A Double-Edged Sword: Security Threats and Opportunities in One-Sided Network Communication

Shin-Yeh Tsai, Yiying Zhang
Purdue University

Abstract

One-sided network communication technologies such as RDMA and NVMe-over-Fabrics are quickly gaining adoption in production software and in datacenters. Although appealing for their low CPU utilization and good performance, they raise new security concerns that could seriously undermine datacenter software systems building on top of them. At the same time, they offer unique opportunities to help enhance security. Indeed, one-sided network communication is a *double-edged sword* in security. This paper presents our insights into security implications and opportunities of one-sided communication.

1 Introduction

Traditional network communication in datacenters takes a *two-sided* form where both the sender and the receiver machines' CPU and software (OS or user-level programs) are involved in processing network requests. Several recent technologies in datacenters enable *one-sided* communication where the receiving machine's CPU and software are completely bypassed in the processing of incoming network requests. Instead, network hardware devices handle incoming requests and directly read/write into host machine's memory (*e.g.*, RDMA, Gen-Z [12], Omni-Path [4]), storage (*e.g.*, NVMe-over-Fabrics [37]), or GPU memory (*e.g.*, GPUDirect [9]).

Because of its low CPU utilization and low-latency performance, one-sided communication has been adopted in distributed in-memory systems [3, 10], fast storage systems [37], and new architectures like resource disaggregation [20, 45]. Prior research efforts and production systems have focused on one-sided communication's performance, scalability, programmability, and network protocols, ignoring one important factor in datacenters, *security*.

We call for attention to the security aspects of one-sided network communication. We study the basic communication patterns and real hardware implementations of one-sided communication to understand its implications in security. Based on our findings, we draw several key insights in fundamental issues, challenges, and potential solutions in achieving secure one-sided communication. The most interesting high-level insight is that one-sided communication is a double-edged sword in security: it can cause security threats and offer opportunities in enhancing security at the same time.

First, one-sided communication's feature of not having any software processing at the receiving side poses unique threats in distributed systems. For example, malicious clients can read or write to remote storage servers with one-sided network requests without being noticed, making it hard to account for errors. Attackers can also easily launch Denial of Service attacks using one-sided communication to swamp the network or the target machine's memory or storage resources.

Second, in order to bypass host processors, hardware devices that enable one-sided communication have to implement functionalities that are traditionally built in software. We identified several security issues in one-sided hardware, some caused by features added for better performance, some by the need to bypass host CPU, and some by ill-designed implementation.

Finally, one-sided communication provides a good opportunity to hide accesses from software, making it easier to achieve user privacy in datacenters. We designed a secure data store system that leverages one-sided communication to enhance the performance of traditional oblivious RAM systems while delivering the same security guarantees.

This paper makes the following contributions:

- As far as we know, this is the first work to discuss security aspects in one-sided communication.
- We identified fundamental limitations in one-sided communication that can pose security threats.
- We discovered new security issues in real one-sided hardware devices.
- We leverage one-sided communication to enhance the performance of an ORAM system.

2 One-Sided Communication in Datacenters

This section provides a background on one-sided communication and applications built based on it.

One-Sided Communication and Enabling Technologies. One-sided communication is a type of network communication pattern where only the sender side's software is involved in network request processing but not the receiver side (thus the term "one-sided"). One-sided communication can take different forms with different technologies and hardware. The most famous and popular technology is RDMA, a network technology that allows senders to access memory on remote machines directly without the involvement of remote

machines' CPUs. Similar to RDMA, Omni-Path [4] is a high-performance fabric architecture developed by Intel to support one-sided network communication. Gen-Z [12] is another interconnect technology designed for fast, direct access to a small (*e.g.*, the scale of a rack) pool of memory devices and it supports one-sided accesses to remote memory.

Apart from memory-based systems, one-sided communication is also used to access storage systems and accelerators. NVMe-over-Fabrics [37] is a technology that allows direct network access to NVMe-based storage devices. GPUDirect [9] is a technology that allows direct access to remote GPU memory from the network.

Applications Based on One-Sided Communication. Compared with traditional two-sided communication, one-sided communication offers key advantages in reducing CPU utilization and in improving performance. These benefits fit datacenters' performance and cost needs, making one-sided communication appealing in recent datacenter systems. For example, Microsoft Azure and Alibaba have deployed RDMA in large, production scale [2, 54]. There has also been a host of research work in RDMA and other one-sided technologies in recent years.

Below, we list a representative set of applications that are based on or support one-sided communication. These include in-memory key-value stores [10, 11, 26, 38, 39], in-memory databases and transactional systems [6, 59, 63], graph processing systems [48, 61], consensus implementations [41, 57], distributed non-volatile memory systems [30, 46, 64], remote swap systems [1, 17], NVMe-based storage systems [18, 31], and resource disaggregation systems [19].

All these previous systems focus on the performance, scalability, cost, programmability, and correctness of one-sided communication. The rest of this paper will demonstrate that one-sided communication presents many new security issues and opportunities and we call for attention to the security aspect of one-sided communication.

3 Threats of One-Sided Communication

While the processing needs at receiver nodes increase CPU utilization in two-sided communication, the receiver's processing software stack provides the means to implement various security defenses. In contrast, the lack of receiver involvement in one-sided network communication poses several security threats in datacenter environments. This session discusses two fundamental threats of one-sided communication.

Threats to accountability. In distributed, multi-tenant datacenter environments, many parties can potentially cause errors (*e.g.*, node failures, data corruption, stealing information). It is important to *account* for errors when they happen. Accountability enables the pinpointing of the party responsible for a problem and allows other parties to be proven innocent [42]. In a threat model where a *server* hosts data that multiple *client*s can read and write, an attacker can be one of the clients that desire to write malicious data or read other clients' data without being noticed.

Under two-sided communication, the receiver handles all incoming network requests and can identify their senders, providing a means for accountability. However, it is fundamentally difficult for one-sided communication to be accountable, because CPU and software are completely bypassed at the receiving side. Attackers can exploit this vulnerability to perform one-sided network operations without being detected. For example, in RDMA-based in-memory storage systems that support one-sided writes [6, 56, 63], an attacker client can write malicious data to any locations in the store without being detected. One possible way benign clients can avoid reading unaccountable data is to authenticate the writer of the data with encryption keys. However, this mechanism needs a fair amount of computation, and the performance overhead is especially large considering one-sided networks' high speeds [32, 36].

Threats in denial of service. Without any processing at the receiving side, one-sided communication also makes it hard to throttle the speed of network requests from any particular sender. This limitation can be exploited to launch different types of Denial of Service (DoS) attacks [43]. Our threat model here is a cloud environment where an attacker has one-sided network accesses to a server that provides some service (*e.g.*, an RDMA-based key-value store, GPU acceleration with GPUDirect) to multiple clients. An attacker can flood the network to the server with a large number of one-sided network requests without being detected.

Another potential DoS attack that is different from traditional network DoS attacks is to exhaust a server's hardware resources that are used to provide one-sided communication services. For example, RDMA NICs (*RNIC*s), devices that enable one-sided RDMA operations, store metadata for network connections and memory spaces in on-NIC SRAM so that RNICs can process incoming one-sided requests without involving the host machine. Attackers can fill the on-NIC SRAM by accessing many different memory spaces, forcing the receiver's RNIC to evict victims' metadata. To make it worse, the server cannot tell which client is the attacker, because the attacker's one-sided network traffic cannot be detected by the server.

Discussion and defenses. Attacks that can successfully exploit the above two vulnerabilities rely on the assumption that their traffic cannot be sniffed by trusted parties. We believe this assumption to be reasonable because most packet sniffing tools require the *root* privilege to run and datacenter administrators usually prohibit packet sniffing. For example, packet sniffing is prohibited by RNIC by default and requires the RNIC administrator privilege to change the default configuration [35, 53, 55].

While many traditional defense mechanisms may not work or will break the one-sided communication pattern, we identified two possible directions for future defenses. First, at

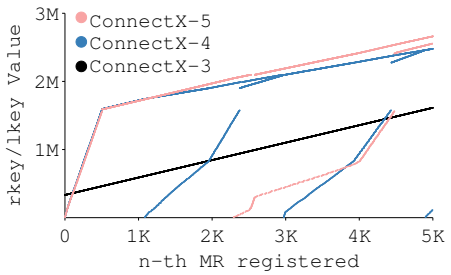

Figure 1: **MR Key Value** *X axis represents the sequence of MRs as they are generated in time.*

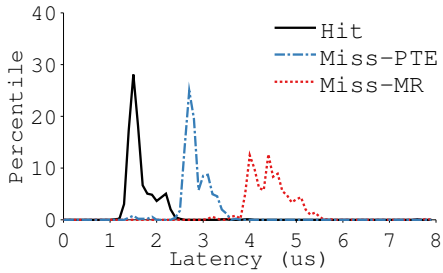

Figure 2: **Timing Differences in ConnectX-5.**

the sender, we can employ defense mechanisms to monitor or control network activities before network requests are sent out with the help of an intermediate layer like LITE [56] or with traffic dump tools [35, 53]. Second, at the receiver, we can enhance the hardware devices that handle one-sided requests with better security features, for example, by programming SmartNICs.

4 Vulnerabilities in One-Sided Hardware

To be able to bypass CPU at the receiver machine, hardware devices that enable one-sided communication need to implement many functionalities that are traditionally implemented in software such as permission checking and address mapping. However, hardware is not flexible and most existing one-sided hardware devices are not programmable. It is hard to add security features in hardware and security is usually an afterthought in hardware design.

A major threat model we envision is a cloud environment that provides in-memory data store services to cloud users, similar to the threat model in Section 3. A victim accesses data that a server hosts by issuing one-sided network requests from a client machine. The attacker runs on another client machine and is another user of the cloud data store service (*i.e.*, the attacker has no access to the victim or the server machines).

This section presents three real cases of security issues in one-sided hardware devices. These security vulnerabilities could potentially be exploited to launch real attacks with the above threat model, but we leave the design of real attacks for future work.

Case 1: hardware-managed "keys" are not secret. RNICs use a pair of "keys", *lkey* and *rkey*, to protect the local and remote access of each *memory region* or *MR*. An RNIC of a machine generates and stores lkey and rkey (together with the memory address) at the time when a user process registers an MR on the machine. Applications running on other machines use the virtual memory address of a registered MR and its rkey to access it. When receiving an RDMA request, the RNIC checks its access permission using the rkey.

Surprisingly, we discovered that RNICs generate rkeys in a predictable, sequential pattern with Mellanox ConnectX-3, ConnectX-4, and ConnectX-5 RNICs, the three most popular generations of RNICs. Figure 1 plots the values of lkey/rkey of the first 5000 MRs that are registered at a host in the order of the registration time (lkey and rkey have exactly the same values for all MRs). For all the three generations of RNICs, there are clear and easy-to-guess patterns in lkey/rkey values. Moreover, the first 500 MRs have sequentially increasing lkey/rkey values.

Most RDMA-based in-memory storage systems use a small number of large MRs [10, 56], making it even easier for attackers to guess MR keys. After guessing both the rkey and the memory address of an MR (the latter can be guessed in a similar way as traditional buffer-overflow attacks), attackers can gain full access to the MR, overwriting victims' memory content or stealing their data.

Case 2: side channels in RNICs. A key functionality that all RNICs support for one-sided communication is to map from virtual memory addresses to DMA (physical) addresses. These mappings are essentially page table entries (PTEs) that are stored in host machines' DRAM. For better performance, RNICs cache hot PTEs in its on-board SRAM. In addition to PTEs, RNICs also cache MR metadata such as lkeys and rkeys for hot MRs. An RNIC fetches metadata from its host machine's main memory when receiving RDMA requests whose metadata is not in the RNIC's SRAM.

We evaluated one-sided RDMA requests' latency when both the data's PTE and MR metadata is in SRAM (hit), when the data's PTE is not in SRAM (PTE miss), and when the data's MR metadata is not in SRAM (MR miss). Figure 2 presents the timing differences of these three cases over 1000 trials of each case. There is a clear difference between hits and misses, which can be used to establish side channels on RNICs. Since RDMA accesses from different applications share the same RNIC SRAM, it is possible to further build real side-channel and covert-channel attacks based on these timing channels.

Case 3: exposing physical memory addresses to remote machines. Another case where RNIC design can threat security is a feature designed to improve performance. By default,

user processes register MRs with RNICs using virtual memory addresses in their address spaces and RNICs store the virtual to physical memory address mapping in their SRAM. To eliminate the space and performance overhead of address mapping, Mellanox ConnectX-4 and later versions of Mellanox RNICs introduce a new feature that allows user processes to register MRs directly using physical memory addresses [34]. Applications running on other machines can use these physical memory addresses to access the MRs.

Exposing physical memory addresses poses many new security threats, since knowing physical memory address layout is the basis of many attacks. For example, knowing physical memory addresses make both rowhammer [16, 24, 62] and throwhammer [52] attacks easier. Although registering MRs with physical address improves performance, we recommend applications builders to take caution when using this feature.

Discussion. While the three cases presented in this section all happen with Mellanox RNICs, we believe that similar issues can happen with other one-sided hardware and that there are deep-rooted reasons for why they happen. There are clear tradeoffs between including more security functionalities in hardware devices and the devices' performance and cost. For one-sided devices, vendors usually choose the latter over security, because what makes one-sided devices appealing is their superior performance and low cost (on saving CPU utilization). For example, the side channel threats in Case 2 are a result of vendors choosing performance (caching hot metadata in on-board SRAM) and cost saving (maximize the utilization of SRAM by not isolating SRAM space across applications) over security.

Mitigating security issues in one-sided hardware implementation is possible. For example, RNICs can use cryptographically generated keys as lkey and rkey (Case 1); they can isolate SRAM for different applications or introduce noise to disturb timing differences (Case 2); and they can remove the exposure of physical memory addresses (Case 3). Certain one-sided hardware vendors have manufactured SmartNICs that supports one-sided network communication [32, 33]. Various defense mechanisms can be implemented on these SmartNICs (*e.g.*, encryption). Despite the promise of the above mitigations, it is still challenging but important to deliver security features with minimal impacts on performance, cost, and hardware complexity.

5 Opportunity of One-Sided Communication

Although one-sided communication poses different threats to datacenter security, it provides one great opportunity to *enhance* security. Users can ensure their privacy by hiding their network activities from receiving servers using one-sided communication. This session discusses how we can leverage this opportunity to develop secure and fast data storage systems.

5.1 Environment and Threat Model

ORAM, or *oblivious* RAM, is a type of technology that makes access patterns "oblivious" by continuously re-encrypting and reshuffling data blocks at storage servers [13–15, 25, 40, 47, 50, 51, 58]. There have been various secure storage systems built based on ORAM in the past [5, 29, 44, 44, 49, 60]. One widely used ORAM system is Path ORAM [49]. The basic idea of Path ORAM is to organize the server storage as a binary tree where each node holds a number of (encrypted) objects (*e.g.*, key-value pairs). Each client caches a small amount of data locally in a *stash*. Client read and write requests are handled in the same Path ORAM protocol to make reads and writes indistinguishable. To perform a client request (read or write) to an object, the client identifies the path (nodes from a leaf to the root) that contains the object using a locally stored mapping table. The client then reads the entire path into its local stash (*read step*). Afterwards, it remaps the requested object to a random leaf node and writes all the objects in the path it has read in the read step (with a new value if it is a client write request). If there are other objects on the path in the client's stash, they can also be written back together.

We adopt the same trust model as previous ORAM systems [29, 44, 49, 60], where trusted clients access an untrusted storage service provider (*e.g.*, in-memory key-value store, NVMe-based storage). Clients either directly access the server machine that the service provider runs on through one-sided communication or access a *trusted proxy* which then accesses the service provider through one-sided communication.

We also adopt the same level of "security" and "privacy" as existing ORAM solutions [29, 44, 49, 51, 60], where the untrusted service provider should gain no information about client data or access patterns. Thus, hiding the content of data through encryption alone is not enough. In addition, no information should be leaked about: 1) which data is being accessed; 2) whether the access is a read or a write; 3) when the data was last accessed; 4) whether the same data is being accessed; or 5) the access pattern (sequential, random, etc).

5.2 One-Sided Oblivious RAM

Although proven to be cryptographically safe, existing ORAM-based systems have high performance overhead, making it too costly to be adopted in many datacenter environments. We now present our improved ORAM system design. Our system is based on Path ORAM [49] but can significantly outperform Path ORAM.

We propose to leverage one-sided communication to hide data access information and to replace (costly) ORAM operations with one-sided operations, thereby improving the performance of ORAM. Although the basic idea is simple, one needs to take extra care when applying it to provide the same level of security guarantees as ORAM. For example, a receiver (the malicious service provider) cannot detect when a one-sided write happens. But it can take snapshots of data content periodically and compare two snapshots to detect

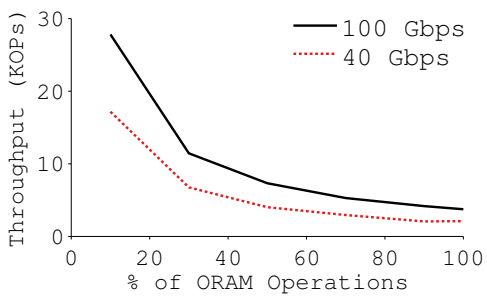

Figure 3: **Read Performance of One-Sided ORAM** *X axis shows the percentage of read operations that we use original ORAM operations to perform. The two lines show the performance when using a 100 Gbps and a 40 Gbps InfiniBand network.*

modifications in between. It can then infer that a write happens by performing frequent snapshotting. Thus, one-sided writes can still leak information to the service provider. Because of this, we use the unmodified ORAM write protocol and only improve ORAM read performance.

Reading data using one-sided communication can be completely invisible to the service provider, even if the provider performs snapshotting. For a read-only storage service, we can safely replace all reads with one-sided communication. However, in practice, most storage services are not read-only and we need to deliver the second guarantee in Section 5.1 — server should not be able to tell a read from a write.

We use a simple technique to achieve this security goal. By default, we perform one-sided reads for client read requests but randomly choose a certain amount of client read requests (*e.g.*, *X*% of all read requests) to perform original ORAM operations. These ORAM-based read operations cannot be distinguished from write operations. Therefore, statistically, we can deliver all the security guarantees. Meanwhile, performing a one-sided read is significantly faster than performing an ORAM operation, since the latter requires read and write of a whole path, while the former only performs a read to one object. Although the algorithmic complexity of the modified Path ORAM is still identical to the original Path ORAM, the performance of our improved ORAM read is significantly better than the original Path ORAM.

We implemented the original Path ORAM protocol and our modified read mechanism using RDMA and tested their performance with two network settings, a 40 Gbps InfiniBand network and a 100 Gbps InfiniBand network. Figure 3 presents our modified read performance when changing *X*% from 10% to 100% (under 100%, our system falls back to original Path ORAM). Here, we use a pure-read workload in the YCSB key-value store benchmark [7, 23], with 32,000 512-byte key-value pairs. We encrypt all data with AES256. With 50% one-sided reads, we achieve around 2× performance of pure Path ORAM.

5.3 Limitations and Discussion

One limitation of our one-sided ORAM solution is the requirement of prohibiting network packet sniffing. As described in Section 3, most systems enable this prohibition by default. However, if an attacker can bypass such prohibition mechanisms (*e.g.*, when it controls a physical machine), it can observe one-sided traffic. Thus, our threat model applies only to other cases, for example, when the attacker only owns a virtual machine.

One-sided NICs usually writes to memory through DMA. There are methods for a server to track (all) DMA activities (*e.g.*, with processor counter monitor [21]). However, it is still hard to pinpoint one-sided traffic, since other types of DMA activities may be happening at the same time. Even if a server can detect one-sided network traffic, it is still challenging to tell whom the sender is and which data is being accessed.

Our current design does not consider parallel client accesses to the secure data store, *i.e.*, we only support either a single proxy that delivers all client requests to the data store or a single client that talks to the data store directly. Supporting parallel client accesses to a secure data store is an important goal in many distributed data store systems [8,44,49,60]. Our design presented here serves as a building block in developing a fully distributed ORAM system.

6 Concluding Remarks and Future Work

This paper provides the first and initial look into the security aspect of one-sided network communication. We demonstrate several vulnerabilities that are rooted in one-sided communication's basic design philosophies. Although we have not built real attacks that exploit these vulnerabilities yet, our findings can be a starting point to explore potential attacks and defenses for future researchers and practitioners.

This work is a warning for future one-sided hardware vendors, software developers who want to use one-sided communication, and datacenters that have or plan to deploy one-sided network systems. We believe that these three parties should work together to achieve the security goals in datacenters while preserving one-sided communication's performance and cost benefits. One promising direction is to leverage programmable network devices like SmartNIC [32, 33] and programmable switches [22, 27, 28] to implement security defenses in hardware.

Although adding security guarantees to existing one-sided communication technologies is not an easy job, we do not believe the future of one-sided communication to be diminishing. For all the vulnerabilities that we have identified in this work, there are defense mechanisms that could potentially work. Moreover, one-sided communication provides a great opportunity to improve privacy, the security property that is increasingly important in today's cloud environments.

References

[1] M. K. Aguilera, N. Amit, I. Calciu, X. Deguillard, J. Gandhi, S. Novakovic, A. Ramanathan, P. Subrahmanyam, L. Suresh, K. Tati, R. Venkatasubramanian, and M. Wei. Remote regions: A simple abstraction for remote memory. In *Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference (ATC '18)*, Boston, MA, USA, July 2018.

[2] Alibaba Cloud. Super computing cluster. https://www.alibabacloud.com/product/scc, 2018.

[3] Apache. Crail: High-Performance Distributed Data Store. https://crail.incubator.apache.org/, 2018.

[4] M. S. Birrittella, M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, K. D. Underwood, and R. C. Zak. Intel omni-path architecture: Enabling scalable, high performance fabrics. In *Proceedings of the 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects (HOTI '15)*, Santa Clara, CA, USA, August 2015.

[5] E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi. Solidus: Confidential distributed ledger transactions via pvorm. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*, Dallas, Texas, USA, October 2017.

[6] Y. Chen, X. Wei, J. Shi, R. Chen, and H. Chen. Fast and general distributed transactions using rdma and htm. In *Proceedings of the Eleventh European Conference on Computer Systems (EUROSYS '16)*, London, UK, April 2016.

[7] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10)*, New York, New York, June 2010.

[8] N. Crooks, M. Burke, E. Cecchetti, S. Harel, R. Agarwal, and L. Alvisi. Obladi: Oblivious serializable transactions in the cloud. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, Carlsbad, CA, USA, October 2018.

[9] Davide Rossetti. GPUDIRECT: Integrating the GPU with a Network Interface. http://on-demand.gputechconf.com/gtc/2015/presentation/S5412-Davide-Rossetti.pdf, 2015.

[10] A. Dragojević, D. Narayanan, O. Hodson, and M. Castro. FaRM: Fast Remote Memory. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI '14)*, Seattle, WA, USA, April 2014.

[11] A. Dragojević, D. Narayanan, E. B. Nightingale, M. Renzelmann, A. Shamis, A. Badam, and M. Castro. No compromises: Distributed transactions with consistency, availability, and performance. In *Proceedings of the 25th Symposium on Operating Systems Principles (SOSP '15)*, Monterey, CA, USA, October 2015.

[12] Gen-Z Consortium. Gen-z overview, 2016. https://genzconsortium.org/wp-content/uploads/2018/05/Gen-Z-Overview-V1.pdf.

[13] O. Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC '87)*, New York, New York, USA, May 1987.

[14] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *Journal of ACM*, 43(3):431–473, May 1996.

[15] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia. Privacy-preserving group data access via stateless oblivious ram simulation. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms (SPDA '12)*, Kyoto, Japan, January 2012.

[16] D. Gruss, C. Maurice, and S. Mangard. Rowhammer.js: A remote software-induced fault attack in javascript. In *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA '16)*, San Sebastián, Spain, July 2016.

[17] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin. Efficient memory disaggregation with infiniswap. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (NSDI '17)*, Boston, MA, USA, March 2017.

[18] Z. Guz, H. H. Li, A. Shayesteh, and V. Balakrishnan. Nvme-over-fabrics performance characterization and the path to low-overhead flash disaggregation. In *Proceedings of the 10th ACM International Systems and Storage Conference (SYSTOR '17)*, Haifa, Israel, May 2017.

[19] Hewlett Packard. The Machine: A New Kind of Computer. http://www.hpl.hp.com/research/systems-research/themachine/.

[20] Hewlett Packard. The Machine: A New Kind of Computer. http://www.hpl.hp.com/research/systems-research/themachine/, 2005.

[21] Intel. Intel performance counter monitor. https://software.intel.com/en-us/articles/intel-performance-counter-monitor, 2012.

[22] X. Jin, X. Li, H. Zhang, R. Soulé, J. Lee, N. Foster, C. Kim, and I. Stoica. Netcache: Balancing key-value stores with fast in-network caching. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*, Shanghai, China, October 2017.

[23] Jinglei Ren. Ycsb-c. https://github.com/basicthinker/YCSB-C, 2015.

[24] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA '14)*, Minneapolis, Minnesota, USA, June 2014.

[25] E. Kushilevitz, S. Lu, and R. Ostrovsky. On the (in)security of hash-based oblivious ram and a new balancing scheme. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '12)*, Kyoto, Japan, January 2012.

[26] B. Li, Z. Ruan, W. Xiao, Y. Lu, Y. Xiong, A. Putnam, E. Chen, and L. Zhang. Kv-direct: High-performance in-memory key-value store with programmable nic. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*, Shanghai, China, October 2017.

[27] J. Liang, J. Bi, Y. Zhou, and C. Zhang. In-band network function telemetry. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos (SIGCOMM '18)*, Budapest, Hungary, August 2018.

[28] M. Liu, L. Luo, J. Nelson, L. Ceze, A. Krishnamurthy, and K. Atreya. Incbricks: Toward in-network computation with an in-network cache. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17)*, Xi'an, China, April 2017.

[29] J. R. Lorch, B. Parno, J. Mickens, M. Raykova, and J. Schiffman. Shroud: Ensuring private access to large-scale data in the data center. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST '13)*, San Jose, CA, USA, February 2013.

[30] Y. Lu, J. Shu, Y. Chen, and T. Li. Octopus: an rdma-enabled distributed persistent memory file system. In *2017 USENIX Annual Technical Conference (ATC '17)*, Santa Clara, CA, USA, July 2017.

[31] J. Markussen, L. B. Kristiansen, H. K. Stensland, F. Seifert, C. Griwodz, and P. Halvorsen. Flexible device sharing in pcie clusters using device lending. In *Proceedings of the 47th International Conference on Parallel Processing Companion (ICPP '18)*, Eugene, OR, USA, August 2018.

[32] Mellanox. Bluefield smartnic. http://www.mellanox.com/related-docs/prod_adapter_cards/PB_BlueField_Smart_NIC.pdf, 2018.

[33] Mellanox. Innova-2 flex open programmable smartnic. http://www.mellanox.com/related-docs/prod_adapter_cards/PB_Innova-2_Flex.pdf, 2018.

[34] Mellanox. Physical address memory region. https://community.mellanox.com/s/article/physical-address-memory-region, 2018.

[35] Mellanox. RDMA/RoCE Solutions. https://community.mellanox.com/s/article/rdma-roce-solutions, 2018.

[36] Mellanox. Connectx-6 en card. http://www.mellanox.com/related-docs/prod_adapter_cards/PB_ConnectX-6_EN_Card.pdf, 2019.

[37] D. Minturn. NVM Express Over Fabrics. 11th Annual OpenFabrics International OFS Developers' Workshop, March 2015.

[38] C. Mitchell, Y. Geng, and J. Li. Using one-sided rdma reads to build a fast, cpu-efficient key-value store. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference (ATC '13)*, San Jose, CA, USA, June 2013.

[39] C. Mitchell, K. Montgomery, L. Nelson, S. Sen, and J. Li. Balancing cpu and network in the cell distributed b-tree store. In *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference (ATC '16)*, Denver, CO, USA, June 2016.

[40] B. Pinkas and T. Reinman. Oblivious ram revisited. In *Proceedings of the 30th Annual Conference on Advances in Cryptology (CRYPTO '10)*, Santa Barbara, CA, USA, August 2010.

[41] M. Poke and T. Hoefler. Dare: High-performance state machine replication on rdma networks. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '15)*, Portland, OR, USA, June 2015.

[42] R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang. Enabling security in cloud storage slas with cloudproof. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference (ATC '11)*, Portland, OR, USA, June 2011.

[43] A. Romanow, J. Mogul, T. Talpey, and S. Bailey. Remote Direct Memory Access (RDMA) over IP Problem Statement. https://tools.ietf.org/html/rfc4297, 2005.

[44] C. Sahin, V. Zakhary, A. El Abbadi, H. Lin, and S. Tessaro. Taostore: Overcoming asynchronicity in oblivious data storage. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP '16)*, San Jose, CA, USA, May 2016.

[45] Y. Shan, Y. Huang, Y. Chen, and Y. Zhang. Legoos: A disseminated, distributed OS for hardware resource disaggregation. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, Carlsbad, CA, October 2018.

[46] Y. Shan, S.-Y. Tsai, and Y. Zhang. Distributed shared persistent memory. In *Proceedings of the 8th Annual Symposium on Cloud Computing (SOCC '17)*, Santa

Clara, CA, USA, September 2017.

[47] E. Shi, T.-H. H. Chan, E. Stefanov, and M. Li. Oblivious ram with o((logn)3) worst-case cost. In *Proceedings of the 17th International Conference on The Theory and Application of Cryptology and Information Security (ASIACRYPT '11)*, Seoul, South Korea, December 2011.

[48] J. Shi, Y. Yao, R. Chen, H. Chen, and F. Li. Fast and concurrent rdf queries with rdma-based distributed graph exploration. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI '16)*, Savannah, GA, USA, November 2016.

[49] E. Stefanov and E. Shi. Oblivistore: High performance oblivious cloud storage. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP '13)*, San Francisco, CA, USA, May 2013.

[50] E. Stefanov, E. Shi, and D. Song. Towards practical oblivious RAM. In *Proceedings of the Network and Distributed System Security Symposium (NDSS '12)*, San Diego, CA, USA, February 2012.

[51] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path oram: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13)*, Berlin, Germany, November 2013.

[52] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi. Throwhammer: Rowhammer attacks over the network and defenses. In *Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference (ATC '18)*, Boston, MA, USA, July 2018.

[53] M. Technologies. Mellanox OFED for Linux User Manual, 2019. http://www.mellanox.com/related-docs/prod_software/Mellanox_OFED_Linux_User_Manual_v4_5.pdf.

[54] Tejas Karmarkar. Availability of linux rdma on microsoft azure. https://azure.microsoft.com/en-us/blog/azure-linux-rdma-hpc-available, 2015.

[55] The Tcpdump Group. tcpdump - Dump Traffic on A Network. https://www.tcpdump.org/manpages/tcpdump.1.html, 2018.

[56] S.-Y. Tsai and Y. Zhang. LITE Kernel RDMA Support for Datacenter Applications. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*, Shanghai, China, October 2017.

[57] C. Wang, J. Jiang, X. Chen, N. Yi, and H. Cui. Apus: Fast and scalable paxos on rdma. In *Proceedings of the 2017 Symposium on Cloud Computing (SoCC '17)*, Santa Clara, CA, USA, September 2017.

[58] X. S. Wang, Y. Huang, T.-H. H. Chan, A. Shelat, and E. Shi. Scoram: Oblivious ram for secure computation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*, Scottsdale, AZ, USA, November 2014.

[59] X. Wei, Z. Dong, R. Chen, and H. Chen. Deconstructing rdma-enabled distributed transactions: Hybrid is better! In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18)*, Carlsbad, CA, USA, October 2018.

[60] P. Williams, R. Sion, and A. Tomescu. Privatefs: A parallel oblivious file system. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*, Raleigh, North Carolina, USA, October 2012.

[61] M. Wu, F. Yang, J. Xue, W. Xiao, Y. Miao, L. Wei, H. Lin, Y. Dai, and L. Zhou. Gram: Scaling graph computation to the trillions. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC '15)*, Kohala Coast, HI, USA, August 2015.

[62] Y. Xiao, X. Zhang, Y. Zhang, and R. 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, August 2016.

[63] E. Zamanian, C. Binnig, T. Harris, and T. Kraska. The End of a Myth: Distributed Transactions Can Scale. *Proceedings of the VLDB Endowment*, 10(6):685–696, 2017.

[64] Y. Zhang, J. Yang, A. Memaripour, and S. Swanson. Mojim: A Reliable and Highly-Available Non-Volatile Memory System. In *Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '15)*, Istanbul, Turkey, March 2015.