BERKELEY LAB

LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

Proactive Data Containers (PDC) for next generation HPC storage

Suren Byna

Computer Staff Scientist

Scientific Data Management Group

Lawrence Berkeley National Laboratory

PDC project

- A new DOE ASCR project to explore next generation storage systems and interfaces
- Project team
  - Quincey Koziol, Houjun Tang, Bin Dong (LBNL)
  - Jerome Soumagne, Kimmy Mu (The HDF Group)
  - Venkat Vishwanath, François Tessier (Argonne National Lab)

HPC data management requirements

Use case	Domain	Sim/EOD/analysis	Data size	I/O Requirements
FLASH	High-energy density physics	Simulation	~1PB	Data transformations, scalable I/O interfaces, correlation among simulation and experimental data
CMB / Planck	[illegible]	EOD/Analysis	[illegible]	...t optimizations
DECam & LSST	[illegible]	[illegible]	[illegible]	...s, data ...s
ACME	Climate	Simulation	~10PB	Async I/O, derived variables, automatic data movement
TECA	[illegible]	[illegible]	[illegible]	...ient data movement
HipMer	Genomics	EOD/Analysis	~100TB	Scalable I/O interfaces, efficient and automatic data movement

Easy interfaces and superior performance

Autonomous data management

Information capture and management

POSIX I/O: Main functionality

POSIX File System	Object Store
chmod	
open	
read	
lseek	
write	
close	
stat	
unlink	

Slide from Glenn Lockwood

POSIX I/O: Metadata

POSIX File System	Object Store
chmod	
open	
read	
lseek	
write	
close	
stat	
unlink	

All files have a common set of metadata
(permissions, owner, group, mtime, atime ctime)

Slide from Glenn Lockwood

POSIX I/O: Stateful

POSIX File System	Object Store
chmod	
open	
read	
lseek	
write	
close	
stat	
unlink	

I/O is stateful
(read/lseek/write operate on file descriptors from open/close)

Slide from Glenn Lockwood

POSIX I/O: Consistent

POSIX File System	
chmod	
open	
read	
lseek	
write	
close	
stat	
unlink	

Reads and writes are atomic and strongly consistent:

"After a write() to a regular file has successfully returned:

- Any successful read() from each byte position in the file that was modified by that write shall return the data specified by the write() for that position until such byte positions are again modified.
- Any subsequent successful write() to the same byte position in the file shall overwrite that file data."

Slide from Glenn Lockwood

What is an object store? Metadata-less

POSIX File System	Object Store
chmod	
open	
read	
lseek	
write	
close	
stat	
unlink	

No prescribed metadata

- object id (e.g., 0xbf5abcdb)
- object size (e.g., 5,368,709,120)

Slide from Glenn Lockwood

What is an object store? Stateless

POSIX File System	Object Store
~~chmod~~	
~~open~~	
read	No state
~~lseek~~	
write	
~~close~~	
~~stat~~	
unlink	

Slide from Glenn Lockwood

What is an object store? Simple

POSIX File System	Object Store
chmod	
open	
read	GET
lseek	
write	PUT
close	
stat	
unlink	DELETE

Slide from Glenn Lockwood

BERKELEY LAB

Examples of object storage systems

- Object storage services
  - Amazon S3, Rackspace Cloud files, HP Cloud object storage, IBM Cloud Object Storage, etc.
- Object-based storage systems
  - Lustre, etc.
  - Ceph
  - DAOS
  - MarFS
  - OpenStack Swift
  - …

What is an object?

- Chunks of a file
- Files (images, videos, etc.)
- Array
- Key-value pairs
- File + Metadata

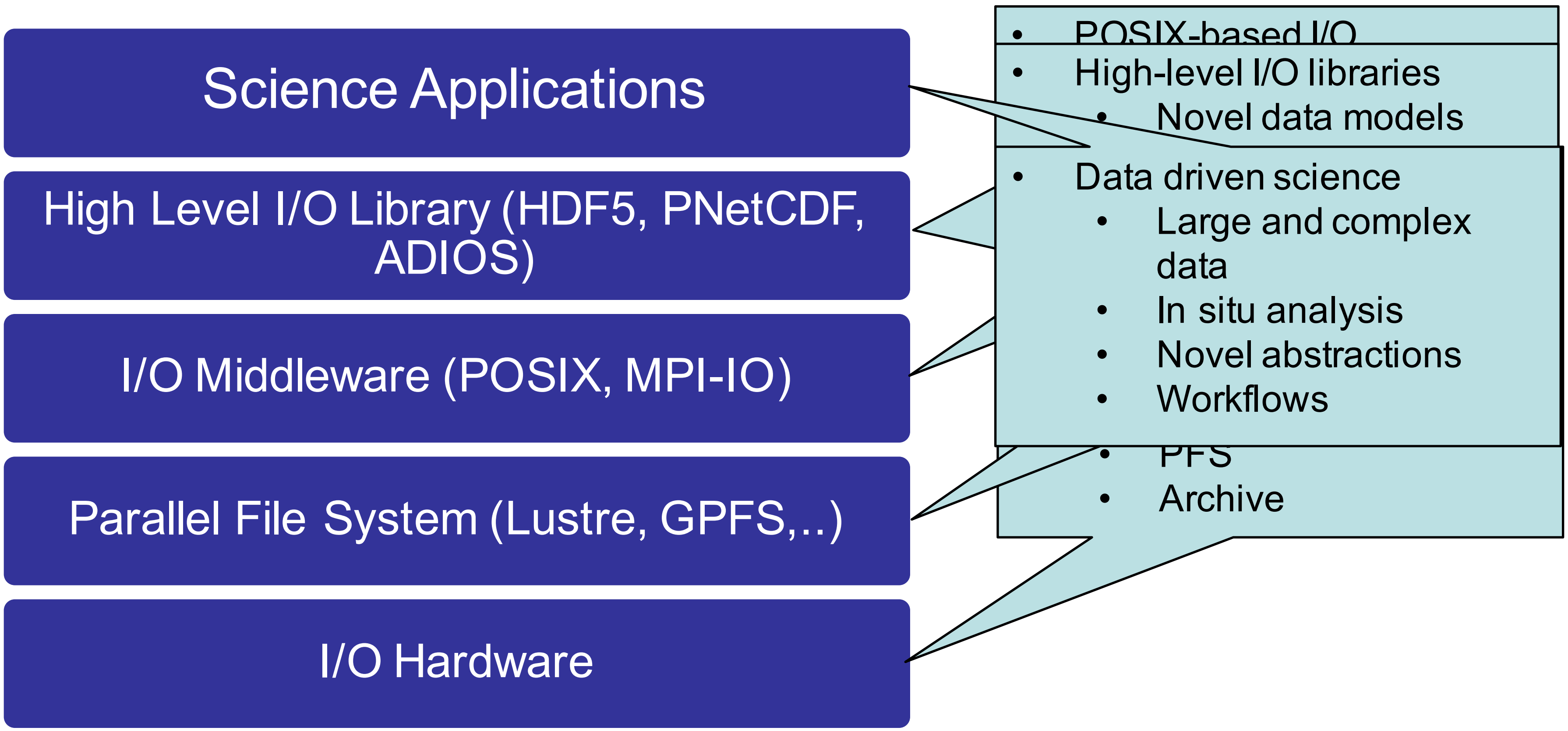
Current parallel file systems

Cloud services (S3, etc.)

HDF5, DAOS, etc.

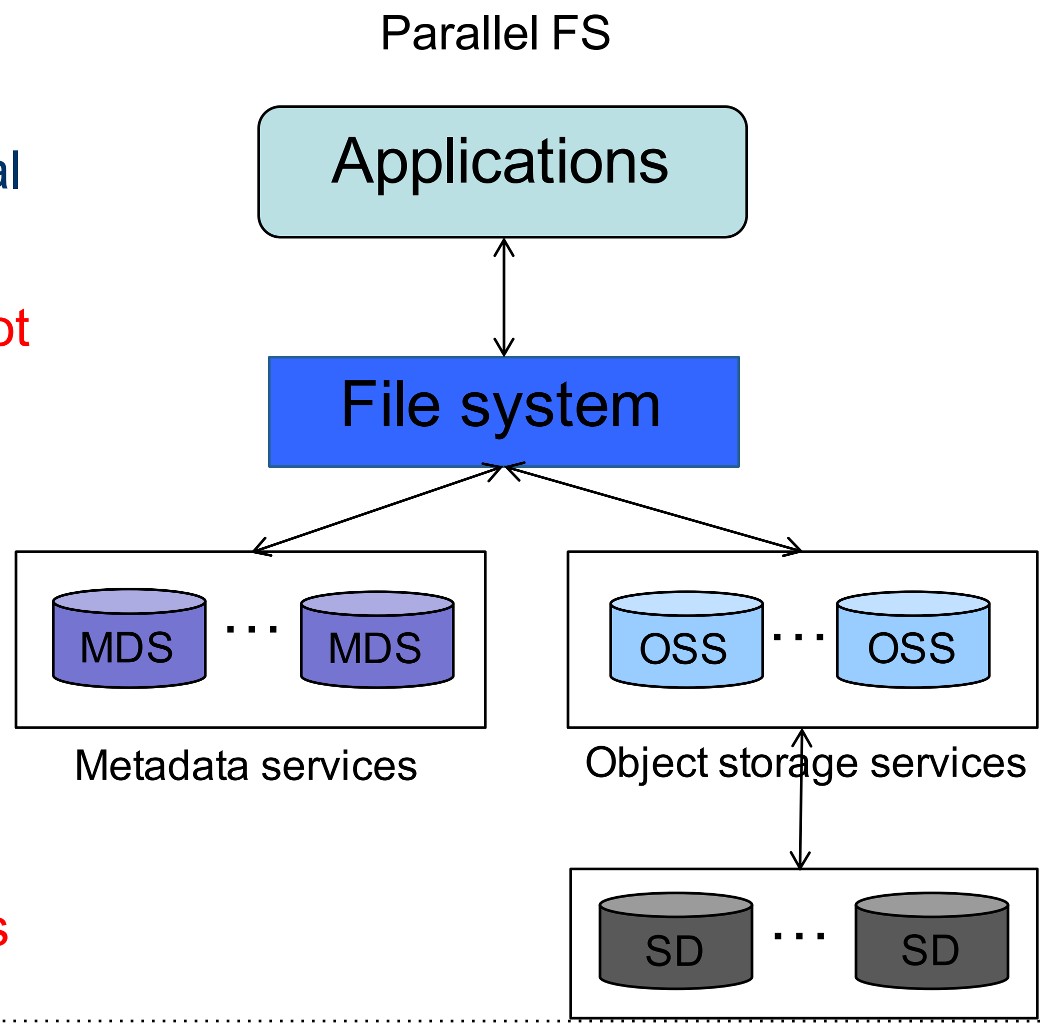
OpenStack Swift, MarFS, Ceph, etc.

Existing parallel I/O stack



Challenges and requirements

- Object-based parallel file systems – Evolutionary patched systems to sequential file systems
- Performance tuning should not be a scientist's job
- Managing millions of directories and files is not scalable
- Lack of information capture and management
- Integrated data management across multiple storage layers is absent



HPC data management requirements

Use case	Domain	Sim/EOD/analysis	Data size	I/O Requirements
FLASH	High-energy density physics	Simulation	~1PB	Data transformations, scalable I/O interfaces, correlation among simulation and experimental data
CMB / Planck	[illegible]	EOD/Analysis	[illegible]	[illegible] optimizations
DECam & LSST	[illegible]	[illegible]	[illegible]	[illegible], data [illegible]
ACME	Climate	Simulation	~10PB	Async I/O, derived variables, [illegible]
TECA	[illegible]	[illegible]	[illegible]	[illegible] data movement
HipMer	Genomics	EOD/Analaysis	~100TB	Scalable I/O interfaces, efficient and automatic data movement

Easy interfaces and superior performance

Autonomous data management

Information capture and management

BERKELEY LAB

Research toward object-centric storage systems

- Two objectives of "**object-centric storage**"
  - Performance
  - Productivity
- Autonomous data management
  - Users and application developers should be free from managing byte streams, files, and directory hierarchy
  - Automatic and efficient data movement and use of memory hierarchy in exascale systems
- Support for extracting information from data
  - Automatic extraction and management of information
  - Simulation time analytics
  - Interaction among multiple datasets (e.g., simulation and experimental/observation data)

PDC interpretation of an object

- Chunks of a file
- File
- Array
- Key-value pairs
- File + Metadata
- Data + Metadata + Provenance + Analysis operations + Information (data products)

Current parallel file systems

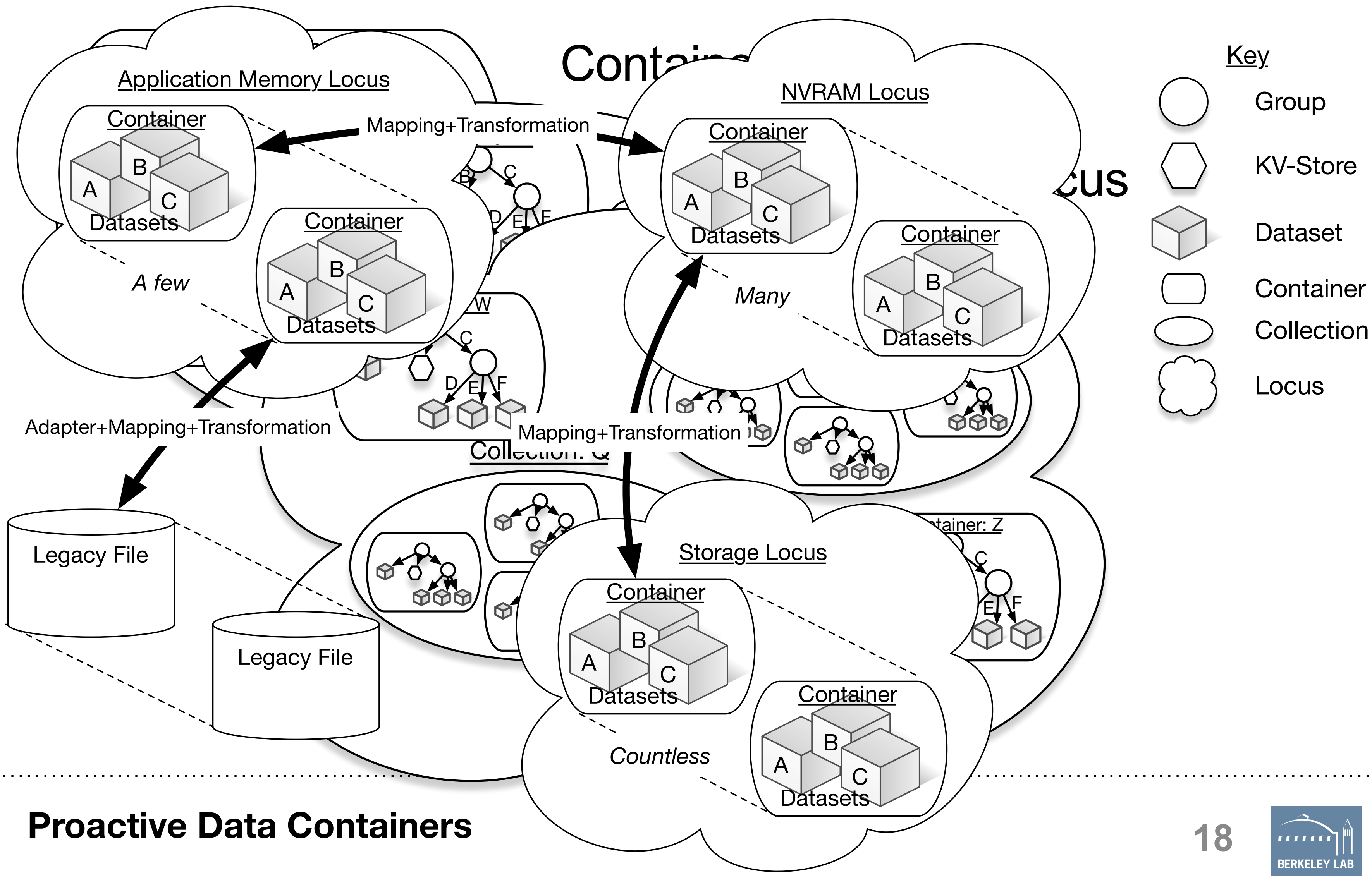
Cloud services (S3, etc.)

HDF5, DAOS, etc.

OpenStack Swift

Proactive Data Containers (PDC)

Proactive Data Containers



PDC System – High-level Architecture

Interface

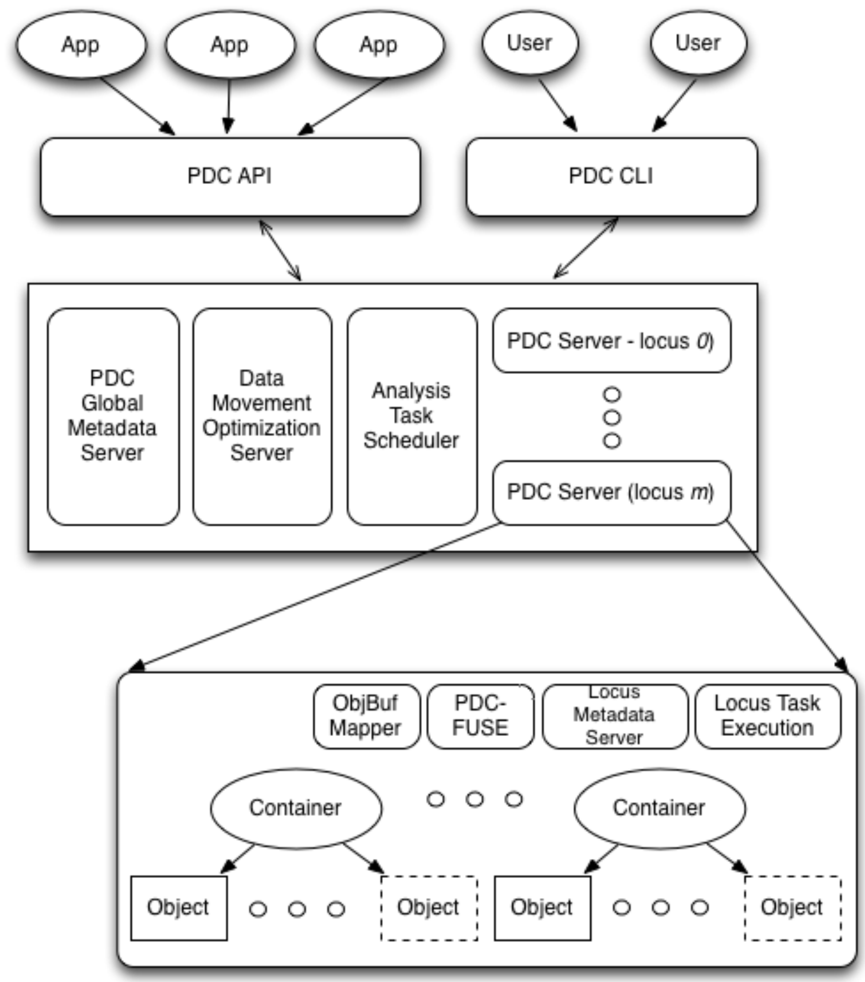
- Programming and client-level interfaces

Services

- Metadata management
- Autonomous data movement
- Analysis and transformation task scheduler

PDC locus services

- Object mapping
- Local metadata management
- Locus task execution



BERKELEY LAB

PDC System – High-level Architecture

Interface

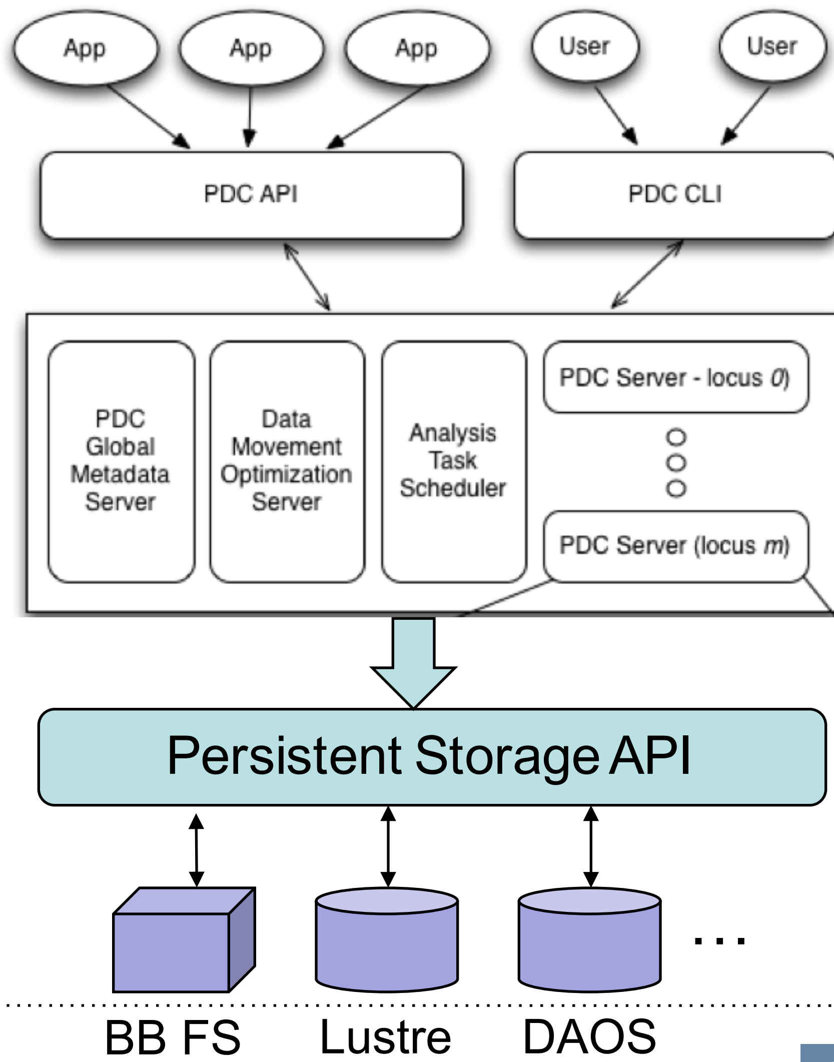
- Programming and client-level interfaces

Services

- Metadata management
- Autonomous data movement
- Analysis and transformation task scheduler

PDC locus services

- Object mapping
- Local metadata management
- Locus task execution



BERKELEY LAB

PDC API – Object manipulation

- Create an object
  - Pass object properties (metadata): name, lifetime, user info, provenance, tags, dimensions, data type, transformations, consistency, etc.
- Open object
  - Memory allocation, etc.
- Create a region
- Map and unmap a region to object
- Close object
- Release object

Metadata Object		
Pre-defined Tag	User-defined Tag	Operations

• Objcet ID
• DataObjLocation
• SystemInfo
• ID Attributes
  • Name
  • AppName
  • Owership
  • TimeStep
	• (UserTag1, Value1)
	• (UserTag2, Value2)
	• (UserTag3, Value3)
	• …
		• Create
		• Delete
		• Search
		• Update

BERKELEY LAB

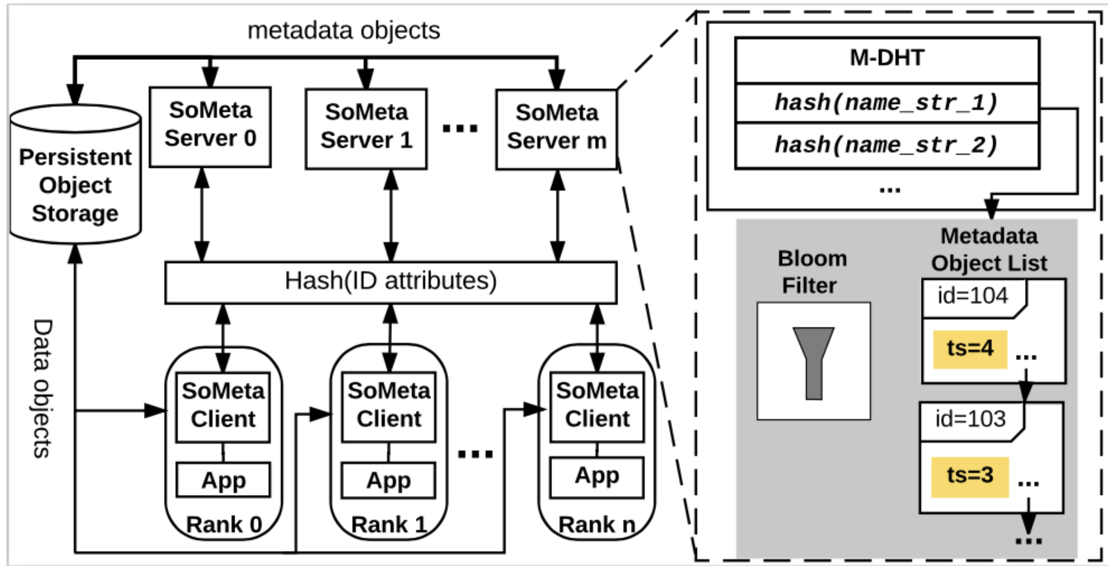
PDC API – Object Access

- Create query with conditions
- Execute query
- Iterate_start
- Iterate_next
- Get_object_handle
- Get_object_info

SoMeta: Metadata management design

Metadata server design

- Distributed metadata servers
- Uses a core on each compute node
- Distributed hash table to select metadata servers



Capabilities

- Create, update, and delete metadata objects
- Metadata objects are searchable
- Attach tags for extended attributes and relationships

Metadata Object		
Pre-defined Tag	User-defined Tag	Operations

• Objcet ID
• DataObjLocation
• SystemInfo
• ID Attributes
  • Name
  • AppName
  • Owership
  • TimeStep
	• (UserTag1, Value1)
	• (UserTag2, Value2)
	• (UserTag3, Value3)
	• …
		• Create
		• Delete
		• Search
		• Update

BERKELEY LAB

Scalable metadata management for object-centric storage

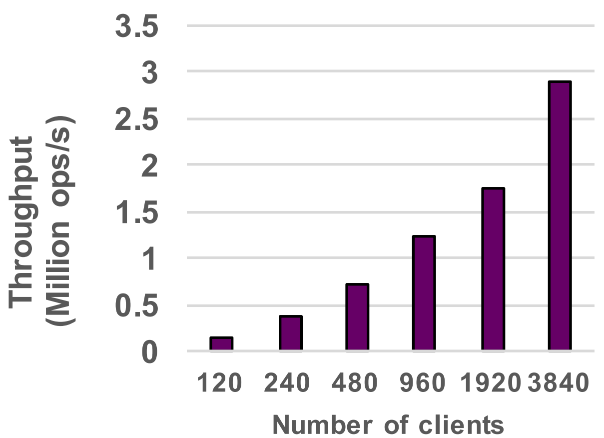
Problem

- Existing metadata management on file systems do not to grow or to shrink based on the need client load
- The load of creating objects by hundreds of thousands to millions of processes cannot be handled by existing file systems

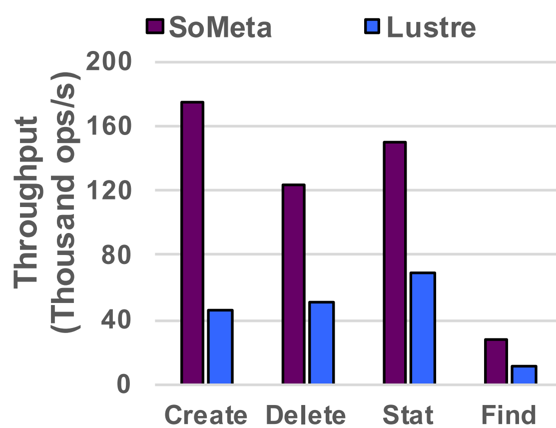
Approach

- Developed a scalable, fault tolerant, user-level, distributed metadata management system for object-centric storage systems
- Our system, SoMeta, provides tagging for storing rich semantic information as well as capabilities to search and retrieve interesting metadata objects, based on keyword search.

Results



Throughput for SoMeta to create objects with unique names.



A comparison of SoMeta and Lustre, where both systems use four metadata servers.

Impact

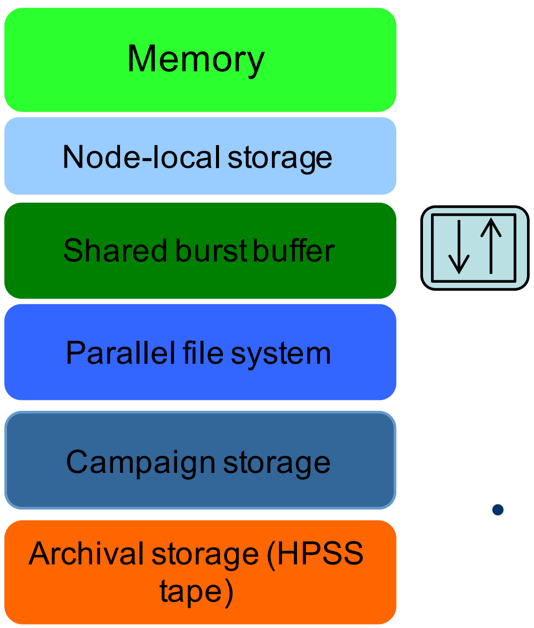
- Existing and future object-centric storage systems are able to use SoMeta for scalable metadata management.
- SoMeta has up to **3.7X speedup** over Lustre in common metadata operations with 4 servers, and scales well with more server processes.
- SoMeta scales well with the number of metadata servers; w/ 120 servers **> 3 million object creation operations per second**

Challenges of deep storage hierarchy

- *Inefficiency* :
  - Staging in/out data to/from burst buffers (BB) compete for resources on BB servers
- *Burden on users*
  - Users or applications have to explicitly make the data movement decisions, which could lead to inefficiency
- *Limited to one level*
  - Staging in or out and transparent caching is aware of a single level storage

BERKELEY LAB

Data Elevator for moving data transparently



- **Contributions**
  - Low-contention data movement library for hierarchical storage systems
  - Offload of data movement task to a few compute nodes or cores
  - Data Elevator on NERSC's Cori Phase I
    - With two science applications, we demonstrated that Data Elevator is **1.2X to 4X** faster than Cray DataWarp ***stage_out*** and up to **4X** faster than writing data to parallel file system
- **Benefits of using Data Elevator**
  - **Transparent data movement**: Applications using **HDF5** specify destination of data file and the Data Elevator transparently moves data from a source to the destination
  - **Efficiency**: Data Elevator reduces contention on BB
  - **In transit analysis**: While data is in a faster storage layer, analysis can be done in the data path

Data Elevator – high-level operation

Start Data Elevator along with an application

DE intercepts HDF5 calls at run time using VOL

…

H5Fopen (..) → Sets up staging area, and give file handle to app
Records the destination in a Berkeley DB table

…

H5Dwrite (..) → Writes the data to staging area
Updates the status

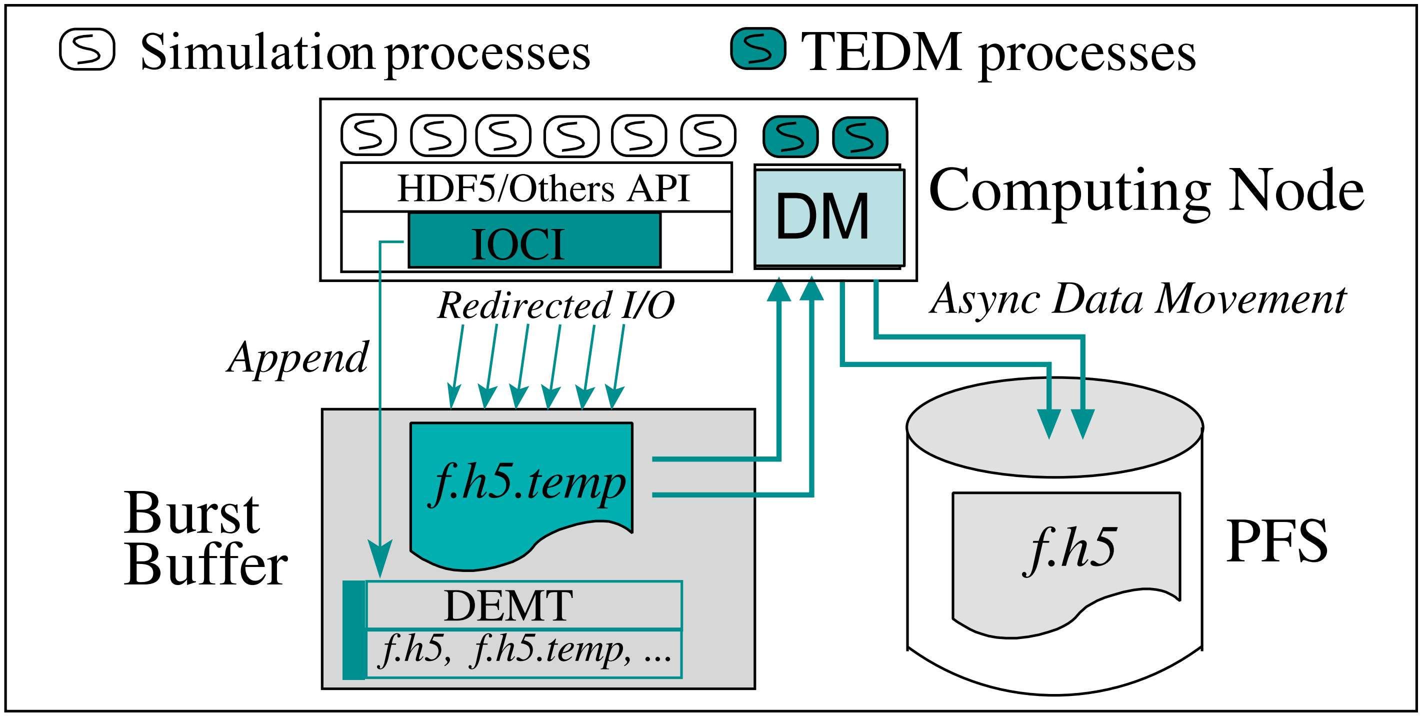
…

H5Fclose (..) → Staging area ready for use
(analysis, transfer, etc.)
Application can continue with computation

…

MPI_Finalize () → Check staging area status
Move any remaining data

Data Elevator design

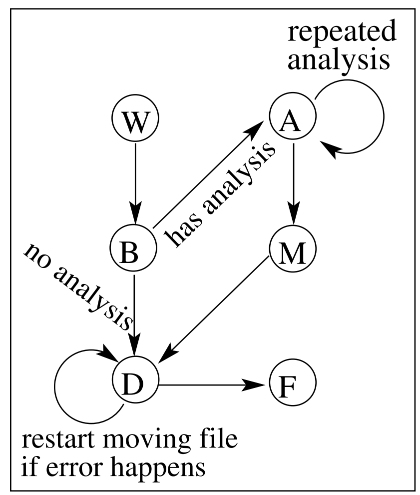


- **Implementation challenges**
  - Transparently intercepting I/O calls
  - Moving data between storage layers efficiently w/ low contention
- **Solutions**
  - IOCI – IO Call Interceptor library - VOL
  - Transparent & Efficient Data Mover processes – Concurrent MPI job

Metadata for managing the state of data

- Metadata Table to manage the data movement status
  - Data written to BB
  - Data is written to BB
  - Request to analyze data and start analysis
  - All data reads are done
  - Data is being written to PFS
  - Data is moved to PFS

Status	Description
W	Start writing to BB
B	Finish writing to BB
A	Start analysis
M	Finish analysis
D	Start moving to PFS
F	Finish moving to PFS



BERKELEY LAB

HPC data management requirements

Easy interfaces and superior performance

Autonomous data management

Information capture and management