EPJ Web of Conferences **214**, 03025 (2019) https://doi.org/10.1051/epjconf/201921403025
CHEP 2018

ATLAS Global Shares implementation in PanDA

Fernando Barreiro Megino[1,*], *Alessandro* Di Girolamo[2], *Kaushik* De[1], *Tadashi* Maeno[3], *Rodney* Walker[4] on behalf of the ATLAS Collaboration

[1]University of Texas at Arlington, Texas, USA
[2]European Organization for Nuclear Research, Geneva, Switzerland
[3]Brookhaven National Laboratory, New York, USA
[4]Ludwig Maximilian University, Munich, Germany

Abstract. PanDA (Production and Distributed Analysis) is the workload management system for ATLAS across the Worldwide LHC Computing Grid. While analysis tasks are submitted to PanDA by over a thousand users following personal schedules (e.g. PhD or conference deadlines), production campaigns are scheduled by a central Physics Coordination group based on the organization's calendar. The Physics Coordination group needs to allocate the amount of Grid resources dedicated to each activity, in order to manage sharing of CPU resources among various parallel campaigns and to make sure that results can be achieved in time for important deadlines. While dynamic and static shares on batch systems have been around for a long time, we are trying to move away from local resource partitioning and manage shares at a global level in the PanDA system. The global solution is not straightforward, given different requirements of the activities (number of cores, memory, I/O and CPU intensity), the heterogeneity of Grid resources (site/HW capabilities, batch configuration and queue setup) and constraints on data locality. We have therefore started the Global Shares project that follows a requirements-driven multi-step execution plan, starting from definition of nestable shares, implementing share-aware job dispatch, aligning internal processes with global shares and finally implementing a pilot stream control for controlling the batch slots that keeps late binding. This contribution will explain the development work and architectural changes in PanDA to implement Global Shares, and describe how the Global Shares project has enabled the central control of resources and significantly reduced manual operations.

1 Motivation

ATLAS requires a vast computing infrastructure and a wide variety of workflows to study collisions and write physics papers (see Figure 1). These workflows have different requirements, e.g. Monte Carlo (MC) Generation and Simulation are CPU intensive, whilst Derivation jobs have higher I/O intensity. Most of the ATLAS Software runs on multi-core batch slots (typically 8 or 16), but there are some remaining workflows, most notably MC Event Generation, that can only run on single-core batch slots.

With the exception of few dedicated resources for real time processing (Trigger and Tier-0), all other workflows are scheduled across the Worldwide LHC Computing Grid (WLCG)[1] and other opportunistically used resources.

* Corresponding author: barreiro [at] uta.edu

Traditional Grid resources are dedicated compute centers in universities or other laboratories. They are ranked into Tier-1, Tier-2 and Tier-3 in an attempt to describe their capabilities, service levels and usage. There is a wide variation in processing, storage and network capacities.

In addition, there are opportunistically used resources, such as Volunteer Computing, Cloud Computing and High Performance Computing (HPC). Due to the nature/architecture of the resource and their integration model, they are only suitable for specific ATLAS workflows. The most favoured workflow for opportunistically used resources is MC Simulation, which has very low input and output requirements and keeps the CPUs busy for several hours. During peak availability periods, these resources can aggregate to around 50% of the ATLAS Computing resources, and are only used for MC Simulation.

However, ATLAS Physics Coordination needs to have control over amount of resources dedicated to each activity and establish a calendar for the planned delivery of physics results. The Global Shares project implements such control centrally in PanDA[2], the Workflow Management System used by ATLAS.

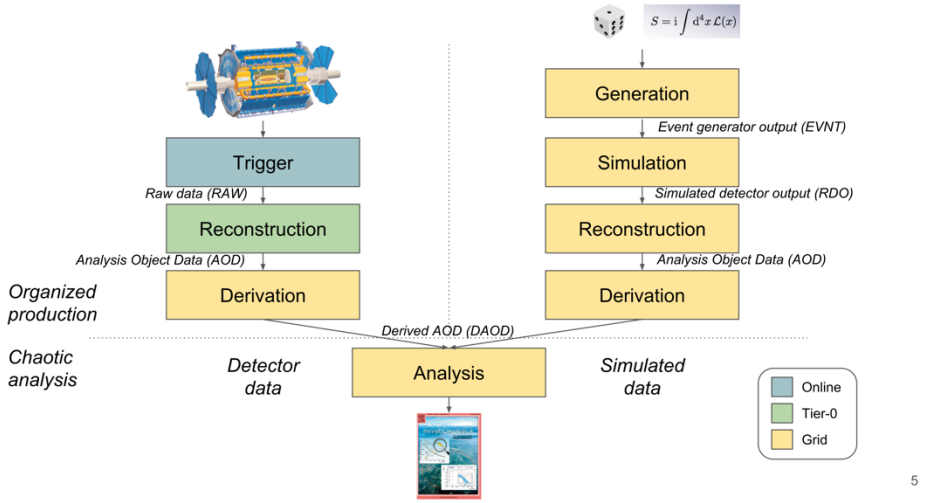

Fig. 1. From collisions to papers: overview of the processing paths for real and simulated data

2 Implementation

2.1 Global Shares definition

Global Shares establish the amount of resources available instantaneously to a certain activity as a fraction of the total amount of resources available to ATLAS. By requirement, Global Shares are measured in currently used HEPSpec06[3], the unit for corepower benchmarking used in HEP community. Note it was not implemented as a quota system measured over a certain period of time.

Global Shares are a nestable structure, where siblings have the preference to occupy unused shares, before the unused share goes to upper levels. A snapshot of the current share definitions, as of July 2018, is shown in Figure 2.

```
name: Analysis, value: 20.0
name: Express, value: 3.0
name: Production, value: 75.0
       name: Derivations, value: 14.8
              name: MC Derivations, value: 7.4
              name: Data Derivations, value: 7.4
       name: Event Index, value: 0.74
       name: HLT Reprocessing, value: 2.2
       name: MC evgen, value: 12.6
              name: MC 16 evgen, value: 10.1
              name: MC Other evgen, value: 2.5
       name: MC root, value: 4.5
              name: MC 16, value: 3.6
              name: MC Other, value: 0.9
       name: MC simul, value: 26.7
              name: MC 16 simul, value: 21.4
              name: MC Other simul, value: 5.3
       name: Overlay, value: 1.5
       name: Reprocessing, value: 7.4
              name: Reprocessing default, value: 5.9
              name: Heavy Ion, value: 1.5
       name: Upgrade, value: 2.2
       name: Validation, value: 2.2
name: Test, value: 2.0
```

Fig. 2. Snapshot of current multi-level Global Share definitions

2.2 Tagging of tasks and jobs

Tasks (collections of jobs) and jobs (workload unit) are tagged with a Global Share at creation time. The tagging is based on a table that defines regular expressions matched against the most common task and job description fields in the Production System[4] (processing type, campaign, working group, transformation). The Global Share field on each job allows to calculate the amount of resources occupied by each share.

2.3 PanDA job generation chain

The PanDA workflow management system has several components for the generation, brokerage and execution of jobs. The most important ones are shown in Figure 3 and described below.

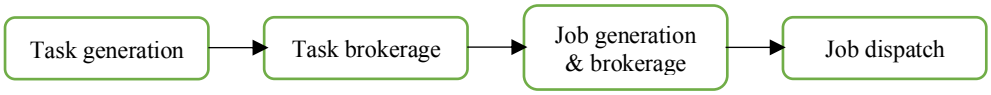

Fig. 3. Main steps in the ATLAS PanDA job generation and execution chain

- **Task generation**: Tasks are received from the Production System interface and inserted into the PanDA database
- **Task brokerage**: Tasks are assigned to its nucleus [5]. The nucleus is a computing site that participates in the processing of the task and where all the output files of the task will be collected. Nuclei candidates are larger sites that have proven reliable.

- **Job generation and brokerage**: A task is split into jobs. Jobs are assigned to be executed at a particular Grid site, based on input data availability, length of the queue and other metrics.
- **Job dispatch**: When a slot at a Grid site is freed up and requests a job, PanDA will decide which of the assigned jobs should run.

There were several changes needed on these components to make them Global Shares oriented.

Job dispatch, the module deciding which job will run next at each site, has been extended to respect Global Share targets. Job dispatch will order the jobs assigned to a site by Global Share preference (the share furthest away from its target) and priority inside the Global Share. This assumes a healthy distribution of jobs of different shares across sites, i.e. avoid all jobs of a particular share assigned to few sites. It also assumes that there are enough jobs of each share available for dispatch, so one share can not block another through the job generation chain.

In order to satisfy the last requirements, we have re-engineered the work-queues in PanDA. All the steps in PanDA's job generation and execution chain run through parallel work-queues. The purpose of the work-queues is to separate activities and to avoid one activity blocking the other. Work-queues also need to separate different slot types: single and multi-core jobs should not block each other, since generally they go to different types of resources. A low priority single core task should not be stuck behind a high priority multi core task, because there might be free single core resources available. Work-queues can also be tuned independently (e.g. upper and lower limits for queued or running jobs).

In the past, work-queues were defined ad-hoc to solve immediate operational problems: new work-queues were manually generated to cure starving activities. There was no methodology in the addition of work-queues so they grew inconsistently and partially overlapped each other.

The re-engineering of work-queues consisted in generating work-queues automatically (see Figure 4): there is a work-queue for each $global\ share\ \times slot\ type$ (currently single core, multi core and high memory). Each work-queue targets to have $n_{activated} \sim 2 \times n_{running}$ jobs in order to guarantee there will be enough jobs for dispatch of each share. This is an empirical target that has proven to work well overtime, since it allows rapid ramp up possibilities without generating many stale jobs.

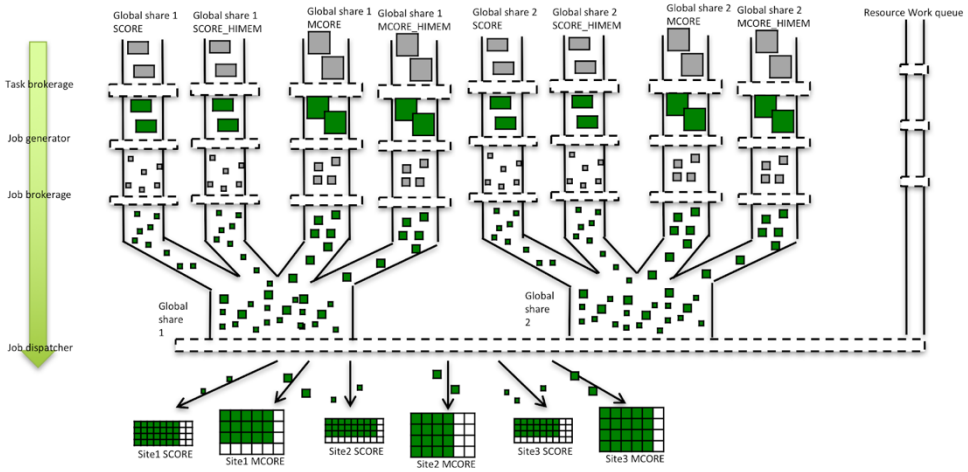

Fig. 4. Automatic work-queue generation by Global Shares in the ATLAS PanDA workload management system

Additionally, it is possible to manually define special work-queues for special resources, e.g. a HPC center. Given the scheduling policies on HPC centers, they generally need to have a large amount of jobs queued to fill a large reservation of the supercomputer.

2.4 Unified PanDA queues

Previous computing models established multiple queues per site, usually a single-core and a multi-core queue. For a few sites there are also separate queues for high and low memory requirements. The batch configuration/implementation behind these queues varies on each site. Some have static limits, but most manage them dynamically.

Some of the Global Shares activities have specific slot requirements. The most notable example is that MC Event Generation runs on single-core, while the other activities usually run on multi-core slots.

Global-Share-unaware pilot schedulers serving separate queues on a same site, usually submit pilots proportionally to the amount of assigned jobs at each queue. They do not respect the ratio needed to honour Global Shares and generate competition between the queues.
In order to control the ratio between single and multi-core slots, we are rolling out the concept of unified queues. Here all the jobs with different slot types are queued together and PanDA needs to control the order of execution. There are two different modes of control:

- Push mode: works naturally on Harvester [6] or ARC Control Tower [7]. This mode loses late binding.

    a) The pilot-submitter asks PanDA for the next job
    b) PanDA dispatches a job according to Global Shares
    c) The pilot, with the exact (i.e. job specific) requirements and job ID, is submitted to the batch system

- Pilot Streaming mode: this is a variation of the pull mode. It is less natural, but keeps the late binding. It is supported by PanDA and Harvester.

    a) PanDA calculates how many pilots of each type are needed, based on the jobs queued and running at the site, and the Global Share targets. The types are generic (i.e. not job specific) categories: single-core, multi-core and high-memory
    b) PanDA sends this information to Harvester
    c) Harvester submits the pilots with the proper requirements to the site
    d) When a pilot starts, it will ask for a job matching the type requirements
    e) PanDA dispatches the job according again to Global Shares

    In this mode PanDA decides how many pilots of each type are submitted, and the decision is not randomly left to the pilot submitter. Figure 5 shows an example at CERN, where PanDA asked Harvester to submit only single-core pilots around June 17, because it was boosting a MC Event Generation (single-core workload) campaign.

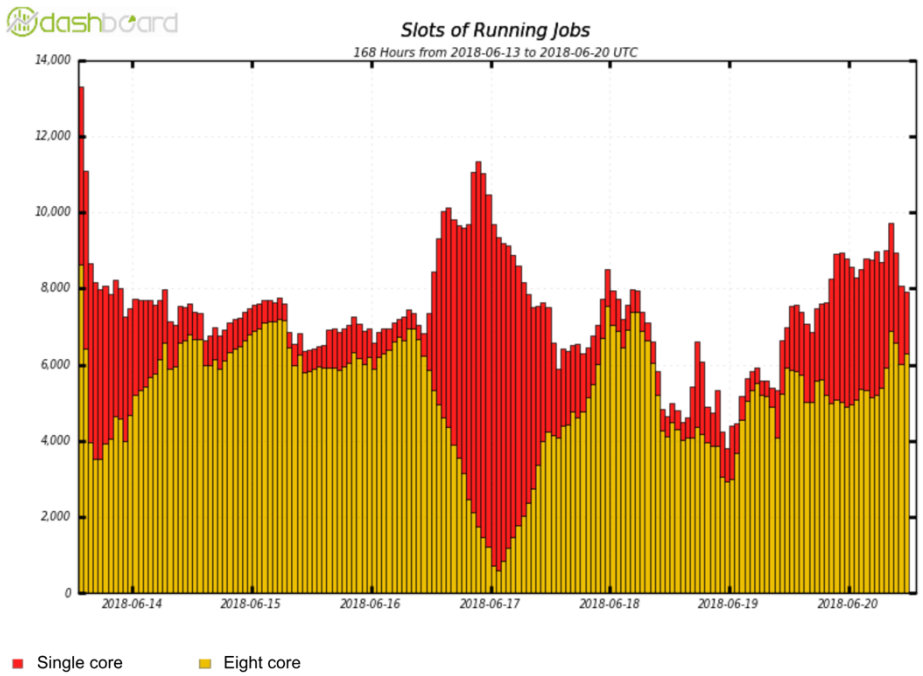

Fig. 5. Number of running cores at the unified queue at CERN, broken down by ATLAS single-core (1) and multi-core (8) jobs. On June 17 PanDA and Harvester changed dynamically the type of the pilots to single-core, in order to boost an Event Generation (single-core/1 workload) campaign.

3 Conclusions and future work

The Global Shares project has enabled the central control of resources by transforming several components of the PanDA workload management system. Manual operations have been significantly reduced and it is much easier to boost a particular share. The operations team, Computing management and Physics Coordination closely watch that shares are being respected. The feedback has been very positive and Global Shares is helping to keep the campaign deadlines.

Current work is focused on extending Unified PanDA queues to the majority of the sites. This step involves a review of site configurations and communication with sites having static partitions.

So far, the implementation has been centered on Central Production tasks and needs to be extended to include User Analysis tasks. Production and Analysis have been historically separated in different PanDA components, because of different security policies and Computing Model requirements. Adding Analysis to the Global Shares depends on having a majority of unified queues and requires some development.

References

1. LHC Computing Grid: Technical Design Report, document LCG-TDR-001, CERN-LHCC-2005-024 (The LCG TDR Editorial Board) (2005)
2. T. Maeno et al, J. Phys. Conf. Ser. **898** 052002 (2017)
3. M. Michelotto et al, J. Phys. Conf. Ser. **219** 052009 (2010)

4. M. Borodin et al, J. Phys. Conf. Ser. **898** 052016 (2017)
5. F. Barreiro Megino et al, J. Phys. Conf. Ser. **898** 052011 (2017)
6. T. Maeno et al. Harvester: an edge service harvesting heterogeneous resources for ATLAS, Proceedings of this conference (CHEP2018)
7. A. Filipcic et al, J. Phys. Conf. Ser. **331** 072013 (2011)