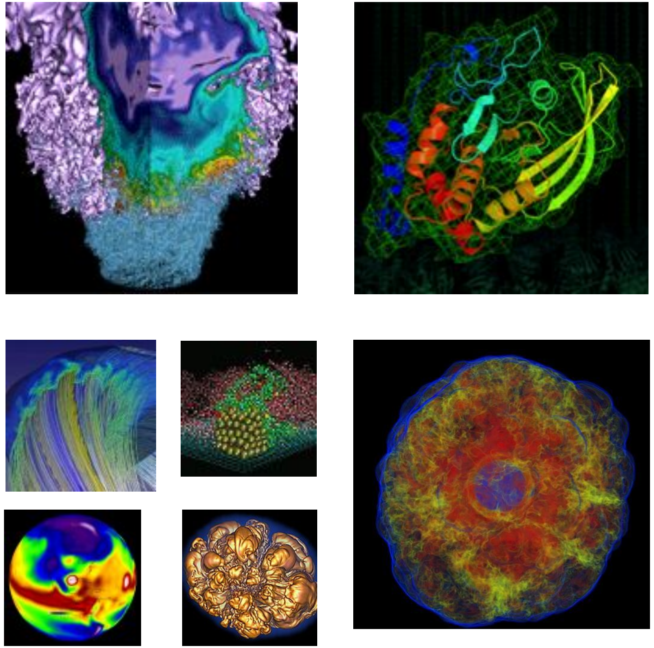
Evaluation of HPC Application I/O on Object Storage Systems



Jialin Liu, Quincey Koziol
Gregory F. Butler
Neil Fortner, Mohamad Chaarawi
Houjun Tang, Suren Byna
Glenn K. Lockwood
Ravi Cheema
Kristy A. Kallback-Rose
Damian Hazen, Prabhat

Nov 12nd, 2018
PDSW-DISCS

U.S. DEPARTMENT OF ENERGY | Office of Science

About the Team

NERSC@LBL: SSG, DAS, ATG

Jialin Liu, Quincey Koziol, Gregory F. Butler

Glenn K. Lockwood, Ravi Cheema

Kristy A. Kallback-Rose

Damian Hazen, Prabhat

CRD@LBL: SDM

Houjun Tang, Suren Byna

The HDF Group

Neil Fortner

Intel

Mohamad Chaarawi

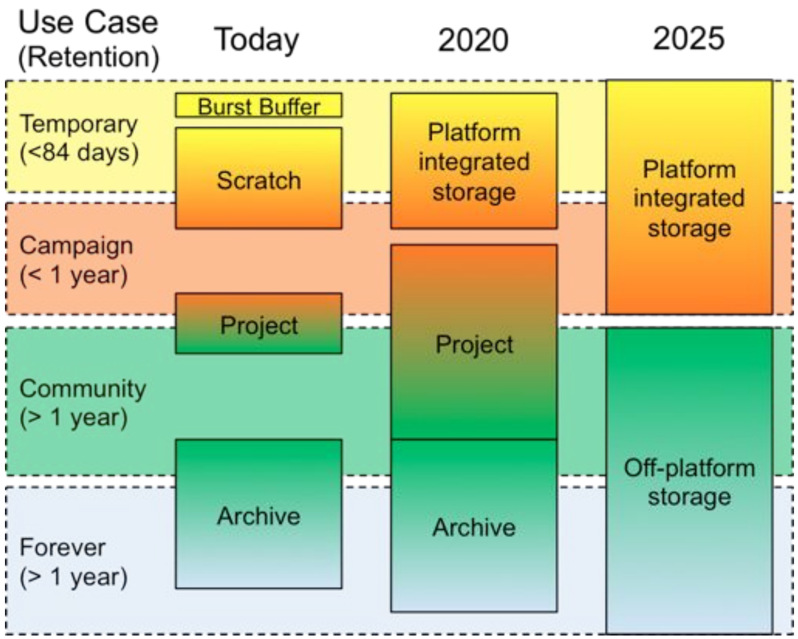
Trends in High Performance Storage

Hardware

- **Now:** SSD for on platform storage
- **Soon:** Storage Class Memory, byte addressable, fast and persistent
- **Soon:** NVMe over Fabrics for block access over high speed networks

Parallel file systems

- **Now**: POSIX-based file system
  - Lustre, GPFS
- **Potential replacement**:
  - Object stores (DAOS, RADOS, Swift, etc.)



POSIX and Object Store

"POSIX Must Die":
- Strong consistency requirement
- Performance/Scalability issue
- Metadata bottleneck

Jeffrey B. Layton, 2010, Linux Magazine

POSIX Still Alive:
- Without POSIX writing applications would be much more difficult.
- Extremely large cruise ship that people love to travel upon

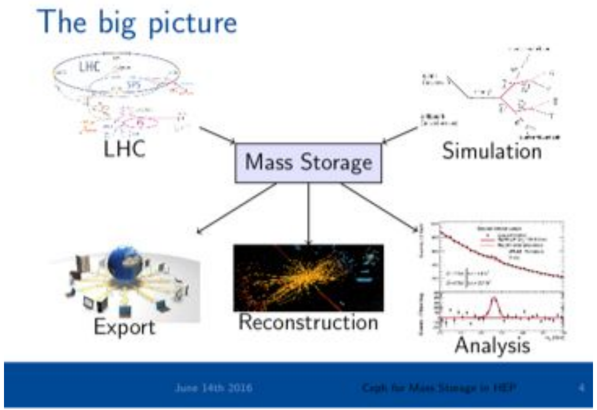
Benefits of Object Store:
- Scalability: no lock
- Disk-friendly I/O: massive read/write
- Durability
- Manageability
- System Cost

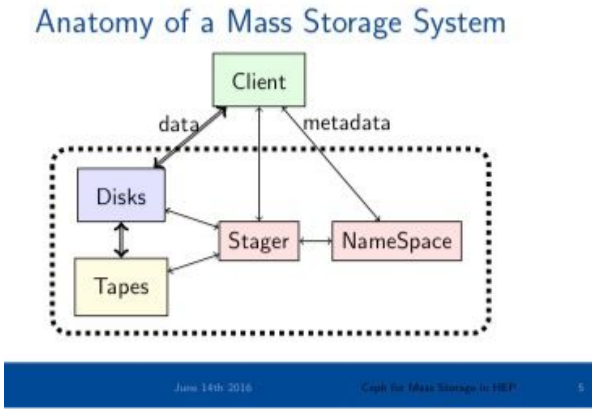
Glenn K. Lockwood, 2017

However:
- Immutable objects: no update-in-place
  - Fine-grained I/O doesn't work
- Parity/replication is slow/expensive
- Rely on auxiliary service for indexing
- Cost in developer time

"POSIX Must Die", Jeffrey B. Layton, 2010, http://www.linux-mag.com/id/7711/comment-page-14/
"What's So Bad About POSIX", Glenn K. Lockwood, NextPlatform: https://www.nextplatform.com/2017/09/11/whats-bad-posix-io/

Object Store Early Adopter: CERN





- ❖ Mainly used for archiving big files
  - ➢ 150PB tape as backend, 10PB disk as cache
  - ➢ 10s of GB/s throughput, single stream to tape: 400MB/s
- ❖ Why Ceph:
  - ➢ delegate disk management to external software
  - ➢ rebalancing, striping, erasure coding

Applications Can't Use Object Store Directly

NERSC

- Problem:
  - Apps are written with today's POSIX APIs: HDF5, MPI-IO, write/read
  - Object Stores only supports non-POSIX: put / get

Dream World

Reality

Motivation

- Evaluate object store systems, with science applications
  – Explore parallel I/O with object store API
  – Understand the object I/O internals
- Understand impact of object store on HPC applications and users
  – How much do HPC applications need to change in order to use object stores?
  – What is the implication to users?

- HPC Users
- HPC Applications

- POSIX Interface
- POSIX File System

Step 1: Which Object Store Technologies?

openstack SWIFT

SCALITY

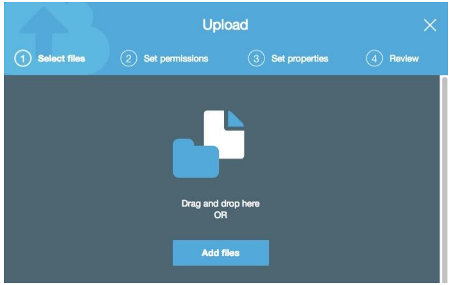
DAOS

MarFS @LANL

Google Storage

ceph

Amazon S3



Mero @Seagate

?

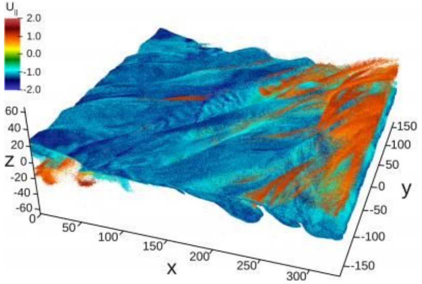
Requirements:

- Open Source
- Community Support
- Non-POSIX
- Applicable to HPC

Step 2: Which HPC Applications?

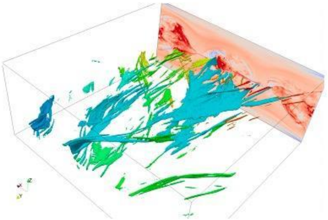
Requirements:

- Scientific Applications
- Representative I/O Pattern



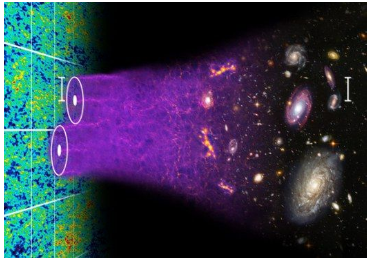
FastQuery identifies 57 million particles with energy < 1.5
Credit: Oliver Rübel et al.

- **VPIC**: Large Contiguous Write



Cluster Identified in Plasma Physics
Credit: Md. Mostofa Ali Patwary et al.

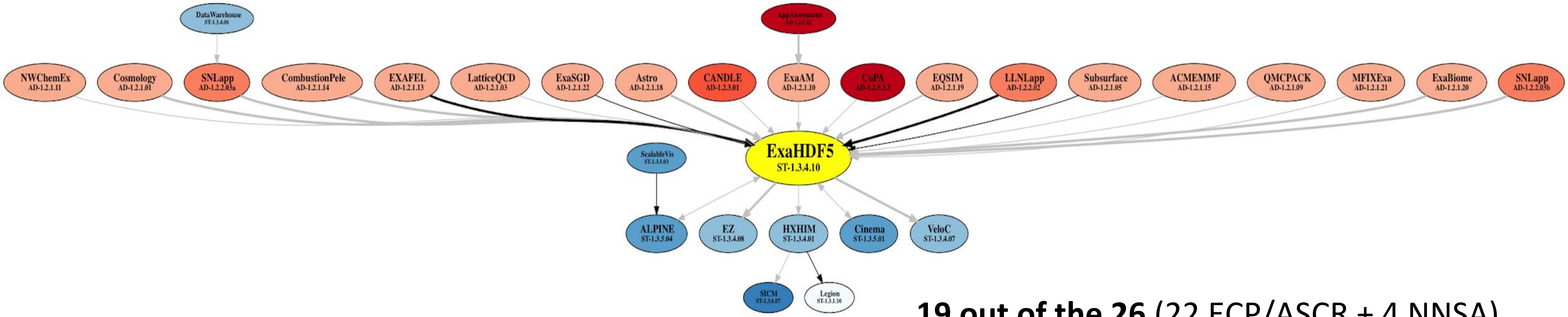
- **BD-CATS**: Large Contiguous Read



Concept of Baryon Acoustic Oscillations, with BOSS survey
Credit: Chris Blake et al.

- **H5BOSS**: Many Random Small I/O

HDF5: Scientific I/O Library and Data Format



19 out of the 26 (22 ECP/ASCR + 4 NNSA) apps currently use or planning to use HDF5 (*Credit: Suren Byna*)

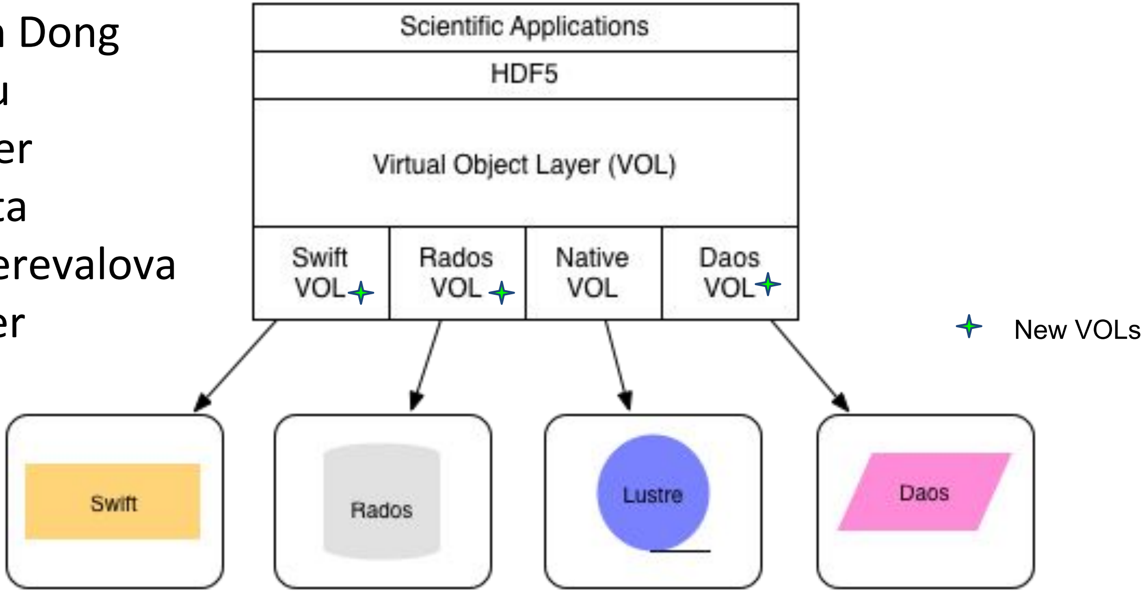
HDF5:

- Hierarchical Data Format v5
- 1987, NCSA&UIUC
- Top 5 libraries at NERSC, 2015
- Parallel I/O

U.S. DEPARTMENT OF ENERGY | Office of Science

HDF5 Virtual Object Layer (VOL)

- A layer that allows developers to intercept all storage-related HDF5 API calls and direct them to a storage system
- Example VOL Connectors:
  - Data Elevator, Bin Dong
  - ADIOS, Junmin Gu
  - Rados, Neil Fortner
  - PLFS, Kshitij Mehta
  - Database, Olga Perevalova
  - DAOS, Neil Fortner
  - ...



Example VOL: Swift

```
int main ()
{
MPI_Init();
…
H5Fcreate();
for (i=0;i<n;i++)
      buffer[i]=i;
H5Dcreate();
H5Dwrite();
H5Fclose();
...
MPI_Finalize();
}
```

HDF5 C Application

```
herr_t H5Dwrite()
{
.
.
.
.
H5VL_dataset_write()
.
.
.
}
```

HDF5 Source Code

```
const H5VL_class_t
{
H5VL_python_dataset_create,
H5VL_python_dataset_open,
H5VL_python_dataset_read,
static herr_t
H5VL_python_dataset_write() {
}
}
```

Generic Python VOL Connector

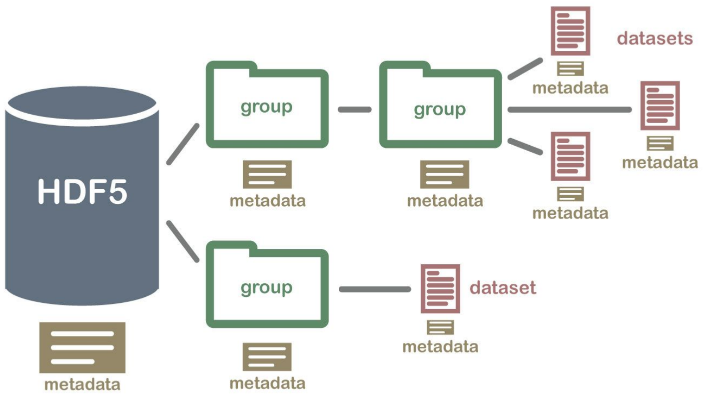
```
static herr_t
H5VL_python_dataset_write() {
    PyObject_CallMethod("Put");
}
```

"Callback function"

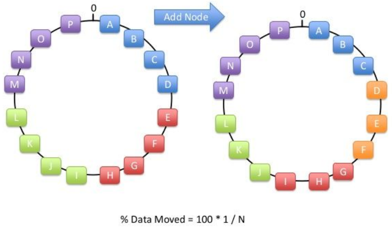
```
import numpy
import swiftclient.service

swift.upload()
```

Python Swift Client

Mapping Data to Object



Consistent Hashing



DAOS:
- HDF5 File -> DAOS Container
- Group -> DAOS Object
- Dataset -> DAOS Object
- Metadata -> DAOS Object

DAOS Object:
- Key: Metadata
- Value: Raw data

RADOS:
- HDF5 File -> RADOS Pool
- Group -> RADOS Object
- Dataset -> RADOS Object

RADOS Object:
- Linear Byte Array: Metadata
- Key: Name
- Value: Raw data

Swift:
- HDF5 File -> Swift Container
- Group -> Swift Sub-Container: 'Group'
- Dataset -> Swift Object
- Metadata -> Extended Attribute

Swift Object:
- Key: Path Name
- Value: Raw data

Parallel Object I/O

Data Read/Write

- Independent I/O
- *Collective I/O is possible in the future*

Metadata Operations

- Native HDF5: Collective or Independent I/O w/MPI to POSIX
- VOLs: Independent - highly independent access to object store
- VOLs: Collective I/O is optional

Data Parallelism for Object Stores

- HDF5 Dataset Chunking is important
- Lack of fine-grained partial I/O in object stores is painful, e.g., Swift

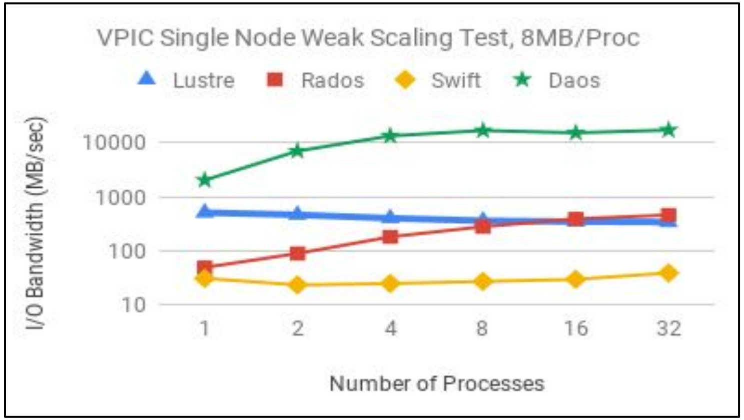
Early Evaluation of Object Stores for HPC Applications

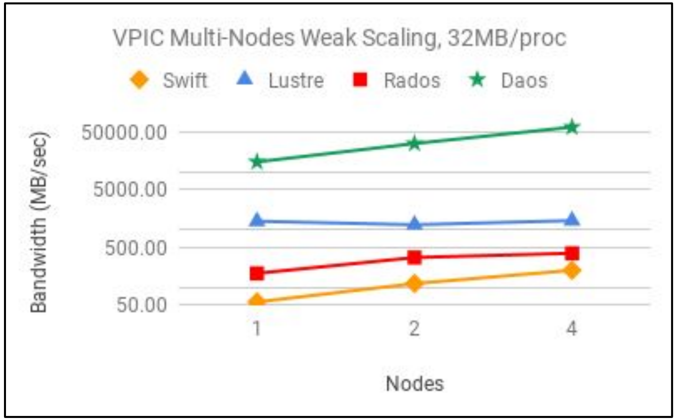
- VOL proof-of-concept
- Compared RADOS and Swift on identical hardware
- Evaluated the scalability of DAOS and Lustre separately
- Compute nodes
  - 1-32 processes
  - 1-4 nodes
- Storage nodes
  - 4 server
  - 48 OSDs

Our Object Store Testbeds

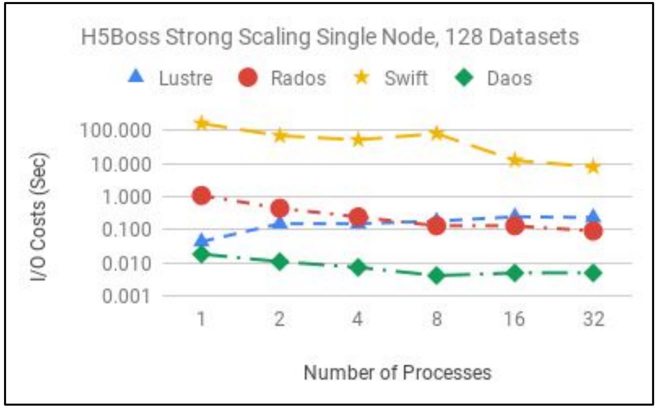
- **Swift, RADOS: Testbed @ NERSC**
  - 4 servers, 1.1 PB capacity, 48 LUNs/NSDs
  - Two failover pairs for Swift, but no failover on Rados
  - Servers are connected with FDR Infiniband
  - Access to server is through NERSC gateway nodes
- **Lustre: Production file system @ NERSC**
  - 248 OST/OSS, 30 PB capacity, 740 GB/sec max bandwidth
  - 130 LNET, Infiniband
- **DAOS: Boro cluster at Intel**
  - 80 nodes, 128G memory each
  - Infiniband single port FDR IO with QSFP
  - Mercury, OFI and PSM2 as network provider

Evaluation: VPIC



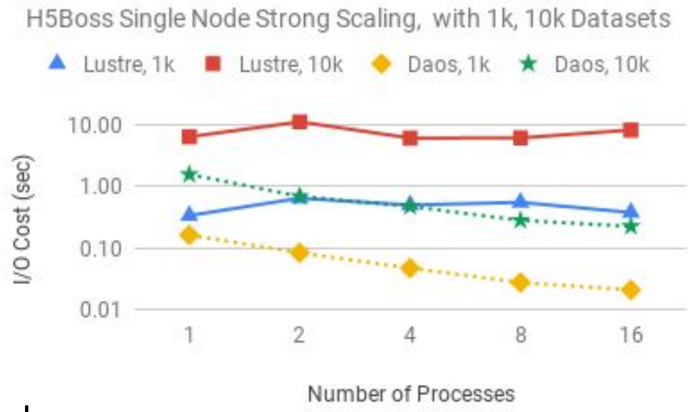


Evaluation: H5BOSS

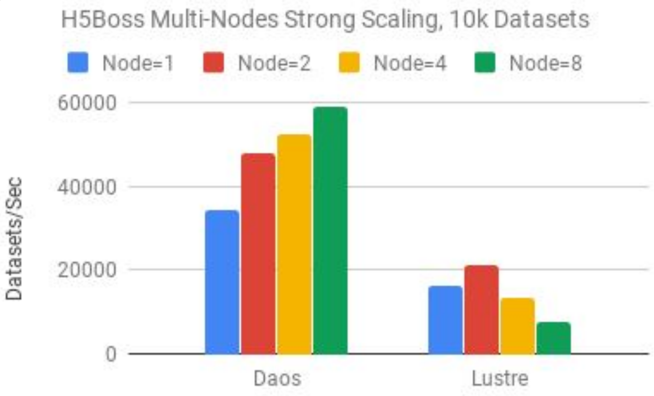


Single Node Test

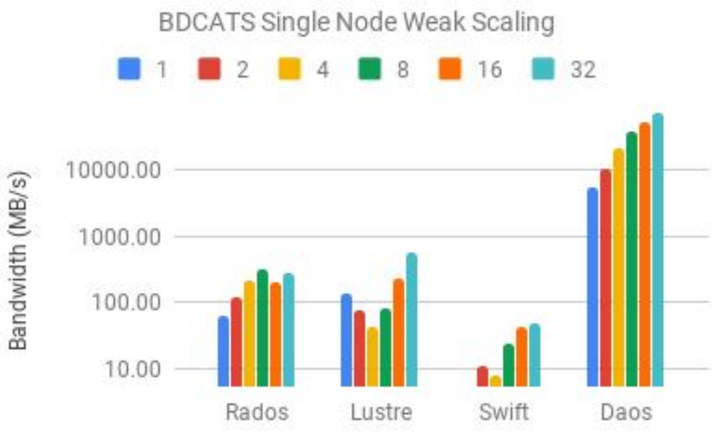
RADOS and Swift both failed with more datasets, and on multiple nodes



Multi-Node Tests

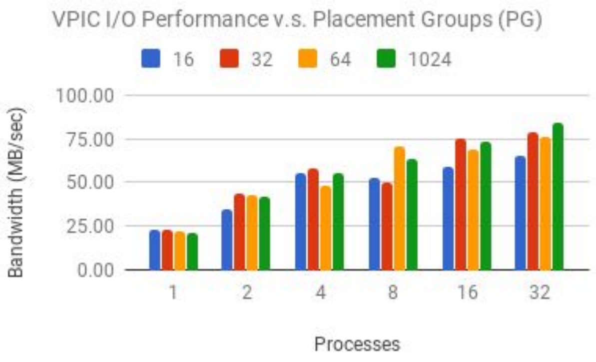


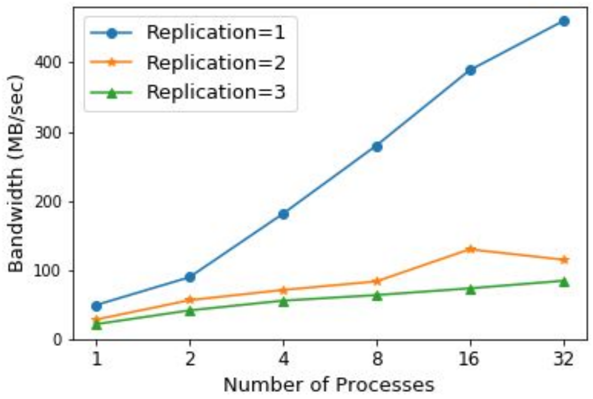
Evaluation: BD-CATS



Observation	
Lustre Read > Write	Lustre Readahead, Less Locking
Rados > Swift	Partial Read, Librados
Daos	Scale with nProc

Object I/O Performance Tuning





- From this we can see:
  - Placement groups are an area to focus on for tuning I/O
  - Disabling replication has large performance benefit (of course!)

- Further investigation needed:
  - Object Stores for HPC need more research and engineering effort
  - Traditional HPC I/O optimizations can be useful in optimizing Object I/O, e.g., Locality aware

Object Store I/O Internals & Notes

- Most object stores are designed to only handle **I/O on entire objects**, instead of finer granularity I/O, such as provided by POSIX, which is required by HPC applications.
- Swift does not support **partial I/O on object**. Although it supports segmented I/O on large objects, the current API can only read/write an entire object. This stops us from performing parallel I/O with chunking support in HDF5.
- **RADOS offers librados for clients** to directly access its OSD (object storage daemon), which is a performance benefit as the gateway node can be bypassed.
- Mapping HDF5's **hierarchical file structure to flat namespace** in object store will require additional tools for users to easily view the file's structure.
- Traditional HPC I/O optimization techniques may be applied in object stores, for example, two-phase collective I/O, as currently each rank issues the I/O to object independently. A two-phase collective I/O-like algorithm is possible when considering the **object locality.**
- Object stores **trade performance for durability**. Reducing the replication size (default is frequently 3) when durability is not a concern for HPC application can increase the bandwidth.

Porting Had Very Low Impact to Apps

NERSC

	VPIC	H5BOSS	BDCATS
SWIFT	7	6	7
RADOS	7	7	7
DAOS	4	4	4

Lines of Code Changed

Possible in Future:

- module load rados
- module load lustre
- module load daos

```
int main()
{
MPI_Init();
...
H5Fcreate();
for (i=0;i<n;i++)
      buffer[i]=i;
H5Dcreate();
H5Dwrite();
H5Fclose();
...
MPI_Finalize();
}
```

Before

H5VLrados_init() ;
H5P_set_fapl_rados() ;

~1-2% code change

```
int main()
{
MPI_Init();
...
H5VLrados_init();
H5Pset_fapl_rados();
H5Fcreate();
for (i=0;i<n;i++)
      buffer[i]=i;
H5Dcreate();
H5Dwrite();
H5Fclose();
...
MPI_Finalize();
}
```

After

U.S. DEPARTMENT OF ENERGY | Office of Science

Conclusion

1. Object stores shows better scalability than POSIX filesystem in various HPC I/O patterns*
2. Object stores are still young for HPC, but traditional HPC I/O optimization may be easily applied
3. HDF5 VOL connectors enable users to use object store <u>transparently</u>, with very small modifications to their applications.
4. *DAOS and other NVM based object stores* are promising for on-platform storage tier

** However, current evaluation scale is small*

Thanks

Project Repo: https://github.com/NERSC/object-store

Rados VOL:

https://bitbucket.hdfgroup.org/users/nfortne2/repos/hdf5_naf/browse?at=hdf5_rados

Daos VOL:

https://bitbucket.hdfgroup.org/users/nfortne2/repos/hdf5_naf/browse?at=refs%2Fheads%2Fhdf5_daosm

Swift VOL:

https://github.com/valiantljk/sci-swift