Burst Buffer *Hands on Tutorial*

ATPESC 2018

Jialin Liu

Data Analytics Service Group

National Energy Research Scientific Computing Center

Q Center, St. Charles, IL (USA)
July 29 – August 10, 2018

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 and KNL 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.

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 cscratch (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/io.git
- cd ATPESC-IO-day/IntroToBB/
- mkdir data
- cp /global/cscratch1/sd/jialin/atpesc18.txt data/

Run the first simple script!

- Go to the directory ATPESC-IO-day/IntroToBB/and look at the example script “scratch.sh”:

  - #SBATCH -C haswell
  - #SBATCH --reservation=” atpesc18-haswell”

  or

  - #SBATCH –C knl
  - #SBATCH –reservation=“atpesc18-knl”

```
#!/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
```

Run the first simple script!

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

```
jialin@cori08: squeue -u jialin
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          13980906   regular job scra   jialin PD       0:00      1 (BurstBufferResources)
```

View Burst Buffer status

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

```
jialin@cori08: scontrol show burst
Name=cray DefaultPool=wlm_pool Granularity=20624MiB TotalSpace=1669255GiB FreeSpace=1354171840MiB UsedSpace=30317
  AltPoolName[0]=dev_pool Granularity=20624MiB TotalSpace=47693GiB FreeSpace=10312GiB 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:
    Name=HDF5LHE CreateTime=2018-05-10T15:12:41 Pool=(null) Size=41248MiB State=allocated UserID=hschulz(76404)
    Name=c20c CreateTime=2018-05-15T11:03:27 Pool=(null) Size=4784768MiB State=allocated UserID=ndk(31805)
    JobID=13970402 CreateTime=2018-08-02T13:12:43 Pool=wlm_pool Size=1289GiB State=staged-in UserID=khl7265(74109
    Name=NCBI_DB2 CreateTime=2018-05-09T15:40:21 Pool=(null) Size=3815440MiB State=allocated UserID=syao(30821)
    JobID=13966415 CreateTime=2018-08-02T19:58:32 Pool=wlm_pool Size=4784768MiB State=staged-in UserID=fbench(420
    Name=no_WT_1999 CreateTime=2018-08-02T13:12:37 Pool=wlm_pool Size=2392384MiB State=allocated UserID=tshep(731
    Name=WT_1999 CreateTime=2018-07-31T13:33:24 Pool=wlm_pool Size=2392384MiB State=allocated UserID=tshep(73143)
    Name=no_WT_1985 CreateTime=2018-07-31T13:33:24 Pool=wlm_pool Size=2392384MiB State=allocated UserID=tshep(731
    Name=WT_1985 CreateTime=2018-07-26T10:14:03 Pool=wlm_pool Size=2578000MiB State=allocated UserID=tshep(73143)
    Name=dev_scratch CreateTime=2018-05-08T23:24:28 Pool=(null) Size=37381GiB State=allocated UserID=dpaul(15448)
    Name=EQSIM CreateTime=2018-06-04T11:52:45 Pool=(null) Size=51560GiB State=allocated UserID=andersp(72257)
    JobID=13923486 CreateTime=2018-07-29T18:37:58 Pool=wlm_pool Size=15261760MiB State=staged-in UserID=detar(438
    JobID=13921374 CreateTime=2018-07-29T18:37:58 Pool=wlm_pool Size=15261760MiB State=staged-in UserID=detar(438
    JobID=13912354 CreateTime=2018-07-28T21:42:21 Pool=wlm_pool Size=15261760MiB State=staged-in UserID=detar(438
```

Stage in data to a scratch allocation

```
#!/bin/bash

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

#### name of the training reservation
#SBATCH --reservation="atpesc18-haswell"

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

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

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

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

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

#### Request a 20GB scratch allocation, striped over BB nodes
#DW jobdw capacity=20GB access_mode=striped type=scratch pool=wlm_pool

#### Stage in a directory. Remember to change this directory to your tr
n also stage_in a file by specifying "type=file".
#DW stage_in source=/global/cscratch1/sd/jialin/io/ATPESC-IO-day/IntroT
ata2  type=directory
```

- 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


```
#### name of the training reservation
#SBATCH --reservation="atpesc18-haswell"

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

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

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

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

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

#### Request a 20GB scratch allocation, striped over BB nodes
#DW jobdw capacity=20GB access_mode=striped type=scratch pool=wlm_pool

#### Stage a file out of the BB onto scratch.
#DW stage_out destination=/global/cscratch1/sd/jialin/io/ATPESC-IO-day/I
STRIPED/hello.txt type=file
```

- 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

```
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

```
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

```
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
```

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.

scontrol show burst | grep 'jialin'

```
#!/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
```

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
```

- 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.

```
#!/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
```

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="atpesc18-haswell" --bbf=”interactive_scratch.conf”
  - salloc -N 1 -t 5 -C haswell --qos=interactive --bbf=”interactive_scratch.conf”
- Play around with the BB interactively!
  - srun -n 8 ./IOR -a MPIIO -g -t 10MiB -b 1000MiB -o $DW_JOB_STRIPED/IOR_file

Lustre, and Darshan Logs

- Compare with Lustre
  - `cd ATPESC_IO_day/IntroToBB/`
  - `mkdir ost5`
  - `lfs setstripe –c 5 –S 20m ost5`
  - `module load darshan`
  - `srun –n 8 ./IOR –a MPIIO –g –t 10MiB –b 1000MiB –o ost5/IOR_file`

- Grab Darshan logs
  - `cd /global/cscratch1/sd/darshanlogs/2018/8/2`
  - `ls -ltr jialin*`

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}
```

- 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 H5py on the BB

- Take a look at “parallel_write.py”
  - salloc -N 1 -q interactive -C haswell --bbf="interactive_scratch.conf" -t 20
- Use collective I/O to write on Burst Buffer/Lustre
  - srun -n 32 python parallel_write.py 1 $DW_JOB_STRIPED/bb1.h5
  - srun -n 32 python parallel_write.py 1 bb1.h5
- Use independent I/O to write on Burst Buffer/Lustre
  - srun -n 32 python parallel_write.py 0 $DW_JOB_STRIPED/bb1.h5
  - srun -n 32 python parallel_write.py 0 bb1.h5

Running IOR on the BB

- **Take a look at “run_IOR.sh”.**

Thank you!

consult@nersc.gov

jalnliu@lbl.gov

Thanks Debbie Bard, Wahid Bhimji, etc, for slides materials.