LSF bsub flags

This page is a compilation of the various `bsub` flags you may use in your jobs and what they do. Please consult the Lilac Cluster Guide page for some basic configurations and some very common job submissions.

These are the arguments given to `bsub` or as `#BSUB` directives in your script's header. These flags are case sensitive

Any argument with the < > braces shows a field which you the user replace in your submission.

Specify Wall time

- `-W <hh:mm>`
- `hh` is time in hours, `mm` is time in minuts
- Warning: `<hh:mm:ss>` syntax is invalid, no seconds may be specified (for those converting from PBS scripts)

Specify *total* number of processes you will run

- `-n <Int>`
- This is the total number over all nodes

Use Job ID Number as part of logfile name

- Specify either the `-e` or `-o` flag (error and output, `-o` implies `-e`)
- Use the variable `%J` as part of the file name argument to those flags
- e.g. `#BUSB -o myjob.%J.log` yeilds file `myjob.12345.log` for job ID `12345`

Notes on the resource request flag `-R`:

- The `-R "..."` flag can have multiple entries in side the quotes, e.g. `span`, `select`, `rusage`, etc.
- Multiple `-R` flags can be specified with different entries.
- The entries of `-R` such as `span`, `select`, `rusage`, etc. can support multiple arguments, comma separated.
    - e.g. `-R "rusage[ngpu_excl_p=1,mem=4]"` will set both the `ngpu_excl_p` and `mem` resource requests. Details about what those do are below

Choose how many processes per node to run

- `-n <N*M> -R "span[ptile=<M>]"`
- `N` = number of nodes, `M` = number of processes per node
- Equivalent to `#PBS -l nodes=N:ppn=M`
- If `N % M != 0`, ("N modulo M") then up to `M` processes will be loaded per node, and the last node will have the remainder.
    - e.g. `-n 7 -R "span[ptile=3]"` will have 2 nodes with 3 processes, and 1 node with 1 process

Start a job in Interactive mode with the bash terminal

- `-Is /bin/bash`
    - Note: You can still do all the other settings as well.
    - This signs you into one of the nodes requested, although all the nodes of your job are available to you.

Select a *specific* node to run your job on

- `-n <N> -R "select[hname==<hostname>]"`
- also specify a not *not* to run on*:* `-n <N> -R "select[hname!=<hostname>]"`

Request a certain amount of memory per cpu core:

- `-n <N> -R "rusage[mem=<Q>]"`
- `<Q>` is the amount of memory *per CPU core* (`<N>`) in GB

Request specific node groups

- `-m <group_name>`
- This is separate from the submission *queue* which changes the rules and types of jobs that can be submitted
- use `bmgroup` to see the list of groups

    - As new nodes with different resources are added, this setting will help you get the resources (like GPU model) you want the easiest.
    - The nodes are searched in Alphabetical order until a node is found that matches requirements, so `lg04` comes before `ls01`

List of groups and differences (4/20/17) :

- `ls-gpu`: New Lilac nodes, GTX 1080's

    - Node Name format: `ls##`
- `lila-gpu-hpc`: Currently being used for testing and has limited access, they have GTX 1080's
    - Node Name format: `ls##`
    - These might be experimental nodes for now
- `lg-gpu`: Old HAL `gg**` Fuchs' lab nodes, GTXTITANX. Fuchs lab has priority
    - Node Name format: `lg##`

Change the queue you are submitting to. Different queues treat the nodes differently.

- `-q <queue>`
- `bqueues` shows what queues are available from command line

Available queues (feed them into `-q <queue>`) :

- `general` Default queue, implied so `-q` is not needed.
    - GPUs are in `process exclusive` mode
- `gpushared` Queue being used for testing, limited access
- `test_hpc` Queue used by the HPC staff for testing

Requesting GPU Specific Resources

Recommended Reading: Lilac GPU Primer

It is important to note that the GPUs you request are multiples of your CPU topology. This only makes sense for 1CPU per GPU, which is why all the resource requests are `ngpus_excel_p=1.`

Trivia: When looking at GPU resource requests for LSF documentation from IBM, you will see `ngpus_shared, ngpus_excl_p, and ngpus_excl_t.` These correspond to `shared`, `process exclusive`, and `thread exclusive` mode respectively. Since all Lilac GPUs are in process exclusive mode, the only valid option is `ngpus_excl_p`.

Request 4 GPUs in `process exclusive` mode on 1 node to run on 4 CPUs

- `-n 4 -R "rusage[ngpus_excl_p=1]"`
    - Note that the `-n <N>` acts as a multiple of `ngpus_excl_p=M` to get `N*M = 4*1 = 4` GPUs

Request 6 GPUs in over 2 nodes (there are 4 gpus/node on Lilac) to run 6 total processes

- `-n 6 -R "rusage[ngpus_excl_p=1] span[ptile=3]"`

Request a specific GPU model

- `-R select[gpu_model0=='GeForceGTX1080']`
- Since there is only one `-R` keyword (`select`), no outside double quotes are needed.

Emulate `shared` mode for 2 GPUs by activating the MPS service from Nvidia

- `-n 2 -env "LSB_START_JOB_MPS=Y" -R "rusage[ngpus_excl_p=1]"`
    - Note: You can ONLY launch CUDA Contexts in this mode, no OpenCL.
    - The default for this option is "no" so you have to specifically request it
    - The GPU will still be in exclusive mode, and the process which will be running on it (inspectable through `nvidia-smi` on the node) will be the "`nvidia-cuda-mps-server.`"
    - Up to 16 Contexts per GPU can be created in this mode.

Logic Operators in Resource Requests

LSF can support logical groups (`{ }`) and the "OR" operator (`||`) as part of the `-R` requirements string. This could be helpful for requesting particular sets of packing or hardware requirements.

- `–n <N> -R "{span[ptile=2] rusage[ngpus_excl_p=1]} || {span[ptile=4] rusage[ngpus_excl_p=1]}"`
- Requests `N` CPUs and GPUs packed either 2 per node, or 4 per node.
