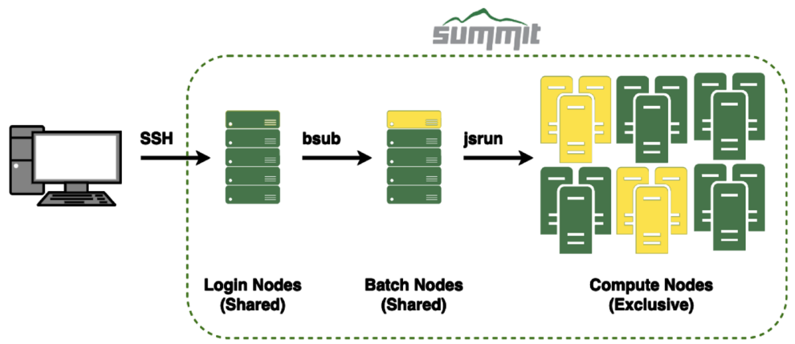
Introduction to Job Submission on Summit

Michael Sandoval

HPC Engineer - User Assistance Group
Oak Ridge Leadership Computing Facility (OLCF)
Oak Ridge National Laboratory (ORNL)

February 28, 2024

bsub: Submit a Job to LSF

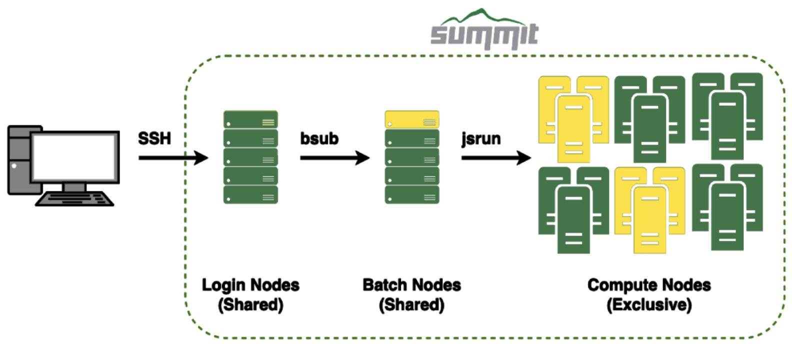


- bsub allocates 1 batch node plus the number of requested compute nodes.

bsub -alloc_flags defines allocation-wide configurations

- Applied to every compute node in your allocation:
  - CPU Simultaneous Multithreading (SMT)
  - GPU Multi-Process Service (gpumps)
  - Burst Buffer (nvme)
- Multiple options require quoting, space separated
  - E.g.: #BSUB –alloc_flags “gpumps smt2 nvme”

Jsrun is the job launcher for Summit



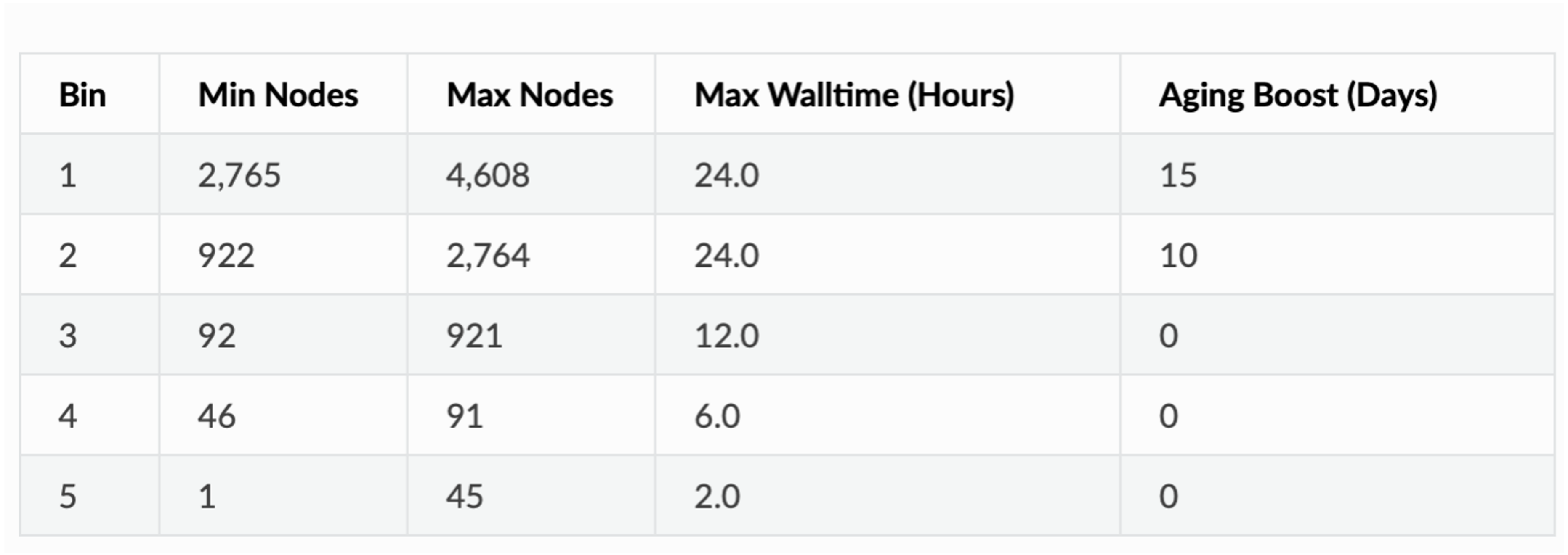
```
#!/bin/bash
#BSUB -P STF007
#BSUB -J myLSFjob
#BSUB -o out.%J
#BSUB -e err.%J
#BSUB -W 30
#BSUB -nnodes 1
#BSUB -alloc_flags nvme

hostname            ← batch1
jsrun -n1 hostname  ← g24n10
hostname            ← batch1
```

```
[suzanne@login3.summit ~]$ bsub batchScript.lsf
Job <1849558> is submitted to default queue <batch>.
[suzanne@login3.summit ~]$ 
```

https://docs.olcf.ornl.gov/systems/summit_user_guide.html#batch-scripts

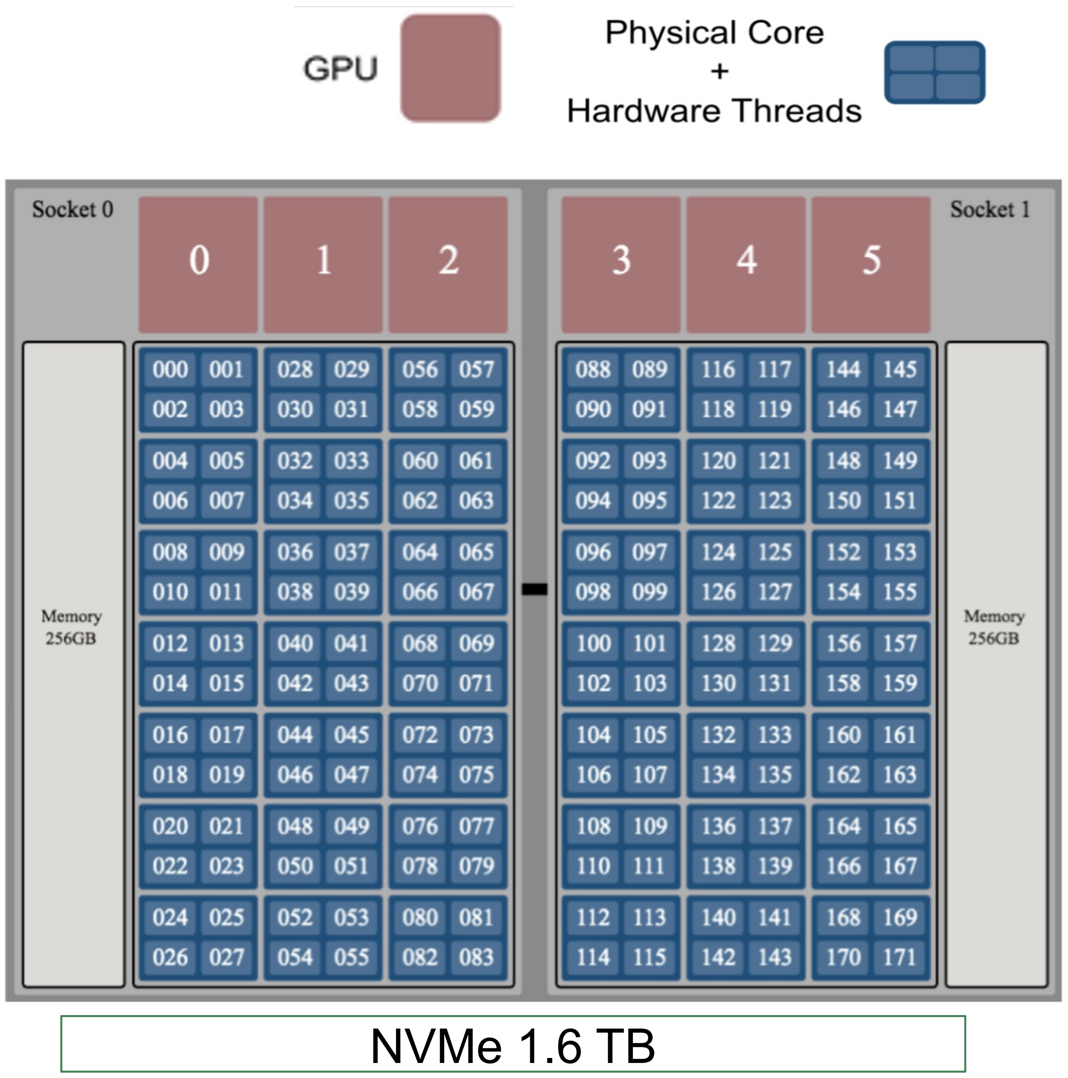
Queue Policy



Bin	Min Nodes	Max Nodes	Max Walltime (Hours)	Aging Boost (Days)
1	2,765	4,608	24.0	15
2	922	2,764	24.0	10
3	92	921	12.0	0
4	46	91	6.0	0
5	1	45	2.0	0

Summit Node

- IBM Power System AC922 Compute Node
- 2 Sockets
  - 3 NVIDIA V100 GPUs
  - 21 usable cores
    - {1,2,4}-way multithreading
  - 256 GB DDR4 RAM
- Exclusive node access

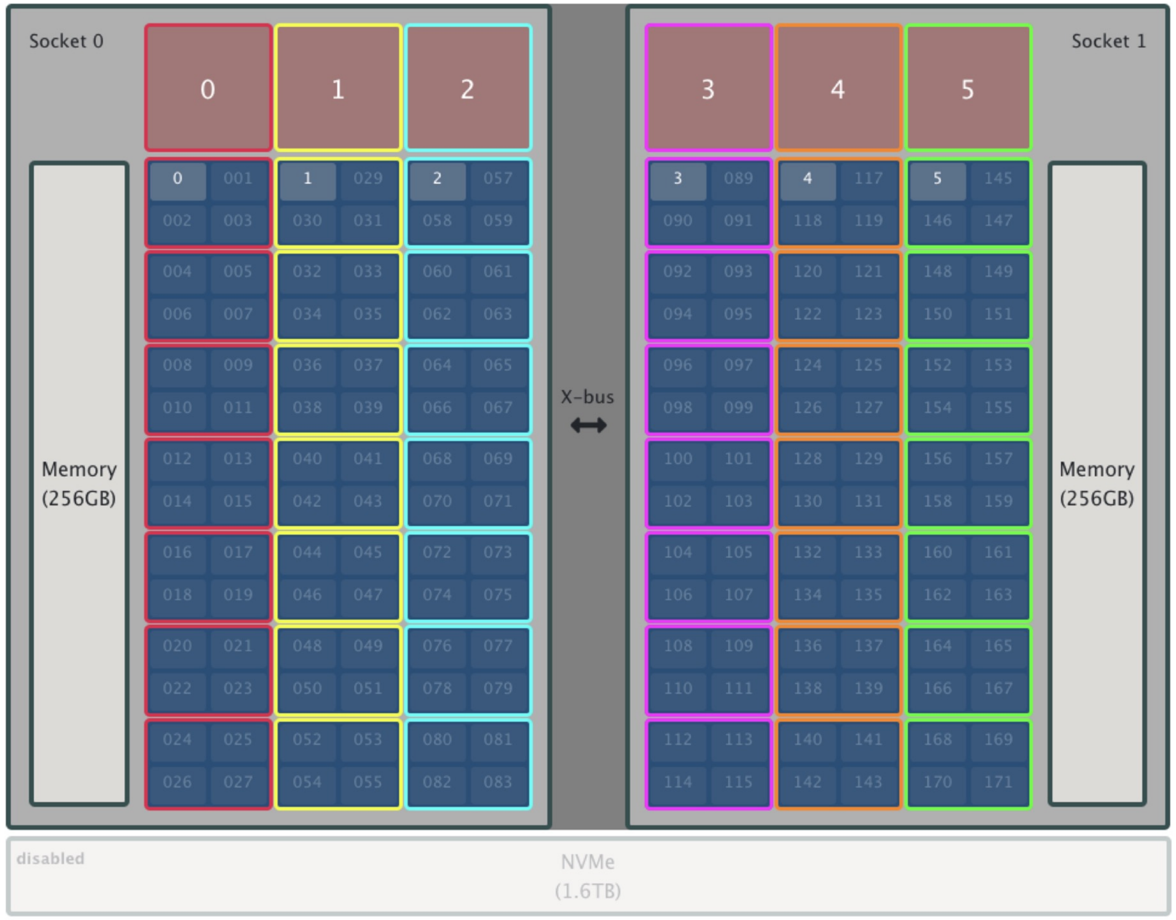


Resource Sets

jsrun [-n #resource sets] [tasks, threads, and GPUs in each resource set] program [program args]

This picture was made with the Job-step-viewer, a tool that lets you visually map your job layout.

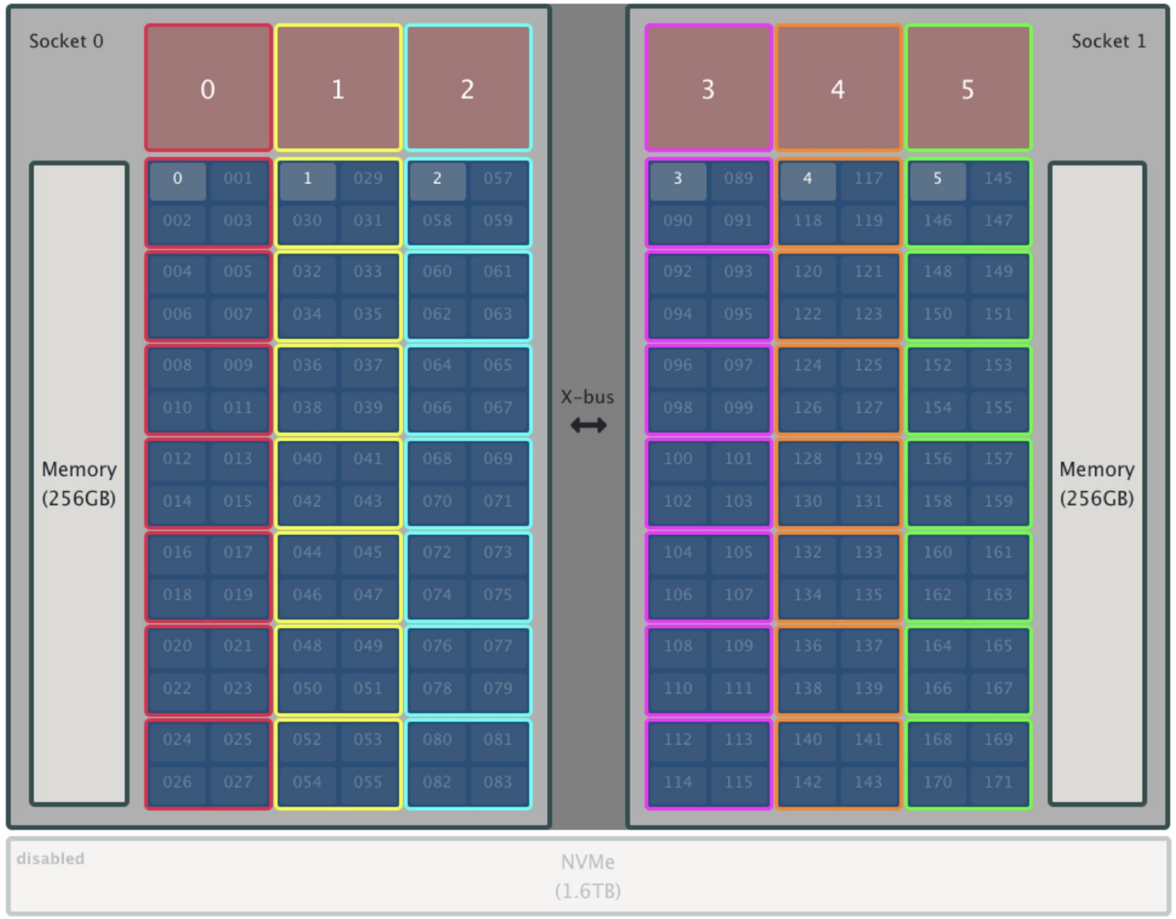
https://docs.olcf.ornl.gov/systems/summit_user_guide.html#job-step-viewer



Resource Sets

jsrun [-n #resource sets] [tasks, threads, and GPUs in each resource set] program [program args]

- Used to shape the compute node to your application by grouping (for example):
  - Physical cores
  - GPUs
- Must contain 1 or more physical cores and 0 or more GPUs
- Not as scary as they seem (but still overwhelming at first)



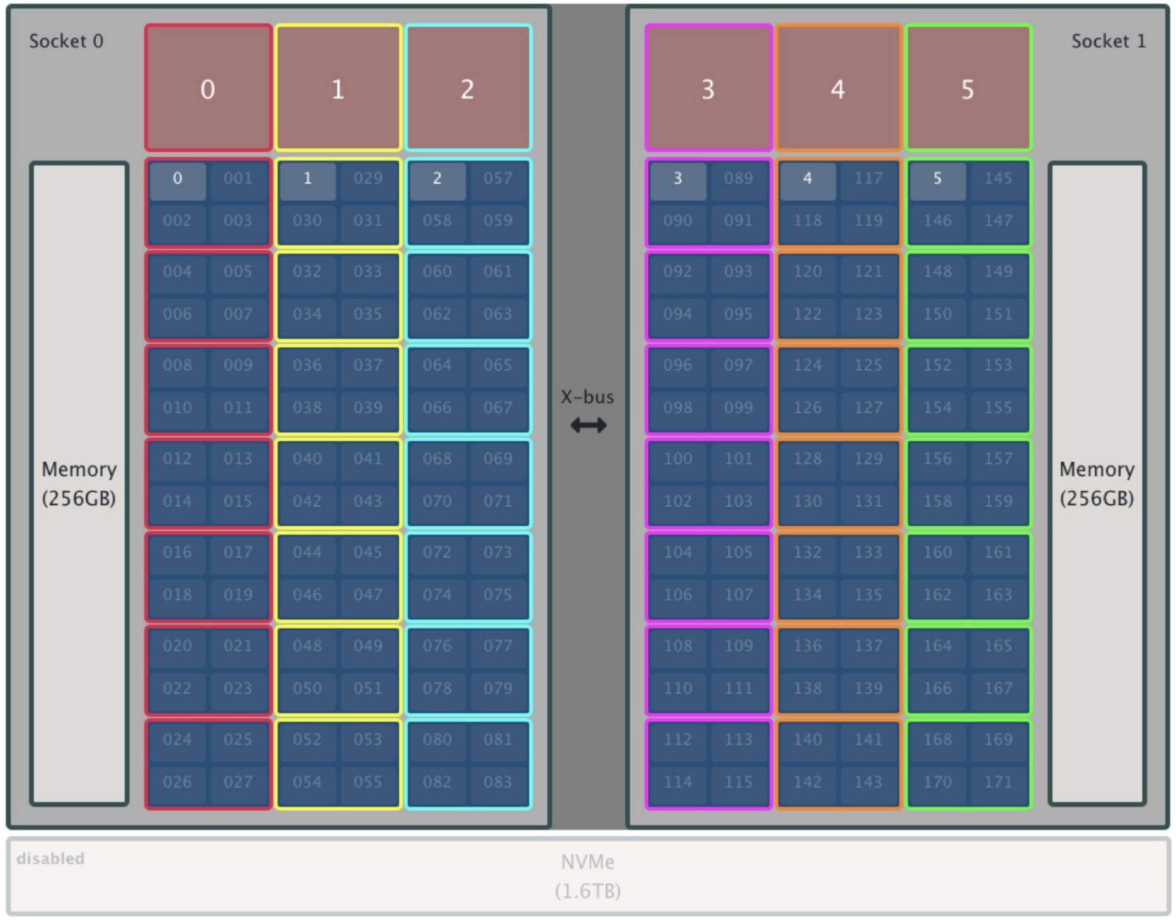
jsrun -n6 -c7 -a1, -g1 ./a.out

Resource Sets

jsrun [-n #resource sets] [tasks, threads, and GPUs in each resource set] program [program args]

Some limitations

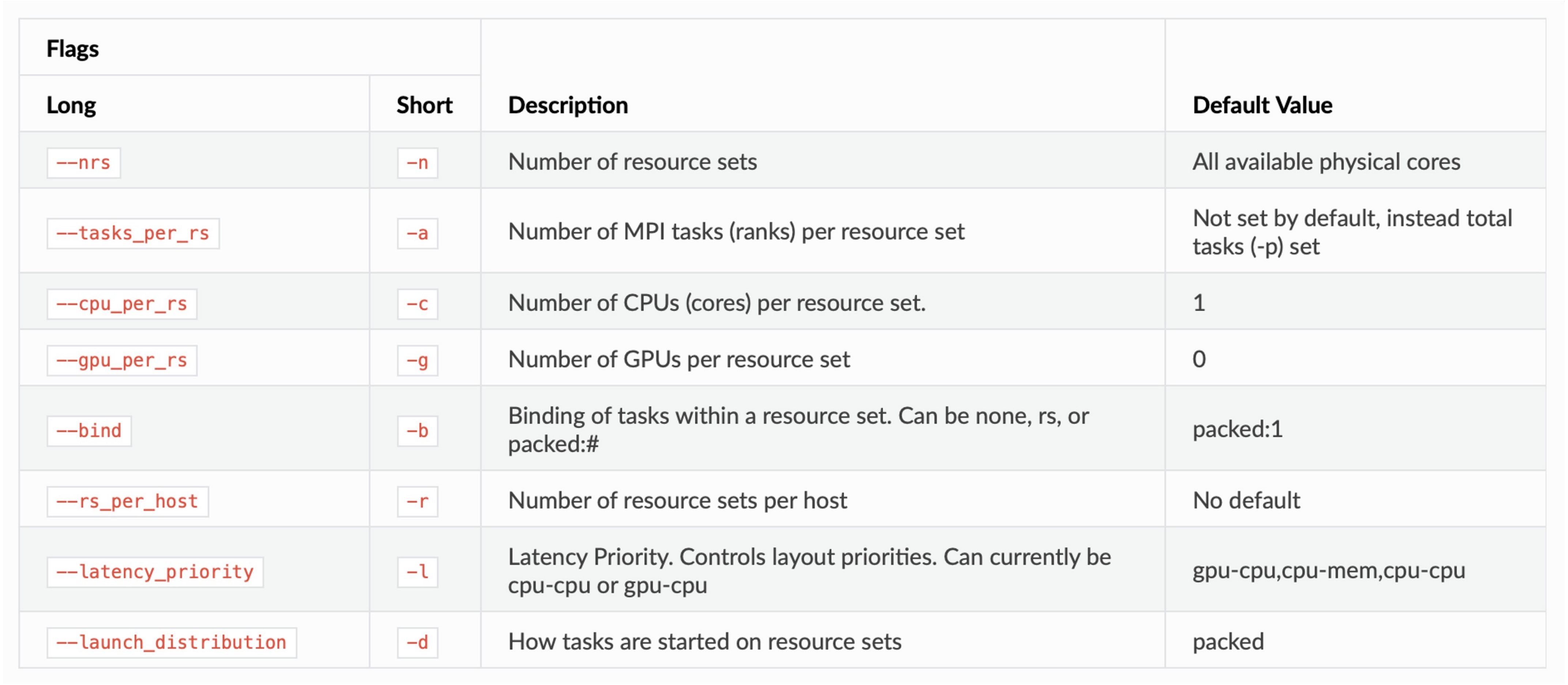
- A resource set may span sockets, but cannot span nodes
  - Creating resource sets within sockets can avoid cross-socket communication
- Memory access requires a physical core or GPU on its host socket



jsrun -n6 -c7 -a1, -g1 ./a.out

Designing a Resource Set

jsrun [-n #resource sets] [tasks, threads, and GPUs in each resource set] program [program args]



Flags			
Long	**Short**	**Description**	**Default Value**
`--nrs`	`-n`	Number of resource sets	All available physical cores
`--tasks_per_rs`	`-a`	Number of MPI tasks (ranks) per resource set	Not set by default, instead total tasks (-p) set
`--cpu_per_rs`	`-c`	Number of CPUs (cores) per resource set.	1
`--gpu_per_rs`	`-g`	Number of GPUs per resource set	0
`--bind`	`-b`	Binding of tasks within a resource set. Can be none, rs, or packed:#	packed:1
`--rs_per_host`	`-r`	Number of resource sets per host	No default
`--latency_priority`	`-l`	Latency Priority. Controls layout priorities. Can currently be cpu-cpu or gpu-cpu	gpu-cpu,cpu-mem,cpu-cpu
`--launch_distribution`	`-d`	How tasks are started on resource sets	packed

Designing a Resource Set

jsrun [-n #resource sets] [tasks, threads, and GPUs in each resource set] program [program args]

1. Understand how the code expects the node to appear
   - How many tasks/threads per GPU?
   - Does each task expect to see a single GPU?
   - Do multiple tasks expect to share a GPU (gpumps)?
2. Create Resource Sets containing the needed GPU to task binding
   - Describe a resource set that meets the requirements above. If the code is written for one GPU per task, consider a resource set with just one GPU
3. Decide on the number of Resource Sets needed
   - After understanding task, thread, and GPU requirements, scale the number of Resource Sets (and number of nodes) as needed.

Examples

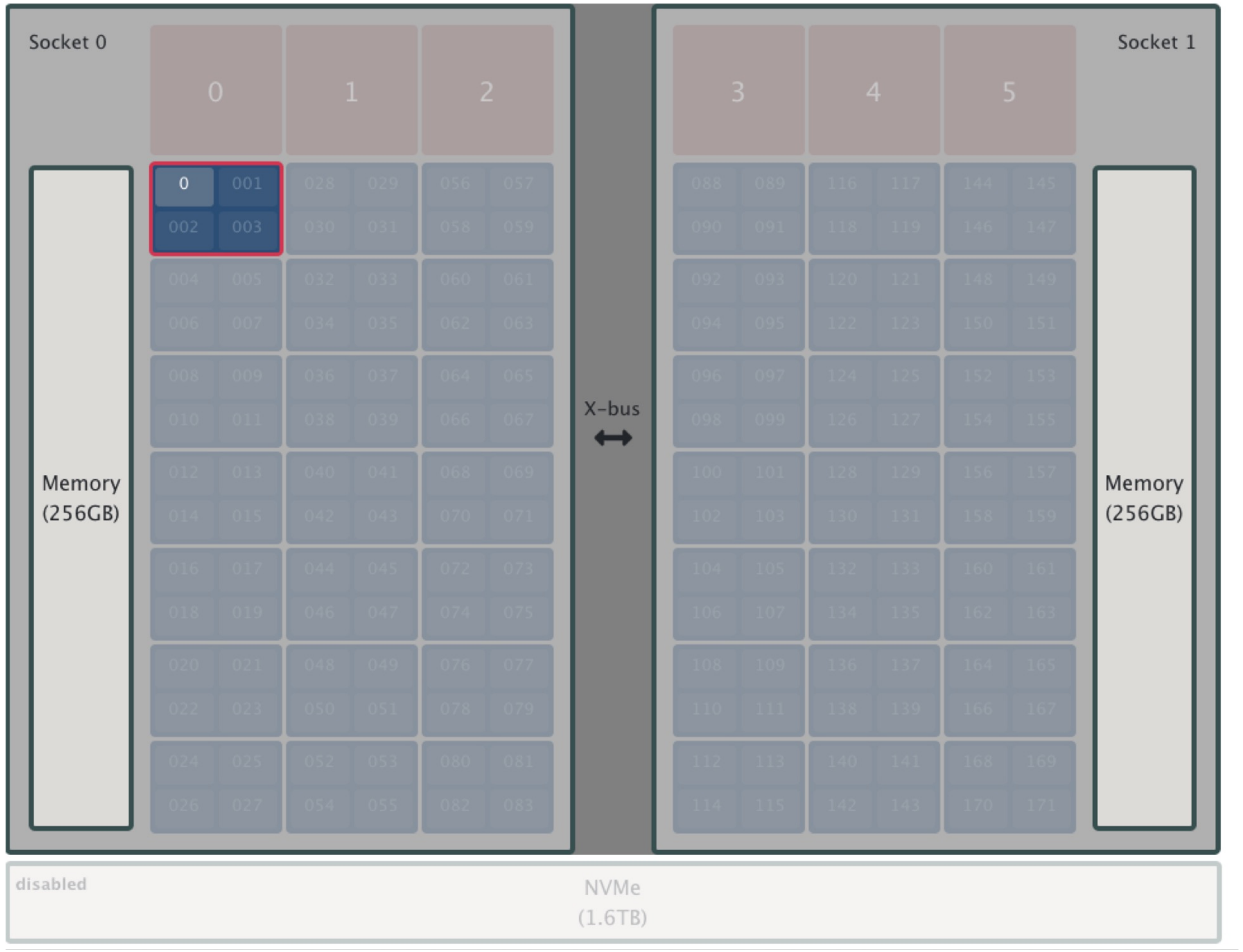
jsrun -n1

- By default you get one core with one task.
- This is the same as

jsrun -n1 -c1 -a1

- Better not to rely on defaults

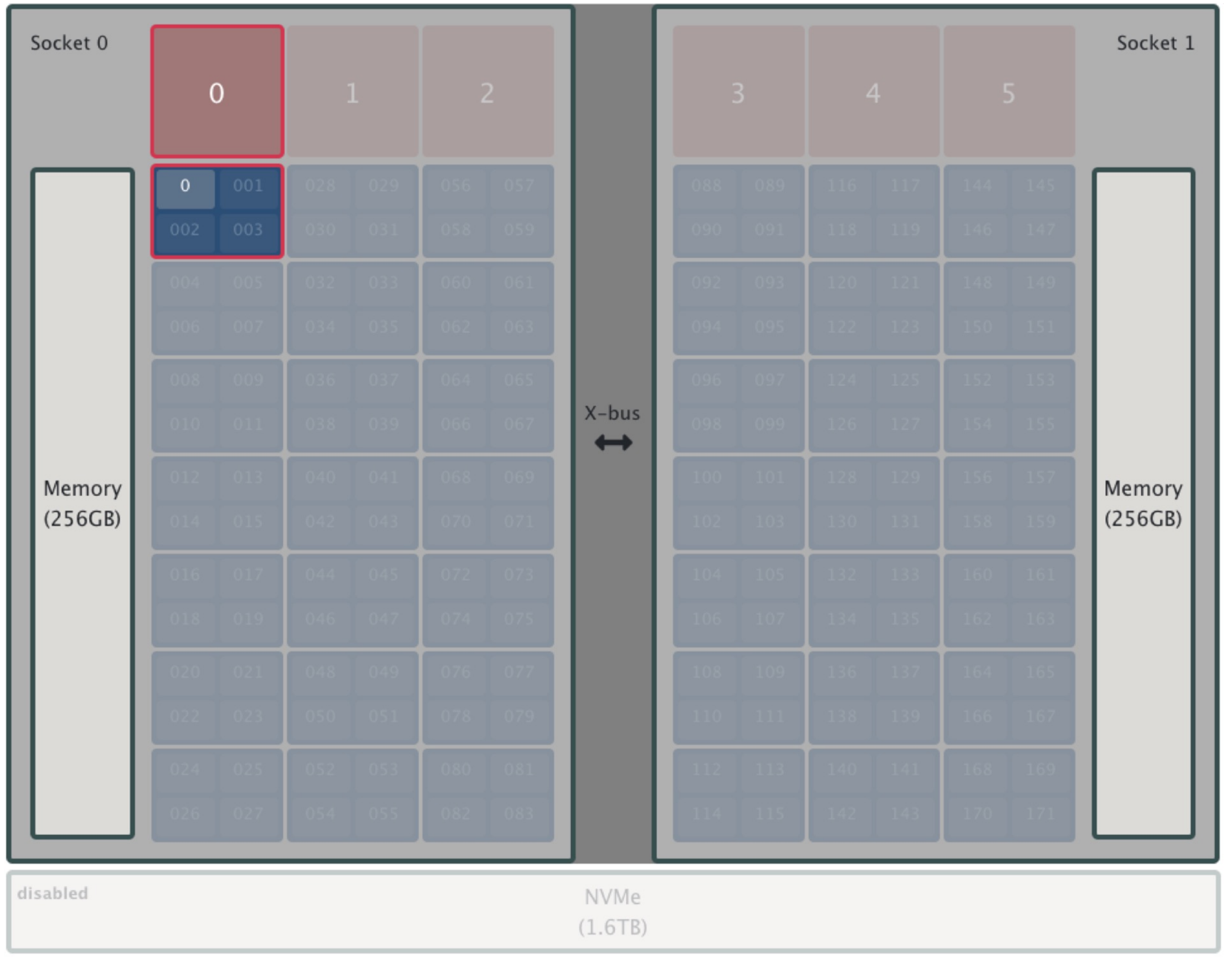
Let's add a GPU . . .



Examples

jsrun -n1 -c1 -a1 -g1

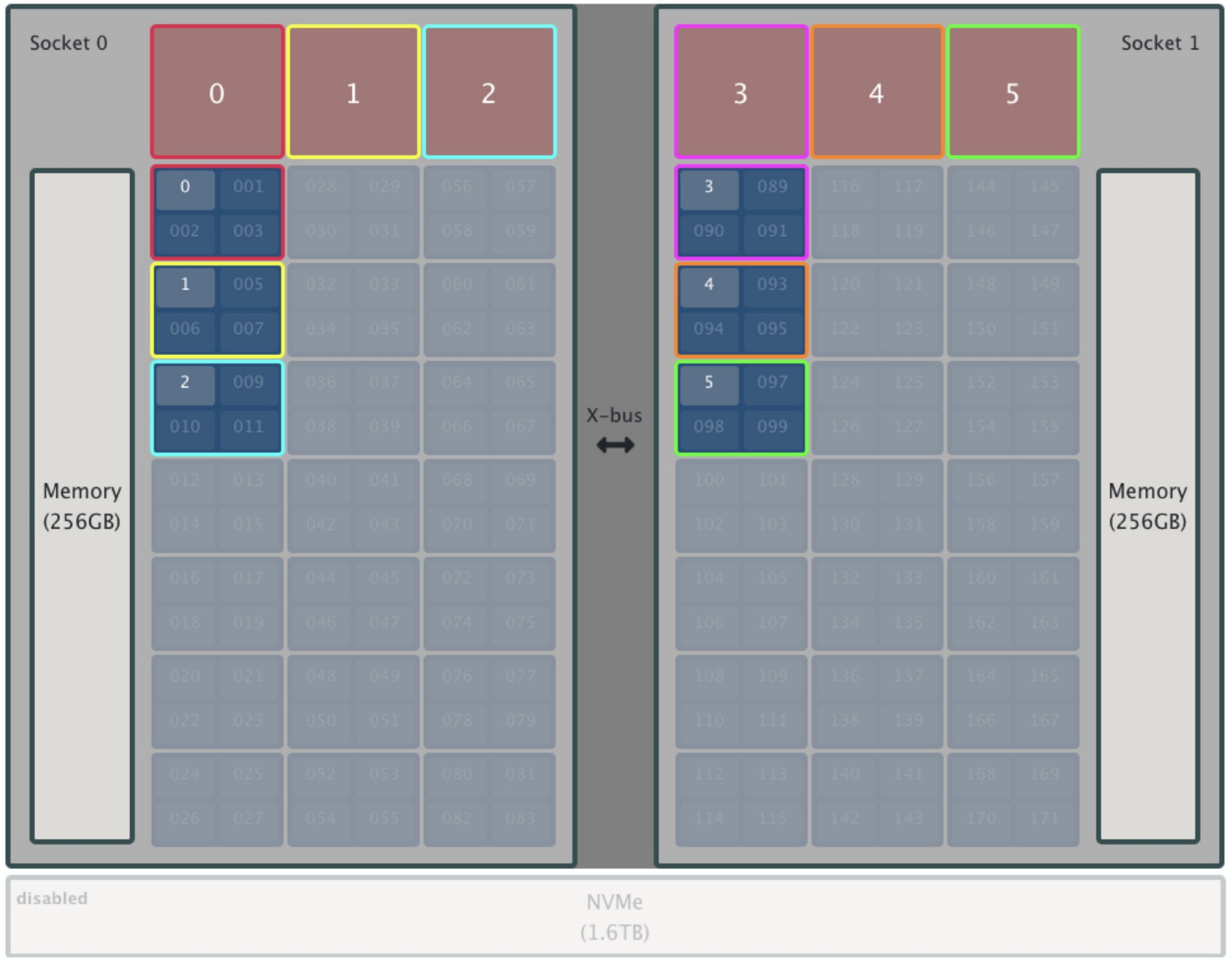
- -g flag specifies the number of GPUs per resource set.
- How many of these sets can we fit on the node?



Examples

```
jsrun -n6 -c1 -a1 -g1
```

- How would this scale to two or more nodes?



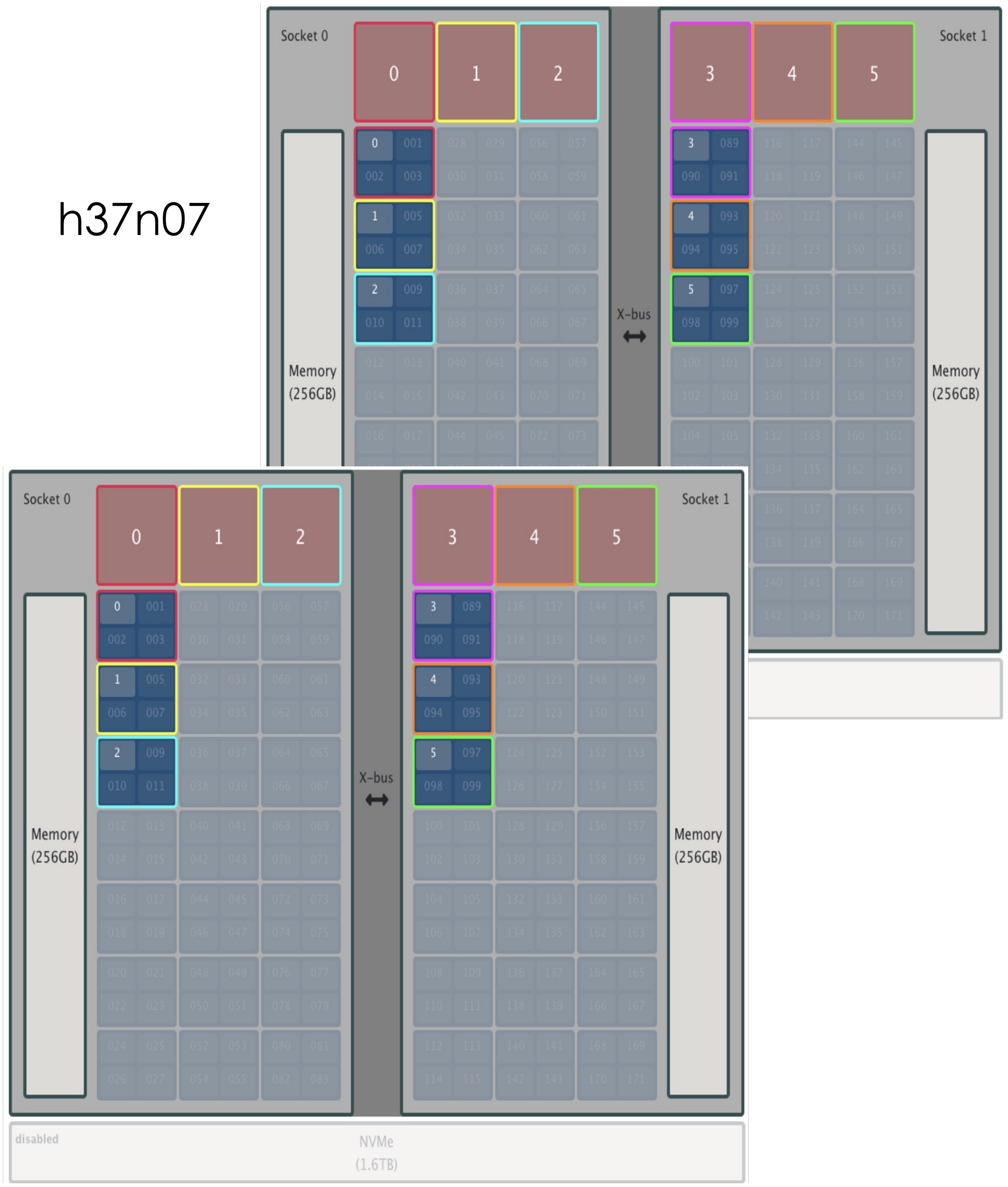
Examples: Multiple Nodes

jsrun -n12 -c1 -a1 -g1

- -n will be the number of nodes you reserved multiplied by the number of resource sets on one node.
- If your resources sets don’t fill both sockets, jsrun may not map the resources set as you expect when you expand to multiple nodes.
- Let's talk about that and threads next.

h37n07

h37n08

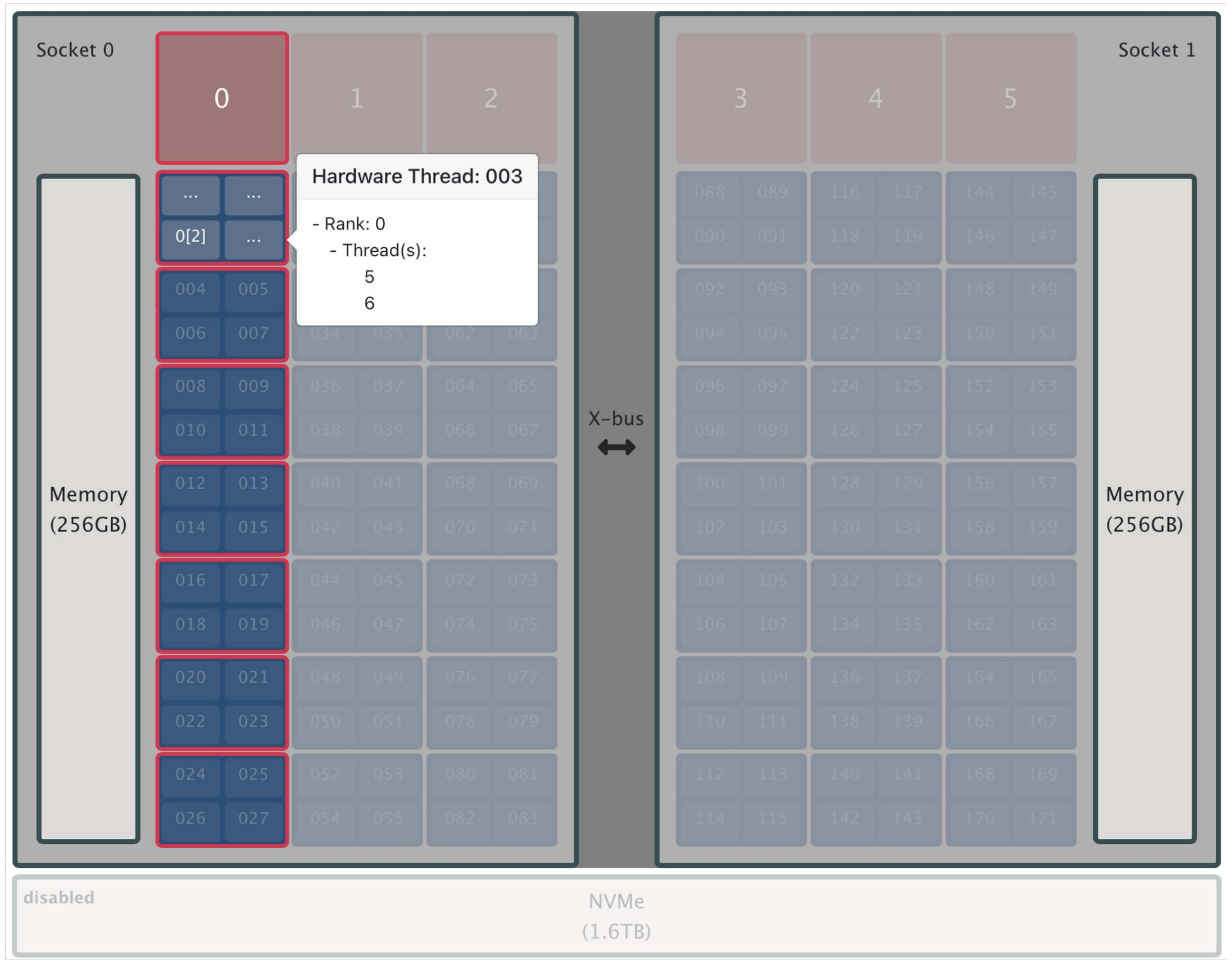


OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Examples: Threading

jsrun -n1 -c7 -a1 -EOMP_NUM_THREADS=7 -g1

- **-EOMP_NUM_THREADS=#** lets you set the number of openMP threads per task.
- Could also do
  export OMP_NUM_THREADS=7 just before the jsrun command.
- All 7 threads are clustered on the same core!
- How do we fix this?
- Hint: `-b packed:1` was the default



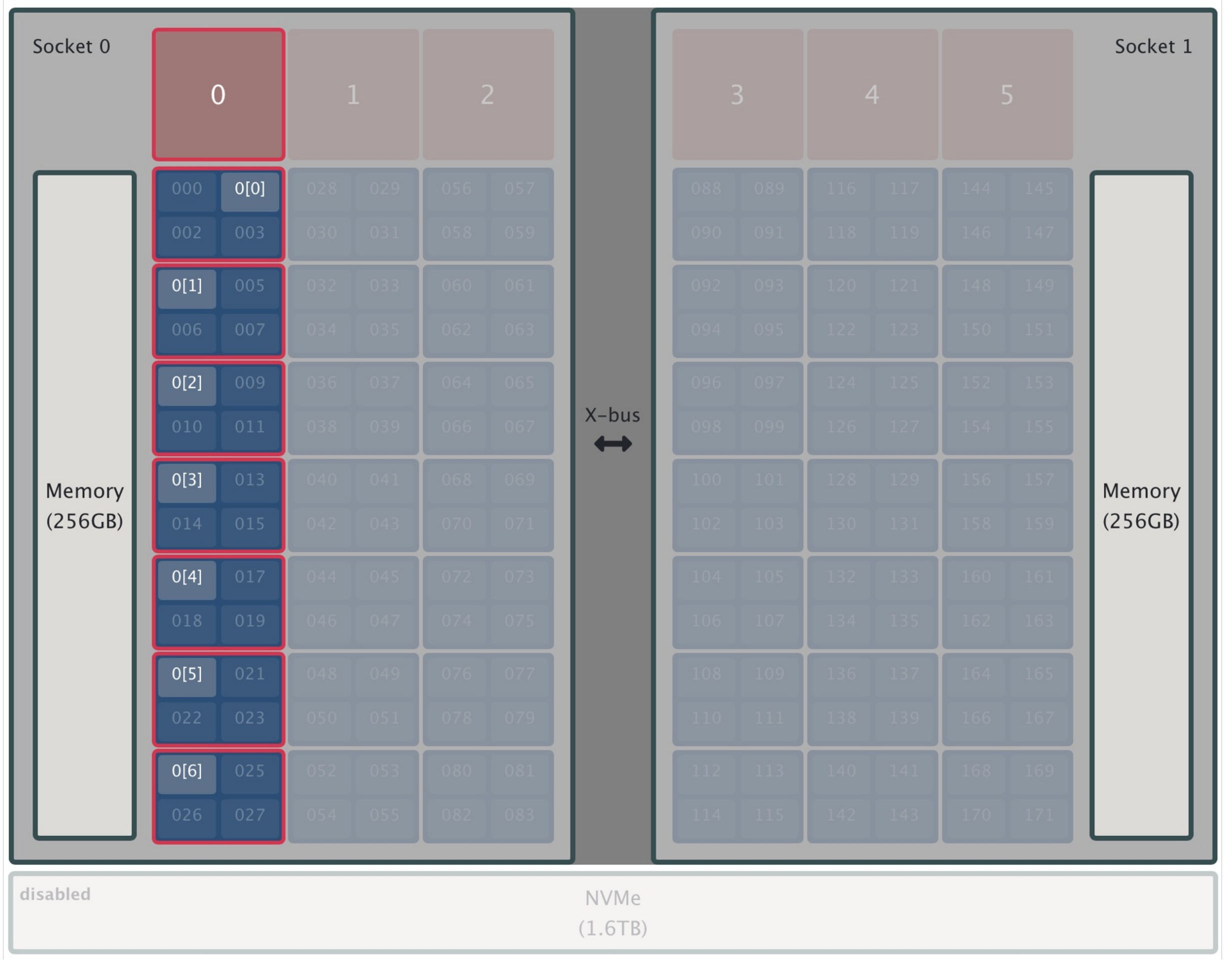
Hardware Thread: 003

- Rank: 0
  - Thread(s):
    5
    6

Examples: Threading -brs

jsrun -n1 -c7 -a1 -EOMP_NUM_THREADS=7 -g1 -brs

- -b controls the thread binding options are: **packed:#, rs, none**
- Allows you to set the number of physical cores available to an MPI task
- -b rs binds the threads to fill the available cores in the resource set
- You could also use -b packed:7, in other words set the binding to arrange 7 cores per task
- How would I do this over 2 nodes?



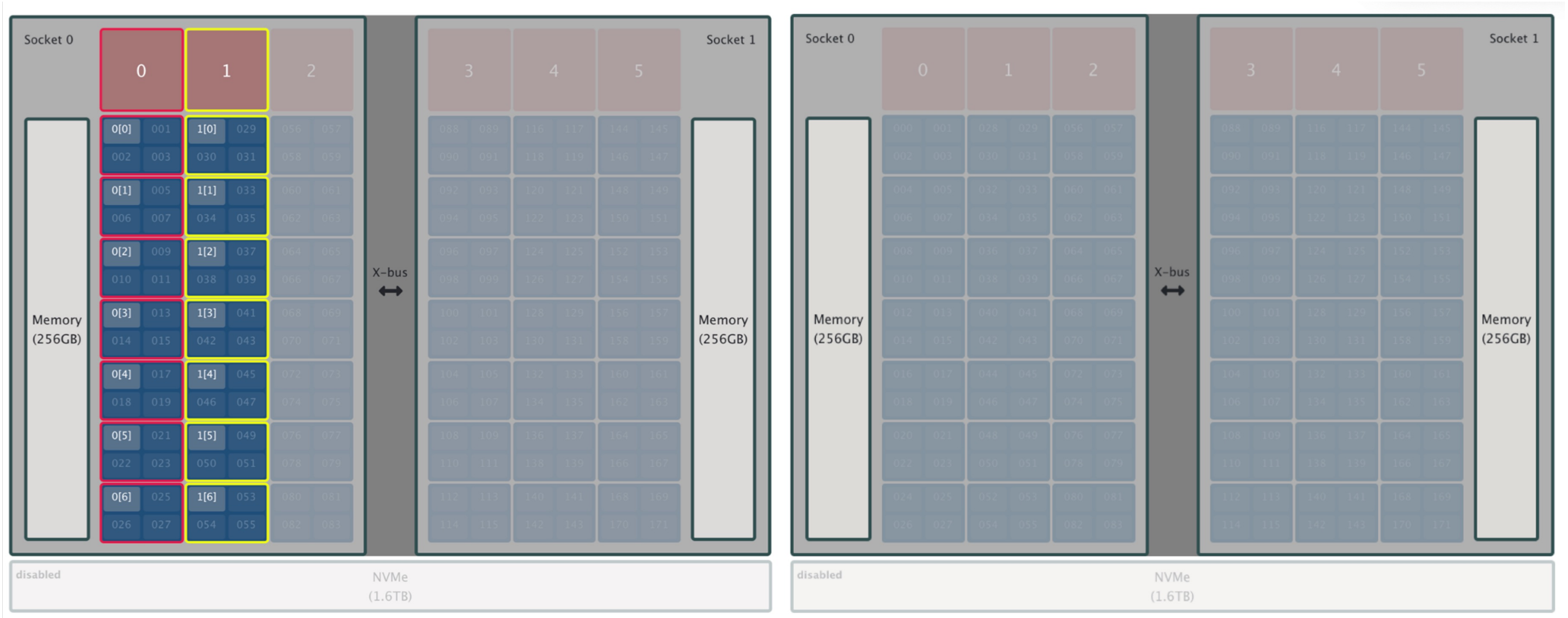
Examples: Threading Multiple Nodes

```
jsrun -n2 c7 -a1 -EOMP_NUM_THREADS=7 -g1 -brs
```

Examples: Threading Multiple Nodes

!Wast of Allocation!

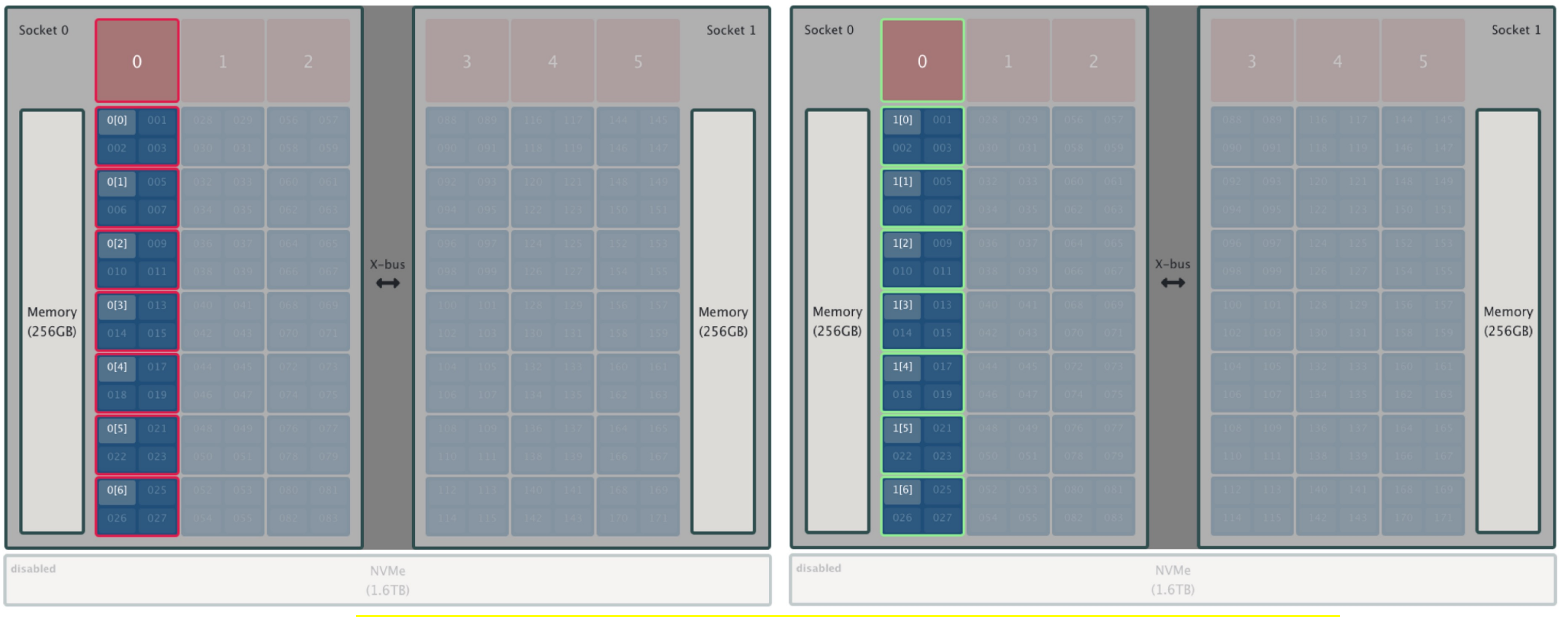
```
jsrun -n2 c7 -a1 -EOMP_NUM_THREADS=7 -g1 -brs
```



Examples: Threading Multiple Nodes

But this still might be a waste of allocation.

```
jsrun -n2 -r1 -c7 -a1 -EOMP_NUM_THREADS=7 -g1 -brs
```



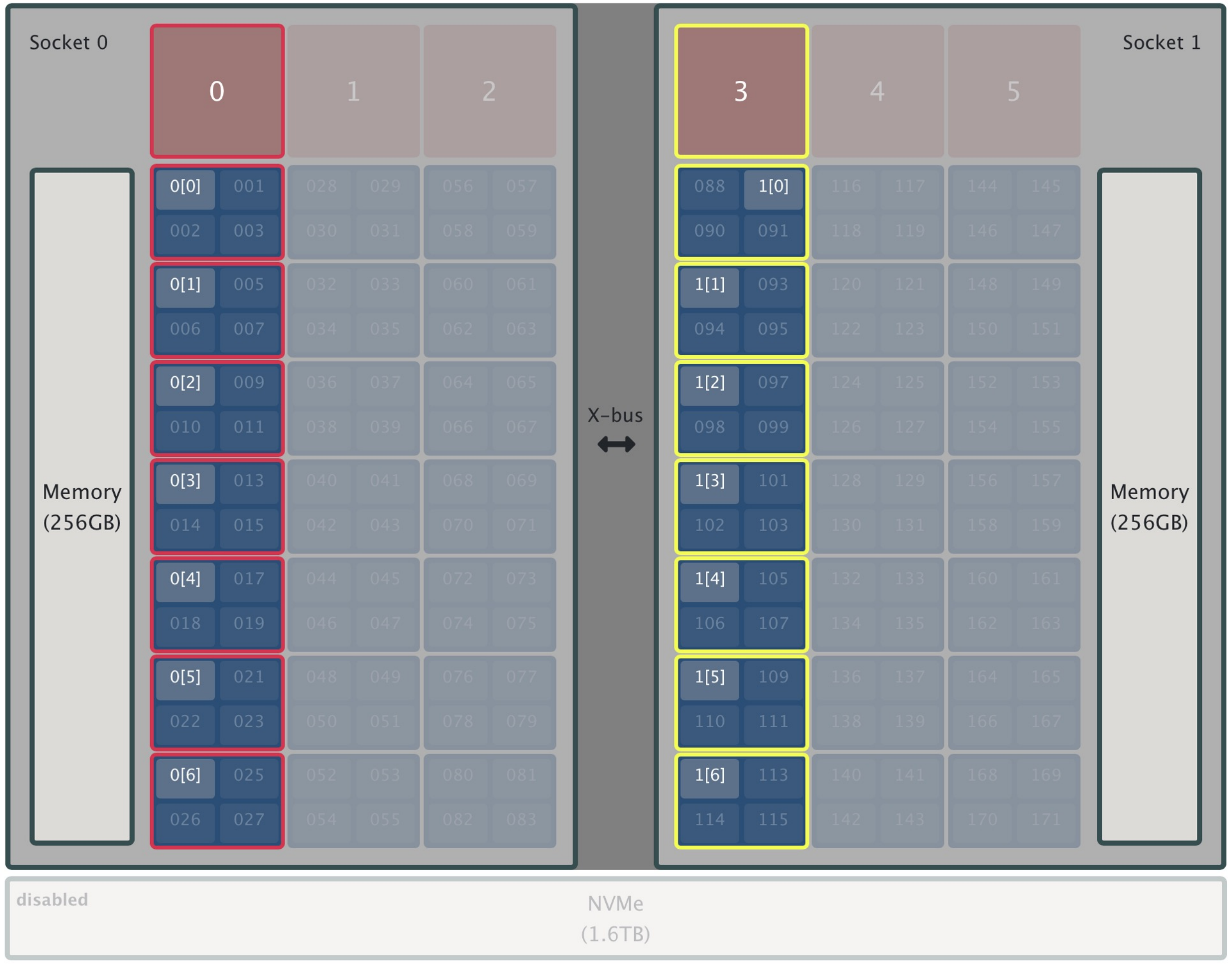
Remember you must have a core or GPU reserved on each socket to get access to that socket's memory,so what you see above does not give each resource set 512 GB, but rather only 265 GB.

Examples: Threading Multiple Nodes

```
jsrun -n2 --rs_per_socket 1 -c7 -a1 -EOMP_NUM_THREADS=7 -g1 -brs
```

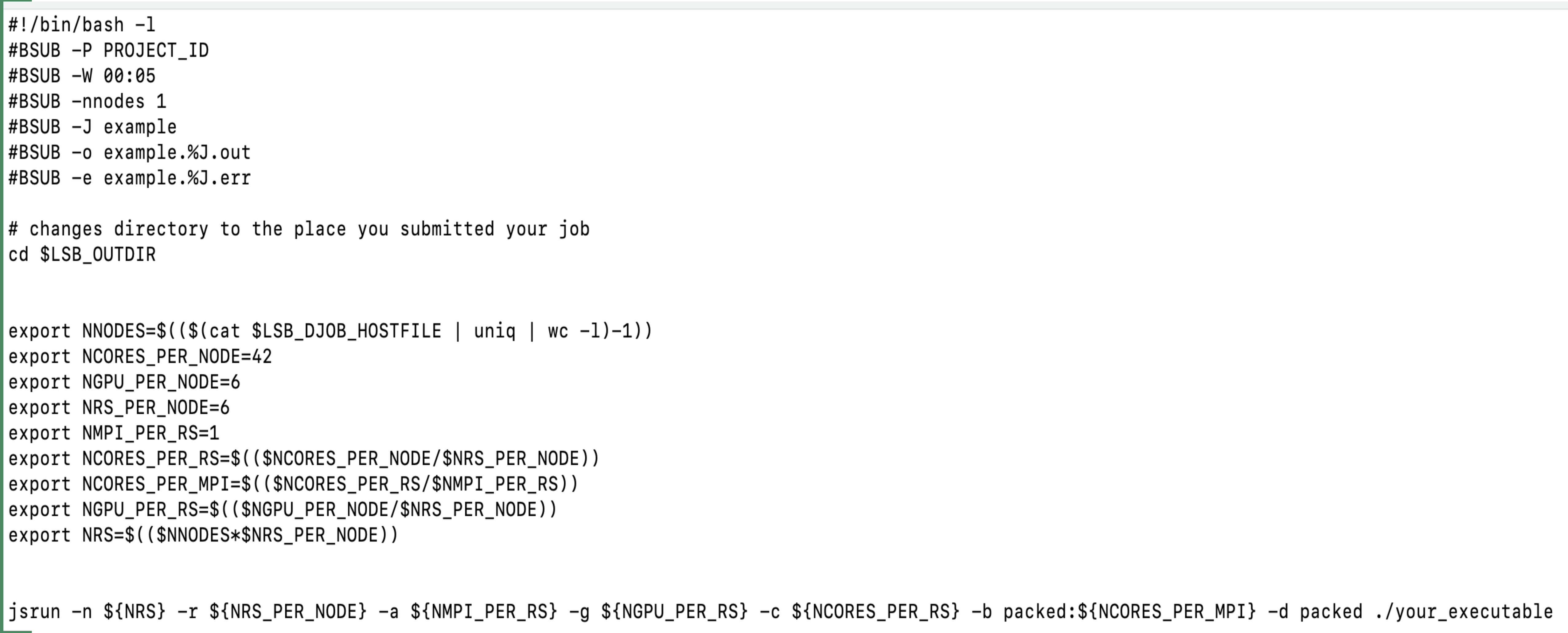
If you used -n 4, you'd get two nodes that look like this.

- Setting the resource sets per socket to 1 gives you one set on each socket, each able to get full access to that node's 265 GB of RAM.
- This gives you a more efficient use of allocation than the configuration on the previous slide.



OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Batch Script Template



```
#!/bin/bash -l
#BSUB -P PROJECT_ID
#BSUB -W 00:05
#BSUB -nnodes 1
#BSUB -J example
#BSUB -o example.%J.out
#BSUB -e example.%J.err

# changes directory to the place you submitted your job
cd $LSB_OUTDIR


export NNODES=$(($(cat $LSB_DJOB_HOSTFILE | uniq | wc -l)-1))
export NCORES_PER_NODE=42
export NGPU_PER_NODE=6
export NRS_PER_NODE=6
export NMPI_PER_RS=1
export NCORES_PER_RS=$(($NCORES_PER_NODE/$NRS_PER_NODE))
export NCORES_PER_MPI=$(($NCORES_PER_RS/$NMPI_PER_RS))
export NGPU_PER_RS=$(($NGPU_PER_NODE/$NRS_PER_NODE))
export NRS=$(($NNODES*$NRS_PER_NODE))


jsrun -n ${NRS} -r ${NRS_PER_NODE} -a ${NMPI_PER_RS} -g ${NGPU_PER_RS} -c ${NCORES_PER_RS} -b packed:${NCORES_PER_MPI} -d packed ./your_executable
```

Resources

- OLCF user docs for running jobs: https://docs.olcf.ornl.gov/systems/summit_user_guide.html#running-jobs
  - Covers main jsrun options with examples of each.
  - Also options for monitoring your jobs.
- man jsrun
  - This has the complete descriptions of all Jsrun options and advice about how combinations of those options will be interpreted
- Job-step-viewer
  - Demo: https://jobstepviewer.olcf.ornl.gov
  - How To: https://docs.olcf.ornl.gov/systems/summit_user_guide.html#job-step-viewer
- **Summit New User Quick Start guide: https://docs.olcf.ornl.gov/quickstart/index.html**
  - Has slides and recordings of past jsrun trainings (and many other new user topics)
  - Hands on exercises with answers for jsrun