HIGH-THROUGHPUT COMPUTING
AND YOUR RESEARCH

Brian Bockelman, Associate Scientist
Morgridge Institute for Research
October 17, 2019

Based on slides from Christina Koch, CHTC

LARGE SCALE COMPUTING

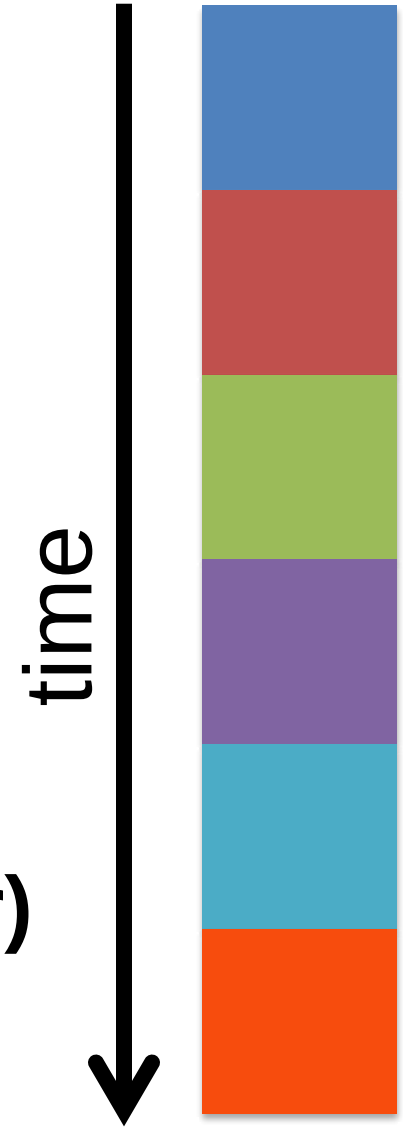
What is large-scale computing?

larger than ‘desktop’

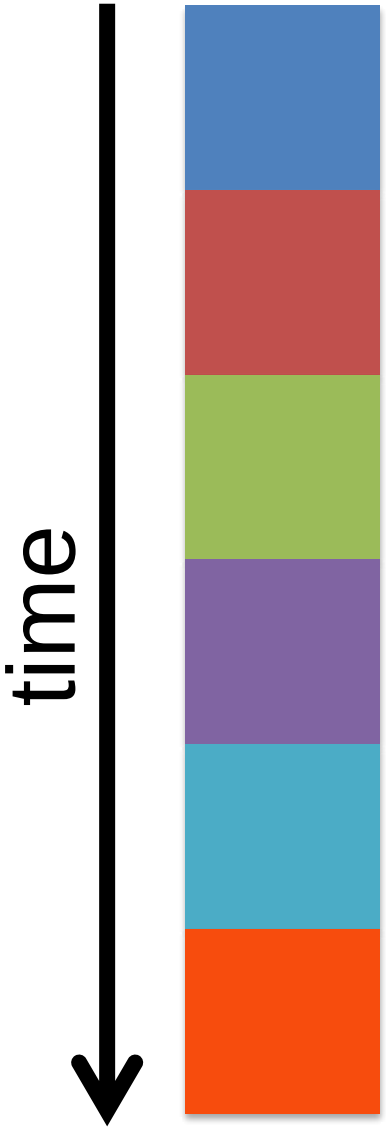
(in memory, data, processors)

Problem:
running many
computations
takes a long time

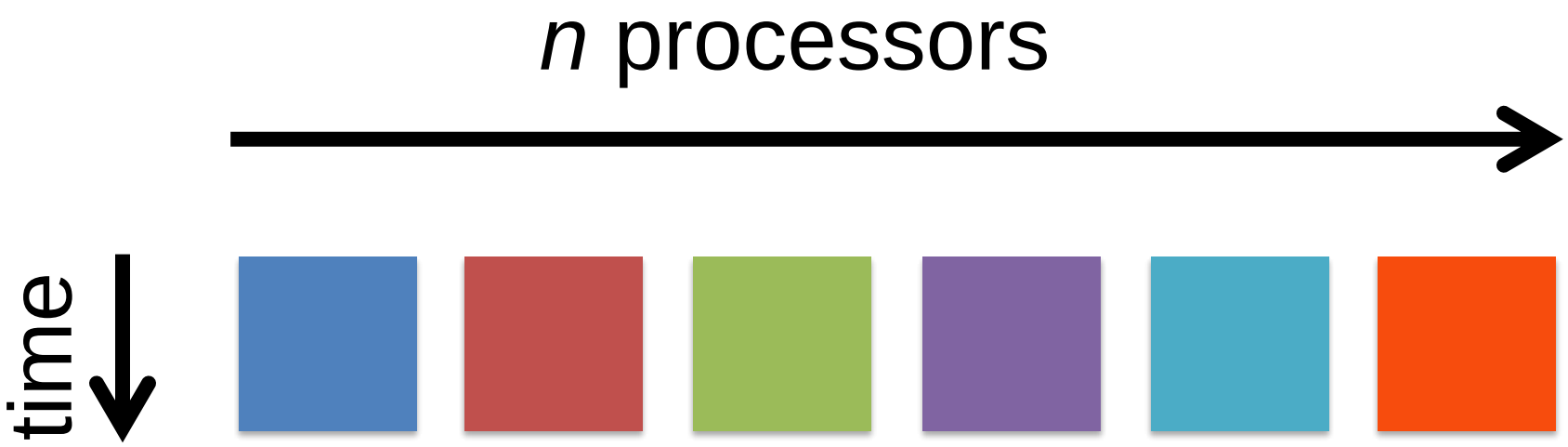
(running on one processor)



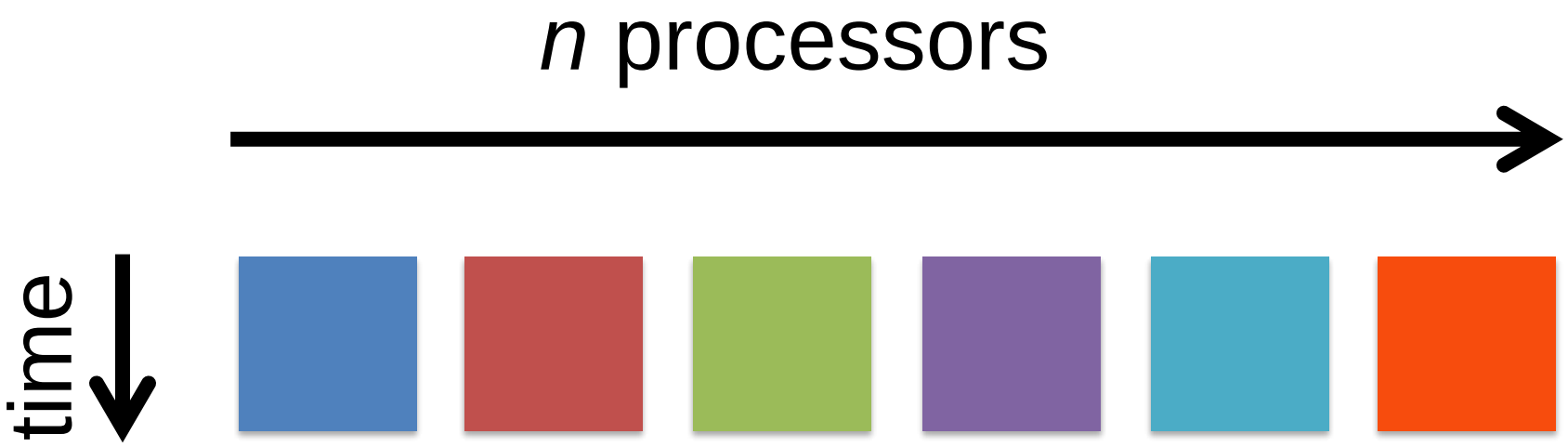
So how do you speed things up?



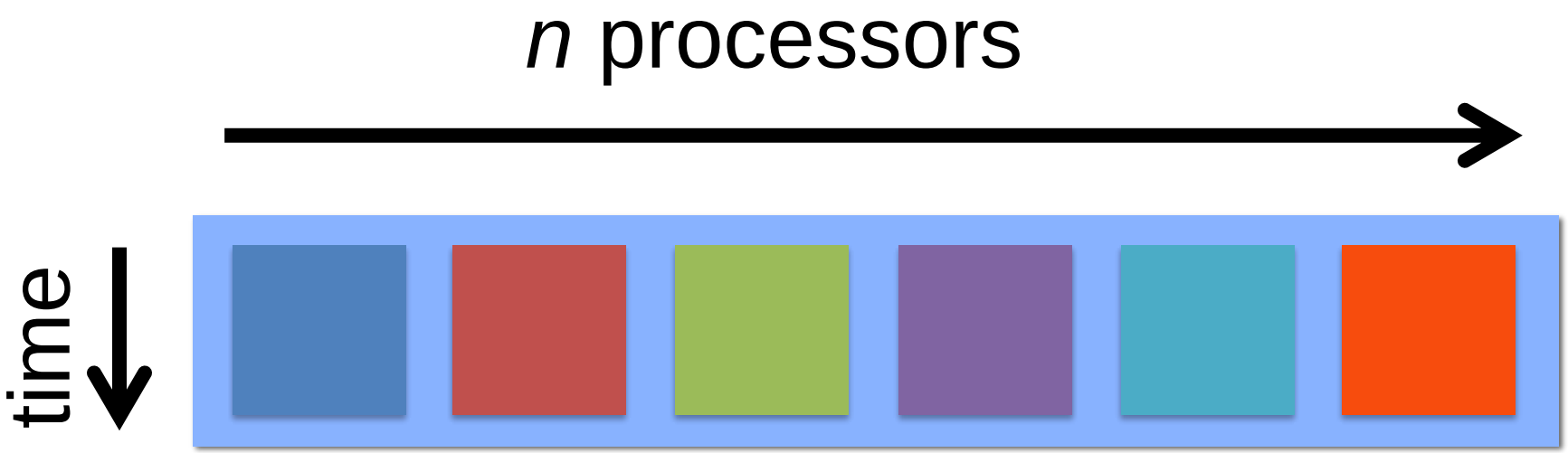
Break up the work!
Use more processors! (parallelize)



High ***throughput*** computing



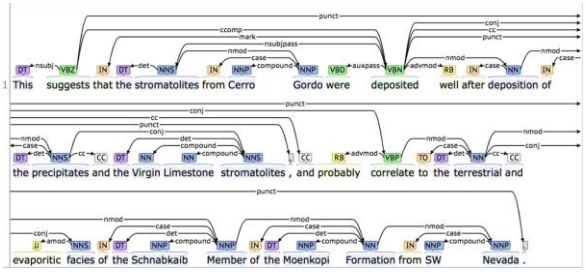
High ***performance*** computing



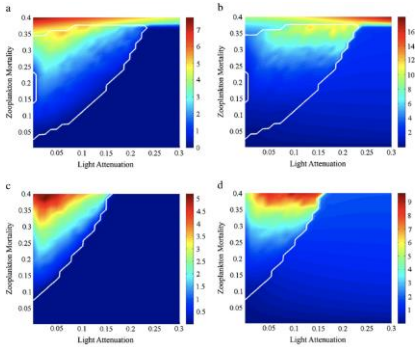
An HTC Analogy

HTC and the World's Largest Cake

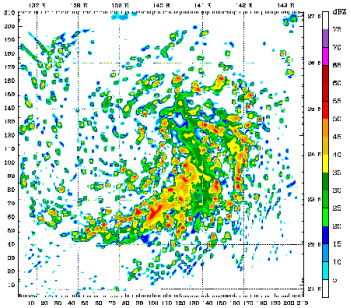
HTC Examples



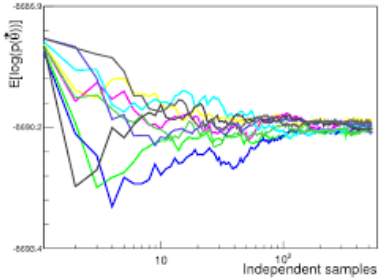
text analysis (most genomics …)



parameter sweeps



multi-start simulations



statistical model optimization (MCMC, numerical methods, etc.)

(multi-)image and sample analysis

HIGH THROUGHPUT COMPUTING

High Throughput Examples

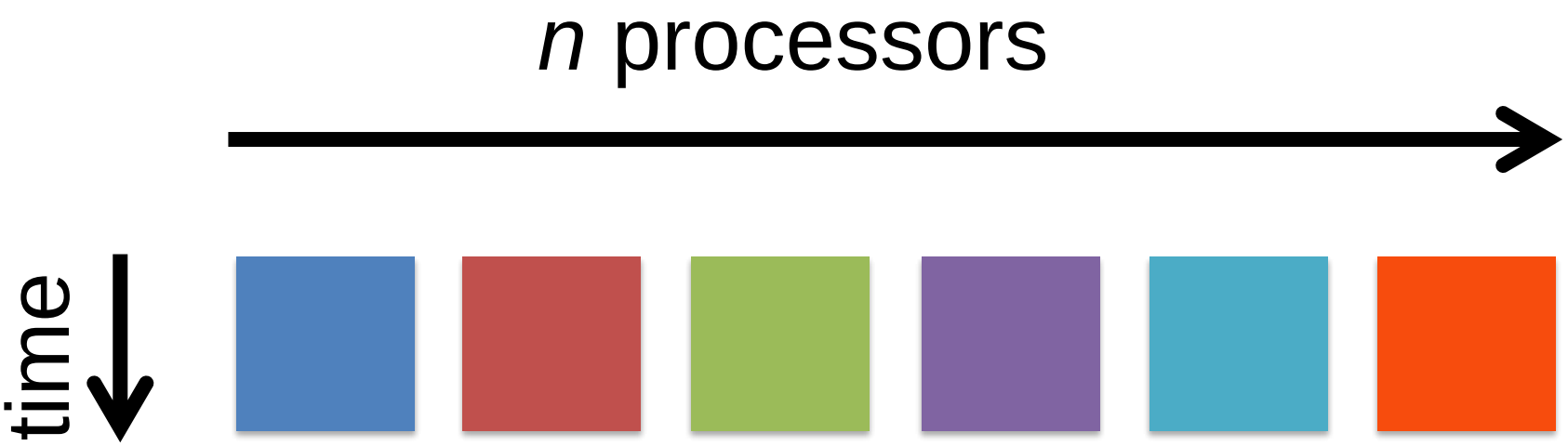
- Test many parameter combinations
- Analyze multiple images or datasets
- Do a replicate/randomized analysis
- Align genome/RNA sequence data

The Philosophy of HTC

- **Break work into many ‘smaller’ jobs**
  - single or few CPUs, short run times, smaller input and output per job
- **Run on as many processors as possible**
  - smaller and shorter jobs are best
  - take dependencies with you (like R)
- **Automate as much as you can**
  - black box programs that use various input files
  - numbered files
- **Scale up gradually**

CENTER FOR HIGH THROUGHPUT COMPUTING

High ***throughput*** computing



WHAT WE NEED

Lots of computers, to run multiple independent computations

CHTC Services

Center for **H**igh **T**hroughput **C**omputing, est. 2006

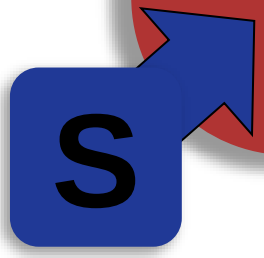
- **Large-scale, campus-shared computing systems**
  - high-throughput computing (HTC) and high-performance computing (HPC) systems
  - **all standard services provided free-of-charge**
  - hardware buy-in options
  - support and training for using our systems
  - proposal assistance
  - **chtc.cs.wisc.edu**

CHTC Accessible Computing

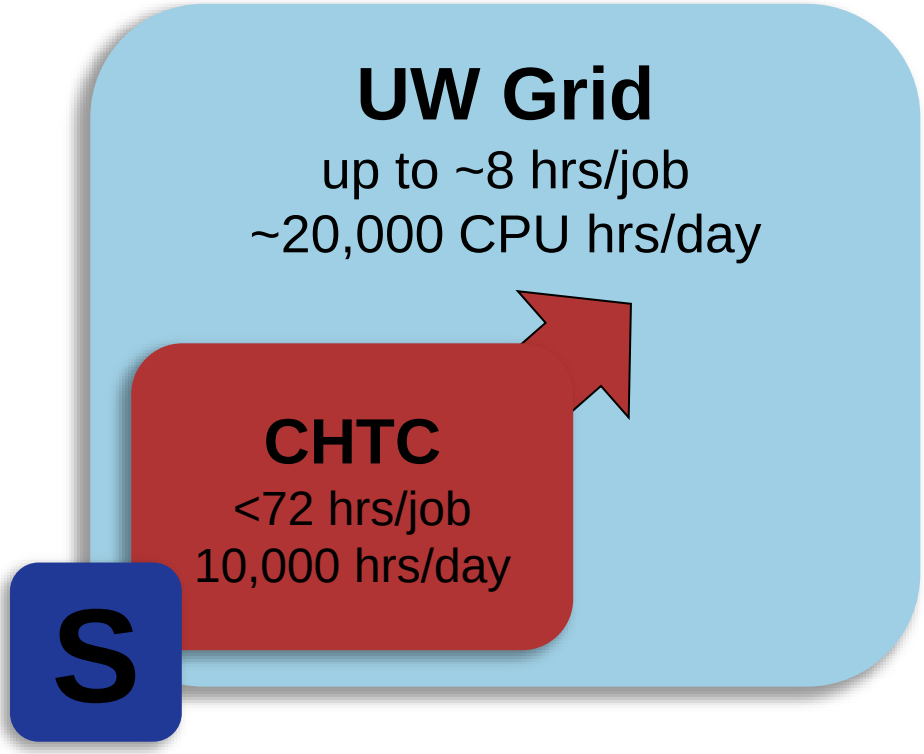
CHTC

time limit: <72 hrs/job

10,000+ CPU hrs/day



CHTC Accessible Computing



CHTC Accessible Computing

Open Science Grid
up to ~4 hrs/job
~200,000 CPU hrs/day

UW Grid
up to ~8 hrs/job
~20,000 CPU hrs/day

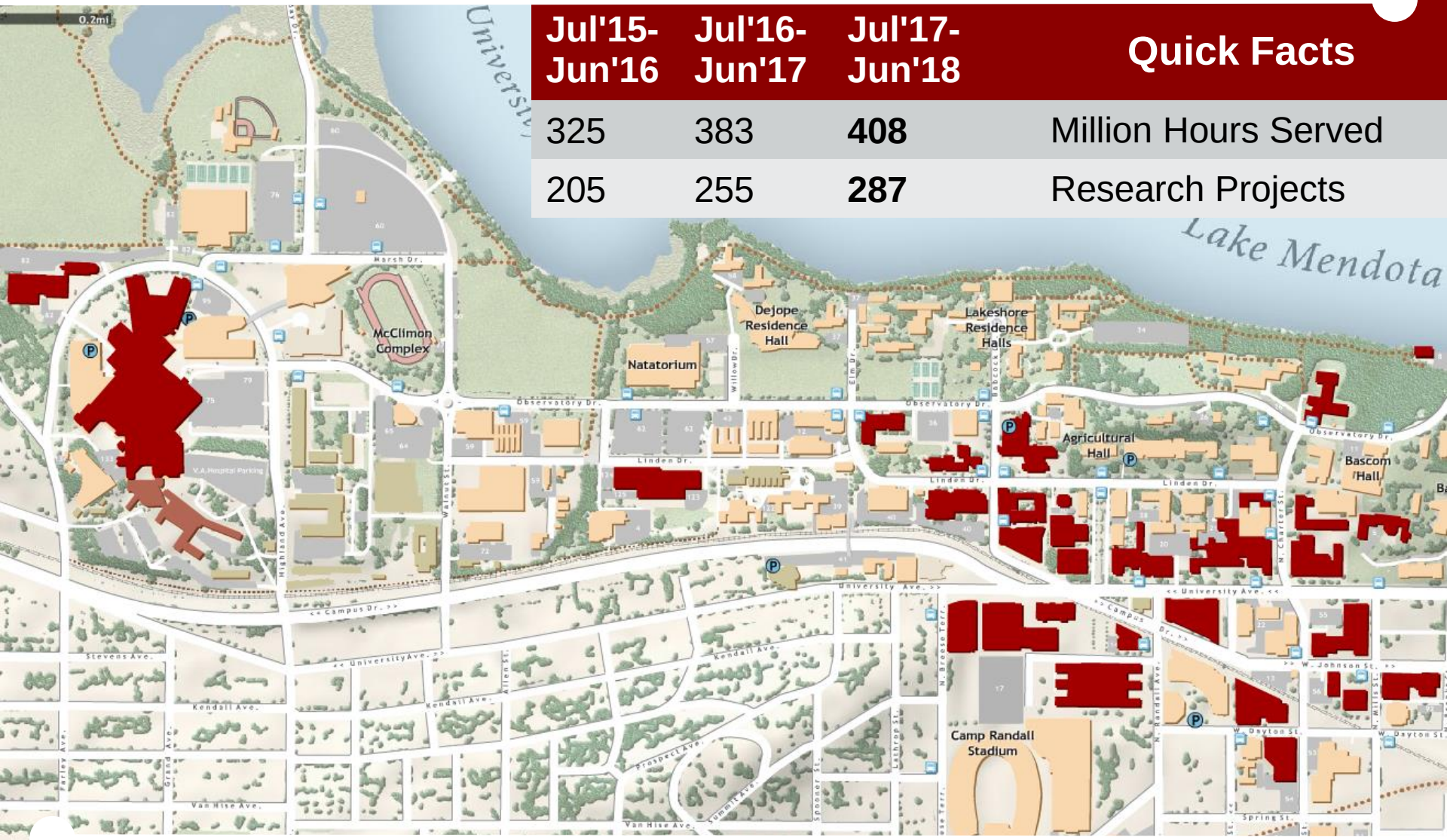
CHTC
<72 hrs/job
10,000 hrs/day

S

CENTER FOR HIGH THROUGHPUT COMPUTING

http://chtc.cs.wisc.edu

Jul'15-Jun'16	Jul'16-Jun'17	Jul'17-Jun'18	Quick Facts
325	383	**408**	Million Hours Served
205	255	**287**	Research Projects

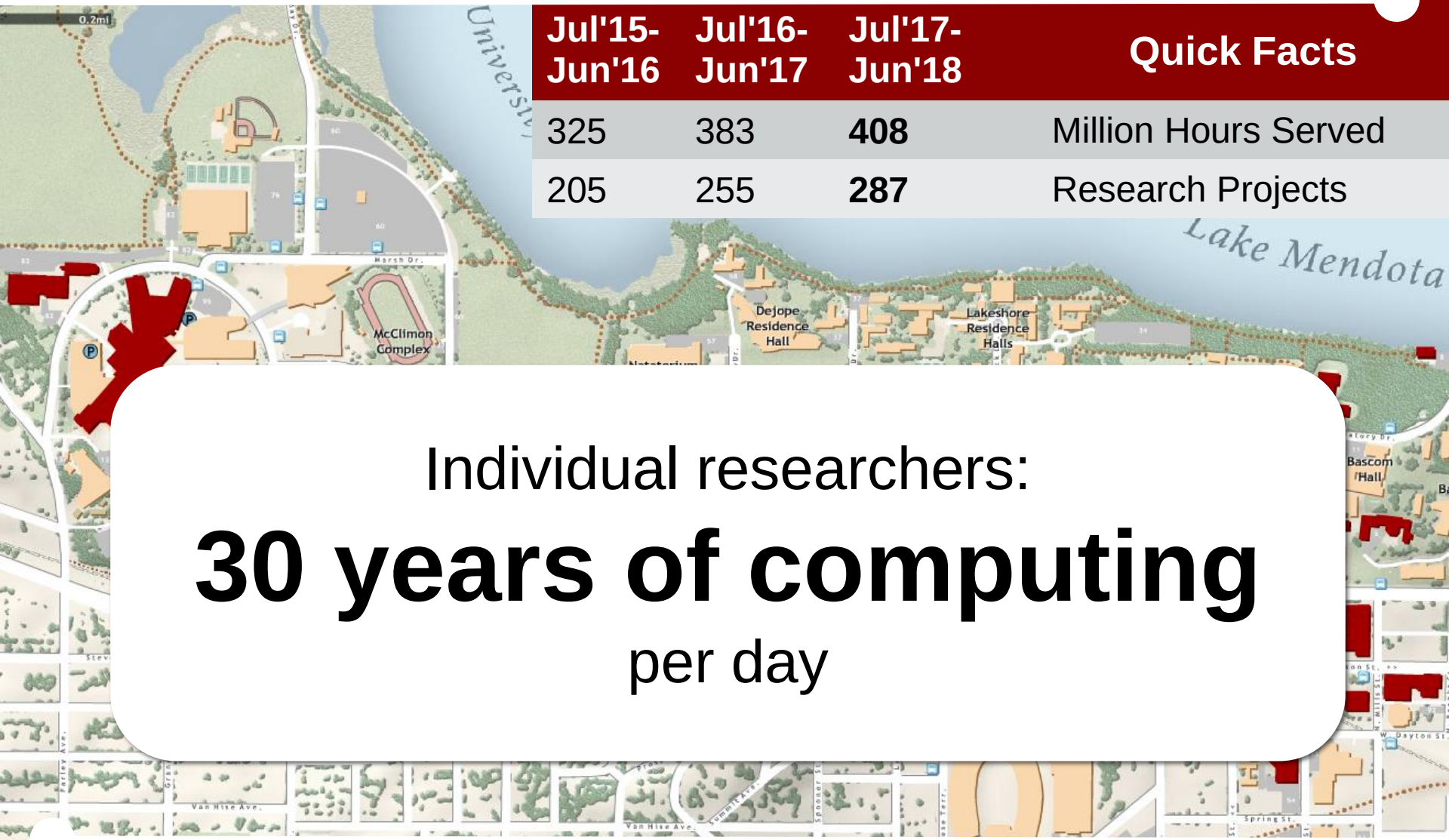


Researchers who use the CHTC are located all over campus (red buildings)

CENTER FOR HIGH THROUGHPUT COMPUTING

http://chtc.cs.wisc.edu

Jul'15-Jun'16	Jul'16-Jun'17	Jul'17-Jun'18	Quick Facts
325	383	**408**	Million Hours Served
205	255	**287**	Research Projects



Individual researchers:

30 years of computing

per day

Researchers who use the CHTC are located all over campus (red buildings)

What else?

- **Software**
  - We can support most unlicensed/open source software (R, Python, other projects)
  - Can also support Matlab
  - For this class, we've helped to put together a 'starter version' of GPGPU-Sim.
- **Data**
  - CHTC cannot *store* data
  - However, you can work with up to several TB of data on our system

How it works

- **Job submission**
  - Instead of running programs on your own computer, log in and submit **jobs**
  - Job = single independent calculation
  - Can run hundreds of jobs at once (or more)
- **HTCondor**
  - Computer program that controls and runs jobs on CHTC's computers

Getting Started

- **Facilitators**
  - Help researchers get started
  - Advise on best approach and resources for your research problem
- **How do I get an account?**
  - Temporary accounts created for this class on learn.chtc.wisc.edu.
  - If you want to use CHTC beyond this class for research, fill out our account request form: http://chtc.cs.wisc.edu/form

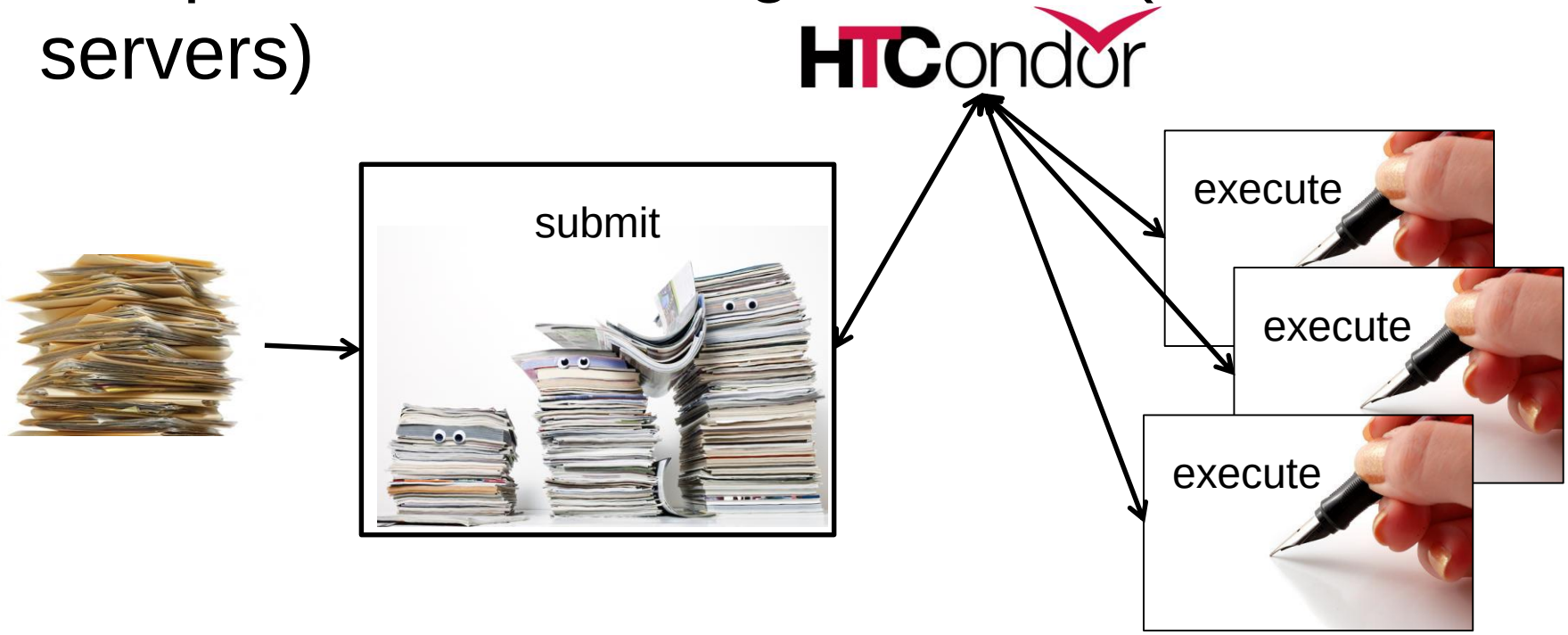
Take a second – login!

- Today we will use the `learn.chtc.wisc.edu` submit host.
  - Your username has already been created and is your NetID.
  - Your password is your NetID password.
- Take a second to login via SSH now!
- Whenever you're off-campus, you'll need to use the VPN to login. See https://it.wisc.edu/services/wiscvpn/

RUNNING A JOB ON CHTC'S HIGH THROUGHPUT SYSTEM WITH HTCONDOR

How It Works (in CHTC)

- Submit jobs to a queue (on a submit server)
- HTCondor schedules them to run on computers that belong to CHTC (execute servers)

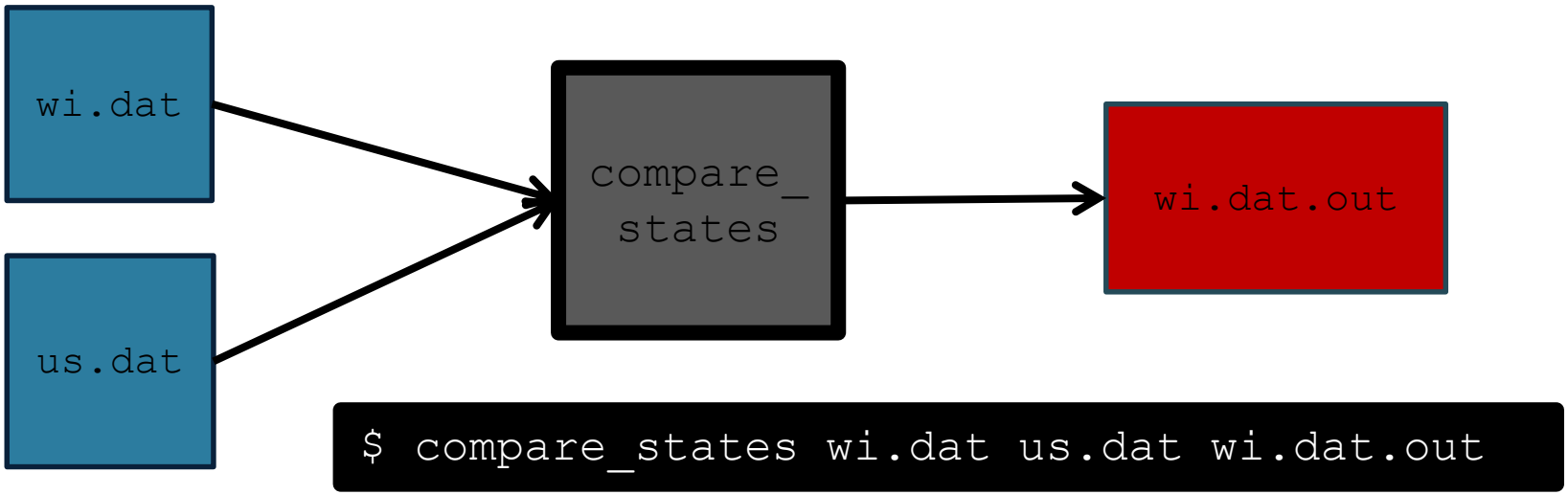


What is HTCondor?

- Software that schedules and runs computing tasks on computers

Job Example

- Consider an imaginary program called "`compare_states`", which compares two data files and produces a single output file.

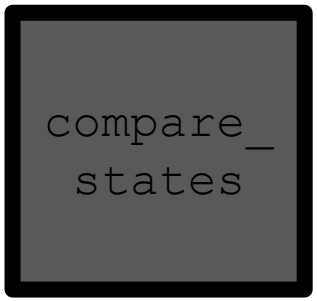


Submit File

job.submit

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

- List your executable and any arguments it takes.



- Arguments are any options passed to the executable from the command line.

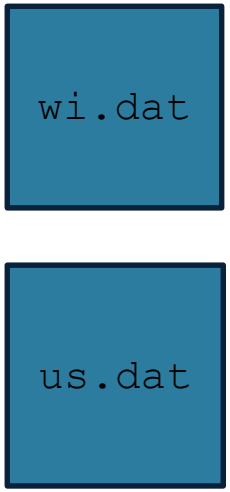
```
$ compare_states wi.dat us.dat wi.dat.out
```

Submit File

job.submit

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

- Indicate your input files.



Submit File

job.submit

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

- HTCondor will transfer back all new and changed files (usually output) from the job.

Submit File

job.submit

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

- `log`: file created by HTCondor to track job progress
- `output`/`error`: captures stdout and stderr

Submit File

job.submit

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

should_transfer_files = YES
transfer_input_files = us.dat, wi.dat
when_to_transfer_output = ON_EXIT

log = job.log
output = job.out
error = job.err

request_cpus = 1
request_disk = 20MB
request_memory = 20MB

queue 1
```

- Request the appropriate resources for your job to run.
- `queue`: keyword indicating "create a job."

Submitting and Monitoring

- To submit a job/jobs:

  `condor_submit` *`submit_file_name`*

- To monitor submitted jobs, use:

  `condor_q`

```
$ condor_submit job.submit
Submitting job(s).
1 job(s) submitted to cluster 128.
```

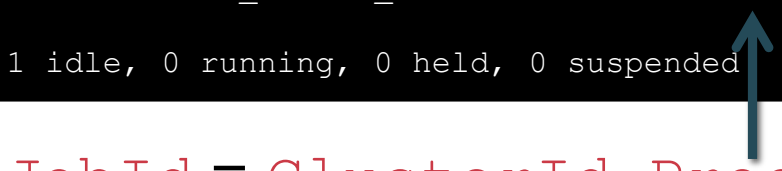
```
$ condor_q
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?... @ 05/01/17 10:35:54
OWNER  BATCH_NAME            SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
alice  ID: 128               5/9  11:09     _      _      1      1 128.0


1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

HTCondor Manual: condor_submit
HTCondor Manual: condor_q

More about `condor_q`

- By default **`condor_q`** shows:
  - user's job only (as of 8.6)
  - jobs summarized in "batches" (as of 8.6)
- Constrain with username, `ClusterId` or full `JobId`, which will be denoted `[U/C/J]` in the following slides

```
$ condor_q
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?... @ 05/09/17 11:35:54
OWNER  BATCH_NAME            SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
alice  ID: 128               5/9  11:09     _      _      1      1 128.0

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```



`JobId` = `ClusterId`.`ProcId`

More about `condor_q`

- To see individual job information, use:

  `condor_q -nobatch`

```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice      5/9  11:09   0+00:00:00 I  0    0.0 compare_states wi.dat us.dat

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

- We will use the `-nobatch` option in the following slides to see extra detail about what is happening with a job

Job Idle

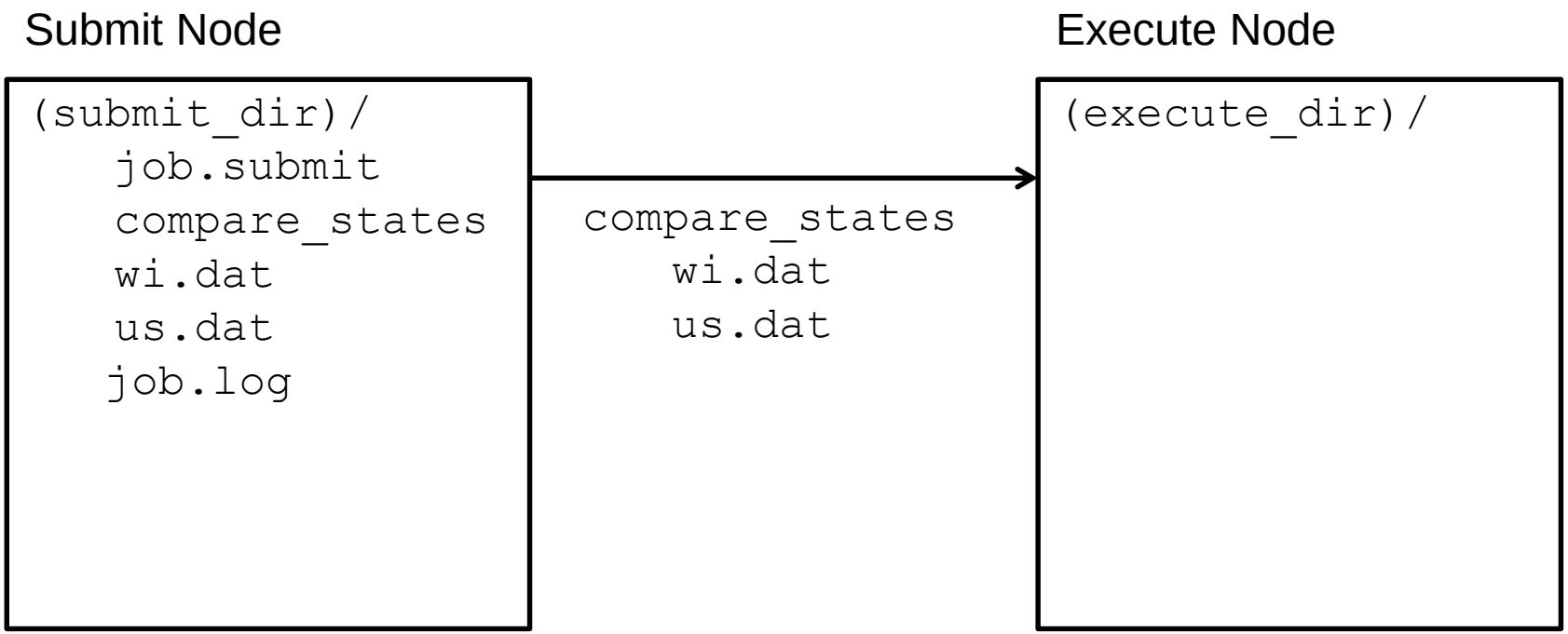
```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice      5/9  11:09   0+00:00:00 I  0    0.0 compare_states wi.dat us.dat

1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

Submit Node

```
(submit_dir)/
    job.submit
    compare_states
    wi.dat
    us.dat
    job.log
```

Job Starts

```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice      5/9  11:09   0+00:00:00 <  0    0.0 compare_states wi.dat us.dat w

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
```



Job Running

```
$ condor_q -nobatch

-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 ID          OWNER       SUBMITTED     RUN_TIME ST PRI SIZE CMD
128.0        alice       5/9  11:09   0+00:01:03 R  0    0.0 compare_states wi.dat us.dat

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
```

Submit Node

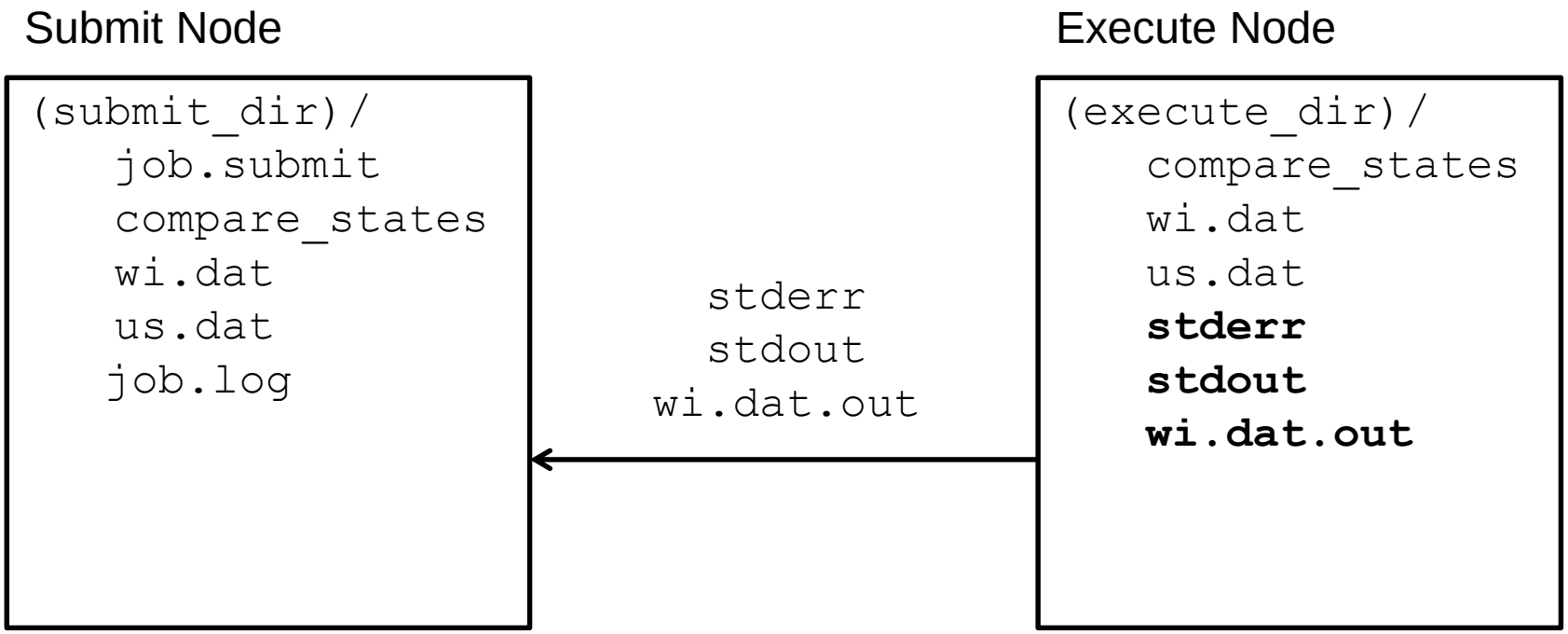
```
(submit_dir)/
    job.submit
    compare_states
    wi.dat
    us.dat
    job.log
```

Execute Node

```
(execute_dir)/
    compare_states
    wi.dat
    us.dat
    stderr
    stdout
    wi.dat.out
```

Job Completes

```
$ condor_q -nobatch
-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 ID          OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
128          alice      5/9  11:09   0+00:02:02 >  0    0.0 compare_states wi.dat us.dat

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
```



Job Completes (cont.)

```
$ condor_q -nobatch


-- Schedd: submit-5.chtc.wisc.edu : <128.104.101.92:9618?...
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD

0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
```

Submit Node

```
(submit_dir)/
    job.submit
    compare_states
    wi.dat
    us.dat
    job.log
    job.out
    job.err
    wi.dat.out
```

TRY IT OUT

https://github.com/bbockelm/gpgpu-sim_htcondor

Submitting Multiple Jobs

```
executable = compare_states
arguments = wi.dat us.dat wi.dat.out

transfer_input_files = us.dat, wi.dat

queue 1
```

Replacing single job inputs

```
executable = compare_states
arguments = $(infile) us.dat $(infile).out

transfer_input_files = us.dat, $(infile)

queue ...
```

with a variable of choice

Possible Queue Statements

multiple "queue" statements	infile = wi.dat queue 1 infile = ca.dat queue 1 infile = ia.dat queue 1 Not Recommended
matching ... pattern	queue infile matching *.dat
in ... list	queue infile in (wi.dat ca.dat ia.dat)
from ... file	queue infile from state_list.txt Recommended wi.dat ca.dat ia.dat state_list.txt

USER EXPECTATIONS

Be responsible!

- These resources are shared and you get to use them for free -- be a good citizen.
- Ask questions if you aren’t sure about something.
- Don’t run programs directly on the submit server.
- Data files should be small. Talk to us if you want to submit jobs with big (> 1GB) of data.

Resource Request

- Jobs are nearly always using a part of a computer, not the whole thing
- Very important to request appropriate resources (memory, cpus, disk) for a job

Resource Assumptions

- Even if your system has default CPU, memory and disk requests, these may be too small!
- Important to run test jobs and use the log file to request the right amount of resources:
  - requesting too little: causes problems for your and other jobs; jobs might by held by HTCondor
  - requesting too much: jobs will match to fewer “slots”

Log File

```
000 (128.000.000) 05/09 11:09:08 Job submitted from host:
<128.104.101.92&sock=6423_b881_3>
...
001 (128.000.000) 05/09 11:10:46 Job executing on host:
<128.104.101.128:9618&sock=5053_3126_3>
...
006 (128.000.000) 05/09 11:10:54 Image size of job updated: 220
    1  -  MemoryUsage of job (MB)
    220  -  ResidentSetSize of job (KB)
...
005 (128.000.000) 05/09 11:12:48 Job terminated.
    (1) Normal termination (return value 0)
        Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote Usage
        Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local Usage
        Usr 0 00:00:00, Sys 0 00:00:00  -  Total Remote Usage
        Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local Usage
    0  -  Run Bytes Sent By Job
    33  -  Run Bytes Received By Job
    0  -  Total Bytes Sent By Job
    33  -  Total Bytes Received By Job
    Partitionable Resources :    Usage  Request Allocated
       Cpus                 :                1         1
       Disk (KB)            :       14    20480  17203728
       Memory (MB)          :        1       20        20
```

TESTING IS KEY!

ALWAYS run test jobs before submitting many jobs at once.

Getting Help

- Christina Koch and Lauren Michael work for CHTC as Research Computing Facilitators.
- It's the facilitator's job to answer questions and help people get started with computing at CHTC.
- General CHTC questions:
  - Email us! chtc@cs.wisc.edu
  - Or come to office hours in the WID:
    - Tues/Thurs, 3:00 - 4:30
    - Wed, 9:30 - 11:30
  - http://chtc.cs.wisc.edu/get-help
- Questions about today's homework? Easiest to start off with contacting Prof. Sinclair or attend office hours.

NEXT STEPS

Building up a workflow

- Try to get ONE job running
  - Follow steps outlined in the git repository.
  - Troubleshoot
- Check memory/disk requirements
- Do a small scale test of 5-10 jobs
  - Check memory + disk requirements *again*
- Run full-scale set of jobs

Homework for Today

- Again, the GitHub repository for a few simple tutorials is at https://github.com/bbockelm/gpgpu-sim_htcondor
  - These have been tested on learn.chtc.wisc.edu; please do your class work there for now.
  - If you want to continue growing your use of CHTC after this class, we will allocate your account onto a larger machine.
- HW2 is posted on the class website.