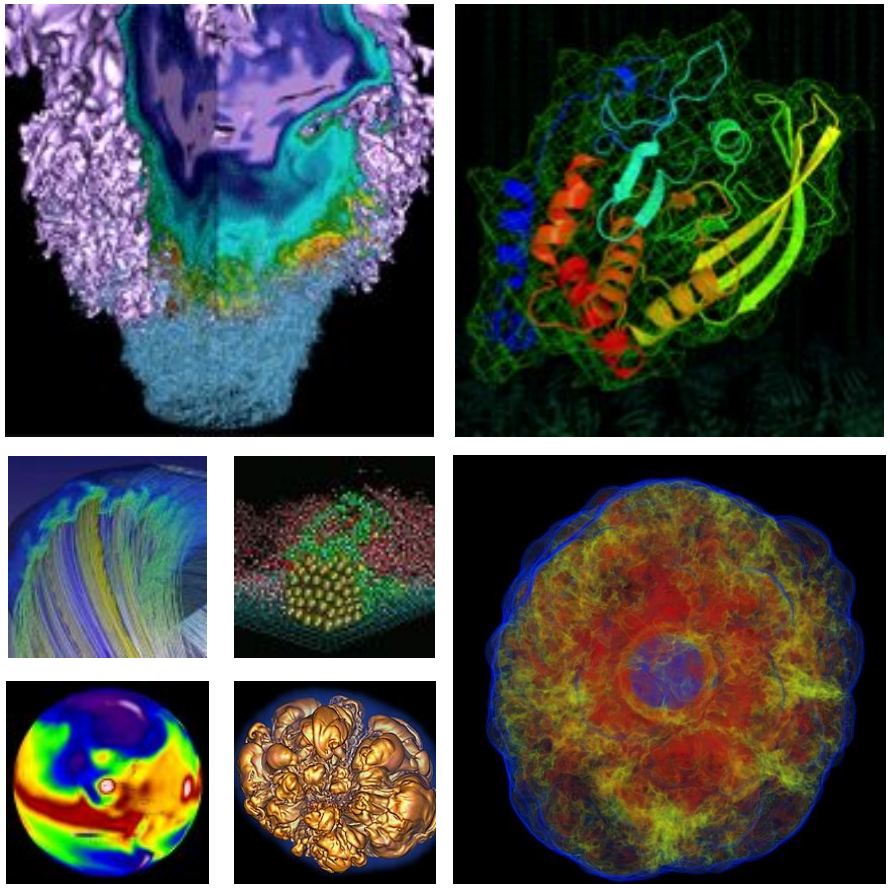
Burst Buffer



Wahid Bhimji

Data and Analytics Services

NERSC New Users Training

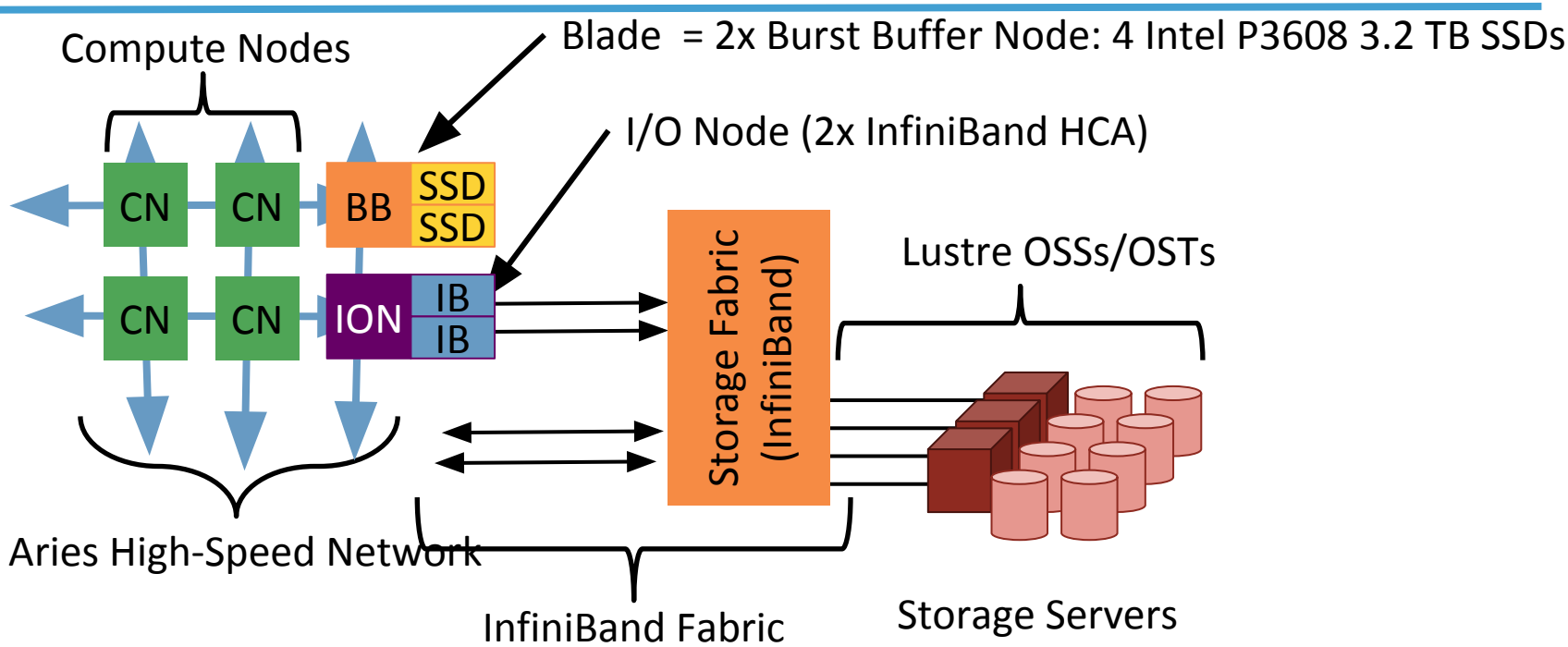
24th Feb 2017

Why an SSD Burst Buffer?

- **Motivation: Handle spikes in I/O bandwidth**
  - Reduce overall application run time
  - Compute resources are idle during I/O bursts
- **Some user applications have challenging I/O patterns**
  - High IOPs, random reads, different concurrency…
- **Cost rationale: Disk-based PFS bandwidth is expensive**
  - Disk capacity is relatively cheap
  - SSD *bandwidth* is relatively cheap

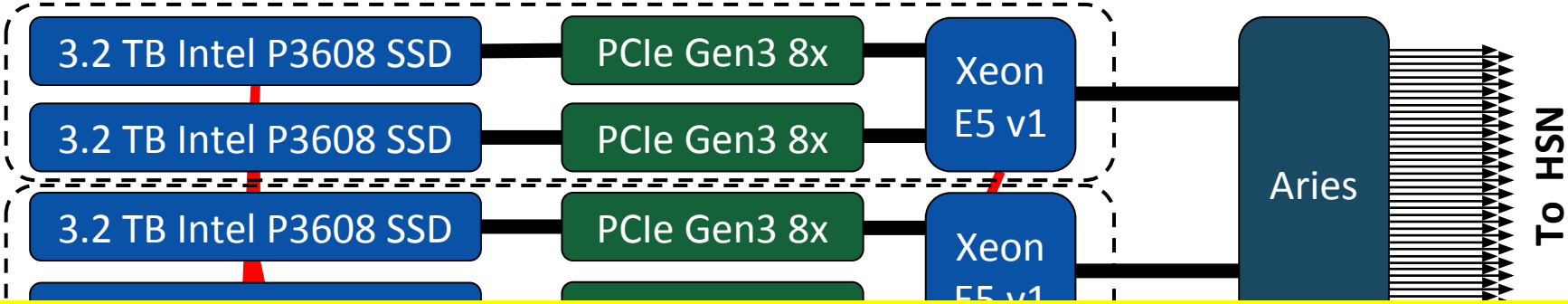
    =>Separate bandwidth and spinning disk
    - Provide high BW without needing PFS capacity
- **Huge POSIX parallel filesystems don't scale**
    - Build filesystems on demand
    - Leverage Cray Aries network speed

NERSC/Cray Architecture

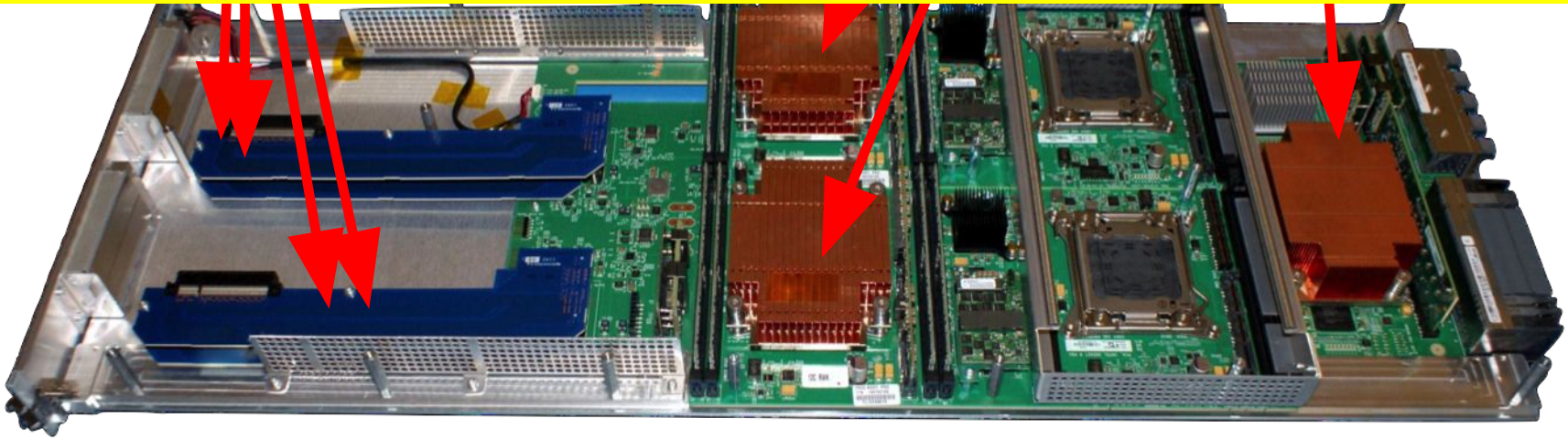


- DataWarp software (integrated with SLURM WLM) allocates portions of available storage to users per-job (or 'persistent').
- Users see a POSIX filesystem
- Filesystem can be striped across multiple BB nodes (depending on allocation size requested)

Burst Buffer Blade = 2xNodes



- ~1.8PiB of SSDs over 288 nodes
- Accessible from both HSW and KNL nodes

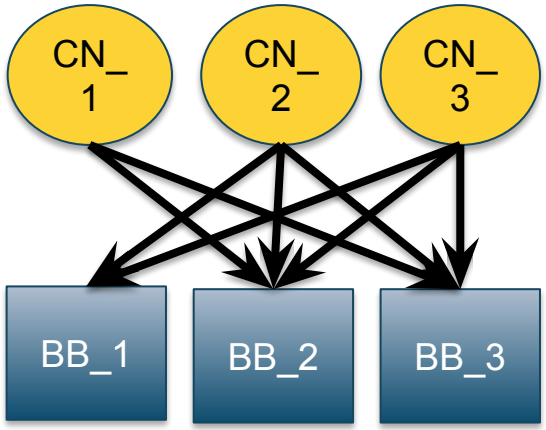


Two kinds of DataWarp Instances

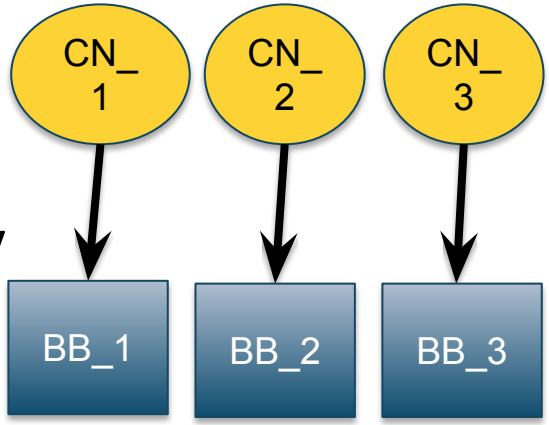
- **"Instance": an allocation on the BB**
- **Can it be shared? What is its lifetime?**
  - **Per-Job Instance**
    - Can only be used by job that creates it
    - Lifetime is the same as the creating job
    - Use cases: PFS staging, application scratch, checkpoints
  - **Persistent Instance**
    - Can be used by any job (subject to UNIX file permissions)
    - Lifetime is controlled by creator
    - Use cases: 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 200 GB at NERSC in wlm_pool, so 1000 GB would normally be placed on 5 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 200 GB at NERSC in wlm_pool, so 1000 GB would normally be placed on 5 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 200 GB at NERSC in wlm_pool, so 1000 GB would normally be placed on 5 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

```
#!/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
```

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
- **Two "pools" of DataWarp nodes, with different granularity**
  - wlm_pool (default): 86.5GiB
    - `#DW jobdw capacity=1000GB access_mode=striped type=scratch pool=wlm_pool`
  - sm_pool: 20.14 GiB
    - `#DW jobdw capacity=1000GB access_mode=striped type=scratch pool=sm_pool`
- **For example, 1.2TiB will be striped over 14 BB nodes in wlm_pool, but over 60 BB nodes in sm_pool**
  - **No guarantee that allocation will be spread evenly ove - may see >1 "grain" on a single node (see script on pre** page if you really care on the layout on a particular job)

Benchmark Performance

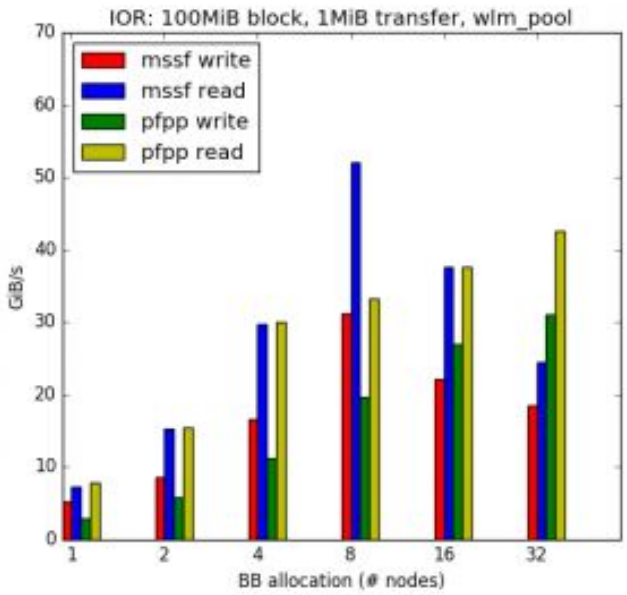
- **Burst Buffer is doing very well against benchmark performance targets**
  - Out-performs Lustre significantly

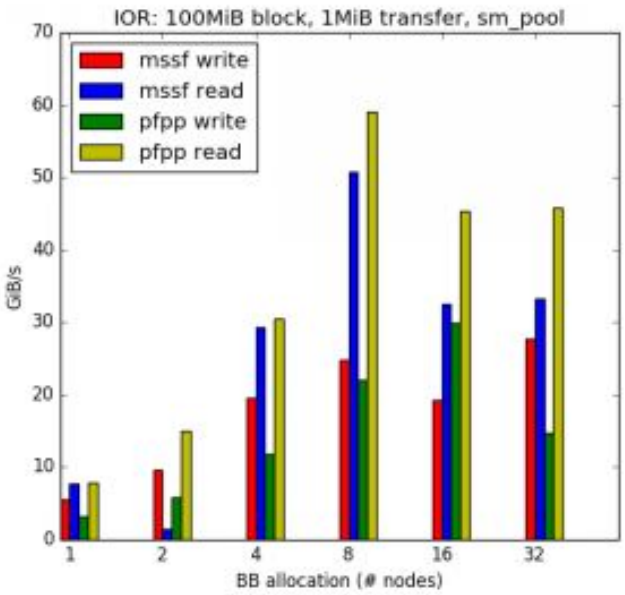
	IOR Posix FPP		IOR MPIO Shared File		IOPS	
	Read	**Write**	**Read**	**Write**	**Read**	**Write**
Best Measured (287 Burst Buffer Nodes : 11120 Compute Nodes; 4 ranks/node)*	1.7 TB/s	1.6 TB/s	1.3 TB/s	1.4 TB/s	28M	13M

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

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





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/

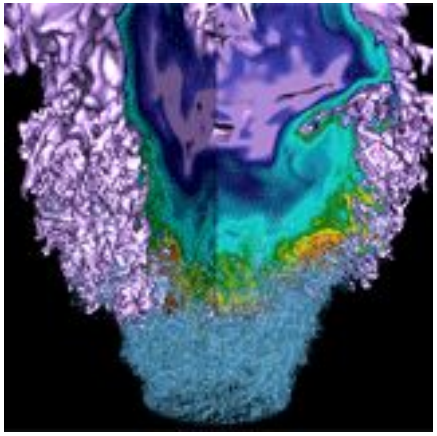
- **Burst Buffer Early User Program Paper**

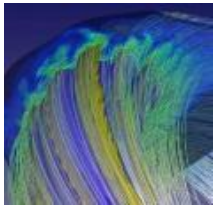
http://www.nersc.gov/assets/Uploads/Nersc-BB-EUP-CUG.pdf

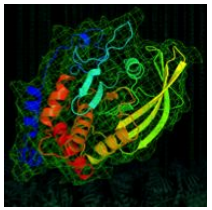
Summary

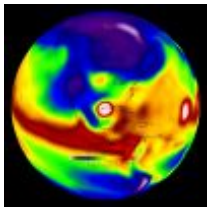
- **NERSC has the first Burst Buffer for open science in the USA**
  - And the first in the world that is being tested for real use cases!
- **Users are able to take advantage of SSD performance**
  - Some tuning may be required to maximise performance
- **Many bugs now worked through**
  - But care is needed when using this new technology!
- **User experience today is generally good**
  - Let us know your experiences...

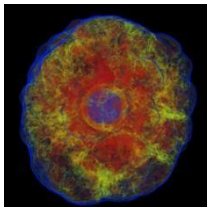
Extra slides

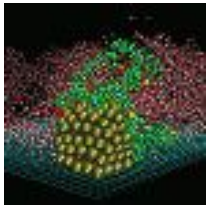












SSD write protection

- **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)