Day 9: Introduction to CHTC

Suggested reading: Condor 7.7 Manual:
http://www.cs.wisc.edu/condor/manual/v7.7/

Chapter 1: Overview
Chapter 2: Users’ Manual (at most, 2.1–2.7)
Turn In Homework
Homework Review
CHTC

Center for High Throughput Computing
Science

Theory

Experiments
Computer Sciences 368

Scripting for CHTC

Science

Theory
Computing
Experiments
• Computing resources for researchers
• Right here on campus
• **Free** for UW–Madison researchers
• Funded by UW, NSF, Dept. of Energy, NIH, …
• Last year: **15 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 (floating-point operations per second)

• Aims to maximize long-term throughput
  – “How many results this week/month/year?”
  – \( \text{FLOPY} \neq (60 \times 60 \times 24 \times 365) \text{ FLOPS} \)
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 (floating-point operations per second)

• Aims to maximize long-term throughput
  – “How many results this week/month/year?”
  – FLOPY ≠ (60 × 60 × 24 × 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

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)

What CHTC Offers
What CHTC Offers

Computing
What CHTC Offers
What CHTC Offers

Computing

People
CHTC Machines

• Hardware
  – ~170 8–12-core 2.6–2.8 GHz Intel 64-bit, 1U servers
  – Typical machine: 12–24 GB memory, ~350 GB 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 100 sites
  – Mostly in U.S., plus others
  – Past year:
    ✦ ~200,000,000 jobs
    ✦ ~514,000,000 CPU hours
    ✦ ~280,000 TB transferred

• Can submit jobs to CHTC, move to OSG

• [http://www.opensciencegrid.org/](http://www.opensciencegrid.org/)
Anyone want a tour?
Condor
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: Condor 7.6.6
  – Condor software alone: ~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 Condor 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 Condor 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)

• Condor: *Slot*
  – One assignable unit of a computing resource
  – Most often, corresponds to one core
  – Thus, typical machines today have 4–40 slots

• Advanced Condor 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 disabled 1 June 2012, unless...
Viewing Slots

`condor_status`

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

```
<table>
<thead>
<tr>
<th>Slot</th>
<th>Owner</th>
<th>CPU</th>
<th>RAM</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:slot6@opt-a001.cht">slot6@opt-a001.cht</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
</tr>
<tr>
<td><a href="mailto:slot7@opt-a001.cht">slot7@opt-a001.cht</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
</tr>
<tr>
<td><a href="mailto:slot8@opt-a001.cht">slot8@opt-a001.cht</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed</td>
<td>Idle</td>
</tr>
<tr>
<td><a href="mailto:slot9@opt-a001.cht">slot9@opt-a001.cht</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Claimed</td>
<td>Busy</td>
</tr>
<tr>
<td><a href="mailto:slot10@opt-a002.ch">slot10@opt-a002.ch</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed</td>
<td>Idle</td>
</tr>
<tr>
<td><a href="mailto:slot11@opt-a002.ch">slot11@opt-a002.ch</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed</td>
<td>Idle</td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>Platform</th>
<th>Claimed</th>
<th>Unclaimed</th>
<th>Matched</th>
<th>Preempting</th>
<th>Backfill</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEL/WINNT51</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>INTEL/WINNT61</td>
<td>52</td>
<td>2</td>
<td>50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X86_64/LINUX</td>
<td>2086</td>
<td>544</td>
<td>284</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>2140</td>
<td>546</td>
<td>1258</td>
<td>336</td>
<td>0</td>
</tr>
</tbody>
</table>
Viewing Jobs

condor_q

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

```
-- Submitter: submit-368.chtc.wisc.edu : <...> : ...
<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0</td>
<td>cat</td>
<td>11/12 09:30</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>explore.py</td>
</tr>
<tr>
<td>6.1</td>
<td>cat</td>
<td>11/12 09:30</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>explore.py</td>
</tr>
<tr>
<td>6.2</td>
<td>cat</td>
<td>11/12 09:30</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>explore.py</td>
</tr>
<tr>
<td>6.3</td>
<td>cat</td>
<td>11/12 09:30</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>explore.py</td>
</tr>
<tr>
<td>6.4</td>
<td>cat</td>
<td>11/12 09:30</td>
<td>0+00:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>explore.py</td>
</tr>
</tbody>
</table>
```

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 from command line. Path is relative to current directory when submitted.
Basic Submit File

```python
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
```

Command-line arguments to pass to executable when run; surround with double quotes `[opt]`
Basic Submit File

```plaintext
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
```

Local files that will receive the contents of standard output and error from the run [opt]
Basic Submit File

```bash
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
```

Condor’s log file from running the job; very helpful, do not omit!
Basic Submit File

```python
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
```

Comma-separated list of input files to transfer to machine [opt]
Basic Submit File

```python
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  # 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)