Stratus: Clouds with Microarchitectural Resource Management

Kaveh Razavi[*]
ETH Zürich
kaveh@ethz.ch

Animesh Trivedi[*]
VU Amsterdam
a.trivedi@vu.nl

Abstract

The emerging next generation of cloud services like Granular and Serverless computing are pushing the boundaries of the current cloud infrastructure. In order to meet the performance objectives, researchers are now leveraging low-level *microarchitectural* resources in clouds. At the same time these resources are also a major source of security problems that can compromise the confidentiality and integrity of sensitive data in multi-tenant shared cloud infrastructures. The core of the problem is the lack of *isolation* due to the unsupervised sharing of microarchitectural resources across different performance and security boundaries. In this paper, we introduce Stratus clouds that treat the isolation on microarchitectural elements as the key design principle when allocating cloud resources. This isolation improves both performance and security, but at the cost of reducing resource utilization. Stratus captures this trade-off using a novel abstraction that we call *isolation credit*, and show how it can help both providers and tenants when allocating microarchitectural resources using Stratus's declarative interface. We conclude by discussing the challenges of realizing Stratus clouds today.

1 Introduction

We are in the midst of a fundamental shift in cloud computing as researchers are pursuing the next-generation of services and infrastructure projects [12, 81, 97]. For example, cloud functions (FaaS, Serverless) enable developers to build bursty, highly-parallel and scalable applications [23,24,39]. Granular computing [51] proposes a new computing fabric consisting of large numbers (1k-1M) of small tasks at scale for a short burst of activity (1-10ms). Traditional monolithic services are now being broken down into hundreds of microservices [27]. Overall, these next-generation services aim to push the performance of current clouds by another order of magnitude.

To meet such performance demands, our computing platforms have also evolved, taking advantage of emerging hardware in commodity computing. Devices such as GPUs [73], FPGAs [71, 82], SmartNICs [22, 41], and programmable storage [17, 74] are now being used in cloud services and applications. Traditionally, these devices are managed by an operating system and a cloud-scale resource manager at the exposed *architectural* interfaces (e.g., number of cores, GPUs, DRAM, etc.). However, as the demand for high performance increases, the attention has gradually been shifting to reason about and even manage (directly or indirectly) *microarchitectural* resources[1] as well [21]. As an example, contention on the last level cache (LLC), a microarchitectural resource, can lead to sub-optimal performance in high-speed networks when building distributed applications [20, 94]. Similarly, mismanagement of various *on-chip* microarchitectural resources inside SmartNICs [18, 40], or unintentional cross-talk inside Non-Volatile Memories (NVM) devices [11, 45], can lead to significant performance degradations.

Performance, however, is not the only issue. Recent high-profile security attacks show that these microarchitectural resources can also be exploited by attackers to leak sensitive information [16, 46, 53, 58, 87, 88, 98] or inject faults in the data [72, 93]. Broadly speaking, these attacks break hardware isolation boundaries in shared microarchitectural resources like DRAM, caches, and instruction execution units to compromise systems. Beyond the CPU, researchers have also demonstrated significant attacks on/using storage [49], FPGA [48], and GPU [25, 103]. More worryingly, as seen recently, these attacks are also possible remotely over the network [50, 85, 86]. With the emergence of faster and diverse hardware, these issues will only worsen.

The core of the problem is the lack of isolation due to the *unsupervised sharing* of microarchitectural resources across different performance and security boundaries. The response from the community has been reactive: microarchitectural resources are currently managed in an ad hoc manner using a mix of techniques to improve either performance [13, 40] or security [31, 47, 55], but never both. For example, Apache Crail,

[*]Equal contributions.

[1]We collectively refer to internal (transparent) resources of the CPU as well as other modern devices as microarchitectural resources.

Resource	Microarchitectural
CPUs	Caches, TLBs, Hyperthreads, ALUs
SmartNICs [40,41]	Caches (memory, requests, connection), TLBs, RMT pipelines, DMA engines
NVM Storage [1,10,11]	Blocks, pages, intenral r/w ports, programmable cores, SRAM
GPUs [5,59,101]	Memories, caches, execution units
Switches [70,75]	SRAM and TCAM memories, Match-action Unit processors, ALUs

Table 1: Architectural and associated microarchitectural resources.

a distributed data store, which is designed for high performance with NVM and RDMA devices [83], has been shown to suffer from low-level microarchitectural attacks [86].

In this paper, we propose Stratus[2], a cloud framework to reason about the sharing of microarchitectural resources in a multi-tenant cloud in a principled manner. We approach this challenge by identifying microarchitectural *isolation* as the desired property on which security and performance properties can be built. Stratus proposes a declarative interface for tenants to specify their isolation constraints, which are evaluated by a cloud provider during resource allocation. Constraint-driven allocation is aided by Cloud Knowledge Base (CKB), which is a data store for storing and querying microarchitectural knowledge in a declarative fashion (similar to SKB in Barrelfish [7,77]). The simultaneous evaluation of security and performance constraints ensures that an optimization does not open a security vulnerability. In order to capture the value and effort of providing such microarchitectural-level isolation, we introduce the concept of *isolation credits*. Cloud providers can charge tenants in credits for satisfying their constraints in resource allocation, thus encouraging tenants to only specify the relevant constraints. From the provider's point of view, the cost of fulfilling a constraint helps them differentiate from competitors by innovating in building appropriate mechanisms. In comparison to previous efforts (microarchitectural management [2,4,13,28,52,89,90,94], declarative approaches [7,54,56,63,76,92,100], and performance profiling [15,16,21]) our proposal differs in (i) scale - not just a single application or machine, but for an entire datacenter; (ii) granularity - not just high-level architectural and operating system-level resources, but microarchitectural from a diverse set of on/off-CPU and in-network devices; (iii) scope - security and performance properties in an *end-to-end* manner.

Threat model. We assume a generic threat model against microarchitectural attacks that do not require physical access to the target. Examples include cache attacks [66,98], speculative execution attacks [87,88] and Rowhammer [14,26,72,93]. These attacks often assume co-residency with a victim process or VM, but recent advances have also made them practical against servers over the network [50,79,85]. Stratus aims to address these attacks by providing abstractions that enable isolation on microarchitectural resources.

[2]Stratus is a type of cloud which is found at very low levels.

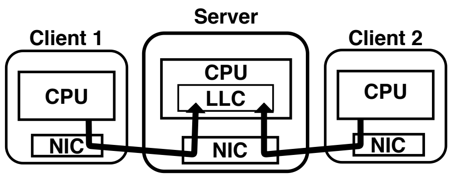

Figure 1: LLC sharing between two clients with DDIO [35].

2 The Case for Stratus

There are three primary trends that are pushing for a more principled approach towards fine-grained microarchitectural resource reasoning. The first trend is *increasing diversity*. With the push for heterogeneous computing, a diverse set of ISAs, accelerators, switches, programmable storage and smart networking devices have entered into mainstream computing [32]. Naturally, these devices also bring associated diverse microarchitectural resources (see Table 1) into the shared cloud computing paradigm. The second trend is the *push for multi-tenancy on modern devices*. After having developed single-tenant applications, now modern devices (RDMA, NVMs, FGPAs) are being deployed in a shared, multi-tenant cloud setting [36,43,44,62,70]. Consequently, they require careful attention towards resource *sharing*, which can have unintended performance implications [13,20,102]. The last trend is the evolving *security threats*. As many of recent high-profile security attacks have demonstrated that an unsupervised or misguided sharing of microarchitectural resources can lead to information leaks, and full system compromises. Such attacks become possible because (a) there is a misplaced trust in the hardware to deliver safe sharing through isolation [33]; (b) a system software lacks any direct visibility to reason about the sharing and isolation at the microarchitectural level.

One could argue that it should be the operating system (OS) on each node that is tasked for the management of microarchitectural resources. While this argument holds for applications that run on a single node, cloud services such as replication [43,84], storage [96], machine learning [37], and even OSes [80] run on a number of different nodes with a variety of accelerator devices. Furthermore, in-networking resources such as SRAM and processing elements from programmable switches are now also being used for building datacenter services [38,70]. Hence, reasoning about security and performance properties in an *end-to-end manner* requires looking beyond end-points, and instead taking a more holistic and distributed approach towards microarchitectural resource management.

LLC Sharing - A Motivating Example In order to sustain data rates of high-performance networks, modern Intel CPUs directly place network data in its LLC [34] (Figure 1). This design immediately raises the question of how a remote LLC (i.e., a microarchitectural resource) should be managed to avoid cross-talk and maximize the performance of multiple competing tenants. Current technologies such as Intel DDIO,

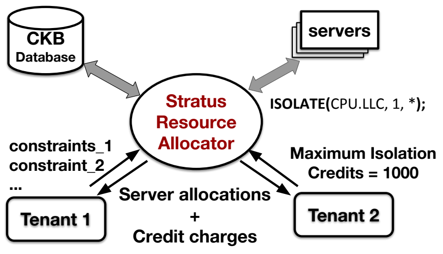
CKB
Database
servers
Stratus Resource Allocator
ISOLATE(CPU.LLC, 1, *);
constraints_1
constraint_2
...
Maximum Isolation Credits = 1000
Server allocations + Credit charges
Tenant 1
Tenant 2

Figure 2: The design of Stratus with CKB.

simply share the LLC slice that is dedicated to I/O traffic among all clients. Such a default policy (with low isolation) not only delivers sub-optimal performance [20, 94], but more worryingly enables side-channel attacks over the network to leak information [50]. The core of both performance and security problems, in this example, is the unsupervised sharing of the LLC. Is it possible to share the remote LLC to improve performance while preserving security?

3 Design of Stratus

Stratus is a cloud framework that aims to capture and reason about microarchitectural *isolation* in a principled manner. *The key insight in building Stratus is that security and performance are the two sides of isolation.* This isolation is expressed by constraints which are predicates attached to resources that a provider must satisfy when allocating those resources. Currently these isolation constraints are hidden underneath various resource allocation strategies for a large spectrum of computing abstractions offered by cloud providers (e.g., functions, containers, VMs IaaS). One point on this spectrum is FaaS-like clouds, where tenants only provide "functions" (without any constraints) and providers are free to optimize their utilization objectives when allocating resources for the function execution [91]. Another extreme can be imagined as a cloud which exposes all of its resources and their status to its tenants, offering them full control over allocations. In the middle, there are IaaS clouds where tenants provide constraints such as the number of cores or the amount of memory encoded in the VM types that the cloud provider offers. Without limiting the current resource allocation strategies, Stratus aims to complement them with microarchitectural constraints to capture performance and security properties. In Stratus, microarchitectural constraints (e.g., isolated LLC cache) are in the majority of the cases decoupled from the more coarse-grained architectural resources (e.g., a core). This allows Stratus to support all existing resource allocation schemes while satisfying tenant-provided microarchitectural constraints.

We envision that tenants provide a set of microarchitectural-specific constraints to Stratus. Stratus then finds a set of suitable servers that can satisfy resource allocation constraints by querying a database that captures available microarchitectural resources. This database is, in essence, similar to the database of architectural resources in popular cloud infrastructures such as OpenNebula [60]. After choosing servers, Stratus uses available mechanisms on each machine to isolate microarchitectural resources for a given tenant. In exchange for these services, the provider can charge the tenant. The amount depends upon the balance between the effort required from the provider and the perceived value of satisfying the given constraints for the tenant. We capture this balance using a new abstraction called *Isolation Credit*.

In the following sections we show how Stratus captures isolation constraints (§3.1), introduce isolation credit (§3.2), discuss how Stratus evaluates a tenant's constraints using our proposal of a cloud knowledge base (CKB) (§3.3), and uses existing mechanisms for enforcing isolation (§3.4). Figure 2 shows the overall interaction among these components in Stratus for principled microarchitectural resource allocation.

3.1 Capturing Tenants Requirements

Stratus allows tenants to express their isolation requirements in a declarative manner. An internal resource allocator uses constraint-logic programming (CLP) to analyze and satisfy the constraints. Constraints enable Stratus to reason in a principled manner if any isolation requirements are violated. Tenants specify isolation constraints using the following syntax:

Listing 1: Syntax of defining a constraint using ISOLATE

```
handle = ISOLATE(resource, scale, quantity);
```

A `resource` is a microarchitectural resource such as LLC, a NIC packet processor, etc. There are two types of microarchitectural resources, hard and soft. Hard resources are the ones that can be partitioned in space and used exclusively by a single tenant such as the LLC. Soft resources are contended in time, such as the DRAM bandwidth. The resources are modeled as they appear in the system topology where the top-level represents top-level architectural resources such as CPU, DRAM, or NIC. `scale` is a scalar quantity between `{0,1}` capturing the extent of the isolation requested. A zero value, which is the default for all resources, indicates no isolation constraints from a tenant and the provider is free to optimize for the maximum utilization. Hard microarchitectural resources can only take discrete values of 0 or 1, whereas soft resources can take any value in between. `quantity` is the minimum number of requested resources for which the constraints must be satisfied. For example, a tenant might only be interested in the first 64 requested cache sets of LLC for network traffic, and not beyond that.

Isolating microarchitectural resources is alone not enough to provide end-to-end isolation. Thus, Stratus allows attaching isolated resources to each other using the ATTACH operator:

Listing 2: Combining (AND) constraints using ATTACH

```
  ATTACH(handle1, handle2, ...);
```

This allows tenants of Stratus to properly isolate network clients of a DDIO-enabled server shown in Figure 1 using the following constraints:

Listing 3: *labeled* constraint, see Listings 1,2 for syntaxes

```
Tenant_i_constraints:
  h1 = ISOLATE(res=CPU.LLC, sc=1, qaunt=64);
  h2 = ISOLATE(res=NIC.*, sc=0, quant=*);
  ATTACH(h1, h2);
```

The wildcard expression (symbol `*`) enables Stratus to (i) extend the isolation to all microarchitectural elements of a given architectural resource; (ii) select any available partition of a given microarchitectural resource. The *labeled constraints* (e.g., $Tenant_i\_constraints$ in the example above) can be attached to a particular type of *allocation* such as virtual machines, containers, or FaaS functions. A tenant can specify:

Listing 4: Using constraints and labels with ALLOCATE

```
ALLOCATE cloud_resources,...where
constraint,...or label;
```

Each microarchitectural resource has multiple properties. For example, a CPU has a type (`x86` or `arm`), and a vendor_id (Intel or AMD). A tenant can also specify constraints on these properties. For example, if a particular attack happens only on Intel CPUs, and not on AMD (e.g., NetCAT [50]) then a client can use a CPU specific allocation constraint as:

Listing 5: Example of Intel-only constraints CPU allocation

```
ALLOCATE VM where IF(CPU.type == Intel)THEN
Tenant_i_constraints;
```

By providing these operations, Stratus offers an expressive and declarative interface to enable selective isolation of microarchitectural resources. This design enables providers to better utilize resources, and encourages tenants not to excessively over-constrain resource allocations. Next, we discuss how a new abstraction in Stratus captures the cost of isolation for both providers and tenants while simplifying microarchitectural resource management for tenants when desired.

3.2 Isolation Credit

There is an inherent tension between tenants and providers when it comes to providing isolation. Strong isolation leads to better performance (e.g., 99.9^{th} percentiles) and security, which are desired properties by tenants. In contrast, providers typically aim for high utilization by co-hosting tenants on shared infrastructure (minimum isolation) to maximize their profits. To capture this tension, we introduce the **isolation credit**, a currency that represents the amount of effort required from a provider to satisfy a tenant's isolation constraints as well as the value derived by the tenant for their workloads.

The cost of providing isolation is not the same across different resources. For example, isolating an entire LLC cache may require other cores on the same processor socket not to be utilized by other tenants. A provider, hence, asks for a certain amount of isolation credits for satisfying an isolation constraint. Tenants can buy credits from their cloud provider and spend them on their isolation requests. The abstraction of isolation credit *quantifies* and *monetizes* the effort required for isolation. It forces a tenant to make sensible isolation requests (the ones that generate the maximum value), and pushes a provider to innovate in low-overhead isolation mechanisms.

Spending isolation credits. Isolation credit can further be used as an abstraction for simplifying the low-level constraint interface of Stratus. For some tenants, microarchitectural constraints might be too low-level and detailed to enumerate. Instead, a tenant can simply provide Stratus with a credit budget that they are willing to spend, and Stratus will explore a strategy to simultaneously optimize performance, security, and utilization properties for the given budget. This strategy incentivises cloud providers to find solutions with efficient isolation mechanisms to offer differentiating services. If another cloud provider offers a better way to charge less isolation credits, the tenant is tempted to run on the second provider.

3.3 Evaluating Constraints

The overall goal of constraints evaluation is to allocate resources on machines that can satisfy all microarchitectural constraints specified by a tenant, while maximizing the utilization (or any other metric) for the cloud provider. To model information and solve constraint allocation, we take inspiration from the System Knowledge Base (SKB) component of the Barrelfish operating system [7,77]. SKB is a service inside Barrelfish for storing and querying hardware knowledge to solve resource allocation constraints. We propose building a distributed version of SKB, called Cloud Knowledge Base (CKB) where we will gather data, model cloud resources, and query for allocations. CKB will manage data gathered from two primary sources. First, factual information from literature and manuals such as the number of TLB entries on a CPU, the number of DRAM banks, or the number of parallel processing units in a SmartNIC. The number of such resources defines how many fully isolated discrete allocations Stratus can do. Second, online measurements to monitor utilization, occupancy, latencies and bandwidths of interconnects, etc. This information is used for soft resource allocation.

Naturally, one key concern is the performance of the constraints evaluation as the system scales. For a VM, a few seconds for allocation time could be fine, but it is not acceptable to launch FaaS functions where allocations must be done in 10s of milliseconds. General constraint solving (SAT solving) to find a solution is a NP-hard problem. However, Stratus has to check *satisfiability* for a given set of possible allocation solutions. Satisfiability checks of Stratus's constraints, which are in the CNF form, scale linearly with the number of constraints. For a given set of solutions (i.e. the list of servers), which can satisfy given constraints, a cloud provider can choose the one that maximizes its utilization or any other objective using existing mechanisms. Looking beyond server resources, in-network resources on all involved switches must also be evaluated, if defined. For example, given a set of in-switch constraints, connections between servers need to be routed differently, or it may be necessary to migrate or refresh previous allocations. In all cases, the space exploration is bounded by the isolation credits provided by a tenant.

3.4 Building on Available Mechanisms

A research question that Stratus addresses is *to what extent microarchitectural resources can be isolated between different tenants.* Microarchitectural resources by definition are not directly exposed to tenants and there is no explicit API for managing them. However, there are mechanisms that can be built to indirectly ensure that microarchitectural resources are allocated and used under given constraints.

As for the CPU resources, there are mechanisms for isolating microarchitectural resources in the memory hierarchy. For example, LLC allocation and sharing is a well-studied problem [20] and there are mechanisms such as page coloring [99] or explicit partitioning [55] that can be used to satisfy Stratus's isolation commands. Partitioning computational resources such as ALU ports inside a core is much more challenging and would require an entire core allocation for satisfying their isolation when needed [3, 9, 30].

Considering off-CPU devices, DRAM resources can be partitioned by the careful selection of memory pages [85]. SmartNICs (e.g., RDMA NICs) contain various on-NIC packet processing units (PUs), co-processors, connection/queue pairs (QPs) states, caches for work queue and memory translation entries, and DMA engines [40, 69]. A careful management of these resources is necessary to ensure high performance [40]. We expect that SmartNICs can either support resource isolation via state/session tracking (like in RDMA QPs), or hardware virtualization (PCI-e SRIOV RNICs), or software virtualization [44, 68]. Sharing of in-network computing resources is an active area of research, where there are very limited mechanisms for ensuring isolation between tenants [8].

For storage, Open-Channel SSDs can be used that expose the microarchitectural resources behind the block abstraction to a host for management [10, 11, 57] In such a design, a host becomes responsible for data placement (thus, implicitly controlling the mapping of a location to die, plane, and parallel I/O ports), sharing (write buffers among tenants), and error handling. We believe Stratus can use these mechanisms to enforce isolation among multiple tenants [29].

4 Open Challenges

Realizing a Stratus cloud requires addressing a number of challenges, three of which we discuss here:

Picking the Right Isolation Constraints. Stratus tenants can either identify microarchitectural resources directly or use isolation credits as the mechanism for achieving isolation. In the former case, a tenant needs to know which microarchitectural resources require isolation, and in the latter case, this task is given to Stratus. For achieving security, tenants or Stratus can provide different isolation policies that mitigate different attacks (e.g., avoiding the execution of other tenants on sibling hardware threads mitigates certain speculative execution attacks [87, 88]). These policies can also be provided by third parties as a collection of open policy libraries that tenants can use. Building these security policies against known attacks will be the first such attempt and it remains to be seen whether the current interface of Stratus is expressive enough for such a task. For achieving improved performance, tenants can again directly ask for isolated microarchitectural resources or provide Stratus the freedom to use isolation credits for improving performance. We envision novel distributed profile-guided tools that enable the tenants or Stratus to reason about the benefits of isolating certain microarchitectural resources versus the accrued cost via isolation credits.

Scalable Allocation. Resource allocation/selection lies in the critical path for fast booting, scheduling, and execution of components that make cloud-scale services. For example, reducing the booting time (including resource acquisitions) of FaaS functions is an active research area [65]. Not just limiting to the latency, operating new computing frameworks like Granular Computing require starting 10s of thousands of small tasks in a few milliseconds [51]. Can Stratus evaluate a tenant's constraints for all of these instances in a reasonable time, at scale? Furthermore, cloud providers may prefer to satisfy these isolation constraints next to other desired constraints such as increasing per-server utilizations. It remains to be seen whether these constraints from both providers and tenants can be solved in an efficient and scalable manner.

Enforcing Isolation. Stratus requires the possibility of isolating microarchitectural elements of any given architectural device that is shared between tenants. While this has proven to be possible for certain microarchitectural elements in CPU and DRAM, the rest – network, storage, in-network computing – is subject to research exploration and development of new hardware interfaces that allow the management of their microarchitectural resources when necessary. The mitigation of speculative execution attacks via the network may require the isolation of speculation effects which is currently a subject of active research [42, 78, 95]. Another challenge is developing novel abstractions that simplify the deployment of microarchitectural constraints. We envision Stratus to introduce microarchitectural resource containers akin to resource containers [6] that can be applied to a given tenant's execution context. Building efficient support for such abstractions and verifying their execution (e.g., using attestation) at the operating system-level are other challenges that need to be addressed in Stratus.

5 Contributions to Workshop Discussion

Expected feedback and discussion points

- *What are we missing from an operational point of view?* Running cloud-scale services is a complex operation and allocating resources is one of the many steps taken in a long process. What are the implications of Stratus decision making on end-to-end operational properties such as fault tolerance, load balancing, etc.?
- *Is isolation credit with a declarative interface the right abstraction for reasoning about microarchitectural resources?* A declarative interface is a powerful and simple interface which has been used to manage resources [54, 100], explore configurations [63], manage heterogeneity [64, 92] and in networking [56, 76]. Furthermore, a previous study has shown that one-dimensional scalar quantities (similar to isolation credits) can be effective in communicating a tenant's intention to its cloud provider [19]. Put together, we believe the abstractions we choose are powerful. We are, however, eager to hear counterarguments.
- *What is the cost of building an efficient and scalable CKB?* Constraint solving at scale, in a bounded time budget is a challenging problem. A recent work from Google shows that it is possible to build an efficient distributed system for solving graph reachability and membership evaluation problems for ACL management [67]. We take inspiration from such designs, but it remains to be seen what performance and scale CKB can deliver.
- In general, we are aiming to spark a discussion of how best to manage microarchitectural resources. Should we invest more in developing better policies and abstractions for the tenants to choose from or should we instead focus on building more expressive and fine-grained mechanisms?

Controversial questions

- *Is microarchitectural resource management really worth it?* In this paper we made a case for microarchitectural resource management in shared clouds. However, we understand that beyond technology, operational costs and complexities might put limits to the realization of this idea.
- *Are hardware manufacturers willing to change hardware to provide better microarchitectural interfaces?* CPU vendors already offer a limited form of mechanisms (Intel CAT, and cache invalidation instructions) to control microarchitectural resources. However, often policies are entangled with mechanisms [61]. Is there an opportunity here to identify the right interface for a variety of devices to expose their microarchitectural resources in a principled manner?
- *What is a principled approach for a new ISA to include microarchitectural resource management?* Moving forward, with a new ISA there is an opportunity to provide proper abstractions for microarchitectural resource management. There are many new trade-offs here: the interface can allocate resources on the fly with added hardware complexity, or the allocations can be reserved.

Acknowledgments

We thank our shepherd, Jon Howell, and the anonymous reviewers for their constructive comments. This work has been supported by NWO 016.Veni.192.262 and by Intel Corporation through the Side Channel Vulnerability ISRA.

References

[1] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy. Design Tradeoffs for SSD Performance. In *Proceedings of the USENIX 2008 Annual Technical Conference*, ATC'08, pages 57–70, Boston, Massachusetts, 2008.

[2] Jeongseob Ahn, Changdae Kim, Jaeung Han, Young-Ri Choi, and Jaehyuk Huh. Dynamic Virtual Machine Scheduling in Clouds for Architectural Shared Resources. In *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'12, pages 19–19, Boston, MA, 2012.

[3] Alejandro Cabrera Aldaya, Billy Bob Brumley, Sohaib ul Hassan, Cesar Pereida García, and Nicola Tuveri. Port Contention for Fun and Profit. In *2019 IEEE Symposium on Security and Privacy (S&P)*, pages 870–887, 2019.

[4] Nadav Amit. Optimizing the TLB Shootdown Algorithm with Page Access Tracking. In *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '17, pages 27–39, Santa Clara, CA, USA, 2017.

[5] Rachata Ausavarungnirun, Vance Miller, Joshua Landgraf, Saugata Ghose, Jayneel Gandhi, Adwait Jog, Christopher J. Rossbach, and Onur Mutlu. MASK: Redesigning the GPU Memory Hierarchy to Support Multi-Application Concurrency. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '18, pages 503–518, Williamsburg, VA, USA, 2018. ACM.

[6] Gaurav Banga, Peter Druschel, and Jeffrey C. Mogul. Resource Containers: A New Facility for Resource Management in Server Systems. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 45–58, New Orleans, Louisiana, USA, 1999.

[7] Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach, and Akhilesh Singhania. The multikernel: A new os architecture for scalable multicore systems. In *Proceedings of the*

ACM SIGOPS 22Nd Symposium on Operating Systems Principles, SOSP '09, pages 29–44, Big Sky, Montana, USA, 2009. ACM.

[8] Theophilus A. Benson. In-Network Compute: Considered Armed and Dangerous. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '19, page 216–224, Bertinoro, Italy, 2019. ACM.

[9] Atri Bhattacharyya, Alexandra Sandulescu, Matthias Neugschwandtner, Alessandro Sorniotti, Babak Falsafi, Mathias Payer, and Anil Kurmus. SMoTherSpectre: Exploiting Speculative Execution through Port Contention. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 785–800, London, United Kingdom, 2019. ACM.

[10] Matias Bjørling. Open-Channel Solid State Drives. `https://events.static.linuxfound.org/sites/events/files/slides/LightNVM-Vault2015.pdf`. Accessed: 2019-10-24.

[11] Matias Bjørling, Javier González, and Philippe Bonnet. LightNVM: The Linux Open-Channel SSD Subsystem. In *Proceedings of the 15th Usenix Conference on File and Storage Technologies*, FAST'17, page 359–373, Santa clara, CA, USA, 2017.

[12] Adrian M. Caulfield, Eric S. Chung, Andrew Putnam, Hari Angepat, Jeremy Fowers, Michael Haselman, Stephen Heil, Matt Humphrey, Puneet Kaur, Joo-Young Kim, Daniel Lo, Todd Massengill, Kalin Ovtcharov, Michael Papamichael, Lisa Woods, Sitaram Lanka, Derek Chiou, and Doug Burger. A Cloud-scale Acceleration Architecture. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-49, pages 7:1–7:13, Taipei, Taiwan, 2016. IEEE Press.

[13] Shuang Chen, Christina Delimitrou, and José F. Martínez. PARTIES: QoS-Aware Resource Partitioning for Multiple Interactive Services. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, pages 107–120, Providence, RI, USA, 2019. ACM.

[14] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks. In *2019 IEEE Symposium on Security and Privacy (S&P)*, May 2019.

[15] Christina Delimitrou and Christos Kozyrakis. Paragon: QoS-aware Scheduling for Heterogeneous Datacenters. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, pages 77–88, Houston, Texas, USA, 2013. ACM.

[16] Christina Delimitrou and Christos Kozyrakis. Bolt: I Know What You Did Last Summer... In The Cloud. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 599–613, Xi'an, China, 2017. ACM.

[17] Jaeyoung Do, Sudipta Sengupta, and Steven Swanson. Programmable Solid-state Storage in Future Cloud Datacenters. *Commun. ACM*, 62(6):54–62, May 2019.

[18] Aleksandar Dragojević, Dushyanth Narayanan, Orion Hodson, and Miguel Castro. FaRM: Fast Remote Memory. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 401–414, Seattle, WA, 2014.

[19] Vojislav Dukic and Ankit Singla. Happiness index: Right-sizing the cloud's tenant-provider interface. In *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, Renton, WA, July 2019.

[20] Alireza Farshin, Amir Roozbeh, Gerald Q. Maguire, Jr., and Dejan Kostić. Make the Most out of Last Level Cache in Intel Processors. In *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, pages 8:1–8:17, Dresden, Germany, 2019. ACM.

[21] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, pages 37–48, London, England, UK, 2012. ACM.

[22] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert Greenberg. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 51–66, Renton, WA, 2018.

[23] Sadjad Fouladi, Francisco Romero, Dan Iter, Qian Li, Shuvo Chatterjee, Christos Kozyrakis, Matei Zaharia, and Keith Winstein. From Laptop to Lambda: Outsourcing Everyday Jobs to Thousands of Transient Functional Containers. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 475–488, Renton, WA, 2019.

[24] Sadjad Fouladi, Riad S. Wahby, Brennan Shacklett, Karthikeyan Vasuki Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 363–376, Boston, MA, 2017.

[25] Pietro Frigo, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU. In *2018 IEEE Symposium on Security and Privacy (S&P)*, pages 195–210, 2018.

[26] Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. TRRespass: Exploiting the Many Sides of Target Row Refresh. In *2020 IEEE Symposium on Security and Privacy (S&P)*, May 2020.

[27] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 3–18, Providence, RI, USA, 2019. ACM.

[28] Qian Ge, Yuval Yarom, Tom Chothia, and Gernot Heiser. Time Protection: The Missing OS Abstraction. In *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, pages 1:1–1:17, Dresden, Germany, 2019. ACM.

[29] Javier González and Matias Bjørling. Multi-Tenant I/O Isolation with Open-Channel SSDs. *Nonvolatile Memory Workshop (NVMW)*, 2017.

[30] Ben Gras, Cristiano Giuffrida, Michael Kurth, Herbert Bos, and Kaveh Razavi. ABSynthe: Automatic Blackbox Side-channel Synthesis on Commodity Microarchitectures. In *Network and Distributed Systems Security (NDSS) Symposium 2020*, NDSS'20, 2020.

[31] Daniel Gruss, Julian Lettner, Felix Schuster, Olga Ohrimenko, Istvan Haller, and Manuel Costa. Strong and Efficient Cache Side-Channel Protection Using Hardware Transactional Memory. In *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, page 217–233, Vancouver, BC, Canada, 2017.

[32] John L. Hennessy and David A. Patterson. A New Golden Age for Computer Architecture. *Commun. ACM*, 62(2):48–60, January 2019.

[33] Tyler Hunt, Zhipeng Jia, Vance Miller, Christopher J. Rossbach, and Emmett Witchel. Isolation and Beyond: Challenges for System Security. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '19, page 96–104, Bertinoro, Italy, 2019. ACM.

[34] Intel. Intel Data Direct I/O Technology Overview. https://www.intel.co.jp/content/dam/www/public/us/en/documents/white-papers/data-direct-i-o-technology-overview-paper.pdf, 2012. Accessed: 2019-05-24.

[35] Intel Corporation. Intel data direct I/O technology (Intel DDIO): A primer. http://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/data-direct-i-o-technology-brief.pdf. Accessed: 2019-05-25.

[36] Zsolt István, Gustavo Alonso, and Ankit Singla. Providing Multi-tenant Services with FPGAs: Case Study on a Key-Value Store. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pages 119–1195, 2018.

[37] Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Wencong Xiao, and Fan Yang. Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 947–960, Renton, WA, July 2019.

[38] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. NetCache: Balancing Key-Value Stores with Fast In-Network Caching. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 121–136, Shanghai, China, 2017. ACM.

[39] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Jayant Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion

Stoica, and David A. Patterson. Cloud Programming Simplified: A Berkeley View on Serverless Computing. *CoRR*, abs/1902.03383, 2019.

[40] Anuj Kalia, Michael Kaminsky, and David G. Andersen. Design Guidelines for High Performance RDMA Systems. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 437–450, Denver, CO, 2016.

[41] Antoine Kaufmann, SImon Peter, Naveen Kr. Sharma, Thomas Anderson, and Arvind Krishnamurthy. High Performance Packet Processing with FlexNIC. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, pages 67–81, Atlanta, Georgia, USA, 2016. ACM.

[42] Khaled N. Khasawneh, Esmaeil Mohammadian Koruyeh, Chengyu Song, Dmitry Evtyushkin, Dmitry Ponomarev, and Nael Abu-Ghazaleh. SafeSpec: Banishing the Spectre of a Meltdown with Leakage-Free Speculation. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, Las Vegas, NV, USA, 2019. ACM.

[43] Daehyeok Kim, Amirsaman Memaripour, Anirudh Badam, Yibo Zhu, Hongqiang Harry Liu, Jitu Padhye, Shachar Raindel, Steven Swanson, Vyas Sekar, and Srinivasan Seshan. Hyperloop: Group-based NIC-offloading to Accelerate Replicated Transactions in Multi-tenant Storage Systems. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 297–312, Budapest, Hungary, 2018. ACM.

[44] Daehyeok Kim, Tianlong Yu, Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Shachar Raindel, Chuanxiong Guo, Vyas Sekar, and Srinivasan Seshan. Freeflow: Software-Based Virtual RDMA Networking for Containerized Clouds. In *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation*, NSDI'19, page 113–125, Boston, MA, USA, 2019.

[45] Ana Klimovic, Heiner Litz, and Christos Kozyrakis. ReFlex: Remote Flash = Local Flash. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 345–359, Xi'an, China, 2017. ACM.

[46] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre Attacks: Exploiting Speculative Execution. In *2019 IEEE Symposium on Security and Privacy (S&P)*, pages 1–19, 2019.

[47] Radhesh Krishnan Konoth, Marco Oliverio, Andrei Tatar, Dennis Andriesse, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. ZebRAM: Comprehensive and Compatible Software Protection Against Rowhammer Attacks. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, page 697–710, Carlsbad, CA, USA, 2018.

[48] Jonas Krautter, Dennis R. E. Gnad, and Mehdi Baradaran Tahoori. FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):44–68, 2018.

[49] Anil Kurmus, Nikolas Ioannou, Matthias Neugschwandtner, Nikolaos Papandreou, and Thomas Parnell. From random block corruption to privilege escalation: A filesystem attack vector for rowhammer-like attacks. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, Vancouver, BC, 2017.

[50] Michael Kurth, Ben Gras, Dennis Andriesse, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. NetCAT: Practical Cache Attacks from the Network. In *2020 IEEE Symposium on Security and Privacy (S&P)*, 2020.

[51] Collin Lee and John Ousterhout. Granular Computing. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '19, pages 149–154, Bertinoro, Italy, 2019. ACM.

[52] Arnaud Lefray, Eddy Caron, Jonathan Rouzaud-Cornabas, and Christian Toinard. Microarchitecture-Aware Virtual Machine Placement Under Information Leakage Constraints. In *Proceedings of the 2015 IEEE 8th International Conference on Cloud Computing*, CLOUD '15, pages 588–595, Washington, DC, USA, 2015. IEEE Computer Society.

[53] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 973–990, Baltimore, MD, August 2018.

[54] Changbin Liu, Boon Thau Loo, and Yun Mao. Declarative Automated Cloud Resource Orchestration. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SOCC '11, pages 26:1–26:8, Cascais, Portugal, 2011. ACM.

[55] Fangfei Liu, Qian Ge, Yuval Yarom, Frank Mckeen, Carlos Rozas, Gernot Heiser, and Ruby B Lee. CATalyst: Defeating Last-Level Cache Side Channel Attacks in Cloud Computing. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, HPCA'16, pages 406–418, 2016.

[56] Boon Thau Loo, Joseph M. Hellerstein, Ion Stoica, and Raghu Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 289–300, Philadelphia, Pennsylvania, USA, 2005. ACM.

[57] Youyou Lu, Jiwu Shu, and Weimin Zheng. Extending the Lifetime of Flash-based Storage Through Reducing Write Amplification from File Systems. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies*, FAST'13, pages 257–270, San Jose, CA, 2013.

[58] A. T. Markettos, R. N. M. Watson, S. W. Moore, P. Sewell, and P. G. Neumann. Through Computer Architecture, Darkly. *Commun. ACM*, 62(6):25–27, May 2019.

[59] Xinxin Mei and Xiaowen Chu. Dissecting GPU Memory Hierarchy Through Microbenchmarking. *IEEE Transactions on Parallel and Distributed Systems*, 28(1):72–86, January 2017.

[60] Dejan Milojicic, Ignacio M. Llorente, and Ruben S. Montero. Opennebula: A cloud management tool. *IEEE Internet Computing*, 15(2):11–14, March 2011.

[61] Jeffrey C. Mogul, Andrew Baumann, Timothy Roscoe, and Livio Soares. Mind the Gap: Reconnecting Architecture and OS Research. In *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, HotOS'13, page 1, Napa, California, 2011.

[62] Mihir Nanavati, Jake Wires, and Andrew Warfield. Decibel: Isolation and Sharing in Disaggregated Rack-Scale Storage. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, page 17–33, Boston, MA, USA, 2017.

[63] Sanjai Narain, Gary Levin, Sharad Malik, and Vikram Kaul. Declarative Infrastructure Configuration Synthesis and Debugging. *Journal of Network and Systems Management*, 16(3):235–258, Sep 2008.

[64] Edmund B. Nightingale, Orion Hodson, Ross McIlroy, Chris Hawblitzel, and Galen Hunt. Helios: Heterogeneous Multiprocessing with Satellite Kernels. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 221–234, Big Sky, Montana, USA, 2009. ACM.

[65] Edward Oakes, Leon Yang, Dennis Zhou, Kevin Houck, Tyler Harter, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. SOCK: Rapid Task Provisioning with Serverless-Optimized Containers. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 57–70, Boston, MA, July 2018.

[66] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Countermeasures: The Case of AES. In *Proceedings of the 2006 The Cryptographers' Track at the RSA Conference on Topics in Cryptology*, CT-RSA'06, pages 1–20, San Jose, CA, 2006. Springer-Verlag.

[67] Ruoming Pang, Ramón Cáceres, Mike Burrows, Zhifeng Chen, Pratik Dave, Nathan Germer, Alexander Golynski, Kevin Graney, Nina Kang, Lea Kissner, and et al. Zanzibar: Google's Consistent, Global Authorization System. In *Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '19, page 33–46, Renton, WA, USA, 2019.

[68] Jonas Pfefferle, Patrick Stuedi, Animesh Trivedi, Bernard Metzler, Ionnis Koltsidas, and Thomas R. Gross. A Hybrid I/O Virtualization Framework for RDMA-Capable Network Interfaces. In *Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '15, page 17–30, Istanbul, Turkey, 2015. ACM.

[69] Phitchaya Mangpo Phothilimthana, Ming Liu, Antoine Kaufmann, Simon Peter, Rastislav Bodik, and Thomas Anderson. Floem: A Programming System for NIC-Accelerated Network Applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 663–679, Carlsbad, CA, October 2018.

[70] Dan R. K. Ports and Jacob Nelson. When Should The Network Be The Computer? In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '19, pages 209–215, Bertinoro, Italy, 2019. ACM.

[71] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo-Young Kim, Sitaram Lanka, James Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. A Reconfigurable Fabric for Accelerating Large-scale Datacenter Services. In *Proceeding of the 41st Annual*

International Symposium on Computer Architecture, ISCA '14, pages 13–24, Minneapolis, Minnesota, USA, 2014. IEEE Press.

[72] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip Feng Shui: Hammering a Needle in the Software Stack. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 1–18, Austin, TX, USA, 2016.

[73] Christopher J. Rossbach, Jon Currey, Mark Silberstein, Baishakhi Ray, and Emmett Witchel. PTask: Operating System Abstractions to Manage GPUs As Compute Devices. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 233–248, Cascais, Portugal, 2011. ACM.

[74] Zhenyuan Ruan, Tong He, and Jason Cong. INSIDER: Designing In-Storage Computing System for Emerging High-Performance Drive. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 379–394, Renton, WA, 2019.

[75] Amedeo Sapio, Ibrahim Abdelaziz, Abdulla Aldilaijan, Marco Canini, and Panos Kalnis. In-Network Computation is a Dumb Idea Whose Time Has Come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, pages 150–156, Palo Alto, CA, USA, 2017. ACM.

[76] Brandon Schlinker, Radhika Niranjan Mysore, Sean Smith, Jeffrey C. Mogul, Amin Vahdat, Minlan Yu, Ethan Katz-Bassett, and Michael Rubin. Condor: Better Topologies Through Declarative Design. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 449–463, London, United Kingdom, 2015. ACM.

[77] Adrian L. Schüpbach. *Tackling OS Complexity with Declarative Techniques*. PhD thesis, ETH Zurich, 2012. https://www.research-collection.ethz.ch/handle/20.500.11850/61055.

[78] Michael Schwarz, Moritz Lipp, Claudio Canella, Robert Schilling, Florian Kargl, and Daniel Gruss. ConTExT: A Generic Approach for Mitigating Spectre. In *Network and Distributed Systems Security (NDSS) Symposium 2020*, NDSS'20, 2020.

[79] Michael Schwarz, Martin Schwarzl, Moritz Lipp, Jon Masters, and Daniel Gruss. NetSpectre: Read Arbitrary Memory over Network. In *European Symposium on Research in Computer Security*, ESORICS'19, 2019.

[80] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 69–87, Carlsbad, CA, 2018.

[81] Vishal Shrivastav, Asaf Valadarsky, Hitesh Ballani, Paolo Costa, Ki Suh Lee, Han Wang, Rachit Agarwal, and Hakim Weatherspoon. Shoal: A Network Architecture for Disaggregated Racks. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 255–270, Boston, MA, 2019.

[82] Ran Shu, Peng Cheng, Guo Chen, Zhiyuan Guo, Lei Qu, Yongqiang Xiong, Derek Chiou, and Thomas Moscibroda. Direct Universal Access: Making Data Center Resources Available to FPGA. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 127–140, Boston, MA, 2019.

[83] Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, Ana Klimovic, Adrian Schuepbach, and Bernard Metzler. Unification of Temporary Storage in the NodeKernel Architecture. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 767–782, Renton, WA, July 2019.

[84] Yacine Taleb, Ryan Stutsman, Gabriel Antoniu, and Toni Cortes. Tailwind: Fast and Atomic RDMA-based Replication. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 851–863, Boston, MA, 2018.

[85] Andrei Tatar, Radhesh Krishnan Konoth, Elias Athanasopoulos, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Throwhammer: Rowhammer Attacks over the Network and Defenses. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 213–226, Boston, MA, 2018.

[86] Shin-Yeh Tsai, Mathias Payer, and Yiying Zhang. Pythia: Remote Oracles for the Masses. In *28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA, 2019.

[87] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *Proceedings of the 27th USENIX Security Symposium*, August 2018.

[88] Stephan van Schaik, Alyssa Milburn, Sebastian Österlund, Pietro Frigo, Giorgi Maisuradze, Kaveh Razavi, Herbert Bos, and Cristiano Giuffrida. RIDL: Rogue in-flight data load. In *2019 IEEE Symposium on Security and Privacy (S&P)*, pages 88–105, May 2019.

[89] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. Graviton: Trusted Execution Environments on GPUs. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 681–696, Carlsbad, CA, October 2018.

[90] Hui Wang, Canturk Isci, Lavanya Subramanian, Jongmoo Choi, Depei Qian, and Onur Mutlu. A-DRM: Architecture-aware Distributed Resource Management of Virtualized Clusters. In *Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '15, pages 93–106, Istanbul, Turkey, 2015. ACM.

[91] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. Peeking Behind the Curtains of Serverless Platforms. In *Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '18, pages 133–145, Boston, MA, USA, 2018.

[92] Yaron Weinsberg, Danny Dolev, Tal Anker, Muli Ben-Yehuda, and Pete Wyckoff. Tapping into the Fountain of CPUs: On Operating System Support for Programmable Devices. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIII, pages 179–188, Seattle, WA, USA, 2008. ACM.

[93] Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, Austin, TX, USA, 2016.

[94] Cong Xu, Karthick Rajamani, Alexandre Ferreira, Wesley Felter, Juan Rubio, and Yang Li. dCat: Dynamic Cache Management for Efficient, Performance-sensitive Infrastructure-as-a-service. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, pages 14:1–14:13, Porto, Portugal, 2018. ACM.

[95] Mengjia Yan, Jiho Choi, Dimitrios Skarlatos, Adam Morrison, Christopher W. Fletcher, and Josep Torrellas. InvisiSpec: Making Speculative Execution Invisible in the Cache Hierarchy. In *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-51, page 428–441, Fukuoka, Japan, 2018. IEEE Press.

[96] Jian Yang, Joseph Izraelevitz, and Steven Swanson. Orion: A Distributed File System for Non-Volatile Main Memory and RDMA-Capable Networks. In *17th USENIX Conference on File and Storage Technologies (FAST 19)*, pages 221–234, Boston, MA, 2019.

[97] Tian Yang, Robert Gifford, Andreas Haeberlen, and Linh Thi Xuan Phan. The Synchronous Data Center. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '19, pages 142–148, Bertinoro, Italy, 2019. ACM.

[98] Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-channel Attack. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 719–732, San Diego, CA, 2014.

[99] Ying Ye, Richard West, Zhuoqun Cheng, and Ye Li. COLORIS: A Dynamic Cache Partitioning System Using Page Coloring. In *2014 23rd International Conference on Parallel Architecture and Compilation Techniques (PACT)*, PACT'14, pages 381–392, 2014.

[100] Qin Yin, Adrian Schüpbach, Justin Cappos, Andrew Baumann, and Timothy Roscoe. Rhizoma: A Runtime for Self-deploying, Self-managing Overlays. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '09, pages 10:1–10:20, Urbanna, Illinois, 2009. Springer-Verlag.

[101] Xiuxia Zhang, Guangming Tan, Shuangbai Xue, Jiajia Li, Keren Zhou, and Mingyu Chen. Understanding the GPU Microarchitecture to Achieve Bare-Metal Performance Tuning. In *Proceedings of the 22Nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '17, pages 31–43, Austin, Texas, USA, 2017. ACM.

[102] Yiwen Zhang, Juncheng Gu, Youngmoon Lee, Mosharaf Chowdhury, and Kang G. Shin. Performance Isolation Anomalies in RDMA. In *Proceedings of the Workshop on Kernel-Bypass Networks*, KBNets '17, page 43–48, Los Angeles, CA, USA, 2017. ACM.

[103] Zhiting Zhu, Sangman Kim, Yuri Rozhanski, Yige Hu, Emmett Witchel, and Mark Silberstein. Understanding The Security of Discrete GPUs. In *Proceedings of the General Purpose GPUs*, GPGPU-10, pages 1–11, Austin, TX, USA, 2017. ACM.