CERN

HTCondor Training

Florentia Protopsalti IT-CM-IS

Overview

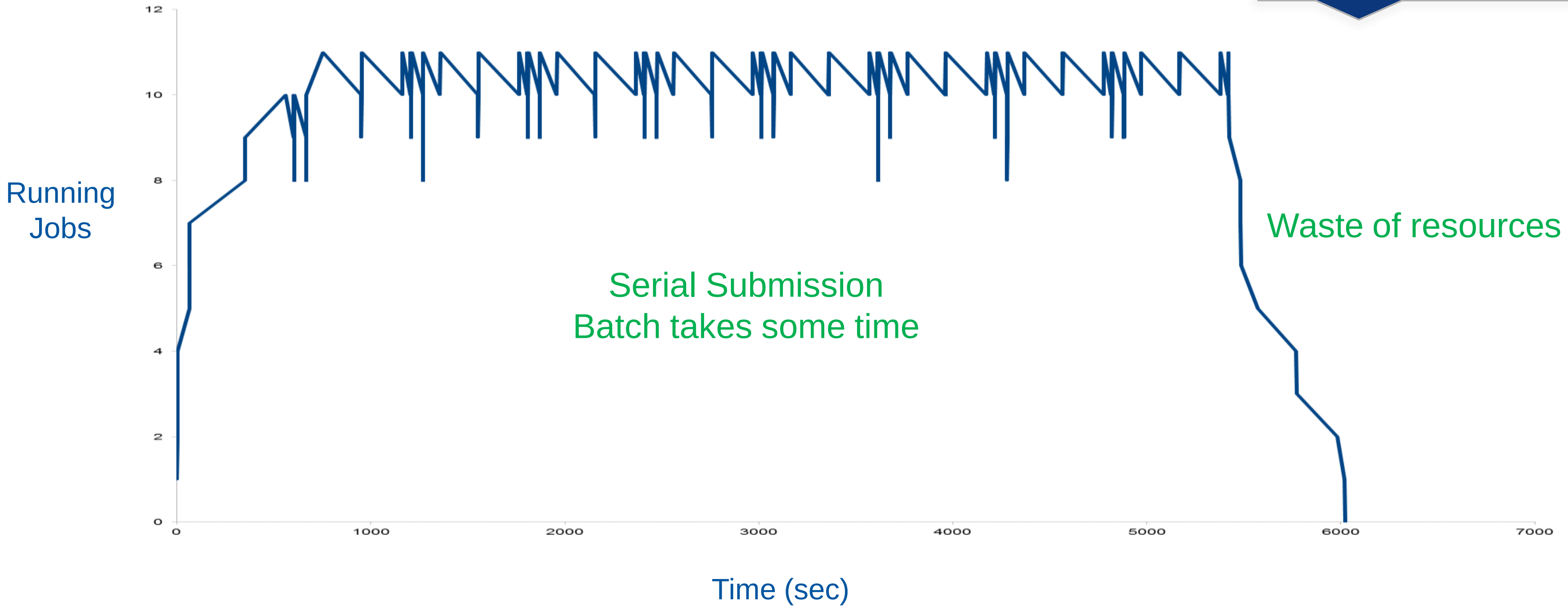
- HTCondor Batch System
- Job Submission
- Investigating Failed Jobs
- Input And Output Files
- Exercises

HTCondor Batch System

Machine Ownership

1K * 100 1h jobs = 100K CPU hours ≈ 11.4 job slots

Submission of 100 jobs – 10 machines

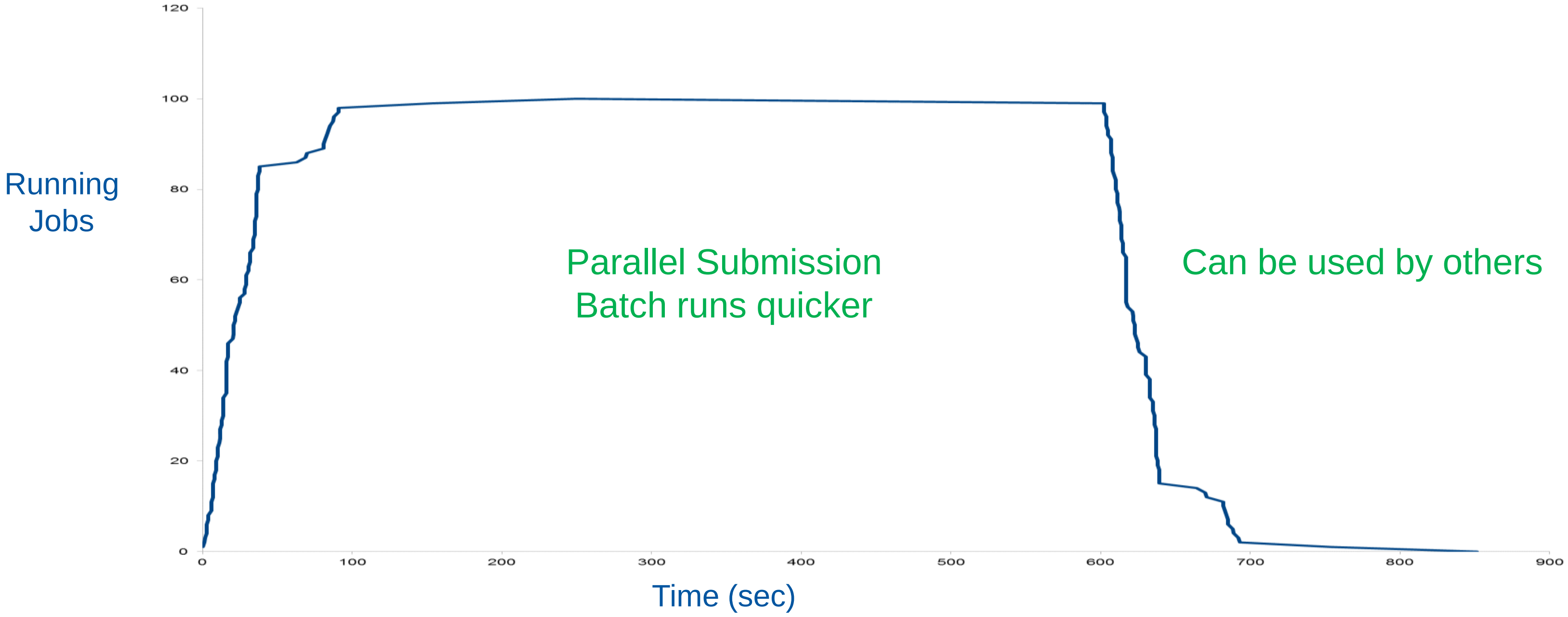


Waste of resources

Serial Submission
Batch takes some time

Timesharing

Submission of 100 jobs – 100 machines



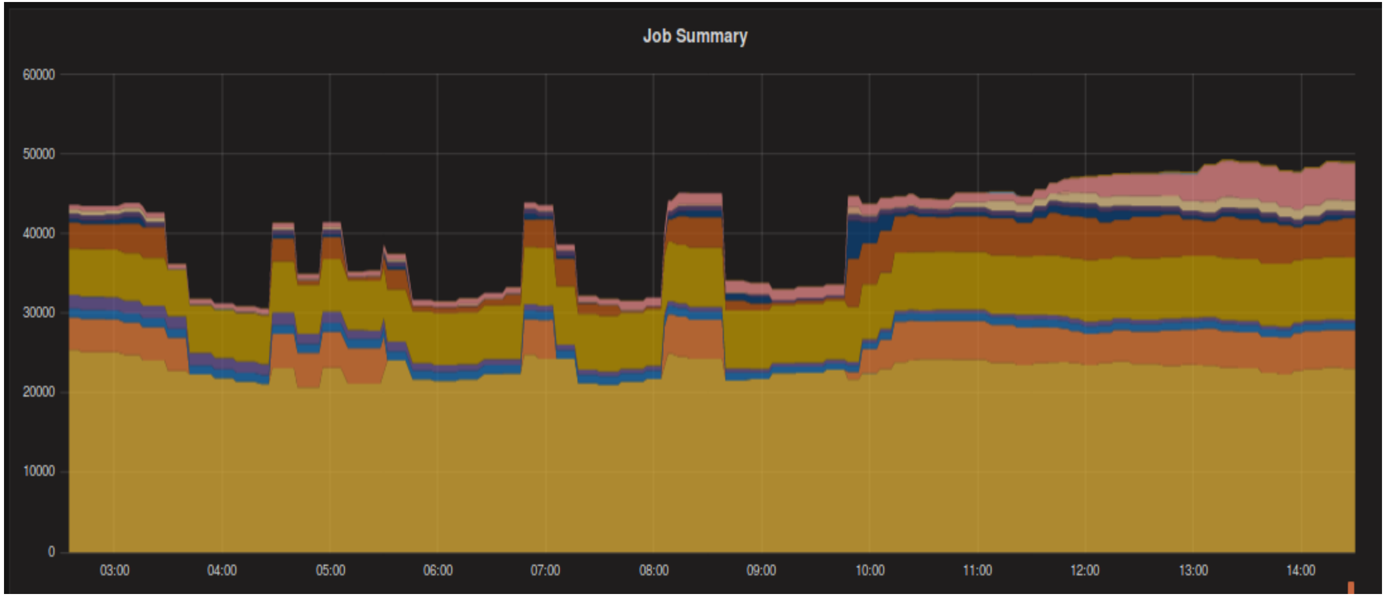
Parallel Submission
Batch runs quicker

Can be used by others

Batch Scheduling

CERN Batch System

Running Jobs



Time (hours)

CERN Batch Service

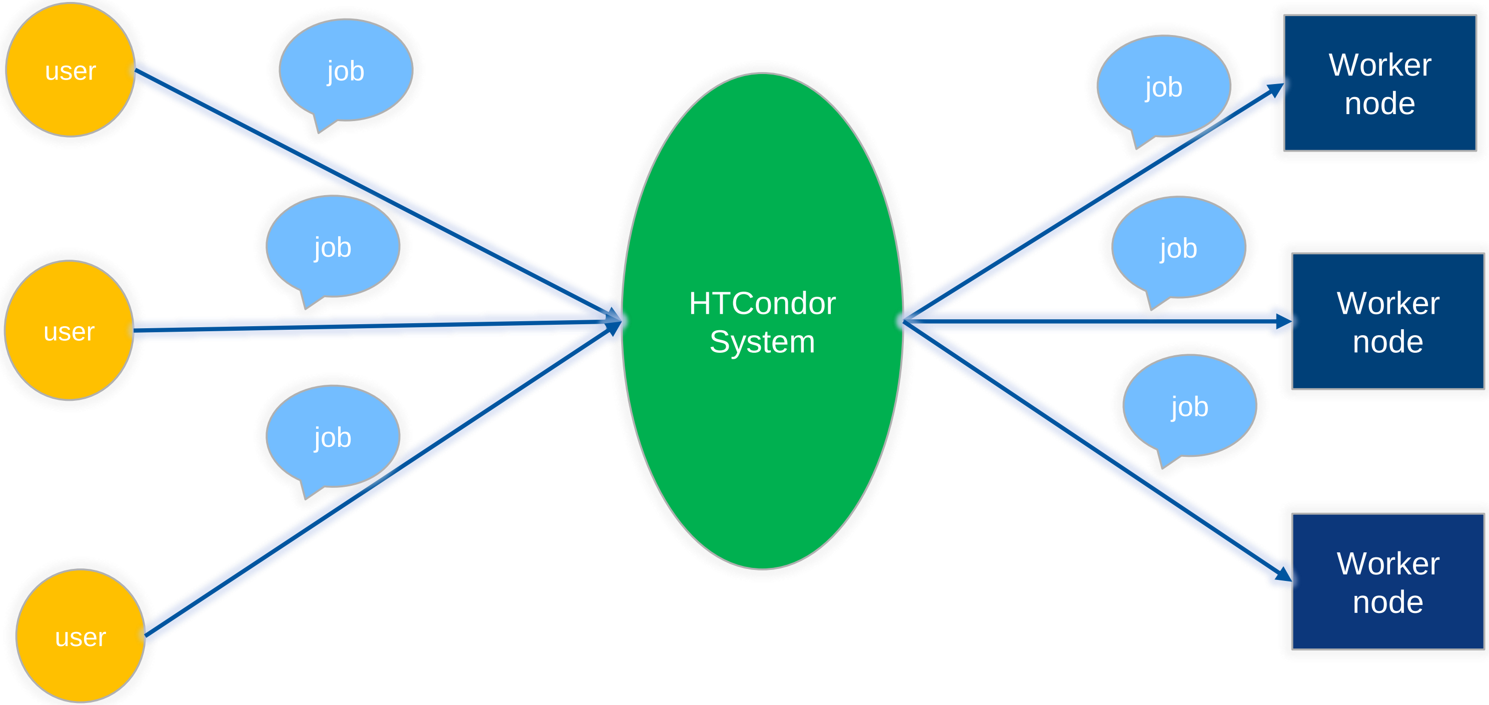
- Delivers computing resources
  - To the experiments and departments for tasks e.g.
    - Physics event reconstruction
    - Data analysis
    - Simulation
- It shares the resources fairly between all users
- Current capacity approximately 120,000 cores

HTCondor

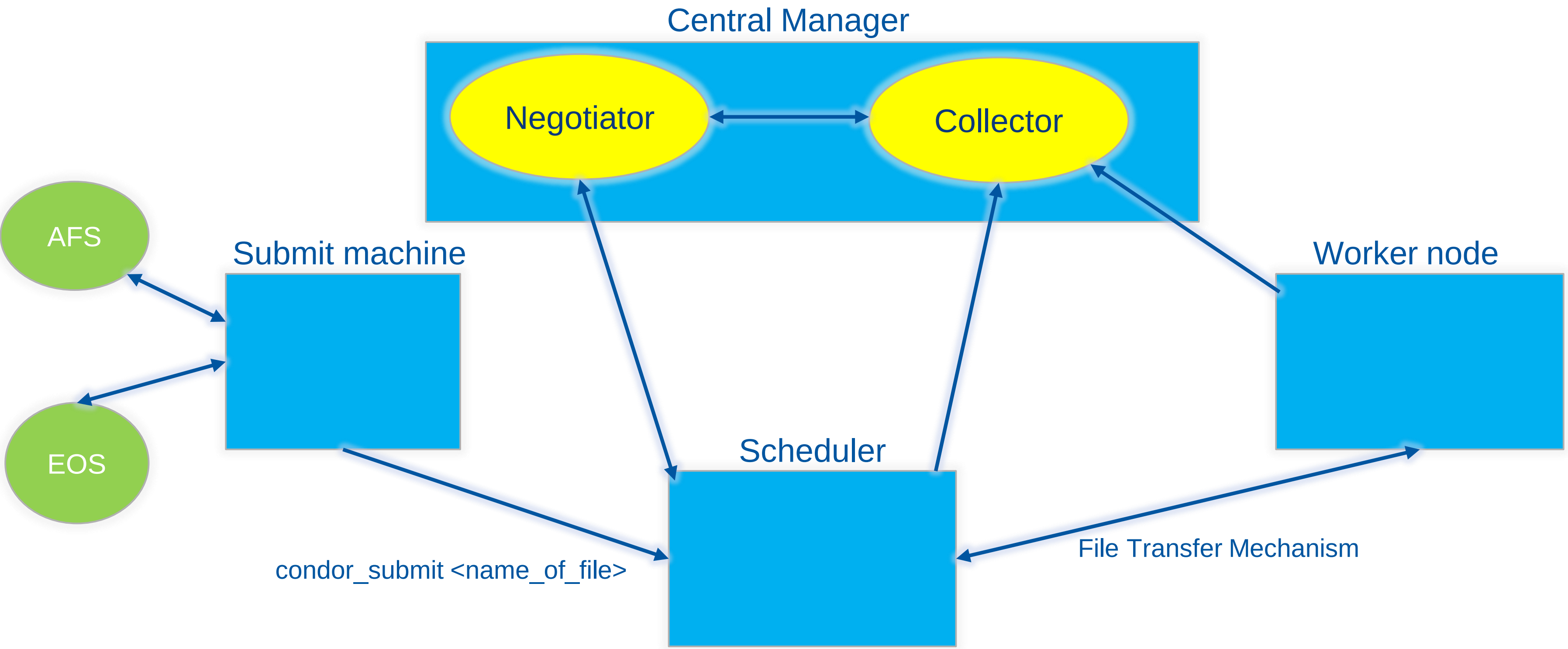
- Open-source batch system implementation
  - Center for High Throughput Computing
    - University of Wisconsin–Madison.
- It provides
  - Job queueing mechanism
  - Scheduling policy
  - Resource monitoring
  - Resource management

http://research.cs.wisc.edu/htcondor/manual/

HTCondor Workflow



HTCondor Service Components

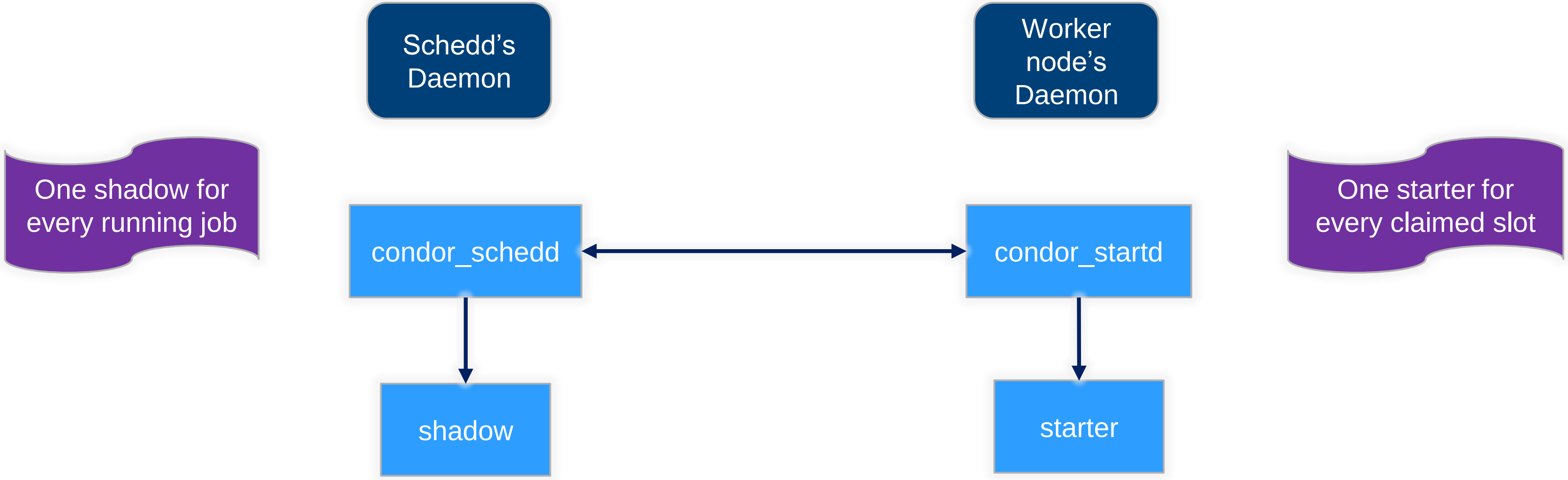


The Central Manager

- Composed of the Collector and Negotiator daemons.

- The Collector
  - Collects information on machines in the pool
  - Collects information on the jobs in the queue
  - Collecting information from all the daemons in the pool
  - It accepts queries from other daemons and user-level command (e.g. condor_q)
- The Negotiator
  - Negotiates between machines and machines requests (job)
  - Asks for a list with all the available machines from the collector
  - Matches jobs and machines considering the job requirements

Negotiator: Matchmaking



ClassAds

- The framework by which Condor matches jobs with machines
    - They are analogous to the classified advertising section of the newspaper
        - users submitting jobs are *buyers* of compute resources
        - machine owners are *sellers*.
- Used for
    - Describing and advertising
        - Jobs
        - Machines
- Matching jobs to machines
- Statistical purposes
- Debugging purposes

Example ClassAds

Job ClassAd

```
AccountingGroup = “name of accounting group"
ClusterId = 9
Cmd = "/afs/cern.ch/…/welcome.sh"
CompletionDate = 0
CondorPlatform = "$CondorPlatform: x86_64_RedHat7 $"
CondorVersion = "$CondorVersion: 8.5.8 Dec 13 2016 BuildID: 390781 $"
DiskUsage = 1
EnteredCurrentStatus = 1493728837
Err = "error/welcome.9.0.err"
ExitBySignal = false
ExitStatus = 0
FileSystemDomain = "cern.ch"
GlobalJobId = "bigbird06.cern.ch#9.0#1493728837"
Hostgroup = "$$(HostGroup:bi/condor/gridworker/share)"
JobPrio = 0
JobStatus = 1
JobUniverse = 5
NumJobCompletions = 0
NumJobStarts = 0
NumRestarts = 0
```

Machine ClassAd

```
COLLECTOR_HOST_STRING = “*.cern.ch, *.cern.ch"
CondorLoadAvg = 0.0
CondorPlatform = "$CondorPlatform: x86_64_RedHat6 $"
CondorVersion = "$CondorVersion: 8.5.8 Dec 13 2016 BuildID: 390781 $"
Cpus = 8
FileSystemDomain = "cern.ch“
JobStarts = 156
Machine = "b658ea5902.cern.ch“
Memory = 22500
RecentJobStarts = 0
SlotType = "Partitionable"
SlotTypeID = 1
SlotWeight = Cpus
Start = ( StartJobs =?= true ) && ( RecentJobStarts < 5 ) && ( SendCredential =?= true )
StartJobs = true
TotalMemory = 22500
TotalSlotCpus = 8
TotalSlotDisk = 223032980.0
TotalSlotMemory = 22500
TotalSlots = 1
```

Job Startup

1. Machines periodically send their ClassAds to the Collector
2. The user submits their job to the Schedd
3. The Schedd informs the Collector about the job
4. The Negotiator queries the Collector about waiting jobs and available machines
5. The Negotiator queries the Schedd about the job for the requirements
6. The Negotiator matches the job with a machine
7. The Schedd contacts the machine and each other

Job Submission

Submit File

- Provides commands on how to execute the job
  - Contains basic information about
    - The executable
    - The arguments
    - Paths for the input and output files
    - The number of the jobs in the queue
    - The names of the jobs in the queue

Condor Output And Logs

- These files are defined in the submit file
  - Output
    - The STDOUT of the command or script
  - Error
    - The STDERR of the command or script
  - Log
    - Information about job’s execution
      - execution host
      - the number of times this job has been submitted
      - the exit value, etc
- Can use relative or absolute paths for all of them
- HTCondor will search for this directory
  - So it should be already created

Requirements

- In the submit file job requirements can be set about
  - Operating System
  - Number of CPUs
  - Memory
  - Specific machines
- Also provide ClassAdd attributes for this job
  - Defined by using “+Name_Of_Variable”
    - E.g +JobFlavour = espresso

- The job is submitted by executing:

condor_submit <name_of_submit_file>

Progress of job submission

- To follow the progress of the job, execute:

condor_wait path_to_log-file [job ID]

- This watches the job event log file
  - Created with the log command within a submit description file
- Returns when one or more jobs from the log have completed or aborted
- It will wait forever for jobs to finish unless a shorter wait time is specified

condor_wait [-wait seconds] log-file [job ID]

Inspecting Queues

- The condor_q command queries the collector
  - For information about the jobs in the queue
- Arguments can be used to filter the jobs of interest
- Possible filters
  - cluster.process
    - Matches jobs in the same cluster.process that are still in the queue
  - owner
    - Matches jobs that are in the queue and they belong to this owner
  - -constraint expression
    - Matches jobs that satisfy this ClassAd expression

Example

```
-- Schedd: bigbird06.cern.ch : <128.142.194.67:9618?... @ 05/02/17 10:04:46

OWNER    BATCH_NAME           SUBMITTED   DONE   RUN    IDLE  TOTAL  JOB_IDS
fprotops CMD: welcome.sh   5/2  10:04  _        -             1           8.0


1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

But which are the possible status of the jobs?

Job States

Idle (I)	• The job is waiting in the queue to be executed
Running (R)	• The job is running
Completed (C)	• The job is exited
Held (H)	• Something is wrong with the submission file • The user executed condor_hold <job_id>
Suspended (S)	• The user executed condor_suspend
Removed (X)	• The job has been removed by the user.

Investigating Failed Jobs

Diagnostics With condor_q

- condor_q displays the current jobs in the queue
- To see the ClassAd of a specific Job Id, execute:

condor_q –l <JobId>

- A very useful option for debugging when a jo stays in idle state is:

condor_q –better <Job Id>

or

condor_q –analyze <Job Id>

- Both display the reason why a job is not running
- They perform an analysis with constraints, owner's preferences about the machines, etc.
- It *sometimes* provides also suggestions about the solution of the problem
- For a more detailed analysis of complex requirements and the job ClassAd attributes, execute:

condor_q –better-analyze <Job Id>

Investigating Jobs

- The reasons for Held jobs can be found with:

condor_q –hold <JobId>

- In the case where a machine is not accepting the job, execute:

condor_q -better –reverse <name of machine> <JobId>

- After the completion of the job condor_history can be used
  - It displays the information of the complete jobs from the history files
- To display the ClassAd of a specific completed Job, execute:

condor_history –l <JobId>

Diagnostics With condor_status

- condor_status queries the Collector asking for information about the machine
- Specifying the machine name as an argument
  - Display the slots, their state and their activity

```
          Name               OpSys    Arch    State    Activity LoadAv  Mem    ActvtyTime
slot1@b6c70d5f39.cern.ch     LINUX    X86_64  Owner    Idle      0.500  10500  0+00:02:47
slot1_1@b6c70d5f39.cern.ch   LINUX    X86_64  Claimed  Busy      0.000  2000   0+00:18:55
```

- To display the ClassAd of a specific machine, execute:

```
condor_status –l <name of the machine>
```

Input and Output Files

Input Files

- Only the executable is transferred
- To use other files the File Transfer Mechanism is required
- Input files are defined in the submit file by adding:

transfer_input_files= path

- The path can be absolute or relative
- Multiple files can be specified using comma separation

Output Files

- Files created or modified during job execution will be transferred back
- The number of output files transferred can be filtered adding in the submit file:

transfer_output_files= “name_of_file”

- This will search for the file in the scratch directory of the executing machine
- Multiple files can be specified using comma separation

- Note that this command is not related to the error, output and log files

Spooling Files

- Another way to filter the number of output files is by using the spool option

  condor_submit –spool <name of submit file>

- The files are stored in the schedd’s spool directory after the completion of the job.
- To transfer all or some of them back to the submit, execute:

  condor_transfer_data <name of User> || <JobId>

- This acts upon all files including the error, log and output files

Exercises

- These will help you to understand HTCondor
- And how we can use it correctly

Let’s play with HTCondor!!

http://cern.ch/htcondor