Intermediate Condor
Monday morning, 11:30am

Scot Kronenfeld <kronenfe@cs.wisc.edu>
OSG Software Team
University of Wisconsin-Madison

Open Science Grid

Before we begin…

- Any questions on the lectures or exercises up to this point?

Open Science Grid

HTC: Reliability

- Two quotes from the first lecture:
  - "We don't care about operations per second, we care about operations per year"
  - "HTC focuses on reliability"
- How should we provide reliability?

Open Science Grid

First: why do jobs fail?

- The computer running the job fails
  - Or the network, or the disk, or the OS, or…
- Your job might be *preempted*:
  - Condor decides your job is less important than another, so your job is stopped and another started.
  - Not a “failure” per se, but it may feel like it to you.

Open Science Grid

Reliability example #1

- When a job fails or is preempted:
  - It stays in the queue (on the schedd)
  - A note is written to the job log file
  - It reverts to “idle” state
  - It is eligible to be matched again
- Relax! Condor will run your job again
  - “It’s like money in the bank”

Reliability example #2: Checkpointing

- When Condor re-runs your job, it starts over from the beginning. ☹
- Unless you use the *standard universe*
  - Huh? What's a "universe"?

Open Science Grid

Condor's Universes

- Condor can support various combinations of features/environments in different "universes"
- Different Universes provide different functionality for your job:
  - Vanilla: Run any serial job (your first job this morn')
  - Standard: Support for checkpointing & remote I/O
  - Java: Special support for Java
  - Parallel: Support for parallel jobs (such as MPI)
  - … and others

Open Science Grid

Process Checkpointing

- Condor's process checkpointing mechanism saves the entire state of a process into a checkpoint file
  - Memory, CPU, I/O, etc.
- The process can then be restarted from right where it left off
- Typically no changes to your job's source code needed—however, your job must be relinked with Condor's Standard Universe support library

Relinking Your Job for Standard Universe

To do this, just place “condor_compile” in front of the command you normally use to link your job:

```
% condor_compile gcc -o myjob myjob.c
```

- OR -

```
% condor_compile f77 -o myjob filea.f
fileb.f
```

Limitations of the Standard Universe

- Condor's checkpointing is not at the kernel level. Thus in the Standard Universe the job may not:
  - fork()
  - Use kernel threads
  - Use some forms of IPC, such as pipes and shared memory
- Many typical scientific jobs are OK
- Must be same gcc as Condor was built with

When will Condor checkpoint your job?

- Periodically, if desired (for fault tolerance)
- When your job is preempted by a higher priority job
- When your job is vacated because the execution machine becomes busy
- When you explicitly run:
  - condor_checkpoint
  - condor_vacate
  - condor_off
  - condor_restart

Access to data in Condor

- Option #1: Shared filesystem
  - Simple to use, but make sure your filesystem can handle the load
  - Not available for today's exercises
- Option #2: Condor's file transfer
  - Can automatically send back changed files
  - Atomic transfer of multiple files
  - Can be encrypted over the wire
  - This is what we'll do in the exercises
- Option #3: Remote I/O

Condor File Transfer

- `ShouldTransferFiles = YES`
  - Always transfer files to execution site
- `ShouldTransferFiles = NO`
  - Rely on a shared filesystem
- `ShouldTransferFiles = IF_NEEDED`
  - Will automatically transfer the files if needed

```
Universe   = vanilla
Executable = my_job
Log        = my_job.log
ShouldTransferFiles   = YES
Transfer_input_files  = dataset$(Process), common.data
Queue 600
```

Condor File Transfer with URLs

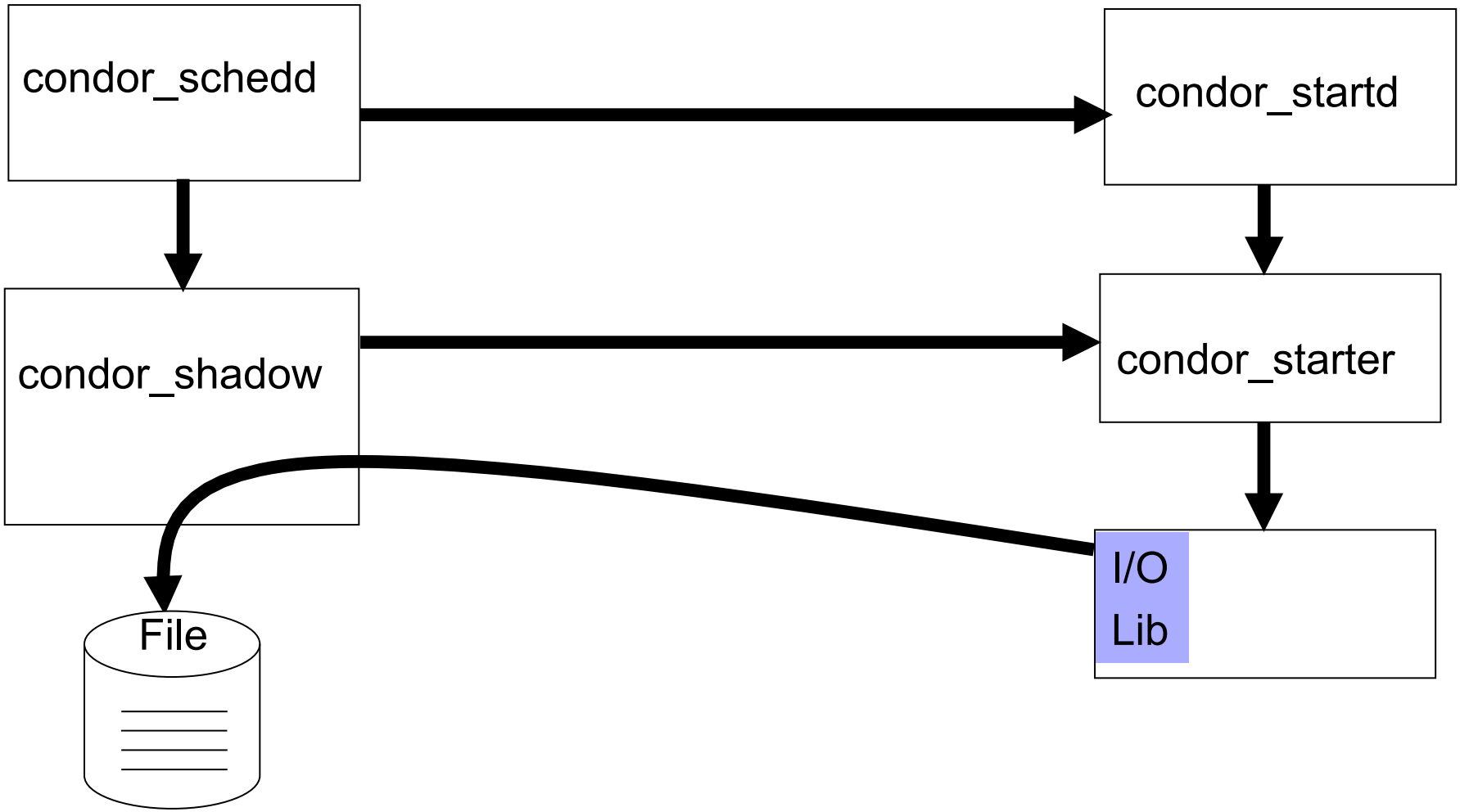
- Transfer_input_files can be a URL
  For example:

```
transfer_input_files = http://www.example.com/input.data
```

Standard Universe: Remote System Calls

- When your job accesses a file, we *trap* the access and do it on the submit computer
  - No file transfer needed
  - Can be faster if you don't access the whole file
- An essential part of checkpointing:
  - We know the state of the job, including the state of its files
  - We can re-run a job on a different machine without losing access to the files
- No source code changes required

Remote I/O


condor_schedd
condor_startd
condor_shadow
condor_starter
I/O
Lib
File

So Scot…

- So far we've run one job at a time.
  Just **one** job.
  ONE.
  This doesn't seem very "high-throughput"

Clusters & Processes

- One submit file can describe lots of jobs
  - All the jobs in a submit file are a *cluster* of jobs
  - Yeah, same term as a cluster of computers
- Each cluster has a unique “cluster number”
- Each job in a cluster is called a “process”
- A Condor “job ID” is the cluster number, a period, and the process number (“20.1”)
- A cluster is allowed to have one or more processes.
  - There is always a cluster for every job

Example Submit Description File for a Cluster

Open Science Grid

```
# Example submit description file that defines a
# cluster of 2 jobs with separate working directories
Universe   = vanilla
Executable = my_job
log        = my_job.log
Arguments  = -arg1 -arg2
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
InitialDir = run_0
Queue
InitialDir = run_1
Queue
```

Becomes job 2.0

Becomes job 2.1

Submitting The Job

```
% condor_submit my_job.submit-file
Submitting job(s).
2 job(s) submitted to cluster 2.

% condor_q
-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  2.0    frieda          4/15 06:56  0+00:00:00 I  0   0.0  my_job
  2.1    frieda          4/15 06:56  0+00:00:00 I  0   0.0  my_job
2 jobs; 2 idle, 0 running, 0 held
```

Open Science Grid

The $(Process) macro

- The initial directory for each job can be specified as run_$(Process), and instead of submitting a single job, we use “Queue 600” to submit 600 jobs at once
- The $(Process) macro will be expanded to the process number for each job in the cluster (0 - 599), so we’ll have “run_0”, “run_1”, … “run_599” directories
- All the input/output files will be in different directories!

Example of $(Process)

```
# Example condor_submit input file that defines
# a cluster of 600 jobs with different directories
Universe   = vanilla
Executable = my_job
Log        = my_job.log
Arguments  = -arg1 -arg2
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
InitialDir = run_$(Process)
Queue 600
```

run_0 … run_599
Creates job 3.0 … 3.599

Open Science Grid

More $(Process)

- You can use $(Process) anywhere:

```
Universe   = vanilla
Executable = my_job
Log        = my_job.$(Process).log
Arguments  = -randomseed $(Process)
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
InitialDir = run_$(Process)
Queue 600
```

Sharing a directory

- You don't have to use separate directories.
- $(Cluster) will help distinguish runs

```
Universe   = vanilla
Executable = my_job
Arguments  = -randomseed $(Process)
Input      = my_job.input.$(Process)
Output     = my_job.stdout.$(Cluster).$(Process)
Error      = my_job.stderr.$(Cluster).$(Process)
Log        = my_job.$(Cluster).$(Process).log
Queue 600
```

Complex sets of jobs

- What if you have a large set of jobs?
  - Different jobs have different purposes
    But all are needed for your science
  - Some jobs must run before others
- Hang tight, we'll talk about this after lunch

Open Science Grid

Job Priorities

- Are some of the jobs more interesting than others?
- `condor_prio` lets you set the job priority
  - Priority relative to your jobs, not other peoples
  - Priority can be any integer
- Can be set in submit file:
  - `Priority = 14`

Open Science Grid

Advanced Trickery

- Pretend: You submit several large sets of jobs
- These jobs are of two different types
  - e.g. analysis vs. simulation
- How can you look just the analysis jobs?

Advanced Trickery cont.

- In your job file:

```
+JobType = "analysis"
```

- You can see this in your job ClassAd

```
condor_q -l
```

- You can show jobs of a certain type:

```
condor_q -constraint 'JobType == "analysis"'
```

- Try this during the exercises!
- Be careful with the quoting…

Open Science Grid

Time for more exercises!

Open Science Grid

Questions?

- Questions? Comments?
- Feel free to ask me questions later:

  Scot Kronenfeld <kronenfe@cs.wisc.edu>
- Upcoming sessions
  - Now – 13:00
    - Hands-on exercises
  - 13:00 – 14:30
    - Lunch