Preprint version, accepted to the 10th ACM International Systems and Storage Conference (SYSTOR'17), May 2017

NVMe-over-Fabrics Performance Characterization and the Path to Low-Overhead Flash Disaggregation

Zvika Guz, Harry (Huan) Li, Anahita Shayesteh, and Vijay Balakrishnan
Memory Solution Lab
Samsung Semiconductor, Inc.
{zvika.guz, harry.li, anahita.sh, vijay.bala}@samsung.com

ABSTRACT

Storage disaggregation separates compute and storage to different nodes in order to allow for independent resource scaling and thus, better hardware resource utilization. While disaggregation of hard-drives storage is a common practice, NVMe-SSD (i.e., PCIe-based SSD) disaggregation is considered more challenging. This is because SSDs are significantly faster than hard drives, so the latency overheads (due to both network and CPU processing) as well as the extra compute cycles needed for the offloading stack become much more pronounced.

In this work we characterize the overheads of NVMe-SSD disaggregation. We show that NVMe-over-Fabrics (NVMf) – a recently-released remote storage protocol specification – reduces the overheads of remote access to a bare minimum, thus greatly increasing the cost-efficiency of Flash disaggregation. Specifically, while recent work showed that SSD storage disaggregation via iSCSI degrades application-level throughput by 20%, we report on negligible performance degradation with NVMf – both when using stress-tests as well as with a more-realistic KV-store workload.

CCS Concepts

•**Hardware** → **External storage;** •**Information systems** → *Flash memory; Storage management;*

Keywords

NVMe-over-Fabrics, performance characterization, network storage

1. INTRODUCTION

NVMe-SDDs (i.e., PCIe-based SSDs) are becoming increasingly popular for serving I/O intensive applications in data centers and enterprise deployments. This is because their vastly superior performance compared to SATA-SSDs and SAS-SSDs (in terms of both bandwidth and latency) caters well to the ever-increasing throughput demands of I/O intensive applications; primarily relational and NoSQL databases. A single current-generation NVMe-SSD sustains up to one million IOPS with a 90μs read latency [1] – more than 5 times better than current generation SAS-SSDs, and 40 times better than SATA-SSDs. Indeed, many large-scale cloud companies have reported using PCIe-based SSDs as part of their infrastructure [2–5].

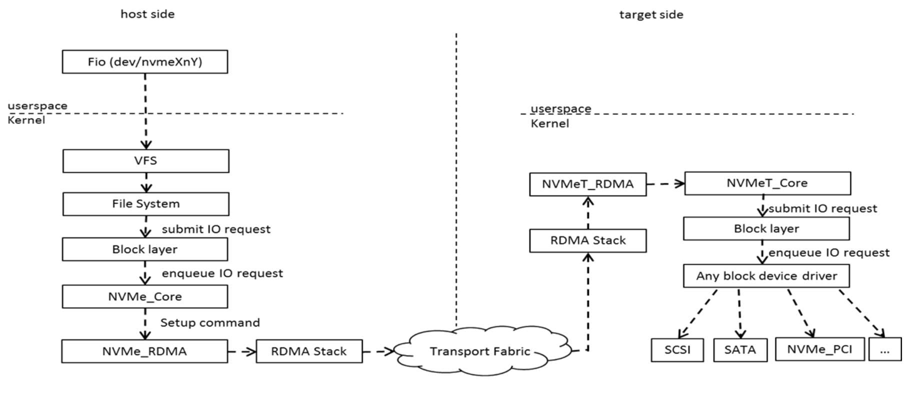
Generally, storage devices can either be co-located within the compute server nodes, or be placed in dedicated storage nodes accessed through the network. Large-scale cloud companies originally used scale-out infrastructure, composing their data centers out of commodity servers that tightly coupled memory, storage, and compute [6,7]. Unfortunately, this approach leads to inefficiencies and resource underutilization, because it fixes the ratio between compute, memory, storage, and network.

Resource underutilization in data centers is a common, well-documented phenomenon [2, 5, 6, 8]. Since the usage of different resources changes over time, predominantly independently from one another, there is no single static resource balance that fits every application that a server supports throughout its lifetime. Since changing these ratios at scale is economically unfeasible [9], server resources are often over-provisioned, leading to an increased total cost of ownership [6]. Specifically, NVMe-SSDs tend to be over-provisioned in terms of both load (IOPS) and capacity: capacity is over-provisioned to allow for future growth, and load is underutilized because other software overheads tend to saturate the CPU well before reaching the drive limits [5, 10]. Indeed, PCIe-based Flash has been argued to be "problematically fast" [11].

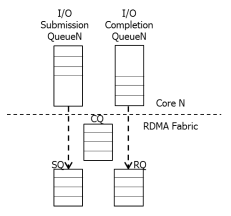
Resource disaggregation is a common approach to mitigate the problem of over-provisioning. Specifically, storage disaggregation decouples compute and storage to different nodes (i.e. different servers), allowing independent scaling of each resource according to dynamic needs. It provides more flexibility when tuning the infrastructure to specific loads because compute and storage can be configured for concrete demands rather than for average or maximum expected loads. In turn, it can greatly increase data center cost efficiency by reducing resource waste.

Despite its potential merits, storage disaggregation comes at a cost. Since storage devices are accessed over a network, interconnect latencies are added to every storage access, adversely affecting access times. Furthermore, the additional network processing and the overheads of the remote storage protocol itself tax the CPU on both the storage server and the compute node, further degrading application

SYSTOR'17 May 22-24, 2017, Haifa, Israel
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5035-8/17/05.
DOI: http://dx.doi.org/10.1145/3078468.3078483



(a) **NVMe-over-Fabrics Stack**



(b) **NVMe-RDMA Queue pairing per core**

Figure 1: NVMe-over-Fabrics Architecture

throughput compared to a direct-attached configuration. Hard-drive (HDD) disaggregation is common because its low performance (hundreds of IOPS and milliseconds latencies) makes network and processing overheads negligible [9, 12–16]. NVMe-SSD disaggregation, on the other hand, is considered significantly more challenging: since these drives are orders-of-magnitude faster than HDDs, the network latencies and protocol overheads become much more pronounced. A recent study on NVMe-SSD disaggregation via iSCSI demonstrated the cost benefits of Flash disaggregation, but also reported an average degradation of 20% in application-level performance [2].

NVMe-over-Fabrics (NVMf) [17] is a recent protocol standard for accessing NVMe devices over RDMA-capable networks (see Section 2). By leveraging RDMA, NVMf offloads data movement to the network card (NIC), thus reducing the processing overheads involved in handling remote I/O requests on both the host and the target. Since processing overheads are a major cause for performance degradation when using disaggregated NVMe-SSD (see Section 4), NVMf is expected to be a better fit as the underlining remote storage protocol for disaggregated storage. However, being a relatively new standard, there is little available performance characterization data of NVMf and its overheads.

This paper revisits NVMe-SSD disaggregation, using NVMf as the remote storage protocol. We show that NVMf minimizes the performance degradation experienced by the application, thus significantly improving the efficiency and viability of NVMe disaggregation. We also present a detailed NVMf performance analysis. First, in Section 4, we use synthetic stress-tests to characterize and break down NVMf latency and throughput overheads, and compare them to those of iSCSI. Then, in Section 5, we use RocksDB as an example for an I/O intensive KV-store, and show that it experiences minimal performance degradation when running on remote storage. Finally, we show that NVMf's low processing overhead enables better scaling at the storage server side compared to iSCSI, further improving cost efficiency.

In summary, this paper argues that NVMf facilitates NVMe-SSD disaggregation: it preserves all the advantages discussed in previous literature, but further minimizes performance impacts and thus significantly improves cost efficiency.

2. NVME-OVER-FABRICS

Non-Volatile Memory Express (NVMe) is an optimized, high-performance, software interface standard for accessing local non-volatile memory devices over PCIe [18]. NVMe is based on a high number of deep, paired Submission and Completion Queues, allocated in host memory. These paired, parallel queues are the interface between the NVMe driver and the NVMe controller, which manage them cooperatively. While other proprietary protocols exist [19,20], NVMe is the predominate, and most widespread, technology for PCIe-based SSD devices. In this paper we use the terms *PCIe-based SSD* and *NVMe-SSD* interchangeably.

NVMe-over-Fabrics (NVMf) [17] is a recent extension to the NVMe standard that enables access to remote NVMe devices over different network fabrics. It eliminates unnecessary protocol translations (such as SCSI) along the I/O path to the remote device, exposing the multiple paired-queue design of NVMe. The goal is to provide applications with fast storage access regardless of whether NVMe-SSDs are attached locally or accessed remotely; NVMf aims to keep the overheads of remote access to less than $10\mu s$. NVMf currently supports two types of fabric transport: RDMA and Fiber Channel. Of the two, RDMA enjoys wider availability both in traditional HPC domain as well as in modern data centers [21–27].

Remote Direct Memory Access (RDMA) is a remote memory management capability that allows direct data movement from one machine's memory to another machine's memory over the network without CPU involvement [28]. RDMA enables low latency, high-throughput remote access by (1) Using zero-copy networking: data is transferred directly to or from application memory, eliminating the need to copy data between network layers as done in TCP/IP, (2) Using kernel bypass: applications perform data transfers directly from user-space without kernel involvement; and (3) Eliminating CPU involvement: RDMA transfers do not consume CPU cycles or pollute the CPU caches.

RDMA can be implemented over different link layer protocols. It was originally associated with InfiniBand [29], requiring special-purpose interconnect. Two recent Ethernet-base alternatives – RoCE and iWARP – provide a more cost-efficient solution that is expected to increase RDMA adoption in the data center domain. RoCE (RDMA over Converged Ethernet) [30] is based on InfiniBand transport

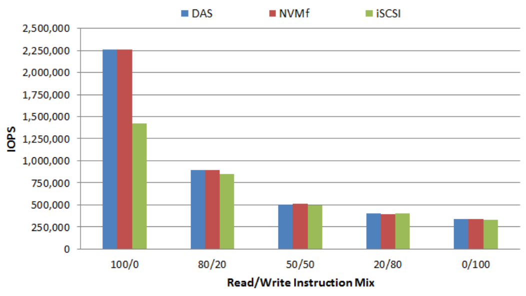


Figure 2: Maximum achievable throughput (in IOPS) for DAS, NVMf, and iSCSI.

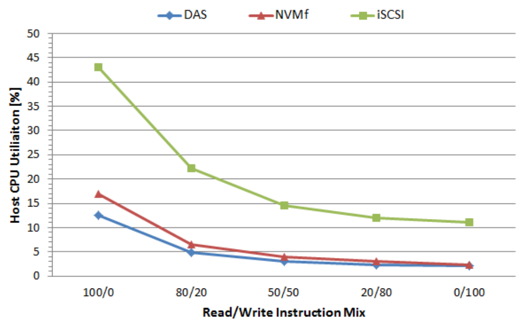


Figure 3: Host CPU utilization for DAS, NVMf, and iSCSI.

over Ethernet, and uses RDMA over Ethernet and UDP/IP frames. iWARP (interent Wide Area RDMA Protocol) [31] uses RDMA over a connection-oriented transport such as TCP. Both iWARP and RoCE require specialized RDMA-capable NICs, but can both operate over modern Ethernet-based switches. This greatly reduces the infrastructure investment compared to InfiniBand[1]. While running iWARP over standard Ethernet is cheaper than running RoCE, RoCE provides lower latencies and has a strong following. Lately, RoCE seems to be gaining momentum and is being adopted by OEMs (e.g., [32]); RoCEv2 is the fabric used in our NVMf evaluation.

Figure 1a shows the stack of NVMe-over-Fabrics for the host and target. The Host driver contains common NVMe host core, and PCIe and RDMA transports. The Target driver contains NVMe target core, and RDMA transport. Both drivers share common NVMe and NVMf data structures, queuing interfaces, commands, and properties, which are all independent of the NVMe Transport layer. NVMe Transport binding (NVMe-RDMA block), binds NVMe-over-Fabrics to NVMe Transport layer and is specific to the Fabric Protocol and physical layer.

Before a transfer is made, the host makes an association (*connection*) to the NVM controller in the target, creating a one-to-one mapping between I/O Submission Queues and I/O Completion Queues. NVMe queues are aligned to CPU cores, and paired with RDMA dedicated Send (SQ) and Response (RQ) Queue Pairs and Completion Queues (See Figure 1b). This queue design with end-to-end pairing exposes the inherent parallelism of NVMe to the remote host. While the association exists, NVMe Host Driver encapsulates the NVMe commands and responses into a fabric-neutral Command Capsule and passes it to the NVMe RDMA Transport. A capsule is the NVMe unit transferred from the host to the NVM subsystem [17].

3. METHODOLOGY

This section describes the hardware and software setup used in our study. Our non-disaggregated baseline configuration uses a dual-socket Xeon E5-2699 v4 server [33] with 128GB DDR4 memory. It is fitted with a single PM1725 NVMe-SSDs [1] – a high-end enterprise drive that supports up to 750 (120) KIOPS of random read (write) traffic, and 3GB/sec (2GB/sec) of sequential read (write) bandwidth. We refer to this configuration as Direct-Attached-Storage (DAS), or *local* server. When evaluating remote storage, we use three such DAS servers and move their drives to a designated *storage server* (referred to as *target*). The three baseline nodes now access the drives over the network, and are referred to as *hosts* (or *datastore servers* [2]). The target node is a quad-socket Xeon E7-8890 v4 server [33], populated with 512GB DDR4 memory and the same three NVMe-SSD drives. To reduce noise and allow for a cleaner comparison, we disable hyperthreading as well as power-saving states on all servers.

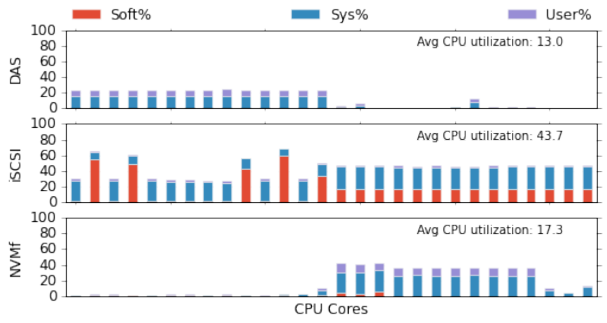
All servers use a single ConnectX-4 100Gb Ethernet NIC [34] that provides RDMA over Ethernet (RoCE) capabilities, and connect through a 100Gb Ethernet switch [35]. For the remote storage configuration, we evaluate both iSCSI and NVMf over RoCE. Since data centers mainly use Ethernet, we use RoCE rather than InfiniBand for the underlining interconnect (as described in Section 2). We made an explicit effort to tune the iSCSI setup, following known best practices (e.g., [2]).

We use two types of workloads in our performance analysis. First, for stress-testing, we use *fio* [36] to generate synthetic I/O traffic. We use 4K random traffic to create the highest load on the CPUs and on the remote storage stacks. We also bypass the operating system page cache by using fio's direct I/O mode. Then, to simulate a more-realistic data center workload, we use RocksDB [37] as an example of a KV store library that is widely used in many production stacks [38]. We use RocksDB's own benchmark utility – db_bench – to generate loads and to measure performance. We model 16B keys with 800B and 10KB values, and use a 80/20 read-write mix. While some RocksDB installations might be used with larger objects [2], we chose smaller objects to further stress our system. We also limit the OS page cache so that more requests would be served from the drives rather than from DRAM.

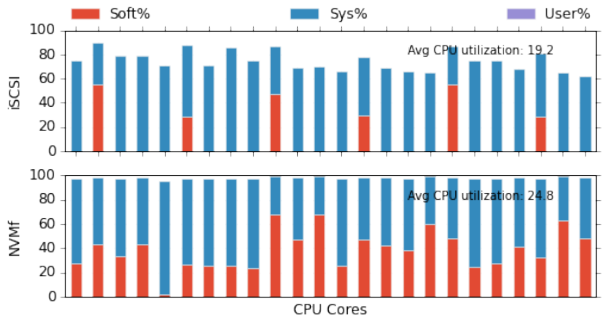
4. NVMF PERFORMANCE ANALYSIS

In this section we use synthetic I/O traffic (generated with *fio*) to study the overheads of NVMf relative to a direct-attached storage. We also characterize remote storage access via iSCSI in order to quantify the performance savings realized through using NVMf and RDMA. In Section 4.1 we examine performance at maximum load, and in Section 4.2 we characterize latencies and overheads at different system operation points.

[1]Unlike iWARP, RoCE requires that switches support Data Center Bridging (DCB)). While old switches do not support DCB, it is becoming common in new high-bandwidth switches.



(a) **Host**



(b) **Target**

Figure 4: Average per-core CPU utilization for both (a) the host and (b) the target during fio read-only test. We consider DAS as a host and show its data in (a). For the sake of presentation clarity, we only show the active cores on the target node.

4.1 Maximum Load Characterization

Figure 2 shows the maximum IOPS attained with different read-to-write ratios for DAS, iSCSI, and NVMf. DAS saturates the three drives in all cases and serves as the upper limit for performance. As can be seen in the figure, NVMf performance (red bars) is practically the same as DAS – the difference is less than 1% for all cases. The iSCSI throughput (green bars), on the other hand, is significantly lower than DAS and NVMf: in the read-only test, iSCSI delivers 1.4 MIOPS, utilizing only 60% of the drives' available IOPS. iSCSI performance is hindered by an inefficient network stack and protocol processing overheads such as unnecessary protocol translation from NVMe to SCSI and back. In addition, and unlike NVMf, iSCSI cannot expose NVMe's inherent parallelism and multiple queues, leaving significant performance on the table.

As more writes are added to the instruction mix, overall performance in all configurations decreases due to the drive's asymmetric read/write performance (One PM1725 drive can sustain 750 and 120 KIOPS for read and write traffic, respectively). When total IOPS, and thus the overall system load drop, iSCSI is able to support the reduced IOPS, achieving performance close to DAS and NVMf. Note that in addition to inferior performance, iSCSI adds significant CPU overhead as shown in Figure 3. While NVMf adds at most 4.3% to the overall host CPU utilization, iSCSI overheads can add a whopping 30%. For real-world workloads that require more compute resources than fio, these overheads will further degrade the application performance as they significantly reduce the overall available system resources.

Figure 4 provides a more in-depth look at CPU utilization for the read-only test, showing per-core utilization on both the host and the target. User time in Figure 4a indicates time spent running the fio application, where Sys time accounts for network and storage protocols, and for Kernel mode processing. In addition to system and user, the host in our iSCSI configuration spends significant time servicing software interrupts (soft). This is a known issue related to TCP/IP significant interrupt handling overheads; it can benefit from distributing interrupts via affinity setting [2]. In our experiments we did not see noticeable improvements by setting interrupt affinity, probably due to the high core count of our machines. For the NVMf host, overall CPU utilization is slightly increased from 13% in DAS to 17% in NVMf, and processing is limited to one socket. On the other hand, in the iSCSI setup, cores on both sockets in the host have a 43% average utilization.

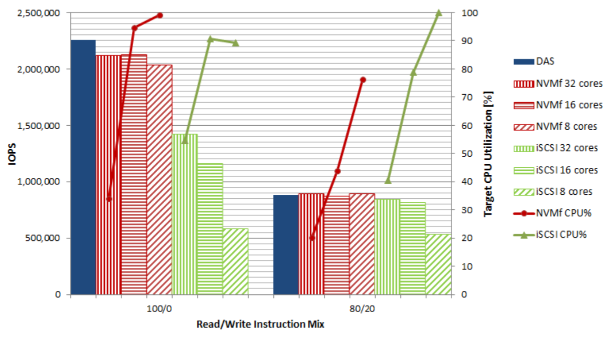
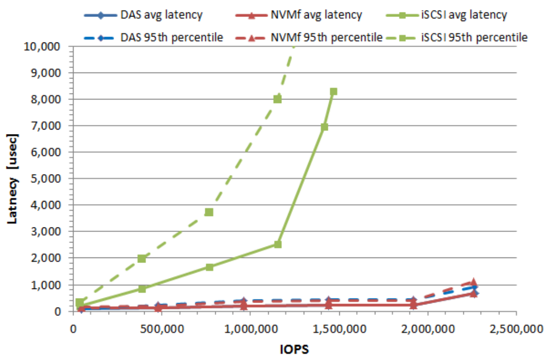


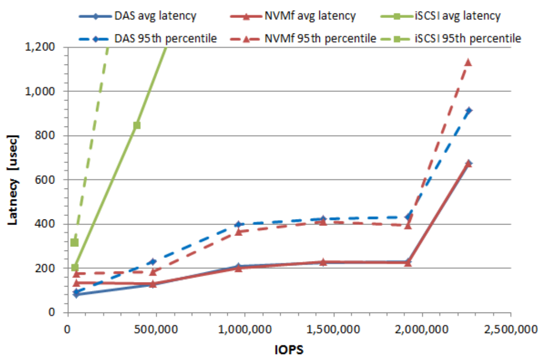
Figure 5: Overall IOPS and cores utilization for different core counts on the target.

On the target side, Figure 4b shows the utilization of the active cores in the system. Out of the 96 cores, only 24 cores are utilized, i.e., only one out of the four available sockets. The average utilization numbers reported in the figure account for all cores, which explains the relatively low numbers. Both NVMf and iSCSI have high utilization on active cores. However, note that iSCSI's overall IOPS performance is 37% lower than NVMf performance for the read-only test (See Figure 2).

Lastly, to further study the target overheads and compute requirements, we repeated the cpu-intensive tests (read-only and 80/20 read/write) while enabling only a subset of the cores available on the the target. As can be seen in Figure 5, since NVMf requires less CPU resources per IO, its performance degrades gracefully as the core count decreases. Even with only 1/12th of the cores enabled (red diagonal stripes) NVMf performance is still within 90% of local-storage for the read-only test. iSCSI performance degrades much more quickly as the number of available cores decreases. For the read-only case, performance falls by more than 2x, to 26% of the max IOPS in the same scenario (green diagonal stripes). NVMf's low overheads on the target node is particularly important for datacenter cost efficiency, because it allows a storage servers to be populated with more drives and fewer



(a) Full view



(b) Zoom-in on 0-1ms range

Figure 6: Average and tail latencies of DAS, NVMf, and iSCSI for different request loads. Since iSCSI saturates early and exhibits significantly higher latencies, we show both the full range view (a), and a zoom-in on DAS and NVMf (b).

processors (or lower-performance processors), thus reducing overall TCO without sacrificing performance.

4.2 Latency and Throughput Analysis

Figure 6 shows the latencies of DAS, NVMf, and iSCSI for different request loads. We use a 4KB, read-only request stream, and show both average as well as tail (95th percentile) latencies. As can be seen in the figure, NVMf performance is almost identical to DAS over the entire load range – the average latencies are within a 3.5% difference except for the first (2% load) run. In fact, NVMf tail latencies are slightly better than DAS for light to medium-high loads. The same phenomenon of slightly higher performance of NVMf compared to DAS was also reported by Mintrum [39], and attributed to interrupt coalescing in the RDMA NIC. Indeed, under all practical loads scenarios, the performance of a locally attached NVMe-SSD and the performance of a remote NVMf access are virtually the same.

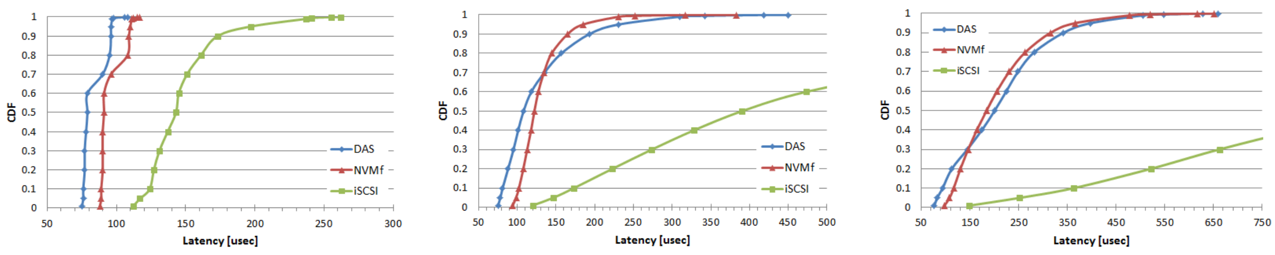
Unlike NVMf, iSCSI latencies are noticeably longer compared to local storage access. First, iSCSI saturates well before DAS (and NVMf) – latencies start climbing exponentially after about 1.5 MIOPS compared to 2 MIOPS for DAS. Second, even for relatively light loads, iSCSI adds significant delays compared to the time needed to access local storage. When operating at 50% of drive capacity, the average latency is 2.5msec – more than 10x the local access latency. The difference in tail latency is even larger: to keep the 95th percentile latency below 2 msec, we can only operate at 17% of the maximum drive IOPS capacity. Conversely, the 95th percentile latency for DAS and NVMf is kept below 2 msec throughout the entire load range. Our results indicate that, unlike iSCSI, NVMf does not introduce additional latency to remote access, and can be deployed in even the most latency-sensitive applications without performance penalty.

To provide further comparison of NVMf and DAS latencies, Figure 7 shows the cumulative distribution (CDF) of the end-to-end read latency under three different loads. The unloaded latencies (Figure 7a; 1 job, 1 in-flight request) for all three configurations are relatively close: the average latencies are 83.5μs, 95.2μs, and 145.8μs for DAS, NVMf, and iSCSI respectively. NVMf adds 11.7μs delay – very close to the specification's target of 10μs. iSCSI adds 62μs. While larger, this is still acceptable for most data-center use cases. Similarly, tail latencies are also close: 95th percentile latencies are 96μs, 110μs, and 197μs for DAS, NVMf and iSCSI respectively (i.e., 14μs overhead for NVMf and 101μs overhead for iSCSI). However, as we increase the load on the system (Figure 7b and Figure 7c), iSCSI's latencies become much higher than DAS while NVMf overheads (both average and tail) stay minimal. Since we wanted to focus on NVMf in these figures, we let iSCSI go out of the figure range.

Lastly, Figure 8 shows a breakdown of unloaded NVMf read latency. We use ftrace [40] to trace the time spent in each module, and use a DAS read test to derive the baseline time needed for local access. Overall, NVMf is 11.7μs slower than a local NVMe access. The NVMf-related modules on the host account for 27% of the overhead (block layer, NVMe driver, and the RDMA stack, see Figure 1a); NVMf target modules (RDMA stack and NVMe_T modules in Figure 1a) account for an additional 38%. The network and switch add another 2.43μs to the overall latency. Finally, an extra 13% is spent outside of any NVMf module. While we do not have further breakdown of this latency, it includes interrupt delivery latency, kernel scheduling delays, and overhead related to switching between user and kernel spaces. Intel's SPDK [41] – the user-mode NVMf driver – is expected to reduce these overheads, because SPDK runs in user space (thus eliminating user-to-kernel transitions) and uses polling rather than interrupts (thus eliminating interrupt handling latencies) [42,43]. Indeed, our preliminary experiments running SPDK for DAS and for the NVMf target node, resulted in a reduction of 2.84μs in both configuration. In other words, using NVMf SPDK Target [41] reduces NVMf additional latency for unloaded read to 8.9μs compared to DAS, and preserves the 11.7μs overhead compared to local access with SPDK.

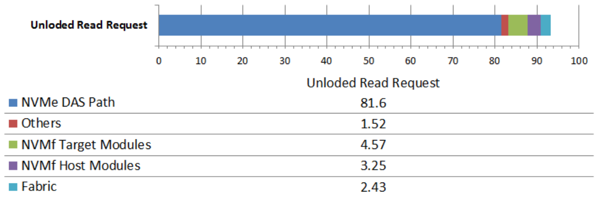
5. DISAGGREGATED KV-STORE

While in Section 4 we use microbenchmark stress-tests to characterize NVMf performance, real-world data center workloads can have very different system characteristics and are generally less demanding of a node's I/O system. Often, attached-drives are only moderately utilized [2,5,44] because other system bottlenecks saturate well before the drives. Moreover, compute-per-disk I/O is higher relative to our



(a) Unloaded system (b) Load=20% (480 KIOPS) (c) Load=42% (960 KIOPS)

Figure 7: Read latency CDF under different loads for DAS, NVMf, and iSCSI.



🟦 NVMe DAS Path	81.6
🟥 Others	1.52
🟩 NVMf Target Modules	4.57
🟪 NVMf Host Modules	3.25
🟦 Fabric	2.43

Figure 8: A breakdown of NVMf end-to-end latency for an unloaded read request.

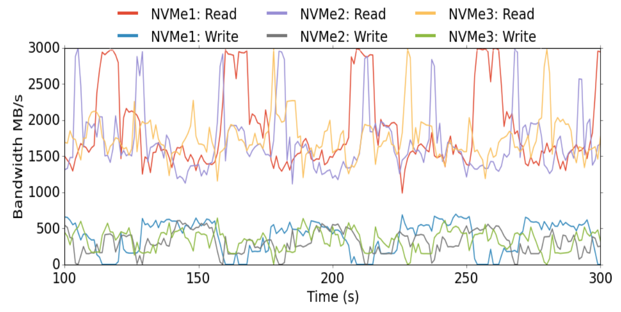


Figure 9: Transient NVMe-SSD bandwidth on the NVMf target node during 200 seconds of RocksDB NVMf run. All three drives gets saturated (3GB/s read) during the bandwidth spikes.

fio test, because (1) fio is inherently less compute-intensive than a full-fledged application, (2) DRAM caching filters out some of the I/Os, an effect not modeled by our direct fio test; and (3) various software overheads associated with data center services (coined "datacenter tax" by Kanev et. al [45]) further increase the processor load. In this section we use RocksDB – a popular KV-store used in many data center production stacks [38] – to characterize a more-realistic Flash disaggregation scenario and to quantify its expected benefits.

Figure 10 shows RocksDB performance (in terms of operations-per-second) using local storage vs. remote storage via both iSCSI and NVMf (See Section 3 for RocksDB and db_bench configuration details). We use 10KB objects as our baseline, similar to Klimovic et. al [2], to represent typical RocksDB production loads. In addition, we also show performance with smaller 800B objects. With smaller objects, more transactions are executed per second, further stressing the remote storage setup. As can be seen in the figure, while iSCSI performance is significantly lower than local storage (40% performance degradation), NVMf performance is within a negligible 2% of DAS. Indeed, in terms of overall performance, the host does not see much difference when the drives are accessed locally vs. remotely over NVMf. In terms of processing overheads (see Figure 11), NVMf does increases the host CPU utilization by 10%. While this modest overhead should be acceptable in most scenarios, the SPDK user-mode NVMf driver is expected to further reduce NVMf overheads on both the client and the host. The host CPU utilization for iSCSI is given for completeness, but comparing it to NVMf is difficult because the overall performance is significantly lower.

Figure 9 shows the fluctuations in NVMe-SSD bandwidth on the NVMf target node during the workload run. These bandwidth fluctuations are a property of RocksDB background processes and are an expected RocksDB behavior. As can be seen in the figure, the NVMf target machine is able to saturate all three NVMe drives: all drives reach 3GB/s (the drives peak read bandwidth) during periods of bandwidth spikes. Both DAS and NVMf setups saturate the NVMe drives. However, note that in the DAS configuration each of the three hosts drives a single NVMe SSD, while for the NVMf configuration the target machine handles all three drives with peak SSD bandwidth of 7.8GB/s.

Lastly, Figure 12 plots the cumulative read latency distribution reported by db_bench for NVMf and DAS. For Figure 12 we use 800B objects, and run at a reduced load (70%) so that we do not operate close to saturation point. We omit iSCSI from this figure. Since we could not reach the same throughput when using iSCSI, any latency comparison would have been meaningless. NVMf read latencies under load are comparable to local storage: NVMf average latency is 11% (60μs) higher than DAS, and the 99th percentile latency overhead is a negligible 2% (83μs).

6. RELATED WORK

Disaggregation is a well known approach to achieve computing hardware modularity and to address imbalanced resource requirements that results in overprovisioning [46]. HDD disaggregation [9, 12–16] is common because network overheads are insignificant compared to the latency of accessing data on a spinning disk. This paper focuses on PCIe-based SSD disaggregation, which poses significant challenges due to the raw throughput and latency of NVMe-SSD. Klimovic et al. [2] evaluate Flash disaggregation with iSCSI. They show that even though iSCSI remote access introduces a 20% throughput drop at the application level, disag-

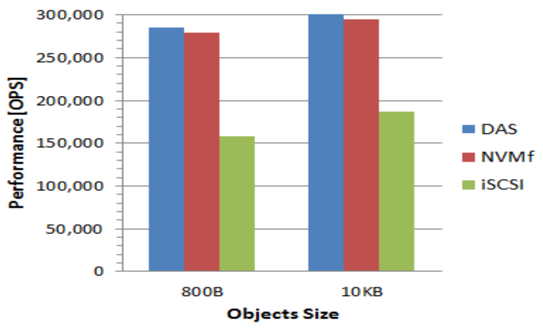


Figure 10: RocksDB maximum throughput for DAS, NVMf, and iSCSI. We show results for two object sizes: 800B and 10KB.

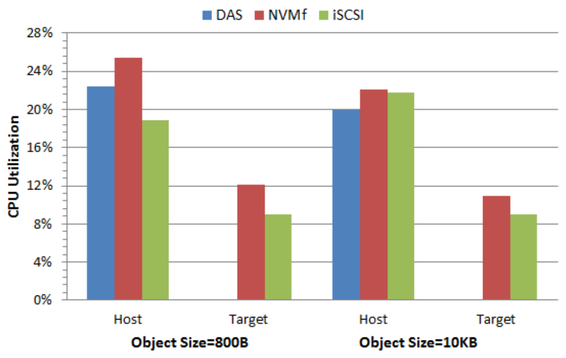


Figure 11: Average CPU utilization on the host and target for RocksDB workloads.

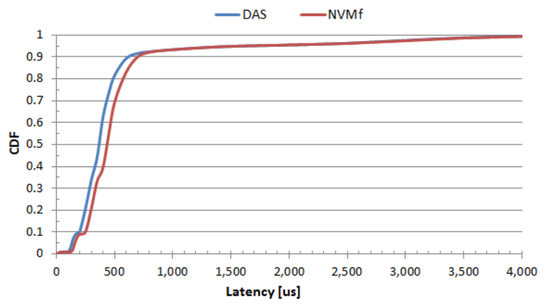


Figure 12: RocksDB read latency CDF for DAS and NVMf, with 800B objects.

gregation makes up for that overhead by independently scaling CPU and Flash resources. We extend their work by evaluating disaggregation over NVMf, and show that NVMf reduces the application performance impact to a bare minimum, thus greatly improving cost efficiency and practicality.

Hyper-Convergence Infrastructure (HCI) is a software-defined approach that bundles commodity servers, each containing both compute and storage, into a clustered pool [47]. It abstracts the underlining hardware through a virtual computing platform, and allows management and scaling of compute and storage through a centralized management interface. HCI simplifies the infrastructure management, and is gaining popularity within the enterprise segment [47, 48]. In this paper we focus on web-scale data centers, which are generally not hyperconverged [49, 50]. Specifically, in order to better utilize hardware resources, web-scale data centers use few types of (sometimes highly-customized) servers, each with a different balance of compute, storage, memory, and network resources (e.g., [14]). Flash disaggregation fits well within this deployment model, as some of the server classes are already tuned specifically for storage. Moreover, NVMf, HCI, and storage disaggregation are not mutually exclusive: Excelero's utilizes NVMf for their NVMesh architecture that supports both hyperconverged and disaggregated storage configurations [51]. Likewise, hybrid HCI and disaggregation architectures are also being explored [52, 53].

Since NVMf v1.0 Specification [17] is relatively new, there is limited literature and analysis available. The design and architecture can be found in [54–56], but none provide detailed performance analysis. Xu et al. [44] provide an in-depth analysis of NVMe drive and their performance benefits for multiple cloud databases. Our work shows how NVMe benefits can be extended over the network using NVMf.

Recently, Intel introduced SPDK [41]: a set of tools and libraries for writing high performance user-mode storage applications that reduce kernel context switches and eliminate interrupt handling overheads. Moving drivers into userspace and operating in a polled mode will improve CPU utilization for both NVMf and iSCSI, as our initial studies have shown. We leave detailed evaluation of SPDK NVMf to future work as tools mature.

NVMf requires an RDMA-capable network. With both RoCE and iWARP enabling RDMA over Ethernet (or IP), RDMA is becoming more popular in the data center domain, and more applications are being ported to utilize it. Some notable implementation examples in addition to NVMf include SRP (SCSI RDMA Protocol) [21], iSER (iSCSI Extensions for RDMA) [22], Windows SMB Direct [23], Luster [24], Hadoop [25], Crail [26], and Ceph [27].

7. CONCLUSION

Datacenter applications' requirements for high-performance storage keep increasing. Consequently, NVMe-SSDs are increasingly being deployed to cater to the ever-growing demand for throughout and low-latency access. As NVMe-SSDs are becoming more prevalent in the datacenter, Flash disaggregation has been proposed to improve resource efficiency and reduce the overall cost of NVMe deployment. However, inefficiencies of existing remote storage protocols have hindered performance and limited disaggregation benefits. This work revisits Flash disaggregation using NVMe-over-Fabrics: a new protocol designed specifically to extend the high performance of NVMe to remote servers.

We provide the first, in-depth analysis of NVMf, characterizing end-to-end performance as well as the resource overheads on both the host and the target. Using synthetic stress-tests and a use-case from the datacenter domain (RocksDB), we compare NVMf with direct attached storage and with disaggregation via iSCSI. We show that NVMf delivers high performance and low latencies with minimal CPU overheads, consistent with its design goals. Our results indicate that for the most part, application will not see noticeable difference between accessing local storage vs. remote storage over NVMf. Lastly, we show that iSCSI introduce significant latencies for NVMe disaggregation (10x in some cases) and can degrade application performance by up to 40%.

8. REFERENCES

[1] Samsung. PM1725 NVMe PCIe SSD. http://www.samsung.com/semiconductor/global/file/insight/2015/11/pm1725-ProdOverview-2015-0.pdf, 2015.

[2] Ana Klimovic, Christos Kozyrakis, Eno Thereska, Binu John, and Sanjeev Kumar. Flash storage disaggregation. In *Proceedings of the Eleventh European Conference on Computer Systems*, EuroSys '16, pages 29:1–29:15, New York, NY, USA, 2016. ACM.

[3] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. Flash reliability in production: The expected and the unexpected. In *14th USENIX Conference on File and Storage Technologies (FAST 16)*, pages 67–80, Santa Clara, CA, 2016. USENIX Association.

[4] HGST. Linkedin scales to 200 million users with PCIe Flash storage from HGST. https://www.hgst.com/sites/default/files/resources/LinkedIn-Scales-to-200M-Users.pdf, 2014.

[5] Jian Ouyang, Shiding Lin, Song Jiang, Zhenyu Hou, Yong Wang, and Yuanzheng Wang. SDF: Software-defined flash for web-scale internet storage systems. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pages 471–484, New York, NY, USA, 2014. ACM.

[6] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

[7] Charlie Manese. Facebook and open compute, designing for efficiency and scale. 2014.

[8] Charles Loboz. Cloud resource usage–heavy tailed distributions invalidating traditional capacity planning models. *J. Grid Comput.*, 10(1):85–108, March 2012.

[9] Kestutis Patiejunas and Amisha Jaiswal. Facebook's disaggregated storage and compute for map/reduce. In *Data @Scale*, 2016.

[10] Brendan Cully, Jake Wires, Dutch Meyer, Kevin Jamieson, Keir Fraser, Tim Deegan, Daniel Stodden, Geoffre Lefebvre, Daniel Ferstay, and Andrew Warfield. Strata: High-performance scalable storage on virtualized non-volatile memory. In *Proceedings of the 12th USENIX conference on File and Storage Technologies (FAST)*, pages 17–31, 2014.

[11] Andrew Warfield. Architecting for "problematically fast" flash. COHO Data Virtualization Field Day 3, 2013.

[12] Amazon. Amazon elastic block store. https://aws.amazon.com/ebs/.

[13] VMWare. VMware vSAN. http://www.vmware.com/products/virtual-san.html.

[14] Facebook Inc. Open compute project. http://www.opencompute.org/projects, 2015.

[15] Edward K. Lee and Chandramohan A. Thekkath. Petal: Distributed virtual disks. *SIGOPS Oper. Syst. Rev.*, 30(5):84–92, September 1996.

[16] James Mickens, Edmund B. Nightingale, Jeremy Elson, Krishna Nareddy, Darren Gehring, Bin Fan, Asim Kadav, Vijay Chidambaram, and Osama Khan. Blizzard: Fast, cloud-scale block storage for cloud-oblivious applications. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 257–273, Berkeley, CA, USA, 2014. USENIX Association.

[17] NVM Express. NVM Express over Fabric 1.0. http://www.nvmexpress.org/, 2016.

[18] Amber Huffman. NVM Express Revision 1.1. http://www.nvmexpress.org/, 2012.

[19] FusionIO. Fusion-io Flash Memory as RAM Relief. 2013.

[20] X bit labs. OCZ Demos 4 TiB, 16 TiB Solid-State Drives for Enterprise, 2016.

[21] T10. SCSI RDMA protocol-2 (SRP-2). http://www.t10.org/ftp/t10/drafts/srp2/srp2r00a.pdf, 2003.

[22] Mike Ko, J Hufferd, M Chadalapaka, Uri Elzur, H Shah, and P Thaler. iSCSI extensions for RDMA specification (version 1.0). *Release Specification of the RDMA Consortium*, 2003.

[23] Microsoft. Improve performance of a file server with SMB direct. https://technet.microsoft.com/en-us/library/jj134210(v=ws.11).aspx.

[24] Chelsio Communications. Luster over iWARP RDMA at 40Gbps.

[25] Xiaoyi Lu, Nusrat S Islam, Md Wasi-Ur-Rahman, Jithin Jose, Hari Subramoni, Hao Wang, and Dhabaleswar K Panda. High-performance design of Hadoop RPC with RDMA over InfiniBand. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 641–650. IEEE, 2013.

[26] IBM Research. Crail. http://www.crail.io/, 2017.

[27] John F. Kim. Accelerating Ceph with flash and high speed networks. Storage Developer Conference. SNIA, 2014.

[28] Ro Recio, P Culley, D Garcia, J Hilland, and B Metzler. An RDMA protocol specification. Technical report, IETF Internet-draft draft-ietf-rddp-rdmap-03. txt (work in progress), 2005.

[29] Jiuxing Liu, Jiesheng Wu, Sushmitha P Kini, Pete Wyckoff, and Dhabaleswar K Panda. High performance RDMA-based MPI implementation over InfiniBand. In *Proceedings of the 17th annual international conference on Supercomputing*, pages 295–304. ACM, 2003.

[30] David Cohen, Thomas Talpey, Arkady Kanevsky, Uri Cummings, Michael Krause, Renato Recio, Diego Crupnicoff, Lloyd Dickman, and Paul Grun. Remote direct memory access over the converged enhanced ethernet fabric: Evaluating the options. In *High Performance Interconnects, 2009. HOTI 2009. 17th IEEE Symposium on*, pages 123–130. IEEE, 2009.

[31] RDMA Consortium. Architectural specifications for RDMA over TCP/IP. Technical report.

[32] Dell Networking. RDMA over Converged Ethernet Technical Brief. 2015.

[33] Intel. Intel Xeon processor E5-2699 v4. https://ark.intel.com/products/91317/Intel-Xeon-Processor-E5-2699-v4-55M-Cache-2_20-GHz, 2016.

[34] Mellanox. Connect X-4 VPI 100Gb. http://www.mellanox.com/related-docs/prod_

adapter_cards/PB_ConnectX-4_VPI_OCP.pdf, 2015.

[35] Mellanox. SN2700. https://www.mellanox.com/related-docs/prod_eth_switches/PB_SN2700.pdf, 2015.

[36] Jens Axboe. FIO. https://github.com/axboe/fio, 2014.

[37] Facebook Inc. RocksDB: A persistent key-value store for fast storage environments. http://rocksdb.org, 2015.

[38] RocksDB users. https://github.com/facebook/rocksdb/blob/master/USERS.md.

[39] Dave Minturn. NVMe over fabrics linux driver eco system. NVMe All Hands Meeting, 2016.

[40] Steven Rostedt. ftrace - Function Tracer. https://www.kernel.org/doc/Documentation/trace/ftrace.txt, 2008.

[41] Storage performance development kit. http://www.spdk.io/.

[42] Benjamin Walker. SPDK: Building blocks for scalable, high performance storage applications. Storage Developer Conference. SNIA, 2016.

[43] Jisoo Yang, Dave B. Minturn, and Frank Hady. When poll is better than interrupt. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST'12, pages 3–3, Berkeley, CA, USA, 2012. USENIX Association.

[44] Qiumin Xu, Huzefa Siyamwala, Mrinmoy Ghosh, Tameesh Suri, Manu Awasthi, Zvika Guz, Anahita Shayesteh, and Vijay Balakrishnan. Performance analysis of nvme ssds and their implication on real world databases. In *Proceedings of the 8th ACM International Systems and Storage Conference*, SYSTOR '15, pages 6:1–6:11, New York, NY, USA, 2015. ACM.

[45] Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks. Profiling a warehouse-scale computer. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 158–169, New York, NY, USA, 2015. ACM.

[46] Sangjin Han, Norbert Egi, Aurojit Panda, Sylvia Ratnasamy, Guangyu Shi, and Scott Shenker. Network support for resource disaggregation in next-generation datacenters. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, page 10. ACM, 2013.

[47] Simon Robinson, John Abott, and Tim Stammers. The emergence of hyperconvergence. Technical report, 451 Research, March 2015.

[48] Julia Parlmer and Stanley Zaffos. How to determine when hyperconverged integrated systems can replace traditional storage. Technical report, Gartner, January 2016.

[49] Kieran Harty. Don't confuse hyperconvergence with web-scale. http://www.networkcomputing.com/data-centers/dont-confuse-hyperconvergence-web-scale/445839104, 2016.

[50] Brandon Salmon. Web scale vs. hyperconverged: Understand the differences. http://www.infoworld.com/article/3005572/enterprise-architecture/web-scale-vs-hyperconverged-understand-the-differences.html, 2015.

[51] Excelero. Excelero NVMesh. https://www.excelero.com/product/nvmesh/.

[52] Simon Sharwood. Disaggregated hyper-convergence thinks storage outside the box. https://www.theregister.co.uk/2016/03/24/disaggregated_hyper_convergence/, March 2016.

[53] Datium. Open convergence. http://www.datrium.com/open-convergence/.

[54] Dave Minturn and J Metz. Under the hood with NVMe over Fabrics. Ethernet Storage Forum. SNIA, 2015.

[55] John Kim and David Fair. How Ethernet RDMA protocols iWARP and RoCE support NVMe over Fabrics. Ethernet Storage Forum. SNIA, 2016.

[56] Brandon Hoff. RDMA Interconnects Paving the Way for NVMe over Fabrics Technology. http://www.roceinitiative.org/, 2016.