National e-Science Centre

Tutorial 4: Condor

John Watt, National e-Science Centre

Tutorials Timetable

Week	Day/Time	Topic	Staff
3	Fri 11am	Introduction to Globus	J.W.
4	Fri 11am	Globus Development	J.W.
5	Fri 11am	Globus Development	J.W.
6	**Fri 11am**	**Condor**	**J.W.**
7	Tue 12pm	SAML/PERMIS (L)	A.S.
7	Wed 12pm	Portals (L)	J.J.
7	Fri 11am	Q & A Session	all
8	Fri 11am	OGSA-DAI (L)	O.A.
10	Tue 12pm	Example Systems (L)	R.S.
10	Fri 11am	Assignment Demos	all

What Is Condor?

- a batch scheduling system
  - allows submission and processing of batch jobs
- a cycle harvesting system
  - carries out computation when processor is idle
- a workload management system
  - allows user to prioritise jobs etc
- can be installed on desktop machines and clusters too

What Is Condor?

- **Developed mainly at Uni. Of Wisconsin, USA**
  - **Free software**
    - Research tool
  - **Binaries available**
    - No source
    - Multi-platform
  - **First version – 1988**
  - **~40 developers**
    - Staff AND students

http://www.cs.wisc.edu/condor

Condor

- Condor converts collections of distributively owned workstations and dedicated clusters into a distributed high-throughput computing (HTC) facility.
- Condor manages both resources (machines) and resource requests (jobs)
- Condor has several unique mechanisms such as :
  - ClassAd Matchmaking
  - Process checkpoint/ restart / migration
  - Remote System Calls
  - Grid 'Awareness'
- Collection of Condor resources is known as a 'pool'

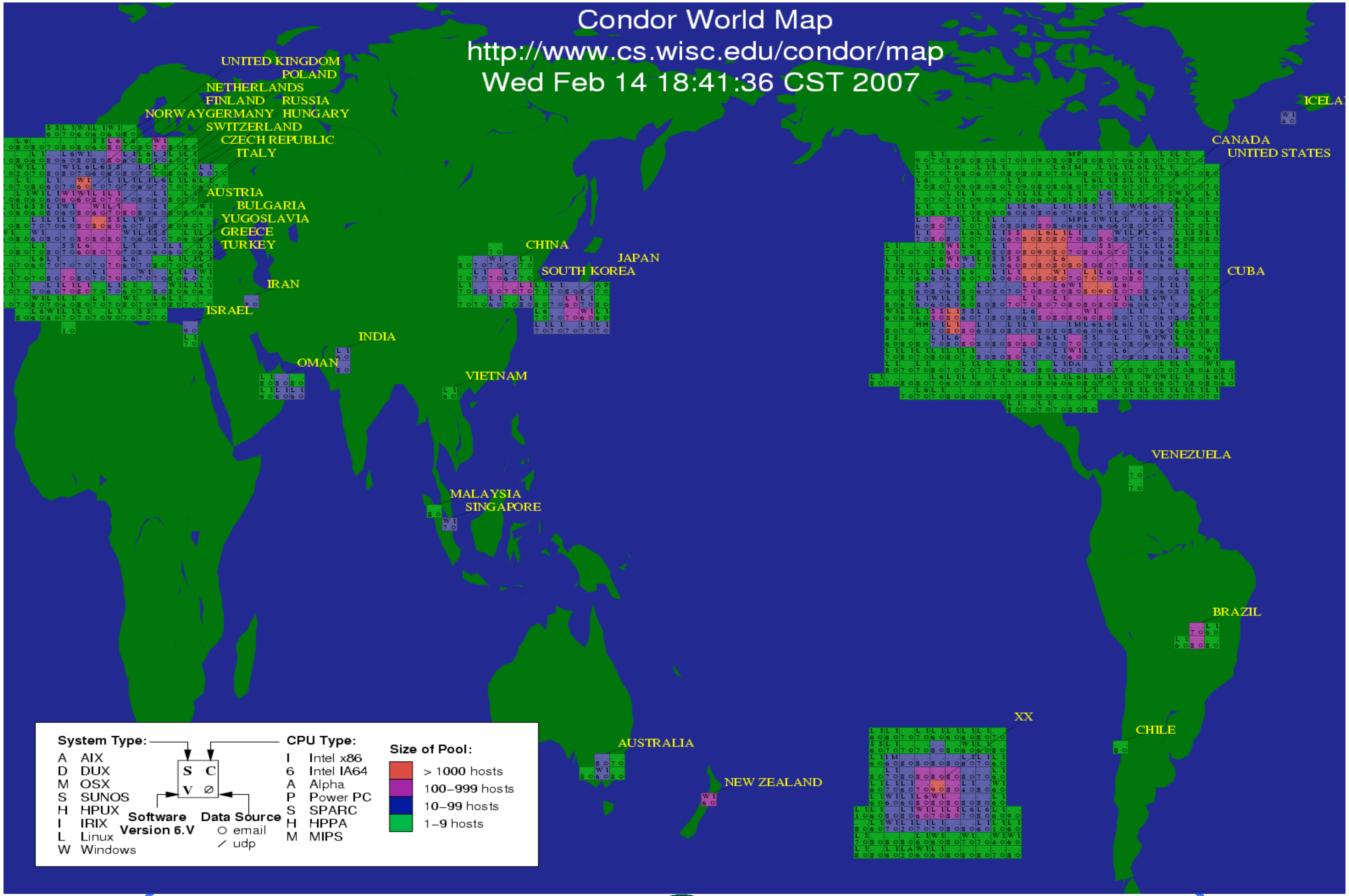
Job Management

- Managing a large number of jobs
  - You specify the jobs in a file and submit them to Condor, which runs them all and keeps you notified on their progress
  - Mechanisms to help you manage huge numbers of jobs (1000's), all the data, etc.
  - Condor can handle inter-job dependencies (DAGMan)
  - Condor users can set job priorities
  - Condor administrators can set user priorities

Resource Types

- Dedicated Resources:
  - Compute Clusters
- Manages
  - Node monitoring, scheduling
  - Job launch, monitor & cleanup

- Non-dedicated resources:
  - Desktop workstations in offices
  - Workstations in student labs
- Non-dedicated resources are often idle ~70% of the time!
  - Condor can effectively harness the otherwise wasted compute cycles from non-dedicated resources



UNIVERSITY *of* GLASGOW

e-Science dti

National e-Science Centre

Condor Pool

- A machine in a Condor pool can have several roles:
  - Central Manager – coordinates all activity (only one per pool), matches jobs with machines, keep tab on status of pool etc.
  - Submit machine – users submit jobs here
  - Worker machine – runs jobs

- These roles are implemented by specific daemons…

Condor Master

- Runs on ALL the machines ALL the time
- Spawns all the other daemons
  - With monitoring and restart if any crash
- Daemons reconfigured from the command line
  - `condor_on/condor_off`
    - Starts/stops a condor resource (but master still runs)
    - Put –master switch to switch off master daemon
  - `condor_reconfig`
    - Reconfigure and reload the master daemon
- Central Manager can control daemons on other pool nodes

Condor Schedd

- Runs on all machines that can SUBMIT jobs
- ‘shadow’ process spawned by schedd
  - When job is submitted, the condor_shadow daemon starts which monitors the job, controls file I/O and handles remote calls
- Schedd represents job requests to the pool
  - `condor_rm` – remove a job from the queue
  - `condor_q` – look at current queue
  - `condor_submit` – submit a job to the queue

Condor Startd

- Runs on all machines that can RUN jobs
- Startd advertises machines attributes to the central manager
  - For subsequent job matching
- Startd spawns a ‘starter’ process when sent job
  - Sets up environment and runs job
  - starter communicates with shadow process on submit machine

  - *Note starter and shadow only exist for the lifetime of the job*

Condor Collector

- Runs only on the Central Manager
  - Collects information about the pool
- All other daemons in the pool report to the collector periodically
  - ClassAds are advertised here
- Collector is queried with the `condor_status` command
  - `condor_status –l` - shows machine ClassAds
  - Condor_status summarises whether machine is busy, idle, matched etc

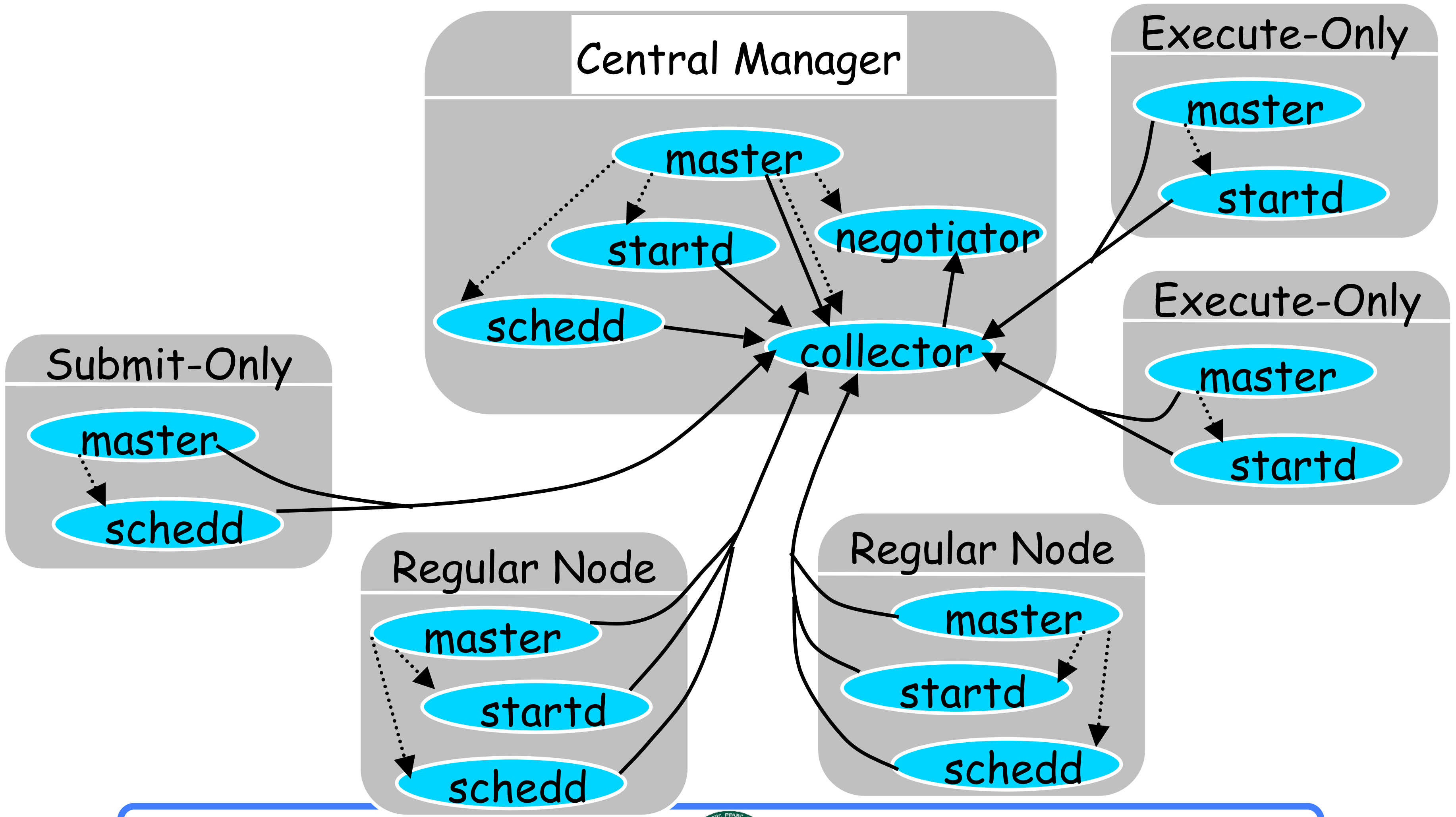
Condor Negotiator

- Runs only on the Central Manager
  - The ‘backbone’ of Condor
- Negotiator responsible for job-to-machine matching (ClassAd matching)
  - Queries the collector periodically for the status of the Condor pool
- Contacts the schedd daemon on each machine with waiting job requests
  - And matches to resources which are suitable

Other Roles

- Nodes in the pool can have one or more roles depending on which combination of the daemons are running
  - schedd + startd = can submit jobs and run jobs
  - schedd only = can submit jobs only
  - startd only = pure worker node, no job submission
  - the central manager itself can also be a submit machine and worker node but this is not recommended

Condor Node Roles



ClassAds

- Condor uses ClassAd Matchmaking to make sure that work gets done within the constraints of both users and owners.
- Users (jobs) have constraints:
  - “I need an Alpha with 256 MB RAM”
- Owners (machines) have constraints:
  - “Only run jobs when I am away from my desk and never run jobs owned by John.”
- Semi-structured data --- no fixed schema

Machine ClassAds

- "The Job Centre"
  - Advertises machines resources to the pool
  - View with `condor_status` command

```
Name          OpSys       Arch   State     Activity   LoadAv Mem   ActvtyTime

labpc-11.nesc LINUX       INTEL  Unclaimed Idle       0.000   495  0+02:05:04
labpc-12.nesc LINUX       INTEL  Unclaimed Idle       0.000   495  0+01:55:04
labpc-13.nesc LINUX       INTEL  Unclaimed Idle       0.000   495  0+01:00:05
labpc-14.nesc LINUX       INTEL  Unclaimed Idle       1.000   494  0+02:00:04
labpc-15.nesc LINUX       INTEL  Unclaimed Idle       0.000   494  0+00:05:04
labpc-16.nesc LINUX       INTEL  Unclaimed Idle       0.000   494  0+00:50:04
labpc-18.nesc LINUX       INTEL  Unclaimed Idle       0.000   494  0+01:20:04
labpc-2.nesc. LINUX       INTEL  Unclaimed Idle       0.000   494  0+02:15:04
labpc-20.nesc LINUX       INTEL  Claimed   Busy       0.000   494  0+03:45:04

                    Total Owner Claimed Unclaimed Matched Preempting Backfill

         INTEL/LINUX    9     0       1         8       0          0        0

               Total    9     0       1         8       0          0        0
```

Job ClassAds

- Workers “Curriculum Vitae”
  - Advertises job’s requirements to the pool
  - View job status with `condor_q`
  - `condor_q –long` shows entire ClassAd

```
-- Submitter: labpc-12.nesc.gla.ac.uk : <130.209.58.162:43501> : labpc-
12.nesc.gla.ac.uk
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
200.1    jones            5/12 11:51    +00:41:20 R   0  7.1 executable.exe

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

Condor Configuration

- A central config file maintains global config parameters for the whole pool

```
/opt/condor-6.8.3/etc/condor_config
```

- In addition, a local config file allows the owner of the machine to set parameters that override the global settings

```
/opt/condor-6.8.3/local.labpc-
  12/condor_config.local
```

- This allows user to stay firmly in control and to not have Condor jobs swamp his machine

Local Configuration

- Parameters can be set in the local config files that make jobs run always and straight away

```
START = True
RANK  =
SUSPEND   = False
CONTINUE   = True
PREEMPT = False
KILL  = False
```

- this sort of configuration means that jobs will run even if the user is working – may cause performance degradation
- most suitable for cluster or quiet pool which mainly runs jobs

Getting Started!

- Choosing a "Universe" for your job
  - Just use VANILLA for tests
  - Will need JAVA for assignment…
- Make your job "batch-ready"
  - Code preparation
- Creating a 'submit description' file
- Run `condor_submit` on your submit description file

Condor Universes

- A ‘Universe’ is an execution environment
  - Standard
  - Vanilla
  - MPI
  - Java
  - Globus etc…
- use vanilla if no source code available
- use standard if source code available
  - Provides checkpointing
  - Needs linked against condor libraries

Code Preparation

- Must be able to run in the background
  - no interactive input, windows, GUI, etc.
- Can still use `STDIN`, `STDOUT`, and `STDERR` (the keyboard and the screen)
  - but files are used for these instead of the actual devices
- Organize data files
  - We are NOT using a shared filesystem
  - So files and data will have to be moved about

Job Submission

- Jobs are submitted by putting instructions into a submit script and then executing

```
condor_submit <scriptname>
```

- Example script:

```
universe                    = vanilla
executable                  = sh_loop
output                      = sh_loop.out
error                       = sh_loop.err
log                         = sh_loop.log
arguments                   = 60
should_transfer_files       = IF_NEEDED
when_to_transfer_output     = ON_EXIT
queue
```

`condor_submit`

- You give `condor_submit` the name of the submit file you have created
- `condor_submit` parses the file, checks for errors, and creates a “ClassAd” that describes your job(s)
- Sends your job’s ClassAd(s) and executable to the condor schedd, which stores the job in its queue
  - Atomic operation, two-phase commit
- View the queue with `condor_q`

Clusters and Processes

- If your submit file describes multiple jobs, we call this a “cluster”
- Each job within a cluster is called a “process” or “proc”
- If you only specify one job, you still get a cluster, but it has only one process
- A Condor “Job ID” is the cluster number, a period, and the process number (“23.5”)
- Process numbers always start at 0

`condor_rm`

- If you want to remove a job from the Condor queue, you use `condor_rm`
- You can only remove jobs that you own (you can't run `condor_rm` on someone else's jobs unless you are root)
- You can give specific job ID's (cluster or cluster.proc), or you can remove all of your jobs with the "*-a*" option.

condor_history

- Once your job completes, it will no longer show up in `condor_q`
- You can use:

```
condor_history
```

- to view information about a completed job

- The status field ("ST") will have either a "C" for "completed", or an "X" if the job was removed with condor_rm

Condor_prio

- `condor_prio` allows you to specify the order in which your jobs are started
- Higher the prio #, the earlier the job will start

```
% condor_q
-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  1.0    frieda           6/16 06:52  0+00:02:11 R  0   0.0  my_job
% condor_prio +5 1.0
% condor_q
-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
  1.0    frieda           6/16 06:52  0+00:02:13 R  5   0.0  my_job
```

Job Controls

- Use `condor_hold` to place a job on hold
  - Kills job if currently running
  - Will not attempt to restart job until released
- Use `condor_release` to remove a hold and permit job to be scheduled again
- Recommend using a logfile in your submit description
  - Good for debug…

Sample Logfile

```
000 (8135.000.000) 05/25 19:10:03 Job submitted from host:
<128.105.146.14:1816>
...
001 (8135.000.000) 05/25 19:12:17 Job executing on host:
<128.105.165.131:1026>
...
005 (8135.000.000) 05/25 19:13:06 Job terminated.
        (1) Normal termination (return value 0)
                Usr 0 00:00:37, Sys 0 00:00:00  -  Run Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:05  -  Run Local Usage
                Usr 0 00:00:37, Sys 0 00:00:00  -  Total Remote Usage
                Usr 0 00:00:00, Sys 0 00:00:05  -  Total Local Usage
        9624   -  Run Bytes Sent By Job
        7146159  -  Run Bytes Received By Job
        9624   -  Total Bytes Sent By Job
        7146159  -  Total Bytes Received By Job
...
```

The Standard Universe and Checkpointing

- Condor’s Process Checkpointing mechanism saves all the 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…

condor_compile

- You need to relink your job for submission to the 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
OR
condor_compile make –f MyMakefile
```

Limitations of 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
  - <u>WE WILL NOT NEED CHECKPOINTING!</u>

The Java Universe

- Condor supports Java applications
  - `condor_submit java.cmd`

Java.cmd:

```
universe java
executable Main.class
arguments Main arg1 InputFile arg2
output Outfile
error ErrFile
queue 6
```

The Java Universe

- **Can submit jobs in vanilla, but…**
  - **Java Universe provides more than just inserting "java" at the start of the execute line**
    - Knows which machines have a JVM installed
    - Knows the location, version, and performance of JVM on each machine
    - Provides more information about Java job completion than just JVM exit code
      - Program runs in a Java wrapper, allowing Condor to report Java exceptions, etc.

```
condor_submit –java
```

- Shows Java supported nodes in your pool

Command Summary

condor_status	View Pool Status
condor_q	View Job Queue
condor_submit	Submit new Jobs
condor_rm	Remove Jobs
condor_prio	Intra-User Prios
condor_history	Completed Job Info
condor_compile	Link Condor library

Finally

- **Look in your $HOME/examples folder**
  - **There are several test applications for use with Condor**
  - **Try running the `sh_loop` job…**
    - Try submitting multiple jobs
    - Monitor them with condor_q, condor_status etc..

- **Next week – 3 tutorials**
  - **SAML/PERMIS**
  - **Portals**
  - **Q & A**