



# RC Primer: Incredibly Easy Parallelization with High Throughput Computing

# Incredibly Easy Parallelization with High Throughput Computing

Date: October 17, 2024

Instructor: Andrew Monaghan

Contributors: Layla Freeborn, Trevor Hall,  
Brandon Reyes

- Website: [www.rc.colorado.edu/rc](http://www.rc.colorado.edu/rc)
- Documentation: <https://curc.readthedocs.io>
- Helpdesk: [rc-help@colorado.edu](mailto:rc-help@colorado.edu)
- Survey: <http://tinyurl.com/curc-survey18>



**Slides**

[https://github.com/ResearchComputing/easy\\_parallelization\\_htc\\_primer](https://github.com/ResearchComputing/easy_parallelization_htc_primer)



# Meet the User Support Team



Layla  
Freeborn



Brandon  
Reyes



Andy  
Monaghan



Michael  
Schneider



John  
Reiland



Dylan  
Gottlieb



Mohal  
Khandelwal



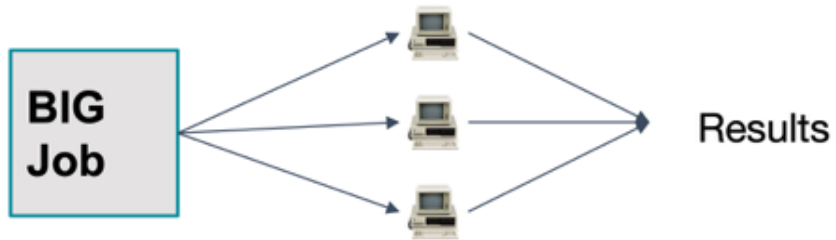
Ragan  
Lee

# Learning Objectives and Outline

- Introduction to high throughput computing (HTC)
- HTC method 1: Job Arrays
- HTC method 2: Load Balancer
- HTC method 3: GNU Parallel
- Open Science Grid

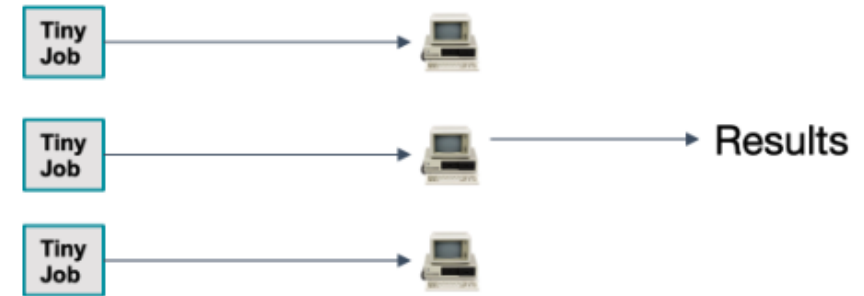
# What is High Throughput Computing (HTC)?

Internal parallelization



Example: Climate Model

External parallelization



Example: Image processing

**This is HTC!**

# Three HTC methods for Alpine

	Job Arrays	Load Balancer	GNU Parallel
<b>Job length?</b>	Best for tasks > 10 min	Best for tasks < 30 min	Best for tasks < 30 min
<b>Where can I use?</b>	Alpine, other HPC resources	Alpine, other HPC resources	Alpine or your laptop/desktop
<b>Max jobs in queue?</b>	1000 across all arrays/jobs; each array member counts as a job	No	No
<b>Max cores per job?</b>	None; works well on one or multiple nodes	None; works well on one or multiple nodes	None but works best on one node/machine
<b>Max cores per task?</b>	None	None but most commonly used for serial tasks	None but most commonly used for serial tasks
<b>Does it reserve a controller core?</b>	No	Yes	No
<b>Other features?</b>	Easy to adapt a "regular" job script to accommodate job arrays	Works well for input files with heterogeneous names ; easy to set up multi-node jobs	Great for replacing/speeding up loops; Can pick up where you left off if job times out

Now let's try some examples....

# Logging into CU Research Computing

login to CURC via your terminal:

```
$ ssh monaghaa@login.rc.colorado.edu
```

...or login to CURC via your browser:

<https://ondemand-rmacc.rc.colorado.edu>

(once logged in, navigate to **Clusters -> Alpine shell**)

*Additional information:*

<https://curc.readthedocs.io/en/latest/access/logging-in.html>

<https://curc.readthedocs.io/en/latest/gateways/OnDemand.html>



# Download the example files

```
]$ cd /scratch/alpine/$USER  
]$ git clone https://github.com/ResearchComputing/easy\_parallelization\_htc\_primer  
]$ cd easy_parallelization_htc_primer/examples  
]$ ls  
cars_mpg_array.sh      cars_mpg_input_args_array.txt      cars_mpg_input_commands_lb.txt cars_mpg.py  
cars_mpg_gnuparallel.sh cars_mpg_input_commands_gnuparallel.txt cars_mpg_lb.sh  
  
]$ cat cars_mpg.py  
import sys  
  
car=sys.argv[1]  
mpg=sys.argv[2]  
  
print("The " + car + " gets " + mpg + " mpg.")
```



# Method 1 Example: Job Arrays

*cars\_mpg\_array.sh* (job script)

```
#!/bin/bash
#SBATCH --time=00:00:10
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --job-name=cars
#SBATCH --output=cars.%A_%a.out
#SBATCH --array=1-5

# load anaconda python
source /curc/sw/anaconda/default

# run workflow
python cars_mpg.py $(sed -n "${SLURM_ARRAY_TASK_ID}p" cars_mpg_input_args_array.txt)
```

← Only request resources for 1 member

← This flag makes the array

*To run this example :*

```
]$ sbatch --reservation=htc
cars_mpg_array.sh
```

*cars\_mpg\_args\_array.txt*  
(arguments)

```
mustang 25
pinto 30
chevette 33
nova 21
cutlass 23
```

# Method 2 Example: Load Balancer

*cars\_mpg\_lb.sh* (job script)

```
#!/bin/bash
#SBATCH --time=00:00:10
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --nodes=1 # optional
#SBATCH --ntasks=6
#SBATCH --job-name=cars
#SBATCH --output=cars.%j.out
```

Request resources  
for all members

```
# load loadbalancer and anaconda python
module purge
module load loadbalance
source /curc/sw/anaconda/default
```

Load the  
'loadbalance'  
module

```
# run workflow
mpirun lb cars_mpg_input_commands_lb.txt
```

To run this example :

```
]$ sbatch --reservation=htc cars_mpg_lb.sh
```

*cars\_mpg\_commands\_lb.txt*  
(arguments)

```
python cars_mpg.py mustang 25
python cars_mpg.py pinto 30
python cars_mpg.py chevette 33
python cars_mpg.py nova 21
python cars_mpg.py cutlass 23
```

# Method 3 Example: GNU Parallel

*cars\_mpg\_gnuparallel.sh* (job script)

```
#!/bin/bash
#SBATCH --time=00:00:10
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --nodes=1 # optional
#SBATCH --ntasks=5
#SBATCH --job-name=cars
#SBATCH --output=cars.%j.out

# load gnu parallel and anaconda python
module purge
module load gnu_parallel
source /curc/sw/anaconda/default

# run workflow
parallel -j $SLURM_NTASKS <cars_mpg_input_commands_gnuparallel.txt
```

Request resources  
for all members

Load the  
'gnu\_parallel'  
module

*To run this example :*

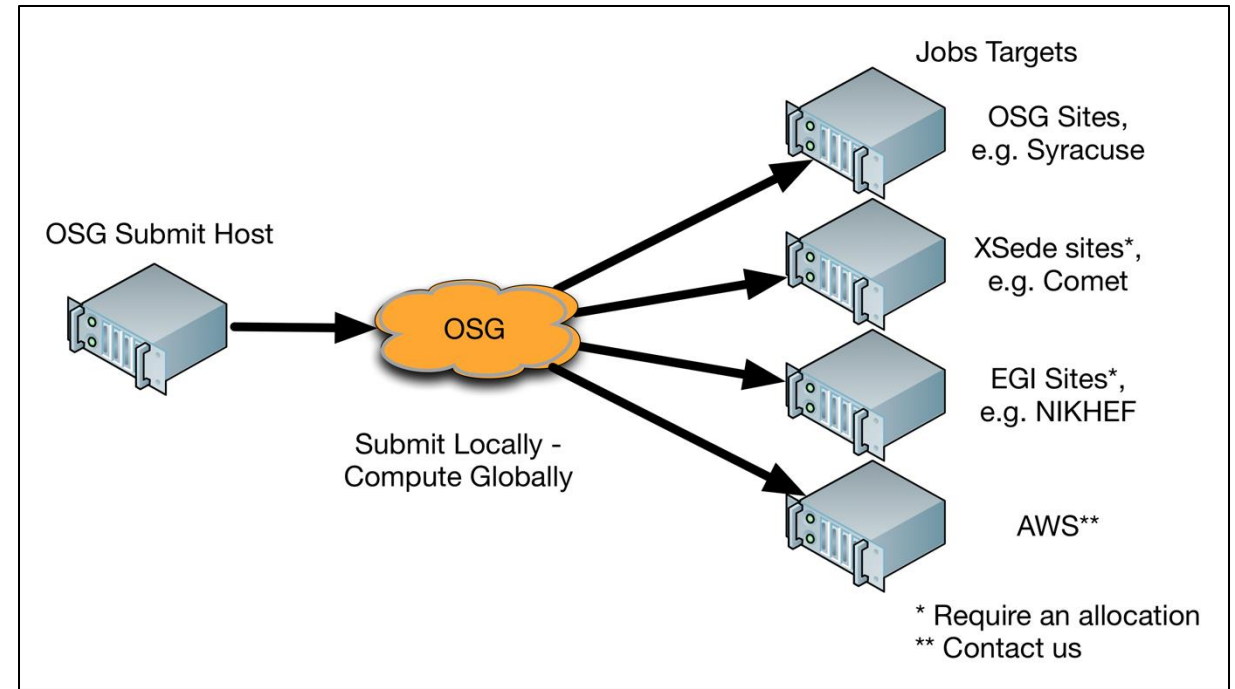
```
]$ sbatch --reservation=htc  
cars_mpg_gnuparallel.sh
```

*cars\_mpg\_commands\_gnuparallel.txt*  
(arguments)

```
python cars_mpg.py mustang 25  
python cars_mpg.py pinto 30  
python cars_mpg.py chevette 33  
python cars_mpg.py nova 21  
python cars_mpg.py cutlass 23
```

# A Community HTC Resource beyond Alpine: Open Science Grid (OSG)

- NSF/DOE-funded service (free!)
- Open to any U.S.-based researcher
- 125 institutions sharing spare computing cycles
- ~1 Billion core hours per year used
- Can get started quickly. Options for dedicated allocation if needed.
- <https://osgconnect.net>



Source: <https://swc-osg-workshop.github.io/OSG-UserTraining-JLab-2019/materials/AHM/01-IntroGrid.html>



# Thank you!

## Survey and feedback

<http://tinyurl.com/curc-survey18>

