UT Southwestern Medical Center
Lyda Hill Department of Bioinformatics

BioHPC

Using the SLURM Job Scheduler

[web] portal.biohpc.swmed.edu
[email] biohpc-help@utsouthwestern.edu

Overview

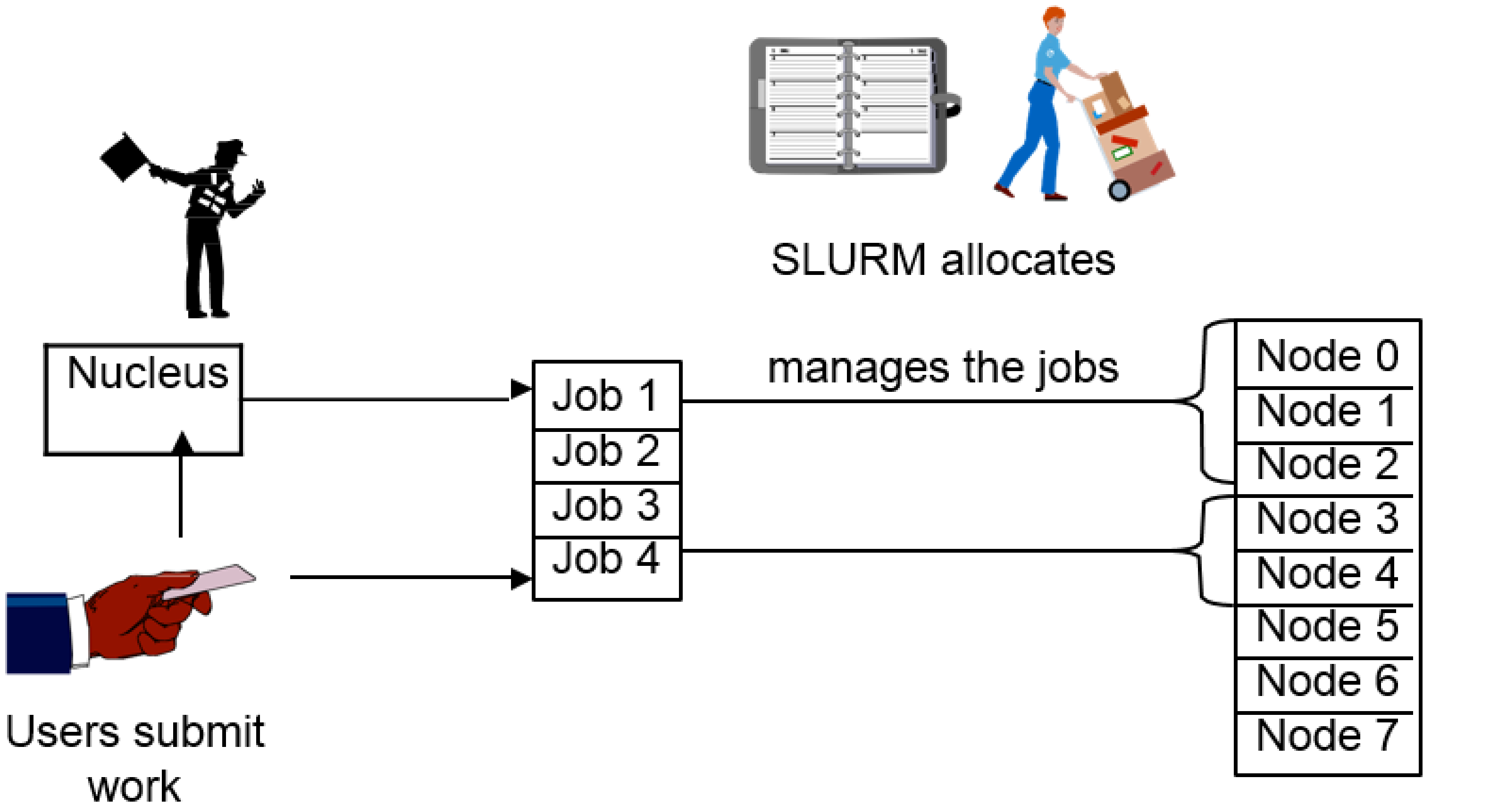
- Part I: What is SLURM? / SLURM Basics
- Part II: `sbatch` scripts for job submission to a single node
- Part III: Running GPU and MPI jobs

Part I: What is SLURM?

- **S**imple **L**inux **U**tility for **R**esource **M**anagement
  - Tell SLURM what your job needs to run
  - It worries about where to put it (and when!)
  - Juggles jobs so the run as quickly and efficiently as possible

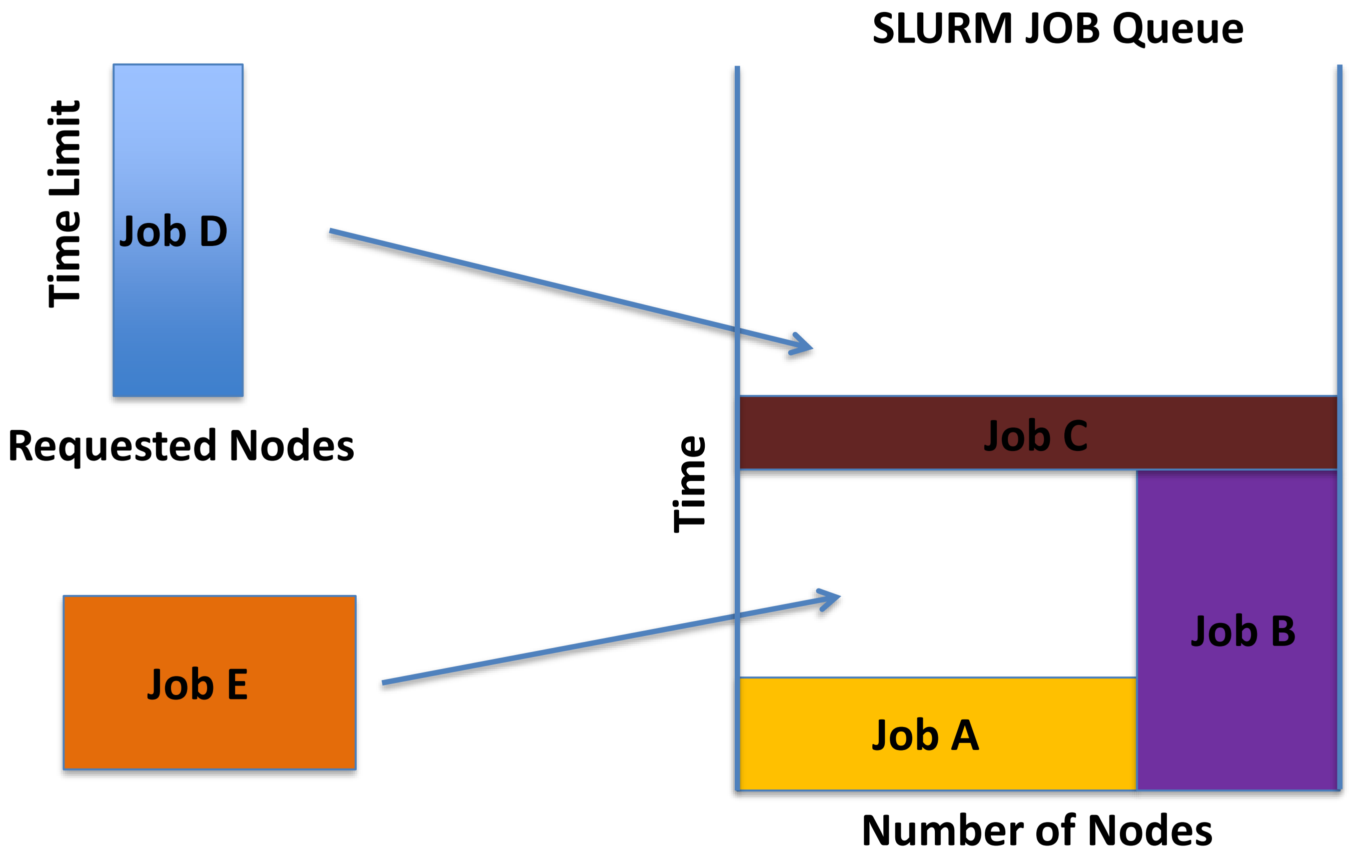
- The "glue" for a parallel computer to execute parallel jobs
  - Make a parallel computer almost as easy to use as a PC
  - *Not magic* – can't make your serial program parallel

Part I: What does SLURM do?



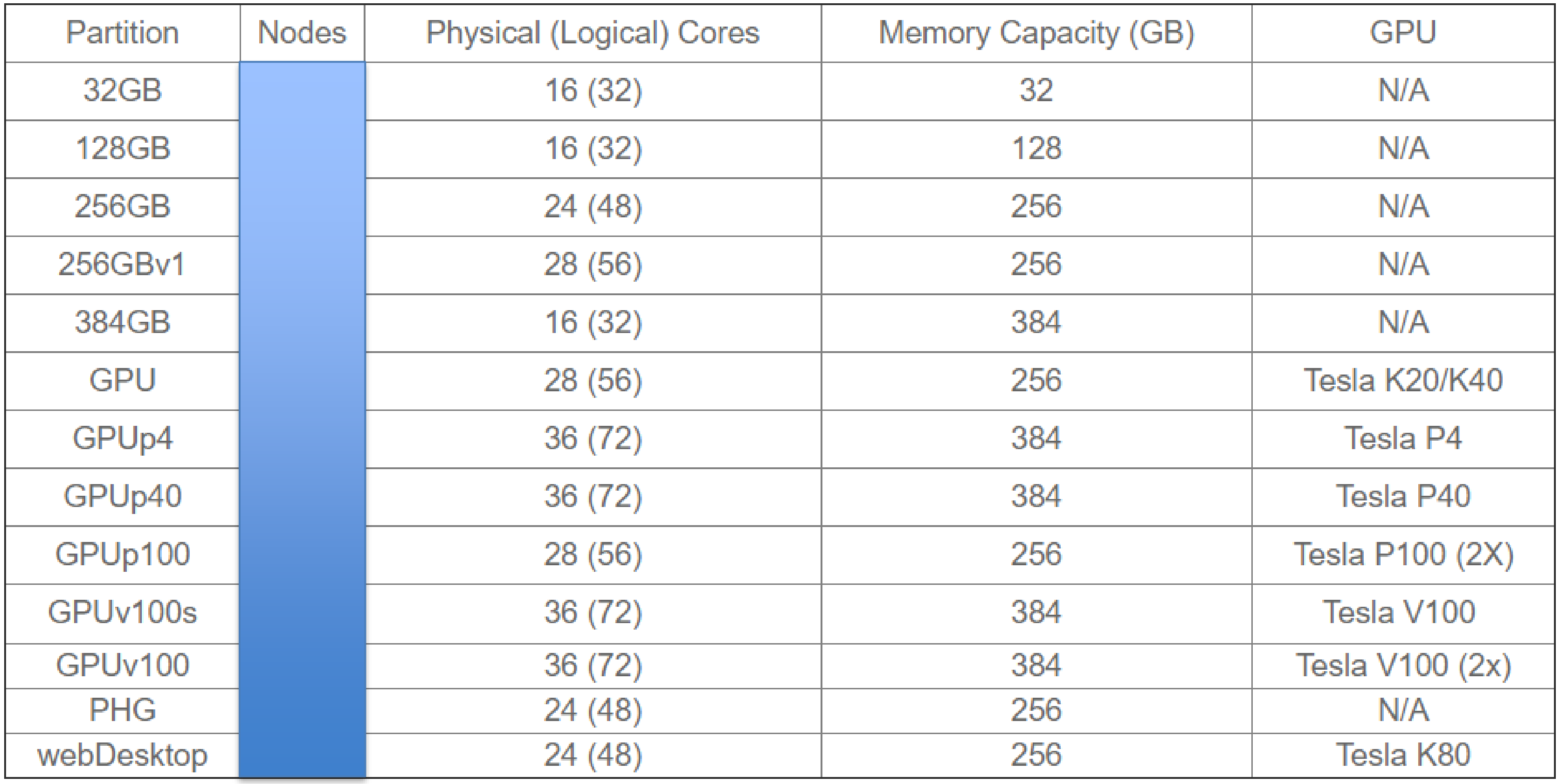
Login node → **SLURM job queue** → **Computer nodes**

Part I: How SLURM schedules jobs (time limits are important!)



Estimated compute time < User specified time limit < 2*Estimated compute time

Part I: Types of nodes in the Nucleus cluster



Partition	Nodes	Physical (Logical) Cores	Memory Capacity (GB)	GPU
32GB		16 (32)	32	N/A
128GB		16 (32)	128	N/A
256GB		24 (48)	256	N/A
256GBv1		28 (56)	256	N/A
384GB		16 (32)	384	N/A
GPU		28 (56)	256	Tesla K20/K40
GPUp4		36 (72)	384	Tesla P4
GPUp40		36 (72)	384	Tesla P40
GPUp100		28 (56)	256	Tesla P100 (2X)
GPUv100s		36 (72)	384	Tesla V100
GPUv100		36 (72)	384	Tesla V100 (2x)
PHG		24 (48)	256	N/A
webDesktop		24 (48)	256	Tesla K80

https://portal.biohpc.swmed.edu/content/guides/slurm/
https://slurm.schedmd.com/quickstart.html

Part I: BioHPC CPU Partitions (or Queue)

Partition – a collection of cluster nodes
CPU nodes are grouped by amount of RAM

279	32GB: NucleusA003-241, B002-041
24	128GB: Nucleus010-033
86	256GB: Nucleus034-161
54	256GBv1: Nucleus126-195
18	384GB: Nucleus082-083, NucleusB042-057

super
any of these

Total 338 compute (CPU) nodes as of April 2021

Part I: BioHPC GPU Partitions (or Queue)

GPU nodes are grouped by type of GPU card

8 GPU* : Nucleus042-049 (K20/K40 card)

16 GPUp4 : NucleusC002-C017 (P4 Card)

16 GPUp40: NucleusC018-033 (P40 Card)

30 GPUv100s: NucleusC036-069 (V100s Card)

70 in GPU partition *any of these*

12 GPUp100: Nucleus162-173 (2xP100 Cards)

2 GPUv100s: NucleusC034-35 (2xV100 Cards)

12 GPU4v100: NucleusC070-081 (4xV100 Cards)

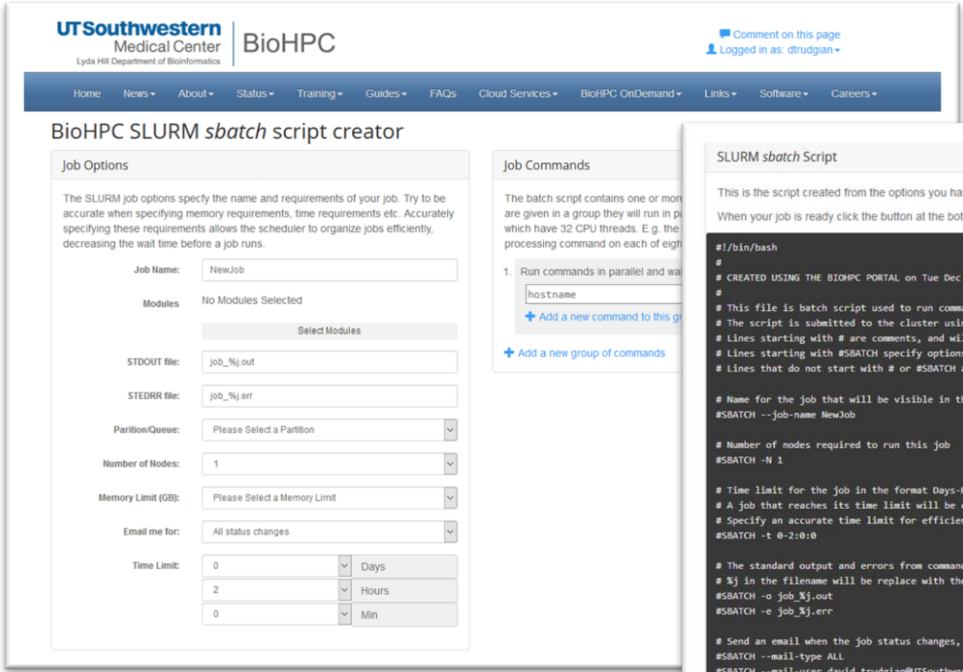
16 GPUA100: NucleusC086-C101 (A100 Card)

Not part of main GPU partition

72 Total GPU nodes as of April 2021

Part I: How to submit a job to the BioHPC cluster

https://portal.biohpc.swmed.edu --> Cloud Services --> Web Job Submission



BioHPC SLURM *sbatch* script creator

Job Options

The SLURM job options specfy the name and requirements of your job. Try to be accurate when specifying memory requirements, time requirements etc. Accurately specifying these requirements allows the scheduler to organize jobs efficiently, decreasing the wait time before a job runs.

Job Name:	NewJob
Modules	No Modules Selected
	Select Modules
STDOUT file:	job_%j.out
STEDRR file:	job_%j.err
Parition/Queue:	Please Select a Partition
Number of Nodes:	1
Memory Limit (GB):	Please Select a Memory Limit
Email me for:	All status changes
Time Limit:	0 Days
	2 Hours
	0 Min

Job Commands

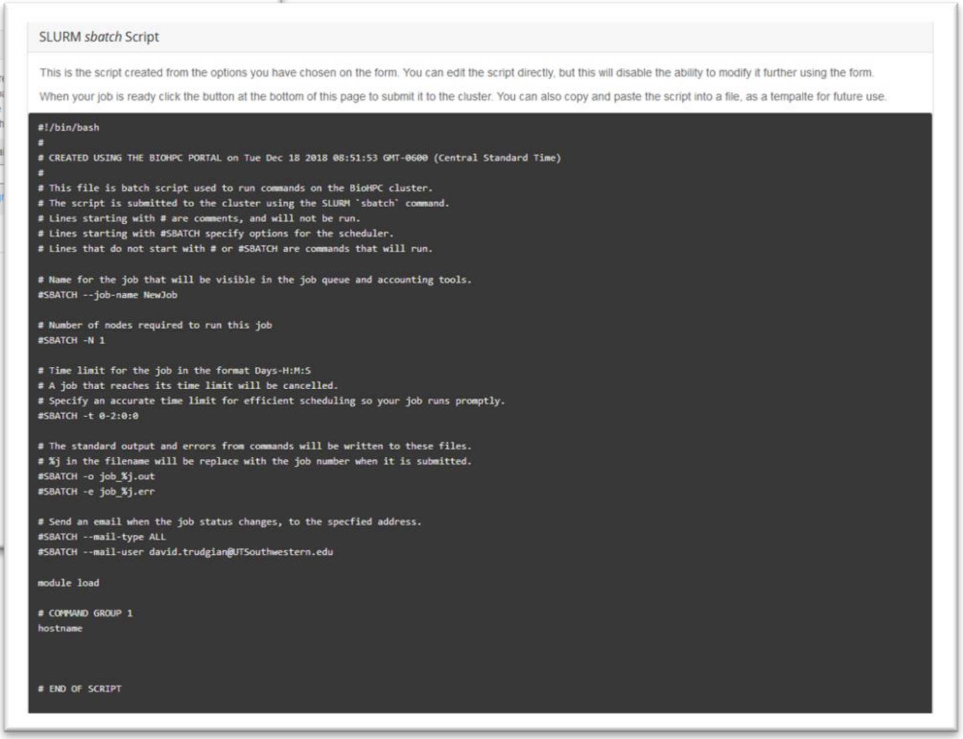
The batch script contains one or more
are given in a group they will run in p
which have 32 CPU threads. E.g. the
processing command on each of eigh

1. Run commands in parallel and wa

hostname

+ Add a new command to this g

+ Add a new group of commands



SLURM *sbatch* Script

This is the script created from the options you have chosen on the form. You can edit the script directly, but this will disable the ability to modify it further using the form.

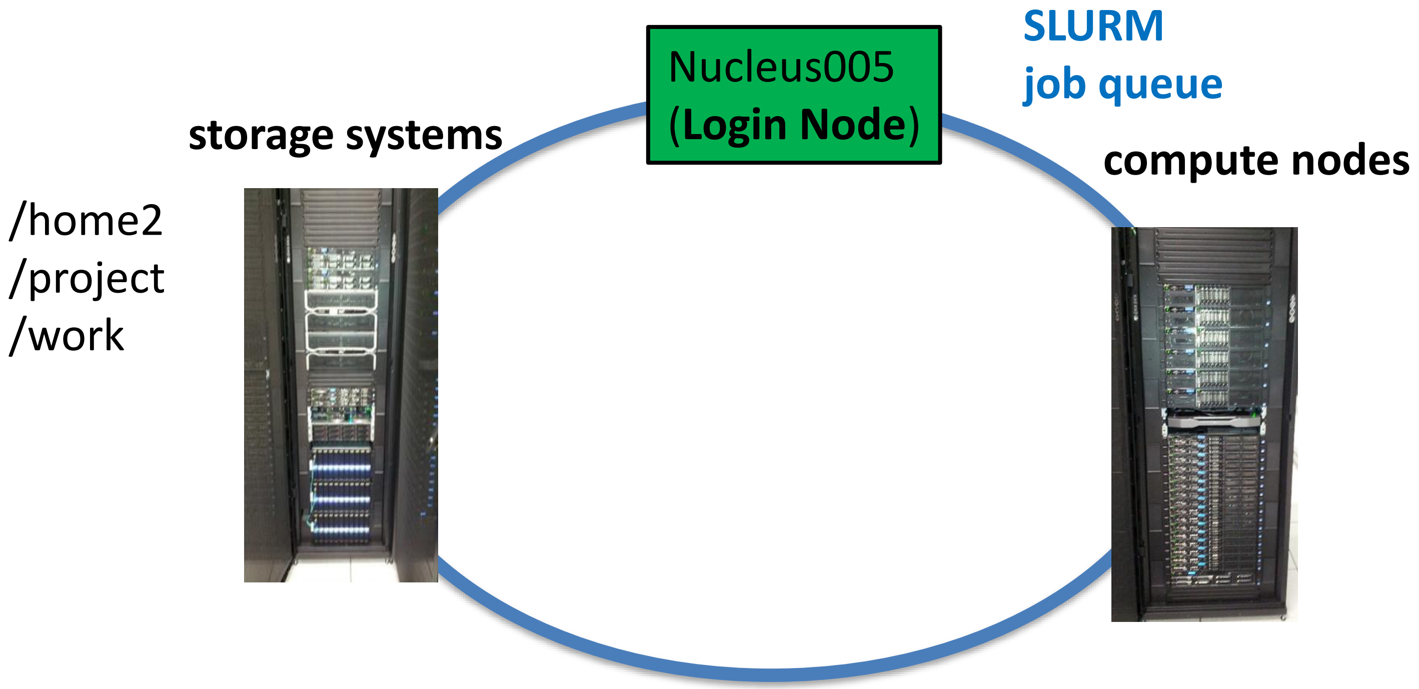
When your job is ready click the button at the bottom of this page to submit it to the cluster. You can also copy and paste the script into a file, as a tempalte for future use.

```
#!/bin/bash
#
# CREATED USING THE BIOHPC PORTAL on Tue Dec 18 2018 08:51:53 GMT-0600 (Central Standard Time)
#
# This file is batch script used to run commands on the BioHPC cluster.
# The script is submitted to the cluster using the SLURM `sbatch` command.
# Lines starting with # are comments, and will not be run.
# Lines starting with #SBATCH specify options for the scheduler.
# Lines that do not start with # or #SBATCH are commands that will run.

# Name for the job that will be visible in the job queue and accounting tools.
#SBATCH --job-name NewJob

# Number of nodes required to run this job
#SBATCH -N 1

# Time limit for the job in the format Days-H:M:S
# A job that reaches its time limit will be cancelled.
# Specify an accurate time limit for efficient scheduling so your job runs promptly.
#SBATCH -t 0-2:0:0

# The standard output and errors from commands will be written to these files.
# %j in the filename will be replace with the job number when it is submitted.
#SBATCH -o job_%j.out
#SBATCH -e job_%j.err

# Send an email when the job status changes, to the specfied address.
#SBATCH --mail-type ALL
#SBATCH --mail-user david.trudgian@UTSouthwestern.edu

module load

# COMMAND GROUP 1
hostname



# END OF SCRIPT
```

Part I: How to submit a job to the BioHPC cluster

Login via SSH to **nucleus.biohpc.swmed.edu**

```
sbatch myscript.sh
```



Part I: Login Node Usage

Nucleus005
(**Login Node**)

nucleus.biohpc.swmed.edu
Gateway of BioHPC cluster
Shared resource: many people use it at the same time

At the login Node

You Can:

view/move/copy/edit files
compile code
submit jobs via SLURM
check job status

You Should Not:

run long-term applications/jobs – *use a batch job*
run short tasks that need large CPU/RAM – *use a webGUI session*

Part I: SLURM commands

- SLURM commands
  - Before job submission: **`sinfo`**, **`squeue`**
  - Submit a job: **`sbatch`**, **`srun`**, **`salloc`**
  - During job running: **`squeue`**, **`scontrol`**
  - After job completed: **`sacct`**
- Manual pages (**`man`**) available for all commands (e.g., **`man sinfo`**)
  - Help option prints brief descriptions of all options
  - Usage option prints a list of the options
  - Almost all options have two formats:

    A single letter option (e.g. "-p super")

    A verbose option (e.g., "--partition=super")

Part I: status reports, queue, jobs

```
$ sinfo  (report status in node-oriented form)

$ sinfo -p GPUp4 (report status of nodes in partition “GPUp4”)

$ sinfo -n Nucleus100

$ squeue

$ squeue -p 128GB

$ squeue -u $USER

$ scontrol show job <jobID>

$ scancel <jobID>

$ sacct –j <jobID>
```

`scontrol` gives more detailed information of the job, but only for recent jobs;

`sacct` keeps a completed history of job status, but only basic information.

Part I: interactive jobs

```
[s191529@Nucleus006 ~]$ salloc --job-name "test" --time 2:00:00 -p GPUA100
salloc: Granted job allocation 2268890
[modulestats] Wrapper already loaded
[s191529@Nucleus006 ~]$ squeue -j 2268890
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2268890   GPUA100     test  s191529  R       0:13      1 NucleusC089
[s191529@Nucleus006 ~]$ ssh NucleusC089
Do your work =D
[s191529@NucleusC089 ~]$ exit
logout
Connection to nucleusc089 closed.
[s191529@Nucleus006 ~]$ scancel 2268890
salloc: Job allocation 2268890 has been revoked.
Hangup
```

Part II: Writing & Submitting SLURM Jobs

- An example job: counting frequencies of words in stories
- A Python 3 script ‘word_freq.py’ we can run and pass the name of a story text file: `$ python word_freq.py stories/story-1.txt`
- We have 16 stories, name ‘story-0.txt’ to ‘story-15.txt’
- Can be found in the ‘stories’ directory in the example .zip file

Part II: Testing your job before submission

- **On your own machine**
  Software and environment might not match the cluster

- **At your local workstation/thin-client**
  Same version of Linux as the cluster, but with a graphical desktop.
  Same software available as on cluster.

- **Reserve a BioHPC compute node**
  WebGUI : reserve a CPU node
  WebGPU: reserve a GPU node

Part II: Basic structure of SLURM script

1_single.sh

```
#!/bin/bash                                        run SLURM script under bash shell

#SBATCH --job-name=1_single
#SBATCH --partition=super
#SBATCH --nodes=1
#SBATCH --time=00-00:01:00                                      format: D-H:M:S
#SBATCH --output=1_single.%j.out
#SBATCH --error=1_single.%j.err                          set up SLURM environment

module add python/3.6.4-anaconda              load software (export path & library)

python3 word_freq.py stories/story-1.txt              command(s) to be executed
```

Part II: More SBATCH options

```
#SBATCH --begin=now+1hour
```
Defer the allocation of the job until the specified time

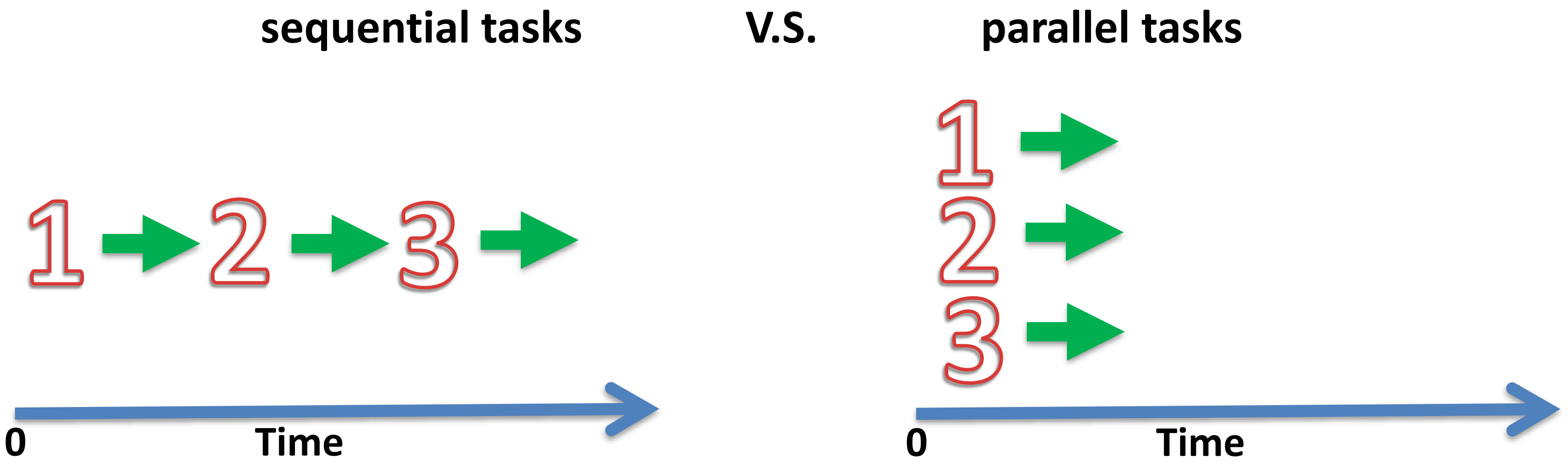
```
#SBATCH --mail-type=ALL
```
Notify user by email when certain event types occur (BEGIN, END, FAIL, REQUEUE, etc.)

```
#SBATCH --mail-user=john.doe@utsouthwestern.edu
```
Use to receive email notification of state changes as defined above

```
#SBATCH --mem=262144
```
Specify the real memory required per node in Megabytes (262144MB=256GB).

```
#SBATCH --nodelist=Nucleus0[10-20]
```
Request a specific list of node names. The order of the node names in the list is not important, the node names will be sorted by SLURM

Part II: Demo 2 & 3 – submit multiple tasks to single node



```
#!/bin/bash

#SBATCH --job-name=multiple
#SBATCH --partition=super
#SBATCH --nodes=1
#SBATCH --time=00-00:01:00
#SBATCH --output=2_multiple.%j.out
#SBATCH --error=2_multiple.%j.err

module add python/3.6.4-anaconda
```

For both sequential and parallel tasks, SLURM environment and the software we needed are the same. The difference is from how your write your commands.

Part II: Demo 2 & 3 – submit multiple tasks to single node

Demo 2: sequential tasks 2_sequential.sh

```
python3 word_freq.py stories/story-1.txt
python3 word_freq.py stories/story-2.txt
python3 word_freq.py stories/story-3.txt
```

Demo 3: parallel tasks 3_parallel.sh

```
# start tasks in background (& means send to background)
python3 word_freq.py stories/story-1.txt > words-1.txt &
python3 word_freq.py stories/story-2.txt > words-2.txt &
python3 word_freq.py stories/story-3.txt > words-3.txt &

# wait for background tasks to finish
wait
```

Part II: Demo 4 – for loops for parallel tasks

4_forloop.sh

```
#!/bin/bash

#SBATCH --job-name=4_forloop
#SBATCH --partition=super
#SBATCH --nodes=1
#SBATCH --time=00-00:01:00
#SBATCH --output=4_forloop.%j.out
#SBATCH --error=4_forloop.%j.err

module add python/3.6.4-anaconda
```

Runs on 1 Node for all 16 tasks

```
for i  in `seq 1 16`; do
    python3 word_freq.py "stories/story-${i}.txt" > "words-${i}.txt" &
done

wait
```

Part II: Demo 5 – job arrays for parallel jobs

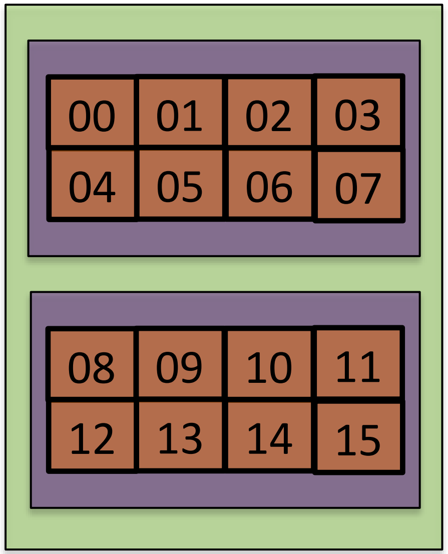
5_jobarray.sh

```
#!/bin/bash

#SBATCH --job-name=5_jobarray
#SBATCH --partition=super
#SBATCH --nodes=1
#SBATCH --array=1-16
#SBATCH --time=00-00:01:00
#SBATCH --output=multiple.%j.out
#SBATCH --error=multiple.%j.err

module add python/3.6.4-anaconda
```

Submits each task as a separate Job, to a separate node

```
python3 word_freq.py "stories/story-
${SLURM_ARRAY_TASK_ID}.txt"
       > "words${SLURM_ARRAY_TASK_ID}.txt" &
wait
```

Part II: BioHPC Cluster Architecture 101



🟩 Node

🟪 Socket

🟧 Core

* 2 logical cores/threads inside each physical core

Question: How many tasks can I submit to each node?

Answer: It depends…..

Part II: Constraints on a single-node job

Each task uses **X** CPU cores

Each task uses **Y** MB of RAM

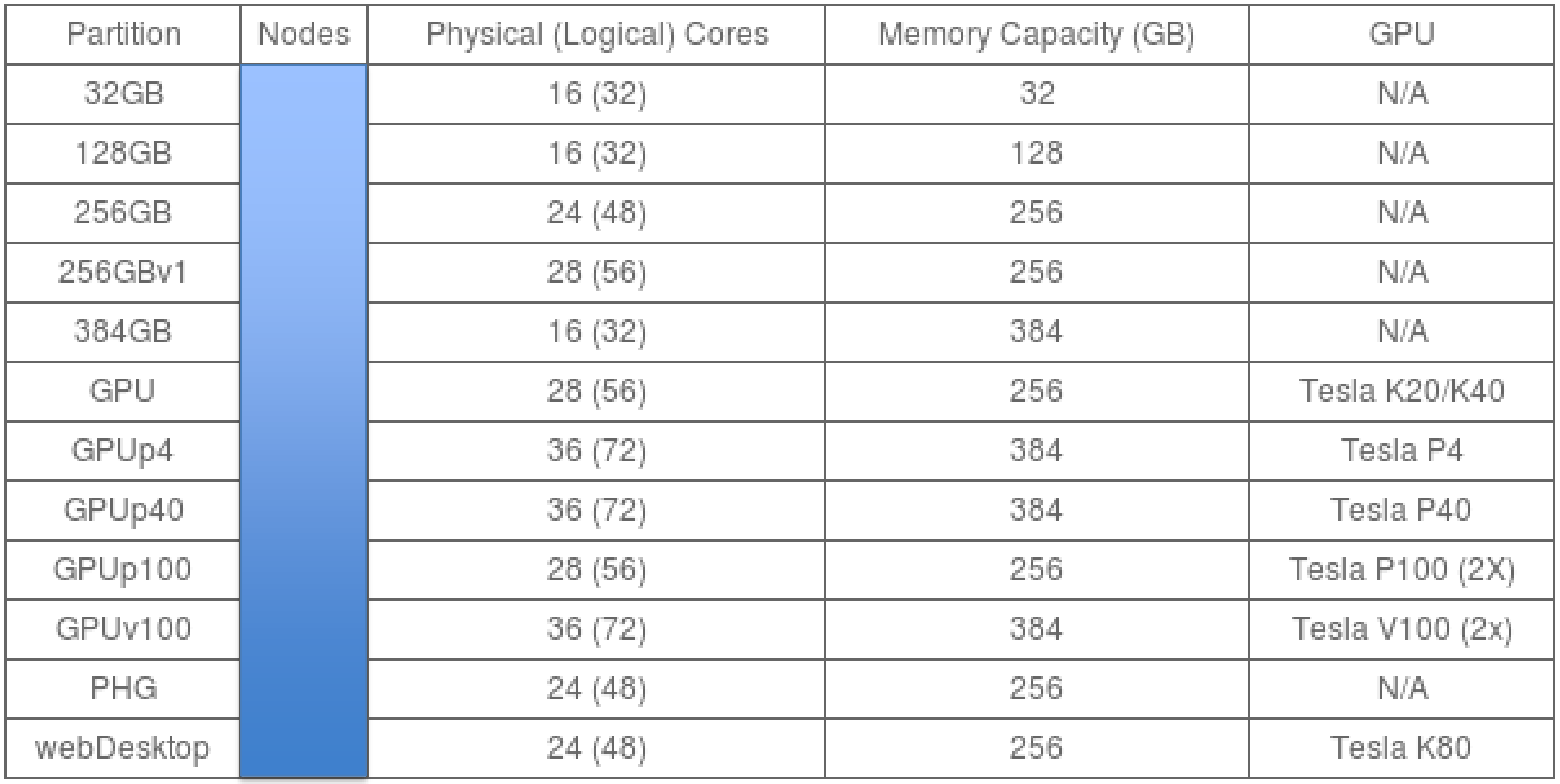
Total of **X** should fit within cores on node

Total of **Y** **must** fit within RAM on node

Sometimes you need to run fewer tasks than you have CPUs cores, to fit inside the RAM available.

UTSouthwestern Medical Center
Lyda Hill Department of Bioinformatics
BioHPC

Part II: Types of Nodes



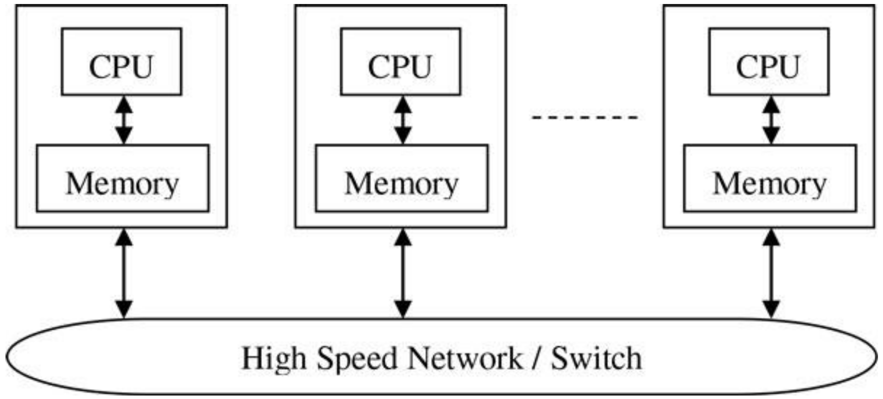
Partition	Nodes	Physical (Logical) Cores	Memory Capacity (GB)	GPU
32GB		16 (32)	32	N/A
128GB		16 (32)	128	N/A
256GB		24 (48)	256	N/A
256GBv1		28 (56)	256	N/A
384GB		16 (32)	384	N/A
GPU		28 (56)	256	Tesla K20/K40
GPUp4		36 (72)	384	Tesla P4
GPUp40		36 (72)	384	Tesla P40
GPUp100		28 (56)	256	Tesla P100 (2X)
GPUv100		36 (72)	384	Tesla V100 (2x)
PHG		24 (48)	256	N/A
webDesktop		24 (48)	256	Tesla K80

https://portal.biohpc.swmed.edu/content/about/systems/

Part III: Beyond Simple CPU Jobs

- GPU job on a single node

- MPI job on multiple nodes (distributed memory)



Part III: Submit a GPU job

6_gpu.sh

```
#!/bin/bash

#SBATCH --job-name=6_gpu
#SBATCH --partition=GPU
#SBATCH --gres=gpu:1
#SBATCH --time=0-00:10:00
#SBATCH --output=6_gpu.%j.out
#SBATCH --error=6_gpu.%j.err

module add cuda80

./matrixMul -wA=320 -hA=10240 -wB=10240 -hB=320
```

Jobs will not be allocated any generic resources unless specifically requested at job submit time.

Using the –gres option supported by sbatch and srun. Format: --gres=gpu:[n], where n is the number of GPUs

Use GPU partition

A(320, 10240) × B(10240, 320)

Simple CUDA matrix multiplication task

Part III: Demo 7 – Submit an MPI job

7_mpi.sh

- MPI jobs run the same MPI program multiple times, on 1 or more nodes.
- Each instance, or **rank** of the program carries out part of the computation.
- The **ranks** communicate during processing. Usually the **rank 0** process collects the result.
- To run an MPI job we need to:

  1. Obtain an allocation of 1 or more nodes;
  2. Make sure each rank will know how to reach the others;
  3. Start the right number of instances of the program on each of the nodes in the allocation.

Part III: Demo 7 – Submit an MPI job

7_mpi.sh

```
#!/bin/bash

#SBATCH --job-name=7_mpi
#SBATCH --partition=super
#SBATCH --nodes=4
#SBATCH --ntasks=16
#SBATCH --time=0-1:00:00
#SBATCH --output=7_mpi.%j.out

# setting a ulimit will rid of many OpenMPI warnings
ulimit -s 10240

module add openmpi/intel/3.1.1

srun ./mpi_pi
```

16 tasks across 4 nodes
(4 tasks per node)

UT Southwestern Medical Center
Lyda Hill Department of Bioinformatics

BioHPC

Questions?

UTSouthwestern Medical Center
Lyda Hill Department of Bioinformatics | BioHPC

ADVANCED DEMOS/SELF STUDY

Extras: Demo A -- submit multiple jobs to single node with srun

```
#!/bin/bash

#SBATCH --job-name=srunSingleNodeMatlab
#SBATCH --partition=super
#SBATCH --nodes=1
#SBATCH –ntasks=16
#SBATCH --time=00-00:01:00
#SBATCH --output=srunSingleNode.%j.out
#SBATCH --error=srunSingleNode.%j.err

module add matlab

srun sh script.sh
```

- Total number of tasks in the current job

SLURM_LOCALID = [0 31] for super

SLURM_LOCALID = [0 47] for 256GB

- **SLURM_LOCALID** : environment variable; Node local task ID for the process within a job. (zero-based)

script.sh

```
#!/bin/bash
matlab –nodisplay –nodesktop -singleCompThread –r "forBiohpcTestPlot($SLURM_LOCALID+1), exit"
```

Extras: Demo B -- submit multiple jobs to multi-node with srun

```
#!/bin/bash

#SBATCH --job-name=srun2NodeMatlab
#SBATCH --partition=super
#SBATCH --nodes=2
#SBATCH –ntasks=16
#SBATCH --time=00-00:01:00
#SBATCH --output=srun2Node.%j.out
#SBATCH --error=srun2Node.%j.err

module add matlab

srun sh script.sh
```

- **SLURM_NODEID** : the relative node ID of the current node (zero-based)
- **SLURM_NNODES** : Total number of nodes in the job's resource allocation
- **SLURM_NTASKS** : Total number of tasks in the current job

- **SLURM_LOCALID** : environment variable; Node local task ID for the process within a job. (zero-based)

script.sh

```
#!/bin/bash

let "ID=$SLURM_NODEID*$SLURM_NTASKS/$SLURM_NNODES+$SLURM_LOCALID+1"
echo "process data $ID on" `hostname`>>namelist.txt
matlab -nodisplay -nodesktop -singleCompThread -r "forBiohpcTestPlot($ID), exit"
```

Extras : GNU parallel – The Problem

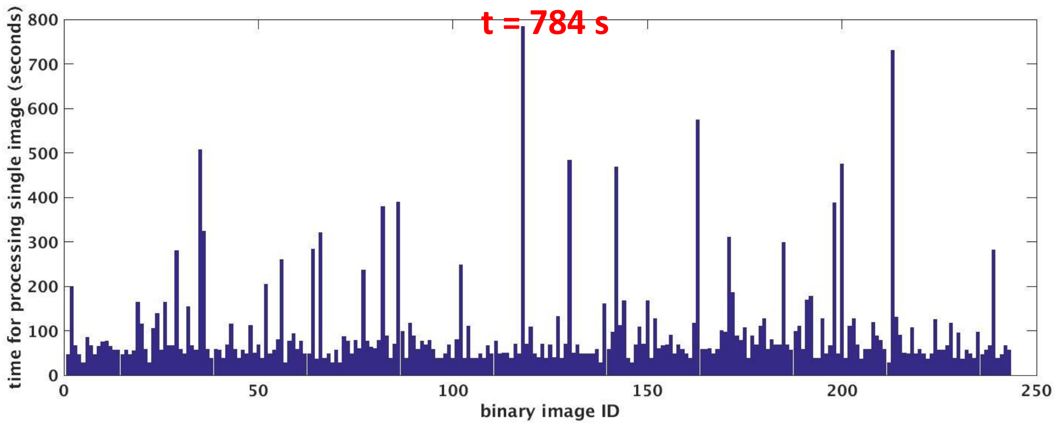
- **Number of tasks > 512**

  16 (maximum running nodes/user) * 32(maximum tasks/node) = 512

- **Input Data is a collection of unorganized files**

  build up connection between filename and SLURM environment variables is not straightforward

- **Unbalanced tasks**



A work distribution algorithm is needed to assign ready tasks to idle threads as efficiently as possible.

Extras : GNU parallel – The Solution

A shell tool for executing jobs in parallel using one or more computers.

http://www.gnu.org/software/parallel/

- **Run the same command on lot of files or tasks**

- **Do it in parallel, but do not run them all in parallel, run only N at the same time**

  simultaneously as it will slow down the computer

  -or-

  exceed the per user node limitation

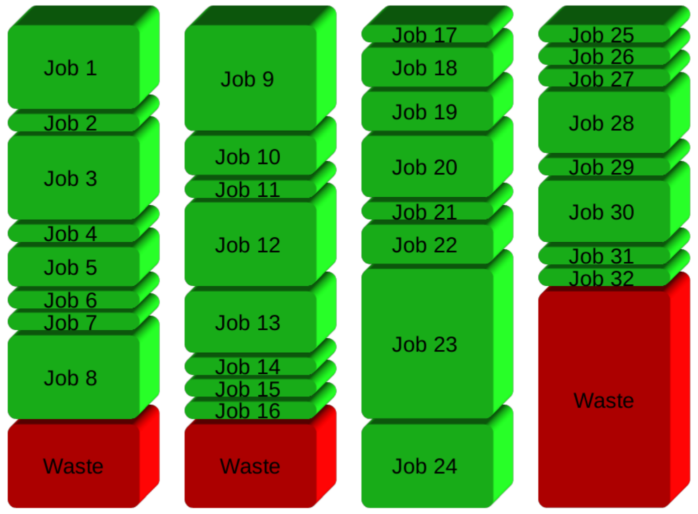
- **Example**

  ```
  parallel -j 32 gzip *
  ```

  (with 32 jobs in parallel)

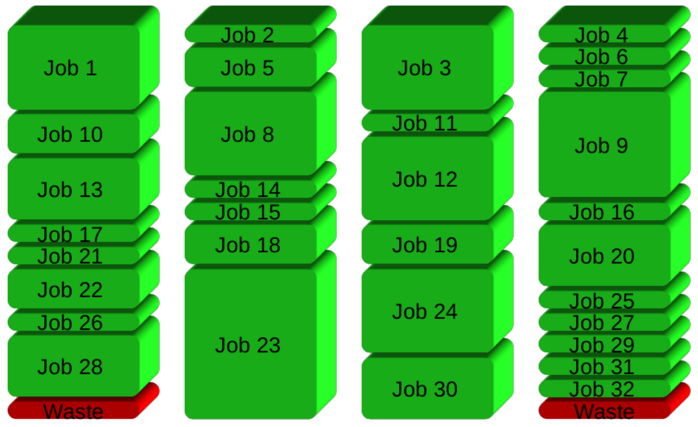
Extras : GNU parallel – Performance

sbatch→srun



parallelize is to run 8 jobs on each CPU

sbatch→parallel→srun



GNU Parallel instead spawns a new process when one finishes - keeping the CPUs active and thus saving time

Usage: sbatch → GNU parallel → srun → base scripts

* Image retrieved from https://www.biostars.org/p/63816/

UTSouthwestern Medical Center
Lyda Hill Department of Bioinformatics
BioHPC

Extras: Demo D -- submit multiple jobs to multi-node with srun & GNU parallel

```
#!/bin/bash

#SBATCH --job-name=parallelExample
#SBATCH --partition=super
#SBATCH --nodes=2
#SBATCH –ntasks=56
#SBATCH --time=1-00:00:00

CORES_PER_TASK=1
INPUTS_COMMAND="ls BWimages"
TASK_SCRIPT="single.sh"

module add parallel

SRUN="srun --exclusive -n -N1 -c $CORES_PER_TASK"
PARALLEL="parallel --delay .2 -j $SLURM_NTASKS -joblog task.log"

eval $INPUTS_COMMAND | $PARALLEL $SRUN sh $TASK_SCRIPT {}
```

- **NTASKS:** size of N <= nodes * core/node

- **INPUTS_COMMAND:** generate a input file list (job pool)
- **TASK_SCRIPT:** base script

single.sh

```
#!/bin/bash

module add matlab/2015a

matlab –nodisplay –nodesktop -singleCompThread –r "fastMatching('$1'), exit"
```