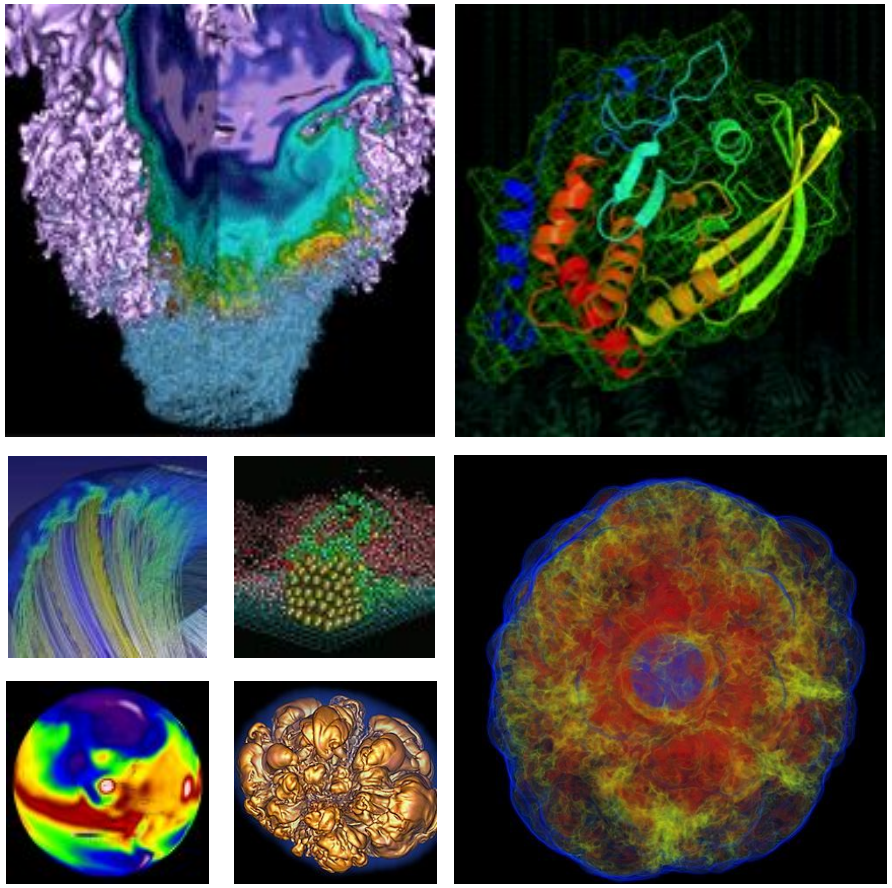
NERSC File Systems and Burst Buffer

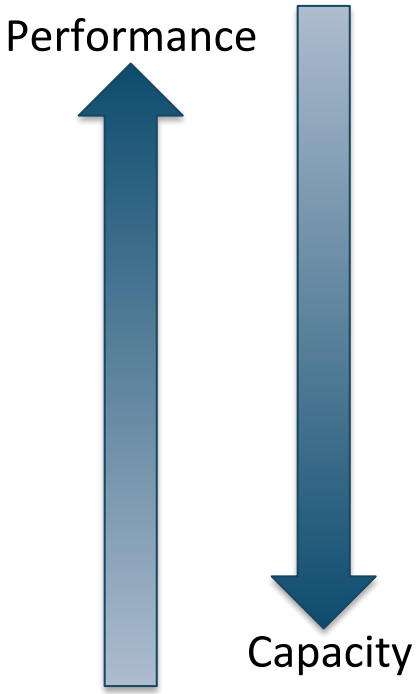


Wahid Bhimji
NERSC Data and Analytics Group

NERSC New User Training

Jan 25th 2019

Simplified NERSC File Systems



Memory

Burst Buffer

Scratch

Project

HPSS

Global Common

1.8 PB SSD Burst Buffer on Cori
Cray Datawarp
1.8 TB/s,
temporary for job or campaign

7.6 PB (Edison) 28 PB (Cori) HDD Scratch
Lustre
700 GB/s,
temporary (12 wk purge)

10.7 PB HDD and SSD Project
Spectrum Scale (GPFS)
150 GB/s, permanent

150 PB Tape Archive
HPSS
Forever

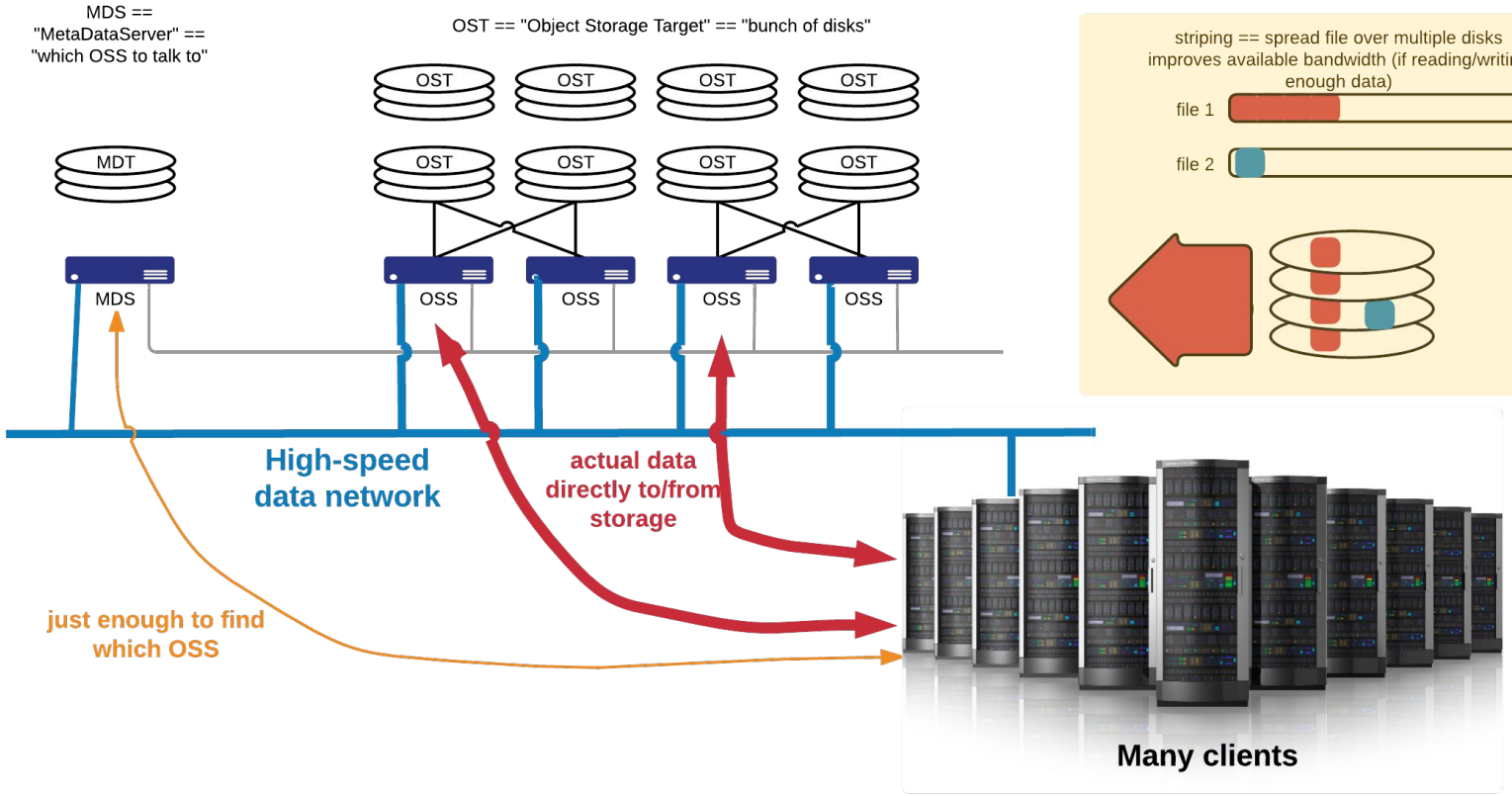
20 TB SSD Software
Spectrum Scale
Permanent,
read-only on computes

Where Do I Put My Data?

- **Burst Buffer**
  - For: Data read in/out by any high IO b/w or IOPS application
  - Truly transient, you must stage in and out from Cori scratch, control with slurm directives (see details later)
  - Can have a ‘persistent’ reservation of up to 20TB, larger on request
  - Scale BW by sizing request, unique MDS so can serve high IOPS workloads

Where Do I Put My Data?

- **Scratch**
  - For data that you don't want to stage into the BB



Where Do I Put My Data?

- **Scratch**
  - Optimal IO by controlling striping (See I/O tips later)
    - By default data is striped across 1 OST, ideal for file per process IO
    - Single shared file IO should be striped according to its size

Size of File	Command
< 1GB	Do Nothing. Use default striping.
~1GB - ~10GB	stripe_small
~10GB - ~100GB	stripe_medium
~100GB - 1TB+	stripe_large

    - Manually query with “lfs getstripe <file_name>” and manually set with “lfs setstripe <empty_file_or_directory_name>”
  - Purged! Files not accessed for more than 12 weeks are automatically deleted

Where Do I Put My Data?

- **Project**
  - For: Large data that you need for the next few years
  - Set up for sharing with group read permissions by default
  - Snapshots automatically back up for the last 7 days. Accidentally delete something? Get it back at /project/projectdirs/<reponame>/.snapshots/<date>
  - Can share data externally by dropping it into a www directory
  - Data is never deleted, usage is managed by quotas
  - Can request quota increases and custom directory names

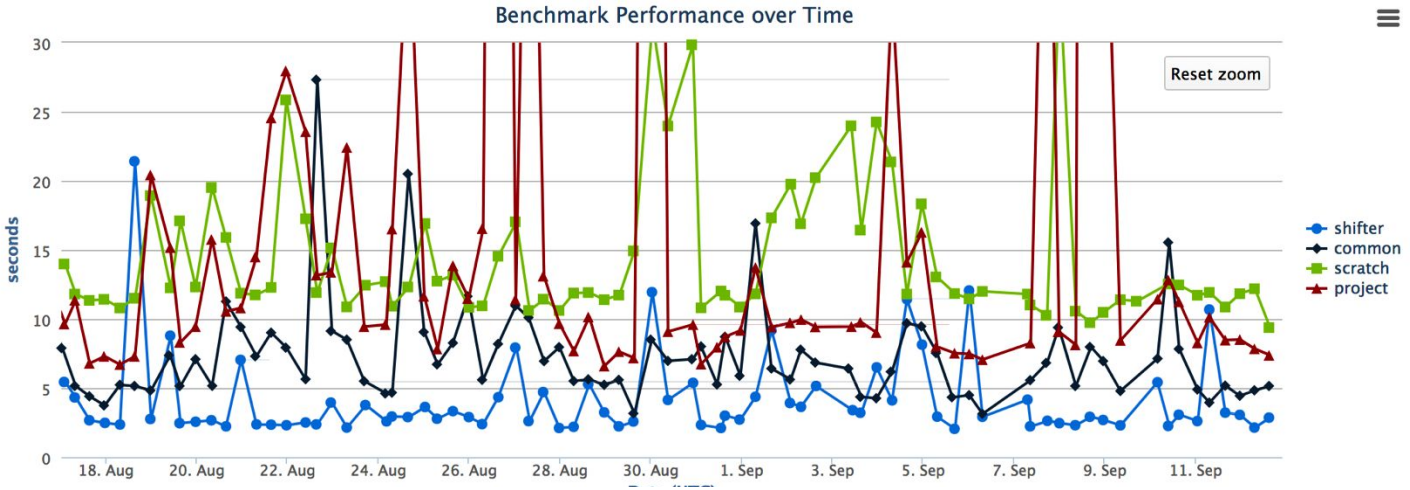
  https://docs.nersc.gov/filesystems/project/

Where Do I Put My Data?

- **HPSS**
  - For: Data from your finished paper, raw data you might need in case of emergency, really hard to generate data
- **HPSS is tape!**
  - Data first hits a spinning disk cache and gets migrated to tapes
  - Files can end up spread all over, so use htar to aggregate into bundles
  - Archive the way you intend to retrieve the data

Where Do I Put My Data?

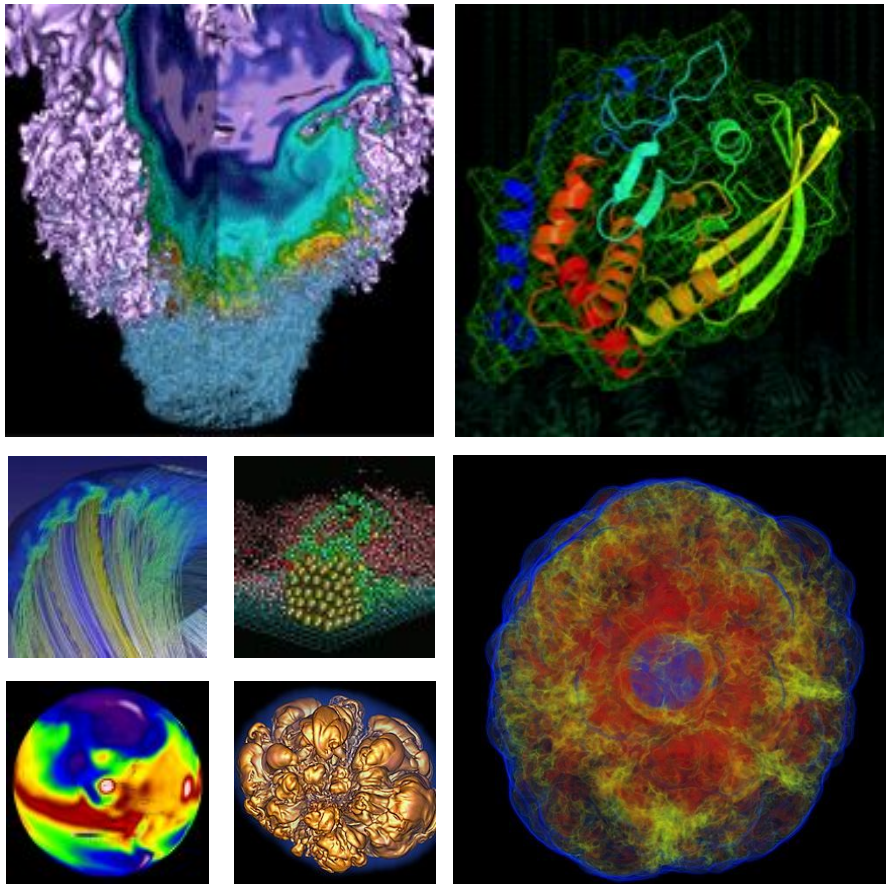
- **Global Common**
  - For: Software stacks
  - Why? Library load performance



- **Group writable directories similar to project, but with a 10 GB quota**
- **Smaller block size for faster compiles than project**

Data Dashboard Demo

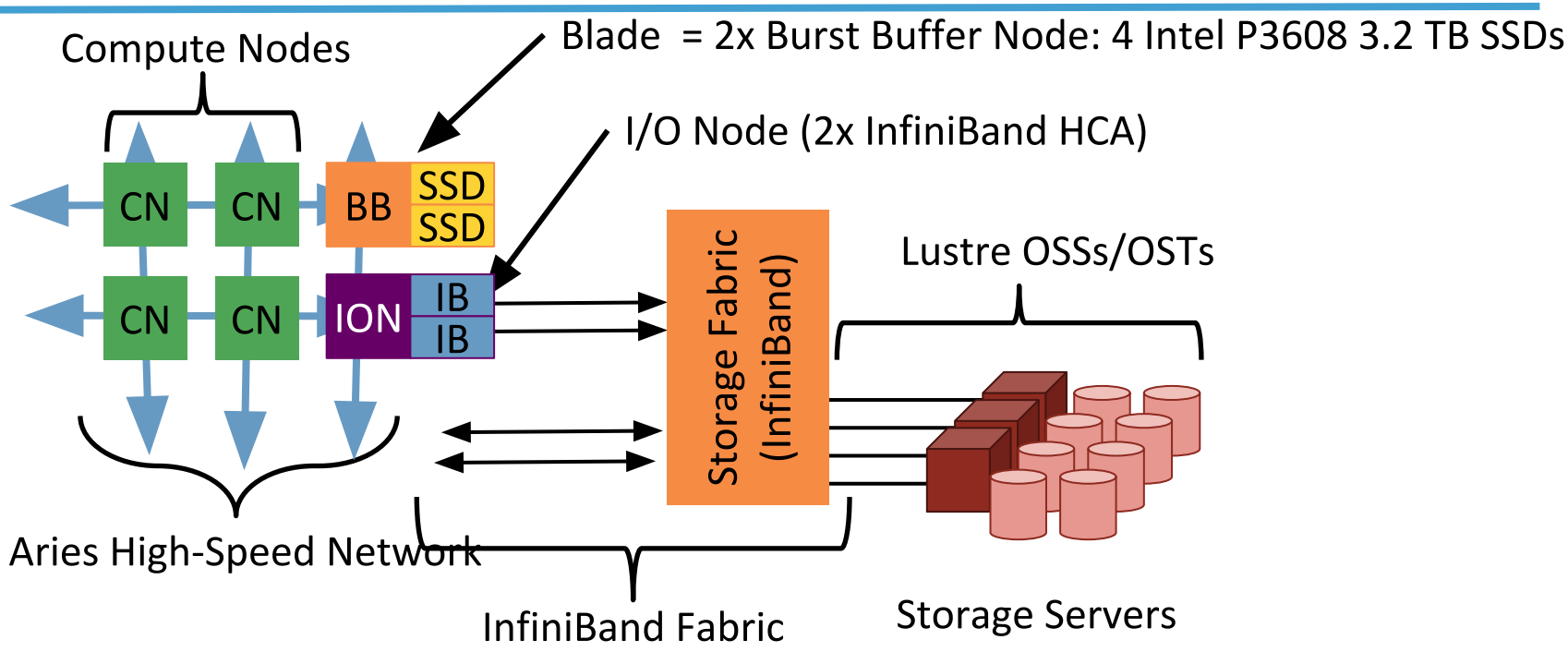
Burst Buffer



Wahid Bhimji
Data and Analytics Services

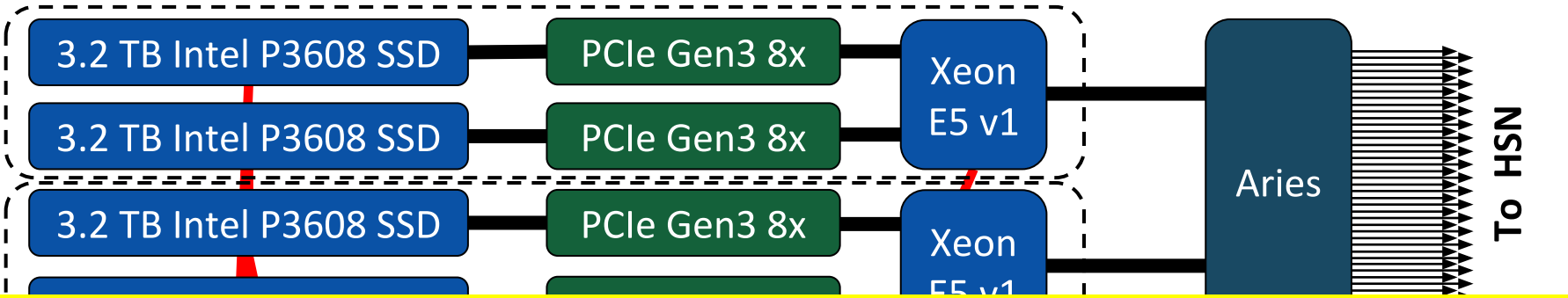
NERSC New Users Training

NERSC/Cray Architecture - Cori

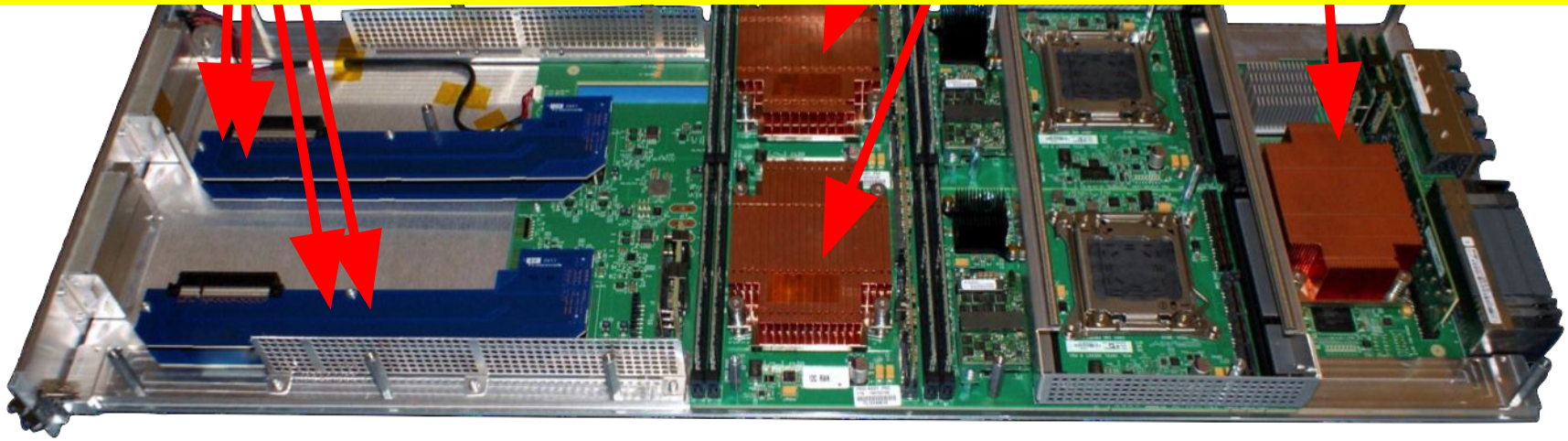


- DataWarp software (integrated with SLURM WLM) allocates portions of available storage to users per-job (or 'persistent').
- Users see a POSIX filesystem
- Filesystem can be striped across multiple BB nodes (depending on allocation size requested)

Burst Buffer Blade = 2xNodes



- ~1.8PiB of SSDs over 288 nodes
- Accessible from all CORI nodes

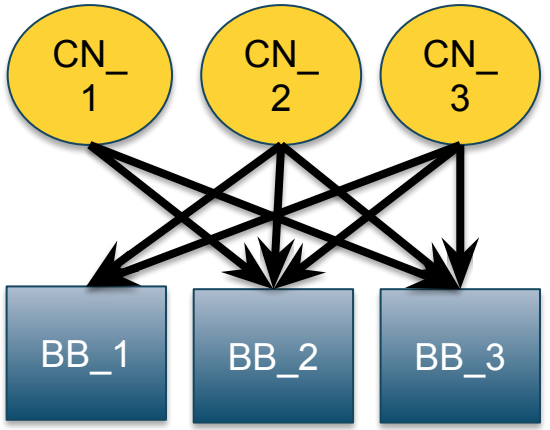


Two kinds of DataWarp Instances

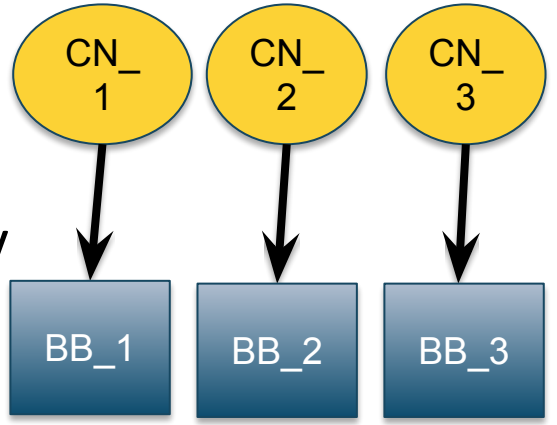
- **"Instance": an allocation on the BB**
- **Can it be shared? What is its lifetime?**
  - **Per-Job Instance**
    - Can only be used by job that creates it
    - Lifetime is the same as the creating job
    - Use cases: PFS staging, application scratch, checkpoints
  - **Persistent Instance**
    - Can be used by any job (subject to UNIX file permissions)
    - Lifetime is controlled by creator
    - Use cases: Frequently reused data(base), Shared data, PFS staging, Coupled job workflow
    - ***NOT for long-term storage of data!***

Two DataWarp Access Modes

- **Striped ("Shared")**
  - Files are striped across all DataWarp nodes
  - Files are visible to all compute nodes
    Aggregates both capacity and BW per file
  - One DataWarp node elected as the metadata server (MDS)



- **Private**
  - File are visible to *only the compute node that created them*
  - Each DataWarp node is an MDS so potentially better metadata performance
  - Like a local disk



How to use DataWarp

- **Principal user access: SLURM Job script directives: `#DW`**
  - Allocate job or persistent DataWarp space
  - Stage files or directories in from PFS to DW; out DW to PFS
  - Access BB mount point via `$DW_JOB_STRIPED`, `$DW_JOB_PRIVATE`, `$DW_PERSISTENT_STRIPED_name`
- **User library API – `libdatawarp`**
  - Allows direct control of staging files asynchronously
  - C library interface
  - https://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/#toc-anchor-8
  - https://github.com/NERSC/BB-unit-tests/tree/master/datawarpAPI

Integration with SLURM

```
#!/bin/bash
#SBATCH -p regular -N 10 -t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=/lustre/inputs destination=$DW_JOB_STRIPED/inputs \
type=directory
#DW stage_in source=/lustre/file.dat destination=$DW_JOB_STRIPED/ type=file
#DW stage_out source=$DW_JOB_STRIPED/outputs destination=/lustre/outputs \
type=directory
srun my.x --indir=$DW_JOB_STRIPED/inputs --infile=$DW_JOB_STRIPED/file.dat \
--outdir=$DW_JOB_STRIPED/outputs
```

- **'type=scratch' – duration just for compute job (i.e. not 'persistent')**
- **'access_mode=striped' – visible to all compute nodes (i.e. not 'private') and striped across multiple BB nodes**
  - Actual distribution across BB Nodes is in units of (configurable) granularity (currently ~80 GB at NERSC in wlm_pool, so 1000 GB would normally be placed on 13 BB nodes)
- **Data 'stage_in' before job start and 'stage_out' after**

Integration with SLURM

```
#!/bin/bash
#SBATCH -p regular -N 10 -t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=/lustre/inputs destination=$DW_JOB_STRIPED/inputs \
type=directory
#DW stage_in source=/lustre/file.dat destination=$DW_JOB_STRIPED/ type=file
#DW stage_out source=$DW_JOB_STRIPED/outputs destination=/lustre/outputs \
type=directory
srun my.x --indir=$DW_JOB_STRIPED/inputs --infile=$DW_JOB_STRIPED/file.dat \
--outdir=$DW_JOB_STRIPED/outputs
```

- **'type=scratch' – duration just for compute job (i.e. not 'persistent')**
- **'access_mode=striped' – visible to all compute nodes (i.e. not 'private') and striped across multiple BB nodes**
  - –Actual distribution across BB Nodes is in units of (configurable) granularity (currently ~80 GB at NERSC in wlm_pool, so 1000 GB would normally be placed on 13 BB nodes)
- **Data 'stage_in' before job start and 'stage_out' after**

Integration with SLURM

```
#!/bin/bash
#SBATCH -p regular -N 10 -t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=/lustre/inputs destination=$DW_JOB_STRIPED/inputs \
type=directory
#DW stage_in source=/lustre/file.dat destination=$DW_JOB_STRIPED/ type=file
#DW stage_out source=$DW_JOB_STRIPED/outputs destination=/lustre/outputs \
type=directory
srun my.x --indir=$DW_JOB_STRIPED/inputs --infile=$DW_JOB_STRIPED/file.dat \
--outdir=$DW_JOB_STRIPED/outputs
```

- **'type=scratch' – duration just for compute job (i.e. not 'persistent')**
- **'access_mode=striped' – visible to all compute nodes (i.e. not 'private') and striped across multiple BB nodes**
  - Actual distribution across BB Nodes is in units of (configurable) granularity (currently ~80 GB at NERSC in wlm_pool, so 1000 GB would normally be placed on 13 BB nodes)
- **Data 'stage_in' before job start and 'stage_out' after**

Integration with SLURM

```
#!/bin/bash
#SBATCH -p regular -N 10 -t 00:10:00
#DW jobdw capacity=1000GB access_mode=striped type=scratch
#DW stage_in source=/lustre/inputs destination=$DW_JOB_STRIPED/inputs \
type=directory
#DW stage_in source=/lustre/file.dat destination=$DW_JOB_STRIPED/ type=file
#DW stage_out source=$DW_JOB_STRIPED/outputs destination=/lustre/outputs \
type=directory
srun my.x --indir=$DW_JOB_STRIPED/inputs --infile=$DW_JOB_STRIPED/file.dat \
--outdir=$DW_JOB_STRIPED/outputs
```

- **'type=scratch' – duration just for compute job (i.e. not 'persistent')**
- **'access_mode=striped' – visible to all compute nodes (i.e. not 'private') and striped across multiple BB nodes**
  - Actual distribution across BB Nodes is in units of (configurable) granularity (currently ~80 GB at NERSC in wlm_pool, so 1000 GB would normally be placed on 13 BB nodes)
- **Data 'stage_in' before job start and 'stage_out' after**

Integration with SLURM

- **Using a *persistent* DataWarp instance**
  - Lifetime different from the batch job
  - Usable by any batch job (posix permissions permitting)
  - `name=xyz`: Name of persistent instance to use

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#BB create_persistent name=myBBname capacity=10GB access=striped type=scratch
```

Delete

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#BB destroy_persistent name=myBBname
```

Use in another job

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#DW persistentdw name=myBBname
mkdir $DW_PERSISTENT_STRIPED_myBBname/test1
srun a.out INSERT_YOUR_CODE_OPTIONS_HERE
```

Integration with SLURM

- **Using a *persistent* DataWarp instance**
  - –Lifetime different from the batch job
  - –Usable by any batch job
  - –`name=xyz`: Name of persistent instance to use

```
1 #!/bin/bash
2 #SBATCH -p debug
3 #SBATCH -N 1
4 #SBATCH -t 00:05:00
5 #BB create_persistent name=myBBname capacity=10GB access=striped type=scratch
```

Delete

```
1 #!/bin/bash
2 #SBATCH -p debug
3 #SBATCH -N 1
4 #SBATCH -t 00:05:00
5 #BB destroy_persistent name=myBBname
```

Use in another job

```
1 #!/bin/bash
2 #SBATCH -p debug
3 #SBATCH -N 1
4 #SBATCH -t 00:05:00
5 #DW persistentdw name=myBBname
6 mkdir $DW_PERSISTENT_STRIPED_myBBname/test1
7 srun a.out INSERT_YOUR_CODE_OPTIONS_HERE
```

Integration with SLURM

- **Using a *persistent* DataWarp instance**
  - Lifetime different from the batch job
  - Usable by any batch job
  - `name=xyz`: Name of persistent instance to use

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#BB create_persistent name=myBBname capacity=10GB access=striped type=scratch
```

Delete

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#BB destroy_persistent name=myBBname
```

Use in another job

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#DW persistentdw name=myBBname
mkdir $DW_PERSISTENT_STRIPED_myBBname/test1
srun a.out INSERT_YOUR_CODE_OPTIONS_HERE
```

Integration with SLURM

- **Using a *persistent* DataWarp instance**
  - Lifetime different from the batch job
  - Usable by any batch job
  - `name=xyz`: Name of persistent instance to use

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#BB create_persistent name=myBBname capacity=10GB access=striped type=scratch
```

Delete

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#BB destroy_persistent name=myBBname
```

Use in another job

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:05:00
#DW persistentdw name=myBBname
mkdir $DW_PERSISTENT_STRIPED_myBBname/test1
srun a.out INSERT_YOUR_CODE_OPTIONS_HERE
```

Tools

- **Slurm command on the login nodes to see your allocation**

```
wbhimji@cori07:~> scontrol show burst
Name=cray DefaultPool=wlm_pool Granularity=82496M TotalSpace=1192325G
UsedSpace=51147520M
  AltPoolName[0]=sm_pool Granularity=20624M TotalSpace=476930G UsedSpace=0
 Flags=EnablePersistent,TeardownFailure
  StageInTimeout=86400 StageOutTimeout=86400 ValidateTimeout=5
 GetSysState=/opt/cray/dw_wlm/default/bin/dw_wlm_cli
  Allocated Buffers:
    JobID=3832168 CreateTime=2017-02-22T12:02:35 Pool=wlm_pool Size=1072448M
State=staged-in UserID=epif(57632)
```

- **Datawarp command on the *compute* nodes for more details:**

```
prompt:#> module load dws
prompt:#> dwstat instances
Inst state sess  bytes  nodes           created  expiration intact label public confs
  29 CA--- 36    16MiB  1     2015-08-21T13:10:05   never     true   blast true       1
  37 CA--- 44    16MiB  1     2015-08-26T08:51:28   never     true   I44-0 false      2
```

example script for extracting job usage information from dwstat at:

http://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/#toc-anchor-6

Striping, granularity and pools

- **DataWarp nodes are configured to have "granularity"**
  - Minimum amount of data that will land on one node
- **"pools" of DataWarp nodes**
  - wlm_pool (default): ~20.14 GiB
- **For example, 1.2TiB will be striped over 61 BB nodes in wlm_pool**
  - No guarantee that allocation will be spread evenly over SSDs - may see >1 "grain" on a single node (see script on previous page if you really care on the layout)

Benchmark Performance

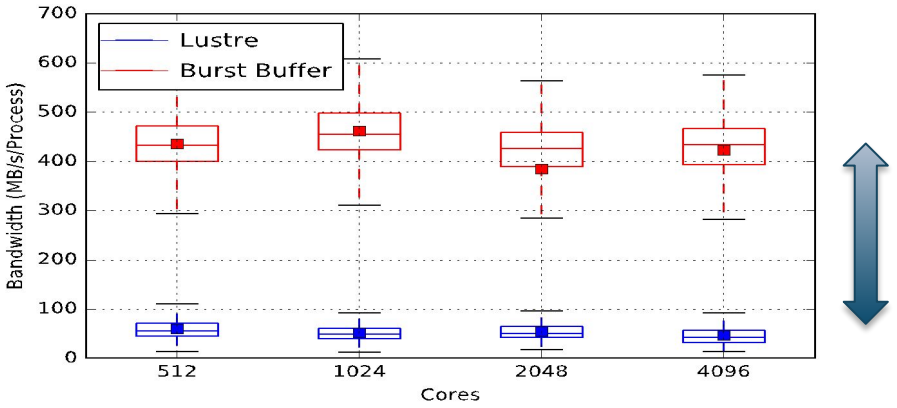
- **Burst Buffer does very well on benchmark performance**
  - Out-performs Lustre significantly

**Bandwidth tests: 8 GB block-size 1MB transfers*
IOPS tests: 1M blocks 4k transfer

	IOR Posix FPP		IOR MPIO Shared File		IOPS	
	Read	**Write**	**Read**	**Write**	**Read**	**Write**
Best Measured (287 Burst Buffer Nodes : 11120 Compute Nodes; 4 ranks)*	1.7 TB/s	1.6 TB/s	1.3 TB/s	1.4 TB/s	28M	13M

- **Many different science codes have seen improvements by switching**

ATLAS QuickAna ROOT/python



x4 improvement

Summary

- **NERSC has a Burst Buffer for open science**
- **Users are able to take advantage of SSD performance and on-demand filesystems**
  - –Flexible configuration
  - –Some tuning may be required to maximise performance
- **Users generally experience good performance and stable service**
  - –But syntax and error-messages can be esoteric
  - –And performance tuning different to other systems
  - –Let us know your issues and experiences…

Resources

- **NERSC Burst Buffer Web Pages**

http://www.nersc.gov/users/computational-systems/cori/burst-buffer/

- **Example batch scripts**

http://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/

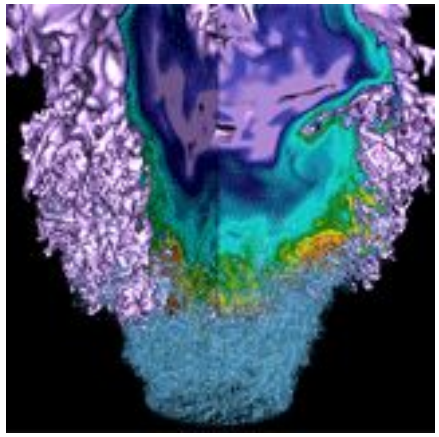
- **Crays DataWarp User Guide**

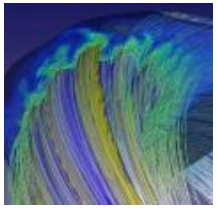
http://docs.cray.com/PDF/XC_Series_DataWarp_User_Guide_CLE60UP04_S-2558.pdf

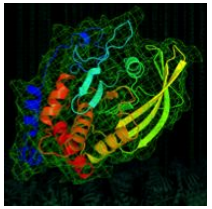
- **Burst Buffer Early User Program Paper**

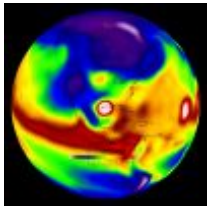
http://www.nersc.gov/assets/Uploads/Nersc-BB-EUP-CUG.pdf

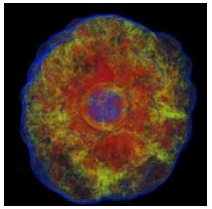
Extra slides

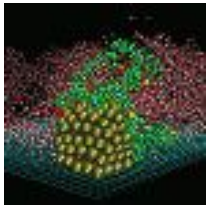








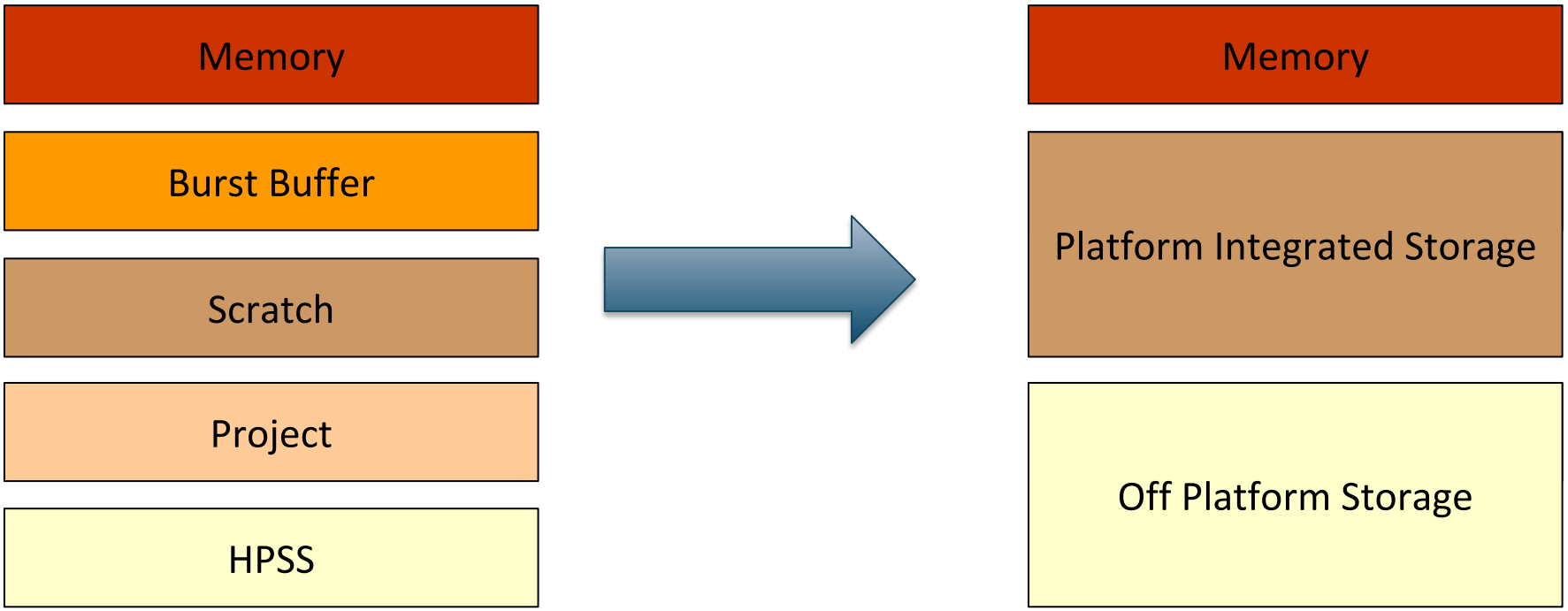




Gaps

- Shared file systems quotas and permissions drift
- Moving between tiers is painful
- Data sizes are growing and it's getting harder to find your data
- NERSC project file system is undersized

Future Plans



SSD write protection

- **SSDs support a set amount of write activity before they wear out**
- **Runaway application processes may write an excessive amount of data, and therefore, "destroy" the SSDs**
- **Three write protection policies**
  - Maximum number of bytes written in a period of time
  - Maximum size of a file in a namespace
  - Maximum number of files allowed to be created in a namespace
- **Log, error, log and error**
  - `--EROFS` (write window exceeded)
  - `--EMFILE` (maximum files created exceeded)
  - `--EFBIG` (maximum file size exceeded)

Performance tips

- **Stripe your files across multiple BB servers**
  - To obtain good scaling, need to drive IO with sufficient compute - scale up # BB nodes with # compute nodes