Integrating HTCondor with ARC

Andrew Lahiff,
STFC Rutherford Appleton Laboratory

HTCondor/ARC CE Workshop, Barcelona
ARC CE architecture

- (Simplified) architecture

- LDAP + BDII
- A-REX job interface
- GridFTP
- LRMS job management scripts
- infoprovider scripts
- job interface
- GridFTP server
- A-REX
- LDAP + BDII

client tools

info

jobs

files

A-REX=ARC Resource-coupled EXecution service
LRMS scripts

- LRMS job management scripts
  - submit-LRMS-job
    - does any required preparations (e.g. create job submission script)
    - submits jobs to the batch system
  - scan-LRMS-job
    - queries batch system about running, idle & completed jobs
    - extracts information about completed jobs
  - cancel-LRMS-job
    - kills jobs when requested by users

- Information system
  - LRMS.pm
    - generates information about the cluster as a whole
LRMS scripts

- In /usr/share/arc

  submit-boinc-job  submit-fork-job  submit-lsf-job
  submit-sge-job   **submit-condor-job**  submit-DGBridge-job
  submit-ll-job    submit-pbs-job    submit-SLURM-job
  cancel-boinc-job **cancel-condor-job**  cancel-fork-job
  cancel-lsf-job   cancel-sge-job    cancel-DGBridge-job
  cancel-ll-job    cancel-pbs-job    cancel-SLURM-job
  scan-boinc-job   scan-DGBridge-job  scan-ll-job
  scan-pbs-job     scan-SLURM-job    **scan-condor-job**
  scan-fork-job    scan-lsf-job     scan-sge-job

Boinc.pm  PBS.pm  SLURM.pm  SLURMmod.pm  LL.pm
DGBridge.pm  SGEmod.pm  SGE.pm  **Condor.pm**  LSF.pm
ARC-CE HTCondor history

- Some main points
  - < 4.1.0
    - partitionable slots not supported
  - 4.1.0 [2014]
    - partitionable slots supported, lots of fixes
  - 5.0.0 [2015]
    - scan-condor-job can read directly from per-job history files
    - job priority passed to HTCondor
Configuration

• The LRMS is specified in /etc/arc.conf

  [common]
  ...
  lrms="condor"
  ...

• ARC will submit jobs to the local schedd

• Other HTCondor-related configuration

  –condor_requirements: sets the requirements expression for jobs submitted to each queue, we use:
    • (Opsys == "linux") && (OpSysAndVer == "SL6")
  –condor_rank: sets the rank expression for jobs
Queues

- At RAL we’ve only ever had one queue per CE with HTCondor + ARC
  - Jobs need to request numbers of cores, memory, CPU time, wall time
- Can specify a HTCondor requirements expression for each queue
  - this gets added to jobs submitted to the queue
- Default requested memory can be taken from queue configuration
Job parameters passed to HTCondor

• Parameters
  – job description
  – number of cores
  – memory
  – job priority

• Limits (added to each job’s periodic remove expression)
  – memory limit (same as requested memory)
  – wall clock time limit
  – CPU time limit
Job parameters passed to HTCondor

• Example ATLAS job (fragment of job ClassAd)

... 
JobDescription = "mc15_13TeV_3618"
RequestCpus = 8
RequestMemory = 16000
JobPrio = -38
JobCpuLimit = 1571520
JobTimeLimit = 196440
JobMemoryLimit = 16384000
PeriodicRemove = false || RemoteUserCpu + RemoteSysCpu > JobCpuLimit || RemoteWallClockTime > JobTimeLimit || ResidentSetSize > JobMemoryLimit
...

...
Something you may want to hack

- /usr/share/arc/submit-condor-job
  - We’ve commented-out this line:
    
    \[
    \text{REMOVE} = "\{\text{REMOVE}\} || \text{ResidentSetSize} > \text{JobMemoryLimit}"
    \]
  - because
    - using cgroups with soft memory limit
    - our SYSTEM_PERIODIC_REMOVE expression will cause jobs using > 3x requested memory to be removed
Accounting

- APEL publisher node not required
  - JURA component of ARC sends accounting data directly to APEL central broker

- Scaling factors
  - Unlike some other batch systems (e.g. Torque), HTCondor doesn’t scale wall & CPU time
  - This is good!
    - limits on normalized time always confuse people
Accounting

• Worker node HTCondor configuration contains, e.g.
  
  \[
  \text{ScalingFactor} = 2.67 \\
  \text{STARTD_ATTRS} = \$(\text{STARTD_ATTRS}) \text{ ScalingFactor}
  \]

• On the CEs
  
  \[
  \text{MachineScalingFactor} = "\$(\text{[ScalingFactor]})"
  \text{SUBMIT_EXPRS} = \$(\text{SUBMIT_EXPRS}) \text{ MachineScalingFactor}
  \]

• ClassAds for completed jobs then contain:
  
  \[
  \text{MATCH_EXP}_{\text{MachineScalingFactor}} = "2.670000000000000E+00"
  \]

• An ARC auth plugin for jobs in the FINISHING state
  
  –scales CPU & wall time appropriately before JURA sends accounting data to APEL
Useful files

- `<control-dir>/job.<grid job id>.errors`
  - dump of the executable submitted to HTCondor
  - stdout/err from condor_submit
  - completed jobs: full job ClassAd + extracted information

- `<session-dir>/<grid job id>/log`
  - HTCondor job user log

- `/var/log/arc/gm-jobs.log`
  - log of jobs starting & finishing
  - includes grid job ID, local HTCondor job id, uid, DN
Monitoring

• Nagios checks & metrics for the consistency in the numbers of running/idle jobs between ARC & HTCondor
  – a sign of problems: scan-condor-job not able to keep up
ARC CEs around the world

• From the information system
  – 73 ARC CEs in total
  – 26 HTCondor
    • the rest are mostly SLURM, PBS/Torque