

Condor in LIGO Festivus-Style

Peter Couvares

Syracuse University, LIGO Scientific Collaboration

Condor Week 2013

1 May 2013

Happy
Festivus!



Who am I? What is LIGO?

- Former Condor Team member ('99-'08).
- Now at Syracuse University focused on distributed computing problems for the LIGO Scientific Collaboration.
- LIGO (the Laser Interferometer Gravitational-Wave Observatory) is a large scientific experiment to detect cosmic gravitational waves and harness them for scientific research. <http://ligo.org/>

Feats of Strength

- LIGO involves amazing science, multiple sites, tens of thousands of cores, hundreds of GPUs, hundreds of active users, gigantic DAGS with millions of jobs, PBs of data.
- LIGO loves Condor. See past Condor Week talks by Duncan Brown, Scott Koranda, myself, and others.
- But I'm here today to be a PITA...

Airing of Grievances

- Condor works so well in LIGO that people notice—and mind—when it doesn't.
- Luckily I'm not the "Condor IT guy" for all of LIGO but nonetheless many issues end up in my court, and this is my attempt to generalize and pose some questions.
- Growing pain points for LIGO:
 - Data-intensive workloads.
 - Diagnosing failures.
 - Priority/Policy configuration.
 - Job factories.
 - GPUs.
 - Checkpointing.
 - Dynamic slots.

Data-Intensive Workloads

- Old, hard problem.
- We have multiple big fast NFS servers serving each cluster. Worked well for a long time.
- Core count increasing faster than NFS throughput. Users increasingly killing file servers, causing confusion and lost CPU and human time.
- We may be reaching the limits of centralized NFS, but any i/o system has its limits – can Condor better help us manage the limits we have, and/or deal with the i/o failures we experience?

Diagnosing Failures

- With larger clusters, more random things go wrong (fileserver outages, user errors, edge-case Condor bugs, etc.).
- It seems like most of them require a Condor expert to debug.
- Debugging process is often the same: user complains that their job won't run, or won't complete for whatever reason:
 - Find job in queue, run `condor_q -analyze`.
 - Grep `SchedLog` and `NegotiatorLog` for relevant info.
 - Find node it ran on, if it ran.
 - Grep `ShadowLog` for relevant info.
 - Grep `StartLog` and `StarterLog.slotN` for relevant info.
 - Turn up debugging level and ask user to call back when it recurs.
 - If debugging level was already up, info is gone because logs rolled. :-/
- Can't some of this be automated?
 - `condor_gather_info!`
- Can there be better info propagation from daemons to end-user?

Priority/Policy Configuration

- I'm not afraid of hairy policy config (see Bologna Batch System), but it would be nice to have some help.
- SU "Sugar" Pool Policy from 10,000 feet:
 - Older compute nodes
 - local LIGO users > LIGO users > others > backfill
 - Newer compute nodes (2064 cores)
 - CPUs: 2/3 LIGO users + 1/3 LHCb users > others > backfill
 - GPUs: local LIGO users > LIGO users > others > backfill
 - Newest compute nodes (144 cores)
 - CMTG > local LIGO users > others > backfill
- This takes many screens of clever config file magic to implement. (And may need to be rewritten for dynamic slots.)
 - Not to mention relative user priority *within* these groups.
- Can Condor give us higher-level tools to express this stuff?

Job Factories

- I keep finding myself in need of a simple job factory.
 - “Watch this LHCb science job queue and submit one LHCb VM job for each science job, up to a limit of 688 VMs. Remove VM jobs as the science-job queue shrinks.”
- GlideinWMS does all this and more, but is it lightweight enough for simple deployments like this? If so, I need the 5-minute quickstart guide!
- Should simple factories (for glide-ins, VMs, etc.) be a first-class Condor feature?

GPUs

- They're great! They suck!
- They probably approximate the future: fast, massively multi-core, unreliable.
- We do a lot of work to manage them in Condor
 - Querying and advertising h/w attributes.
 - Identifying and working around failures.
- Can Condor handle more of this management for us?

Checkpointing

- DMTCP is the future but...
- Standard universe still works seamlessly for the restricted cases where it works.
- DMTCP – checkpoints aside – doesn't work so seamlessly yet for real use.
 - No periodic checkpointing, checkpoint/resume, ckpt servers.
- We keep asking our users to test DMTCP, but...
- Very hard to get users excited about porting something that works well to something that doesn't.
- Bottom line: DMTCP needs to be closer to a drop-in replacement for standard universe before users will be willing to give it a shot.

Dynamic Slots

- We want them, we need them.
- Still a PITA for us:
 - Issues with fetchwork
 - Issues with GPUs
 - Porting static-slot policy expressions
 - Retraining users and rewriting submit file generation code

Followup

- Todd knows where I live.
- Email: <pfcouvar@syr.edu>
- Better yet, grab me in the hallway.
- I'm happy to talk to anyone here about ideas you might have for these issues – or about how LIGO uses Condor to get a tremendous amount of work done despite these issues.