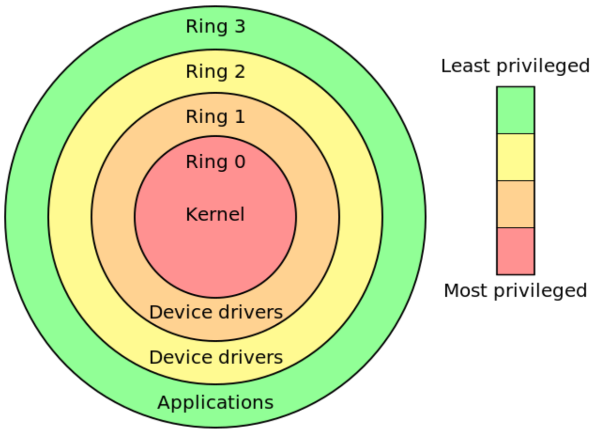

Figure 1.7: Privilege rings for the x86 available in protected mode

source: [2]

- **Xen hypervisor**: It's a software layer that installs directly on top of the hardware and is responsible for managing I/O, interrupts, the CPU and so on.
- **Guests Domains/Virtual Machines**: are virtualized environments, each running its own system. Each VM runs in isolation from the others.
- **Control domain (Domain 0)**: This is the first virtual machine to be started by the system. Domain 0 is the only VM with direct access to the hardware. It contains the tools and services needed to manage other VMs (Guest VMs). It provides interfaces to other OSes for I/O management.

1.3.2 I/O Virtualization in Xen

I/O virtualization is a key aspect of virtualization environments. Xen implements 3 I/O virtualization modes.

- **The Paravitualized(PV) split driver model**: Each guest domain has a driver called front-end driver; Domain 0 has a driver called back-end driver, which is responsible for collecting/transmitting requests/responses from the front-end drivers. The back-end driver in turn communicates with the hardware via native drivers, figure 1.9.
- **Device Emulation Based I/O**: Used in Hardware-Assisted Virtualization (HVM), I/Os are emulated using software. In the case of Xen, QEMU is used for I/O virtualization, figure1.10.

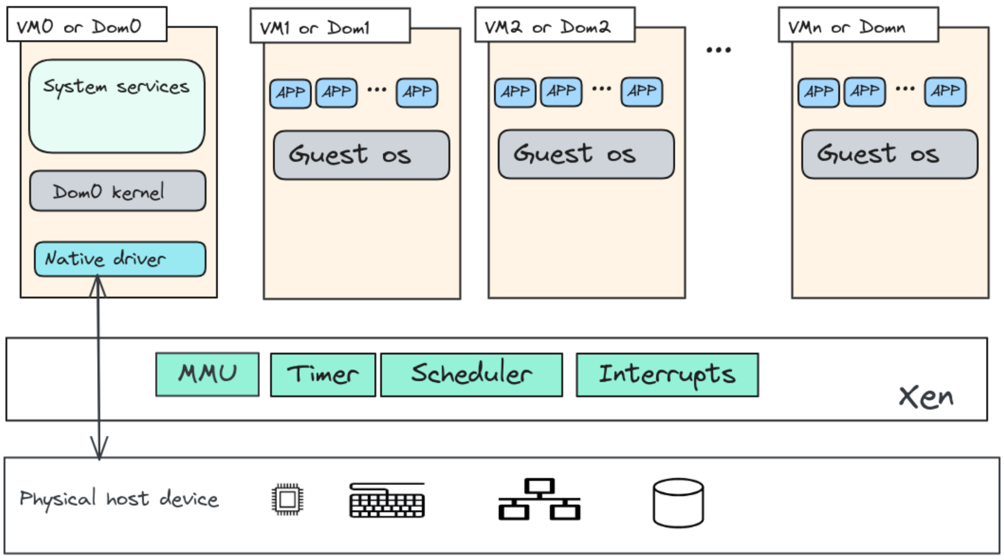

Figure 1.8: Xen project architecture

- **Passthrough**: Allows guest domains direct access to hardware.

1.4 Conclusion

Xen is a virtualization software widely used on the market (some cloud providers using Xen: Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, Alibaba Cloud, Tencent Cloud etc.), and has served as the basis for the development and improvement of other virtualization systems. Since its creation, Xen has benefited from hardware support from Intel and AMD, enabling it to enhance its virtualization features. Although necessary for improving virtualization performance, Intel's integrated hardware features are not easy to get to grips with, and can sometimes lead to security breaches, reduced performance of applications running in VMs, and hypervisor bugs.

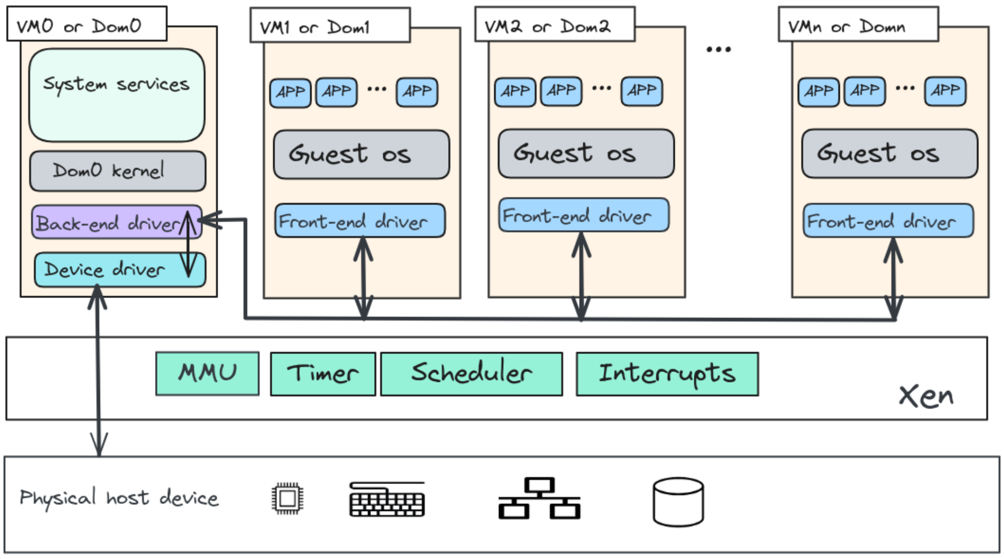

Figure 1.9: Xen Paravirtualization I/O management

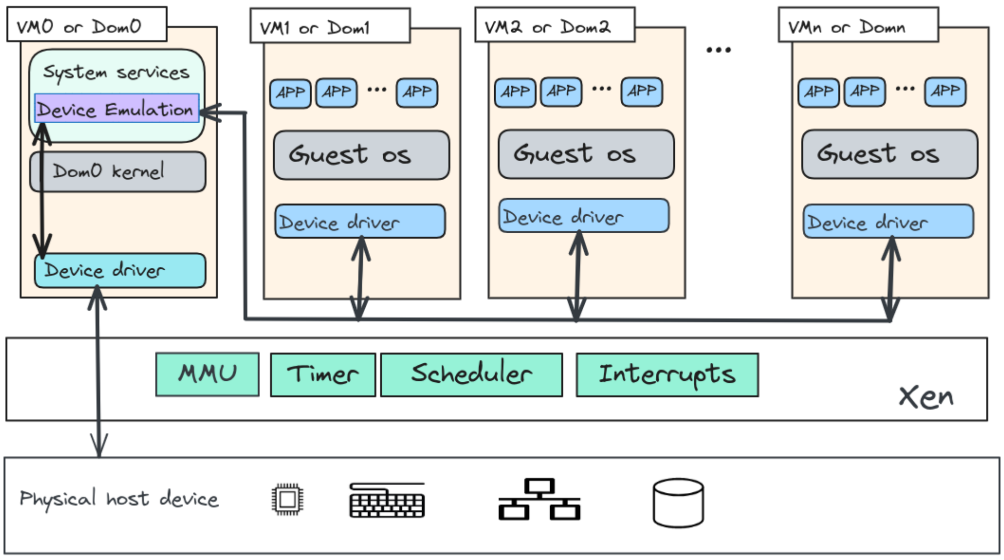

Figure 1.10: Xen HVM I/O management

2

Intel virtualization feature impact on xen hypervisor performance

2.1 Introduction

Since 2007, Intel has been adding hardware features to its server ranges to enhance the performance of virtualization systems. Over time, these hardware features have been incorporated into virtual machine management software or hypervisors such as Xen. In addition to the integration of hardware features into Xen code, patches continue to be released to improve support for the various hardware features. Additional patches are sometimes needed to fix bugs and vulnerabilities introduced when hardware features are taken into account, to improve hypervisor performance, particularly for VM management tasks (creation, deletion, migration, backup), and to improve the performance of applications running in VMs. In this work, we evaluate the impact of hardware features introduced by Intel since 2007 on the performance of a hypervisor widely used on the Xen cloud market.

2.2 Data collection

The first stage of our work consisted in collecting a set of data to enable us to better conduct this study.

2.2.1 Intel hardware feature and processor

Since our study focuses on the features introduced by Intel to enhance virtualization, we have created a tool to collect data on Intel hardware. For each Intel hardware component (processor, NIC, etc.), this tool collects the history of all data for that device, from its release to the present day. Since we're focusing on the harware features that only concern Intel processors, we've built up a dataset of 1048,576 processors with 129 columns, each of which provides information on Intel processors such as Product Collection, Code Name, Vertical Segment (Mobile or Desktop), Processor Number, Total Cores, number of Performance-cores, number of Efficient-cores, Total Threads, Max Turbo Frequency, Lauch Date, etc.

2.2.1.1 Exploratory Data Analysis

In order to make our data easily exploitable, we first carried out an exploratory analysis of the data to see if it contained any undesirable elements. After the exploratory phase, we noticed a few problems in this data set, notably in the Lauch Date column, which is coded by the launch quarter followed by the last 2 initials of the launch year, e.g. Q4'23, which refers to the 4th quarter of the year 2023. Some launch dates had "O" instead of "Q", which caused problems. We did this for each of the columns considered in this work. In fact, we didn't use all 129 columns, we extracted a subset for this study. Once the entire dataset had been cleaned, we sorted the processors by year and quarter of output, according to the "Launch Date" column.

2.2.1.2 Intel processor by Harware feature

For each hardware feature, we have extracted the set of processors on which it is supported. We have therefore built up a dataset for each hardware feature containing information on the first processor families on which it was introduced and the list of processors that continue to incorporate it.

2.2.2 Xen and Intel hardware feature

Hardware features released by Intel to enhance virtualization are integrated into every version of Xen whose release date is greater than the release date of the feature. We've built up a dataset that enables us to find out for each hardware feature the Xen version it was introduced in, as well as the set of Xen versions on which it continues to be supported. To make this task easier, we've set up a tool that lets you download all Xen versions, and check which hardware features each version supports. The following figure shows this dataset.

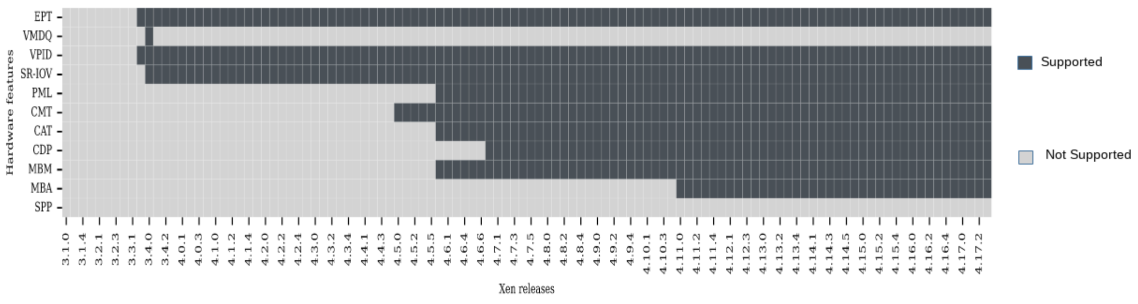

Figure 2.1: Intel hardware feature supported by every Xen version

2.2.3 Grid’5000 Intel processors

Servers containing Intel features are often intended for cloud platforms. All experiments were carried out on Grid’5000[17], which is a flexible, large-scale testbed for experimental research in all areas of computing, with a particular focus on parallel and distributed computing, including Cloud, HPC, Big Data and AI. Grid’5000 offers a wide range of resources for reproducing the conditions of a cloud platform. For each hardware feature, we have selected all the Intel processors in Grid’5000 on which they are supported, as shown in the figure below.

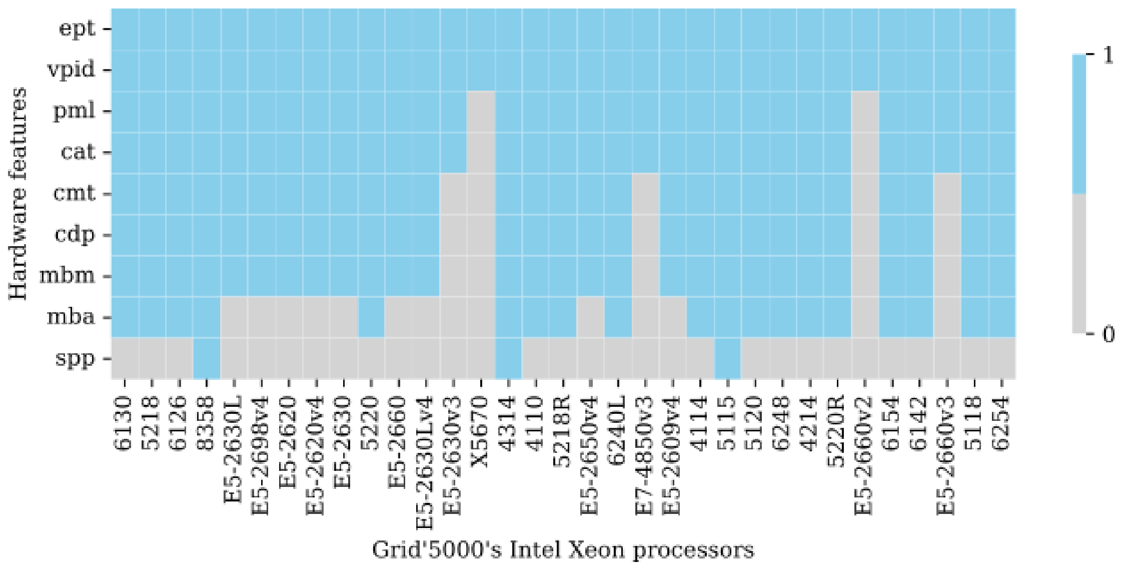

Figure 2.2: Grid’500 processor for each hardware feature

2.3 Performance evaluation

The data collected above enables us to know which versions of Xen and which processors on the Grid'5000 platform we can evaluate for each hardware feature. In our study, we consider several metrics to understand the impact of each feature. The metrics used here can be grouped into 2 categories, namely **Ecosystem related metrics** and **Performance related metrics**. To help us, we've designed a tool called **HaEvol** which makes it easy to analyze the impact of each hardware feature on the virtualization subsystems (Memory, CPU, Disk, NIC etc.), as well as the impact on the hypervisor ecosystem.

2.3.1 Ecosystem related metrics

We collect metrics relating to the hypervisor ecosystem. These metrics show how each hardware feature impacts the Xen ecosystem. For each feature we present the metrics linked to modifications in Xen code, so we have the number of lines of code (LOCs) added, the number of LOCs removed, the number of files updated, the number of patches released. Other metrics are also taken into account, such as the number of bugs, CVEs and major patches.

1. **Code changes**: We evaluate the modifications made by each hardware feature to the Xen architecture. In this category we look at the number of files updated, the number of lines of code added, the number of lines of code removed, the number of patches released. This group highlights the hardware features that have involved the most changes in the Xen ecosystem. Figure 2.3 shows the modifications made by each hardware feature to the Xen code.

   We can see that the feature requiring the most modifications is **CMT**(Cache Monitoting Technology). This may be due to the fact that CMT was the first feature to introduce LLC management. Indeed, shared resources are quite difficult to manage, and one of the elements having a major impact on the performance of virtualization environments is LLC management. Features such as **CAT**, **CDP** are based on **CMT** and have therefore improved L3 cache management.

2. **Bugs**: This is the number of bugs produced by each hardware feature. By bugs we mean any error reported by a user and officially marked by the Xen community as a misbehavior. This metric enables us to understand the potential reasons for bugs, as well as the period of

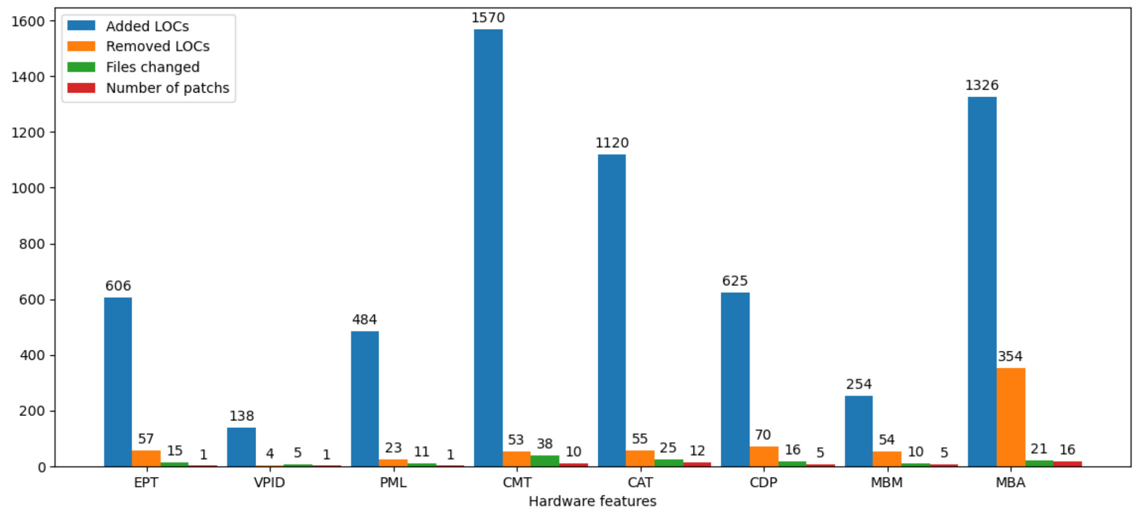

Figure 2.3: Code changes for each hardware feature

discovery after the feature has been taken into account, which could guide developers in code verification and testing.

3. **CVEs (Common Vulnerability and Exposures**: As with bugs, we also look at the number of CVEs discovered per feature. This metric can be used to identify the most vulnerable virtualization sub-systems.

4. **Major patches**: This is the number of patches that have been provided to improve the performance of each feature. This group of metrics provides an overview of the evolution of each feature over time and its impact on the hypervisor.

2.3.2 Performance-related metrics

Performance here is assessed on 2 levels: firstly, the impact on administrative tasks such as VM creation, deletion and snapshotting. Secondly, on the activities of the applications running in the VMs.

2.3.3 VMs Activities metrics

In a virtualization environment, the applications running in the VMs stress the virtualization subsystems (Network, Memory, Disk ...). We therefore chose a set of benchmark applications to see the effect of each feature on the various virtualization subcomponents. The following table shows the selected applications and their corresponding vSUBs.

Benchmarks	**Description**	**Main vSUB**
Byte-Unixbench	Benchmark suite for some operating system operations	CPU
Fio	Tool for stressing the storage system with specific read/write patterns.	Disk
Parsec	Benchmark suite composed of multithreading programs with shared-memory designs.	Memory
Redis-benchmark	Tool for stressing Redis, an in-memory database for storing key-value objects	Memory
Netperf	Tool for generating HTTP workloads and measure throughput and latency	Network
Apache Bench	Tool for benchmarking http servers	Network

Table 2.1: Summary of benchmark use

2.3.3.1 Benchmark automation

HaEvol lets us define benchmark scenarios such as the number of VMs to run, the CPU load of each VM and the memory load.

The benchmark process is as follows:

1. We retrieve the version of Xen before integrating the hardware feature.
2. We apply only the patches linked to this feature and call this version of Xen “Xen_feature_version”, where feature is one of the 9 hardware features studied here.
3. We run the benchmarks on the version of Xen before the feature was introduced, the custom Xen version and all the versions of Xen that have been released since the integration of the hardware feature up to the next feature, as shown in figure 2.4.

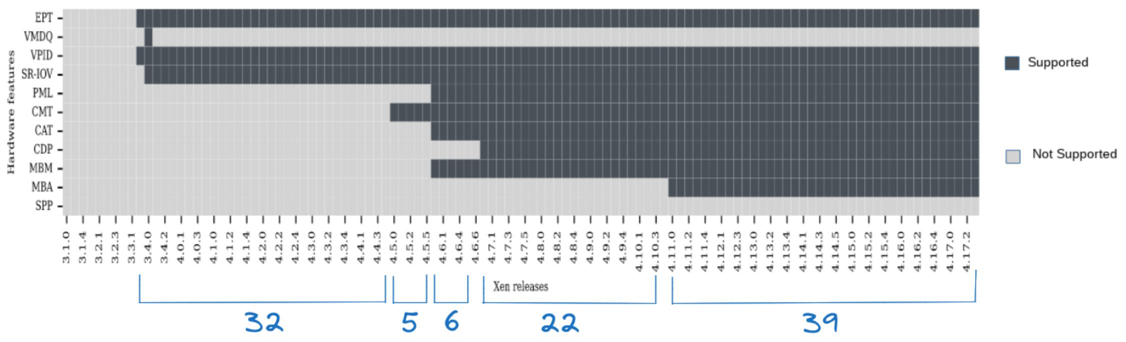

Figure 2.4: Number of xen version to evaluate for each feature or group of feature

4. For each version of Xen mentioned above and for each benchmark application, we pass to our **HaEvol** tool the number of VMs to be run, and the CPU and memory load of each VM.

5. **HaEvol** initiates benchmark execution.

The table 2.2 shows the different scenarios used:

Parameter	**Description**	**Value**
background_VMs	Number of background VMs that should be running	10
host_cpu_load	Host CPU saturation or usage	2
host_memory_load	Host memory load or usage	4096
round	Number of times the benchmark must be run	10

Table 2.2: Benchmark scenarios

2.3.3.2 Benchmarks results

We started our analysis with MBA, so we assessed its impact on each virtualization subsystem using the benchmarks in the table 2.1.

1. **FIO**: Being a benchmark that stresses the disk by performing I/O operations, we've defined 4 scenarios: Sequential readings, Sequential writes, Random reads, Random writes. As defined in the table 2.2, we launch FIO execution in 10 VMs simultaneously and retrieve the **Bandwidth**, **Latency** and **IOPS**(number of I/O per second) on one of these VMs. We ran our benchmarks on 12 versions of Xen since version 4.10.4, including our custom version

containing MBA, named here Xen 4.10.4_mba. After running the benchmarks 10 times, we present the mean, median and 95th percentile to assess which statistical metric best reflects the impact of I/O operations on the disk.

- **Bandwidth**: figure 2.5 shows the FIO bandwidth for each of the 4 scenarios.

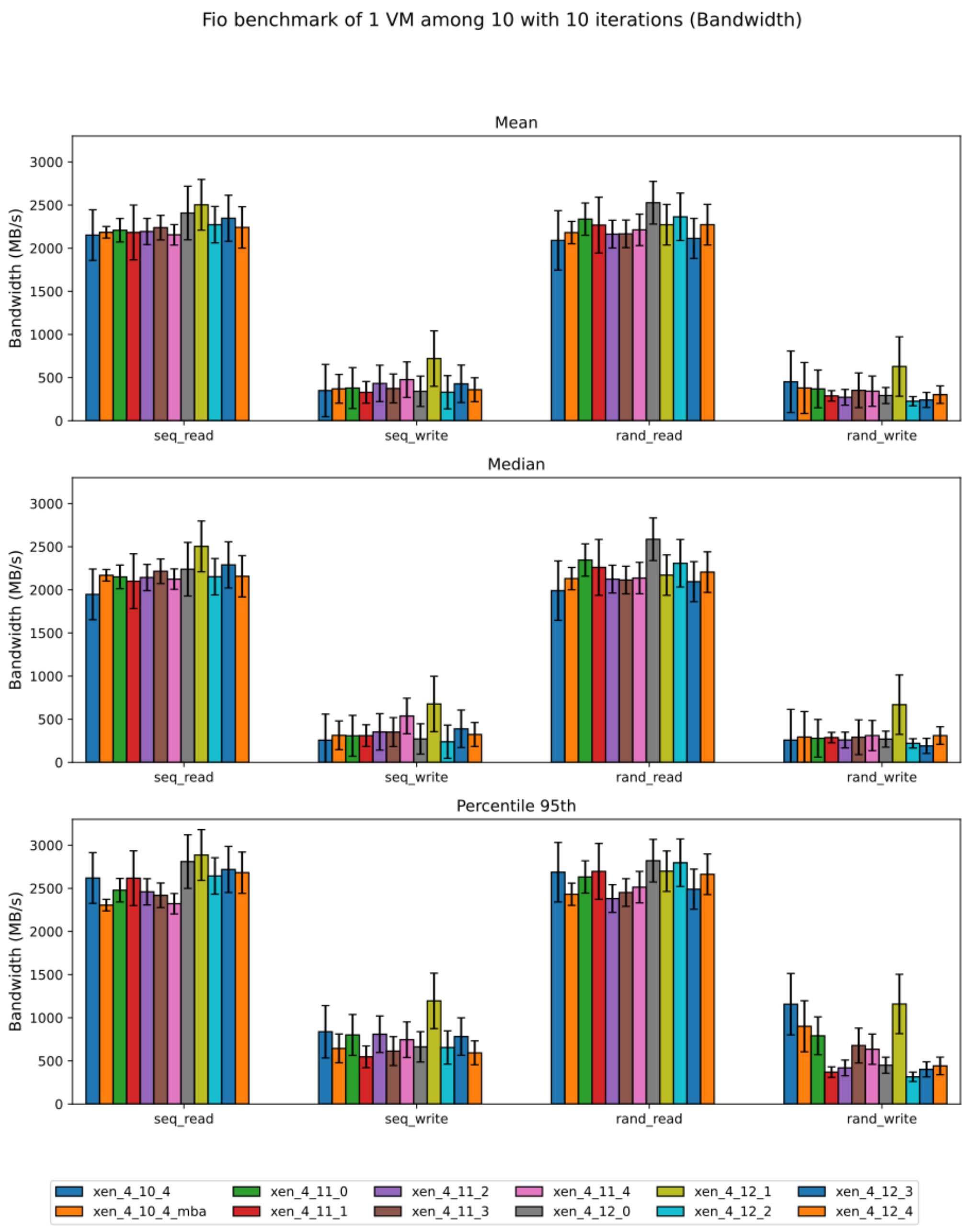

Figure 2.5: Fio bandwidth

Following the median and mean, we can see that the bandwidth of Xen version 4.10.4 is lower than that of all other versions for sequential read, sequential write and random read operations.

- **IOPS**: We can also see on figure 2.6 that the number of I/O per second of Xen 4.10.4 is lower than that of all the other versions containing the MBA patches according to the mean and median.

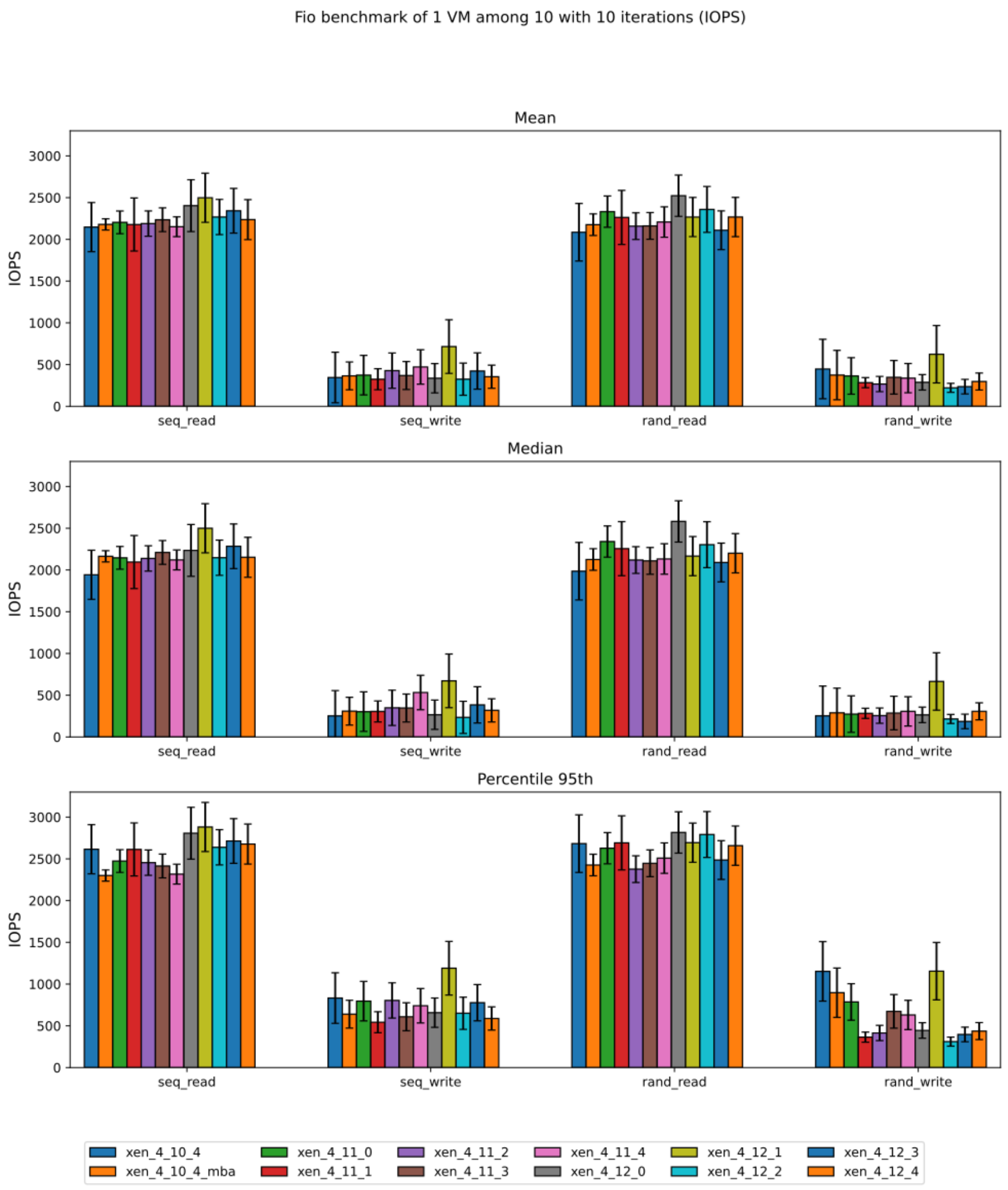

Figure 2.6: Fio IOPS

- **Latency**: Figure 2.7 shows that the latency of Xen version 4.10.4 is lower for sequential and random read operations. No particular trend can be observed for sequential and random write operations.

We can therefore conclude that MBA may not have significantly improved I/O operations, but it has had an impact on disk-based I/O management.

2. **Redis-benchmark**: Redis-benchmark is a tool for evaluating the performance of a redis server by sending a large number of requests to the server. We run redis-benchmark simultaneously in all our VMs so that we can stress each VM's memory. When VM memories are stressed, this has a direct impact on physical memory. The metrics observed here are latency and throughput on one of the running VMs.

   - **Latency** : Figure 2.8 shows Redis latency on a VM.

     No particular trend can be observed, with the exception of version 4.12.2, which has a relatively high latency. It would therefore be advisable to carry out a more in-depth analysis of this version to understand the reasons for its relatively high latency.

   - **Throughput**: Figure 2.9 shows Redis throughput on a VM.

     There is no observed monotony in throughput between Xen 4.10.4 which does not contain the MBA feature and other versions which do. It could therefore be said that MBA has not had an impact on memory bandwidth management operations.

The first observation that is made when looking at these different benchmarks is that MBA does not have a significant impact on I/O management as well as on memory bandwidth. In order to affirm or contradict this provisional result, we must continue to analyze other versions of Xen and perform other benchmarks to have a global overview of the impact that MBA could have on Xen.

2.4 Futher Work

The next step is to run the rest of the Xen versions for MBA, since we've stopped at version 4.12.4, as well as the other benchmarks; then for the other hardwares feature perform the same work as shown in figure 2.4.

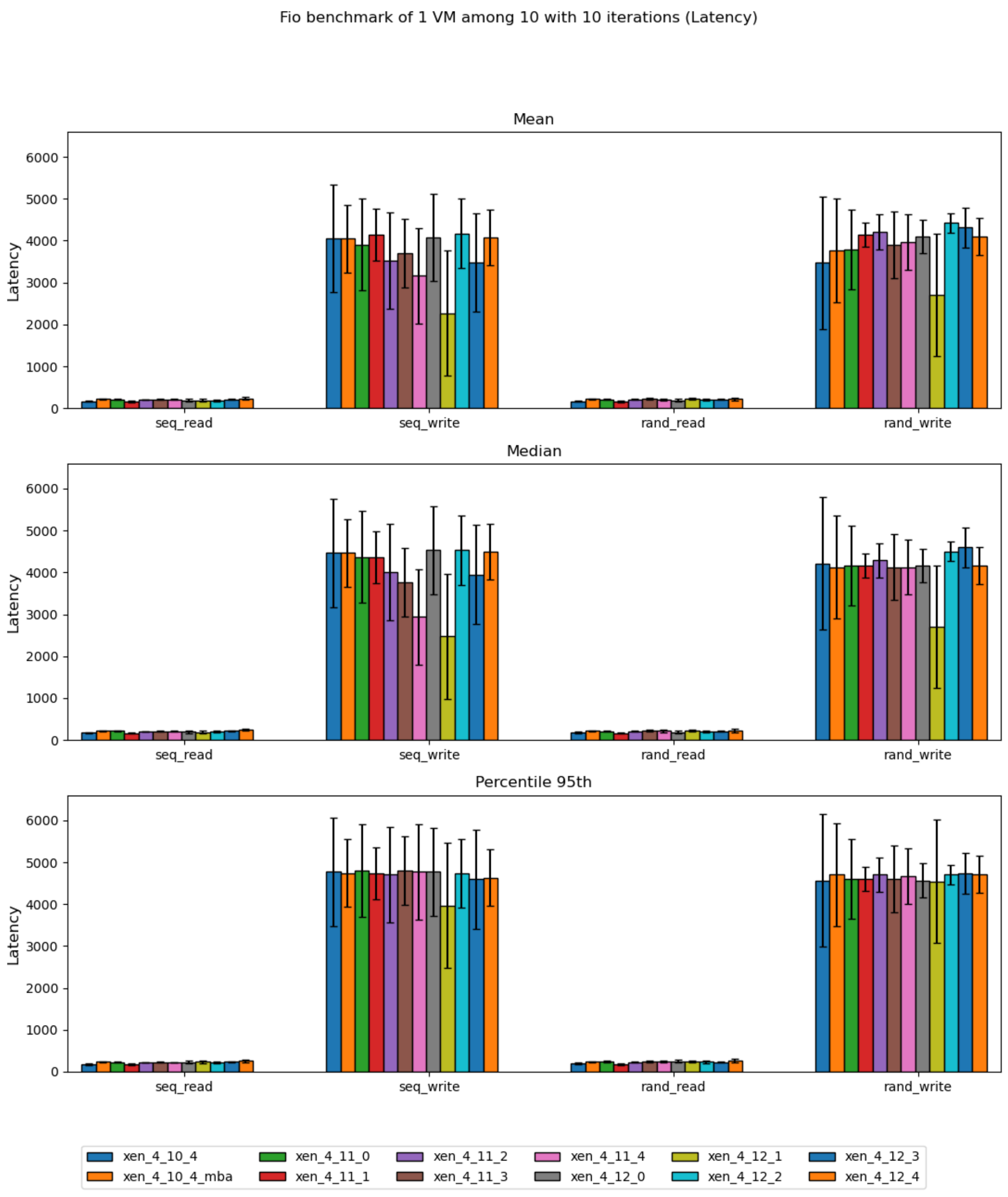

Figure 2.7: Latency of Fio

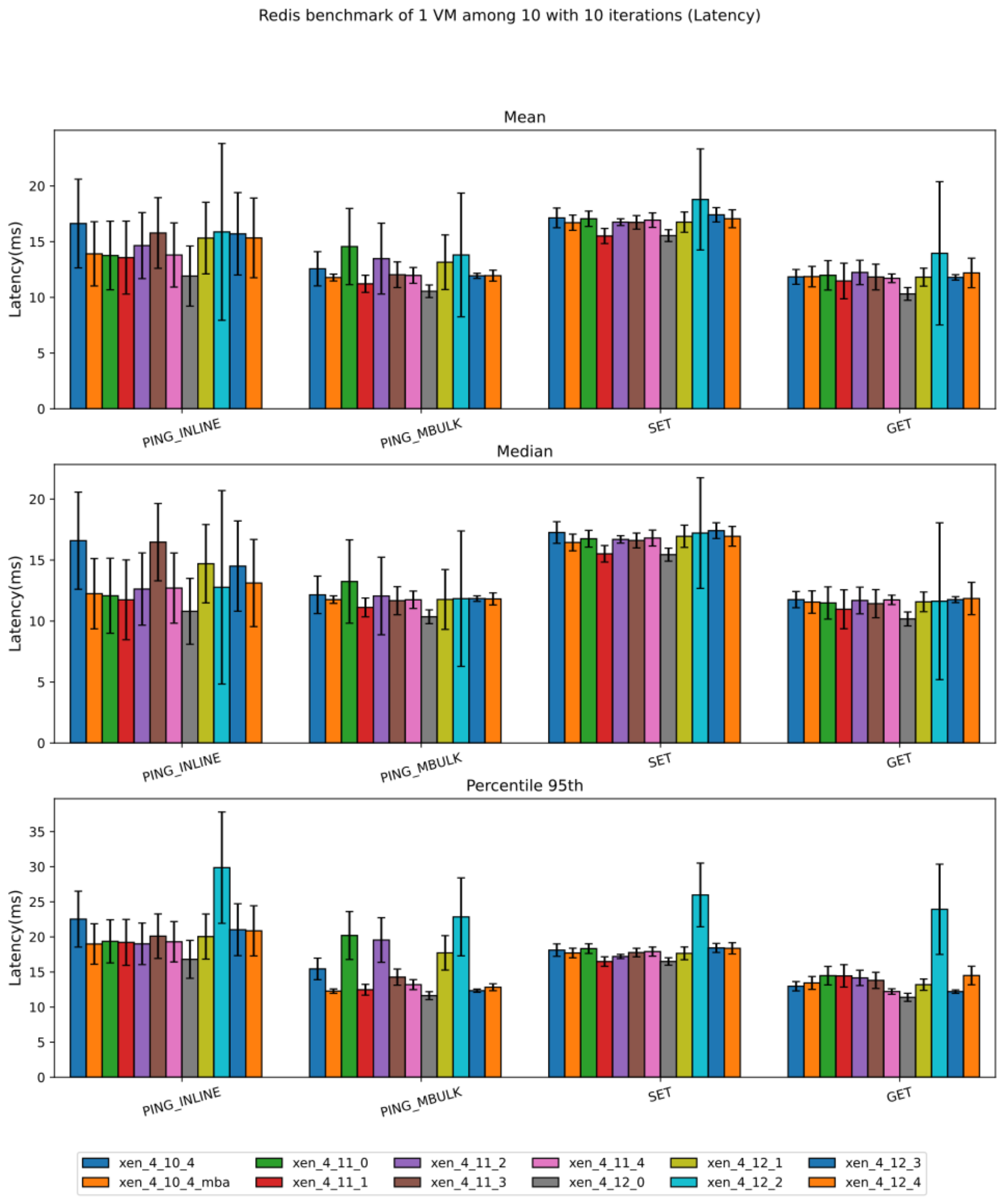

Figure 2.8: Latency of Redis-benchmark

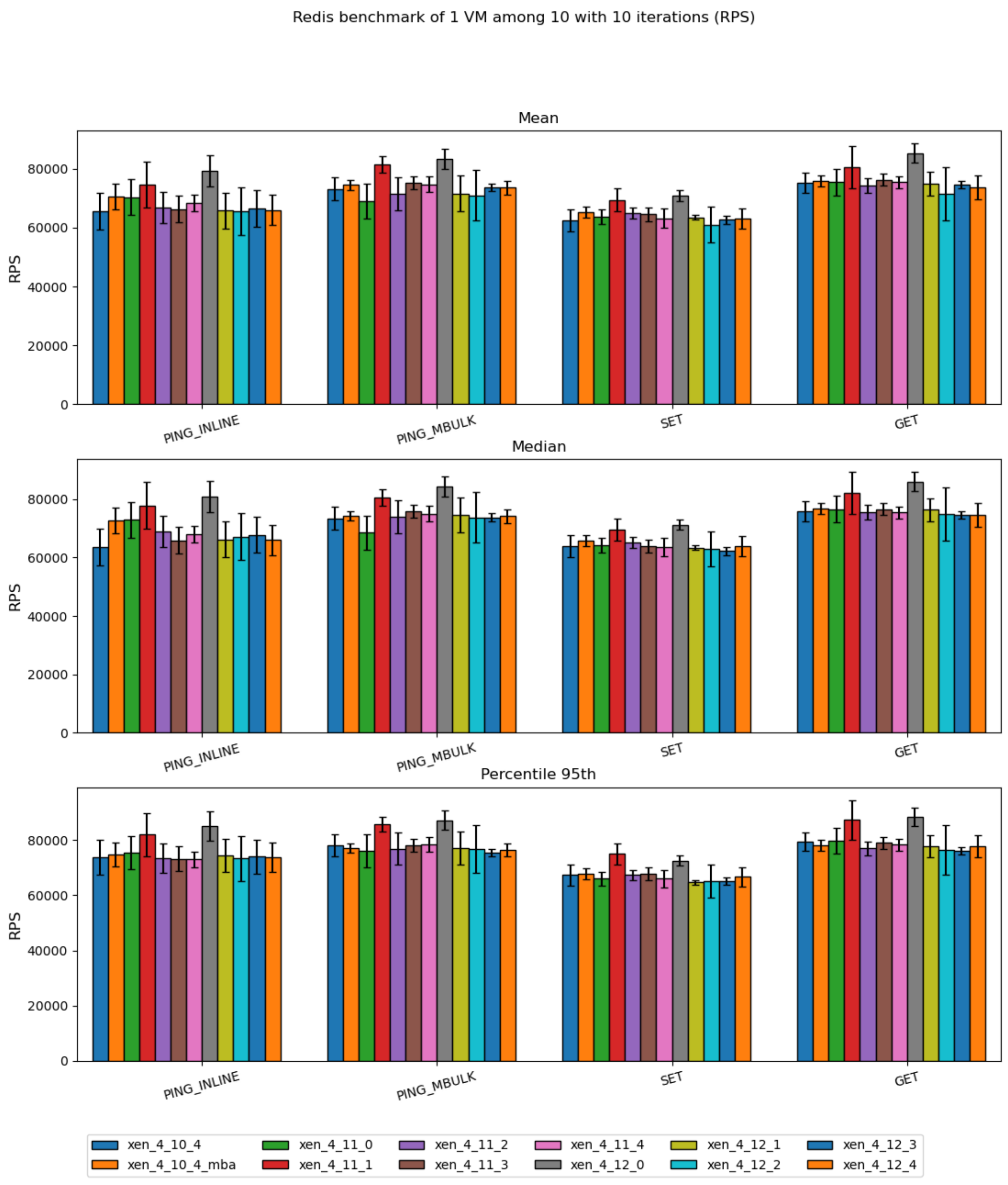

Figure 2.9: Throughput of Redis-benchmark

3

Conclusion

In this work, we set out to evaluate the impact on Xen performance of the hardware features introduced by Intel since 2007. We first collected and formed datasets containing for each hardware feature the list of Intels processors on which they are found, then we established a dataset that allows us to have for each feature the first version of Xen to have integrated it as well as the list of Xen versions on which it is supported. Once these elements had been gathered, we rescenced the list of all the processors on the Grid'5000 platform that support each feature, enabling us to carry out our evaluations directly on the processors specific to each feature, and also to reproduce the execution conditions of a cloud environment. Once we had established the right environment for our evaluations, we set up the **HaEvol** tool to help us run the various benchmarks. Given that we have a large number of Xen versions on which to run our tests, it was imperative for us to have a tool that allowed us to automate the various benchmarks. In addition to benchmarking, we also assessed the impact of each hardware feature on the Xen ecosystem, presenting the changes that had to be made to Xen's structure.

Acknowledgments

I would like to express my deepest gratitude to Prof., David Bromberg and Dr., Djob Mvonde for offering me this wonderful internship opportunity within the WIDE team of the IRISA laboratory.

I would especially like to thank my supervisors David Bromberg and Djob Mvondo, for their trust, guidance and support throughout this project. Their invaluable advice, expertise and availability were essential to me throughout this internship.

I would also like to express my gratitude to the entire WIDE team, who warmly welcomed and integrated me into their structure. The exchanges and discussions I had with each member of the team greatly enriched my internship experience.

Finally, I would like to thank the University of Rennes 1 for allowing me to carry out this research internship, which was a highly formative and enriching experience for me.

I'm very grateful for this opportunity, which first of all enabled me to discover the fascinating field of systems research, particularly in the cloud, where I was able to put my knowledge into practice and acquire new skills.

Bibliography

[1] Intel. Intel® vmdq technology overview, 2008. https://www.intel.sg/content/dam/www/public/us/en/documents/white-papers/vmdq-technology-paper.pdf.

[2] CC BY-SA 3.0 Hertzsprung at English Wikipedia. Privilege rings for the x86 available in protected mode, https://commons.wikimedia.org/w/index.php?curid=8950144. https://www.intel.sg/content/dam/www/public/us/en/documents/white-papers/vmdq-technology-paper.pdf.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *19th ACM Symposium on Operating Systems Principles*, pages 164–177, 2003.

[4] Jeff Casazza. First the tick , now the tock : Next generation intel microarchitecture (nehalem). pages 1–9, 2009.

[5] VMware. Performance evaluation of intel ept hardware assist. 136362:1–14, 2009.

[6] Darren Abramson. Intel ® virtualization technology: Hardware support for efficient processor virtualization. *Intel Technology Journal*, 10, 2006.

[7] Stella Bitchebe, Djob Mvondo, Laurent Réveillère, Noël De Palma, and Alain Tchana. Extending intel pml for hardware-assisted working set size estimation of vms, 2021.

[8] Intel Corporation. Intel ® 64 and ia-32 architectures software developer ' s manual volume 3a : System programming guide , part 1. *System*, 3:1807–1814, 2010.

[9] Fei Guo, Yan Solihin, Li Zhao, and Ravishankar Iyer. A framework for providing quality of service in chip multi-processors. pages 343–355, 12 2007.

[10] Ravi Iyer, Li Zhao, Fei Guo, Ramesh Illikkal, Srihari Makineni, Don Newell, Yan Solihin, Lisa Hsu, and Steve Reinhardt. Qos policies and architecture for cache/memory in cmp platforms. *ACM SIGMETRICS Performance Evaluation Review*, 35:25–36, 06 2007.

[11] Khang Nguyen. Code and data prioritization - introduction and usage models in the intel® xeon® processor e5 v4 family, 11 2016. https://www.intel.com/content/www/us/en/developer/articles/technical/introduction-to-code-and-data-prioritization-with-usage-models.html.

[12] Khang Nguyen. Introduction to memory bandwidth monitoring in the intel® xeon® processor e5 v4 family, 11 2016. https://www.intel.com/content/www/us/en/developer/articles/technical/introduction-to-memory-bandwidth-monitoring.html?wapkw=introduction-to-memory-bandwidth-monitoring.

[13] Yaocheng Xiang, Chencheng Ye, Xiaolin Wang, Yingwei Luo, and Zhenlin Wang. Emba: Efficient memory bandwidth allocation to improve performance on intel commodity processor. 2019.

[14] Khawar Munir Abbasi Andrew J Herdrich, Marcel David Cornu. Introduction to memory bandwidth allocation, 12 2019. https://www.intel.com/content/www/us/en/developer/articles/technical/introduction-to-memory-bandwidth-allocation.html.

[15] Zhang Yi Z. Intel ept-based sub-page write protection support. https://lwn.net/Articles/736322/.

[16] Xen project software overview. https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview.

[17] Grid5000:home. https://www.grid5000.fr/w/Grid5000:Home.