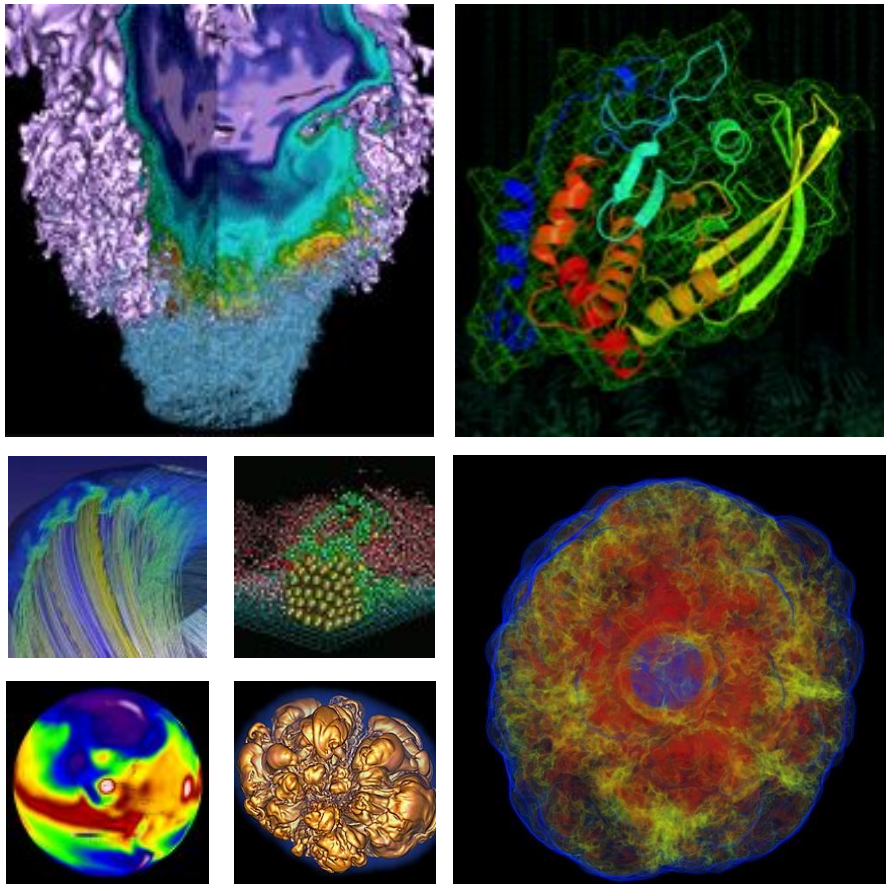
Getting your hands on Cori's Burst Buffer



Debbie Bard
Data and Analytics Services

ATPSEC 2017 IO Day

Scratch allocation

```
#!/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**

Scratch allocation

```
#!/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**

Scratch allocation

```
#!/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**

Scratch allocation

```
#!/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**

Step 1: log onto Cori

- **We have temporary user accounts for NERSC (that will expire at the end of the day) and a reservation of Haswell nodes on Cori.**
- **Using your training account (or your own NERSC account): ssh username@cori.nersc.gov**
  - Note that if you use your own NERSC account you won't be part of our compute reservation but you can still submit jobs that run on the Burst Buffer.

```
                    complete.

Cori:           05/16/17 17:00-05/17/17 17:00 PDT, Dedicated runs.
                 Cori HPL Dedicated run. Logins and job submission available.
                 Jobs will remained queued until the dedicated run is complete.

---------------------------------- Past Outages ----------------------------------

Cori:           05/04/17 8:00-16:30 PDT, Scheduled maintenance.
                 Cori will be undergoing planned maintenance during this time
                 period.

Cori:           05/04/17 17:22-22:15 PDT, System in degraded mode.
                 Cori is experiencing a possible hardware issue that is under
                 investigation. Logins and job submission are available but new
                 jobs will not start

For past outages, see: http://my.nersc.gov/outagelog-cs.php

------------------------------------------------------------------------------

train61@cori07:~>
```

Copy over the example scripts and test data

- **Pull down the example scripts and the test data file onto your scratch space**
  - We're using scratch (i.e. Lustre) because it's currently the only filesystem the burst buffer can access on Cori.

```
cd $SCRATCH
git clone https://github.com/NERSC/train.git
cd train/atpesc-IO-day/IntroToBB/
mkdir data
cp /global/cscratch1/sd/djbard/test1Gb.db data/
```

Run the first simple script!

- **Go to the directory `train/atpesc-IO-day/IntroToBB/examples` and look at the example script "scratch.sh":**

- **Note: on this slide the wrong reservation name is given - but should be correct in the github scripts. Use:**
  - #SBATCH -C haswell
  - #SBATCH --reservation="atpesctrain"

```
#!/bin/bash

#### Which partition? Use "regular" for the reservation
#SBATCH -p regular

#### name of the training reservation
#SBATCH --reservation="csgftrain"

#### How many nodes?
#SBATCH -N 1

#### How long to run the job?
#SBATCH -t 00:1:00

#### Our reservation is for KNL nodes
#SBATCH -C knl

#### Name the job
#SBATCH -J "job_scratch"

#### Set the output file name
#SBATCH -o "job_scratch.log"

#### Request a 200GB scratch allocation, striped over BB nodes, in the default
pool (which has 82GiB granularity, so this gives you grains on 3 BB nodes)
#DW jobdw capacity=200GB access_mode=striped type=scratch pool=wlm_pool

#### print the mount point of your BB allocation on the compute nodes
echo "*** DW path is:"
echo $DW_JOB_STRIPED

#### Write a file on the BB
echo "*** saying hello...."
echo "Hello! I'm on the Burst Buffer!" > $DW_JOB_STRIPED/hello.txt

echo "*** ls -larth $DW_JOB_STRIPED/"
ls -larth $DW_JOB_STRIPED/

echo "*** cat $DW_JOB_STRIPED/hello.txt"
cat $DW_JOB_STRIPED/hello.txt
```

"atpesctrain"

haswell

Run the first simple script!

- **Submit the job using “sbatch scratch.sh”**
- **User “sqs” and “squeue” to view the status of your job**

```
train61@cori07:~> cd  $SCRATCH/CUG2017/examples
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> ls
IOR                  destroy_persistent.sh  scratch.sh   stage_out.sh
create_persistent.sh  run_IOR.sh            stage_in.sh  use_persistent.sh
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples>
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples>
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> sbatch  scratch.sh
Submitted batch job 4883133
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> sqs
JOBID             ST   REASON       USER         NAME         NODES        USED         REQ
UESTED    SUBMIT               PARTITION    RANK_P       RANK_BF
4883133           PD   BurstBuffe   train61      scratch      1            0:00         5:0
0         2017-05-08T10:19:56   debug        11811        N/A
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> sqs
JOBID             ST   REASON       USER         NAME         NODES        USED         REQUESTED    SUBMIT               PARTITION    RANK_P       RANK_BF
4883133           PD   BurstBuffe   train61      scratch      1            0:00         5:00         2017-05-08T10:19:56   debug        11811        N/A
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> squeue -u train61
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           4883133     debug  scratch  train61 PD       0:00      1 (BurstBufferResources)
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> squeue -u train61 -l
Mon May  8 10:20:53 2017
             JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
           4883133     debug  scratch  train61  PENDING       0:00      5:00      1 (BurstBufferResources)
```

Run the first simple script!

- **Submit the job using “sbatch scratch.sh”**
- **User “sqs” and “squeue” to view the status of your job**

```
train61@cori07:~> cd  $SCRATCH/CUG2017/examples
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> ls
IOR                  destroy_persistent.sh  scratch.sh   stage_out.sh
create_persistent.sh  run_IOR.sh            stage_in.sh  use_persistent.sh
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples>
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples>
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> sbatch  scratch.sh
Submitted batch job 4883133
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> sqs
JOBID             ST   REASON       USER         NAME         NODES        USED         REQ
UESTED    SUBMIT               PARTITION    RANK_P       RANK_BF
4883133           PD   BurstBuffe   train61      scratch      1            0:00         5:0
0         2017-05-08T10:19:56   debug        11811        N/A
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> sqs
JOBID             ST   REASON       USER         NAME         NODES        USED         REQUESTED    SUBMIT               PARTITION    RANK_P       RANK_BF
4883133           PD   BurstBuffe   train61      scratch      1            0:00         5:00         2017-05-08T10:19:56   debug        11811        N/A
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> squeue -u train61
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           4883133     debug  scratch  train61 PD       0:00      1 (BurstBufferResources)
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> squeue -u train61 -l
Mon May  8 10:20:53 2017
             JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
           4883133     debug  scratch  train61  PENDING       0:00      5:00      1 (BurstBufferResources)
```

View Burst Buffer status

- **User "scontrol show burst" to look at what the Burst Buffer is doing right now**

```
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples> scontrol show burst
Name=cray DefaultPool=wlm_pool Granularity=82496M TotalSpace=1192325G FreeSpace=893277G UsedSpace=272814272M
  AltPoolName[0]=dev_pool Granularity=20624M TotalSpace=47693G FreeSpace=47693G UsedSpace=0
  AltPoolName[1]=sm_pool Granularity=20624M TotalSpace=476930G FreeSpace=476930G UsedSpace=0
  Flags=EnablePersistent,TeardownFailure
  StageInTimeout=86400 StageOutTimeout=86400 ValidateTimeout=5 OtherTimeout=300
  GetSysState=/opt/cray/dw_wlm/default/bin/dw_wlm_cli
  Allocated Buffers:
    JobID=4829054_2(4829056) CreateTime=2017-05-04T17:01:02 Pool=wlm_pool Size=82496M State=staged-in UserID=aclum(16603)
    JobID=4883133 CreateTime=2017-05-08T10:21:51 Pool=wlm_pool Size=247488M State=staged-in UserID=train61(47535)
    JobID=4831969 CreateTime=2017-05-05T09:37:25 Pool=wlm_pool Size=8414592M State=staged-in UserID=jyc(57038)
    JobID=4831963 CreateTime=2017-05-05T09:37:25 Pool=wlm_pool Size=3299840M State=staged-in UserID=jyc(57038)
    Name=octotiger CreateTime=2017-03-29T01:18:15 Pool=(null) Size=10312G State=allocated UserID=sithhell(61642)
    JobID=4829283 CreateTime=2017-05-07T19:21:56 Pool=wlm_pool Size=21036480M State=staged-in UserID=bharti(70083)
    Name=PDC_BB CreateTime=2017-04-24T17:14:51 Pool=(null) Size=164992M State=allocated UserID=bharti(70083)
    Name=my_persistent_reservation CreateTime=2017-05-07T12:18:33 Pool=wlm_pool Size=329984M State=allocated UserID=djbard(61692)
    Name=celestebb CreateTime=2017-04-09T11:25:05 Pool=(null) Size=114339456M State=allocated UserID=kpamnany(69499)
    Name=celestebb2 CreateTime=2017-03-24T20:14:11 Pool=(null) Size=114339456M State=allocated UserID=kpamnany(69499)
  Per User Buffer Use:
    UserID=aclum(16603) Used=82496M
    UserID=train61(47535) Used=247488M
    UserID=jyc(57038) Used=11714432M
    UserID=sithhell(61642) Used=10312G
    UserID=bharti(70083) Used=21201472M
    UserID=djbard(61692) Used=329984M
    UserID=kpamnany(69499) Used=228678912M
train61@cori07:/global/cscratch1/sd/train61/CUG2017/examples>
```

Stage in data to a scratch allocation

```
#### name of the training reservation
#SBATCH --reservation="csgftrain"

#### How many nodes?
#SBATCH -N 1

#### How long to run the job?
#SBATCH -t 00:1:00

#### Our reservation is for KNL nodes
#SBATCH -C knl

#### Name the job
#SBATCH -J "job_stage_in"

#### Set the output file name
#SBATCH -o "job_stage_in.log"

#### Request a 200GB scratch allocation, striped over BB nodes
#DW jobdw capacity=200GB access_mode=striped type=scratch pool=wlm_pool

#### Stage in a directory. Remember to change this directory to your training acco
unt scratch directory! You can also stage_in a file by specifying "type=file".
#DW stage_in source=/global/cscratch1/sd/djbard/train/csgf-hpc-day/IntroToBB/data
 destination=$DW_JOB_STRIPED/data  type=directory



#### print the mount point of your BB allocation on the compute nodes
echo "*** The path to my BB allocation is:"
echo $DW_JOB_STRIPED

#### Check what's been staged in to your allocation
echo "*** what's on my scratch allocation?"

echo "ls -lrtha  $DW_JOB_STRIPED/ "
ls -lrtha  $DW_JOB_STRIPED/

echo "ls -lrtha  $DW_JOB_STRIPED/data/  "
ls -lrtha  $DW_JOB_STRIPED/data/
```

"atpesctrain"

haswell

- Look at "stage_in.sh".
- Edit it to point to your OWN file/directory that you want to stage_in
- Submit the job
- Check the output log file - did the directory stage in as expected?

Stage out data from a scratch allocation

```
#!/bin/bash

#### Which partition?
#SBATCH -p regular

#### name of the training reservation
#SBATCH --reservation="csgftrain"

#### How many nodes?
#SBATCH -N 1

#### How long to run the job?
#SBATCH -t 00:01:00

#### Our reservation is for KNL nodes
#SBATCH -C knl

#### Name the job
#SBATCH -J "job_stage_out"

#### Set the output file name
#SBATCH -o "job_stage_out.log"

#### Request a 200GB scratch allocation, striped over BB nodes
#DW jobdw capacity=200GB access_mode=striped type=scratch pool=wlm_pool

#### Stage a file out of the BB onto scratch.
#DW stage_out destination=/global/cscratch1/sd/djbard/train/csgf-hpc-day/IntroToBB
/data/hello.txt source=$DW_JOB_STRIPED/hello.txt type=file



#### print the mount point of your BB allocation on the compute nodes
echo "*** DW path is:"
echo $DW_JOB_STRIPED

#### Write a file on the BB
echo "*** saying hello...."
echo "Hello! I'm on the Burst Buffer!" > $DW_JOB_STRIPED/hello.txt
echo "*** ls -larth $DW_JOB_STRIPED/"
ls -larth $DW_JOB_STRIPED/
echo "*** cat $DW_JOB_STRIPED/hello.txt"
cat $DW_JOB_STRIPED/hello.txt
```

"atpesctrain"

haswell

- Look at "stage_out.sh". Edit it to point to your OWN scratch directory
- Submit the job
- Check your scratch directory - did "hello.txt" stage out as expected?

Persistent reservations

- **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
```

Persistent reservations

- **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

```
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
```

Persistent reservations

- **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
```

Persistent reservations

- **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
```

Create a persistent reservation

- Look at "create_persistent.sh"
- Edit it to rename the persistent reservation!
- Submit it, then use "scontrol show burst" to check it's been created.

```
#!/bin/bash

#### Which partition?
#SBATCH -p regular

#### name of the training reservation
#SBATCH --reservation="csgftrain"

#### How many nodes?
#SBATCH -N 1

#### How long to run the job? This doesn't need to be long as we're not actuall
y executing anything on the compute node.
#SBATCH -t 00:1:00

#### Our reservation is for KNL nodes
#SBATCH -C knl

#### Name the job
#SBATCH -J "job_create_persistent"

#### Set the output file name
#SBATCH -o "job_create_persistent.log"


#### Create a persistent reservation (PR). Choose what size and name you want t
o give it. Note that you MUST change this name! Every PR name needs to be uniqu
e - if there already exists a PR of this name the job will fail.
#BB create_persistent name=my_persistent_reservation capacity=300GB access_mode
=striped type=scratch
```

"atpesctrain"

haswell

Use a persistent reservation

```
#!/bin/bash

#### Which partition?
#SBATCH -p regular

#### name of the training reservation
#SBATCH --reservation="csgftrain"

#### How many nodes?
#SBATCH -N 1

#### How long to run the job?
#SBATCH -t 00:01:00

#### Our reservation is for KNL nodes
#SBATCH -C knl

#### Name the job
#SBATCH -J "job_use_persistent"

#### Set the output file name
#SBATCH -o "job_use_persistent.log"


#### Specify which persistent reservation you will access. Remember to use the cor
rect reservation name!
#DW persistentdw name=my_persistent_reservation

#### Stage in a directory. Remember to change this directory to your training acco
unt scratch directory! You can also stage_in a file by specifying "type=file".
#DW stage_in source=/global/cscratch1/sd/djbard/train/csgf-hpc-day/IntroToBB/data
 destination=$DW_JOB_STRIPED/data  type=directory



#### print the mount point of your BB allocation on the compute nodes
echo "*** The path to my BB allocation is:"
echo $DW_PERSISTENT_STRIPED_my_persistent_reservation

#### Check what's been staged in to your allocation
echo "*** What's on my persistent reservation?:"

echo "*** ls -lrt  $DW_PERSISTENT_STRIPED_my_persistent_reservation "
ls -lrt  $DW_PERSISTENT_STRIPED_my_persistent_reservation


echo "*** ls -lrt  $DW_PERSISTENT_STRIPED_my_persistent_reservation "
ls -lrt  $DW_PERSISTENT_STRIPED_my_persistent_reservation/data
```

"atpesctrain"

haswell

- **Look at "use_persistent.sh"**
- **Change:**
  - The PR name,
  - The stage_in path
  - The variable $DW_PERSISTENT_STRIPED_name

Destroy a persistent reservation

- **Look at "destroy_persistent.sh"**
- **Change the name to your own PR name.**
- **Submit it, then use "scontrol show burst" to check it's been destroyed.**
  - If you get the name wrong, you'll get NO error messages.

```
#!/bin/bash

#### Which partition?
#SBATCH -p regular

#### name of the training reservation
#SBATCH --reservation="csgftrain"

#### How many nodes?
#SBATCH -N 1

#### How long to run the job?
#SBATCH -t 00:1:00

#### Our reservation is for KNL nodes
#SBATCH -C knl

#### Name the job
#SBATCH -J "job_destroy_persistent"

#### Set the output file name
#SBATCH -o "job_destroy_persistent.log"


#### Destroy the persistent reservation. All data on the reservation will be lost.
# Remember to use the correct reservation name!
#BB destroy_persistent name=my_persistent_reservation
```

"atpesctrain"

haswell

Using the Burst Buffer interactively

- **Look at "interactive_scratch.conf":**

```
#DW jobdw capacity=100GB access_mode=striped type=scratch
```

- **Request an interactive session using "salloc":**

```
salloc -N 1 -t 5 -C haswell --reservation="atpesctrain"
--bbf="interactive_scratch.conf"
```

```
salloc -N 1 -t 5 -C knl --qos=interactive
--bbf="interactive_scratch.conf"
```

- **Play around with the BB interactively!**
  - Try running IOR using the executable in your examples directory:

```
srun -n 8 ./IOR -a MPIIO -g -t 10MiB -b 100MiB -o
$DW_JOB_STRIPED/IOR_file
```

- **Also "interactive_persistent.conf" for a PR.**

Using the Burst Buffer interactively

```
djbard@nid00578:~/BB/CUG17> ./IOR -a MPIIO -g -t 20MiB -b 100MiB -o $DW_JOB_STRIPED/IOR_file
IOR-3.0.1: MPI Coordinated Test of Parallel I/O

Began: Mon May  8 10:03:07 2017
Command line used: ./IOR -a MPIIO -g -t 20MiB -b 100MiB -o /var/opt/cray/dws/mounts/batch/4882533_striped_scratch//IOR_file
Machine: Linux nid00578

Test 0 started: Mon May  8 10:03:07 2017
Summary:
        api                = MPIIO (version=3, subversion=1)
        test filename      = /var/opt/cray/dws/mounts/batch/4882533_striped_scratch//IOR_file
        access             = single-shared-file
        ordering in a file = sequential offsets
        ordering inter file= no tasks offsets
        clients            = 1 (1 per node)
        repetitions        = 1
        xfersize           = 20 MiB
        blocksize          = 100 MiB
        aggregate filesize = 100 MiB

access    bw(MiB/s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)   close(s)   total(s)   iter
------    ---------  ---------- ---------  --------   --------   --------   --------   ----
MPIIO WARNING: DVS stripe width of 32 was requested but DVS set it to 2
See MPICH_MPIIO_DVS_MAXNODES in the intro_mpi man page.
MPIIO WARNING: DVS stripe width of 32 was requested but DVS set it to 2
See MPICH_MPIIO_DVS_MAXNODES in the intro_mpi man page.
write     1385.56    102400     20480      0.002353   0.069377   0.000442   0.072173   0
MPIIO WARNING: DVS stripe width of 32 was requested but DVS set it to 2
See MPICH_MPIIO_DVS_MAXNODES in the intro_mpi man page.
MPIIO WARNING: DVS stripe width of 32 was requested but DVS set it to 2
See MPICH_MPIIO_DVS_MAXNODES in the intro_mpi man page.
read      2458.82    102400     20480      0.000481   0.039974   0.000215   0.040670   0
remove    -          -          -          -          -          -          0.000232   0

Max Write: 1385.56 MiB/sec (1452.86 MB/sec)
Max Read:  2458.82 MiB/sec (2578.26 MB/sec)

Summary of all tests:
Operation   Max(MiB)   Min(MiB)  Mean(MiB)     StdDev    Mean(s) Test# #Tasks tPN reps fPP reord reordoff reordrand seed segcnt blksiz xsize aggsize API
write        1385.56    1385.56    1385.56       0.00    0.07217 0 1 1 1 0 0 1 0 0 1 104857600 20971520 104857600 MPIIO 0
read         2458.82    2458.82    2458.82       0.00    0.04067 0 1 1 1 0 0 1 0 0 1 104857600 20971520 104857600 MPIIO 0

Finished: Mon May  8 10:03:07 2017
```

Using dwstat

```
#!/bin/bash

#### Which partition?
#SBATCH -p regular

#### name of the training reservation
#SBATCH --reservation="csgftrain"

#### How many nodes?
#SBATCH -N 1

#### How long to run the job?
#SBATCH -t 00:5:00

#### Our reservation is for KNL nodes
#SBATCH -C knl

#### Name the job
#SBATCH -J "job_dwstat"

#### Set the output file name
#SBATCH -o "job_dwstat.log"


#### Request a 200GB scratch allocation, striped over BB nodes
#DW jobdw capacity=200GB access_mode=striped type=scratch pool=wlm_pool



#### print the mount point of your BB allocation on the compute nodes
echo "*** DW path is:"
echo $DW_JOB_STRIPED


#### find out which DW nodes your allocation is striped over
echo "*** what nodes are my allocation striped over?"
module load dws

sessID=$(dwstat sessions | grep $SLURM_JOBID | awk '{print $1}')
echo "session ID is: "${sessID}
instID=$(dwstat instances | grep $sessID | awk '{print $1}')
echo "instance ID is: "${instID}
echo "fragments list:"
echo "frag state instID capacity node"
dwstat fragments | grep ${instID}
```

"atpesctrain"

haswell

- **Dwstat is a useful tool to learn exactly what's going on with the BB allocations.**
  - Only accessible from compute nodes.
- **Look at "dwstat.sh" for how to find what DW nodes *your* BB allocation is striped over.**

Running IOR on the BB

- **Take a look at "run_IOR.sh"**
  - This script sets all the options to run IOR. You can edit it to run on a scratch BB allocation, or on scratch, or in your home directory.

```
#!/bin/ksh

#### Which partition?
#SBATCH -p debug

#### name of the training reservation
##SBATCH --reservation="CUG2B"

#### How many nodes? Let's try driving IOR from 4 nodes. The BB allocation will be mounted on all of the
m.
#SBATCH -N 4

#### How long to run the job? IOR is fast, so we only need 5 minutes.
#SBATCH -t 00:5:00

#### Our reservation is for Haswell nodes
#SBATCH -C haswell

#### Name the job
#SBATCH -J "job_run_IOR"


#### Request a 200GB scratch allocation. Note that you will need to request enough space on the BB to ac
comodate your IOR test - this will be (# nodes * # ranks per node * block size)
#DW jobdw capacity=200GB access_mode=striped type=scratch
```

"atpesctrain"

Running IOR on the BB

```
#### Set the current directory
DIR=$PWD

#### Tell IOR that you'll be running on the Burst Buffer allocation. Change this if you want to write to
 another location (for example, your home or scratch directory).
TESTDIR=$DW_JOB_STRIPED

#### Define the transfer size for IOR
TRANSFER_SIZE=1MiB

#### Define the block size for IOR
BLOCK_SIZE=100MiB

#### How many MPI ranks do we use per node?
RANKS_PER_NODE=32

#### How many nodes did we ask for?
NODES=$SLURM_JOB_NUM_NODES

#### Define how many total MPI ranks we will run. This is the nubmer of nodes times the number of ranks
per node we spcified earlier.
RANKS=$(( $NODES*$RANKS_PER_NODE ))

#### IOR options.
OPTIONS="-g -t $TRANSFER_SIZE -b $BLOCK_SIZE -o ${TESTDIR}/IOR_file -v"
```

```
#### Run IOR to test for all MPI ranks writing to a single shared file.
#### First define the name of the output file.
BASE=$DIR/mssf_${TRANSFER_SIZE}_${BLOCK_SIZE}_${NODES}nodes_${RANKS}files_wlmpool${DWN}_${SLURM_JOBID}
print ${BASE}
print ${RANKS}
print "   ", ${OPTIONS}
#### Now run IOR.
time srun -n ${RANKS} --ntasks-per-node=${RANKS_PER_NODE} ./IOR -a MPIIO ${OPTIONS} < /dev/null >> ${BAS
E}.IOR


#### Run IOR to test for all MPI ranks writing to a single file per process
#### First define the name of the output file
BASE=$DIR/pfpp_${TRANSFER_SIZE}_${BLOCK_SIZE}_${NODES}nodes_${RANKS}files_wlmpool${DWN}_${SLURM_JOBID}
print ${BASE}
#### Now run IOR
time srun -n ${RANKS} --ntasks-per-node=${RANKS_PER_NODE} ./IOR -a POSIX -F -e ${OPTIONS} < /dev/null >>
 ${BASE}.IOR
```