

# DATA ACCESS - Near Data Processing PERSPECTIVE

Gaurav Kaul

Systems Architect, DCG

# Agenda

The Challenge : “Memory Wall”/Von Neumann Bottleneck

Traditional Solutions

New Technology Opportunities – 3D Stacking, NVM, MCP (Xeon+FPGA)

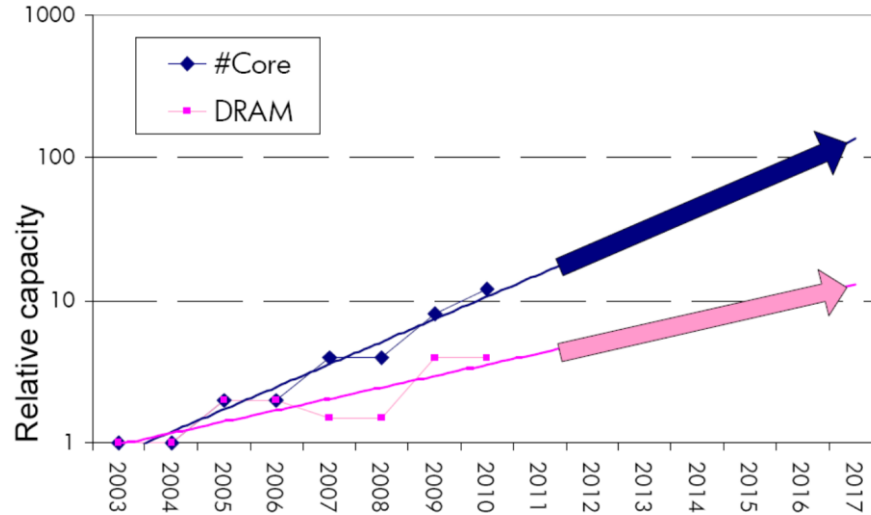
Programming Model for Near Data Processing

Example Applications – HPC Simulation using GANs/DL

Summary / Call-to-Action

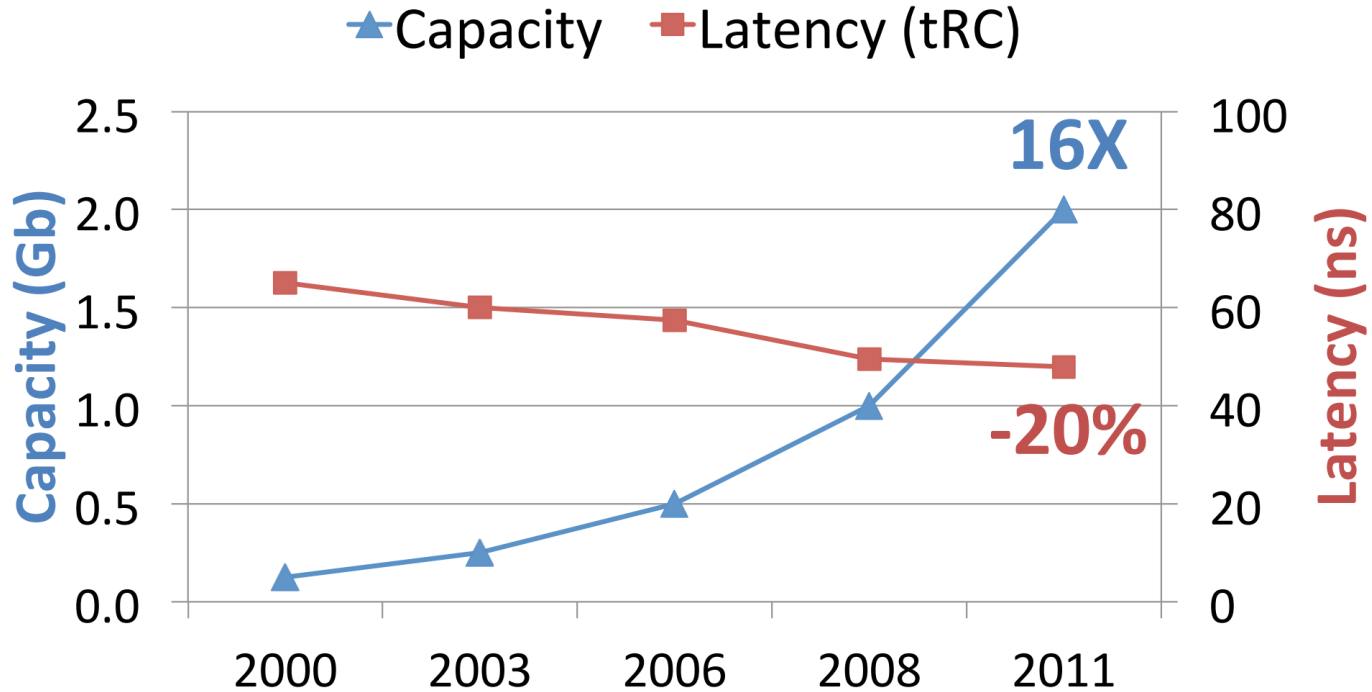
# The Memory Wall

Core count doubling ~ every 2 years  
DRAM DIMM capacity doubling ~ every 3 years



Source: Lim et al., ISCA 2009.

# The Memory Wall (contd.)



# Von Neumann Bottleneck



The term "Von Neumann bottleneck" was coined by John Backus in his 1977 ACM Turing Award lecture.

According to Backus:

*"Surely there must be a less primitive way of making big changes in the store than by pushing vast numbers of words back and forth through the Von Neumann bottleneck. Not only is this tube a literal bottleneck for the data traffic of a problem, but, more importantly, it is an intellectual bottleneck that has kept us tied to word-at-a-time thinking instead of encouraging us to think in terms of the larger conceptual units of the task at hand. Thus programming is basically planning and detailing the enormous traffic of words through the Von Neumann bottleneck, and much of that traffic concerns not significant data itself, but where to find it."*

# The Challenge

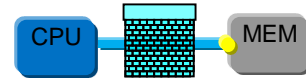
## Conventional computing architecture is CPU-centric

- All data is moved to/from memory for computation



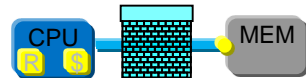
## The “Memory Wall” (aka [Von Neumann bottleneck](#))

- Latency/Bandwidth → Perf.; Mitigations expensive
- Data movement → Energy; Significant



## Traditional approach: *Memory in Processor*

- Registers, cache hierarchy, dedicated buffers
- Reduce data movement by caching and re-use



# Memory Wall : *Performance*

## Latency

- Ever larger, deeper caches
- uArch (Spec., OOO, HT, ...)
- Data prefetching (sw/hw)
- Fast (optical?) interconnect
- ...

→ Die\$\$\$

→ Die\$\$, Complexity(TTM)

→ Application dependent

→ Tech\$\$, E-O-E overhead

## Bandwidth

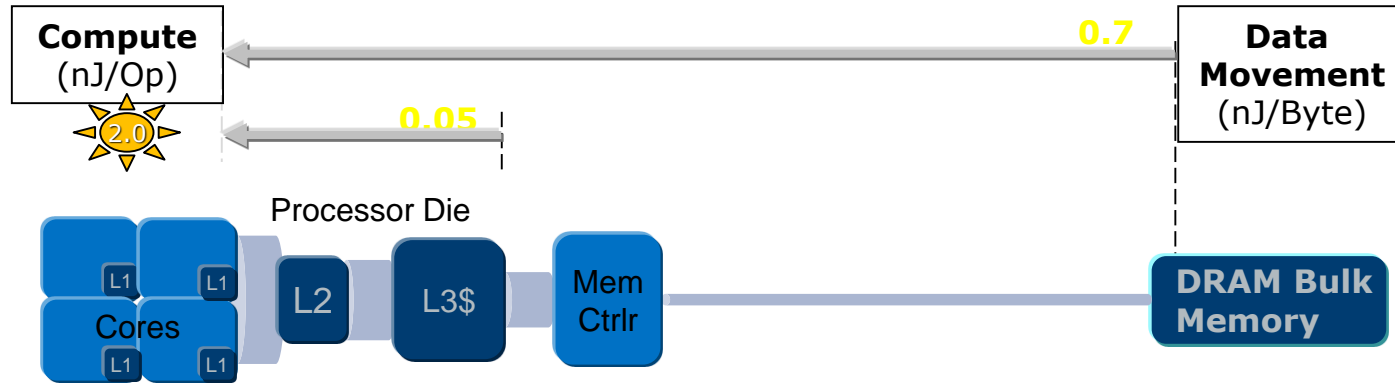
- More memory controllers
- More/Faster mem. channels

→ Die\$, Complexity(TTM)

→ Pin\$\$, Power, Elec(TTM)

**Traditional approaches to scaling the  
“memory wall” are expensive**

# Memory Wall : *Energy*



Energy/Byte :: Energy/Operation == 1 :: 3

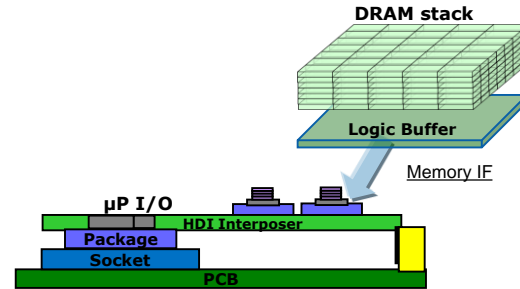
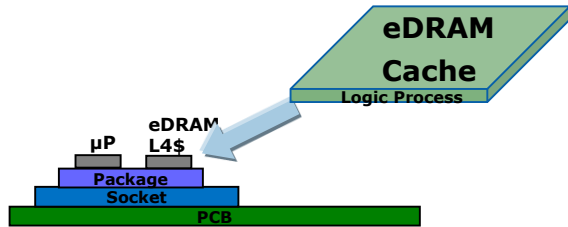
Nominally, an operation needs upto 24 bytes ( $\rightarrow \sim 17$  nJ/Op)

Caching+reuse reduces data movement energy ( $\rightarrow \sim 1.2$  nJ/Op)

**Data movement across the memory wall limits performance and is energy expensive!**



# Emerging Tech Opportunities



## eDRAM

- Logic compatible memory tech
- High BW Memory Cache
- On-die or Multi-chip Pkg
- Reduces nJ/Byte by ~5x

## 3D Stacked Memory (HBM/HMC)

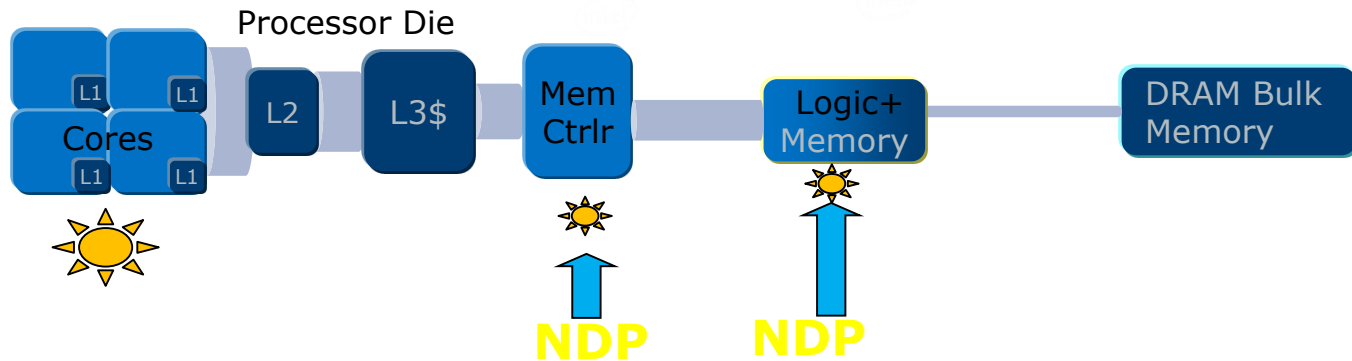
- Low power/High BW interface
- Stacked memory w/ logic buffer
- Higher density DRAM arrays

**A new level in the memory hierarchy  
that combines logic + memory**

# The NDP Spectrum

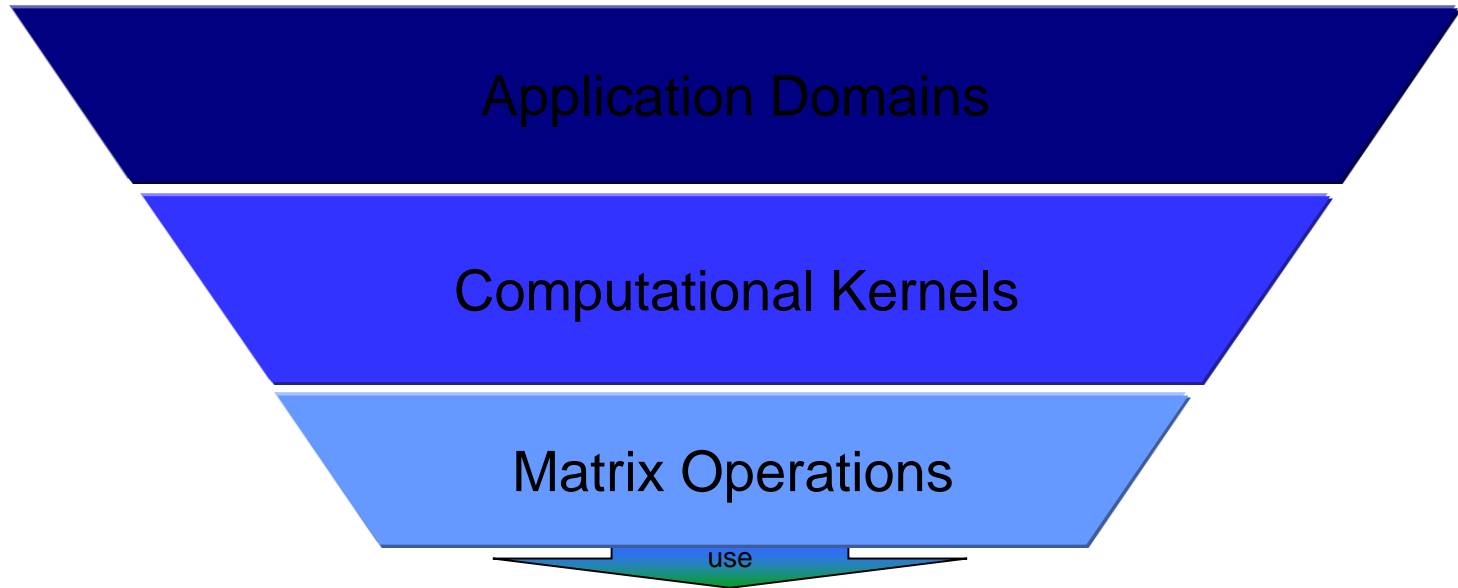
Execute arith & address ops near memory sub-system to:

- Reduce data movement costs (↓ energy, ↑ performance)
- Enable wide parallel operations (exploit memory BW)
- Complement, even improve, caching



**Enables a “continuum” of compute into the memory hierarchy**

# NDP Primitives



use

## NDP Primitives

Block  
Copy/Zero

Gather/  
Scatter

Stream  
Filters

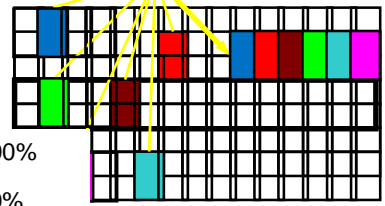
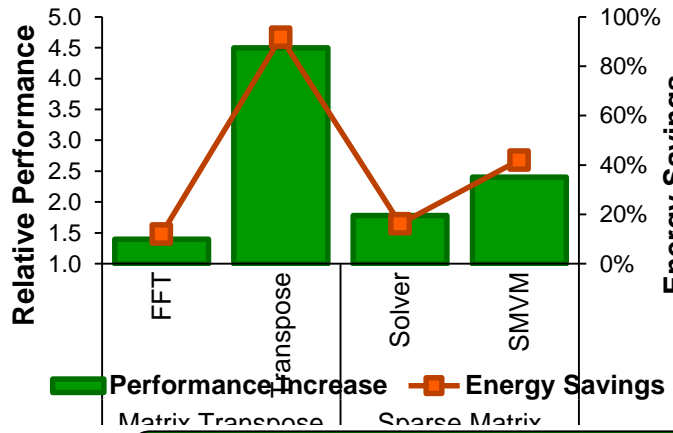
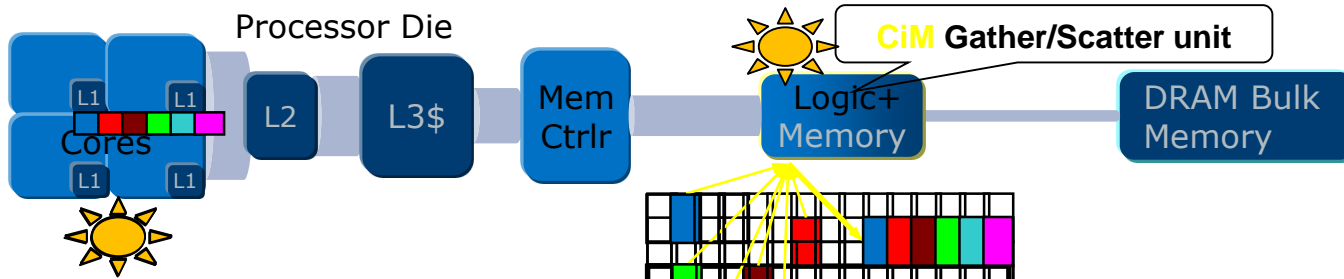
Data  
Comparison

Atomic  
Reduction

Synchron-  
ization

**Diverse set of application domains  
can benefit from NDP**

# Case Study: Gather/Scatter CiM

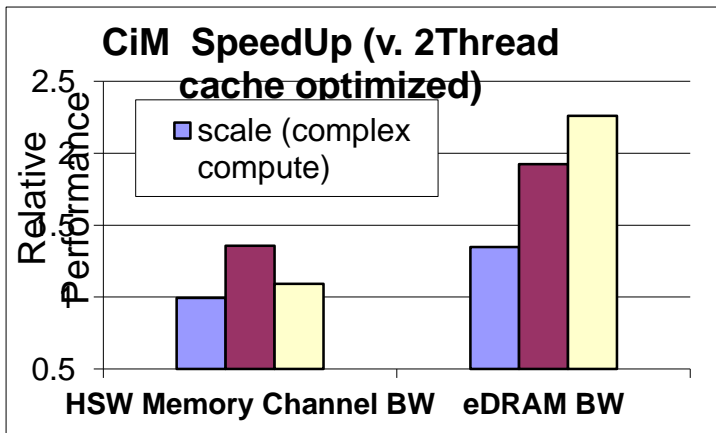
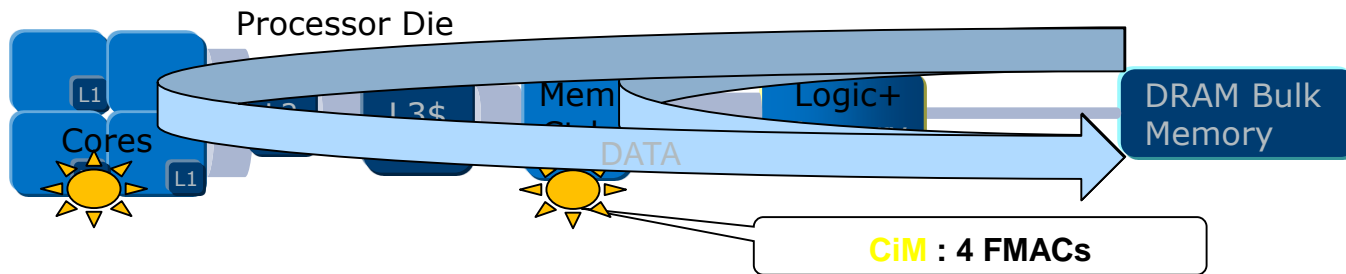


Extends LRB gather/scatter with CiM benefits:

- Reduced data movement
- Increased cache effectiveness
- Increased access concurrency

**Gather/Scatter CiM improves cache locality through optimized data movements**

# Case Study: Stream Filter CiM



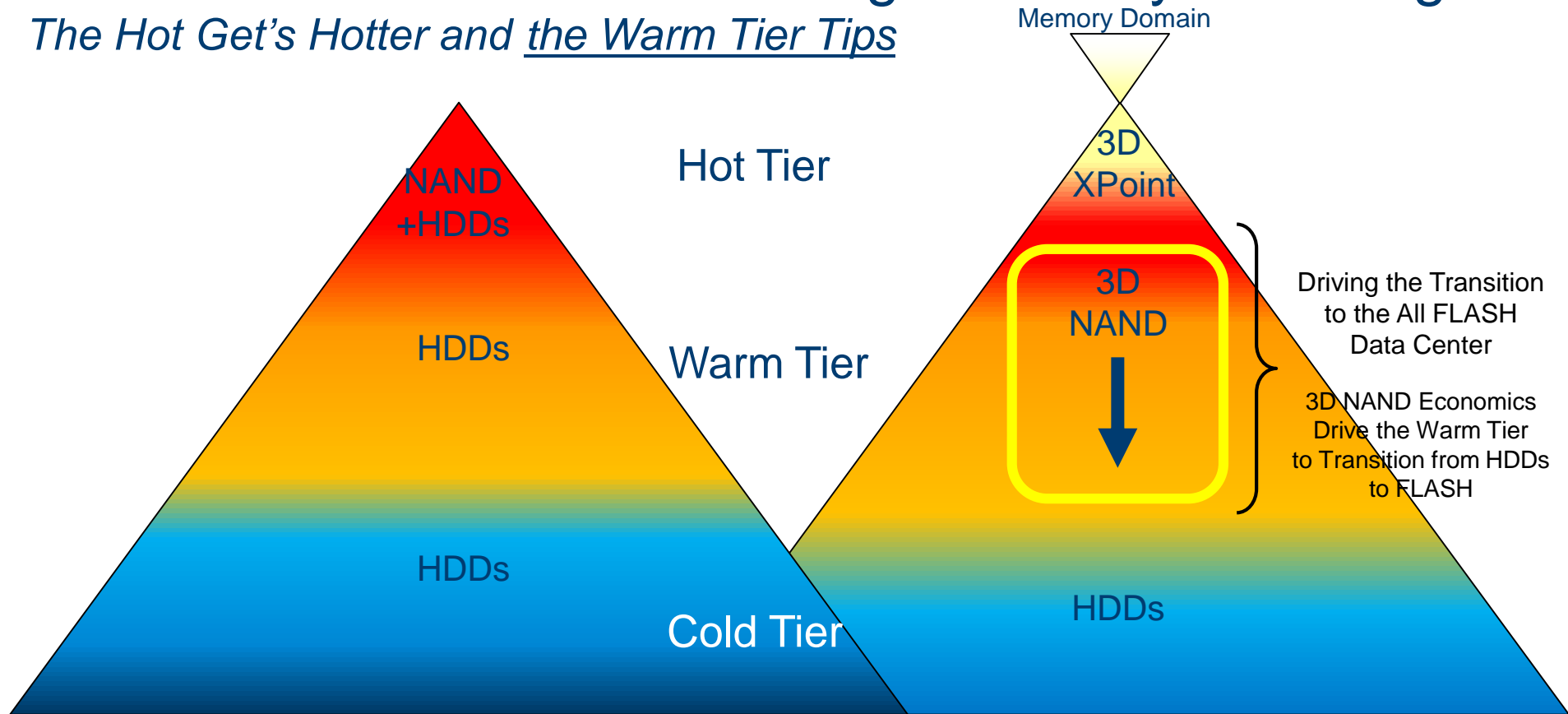
- Recovers “Lost” BW in processor
  - Control and buffering overhead
- - By avoiding cache pollution
- - If implemented in

**Filter CiM improves performance through better use of bandwidth**

# NVME revolution

# Market Motivation: The Evolving Hierarchy of Storage

*The Hot Get's Hotter and the Warm Tier Tips*



# Ruler SSD Form Factor Project

## Objectives

- Maximize FLASH TCO: Understand the TCO and engineer to exploit it
  - 3D NAND economics are enabling the tipping point. How can we accelerate this?
- Avoid the limitations of classical SFF solutions
- Enable scale: Desired a form factor that will support Very High system performance
  - 1PB per Rack U was our stake in the ground objective
  - Now you can buy leading edge HPC bandwidth/IOPS
- Enable dynamic range: Desired a form factor that can efficiently service a wide range of system configurations and performance points
  - Cover the needs of both Scale Up and Scale Out platforms
  - Support wide range of configurations from modest to high performance
- Attack system cost/complexity points
  - Learn from legacy form factor experience



# Common PCIe Solid State Storage Form Factors



110mm

⏻ 6-8.25W

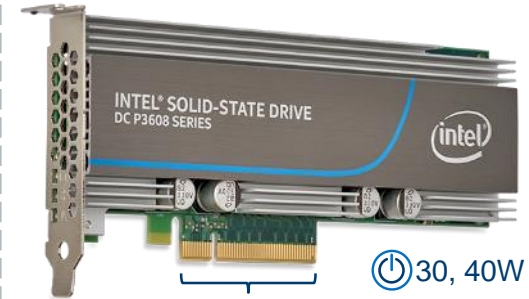
## U.2



7mm { ⏻ 8, 10, 12W

15mm { ⏻ 10 - 25W

## AIC



⏻ 30, 40W

x8



⏻ 25W

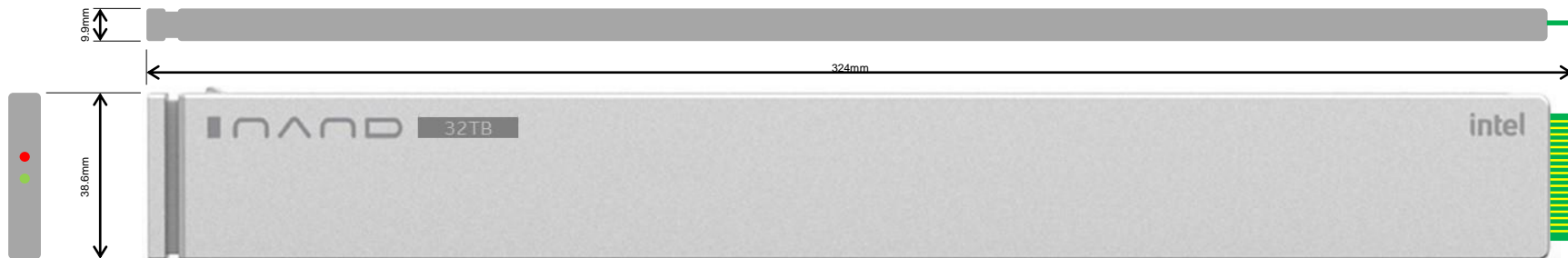
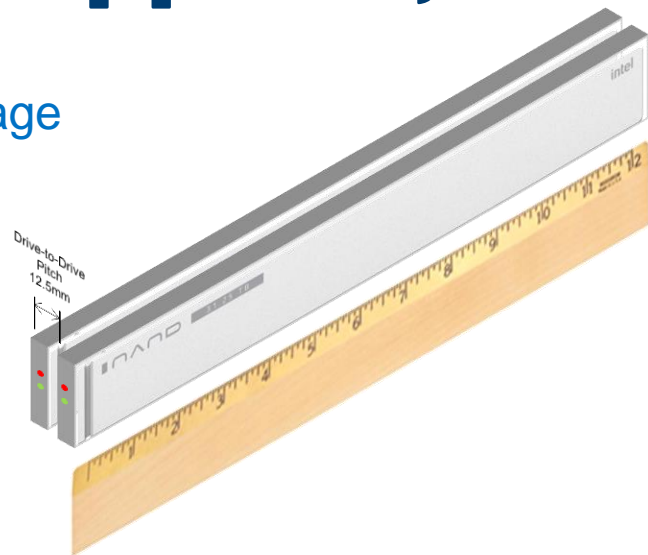
x4



# Form Factors

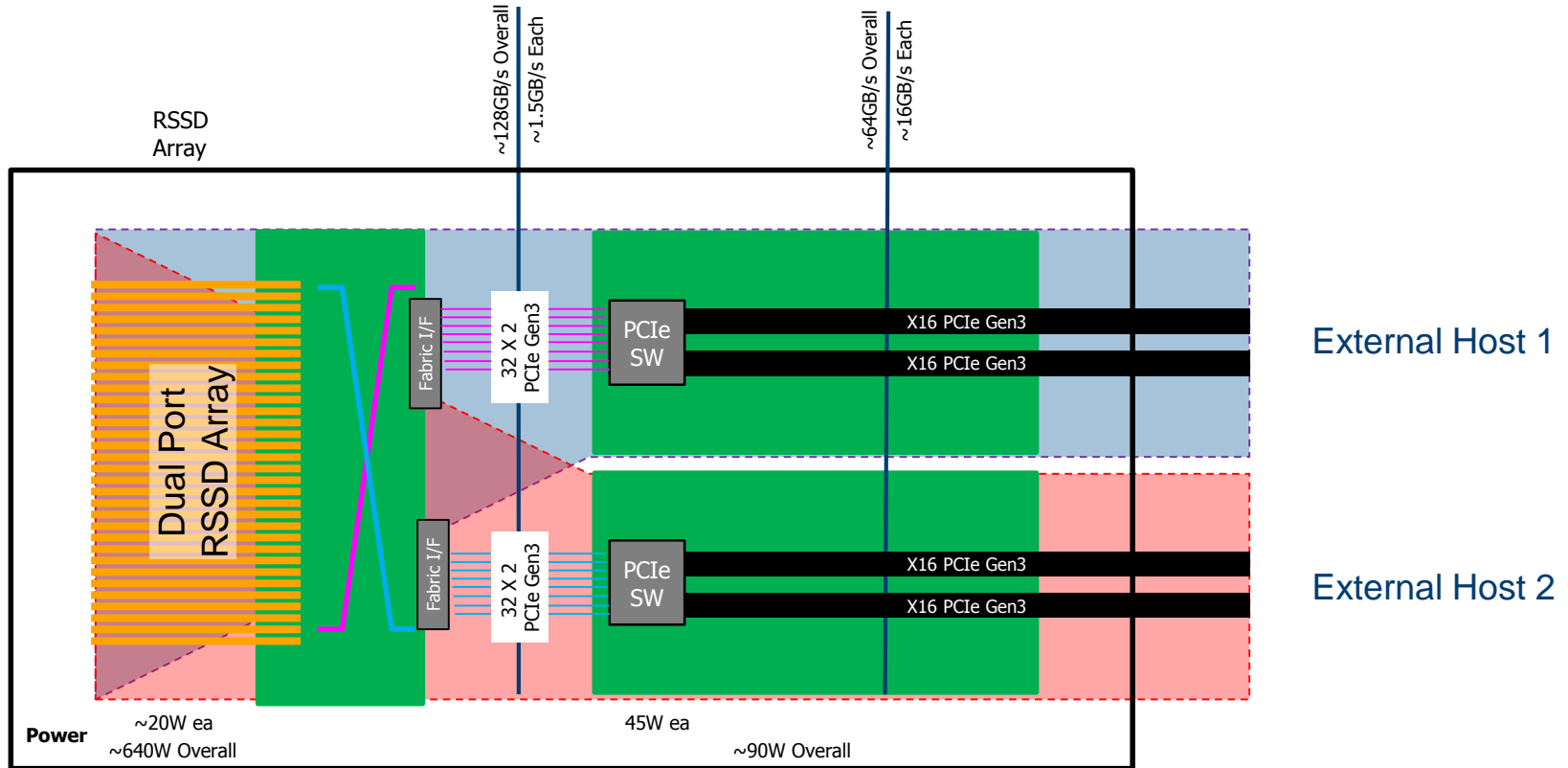
# Front Loading, Hot Swappable, Ruler SSD (RSSD)

- RSSDs are optimized for high density 1U storage
  - Modules are 38.6mm tall and packed at a 12.5mm pitch
- RSSDs are front loading and hot swappable
- RSSDs natively incorporate LEDs and a latching mechanism



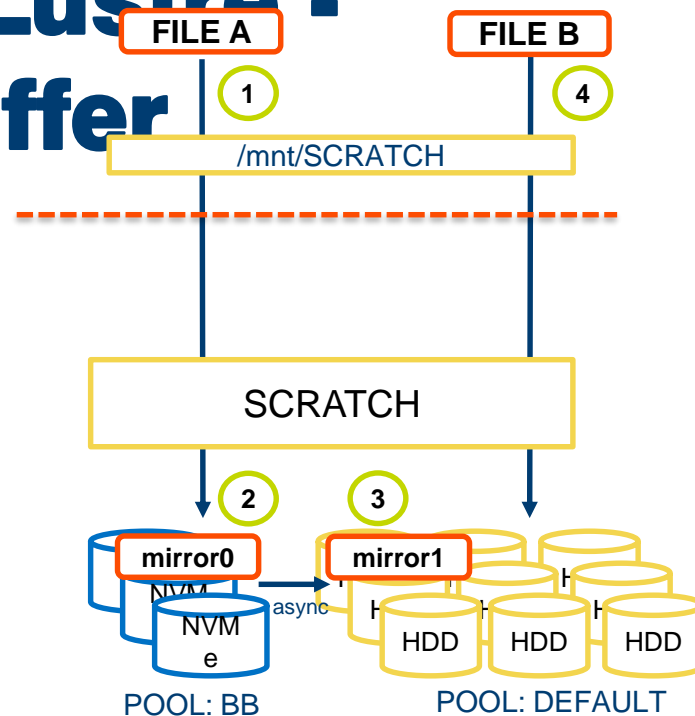
# 1U JBOD

*Allows Provisioning Compute Resources to the Application's Needs*



# Example USE CASE Lustre - Checkpoint Burst Buffer

1. Users can access to a unified global file system and accelerate I/O using FLR layout
2. FileA is written data on a pool of NVMe devices
3. When the write is complete a mirror is created in the HDD based pool. Mirror0 can be removed when the job is completed.
4. FileB is using the default layout and is written into the HDD based pool



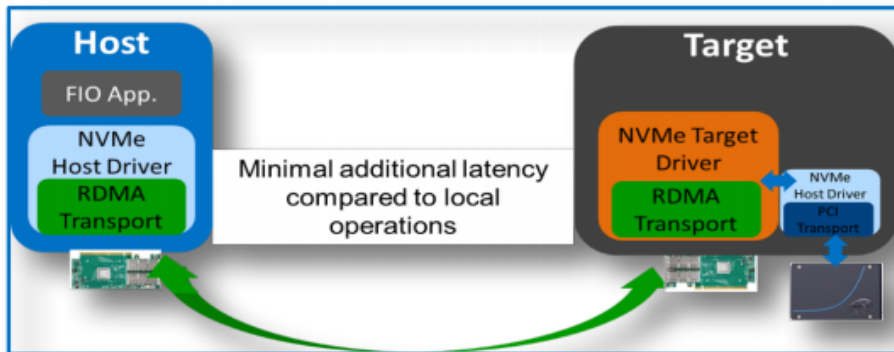
# NVMe OVER fabric

# NVMe\* over Fabrics

## Host and Target Stack Status

### Linux\* Host and Target Kernel drivers operational

- Target configured with NVMe\* PCIe\* SSD
- To ensure fabrics transport is fabric agnostic it has been tested on InfiniBand\*, iWARP, RoCE RDMA adapters
- Host has been tested with multiple target implementations
- Included in upstream kernel since 4.5; 4.8.4 has the latest changes



### NVMe-over-OPA status:

- Compliant with NVMeOF specs 1.0
- OPA driver implemented specs in 10.2 driver release
- NVMe-over-OPA qualification and support at Q4'16
- Presented at SC16 by EchoStreams, RSC Group and others.

# 3D Xpoint™ Technology

## Cross-Point Structure

Selectors allow dense packing and individual access to bits

# 3d XPOINT

## Scalable

Memory layers can be stacked in a 3D manner



## Breakthrough Material Advances

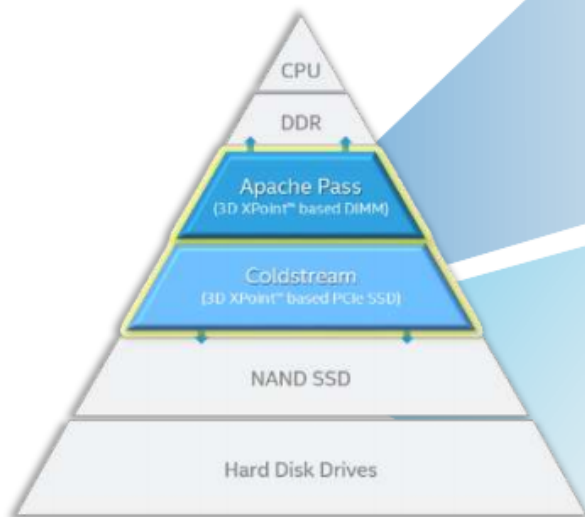
Compatible switch and memory cell materials



## High Performance

Cell and array architecture that can switch states 1000x faster than NAND

# Why 3D XPoint™ NVM in SSD and DIMM?



## Apache Pass (AEP)

(For the Purley 2S+ platform in 2018)

### 1. Highest performance

- Bandwidth, latency, IOPS

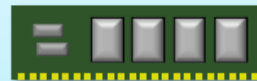
### 2. Highest endurance

### 3. System memory

- DRAM replacement
- Persistent memory

### 4. Diskless platform

- Soft RAID and disk image boot



Data granularity:  
64B cacheline

## Coldstream

(For platforms supporting PCIe from 2016)

### 1. Serviceability

- PCIe standard interface
- Hot add / remove
- Hardware or software RAID support

### 2. Higher capacity (vs. AEP)

- Up to 3 TB per drive

### 3. Lower \$/GB (vs. AEP)

### 4. Higher endurance & performance (vs. NAND)

- NAND / HDD upgrade or replacement

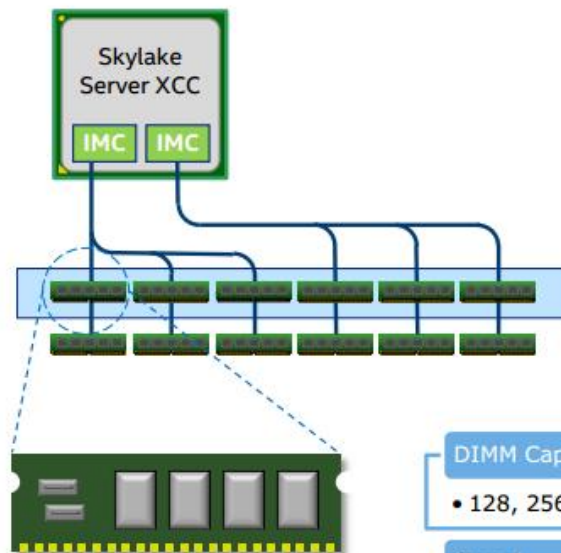


Data granularity:  
512B/4KB block



# Apache Pass Overview

(3D XPoint™ based Memory Module for the Data Center)



Apache Pass

- DDR4 electrical & physical
- Close to DRAM latency
- Cache line size access

## DIMM Capacity

- 128, 256, 512GB

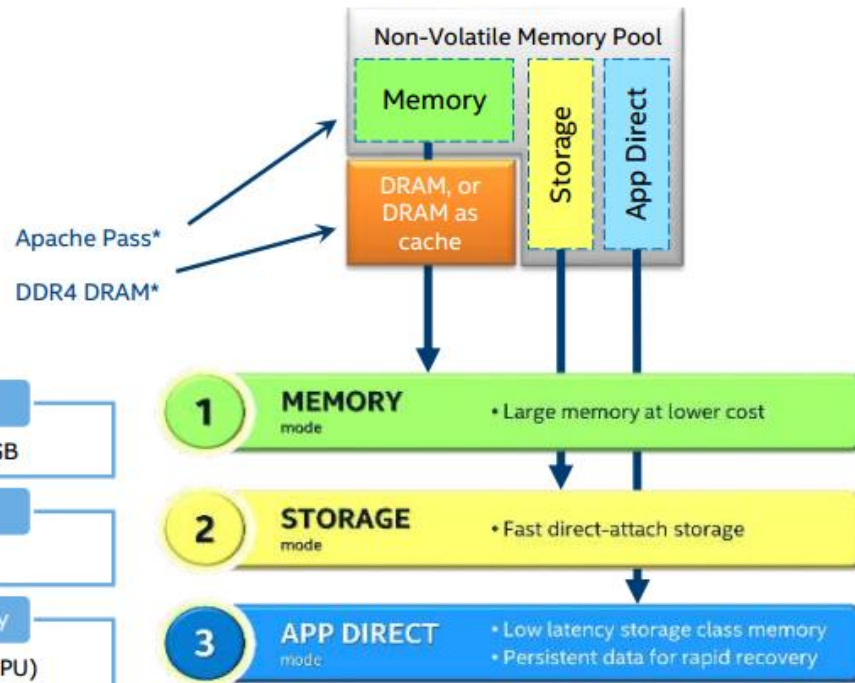
## Speed

- 2666 MT/sec

## Platform Capacity

- 6TB (3TB per CPU)

## Flexible, Usage Specific Partitions



\* DIMM population shown as an example only.

# DAOS – OPEN SOURCE DISTRIBUTED OBJECTS

# Distributed Asynchronous Object Storage

Lightweight Storage Stack

Ultra-fine grained I/O

New Storage Model

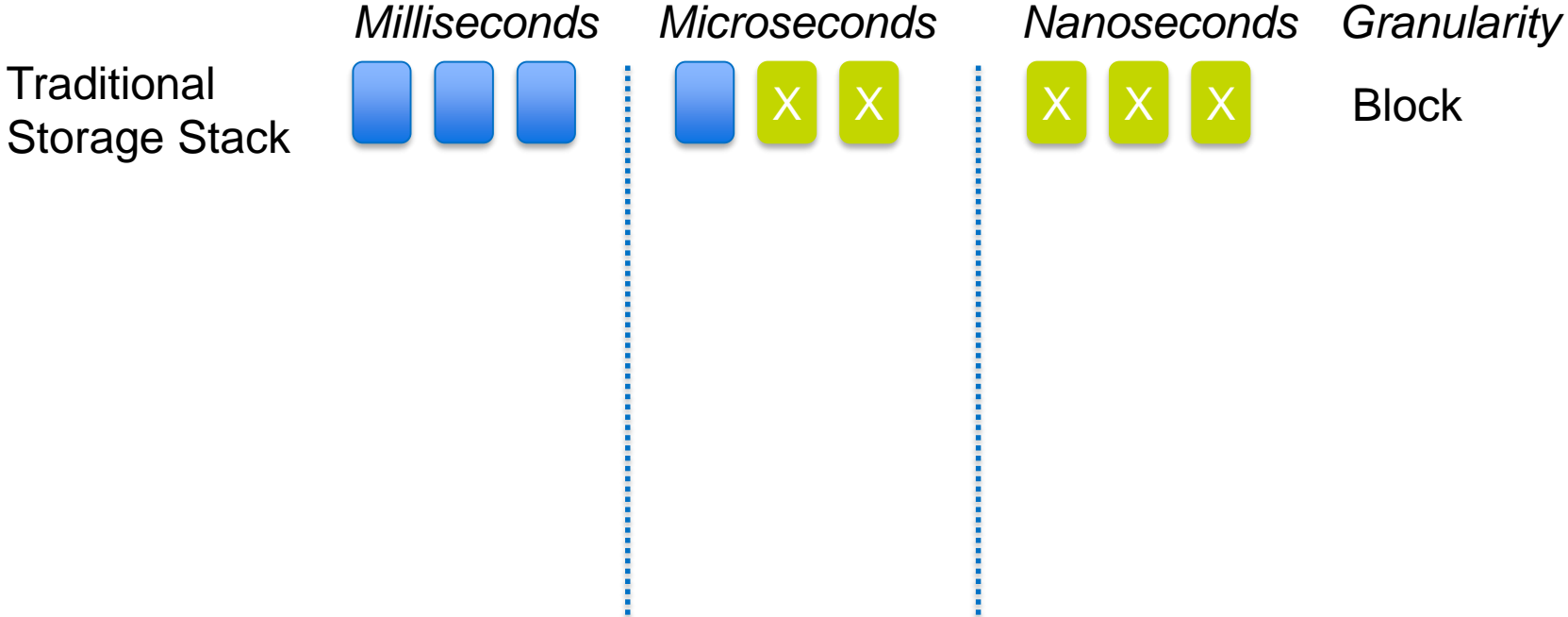
Extreme Scale-out & Resilience

Multi-Tier

New Workflow Methodologies

**Open source - APACHE 2.0 License**

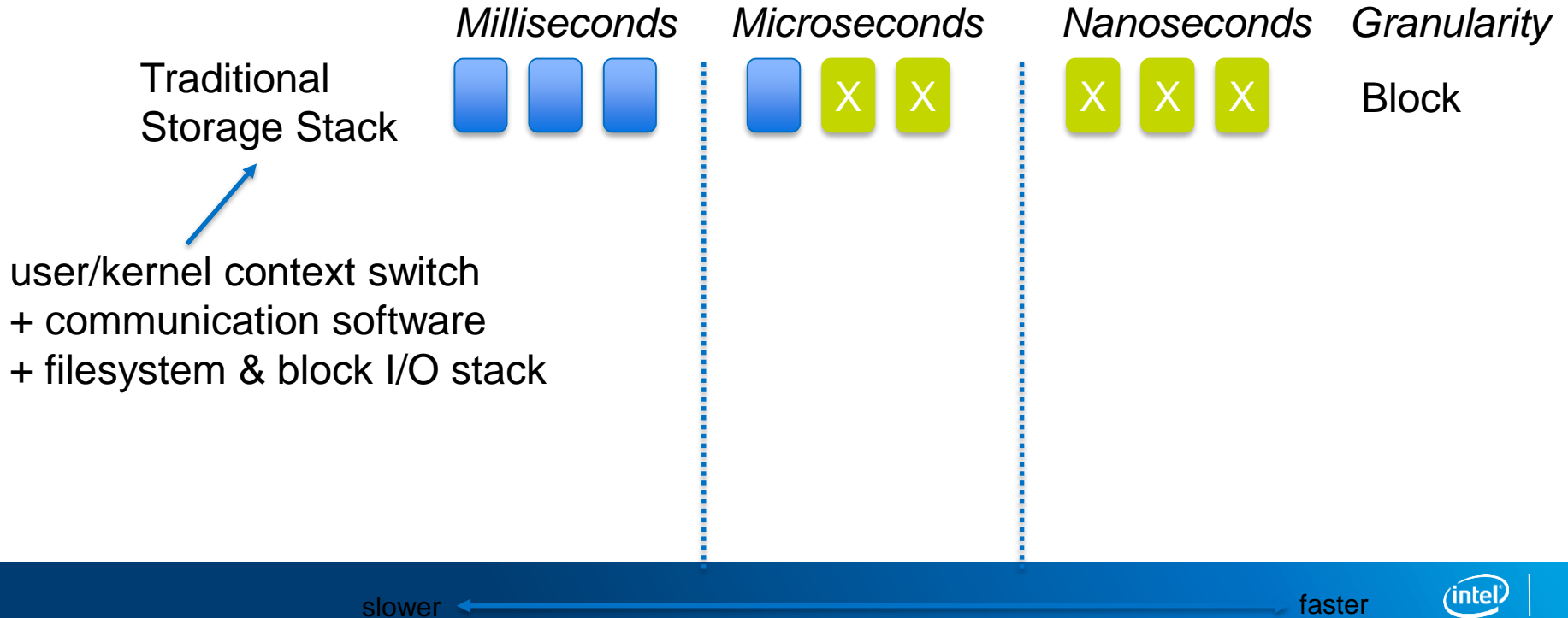
# Storage Latency



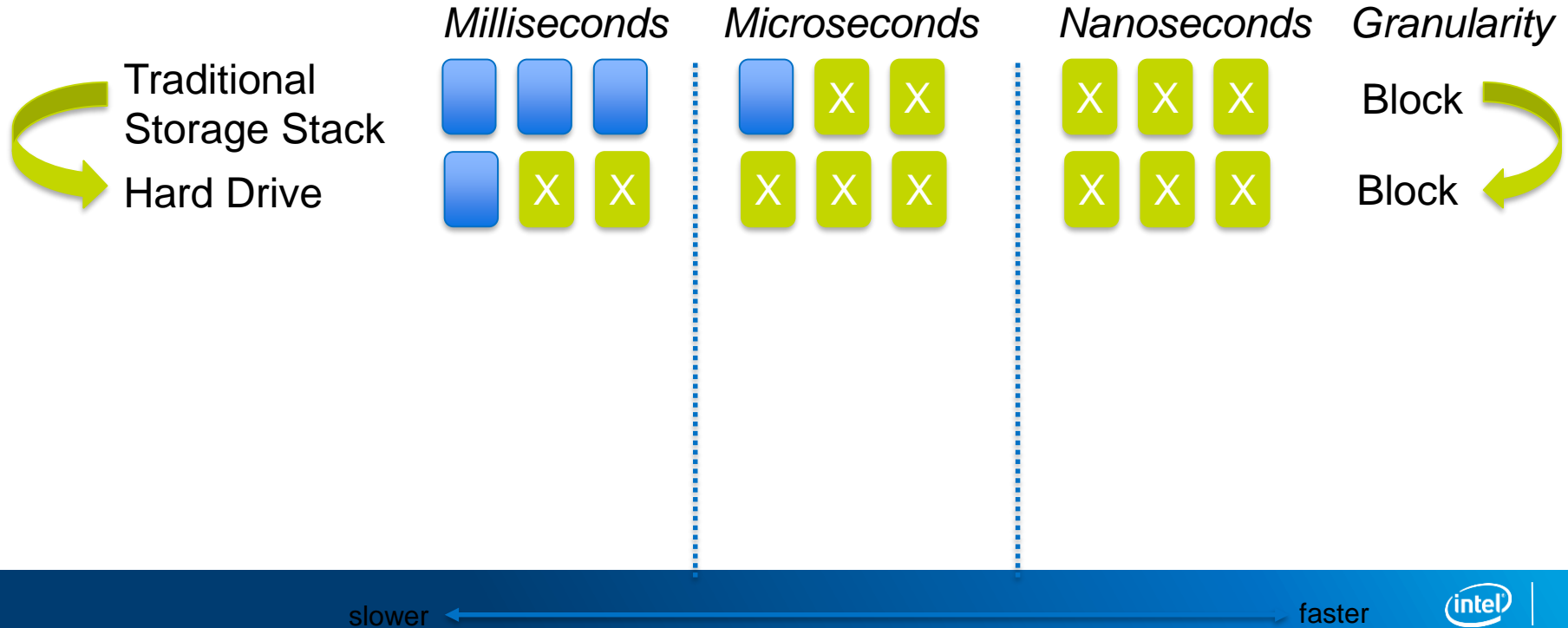
slower ← → faster



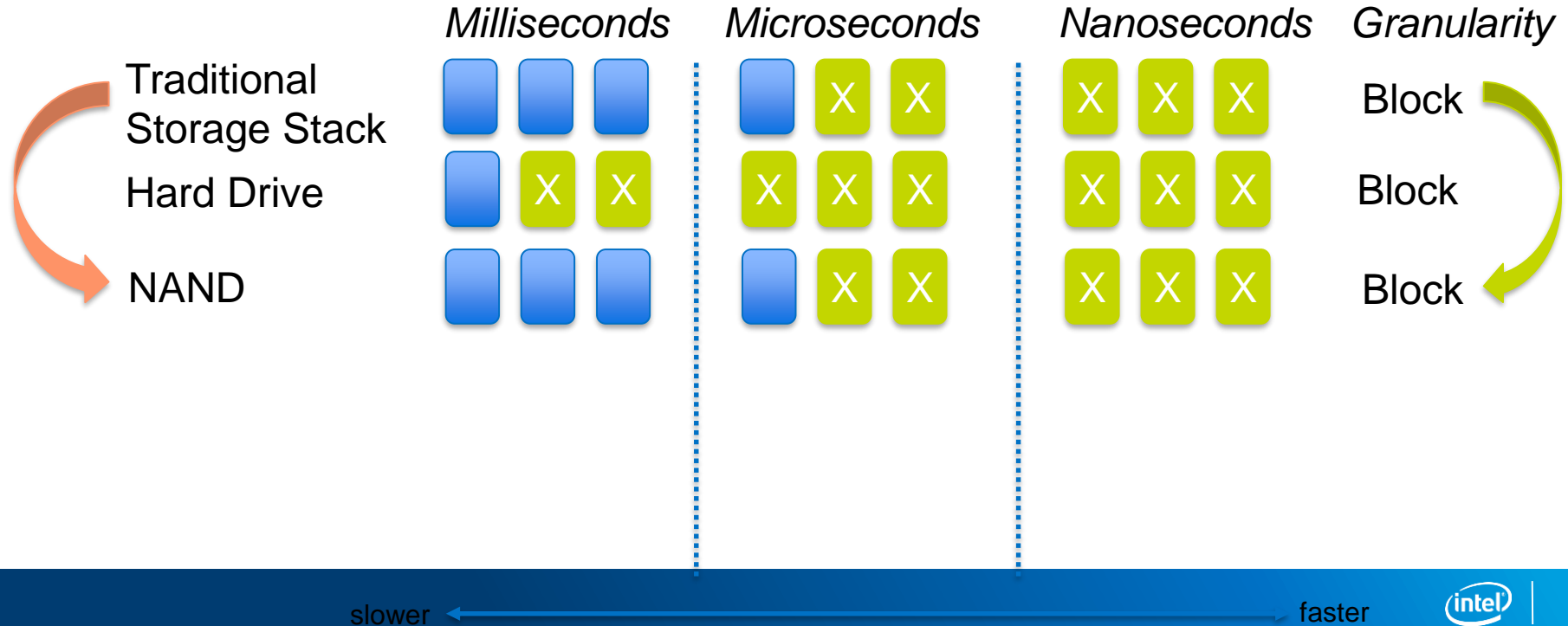
# Storage Latency



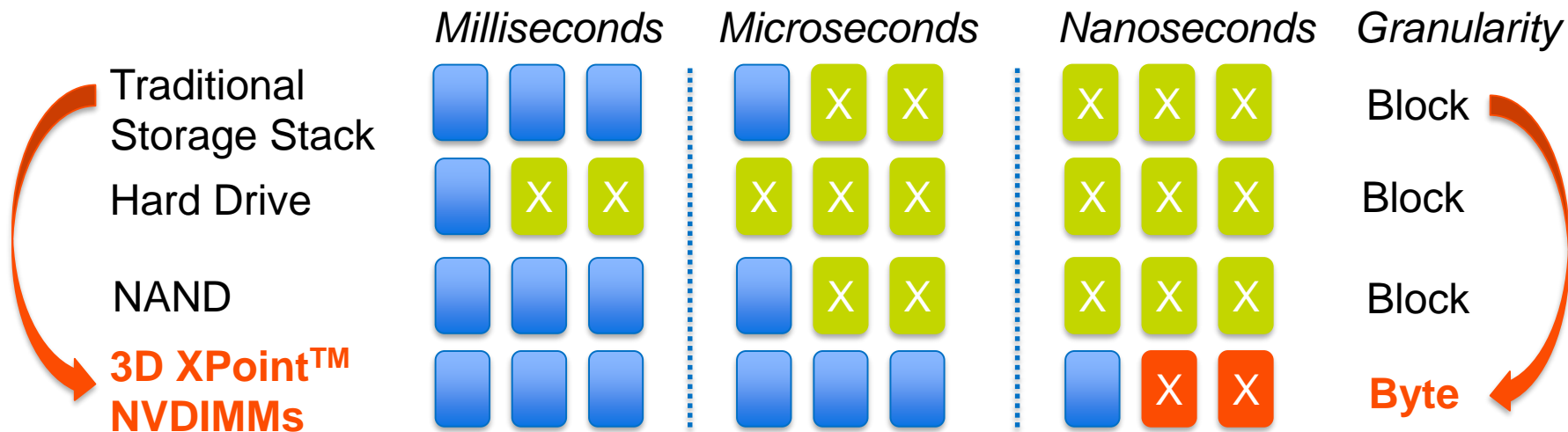
# Storage Latency



# Storage Latency



# Disruptive Technology



Entirely masks HW capabilities!













slower

faster





# Disruptive Technology

	Milliseconds	Microseconds	Nanoseconds	Granularity
Traditional Storage Stack				Block
Hard Drive				Block
NAND				Block
3D XPoint™ NVDIMMs				Byte

Call for a **new** storage stack to deliver **HW** performance  
Up to **1000x** increase in data **velocity!**

slower

faster



# Lightweight Storage Stack

End-to-end OS bypass

Mercury userspace function shipping

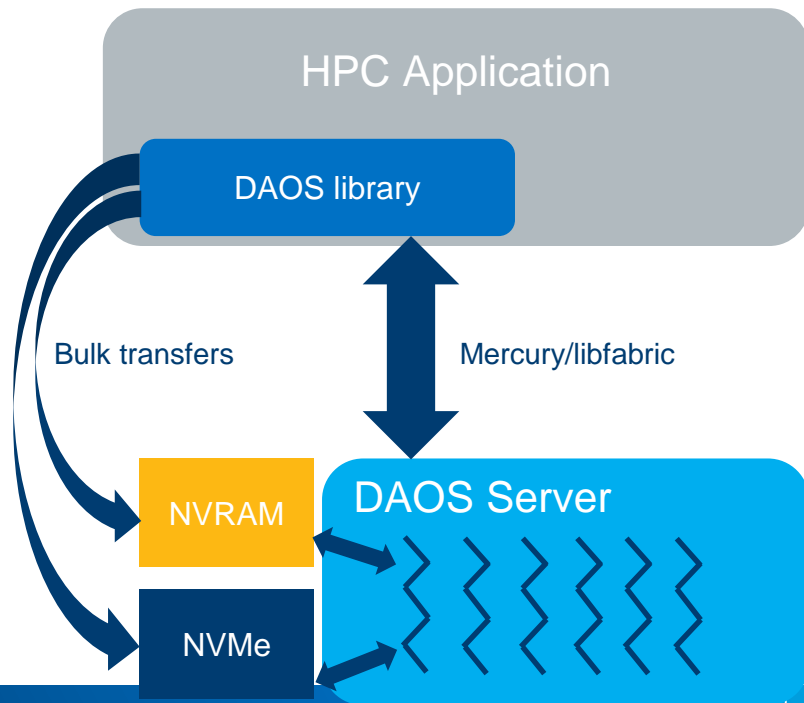
- MPI equivalent communications latency
- Built over libfabric

Applications link directly with DAOS lib

- Direct call, no context switch
- No locking, caching or data copy

Userspace DAOS server

- Mmap non-volatile memory (NVML)
- NVMe access through SPDK\*
- User-level thread with Argobots\*\*



# Ultra-fine grained I/O

Mix of storage technologies

NVRAM (3D XPoint™ NVDIMMs)

- DAOS metadata & application metadata
- Byte-granular application data

NVMe (NAND, 3D NAND, 3D XPoint™)

- Cheaper storage for bulk data
- Multi-KB

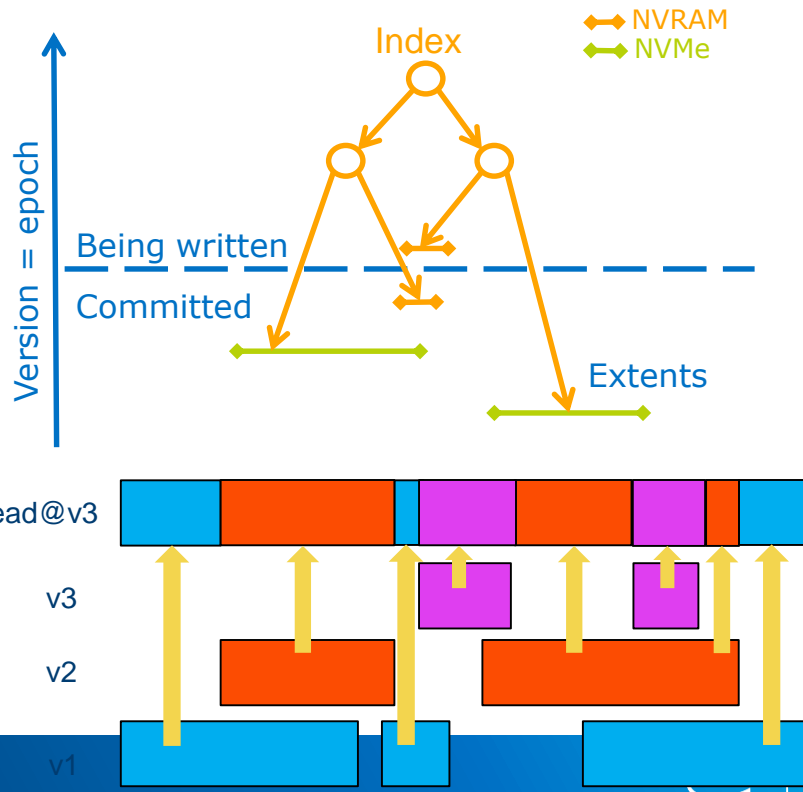
I/Os are logged & inserted into persistent index

All I/O operations tagged/indexed by version

Non-destructive write: log blob@version

Consistent read: blob@version

No alignment constraints



# New Storage Model

Storage Pool

Reservation of distributed storage within a tier

Integration with resource manager

Container

Aggregate related datasets into manageable entity

Distributed across entire storage pool

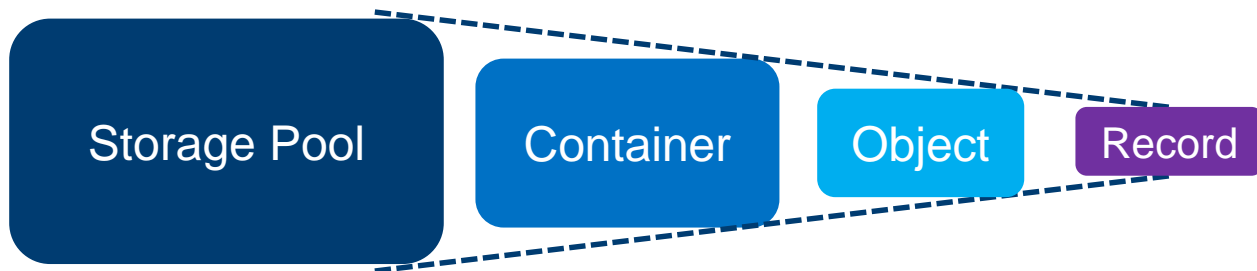
Unit of snapshot/transaction

Object

- Collection of related arrays/values with own distribution/resilience schema
- Key-value store with flexible multi-level key
  - fine-grain control over colocation of related data

Record

- Arbitrary binary blob from single byte to several Mbytes



# Extreme Scale-out & Resilience

Scalable communications

Track jobs and not individual nodes

Tree-based communication

Scalable I/O

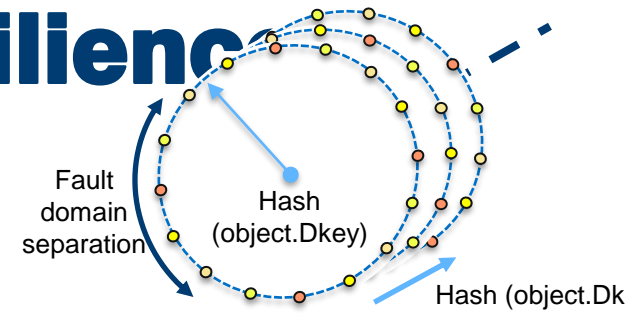
Lockless, no read-modify-write

Ad hoc concurrency control mechanism

Shared-nothing distribution & redundancy schema

Algorithmic & progressive layout

Replication/erasure code with declustered placement



## Storage system failure

- Failed target(s) evicted
- Automated online rebuild & rebalancing

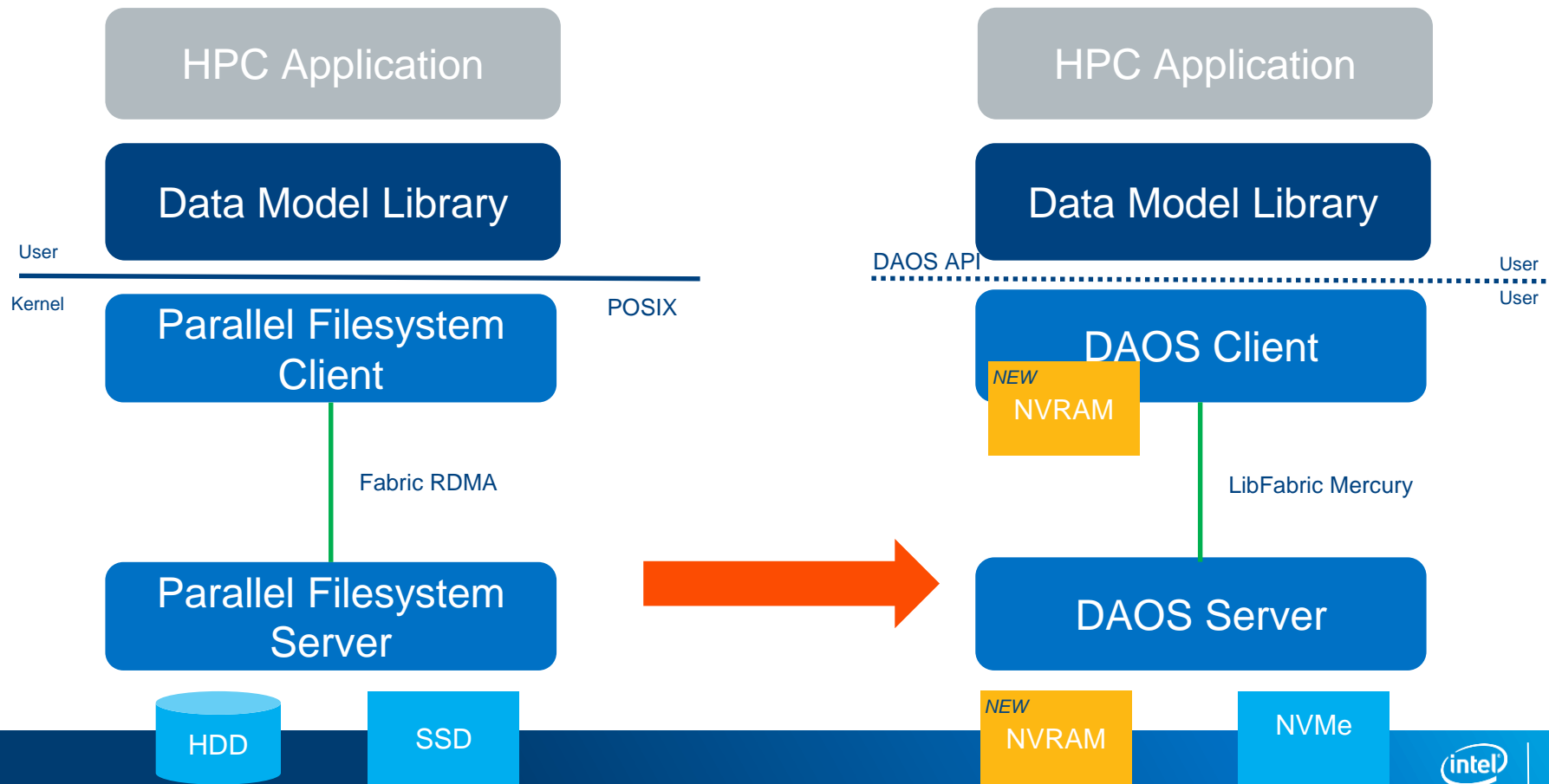
## End-to-end integrity

- Checksums can be provided by I/O middleware
- Stored and verified on read & scrubbing

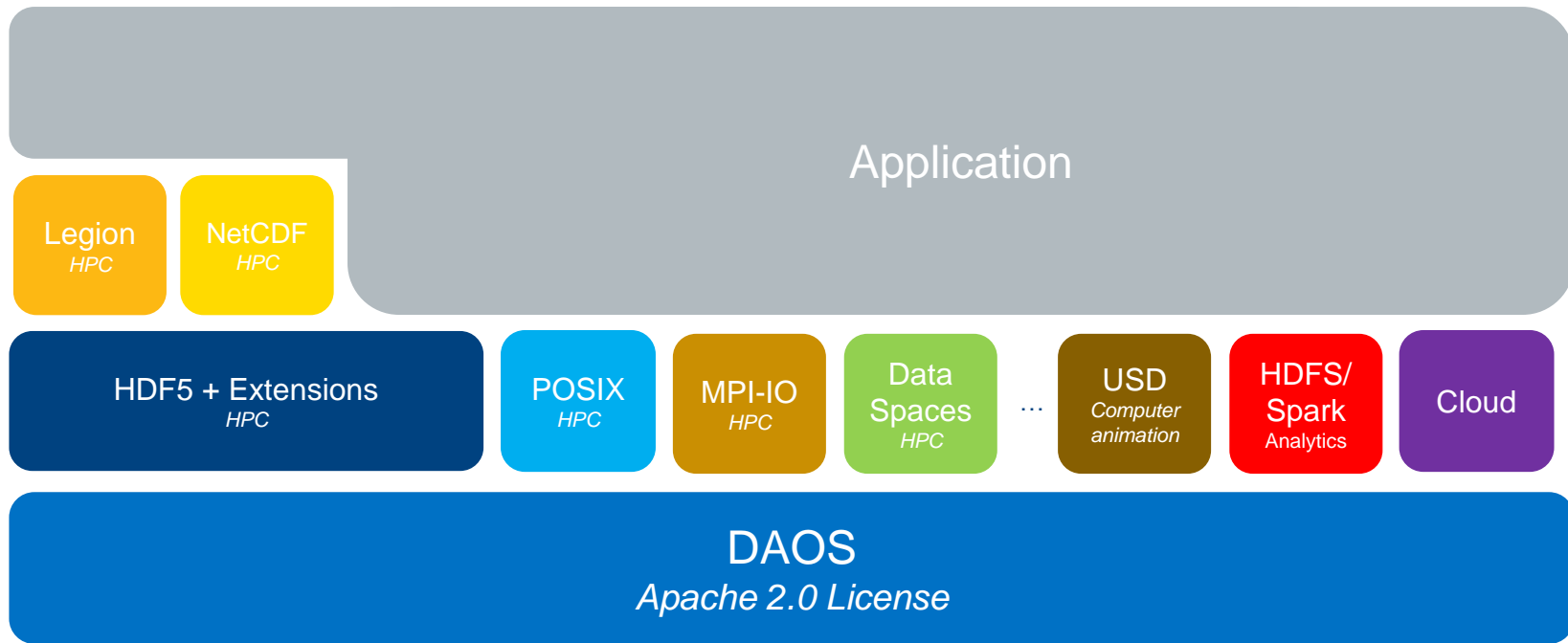
## Application failure

- Scalable distributed transaction exported to I/O middleware
- Automated recovery to roll back uncommitted changes

# DAOS Deployment



# DAOS Ecosystem



# Summary

Energy constrains our platform perf – top to bottom

Data movement consumes much of that energy

**CiM** saves energy and improves performance by optimizing & reducing data movement

Upcoming memory transitions present a great opportunity to exploit **CiM**



# Call-to-Action

- Choose key workloads in segments to drive definition
- Define ISA extensions & platform architecture
- Address multisolet/distributed memory challenges
- Implement performance libraries using CiM primitives
- Prototype platforms for implementation tradeoffs

Develop HW/SW deployment & enabling strategy

