Storage Systems (StoSys)
XM_0092

Lecture 7: Networked NVM Storage

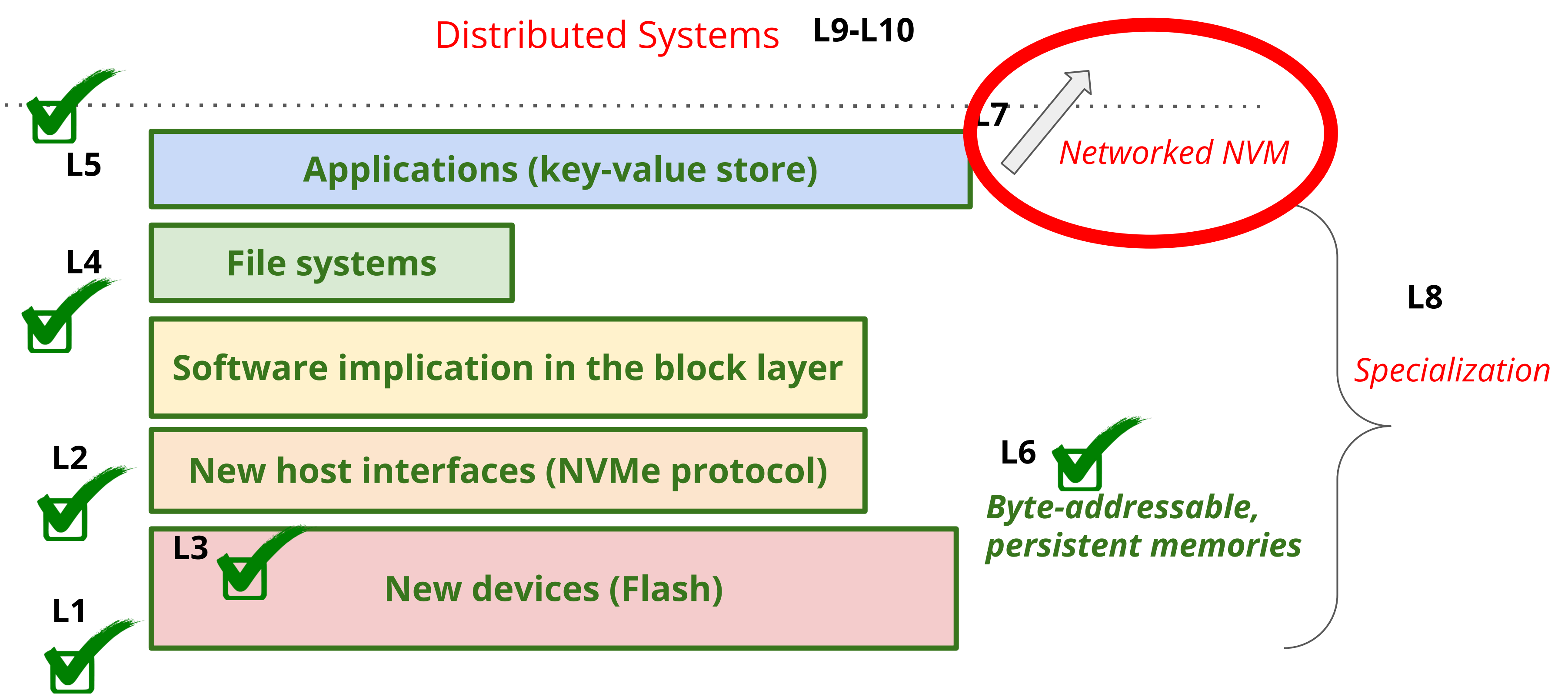
Animesh Trivedi
https://stonet-research.github.io/
Autumn 2023, Period 1

VRIJE UNIVERSITEIT AMSTERDAM

Syllabus outline

1. ~~Welcome and introduction to NVM (today)~~
2. ~~Host interfacing and software implications~~
3. ~~Flash Translation Layer (FTL) and Garbage Collection (GC)~~
4. ~~NVM Block Storage File systems~~
5. ~~NVM Block Storage Key-Value Stores~~
6. ~~Emerging Byte-addressable Storage~~
7. Networked NVM Storage
8. Trends: Programmability
9. Distributed Storage / Systems - I
10. Distributed Storage / Systems - II
11. Emerging topics

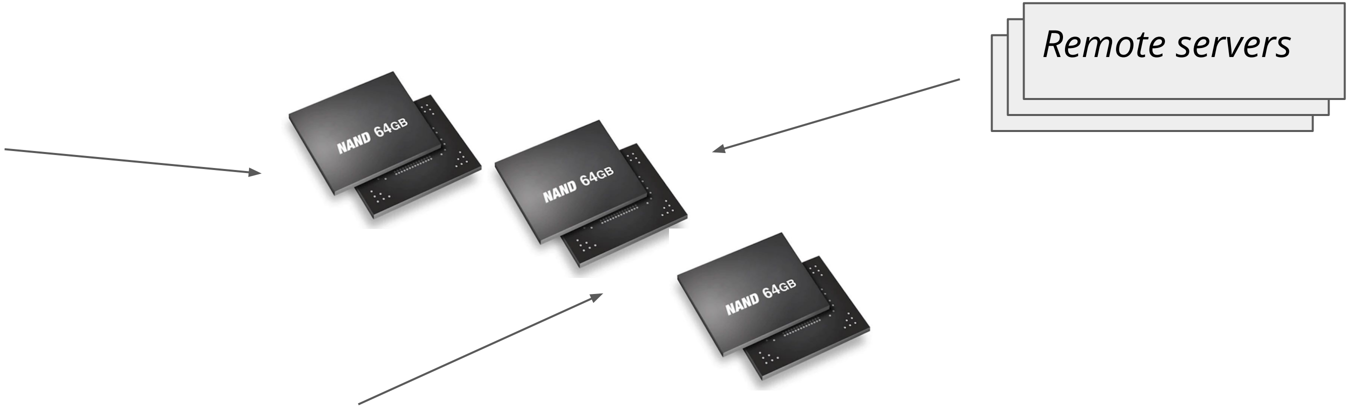
The layered approach in the lectures



Networking Storage

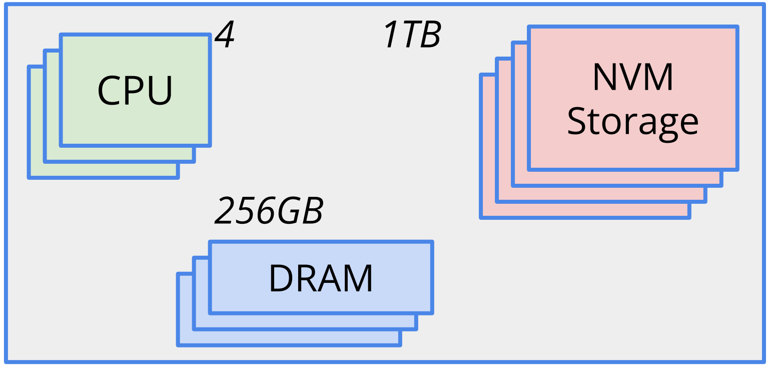
Question 1: why do we want to network storage?

Question 2: what do you think when I say networked storage? (ever heard of NAS, SAN, FC, iSCSI?)



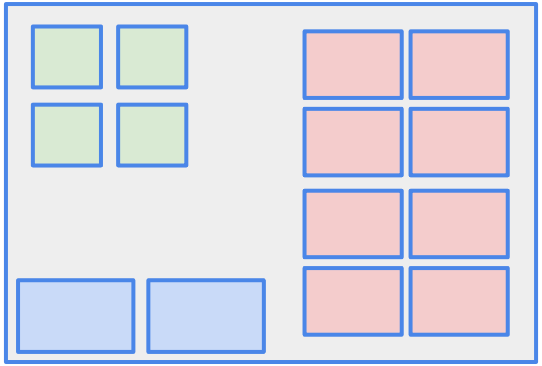
Server and Workloads

What do we have inside a single server: CPU cores, DRAM, and some storage



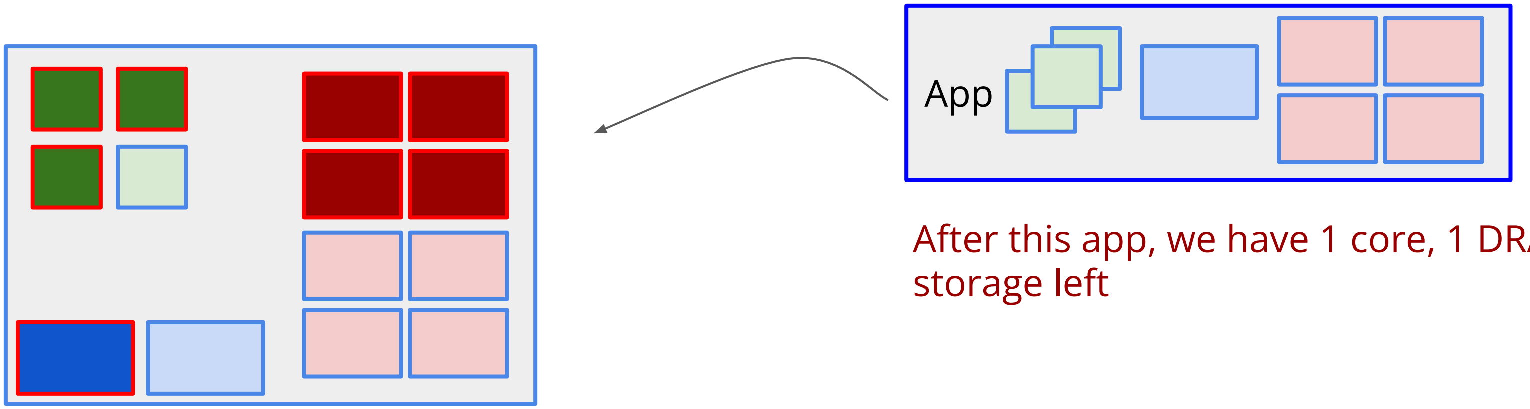
Server and Workloads

What do we have inside a single server: CPU cores, DRAM, and some storage



Server and Workloads

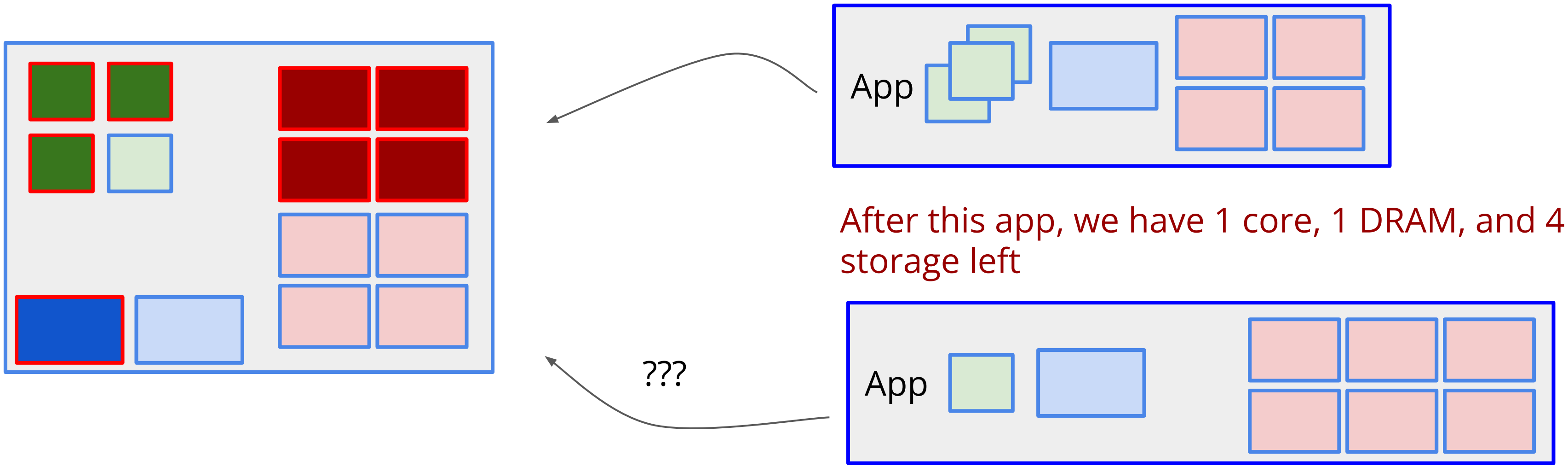
What do we have inside a single server: CPU cores, DRAM, and some storage



After this app, we have 1 core, 1 DRAM, and 4 storage left

Server and Workloads

What do we have inside a single server: CPU cores, DRAM, and some storage



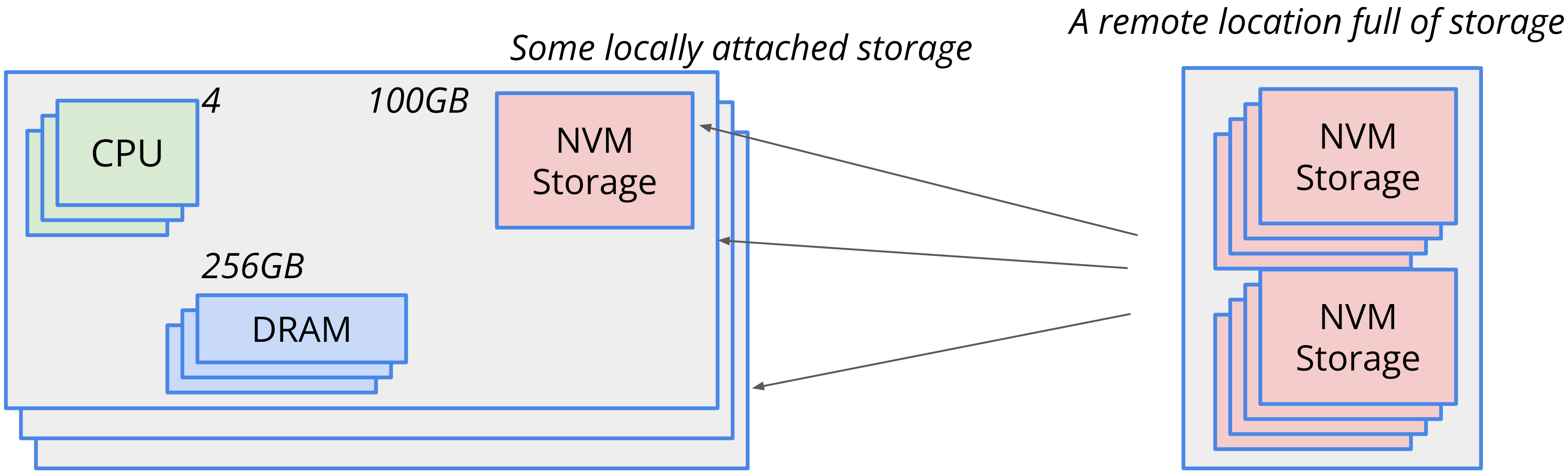
What happens if an application needs

- 3 cores only, or 5 cores?
- 1.1 TB of NVM or only 500 GB?
- 128GB of DRAM, or 512 GB of DRAM

Issues

- Low resource utilization
- High cost of running infrastructure
    - Total cost of ownership (TCO)

Idea: Disaggregation (Storage)

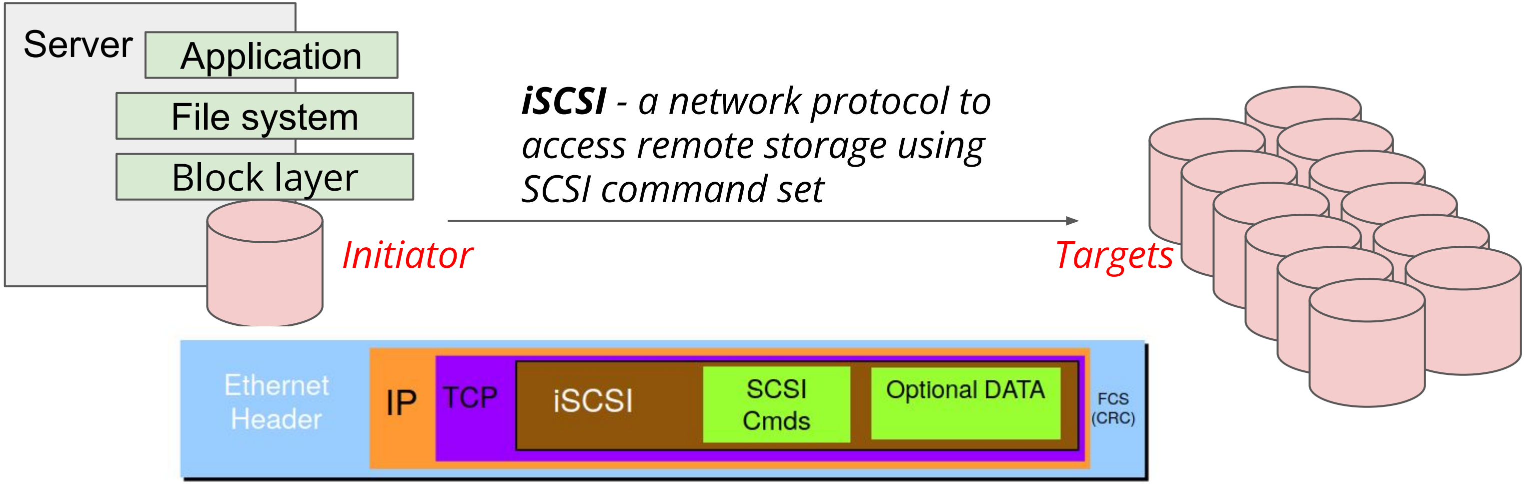


Slice and give out storage capacity from a remote location (dedicated storage servers)

The idea is not new : this is how even HDD based storage systems are also deployed. Benefits:

(i) on-demand device capacity provisioning, no underutilization
(ii) centralized provisioning, and management, a single point of upgrade to all
(iii) low cost TCO, as systems resources are fully utilized (with a mix of workloads)

How to Access Remote Storage - SAN



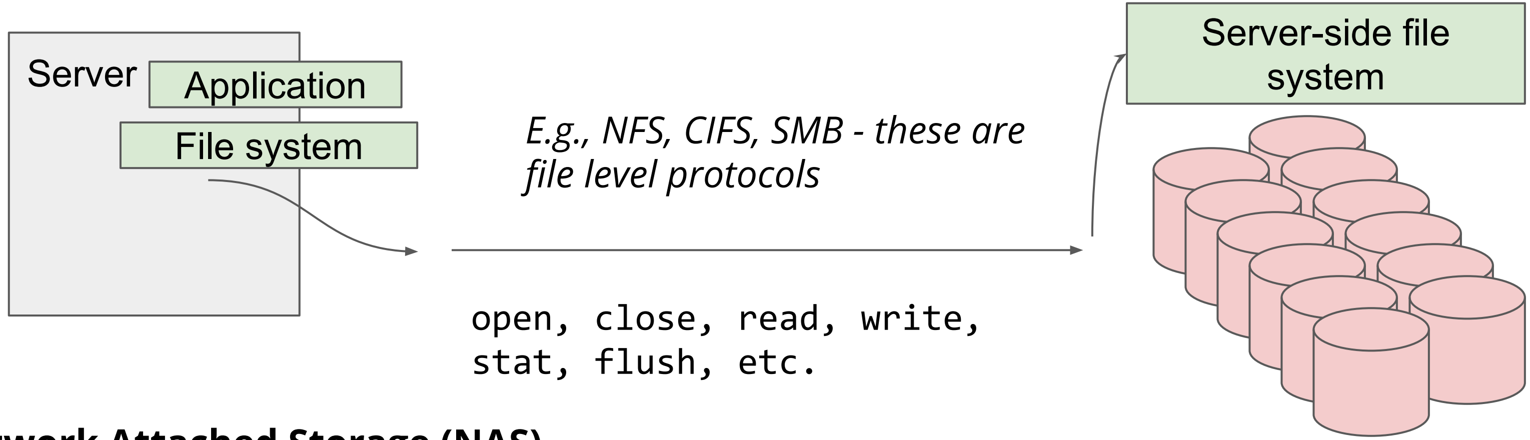
Storage Area Network (SAN)

- One of the most popular way of deploying “remote” block storage
- Block storage size can be anything, configured on demand (persistent or ephemeral)
- Deployable on the common data center networking infrastructure: Ethernet, TCP, IP

There are other ways to do SAN as well like ATA over Ethernet (AoE), Fiber Channel (FC), etc.

IP Storage Protocols: iSCSI, https://www.snia.org/sites/default/education/tutorials/2011/spring/networking/HufferdJohn-IP_Storage_Protocols-iSCSI.pdf

How to Access Remote Storage - NAS

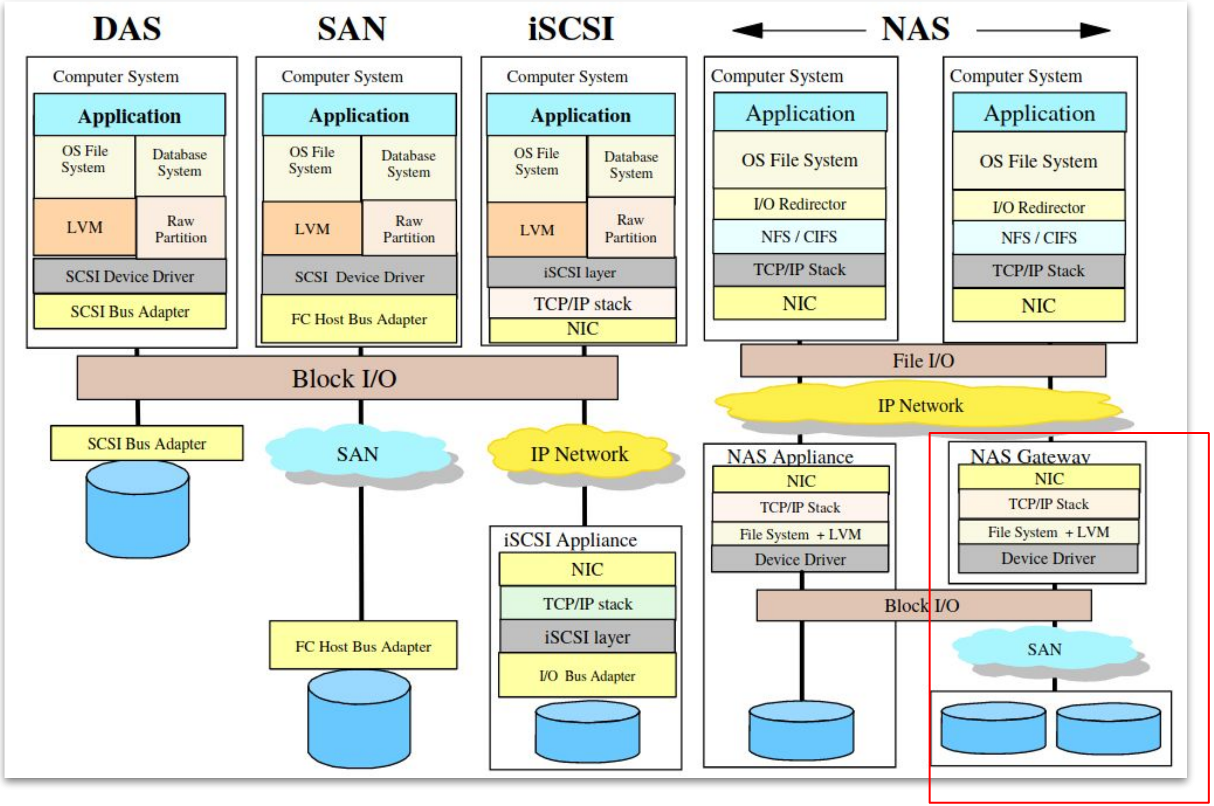


Network Attached Storage (NAS)

- Deployment abstraction is **a file**
  - can be a just a point-to-point file system (NFS), e.g., https://www.rfc-editor.org/rfc/rfc1813
  - a shared, parallel file system (like GPFS, GlusterFS, Ceph) running on distributed block devices
- Capacity provisioning and scaling is done at the file system level

In the cloud, similar example would include Hadoop FS

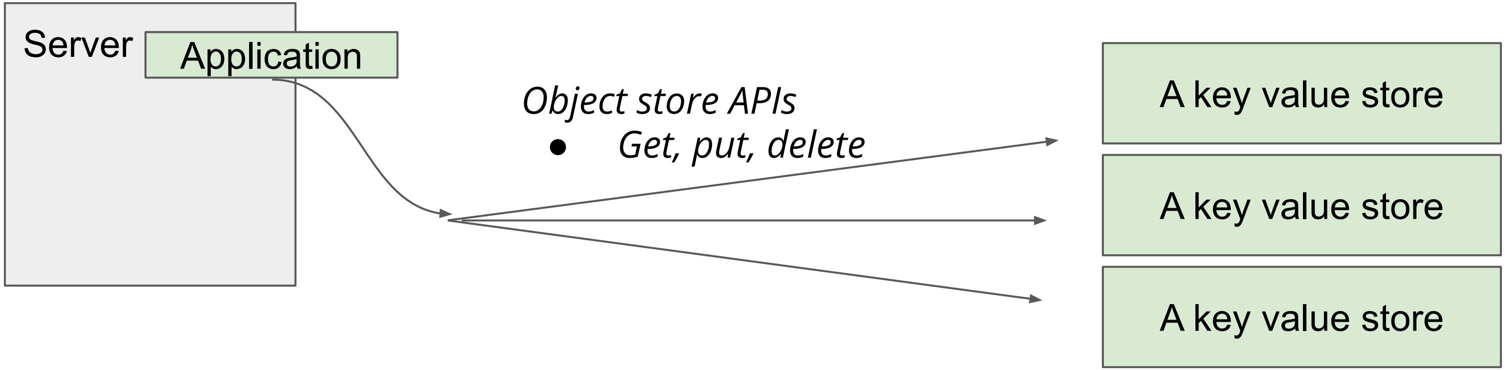
Accessing Remote Storage



NAS on SAN

NAS and iSCSI Technology Overview, https://www.snia.org/sites/default/education/tutorials/2007/fall/storage/WolfgangSinger_%20NAS_and_ISCSI_Technology.pdf

How to Access Remote Storage - Object



If not being restricted to files or blocks for storage, objects are **flexible** (flat namespace, simple locking), **scalable** (can be distributed over multiple servers), and can support **multiple consistency models**

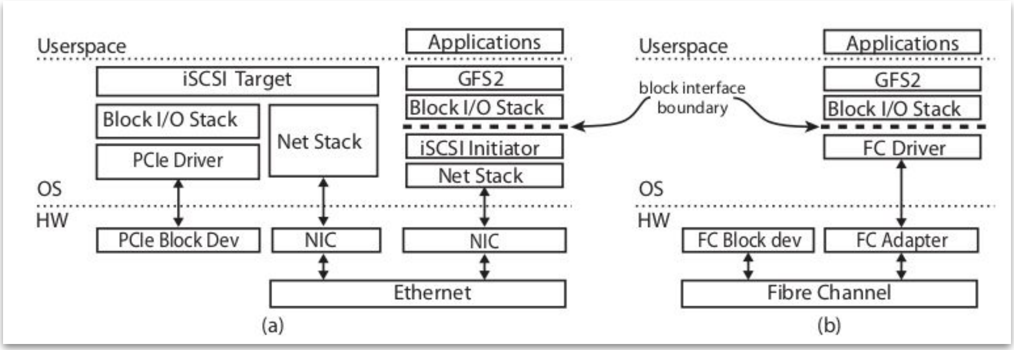
Examples: redis S3 Microsoft Azure Blob Storage

Object Storage 101 Understanding the What, How and Why behind Object Storage Technologies, https://www.snia.org/sites/default/files/Object_Storage_101.pdf

What is the Basic Challenge Here?

Software Time	Network Time	Storage Media Time

Total operation latency. (Often) Mostly dominated by the storage media access time, that was HDD performance

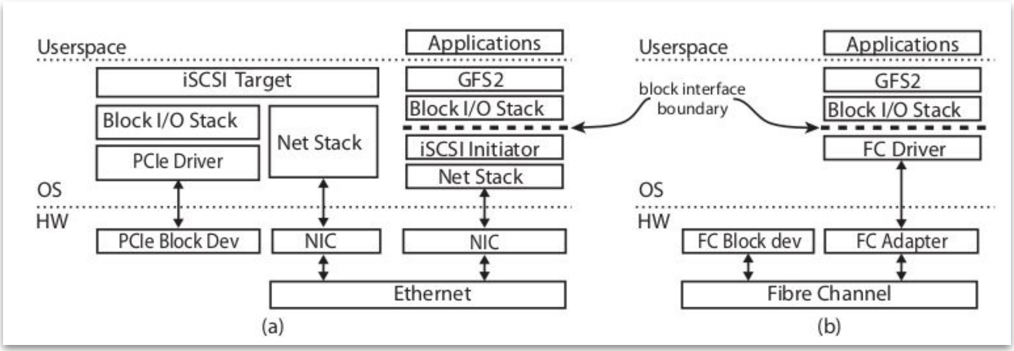


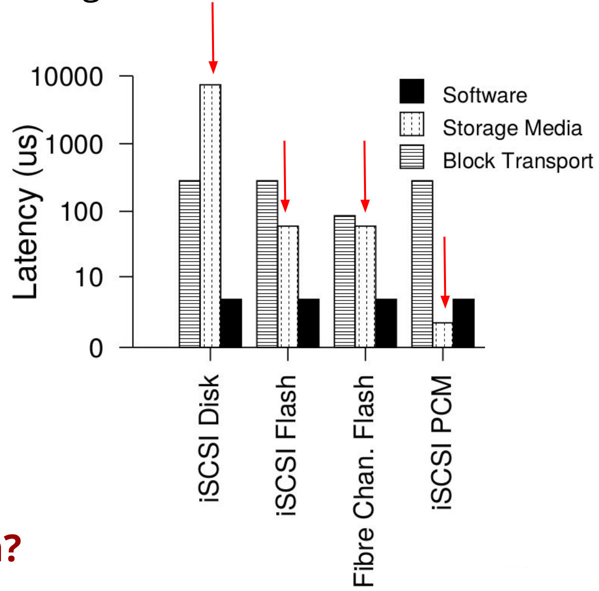
Adrian M. Caulfield and Steven Swanson. QuickSAN: a storage area network for fast, distributed, solid state disks. In ISCA 2013.

What is the Basic Challenge Here?

Software Time	Network Time	Storage Media Time

Total operation latency. (Often) Mostly dominated by the storage media access time, that was HDD performance

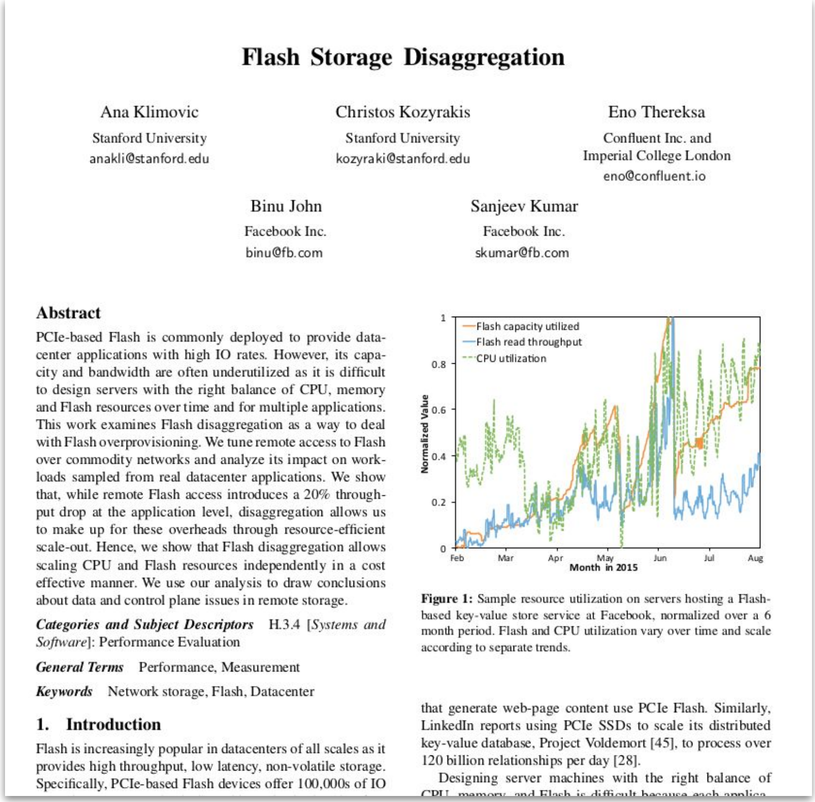




As storage media access time improved, <u>software</u> and <u>network</u> time became the new bottlenecks - what can we do about them?

Adrian M. Caulfield and Steven Swanson. QuickSAN: a storage area network for fast, distributed, solid state disks. In ISCA 2013.

Understanding iSCSI with Disaggregated Flash



Flash Storage Disaggregation

Ana Klimovic
Stanford University
anakli@stanford.edu

Christos Kozyrakis
Stanford University
kozyraki@stanford.edu

Eno Thereksa
Confluent Inc. and
Imperial College London
eno@confluent.io

Binu John
Facebook Inc.
binu@fb.com

Sanjeev Kumar
Facebook Inc.
skumar@fb.com

Abstract

PCIe-based Flash is commonly deployed to provide datacenter applications with high IO rates. However, its capacity and bandwidth are often underutilized as it is difficult to design servers with the right balance of CPU, memory and Flash resources over time and for multiple applications. This work examines Flash disaggregation as a way to deal with Flash overprovisioning. We tune remote access to Flash over commodity networks and analyze its impact on workloads sampled from real datacenter applications. We show that, while remote Flash access introduces a 20% throughput drop at the application level, disaggregation allows us to make up for these overheads through resource-efficient scale-out. Hence, we show that Flash disaggregation allows scaling CPU and Flash resources independently in a cost effective manner. We use our analysis to draw conclusions about data and control plane issues in remote storage.

Categories and Subject Descriptors H.3.4 [*Systems and Software*]: Performance Evaluation

General Terms Performance, Measurement

Keywords Network storage, Flash, Datacenter

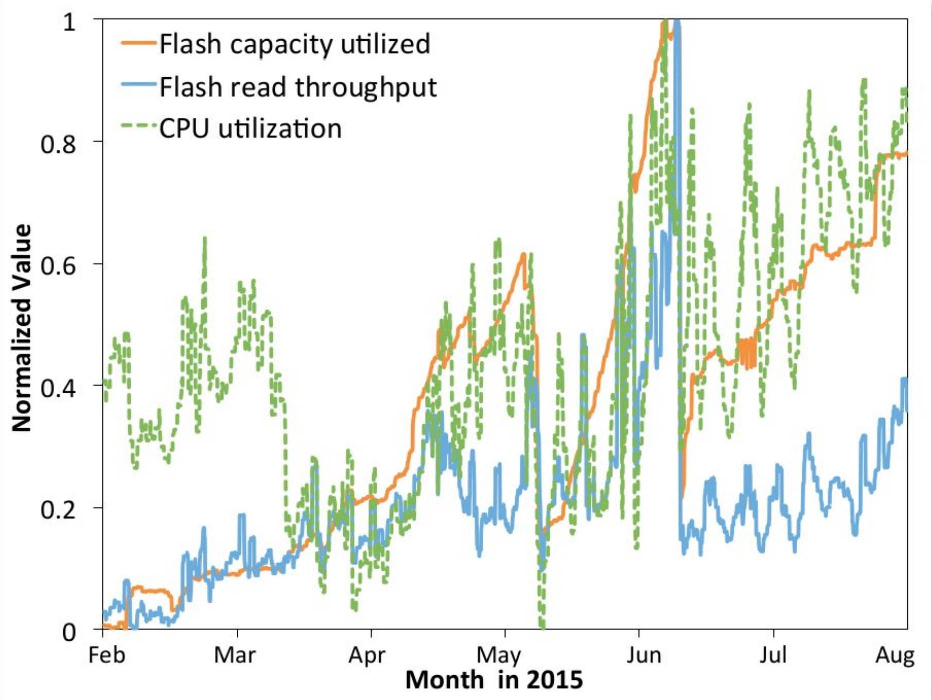
1. Introduction

Flash is increasingly popular in datacenters of all scales as it provides high throughput, low latency, non-volatile storage. Specifically, PCIe-based Flash devices offer 100,000s of IO

Figure 1: Sample resource utilization on servers hosting a Flash-based key-value store service at Facebook, normalized over a 6 month period. Flash and CPU utilization vary over time and scale according to separate trends.

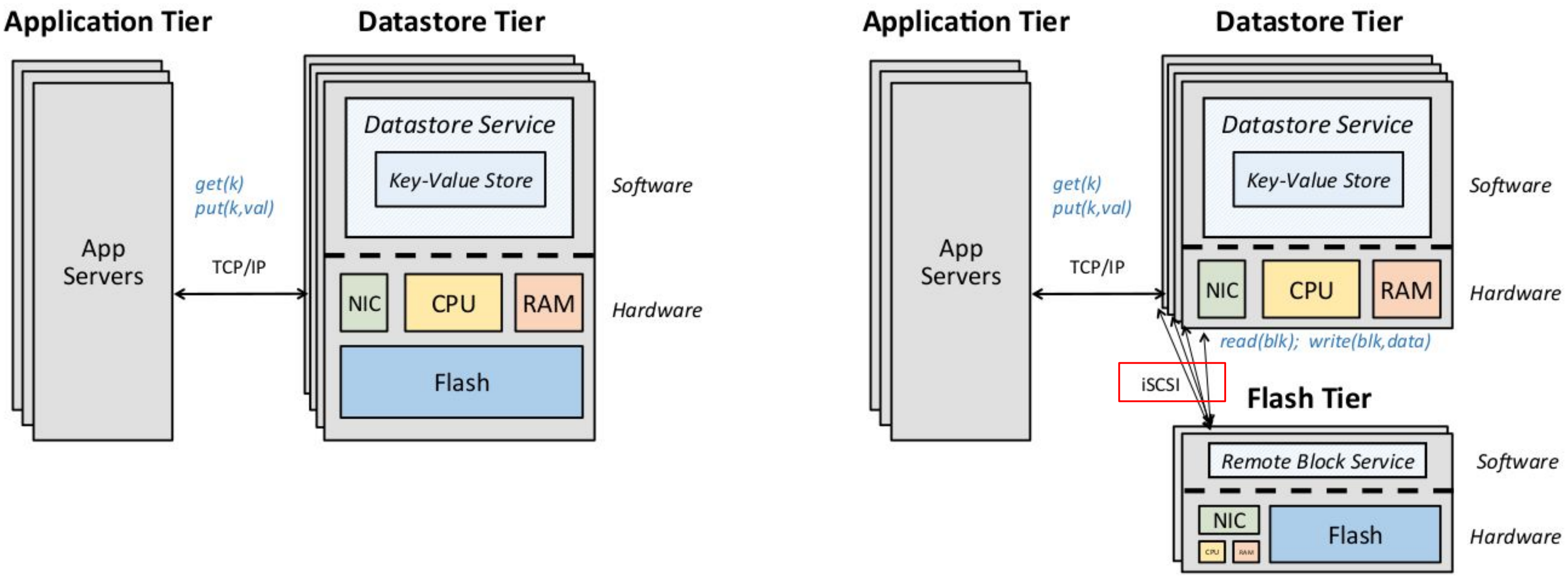
that generate web-page content use PCIe Flash. Similarly, LinkedIn reports using PCIe SSDs to scale its distributed key-value database, Project Voldemort [45], to process over 120 billion relationships per day [28].

Designing server machines with the right balance of



Under utilization of resources

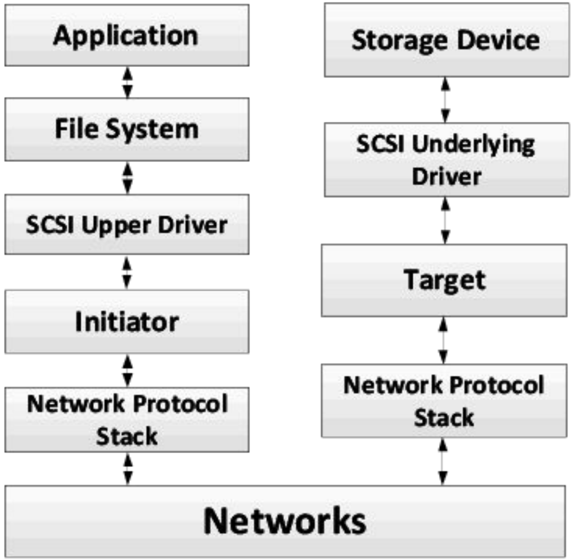
Deployment Setup with Disaggregated Flash

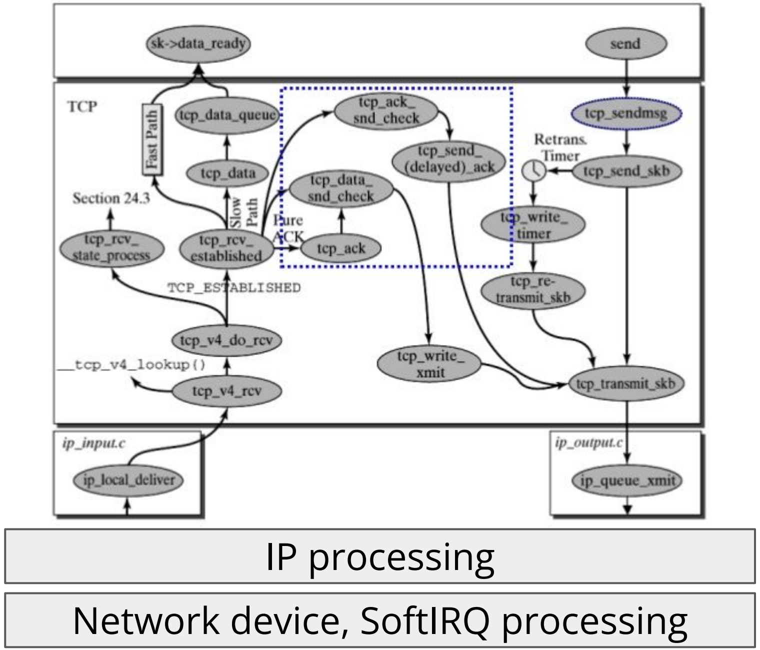


(a) Direct-attached (local) Flash

(b) Disaggregated (remote) Flash

iSCSI Processing (+Networking) in Linux



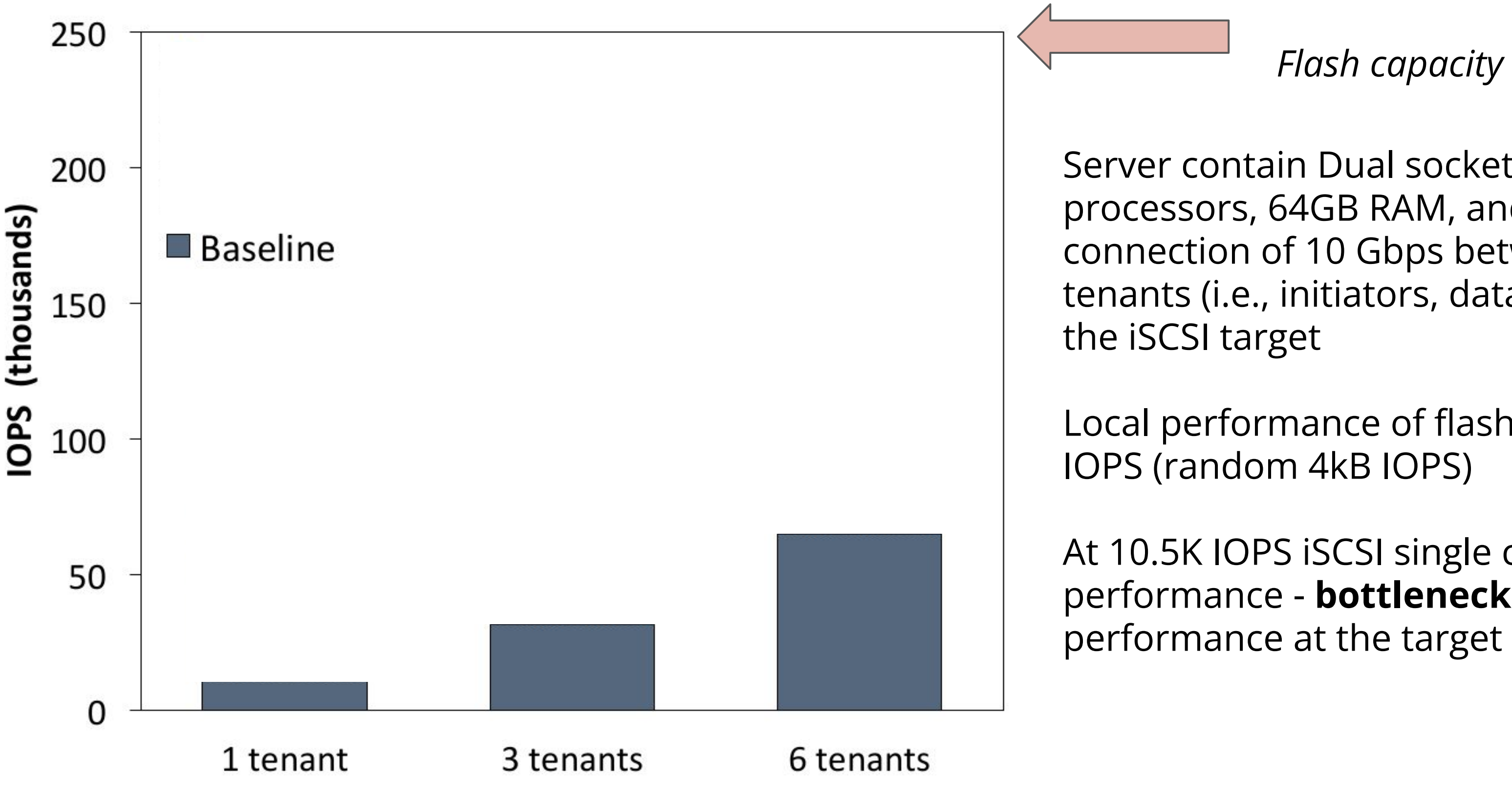


- Initiator and Target iSCSI terminology
- iSCSI become a high-level protocol on top of conventional TCP/IP processing

For more details, see Advanced Network Programming, (Bsc, 3rd year Programming Minor course)

https://the-linux-channel.the-toffee-project.org/index.php?page=3-links-linux-kernel-network-stack-and-architecture

Understand Network Optimizations

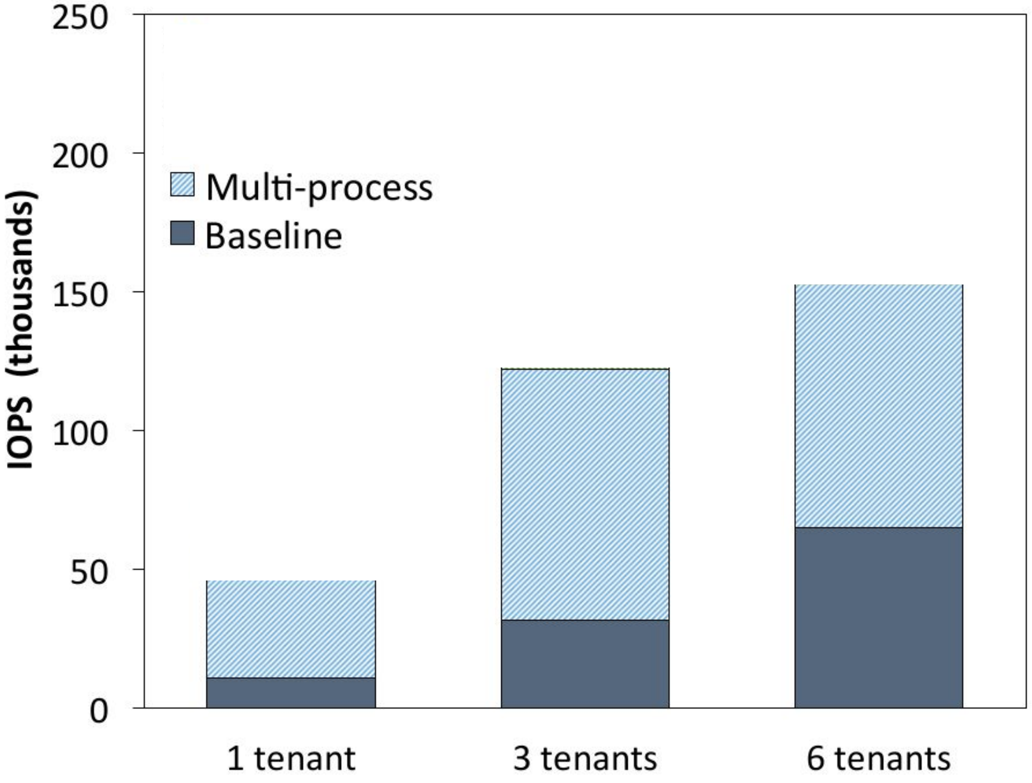


Server contain Dual socket Xeon processors, 64GB RAM, and network connection of 10 Gbps between tenants (i.e., initiators, datastores) and the iSCSI target

Local performance of flash is at 250K IOPS (random 4kB IOPS)

At 10.5K IOPS iSCSI single client performance - **bottleneck:** CPU performance at the target

Understand Network Optimizations



Optimize network processing scalability

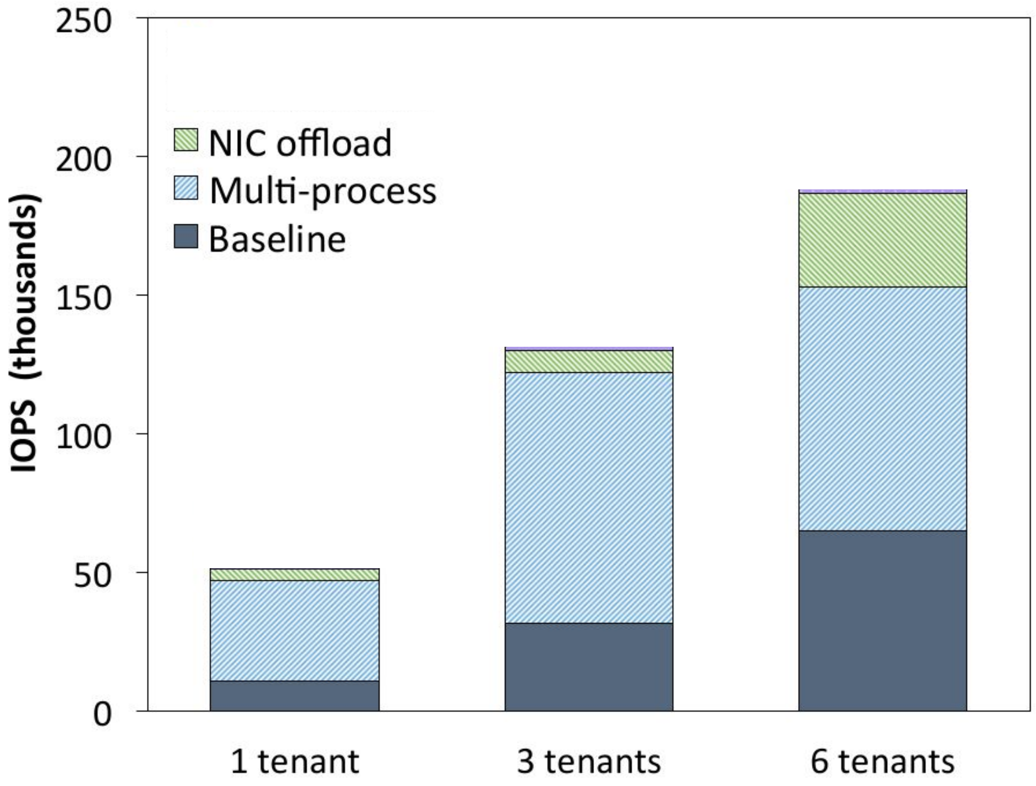
By default iSCSI uses 1 request processing thread per session

Use multiple threads per session to leverage multicore systems (use 6 out of the 8 cores available, *why?*)

Almost 4-5x gains

(not shown) With 8 tenants it can do 250K (device bounded)

Understand Network Optimizations



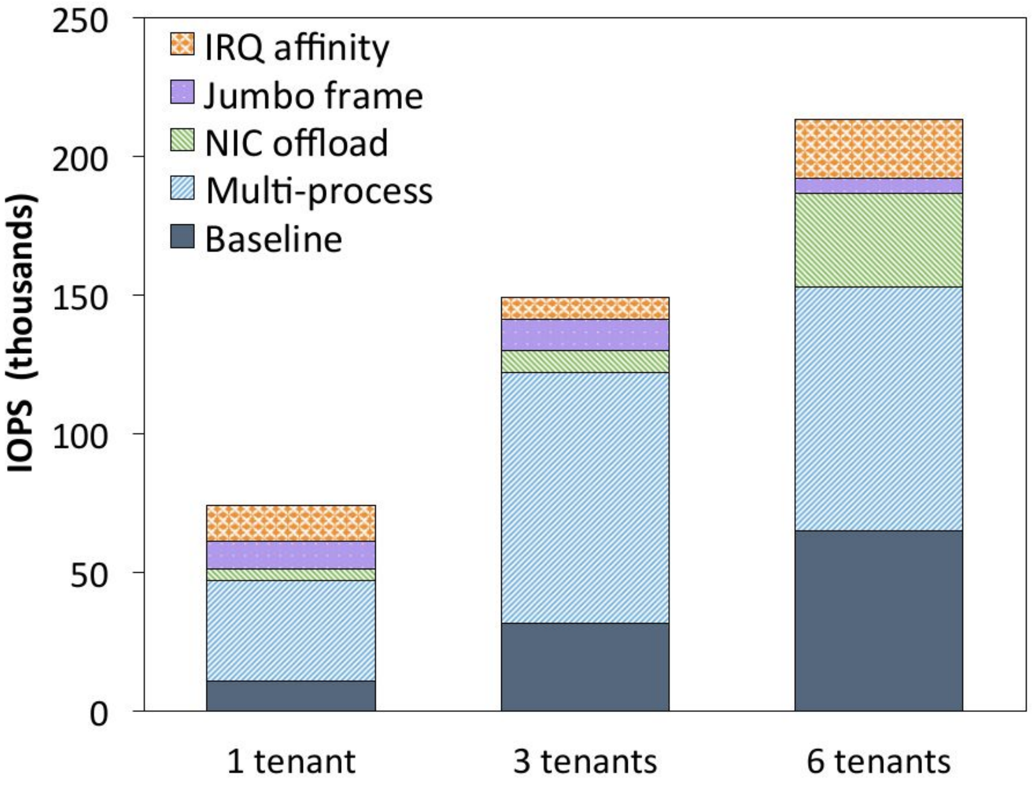
Optimize network offloading

Enable TSO and LRO offloading

TCP segmentation offloading (TSO)
Large receive offloading (LRO)

These network controller features help to reduce per packet overheads by coleasing multiple 1500 bytes packets into a large segments (~64kB)

Understand Network Optimizations



Optimize network offloading

Enable jumbo frames and IRQ affinity

Jumbo frames: default Ethernet frames are 1500 bytes, jumbo frames 9000 bytes
→ Help to reduce per packet overheads

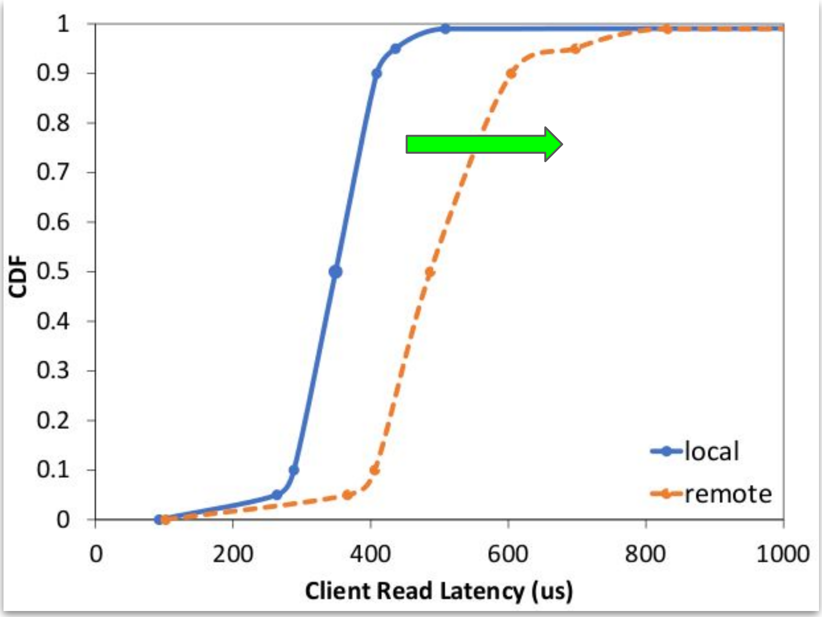
IRQ affinity is used to distributed interrupts from NICs to all cores for scalable processing

Application-Level Performance

Run RocksDB on disaggregated flash devices

Remote flash does increase the 95th percentile latency by 260μs

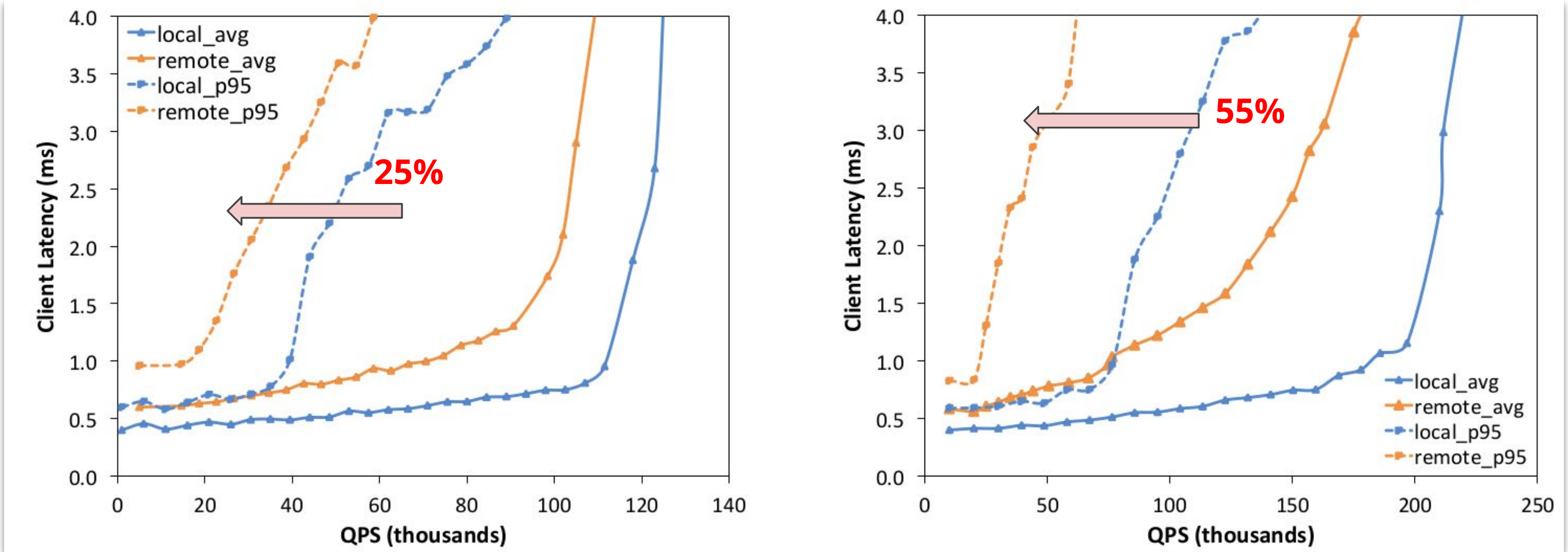
- **Is this acceptable?** Depends upon the application. If your SLOs are in mseconds then yes
  - FB's use-cases are in mseconds
- If they are in 100 useconds - No



Unloaded latency

What happens when multiple tenants share flash devices over the network?

Multi-Tenancy Loaded Latency



Comparison points: **local** (when each tenant has its own local flash) vs. **remote** when shared between 2 (left) and 3 (right) tenants

Observations: QPS is degraded by ~20%, but tail suffers significantly as we increase multi-tenancy
Left figure 2x application flash sharing, right 3x applications- notice the tail latencies

When does Disaggregation Make Sense?

Let's do a first-order approximation for the benefits of disaggregation

$$C_{direct} = \max\left(\frac{\text{GB}_t}{\text{GB}_s}, \frac{\text{IOPS}_t}{\text{IOPS}_s}, \frac{\text{QPS}_t}{\text{QPS}_s}\right) \cdot (f + c)$$

Maximum capacity required

Maximum capacity per machine

Sum of flash + compute capacity in a single server

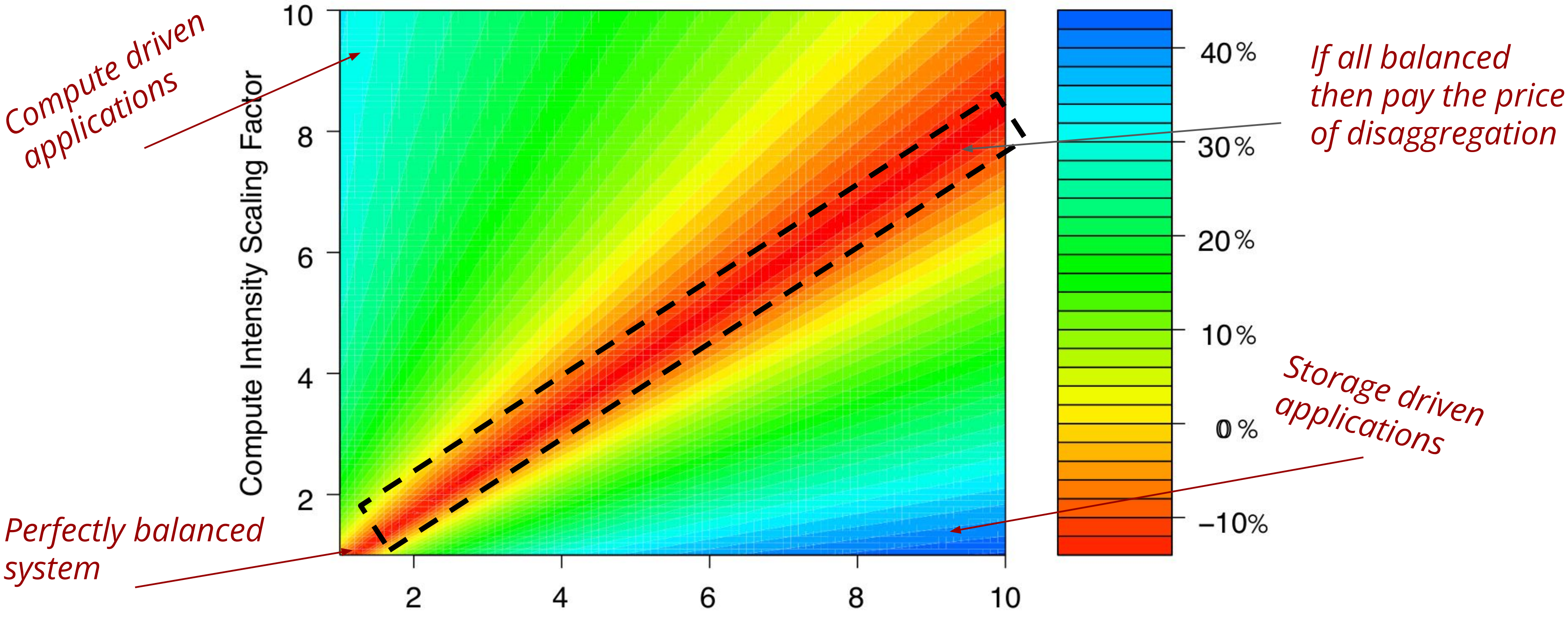
What are the minimum number of servers needed to support an application?

$$C_{disagg} = \max\left(\frac{\text{GB}_t}{\text{GB}_s}, \frac{\text{IOPS}_t}{\text{IOPS}_s}\right) \cdot (f + \delta) + \left(\frac{\text{QPS}_t}{\text{QPS}_s}\right) c$$

Only flash requirements
Multiplied with the cost of flash
+ disaggregation overheads (20%)

Completely separate scaling of compute requirements

When does Disaggregation Make Sense



When does disaggregation makes sense: *when compute and storage demands scale at a different rate (which in real world happens often)*

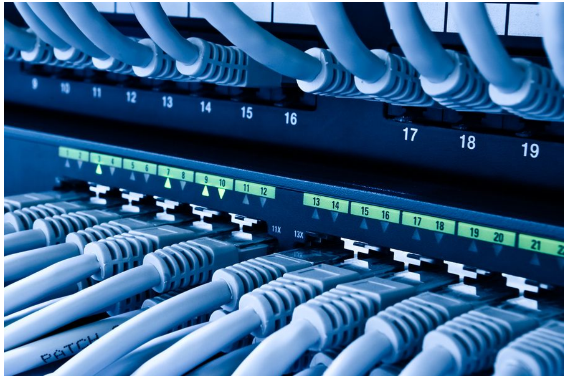
What are the Challenges with Storage Disaggregation

1. Come up with a better protocol than iSCSI? (hint: we did already for locally connected flash)
2. What can we do to improve multi-tenancy for disaggregated flash?
3. What kind of joint network and storage optimizations we can do to decrease the software cost of accessing remote storage?
4. Come up with a better remote data access API than just simple block, files, or objects?
5. Very active area of research!

And many other variants of these themes, let's start with a better protocol

Faster Storage Needs a Faster Network

We are seeing networking performance improve from **100 Gbps to 200 Gbps and 400 Gbps**



They can deliver < 1-10 usec latencies to access remote DRAM buffers

New ways of doing network operations like RDMA enabled networks like InfiniBand, iWARP, RoCE

- Allows network processing inside the network controller (not the CPU)

How do we leverage the performance of these networks with storage?

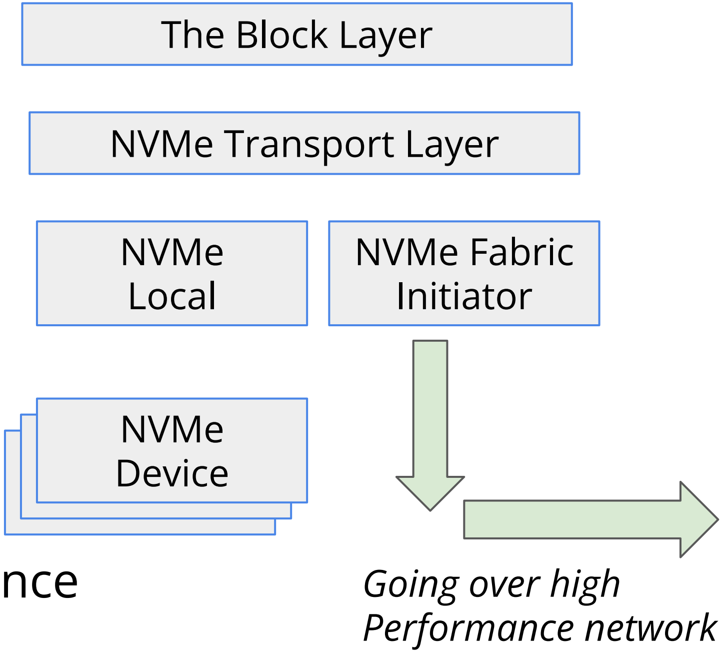
NVM Express over Fabrics (NVMe-oF)

NVM Express

- Command and completion queues
- PCIe directly mapped queues
- Light-weight protocol

NVMe over Fabrics is a networked extension of this idea

What is the "Fabrics" here?
It is an umbrella term to cover high-performance networks like RDMA networks



Remote Direct Memory Access (RDMA)

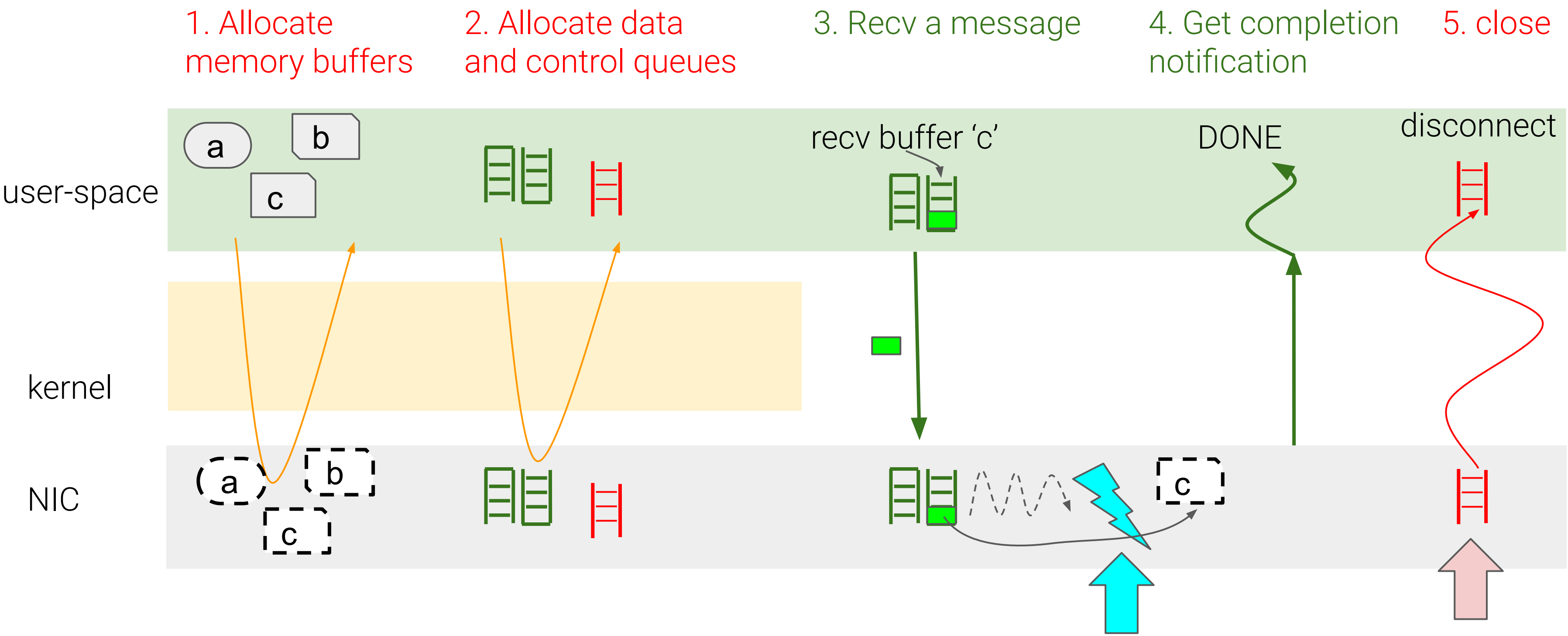
A Userspace networking technology, applications have

- Directly mapped RX/TX queues in the userspace
- Can execute **send/recv commands**
- Can execute **remote memory read/write** operations
- Poll or interrupt driven completion notifications
- All networking processing is offloaded to the hardware (Network controller)

The interesting thing for us here is that RDMA is also (i) a queue-based; (ii) post commands; (iii) poll for completion - type network operation

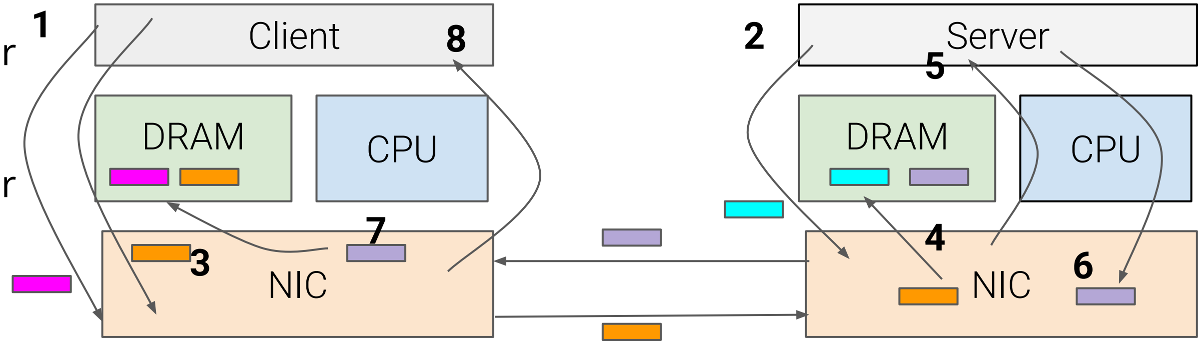
Animesh Trivedi, Patrick Stuedi, Bernard Metzler, Roman Pletka, Blake G. Fitch, and Thomas R. Gross. 2013. Unified high-performance I/O: one stack to rule them all. In Proceedings of the 14th USENIX conference on Hot Topics in Operating Systems (HotOS'13). USENIX Association, USA, 4.

RDMA Operations



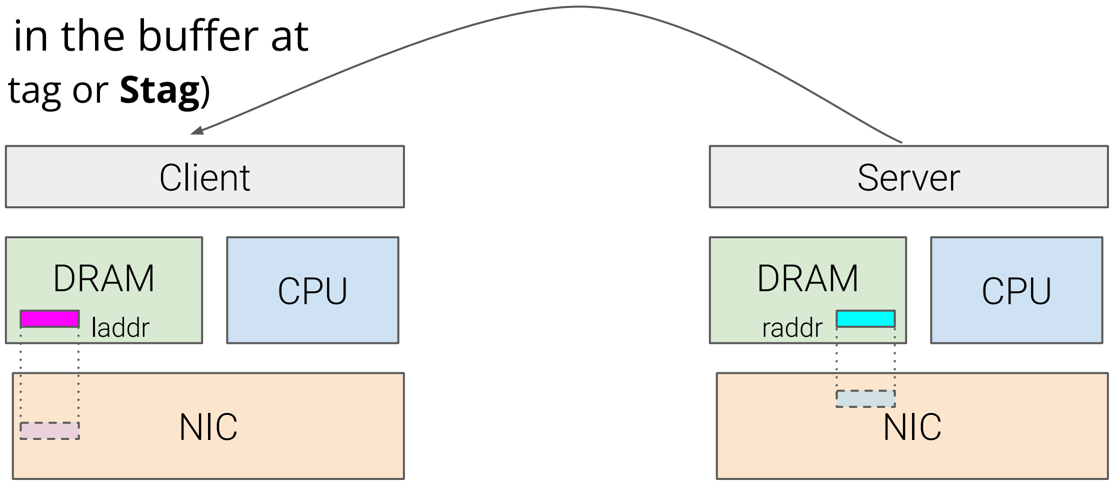
RDMA: Two-Sided Send Recv Operations

1. Client posts a receive buffer (pink)
2. Server posts a receive buffer (cyan)
3. Client sends a buffer to the server (orange)
4. Server's NIC receives the buffer and deposit it into the cyan buffer
5. NIC notifies the server
6. Server prepares a response and send back the purple buffer
7. Client NIC receives the purple buffer and deposit it into the pink buffer
8. NIC notifies the client



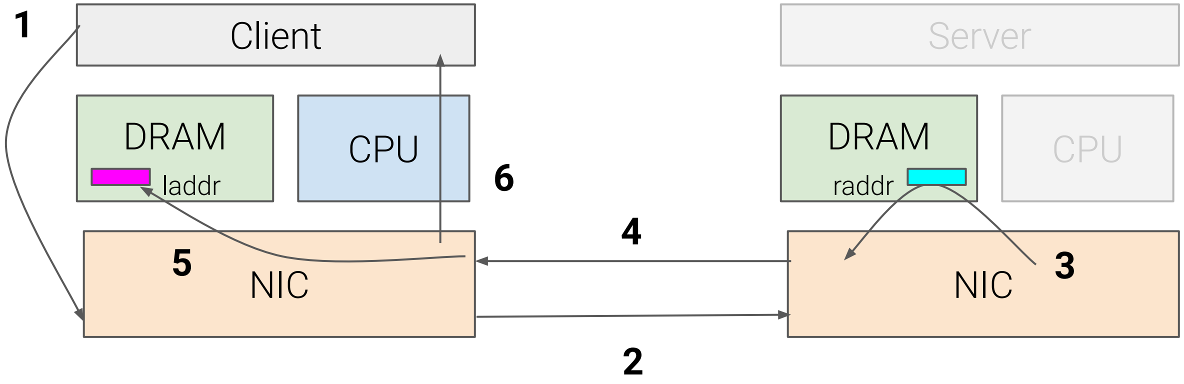
RDMA: One-sided Read Operation

Hey! Your content is stored in the buffer at 'raddr' (+ a tag, called steering tag or **Stag**)



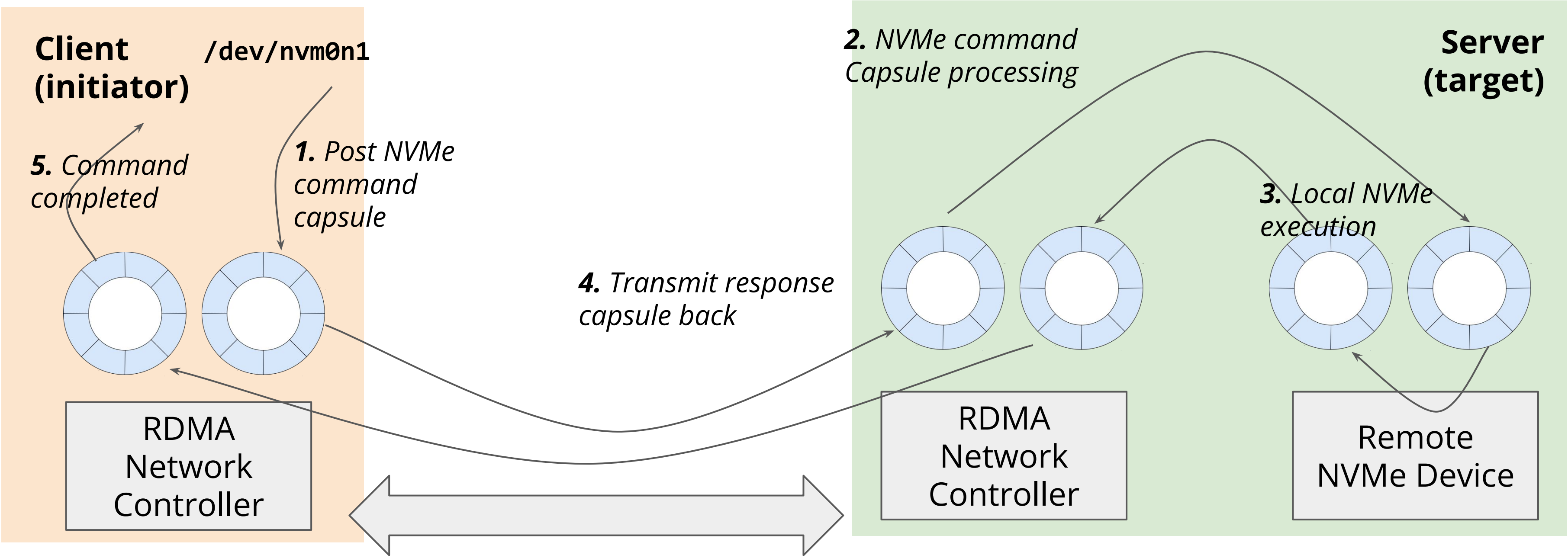
RDMA: One-sided Read Operation

1. Client: READ remote memory address (raddr) to local address (laddr)
2. Client: posts READ request
3. Server: read local (raddr) - local DMA operation
4. Server: TX data back to client NIC
5. Client: local DMA to (laddr) buffer in DRAM
6. Client: interrupt the local CPU/OS to notify completion about the client's READ operation



RDMA operations are like remote "DMA" - defined for specific remote memory locations

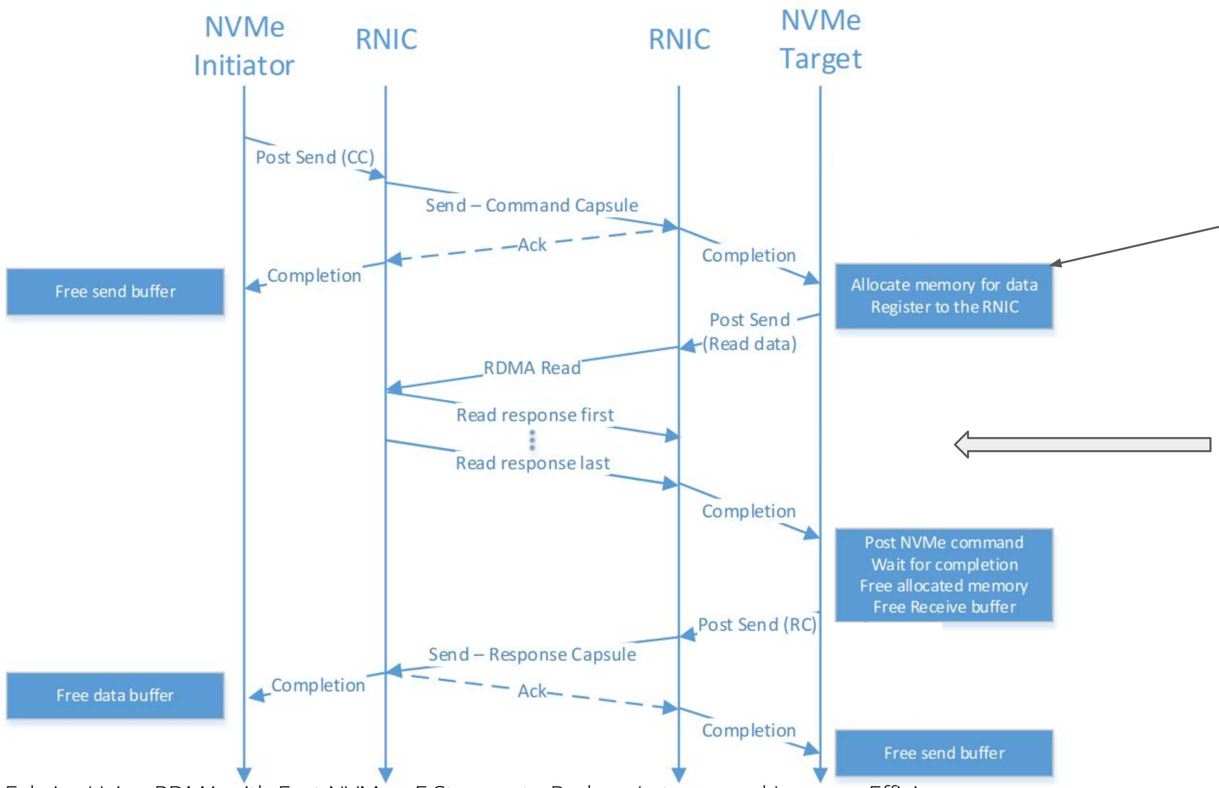
NVMe-oF = RDMA + NVM Express



High-performance Network, 100 Gbps Ethernet

At no point in time we have to use any legacy protocols like SCSI, or socket/TCP network transfers

NVMe-oF Write

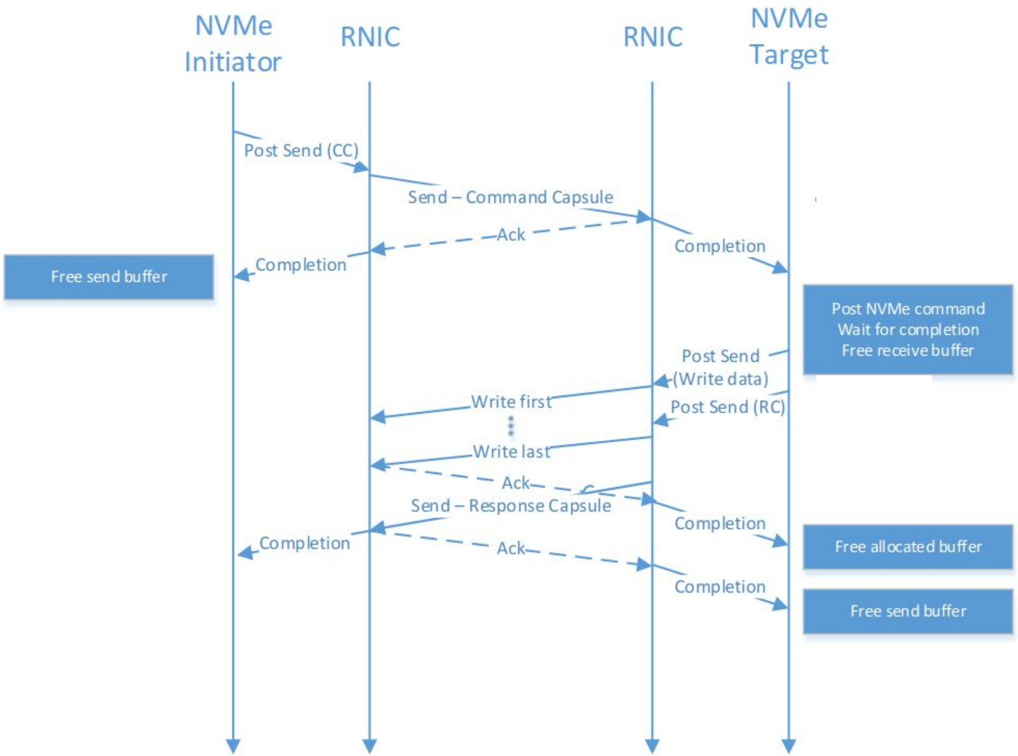


Shared common buffer for network and storage

Use one-sided RDMA operations

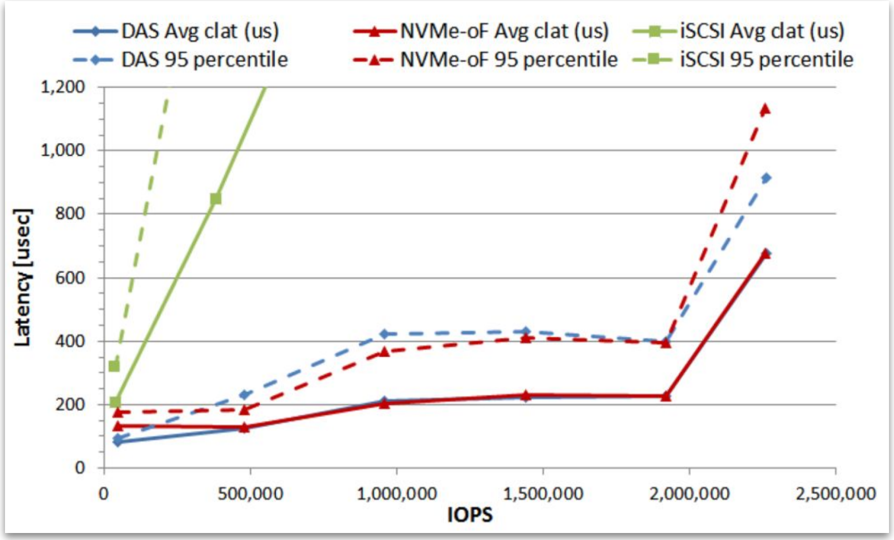
Ethernet Storage Fabrics Using RDMA with Fast NVMe-oF Storage to Reduce Latency and Improve Efficiency, https://www.snia.org/educational-library/ethernet-storage-fabrics-using-rdma-fast-nvme-storage-reduce-latency-and-improve

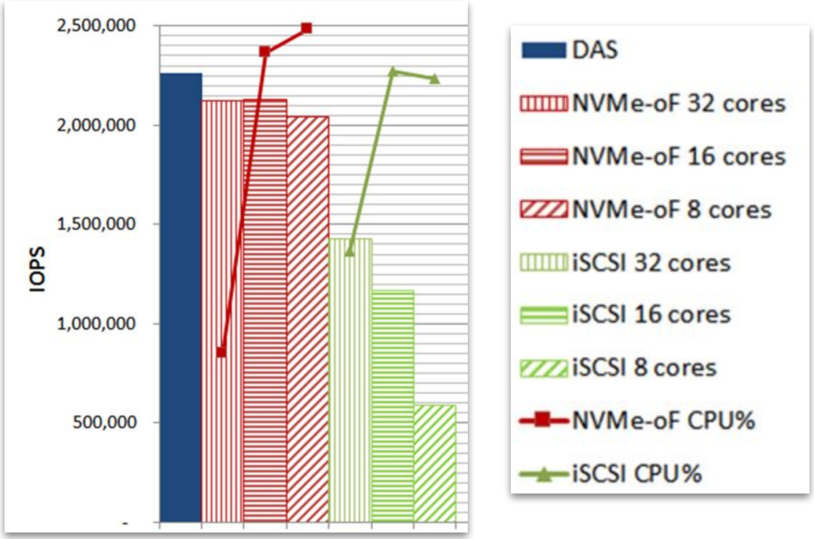
NVMe-oF Read



Ethernet Storage Fabrics Using RDMA with Fast NVMe-oF Storage to Reduce Latency and Improve Efficiency, https://www.snia.org/educational-library/ethernet-storage-fabrics-using-rdma-fast-nvme-storage-reduce-latency-and-improve

NVMe-oF Performance

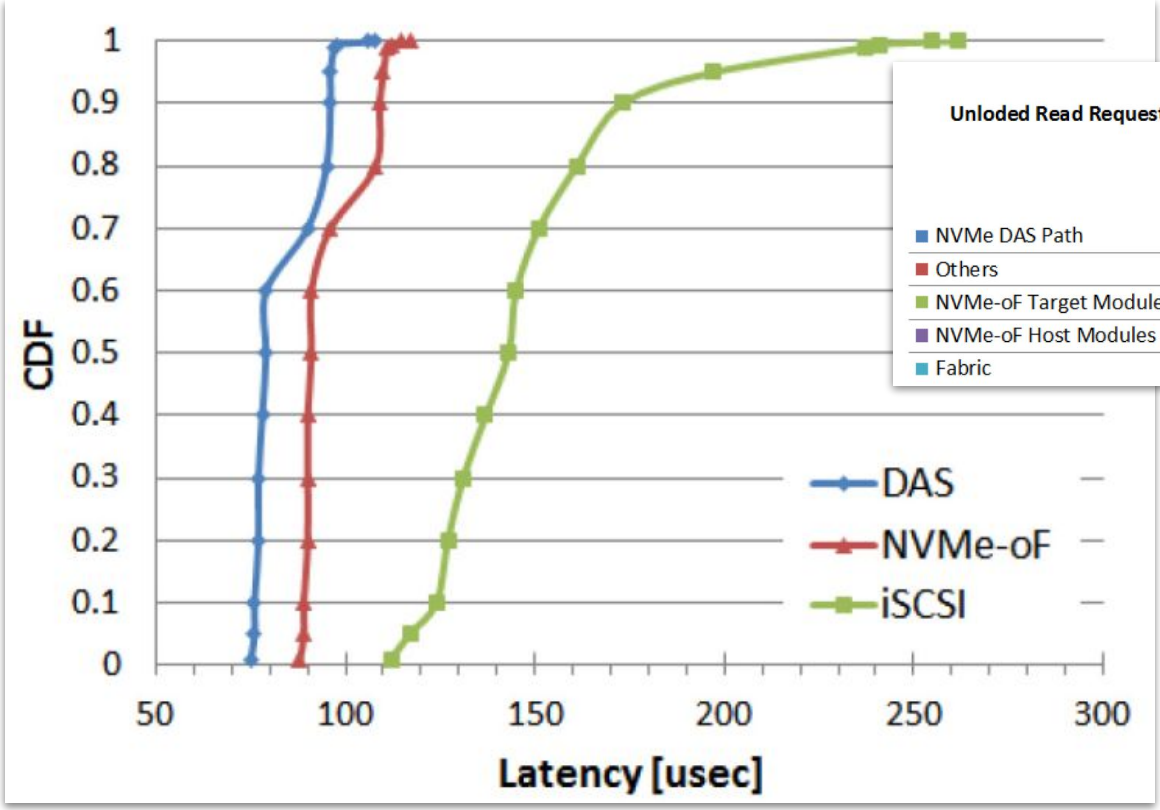


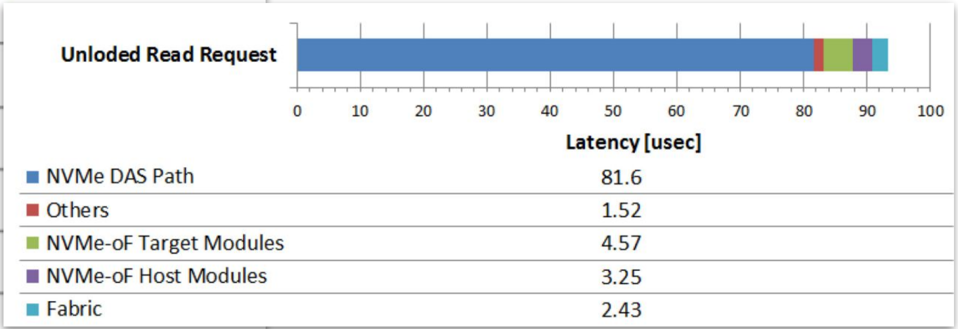


In comparison to iSCSI, NVMoF provides performance very close to a locally attached storage

Zvika Guz, Harry (Huan) Li, Anahita Shayesteh, and Vijay Balakrishnan. 2018. Performance Characterization of NVMe-over-Fabrics Storage Disaggregation. ACM Trans. Storage 14, 4, Article 31 (December 2018), 18 pages.

NVMe-oF Latency Performance





Latency is dominated by the NVMe directly attached storage path on the target side

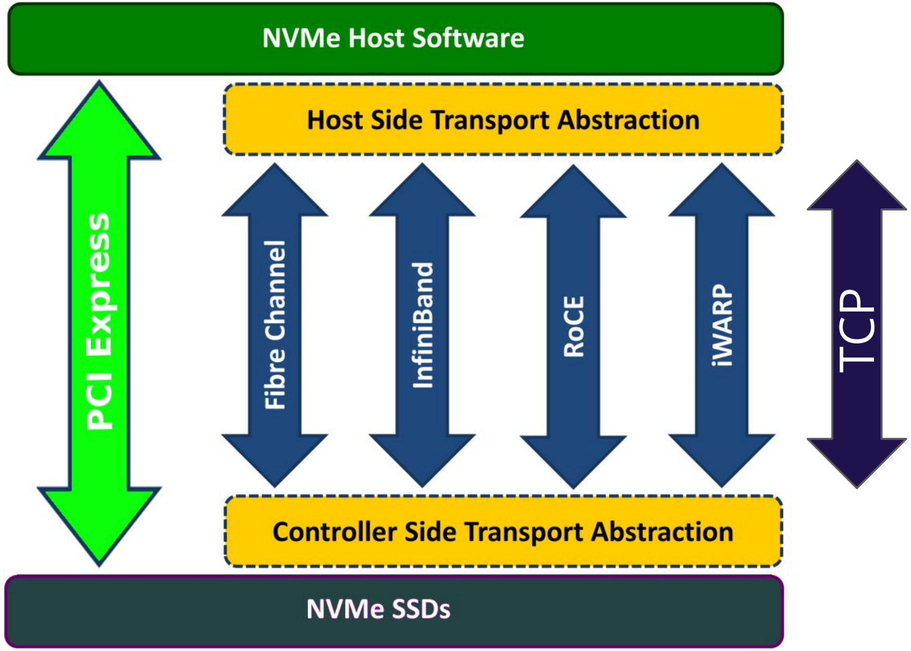
NVM over Fabrics

Is the dominant and standard way to deploy networked flash

Supports various high-performance Networks like RDMA

- New specification on TCP/socket is now also available (not offloaded)

Is constantly being updated to accommodate new changes



Thesis (available): Understanding and optimizing NVMoF/TCP (+scheduling QoS)

https://www.dolphinics.com/solutions/nvme_over_pcie_fabrics.html and https://nvmexpress.org/welcome-nvme-tcp-to-the-nvme-of-family-of-transports/

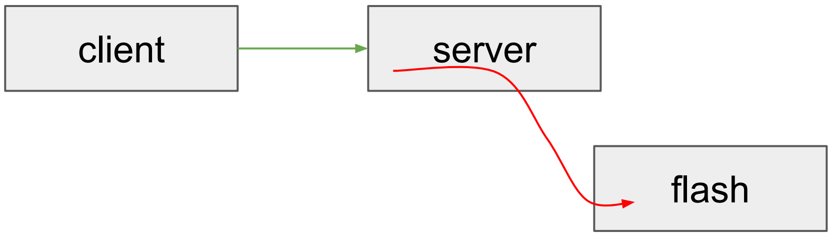
Thinking outside the Box

NVMe-oF is equivalent to iSCSI (hence, a SAN solution)

RDMA allows to read/write remote memories directly

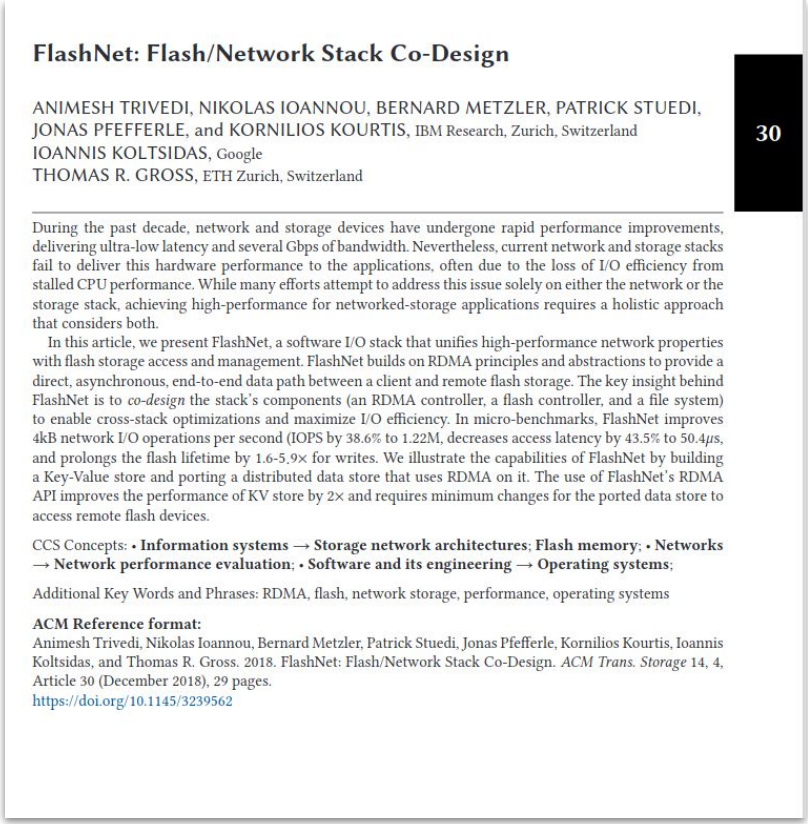
Quite popular inside data center due to its performance to build

- Key-value stores and caches
- Transaction systems
- File systems
- Distributed data structures
- Consensus and ordering



Can we think of extending RDMA operations to directly access Flash location?

FlashNet: Building a Unified Network/Storage Stack (2018)



FlashNet: Flash/Network Stack Co-Design

30

ANIMESH TRIVEDI, NIKOLAS IOANNOU, BERNARD METZLER, PATRICK STUEDI, JONAS PFEFFERLE, and KORNILIOS KOURTIS, IBM Research, Zurich, Switzerland
IOANNIS KOLTSIDAS, Google
THOMAS R. GROSS, ETH Zurich, Switzerland

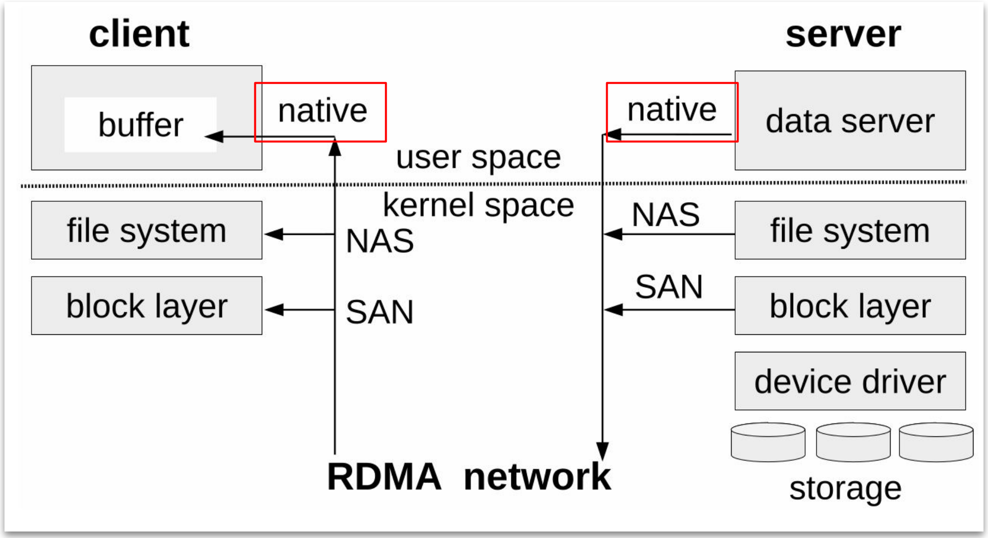
During the past decade, network and storage devices have undergone rapid performance improvements, delivering ultra-low latency and several Gbps of bandwidth. Nevertheless, current network and storage stacks fail to deliver this hardware performance to the applications, often due to the loss of I/O efficiency from stalled CPU performance. While many efforts attempt to address this issue solely on either the network or the storage stack, achieving high-performance for networked-storage applications requires a holistic approach that considers both.

In this article, we present FlashNet, a software I/O stack that unifies high-performance network properties with flash storage access and management. FlashNet builds on RDMA principles and abstractions to provide a direct, asynchronous, end-to-end data path between a client and remote flash storage. The key insight behind FlashNet is to *co-design* the stack's components (an RDMA controller, a flash controller, and a file system) to enable cross-stack optimizations and maximize I/O efficiency. In micro-benchmarks, FlashNet improves 4kB network I/O operations per second (IOPS by 38.6% to 1.22M, decreases access latency by 43.5% to 50.4μs, and prolongs the flash lifetime by 1.6-5.9× for writes. We illustrate the capabilities of FlashNet by building a Key-Value store and porting a distributed data store that uses RDMA on it. The use of FlashNet's RDMA API improves the performance of KV store by 2× and requires minimum changes for the ported data store to access remote flash devices.

CCS Concepts: • **Information systems → Storage network architectures**; **Flash memory**; • **Networks → Network performance evaluation**; • **Software and its engineering → Operating systems**;

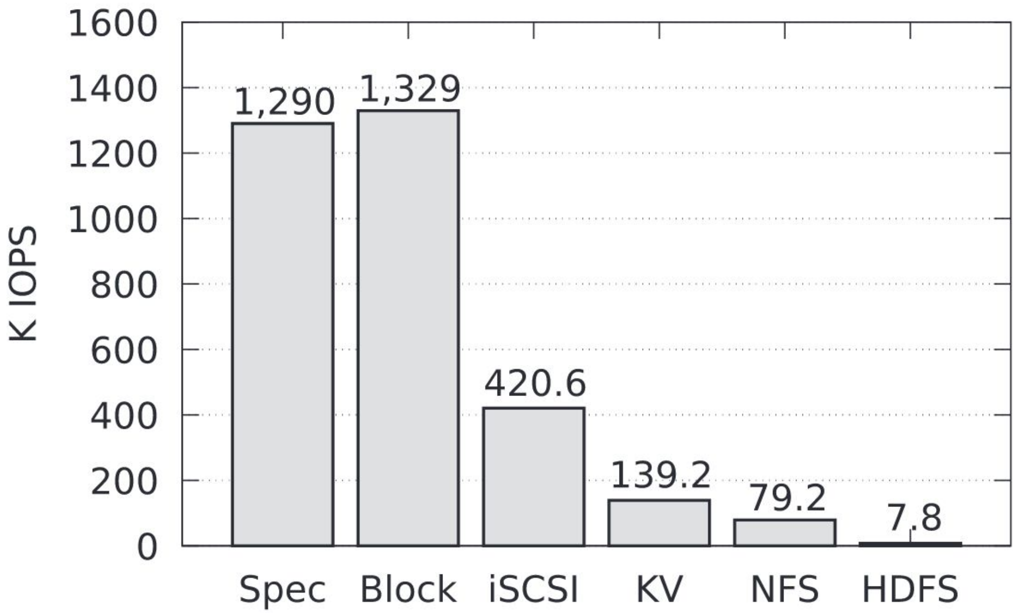
Additional Key Words and Phrases: RDMA, flash, network storage, performance, operating systems

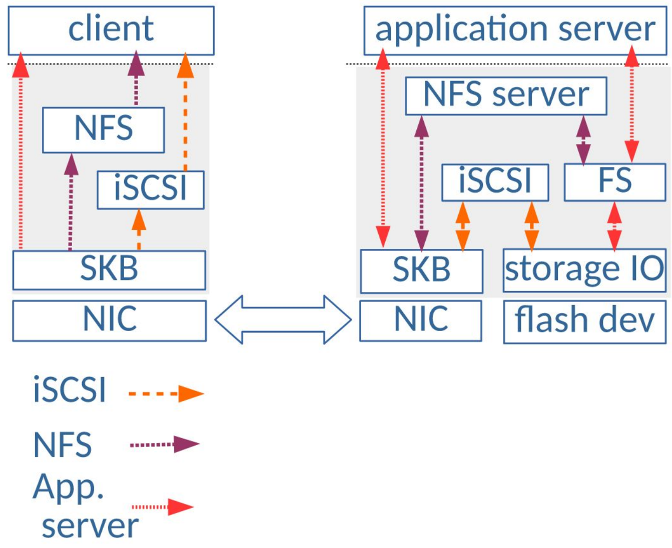
ACM Reference format:
Animesh Trivedi, Nikolas Ioannou, Bernard Metzler, Patrick Stuedi, Jonas Pfefferle, Kornilios Kourtis, Ioannis Koltsidas, and Thomas R. Gross. 2018. FlashNet: Flash/Network Stack Co-Design. *ACM Trans. Storage* 14, 4, Article 30 (December 2018), 29 pages.
https://doi.org/10.1145/3239562



PhD Thesis, A. Trivedi, End-to-End Considerations in the Unification of High-Performance I/O, https://doi.org/10.3929/ethz-a-010651949

Number of Steps to Access Remote Data

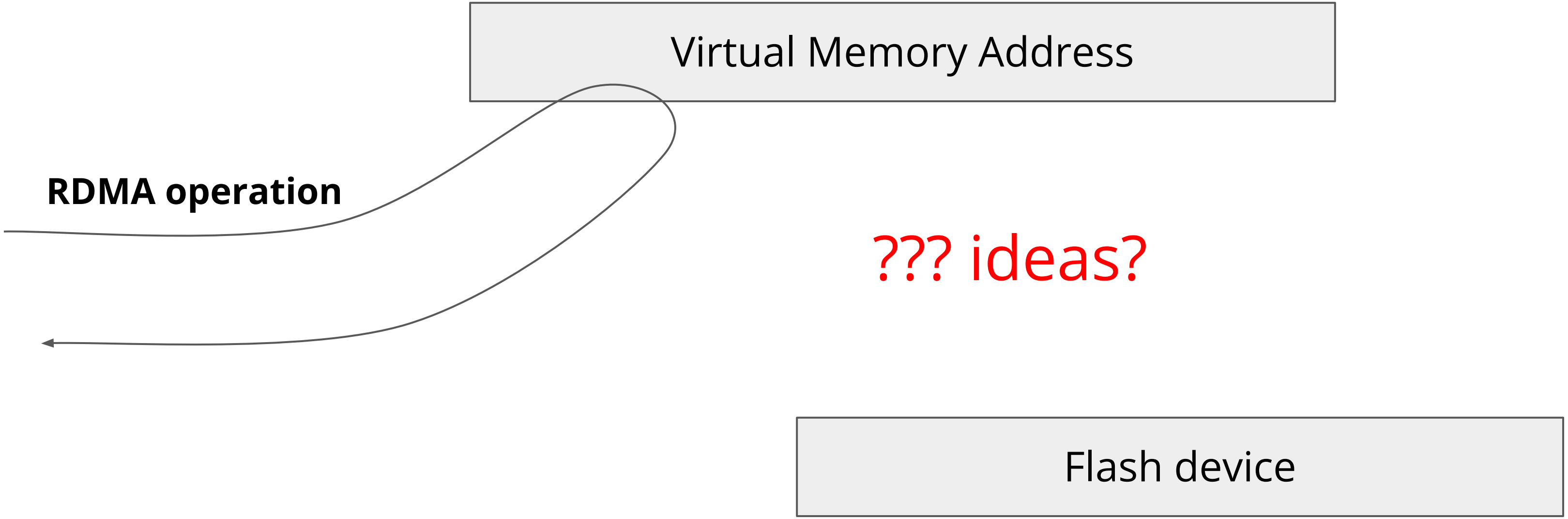




Going over block protocols (iSCSI), application (KV), file system (NFS), or cloud-FS (HDFS) costs performance (mix of network and storage overheads) → can we do something better?

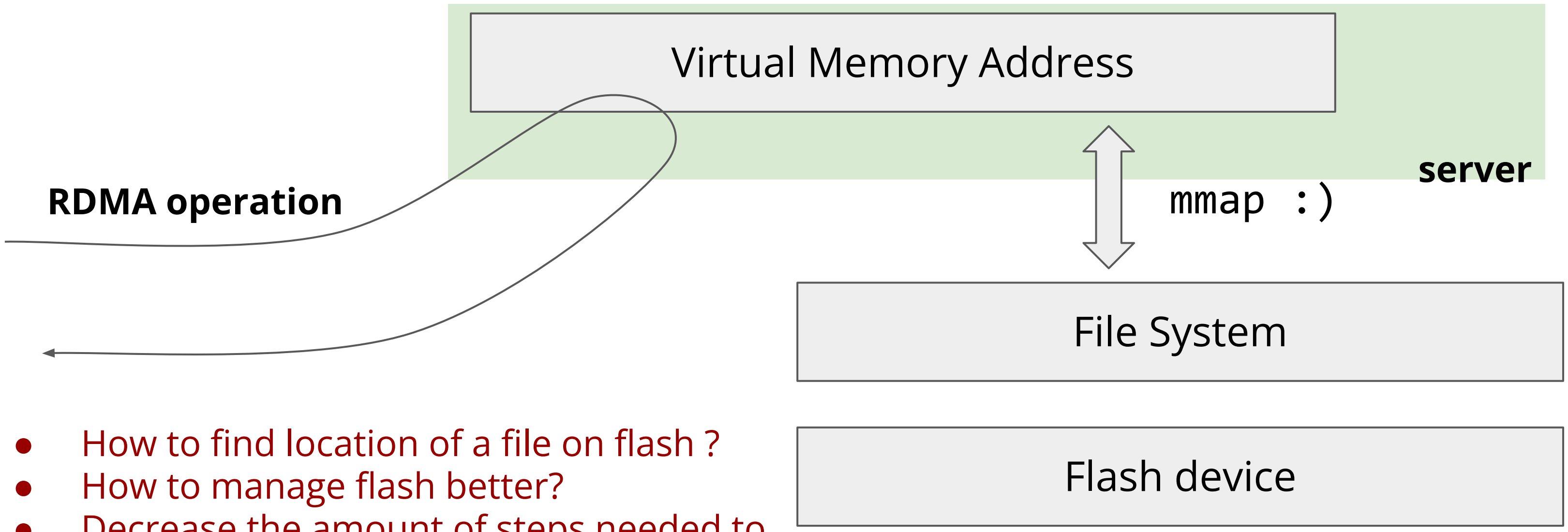
FlashNet: Basic Challenge

RDMA operations are defined for a memory location, how do we get a memory location for a flash?



FlashNet: Basic Challenge

RDMA operations are defined for a memory location, how do we get a memory location for a flash?

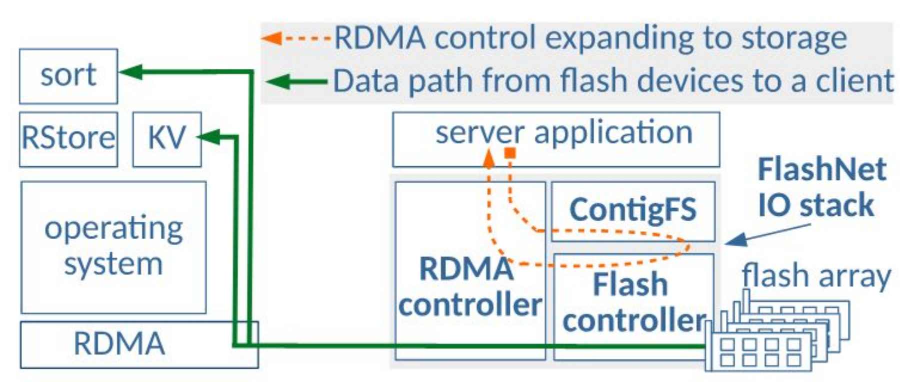


- How to find location of a file on flash ?
- How to manage flash better?
- Decrease the amount of steps needed to lookup things

FlashNet Stack

<u>Co-development</u> of a software:

1. Flash Controller
2. File system
3. RDMA Controller



RDMA controllers helps to on-demand fetch pages from the file system

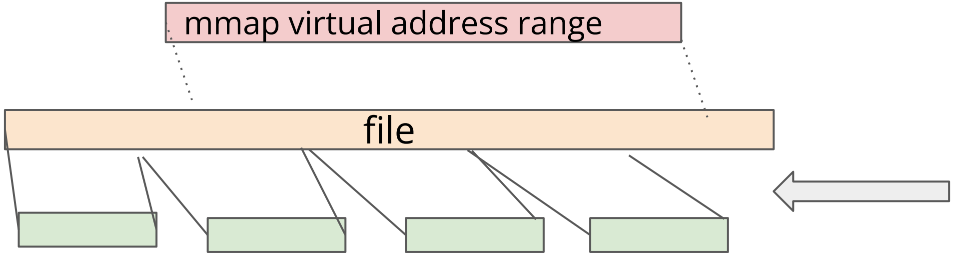
File system is like DFS, hence, large contiguous files (easy offset calculation)

Flash controller manages devices and uses RDMA access statistics for flash device management and page sharing between I/O operations

Put together they help to translate quickly between a network request and flash address

Abstraction Translation

RDMA identifies a memory location using a tag



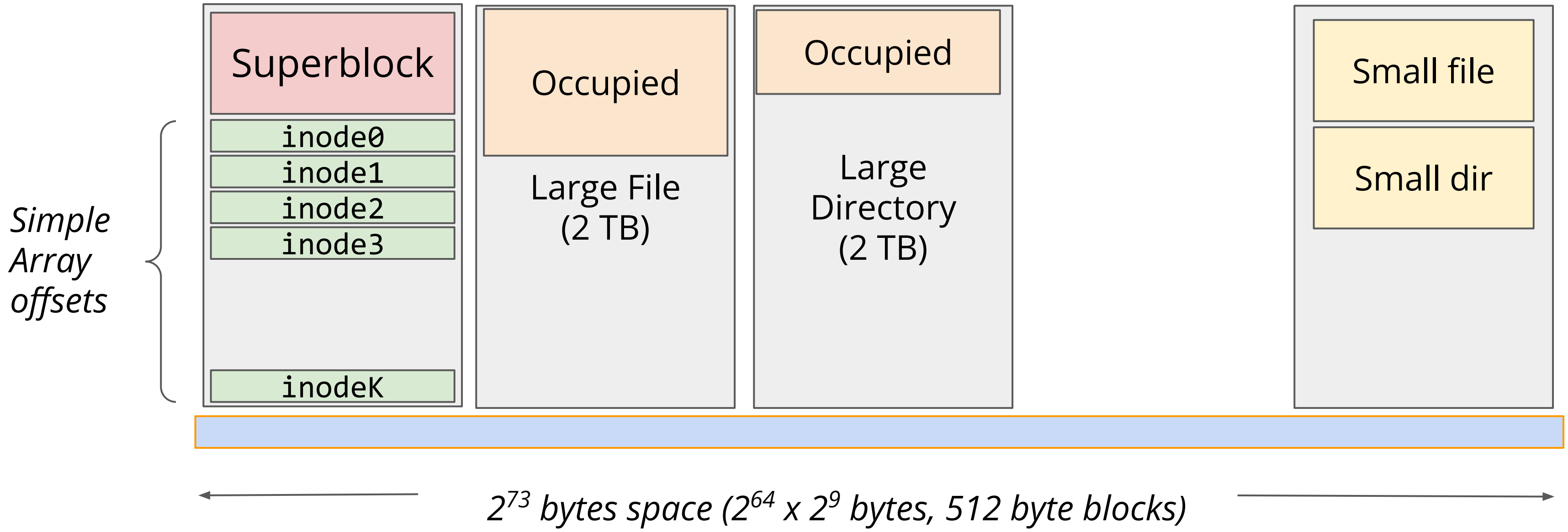
Flash logical address locations (FS does this translation)

This file offset to a local on flash LBA is done by the file system (ext4, F2FS)

So for any random file offset you need to ask from the file system where is the data stored

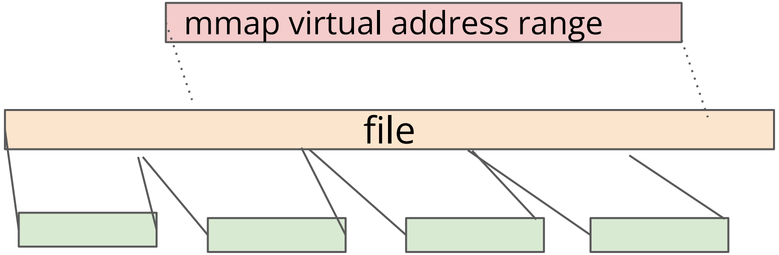
ContigFS

- Builds on the same idea as DFS (lecture 4) on virtualized flash devices
- All files are contiguously allocated (logically)

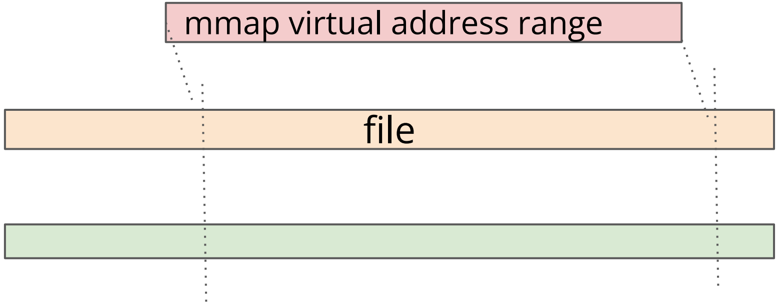


Abstraction Translation

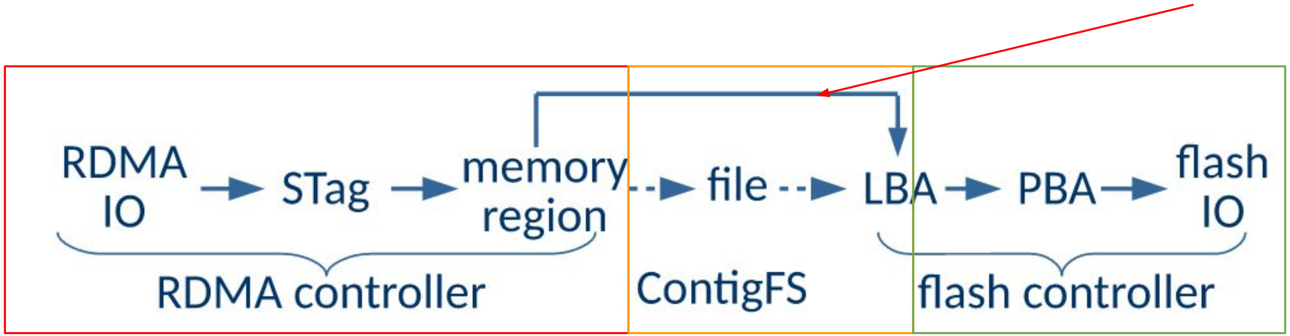
RDMA identifies a memory location using a tag



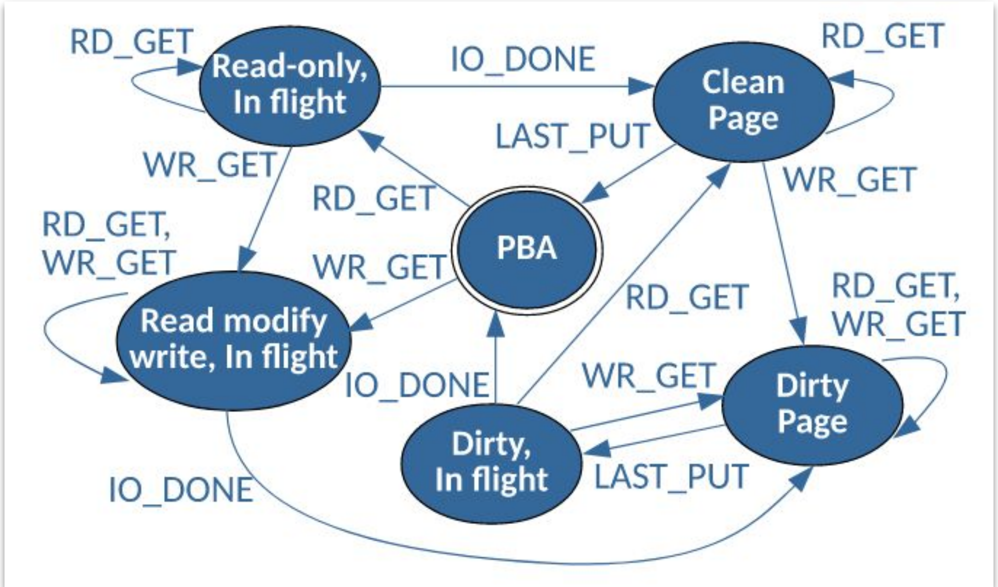
Flash logical address locations (FS does this translation)



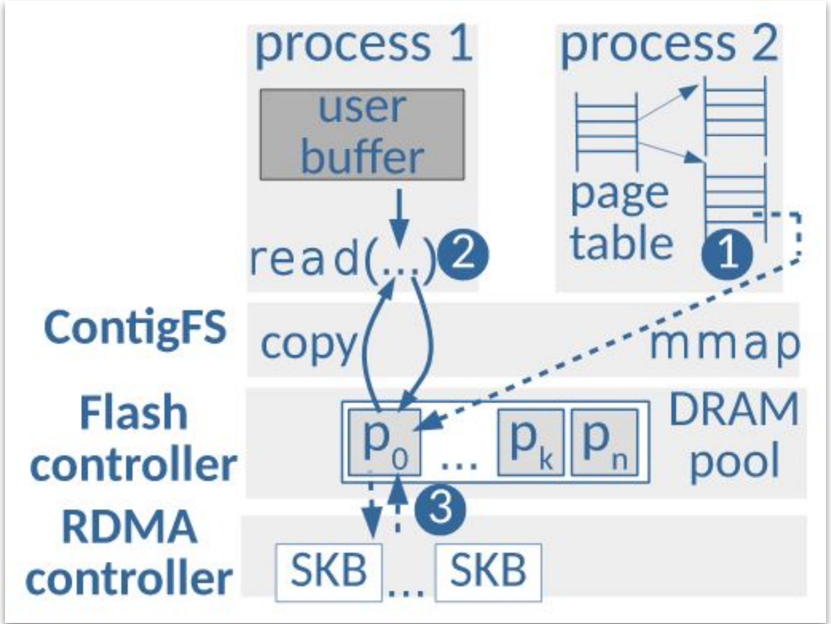
With FlashNet, simple translation



Flash Page Management

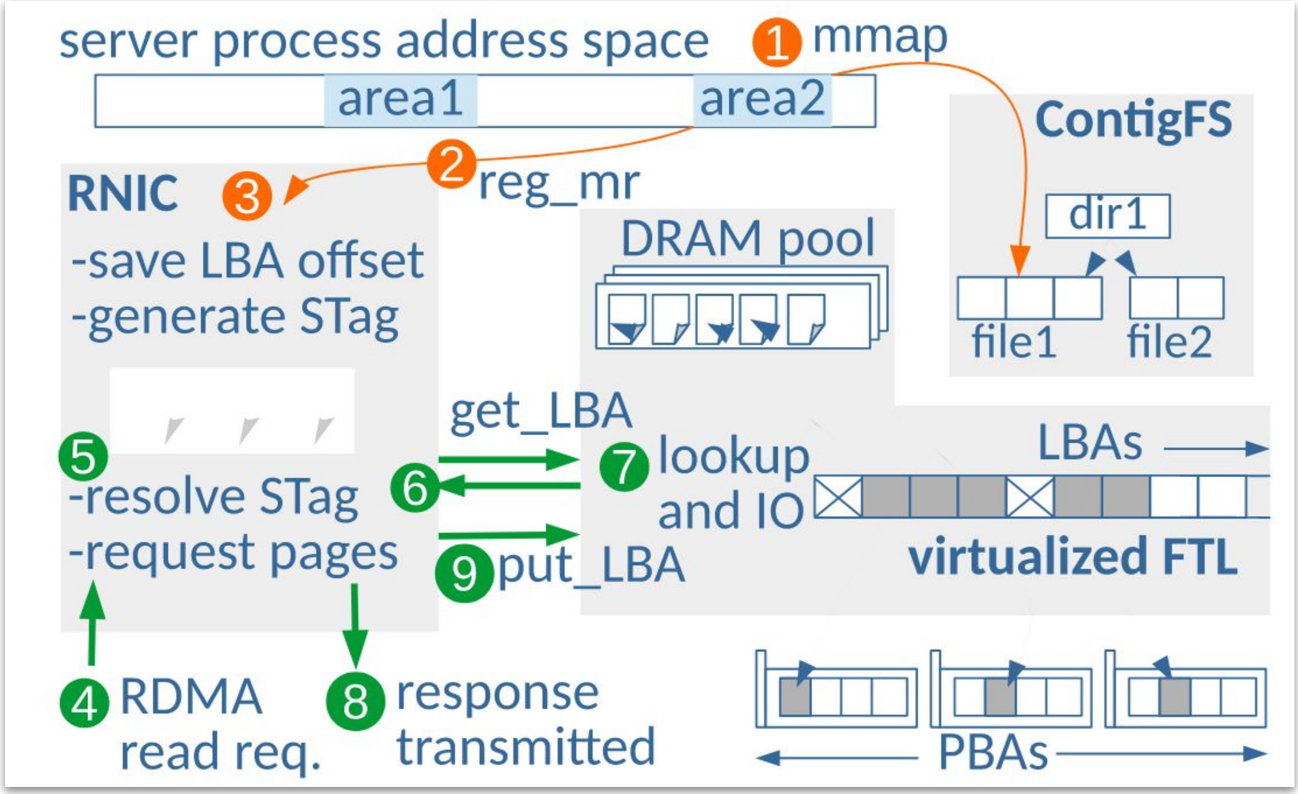


Flash pages in the host go through this state machine when in use



A simple shared DRAM page pool where all I/O happens

A Complete Operation

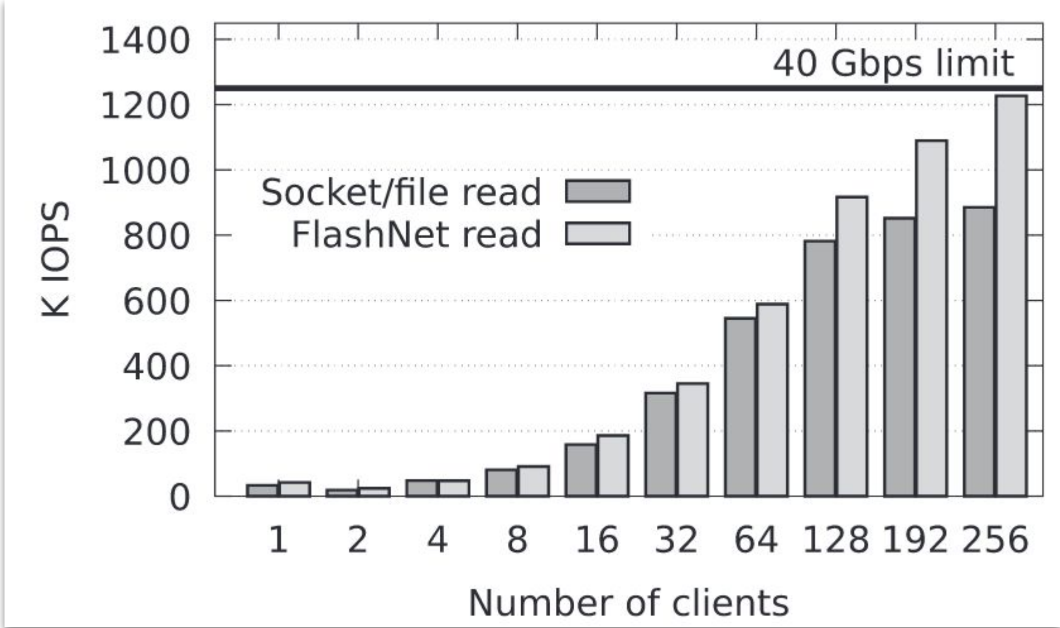


- DRAM DMA pool is shared between
  - RDMA
  - Mmap
  - Local read/write

- get/put LBAs counters help with identifying hot and cold flash pages

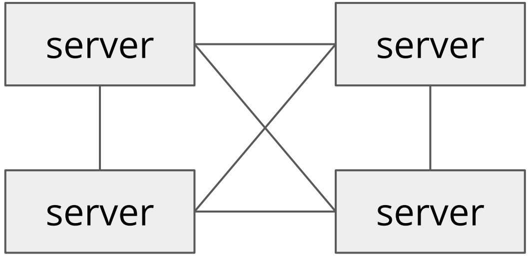
- An RDMA controller can easily do an offset calculation from a virtual address to a flash LBA address (hence, no need to involved the file system)

Application-Driven Remote Storage Access



	network	storage	I/O drivers	scheduling	kernel	app-logic	misc.
Socket/file	19.3%	7.3%	6.7%	15.8%	40.1%	4.7%	6.1%
FlashNet	20.6%	0.8%	6.4%	8.4%	46.7%	11.7%	5.4%

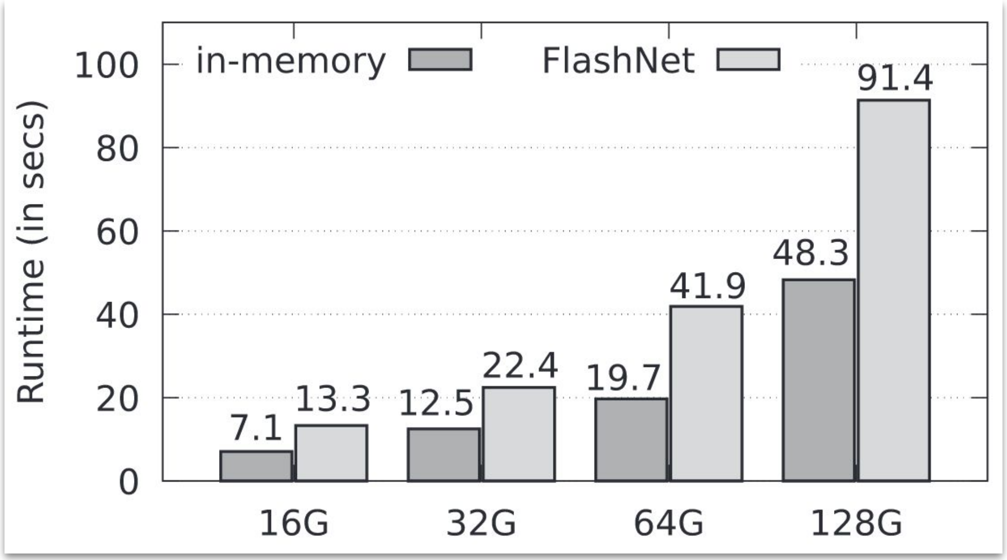
Application Performance



Doing a distributed sorting over 4 machines

In-memory all data is stored in memory, all network traffic is using RDMA

FlashNet, all data is in Flash, and accessed using the **"same" RDMA** operations



The performance gap is purely from flash I/O performance

What you should know from this lecture

1. What is Storage Disaggregation and why is it useful
2. What are the options to access data stored on a remote server
    a. Storage Area Network: iSCSI (block)
    b. Network Attached Storage: NFS (files)
    c. Object/Key-Value stores : like S3, redis (application-driven protocols)
3. What is NVMe-oF and how does it relate to RDMA networking
4. Why was NVMe-oF invented
5. What is FlashNet and what does it tries to optimize
6. How does FlashNet (an application-level RDMA operation) related to NVMe-oF (a block-level protocol)

[Further reading]
SIGCOMM 2022 (Aug, 2022)

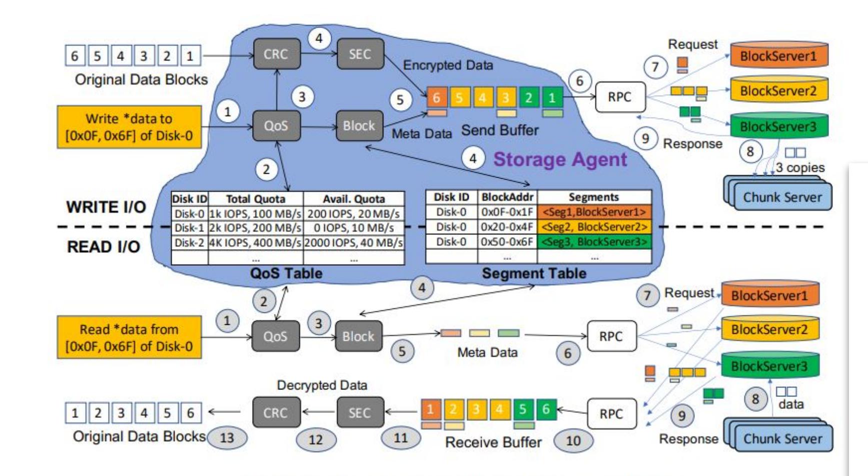


Figure 2: The internal structure and workflow of storage agent (SA).

From Luna to Solar: The Evolutions of the Compute-to-Storage Networks in Alibaba Cloud

Rui Miao*, Lingjun Zhu*, Shu Ma, Kun Qian, Shujun Zhuang, Bo Li, Shuguang Cheng, Jiaqi Gao, Yan Zhuang, Pengcheng Zhang, Rong Liu, Chao Shi, Binzhang Fu, Jiaji Zhu, Jiesheng Wu, Dennis Cai, Hongqiang Harry Liu

Alibaba Group

*Co-first authors

ABSTRACT

This paper presents the two generations of storage network stacks that reduced the average I/O latency of Alibaba Cloud's EBS service by 72% in the last five years: LUNA, a user-space TCP stack that corresponds the latency of network to the speed of SSD; and SOLAR, a storage-oriented UDP stack that enables both storage and network hardware accelerations.

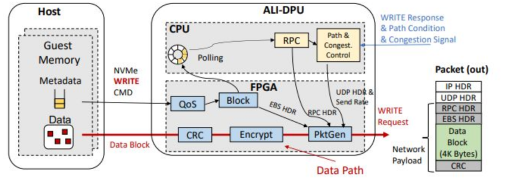
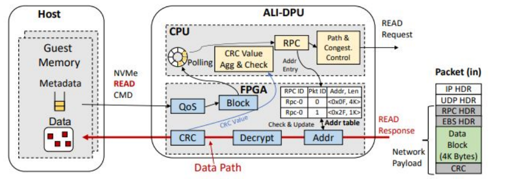


Figure 12: The workflow of a WRITE request in SOLAR.



perfectly applies to commodity DPUs (data processing units).

CCS CONCEPTS

• **Networks** → **Network protocol design**; • **Information systems** → **Cloud based storage**;

KEYWORDS

Storage Network; In-network Acceleration; Data Processing Unit

ACM Reference Format:

Rui Miao*, Lingjun Zhu*, Shu Ma, Kun Qian, Shujun Zhuang, Bo Li, Shuguang Cheng, Jiaqi Gao, Yan Zhuang, Pengcheng Zhang, Rong Liu, Chao Shi, Binzhang Fu, Jiaji Zhu, Jiesheng Wu, Dennis Cai, Hongqiang Harry Liu. 2022. From Luna to Solar: The Evolutions of the Compute-to-Storage Networks in Alibaba Cloud. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3544216.3544238

1 INTRODUCTION

Elastic Block Storage (EBS) is a fundamental service that provides persistent data hosting in virtualized disks (VDs) to cloud users [4, 5, 7, 8]. It has to be highly reliable (*e.g.*, "nine 9s" for data integrity [4]) and fast (*e.g.*, sub-millisecond for I/O latency), given that the VDs directly interact with the cloud users' operating systems in real-time. As the "compute-storage separation" or "storage disaggregation" architecture of EBS has been widely adopted by mainstream cloud providers, the network that interconnects the compute and storage clusters turns into an essential bottleneck of EBS' overall performance.

Nonetheless, it does not mean that the network solution with the best performance is always suitable to EBS because there are multiple dimensions of requirements on a storage network. For example, storage networks should also support a massive number of connections, long network distances, various types of hardware configurations, be compatible with the computation architecture, and limit the cost for cloud providers. Therefore, designing a storage network is highly challenging.

This paper presents the motivations, challenges, design choices, deployment experiences, and lessons of two significant upgrades on the EBS network of Alibaba Cloud ("AliCloud" for short) in recent five years: LUNA and SOLAR.

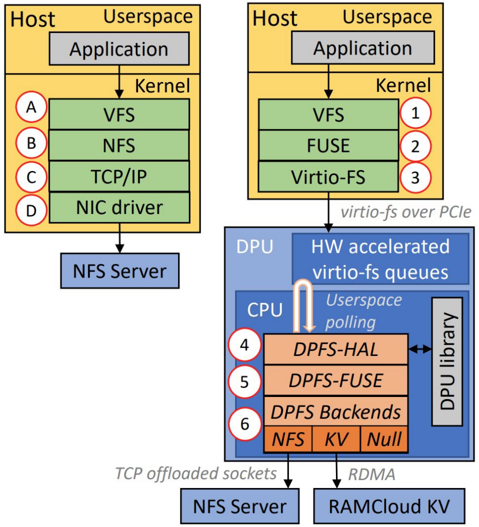
LUNA was designed to replace the kernel TCP stack for coor-

https://dl.acm.org/doi/abs/10.1145/3544216.3544238

[Further reading] File System Virtualization

	ext4	ext4 + NVMe-oF	XFS	Btrfs
I/O operations	5.2	13.7	3	4.6
Total Bytes (in KiB)	44.7	46.8	12	125.3
Amplification	11.2x	11.7x	3x	16x

Table 1: Analysis of storage (block) or network (packets with NVMoF) operations for a single 4KiB file write.



DPFS: DPU-Powered File System Virtualization

Peter-Jan Gootzen*
IBM Research
Zurich, Switzerland
peter.jan.gootzen@ibm.com

Jonas Pfefferle
IBM Research
Yorktown Heights, USA
jpf@ibm.com

Radu Stoica
IBM Research
Zurich, Switzerland
rst@zurich.ibm.com

Animesh Trivedi
VU Amsterdam
Amsterdam, Netherlands
a.trivedi@vu.nl

ABSTRACT

As we move towards hyper-converged cloud solutions, the efficiency and overheads of distributed file systems at the cloud tenant side (i.e., client) become of paramount importance. Often, the client-side driver of a cloud file system is complex and CPU intensive, deeply coupled with the backend implementation, and requires optimizing multiple intrusive knobs. In this work, we propose to decouple the file system client from its backend implementation by virtualizing it with an off-the-shelf DPU using the Linux `virtio-fs` software stack. The decoupling allows us to offload the file system client execution to a DPU, which is managed and optimized by the cloud provider, while freeing the host CPU cycles. DPFS, our proposed framework, is 4.4× more host CPU efficient per I/O, delivers comparable performance to a tenant with zero-configuration and without modification of their host software stack, while allowing workload and hardware specific backend optimizations. The DPFS framework and its artifacts are publically available at https://github.com/IBM/DPFS.

CCS CONCEPTS

• **Networks** → Network File System (NFS) protocol; • **Software and its engineering** → **File systems management**; • **Information systems** → *Cloud based storage*; • **Hardware** → *Networking hardware.*

KEYWORDS

DPU, SmartNIC, Offloading, File system, Virtualization, Cloud, Storage, Framework, Datacenter, RDMA, NFS, Virtio-fs, FUSE

1 INTRODUCTION

File systems are a popular choice for cloud data storage with offerings such as traditional distributed file systems (HadoopFS, Ceph, GlusterFS), and *cloud-native file systems* (CNFS) services like Amazon EFS [3], AliBaba Pangu [10] or Azure Files [27]. With the recent push for hyper-converged infrastructure [13], there is a need for an efficient, scalable and high-performance cloud-native file system service.

Building a high-performance, scalable cloud-native file system for applications is a challenging task. First, the raw performance of storage and networking devices are constantly increasing, while the

*Also with VU Amsterdam.

SYSTOR '23, June 5–7, 2023, Haifa, Israel
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9962-3/23/06.
https://doi.org/10.1145/3579370.3594769

	ext4	ext4 + NVMe-oF	XFS	Btrfs
I/O operations	5.2	13.7	3	4.6
Total Bytes (in KiB)	44.7	46.8	12	125.3
Amplification	11.2x	11.7x	3x	16x

Table 1: Analysis of storage (block) or network (packets with NVMoF) operations for a single 4KiB file write.

CPU performance improvements have stalled [29, 44]. As a result, delivering the full speed of I/O devices in a disaggregated storage setting takes a considerable amount of CPU resources [17, 43]. For example, Alibaba reports 12 CPU cores are required to deliver 200 Gbps of block-level traffic [26]. At the file system level, LineFS reports that with Ceph a single fully-utilized CPU only delivers ~10 Gbps bandwidth on a 100 Gbps link [16]. The question of CPU efficiency is also important for bare-metal machines, which have become popular in clouds recently [6, 34, 48]. Second, client-side CNFS logic can be complex and bloated, as it has to implement logic for communication and coordination with metadata and data servers, client-side buffer and connection management, caches, etc. As a result, it is not uncommon for distributed file system clients to consume GBs of DRAM and a significant amount of CPU cycles, thus limiting how many concurrent tenants (VMs, containers) can be packed on a server [2, 21]. Lastly, the close coupling of the file system API and its implementation makes it difficult to deploy new extensions or optimizations. For example, a bare-metal tenant using Ceph can not easily switch to HopsFS [30] or InfiniFS [25] without significant disruptions if it experiences metadata scalability challenges. Furthermore, many of these CNFS come with hundreds of performance knobs and features, which requires explicit deployment and optimizations from the tenant side to extract the best possible performance.

To address the aforementioned challenges, we propose to *virtualize the access to a file system by offloading the file system client to a DPU* to offer a tenant-transparent, light-weight, high-performance file system service. Such a design has multiple advantages: First, virtualization decouples the file system API from its backend implementation, which enables us to optimize the backends to support multiple workload needs such as multiple APIs [22], scalable metadata lookups with KV stores, decoupling of data from metadata management [19]. A limited form of such decoupling is currently offered by cloud providers in the form of an NFS gateway to the CNFS client [3, 12, 27]. We argue this approach gives away control of the file system client from the cloud provider, and demonstrate that the Linux kernel NFS client has high overheads (§3.4). Second, by offloading (and leveraging the hardware acceleration of the DPU) the file system implementation, we free host CPU resources for the tenant. One can argue that offloading capabilities can also be leveraged by the host either at the block, or application level. A block-level offloading allows a fully offloadable I/O stack [20, 28],

Further Reading

- Adrian M. Caulfield and Steven Swanson. 2013. QuickSAN: a storage area network for fast, distributed, solid state disks, in the ACM ISCA '13.
- Ana Klimovic, Christos Kozyrakis, Eno Thereska, Binu John, and Sanjeev Kumar. 2016. Flash storage disaggregation, in the EuroSys 2016.
- Ana Klimovic, Heiner Litz, and Christos Kozyrakis. 2017. ReFlex: Remote Flash ≈ Local Flash. In the ACM ASPLOS 2017.
- Zvika Guz, Harry (Huan) Li, Anahita Shayesteh, and Vijay Balakrishnan. 2018. Performance Characterization of NVMe-over-Fabrics Storage Disaggregation. ACM Trans. Storage 14, 4, Article 31 (December 2018), 18 pages.
- Animesh Trivedi, Nikolas Ioannou, Bernard Metzler, Patrick Stuedi, Jonas Pfefferle, Kornilios Kourtis, Ioannis Koltsidas, and Thomas R. Gross. 2018. FlashNet: Flash/Network Stack Co-Design. ACM Trans. Storage, 2018.
- Jaehyun Hwang, Qizhe Cai, Ao Tang, Rachit Agarwal, TCP ≈ RDMA: CPU-efficient Remote Storage Access with i10. NSDI 2020, pages, 127-140.
- Animesh Trivedi, Patrick Stuedi, Bernard Metzler, Roman Pletka, Blake G. Fitch, and Thomas R. Gross. 2013. Unified high-performance I/O: one stack to rule them all. In ACM HotOS 2013.
- Brendan Cully, Jake Wires, Dutch Meyer, Kevin Jamieson, Keir Fraser, Tim Deegan, Daniel Stodden, Geoffrey Lefebvre, Daniel Ferstay, and Andrew Warfield. 2014. Strata: scalable high-performance storage on virtualized non-volatile memory. In Proceedings of the 12th USENIX FAST, 2014.