Condor at the RACF


W I L L I A M  S T R E C K E R - K E L L O G G
Upgrade to 7.6.x

- Move to 7.6.4 done in October time-frame for RHIC experiments
  - Everything went better than expected

- 7.6.6 for ATLAS done in February, also went smoothly

- Small experiments done with RHIC upgrade
  - A few hiccups—caused LSST (ASTRO) to abandon Condor in favor of a homegrown batch system
Repackage

- Why? Easy upgrades, configuration management
- One pitfall—CMake silently failing to find globus-libs at build time and building without support

- Requires:
  - globus-callout
  - globus-common
  - globus-ftp-client
  - globus-ftp-control
  - globus-gass-transfer
  - globus-gram-client
  - globus-gram-protocol
  - globus-gsi-callback
  - globus-gsi-cert-utils
  - globus-gsi-credential
  - globus-gsi-openssl-error
  - globus-gsi-proxy-core
  - globus-gsi-proxy-ssl
  - globus-gsi-sysconfig
  - globus-gssapi-error
  - globus-gssapi-gsi
  - globus-gss-assist
  - globus-io
  - globus-libtool
  - globus-openssl
  - globus-openssl-module
  - globus-rsl
  - globus-xio
  - globus-xio-gsi-driver
  - globus-xio-popen-driver

- Most have one library and a README
  - Instead build new condor-libs package
  - Out of standard library search paths & set RPATH
Repackage

- Move away from old way:
  - (tarball + path-twiddling) = new RPM
- New package buildable from any git snapshot of Condor repository—verified in SL5 & 6.
- CMake works (almost) perfectly—would not have been possible with previous build system
- Dynamic linking a huge plus
  - Size reduced from 177Mb → 44Mb compressed!
ASTRO (LSST) Condor Move

- Two problems—eventually caused a move away from Condor to home-grown batch system (for now).
  - First, wanted parallel universe with dynamic slots. Broken in 7.4.2 [#968]
    - Considered special whole-machine slot queue
      - $(DETECTED_CORES) + 1$ Slots, one weighted differently
    - Drawbacks incl. complexity and resource starvation in on relatively small farm (34 nodes)
Move to 7.6 brought promised change with dynamic slots and the parallel universe.

In 7.6.3—chirp bug, missing leading “/” in path names, caused MPI jobs to fail [#2630]

- Found workaround involving different MPI setup script and some software changes
- Fixed in 7.6.4(?)—too late for them:
  - Eventually gave up and wrote own system…
New Scales

- Single largest pool is ATLAS farm, ~13.5k slots!
- Negotiation cycle only 1 or 2 minutes
- `condor_status` takes a whole second!
- Group quotas help with negotiation cycle speed
- More small experiments in common pool:
  - `DAYABAY, LBNE, BRAHMS, PHOBOS, EIC, (formerly) ASTRO`—totals a few hundred CPUs.
  - WISC machines and dedicated OSG slots are still in the ATLAS pool
New Scales

- STAR pool has most user diversity, ~40 active users with lots of short running jobs
  - Negotiation cycle still only $O(5\text{min})$ without any limiting time per-user
  - Worst case many different Requirements
- PHENIX pool mostly runs with a few special users (reconstruction, simulation, and analysis-train).
- Wish for FIFO/Deadline option for reconstruction jobs
Hierarchical Group Quotas

- After upgrade to 7.6.6 moved ATLAS to HGQ
- More success had using ACCEPT_SURPLUS flag than was had with AUTO_REGROUP
- Behavior more stable, no unexplained jumps:

Even with queues supplied with ample Idle jobs, this sometimes happened with AUTO_REGROUP.
Hierarchical Group Quotas

- Nice organization and viewing of totals of each sub-group running; groups structured thus:

```
Hierarchical Structure:
- atlas
  - prod
    - test
    - cvmfs
    - mp8
  - software
  - analysis
    - short
    - long
```
ATLAS Multicore

- New queue (mp8) has hard-coded 8-core slots
- Just in testing, but some new requirements
  - Overhaul of monitoring scripts needed
    - Number of jobs running becomes weighted sum
    - Tested interplay with group quotas—some hiccups
- Will likely move to use dynamic slots if someday more than just 8-core jobs are desired
  - Interested in anyone’s experience with this
Configuration Management

- Done with a combination of Puppet, git, and homegrown scripts
  - Problems encountered on compute farm:
    - Certificate management
    - Node classification
    - Puppet master load
    - QA process
- Ultimate goal: use exported resources to configure location of each experiment's central manager
  - Config files, monitoring all updated automatically
  - Bring up a new pool with push-button ease
Poor Man’s Cloud

- Problem
  - We want users to be able to run old OS's after entire farm goes to SL6
  - Not to have to support one or two real machines of each old OS as legacy.

- Keep It Simple (Stupid)
  - With current hardware, nothing extra
  - Avoid using Open* etc...
  - Not an official cloud investigation, just a way to use virtualization to ease maintenance of legacy OS’s
Poor Man’s Cloud—Requirements

- Users cannot run images they provide in a NAT environment that does not map ports < 1024 to high ports—could edit our NFS(v3)!
  - Anything that uses UID-based authentication is at risk if users can bring up their own VM's

- Need access to NFS for data, user home directories, and AFS for software releases, etc...

- Cannot handle network traffic of transferring images without extra hardware (SAN, etc...)
Poor Man’s Cloud—Distribution

- Distribution done through a simple script that fetches/decompresses from webservice
- Allowed images listed in checksum file on webserver
  - Automatically downloads new images if out of date and re-computes the checksums.
- QCOW2 image created for each job with read-only backing store of local image copy
  - Diffs get written in condor’s scratch area (or we setup read-only-root in our images)
Poor Man’s Cloud—Instantiation

- Instantiation done by same setuid-wrapper after potential image-refresh.
- Wrapper execs program that uses libvirt/qemu to boot an image

  - First guest-fish writes a file with the user to become and a path to execute
    - Information comes from job description
    - Wrapper has rc.local that becomes user and executes the script as passed into the job
Poor Man’s Cloud—Getting Output

- Most likely place is NFS—users can run the same code and write to the same areas as they would in a non-virtual job.

- Wrapper can optionally mount a data-disk (in scratch area) that is declared as condor job output.
  - Future extension to untrusted VM’s would require port-redirection and only allow output this way.
    - Input provided in similar manner or via file-transfer-hooks and guest-fs injection.
With addition of LIBVIRT_XML_SCRIPT option using the VM universe for instantiation becomes possible.

Use of guest-fs to inject user code and actual instantiation can be done by Condor now.

- Restrictions on which VM’s are trusted can be managed in this script.

Still need setuid wrapper to do image-refresh.

- Use a pre-job-wrapper or just require it of the users.
Thanks!