

Computational Storage Drives Compute Where The Data Is

Mats Öberg, AVP, Marvell

Disclaimer

This presentation contains forward-looking statements within the meaning of U.S. federal securities laws that involve risks and uncertainties. Forward-looking statements include, without limitation, any statement that may predict, forecast, indicate or imply future events or achievements. Actual events or results may differ materially from those contemplated in this presentation. Forward-looking statements are only predictions and are subject to risks, uncertainties and assumptions that are difficult to predict, including those described in the “Risk Factors” section of our Annual Reports on Form 10-K, Quarterly Reports on Form 10-Q and other documents filed by us from time to time with the SEC. Forward-looking statements speak only as of the date they are made. Readers are cautioned not to put undue reliance on forward-looking statements, and no person assumes any obligation to update or revise any such forward-looking statements, whether as a result of new information, future events or otherwise.

Agenda

1. What is computational storage
2. Simple example
3. Standardization efforts
4. Computational storage drives
5. Existing solutions
6. Explore computational storage at FMS
7. Discussion

What is computational storage?

Storage Networking Industry Association (SNIA) definition

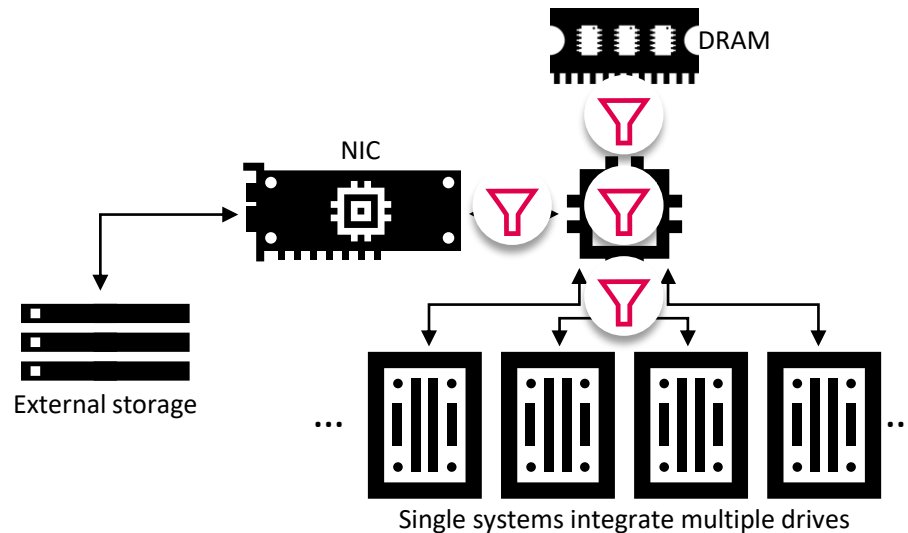
Computational storage is defined as architectures that provide computational Storage Services coupled to storage, offloading host processing, or reducing data movement

- Simply stated: Bring compute to data
 - Offload host
 - Reduce data movement
 - Avoid network bottlenecks
 - Reduce power



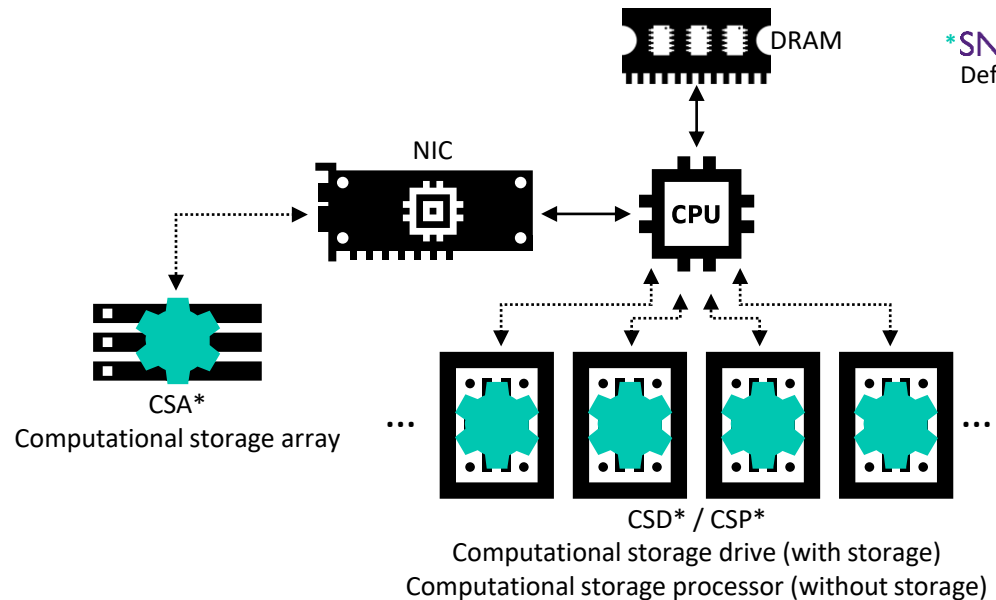
Traditional → computational storage

Traditional CPU-centric architecture



- Centralized compute
- DRAM throughput and capacity challenges
- Massive data movement (in-server & network)
- Fixed compute as workload capacity grows

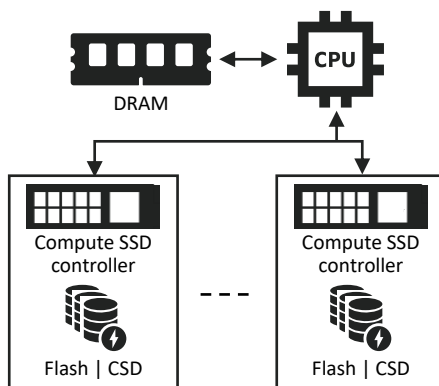
Computational storage architecture



- Parallelize compute (utilization ↑)
- Optimize DRAM throughput & utilization
- Minimize data movement (power/latency)
- Scale compute with capacity

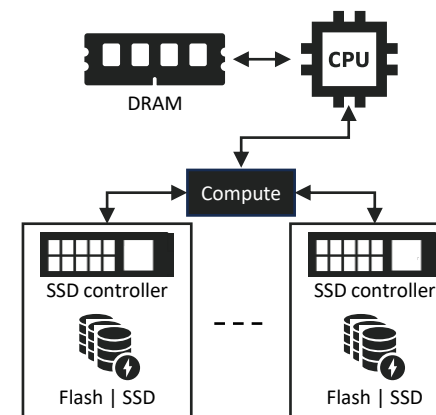
Computational storage drive vs. processor

CSD:
Computational
Storage Drive





- 👍 Scalable
- 👍 Reduced data movement
- 👍 Simple deployment model
- 👎 Additional cost per CSD
- 👎 Hard to optimize system

CSP:
Computational
Storage Processor



- 👍 Powerful and flexible. (E.g., both CPU and HW accelerator based CSPs in same box)
- 👍 Supports RAID and data striped over multiple drives
- 👍 Optimized to available bandwidth
- 👎 Not scalable with additional drives
- 👎 Additional component in data path

 APPLICATION
DATABASE, ANALYTICS Analytical Queries Data Format Conversion KV-Store CDN Media Scaling Transcoding SEARCH Fuzzy Search Filtering & Matching AI/ML Image Recognition Object Detection HPC/Genomics Highly parallelized workloads
 INFRASTRUCTURE
STORAGE Compression (GZIP, Z-standard) Deduplication (SHA-256) Erasure Coding (RS) Security (AES) Authentication (SHA) Error Checking (CRC, Checksum) Virtualization (VF, PF) Reliability (RAID)

Computational Storage Functions

- Fixed functions
 - E.g., compression, encryption, filtering, matching
- Programmable Functions
 - Dynamically reprogrammable by end user
- Application challenge: API integration, standardization
- Storage infrastructure: Transparent (e.g., compression) or API
- Value: Pipeline services on same computational storage drive (CSD)
 - Multiple impact to performance and scaling



Examples of compute operation

Deduplication

Filtering

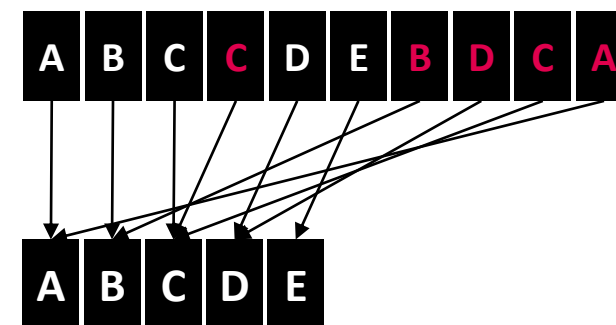
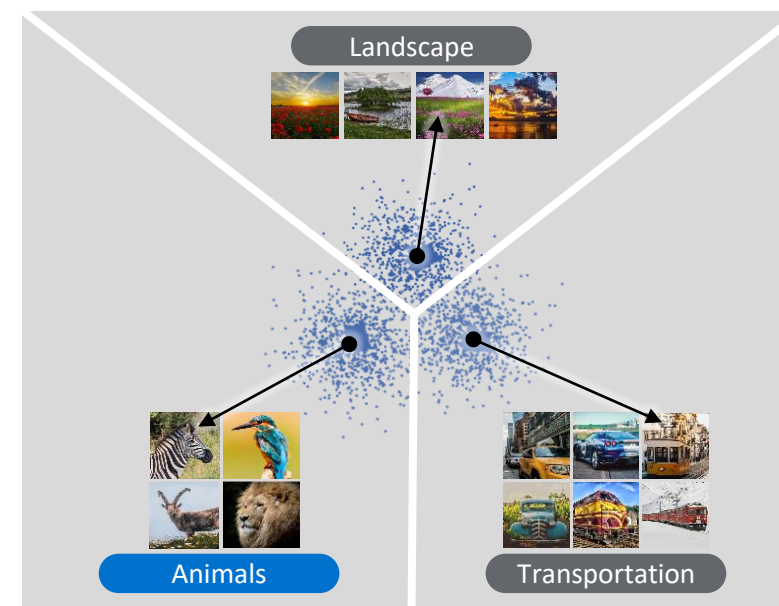
Aggregation

Classification

Name	Street	City	State	Email	Ph.
Andy Banner			NH		
Bruce Wayne			NY		
David Clapton			CA		
Jennie Adams			NY		
Tom Johnson			NY		
Tony Stark			TN		

Filter

Name	Street	City	State	Email	Ph.
Bruce Wayne			NY		
Jennie Adams			NY		
Tom Johnson			NY		



Computational storage example

- FMS 2018 demo

FMS 2018 demo

- AI at the SSD Storage Edge
- Two use cases
 - Host co-processor
 - Background pre-processing
- FPGA running Nvidia NVDLA



SSDs natively include all elements of a compute entity



Include additional entities (HW and/or FW/SW) for data processing acceleration



Efficiency at its most:

Power ● Performance ● Cost

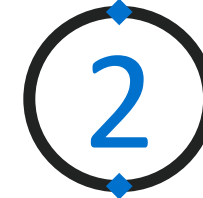
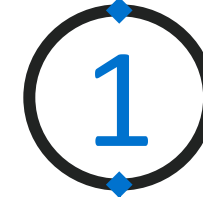
SSD THAT THINKS

USAGE EXAMPLE 1

Host co-processor

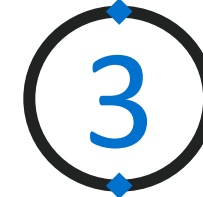
Storage device is used as host offload engine where proximity to the data is an advantage

Host generates a task / query



Task is sent to Storage device

Computation at Storage device



(Only) Metadata or Computation result is sent back to host

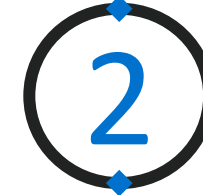
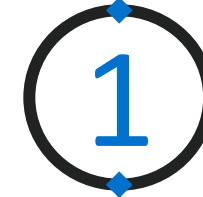
SSD THAT THINKS

USAGE EXAMPLE 2

Background data pre-processing

Storage device is used to pre-process stored data, and tag / index generation

Data is stored in the Storage device

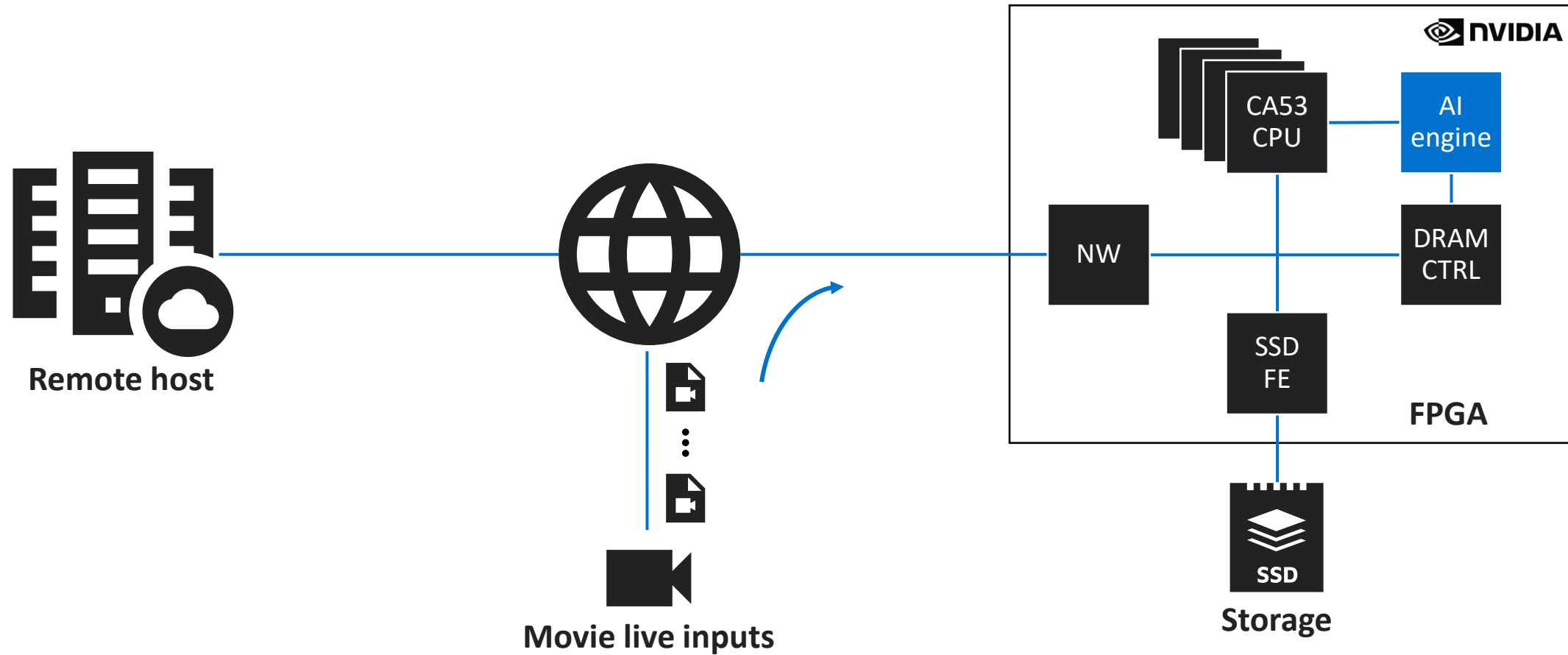


Storage device pre-processes the data and generates metadata / tags / indexes, etc.



Host retrieves metadata upon need

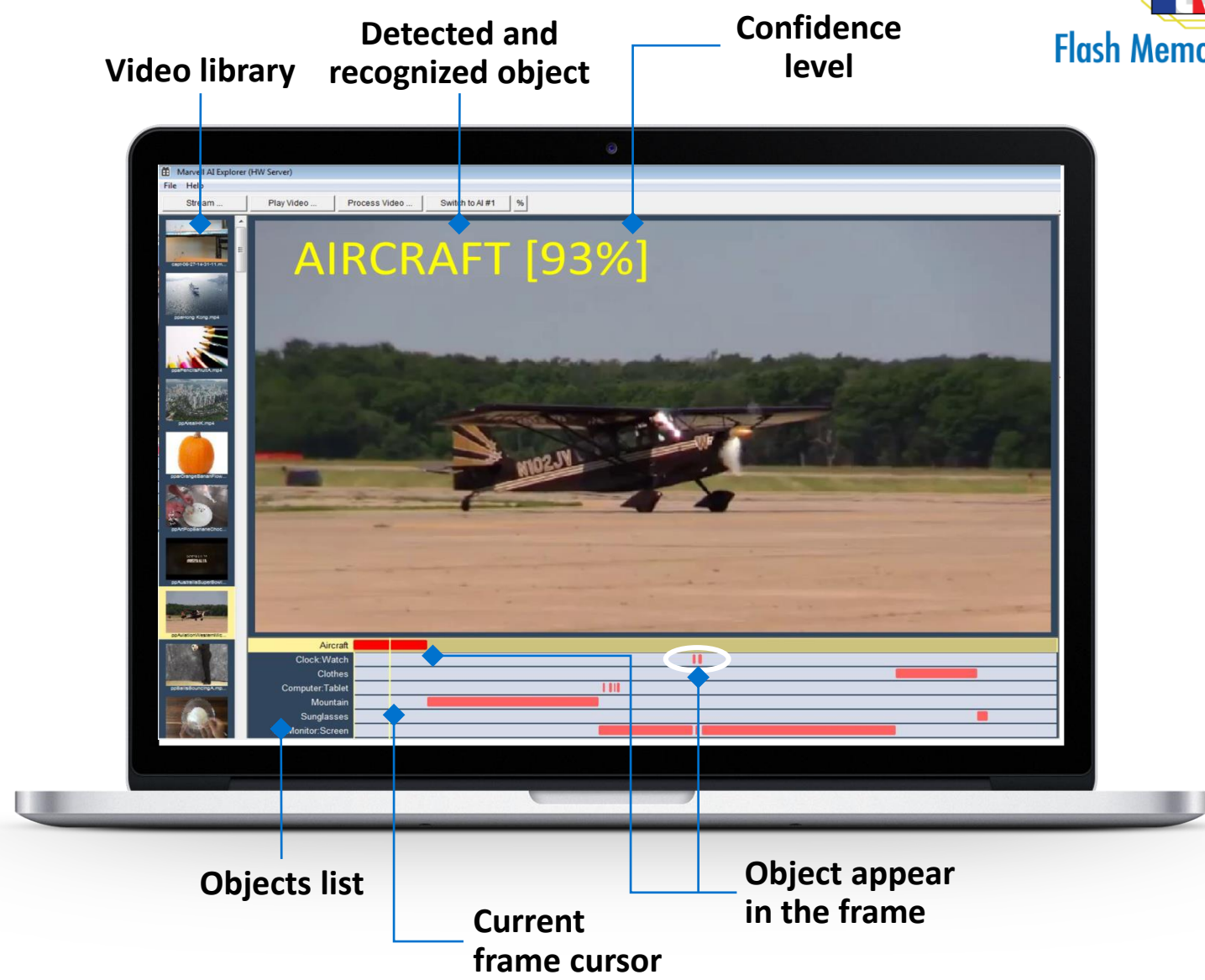
AI demo | Setup





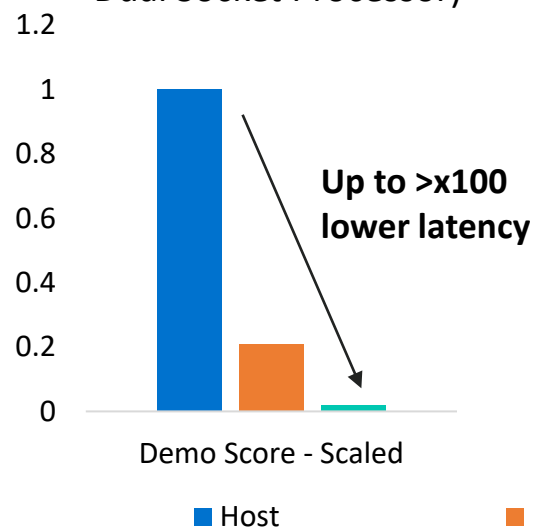
AI DEMO

SCREEN VIEW

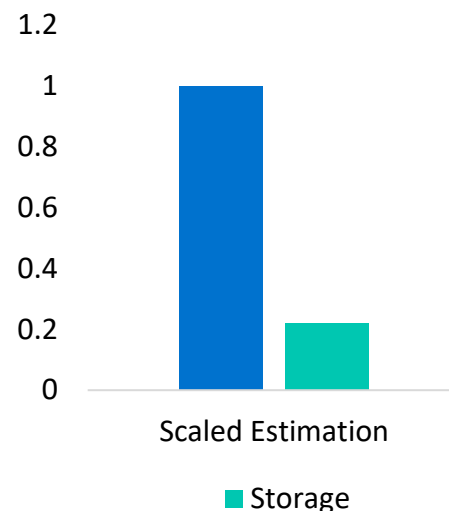


AI demo | Results

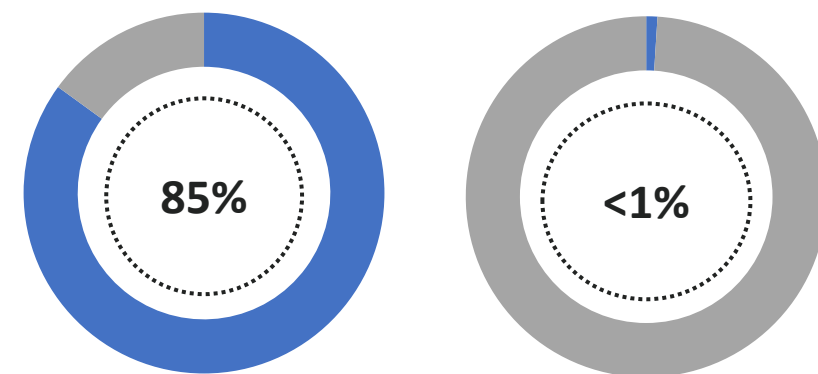
LATENCY RATIO
(24*8TB SSDs Database
Dual Socket Processor)



POWER RATIO
(24*8TB SSDs Database
Dual Socket Processor)



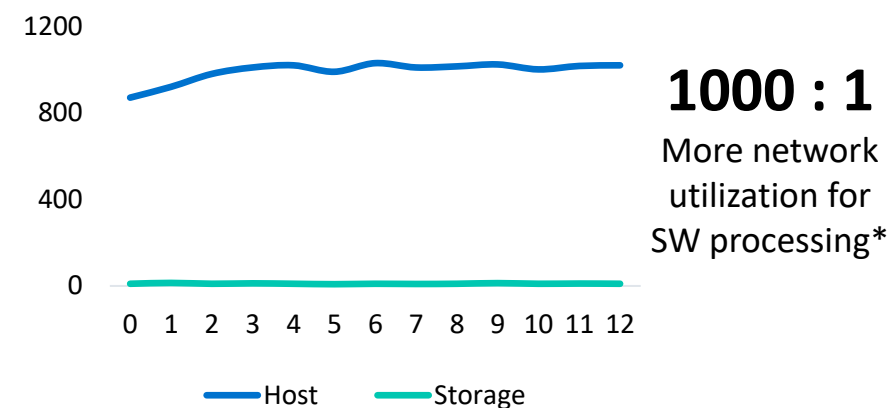
HOST CPU UTILIZATION



HOST PROCESSING

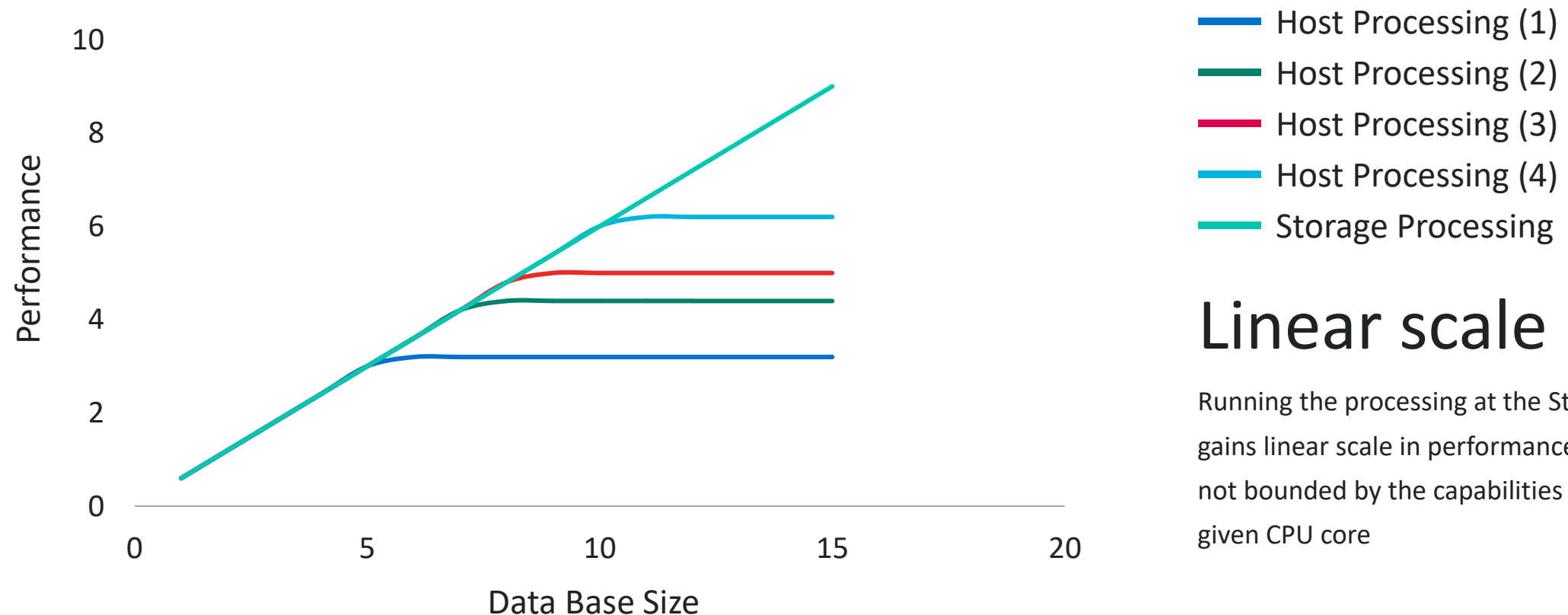
STORAGE PROCESSING

**NETWORK CAPACITY RATIO
OVER TIME**



* Per the demo database. May vary up or down with different databases

AI demo | Performance at scale



Linear scale

Running the processing at the Storage gains linear scale in performance, and is not bounded by the capabilities of a given CPU core

Standardizing computational storage



- SNIA computational storage technical working group
 - With permission from CS TWG Co-chairs
 - Bill Martin
 - Jason Molgaard



- NVMe computational storage
 - With permission from NVM Express Computational Storage Task Group:
 - Bill Martin
 - Kim Malone

SNIA | The continued growth of experience

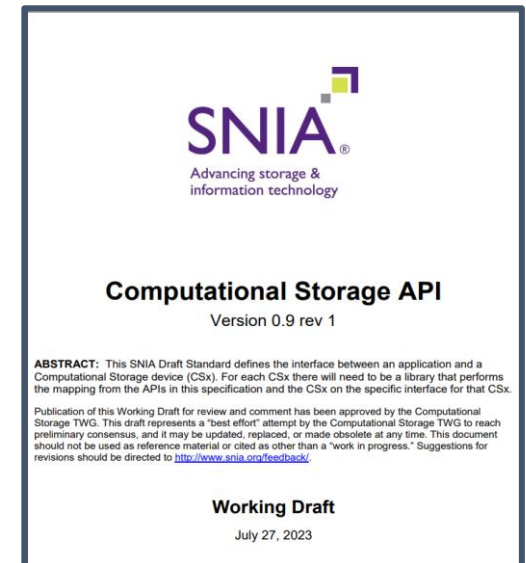
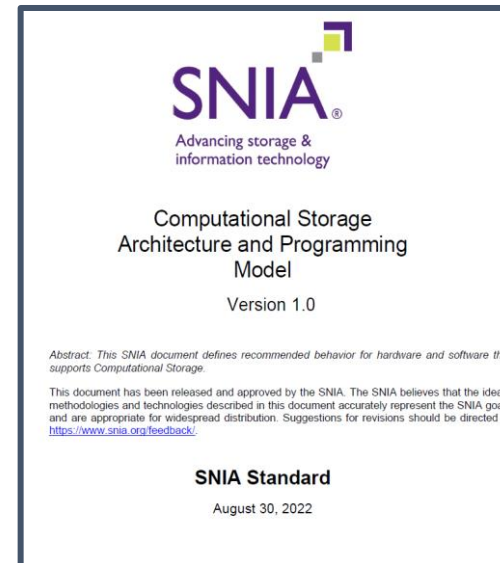
- CS TWG is continuing to see growth
 - **48** companies, **258** individual members
- Work within **SNIA** efforts
 - **CS SIG** – Webinars, Blogs, Events
 - **SDXI** – Sub-Group Collaboration
 - **Security** TWG – Addressing Security
- Collaborating with external groups
 - **NVM Express** – Computational Programs

48 Participating Companies - 258 Member Representatives



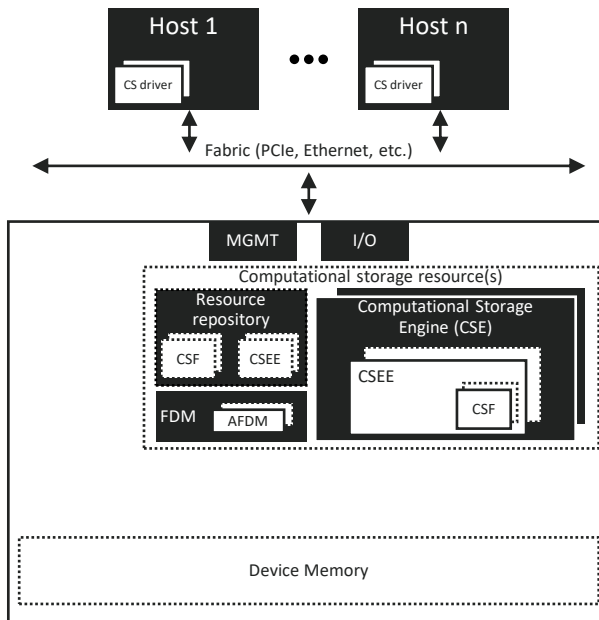
SNIA[®] | Current progress of TWG output

- **Architectural Document v1.0 has been released**
- v1.1 under development
 - Security enhancements for multiple tenants
 - Chaining of commands
 - Expansion of use cases
- **API v0.9 public review version also available**
- **API v1.0 under development**
 - Abort/reset handling
 - Device memory
 - NVMe computational programs support
 - Other miscellaneous updates

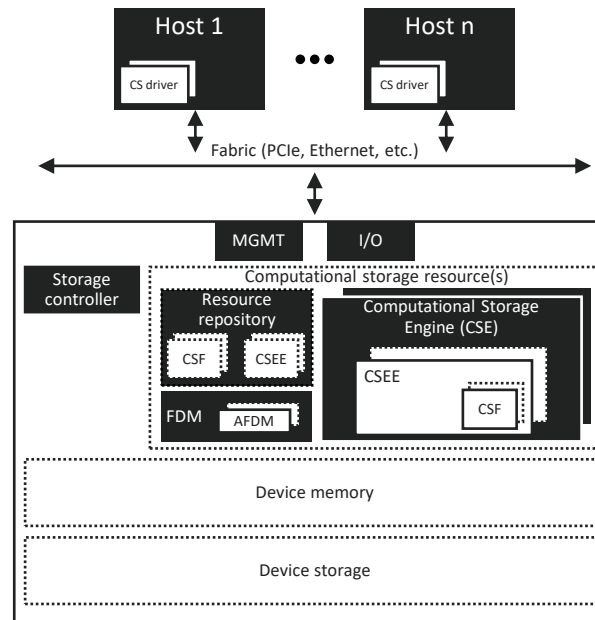


SNIA | Computational storage architecture

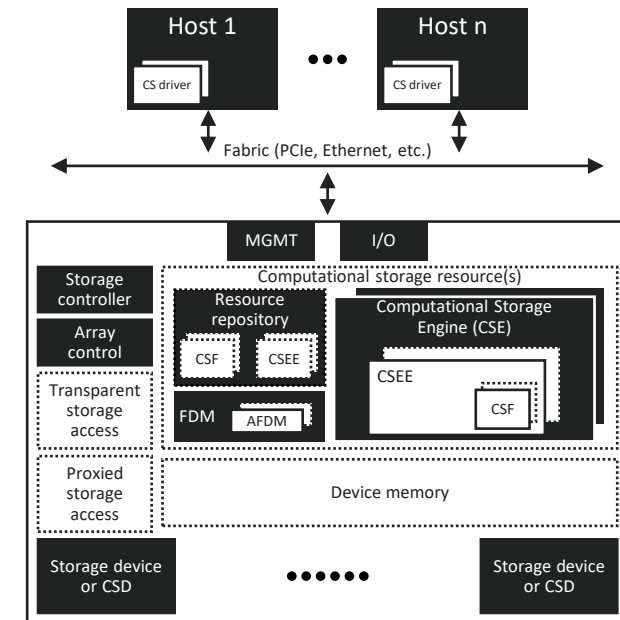
Computational Storage Processor (CSP)



Computational Storage Drive (CSD)



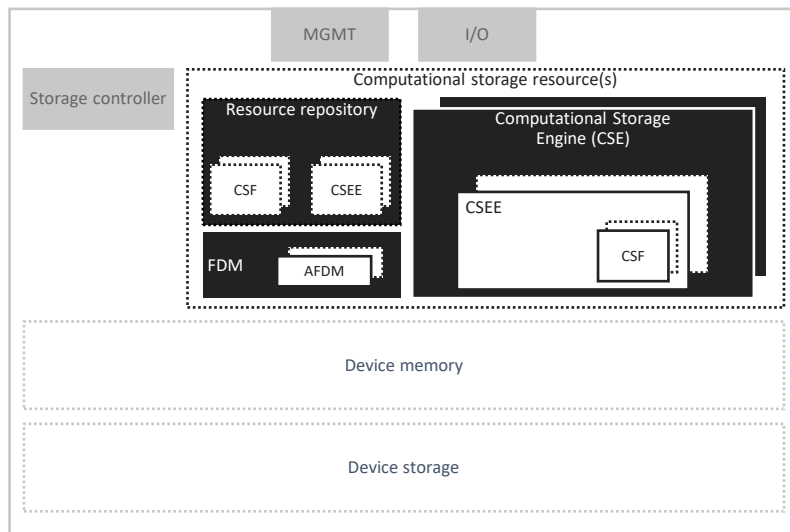
Computational Storage Array (CSA)



CSx = Computational storage **device** – CSP or CSD or CSA

SNIA | Deep dive of the CSx resources

Computational Storage Drive (CSD)



CSR | Computational Storage Resources are the resources available in a CSx necessary for that CSx to store and execute a CSF.

CSF | A Computational Storage Function is a set of specific operations that may be configured and executed by a CSE in a CSEE.

CSE | Computational Storage Engine is a CSR that is able to be programmed to provide one or more specific operation(s).

CSEE | A Computational Storage Engine Environment is an operating environment space for the CSE.

FDM | Function Data Memory is device memory that is available for CSFs to use for data that is used or generated as part of the operation of the CSF.

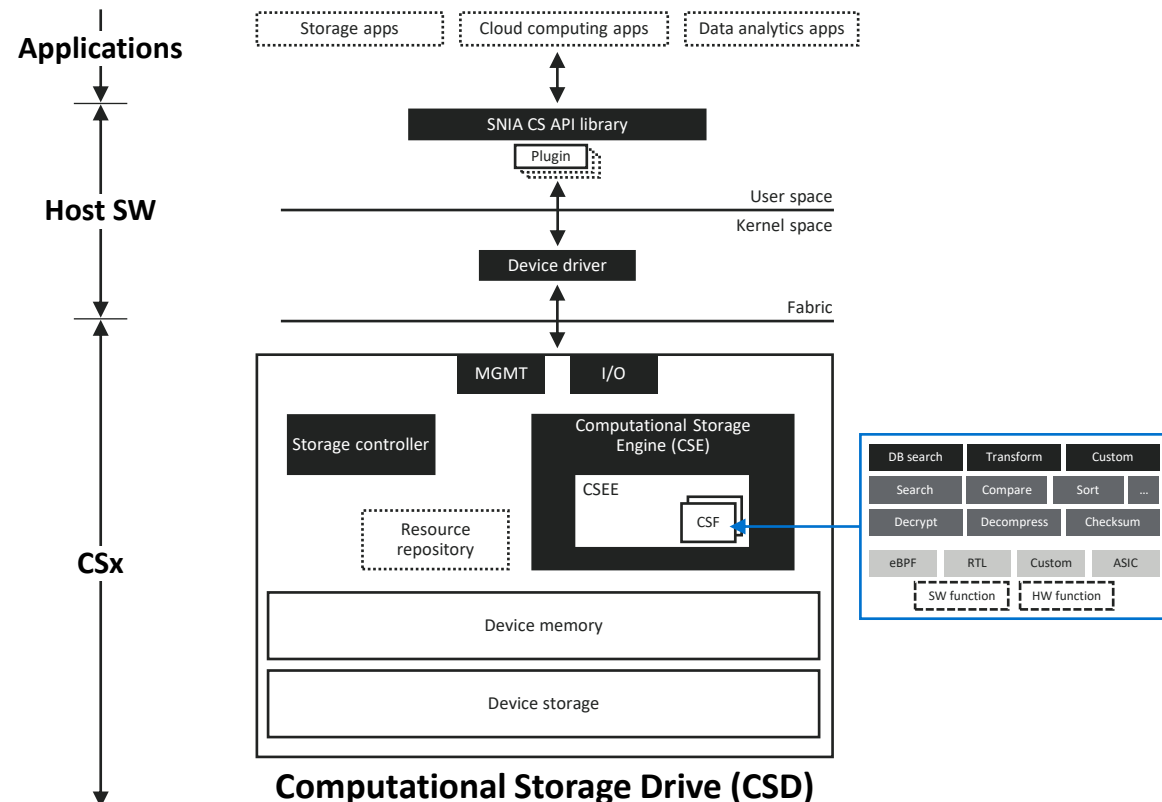
AFDM | Allocated Function Data Memory is a portion of FDM that is allocated for one or more specific instances of a CSF operation



API Use in Computational Storage

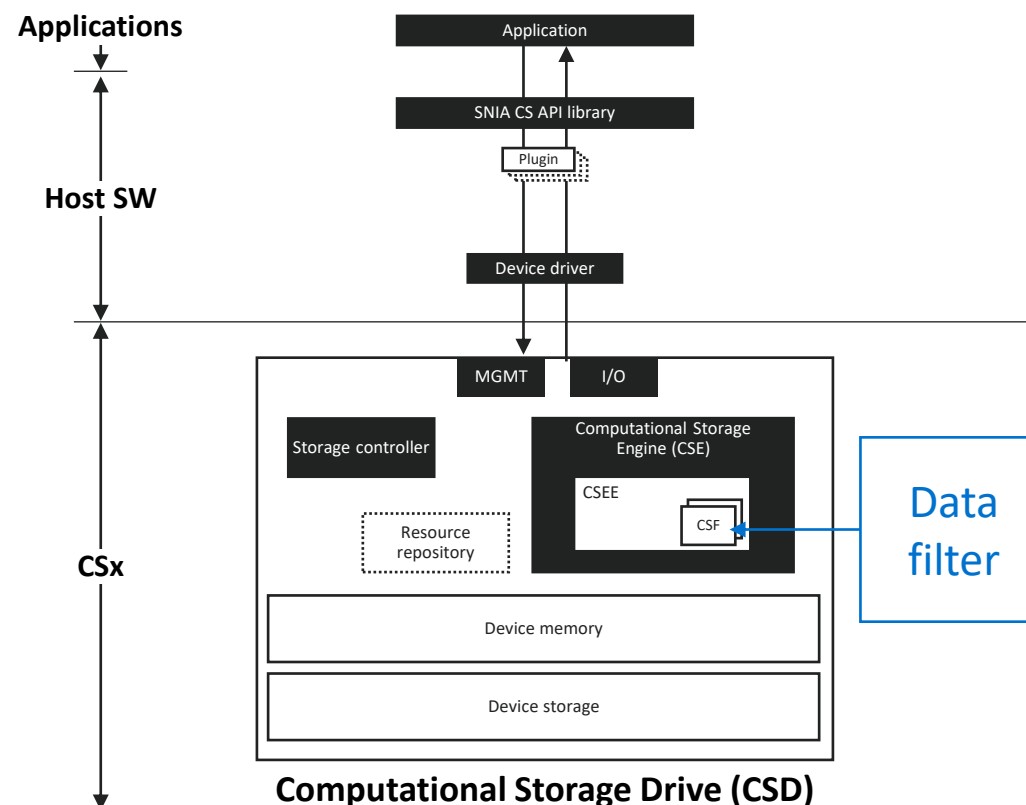
SNIA | Computational storage APIs

- One set of APIs for all CSx types
 - CSP, CSD, CSA
- APIs hide device details
 - Hardware, Connectivity (local/remote)
- Abstracts device details
 - Discovery
 - Access
 - Device Memory (mapped/unmapped)
 - Near Storage Access
 - Copy Device Memory
 - Download CSFs
 - Execute CSFs
 - Device Management
- Extensible Interface
 - Plugins connect CSx to abstracted APIs
 - Hides vendor specific implementation details
- APIs are OS agnostic

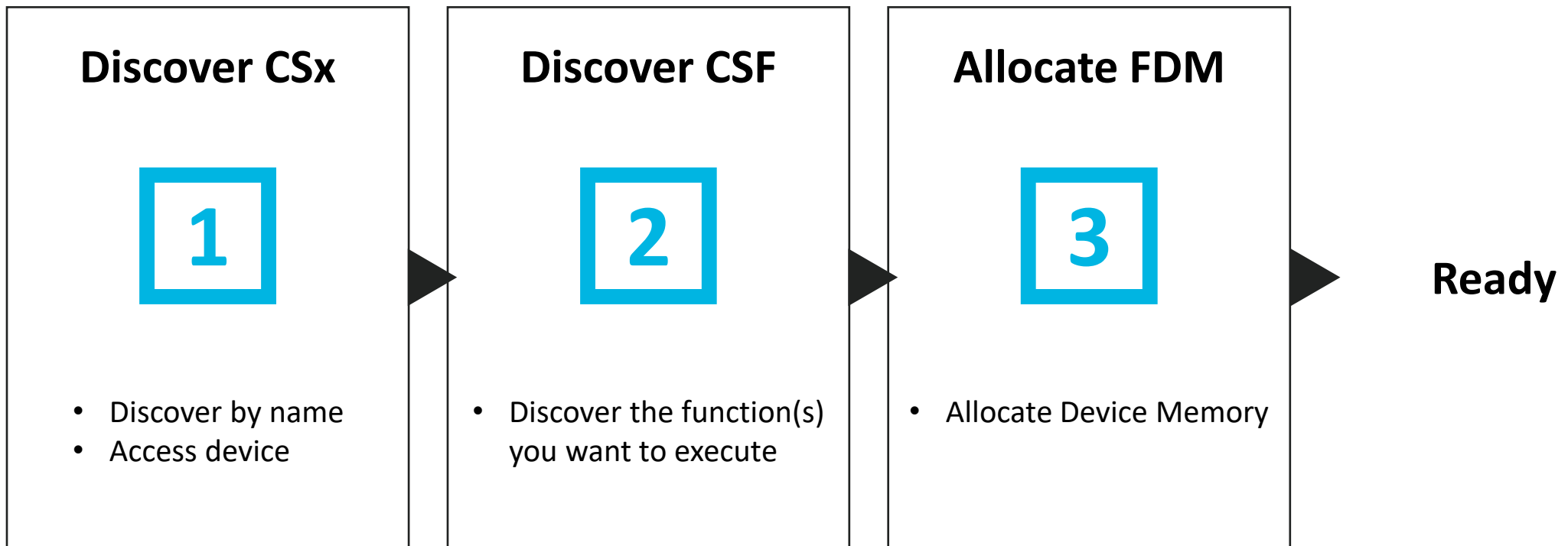


SNIA | Example use case – Abbreviated

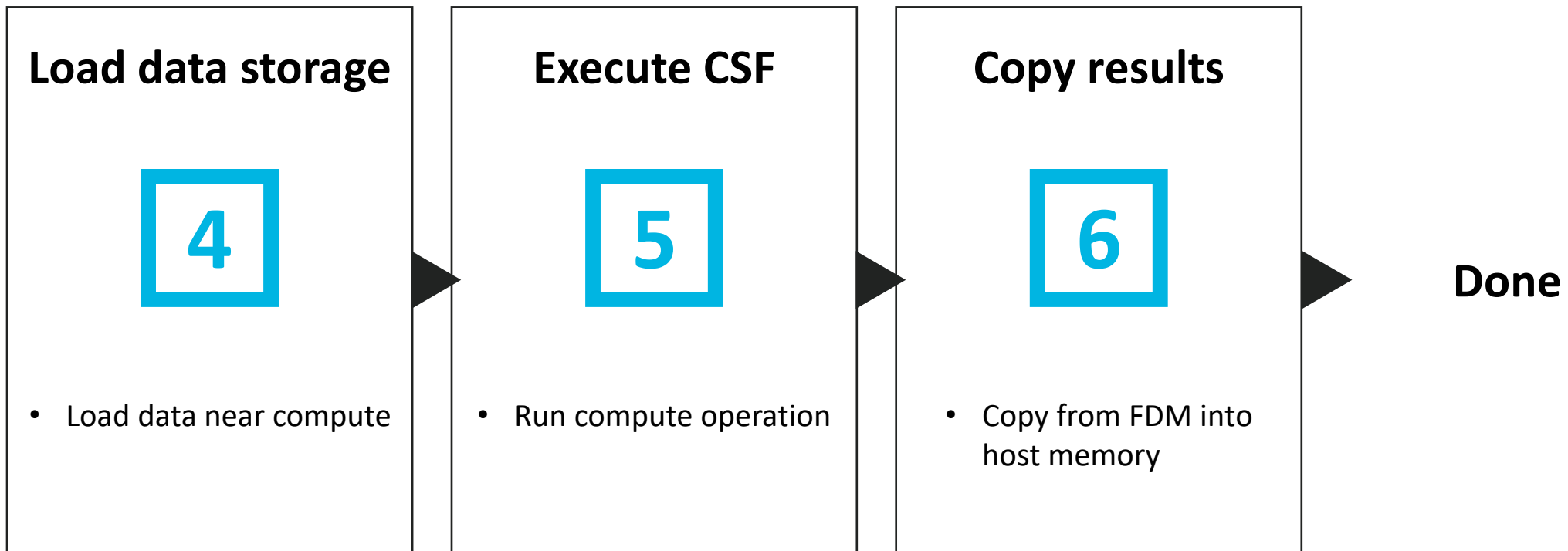
- Execute data filter CSF
 - Allocate Device Memory (FDM)
 - Load data from storage
 - Run data filter CSF on loaded data
 - Copy results to host memory



SNIA[®] | Prepare for computational storage (setup)



SNIA[®] | Perform computational storage I/O (run)



SNIA[®] | Usage summary

ONLY

6 steps

1

```
csGetCSxFromPath("my_file_path", &length, &csxBuffer);  
csOpenCSx(csxBuffer, &MyDevContext, &devHandle);
```

2

```
csGetCSFid(devHandle, "filter", &infoLength, &count, &csfInfo);
```

3

```
csAllocMem(devHandle, CHUNK_SIZE, &f, &afdmHandle1, NULL);
```

4

```
csQueueStorageRequest(storReq, storReq, NULL, NULL, NULL, &compVal);
```

5

```
csQueueComputeRequest(compReq, compReq, NULL, NULL, NULL, &compVal);
```

6

```
csQueueCopyMemRequest(copyReq, copyReq, NULL, NULL, NULL, NULL);
```



Bridging the Gap

Architecture to Implementation

Correlation of SNIA/NVMe terms

SNIA[®] terms

Computational Storage Engine (CSE)
Computational Storage Engine Environment (CSEE)
Resource Repository <ul style="list-style-type: none"> • Downloaded CSF and CSEE • Pre-loaded CSF and CSEE
Activation
Function Data Memory (FDM)
Allocated FDM (AFDM)
Device Storage

nvm EXPRESS[®] terms

Compute Namespace
Programs <ul style="list-style-type: none"> • Downloaded programs • Device-defined programs
Activation
Subsystem Local Memory (SLM) Namespace(s)
Not defined by NVMe (managed by host)
NVM Namespace(s)

Differences between SNIA and NVMe

SNIA[®]

- Defines CSEE
- CSF can directly access AFDM or Storage
- Supports an indirect model

nvm EXPRESS[®]

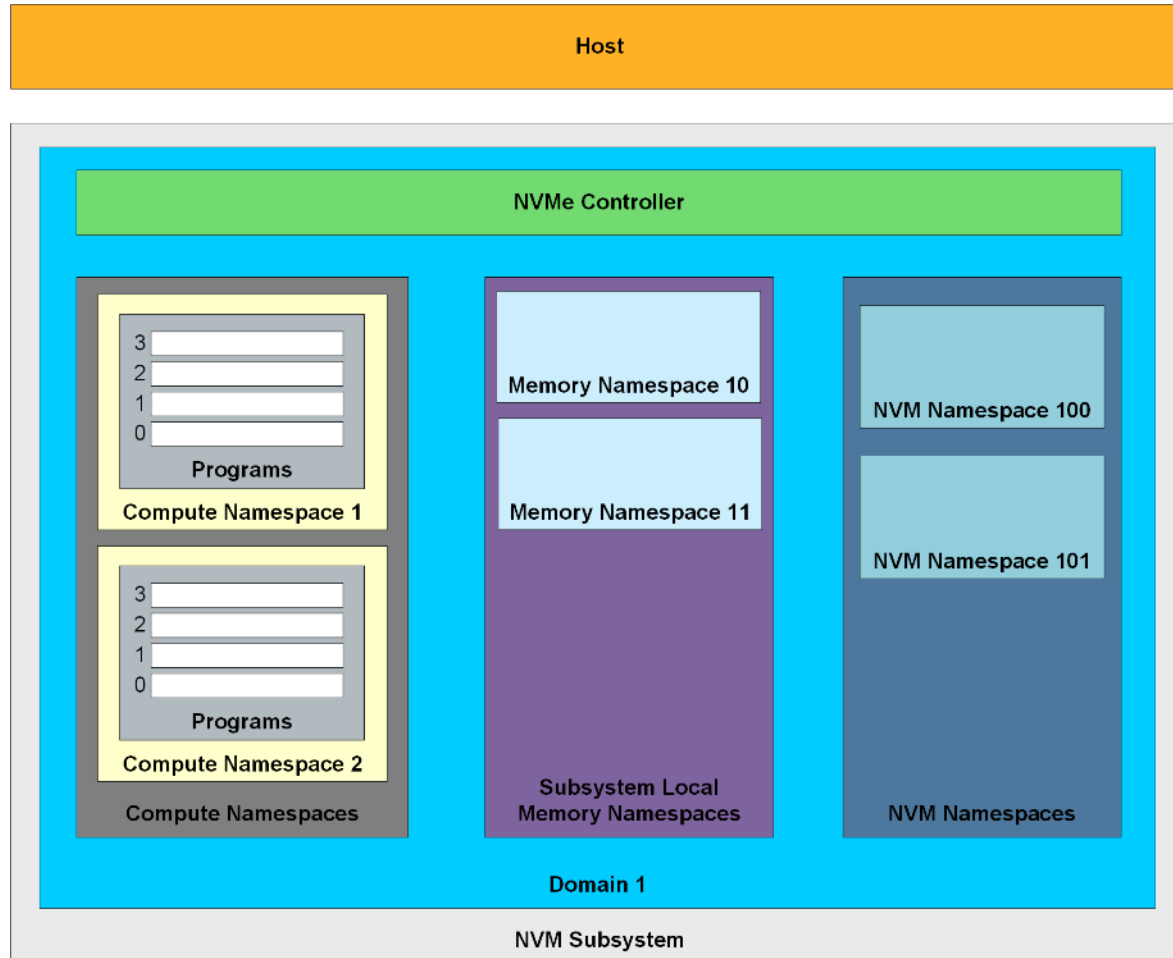
- A compute namespace contains both CSE and CSEE
- Program has access to memory described by a Memory Range Set
- Direct Execute command only



NVM Express[®] (NVMe[®]) Computational Storage

Sponsored by NVM Express[™] organization, the owner of NVMe Family of Specifications

Major Architectural Components



The NVMe Express[®] (NVMe[®]) computational storage architecture involves several types of namespaces:

- Compute namespaces (new)
- Memory namespaces (new)
- NVM namespaces
 - NVM, Zoned and Key Value namespaces

Compute Namespaces

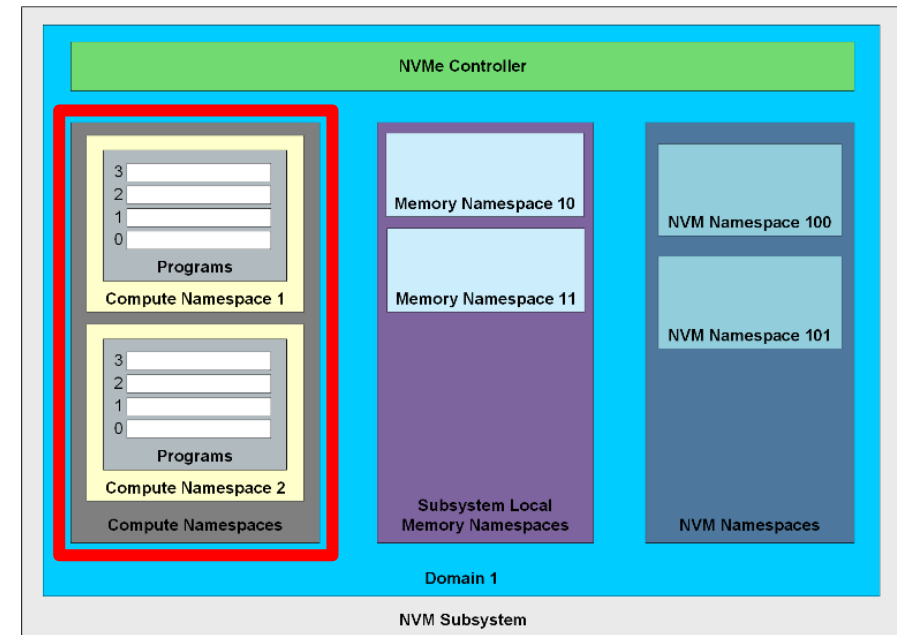
A compute namespace:

- Is a namespace in an NVM subsystem that is able to execute one or more programs
- Is a namespace that is associated with the Computational Programs I/O command set
- Is a namespace that contains compute resources

TP4091: Computational Programs

New Computational Programs I/O command set for compute namespaces

- New commands include:
 - Execute program
 - Load program
 - Activate program
 - Create/Delete Memory Range Set
- Provides log pages for program discovery

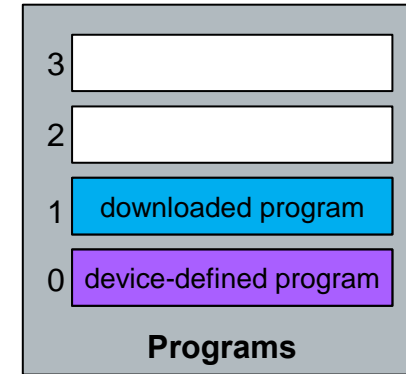


Flash Memory Summit

33 **nvm**
EXPRESS®

Computational Programs

- Conceptually similar to software functions
 - Called with parameters and run to completion
- Are addressed via a compute namespace program index
- May be identified by a globally unique program identifier
- Operate only on data in Subsystem Local Memory
- May be device-defined or downloadable
 - Device-defined programs
 - Programs provided at time of manufacture e.g., compression, encryption
 - Downloadable programs
 - Programs that are loaded to a Computational Programs namespace by the host
- A program may only be able to execute on a subset of the compute resources in an NVM subsystem
 - A program may be implemented in an ASIC
 - A program may be executed on a CPU core



Memory Namespaces

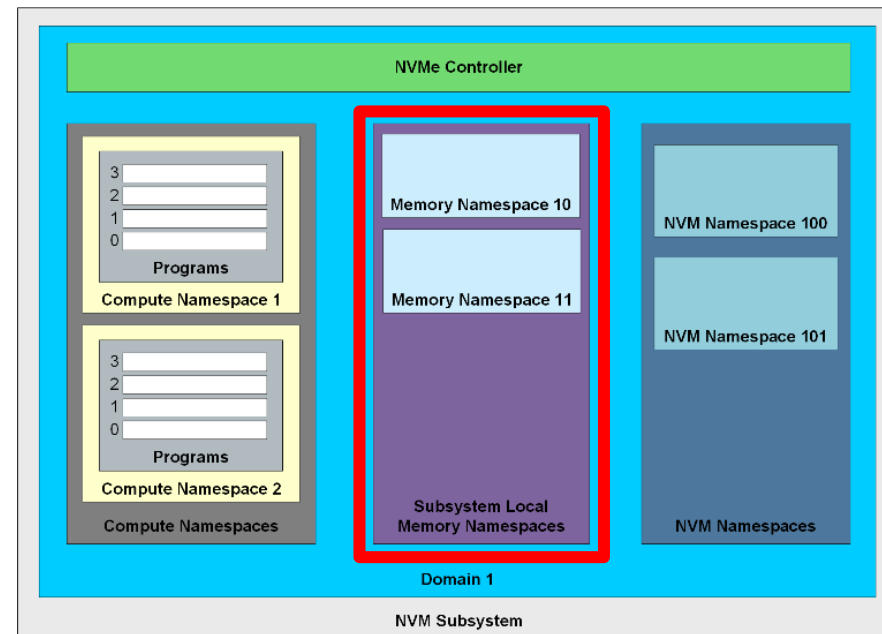
A memory namespace:

- Is a namespace in an NVM subsystem that provides host command access to memory in the NVM subsystem
- Is a namespace that is associated with the Subsystem Local Memory I/O command set
- Is a namespace used by the Computational Programs command set to provide access to SLM for program execution

TP4131: Subsystem Local Memory (SLM)

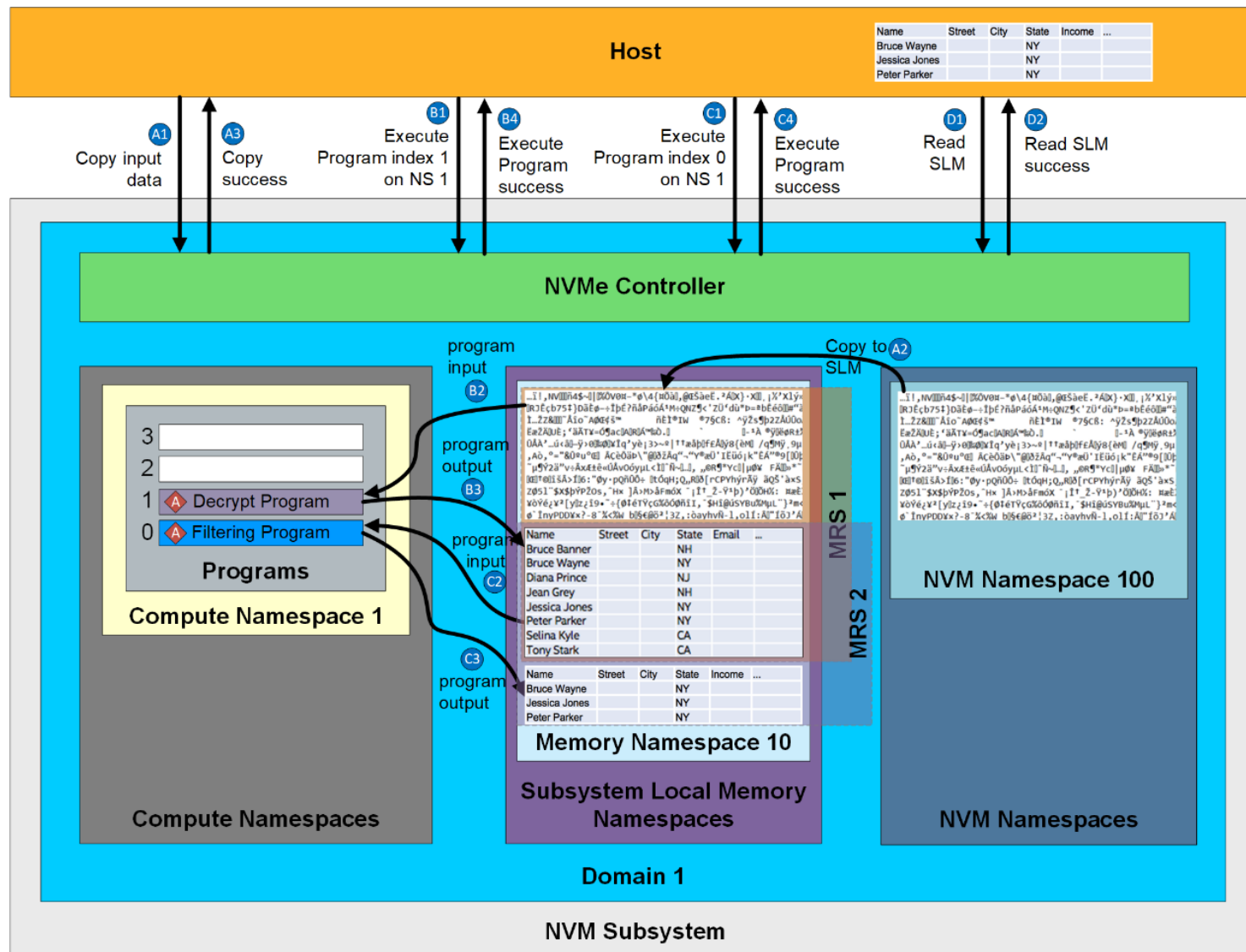
New Subsystem Local Memory I/O command set for memory namespaces

- New commands include:
 - Memory read and memory write
 - Commands for transferring data between host memory and a memory namespace
 - Memory copy
 - Command for copying data from NVM and memory namespaces to a memory namespace



This presentation discusses NVMe® technology work in progress, which is subject to change without notice.

Flow: Execute Program – Filter Encrypted Data



Precondition:

- Memory Range Sets MRS1 and MRS2 have been created

Flow steps

- Copy encrypted data into SLM
- Execute Program 1 on compute NS 1 using MRS1
- Execute Program 0 on compute NS 1 using MRS2
- Read filtered data from SLM to host

Help progressing computational storage



- Join the SNIA Technical Work Group
 - Go to [SNIA Member Portal](#)
 - Select [TWG: Computational Storage](#)
 - Click on the “Join Group”
- CS TWG meetings
 - Wed 10-11am Pacific time



- Join the task group
 - Go to the NVMe workgroup portal
 - Select the CS Task Group
 - Click on the “Join Group” link
- Task group meetings
 - Thursdays 9–10am Pacific time

Computational Storage Drives

Computational storage drives

- **Pros**

- Scalable
- Minimized data movement
- Simple deployment

- **Cons**

- \$/drive
- Fixed compute per drive – hard to optimize
- Power
- Data striping

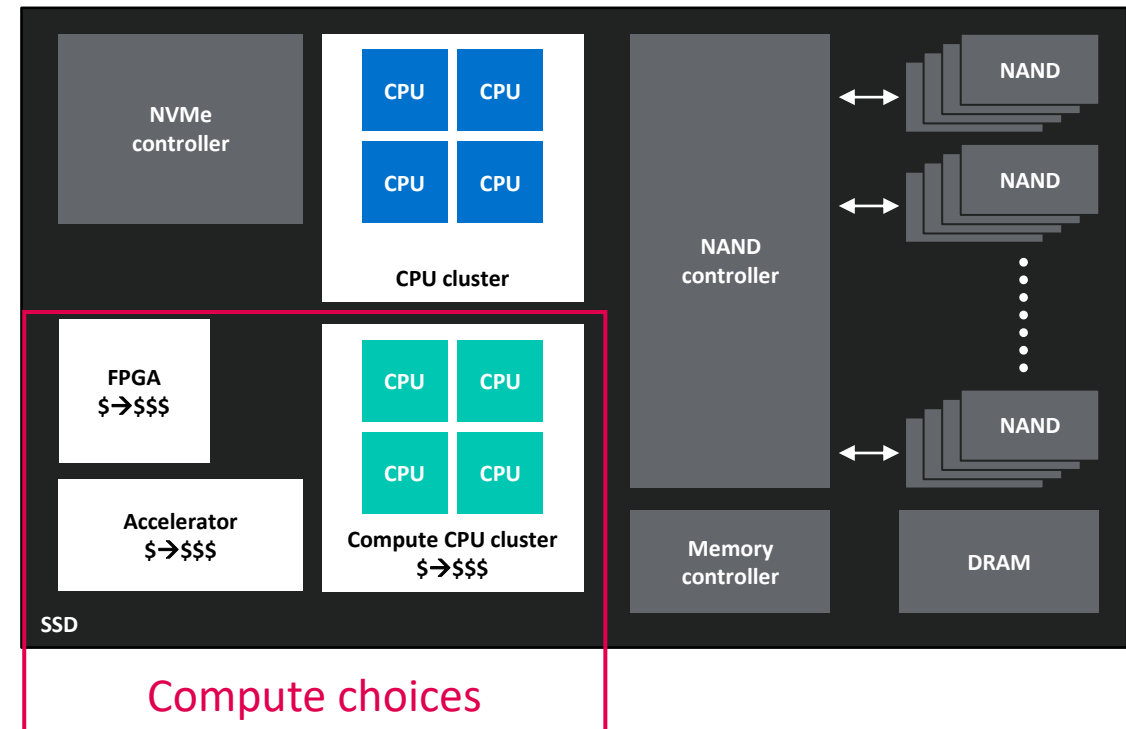
Push to minimize SSD cost and power

- No room for CSD to replace SSD?
- CSDs niche products?

Is there a killer app?

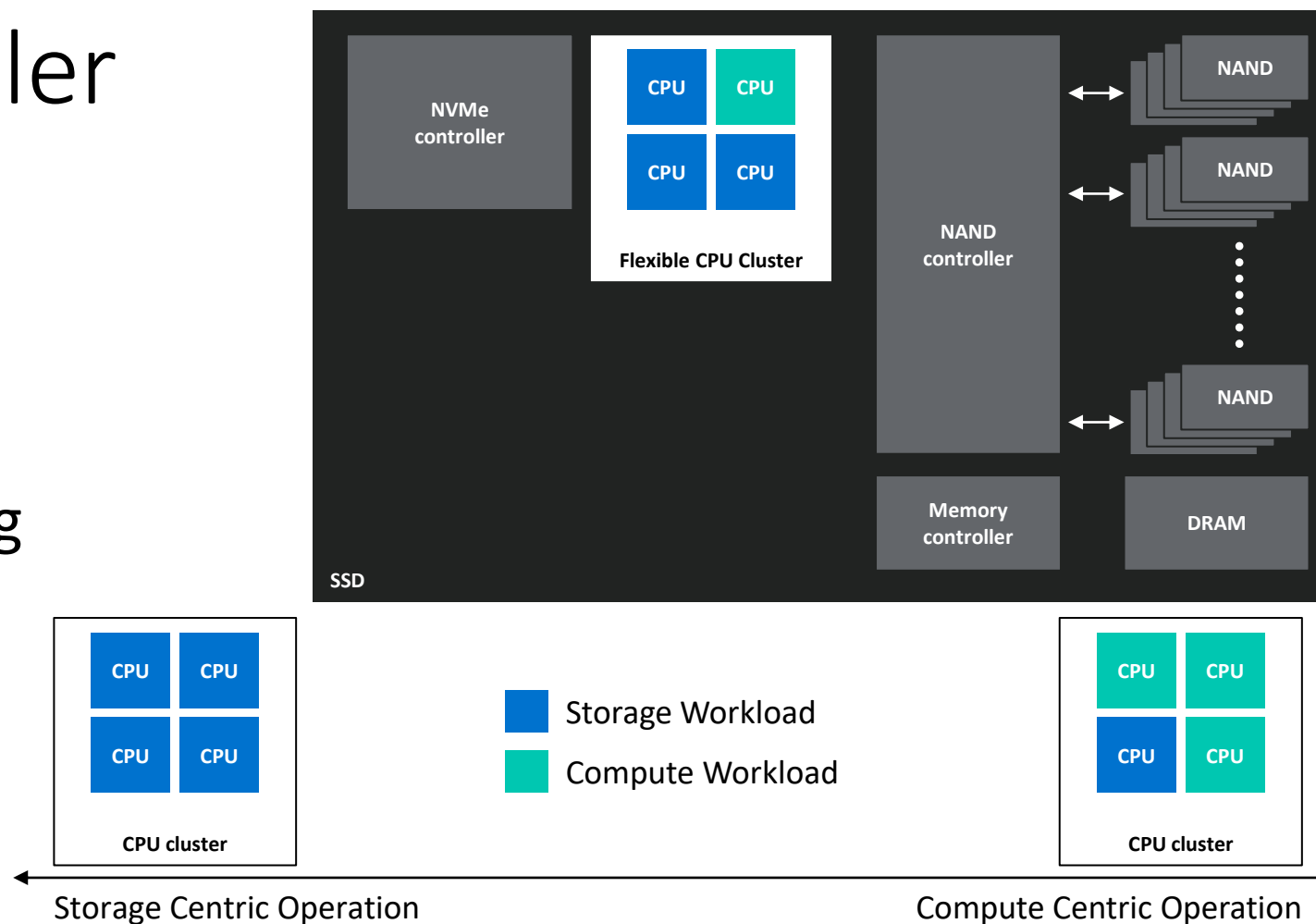
Compute inside SSD

- Computational storage drive
 - Compute power vs cost
 - FPGA
 - Hardware accelerator
 - CPU cores for compute
 - Memory
- Niche applications vs. general purpose



Flexible controller

- Workload balancing
- Flexible CPU usage
 - Storage vs Compute
- Background processing



Common concerns

- What about...
 - Compressed data: Would need to decompress data locally
 - Encrypted data: Would need ability to decrypt the data locally
 - Data striped over several drives: A single device may not contain the whole file
 - With SSD the need for striping data is smaller. Maybe consider moving away from widespread striping
 - Block storage: Storage devices usually have no idea of file system
 - Object storage, or key-value (KV) are solutions, but block storage is widely deployed
 - Other solutions include mounting file system in CSD, or virtual objects

What Exists Today

Examples only

NGD systems – too early?

- Folded in 2022
- Newport System
 - NVMe CSD
 - ASIC based
 - Quad core ARM CPU
 - Linux OS capable
- World's first NVMe CS drive
- Many partners

Scaleflux

- CSD1000 and CSD2000
 - FPGA acceleration
 - Database filtering
 - Compact HW acceleration
 - Deployed at Alibaba
 - Transparent inline compression
 - 2:1 compression
 - Deployed in all drives
 - Unencrypted data only
- CSD3000
 - SOC acceleration
 - Reduced cost
 - Native NVMe support

Samsung SmartSSD

SmartSSD1.0

- FPGA based
- Supports (Examples)
 - Search
 - Database scan and filtering
 - Video transcoding
 - Analytics
 - Compression
 - Encryption

SmartSSD2.0

- Bigger FPGA
- ARM CPUs
- NVMe
- TP4091
- eBPF
- Fixed functions

Lewis Rhodes Labs

NPUsearch™

- Integrated into Samsung 4TB SmartSSD™
- Predictable search times
 - 4TB regex search <25minutes
- Decreased data movement

ExtremeSearch™

- 96TB self-searching storage appliance
- NPUsearch™
- 24 4TB SmartSSD™
- 96TB regex search < 25minutes

Eideticom

NoLoad™ Computational Storage Processor

- Line-rate compression on Lustre/ZFS
- FPGA accelerator card
- NVMe

Partnered with Los Alamos National Laboratory (LANL)

- NoLoad™ used in ABOF

Los Alamos National Laboratory

Big data, big jobs

- 10 PB DRAM
- 100 PB Flash
- 500PB HDD
- Few jobs – very big, very long

Several Custom Implementations

- Accelerated Box of Flash (ABOF)
- Key-value Store Computational Storage Device
- DeltaFS – Near-device Indexing and Analytics
 - Large-scale physics simulations
 - File-based I/O → record- and columnar-indexed I/O

Learn More at FMS

Wednesday

8:30 - 9:35 AM NVME-201-1 Computational Storage and Subsystem Local Memory

Thursday

8.30-9.35 AM SARC-301-1 Computational Storage Killer Apps (System Architectures Track)

9:45-10:50 AM SARC-302-1 Computational Storage Use Cases (System Architectures Track)

11:00-12:05 PM SARC-303-1 Computational Storage Optimization (System Architectures Track)

12:10-1:15 PM SARC-304-1 Future of Computational Storage (System Architectures Track)

12:10-1:15 PM SARC-304-3 Computational Storage and AI (System Architectures Track)

Discussion

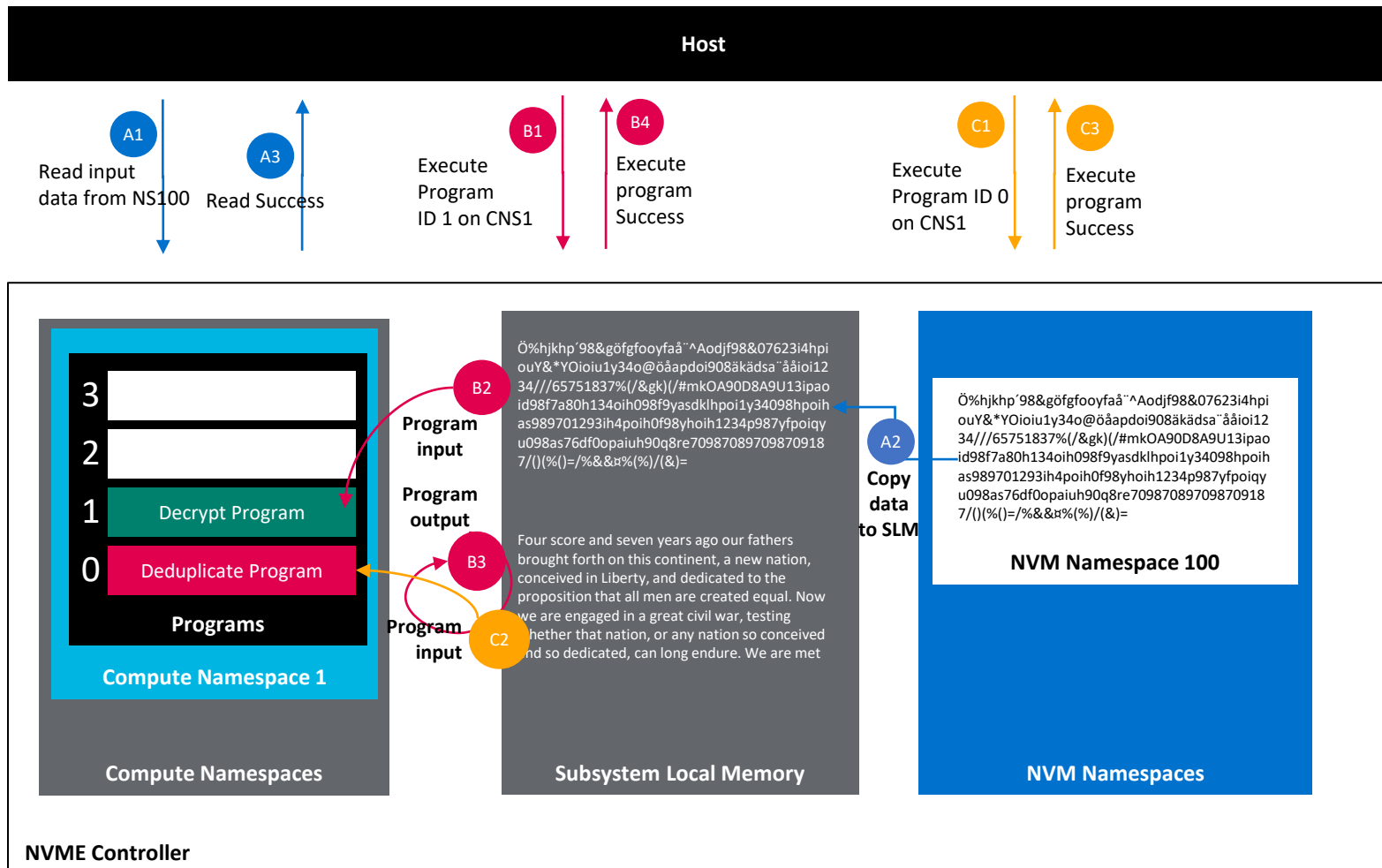
- What is the killer app for computational storage?
 - Topic of session SARC 301-1 on Thursday at 8.30 AM
- CSD, CSA or CSP? Pros and cons
- Reasons for computational storage success or failure

Thank You!

Backup with Simple example

Simple Example

Simple offline deduplication example



Flow steps

- A** Read encrypted data into SLM
- B** Execute Program ID 1 on CNS1
- C** Execute Program ID 0 on CNS1

Repeat for next
block of data