Lilac Cluster Guide

The Lilac cluster is now available to the general MSKCC research community. The system is running the CentOS 7 Linux operating system along with the LSF scheduler from IBM. This sits on top of a large GPFS storage cluster for high performance data access.

Slides from Nov 2019 UG: UG_10_2019.cp1.pptx

Slides from Oct 2018 UG:Lilac_UG_102018.pdf

Slides from Sep 2017 UG: UG_09202017_final .pptx

Slides from June 22 2017 UG: LSF_UG_1.potx

Queues

The LILAC cluster uses LSF (Load Sharing Facility) 10.1 FP8 from IBM to schedule jobs. The default LSF queue, 'cpuqueue', includes subset LILAC compute nodes and should be used for CPU jobs only. The gpuqueue queue should be used for GPU jobs only.

The cluster currently has ten partitions: 122 nodes, ~6,840 cores, ~475 GPUs, ~50 TB of RAM, 2.6 PB usable computational storage /data and 1.7PB usable warm storage /warm.

Queue name	Hosts	Max memory per host	Max number of CPU-threads per host	Default (thread/mem/walltme)	Max Walltime	Access
cpuqueue(default)	ls* lg* lp* lt* lu* lx* ly* lw*	466GB 720GB 1.35TB	68 76 76	1/2GB/1hour	7 days	All users
gpuqueue	ls* lg* lp* lt* lu* lv* lx* ly* lw* ld*	489GB 758GB 1.4TB 489GB	72 80 80 80	1/2GB/1hour	7 days	All users test users
long	lg01-06 ld01-05	489GB	72 (lg hosts) 80 (ld hosts)	1/2GB/1hour	30 days	ACL

*note. Some subsets of compute servers are purchased by partner PIs or individual departments. These systems are placed into the cluster under special queue configurations that enable prioritization for the contributing group. All users can run jobs with Walltime < 6 hours on lg-gpu, lp-gpu, ld01-05 and ly-gpu. All users benefit from such systems as they allow jobs to run on them while they are idle or under low utilization. The rules for group owned nodes within the LSF queuing system

The LILAC cluster includes different types of nodes. Currently we have following host groups:

ls-gpu: ls01-ls18 (4xGeForce GTX 1080)

lt-gpu: lt01-lt08 (4xGeForce GTX 1080Ti)

lg-gpu: lg01-lg06 (lg01-03 4xTITANX; lg04-06 4xPascal)

lp-gpu: lp01-35 (4xGeForce GTX 1080Ti)

ld-gpu: ld01-07 (8xTeslaV100-SXM2-16GB)

lv-gpu: lv01 (4xTeslaV100 PCIE-16GB)

lu-gpu: lu01-lu10 (4xGeForce RTX 2080)

lx-gpu: lx01-lx13 (4xGeForce RTX 2080 Ti)

ly-gpu: ly01-09 (4xQuadro RTX 6000)

lw-gpu: lw01-02 (8xGPUs GeForce RTX 2080 Ti)

Job resource control enforcement in LSF with cgroups

LSF 10.1 makes use of Linux control groups (cgroups) to limit the CPU cores, number of GPUs and memory that a job can use. The goal is to isolate the jobs from each other and prevent them from consuming all the resources on a machine. All LSF job processes are controlled by the Linux cgroup system. Jobs can only access the GPUs which have been assigned to them. If the job processes on a host use more memory than it requested, the job will be terminated by the Linux cgroup memory sub-system.

LSF cluster level resources configuration (Apr 3 2018)

GPUs are consumable resources per host, not per slot. Job can request N CPUs and M GPUs per host, where N>M, N=M and N<M.

Please, check examples in **Requesting Different CPU and GPU Configurations.**

Memory is consumable resource in GB per slot (-n).

Job Submission

LSF supports a variety of job submission techniques. By accurately requesting the resources you need, you can have your jobs execute as quickly as possible on available nodes which can process them.

A job consists of two parts: resource requests and job steps. A resource request consists of number of CPU threads, GPUs, wall time (expected duration), amount of memory per CPU thread, etc. Job steps specify tasks which must be done, software which must be run, etc.

Typically a user submits a job submission script which contains resource requests and job steps to LSF. A submission script is usually a shell (bash) script. The job will advance in the queue until it reaches the top. At this point LSF will allocate the requested cores, GPUs, and memory, and execute it.

Alternately users can submit jobs directly on the command line:

```
bsub -n1 -o %J.stdout -eo %J.stderr -W 4:00  my_executable
```

The `-o` flag without `-e` by itself will automatically combine the two logs into the single file. The `%J` variable is a special variable in the job submission which is replaced with your job ID.

ls-gpu and lt-gpu host partition have /fscratch NVMe based 1.9TB scratch partition for running jobs. This partition /fscratch doesn't clean automatically after the job is finished.

To request the node with /fscratch partition:

```
bsub -n 1 -R "fscratch" ...
```

More information on job submission and control

For more information on the commands to submit and manage jobs, please see the following page: LSF Commands

Simple Submission Script

There are default values for all batch parameters, but it is a good idea to always specify the number of threads, GPUs (if needed), memory per thread, and expected wall time for batch jobs. To minimize time spent waiting in the queue, specify the smallest wall time that will safely allow your jobs to complete.

```
#!/bin/bash
#BSUB -J myjobMPI
#BSUB -n 144
#BSUB -R span[ptile=72]
#BSUB -R rusage[mem=4]
#BSUB -W 04:00
#BSUB -o %J.stdout
#BSUB -eo %J.stderr

cd $LS_SUBCWD
mpirun -np 144 /my/bin/executable < my_data.in
```

Note that the memory requirement (`-R rusage[mem=4]`) is in GB (gigabytes) and is PER CORE (`-n`) rather than per job. A total of 576GB of memory will be allocated for this example job.

Submission Example

Submit a batch script with the bsub command:

```
bsub < myjob.lsf
```

Interactive Jobs

Interactive batch jobs provide interactive access to compute resources, such as for debugging. You can run a batch-interactive using “bsub -Is”.
Here is an example command to create an interactive shell on a compute node:

```
bsub -n 2 -W 2:00 -q gpuqueue -gpu "num=1" -R "span[hosts=1]"  -Is /bin/bash
```

GPU Jobs

LILAC GPUs offer several modes. All GPUs on LILAC are configured in EXCLUSIVE_PROCESS compute mode by default.

EXCLUSIVE_PROCESS means a GPU is assigned to only one process at a time, and individual process threads may submit concurrent work to the GPU. So only one context is allowed per GPU, which is usable from multiple threads simultaneously.

The other GPU compute mode is DEFAULT (or SHARED).

In DEFAULT mode, multiple processes can use the GPU simultaneously. Individual threads in each process may submit work to the GPU simultaneously -- multiple contexts are allowed on a GPU.

Nvidia Multi-Process Service (MPS) enables LILAC to support multiple contexts per GPU.

The MPS runtime architecture is designed to transparently enable cooperative multi-process CUDA applications to utilize Hyper-Q capabilities, which enables multiple CUDA kernels to processes concurrently on a single GPU.

Without MPS, processes sharing a GPU would need to be swapped on and off a GPU, which could only hold one context at a time. The MPS server shares one set of scheduling resources between all its clients, eliminating the overhead of swapping the GPU between contexts or processes.

https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf

The new LSF syntax to submit GPU job:

https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_command_ref/bsub.gpu.1.html#reference_kqd_3kh_xz

The default GPU settings are: "num=1:mode=exclusive_process:mps=no:j_exclusive=yes"

To submit GPU job with default GPU settings:

```
bsub -q gpuqueue -gpu -
```

To submit MPS jobs:

```
bsub -q gpuqueue -n 1 -gpu "num=1:mps=yes"
```

To submit job array of two jobs and request one GPU in Shared mode per job:

```
bsub -q gpuqueue -J "shared[1-2]" -n 1 -gpu "num=1:mode=shared:mps=no:j_exclusive=yes"
```

The LILAC cluster will include different GPU models, as well as non-GPU nodes Currently the LILAC nodes have GTX 1080, GTX TITANX and TITANX (Pascal) GPUs.

To check GPU type per node:

```
lshosts -w -gpu
```

To check available GPU type per node:

```
lsload -w -gpuload
lsload -gpu
```

To submit LSF jobs and request a particular GPU type:

```
bsub -q gpuqueue -n 1  -gpu "num=1:gmodel=NVIDIAQuadroRTX6000"
```

To check current GPUs status on LILAC (host name, total GPUs per host, allocated GPUs per host):

```
bhosts -o "HOST_NAME NGPUS NGPUS_ALLOC"
```

To check current GPUs status and GPUs mode on LILAC:

```
bhosts -o "HOST_NAME NGPUS NGPUS_ALLOC NGPUS_SHARED_AVAIL NGPUS_SHARED_ALLOC NGPUS_EXCL_AVAIL NGPUS_EXCL_ALLOC"
```

To submit an LSF job to a host group or host groups:

```
bsub -m ls-gpu
bsub -m "ls-gpu lt-gpu lp-gpu lg-gpu"
```

For more information on GPU resources, terminology and fine grain GPU control, please see the Lilac GPU Primer.

Requesting Different CPU and GPU Configurations

Warning: Please use -R "span[ptile=number_of_slots_per_host]" to get requested number of slots and requested number of GPUs on the same host, otherwise LSF may try to distribute the job among many hosts

N CPUs and M GPUs on the same node (ls,lp,lt,lg nodes have 72 core-threads and 4 GPUs, ld nodes have 80 core-threads and 8 DGX GPUs)

```
bsub -q gpuqueue -n N -gpu "num=M" -R "span[ptile=N]"
```

One whole 72-thread node and 4 GPUs:

```
bsub -q gpuqueue -n 72 -gpu "num=4" -R "span[ptile=72] rusage[mem=6]"
```

One whole 80-thread node and 8 DGX GPUs:

```
bsub -q gpuqueue -n 80 -gpu "num=8" -R V100 -R "span[ptile=80] rusage[mem=6]"
```

N CPUs and N GPUs, using 1 CPU and 1 GPU per node:

```
bsub -q gpuqueue -n N -gpu "num=1"-R "span[ptile=1]"
```

N CPUs and N GPUs, using 2 CPUs and 2 GPUs per node:

```
bsub -q gpuqueue -n N -gpu "num=2" -R "span[ptile=2]"
```

Parallel Jobs

LSF uses the blaunch framework (aka hydra) to allocate GPUs on execution nodes. The major versions of MPI integrate with blaunch.

```
bsub -q gpuqueue -I -n 4 -gpu "num=1" -R "span[ptile=2] " blaunch 'hostname; echo
CUDA_VISIBLE_DEVICES=$CUDA_VISIBLE_DEVICES; my_executable"
```

Job options and cookbook

For the set of common `bsub` flags and more examples as cookbook, please see: LSF bsub flags