HOW TO MANAGE NOISY OR NOSY NEIGHBORS?

John M. Acken [1] and Naresh K. Sehgal [2]

[1] Portland State University, Portland, Oregon
[2] Securing the Cloud, LLC, Santa Clara, California

ABSTRACT

One of the biggest challenges for an IT administrator in a private or public data centre is to ensure a fair usage of resources between different Virtual Machines (VMs). If a particular VM does excessive network traffic, I/O or memory access, then other VMs running on that same server will experience a slowdown in their access of the same resource. This phenomenon is called noisy neighbours and results in a performance variation experienced by users over time, due to over consumption of shared resources by other VMs. This interference will occur even if each VM has adequate resources for its assigned workload due to conflict in the shared server resources. In this paper, we look at real-life data from Cloud based VMs showing the performance variability and present methods to detect it. A variation of this phenomenon is when a noisy neighbours can decipher confidential contents of a victim VM. We term it as a nosy neighbour, which can pose security risks. We present specific methods to prevent both noisy and nosy neighbours.

KEYWORDS

Cloud, Performance, Multi-tenants, Virtual Machines, Security, Noisy Neighbour, Nosy Neighbour

1. INTRODUCTION

Data center servers are powerful machines. No single application can possibly utilize their computational power on a 24x7 basis over long periods of time. Hence, there is a good business case to share a data center's resources among many different users. The enabling technology is called virtualization, which creates a layer of abstraction between applications and the underlying as shown in Fig.1.

David C. Wyld et al. (Eds): MLCL, CSITA, NMCO, AIRN, CONEDU – 2026
pp. 01-10, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160101

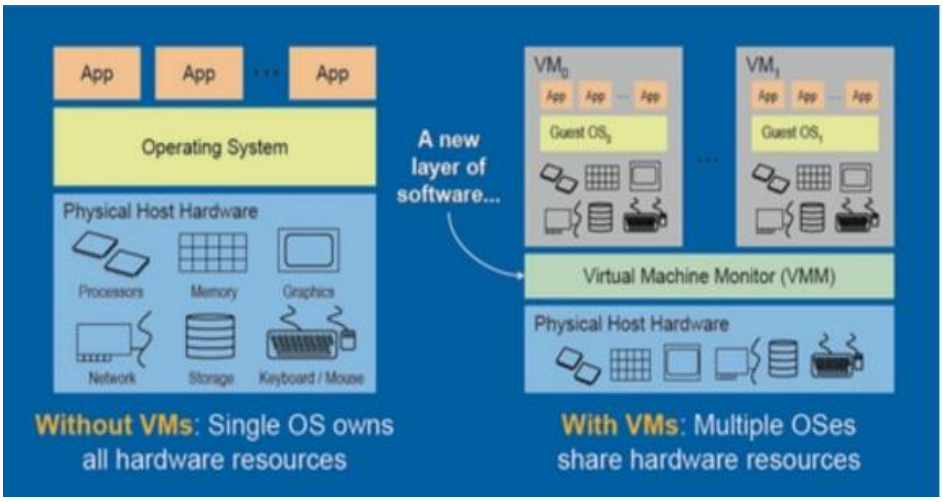

Fig.1: Operating System (OS) vs. Virtualization [1]

Virtualization [1] refers to the act of creating a virtual image of the computer hardware, including CPU, memory, storage, and network elements. It is done to isolate the software stack from the underlying hardware. This isolation is the form of a virtual machine (VM), in which the user jobs run, isolating them from other VMs. A layer of system software, called virtual machine monitor (VMM) or hypervisor, resides between the VMs and the platform hardware. It plays a role like an operating system (OS), managing various VMs, just as the OS man- ages the user processes. The only difference is that each VM may belong to a different and independent user, known as a guest or tenant. A VM may contain its own OS, enabling Windows and Linux to coexist on the same hardware platform.

Virtualization makes each user feel that he or she has full access and control of the machine. There is time-sharing between different users on the same machine. With multi-cores, it is possible to allocate each user to a dedicated core, but still some I/O resources will need to be shared between the users. This may result in occasional performance bottlenecks.

Virtualization is essential to enabling a Public Cloud. Multiple users on the same machine pay a fractional cost to rent the system. It is like public airlines with many paid passengers, each buying a ticket for the journey, and the cost of an expensive airplane is amortized over many such trips. The only key difference is that in a Public Cloud, the users are not aware of each other, nor can they see the content of other Virtual Machines (VMs).Cloud computing has two problems caused by multi-tenancy, namely Noisy and Nosy neighbors. Noisy neighbors inadvertently impact the performance of shared resources. Nosy neighbors intentionally monitor shared resources.

Managing a Cloud operation is like managing any other shared resource. One of the biggest challenges for an IT administrator in a private or public data center is to ensure a fair usage of resources between different VMs. Imagine checking into a hotel after a long flight, hoping to catch a good night's sleep before the next day's business meetings. But suddenly your next-door neighbor decides to watch a loud TV program. Its sound will awaken you for sure, but what's the recourse? Not much, as it turns out, other than calling the hotel manager and making a request to noisy neighbors to lower the TV's volume. A similar situation can happen in a Cloud data center with multiple VMs from different customers sharing the same physical server or any set of resources. Your neighbor in this case may be another Cloud user, running a noisy job with too many memory or disk read/write interactions. Obviously, this will cause a slowdown in other jobs running on the same-shared hardware, as any new read/write request will need to wait in a queue behind the previously issued read/write requests by a noisy neighbor.

In a Public Cloud, once a persistent noisy neighbors' presence is detected, it is beyond the scope of an individual user to avoid it. The reason is that one user has no control over another user's tasks. However, it can be avoided by stopping the task and requesting a different machine. This may interrupt the service, so a better way is to start another server in parallel and migrate the task in a seamless manner, if possible.

It is nontrivial for a Public Cloud user to know if her job is slowing down due to a noisy neighbor's problem, because IT manager isolates each customer's tasks. However, when these tasks use any shared resource, such as a memory controller, then tasks may slow down due to resource contention. This can be detected by careful monitoring and logging on the time it takes for individual tasks at different times. A comparison will reveal if the same tasks, such as a memory access time, are increasing or decreasing in a substantial manner. This change can be attributed to noisy neighbors. This results in a performance variation experienced by a VM user over time [2], as shown in Figure 2 below.

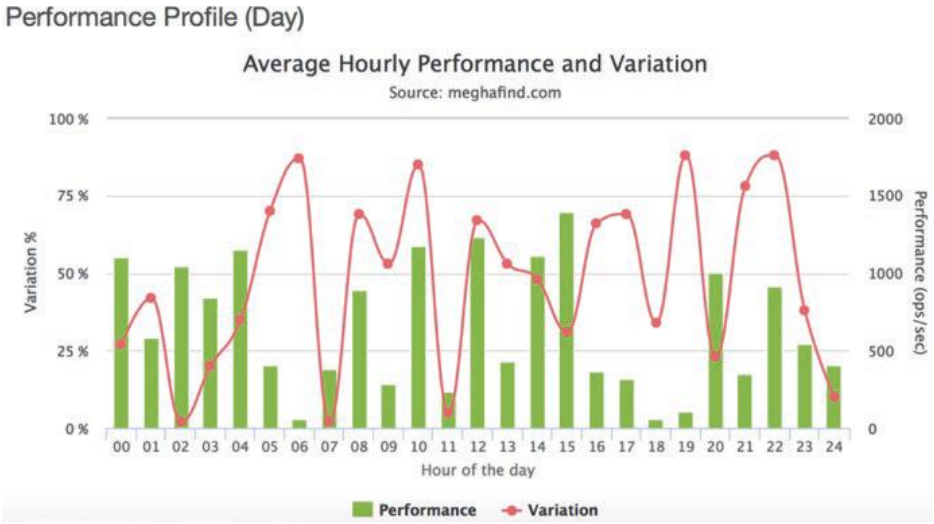

Figure 2: An extreme example of performance variation due to noisy neighbors [3]

To ensure customer privacy, many Public Cloud operators have a stated policy of not looking into customer VMs. This makes it harder for Cloud IT managers to do workloads characterization on a shared server.

2. PHENOMENON AND CAUSES OF NOISY NEIGHBORS

If a particular VM does excessive network traffic, I/O or memory access, then other VMs running on that same server will experience a slowdown in their access of the same physical resource. This phenomenon is called noisy neighbors. Noisy Neighbors result in a performance variation experienced by a VM user over time, due to over consumption of shared resources by other virtual machines. This interference will occur even if each VM has adequate resources for its assigned workload due to conflict in the shared server resources. Some potential causes of noise sources include [4] the following. This is a limited set of examples and not exhaustive list of causes.

- **Simultaneous Multithreading (SMT):** This is a method of partitioning a core for simultaneous execution of multiple application threads. This is useful since relatively few

applications exploit the max instruction-level parallelism of 4 to 5 instructions-per-cycle (IPC) of modern CPUs. So, SMT trades instruction-level parallelism for thread-level parallelism to make better use of underutilized core resources. Various SMT implementations split core resources both statically and dynamically. In other words, enabling SMT robs threads of resources they could access.

- **Last Level Cache (LLC):**While the L1/L2 caches are private to each core of a multicore CPU, they share the LLC among them. This shared space forms yet another point of contention rife with noisy neighbor effect potential.

- **Memory Controller:**Memory Controllers (MC) are complex traffic cops. They must juggle requests fairly from every core on the socket. Rearrange them appropriately to maximize DIMM row buffer hits. Respect DIMM timing constraints and bank refresh scheduling. And they must do all this while preserving memory ordering guarantees. Space constraints limit how much intelligence can be implemented in the MC to protect against bandwidth-hogging applications. All these factors impose a noisy neighbor effect.

- **DRAM:** Server memory subsystems organize into a hierarchy. Memory controller -> memory channel -> DIMM rank -> DRAM chip -> DRAM bank. Read/write concurrency operates at DRAM bank granularity. Therefore, the more banks available, the higher the request concurrency level. In addition to the bank-level concurrency constraint, DRAM requires periodic bank refreshes that preclude concurrent r/w access. While LPDDR and DDR5 offer finer-grained bank-level refresh modes, the more commonly deployed DDR3/4 refreshes at the rank-level, thereby preventing all concurrent r/w access during operation. Therefore, not only do concurrent bank-level requests pose a noisy neighbor effect, but also the refresh schedule of the DIMM itself.

- **Network Controller:** Bandwidth is shared between VMs, and one VM can do excessive I/O at the cost of others, causing dropped packets or delays. An example is of a VoIP service provider, who may be hosting its services in the Cloud. Since all incoming and outgoing phone calls are handled as packets, there is a finite capacity in a server to handle a certain number of calls. This handling may involve copying an incoming packet from network card to the main system memory, then examining its destination and routing it to the next address, before examining or modifying its contents to comply with any security or QoS requirements.

In case of a file transfer, if data is lost due to dropped packets, TCP/IP re-transmits to ensure correct delivery. So, if a file's transfer normally takes 10 seconds, then assuming with a 10% loss, it will take 11 seconds. Users may not care or notice the extra second.

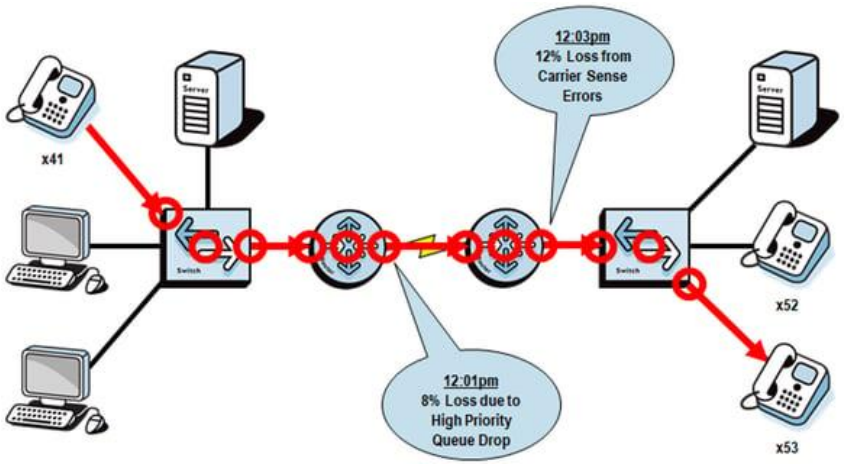

Figure 3: Packet loss in Voice over IP Calls [6]

However, with real-time protocols such as VoIP or video transmission, 10% loss will equate to a poor experience [6]. Now imagine if a call got dropped, or impacted by latency/jitter/loss, happened to be a 911 emergency call that loss would be unacceptable. As shown in Fig. 3, packet loss can happen for several reasons but for the purposes of this paper, we shall focus on QoS issues arising only due to noisy neighbors in a data center. If a server runs out memory or CPU resources due to unfair utilization by other VMs, then it may drop a call. The data centre operator needs to ensure that QoS is being maintained for all critical calls. This may require identification of VMs carrying VoIP traffic, and within that the call packets belonging to higher priority calls such as for 911 emergency are never dropped.

3. WAYS TO DETECT NOISY NEIGHBORS

Performance tests were conducted [2] with 24 users, 6 Public Cloud vendors, and using a custom-built application from which 351 samples were collected and analyzed as outlined in Table1.Nearly identical VMs were used across all Cloud Service Providers (CSP) representing the baseline of their offerings. For example, for AWS we used a T-series VM with 2 vCPUs, 0.5 GiB, Arm-based Graviron2 CPU. Its equivalent VMs were used for other Cloud vendors. We built a user-level application (called “MegaApp”) collects the metrics, using a real-time clock provided by the operating system. The application creates measurement threads and schedules them to run on all CPU cores on a server.

Table 1: Attributes of our study [2]

Attribute	Value
Number of users	24
Number of samples	351
Number of Cloud providers	6
Number of guest operating	2 (Linux and Windows)
Number of VMMs or host operating systems	6

These scores are dependent on the CPU, memory speed and data access architecture, storage technology (speed, hard disk vs. SSD), operating system, threads scheduling and management,

virtualization machine, virtualization guest OS, and any other applications running when the measurements are performed. The following components were measured:

- CPU score is a measure of how many seconds certain thousands of CPU instructions are complete. It is measured in real time by performing integer, floating-point, and string operations. Lower duration scores are better than the higher duration scores.

- Memory score is a measure of how many seconds certain thousands of memory instructions are complete. It is measured in real time by allocating blocks of memory and reading and writing to the allocated memory blocks. Lower duration scores are better than the higher duration scores.

- Storage score is a measure of how many seconds certain thousands of file creation, deletion, and input and output operations take to complete. It is measured in real time by creating and deleting files and reading and writing to the created files. Lower duration scores are better than the higher duration scores.

- Total score is a simple addition of CPU, memory, and storage scores. The units for each of these metrics in the seconds it took to execute the benchmark program. A higher value of this score indicates that the server is running slower.

$$CV = \sigma \text{ (Total Score)}/\mu \text{ (Total Score)}$$

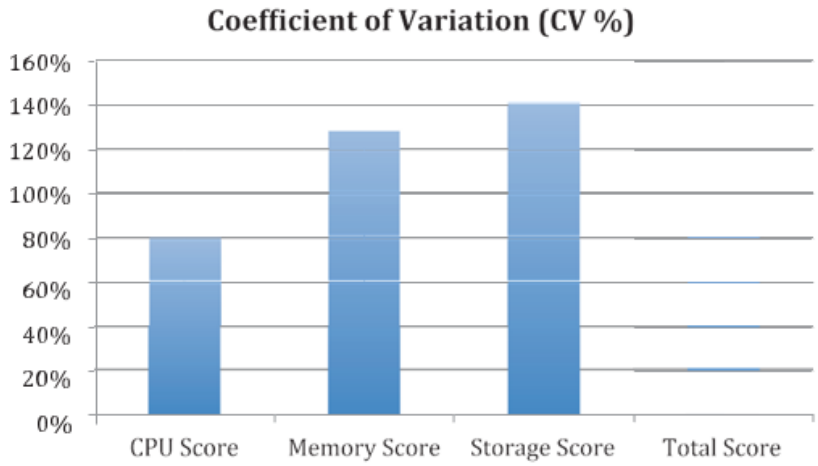

Figure 4: Variations of scores on a Cloud VM [2]

4. PREVENTING NOISY NEIGHBORS

It is non-trivial to detect a noisy neighbor, because performance in a shared server can depend on a host of factors. These may include a higher number of VMs placed on that server, no single of which is consuming an unfair share of resources, but collectively the load on the server is higher impacting the QoS for all VMs. Even such a situation will be detectable using the micro-benchmarking scheme proposed in the previous section. Question is the best way to avoid such a situation, as prevention is often better than the cure.

This can be accomplished by the datacenter operator by benchmarking available servers as described in the previous section, to determine how many units of computing, storage and networking load each can carry. Then, as different jobs arrive, these may get placed on various servers in a round-robin manner until the server has reached its capacity [10]. Residual capacity for each server can be constantly checked by running a microbenchmark to ensure that existing jobs are also getting their fair share of resources [2].

Another method is to collect metrics from various components in a server and see which VM is over consuming the resources. Then the offending VM can be isolated and migrated to another server. Alternatively, a victim VM which is carrying mission critical jobs, such as 911 emergency calls, can be migrated to a relatively lightly loaded server. However, such migration should be done with care to ensure that QoS for that VM is minimally impacted during the migration.

A Cloud operator can track usage by different customers to identify their resource usage patterns. This involves creating a database of resources utilized by various customers and their VMs in the datacenter over time. Typically, usage patterns tend to repeat as in the case of regression related workloads or VoIP users with certain peak times for calls etc. This usage data can be used to train an orchestration engine for new VM placement in a datacenter. If done correctly, this will ensure that VMs which tend to overuse certain types of resources will not be placed on the same server to avoid contention. Goal is to avoid all CPU, memory or network intensive jobs to be on the same server, so VMs do not compete for the same resource. Balanced loading implies a scheme that evenly distributes CPU, Memory and Network intensive jobs such that the mix on any given server has a balanced loading for each resource type [3] [5].

5. PHENOMENON AND CAUSES OF NOSY NEIGHBORS

Hardware resources shared by various users and VMS allow potential access to information from a target VM to a malicious VM. These shared resources include data busses, I/O ports, Cache memory, main memory, power supply lines and states in control circuitry. Below is a limited set of examples and not exhaustive list of causes.

- **Uncleared Cache:** When processes swap in an out of the execution stream, their data remains in the cache. It is sometimes possible for the swapped process to read cache locations containing data left behind by a previous process. This can occur at any level of the cache; however, it is easier on the higher levels of cache due to more sharing and larger size. The ease of attack depends upon the cache consistency policies [7]that are implemented for a specific system.

- **Uncleared or unprotected main memory:** Virtual memory maps a running process memory address to a different physical address. This was originally created to allow a process to address a larger address space than is physically implemented in the hardware. The use of virtual memory addressing has been expanded to allow multiple processes and VM's to access the same address space located in different physical memory locations. Low level (machine language) code can directly access physical memory by passing the virtual memory address mapping. Thus, a malicious process can read the memory of a victim process.

- **Main Memory Addressing:** A common problem in software is array or stack bounds checking. However, a malicious process can intentionally violate the array limit to address physical memory beyond its allocated size to access some target memory. Similarly, the stack, especially the routine return stack, can be accessed out of order such that it can reach beyond the attacker's process memory limits.

- **Control Flow and execution timing:** To improve performance, CPU's do speculative execution, which can load values and instructions ahead of the instruction pointer (IP) by guessing a path. This speculation uses an address that is already in the cache to save time. When the speculation is wrong, there is a miss, which is slower than a hit. The infamous Spectre and Meltdown attacks [8] used this concept to ascertain data values by measuring the time to access memory. This is called a side channel attack because the data is not directly read. Rather the data is indirectly ascertained by utilizing some side channel measurement, in this case the timing of execution.

- **Power measurements:** Another example of a side channel attack is to measure the power consumption of a chip. For example, Correlation Power Analysis can be used to find the secret key used in the AES algorithm [9].

6. DETECTING AND PREVENTING NOSY NEIGHBORS

Security includes both prevention of successful attacks and detection of potential attacks. Some attacks are easier to prevent than to detect. This can be a long-term problem, because malicious agents can keep trying to attack until they find a vulnerability. Without detecting these failed attempts, defenders may not be able to improve their defense from future attacks. Therefore, it is useful to detect and record whether an attack was successful or not. Basically, detection of attacks and potential attacks can be used to measure trust [3]. The failure of trust, or detection of an attack can have different levels of responses.

- **Uncleared Cache:** This simplest prevention of this attack is to clear the cache when it is released by a process being swapped out or finishing. However, this has a performance cost if every time a process exits the cache it needs to be cleared. This is even more complicated for the higher-level caches which are shared between processes. Some way must be used to only clear the memory locations assigned to the released process. One way to achieve this is to tag every instruction and data location with the associated process id (PID). This is a huge overhead in memory usage. This is used for prevention by only allowing matching PIDs to access data or instructions left by the same process.

- **Uncleared or unprotected main memory:** Just as with the caches, one approach is to clear the main memory when a process swaps out. However, this does not solve the problem of VM's accessing the same main memory. Because virtual memory maps a running process memory address to a different physical address, the hardware can check the PIDs, if they are noted in the translation look aside buffer which translates the addresses the access can be either prevented or monitored [11].

- **Main Memory Address bounds checking:** This is difficult because hardware does not know dimensions or boundaries set by the software. This requires some extra software modification to pass the information to the hardware for bounds checking. This should trigger a trap so the software can handle an exception, however the malicious process will attempt to ignore the trap. This is where detection is applied by keeping track of processes repeatedly violating the bounds and passing a trust evaluation of a PID to a hardware security checker.

- **Control Flow and execution timing:** A simple defense is to turn off speculative execution. This in fact was the original defense against Specter and meltdown attacks [8].However, this caused a huge performance penalty. So, later patches to the hardware

prevented the attacks. In the true spirit of competition, the attackers modified the attacks to defeat the patches. This back and forth between attackers and defenders has continued through several iterations.

- **Power measurements:** There are several methods applied to defeat power side channel attacks. The simplest one is to add a power averaging circuit that adds power consumption during the cycles of low power. This makes a flat power signature independent of what the secret data is. However, power consumption it a problem for computers already, and adding more power consumption is not always acceptable. Another Solution is to interchange process operations so the order of execution will not create a consistent power signature [9].

7. CONCLUSIONS AND FUTURE WORK

Noisy or Nosy neighbors pose serious issues for Cloud operators and users. Following steps are suggested proactively manage these issues:

- Predict a repetitive problem before it happens, based on past behavior patterns
- Job assignments for scheduling based on anticipated resource consumption
- Track internal metrics such as cache hits/misses, Memory I/O, Storage I/O, Network I/O saturation.

Once a VM is ascertained to be aggressor then it needs to be isolated or terminated. Similarly, a victim VM running a mission critical job should be migrated to another server to ensure its optimal performance.

REFERENCES

[1] Bhatt, P. C. P. An introduction to operating systems concepts and practice (GNU/LINUX) (4th ed., pp. 305–311). Prentice Hall India Pvt Ltd, New Delhi, Jan 2014 pp. 558–562, and pp. 681.

[2] Shankar, S., Acken, J. M., & Sehgal, N. K. (2017). Measuring Performance Variability in the Clouds. *IETE Technical Review*, *35*(6), 656–660. https://doi.org/10.1080/02564602.2017.1393353

[3] Cloud computing with security and scalability, NK Sehgal, PCP Bhatt, JM Acken. Springer, https://link.springer.com/book/10.1007/978-3-031-07242-0

[4] Dawson, Mark,Noisy Neighbor Effect: How to Manage it?Aug 19. 2021.https://www.jabperf.com/noisy-neighbor-effect-and-ways-of-handling-it/

[5] Mulia, W. D., Sehgal, N., Sohoni, S., Acken, J. M., Stanberry, C. L., & Fritz, D. J. (2013). Cloud Workload Characterization. IETE Technical Review, 30(5), 382–397. https://doi.org/10.4103/0256-4602.123121

[6] Path Solutions, Where Does VoIP Packet Loss Come from? Detecting and Preventing VoIP Packet Loss, April 4, 2019.https://www.pathsolutions.com/blog/voip-packet-loss-troubleshooting

[7] S. S, S. S. Garag, A. Phegade, D. Gusain and K. Varghese, "Design of a Multi-Core Compatible Linux Bootable 64-bit Out-of-Order RISC-V Processor Core," 2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID), Hyderabad, India, 2023, pp. 42-47, doi: 10.1109/VLSID57277.2023.00023.

[8] A. Prout et al., "Measuring the Impact of Spectre and Meltdown," 2018 IEEE High Performance extreme Computing Conference (HPEC), Waltham, MA, USA, 2018, pp. 1-5, doi: 10.1109/HPEC.2018.8547554.

[9] A. T. Mozipo and J. M. Acken, "Analysis of Countermeasures Against Remote and Local Power Side Channel Attacks using Correlation Power Analysis," in IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 6, pp. 5128-5142, Nov.-Dec. 2024, doi: 10.1109/TDSC.2024.3370711

[10] Aljobory K, Yazici MA. Edge Server Selection with Round-Robin-Based Task Processing in Multiserver Mobile Edge Computing. Sensors (Basel). 2025 May 30;25(11):3443. doi: 10.3390/s25113443. PMID: 40968968; PMCID: PMC12158191.

[11] A. Awad, A. Basu, S. Blagodurov, Y. Solihin and G. H. Loh, "Avoiding TLB Shootdowns Through Self-Invalidating TLB Entries," 2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT), Portland, OR, USA, 2017, pp. 273-287, doi: 10.1109/PACT.2017.38

AUTHORS

John M. Acken is a faculty member in the Electrical and Computer Engineering Department, Portland State University, Portland, OR. John received his BS and MS degrees in electrical engineering from Oklahoma State University and PhD degree in electrical engineering from Stanford University. His projects include technology and devices for information security and identity authentication. John has worked as an electrical engineer and manager at several companies, including the US Army, Sandia National Labs in Albuquerque, New Mexico, and Intel in Santa Clara, CA. John's time in the US Army was in the Army Security Agency, a branch of NSA during the Vietnam War.

Naresh K. Sehgal is the CTO at Securing the Cloud llc, and before that worked at NovaSignal Corp as the Sr. VP of Cloud Engineering and CISO. Before that he was at Intel for more than 31 years in various engineering and management roles. Naresh has earned a B.E. (Electrical Engineering) from Punjab Engineering College, M.S. and Ph.D. from Syracuse University in Computer Engineering. Naresh has taught Cloud Computing at Santa Clara University, where he also earned an MBA.