Hiding the Complexity: Building a Distributed ATLAS Tier-2 with a Single Resource Interface using ARC Middleware

S Purdie[1], G Stewart[1], M Kenyon[1,4], S Skipsey[1], A Washbrook[2], W Bhimji[2] and A Filipčič[3]

[1] Particle Physics Experiment (PPE) Group, Kelvin Building, University of Glasgow, University Avenue, Glasgow G12 8QQ, UK
[2] Department of Physics, University of Edinburgh, Edinburgh, EH9 3JZ, UK
[3] Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia
[4] Now at: IT Department, CERN, Route de Meyrin 1121, Geneva 23, Switzerland

E-mail: s.purdie@physics.gla.ac.uk

Abstract. Since their inception, Grids for high energy physics have found management of data to be the most challenging aspect of operations. This problem has generally been tackled by the experiment's data management framework controlling in fine detail the distribution of data around the grid and the careful brokering of jobs to sites with co-located data. This approach, however, presents experiments with a difficult and complex system to manage as well as introducing a rigidity into the framework which is very far from the original conception of the grid.

In this paper we describe how the ScotGrid distributed Tier-2, which has sites in Glasgow, Edinburgh and Durham, was presented to ATLAS as a single, unified resource using the ARC middleware stack. In this model the ScotGrid 'data store' is hosted at Glasgow and presented as a single ATLAS storage resource. As jobs are taken from the ATLAS PanDA framework, they are dispatched to the computing cluster with the fastest response time. An ARC compute element at each site then asynchronously stages the data from the data store into a local cache hosted at each site. The job is then launched in the batch system and accesses data locally.

We discuss the merits of this system compared to other operational models and consider, from the point of view of the resource providers (sites), and from the resource consumers (experiments); and consider issues involved in transitions to this model.

1. Introduction

There are two key use cases of the Grid within the Particle Physics communities: simulation of detectors and analysis of recorded data. Of the two, simulation is more straightforward, as it involves a significantly smaller quantity of data, and is not the primary focus of this work. Analysis is the more tricky of the two, given that it depends on being able to access large quantities of data. The LHC experiments initially planned on the following the MONARC [1] model; with hierarchical distribution models. Within this model, there is a primary site for each country, to which data is sent initially, and from there the data is distributed to Tier-2 sites for analysis. The details of this distribution process are left up to each VO, and this is the point where there is a trade off to be made. There are many existing clusters that could be connected to the Grid, where some share of the cluster could be used. The more sites that there are connected, the more complicated it is to distribute data to them, and ensure that things are 'well distributed' [1]. There is also the need to have storage at each site, which is non-trivial to maintain.

Here we describe how the ScotGrid distributed Tier-2, which has sites in Glasgow, Edinburgh and Durham, was presented to ATLAS[2] as a single, unified resource using the ARC[3] middleware stack. In this model the ScotGrid 'data store' is hosted at Glasgow and presented as a single ATLAS storage resource. As jobs are taken from the ATLAS PanDA framework, they are dispatched to the computing cluster with the fastest response time. An ARC compute element at each site then asynchronously stages the data from the data store into a local cache hosted at each site. The job is then launched in the batch system and accesses data locally.

2. ARC

ARC middleware [2] does a number of things differently from gLite middleware, but the key aspect for this work is the manner in which jobs data requirements are handled, and the local cache. When a job is received by the ARC CE, the CE stages in all the required data files before sending the job to the batch systems, rather that staging in files at that start of the job. This change requires some disc space allocated at the CE to use for the staged files, but does mean that the time taken to stage in files is not felt by the job - which means that the typical CPU efficiency of the job tends to be better. This disk space is used as a cache, so multiple requests for the same file are made, it can be served from the local cache if it's present.

ARC is proven technology, being used primarily by the Nordic Data Grid Federation to support the LHC experiments in addition a number of other activities; as well as several other NGI's.

Although primarily used via it's own job submission and management tools, there exists plugins for the the gLite WMS to allow it to submit to ARC; further the syntax and semantics of ARC's native tools are similar to Globus GRAM (and hence lcg-CE), so it's not too difficult to adapt tool chains to submit to ARC. More tricky is to adjust the job profile to gain maximum benefit from the ARC caching mechanisms. Directly submitted jobs with explicit data requirements are straightforward here, whilst pilot job frameworks require more work to adapt. For the ATLAS experiment this work had already been done, resulting in a piece of software called the ARC Control Tower, which effectively translates the ATLAS pilot jobs into ones with explicit data requirements. There are discussions in progress amongst the ARC developers to offer pre-caching features that would make supporting pilot frameworks easier, so this is expected to involve less work in future.

3. Installation of ARC

To be a competitive model it is important that the upfront costs of change are not prohibitive. ARC is free software, but the most important resource is site and VO administrators time. In this case that relates to three primary things: installation time taken; maintenance overhead and co-existence with current CE's.

The installation in terms of deploying the required packages was near trivial, consisting of adding the appropriate YUM [4] repositories, and invoking YUM once [5]. An area of disk space is

[1] The precise meaning of well distributed is up to the VO in question.

[2] A Toroidal LHC ApparatuS

[3] Advanced Resource Connector

[4] Yellowdog Updater, Modified: the system under Scientific Linux (inherited from its Red Hat lineage) that maintains and updates software. ARC also offers Debian repositories, which were not used here.

[5] In practice, it was needed to do it twice at Glasgow, as SP missed one of the required repositories first time (the Red Hat EPEL repository).

needed for the cache, and here we used Lustre [3]. The final step in installation is configuration, which is contained in a single file, arc.conf. Although there are three components in a running CE, the same file is used for all parts, which does simplify things.

3.1. Configuration

Configuration was generally straightforward, working from a model configuration in use at another site. One key complication was that the mechanism traditionally used for pool account mapping within gLite and ARC sites is different; with gLite sites generally using LCMAPS, whilst ARC sites typically use a shell script to manage pools. ARC does have support in for using LCMAPS, but at the time this works was being done there wasn't a production 64bit LCMAPS library available, and the early versions tried didn't load properly on our test system. At this point, it was decided to separate the two issues, and leave the dynamic mapping to the side; using a small, static, gridmap in place to allow the rest of the planned work to continue. This was configured with a few accounts for development purposes, and some ATLAS production users for the testing, all matched to the dynamic pool accounts by hand.

One piece of configuration that was missed initially is that because the CE stages in data for the the jobs, it needs to have the network stack tuned like a disk pool node; until this was done the throughput was noticeably slower than might be expected.

Outside of the time spend on account mapping issues (which shouldn't be a problem for sites in general, once production 64bit LCMAPS libraries are available), it took around a day of work, spread over a week, in order to configure the ARC CE, until it would accept jobs and pass them through the batch system. That's roughly comparable with the time budget used for setting up a secondary CREAM CE at Glasgow. It is expected that the authors could guide other sites through the process, such that it would take less time for them.

3.2. Testing

Initial tests were carried out with the ARC command line tools, including the very useful *ngtest* program. *ngtest* uses pre-canned test jobs, meaning that a site admin does not need to learn the ARC syntax or command line tools in order to test things. One of the tests used files that need to be staged in from multiple sources, to ensure the input staging works.

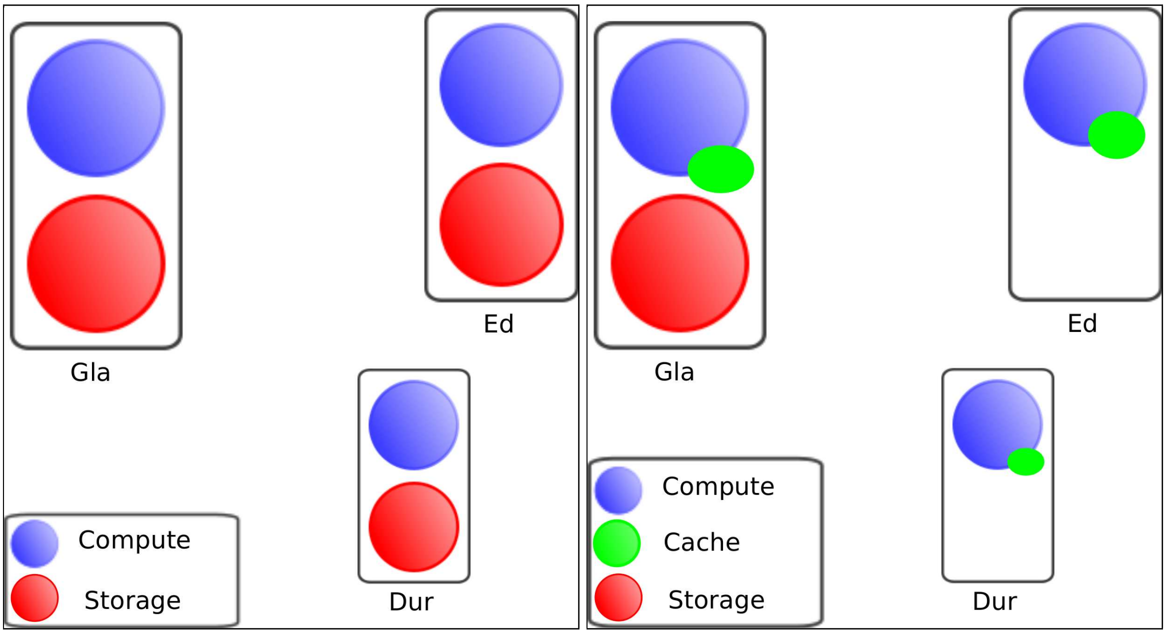
Once the basic operation was assured, a test of the cache was performed by joining the ARC CE in Glasgow to the NDGF cloud for ATLAS. This let the ATLAS PanDA framework send jobs to Glasgow, where they staged in required files from the NDGF datastores in Scandinavia. This is a more extreme separation of compute node and data store then planned for the Scotgrid layout, and therefore a more taxing test. It was also easier to set up this arrangement as a testbed with real jobs (involving less work from the VO).

4. Scotgrid 2.0

The model planned for Scotgrid is actually the old model, as originally envisaged in the heady days when the Grid was young and we were naïve. Having a single, large, data store at one site (Glasgow), with compute facilities are the other sites, and staging the data from Glasgow to the other sites as needed. This plan was abandoned earlier when it became apparent that the staging of data from the data store direct to the worker node as needed resulted in the jobs spending too long waiting on the transfer, such that the job efficiencies were low.

By using the ARC CE to stage the data in before the the job arrives at the worker node, the jobs can hit very high efficiencies, and still have a remote data store. This does require some storage at each site, but this is managed by the CE as a cache, and is not critical to correctness of operation [6]. Under this model all the storage could be consolidated at one location, from the

[6] It is, however, important for efficiency of operation



Current storage model

Proposed storage model

Figure 1. Current and proposed storage layout

point of view of management of long term storage. This simplifies the use of the sites by VO's, and also reduces the minimum necessary infrastructure for a site to have, before it can be use effectively.

Figure 1 shows visually the change enabled by this model. Although there are slightly more entities required in the proposed model, the maintenance of the cache is much simpler, resulting in a lower workload for site admins. The VO admins definitely see a simpler storage model, as the use of the caches is invisible to them.

5. Transition plans

There's no advantage in a model that requires a 'big bang' change. It is politically and practically impossible to get all users to agree to change how they do things at precisely the same point in time. Therefore for the model to be viable, it is necessary to be able to run it in parallel with the current models. For Scotgrid, this means that ARC must fit in with the way gLite does things, so that there is ability to run both in parallel.

The user mapping to pool accounts via LCMAPS has already been mentioned. Although strictly having the exact same mappings is not necessary, it is useful, and a good way to ensure that the requirement of mapping VOs to groups (where appropriate) is maintained.

The existing Storage element can be left as is; provided that some disk space can be obtained for use in the cache. It is expected that, where a transition is envisaged, storage nodes may be moved piecemeal from the Storage Element to be used as the cache; such an arrangement would require the use of a filesystem such as Lustre [3] or Ceph [4] to facilitate that.

The most crucial part in the transition is the change in the user interface. In this case, it is twofold - partly that ARC uses a different syntax to describe jobs, and partly that there is a different model in place for how jobs operate. This is foreseen to be the slowest part of the transition, as the keener users may move quickly, but there is likely to be a few users that do not wish to change scripts that the currently use, for no benefit to them. Both of these changes, however, are reasonably small in overall impact. For the different syntax, the principles embodied by the tools are similar in both cases. The biggest difference in the job submission model is that in ARC one submits direct to the CE, rather than an intermediate, and, because data staging only happens at the start and end, it is not necessary (in general) to have a valid proxy during the job. These make the transition hold positive aspects for end users.

Primarily, however, these changes in user experience mean that such a transition should be considered a long term change. Whilst the big (LHC) VO's may move quickly, the small VO's may need a lot more time to carry out the transition - making the co-existence of both models crucial.

6. Conclusion

ARC setup is straightforward, and can be done in a manner compatible with existing gLite CE's; which would be easy to automate with a matching bitness of LCMAPS. The evidence from the trial runs with Glasgow as a compute node backed from NDGF storage in Scandinavia showed that the single storage for multiple model would not be detrimental to the performance of the clusters. The one barrier is VO adoption of the ARC element; something that can be done today with some of the VO's.

Bibliography

[1] The MONARC Collaboration. Monarc - phase 2 report. May 2000. http://www.cern.ch/MONARC/docs/phase2report/Phase2Report.pdf.

[2] M Ellert, M Grønger, A Konstantinov, B Kónya, J Lindemann, I Livenson, J L Nielsen, M Niinimäki, O Smirnova, and A Wäänänen. Advanced resource connector middleware for lightweight computational grids. *Future Generation Computer Systems*, 23, 2007.

[3] Lustre filesystem. Website, 2011. http://www.lustre.org/.

[4] Ceph filesystem. Website, 2011. http://ceph.newdream.net/.