Dramatic Changes in HPC Storage

Peter Braam

peter@braam.io

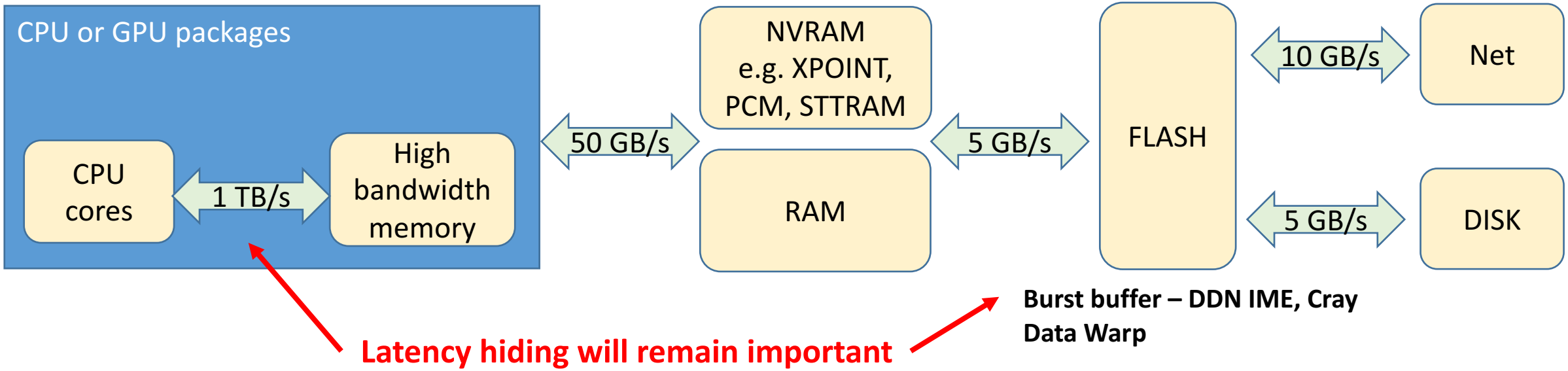
2017-12

Contents

- Storage Tiers
- Software and IO performance
- Notable deployments
- Emerging Storage Software
- Challenges and Conclusions

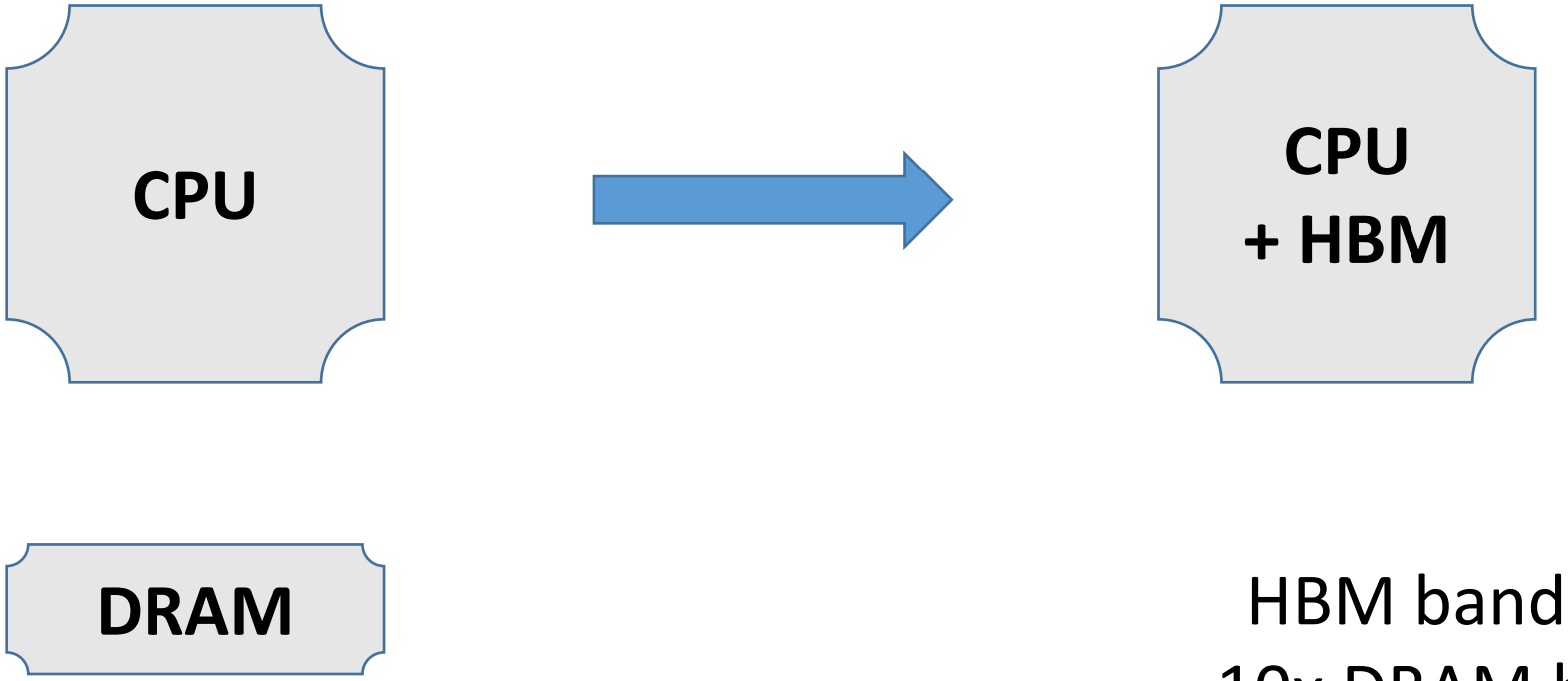
Speaker: introduced Lustre and other ideas. Presently independent researcher. Focus on future I/O. Consultant to SKA telescope effort.

Storage Tiers



Node BW (GB/sec)	**1 TB/s**	**50 GB/s /bus (NV for now 10x slower)**	**3 GB/s (/device)**	**5 GB/s (/enclosure)**
Cluster BW (TB/sec)	**1 PB/s**	**100 TB/s**	**10's TB/s**	**- 1 TB GB/s**
Software	**Language level**	**Language level / PGAS DAOS**	**Parallel file systems**	**Parallel FS Campaign Storage**
Key features	**transparent computation**	**transparent computation PGAS or ultra-fast storage**	**name space scientific formats FS style container**	**bulk data movement - many files - subtrees of MD**

Most Important Architecture Shift



HBM bandwidth ~=
10x DRAM bandwidth

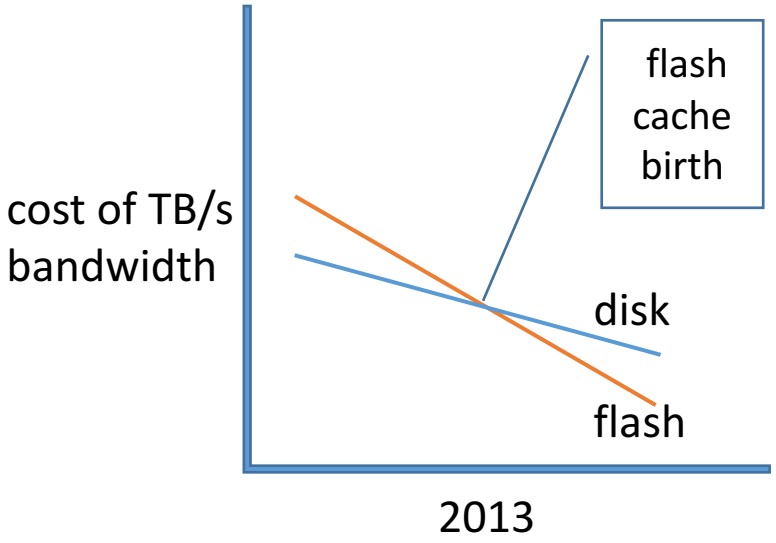
Everything shifts scale .....

Tier $

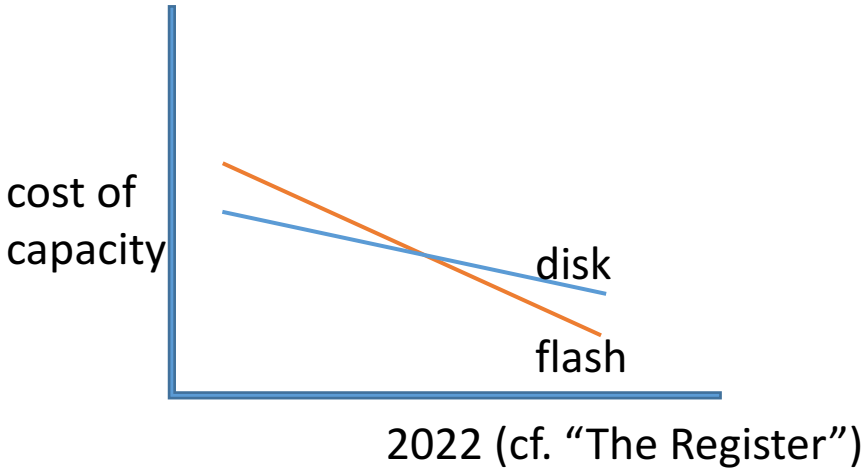
	High Bandwidth Memory	RAM	NVRAM XPOINT / PCM / STTRAM (1 bus)	FLASH (1/3 device)	DISK (10 disks)	TAPE (2 drives)
BW Cost $/ (GB/s)	?	$10	>$10	$200	$2K	$30K
Capacity Cost $/GB	~ RAM?	$8	<$8	$0.3	$0.02	$0.01

Economic Models

⇒***Flash cache:*** *Disk vs Flash for bandwidth?*



⇒***Solid State Capacity Tier?***



⇒***Staging:*** *At what point is asynchronous staging in burst buffer cheaper than waiting for IO? [workload dependent answer]*

⇒***NVRAM****: when does extreme (read) bandwidth pay off?*

⇒***Archive:*** *Trade-off between spin-down (SMR) disk archives vs tape archives*

⇒***Write contents of RAM*** *to storage in ~5 minutes?*

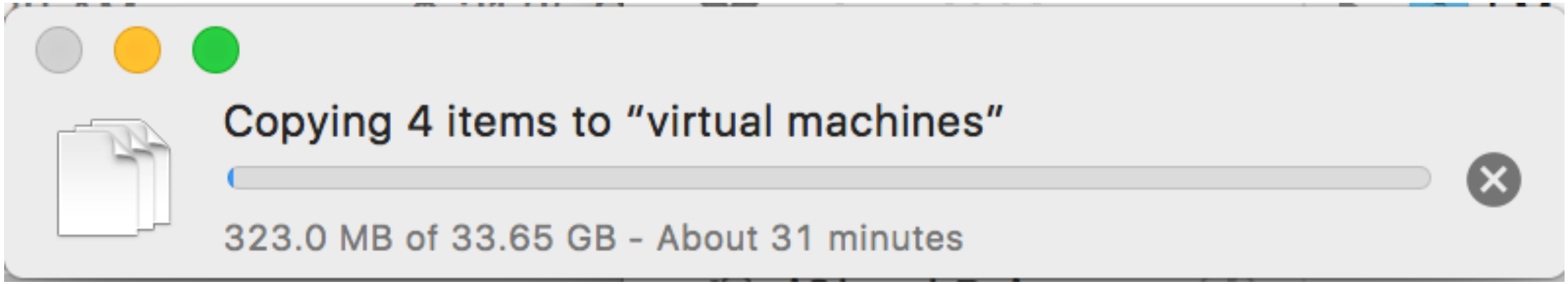
⇒***Cost of using vs owning*** *capacity?*

- Systems will have more tiers of storage
- Moving data between tiers will be essential
- Prepare for further hardware improvements!

- Should really use total cost of ownership
- Should compare with cloud

Software & IO Performance

Is this still true?



3% of HW performance (SSD - 2014)

Evolution of understanding

2000-2005

parallel file systems: impressive benchmarks, problems for applications

2005-2012

ADIOS / PLFS: data layout, aggregation to the rescue

2010

Object storage has scalability, lacks distributed parallelism for HPC

2013

Staging for harder problems, transactions for workflows, log structures

3 desirable API's

- File System
- Object
- Rich data library – e.g. HDF5

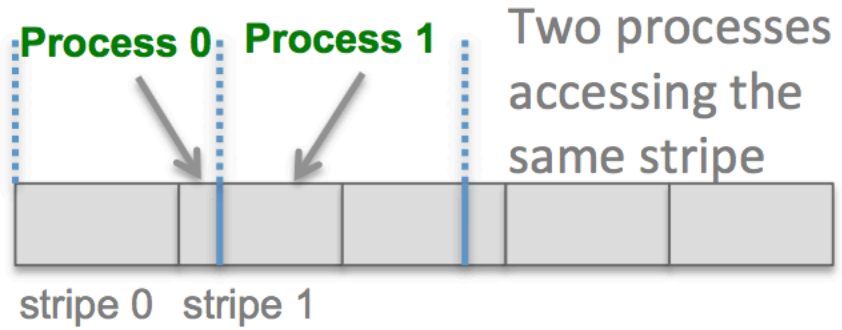
! **Storage will offer FS & Object & HDF5 API's to users**

Parallel File System Trouble

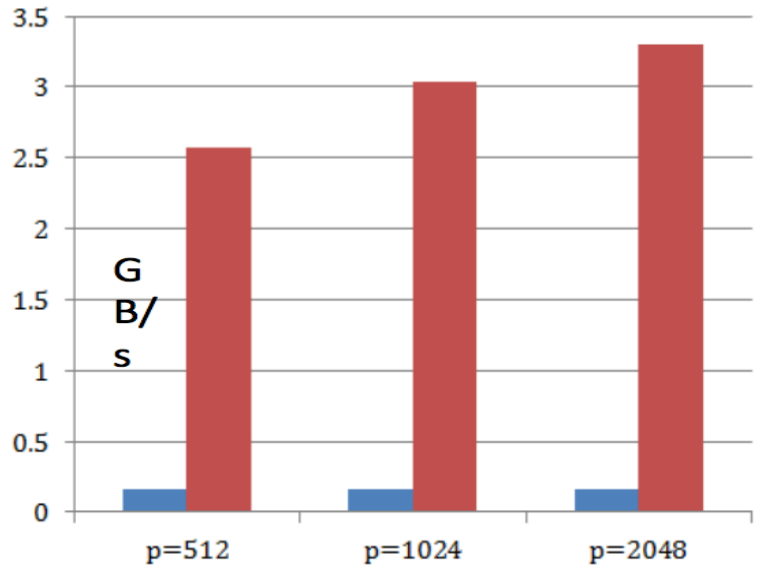
Too many files

Too many separate bits of data in one file

Wrong alignment



- ~ 2010: ADIOS library addresses these issues



Tianhe-1A, ADIOS AMR .vs. Original IO

What does ADIOS really do?

What needs to be written?

- New API – not POSIX, very simple
- Form group of processes
- Declare what items and how many need to be read / written
- Do IO asynchronously

How will it be written?

- External specification of layout
- Plugins for storage infrastructure

!

Could this beautiful understanding become an automatic runtime optimization?

IO Library Comparison

Operation	POSIX	MPI-IO	PNetCDF	HDF5	NetCDF4	ADIOS
Noncontiguous Memory	X	X	X	X	X	X
Collective I/O		X	X	X	X	X
Portable Format		X	X	X	X	X
Self Describing			X	X	X	X
Attributes			X	X	X	X
Chunking				X	X	X
Hierarchy				X	X	X
Sub Files			X	X		X
Resilient Files						X
Streams/Staging						X
Customize data transforms				X	X	X
Parallel data compression						X

Storing semi-structured data

HDF5

- HPC standard for arrays, KV store, sub-file in file and more
- Surprisingly small overlap with custom data layout for cloud

- Other formats (e.g. NetCDF) starting to leverage HDF5
- HDF5 beginning to use sophisticated lower layers (e.g. ADIOS)

Desired: Integrated HDF5 solutions. POSIX must stay alongside (legacy)

Write back caches – logs & containers

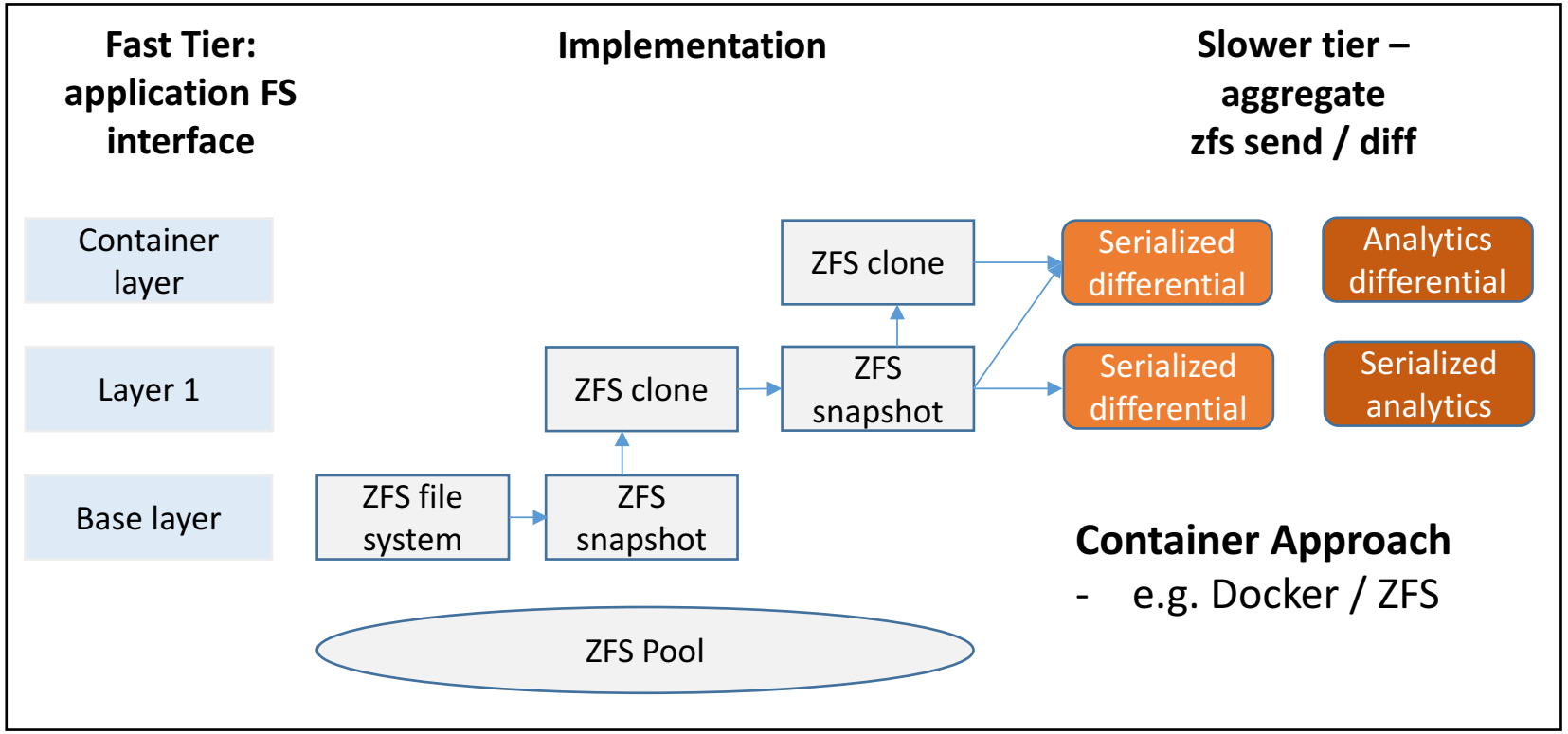
Hierarchy has fast & slow side

Hence:

- Create fine grained data
- Pack in a “log”
- Move log
- Avoid small writes

Examples:

1



2

DDN IME software
- scalable log based storage system

Cray Datawarp

Workflows and distributed transactions

Processing is evolving to workflows – e.g. *simulation -> analytics*

Coordination of IO in the workflow: requires group transactions

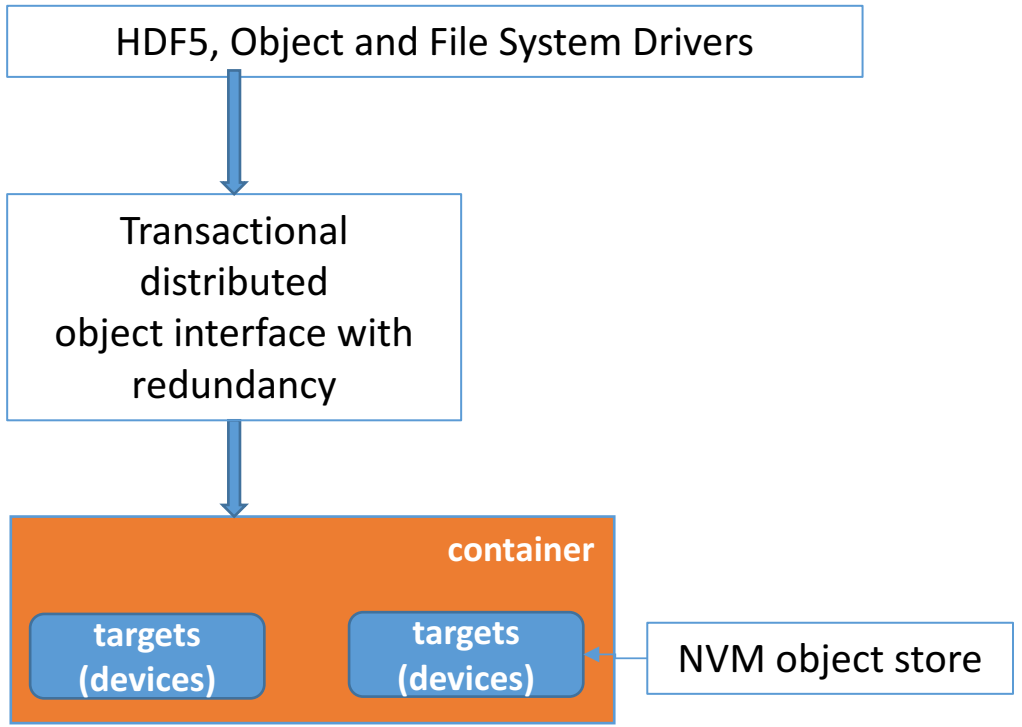
1 **Precursors** | Lustre metadata epochs | Object stores with snapshots | D2T – flexible API

2 **DAOS – DOE/Intel/HDF5 group collaboration**

2012 – 2015: initial prototype based on Lustre / ZFS
2015 - : 2nd pre-production NVM implementation

Key capabilities:
- IO Process Groups with distributed transactions
- Scales to 100K's servers, 1B client processes
- Low sw overhead, Redundancy, NVM emphasis

Application:
- Underpinning for HDF5 and legacy file system
- Probably not so easy to use directly



Partial File Staging

Problem

Hardest IO problems: **massive I/O level exchange of small data**
E.g. in adaptive meshes refinement (AMR)

Solution

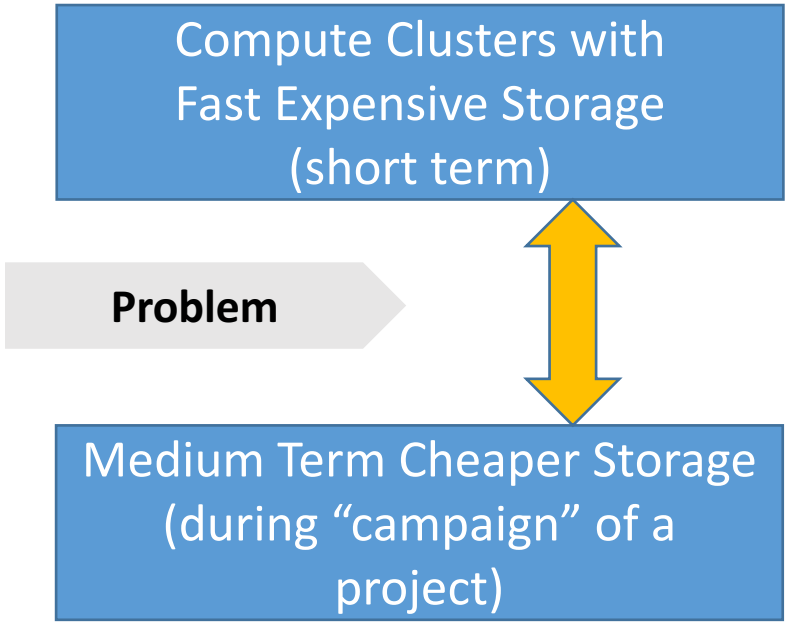
Principle: Data Staging

Avoid reading from each node
Re-organize data, and cache in the network
Read from the cache (avoids many small reads)

Implementation

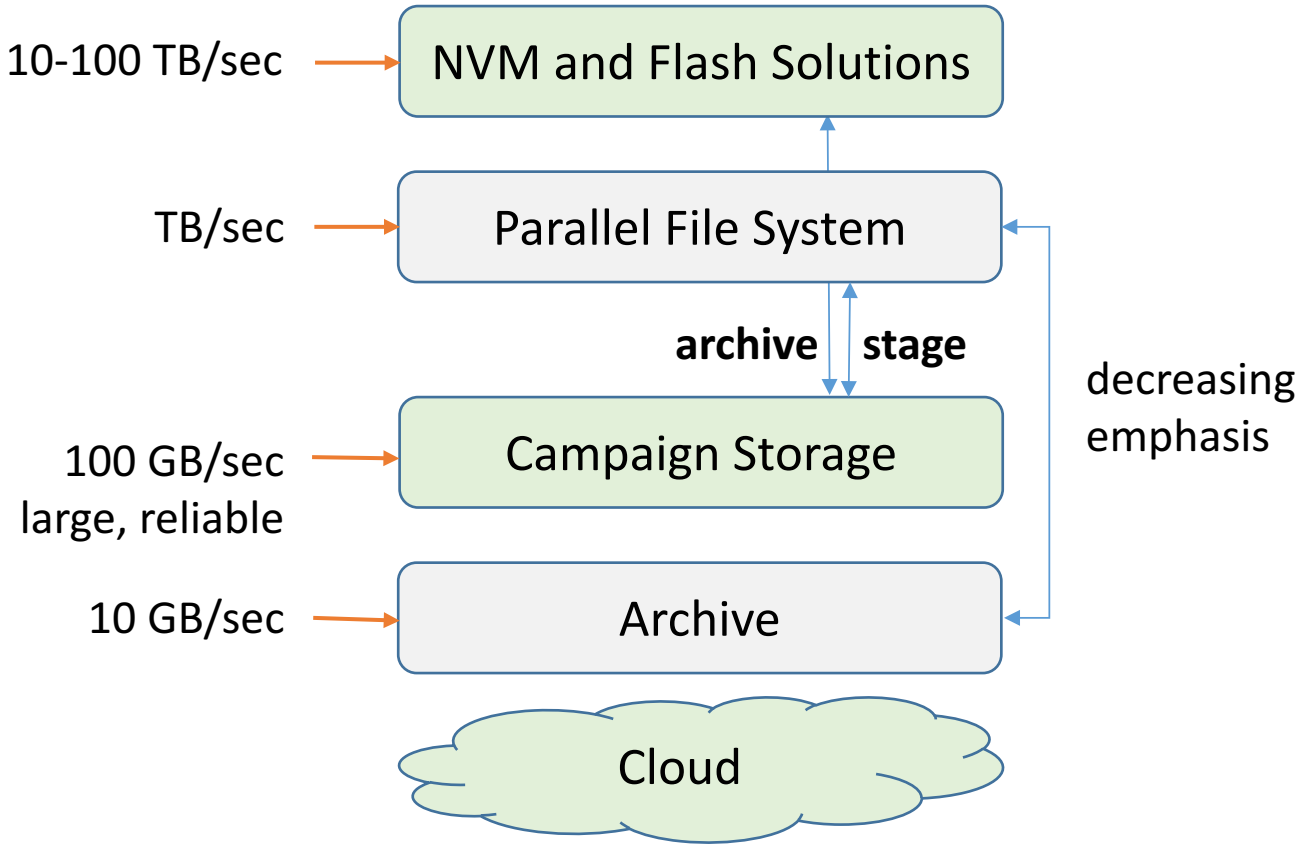
ADIOS with data spaces
Custom libraries – e.g. PUM library from LRZ

Whole file staging and archiving



Alternatives:
1. HSM to object storage
2. IPFS, upspin.io

Campaign Storage (MARFS from LANL)



Recap of mechanisms

1. Layouts
2. Buffering
3. Transactions
4. Write back caching
5. Partial and whole file staging

Not all of these are readily
available in deployed IO solutions

Notable Deployments

Sample deployments 2000 - 2017

2000 – 2017

Racks with compute nodes
Disk enclosures
up to 1TB/sec, disk size = 100x RAM
Lustre / GPFS

2015-

Flash caches: 10x RAM, 5 TB/sec
Lustre / GPFS / DDN IME / Cray DataWarp

Future Deployments

2020 - Capability System at LANL

- Many Lustre/ZFS flash servers?
- 5-10 TB/sec, 2-5 PB memory, ~100 PB flash
- Maybe no burst buffer
- Secondary tier: Campaign Storage

2025 SKA

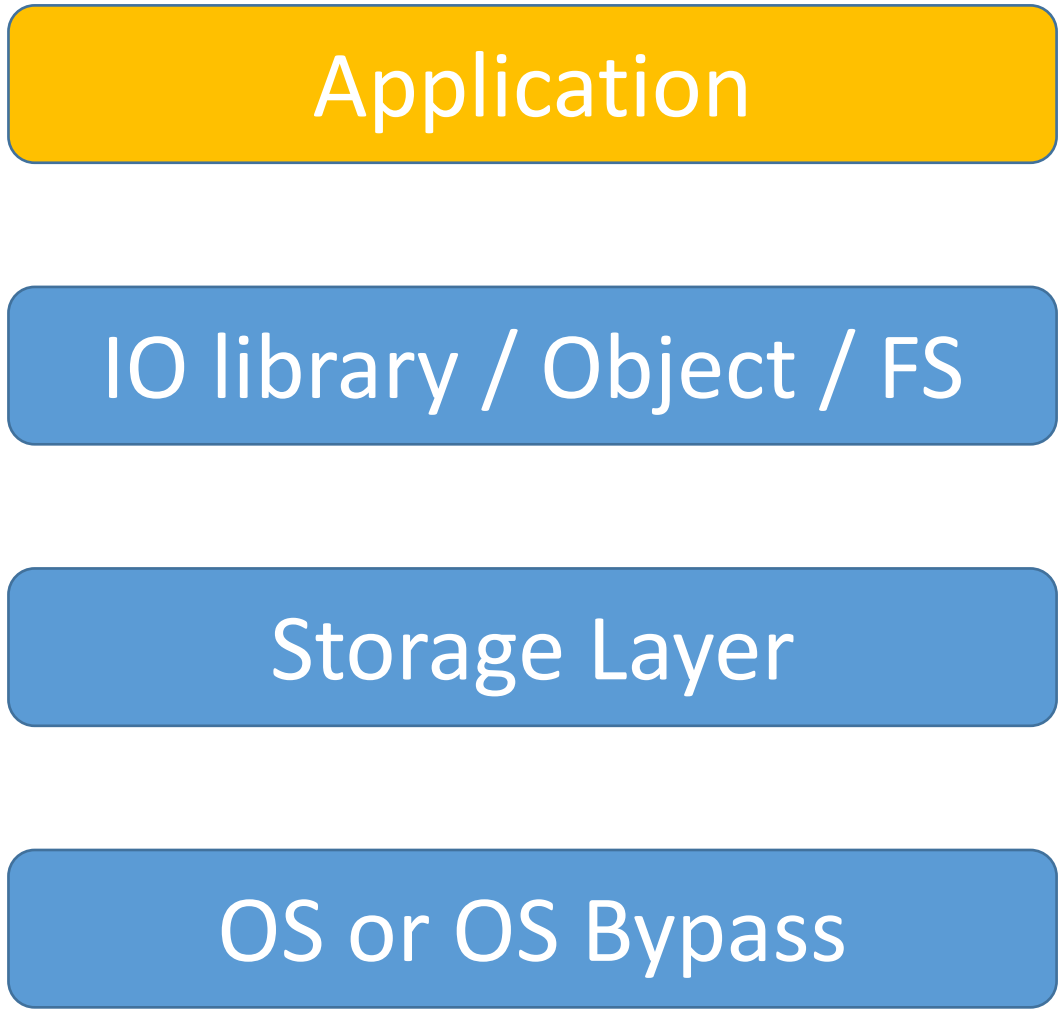
- 10 TB/sec read, 1TB/sec write,
- 1EB output / archive (~0.5 PB/day)
- 200PB/sec memory bandwidth

2022 US Exascale

- 10PB RAM
- IO: ½ memory in 3 mins ~ 30TB/sec
- Two tiers: capacity and performance
- Total storage - 0.5 EB
- 1M stats / sec,
- namespace ops < 100K/sec

Emerging Storage Software

Layers


Application
IO library / Object / FS
Storage Layer
OS or OS Bypass

HDF5 ?

Exa-scale storage layer candidates

1. Evolving parallel file system technology – Lustre / GPFS
2. DAOS – objects for NVRAM data & metadata, transactions
3. Emerging research projects (CEPH/Empress, SAGE, ADIOS)
4. Evolving or new proprietary solution

! New system will be costly to develop
Perhaps too many offerings may solidify incumbents

Challenges & Conclusions

Technical Questions / Suggestions

- Leverage local storage
- Direct IO to accelerators
- ML for layout and scheduling
- Declarative storage software
- NV-transactional memory

Challenges

Establish standard API

library (HDF5),
object (which?) api and
POSIX as a new standard?

Could a cloud provider like HPC on Azure push this?

Get full parallel FS alternative

Conclusions

- Beautiful simpler, convincing systems are emerging
- Hardware developments have been fantastic

Set our eyes for 2020's on distributed storage
that we can truly love using
with 1PB/sec IO, 1B/s namespace creates

Thank you