Lightweight sites: BOINC Backfilling

David Cameron (University of Oslo)
Shaojun Sun (University of Wisconsin-Madison)

ATLAS Sites Jamboree and HPC strategy, CERN, March 7, 2019
BOINC Status

- New Boinc monitoring appears reliably since 16 Dec, 2018

Average 8800 CPU days/day - equivalent of ~10k core grid cluster

Average 3.3M simulation events/day processed

Shaojun Sun (University of Wisconsin-Madison)

ATLAS Sites Jamboree and HPC strategy, March 7, 2019
BOINC Site Monitoring

2 weeks ago per-site monitoring was finally implemented

- Site admin creates BOINC account with same name as ATLAS site (e.g. AGLT2)
- BOINC wrapper sets job modification host to username@hostname
- MONIT monitoring maps these jobs to site_BOINC

**All running BOINC jobs per site**

**Completed jobs per queue for AGLT2**

*Links:* Select resource type “cloud_special” and group by “computingsite”

Shaojun Sun (University of Wisconsin-Madison)

ATLAS Sites Jamboree and HPC strategy, March 7, 2019
Harvester-BOINC Architecture for Trusted Resources

- **Motivations:**
  - Current architecture based on ARC-CE was designed around never giving credentials to untrusted resources
  - Allowing trusted resources to directly access grid storage addresses potential data management scaling issues
- Run a private BOINC server for trusted resources
- Harvester submits jobs (pilot wrapper & proxy) to BOINC server via Boinc Python API
- BOINC clients on WN pick up pilot wrappers which pull real payload from Panda server and interact with grid storage
Accomplished tasks

- Set up a PanDA queue called BOINC-BACKFILL.
- Set up a Harvester instance.
- Tested Boinc Remote submission API.
- Implemented Harvester-Boinc submitter, monitor and sweeper.
- Defined a few ES tasks for testing (e.g. 17248200, 17270723).
- Finished some ES jobs.
Harvester Instance Configuration

- Harvester supports various workflows: 1-to-1 and pull, 1-to-1 and push, 1-to-many and push, to name a few.
- We use NoJob & Pull workflow: Harvester submits jobs (pilot wrapper & proxy) without fetching/updating panda jobs; Pilot get jobs from PanDA server
- We are currently using multi-threading mode; can be switched to multi-processing mode with Apache+UWSGI
- We are using SQLite database; can be switched to other database backends, for example, MySQL.
- Thanks to Fa-Hui.

Shaojun Sun (University of Wisconsin-Madison)
Next steps

- Commission harvester-boinc plugins into production harvester
- Run some real tasks
  - Currently BOINC has manual task submission (short jobs and fixed release)
  - In this new mode any simulation task could run, but event service or short jobs are preferable for backfilling
- Currently BOINC_BACKFILL is attached to CERN DDM endpoints
  - Ideally a job could dynamically change its own configuration to read from any endpoint and write to local storage or nucleus (how to get input data, which is not local?)
  - i.e. BOINC wrapper reassigns an activated job to a different panda queue (possible in panda?)
  - Or, run only event service and upload to CERN OS
How to connect your site

- Twiki entry page: https://twiki.cern.ch/twiki/bin/view/AtlasComputing/VolunteerComputing
- Create a BOINC account with the same name as your ATLAS site (as defined in AGIS)
- Required software on worker node:
  - CVMFS
  - Singularity (if not SLC6)
  - BOINC client ← this should be the only thing you don’t already have
- Either install BOINC from eg EPEL or use simple install script (no root required)
  - wget http://atlasathome.cern.ch/Atlas-test/scripts/install_atlasathome.sh
  - sh install_atlasathome.sh --ProDir /opt/BOINC --uid your_boinc Authenticator
  - ... and jobs start running
- For Backfill mode, once things are ready you will need to ask for an account on the private BOINC server
Experienced Issues

- **StageIn error**: BOINC_BACKFILL queue set to TEST
- **StageOut error**: failed to access Object Store
- **Pilot wrapper** could not get PanDA jobs due to not enough activated jobs.
  - added `skip_scouting` in task parameter for testing tasks
  - reduced the numbers of `maxWorkers` and `nQueueLimitWorker`