

Day 9: Introduction to CHTC

Suggested reading: HTCondor 7.8 Manual:

<http://research.cs.wisc.edu/htcondor/manual/v7.8/>

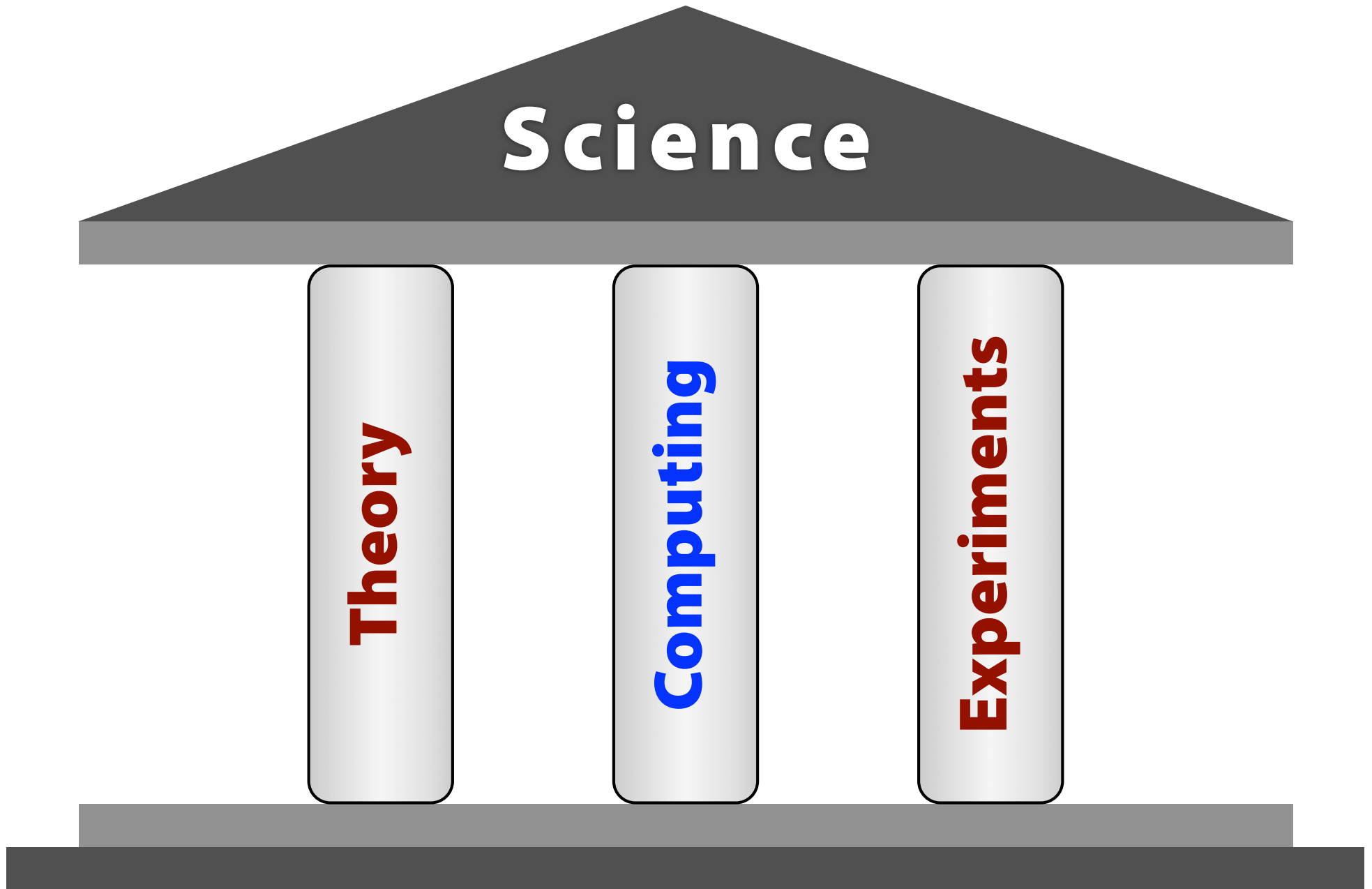
Chapter 1: Overview

Chapter 2: Users' Manual (at most, 2.1–2.7)

Homework Review

CHTC

Center for High-Throughput Computing

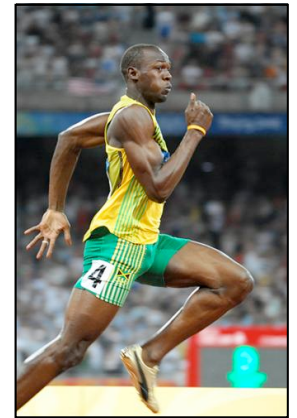




- Computing resources for researchers
- Right here on campus
- **Free** for UW–Madison researchers
- Funded by UW, NSF, Dept. of Energy, NIH, ...
- Last year: **13 million** CPU hours *delivered*

High-Throughput Computing

- “... use of many computing resources over long periods of time to accomplish a computational task” — Wikipedia (retrieved 7 Nov 2011)
- Not high-performance computing (HPC)
 - TOP500 list of supercomputers
 - FLOPS (**f**loating-**p**oint **o**perations **p**er **s**econd)
- Aims to maximize long-term throughput
 - “How many results this week/month/year?”
 - FLOP**Y** \neq $(60 \times 60 \times 24 \times 365)$ FLOPS



The Hope (& Hype) of Distributed Computing

- Do a *lot* of computing
- Always be available and reliable
- Degrade gracefully
- Spread the workload automatically
- Grow (and shrink) easily when needed
- Respond well to temporary overloads
- Adapt easily to new uses

Adapted from: Enslow, P. H., Jr. (1978). What is a “distributed” data processing system? *Computer*, 11(1), 13–21. doi:10.1109/C-M.1978.217901

Definition of Distributed Computing

Multiplicity of resources

- General purpose; not same, but same capabilities
- More replication is better

Component interconnection

- Networked, loosely coupled

Unity of control

- Not centralized control (single point of failure)
- Unified by common goal, and hence policy

System transparency

- Whole system appears as one virtual system to user

Component autonomy

- Autonomous (act locally) but cooperative (think globally)

Enslow, P. H., Jr., & Saponas, T. G. (**1981**). *Distributed and decentralized control in fully distributed processing systems: A survey of applicable models* (GIT-ICS-81/02). Georgia Institute of Technology.

What CHTC Offers



CHTC Machines

- **Hardware**

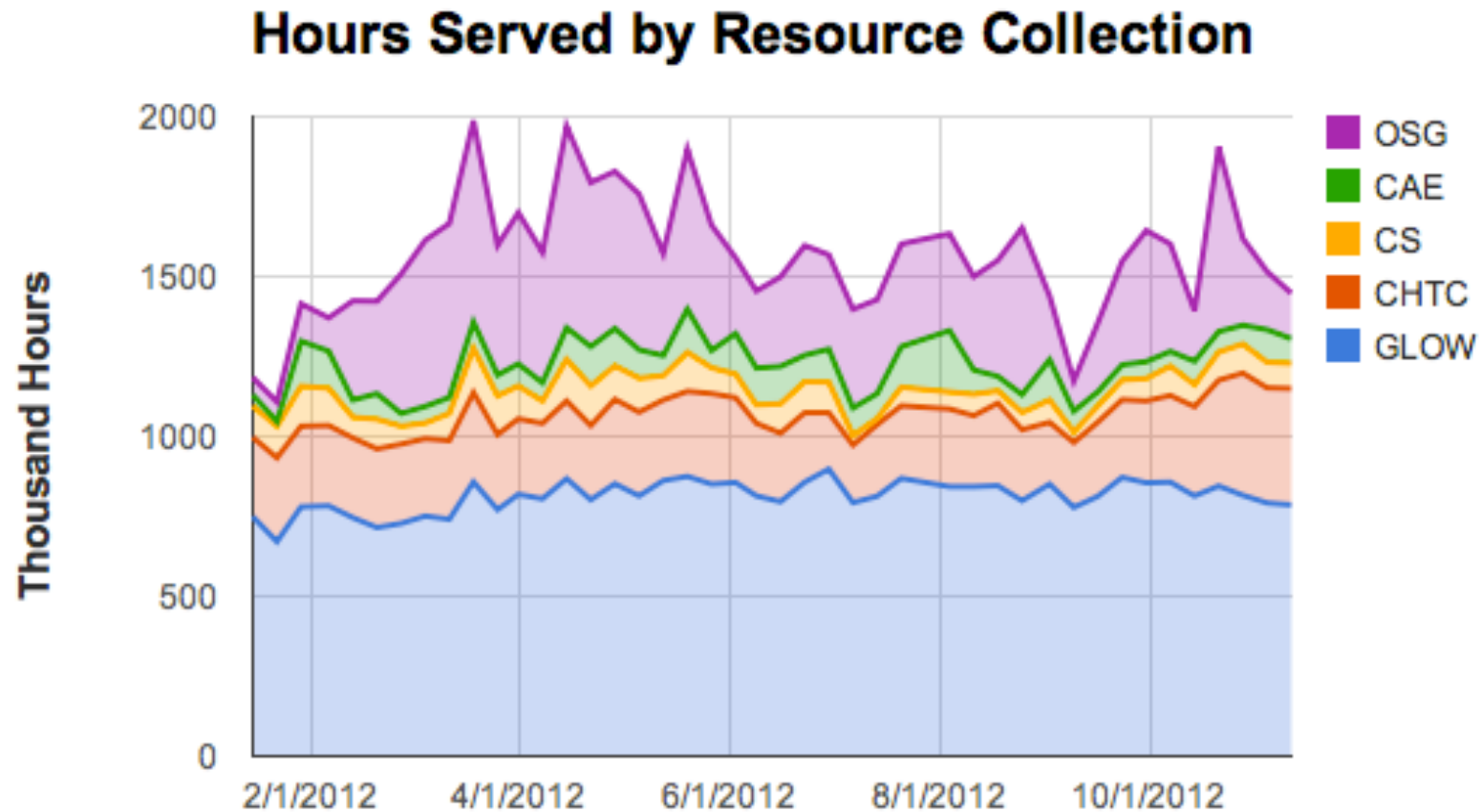
- ~200 8–16-core 2.6–2.8 GHz Intel 64-bit, 1U servers
- Typical machine: 12–96 GB memory, ~0.5–2 TB disk
- 1 Gbit Ethernet (good for file transfer, not MPI)

- **Software**

- Scientific Linux 5 (var. of RHEL 5); some RHEL 6
- Languages: *Python*, C/C++, Java, Perl, Fortran, ...
- Extra software (no licenses): R, MATLAB, Octave

- **Location:** Mostly in CompSci B240, some in WID

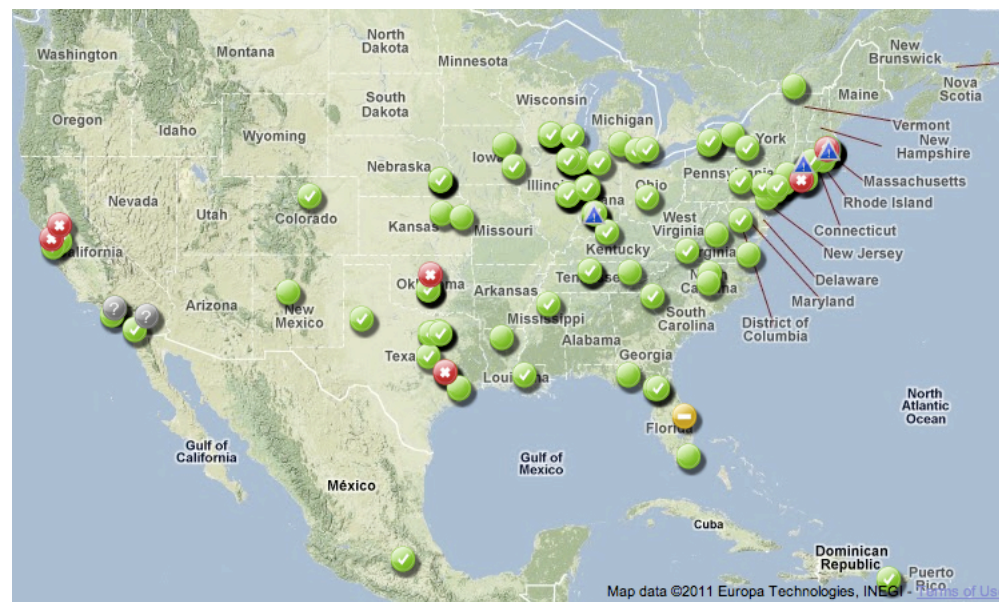
CHTC Usage Statistics



~35,000 hours per day
~1,000,000 hours per month
~15,000,000 hours per year

Open Science Grid

- HTC scaled *way* up
 - Over 110 sites
 - Mostly in U.S., plus others
 - Past year:
 - ✦ ~195,000,000 jobs
 - ✦ ~720,000,000 CPU hours
 - ✦ ~322,000 TB transferred
- Can submit jobs to CHTC, move to OSG
- <http://www.opensciencegrid.org/>



Anyone want a tour?

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 7.8.6
- 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



What Does HTCondor Do?

- **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

Jobs

- = Computer programs
- Not *interactive* (e.g., Word, Firefox, email)
- **Batch processing**: Run without human intervention
 - Input: command-line arguments, files, downloads?
 - Run: do stuff
 - Output: standard output & error, files, DB update?
- **Scheduling**
 - Reserved: Person gets time slot, computer runs then
 - Opportunistic:
Person submits job, computer decides schedule

Machines

- Terminology
 - A *machine* is a physical computer (typically)
 - May have multiple *processors* (computer chips)
 - These days, each may have multiple *cores* (CPUs)
- HTCondor: *Slot*
 - One assignable unit of a computing resource
 - Most often, corresponds to one core
 - Thus, typical machines today have 4–40 slots
- Advanced HTCondor feature: Can request multiple cores for a single slot (that uses parallel computing)

Matchmaking

- Two-way process of matching jobs and machines
- **Job**
 - Requirements, e.g.: OS, architecture, memory, disk
 - Preferences, e.g.: owner, speed, memory, disk, load
- **Machine**
 - Requirements, e.g.: submitter, time of day, usage
 - Preferences, e.g.: submitter, memory, disk, load
- **Administrator**
 - Preferences, e.g.: prior usage, priority, various limits
- Thus: Not as simple as waiting in a line!

Running Jobs

Our Submit Machine

- **Access**

- Hostname (ssh): **submit-368.chtc.wisc.edu**
- If enrolled, get account info from me

- **Rules**

- Full access to all CHTC resources (i.e., machines)
- All UW Information Technology policies apply
<http://www.cio.wisc.edu/policies.aspx>
- OK for research *and training*
- Usage is monitored

- **Notes**

- No backups! Keep original files elsewhere
- Accounts will be deleted early January 2013, unless...

Viewing Slots

condor_status

- With no arguments, lists *all* slots currently in pool
- Summary info at end
- For more options: **-h**, HTCondor Manual, next class

```
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 |
|---------------|-------|-------|---------|-----------|---------|------------|----------|
| INTEL/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 | 0 | 0 | 0 |

Viewing Jobs

condor_q

- With no args, lists *all* jobs waiting or running here
- For more options: **-h**, HTCondor Manual, next class

```
-- Submitter: submit-368.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
```

condor_q *owner*

- Just one owner's jobs (e.g., your own)

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
```

Program to run.
Must be runnable

Command-line
arguments to pass

Condor's log file
from running the job; very helpful,
do not

Comma-separated list
of input files to
transfer to
machine *[opt]*

Must have this to run job!

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*.

- One **condor_submit** yields one *cluster* (in queue)
- Each **queue** statement yields one *process*
- **condor_q**: ID is *cluster.process* (e.g., 8.0)
- We will see how to set up multiple jobs next time

Remove a Job

```
condor_rm cluster [...]
condor_rm cluster.process [...]
```

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

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

Homework

Homework

- Run a job... or several!
 - I supply a Python script — a bit like homework #1
 - How many of your past homeworks can you run?
 - Do you have any other jobs to run?
- Turn in submit file + resulting log, out, and err files
- Watch for errors and hung jobs!!!
 - Be sure your script runs from command line
 - Monitor log file
 - Remove hung jobs (see homework)