Launching Multiple jsruns

OLCF jsrun Tutorial

Chris Fuson
Feb 18, 2020
## Launching Multiple Jsruns

- Jsrun provides ability to launch multiple jsrun job launches within single batch job allocation
- Within single node, across multiple nodes

### Sequential
- Job launches execute one at a time
- Launch does not start until previous completes
- Bach job node allocation equal to largest jsrun
- Batch job walltime greater than sum of all jsruns

### Simultaneous
- Multiple job launches start at the same time
- Jsruns will not share core/gpu resources
- Batch job node allocation equal to sum of jsruns
- Batch job walltime greater than longest task
#!/bin/bash

#BSUB -W 2:00
#BSUB -nnodes 2
#BSUB -P abc007

cd $MEMBERWORK/abc007

jsrun -n12 -r6 -g1 -a2 -c2 ./a.out fileA

jsrun -n1 -r1 -a12 -c12 ./post_process fileA fileB

jsrun -n12 -r6 -g1 -a2 -c2 ./a.out fileB

jsrun -n1 -r1 -a12 -c12 ./post_process fileB fileC

jsrun -n12 -r6 -g1 -a2 -c2 ./a.out fileC
# Multiple Jsruns Simultaneously

```bash
#!/bin/bash

#BSUB -W 1:00
#BSUB -nnodes 6
#BSUB -P abc007

cd $MEMBERWORK/abc007

jsrun -n12 -r6 -g1 -a2 -c2 ./a.out fileA &
jsrun -n12 -r6 -g1 -a2 -c2 ./a.out fileB &
jsrun -n12 -r6 -g1 -a2 -c2 ./a.out fileC &

wait
```

- Walltime should be long enough to run longest running jsrun
- Allocated nodes should be as large as the sum of all simultaneous jsruns
- Placing jsruns in background allows each to run at same time. Jsrun will place each on separate resources.
- UNIX wait ensures script does not exit before backgrounded work completes.
- Without wait, batch job will exit before jsruns are complete.

<table>
<thead>
<tr>
<th>Simultaneous Nodes</th>
<th>Max Walltime</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>00:30:00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nodes Required</th>
<th>Time Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>00:25:00</td>
</tr>
<tr>
<td>+</td>
<td>2</td>
</tr>
<tr>
<td>00:30:00</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>2</td>
</tr>
<tr>
<td>00:25:00</td>
<td></td>
</tr>
</tbody>
</table>
--immediate: Another Simultaneous Method

- jsrun’s –immediate (-i) paired with ‘jswait all’
- Queue the job step and return
- By default STDOUT/ERR sent to /dev/null
  - --stdio_stdout (-o), --stdio_stderr (-k) change location
  - If unable to open specified file, will use /dev/null

$ jsrun -n1 -a1 -c1 -g6 js_task_info &; wait
Task 0 ( 0/1, 0/1 ) is bound to cpu[s] 0-3 on host h49n16 with OMP_NUM_THREADS=4 and with OMP_PLACES={0:4} and CUDA_VISIBLE_DEVICES=0,1,2,3,4,5

$ jsrun -n1 -a1 -c1 -g6 -i --stdio_stdout file.o --stdio_stderr file.e js_task_info; jswait all

$ cat file.e
Task 0 ( 0/1, 0/1 ) is bound to cpu[s] 0-3 on host h49n02 with OMP_NUM_THREADS=4 and with OMP_PLACES={0:4} and CUDA_VISIBLE_DEVICES=0,1,2,3,4,5
Viewing jsrun Queue -- *jslist*

- jsrun placement managed by IBM’s CSM (Cluster System Management)
- Aware of all jsrun allocations within LSF job; allows multiple per node, multi
  node, ...
- *jslist* – view the jsrun queue
  - Will show completed, running, and queued
  - Within interactive batch job will show current CSM allocation
  - Outside of a job can use –c to specify CSM allocation

```
$ bsub test.lsf
Job <26238> is submitted to default queue <batch>.

$ bjobs -l 26238 | grep CSM_ALLOCATION_ID
Sun Feb 16 19:01:18: CSM_ALLOCATION_ID=34435;

$ jslist -c 34435

<table>
<thead>
<tr>
<th>parent ID</th>
<th>ID</th>
<th>nrs</th>
<th>cpus per RS</th>
<th>gpus per RS</th>
<th>exit status</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>Running</td>
</tr>
</tbody>
</table>
```
Multiple Simultaneous Job Steps Example

`summit-login3`> `bsub -Is -nnodes2 -Pabc007 -W1:00 $SHELL`

Allocate 2 nodes

`summit-batch3`> `jsrun -n1 -a1 -c1 -g6 -bpacked:1 csh -c "js_task_info; sleep 30" &`

Task 0 ( 0/1, 0/1 ) is bound to cpu[s] 0-3 on host a01n02

All GPUs on node, 1 CPU

`summit-batch3`> `jsrun -n1 -a1 -c42 -g0 -bpacked:1 csh -c "js_task_info; sleep 30" &`

Task 0 ( 0/1, 0/1 ) is bound to cpu[s] 0-3 on host a01n01

Requires all cores on node, placed on separate node

`summit-batch3`> `jsrun -n1 -a1 -c1 -g1 -bpacked:1 csh -c "js_task_info; sleep 30" &`

Not enough free resources, waiting on completion of running step

`summit-batch3`> `jslist`

Note: In a batch job, backgrounded tasks require `wait` command

<table>
<thead>
<tr>
<th>ID</th>
<th>ID</th>
<th>nrs</th>
<th>cpus per RS</th>
<th>gpus per RS</th>
<th>exit status</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>Running</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>1</td>
<td>42</td>
<td>0</td>
<td>0</td>
<td>Running</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Queued</td>
</tr>
</tbody>
</table>

`jslist` command displays job steps
Questions?

• Documentation
  – docs.olcf.ornl.gov
  – Man pages
    • jsrun, bsub

• Help/Feedback
  – help@olcf.ornl.gov