ORION: A Distributed File System for Non-Volatile Main Memories and RDMA-Capable Networks

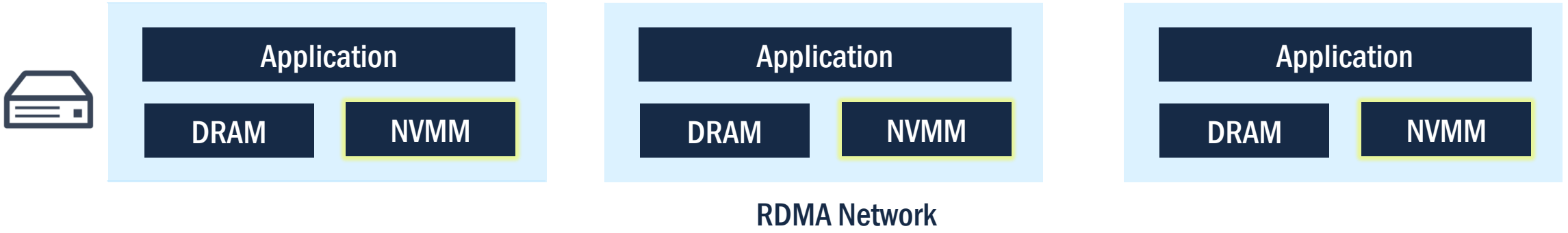
Jian Yang, Joseph Izraelevitz, Steven Swanson

Non-Volatile Systems Laboratory
Department of Computer Science & Engineering
University of California, San Diego

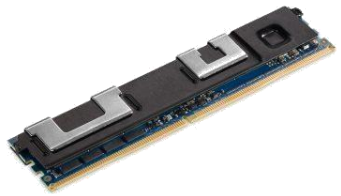
NVSL

UCSDCSE Computer Science and Engineering

Accessing NVMM as Remote Storage

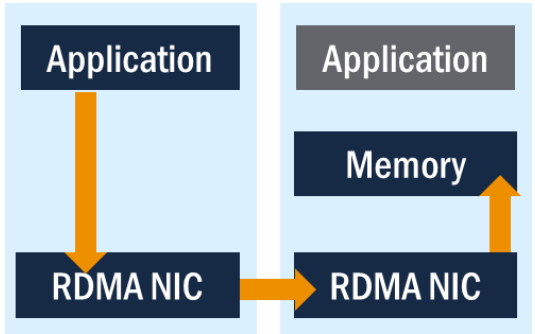


Non-Volatile Main Memory (NVMM)



- PCM, STT-RAM, ReRAM, Intel 3DXPoint
- **Performant:** DRAM-class latency/BW
- **Byte-addressable**
- **Persistent** over power failures

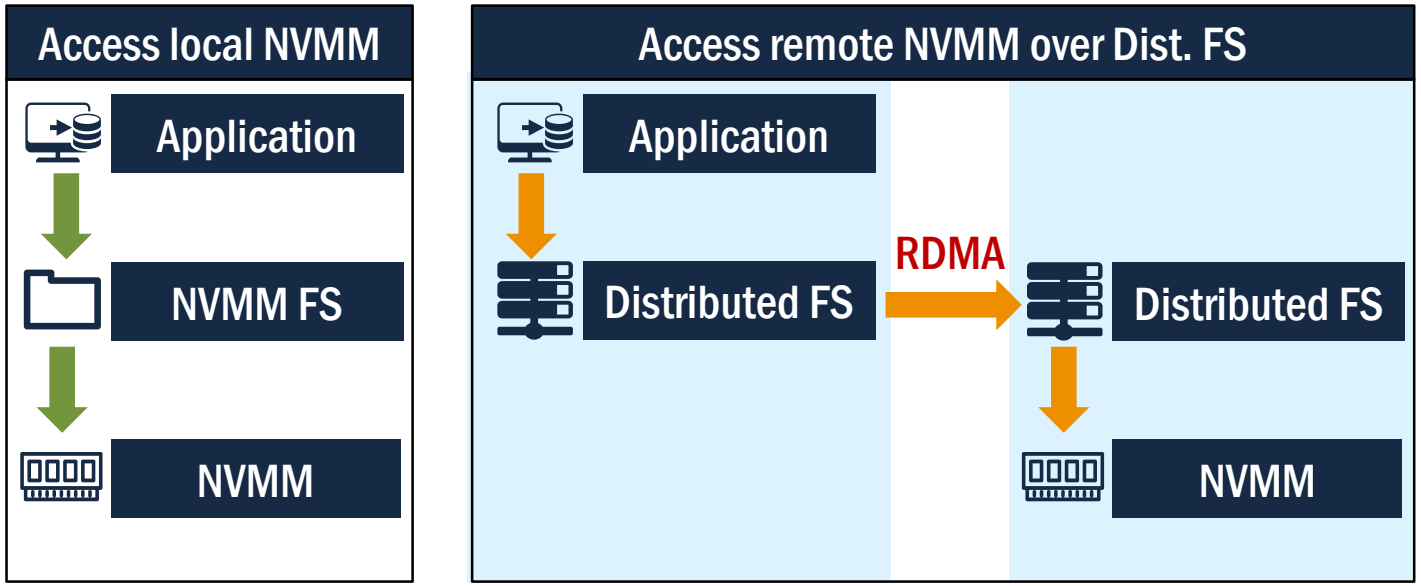
Remote Direct Memory Access (RDMA)

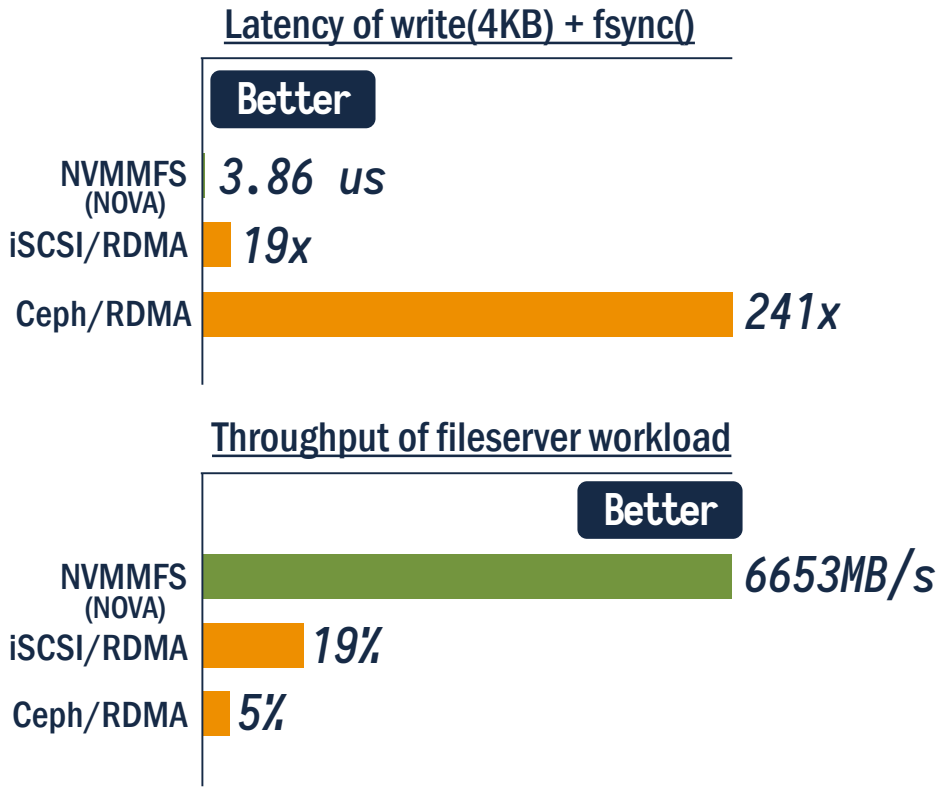


- DMA from/to remote memory
- Two-sided verbs (Send/Recv)
- One-sided verbs (Read/Write)
- Bypasses remote CPU
- Byte-addressable

NVSL

Accessing Local NVMM *vs.* Remote NVMM

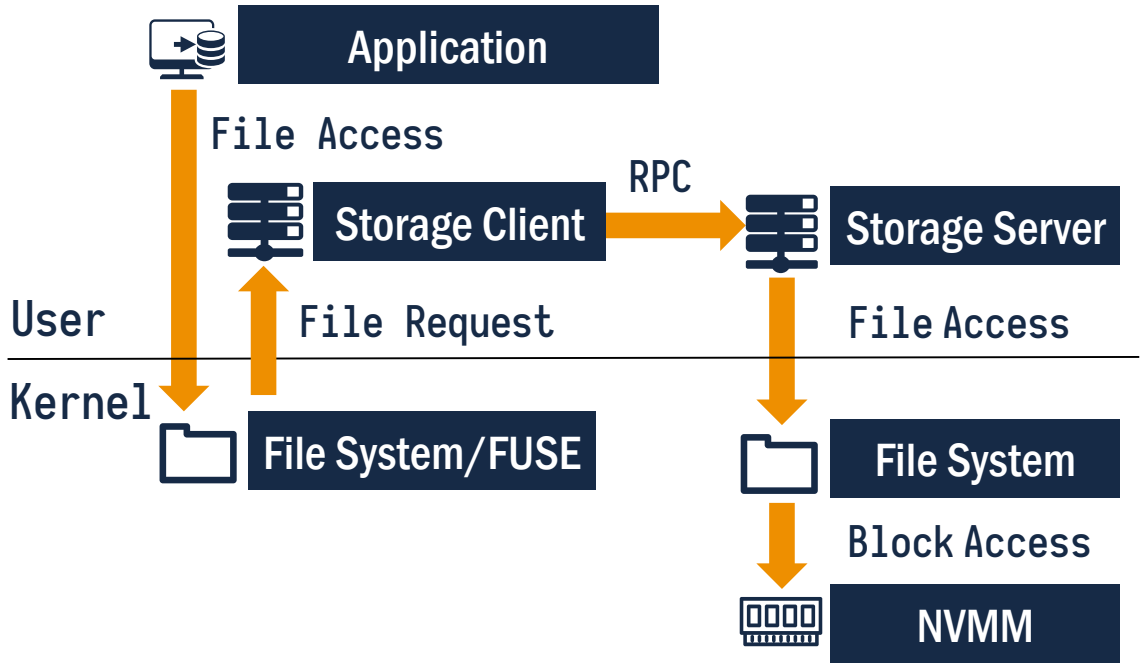




NVSL

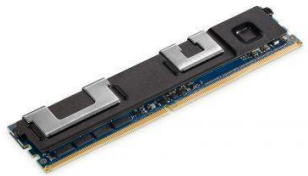
Issue #1: Existing Dist. FSs are Slow on NVMM

- Layered Design
- Indirection overhead
- Expensive to persist (e.g., `fsync()`)



NVSL

NVMM is Faster than RDMA

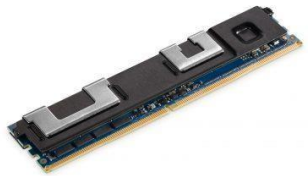


	Harddrive (NVMe)		RDMA Network	NVMM
Latency	70 µs	>>	3 µs	300 ns
Bandwidth	1.3-3.2 GB/s	<	5 GB/s	2-16 GB/s
Access Size (@ Max BW)	4 KB (Page)		2 KB (MTU)	64 Byte (Cacheline)

Networking is faster than storage

NVSL

NVMM is Faster than RDMA



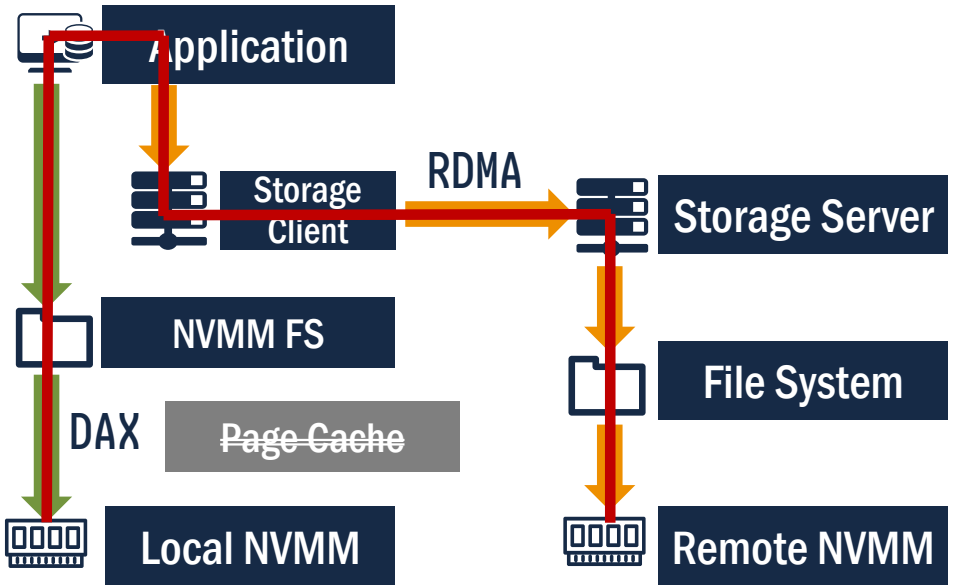
	Harddrive (NVMe)	RDMA Network		NVMM
Latency	70 µs	3 µs	>>	300 ns
Bandwidth	1.3-3.2 GB/s	5 GB/s	<	2-16 GB/s
Access Size (@ Max BW)	4 KB (Page)	2 KB (MTU)		64 Byte (Cacheline)

NVMM is faster than networking

NVSL

Issue #2: Lack of Support for Local NVMM

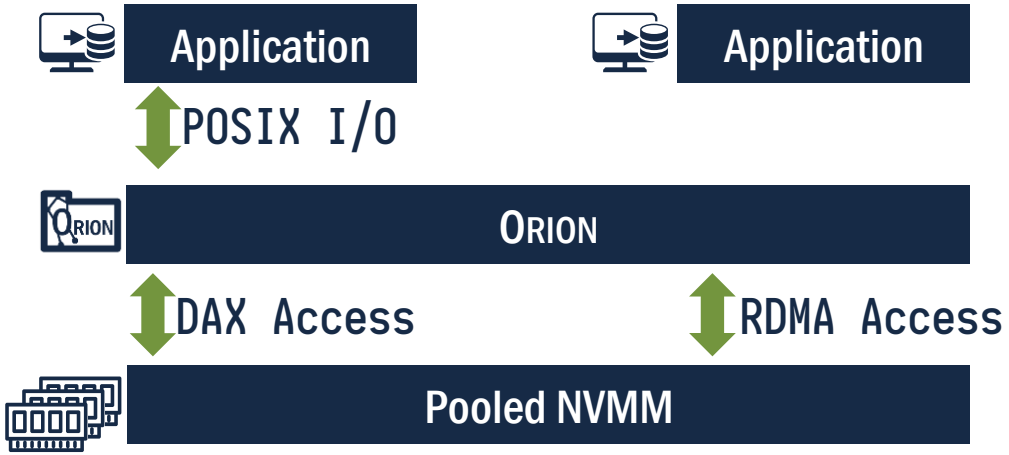
- Use case of converged storage
  - Local NVMM supports Direct Access (DAX)
- Existing systems do not store data at local
- Run Local FS and Dist. FS
  - Expensive to move data



NVSL

ORION: A Distributed File System for NVMM and RDMA-Capable Networks

- A clean slate design for NVMM and RDMA
- A unified layer: kernel FS + networking
- Pooled NVMM storage
- Accessing metadata/data directly over Direct Access (DAX) and RDMA
- Designed for rack-scale scalability



NVSL

Outline

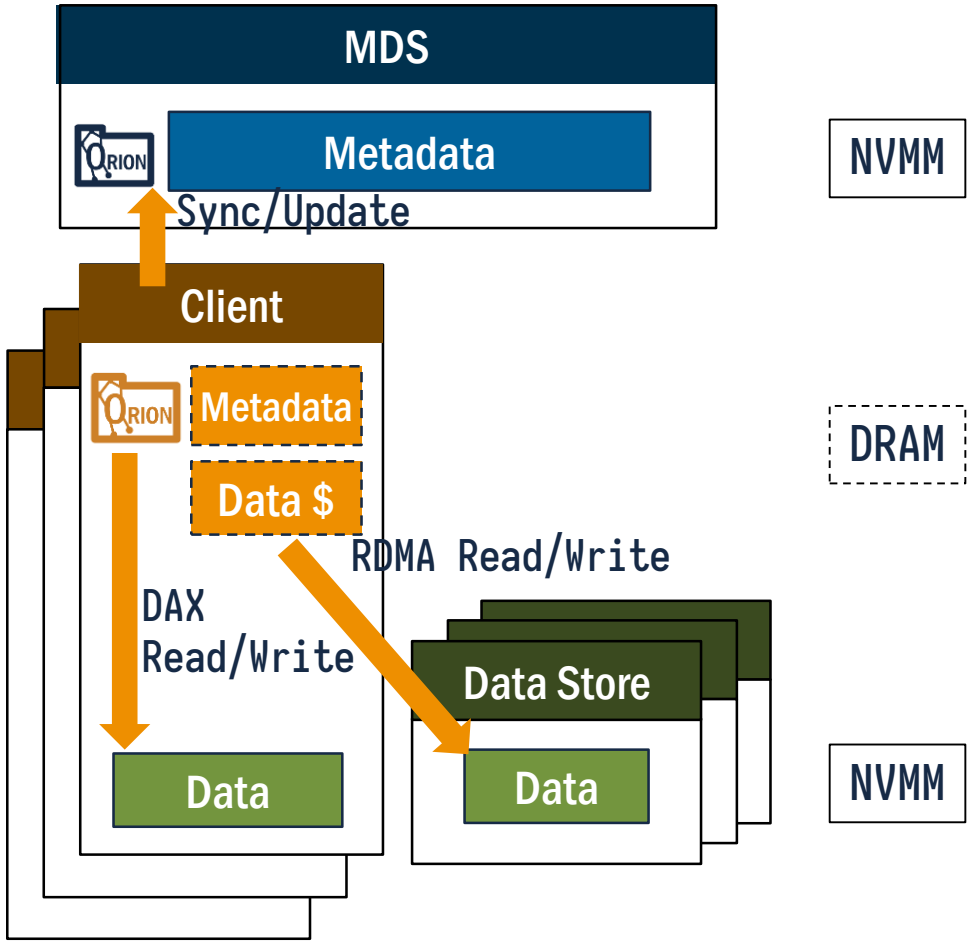
- Background
- Design overview
- Metadata and data management
- Replication
- RDMA persistence
- Evaluation
- Conclusion

NVSL

ORION: Cluster Overview

- **Metadata Server (MDS)**: Runs ORIONFS, keeps authoritative metadata of the whole FS
- **Client**: Runs ORIONFS, keeps active metadata and cached data. Access local NVMM
- **Data Store (DS)**: Pooled NVMM data

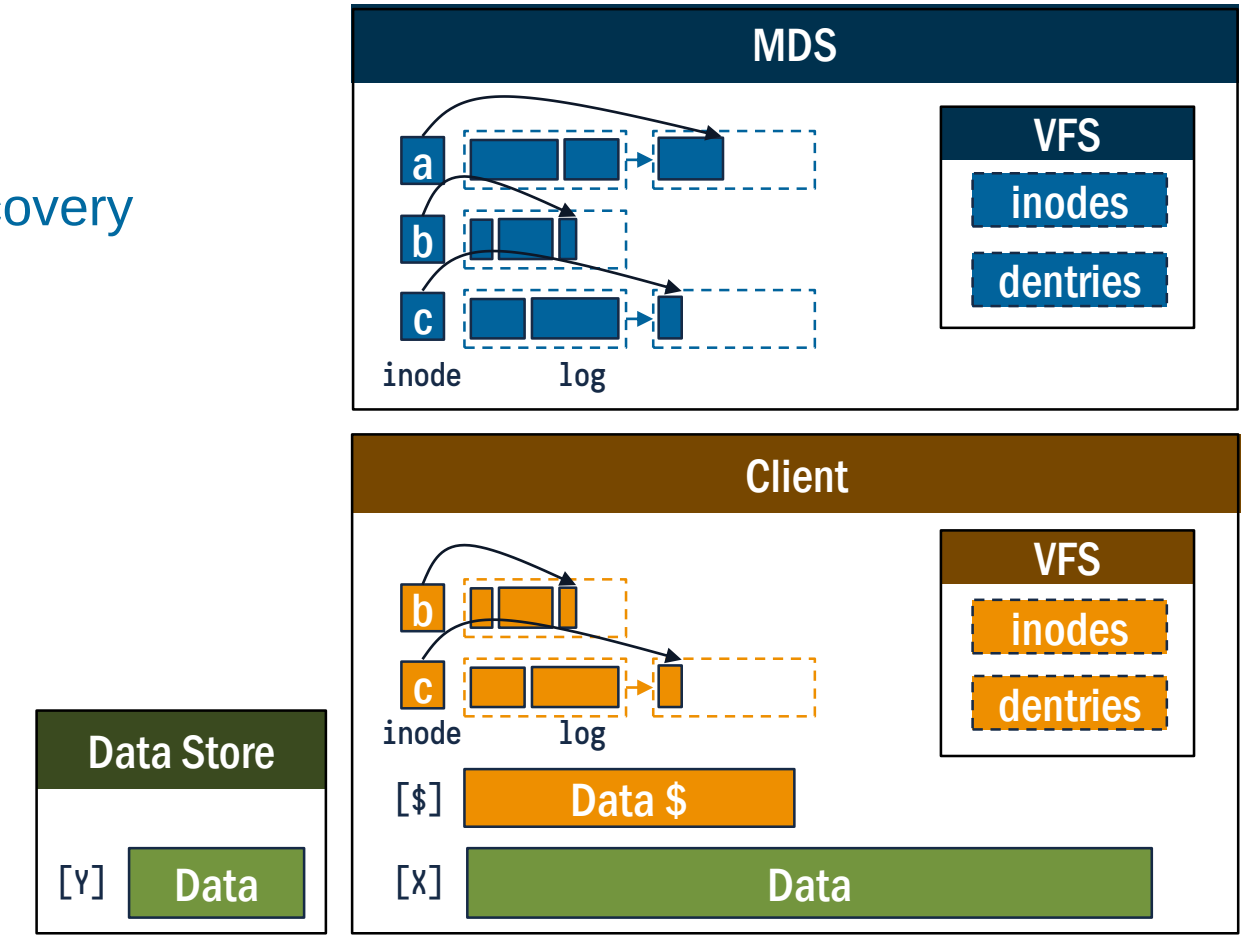
- Metadata Access: Clients <=> MDS (Two-sided)
- Data Access: Clients => DSs (One-sided)



NVSL

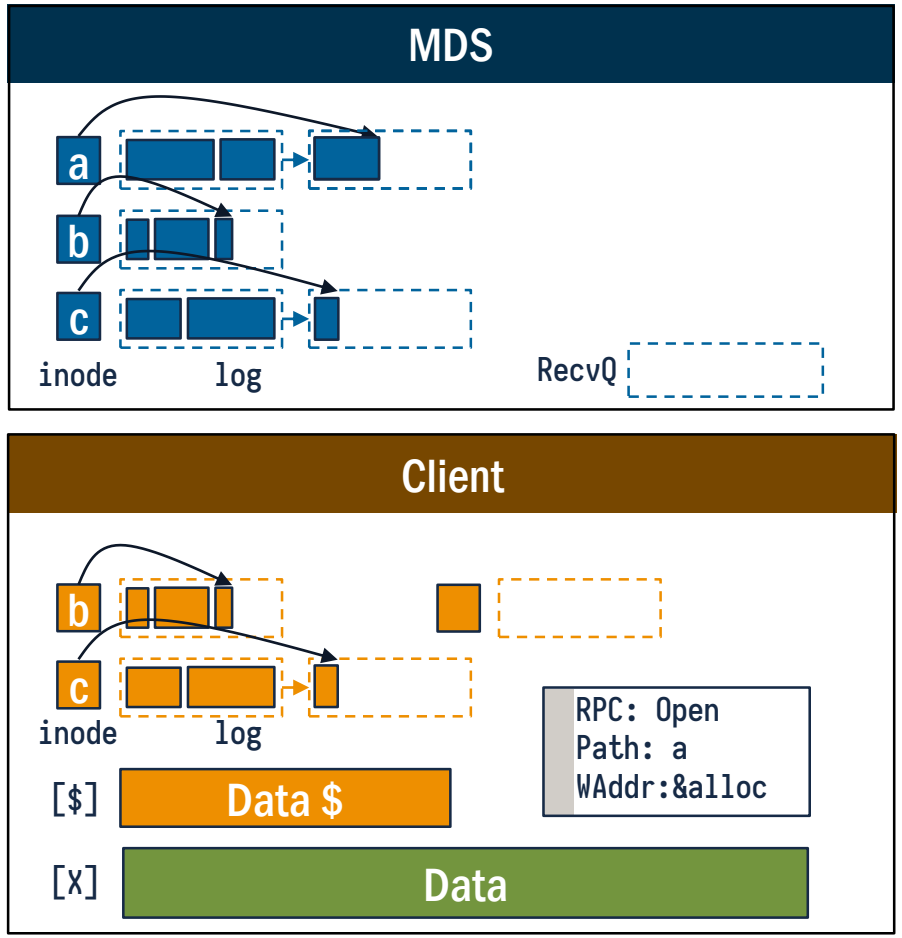
The ORION File System

- Inherited from NOVA [Xu, FAST 16]
  - Per-inode metadata (operation) log
  - Build in-DRAM data structures on recovery
  - Atomic log append
- Metadata:
  - DMA (Physical) memory region (MR)
  - RDMA-able metadata structures
- Data: Globally partitioned



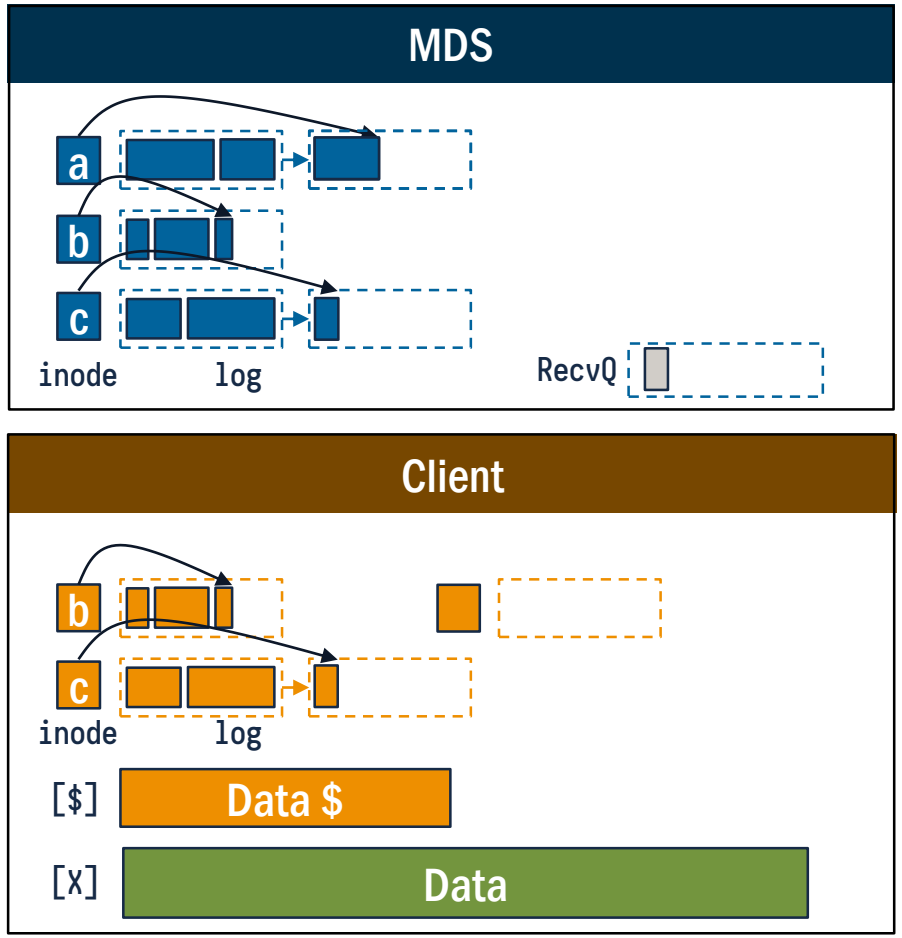
Operations in ORION File System

- Open(a)
  - Client Allocate inode



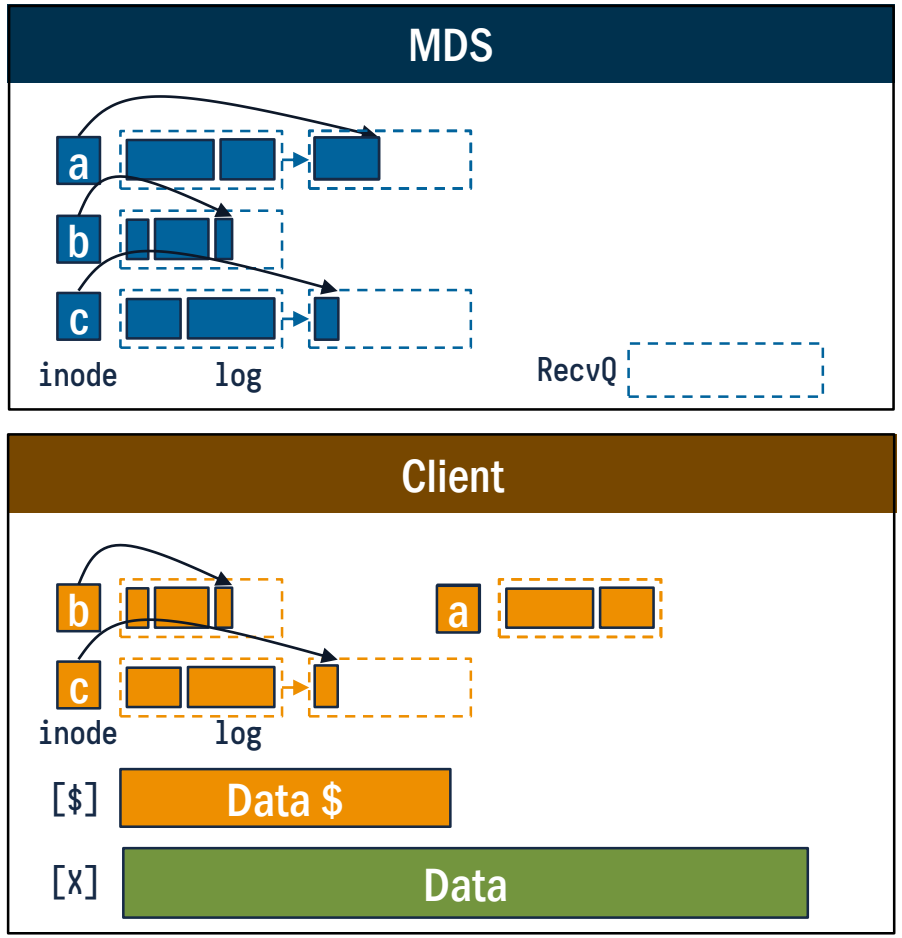
Operations in ORION File System

- Open(a)
  - Client Allocate inode
  - Client Issue an RPC via **RDMA_Send**



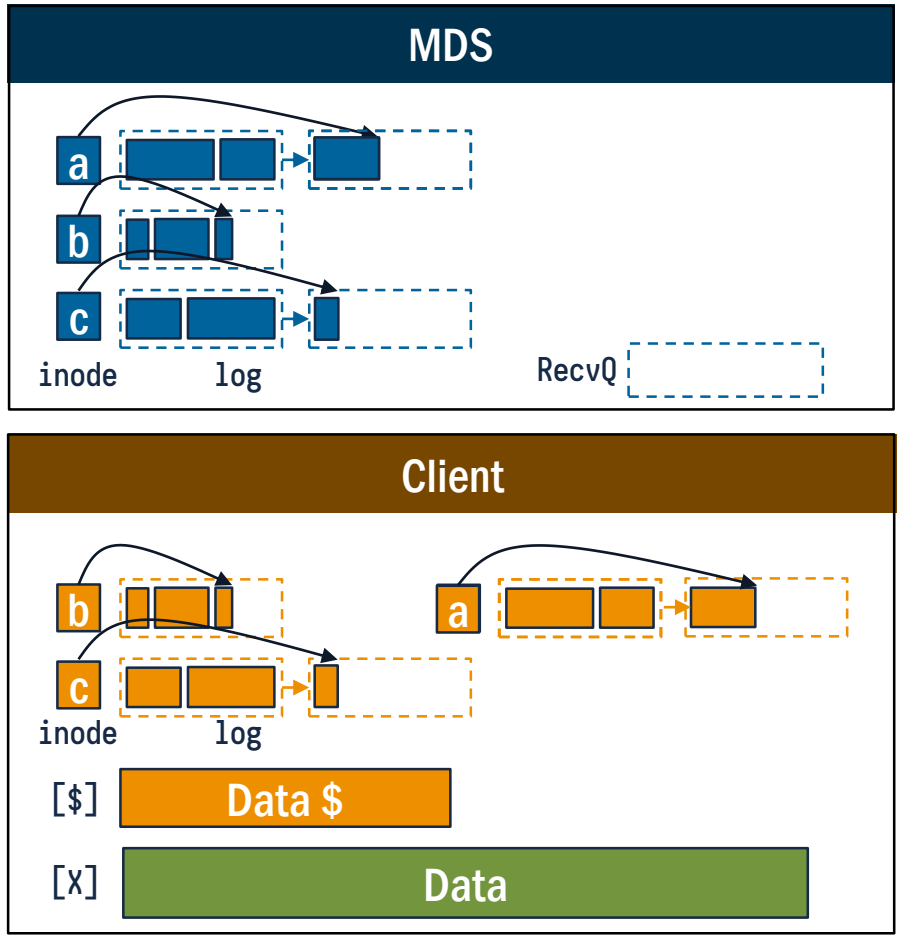
Operations in ORION File System

- Open(a)
  - Client Allocate inode
  - Client Issue an RPC via **RDMA_Send**
  - MDS **RDMA_Write** to allocated space



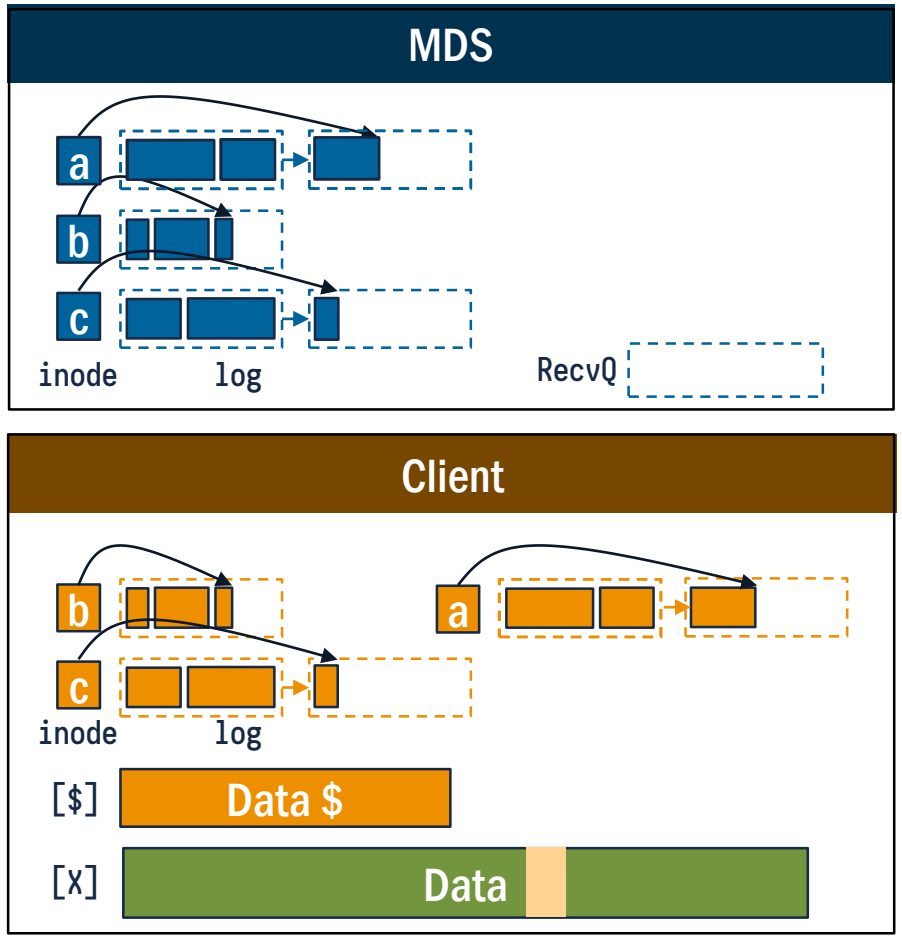
Operations in ORION File System

- Open(a)
  - Client Allocate inode
  - Client Issue an RPC via **RDMA_Send**
  - MDS **RDMA_Write** to allocated space
  - Client **RDMA_Read** the rest of the log



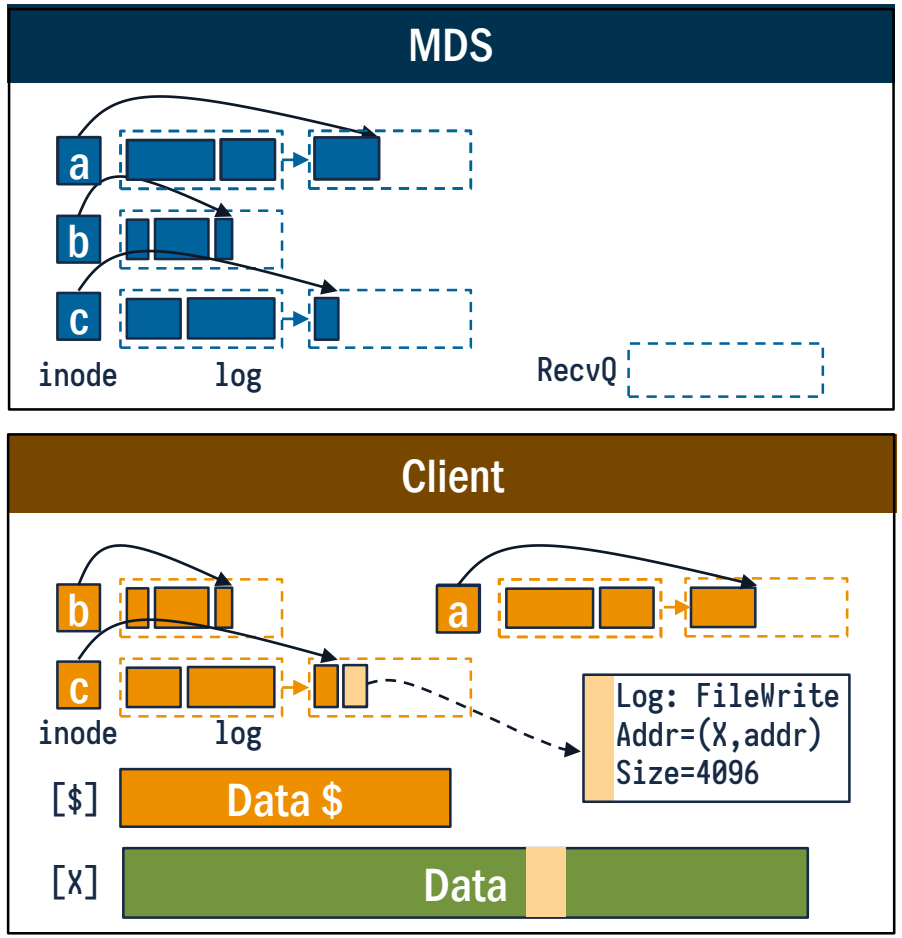
Operations in ORION File System

- Write(c)
  - Client Allocate & CoW to client-owned pages



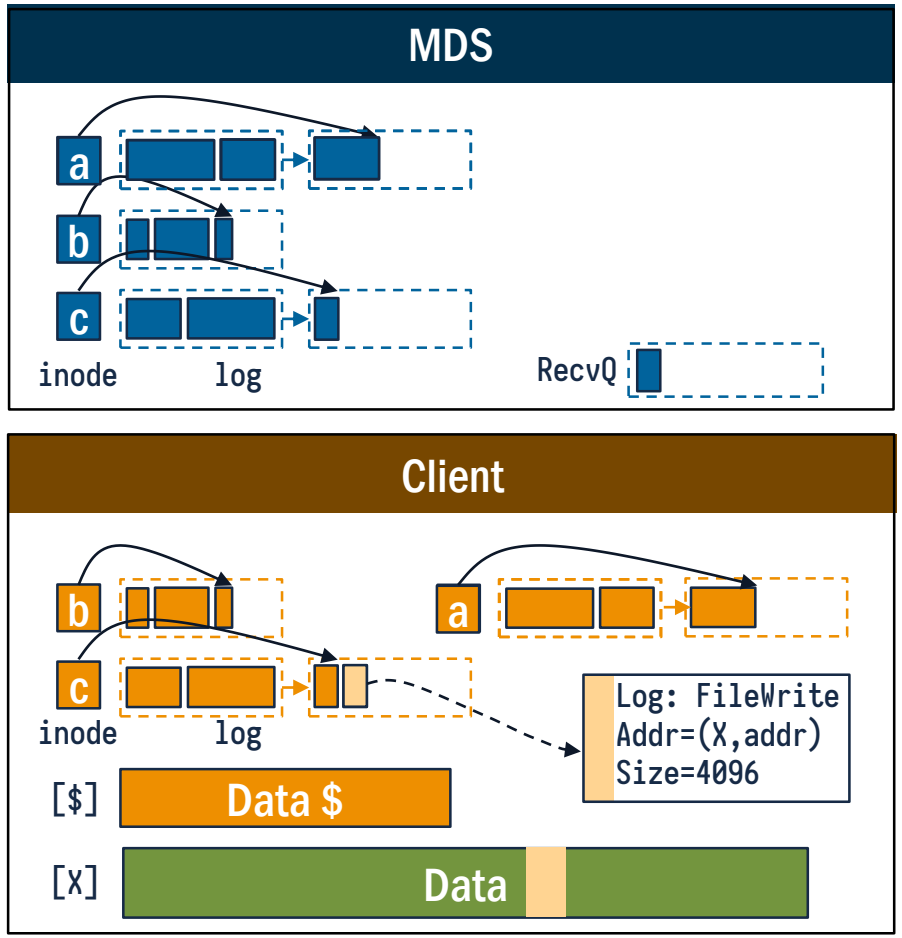
Operations in ORION File System

- Write(c)
  - Client Allocate & CoW to client-owned pages
  - Client Append log entry



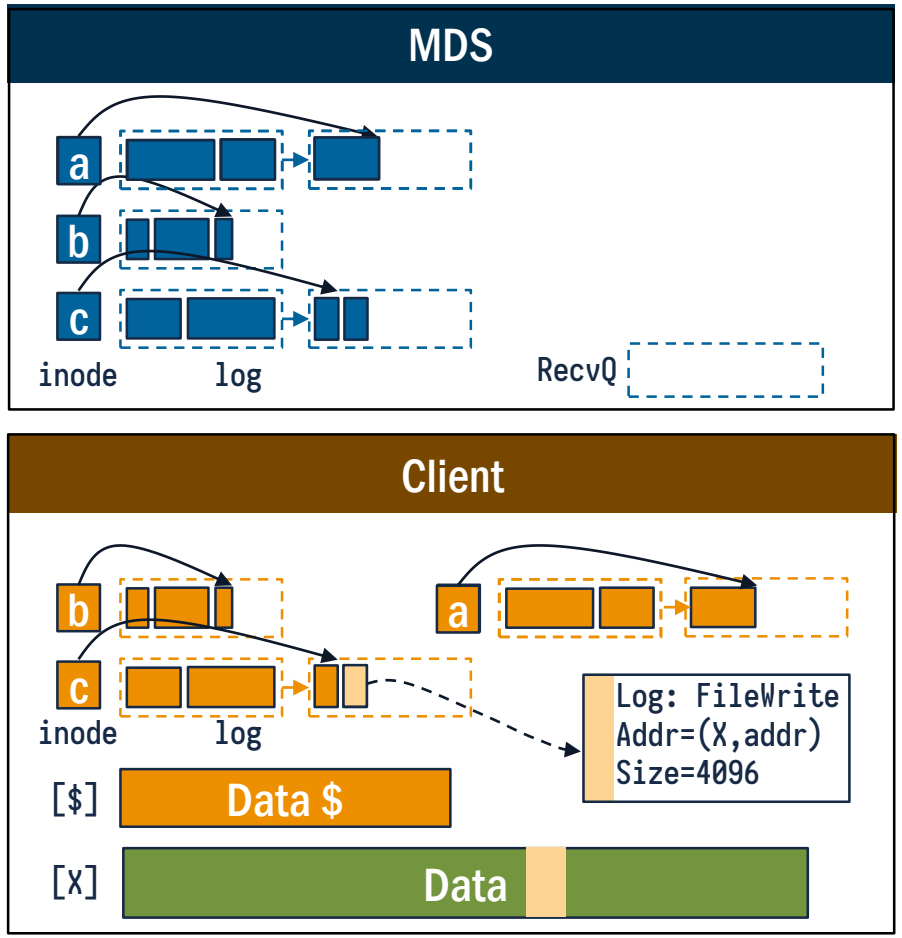
Operations in ORION File System

- Write(c)
  - Client Allocate & CoW to client-owned pages
  - Client Append log entry
  - Client Commit log entry via **RDMA_Send**



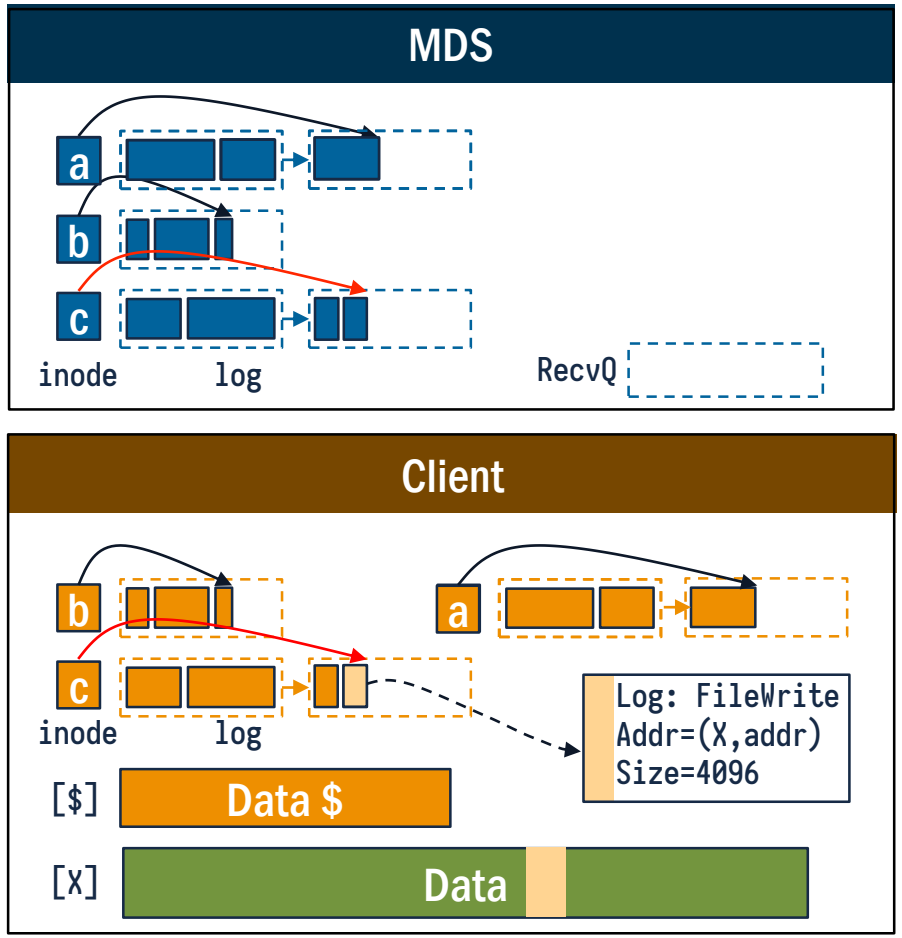
Operations in ORION File System

- Write(c)
  - Client Allocate & CoW to client-owned pages
  - Client Append log entry
  - Client Commit log entry via **RDMA_Send**
  - MDS Append log entry



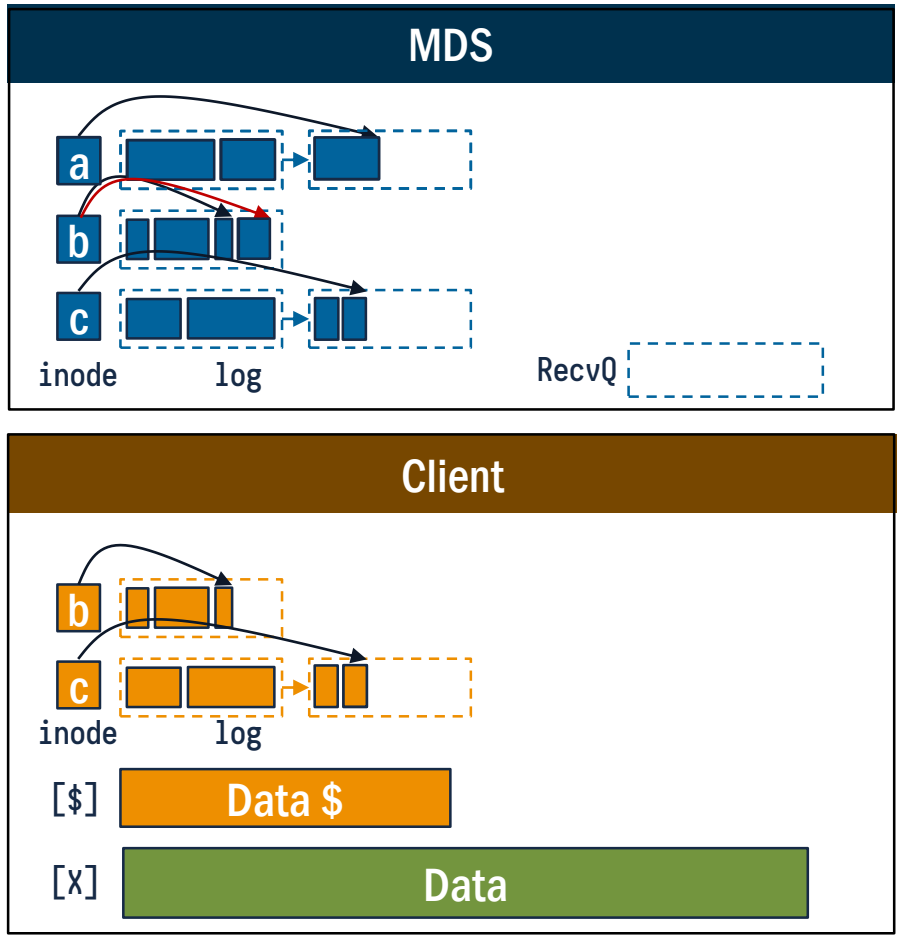
Operations in ORION File System

- Write(c)
  - Client Allocate & CoW to client-owned pages
  - Client Append log entry
  - Client Commit log entry via **RDMA_Send**
  - MDS Append log entry
  - Client MDS Update tail pointers atomically



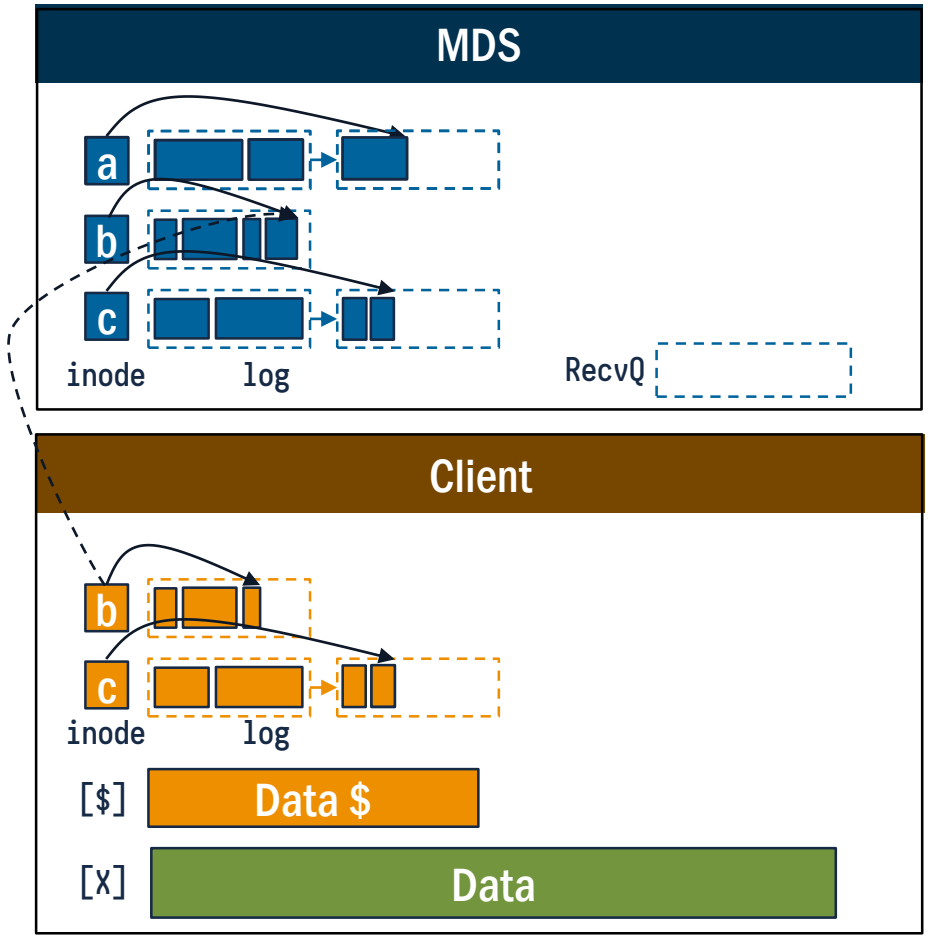
Operations in ORION File System

- Tailcheck(b)
  - MDS Log commit from another client



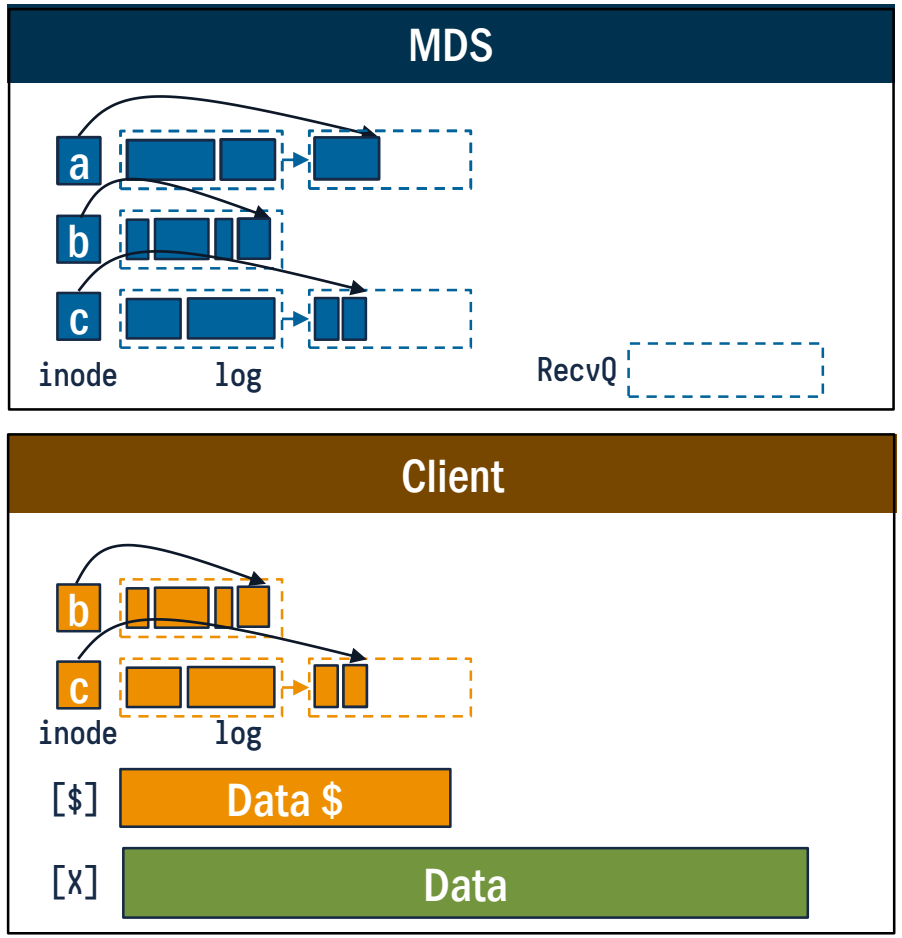
Operations in ORION File System

- Tailcheck(b)
  - MDS Log commit from another client
  - Client **RDMA_Read** remote log tail



Operations in ORION File System

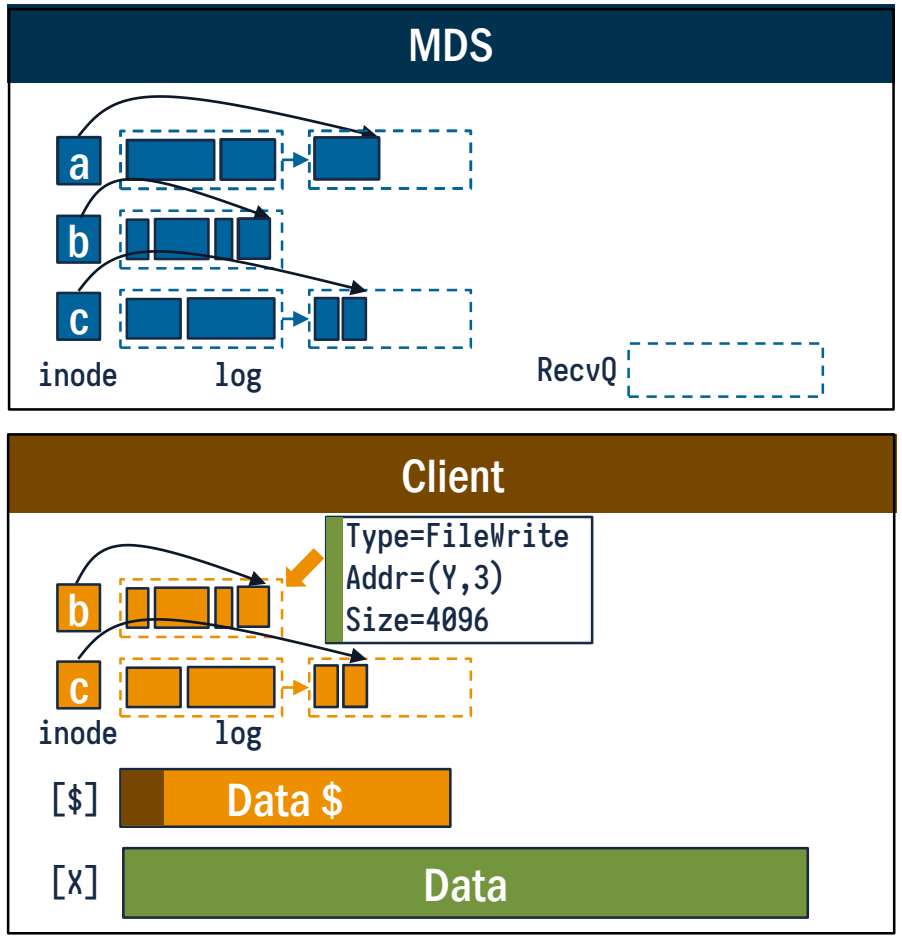
- Tailcheck(b)
  - MDS Log commit from another client
  - Client **RDMA_Read** remote log tail
  - Client Read from MDS if Len(Local) < Len(Remote)



Operations in ORION File System

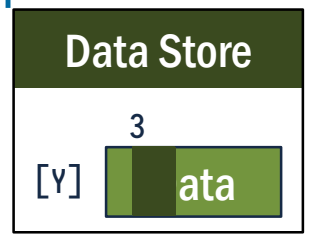
- Read(b)
  - Client Tailcheck (async)
  - Client **RDMA_Read** from data store

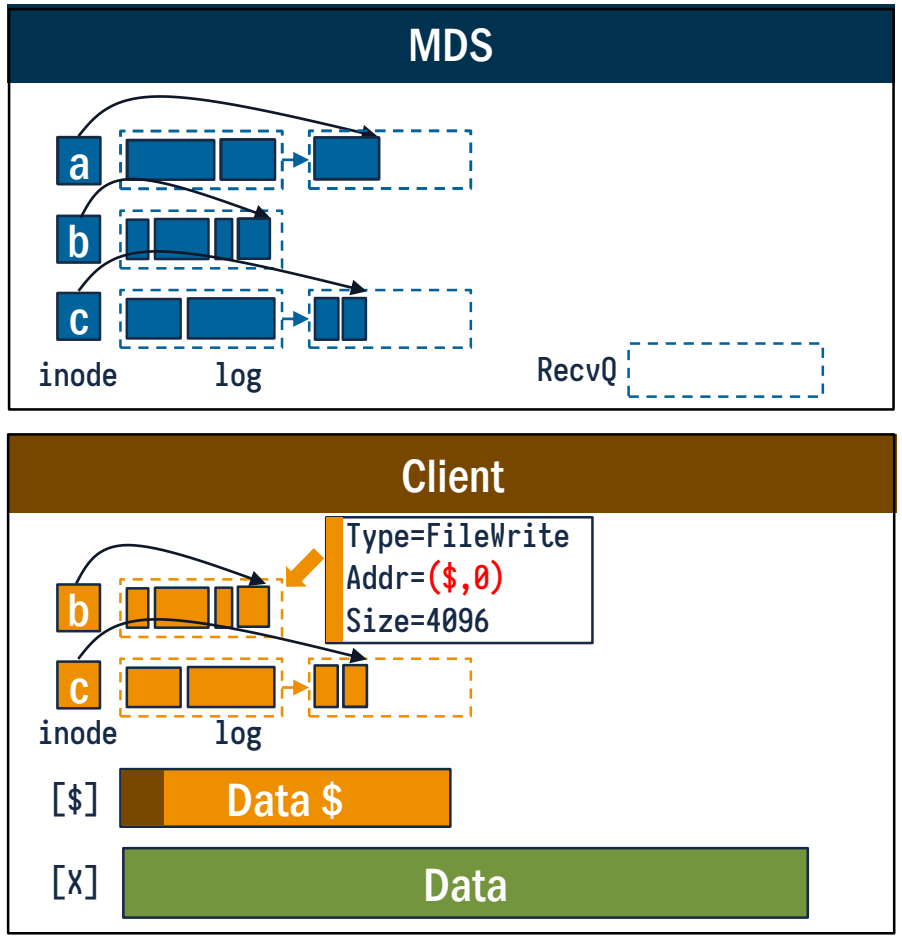
- Data locality
  - Future reads will hit DRAM cache
  - Future writes will go to local NVMM



Operations in ORION File System

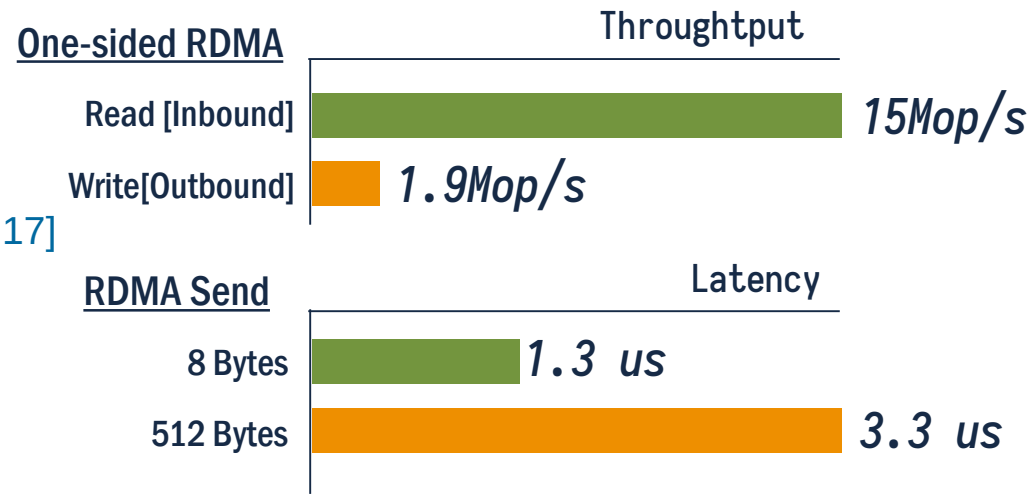
- Read(b)
  - Client Tailcheck (async)
  - Client **RDMA_Read** from data store
  - Client In-place update to log entry
- Data locality
  - Future reads will hit DRAM cache
  - Future writes will go to local NVMM





Accelerating Metadata Accesses

- Observations:
  - RDMA prefers inbound operations [Su, EuroSys 17]
  - RDMA prefers small operations [Kalia, ATC 16]



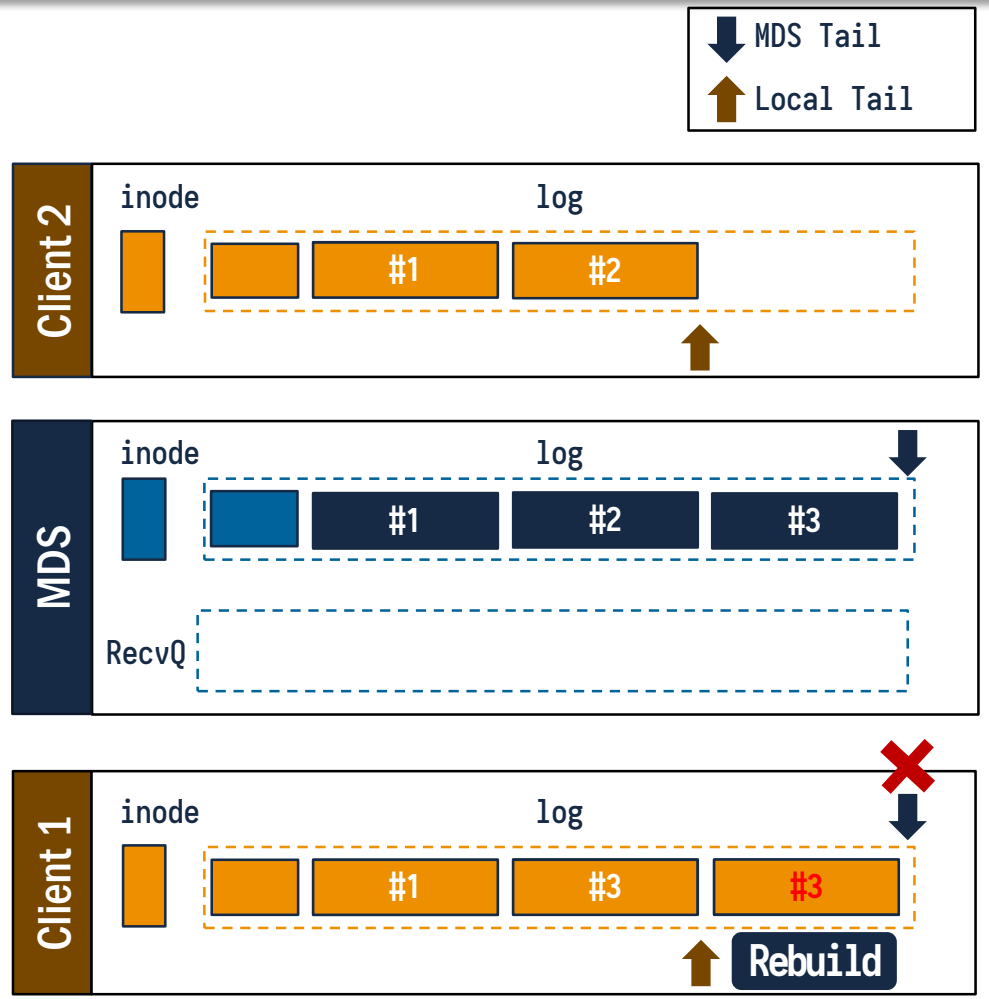
- MDS request handling:
  - **Tailcheck** (8B RDMA_Read): MDS-bypass
  - **Log Commit** (~128B RDMA_Send): Single-inode operations
  - **RPC** (*Varies*): Other operations, less common

NVSL

Optimizing Log Commits

- Speculative log commit:
  - Return when RDMA_Send verb is signaled
  - Tailcheck before send
  - Rebuild inode from log when necessary
  - RPCs for complex operations (e.g. O_APPEND)

- Log commit + Persist: ~ 500 CPU Cycles
  (memcpy) (flush+fence)



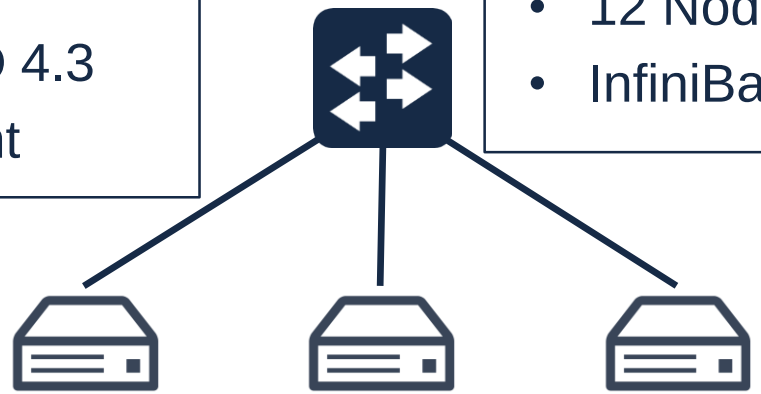
Evaluation

ORION Prototype

- ORION kernel modules (~15K LOC)
- Linux Kernel 4.10
- RDMA Stack: MLNX_OFED 4.3
- Bind to 1 core for each client

Networking

- 12 Nodes connected to a switch
- InfiniBand Switch (QLogic 12300)

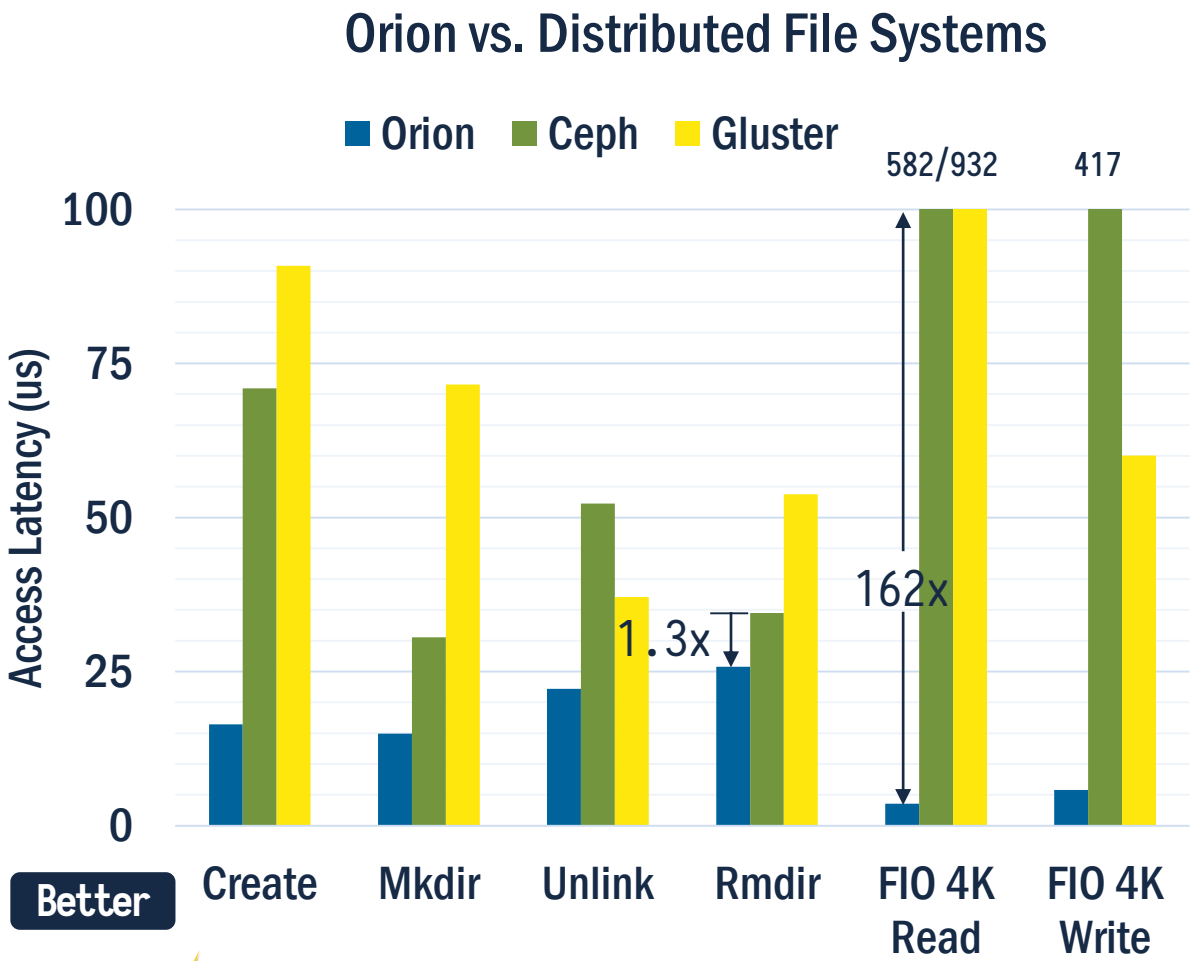


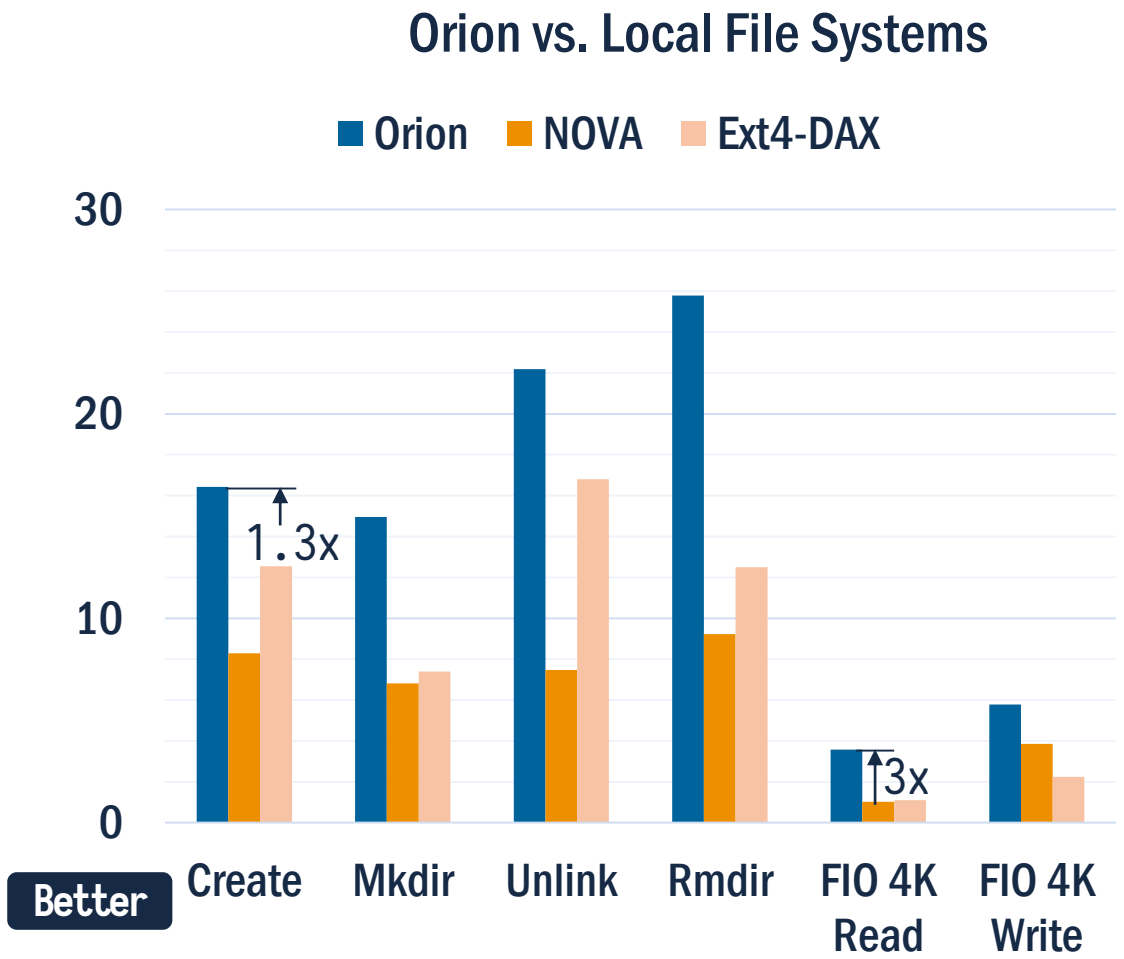
Hardware

- 2x Intel Westmere-EP CPU
- 16GB DRAM as DRAM
- 32GB DRAM as NVMM
- RNIC: Mellanox ConnectX-2 VPI (40Gbps)

NVSL

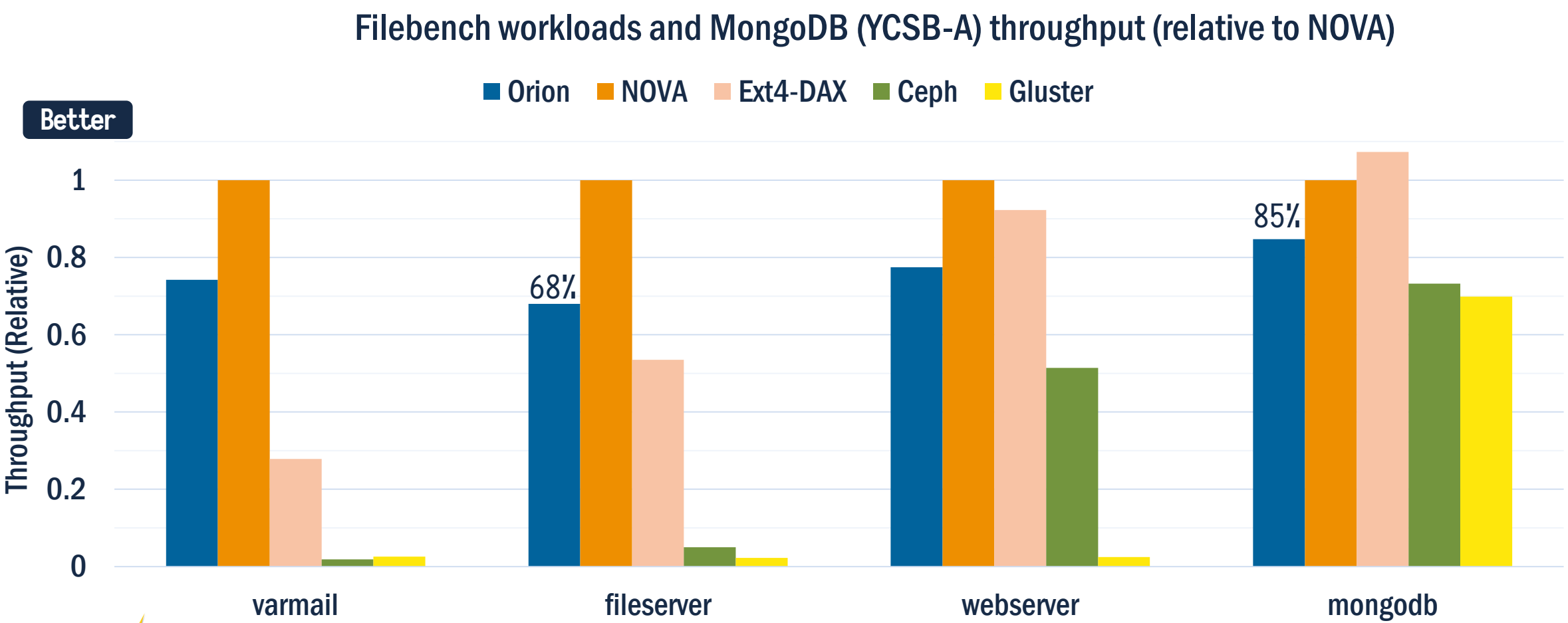
Evaluation: File Operations





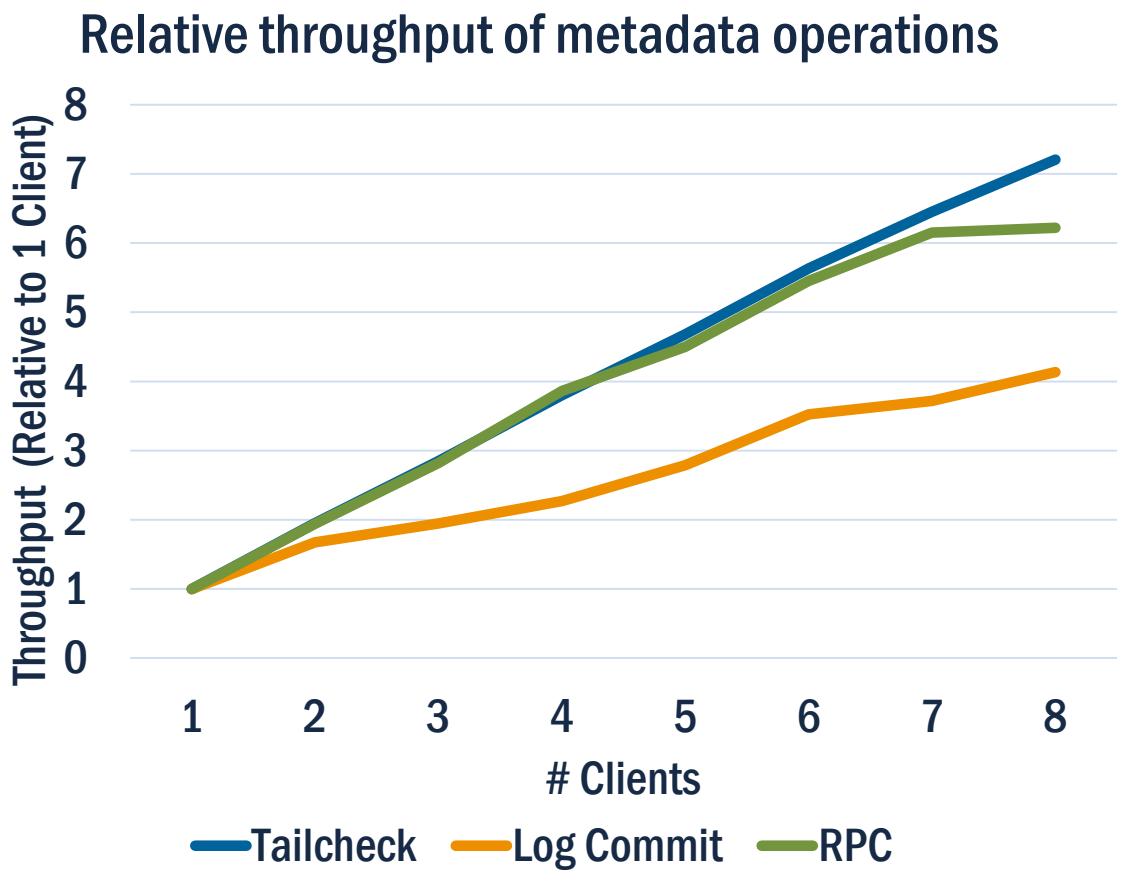
NVSL

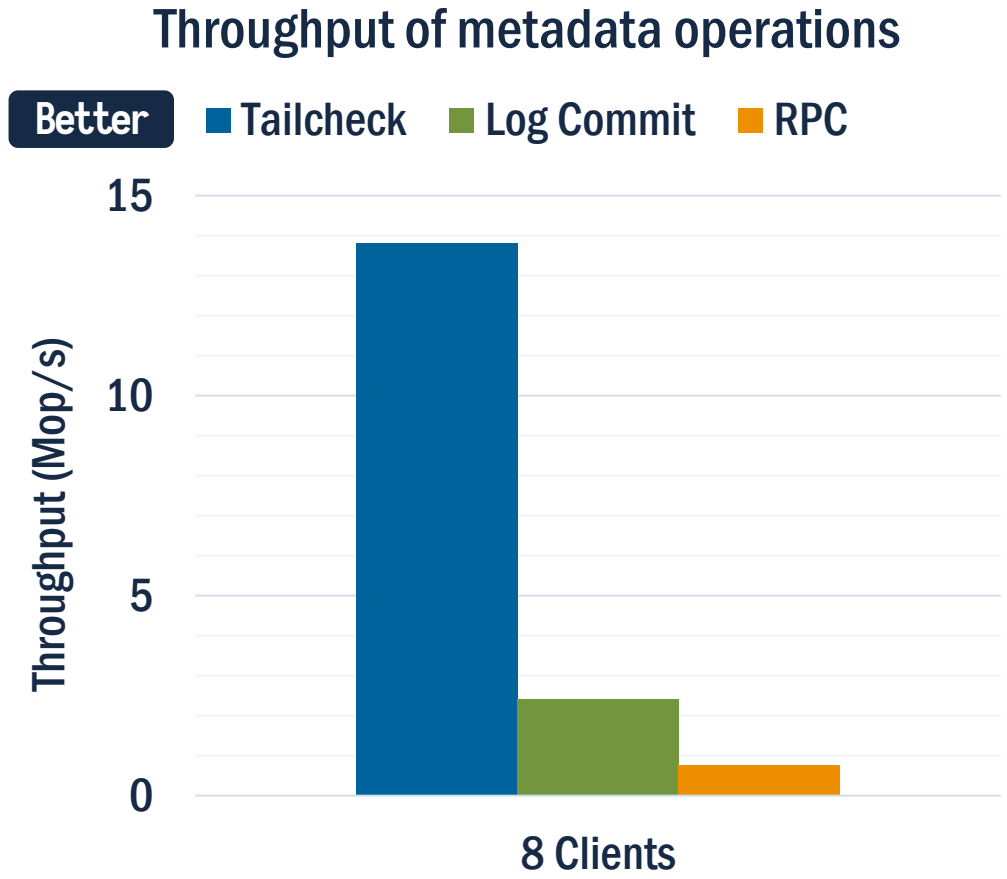
Evaluation: Applications



NVSL

Evaluation: Metadata Accesses





NVSL

Conclusion

- Existing distributed file systems lack of NVMM support and have significant software overhead
- ORION unifies the NVMM file system and the networking layer
- ORION provides fast metadata accesses
- ORION allows DAX to local NVMM data
- Performance comparable to local NVMM file systems

NVSL