

The ATLAS ARC backend to HPC

S. Haug¹, M. Hostettler¹, F. G. Sciacca¹, M. Weber¹ on behalf of the ATLAS collaboration

Albert Einstein Center, Universität Bern, Switzerland

E-mail: gianfranco.sciacca@lhep.unibe.ch

Abstract. The current distributed computing resources used for simulating and processing collision data collected by ATLAS and the other LHC experiments are largely based on dedicated x86 Linux clusters. Access to resources, job control and software provisioning mechanisms are quite different from the common concept of self-contained HPC applications run by particular users on specific HPC systems. We report on the development and the usage in ATLAS of a SSH backend to the Advanced Resource Connector (ARC) middleware to enable HPC compliant access and on the corresponding software provisioning mechanisms.

1. Introduction

The *Worldwide LHC Computing Grid* (WLCG) [1] has been specifically set up to meet the needs of the ATLAS [2] and other CERN *Large Hadron Collider* (LHC) experiments' software stacks. It is largely made of dedicated x86 Linux interconnected clusters executing workloads submitted by a large number of users through complex global frameworks. While the WLCG has been successful in providing an adequate environment for achieving the physics goals of the experiments during the first LHC run (Run 1), with over 170 computing sites worldwide capacity is stretched already now and the model is expected not to scale beyond 2020, within the foreseen budget constraints.

With the new physics Run 2 about to start at CERN, computing for the LHC faces serious challenges in view of the expected increase in CPU needs over the next few years [3]. Significant increases in the trigger output rates and detector occupancy at higher energies and luminosities imply larger amounts of data to be processed. Correspondingly, an increase in the amount of CPU resources is required. The area of simulation is also important, since significant CPU resources are devoted to this activity. A general CPU volume increase close to a factor of two in 2015 compared with 2012 is anticipated. A further increase of 20% per year until 2018 is also predicted. Beyond that, with the expected detector upgrades, the current model is no longer expected to scale at manageable costs. A combination of innovation, new technical solutions and new strategies in securing resources is therefore needed.

High Performance Computing (HPC) centres worldwide provide general purpose high-grade (non distributed) systems and are used for a wide range of computationally intensive tasks in various fields, including climate research, weather forecasting, molecular modelling, and quantum mechanics. The use of such systems is typically regulated by strict rules, with single users granted access in order to run self-contained applications that are specifically developed and built for the system's architecture. Often no super-user access is contemplated, no modifications of the system allowed, such as establishing persistent services on the system, and outbound network access is generally denied.



Therefore in ATLAS the use of multi-purpose high-end HPC systems is proposed. In this paper we describe a solution developed at the University of Bern that enables execution of CPU intensive ATLAS simulation workloads on HPC systems, addressing specifically the following challenges:

- compliance with tight access rules;
- application porting and provisioning;
- workload management;
- input data provisioning and output data retrieval.

In section 2 we discuss the application porting and provisioning solution. The ATLAS workload management system is briefly introduced in section 3, while in section 4 we describe the HPC policy compliant solution developed. We then present the current status of integration in the ATLAS distributed computing infrastructure and look at possible further developments.

2. ATLAS application porting and provisioning to HPC systems

Traditionally, a variety of RISC processor families were featured in the majority of TOP500 [4] supercomputers worldwide up to the early 2000s, including Alpha, MIPS, PA-RISC, and SPARC. From the beginning of the new century, the 32-bit x86 and later 64-bit x86_64 have started growing considerably, changing dramatically the face of supercomputing: as of November 2014, up to 70% of the TOP500 supercomputers' capacity is based on x86_64 CPUs (Intel EMT64 and AMD AMD64 instruction set) (Fig. 1).

The LHC experiments software stacks have been built for the x86 architecture from the onset of the development. A major move towards the x86_64 architecture occurred over the last few years. As of today practically all ATLAS applications are built for x86_64. This means that we are potentially able to run unmodified ATLAS pre-built binaries on a large percentage of the worldwide HPC resources without any explicit need for porting.

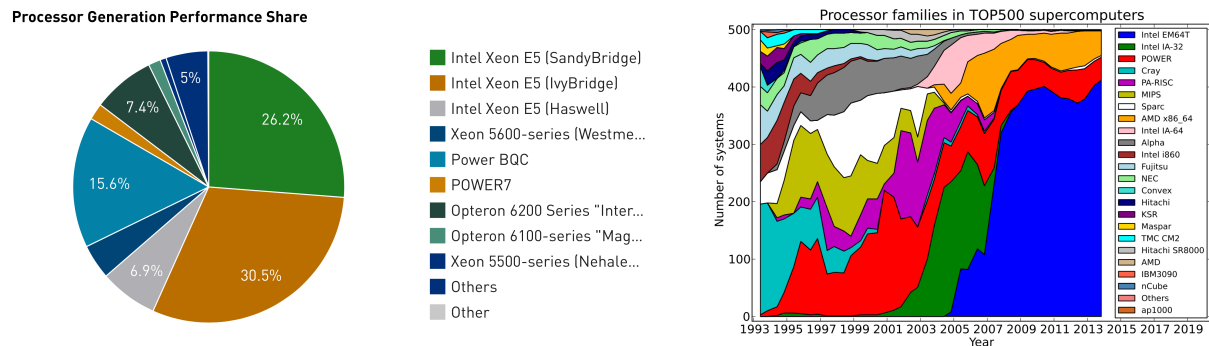


Figure 1. The share of different CPU architectures for the TOP500 systems worldwide [5]

Based on the considerations made earlier about the general features and policies of HPC systems, we have focussed our attention on the workloads that are most CPU intensive and require moderate amounts of I/O network traffic. Monte Carlo simulations suit these criteria very well, consisting of the following steps: event generation (*SHERPA*, *Pythia*, *Alpgen*) [6] and detector simulation (*Geant4*) [7]. In 2015 it is expected that Monte Carlo simulation will account for over 50% of the total CPU usage for ATLAS.

The ATLAS software framework *Athena* is an implementation of *Gaudi* [8], a component-based architecture, which was designed for a wide range of physics data-processing applications. Athena consists of a set of individual tools for the different workloads, which are integrated into the general

framework. The software is available in pre-compiled releases for x86-based Linux systems from the *CernVM File System (CVMFS)* [9]. The software is delivered centrally from CERN using a scalable hierarchy of standard HTTP caches. On the computational nodes, a specially crafted file system is used that ensures data integrity and provides fault-tolerance. On WLCG clusters, this is mounted locally via a FUSE module that must be installed as root. On HPC clusters, no persistent services can be generally setup and run on the nodes and system-wide customisations are generally not permitted. The initial approach was to mount the CVMFS file system as a normal user via the *Parrot* [10] virtual file system wrapper. This is a tool for attaching existing programs to remote I/O systems through the file system interface. Although the method was effective, it proved itself unreliable in a multi-threaded environment, whereby multiple processes trying to access the same file would lead to a deadlock.

It was instead chosen to copy the relevant portions of the ATLAS software needed to a shared area available to the computational nodes on the remote HPC system and keep them synchronised via rsync. Each release of the ATLAS software stack is of a manageable 500 GB size, however, it occupies about 500000 inodes on the file system, so care must be taken not to come close to exhausting the inode limit on the remote HPC file system.

3. Workload Management

The *Production AND Distributed Analysis* framework (PanDA) [11] is the ATLAS approach to a data-driven workload manager. It has been designed and developed specifically by ATLAS in order to meet the challenging requirements on throughput, scalability, robustness, minimal operations manpower, and efficiently integrated data/processing management. A generic workload usually describes inputs, outputs and the Athena action to apply. All the workloads created centrally (such as in the case of large Monte Carlo campaigns) or by single users are stored in the global PanDA database, and a dedicated scheduler takes care of the distribution, management and monitoring of all the workloads for their full lifetime. About one million ATLAS PanDA jobs are run each day on all ATLAS computing sites. The average computing power consumed by those jobs totals to more than 150 000 cores (x86).

In Europe, the PanDA workloads run on top of the European grid computing middleware (*ARC* or *CREAM* Compute Elements). In the US, the Open Science Grid middleware is used (*HTCondor* or *Globus GRAM* Compute Elements).

From the computing site point of view, the middleware is the element of communication between computational resources and PanDA. The different middleware offer different approaches and implementations. Typically, a range of services run as root on the Computing Element (CE), implementing the gateway between the distributed computing infrastructure and the local resources. In some cases, middleware is also required on the computational nodes. The middleware requirements to interface the workload management are a challenge, since these services cannot generally be implemented on demand in a HPC centre, and specifically on computational nodes.

4. Access to Resources: Job Submission and Output Retrieval with ARC

4.1. The ARC Compute Element

At the University of Bern, the *Advanced Resource Connector* (ARC) [12] middleware is used. The ARC-CE receives workloads from clients, either individual users or the PanDA servers. In our discussion here we will consider the latter case, although the workflow for individual users is functionally identical. The ARC-CE handles user-authentication, job submission to the backing cluster and staging of input and output files. A schematic of the ARC-CE functionality can be seen in Fig. 2.

The A-REX service manages the life-cycle of each workload: it performs the user authentication and mapping to one from a pool of local users defined on the cluster. Then it maps the job specifications (for example memory and cpu/walltime requests) to instructions for the backing Local Resource Management System (LRMS). As soon as all input files are available, the job is submitted to the LRMS. The status of the job is queried regularly, and upon successful termination, the staging of

the output data is initiated. Both the input and output staging operations are performed by the GridFTP service.

All the job-related files are stored in a non-persistent “session directory” which must be made available to all nodes. A caching mechanism can also be enabled, so that input files already existing are not staged again unnecessarily. With this approach, all necessary services are provided by the ARC-CE. Neither additional middleware nor explicit outbound network connectivity are required for the computational nodes, thus complying with the typical HPC system requirements.

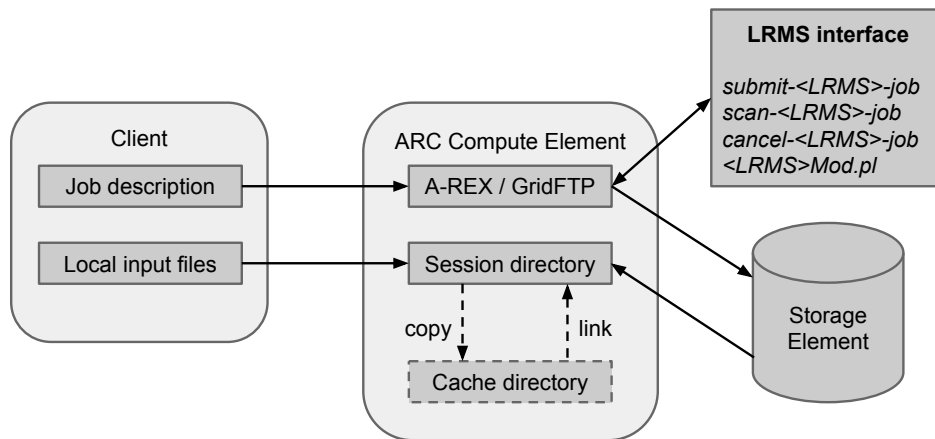


Figure 2. Schematic of the functionality of the ARC-Compute Element.

Additional services are run on the ARC-CE (omitted in Fig. 2 for simplicity). For example, an information system based on a LDAP server run in conjunction with a Berkeley Database Information Index (BDII) is critical for the operations. It provides detailed information about the grid services exposed by the instance and the status of the underlying computational resources, which is needed for various different tasks. The PanDA servers use this information to discover computational resources and match workloads based on their details. Another auxiliary service is run on the ARC-CE to register the resource to one or more Grid Index Information Service (GIIS), which exposes information for public monitoring.

4.2. The ARC interface to HPC

The ARC middleware described implies that a service on the ARC-CE submits to a LRMS fronting a computing cluster, while on HPC the access to the system for job submission and data management is often only provided to specific users over the Secure Shell (SSH) protocol to dedicated login nodes. The ARC-CE services may not be run on HPC dedicated login nodes, on which as we said, permanent services are generally not allowed.

We developed an extension to the ARC middleware to include an interface to submit and manage ATLAS PanDA workloads as jobs to a resource manager of a remote HPC machine. This makes it possible to run an ARC-CE frontend to a HPC system on a server or virtual machine which is completely independent from the HPC system, and which does not have to be in the same computing centre. Furthermore, by exploiting the shared session directory feature of ARC, no outbound network access is required from the compute nodes of the HPC system. In our approach it is required that both the access to the session directory and calls to the HPC scheduler are transparently handled over an SSH connection to the remote system. The integration concept is shown in Fig. 3, and consists of the following:

- All ARC-related services run on a dedicated machine at the University of Bern. No services are exposed from the HPC machine. The ARC session directory is provisioned on a suitable shared area on the HPC system.
- Incoming PanDA workloads are copied to the HPC system over a persistent SSHFS connection (for the session directory). The executable is submitted to the remote SLURM queue by running *sbatch* over SSH.
- The job status is queried regularly by running *squeue* over SSH.
- After a job completes on the HPC machine, the data created in the session directory is copied back over SSHFS and uploaded to an ATLAS Storage Element for persistent storage or further analysis.

For this implementation, a virtual machine at the University of Bern was successfully used as the ARC frontend to the “Todi” Cray HPC machine at CSCS (Swiss National Supercomputing Centre).

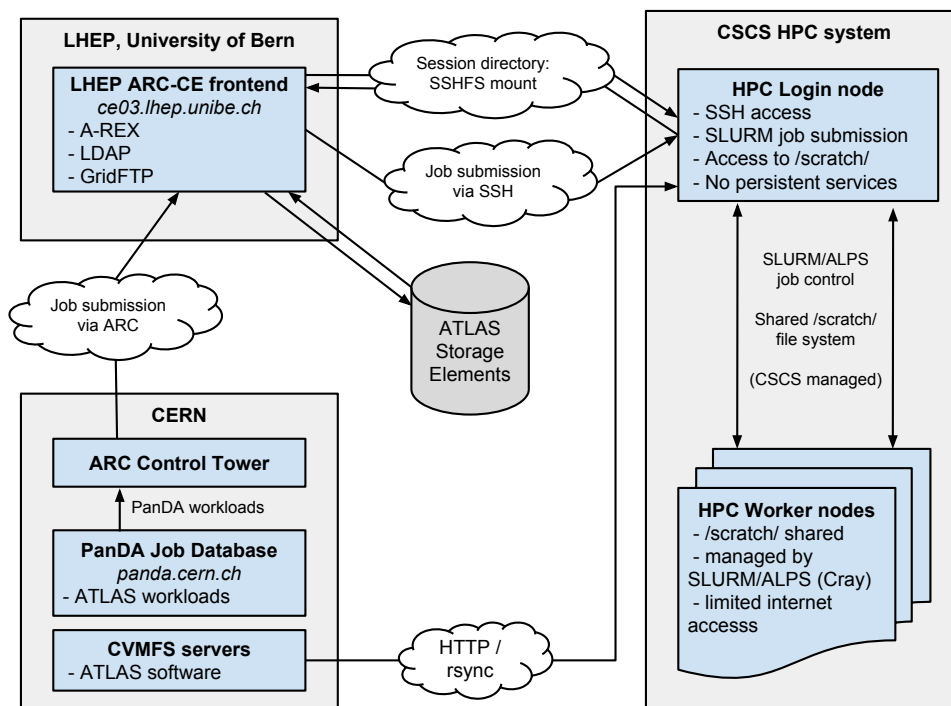


Figure 3. The architecture of the ARC-CE as a frontend to a remote HPC system. Clouds denote connections made over the network.

4.2.1. File system access: SSHFS

SSHFS (SSH File System) is an open-source file system driver to mount and transparently access remote file systems over SSH and its SFTP (secure file transfer) functionality. In order to allow transparent access from the ARC-CE frontend to job session directories available on the HPC system, a shared file system available on the HPC system (e.g. scratch) is mounted on the ARC-CE frontend through SSHFS. This is possible since no long-term storage of session directories is required.

While SSHFS allows transparent access to the remote file system, some system level modifications needed be implemented on the side of the ARC interface implementation: for example, care must be taken to synchronize the file system paths and file ownership between the ARC-CE frontend and the

interfaced HPC system. Specifically, the path to the session directory is fixed in the ARC-CE configuration and must be the same as the one on all compute nodes. Additionally, to avoid file ownership problems, the user id and group id of the user account used to log into the remote HPC system must be mapped on the ARC-CE to both root and the local user ARC uses for running Grid jobs. The remote file system mounted over SSHFS can also be used by ARC to maintain the cache of recently used input files on the remote HPC system.

4.2.2. Remote Resource Manager access

The CSCS Cray HPC systems use a Cray-specific combination of SLURM (Simple Linux Utility for Resource Management) and the Cray ALPS (Application Level Placement Scheduler) to manage jobs. Job scripts are submitted through the standard SLURM command line tool sbatch from the HPC login node, which queues the job and reserves the requested resources. When a job gets to run, SLURM executes the submitted job script on a HPC service node. The job script must then call the ALPS aprun command to execute a script or binary on the reserved compute nodes. The job status can be queried from the HPC login nodes through standard SLURM command line tools (e.g. squeue or scontrol).

To run and manage jobs on the HPC system through the ARC-CE frontend, a script has been implemented (sshsbatch) to call the SLURM command line tools on the remote system transparently over SSH. It uses the password-less SSH Public Key method for authentication and optionally allows using SSH connection sharing to minimize the resource overhead of the SSH connection. The script is not called directly, but by symbolic links named after the SLURM commands to run on the remote system (e.g. sbatch, scancel, scontrol or squeue).

Additionally, the ARC-CE interface to SLURM was modified to generate a special job script, which takes into account the hybrid SLURM/ALPS architecture of Cray HPC systems and runs the job through ALPS aprun when it is executed by SLURM. While this implementation is specific to the system targeted, the principle is not, and the script can be easily adapted to access other HPC resource managers remotely.

5. Status of integration

The solution is in use at HPC centres in Switzerland, Germany and China. At CSCS, ATLAS Monte Carlo production workloads run routinely on the “Todi” Cray used for the development. On this system, scalability tests have been performed and steady streams of up to 100 concurrent 16-core jobs have been run steadily. More details on the specific implementation and performance are to be found in [13]. At the SuperMUC HPC system at the Leibniz Supercomputing Centre in Munich, Germany, a version of this solution adapted to the LoadLeveler resource manager is in production. Both run unmodified ATLAS workloads. Work is ongoing in integrating the Pi supercomputer in Shanghai and more European and Chinese centres are expected to be integrated in the near future. The SSH backend is expected to be integrated in a future release of the ARC middleware.

6. Conclusions

The capability of exploiting existing general purpose HPC resources is crucial to guarantee that enough CPU power will be available for the next few years to ATLAS and the other LHC experiments that face challenges in meeting increasing computational requirements within the constraints of current budgets. The SSH backend developed at the University of Bern allows the transparent integration of remote HPC systems within the ATLAS distributed computational infrastructure, in compliance with the strict HPC policies. Application software is provisioned via rsync to the remote system. Remote access to the resources is accomplished by exploiting the architecture of the ARC Compute Element, extended by a module operating through SSHFS to implement all communication needed with the HPC system for payload submission, execution, monitoring and output retrieval. The solution ensures that the job execution on a remote HPC system is fully integrated within the ATLAS distributed computing infrastructure.

References

- [1] I. Bird, "Computing for the Large Hadron Collider", *Annual Review of Nuclear and Particle Science*, Vol. 61: 99-118, 2011
 Les Robertson, "Computing Service for LHC: From Clusters to Grids", Chapter 3 of *From the Web to the Grid and Beyond, Computing Paradigms Driven By High Energy Physics*, R. Brun, F. Carminati, G.G. Carminati, eds., Springer-Verlag Berlin Heidelberg, 2012
- [2] ATLAS Collaboration 2008, "The ATLAS Experiment at the CERN Large Hadron Collider", *JINST* 3 S08003 doi:10.1088/1748-0221/3/08/S08003
- [3] I. Bird et al., "Update of the Computing Models of the WLCG and the LHC Experiments", CERN-LHCC-2014-014, 2014
- [4] TOP500 Supercomputing Sites, <http://www.top500.org>
- [5] From <http://www.top500.org>: used under the Commons Attribution-Share Alike 3.0 Unported license
- [6] T. Gleisberg et al., "Event generation with SHERPA 1.1", *JHEP*02 007, 2009
 T. Sjöstrand et al., "An introduction to PYTHIA 8.2", *Comp. Phys. Comm.* 191, 159-177, 2015
 M. L. Mangano et al., "ALPGEN, a generator for hard multiparton processes in hadronic collisions", *JHEP*07 001, 2003
- [7] The Geant4 Collaboration, S. Agostinelli et al., "Geant4, a simulation toolkit", *Nuclear Instruments and Methods in Physics Research A* 506, 250-303, 2003
- [8] Barrand G. et al., "GAUDI - A software architecture and framework for building LHCb data processing applications", *Comp. Phys. Comm.* 140, 45-55, 2001
- [9] J Blomer et al., "CernVM-FS: delivering scientific software to globally distributed computing resources", proceedings of the first international workshop on Network-aware data management, 49-56, 2012
- [10] D. Thain and M. Livny, "Parrot: Transparent User-Level Middleware for Data Intensive Computing", Workshop on Adaptive Grid Middleware at PACT, January 2003
- [11] T. Maeno on behalf of the PANDA team and the ATLAS collaboration, "PanDA: distributed production and distributed analysis system for ATLAS", *J. Phys.: Conf. Ser.* 119 062036, 2008
 P. Nilsson et al. on behalf of the ATLAS collaboration, "The PanDA System in the ATLAS Experiment", *PoS (ACAT08)*027, 2008
- [12] M. Ellert et al., "Advanced Resource Connector middleware for lightweight computational Grids", *Future Generation Computer Systems* 23 (2007) 219-240
- [13] M. Hostettler et al., "ATLAS Computing on CSCS HPC", this proceedings.