Bring Your Own HPC Capacity

European HTCondor Workshop 2022

Todd Tannenbaum, Mátyás Selmeci, and Todd L Miller

Evolution History

- First Phase:
  - condor_annex (self-service cloud bursting tool)
- Second Phase:
  - split-starter/lumberjack (HPC systems w/o outbound networking)
  - XD-SUBMIT (HPC systems with outbound networking)
- Third Phase:
  - HPC Annex (self-service HPC system tool)

XD-SUBMIT

- Exploratory work: “what will it take to do this?"
- Hook up HPC centers to the Open Science Pool using the same infrastructure that we use for every other site
- Steps for OSG/PATh staff:
  - Obtain one user account from the HPC center for automated logins (SSH keypair, no 2FA)
  - Ask the user to add the new account to their allocation
  - Set up HTCondor-CE in front of the HPC site to accept pilot submissions and run them on the HPC site
  - Add GlideinWMS “factory” configuration so pilots with the correct parameters get submitted
  - Add GlideinWMS “frontend” configuration so the user's jobs can trigger pilot submission
  - Test!

Drawback: Poor User Isolation

- One account per site, all pilots use that account
- Users use separate allocations, but they all have the same file system permissions -- especially a problem with a shared file system
- Users can interfere with each other's files, or use up the account's disk quota

Considerations for Allocation-Based Submission

- Only one project's jobs should result in pilot submission -- and only when desired
  - OSG/PATh staff need to keep track of user allocations and modify configs as needed
- Only jobs from that project should run on that pilot
  - Pilot needs to know what project "asked for it"
- Conversely: pilot should always be able to run jobs from that project
  - Idle pilots still consume allocations (SUs)
- Pilots should be shaped to fit the user's workflow
  - Affects queue choice, requested resources, environment (e.g., loaded modules)

Drawback: Staff Effort Needed

- User had no control over the mechanisms for launching pilots
  - nor the number of pilots
  - nor the size of a pilot
  - nor the features
- User had no "panic button"
- User depended on OSG/PATh staff for changes
- Therefore OSG/PATh staff needed to monitor closely to avoid waste, and be quickly available to take care of user requests

Results and Lessons Learned

- ~1.5 million core-hours for Chemistry and Grav. Wave Astronomy jobs
- Learned how to run on several HPC sites
  - Each HPC site is unique -- queues, sbatch parameters, proxies, environment, filesystems, etc.
  - Tooling was written to adapt site environments to run our users' jobs (which has been reused elsewhere on the Open Science Pool)
- Experienced need for a "self-service" model
  - For proper isolation, users must be able to use their own identities, not just their allocations
  - Straightforward changes shouldn't require contacting staff

A Self-Service Tool

- Use what we learned about running pilots on these HPC systems, but start them as the researcher.
- Let's automate it and try to make it simple for the researcher.
- What about 2FA?
  - Write a command-line tool that gives us the opportunity to prompt the researcher to log in.

HPC Annex

- Operational
  - Glideins are user-specific (and totally unshared).
  - No staff involvement.
  - Direct user control over allocation use and pilot “shape”; user can shut pilots off remotely.
  - Pilots shut themselves off if idle.
- Architectural
  - Annex = A named set of EPs
  - State about the Annex is persistent at the AP
  - Jobs <-> Annex are bound to each other

A Demonstration

› I'll mostly be following one of the recipes.

https://htcondor.com/web-preview/preview-ospool-byor/ospool/byor/stampede2

Implementation (1)

- `htcondor annex create example debug@stampede2`
  1. Create IDTOKEN and PASSWORD file
     a. Used to authorize connections from the pilot to the collector
     b. Used to authorize connectiosn from the user for shutdown
  2. Check if a job targeting `example` exists.
  3. Start SSH and ask the user to finish. Invoke it with the `ControlMaster` option so that the connection persists.
  4. Make temporary directory via another SSH connection.

Implementation (2)

5. Populate the remote temporary directory via `scp`.
   - IDTOKEN, PASSWORD
   - scripts: pilot set-up, pilot, multi-pilot
   - `.sif` files, if any were detected in step 2
6. Submit a “state-tracking” local-universe job.
   5. Uses HTCondor’s cron-like scheduling to poll the collector
   6. Goes idle and stays that way once the capacity has joined the pool
   7. Records information about the capacity request at submit time and is updated with information about the SLURM job.
7. Invoke the pilot set-up script

Implementation (3)

- The pilot set-up script:
  1. Creates a pilot directory on shared storage and copies everything else there.
  2. Downloads an HTCondor tarball from the official site and a supplemental config tarball from elsewhere.
  3. Unpacks the binaries and runs the `make-personal-from-tarball.sh` from the tarball.
  4. Writes EP configuration. Unpacks the config tarball on top.
  5. Write and submit a SLURM job, which calls the multi-pilot script.

Implementation (4)

- The pilot set-up script has been sending the front-end debugging information; sending the SLURM job ID is the last piece before it terminates.
- The multi-pilot, when SLURM eventually starts it, reads the nodes file and forks SSH to start the pilot on each node in the job. It then waits for all of the pilots to finish before removing the pilot directory.
- The pilots copy the `local` directory so their logs don't collide and then exec `condor_master`.

Status

- Deployed at the OSG Connect access points. ☺
  - Stampede 2, Bridges 2, Expanse, and Anvil supported.
- Deployed. ☹
  - The deployment was not trivial, because of the need to create/configure another collector.
  - Work is in progress to make enabling the tool turning a single knob, ideally one that's on by default.
    - ^^^ Key to accomplish this: Direct connect of an EP to AP
- Future Work
  - Direct Connect
  - DAGMan

Thank you!

This work is supported by NSF under Cooperative Agreement OAC-2030508 as part of the PATh Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.