Run batch jobs

Batch resources available to the SLAC Analysis Computing Facility users

SLAC uses the LSF (Load Sharing Facility) batch system. LSF replica your current environment setup when submitting jobs. This includes your current working directory and any Unix environment variable setups. All batch nodes available to the ATLAS users have

- CVMFS (/cvmfs/atlas.cern.ch, /cvmfs/sft.cern.cn, etc.)
- Access to SLAC networked storages (/nfs, /gpfs, etc.)
- Large /scratch space for temporary use (please clean up your files from there after your job finishes).
- Outbound network connectivity.
- Singularity container on all batch nodes that run CentOS 7 operation system.

The following are examples of using LSF:

Submit a job.

```
$ cat myjob.sh
#!/bin/sh
#BSUB -W180
pwd
echo "hello world"

$ bsub < myjob.sh
Job <96917> is submitted to default queue <medium>.
```

This will submit a job to LSF. The "pwd" command will print out the job's working directory, which should be the same directory where this job is submitted. The **#BSUB -W180** directive tells LSF that the job's maximum run time limit (wall clock time) is 180 minutes. After that the job will be killed. If **#BSUB -Wnnn** isn't specified, your job get the default, which is 30 minutes. You can but don't have to specify a batch queue. If no queue is given at the submission time, LSF will choose one for your job according to job's run time limit.

Check job status.

Check status of all my job

```
$ bjobs
```

Of a specific job

```
$ bjobs -l <job ID>
```

To kill a job.

```
$ bkill <job ID>
```

Refine batch submission

SLAC batch resources consist of several generation of hardwares. They are listed at the the shakeholder's priority page. Some of the batch nodes run RHEL 6 operation system, while others run CentOS 7 operation system. **Singularity container technology is available on the CentOS 7 batch nodes.**

- To run your job on a RHEL 6 batch node only, use: bsub -R "select[rhel6]" ...
- To run your job on a CentOS 7 batch node only, use: bsub -R "select[centos7]" ...

By default, your job will ask for 1 CPU core (one batch slot) and will allow maximum of 4GB of RAM. If your job exceed the RAM limit, your job will be killed. If you need more CPU core or RAM, you can ask for more than one CPU cores when submitting your jobs. For example:

- bsub -n 4 -R 'span[hosts=1]' ... will submit a job requesting 4 core and (4x 4GB =) 16GB RAM, and allocate all 4 cores on one machine (This is what "span[hosts=1]" is for)

Of course, the more resource you ask, the harder to schedule the jobs, and hence the pending time will be longer.

Below is an long example of resource selection in LSF, for you to pick and choose from:

- -R "select[! preempt && rhel60 & cvmfs && inet && bullet && hname != bullet0030] rusage[scratch=5.0:duration=1440:decay=1, mem=2000:decay=0] span[hosts=1]"

It requests the job be dispatched to a machine where

1. Your job won't be preempted by someone else's higher priority job
2. The machine run RHEL6 operating system (we have "rhel60" and "centos7")
3. The machine should have CVMFS,
4. and outbound internet connection (the "inet" key word above)
5. The machine should be part of the "bullet" cluster (Other clusters we have: fell, hequ, dole, kiso, deft and bubble, all run "rhel60" except the last two, which run "centos7"), and not on host bullet0030
6. Reserve 5GB of free space under /scratch, and you job will reserve it for 1440 minutes, and the reserved amount will decay linearly from 100% to 0 during this period.

7. Reserve 2000MB of RAM, no decay (the default)
8. span[hosts=1] means if you request more than one batch slots (the -n option above), schedule all of them on one machine.

Note:

- a. For 6 or 7 to work, the machine should have that amount of resource available at the time the job is dispatched.
- b. CVMFS cache is usually stored under /scratch/cvmfs2_cache. (This is a way to make sure that there are free space for CVMFS cache so your job won't get error when accessing CVMFS)
- c. Most SLAC batch users doesn't use 6 or 7, even if they do, they can use more (because the amount specified in 6 or 7 are "reserved", not maximum). So after your job started at a machine, something bad can still happen (run out of memory, /scratch etc.) due to the activities on that machine.

Please refer to the LSF document to get familiar with the basic usage of LSF.

Batch resources available to the SLAC ATLAS Department users

The above information describes running batch jobs on the "general fair share" queues, which is available to everyone who has a SLAC unix account. ATLAS users have a relatively higher priority on those resource according to the shakeholder's priority page. The following info is for SLAC ATLAS Department users only.

Private batch resource owned by the SLAC ATLAS Department

In addition to the "general fair share" resource, SLAC ATLAS Department users can run jobs in a dedicated LSF queue "atlas-t3". The corresponding batch cluster runs RHEL6. The following command show who can use the dedicate LSF resource, and who can add/remove users to the dedicated resource.

```
$ ypgroup exam -group atlas-t3
Group 'atlas-t3':
        GID:     3104
        Comment:
        Last modified at Oct 14 00:22:52 2015 by yangw
        Owners:  sch, sudong, young, zengq
        Members: acukierm, bpn7, laurenat, makagan, osgatlas01, rubbo, zengq, zihaoj

        This is a secondary group.
```

The above shows the UNIX group "atlas-t3". People in the "Owners" line can add/remove members of this group. People in the "Member" line can run jobs in the dedicate queue. (Owners are not members).

The following is an example job script for users to submit jobs to the atlas-t3 queue:

```
$ cat job-script.sh
#!/bin/sh
# run in LSF queue atlas-t3 and run up to 120 minutes (wall time)
#BSUB -q atlas-t3
#BSUB -W 120
#BSUB -R "select[rhel60 && cvmfs && inet] rusage[scratch=5.0, mem=1000:decay=0]"

# create a unique working directory on batch node's /scratch space
myworkdir="/scratch/`name -n`$$"
mkdir $myworkdir
cd $myworkdir

# run payload
task1 < input_of_task1 > output_of_task1 2>&1 &
task2 < input_of_task2 > output_of_task2 2>&1 &
wait  # wait for the tasks to finish

# save the output to storage, use either "cp" to copy to NFS spaces, or "xrdcp" to copy to the xrootd spaces
cp myoutput_file /nfs/slac/g/atlas/u02/myoutput_file
xrdcp myoutput_file root://atlprf01:11094//atlas/local/myoutput_file

# clean up
cd /scratch
rm -rf $myworkdir

$ bsub < job-script.sh  # submit the job
```

In the above script, #BSUB... lines are more than just comments in this unix Bourne Shell script. They are also directives for the LSF "bsub" command. The first two #BSUB directives tell LSF that the batch queue is "atlas-t3" and the wall time limit is 120 minutes. Please always specify a wall time. Otherwise, your jobs will be killed after 30 minutes (wall time). The third #BSUB directive is optional. It tells LSF that the job wants to run on RHEL6 platform ("rhel60") with cvmfs ("cvmfs") and outbound internet connection ("inet"), and that the job needs up to 5GB of space under /scratch, 1000MB of RAM (these are advises to the LSF scheduler, not caps or limits).

With the two "&" at the end of the task lines (task1 and task2), the two tasks run simultaneously. If you want them to run sequentially, remove the two "&".

Batch resource for general users

If you remove the above "#BSUB -q atlas-t3" directive, your job will be submitted to a general "fair share" queue. This is one of a group of batch queues (short, medium, long and xlong). LSF will choose the queue for you based on the wall time specified. The general "fair share" queues are a much larger pool of resource. They are available to everyone at SLAC. They are heavily loaded most of the time.

Tips:

Kill all your jobs

If you made a mistake and want to kill all your batch jobs, use the following command

```
bjobs | grep -v JOBID | awk '{print "bkill", $1}' | sh
```

Check the load of the cluster

To check how many jobs are currently running and pending in atlas-t3 queue:

```
bqueue atlas-t3
```

You can also check the Ganglia monitoring page to see the load on the cluster.

Use the batch slots efficiently

If you find that your batch job is not resource efficient (CPU/wall time ratio is low AND memory (RA(M) usage is lower (less than 1.5GB, use "bjobs -l job_id" to find out)), try putting two tasks in one batch job (in the above script) and let them run in parallel. **But be careful**. Don't run too many tasks in a batch job in parallel because it is possible to overwhelm some components in the tightly coupled computing environment.

Use the following command to find out if your jobs are reasonable efficient:

```
$ export LSB_BJOBS_FORMAT="id:7 user:11 stat:5 exec_host:10 submit_time:13 start_time:13 max_mem:12 cpu_used
run_time"
$ bjobs | grep DONE
JOBID   USER        STAT  EXEC_HOST  SUBMIT_TIME   START_TIME    MAX_MEM      CPU_USED RUN_TIME
867402  bpn7        DONE  atlprf11   Nov 19 16:18  Nov 20 02:09  654 Mbytes   462.3 second(s) 956 second(s)
...
```

The job used 462 CPU seconds during its 956 seconds run time. The maximum RAM it used was 654MB. The batch nodes are configured to accommodate 2GB RAM per CPU (on average). The CPU time and RAM that are not used by the job is wasted. So in the above job script, one can put two of such tasks in the script and let them run in parallel.