Campaign Storage

storage for tiers
space for everything

Peter Braam

Co-founder & CEO

Campaign Storage, LLC

2017-05

campaignstorage.com

Contents

- Brief overview of the system
- Creating and updating policy databases
- Data management API’s

Thank you

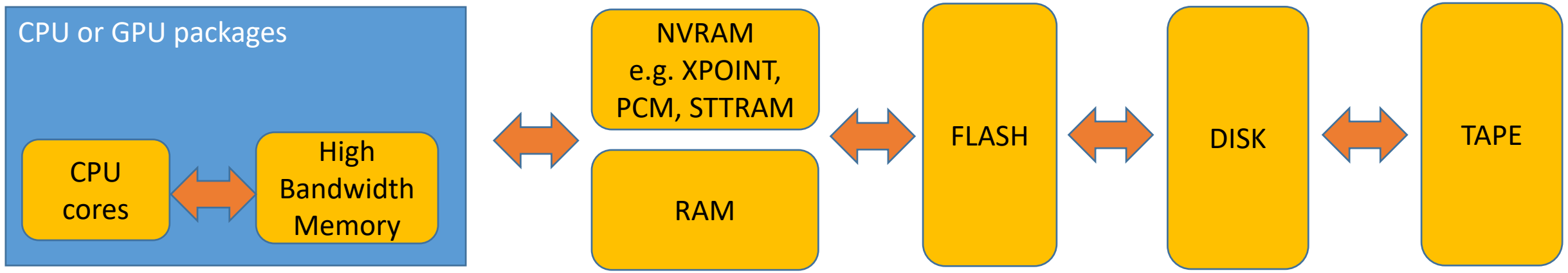
The reviewers of our paper asked quite a few insightful questions

Thank you.

Campaign Storage

Invented at LANL

Being productized at Campaign Storage

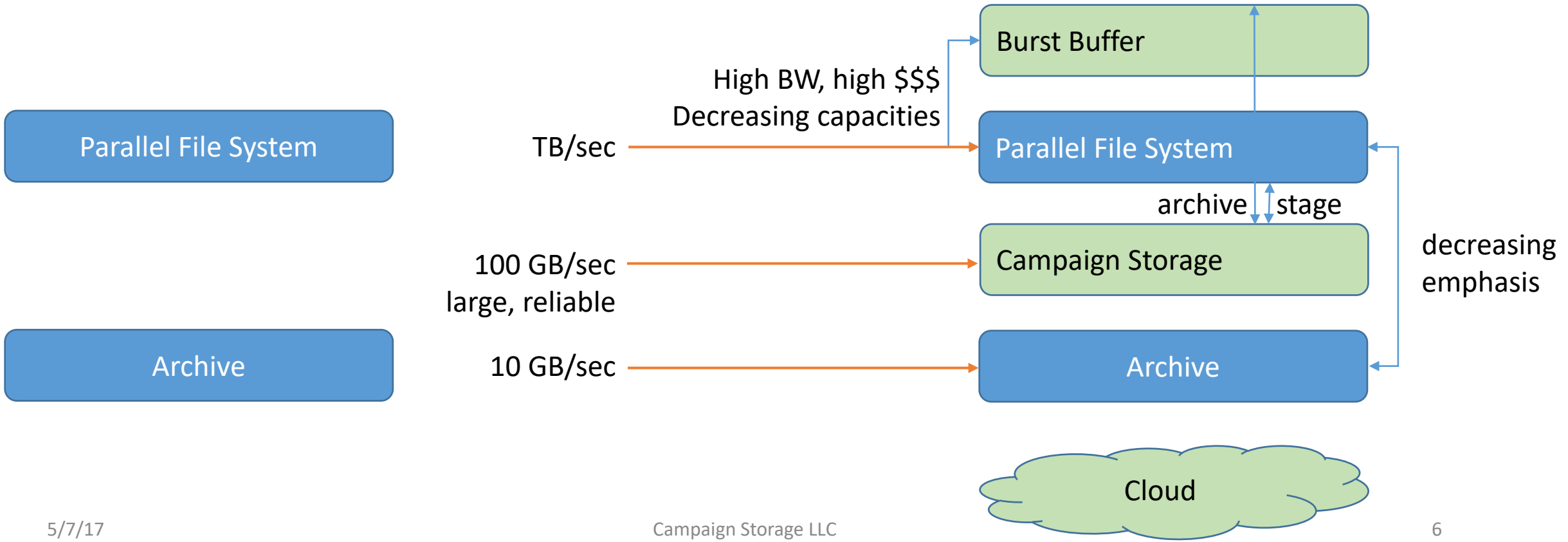


BW Cost $/ (GB/s)	**$10 (CPU included!)**	**$10**	**$200**	**$2K**	**$30K**
Capacity Cost $/GB	**$**	**$8**	**$0.3**	**$0.05**	**$0.01**
Node BW (GB/sec)	**1 TB/s**	**100 GB/s**	**20 GB/s**	**5 GB/s**	
Cluster BW (TB/sec)	**1 PB/s**	**100 TB/s**	**5 TB/s**	**100 GB/s**	**10's GB/s**
Software	**Language level**	**Language level HDF5 / DAOS**	**DDN IME Cray Data Warp**	**Parallel FS Campaign Storage**	**Archive Campaign**

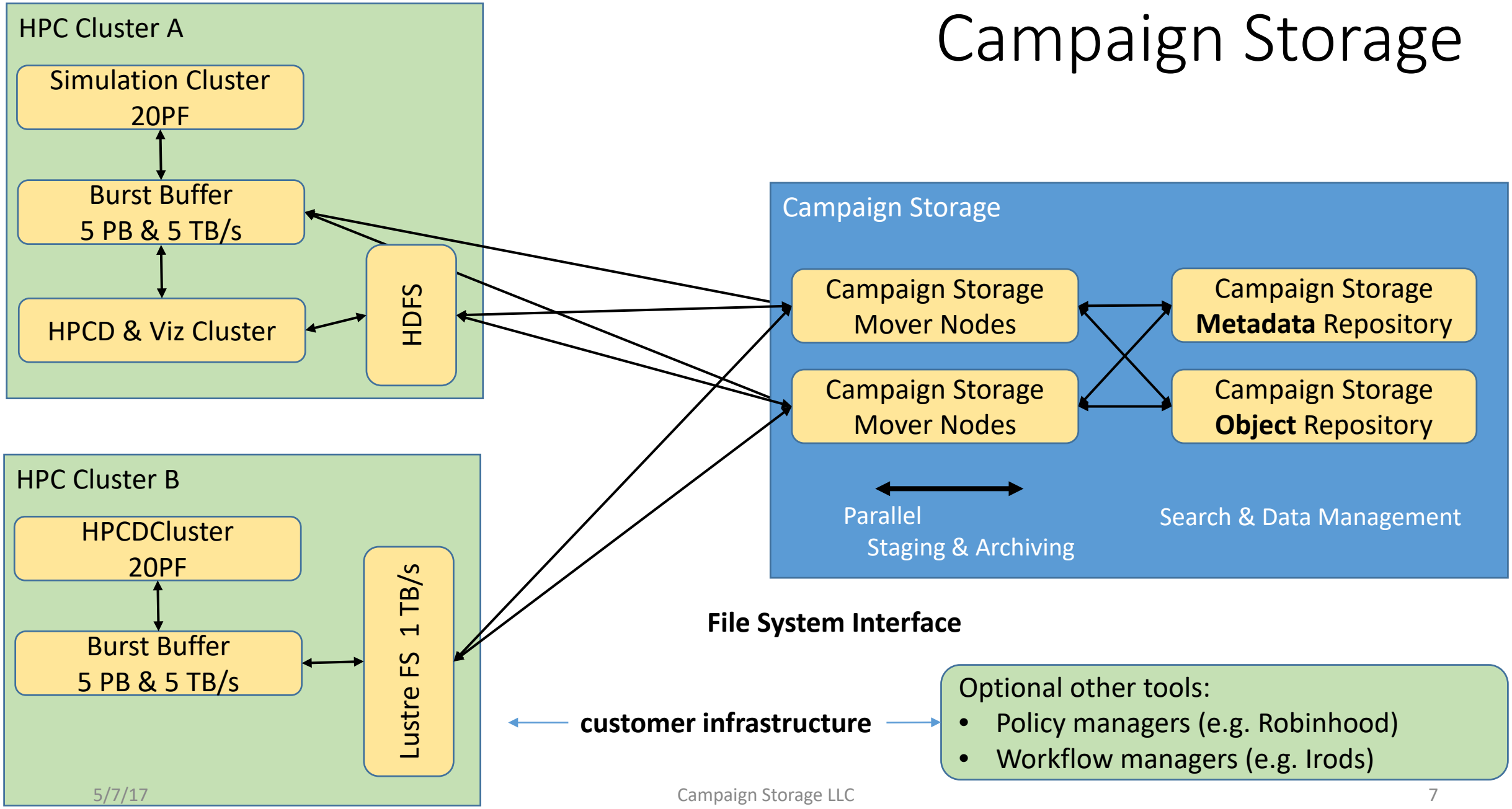
Campaign Storage - a new tier

Old World

New World



Campaign Storage



Campaign Storage

It is …

A file system - staging and archiving

Built from low cost HW but:

- **Industry standard object stores**
- **Existing metadata stores**

High integrity

High capacity, ultra scalable

Not highest BW or lowest latency

- 10-100x higher than archives
- 10x lower than PFS

It is not …

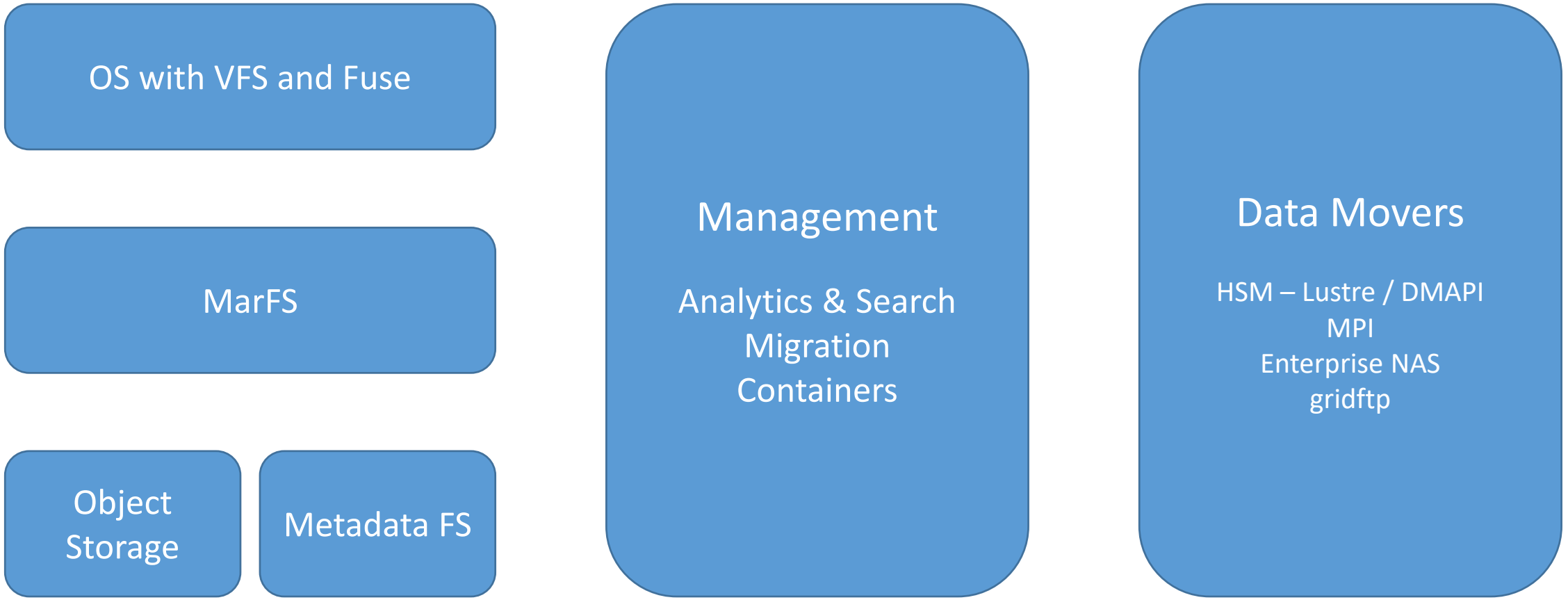
General purpose file system

- Wait … these don't exist actually

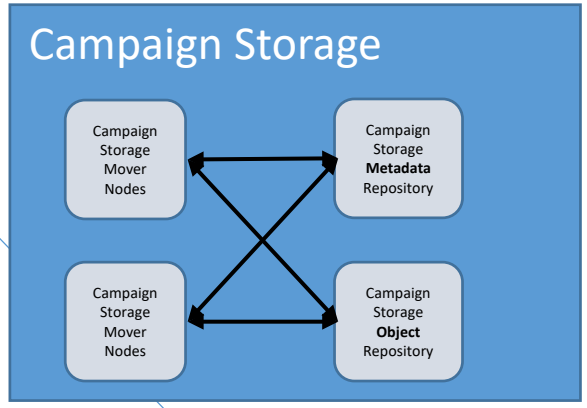
Using object stores has problems

- Data mover support takes effort
- We will ease that pain

Implementation - modules



Campaign Storage - deployment



deploy

Move & Manage

Nodes: 1-100's
- Mount MarFS & other FS

Mover software
- Software on mover node

Management
- Search analytics in MarFS
- 3rd party movement
- Containers

Object Repository

Disk object stores
- Commercial & OSS

Archival object stores
- Black Pearl

Full POSIX objects
- Stored in metadata FS

Metadata Repository

Some nearly POSIX distributed FS with EA's
- Lustre / ZFS
- GPFS

Policy Databases

Traditional approach

Database with a record for each file

Found in HPSS, Robinhood, DMF etc

Used for

Understanding what is in the file system

which files are old, recent, big, belong to group, on device

Assist in automatic (“policy”) or manual data management

Typically histogram ranges are computed from search results

Challenges

Challenges

- Performance – both ingest and queries
  - queries on 100M file database can take minutes
- Scalability
  - Requires significant RAM (e.g. 30% of DB size)
  - Handling more than 1B files is very difficult presently
- Never 100% in sync
- Adds load to premium storage

Approaches

Horizontally scaling key value store

- LANL is exploring this

A variety of proprietary approaches – e.g. Komprise

Histogram analytics

- Maintaining aggregate data has it own challenges:
  - e.g. How to measure the ***change in size*** of a file
- Very few changelogs record old size

Analytics - subtree search

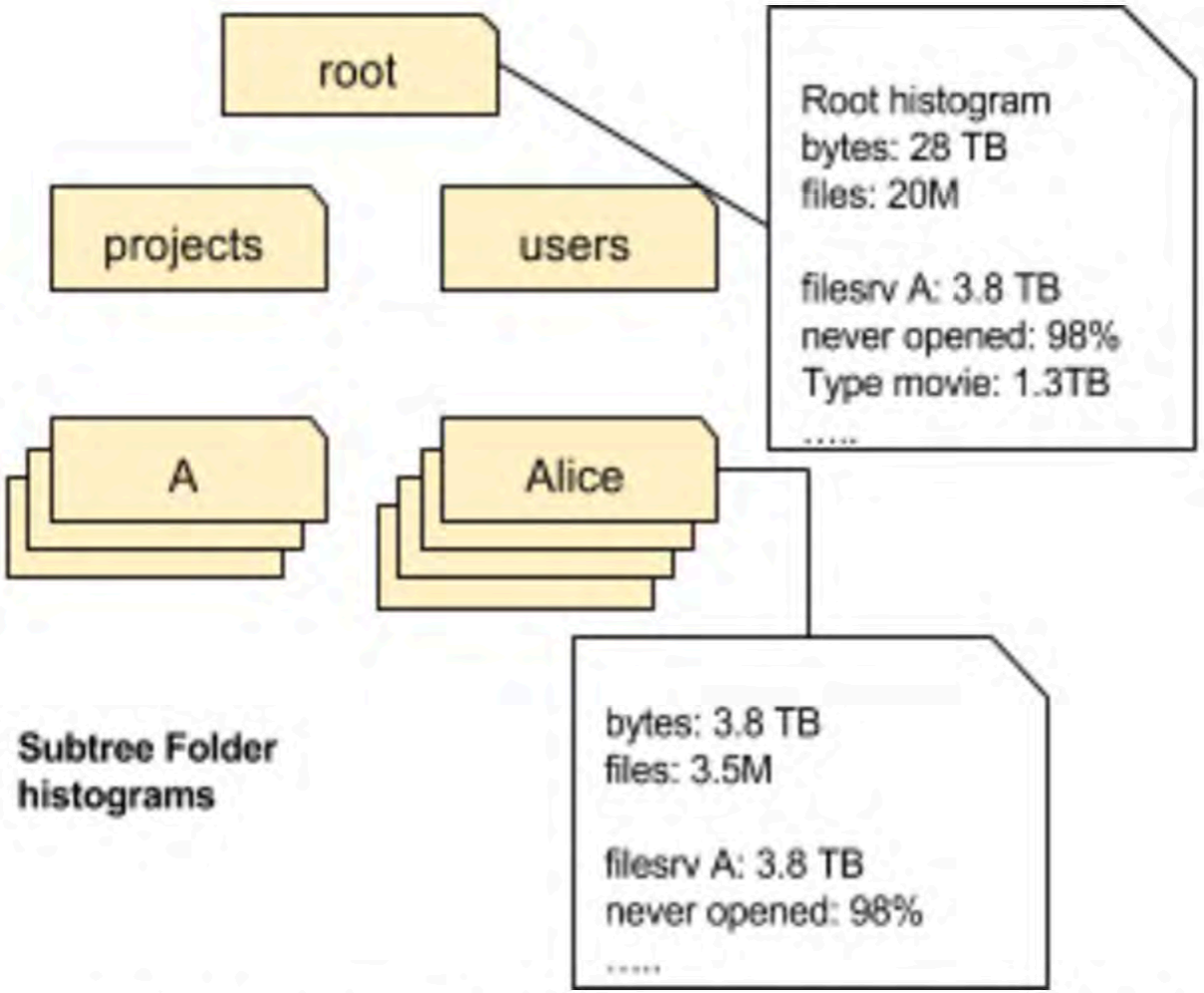
Every directory has **histogram** recording properties of its subtree

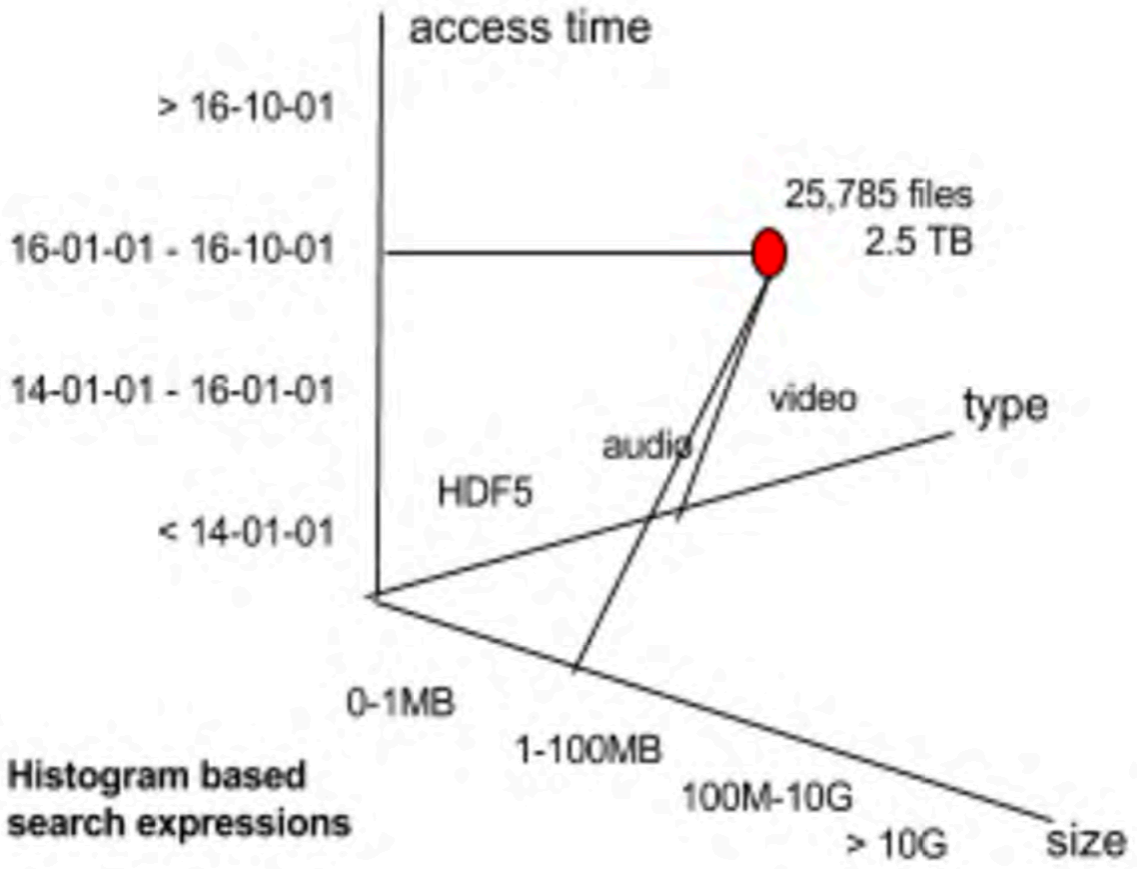
- encode: #files, #bytes in subtree have a property?
- Limited granularity, limited relational algebra
  - Store perhaps ~100,000 properties per directory

Examples:

- Quota in subtree? User/group database for subtree?
- What fileservers contain files?
- Geospatial information in file?
- (file type, size, access time) tuples
  - Allows limited relational algebra

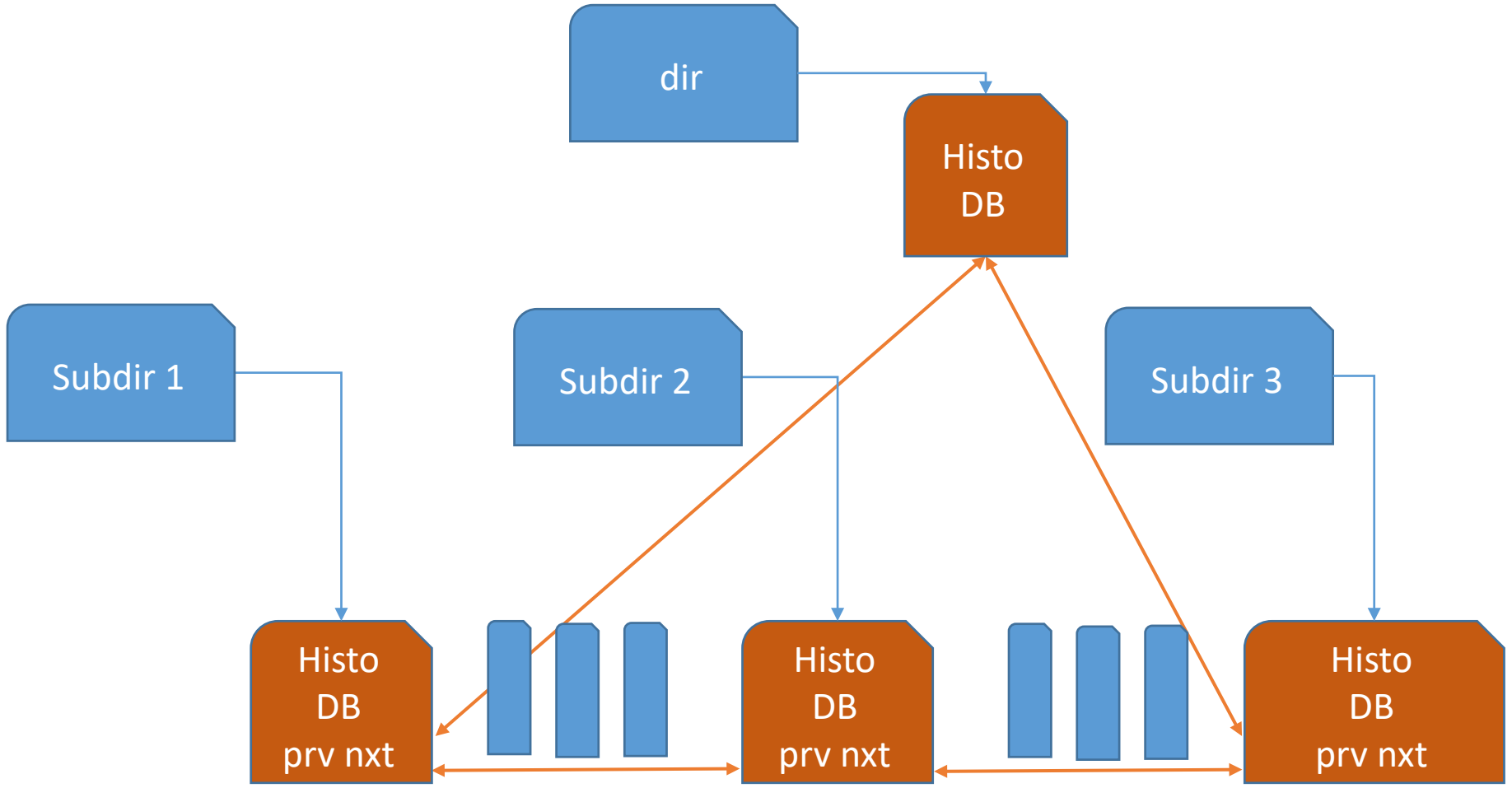
Not a new idea. Can be added to ZFS & Lustre





Include e.g. linked list of subdirectories and database of parents of files link count > 1

Iterate over subdirectories



Key properties

Generate initially from a scan, then update with changelogs

mathematically prove

histo(changelog ∘ FS1) = histo_update(changelog) + histo(FS1)

Additive property:

histograms can be added, either increase count or add new bars

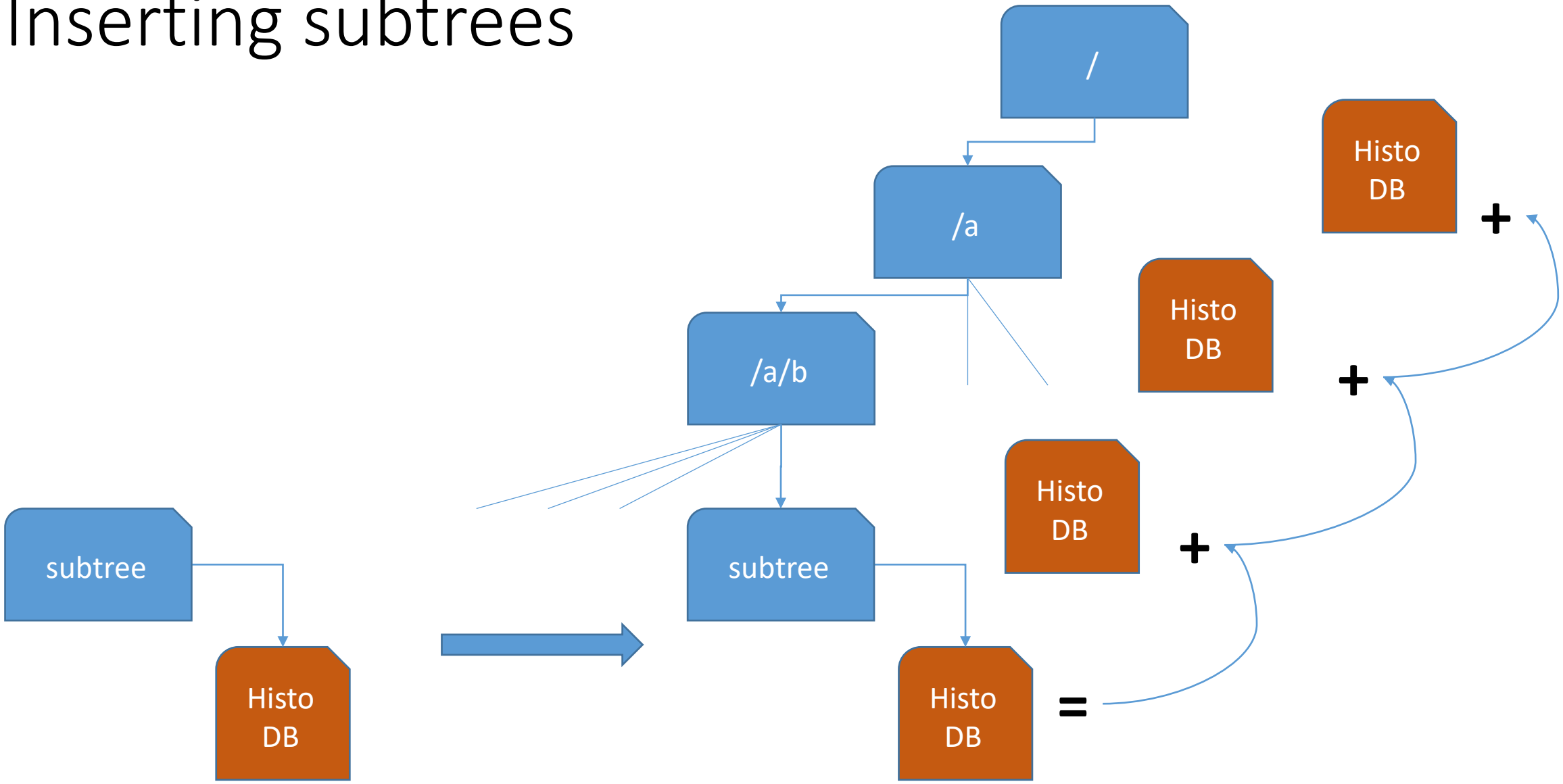
histo(dir) = sum histo(subdirs) + contributions(files in dir)

this is **Merkl tree** property – graft subtrees with simple addition

Keep 100% consistent with snapshots

Space consumption on par with policy database with 100K histogram buckets

Inserting subtrees



Evaluation

A single histogram lookup may provide the overview that a policy search provided

But

A histogram approach may has insufficient data for efficient general searches. Adapting histograms can be costly – how common is this?

Missing Storage API’s

Reflect on Storage Software

Since 1980's a utility has been added "afs" "bfs" "cfs" … "zfs"

- implements a set of non-standardized features
- file sets, data layout, ACL's
  - ACL's and extended attributes became part of POSIX in 2000's

Storage software almost always centers around batch data operations:

- caches do this inside the OS
- utilities do this – rsync, zip,
- cloud software does this – dropbox
- containers do this - Docker

Lack of standardized API's

Unnecessarily complicated software

Not portable, locked in to a platform

Example - data movement across many files

- Objective store batches of files
- New concept: file level I/O vectorization
  - Includes server driven ordering
  - Packing small files into one object
  - Cache flushes

```
int copy_file_range(copy_range *r, uint count, int flags)

struct copy_range {
        int source_fd;
        int dest_fd;
        off_t source_offset;
        off_t dest_offset;
        size_t length;
}
```

Extending the API - alternatives

In some areas concepts must be defined
- data layout
- sub-sets and subtrees of file systems (very similar to “mount”)

DB world solved this problem – SQL as a domain specific language
- A file level data management solution could build on:
  - asynchronous data and metadata API’s
  - batch / transaction boundaries
  - intelligent processing
- Possibly a better approach than more API calls
  - evidence is seen in SQFSCK

New problems will keep appearing, e.g. doing this in clusters

Thank you

Metadata Movement

Batch metadata handling

Well studied problem, not easily productized

Several sides to the problem

1. scale out the server side – data layout
2. bulk communication
   in many cases this utilizes replay of operations
3. tree requires linking subtrees and subsets

Conflicting demands between latency & throughput

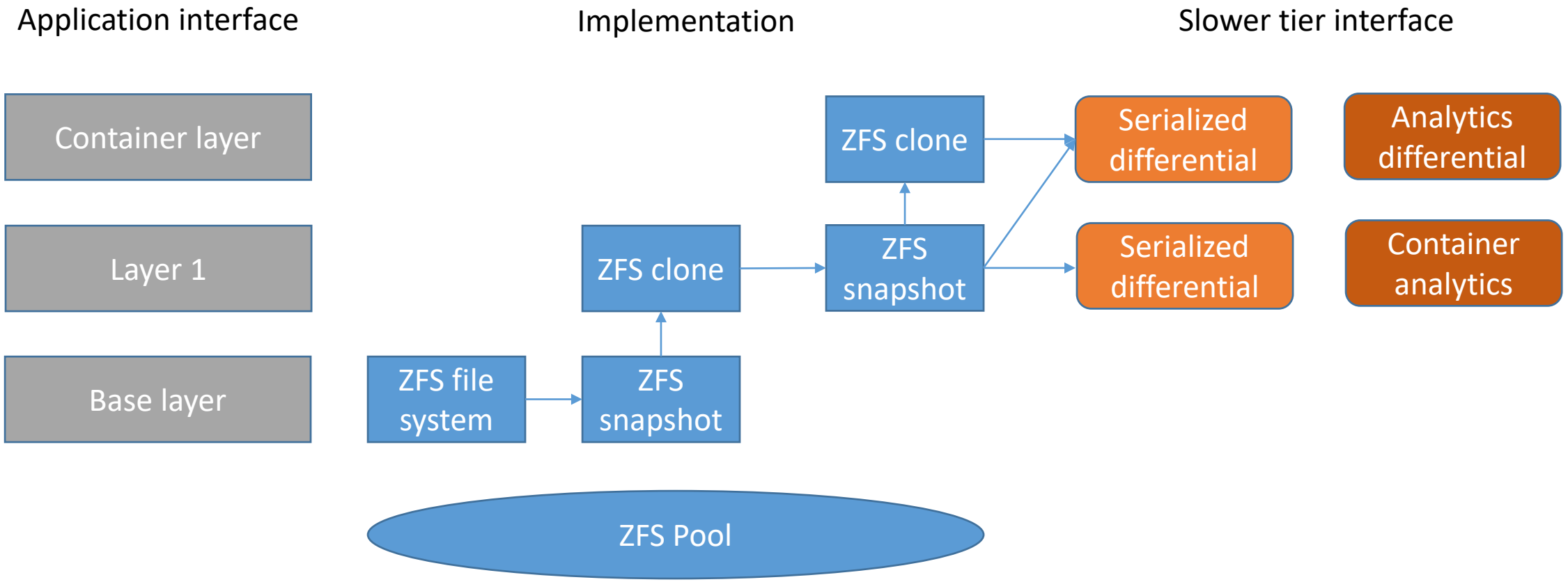
Role of containers

Fundamentally Unlikely

different tiers perform data movement at similar granularity

Containers are a must-have

Example Container Functionality



Containers as distributed namespace

Requires being able to locate the container

- Location database: a subtree resides on a node

Performance will scale well ***as long as*** containers can be large enough

- Fragmented vs. co-located metadata
- Local node performance x #nodes

Related to STT trees, not identical. CMU published a series of papers on this.

Other approaches / key unsolved problems

Other approaches:

- Peer to peer metadata protocols
- LANL scaled them to 1B file creates / sec (in an experiment)

- Allow conflicts

Distributed namespace consistency

- An “epoch” approach tracking dependent updates should work
- There is little understanding of fragmented vs contiguous MD

Conclusions

Campaign Storage: bulk data store, archive – focus on data movement

Massive data handling at file level is important

Amazon introduced S3FS, Dropbox and Gdrive rule

Search, batch metadata movement key ingredients

Richer API's or a DSL could create a better eco system

campaignstorage.com

Thank you