SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Analysis of the Noisy Neighbor Problem in AWS

Youssef Jemal

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Analysis of the Noisy Neighbor Problem in AWS

Analyse der "Noisy Neighbor" Problem in AWS

Author:	Youssef Jemal
Examiner:	Prof. Dr. Leis Viktor
Supervisor:	Till Steinert
Submission Date:	22/08/2025

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 22/08/2025 Youssef Jemal

1 Background

1.1 Simultaneous Multi-threading

Before we dive deeper into simultaneous multithreading, it's important to understand which problem it actually tries to solve and what the motivation behind it is. A processor consists of a few hundred registers, load/store units and a couple of multiple arithmetic units. The main goal is to keep all these resource as busy as possible. To reach this, multiple techniques have been employed such as instruction pipelining, superscalar architecture and out-of-order execution, that improve instruction throughput and resource utilization of the processor. Pipelining is a technique that breaks down the execution of an instruction into several distinct stages, with each stage using separate hardware resources. During each CPU cycle, instructions advance from one stage to another. This allows the CPU to work on multiple instructions simultaneously, each being on a different stage. In a perfect scenario, where all instructions are independent, the processor can work simultaneously on *n* instructions, with *n* being the depth of the pipeline, i.e., the number of stages. The following table depicts a simple example of a five-stage pipeline. At the 5th clock cycle, the CPU is simultaneously working on 5 instructions.

Clock Cycle / Instr. No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX

Figure 1.1: Basic five-stage pipeline (IF = Instruction fetch, ID = Instruction decode, EX = execute MEM = memory read, WB = Write back to memory)

Modern processors are also superscalar. This means that each processor, can start executing more than one instruction simultaneously by dispatching them to different execution units. For example it's able to fetch two instructions on the same time. Issue width is an important characteristic of modern CPUs and it represents the maximum number of instructions that can be executed simultaneously in a single clock cycle. Although these optimizations significantly increase the processor throughput, a relatively big independency between the instructions is required to be able to fully utilize the parallelism. Out-Of-Order execution partially solves this problem but is still not enough as it still dispatches instructions from the same thread, where the dependency between the instructions is generally high. The wastages that occur on the processor can be categorized into two categories: Horizontal waste and vertical waste. Horizontal waste occurs when the CPU is not able to fully saturate the issue width of the processor. Vertical waste occurs when the processor is not able to start any instruction at all because of their dependency to the currently executing instructions.

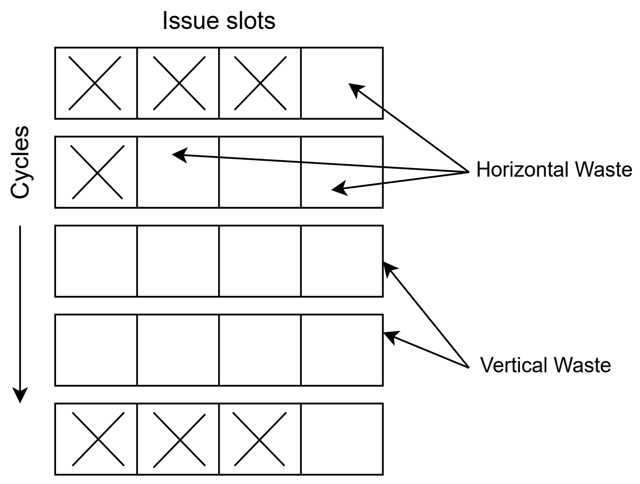

Figure 1.2: Vertical waste vs. horizontal waste

This is where Simultaneous Multi-Threading (SMT) comes into play. SMT is a technique that helps enhance the overall efficiency of superscalar CPUs by improving the parallelization of computation [3]. This technology allows the physical core to dispatch instructions from more than one thread [1] without requiring a context switch, effectively transforming each physical core into two (or more) "logical" cores. The idea is that instructions from different threads provide greate independency, which results in a better utilization of the core's execution resources. To be able to achieve this, some resources of the processor are duplicated, e.g., those that store the architectural state such as registers and program counters. However, the logical cores still share the same execution resources, which can create conflicts, especially if both threads have the same workload nature, e.g., both are float heavy [3]. This technology can improve CPU throughput by taking advantage of idle time that the core formerly spent waiting for other instructions to be completed because of the dependeny between them [1] e.g., when the core is waiting for data from memory after a cache miss.

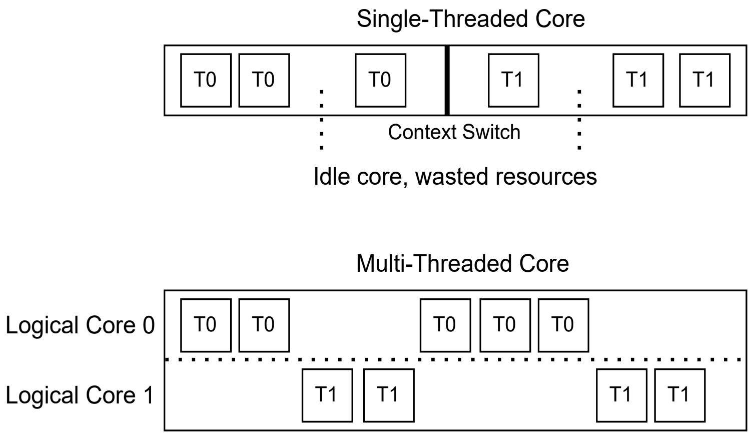

Figure 1.3: Single-Threaded Core vs. Multi-Threaded Core

Both Intel and AMD implement this technology in their modern CPUs, providing two threads per physical core. Intel brands it as Hyper-Threading, while AMD uses the standard term SMT. The option to turn this feature off is always available via the BIOS/UEFI settings. In the AWS dedicated hosts that run on an Intel or AMD CPU with hyperthreading enabled, the number of vCPUs is always the double of the number of physical cores. This, however, opens up the possibility for CPU contention, if two virtual machines have access to vCPUs that share the same underlying physical core. Unlike Intel and AMD CPUs, AWS-designed Graviton processors, that are built around the ARM architecture, do not support hyper-threading [2] and expose one execution context, i.e., vCPU for each physical core. This allows for a better CPU isolation between the different tenants.

1.2 Virtualization

Virtualization is a technology that allows the creation of isolated virtual environments also known as Virtual Machines that run on the same physical server. Each VM has its own operating system and acts as an independent physical computer. These VMs are called "guests" and the physical server is called "host". This technology is crucial for the Infrastructure-as-a-Service (IaaS) model that's offered by cloud providers, as it allows for a greater resource- and cost-efficiency by dividing the physical server into different instance types, driving the price of compute resources down. Users can accordingly choose the instance type with the allocated resources that are optimal for their workload.

The main components that handle the necessary tasks for virtualization are the Virtual Machine Monitor (VMM) also called hypervisor and the management domain. Most of the instructions that are executed by Virtual Machines run natively on the CPU and do not require intervention from the VMM. However, when a privileged instruction, which is not available to regular applications, is encountered, the CPU raises a trap. The trap signals to the VMM to intervene and emulate the behavior of the instruction. After the emulation is finished, the control is then given back to the guest OS.

Virtualization cannot be accomplished by the VMM alone, as it does not virtualize hardware and therefore can not grant the guests access to the underlying hardware devices such as network interface, storage drives, and input peripherals. Device models are required for this. They are basically software components that communicate with the shared hardware and expose multiple virtual device interfaces to the VMs. These device models, along with other management software, run in a special privileged virtual machine called management domain which represents the host's operating system and has access to all the underlying hardware. This domain is called domain zero or dom0 in the Xen project and root/parent partition in the Hyper-V project. Since the device models are software-based, they compete for resources for CPU and system resources along with the existing VMs and can negatively affect the performance of these guests. The following figure summarizes the architecture of a traditional virtualization systems.

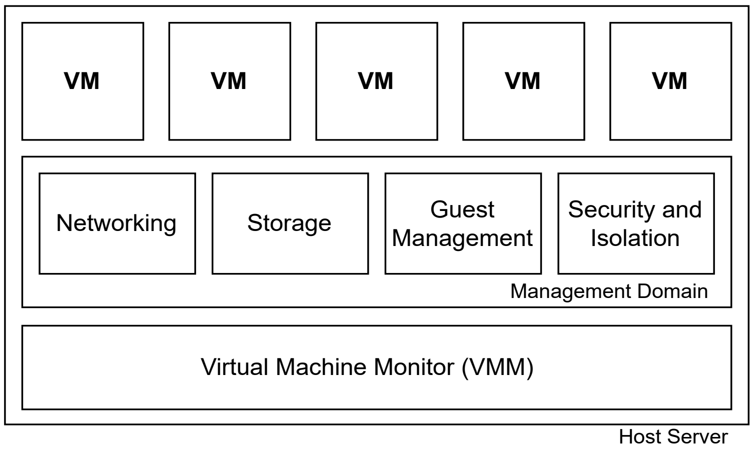

Figure 1.4: Architecture of traditional virtualization Solution

1.2.1 Evolution of Virtualization Solutions

Virtualization technology has evolved significantly. It began with full software virtualization, where the guest OS is unmodified and "unaware" of the virtual environment. Privileged instructions are trapped by the CPU and the hypervisor emulates the sensitive instructions using binary translation. This is, however, very slow and can make the host apps run 2x to 10x slower. Then paravirtualization was introduced, where the guest OS is modified to interact directly with the hypervisor via "hypercalls", removing the abstract emulation layer that is found in full software virtualization. The next major leap was hardware assisted virtualization (HVM) which introduced virtualization support directly on the hardware level by providing highly efficient and fast virtualization commands. This provides a significant improvement in comparison to the previous virtualization techniques. Intel offers this under the Intel Vt-x technology that provides virtualization of CPU and memory. Another important example is Single Root Virtualization (SR-IOV), which is a technology that allows physical PCIs device such as Network Interface Card (NIC) to expose multiple virtual devices to the hypervisor. The hypervisor can then provide the different virtual machines with direct hardware access to thiese virtual devices, which increase I/O performance significantly.

1.2.2 The AWS Nitro System

The Nitro System is a result of a multi-year incremental process of AWS re-imagining the virtualization technology in order to optimize it specifically for their EC2 data centers. The main idea was to decompose the software components that are on the management domain and offload them to independent purpose-built server components. This helps minimize the resource usage caused by software running in the management domain, effectively allowing a near "bare-metal" performance. The following figure depicts the new AWS microservice architecture for virtualization.

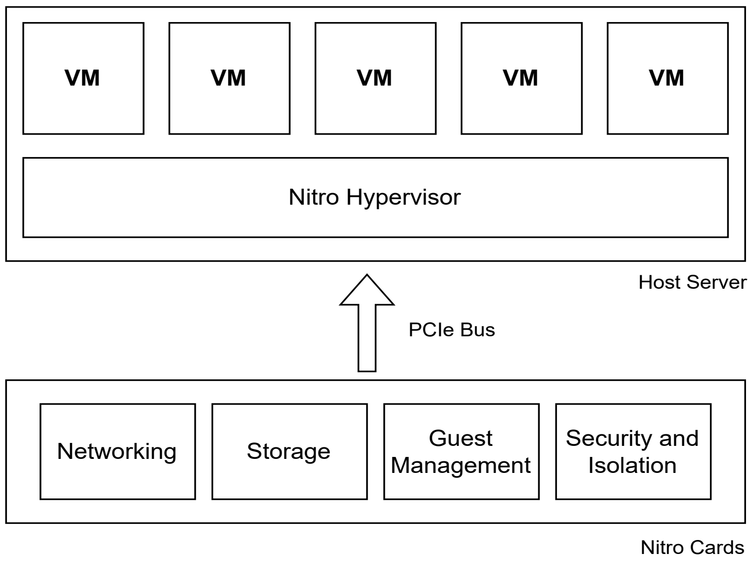

Figure 1.5: Architecture of Nitro System Virtualization

There are three main components in the AWS Nitro System.

The Nitro Cards

These are dedicated hardware components that operate independently from the EC2's server main board (CPU and memory) and are physically attached to it via PCIe. They are responsible for executing all the tasks that concern the outside world in relation to the EC2 Server. They provide all I/O Interfaces such as the ones for storage and networking. When the host's system is using Nitro hypervisor, i.e., not a bare-metal, then these card employ the previously explained SR-IOV technology to provide direct hardware interfaces to the VMs. Example of nitrocards are Nitro card for EBS, Nitro Card for Instance storage, and Nitro Controller, which provides the hardware root of trust of the Nitro System.

The Nitro Security Chip

The Nitro Security Chip extends the hardware root of trust and control over the system main board including CPU and memory. It's managed by the Nitro Controller mentioned previously and plays a crucial role in enabling AWS to offer bare metal instances. In virtualized environments, the hypervisor is responsible for securing the host's hardware assets. However, in bare metal modes, when no hypervisor is present, The Nitro Security Chip assumes this role and ensures the security of the system firmware from tampering attempts through the system CPUs.

The Nitro Hypervisor

The third component is the AWS Nitro Hypervisor. This hypervisor has much less responsibilities than normal hypervisors, as a lot of functions are offloaded by dedicated hardware. It has three main functionalities: partitioning memory and CPU by usiing the virtualization commands provided by the underlying processor, assigning the virtual hardware interfaces provided by the Nitro cards to the Virtual Machines, and handle the machine management commands that come from the Nitro Controller (start, terminate, stop etc.).

Abbreviations

SMT	Simultaneous Multi-Threading
VMM	Virtual Machine Monitor
IaaS	Infrastructure-as-a-Service

Bibliography

[1] Intel Corporation. *What Is Hyper-Threading?* `https://www.intel.com/content/www/us/en/gaming/resources/hyper-threading.html`. Accessed: 2025-05-23. 2025.

[2] N. Thorat. *AWS Graviton: Unlocking Performance and Cost Efficiency in the Cloud*. `https://www.cloudyali.io/blogs/aws-graviton-performance-cost-efficiency`. Accessed: 2025-05-23. 2024.

[3] Wikipedia contributors. *Hyper-threading — Wikipedia, The Free Encyclopedia*. Accessed: 2025-05-23. 2025.