# Introduction to HTC

2015 OSG User School, Monday, Lecture 1

Greg Thain

University of Wisconsin–
Madison

Center For High Throughput
Computing

# Welcome!

# Why Are We Here?

***Transform***
Your Research
With Computing

# Overview

# Overview of Week

| | |
|---|---|
| **Monday** | • High Throughput Computing locally<br>• Miscellaneous<br>‣ Survey<br>‣ UW reimbursement form<br>‣ |
| **Tuesday** | • Distributed High Throughput Computing<br>• Security<br>• Tour of Wisconsin Institutes for Discovery |
| **Wednesday** | • Distributed storage<br>• Practical issues with DHTC |
| **Thursday** | • From science to production<br>• Principles of HTC<br>• HTC Showcase<br>• Next steps |

# Overview of a Day

- Short introductory lectures
- Lots of hands-on exercises
- Some demos, interactive sessions, etc.
- Optional evening sessions
  - Monday – Wednesday, 7–9 p.m.
  - Union South (check TITU)

    School staff on hand
  -

# Keys to Success

- Work hard
- Ask questions!
  … during lectures
  … during exercises
  … during breaks
  … during meals
  … in person is best, email is OK
- If we do not know an answer, we will try to find the person who does

# Ready?

# One Thing

# Computing is Cheap!

Open Science Grid

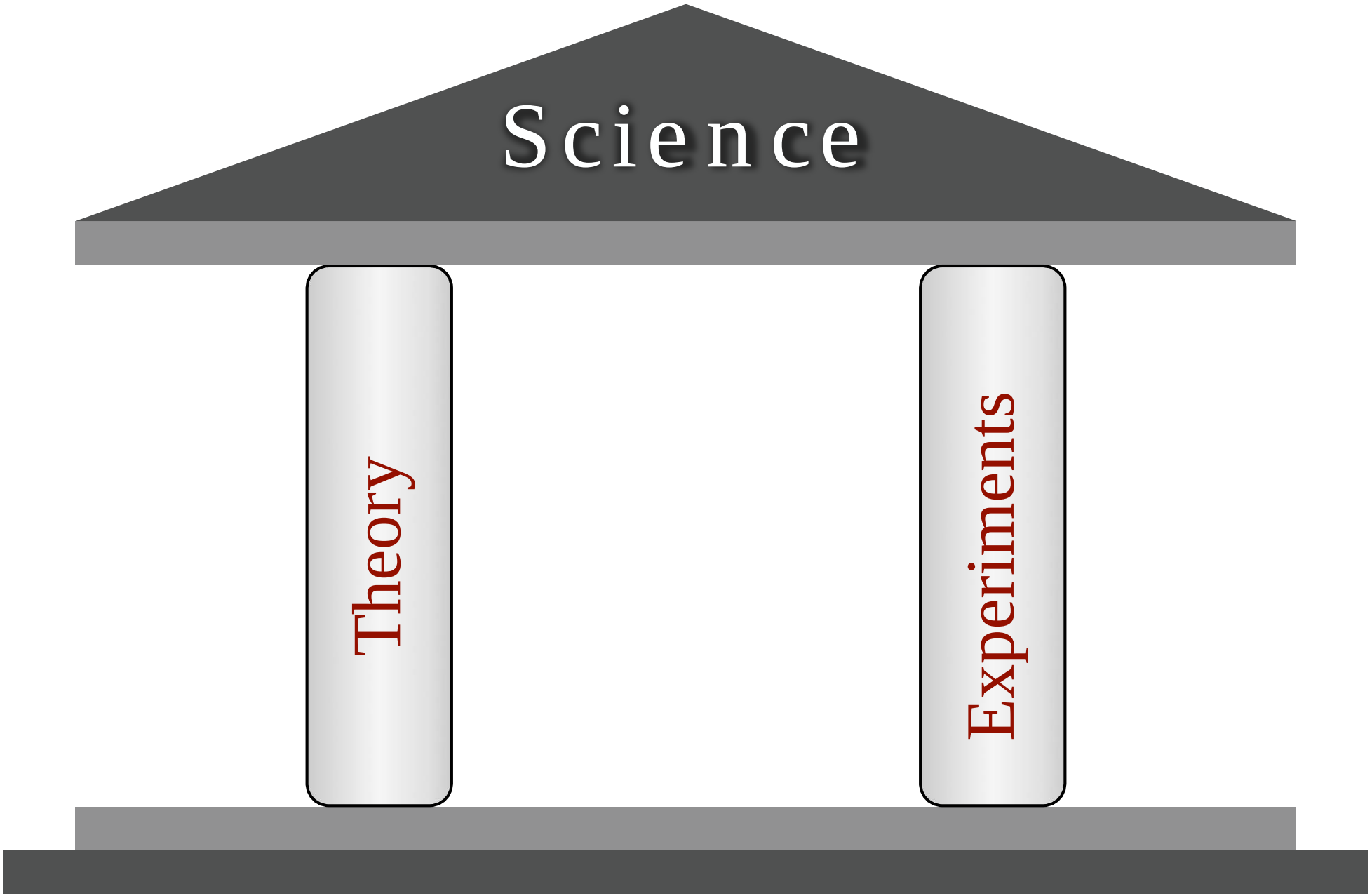
# Goals For This Session

- Understand the basics of High Throughput Computing
- Understand a few things about HTCondor, which is one kind of HTC system
- Use basic HTCondor commands
- Run a job locally!

# Why HTC?

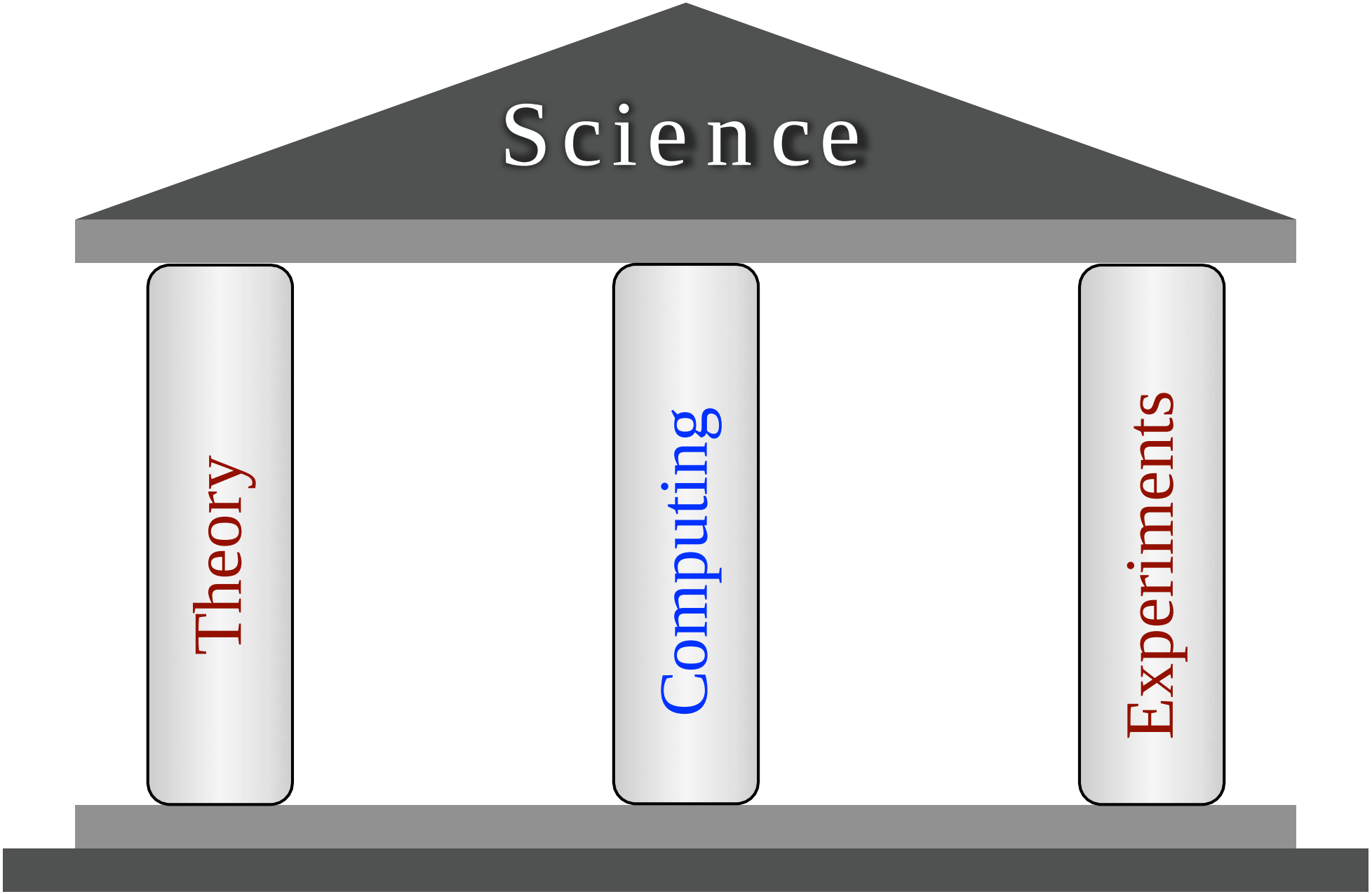
Open Science Grid

# Computing in Science



Open Science Grid

# Computing in Science



Open Science Grid

# Example Challenge

- You have a program to run (simulation, Monte Carlo, data analysis, image analysis, stats, …)
- Each run takes about 1 hour
- You want to run the program $8 \times 12 \times 100$ times
- 9600 hours $\approx$ 1.1 years … running nonstop!
- Conference is next week

Open Science Grid

# Distributed Computing

- Use many computers to perform 1 computation
- Example:
  - 2 computers => 4,800 hours ≈ ½ year
  - 8 computers => 1,200 hours ≈ 2 months
  - 100 computers => 96 hours = 4 days
  - 9,600 computers => 1 hour! (but …)

These computers are no faster than your laptop!

Open Science Grid

# Performance vs. Throughput

- High *Performance* Computing (HPC)
  - Focus on biggest, fastest systems (supercomputers)
  - Maximize operations per *second*
  - Often requires special code
  - Often must request and wait for access
- High *Throughput* Computing (HTC)
  - Focus on using all resources, reliably, all the time
  - Maximize operations per *year*
  - Use any kind of computer, even old, slow ones
  - Must break task into separate, independent parts
  - Access varies by availability, usage, etc.

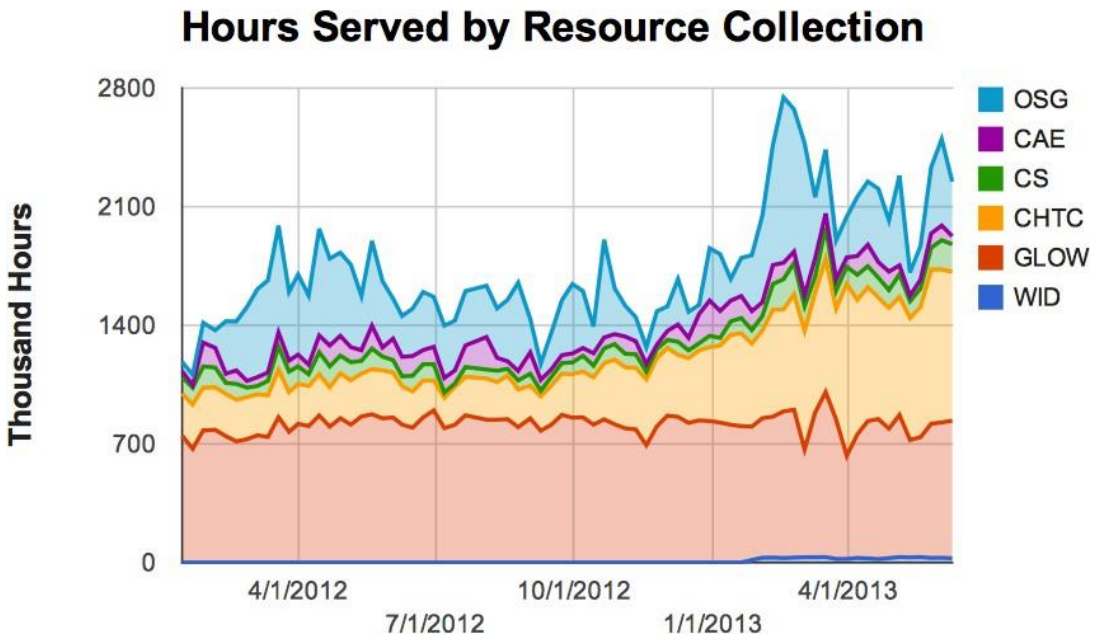
# HPC vs HTC: An Analogy

Open Science Grid

# HPC vs HTC: An Analogy

# Example HTC Site (Wisconsin)

- Our local HTC systems
- Recent CPU hours:
  ~ 280,000 / day
  ~ 8.3 million / month
  ~ 78 million / year



# Open Science Grid

- HTC scaled way up
  - Over 110 sites
  - Mostly U.S.
  - Some others
  - Past year:
    ~170 million jobs
    ~770 million CPUhours
    ~372 petabytes transferred


- Can submit jobs locally, move to OSG
- http://www.opensciencegrid.org/

Open Science Grid

# Other Distributed Computing

- Other systems to manage a local cluster:
  - PBS/Torque
  - LSF
  - Sun Grid Engine/Oracle Grid Engine
  - SLURM
- Other wide-area systems:
  - European Grid Infrastructure
  - Other national and regional grids
  - Commercial cloud systems used to augment grids
- HPC
  - Various supercomputers (e.g., TOP500 list)
  - XSEDE

# HTCondor

# HTCondor History and Status

- History
  - Started in 1988 as a “cycle scavenger”
  - Protected interests of users and machine owners
- Today
  - Expanded to become CHTC team: 20+ full-time staff
  - Current production release: HTCondor 8.4.0
  - HTCondor software: ~700,000 lines of C/C++ code
- Miron Livny
  - Professor, UW–Madison CompSci
  - Director, CHTC
  - Dir. of Core Comp. Tech.,
  - WID/MIR Tech. Director & PI, OSG

Open Science Grid

# HTCondor Functions

- Users
  - Define jobs, their requirements, and preferences
  - Submit and cancel jobs
  - Check on the state of a job
  - Check on the state of the machines
- Administrators
  - Configure and control the HTCondor system
  - Declare policies on machine use, pool use, etc.
- Internally
  - Match jobs to machines (enforcing all policies)
  - Track and manage machines
  - Track and run jobs

Open Science Grid

# Terminology: *Job*

- ***Job*:** A computer program *or* one run of it
- *Not* interactive, *no* GUI (e.g., not Word or email)

  (How could you interact with 1,000 programs running at once?)

  1. Input: command-line arguments and/or files
  2. Run: do stuff
  3. Output: standard output & error and/or files
- Scheduling
  - User decides when to *submit* job to be run
  - System decides when to run job, based on policy

# Terminology: *Machine, Slot*

- *Machine*
  - A *machine* is a physical computer (typically)
  - May have multiple *processors* (computer chips)
  - One processor may have multiple *cores* (CPUs)
- HTCondor: *Slot*
  - One assignable unit of a machine (i.e., 1 job per slot)
  - Most often, corresponds to one core
  - Thus, typical machines today have 4–40 slots
- Advanced HTCondor feature: Can get 1 slot with many cores on 1 machine, for MPI(-like) jobs

# Terminology: Matchmaking

Two-way process of finding a slot for a job

Jobs have requirements and preferences

*E.g.: I need Red Hat Linux 6 and 100 GB of disk space, and*
*prefer to get as much memory as possible*

Machines have requirements and preferences

*E.g.: I run jobs only from users in the Comp. Sci. dept., and prefer to run ones that ask for a lot of memory*

*Important jobs may replace less important ones*

Thus: Not as simple as waiting in a line!

# Running a Job

# Viewing Slots

## condor_status

- With no arguments, lists *all* slots currently in pool
- Summary info is printed at the end of the list
- For more info: exercises, **-h**, manual, next lecture

```
slot6@opt-a001.cht  LINUX      X86_64 Claimed   Busy      1.000  1024  0+19:09:32
slot7@opt-a001.cht  LINUX      X86_64 Claimed   Busy      1.000  1024  0+19:09:31
slot8@opt-a001.cht  LINUX      X86_64 Unclaimed Idle      1.000  1024  0+17:37:54
slot9@opt-a001.cht  LINUX      X86_64 Claimed   Busy      1.000  1024  0+19:09:32
slot10@opt-a002.ch  LINUX      X86_64 Unclaimed Idle      0.000  1024  0+17:55:15
slot11@opt-a002.ch  LINUX      X86_64 Unclaimed Idle      0.000  1024  0+17:55:16
```

```
                     Total Owner Claimed Unclaimed Matched Preempting Backfill

       NTEL/WINNT51      2     0       0         2       0          0        0
      INTEL/WINNT61     52     2       0        50       0          0        0
       X86_64/LINUX   2086   544    1258       284       0          0        0

             Total    2140   546    1258       336
```

# Viewing Jobs

## condor_q

- With no args, lists all jobs waiting or running here
- For more info: exercises, **-h**, manual, next lecture

```
-- Submitter: osg-ss-submit.chtc.wisc.edu : <...> : ...
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
    6.0   cat          11/12 09:30   0+00:00:00 I  0   0.0  explore.py
    6.1   cat          11/12 09:30   0+00:00:00 I  0   0.0  explore.py
    6.2   cat          11/12 09:30   0+00:00:00 I  0   0.0  explore.py
    6.3   cat          11/12 09:30   0+00:00:00 I  0   0.0  explore.py
    6.4   cat          11/12 09:30   0+00:00:00 I  0   0.0  explore.py

5 jobs; 5 idle, 0 running, 0 held
```

# Basic Submit File

```
executable = word_freq.py
universe = vanilla
arguments = "words.txt 1000"

output = word_freq.out
error = word_freq.err
log = word_freq.log

should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_input_files = words.txt

queue
```

# Submit a Job

**condor_submit *submit-file***

- Submits job to local submit machine
- Use condor_q to track

**Submitting job(s).**
**1 job(s) submitted to cluster *NNN*.**

Each condor_submit creates one Cluster

A *job ID* is written as ***cluster*.*process*** (e.g., **8.0**)

We will see how to make multiple processes later

# Remove a Job

**condor_rm *cluster* *[...]***
**condor_rm *cluster*.*process* *[...]***

- Removes one or more jobs from the queue
- Identify jobs by whole cluster or single job ID
- Only you *(or admin)* can remove your jobs

**Cluster *NNN* has been marked for removal.**

# Your Turn!

# Thoughts on Exercises

- Copy-and-paste is quick, but you may learn more by typing out commands yourself
- Experiment!
  - Try your own variations on the exercises
  - If you have time, try to apply to your own work
- If you do not finish, that's OK — you can make up work later or during evenings, if you like
- If you finish early, try any extra challenges or optional sections, or move ahead to the next section if you are brave

# Exercises!

- Ask questions!
- Lots of instructors around
- Coming next:

| | |
|---|---|
| Now – 10:30 | Hands-on exercises |
| 10:30–10:45 | Break |
| 10:45–11:15 | Lecture |
| 11:15–12:15 | Hands-on exercises |