STORAGE DEVELOPER CONFERENCE

SDC 23

BY Developers FOR Developers

The Case for NFS-eSSDs

David Flynn, Hammerspace Founder & CEO

Why Parallel NFS is Relevant Now More Than Ever

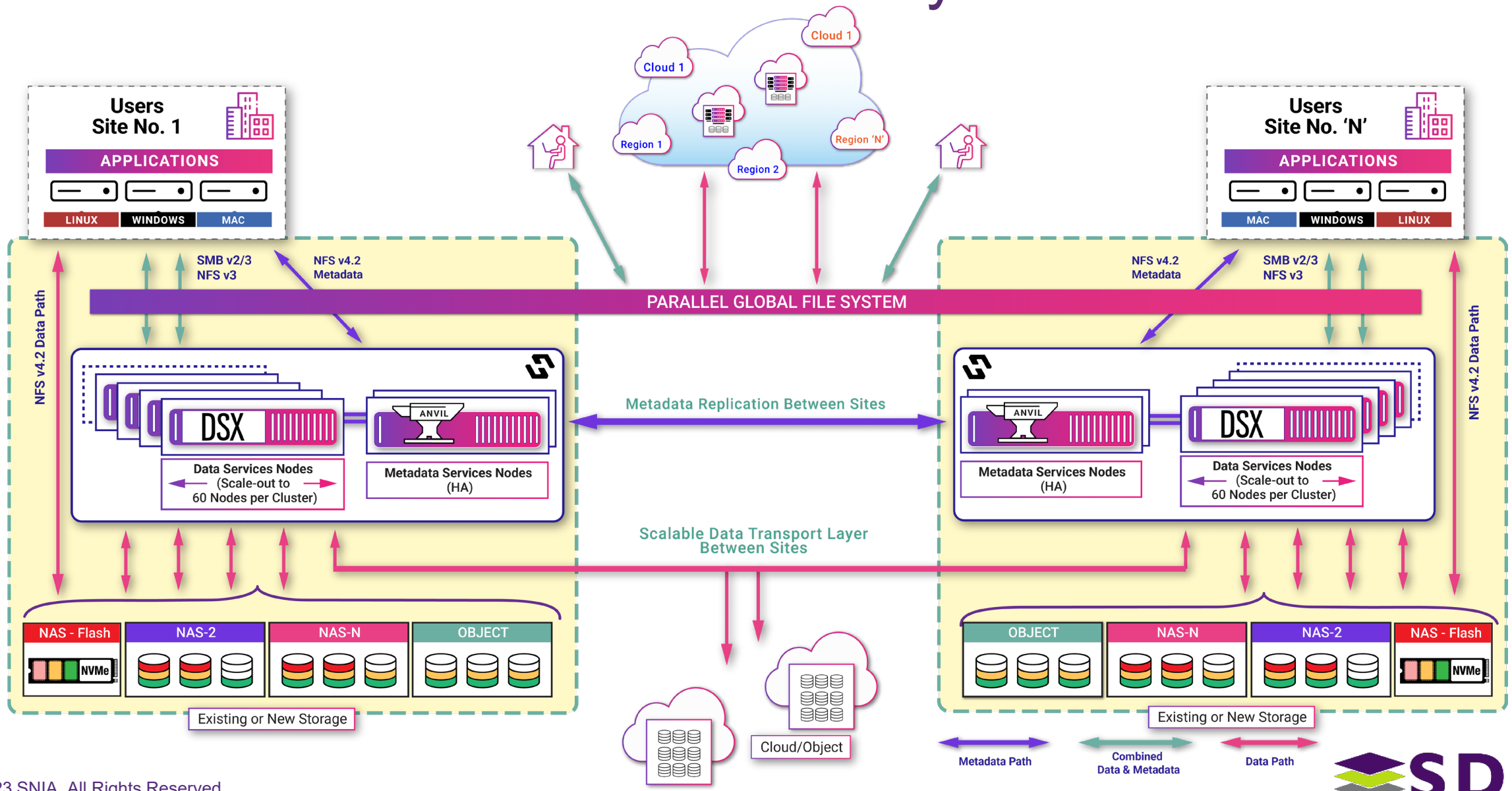
The Current Reality:

- Data orchestration is an absolute requirement across silos, sites, & clouds.
- High-performance requirements have gone mainstream.
- The world is moving to software-defined on commodity infrastructure.
- Linux is ubiquitous → enables a sophisticated, standards-based, open-source client to come built-in (not third-party).

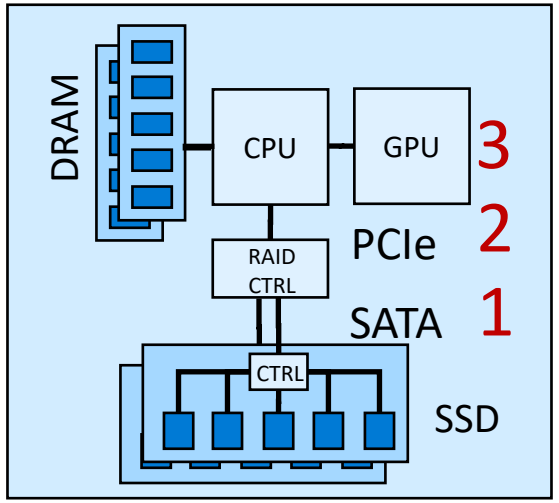
Therefore:

- NFS 4.2 solves these problems.
  - File access that bridges storage silos, sites & clouds.
  - Parallel file system with no need to install third-party client & management tools.
  - Avoids need to rewrite apps to use object storage.

Unstructured Data Orchestration System in Action

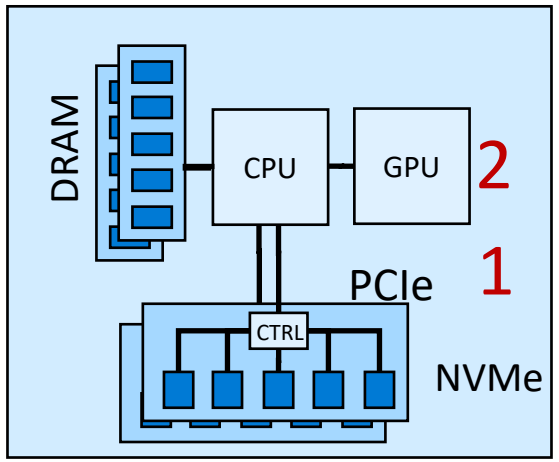


Direct Attached Storage



The RAID controller is the bottleneck and adds an additional serial data retransmission.

Direct Attached Storage - NVMe

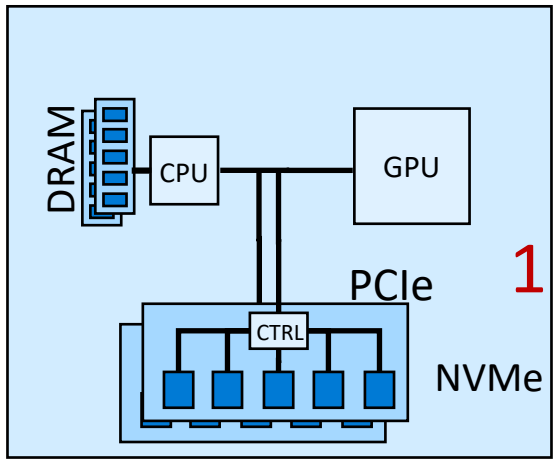


NVMe eliminates the RAID controller

Direct Attached Storage – NVMe and GPU Direct

GPU Direct eliminates the host CPU and memory

But, what about with shared storage?

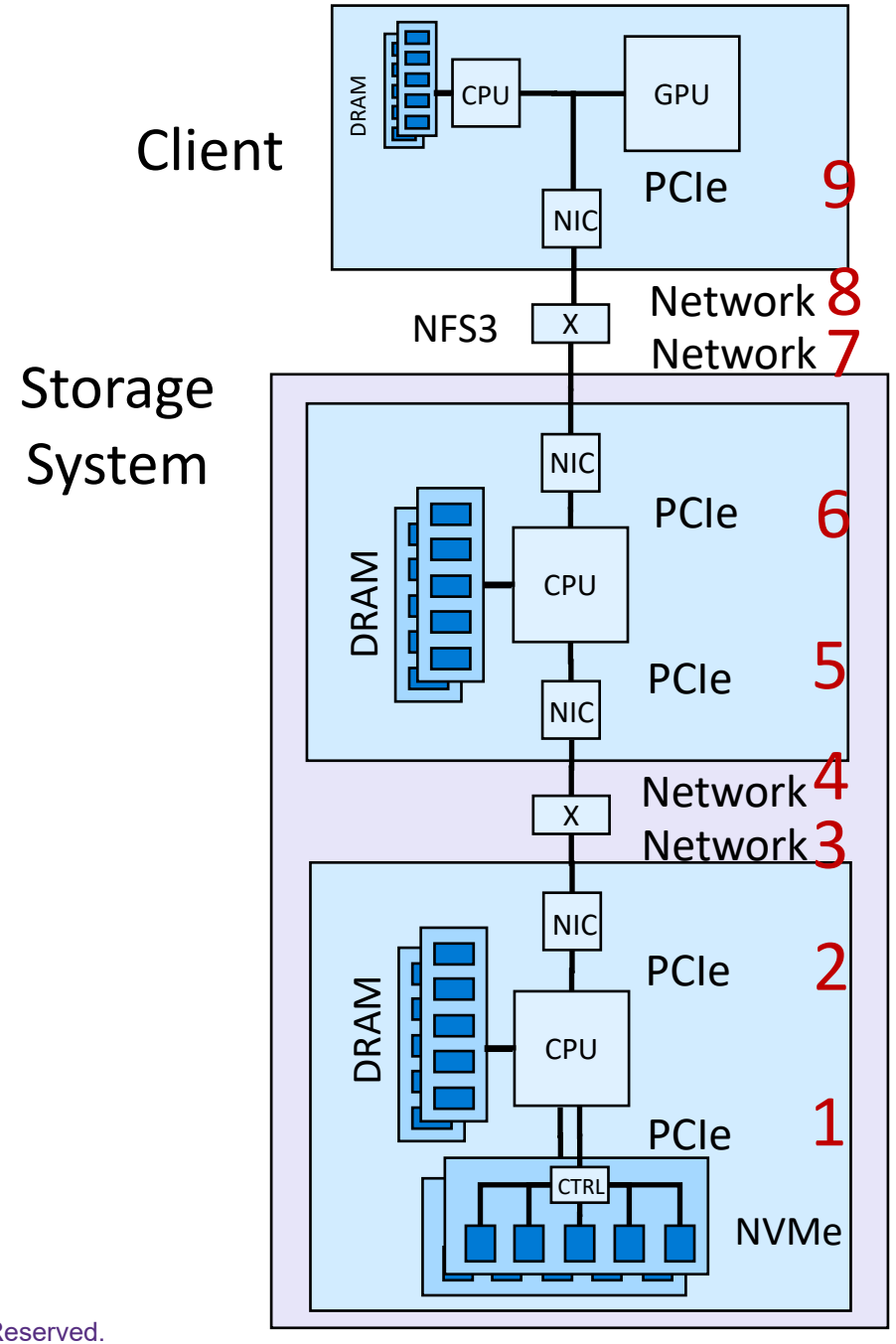


DENTRY to INODE mapping

File offset to block mapping

Block to flash address mapping

Network Attached Storage
(e.g. NetApp, Isilon, Pure, Qumulo, Ceph)



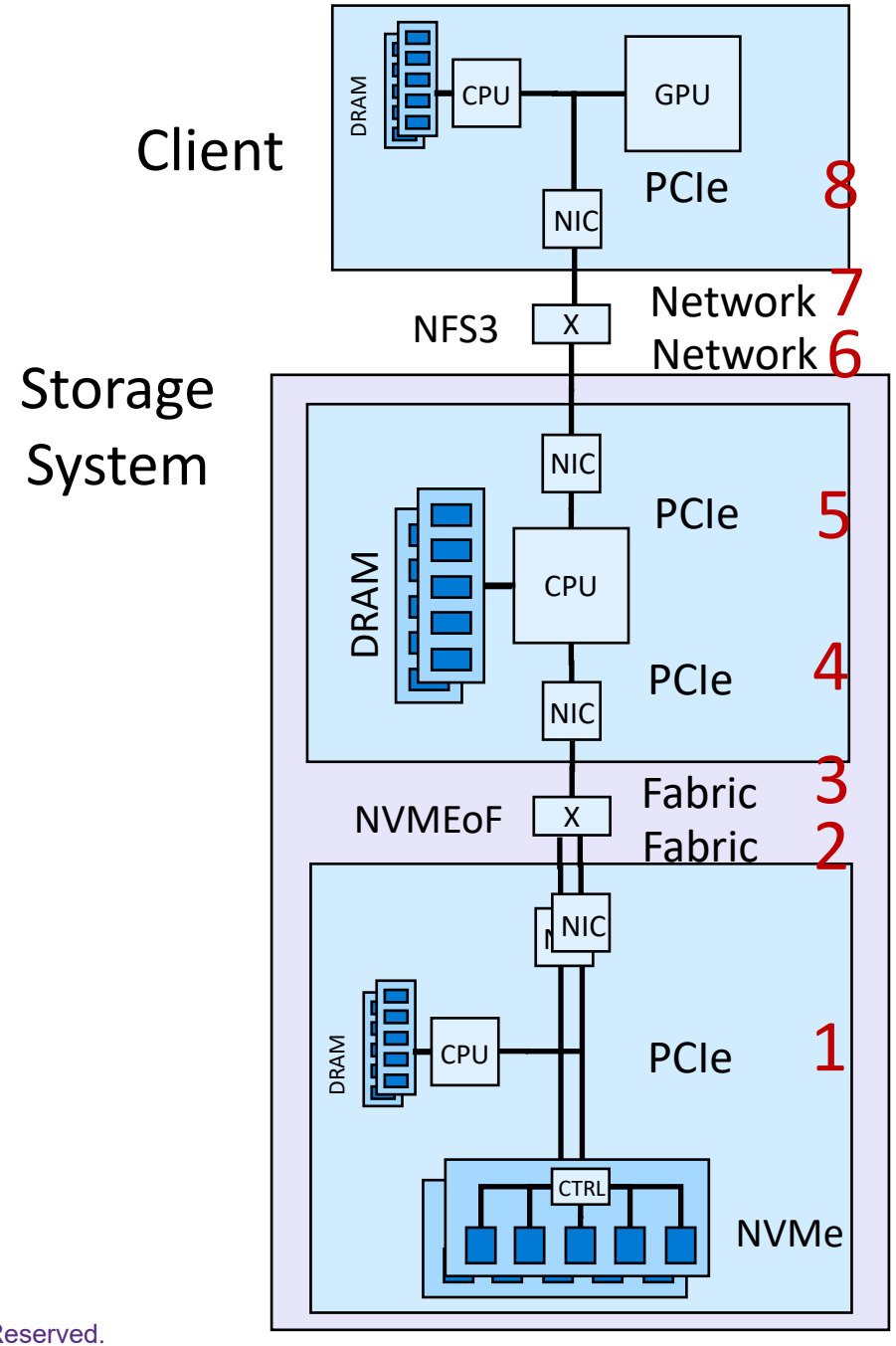
DENTRY to INODE mapping

File offset to block mapping

The storage back-end host (CPU and memory) is the first bottleneck.

Block to flash address mapping

Network Attached Storage
(using NVMEoF, e.g. VAST, Weka)



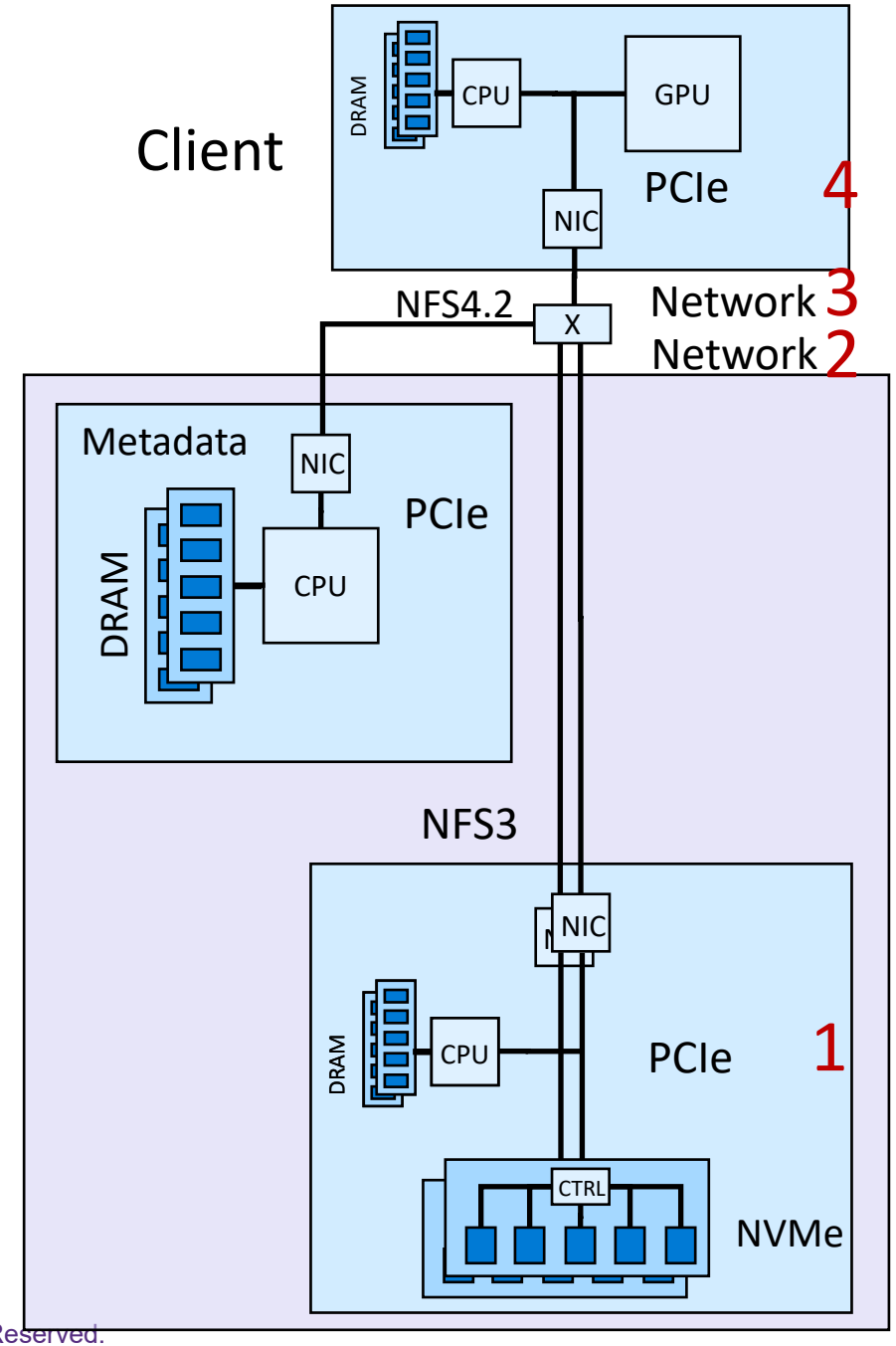
DENTRY to INODE mapping

File offset to block mapping

Block to flash address mapping

The file server front end is an even bigger bottleneck.

Network Attached Storage
(using NFS4.2, e.g. Hammerspace)



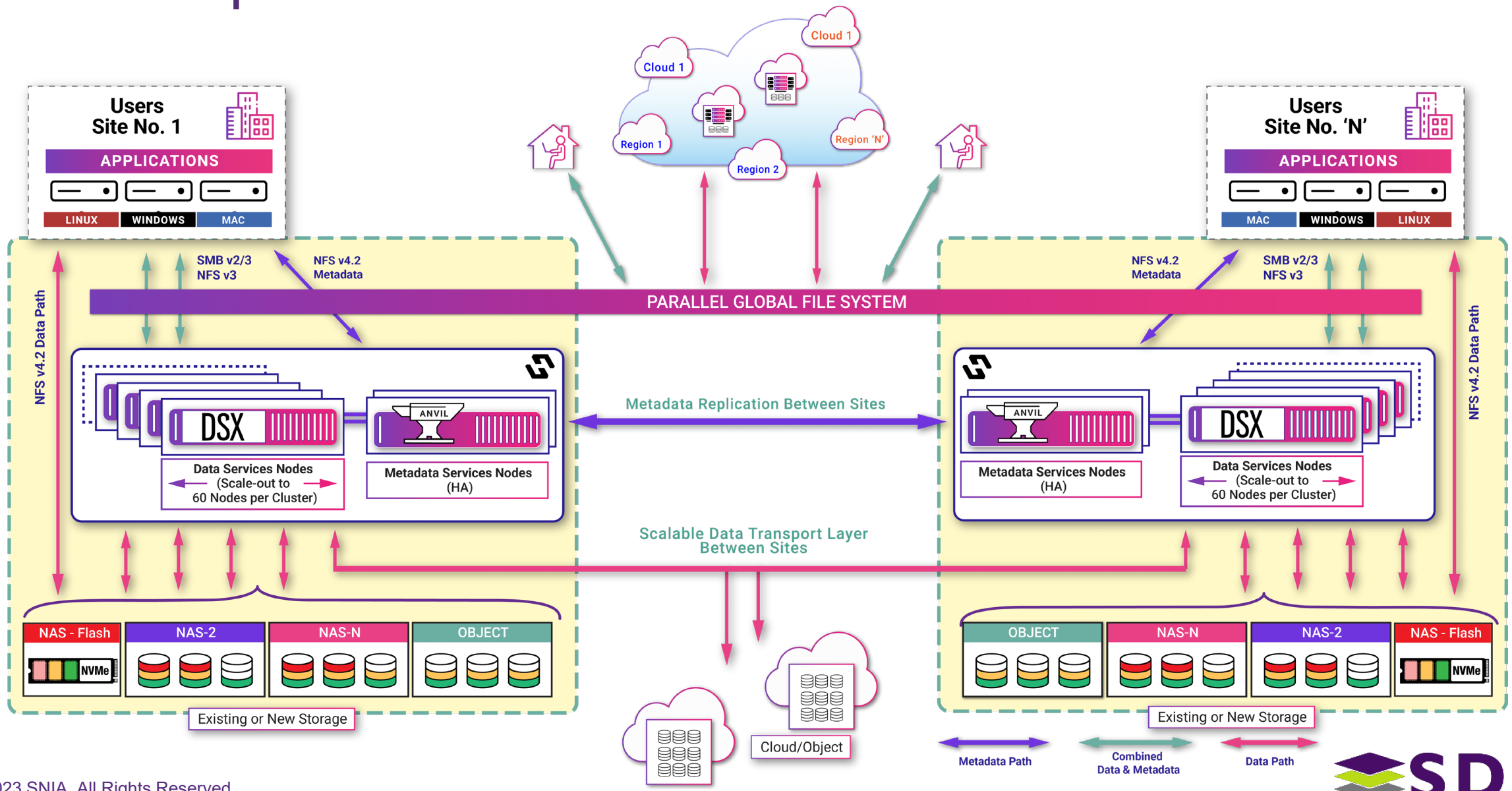
DENTRY to INODE mapping

File offset to block mapping

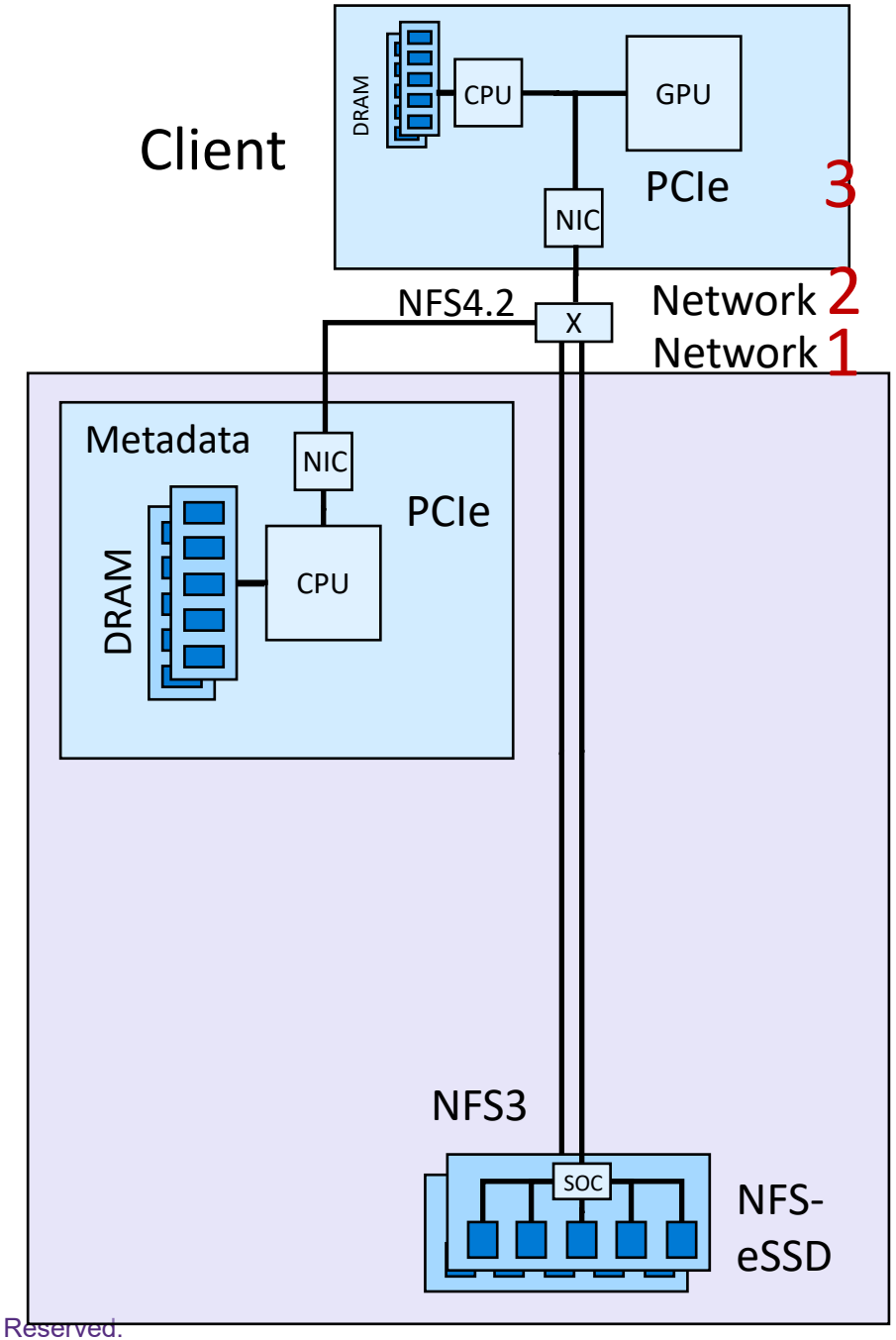
Block to flash address mapping

NFS4.2 has no bottlenecks, eliminates 4 of 9 data retransmissions, and doesn't need NVMEoF – or even an internal network!

Hammerspace Architecture



Network Attached Storage
(using NFS4.2 and NFS-eSSD)



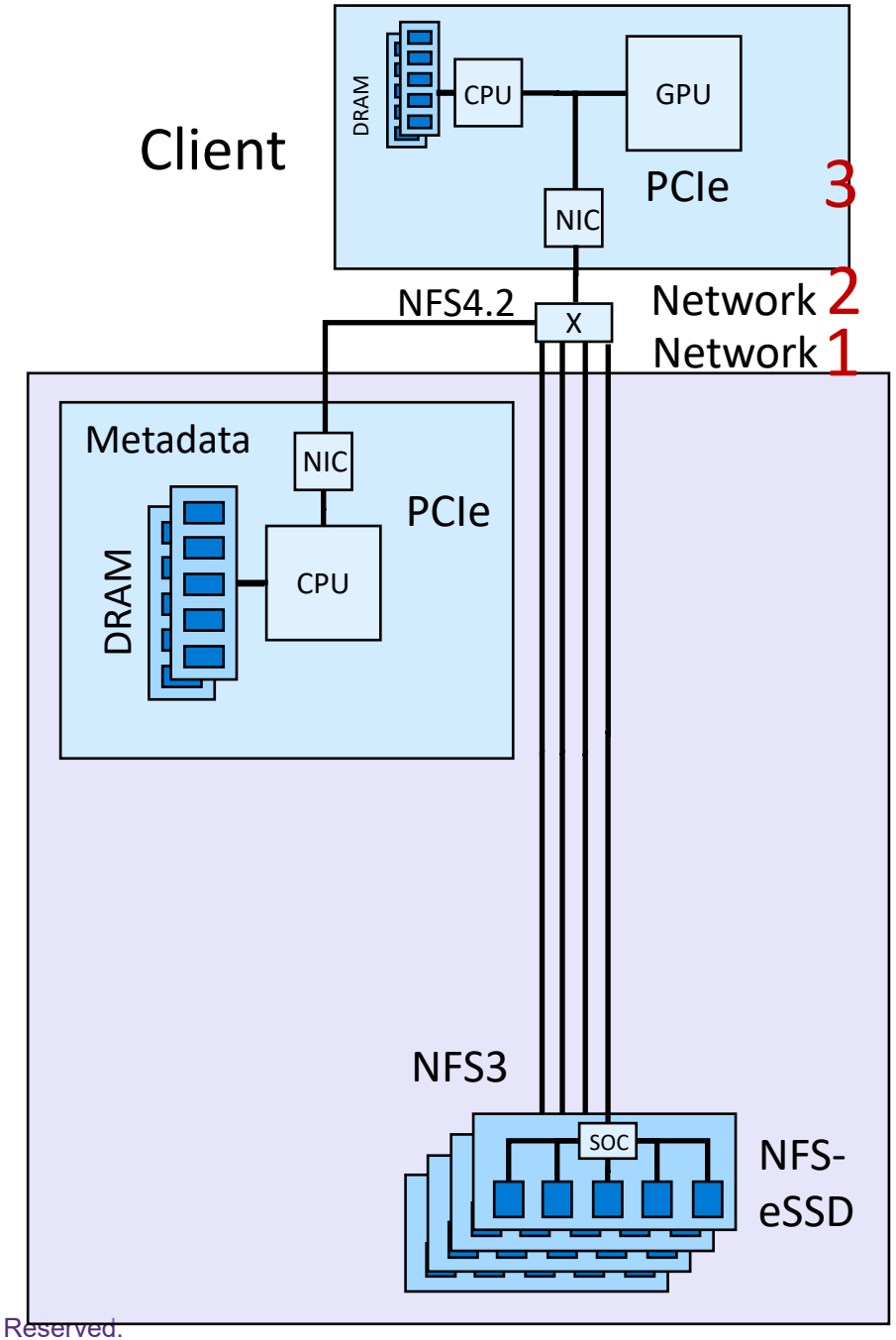
DENTRY to INODE mapping

File offset to block mapping

Block to flash address mapping

Network Attached Storage
(using NFS4.2 and NFS-eSSD)

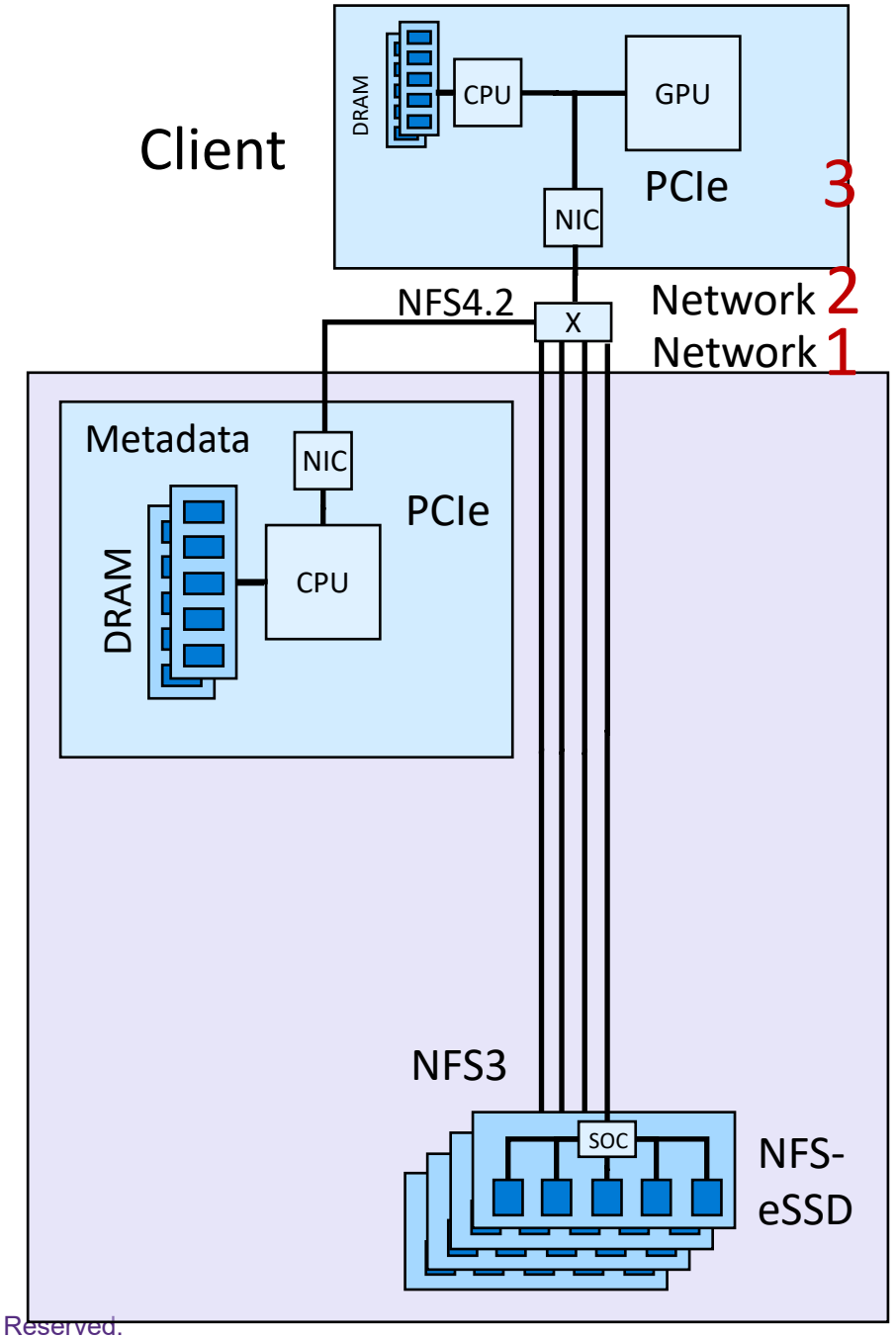
Network Attached Storage
(using NFS4.2 and NFS-eSSD)C



DENTRY to INODE mapping

NFS4.2 with proposed NFS-eSSDs eliminates 6 of 9 data retransmissions, eliminates the double mapping layers, and scales 1x1 with network ports!

File offset to flash address mapping



Benefits

Lower latency

Lower power consumption

Lower operational (and capital) costs

Lower write amplification

Higher density without sacrifice of potential performance

Higher access density

Better inherent reliability, availability and serviceability

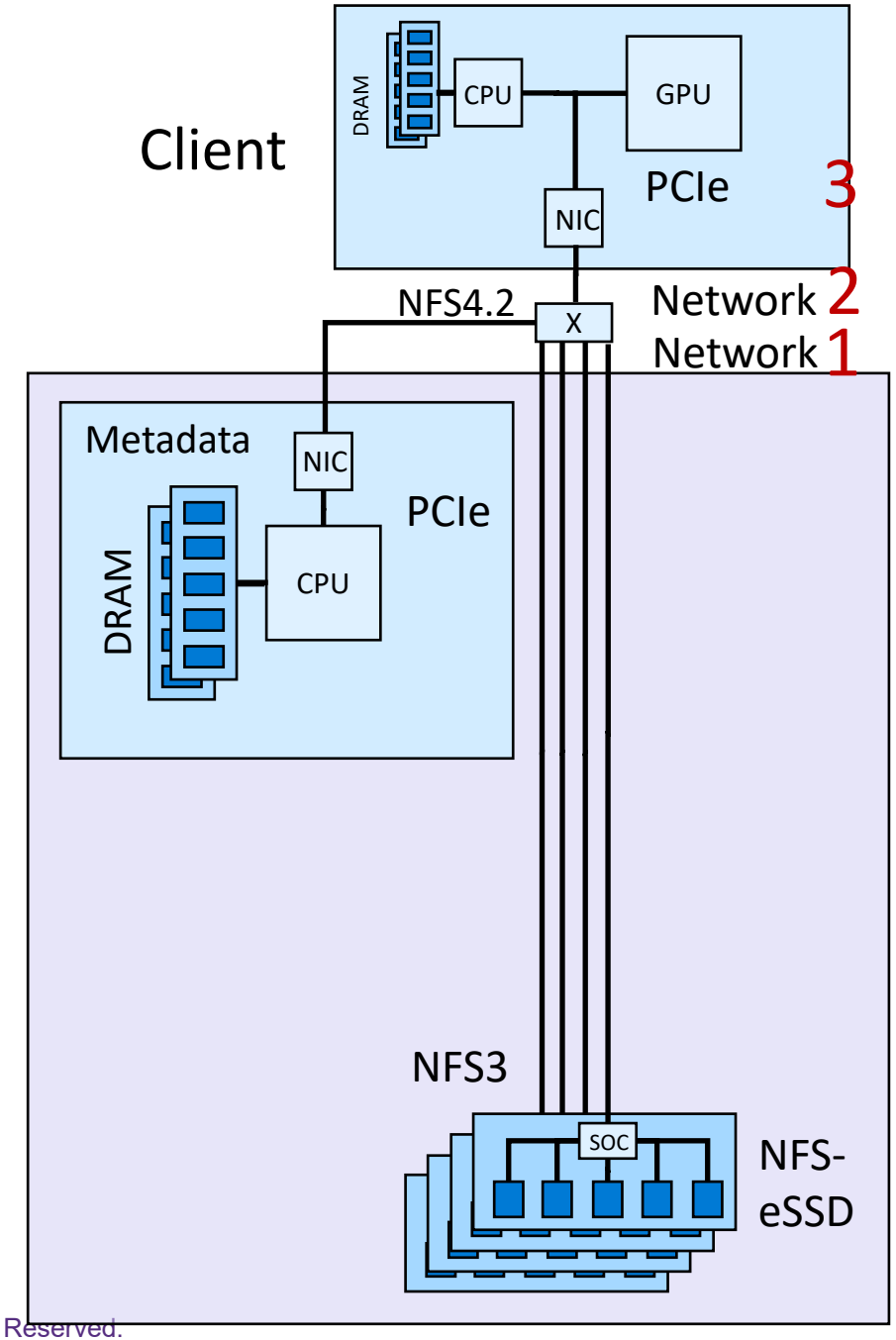
Much wider dynamic range of scale

- Scale up (hyperscale)
- Scale down (SOHO, maybe on USB-C)

Enables Computational Storage

- Compression, deduplication, encryption, erasure / error coding, copy / clone, filter, search, join, map reduce, etc. can be offloaded to the SSD now that it understands file layout

File offset to flash address mapping



Why Now

AI/ML workloads demanding efficient performance

Data governance / cloud computing needs orchestration

Flash performance can easily saturate PCIe/Ethernet

E1.S and other form factors (density and power)

64-bit processor IP availability

Processor performance density

IPv6, RoCE

Embedded Linux with

High performance, lightweight filesystems (XFS)

High performance, lightweight NFS server (kNFSd)

Standardized Parallel NFS 4.2 Flexible Files

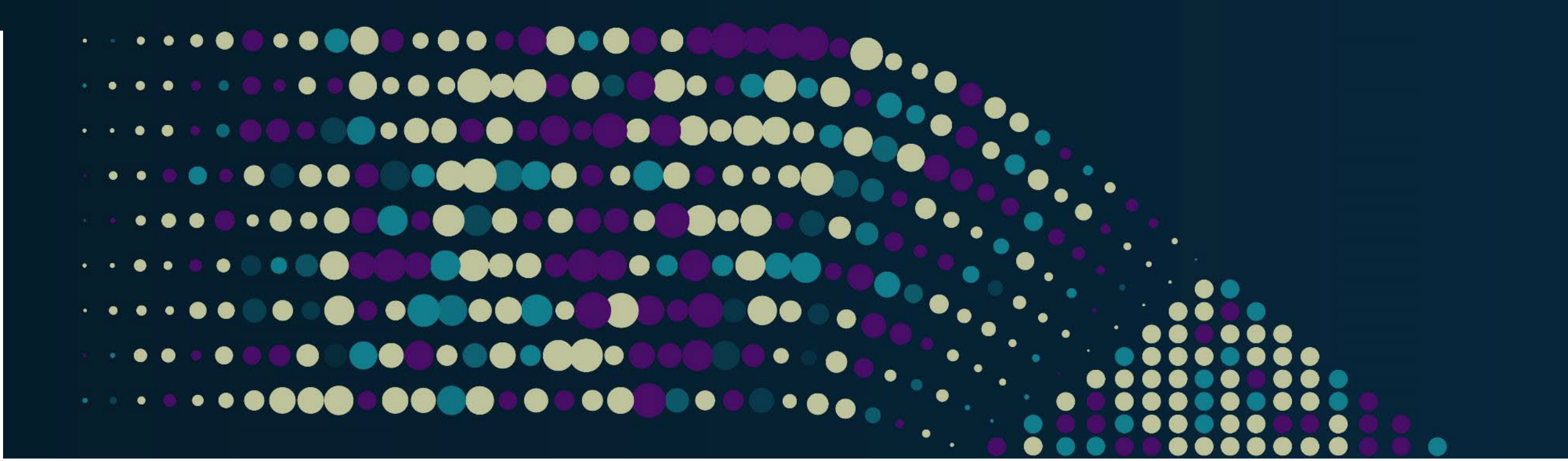
File offset to flash address mapping

STORAGE DEVELOPER CONFERENCE

SDC 23

BY Developers FOR Developers

Questions?



Please take a moment to rate this session.

Your feedback is important to us.

