Managing the ATLAS Grid through Harvester

Fernando Barreiro Megino, Aleksandr Alekseev, Frank Berghaus, David Cameron, Andrej Filipcic, Ivan Glushkov, Fa-Hui Lin, Tadashi Maeno, Nicolo Magini
On behalf of the ATLAS collaboration

CHEP 2019, Adelaide, Australia
Motivation

- Worker submitter capable of interfacing different resources (grid, cloud, HPC)
- Improve grid stability and usage efficiency
- Tighter integration with the Workload Management System (PanDA)
- Control ratio of pilot types (e.g. single vs multi core) according to ATLAS priorities/Global Shares
- Better monitoring
- This presentation focuses on the Grid usage, please see our other presentations
  - Harvester and Kubernetes
  - Harvester and HPC

https://github.com/HSF/harvester
Harvester global picture
Harvester: design decisions

● Flexibility to run in lightweight or high performance mode
  ○ Support for SQLite or MySQL/MariaDB
  ○ Possibility to run in multi-process mode through uWSGI framework

● Fast integration of new resources (CEs, batch, cloud API...)
  ○ Common core
  ○ Resource specific libraries (few hundred lines of code each). Typically:
    ■ Submitter: how to submit workers to the resource
    ■ Monitor: how to poll the status of the workers
    ■ Sweeper: how to clean up a worker
  ○ Plugins exist for HTCondor, ARC CE, GCE, K8s, SAGA, PBS, Slurm, Cobalt
  ○ Also different plugins for data movement needed in HPC environment
  ○ Possibility of running through RPC
Harvester workflows

- **Push**: the queued worker is already tied to a job. Early binding
- **Pull**: the worker will retrieve a job once it starts running. Late binding.

Typically several independent queues per core&memory combination at each site, leading to internal competition. Used to be most popular workflow.

- **Pull UPS** (Unified Pilot Streaming): Intended for **unified** queues, i.e. only one queue per site. The WMS\(^1\) tells Harvester the ratio of workers out of a set of core&memory categories:
  - The worker ratio is calculated according to ATLAS shares/priorities
  - The batch system respects corecount and memory specified in the job description file
  - Wherever possible, sites were migrated to this workflow. Currently most popular workflow.

\(^1\) *WMS: Workload Management System, in our case PanDA*
Harvester commissioning evolution

Mostly production migration: typically long, multi core workers

Mostly analysis migration: typically short, single core workers
Central Harvester infrastructure

Grid Harvester A

Grid Harvester B

ARC Harvester

CEs

CEs

CEs

CEs

CEs

CEs

CEs

Condor Schedds

(ARC CE, CREAM CE, HTCondor CE...)

ARC Harvester

CERN MySQL
DB on Demand

Servers
hosted on
CERN
OpenStack
Harvester worker monitoring: fast and flexible

Based on CERN IT provided Elastic Search, Kibana & Grafana instances

Filters
Worker information and links to logfiles/jdl

Tables and plots
Service&infrastructure monitoring

Service monitoring for Harvester and Condor instances: memory, disk, CPU, worker submission rates. Alerting system to report issues based on most relevant metrics

Grid Infrastructure monitoring: find broken or inactive sites/CEs. Future operational automation
Results and conclusions

Running job slots last 2 years. Grid only (HPC & cloud are excluded)
Conclusions

- 2 years from first discussions until project fully rolled out
- Better usage of the grid
  - More efficient and stable grid usage
  - Respecting ATLAS global shares, i.e. running the jobs that matter to the experiment
- Smarter worker submission, but also dependence on more components and their intercommunication
- Details to iron out
  - Grand unification of analysis and production queues
  - Handling single core spikes and higher submission rates
  - Automation/better handling of issues on the Grid