SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

Adaptive Distributed NVMe over Fabrics Namespaces (ADNN)

Scott Peterson
Intel

Contents

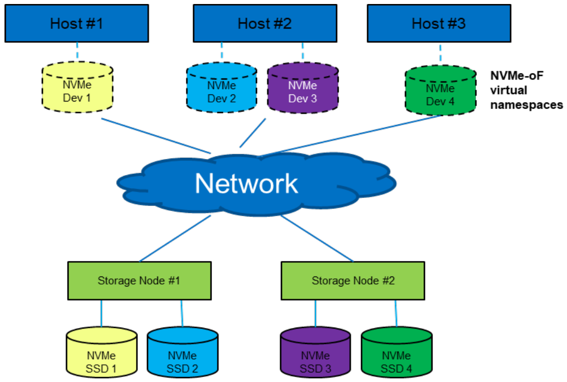
- NVMe-oF and scale-out storage
  - Why would you want to use NVMe-oF there, and what are the issues with that
- What ADNN does and how it improves NVMe-oF
- How ADNN works
  - Some familiar design patterns are reused
  - Some building blocks and a containing framework are defined
- Experimental results
  - ADNN reference implementation with a Ceph back end
- The ADNN SPDK reference implementation
  - Where to get it, and what happens next

2020 Storage Developer Conference. © Intel. All Rights Reserved.

Block Storage Landscape

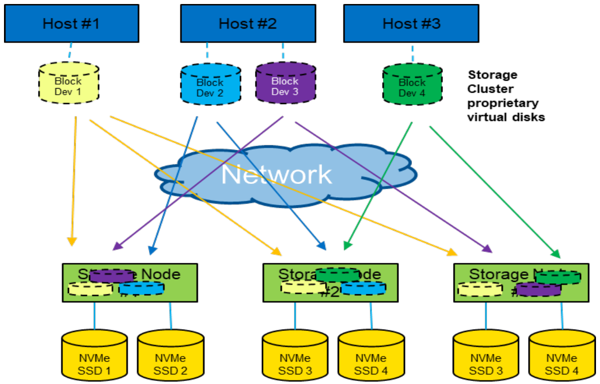
NVMe/NVMe-over-Fabrics

- Industry standard block interface for local and remote storage
- High Performance, Low Latency
- Standard Client driver/stack supported across all OSes
- Low footprint Client driver/stack (low CPU and memory utilization)
- SmartNIC offload friendly
- Single storage device or system back-end (point-to-point)
- Limited Storage Services



Scale Out Block Storage

- Scalable Capacity and Performance
- Highly Available, self-healing
- Rich Volume Services
- Custom Client driver/stack with custom protocol
- High footprint client driver/stack
- SmartNIC offload challenging

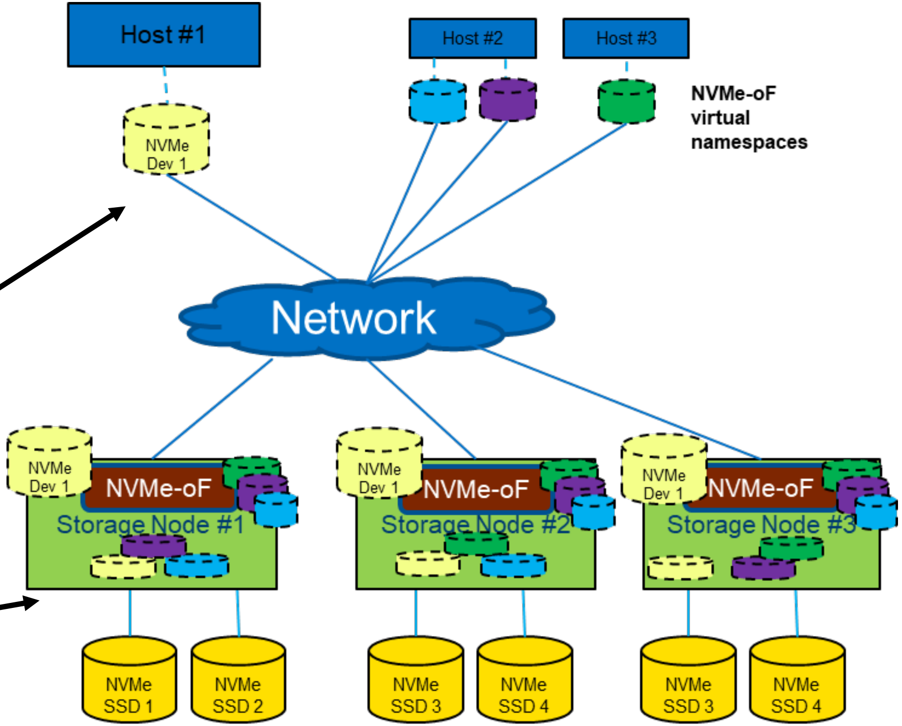


2020 Storage Developer Conference. © Intel. All Rights Reserved.

Adaptive Distributed NVMe-oF Namespaces: Best of Both Worlds

What if we combine NVMe-oF at the client with Scale out Storage as the storage back-end?

- Industry standard block interface for local and remote storage
- High Performance, Low Latency
- Standard Client driver/stack supported across all OSes
- Low footprint Client driver/stack (low CPU and memory utilization)
- SmartNIC offload friendly
- Scalable Capacity and Performance
- Highly Available, self-healing
- Rich Volume Services



2020 Storage Developer Conference. © Intel. All Rights Reserved.

ADNN Technology

ADNN Principles

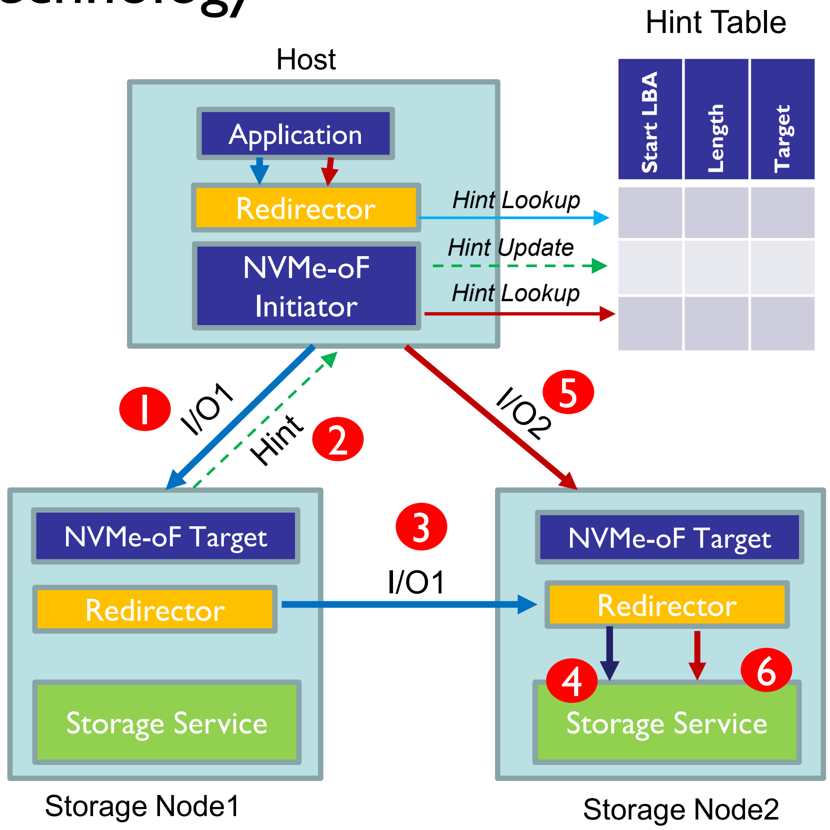
- Enables NVMe-oF volume exposed @host to be spread across several storage nodes
- Delivers IO for each volume *extent* *directly* to the storage node it's located on
- Adapts when volumes or its extents are moved

ADNN Features

- Flexible and extensible mapping functions to map volumes onto nodes
  - Covers a wide-range of storage back-ends
  - Can be easily extended to new storage back-ends
  - Hint/map table
- Learn volume/extent locations through in-line hints
  - Automatically adapt to changing volume and storage back-end topologies
  - Eliminate need for strongly coupled storage management agent at the host, making it easy for bare-metal host provisioning and life-cycle management
- Use standard NVMe-oF capabilities
  - No new standardization necessary
  - Backward compatible with existing NVMe-oF Initiators

ADNN Architecture

- Layer NVMe-oF target Gateway at each storage node
- Add redirector module to NVMe-oF at initiator and target
  - Compute volume/extent mapping for I/O steering
  - Send and Process Hints to maintain mapping consistency



2020 Storage Developer Conference. © Intel. All Rights Reserved.

ADNN System Components

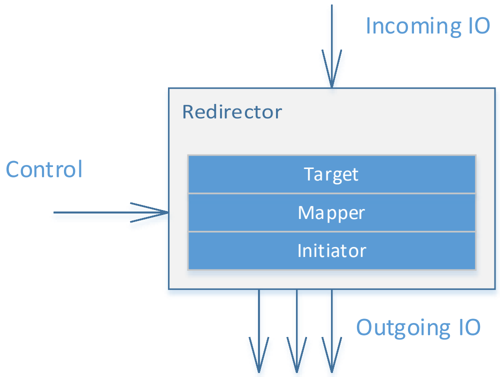
- Redirector
  - Basic SW/HW building block used at many points of a ADNN system
  - Selectively forwards IO to functionally equivalent targets based on block address
  - Learns better destinations from adjacent redirectors
  - Also used in storage nodes, with configuration from storage back-end, to route IO to underlying devices (or forward over fabric)
- Location hints
  - Unit of location information exchanged between redirectors
  - A variety of hint types mimic common data placement techniques, including striping and hashing
- Distributed Volume Manager (DVM) abstraction
  - Interface presented to ADNN hosts by a storage back end
  - Defines redirector roles (host or egress) and source of truth for the location of all a volume’s regions
  - Provides discovery information for stateless configuration of clients regardless of specific storage back-end

2020 Storage Developer Conference. © Intel. All Rights Reserved.

Redirector – The basic building block

A redirector chooses an IO target based on its starting LBA

- Possible targets revealed by Discovery Service
- Mapper table is generated from location hints
  - Hints are learned from targets by default
  - Selected redirectors receive hints via their control path
  - Redirectors inside the DVM get "authoritative" hints this way
    - Authoritative hints are never discarded or replaced by learned hints
- IO is forwarded between redirectors
  - Redirector in the host chooses the best known target
    - Based on its accumulated hints
    - If it has no hints it uses the first available target
  - Redirectors in storage nodes have better location information
    - The target chosen by the host may complete that IO by forwarding
- Redirectors learn from forwards, take direct path next time
  - If a redirector target forwards, it tells the initiator where it should have gone
  - Next IO to that LBA region probably takes direct path



2020 Storage Developer Conference. © Intel. All Rights Reserved.

Location hint essentials

Hints identify an LBA range, and a target

- Simplest hint sends all IO in LBA range to one place
  - Variants send just reads, or just writes
  - Other variants provide a list of alternative destinations (ordered by preference)
- Algorithmic hints define more complex computed mappings
  - Single striping hint locates many small strips
  - A single hashing hint locates all Ceph RBD image objects
- Redirectors combine overlapping hints
  - The most specific hint that matches an IO is applied
  - Short simple hints can describe exceptions to larger hints
- Hint retention is best effort
  - System designers will minimize the number of hints required
    - If few enough they may all be sent on host connect, and learning from forwards may not be required
  - If hosts can't retain all of them, some may be discarded
    - These will eventually be relearned
  - Hint discarding is like cache eviction: drop what's not helping
- Missing hints impact performance, not correctness
  - Optimal performance with complete correct map, but functional with incomplete or stale map

Simple hint
Start LBA
Length
Direction (R/W/both)
Target(s)

Striping hint
Start LBA
Length
Stripe size
Extent count
Target extent list

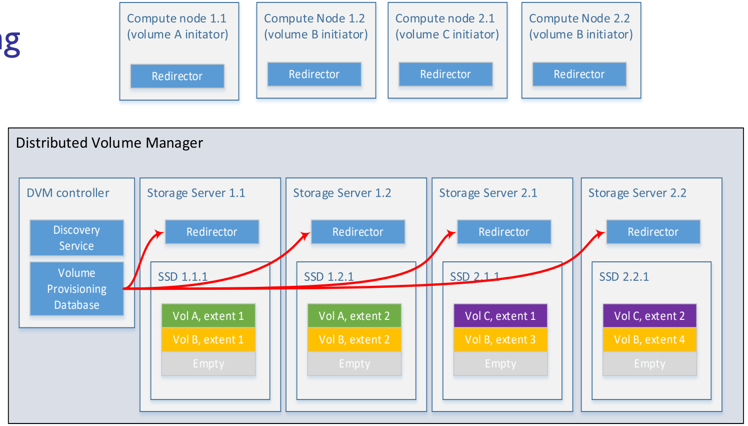
Hashing hint
Object size
Object name format
Hash function
Hash table log page

2020 Storage Developer Conference. © Intel. All Rights Reserved.

Distributed Volume Manager (DVM)

The boundary between ADNN clients and the storage cluster(s) they use

- Anything that supports block can be a ADNN DVM
  - DVM is essentially a generic distributed storage interface
- DVM interface defines a boundary for control and coupling
- Everything outside the DVM is loosely coupled
  - Hosts learn from any DVM the same way
    - Targets from a NVMe-oF discovery service
    - Location hints from those discovered targets
- Things inside the DVM may be tightly coupled
  - Storage cluster mechanisms for internal consistency
    - Logical Namespace (LN) allocation and placement, RAID, etc.
  - This includes the egress redirectors inside the DVM
    - They probably get everything via their control interface
    - The timing of changes is carefully managed to avoid loops
  - Egress redirectors may include system specific interfaces
    - Inside the DVM the cluster's rules and mechanisms apply



2020 Storage Developer Conference. © Intel. All Rights Reserved.

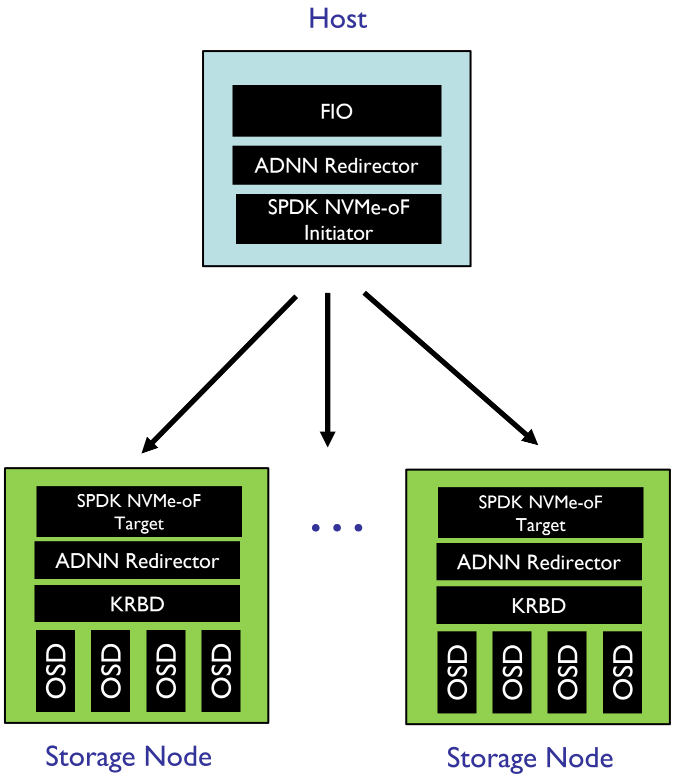
ADNN Reference Implementation

- ADNN redirector as an SPDK bdev
  - Building block for ADNN systems
  - Configured with one or more targets, and optional initial hints
  - Host redirector support:
    - Redirector can inherit its NGUID, size, and alignment from its default targets (which are egress redirectors in a DVM)
    - Intended to enable stateless host configuration
  - Egress redirector support:
    - Configured hints can be designated as “authoritative” (never replaced by a learned hint)
    - Authoritative hints in egress redirectors may cross namespaces (to map logical namespace extents to physical storage)
      - We avoid this outside the DVM to simplify system design
  - Bdev responds to Get Log Page passthrough commands for location hint log page (in vendor specific range)
  - Reads hints from its targets the same way
  - Support scripts for generating hash hints for RBD images are included
  - Redirector bdevs can be connected to each other in one SPDK app without NVMe-oF for testing

2020 Storage Developer Conference. © Intel. All Rights Reserved.

ADNN/Ceph PoC

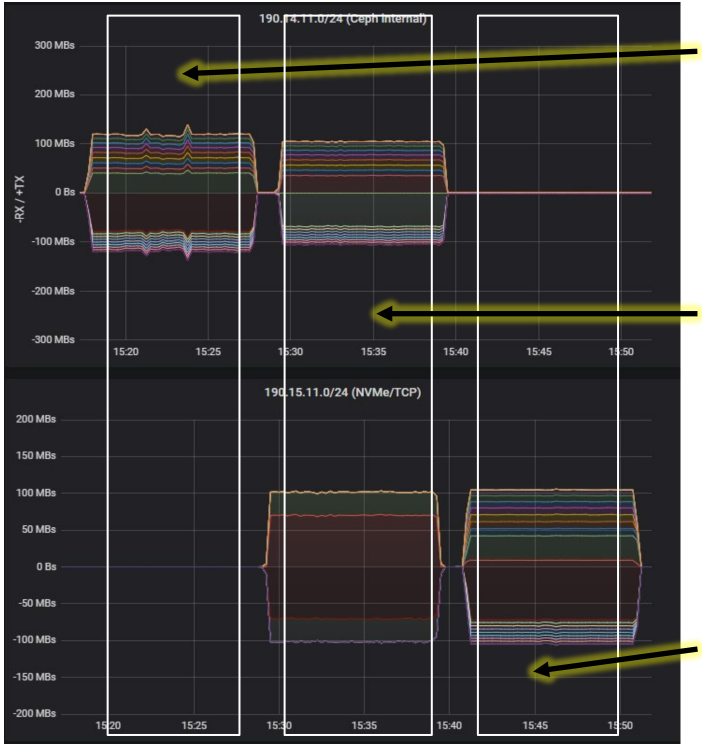
- Setup
  - Virtual cluster (8 OSD nodes, 32 OSDs, 3 monitors, 1 client)
  - Fio to ADNN redirector bdev in host
  - Host redirector has hash hint for an RBD image
    - Image is in **unreplicated pool** to simplify traffic analysis
  - NVMe/TCP from host to each OSD node
    - NVMe/TCP and Ceph use different networks
  - NVMe/TCP target namespaces are all redirector bdevs
  - Redirectors in targets send all IO to RBD
    - KRBD used here, could be RBD bdev
- PoC Goals
  - Show hash hint delivers all host IO to the correct OSD node
    - We plot network traffic on both networks for all nodes
    - We compare the traffic in three cases:
      - Normal RBD client in host
      - NVMe/TCP to target & RBD client in one storage node
      - ADNN redirector with hash hint to targets in all OSD nodes
    - Show nothing is forwarded by Ceph in ADNN hash hint case
  - Show host CPU usage is lower with ADNN than with RBD client
  - Show latency is lower with ADNN than with the gateway.



2020 Storage Developer Conference. © Intel. All Rights Reserved.

ADNN/Ceph PoC: Gateway Forwarding Eliminated

- Top graph is traffic on Ceph network
- Bottom graph is traffic on NVMe-oF network
- RBD Baseline case (left):
  - Traffic on only the Ceph network
- RBD Gateway case (center) shows:
  - All IO traverses NVMe-oF network to gateway **PLUS**
  - Most IO also traverses the Ceph network (two hops)
- ADNN case shows (right):
  - All IO traverses only the NVMe-oF network
  - No traffic on Ceph network
  - ADNN delivers each IO to the correct OSD node.
- Latency
  - QD=1
    - Read latency about the same in all 3 cases
    - ADNN increases write latency ~1% over gateway
  - QD=64
    - ADNN read and write latency ~40% lower that KRBD
    - Surprising, but repeatable in this test config.
    - Not completely understood.



RBD Baseline:
IO flows from client to all OSDs via Ceph network.

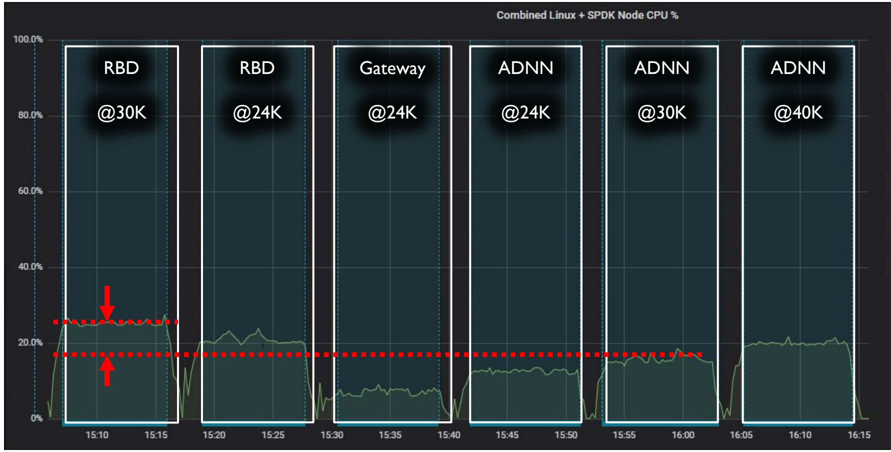
RBD Gateway:
IO flows from client to gateway via NVMe-oF, then to each OSD over Ceph network

ADNN:
IO flows from client to each OSD via NVMe-oF

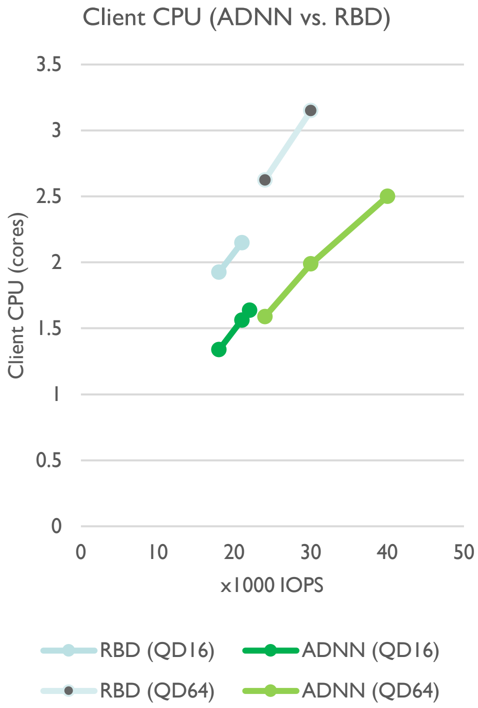
2020 Storage Developer Conference. © Intel. All Rights Reserved.

ADNN/Ceph PoC: Client CPU Usage Results

- Test Case: 4K random 70/30 read/write, QD=64
- ADNN saves >50% of one core @30K IOPS (measured as ~6.5% of 8 cores below)
  - 5 cores @300K (projected)
  - Expected to be higher with increasing performance and scale
    - IOPS limits here chosen to stay within virtual testbed capabilities
- ADNN overhead: 7% of one core on each Ceph OSD node
  - This includes the RBD overhead moved from the host to the 8 OSD nodes



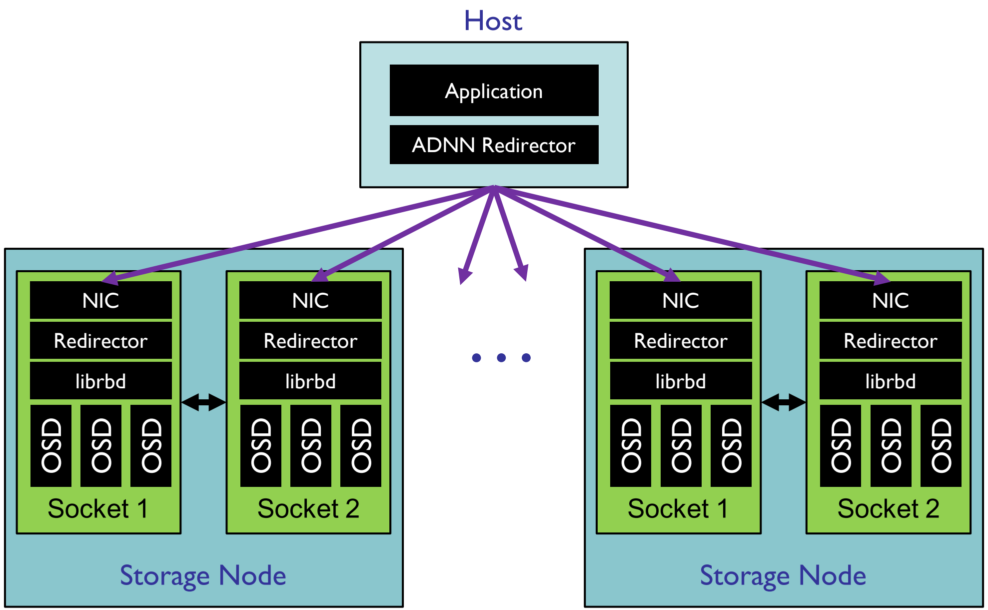
	max	avg	current
nemo-osd-207	40.292%	17.863%	0.700%
nemo-osd-208	34.140%	20.269%	0.700%
nemo-osd-202	33.551%	19.373%	0.563%
nemo-osd-201	33.142%	20.942%	0.738%
nemo-osd-206	32.506%	19.808%	0.546%
nemo-osd-203	32.504%	19.409%	0.588%
nemo-osd-205	31.348%	17.512%	0.438%
nemo-osd-204	30.002%	16.756%	0.554%
nemo-client-101	27.490%	15.036%	0.125%
nemo-mon-252	25.800%	1.629%	0.550%
nemo-mon-253	18.400%	1.672%	1.000%
nemo-mon-251	17.050%	2.781%	2.225%



2020 Storage Developer Conference. © Intel. All Rights Reserved.

NUMA placement with ADNN

- Use multiple egress redirectors in multi-socket storage nodes
  - One target & redirector per NUMA node (that has storage and a NIC)
  - DVM maps storage devices / OSDs to the redirector in its NUMA node
  - Hosts see one ADNN target per NUMA node in DVM
  - 2-socket Ceph OSD nodes shown



2020 Storage Developer Conference. © Intel. All Rights Reserved.

Use ADNN with your distributed storage

Now is the time to help define the form and direction ADNN takes

- Get ADNN from SPDK Gerrit
  - https://review.spdk.io/gerrit/c/spdk/spdk/+/4325 (based on SPDK v20.07)

```
git clone https://review.spdk.io/gerrit/spdk/spdk
git fetch "https://review.spdk.io/gerrit/spdk/spdk" refs/changes/25/4325/1
git checkout -b adnn FETCH_HEAD
```

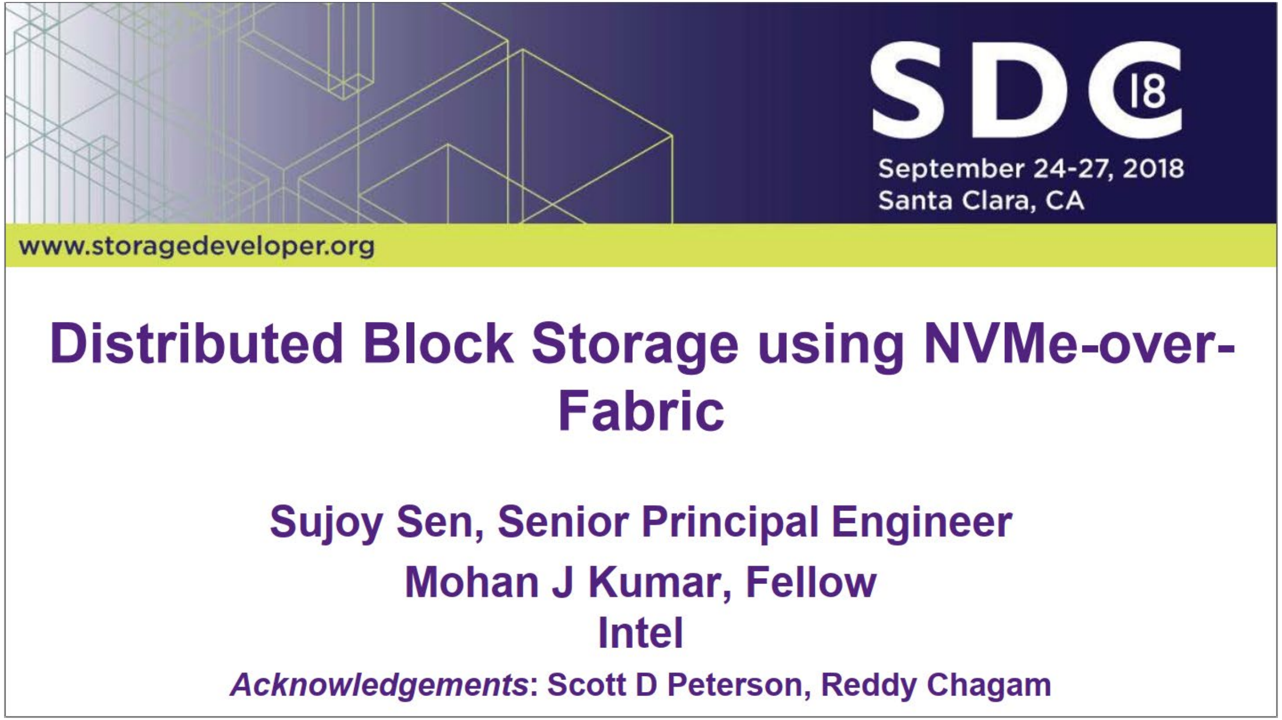
- Try an ADNN example config
  - Use Ceph as a back-end with the included scripts (see module/bdev/redirector/scripts)
  - Construct a simple DVM with the SPDK CLI
    - e.g. concatenate block devices from a couple nodes into an ADNN logical namespace
- Connect ADNN to your distributed storage system
  - Can your system's placement be described with ADNN hints?
- Help refine ADNN into something production ready
  - Behavior details and additional hint types still being refined and defined.
  - What must change before ADNN could be standardized?
  - Comment on the patch, or email me (scott.d.peterson@intel.com)

2020 Storage Developer Conference. © Intel. All Rights Reserved.

Please take a moment to rate this session.

Your feedback matters to us.

ADNN is the result of this prior work



2020 Storage Developer Conference. © Intel. All Rights Reserved.

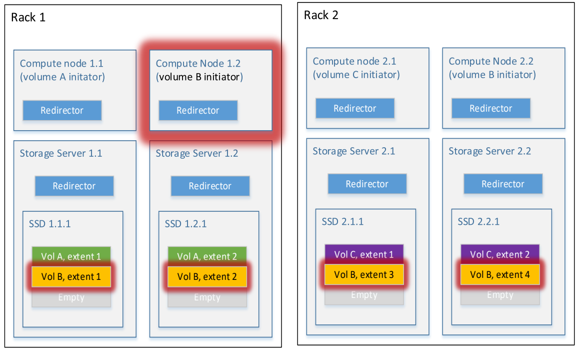
Backup

Example of a redirector learning from hints

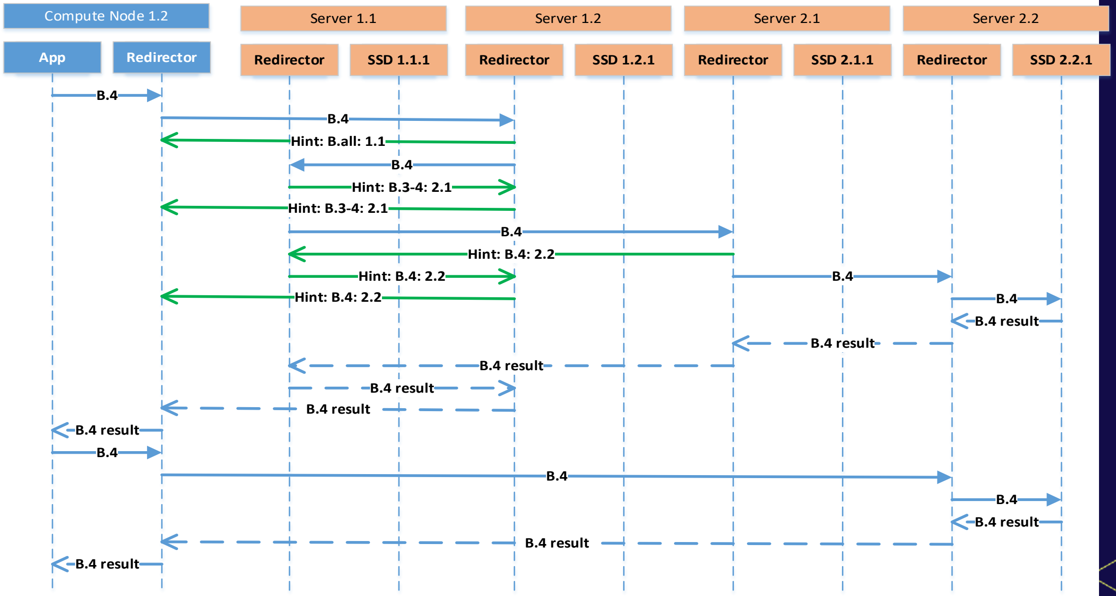
Contrived example to illustrate learning from forwards

- 3 forwards on the first IO to B.4
  - Real systems probably have one forward at most
- Zero forwards on the second IO to B.4

1 - Nodes & Extents



3 – IO forwarding path



2 - Initial hint tables

Server 1.1		Server 1.2		Server 2.1		Server 2.2	
Vol B.1	SSD 1.1.1	Vol B.all	Server 1.1	Vol B.all	Server 1.1	Vol B.all	Server 1.1
Vol B.2	Server 1.2	Vol B.2	SSD 1.2.1	Vol B.3	SSD 2.1.1	Vol B.4	SSD 2.2.2
Vol B.3-4	Server 2.1			Vol B.4	Server 2.2		

4 - Resulting hint tables (learned hints in green)

Compute 1.2		Server 1.1		Server 1.2		Server 2.1		Server 2.2	
Default	Server 1.2	Vol B.1	SSD 1.1.1	Vol B.2	SSD 1.2.1	Vol B.3	SSD 2.1.1	Vol B.4	SSD 2.2.2
Vol B.all	Server 1.1	Vol B.2	Server 1.2	Vol B.3-4	Server 2.1	Vol B.4	Server 2.2		
Vol B.3-4	Server 2.1	Vol B.3-4	Server 2.1	Vol B.4	Server 2.2				
Vol B.4	Server 2.2	Vol B.4	Server 2.2						

2020 Storage Developer Conference. © Intel. All Rights Reserved.