arcControlTower: the System for Atlas Production and Analysis on ARC

Andrej Filipčič for the ATLAS collaboration
Jozef Stefan Institute, Ljubljana, Slovenia
E-mail: andrej.filipcic@ijs.si

Abstract.
PanDA, the Atlas management and distribution system for production and analysis jobs on EGEE and OSG clusters, is based on pilot jobs to increase the throughput and stability of the job execution on grid. The ARC middleware uses a specific approach which tightly connects the job requirements with cluster capabilities like resource usage, software availability and caching of input files. The pilot concept renders the ARC features useless. The arcControlTower is the job submission system which merges the pilot benefits and ARC advantages. It takes the pilot payload from the panda server and submits the jobs to the Nordugrid ARC clusters as regular jobs, with all the job resources known in advance. All the pilot communication with the PanDA server is done by the arcControlTower, so it plays the role of a pilot factory and the pilot itself. There are several advantages to this approach: no grid middleware is needed on the worker nodes, the fair-share between the production and user jobs is tuned with the arcControlTower load parameters, the jobs can be controlled by ARC client tools. The system could be extended to other submission systems using central distribution.

1. Introduction
The central production and analysis system for ATLAS [1] is a complex job and file distribution system using three different grid flavors, namely Open Science Grid, gLite and ARC. In order to overcome the difficulties with the specifics of the various grid approaches as well as site and data storage organization, ATLAS has developed a general database driven job submission system PanDA [2] which is grid flavor independent and is capable of distributing close to one million jobs per day to the computing resources.

Past instabilities of the grid middleware and site resources led the job concept to the pilot paradigm, where a generic pilot job picks the payload from the central server at the job startup. The advantages of such a system are clear when the job efficiency and throughput are compared to the previous direct submission system. However, such a system completely ignores the job resources and site capabilities. The ARC middleware [3] provides several unique capabilities like the embedded data caching system and site resources brokering which are completely useless within the pilot system.

In order to use the ATLAS pilot system together with the unique ARC features, the arcControlTower job submission and control system has been developed. The arcControlTower plays the role of the pilot factory and the pilot job itself. The preprocessing, postprocessing and server communication parts of the pilot job are performed within the arcControlTower, while the job running on a grid node runs in a completely local, i.e. non-grid mode.
2. Nordugrid and ATLAS ND cloud

Figure 1. The ARC architecture

The ND cloud is one of the ATLAS distributed computing clouds with resources spanning over the five countries, Denmark, Norway, Slovenia, Sweden and Switzerland. As opposed to other clouds, it is not organized as Tier-2 centers attached to a central Tier-1. The computing resources are distributed among 14 clusters (figure 2) playing even roles in the production system. The storage is organized around the central NDGF-T1 providing the central dCache services, while the storage pools and tape libraries are distributed through the sites. There is no need for a special local storage element attached to a particular site, therefore a small Tier-3 cluster with ARC services can be included in the production system transparently.

The ARC architecture shown in figure 1 differs from other middlewares in the cluster setup. The middleware is in general only installed on the ARC frontend, which performs all the grid related operations like transferring the input and output files, translating the grid job to the local batch job, controlling the status of the jobs and providing the interfaces for the job control. The input files are kept cached on the ARC frontend, so the subsequent jobs can reuse the input files without additional transfers. The nodes do not require any grid middleware since the jobs usually run in a local batch mode with input files available on the frontend through a shared filesystem and output files transferred to the session directory on the frontend at the end of the job.

3. arcControlTower architecture

The ARC architecture, most prominently the non-grid node concept, is not compatible with the pilot system. In principle, the ARC middleware can be used on nodes as well so the features requested by the true pilot jobs are provided, but most of the sites prefer the non-grid cluster setup due to a clear separation of the local batch cluster and the grid being provided by a single add-on server.

To enable the pilots to run on the non-grid nodes, the operations performed by the pilot job have to be separated in the grid-related and the local operations. The arcControlTower has been developed to provide this functionality. A simplified workflow is shown in figure 3.

The arcControlTower performs the following job operations:

• payload retrieval from the PanDA server
Figure 3. The arcControlTower workflow

- PanDA job status control through the pilot callbacks
- pilot job translation to ARC job
- ARC job submission and control

The arcControlTower is written in python and is organized as a database driven event-loop system which propagates the jobs through states. The state flow is not necessarily linear which allows for features like the job resubmission or rerunning of the ARC jobs in case of download or upload failures. Each state has a corresponding modular action which propagates the job to usually an upstream state. The arcControlTower controls several jobs in the same state at once.

The major components are
- aCTMain.py: initialization and process control
- aCTConfig.py: xml configuration interface
- aCTDB.py: database interface
- aCTAutopilot.py: interface to the PanDA server
- aCTSubmitter.py: ARC job submission module
- aCTDownloader.py: ARC job status and retrieval module

The PanDA interface provides the pilot-job functionality. It retrieves the job payload from the PanDA server as the regular pilot would do and executes the corresponding callbacks to set the job status in PanDA. The payload is parsed for the input and output files, the required job resources like walltime, memory and requested software. Then the ARC job in xrl description language is constructed and is automatically brokered and submitted to one of the clusters within the ND cloud. The payload is embedded in the ARC job so the regular ATLAS pilot wrapper can be used for the job execution. The pilot wrapper gets the payload from the job description instead from the PanDA server and in this mode all the grid operations are skipped. The ARC job execution is then monitored and regular heartbeats are sent to the PanDA server by the arcControlTower. When completed, the arcControlTower fetches the job status from the cluster and performs the postprocessing operations like LFC registration, PanDA server completion callback and job cleanup.

All the configuration variables are stored in a single xml file, which is reparsed if modified at each loop, so the parameters can be changed on-the-fly. The number of active jobs is automatically determined by the ratio between the number of all and running jobs to maximize the usage of allocated ATLAS slots on the Nordugrid clusters. Cluster downtime is handled automatically through the ATLAS ARC Grid Information Index Service and through the PanDA monitor.
4. Benefits
The ND cloud with arcControlTower has some unique features. First of all, it allows for the pilot concept to be used with the predictive direct job submission and thus provides the ARC to be used within the common, unified ATLAS distributed computing system. Since the jobs are defined before submission, a full resource requirements can be provided at the submission time. Special jobs with longer walltime or larger memory can be submitted without any manual intervention or pilot factory tuning. If some cluster does not provide the requested job resources or if the software is missing, it will be automatically rejected by the broker and the job would be submitted elsewhere. If none of the clusters provides the software, the job will wait in the arcControlTower until the software is installed on at least one cluster.

All the clusters use a single storage endpoint, NDGF-T1, thus all the input or output file transfers are remote. The ARC caching reduces the input file transfers to a minimum, which is especially important for analysis jobs with large input files. The ACIX service provides the lookup for input file availability on the cluster cache. This is used in the job brokering algorithm, where the cluster with more input files cached is preferred for job submission.

The arcControlTower proved to be a stable system for ATLAS distributed computing, with high job efficiency and low failure rate. Special tasks which cannot be brokered by the PanDA server in the usual way due to special requirements are executed in the ND cloud.

5. Prospects
The arcControlTower is in operation for almost two years now. A rewrite of the code is underway for faster throughput and more generic usage. The current sqlite3 database will be replaced with a generic database interface. The codebase will be separated into two major parts, one controlling the PanDA server and jobs, and the other with a generic ARC job control. The latter could be used as a generic job submission system.

References
[1] ATLAS Collaboration 2008 JINST 3 S08005
[2] T. Maeno: Overview of ATLAS PanDA Workload Management, this proceedings, CHEP 2010
[3] The NorduGrid Collaboration 2006 Advanced Resource Connector middleware for lightweight computational
Grids Future Generation Computer Systems 23 219-40