OPENFABRICS ALLIANCE

# Burst Buffers

11TH ANNUAL
INTERNATIONAL OPENFABRICS SOFTWARE DEVELOPERS' WORKSHOP

#OFADevWorkshop

rpearson@cray.com

# Historical Background

# ~1984 'Burst Buffer'

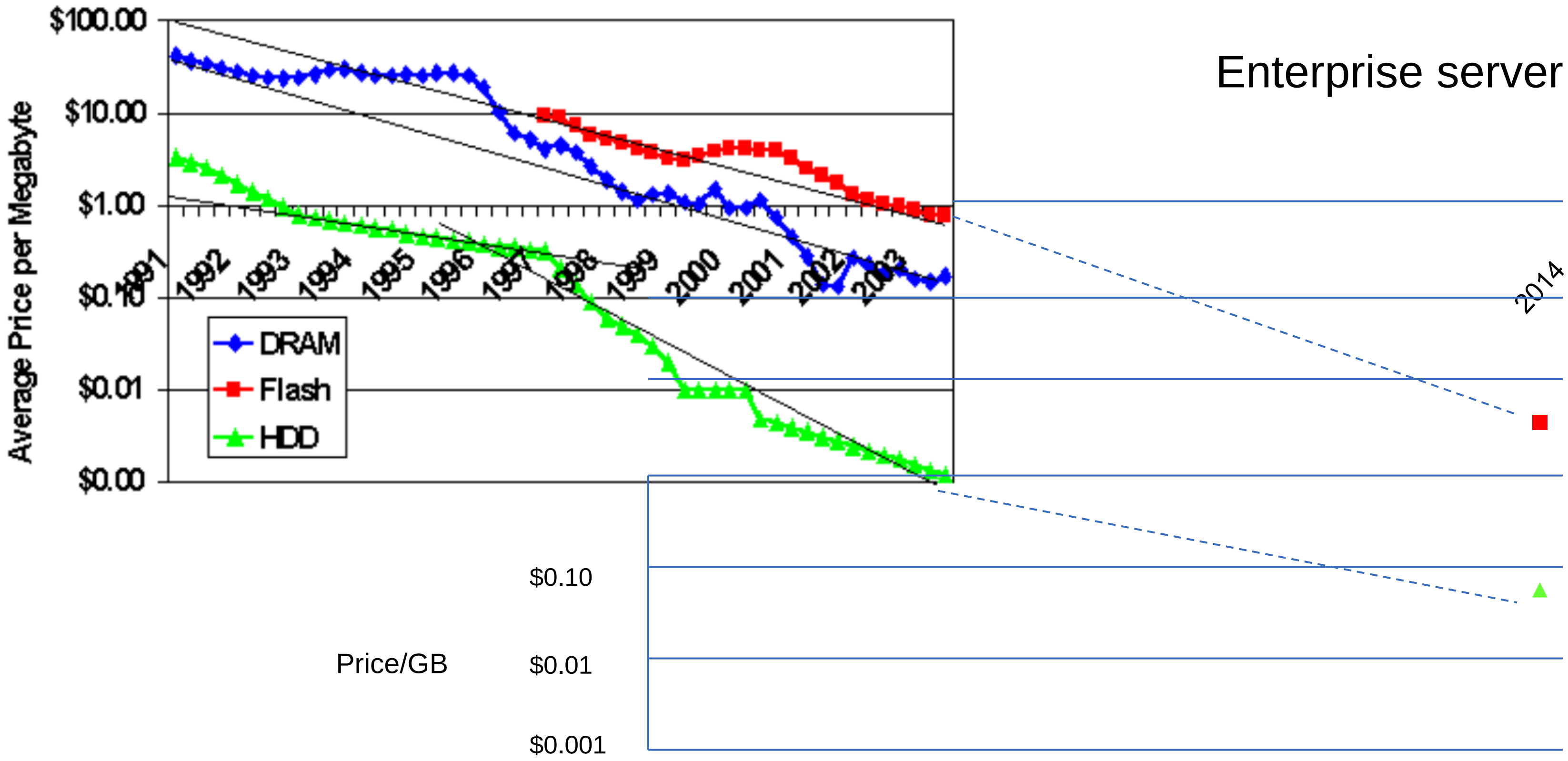
- System:
  - 4 nodes
  - 128 MB SRAM (16M words)
- IO:
  - 1.2 GB HDDs up to 32
  - 6 MB/s channel speed
- 'SSD':
  - 1024 MB (DRAM)
  - 1000 MB/s channel speed

And then … not much for 30 years …

# ~2015 'Burst Buffer'

- TN8 RFP required a 'burst buffer' solution
  - Trinity: checkpoint/restart to support 90% compute efficiency
  - NERSC8: support large job mix many with challenging IO
- => Cray Burst Buffer solution: aka 'DataWarp'
  - Moore's law has had 30 years to work its magic
  - Quickly expanding into most other mid to high procurements

# NV vs HDD - Cost/Capacity




HDD capacity is still scaling but BW and IOPs are near to flat

# Head to Head Currently

- Compare two typical high end devices:

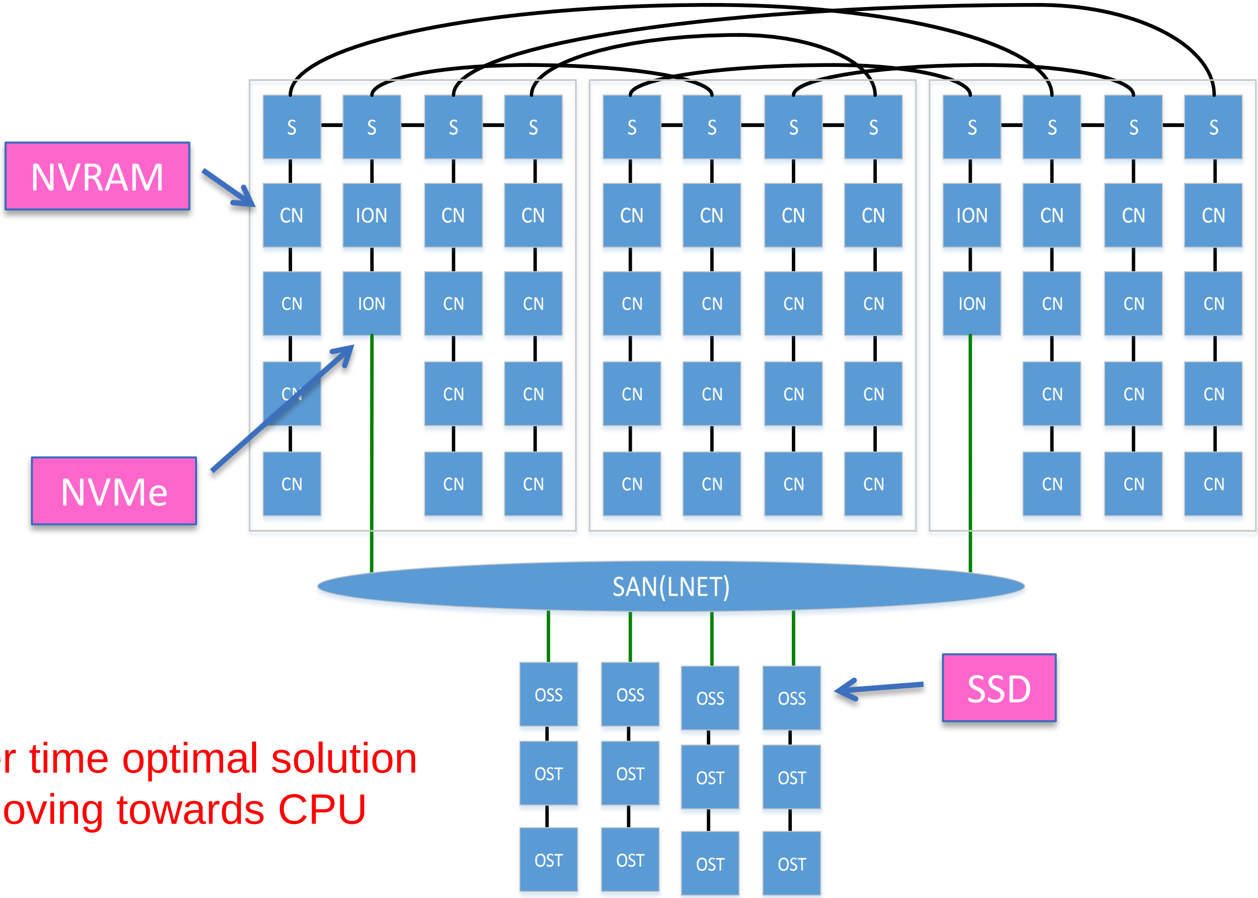
  - 6TB HDD:
    - Capacity ~= 6 TB Cap/$ ~= 20 GB/$
    - Seq BW ~= 150 MB/s BW/$ ~= 0.5 MB/s/$
    - IOPs ~= 150/s IOPs/$ ~= 0.5 IOP/s/$
    - Cost ~= $300
    - HDD lower % of PFS cost (30%)

  - 3TB NVMe SSD:
    - Capacity ~= 4TB Cap/$ ~= 0.5 GB/$
    - Seq BW ~= 3GB/s BW/$ ~= 0.4 MB/s/$
    - IOPs ~= 200,000/s IOPs/$ ~= 25 IOP/s/$
    - Cost ~= $8,000
    - SSD higher % of BB cost (70%)

Solution cost ratios
SSD/$:HDD/$
Cap: ~1/20X
BW: ~2X
IOPs: ~100X

# Hardware Architecture

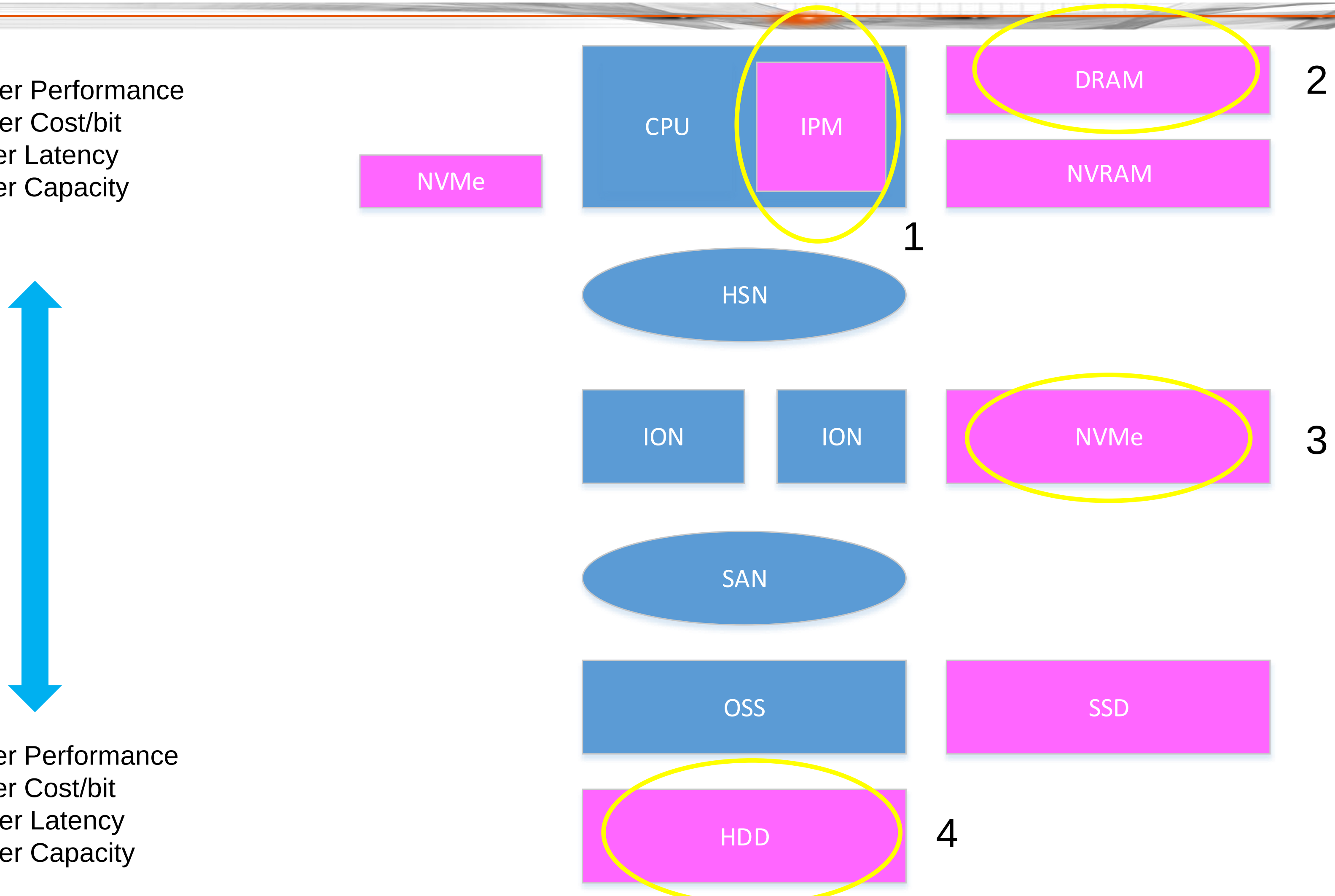
# HPC System with PFS




Over time optimal solution
is moving towards CPU

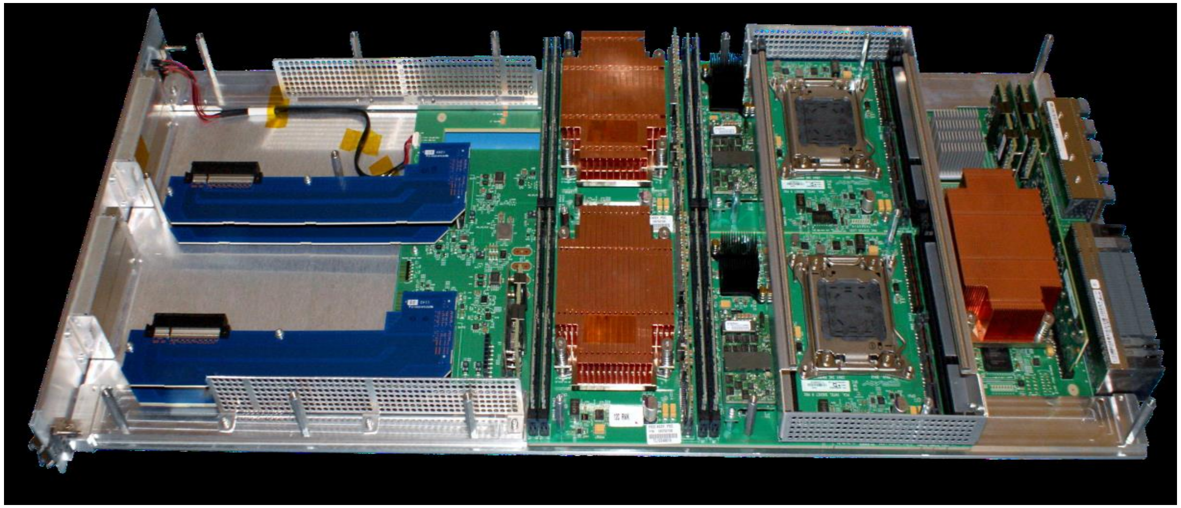
# Memory/Storage Hierarchy

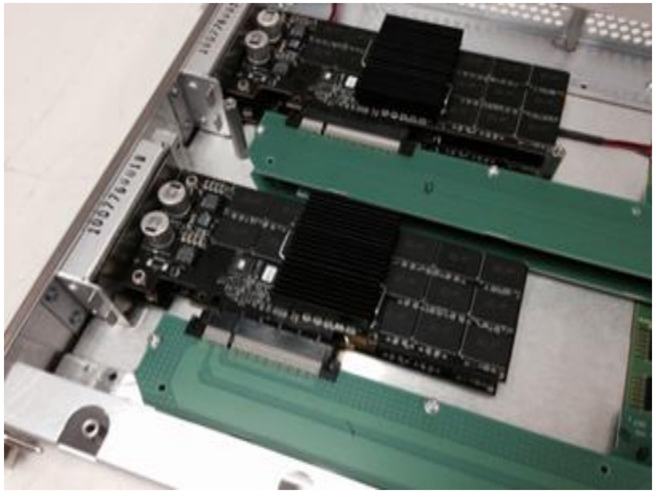
Higher Performance
Higher Cost/bit
Lower Latency
Lower Capacity

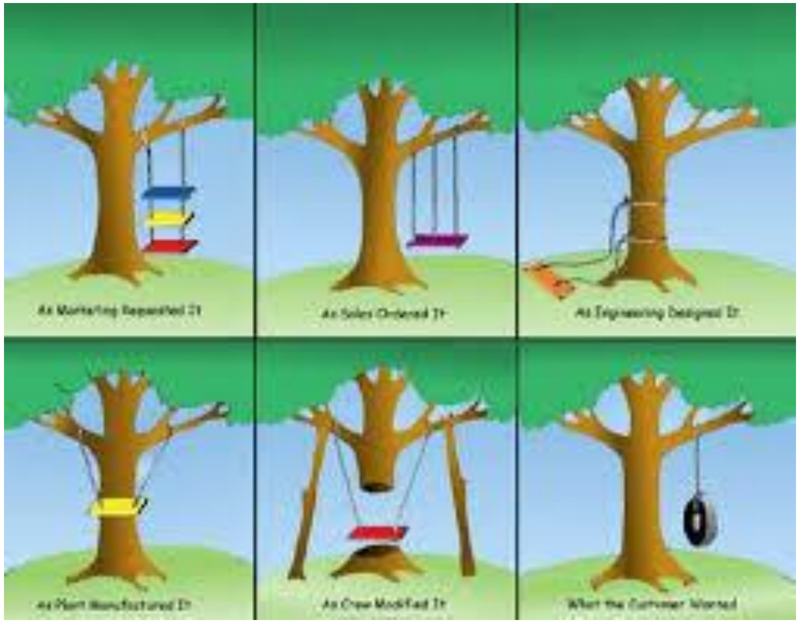



Lower Performance
Lower Cost/bit
Higher Latency
Higher Capacity

# IO/BB Blade








# Burst Buffer Use Cases

# TN8 'Burst Buffer' Use Case Requirements

- Checkpoint-Restart
  - Improves system efficiency for large and long jobs
- Pre Stage/Post Drain
  - Improves system efficiency by overlapping long IO
- Bursty IO Patterns
  - Shortens IO
- Private Storage
  - Virtual private disk or cache
- Shared Storage
  - Improve work flow management
  - Higher performance for critical data
- In Transit Analysis
  - Visualization or analysis as data is saved off

# Use Case: File System (PFS) Cache

- Cache for PFS data (ex. Lustre, GPFS, PanFS, …)
- Checkpoints, periodic output, intermediate results
  - Some data may never need to move to PFS
- Explicit movement of data to/from PFS
  - Application library API
  - Job commands API
- Implicit movement of data to/from PFS
  - Read ahead, write behind default behavior
  - API (library & command) available to control behavior

# Use Case: Application Scratch

- “out of core” algorithms
- Like a big /tmp
- Data typically never touches PFS
  - But it can

# Use Case: Shared Data

- Shared input (for example read-only DB or intermediate results)
- In-transit and ensemble analysis
- Accessed by multiple jobs concurrently or serially
  - Related jobs (e.g. WLM job dependencies)
  - Unrelated jobs
- Some data may never need to move to/from PFS

# Use Case: Swap

- Compute node swap
  - For apps that need it
  - Intended for limited or transient overcommit of memory
    - Swap is always much slower than local memory

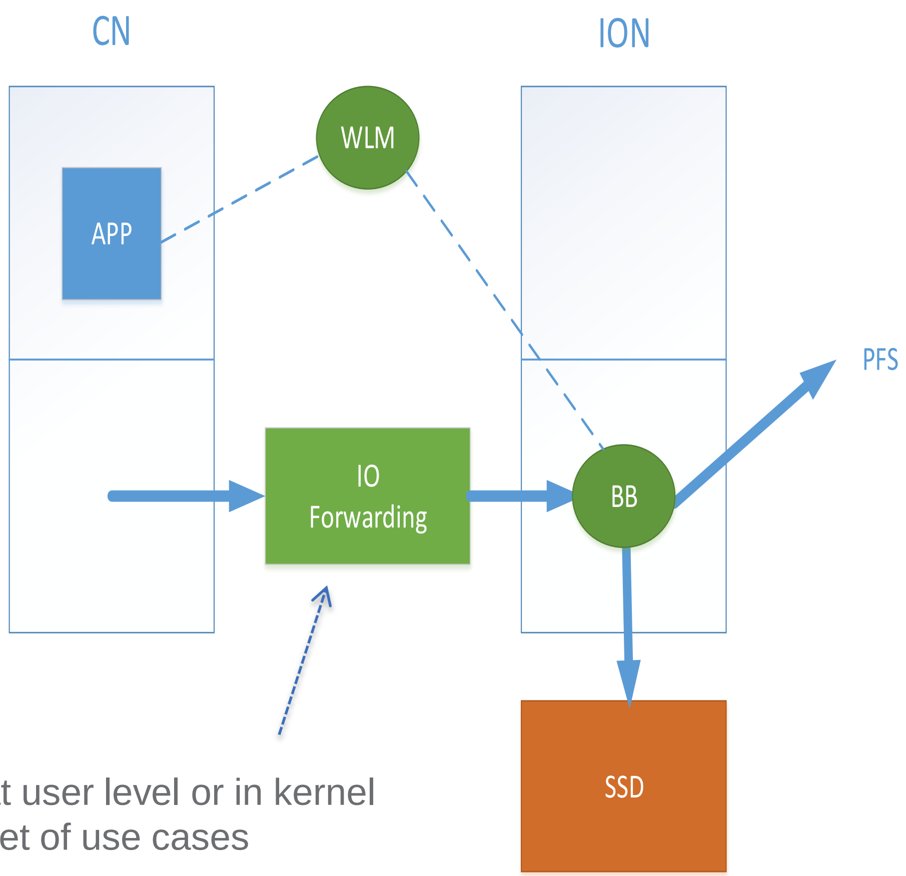
# Use Case:  Apps Running on BB

- Leverage local SSD performance (IOPs and BW)
  - For the data that is local
- MPMD app launch
  - Specific executable & ranks on BB nodes
- BB nodes used for this purpose are dedicated for this use only
  - They are not used for dynamically allocated BB instances as described below
  - They are treated as compute nodes, requested via the WLM and allocated to jobs exclusively
  - Administrator can add and remove nodes

# Motivation

- Place the SSDs directly on the HSN
  - Make use of a valuable existing resource
  - Avoid having to provision bandwidth to external SSDs
  - Match SSD bandwidth with HSN bandwidth
- Decouple application I/O from PFS I/O
  - Compute & PFS I/O overlap
  - Reduce elapsed time
- More cost effective PFS
  - Provision for capacity rather than bandwidth
  - SSD bandwidth is cheaper than PFS bandwidth
  - But SSD capacity is more expensive then PFS capacity

# High Level SW View




IO Forwarding can be done at user level or in kernel
In kernel supports broadest set of use cases

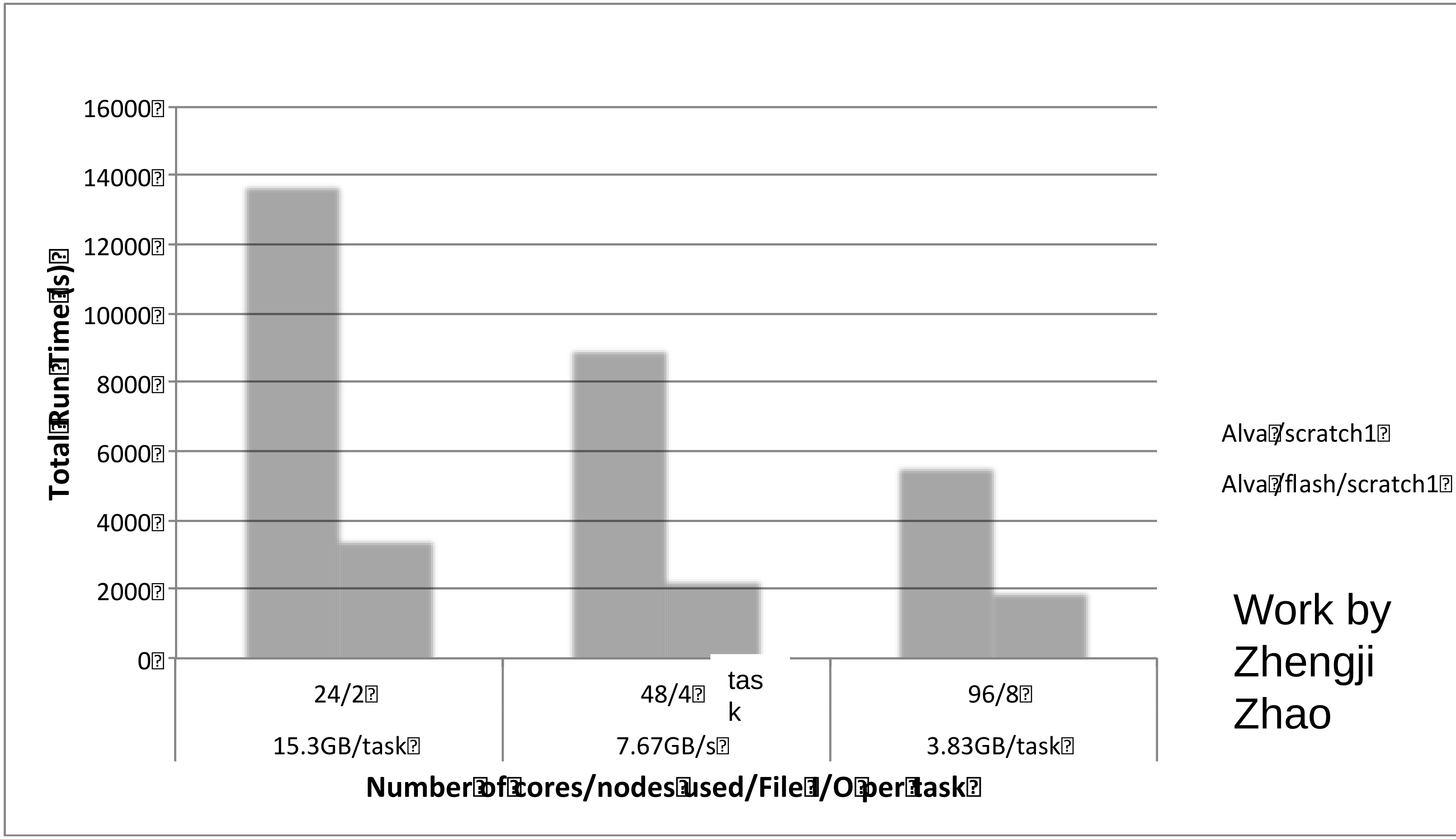
# Compute Node Access Modes

- Striped
  - Files are striped across all BB nodes assigned to an instance
  - Files are visible to all compute nodes using the instance
  - Aggregates both capacity and bandwidth per <u>file</u>
  - For scratch instance one BB node elected as the “MDS” server
    - For cached instances the PFS holds the metadata so every BB node can be an “MDS” server
- Private
  - Files are assigned to one BB node
  - Files are visible to only the compute node that created it
  - Aggregates both capacity and bandwidth per <u>instance</u>
  - Each BB nodes is an “MDS” server
- Load Balanced
  - Files are replicated (read only) on all BB nodes
  - Files are visible to all compute nodes using the instance
  - Aggregates the bandwidth per file
  - Each BB nodes is an “MDS” server

# Some Early Results (NERSC BB Testbed)

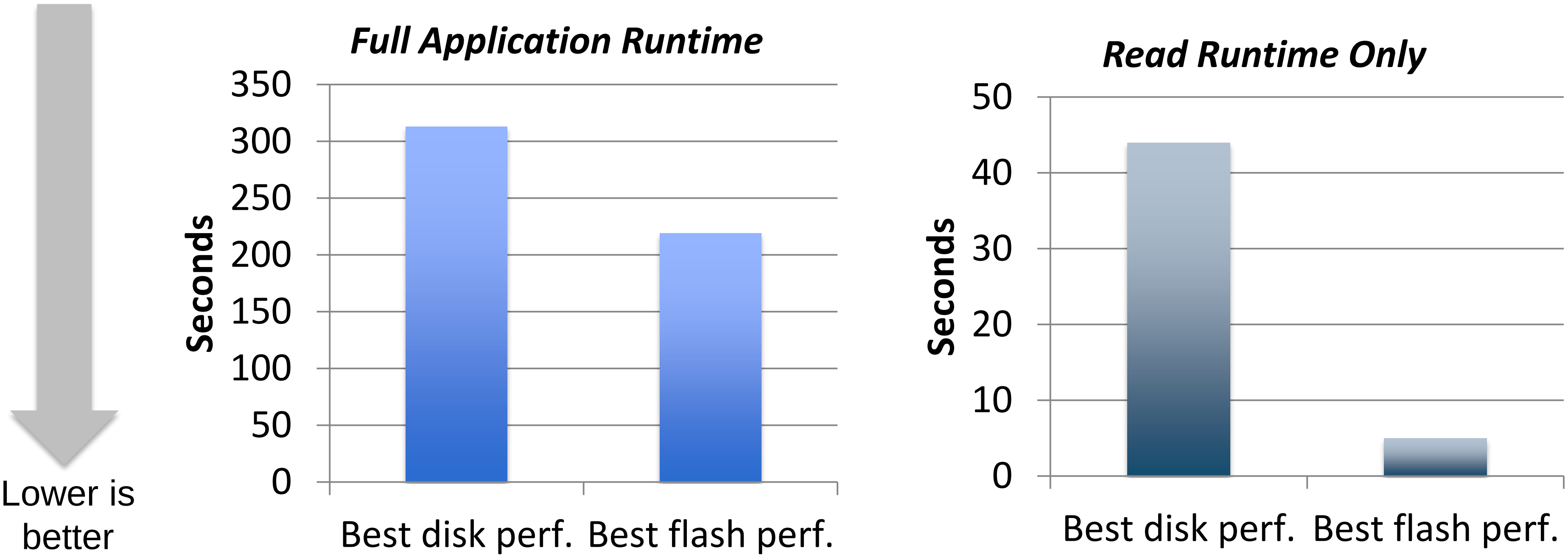
# NWChem Out-of-Core Performance: Flash vs Disk on BB testbed

Lower is better




- NWChem MP2 Semi-direct energy computation on 18 water cluster with aug-cc-pvdz basis set
- Geometry (18 water cluster) from A. Lagutschenkov, e.tal, *J. Chem. Phys.* **122**, 194310 (2005).

# TomoPy performance comparison between flash and disk file systems on BB testbed




- This I/O intensive application runtime improves by 40% with the only change switching from disk to flash
- Read performance is much better when using Flash: ~8-9x faster than disk
- Disk performance testing showed high variability (3x runtime), whereas the flash runs were very consistent (2% runtime difference)

Work by Chris Daley

# Thank You

OPENFABRICS ALLIANCE

11TH ANNUAL INTERNATIONAL OPENFABRICS SOFTWARE DEVELOPERS' WORKSHOP