Memory Technologies & Distributed Storage

Peter Braam

peter@braam.io

2018-05

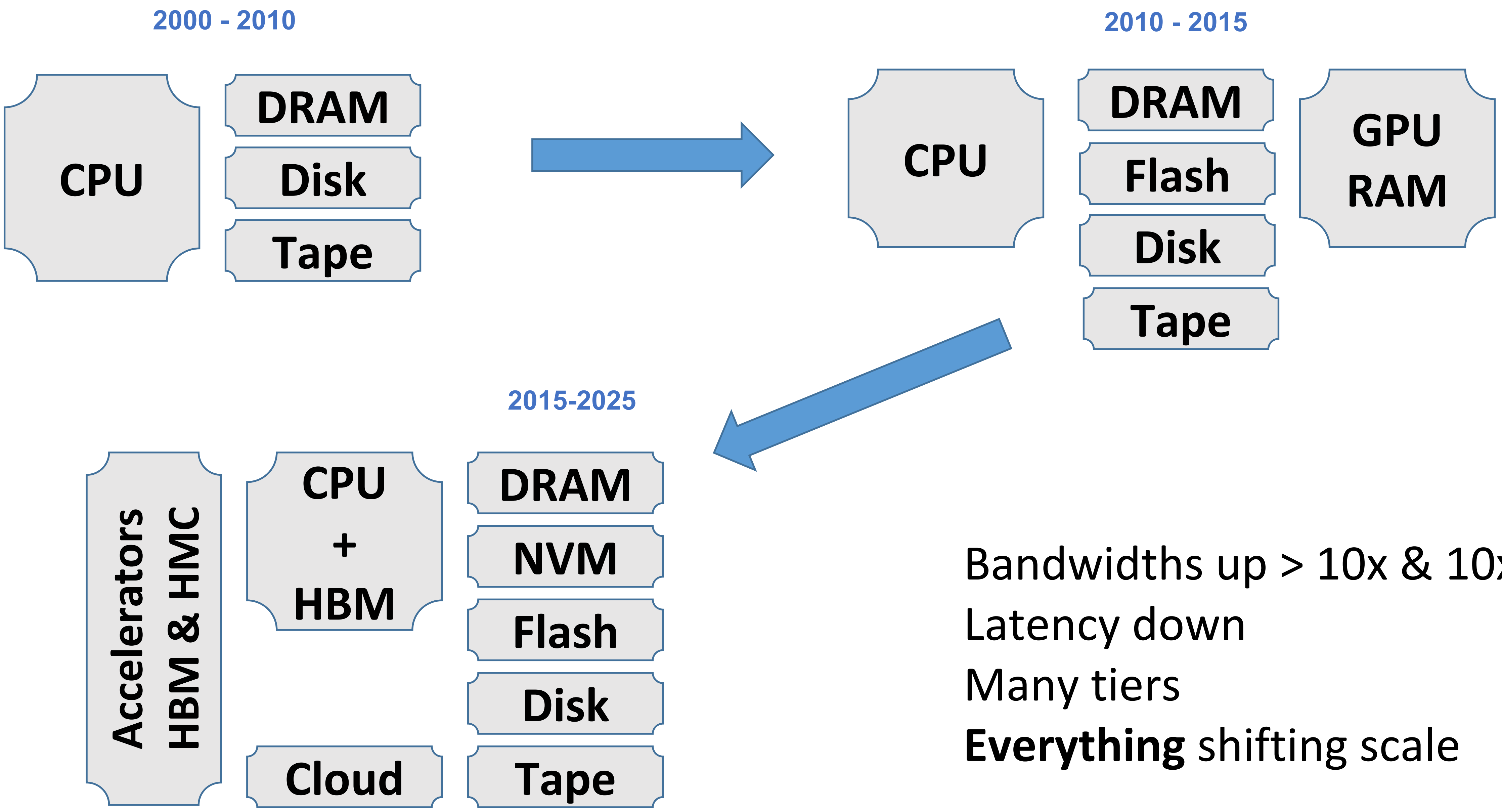
Contents

2000
reflection
storage for
HPC
2025
research

Speaker: Storage architect and independent researcher. Introduced Lustre and other ideas.   Work on SKA telescope effort with Cambridge University.

Storage Tiers

Architecture expanding dramatically



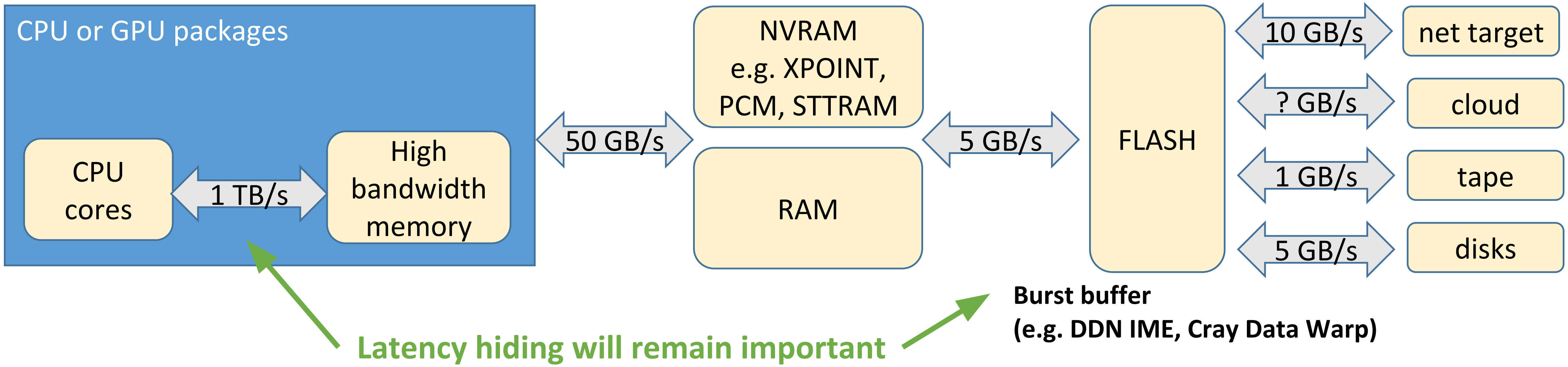
Bandwidths up > 10x & 10x

Latency down

Many tiers

Everything shifting scale

.....



Node BW (GB/sec)	**1 TB/s**	**50 GB/s /bus (NV write ~10x slower)**	**3 GB/s (/device)**	**5 GB/s (/enclosure)**
Latency	**100 ns**	**100 ns (NV Write 1 us)**	**10 - 100 us**	**10 ms - 1 min**
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**
Purpose	**transparent computation**	**transparent computation PGAS and ultra-fast storage**	**name space scientific formats FS style container**	**bulk data movement - many files - subtrees of MD**

NVM and software overhead

NVM device access:	1**us**
NVMe flash read access:	100us

There is a new problem …

Software needs 10-100x speedup

User/kernel:	1us
Network stack:	10us
FS + device stack call:	100us

DAX like mechanisms

New challenges for kernel MM

Considerations About Tiers

Migration

RAM tiers are for computation

migrate **pointers, cache lines, pages**

on large memory ranges when more effective

Flash / disk is 5x faster with large IO

form **containers** at nvRAM level

Program memory layout: re-usable?

HDF5 is a file internal layout specification

Is the next step - NumPy?

Persistence

NVRAM will be the fastest storage device

most demanding storage applications

but - write is not yet like RAM - caching!!

Tier $

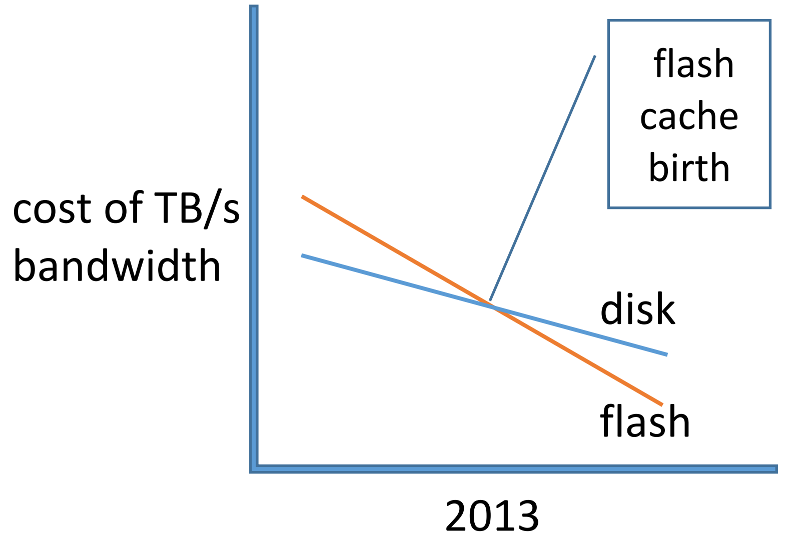
	High Bandwidth Memory	RAM	NVRAM XPOINT / PCM / STTRAM (1 bus)	FLASH (1/3 device)	DISK (10 disks)	TAPE (2 drives)
BW Cost $/ (GB/s)	~ RAM?	$10	>$10	$200	$2K	$30K
Capacity Cost $/GB	~ RAM?	$8	<$8	$0.3	$0.02	$0.01

⟷

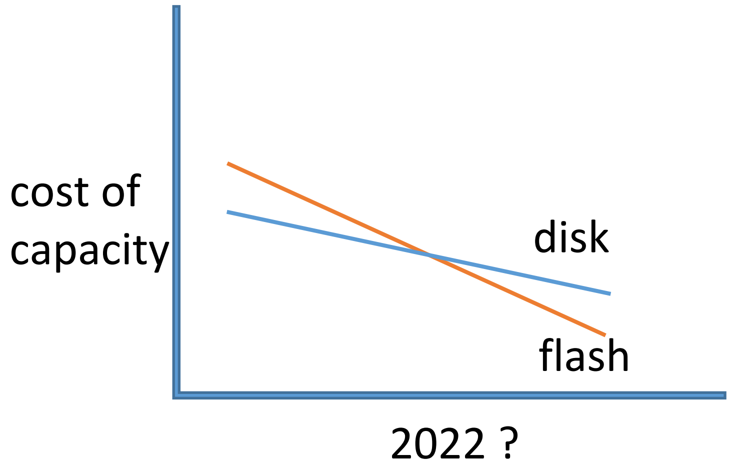
1000x cost difference

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]*

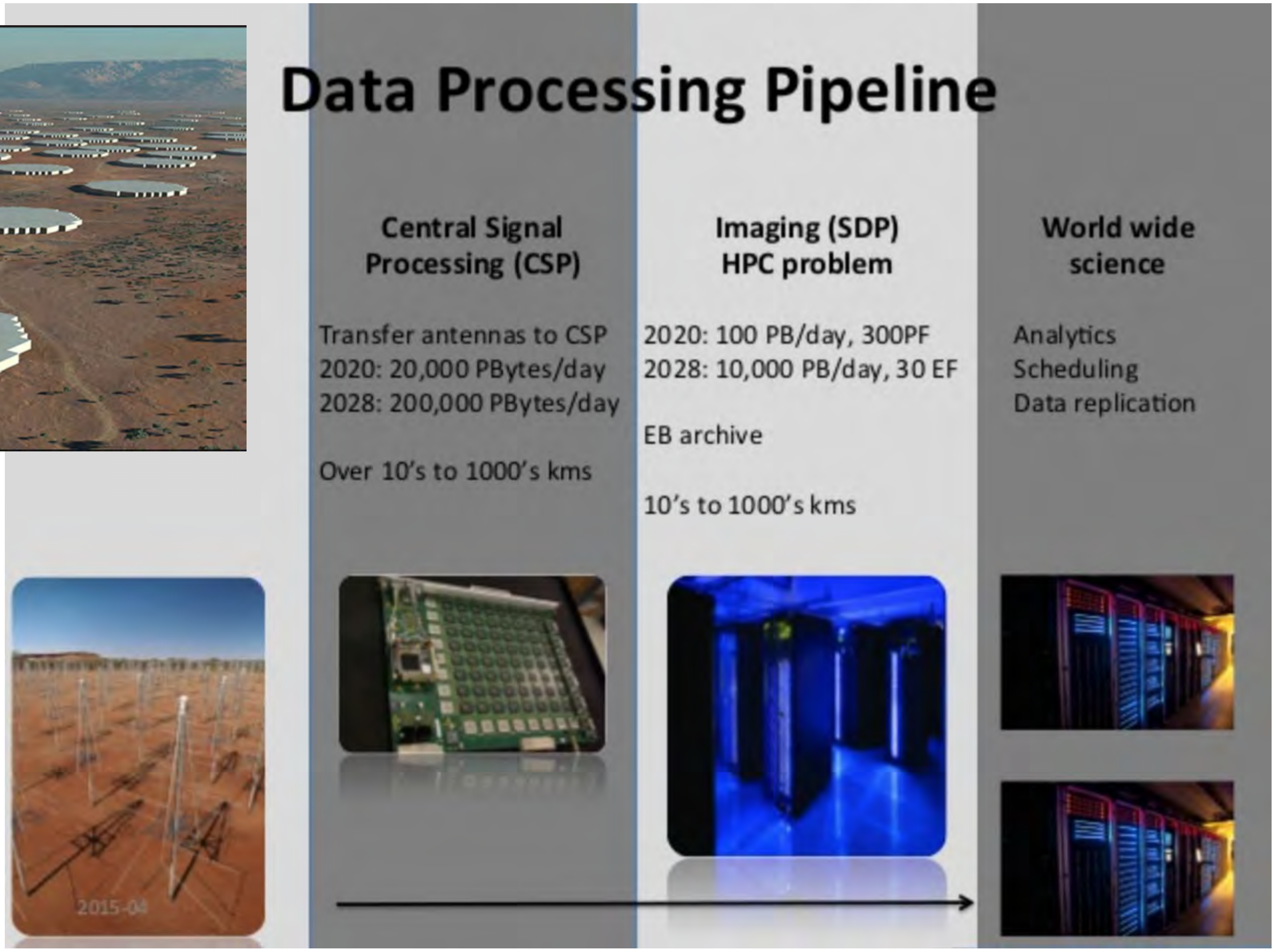
⇒ ***NVRAM****: when does extreme (read) bandwidth pay off?*

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

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

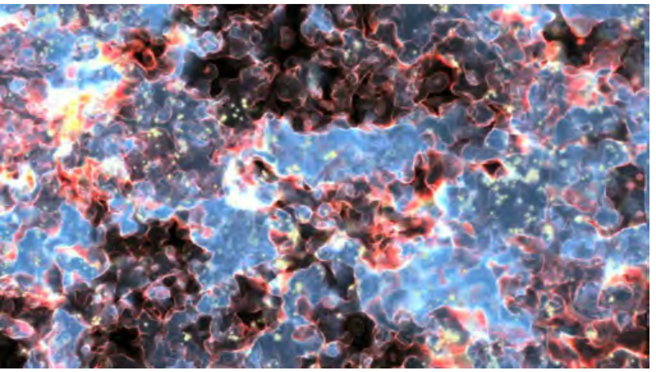
⇒ ***Cost of using vs owning*** *capacity?*

SKA telescope



Data Processing Pipeline

Central Signal Processing (CSP)	Imaging (SDP) HPC problem	World wide science
Transfer antennas to CSP 2020: 20,000 PBytes/day 2028: 200,000 PBytes/day Over 10's to 1000's kms	2020: 100 PB/day, 300PF 2028: 10,000 PB/day, 30 EF EB archive 10's to 1000's kms	Analytics Scheduling Data replication



Interesting extreme economic example

SKA radio telescope requires ~300PF/s compute with

200 PB/sec memory bandwidth (to create images etc):

- upcoming HBM product may support 50 pJ / byte
- so energy consumption 10^{17} x 50×10^{-12} = 5 MW
- prior to awareness this was the entire energy budget

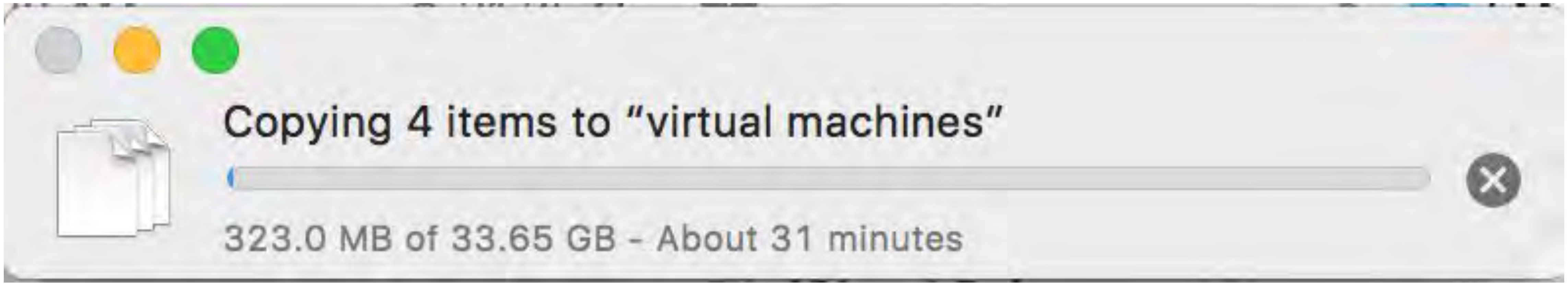
! □

- Systems will have more tiers of storage
- Moving data between tiers will be essential
- Transparency between tiers
- Handling the fastest tiers

- Richer cost model

Software & IO Performance

The Drama



Copying 4 items to “virtual machines”

323.0 MB of 33.65 GB - About 31 minutes

3% efficiency

Evolution of understanding

2000-2005

- parallel file systems: impressive benchmarks but problems for applications

2005-2012

- ADIOS / PLFS: data layout & aggregation to the rescue

2010

- object storage: has scalability, lacks distributed, shared IO, names

2013

- Staging for harder problems, transactions for workflows, log structures

3 desirable API's

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

!

Storage will offer all 3 API's: FS - Object - HDF5

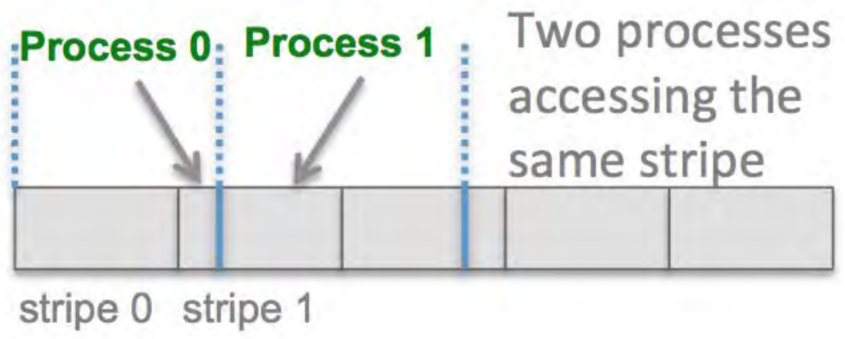
Parallel File System Trouble

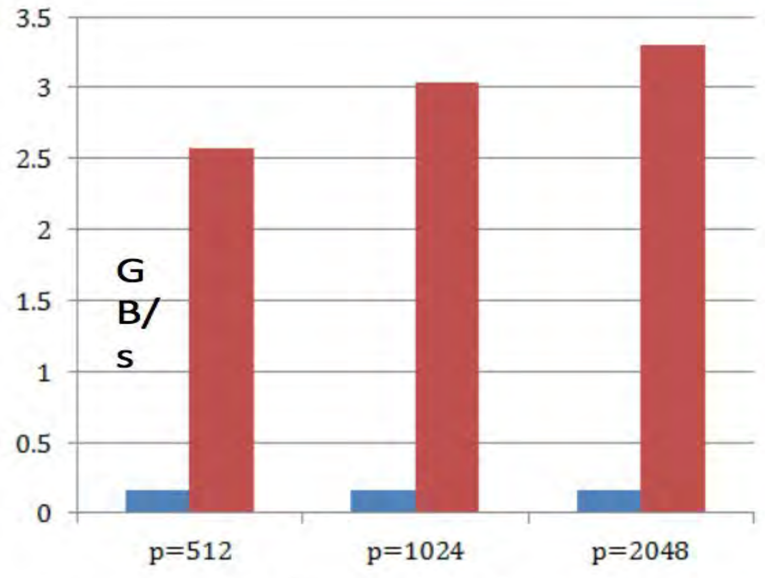
Too many files

Too many separate bits of data in one file

Wrong alignment

- ~ 2010: ADIOS library addresses most 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 data be written?

- External specification of layout
- Plugins for storage infrastructure
- Align data & stripes & devices

! □

IOR’s successful data layout as an automatic optimization?

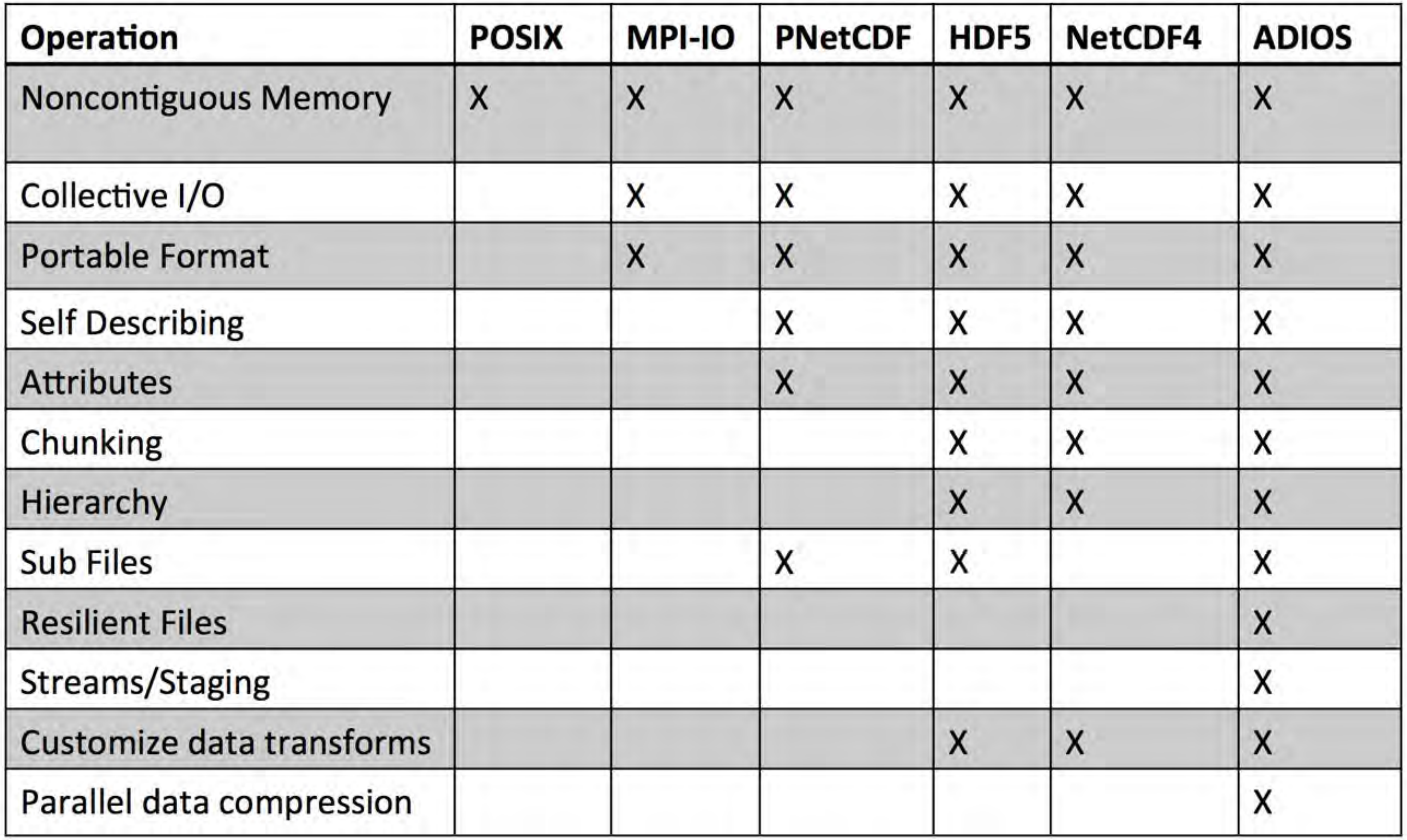
HDF5 - storing semi-structured data

- 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: Integrate HDF5 solutions with objects as well as POSIX.

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

Oakridge National Laboratory

Write back caches – logs & containers

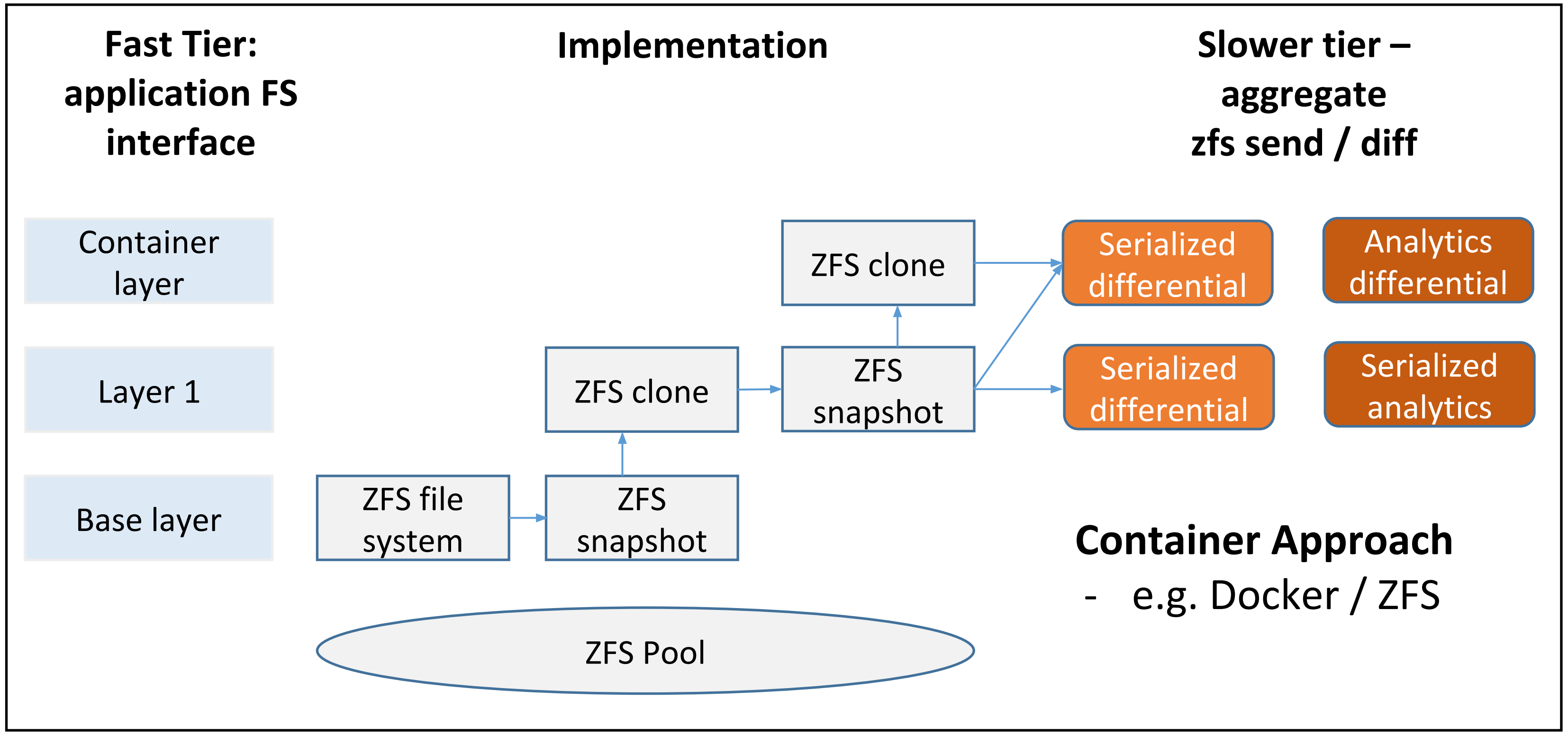
Hierarchy has fast & slow side

Hence:

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

Examples:

1



Container Approach

- e.g. Docker / ZFS

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

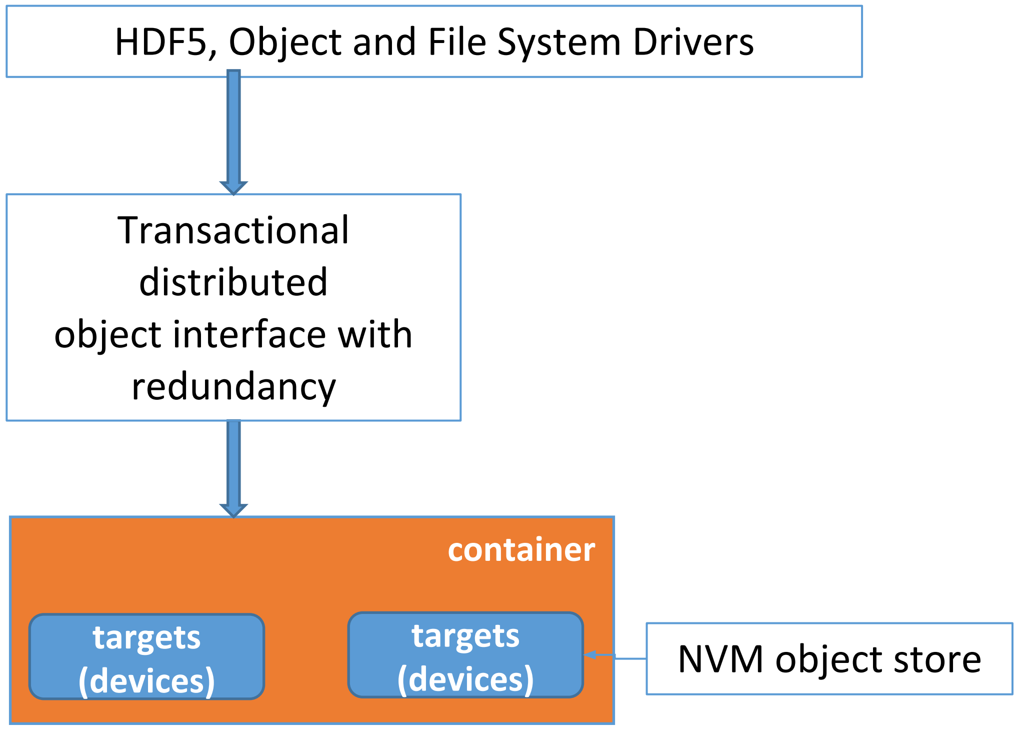
2015 - : 2nd pre-production NVM implementation

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


HDF5, Object and File System Drivers
Transactional distributed object interface with redundancy
container
targets (devices)
targets (devices)
NVM object store

Partial File Staging

Problem

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

Solution

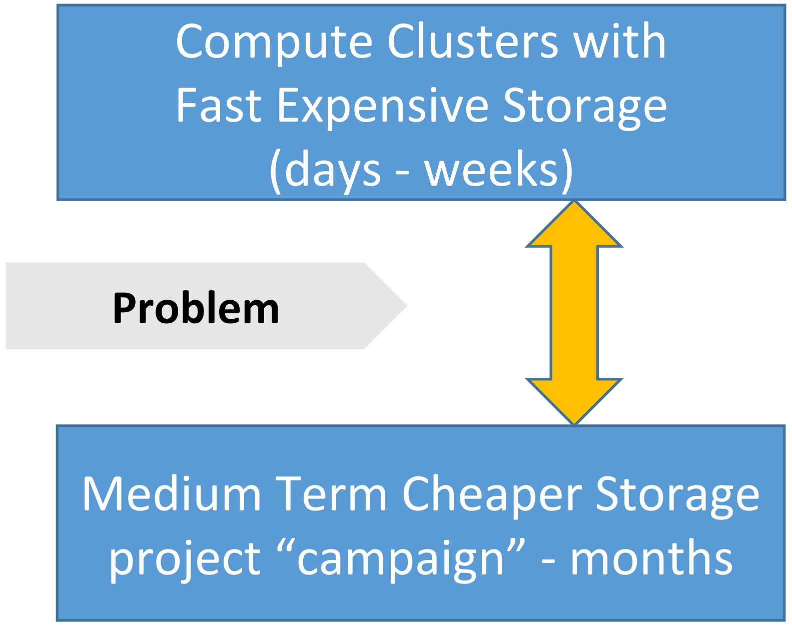
Principle: Data Staging

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

Implementation

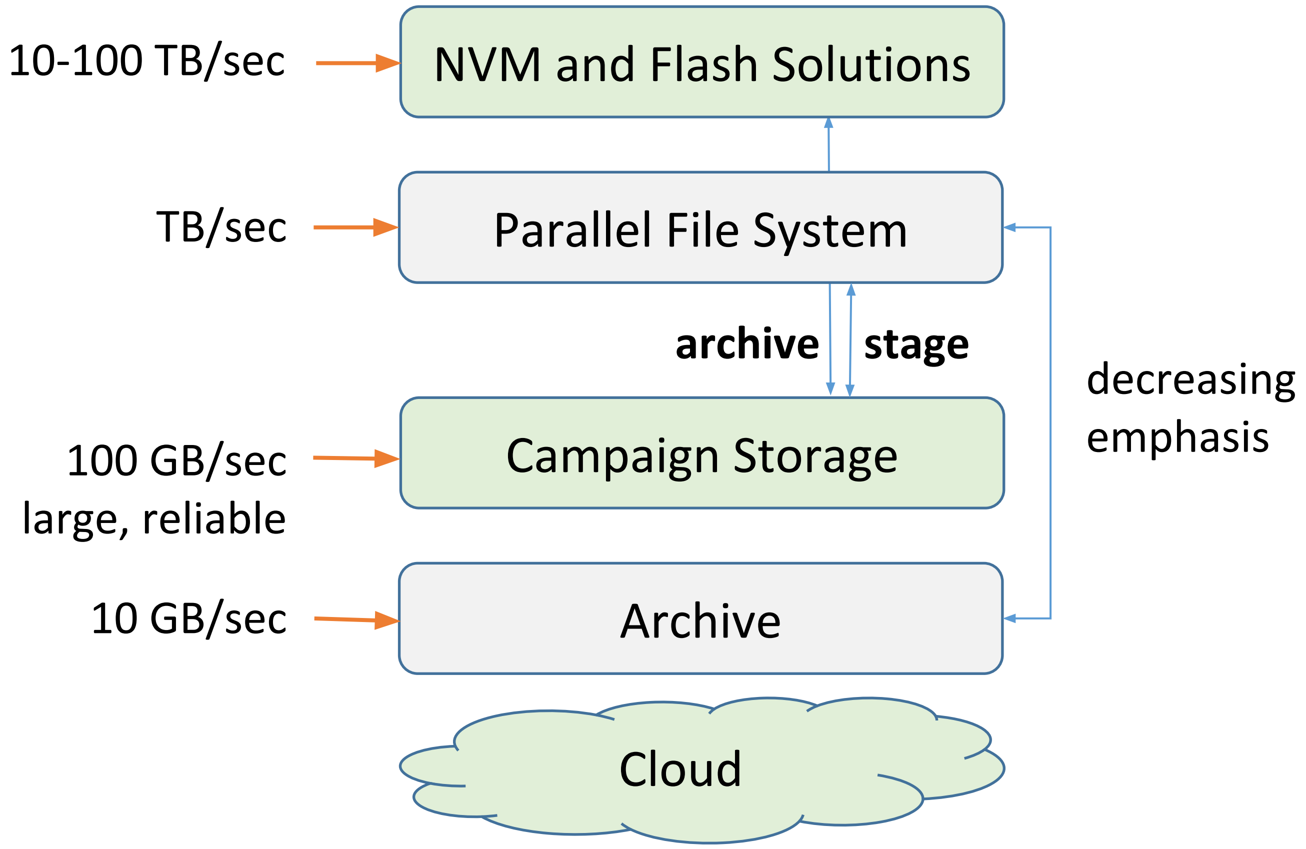
ADIOS with data spaces
Custom libraries – e.g. PUM library from LRZ

Whole file staging and archiving


Compute Clusters with Fast Expensive Storage (days - weeks)
Problem
Medium Term Cheaper Storage project “campaign” - months

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

Campaign Storage (MARFS from LANL)



Recap of software 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

HPC 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
Increasing use of object stores

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

10 Questions

- 1 - Role of AI / ML for storage

AI to apply data layout & aggregation automatically?

- So far limited success
- Should be doable. Fast adaptive learning vs learn from all apps?

Design a new storage level tree layout (Google 2017)

- Computer improved btree
- Get used to AI generated code - like weird chess moves

- 2 - cloud deployments

HPC in the cloud is growing fast in places like Amazon, Azure, …

A piece of a parallel distributed file system is requested

- synchronization of data
- global namespaces with reasonable performance

How to integrate this with the cloud native object stores?

- map file system devices to objects? map files to objects?

- 3 - Exa-scale storage 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 only solidify incumbents

- 4 - Application formats vs. File formats

Python NumPy array computations becoming ***incredibly*** popular

They dominate AI, much of University research computing

Is storing this as files, in memory format, a good idea?

No overheads and conversions

Would require an app and perhaps transactions

- 5 - Further software support of NVM

Persistent memory development kit offers transaction model

What **really** is a persistent memory computation? PL implications?

- Are they restartable?
- Are automatic program transformations to achieve persistence
- Tiers of memory with stackable API

- 6 - Much higher performance NVM file system

DAX is a good part of the story

How to do metadata OS bypass?

- 7 - Distribute storage - local node performance

Network access to storage devices has always been a compromise

    I don’t think that‘s likely to change

Build a coherent system that almost always leverages local storage

- 8 - Accelerators

Numerous memory systems must be targeted by an IO API

Linux heterogeneous memory subsystems are in progress

- 9 - Declarative storage organization

Contrast SQL schema definitions

and

Low level stuff like

- address spaces, stripe patterns, allocations, alignments
- building all data management apps from scratch

is the barrier to addressing really so high?

- 10 -

What is the future of archiving?

Conclusions

Conclusions

- More fundamental questions are being asked than 10 years ago
- Hardware developments have been and likely will remain fantastic

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

Thank you

peter@braam.io