

# High Throughput Computing on RMACC Summit and Beyond

Andrew Monaghan

University of Colorado Boulder

[https://github.com/ResearchComputing/HTC\\_Short\\_Course\\_Fall\\_2019](https://github.com/ResearchComputing/HTC_Short_Course_Fall_2019)

*Adapted from slides and examples by Aaron Holt, CURC*

# Be Boulder.



University of Colorado **Boulder**

# Please Sign in:

<https://tinyurl.com/yxp2ul9u>

# Login and download course materials

If you have a Summit account:

```
ssh -X <identikey>@login.rc.colorado.edu
```

If you don't have an account:

```
ssh -X user00NN@login1.rc.colorado.edu
```

*(for latter case, I will provide you with the “NN” and password)*

...Once you are logged in, type the following three commands:

```
ssh scompile
```

```
cd /scratch/summit/$USER
```

```
git clone https://github.com/ResearchComputing/HTC_Short_Course_Spring_2019
```



# What is HTC?

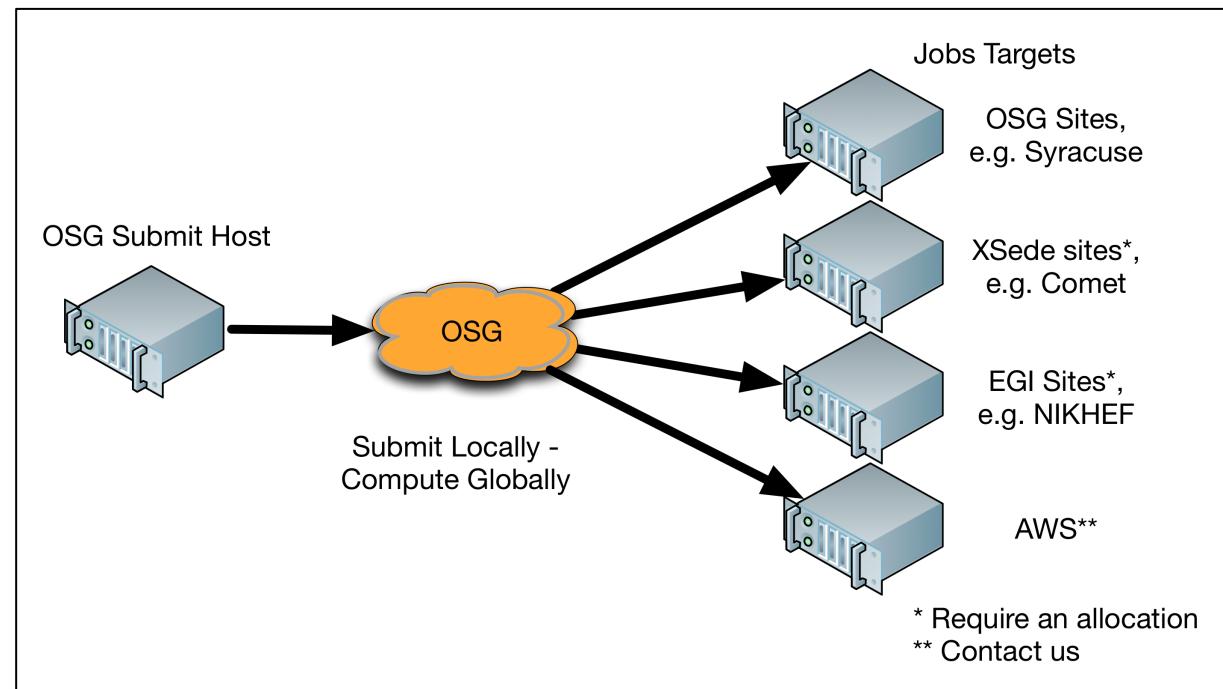
- High throughput computing (HTC) recognizes that most scientists work on research timescales.
- They value how many computing cycles they can get over a period of weeks-to-months rather than a period of hours-to-days.
- Processor speed not a concern – volume is more important.
- Processor type/homogeneity not important either.
- Typical workflow may have hundreds-to-thousands of jobs.
- Often comprised of many serial tasks that can be run independently.
- Goal is to submit lots of jobs with minimal effort and oversight.

<http://research.cs.wisc.edu/htcondor/htc.html>

# Example of ‘pure’ HTC resource

## 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>



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

# Can I do HTC on RMACC Summit?

- Not quite at the scale of OSG.
- But you can “wrap up” 100s-to-1000s of jobs and submit them all in one job script to any of the partitions/qos’s on Summit
- Depending on the timescales of your workflow, this might also be called **“many task computing” (MTC)**, which focuses on lots of tasks over shorter timescales.

# How can I do HTC/MTC on Summit?

There are two primary ways, and we will cover both today:

## 1. Job arrays

- Submit one Slurm job script that submits **X jobs simultaneously**
  - Great for lots of tasks that have heterogeneous run times
  - Each job usually runs a serial task but can be a multi-core task if needed
  - Jobs run as cores become available
  - ...works best for longer jobs (> 30 min)
  - Summit limits: Maximum array size of 1001; 100 jobs can run at once in ucb-general

## 2. GNU Parallel \*\*or\*\* CURC Load Balancer

- Submit one Slurm job script that runs **X tasks** on N requested cores
  - All under one job that works through all X tasks in order, running N at a time (e.g., X=10,000 tasks on N=24 cores)
  - Great for lots of tasks that may have similar run times
  - ...works best for shorter jobs (< 30 min)

# Job Arrays

- It is easy to adapt an existing job script to run as an array of jobs.
  - Add the following directive: `#SBATCH --array=n-N`
    - E.g., `#SBATCH --array=1-200`
    - You can also specify unevenly spaced arrays: `#SBATCH --array=1,4,5,9,22`
  - You can then just submit your job and it will run from n to N.
  - If your jobs depend on the array index, you can use the environment variable `$SLURM_ARRAY_TASK_ID` to invoke the present array number.
    - E.g., typing “`python myscript.py file_{$SLURM_ARRAY_TASK_ID}.csv`” will run a python script over the array, each time processing a different file.



# Example 1: job\_array\_ex1.sh

```
#!/bin/bash

#SBATCH --job-name=htcjob
#SBATCH --output=outfile/htcjob.%A_%a.out
#SBATCH --partition=shas
#SBATCH --qos=normal
#SBATCH --time=00:00:05
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --array=1-12

# Run your program
echo #####
echo "start array job ${SLURM_ARRAY_TASK_ID}"
echo "This job is performing $SLURM_NTASKS tasks"
echo "This job has reserved $SLURM_CPUS_PER_TASK cores for $SLURM_NTASKS tasks"
JOBMEM=$(echo "scale=2; $SLURM_CPUS_PER_TASK*$SLURM_MEM_PER_CPU/1000" | bc)
echo "Each task has $JOBMEM GB of memory allocated to it"
```



# Example 1: job\_array\_ex1.sh

```
#!/bin/bash
```

```
#SBATCH --job-name=htcjob
#SBATCH --output=outfiles/htcjob.%A_%a.out
#SBATCH --partition=shas
#SBATCH --qos=normal
#SBATCH --time=00:00:05
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --array=1-12
```

```
# Run your program
echo #####
echo "start array job ${SLURM_ARRAY_TASK_ID}"
echo "This job is performing $SLURM_NTASKS tasks"
echo "This job has reserved $SLURM_CPUS_PER_TASK cores for $SLURM_NTASKS tasks"
JOBMEM=$(echo "scale=2; $SLURM_CPUS_PER_TASK*$SLURM_MEM_PER_CPU/1000" | bc)
echo "Each task has $JOBMEM GB of memory allocated to it"
```

Let's run the example:

```
sbatch job_array_ex1.sh --reservation=htc
```



# Example 2: job\_array\_ex2.sh

- Example 2 is an extension of Example 1
- We load python and R modules and then add lines to call python and R scripts
  - Both scripts read in array-ordered files from the “`./infiles`” directory as command-line arguments, and print the output to the “`./outfiles`” directory.
- Examine `job_array_ex2.sh`  
`more job_array_ex2.sh`
- Run `job_array_ex2.sh`  
`sbatch job_array_ex2.sh --reservation=htc`

# GNU Parallel

- GNU parallel is a shell tool for executing tasks in parallel using one or more computers.
  - In its simplest form, GNU parallel is a parallel replacement of a **do** or **for** loop.
- Options to specify how many tasks should run in parallel, display output in order, delay starts, limit resources and more!
- Let's get started with some command line examples. Start an interactive job:
  - `sinteractive --ntasks=12 --reservation=htc`
  - `ml gnu_parallel`



# GNU Parallel Useful Options

- View what commands parallel will run without executing them:
  - `seq 10 | parallel --dry-run echo {}`
- Limit number of tasks running at one time:
  - `seq 10 | parallel -j 2 echo {}`
- Stagger start times for processes to avoid I/O overload:
  - `seq 10 | parallel --delay 2.0 echo {}`
- See all the options:
  - `man parallel`



# Let's try working with files

- Let's run our python script on all of the input files in `./infiles`:
  - `find infiles/*.txt | parallel python python_array_test.py {}`
- Or we can read the commands and arguments from a file
  - `find infiles/*.txt | parallel --dry-run python python_array_test.py {} > commands.txt`
  - `parallel < commands.txt`
-

# Example: gnu\_parallel\_ex1.sh

- Job script is similar to job scripts for job arrays except:
  - Does not have the `#SBATCH --array=n-N` flag
  - Requires loading the `gnu_parallel` module
  - You will likely increase `--ntasks` because you want to spread tasks across many cores
  - All standard output goes to a single `*.out` file (unless you redirect output)
- This example does the same tasks as `job_array_ex2.sh`:
  - We load python and R modules call python and R scripts
    - Both scripts read in files from the “`./infiles`” directory as command-line arguments, and print the output to the “`./outfiles`” directory.
- Examine job script

```
more gnu_parallel_ex1.sh
```
- Run job script

```
sbatch gnu_parallel_ex1.sh --reservation=htc
```



# CURC Load Balancer

- In-house tool for executing tasks in parallel using one or more RMACC Summit nodes.
- Very simple to use:
  - Step 1: Create file that lists tasks you want to run
    - One task per line
    - If multiple commands per task can separate each command with semi-colon “;”
    - ...or if each task is a complex list of commands, can create a bash script for each task
  - Step 2: load modules and execute using “`mpirun lb commands.txt`”
- Let's try an example...



# Example: load\_balance\_ex1.sh

- Job script is similar to job scripts for job arrays except:
  - Does not have the `#SBATCH --array=n-N` flag
  - Requires loading the `intel`, `impi` and `loadbalance` modules
  - You will likely increase `--ntasks` because you want to spread tasks across many cores
  - All standard output goes to a single `*.out` file (unless you redirect output)
- This example does the same tasks as `gnu_parallel_ex1.sh`:
  - We load python and R modules call python and R scripts
    - Both scripts read in files from the “`./infiles`” directory as command-line arguments, and print the output to the “`./outfiles`” directory.
    - We will reuse the “`commands.txt`” file we created during the gnu parallel exercises.
- Examine job script  
`more load_balance_ex1.sh`
- Run job script  
`sbatch load_balance_ex1.sh --reservation=htc`



# Which HTC/MTC tool should I use?

	Job Arrays	GNU Parallel	Load Balancer
<b>Job length?</b>	Best for jobs > 30 min	Best for jobs < 30 min	Best for jobs < 30 min
<b>Where can I use?</b>	RMACC Summit, other HPC resources	RMACC Summit or your laptop/desktop	RMACC Summit, other HPC resources
<b>Max jobs in queue?</b>	1001 (per array)	No	No
<b>Max cores per job?</b>	None; works well one one or multiple nodes	None but works best on one node/machine	None; works well one one or multiple nodes
<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	No	Yes
<b>Other features?</b>	Easy to adapt a "regular" job script to accommodate job arrays	Great for replacing/speeding up loops; Can pick up where you left off if job times out; commonly found on any HPC system	Works well for input files have heterogeneous names ; easy to set up multi-node jobs



# Other HTC/MTC resources

- On Blanca: If you belong to a group with a Blanca node, this is a fantastic place to do HTC/MTC!
- On OSG (if you grow beyond Summit)  
<https://opensciencegrid.org/about/introduction/>  
(Need help getting started on OSG? Contact me)

# The end

- We'd love your feedback: <http://tinyurl.com/curc-survey18>
- CURC Load Balancer documentation  
<https://curc.readthedocs.io/en/latest/software/loadbalancer.html>
- GNU Parallel documentation  
<https://curc.readthedocs.io/en/latest/software/GNUParallel.html>  
[https://www.gnu.org/software/parallel/parallel\\_tutorial.html](https://www.gnu.org/software/parallel/parallel_tutorial.html)
- Slurm job arrays  
[https://slurm.schedmd.com/job\\_array.html](https://slurm.schedmd.com/job_array.html)
- I would be happy to help you get started with HTC/MTC on Summit or OSG  
[andrew.monaghan@colorado.edu](mailto:andrew.monaghan@colorado.edu)