Using the Burst Buffer on Cori

Tony Wildish
Dan Udwary

June 23, 2017
Full documentation

• On the NERSC website, at
  – https://www.nersc.gov/users/computational-systems/cori/burst-buffer/
What is a Burst buffer?

• Unlike Genepool, Cori batch nodes have no local disk
• Global filesystems can be slow for some workloads
  – Applications with large I/O requirements
  – Random-access I/O, instead of streaming
  – Problem magnifies with scale
    • Many jobs reading same reference DBs etc

• Burst buffer is fast disk you can use in batch, on Cori
  – Physically part of Cori, close to the compute nodes
  – Much faster (SSDs instead of spinning disks)
  – Smaller volume (10’s – 100’s GB, not TB)
  – User-configurable properties
    • Lifetime, performance
HPC memory hierarchy

Past

Fast, expensive

Slow, cheap

CPU

L1/2/3 cache

Memory (DRAM)

Storage (HDD)

/global/homes, projectb, scratch
HPC memory hierarchy

Past

- CPU
- Memory (DRAM)
- Storage (HDD)

Present

- CPU
- Near Memory (HBM)
- Far Memory (DRAM)
- Near Storage (SSD)
- Far Storage (HDD)
HPC memory hierarchy

- CPU
- Near Memory (HBM)
- Far Memory (DRAM)
- Near Storage (SSD)
- Far Storage (HDD)

Burst Buffer
Accessing the Burst buffer

• Cori only, not available on Edison or Genepool
  – ~1.8 PB available, spread over 288 nodes
  – Accessible to both Haswell and KNL partitions
  – Batch nodes only, not available on login nodes

• Create/delete Burst buffer reservations
  – Use #DW or #BB directives in your batch jobs
    • At top of script, just below any #SBATCH directives
  – Granularity: ‘pool’ size fixed, but can ask for any capacity you want
    • 80 GB default granularity
    • 20 GB – for really intense I/O, add ‘pool=sm_pool’ to commands

• View existing reservations
  – ‘scontrol show burst | grep $USER’
Burst buffer characteristics

- **Per-job reservation – scratch space**
  - Lasts as long as the batch job it’s created for
  - Only visible to that batch job
  - Used for:
    - staging files in/out of job
    - fast scratch space
    - Checkpoints

- **Persistent reservation – sharing data**
  - Can be shared among jobs
  - Lifetime controlled by person who creates it
  - Used for
    - staging files in/out of jobs
    - sharing data (reference files)
    - coupling job workflows
  - Not for long-term storage of data!
    - No guarantees, instance may disappear at any time
Using Burst buffer as scratch

```bash
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -C haswell
#SBATCH -t 00:15:00

#DW jobdw capacity=200GB access_mode=striped type=scratch
```

Currently, the only option

Required keywords

Size

**striped**: all jobs in multi-node job share the same space

**private**: each node in multi-node job gets its own space

Currently, the only option

Required keywords

Size

**striped**: all jobs in multi-node job share the same space

**private**: each node in multi-node job gets its own space
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -C haswell
#SBATCH -t 00:15:00

#DW jobdw capacity=200GB access_mode=striped type=scratch

cd $DW_JOB_STRIPED

cp $HOME/my-file.dat .

./do-something --with my-file.dat --output my-output.dat

cp my-output.dat $HOME/

‘#DW’ only gets you capacity it’s up to you to actually use it!

$DW_* environment variables point to the space on disk

$DW_JOB_PRIVATE if mode=private

# Save your output, or lose it!
#!/bin/bash

#SBATCH -p debug
#SBATCH -N 1
#SBATCH -C haswell
#SBATCH -t 00:15:00

#DW jobdw capacity=20GB access_mode=striped type=scratch

#DW stage_in source=/global/cscratch1/sd/username/path/to/filename
destination=${DW_JOB_STRIPED}/filename type=file

#DW stage_out source=${DW_JOB_STRIPED}/dirname
destination=/global/cscratch1/sd/username/path/to/dirname type=directory

- Staging in/out happens before/after the job runs
  - #DW directives at top of script, not inline
- Not counted against your batch-job time
- Can’t use environment variables – why not?

Full path, no environment variables!
Stage in/out files or directories
Staging data to the Burst buffer

- Staging in/out happens before/after the job runs
  - #DW directives at top of script, not inline

```bash
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -C haswell
#SBATCH -t 00:15:00

#DW jobdw capacity=20GB access_mode=striped type=scratch

#DW stage_in source=/global/cscratch1/sd/username/path/to/filename
destination=${DW_JOB_STRIPED}/filename type=file

#DW stage_out source=${DW_JOB_STRIPED}/dirname
destination=/global/cscratch1/sd/username/path/to/dirname type=directory
```

- Not counted against your batch-job time
- Can’t use environment variables
  - Your job hasn’t ‘logged in’ yet!

Full path, no environment variables!

Stage in/out files or directories
Persistent Burst buffer reservations

• Use #BB directives to create/delete, #DW to use it
  – Create batch jobs to create/delete the reservation
  – No lifetime guarantees, always back up valuable data!
  
    – #BB create_persistent name=TW_BB capacity=80GB access=striped type=scratch
      • Create my persistent reservation
      • $DW_PERSISTENT_STRIPED_TW_BB points to directory

  – #DW persistentdw name=TW_BB
    • Use it in subsequent batch jobs

  – #BB destroy_persistent name=TW_BB
    • It’s your responsibility to destroy the reservation yourself
• Want an interactive session for debugging, with Burst buffer? You can do that!
  – Create a file with the same #DW or #BB directives you’d put in a batch script
  – Use the --bbf flag to salloc to create the burstbuffer allocation
    • > salloc --qos=interactive -C haswell -t 01:00:00 --bbf="mybbf.conf"
  – N.B. the quotes around the filename are obligatory!
  – Can create temporary reservations, for lifetime of interactive session, or create/use/delete persistent reservations
Best practices

• Experiment, to see if using the Burst buffer helps your application
  – Not everything will benefit, try it and see
  – Don’t forget to try the data-staging in/out too!

• Prefer per-job scratch to persistent reservations
  – Easier to manage

• Choose unique names for persistent reservations
  – Make them meaningful

• Clean up persistent reservations when done
  – Build it into your workflow
Environment variables

- Only one DW_* environment variable will be set at a time
  - DW_JOB_STRIPED,
    DW_JOB_PRIVATE,
    or DW_PERSISTENT_STRIPED_*
  - But which? Don’t want to keep changing your batch scripts just because you changed Burst buffer reservation!

```
v=`env | egrep ^DW_`
variable=`echo $v | awk -F '{ print $1 }'`
value=`echo $v | awk -F '{ print $2 }'`
echo “I found a variable called $variable with value $value”
```
Exercises!

1. Create a config file to specify a 40 GB persistent reservation
   1. Use salloc to get an interactive session and create this reservation
   2. Copy some files to the burst buffer directory
   3. Terminate your interactive batch session

2. Use the scontrol command to list information about the persistent reservation you created

3. Create a batch job to list files on the persistent reservation
   1. Submit it, wait until it runs

4. Create a config file to destroy the persistent reservation, ‘execute’ it with salloc