Interfacing the ATLAS grid production system with IDRIS: Activity Report

E. Vamvakopoulos, F. Hernandez [1] and A. Ansari [2]

[1] CC-IN2P3

vamvakop@cc.in2p3.fr, fabio@in2p3.fr

[2] IDRIS

Agnes.Ansari@idris.fr

August 18, 2017

1 Introduction

The ATLAS collaboration has recently organized a new computational activity: HPCATLAS. The aim of this activity is to interface, in an efficient way, its ATLAS Distributed Computing System with HPC environments operating resources for intensive scientific computing [1]. HPC environments are attractive for the ATLAS collaboration as they have the potential to contribute significant amount of computational resources. The utilization of those resources could happen in an opportunistic way (i.e. back-fill mode) or as a result of allocated CPU time, depending on the policies of each center. In addition, ATLAS future computational needs for RUN3 period will be massive due to future increase in the trigger output rates and detector operation at higher energy and luminosity of LHC accelerator. Those developments will entail processing and simulation of larger volumes of data.

In the past few years, many HPC centers around the world have been interfaced with the ATLAS computing system: SuperMUC, MPCDF, CSCS, C2PAP, NERSC, OLCF/TITAN and others [3]. In the first quarter of 2017 usage of HPC systems by ATLAS (in terms of wall-clock time) represented

5.82% of the total usage over a total of 764.9 million hours [2]. Furthermore, the 58% of total usage for the reported period corresponds to Monte Carlo simulation which confirms the importance of this activity. In figure 1, we present the total number of HPC CPU slots used by the ATLAS collaboration for a representative period (first quarter of 2017).

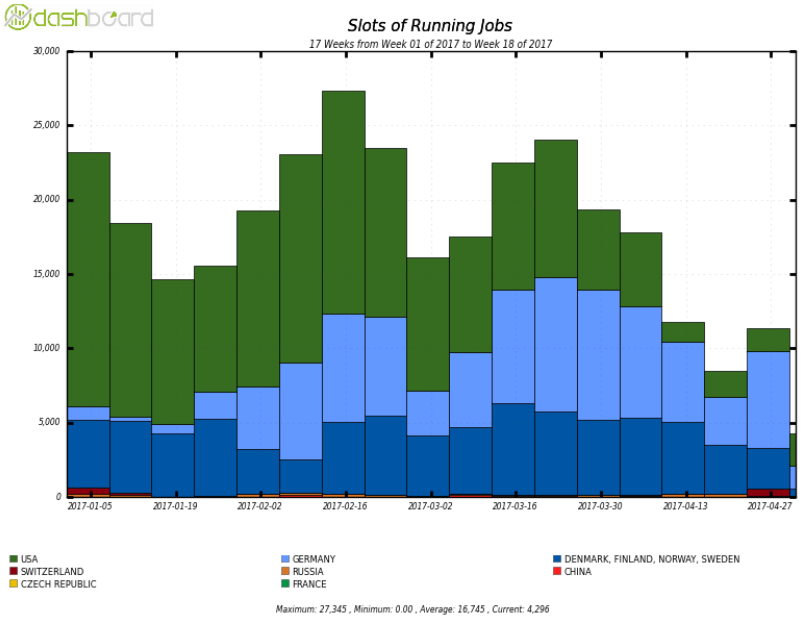

Figure 1: Evolution of HPC slots used by the ATLAS collaboration along a representative period (first quarter of 2017). The different colors in stacked bars correspond to different countries' contribution of HPC resources.

In this document, we report on our work aiming at demonstrating the technical feasibility of executing ATLAS computing workflows at IDRIS. This is the first time this work is performed for a French HPC center. In the next section we describe the execution environment at IDRIS, the CNRS' Institute for Development and Resources in Intensive Scientific Computing, section 3 presents in more detail the integration work and section 4 presents the obtained results before concluding.

2 IDRIS HPC environment

At IDRIS, there are two machines currently in operation for intensive numerical computing: an IBM BlueGene-Q with PowerPC-A2 CPU, 90k slots and 2GB per CPU (named Turing) and an IBM x86 machine based on Intel E5-4650@2.7GHz with 10k CPU slots and InfiniBand interconnection (named Ada). The vast majority of the Ada nodes have 128GB of RAM (304 nodes), and 28 nodes have 256GB. The exploitation of both machines is done via IBM Load-Leveler 5.1 as batch system (one instance per machine). The compute nodes of both machines have all access to the local GPFS file-system which hosts the user's home and the working directories for the applications. A temporary storage area is also supported.

IDRIS HPC environment exhibits the typical characteristics of a supercomputer center, such as:

- Tight access rules (i.e. only via ssh)
- Outbound internet connections from the compute nodes are not permitted
- Execution of long-lived applications (i.e. services) is not allowed
- Different CPU architectures
- Shared network filesystem (i.e. GPFS)

For the work object of this report, we decided to focus on the Ada machine. We did not explore the Turing as a test option, because the ATLAS software (Monte Carlo) is not ready to smoothly exploit the PowerPC architecture [4] and the Turing machine is close to the end of its life. The Ada machine consists of x86_64 Intel CPUs and the nodes are running RedHat Enterprise 6.x. Those characteristics permit us to use ATLAS software and related tools (i.e. Parrot clients, cvmfs) as is and avoid complicated porting of the software packages. Practically, all ATLAS packages can run in the Ada machine. The most stringent constraint is the lack of external WAN connectivity from the compute nodes. For this demonstrator, we focused on executing ATLAS Monte Carlo jobs (MC) on Ada . Those jobs are

CPU-bound, need relatively small amount of RAM (2GB per core), don't need external connectivity in order to access central databases (i.e. Frontier System) [17] the amount of input and output data they handle is negligible (order of Mbytes per job) [5], their wall-clock time can vary from a few hours up to 96 hours [6].

ATLAS software can run in two modes: the traditional multi-core mode [7] and the event service mode [8]. The design of the event service mode allows for jobs to be preempted without loosing the work performed up to its preemption. This is attractive because it gives more operational flexibility. However, for the first phase of our tests, we focused our effort on the stability and performance (in terms of CPU efficiency and event throughput) of traditional multi-core job.

3 Implementation

In order to interface the ATLAS computing system to the Ada machine, we choose to use the ARC-CE computing element. ARC-CE acts as a gateway to collect job submission requests emitted by authorized production users, submits them to the Ada workload management system, monitors their execution status and collects the job results. This solution has been originally proposed by S. Haug et al. [13] and is used by ATLAS for integrating several HPC sites to its computing system (i.e. NorduGrid, SuperMuc, Hydra) [14]. At CC-IN2P3, we installed an ARC-CE instance in the private cloud infrastructure. The corresponding virtual machine have typical characteristics: 8 vCPU, 8GB RAM, 80GB local disk, 1Gbit/sec internet network card. This configuration permits to exploit the Ada resources from a host at CC-IN2P3 in the least possible invasive way. The components of the ARC-CE solution are the following (figure 2).

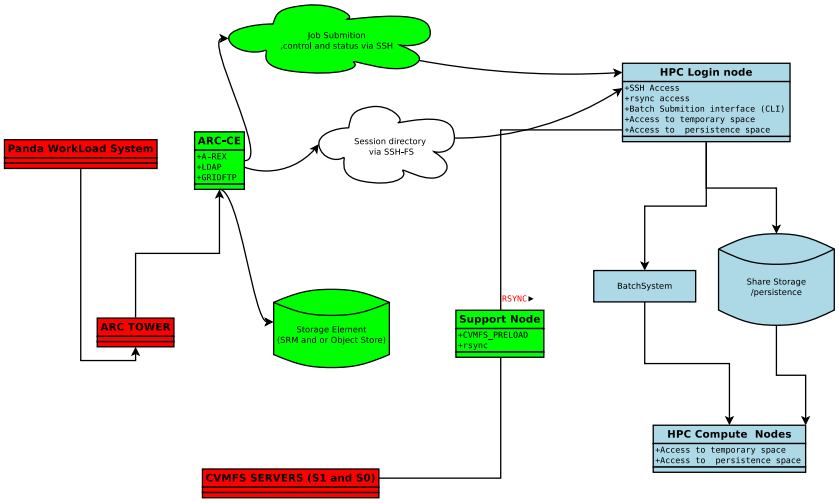

Figure 2: Context Diagram of ARC-CE solution interface. Blue objects represent ADA resources, Red objects represent services which are installed at CERN and are part of the ATLAS grid distributed system. The objects in green represent services which are installed at CC-IN2P3.

3.1 Dedicated IDRIS environment

Two user accounts, belonging to the same group, were created in Ada for the purposes of this work. The first account is used for job submission and the second is used for deployment of ATLAS software in Ada. Commands are sent to Ada machine in a secure way via a ssh connection which uses RSA-based public key authentication. A dedicated class in Ada's batch system was created, namely atmt32. This queue was aimed to run jobs in a back fill mode with a low priority. This allows to run jobs which are not going to compete with the Genci jobs. The execution wall-clock upper limit for jobs in this class was 1.5 hours and allows to run 32 threads in parallel. All production jobs were supposed to be submitted in this queue. The standard disk quotas were extended to 4.5 TB and 4.5 millions of files to fulfill the requirements of the project.

3.2 ARC-CE grid gateway

ARC-CE is a grid gateway. The grid gateway interfaces the batch system of a computer farm with the worldwide grid system [10]. ARC-CE translates a job request based on a set of well-defined attributes (JDL) into an ordinary batch job request (LRMS). Also, ARC monitors the job life-cycle in the batch system and can issue simple batch system commands like submitting a job, getting the status of a job, canceling a job. The authentication and the authorization take place via X509 grid certificates. The communication with the actual ARC-CE clients (i.e. ARC-TOWER) take place via the gridftp protocol. Minor changes in ARC-CE batch system interface were applied to make it compatible with the IDRIS batch execution environment. ARC-CE can perform data management like stage-in and stage-out from the computer farm's filesystem (local or networked) of the job's data. The ARC-CE data management component has the ability to connect and operate with different data transfer protocols, such as srm, gridftp, https, ftp, object store etc. This gives to ARC-CE a unique feature, from a technical point of view. It can manage all the data transfers from/to a central grid storage. In addition, the description of data files and transfer endpoint service is part of the job request (JDL). The ARC-CE data management component supports data caching capabilities. All data management tasks are performed at the ARC-CE node, on top of a session directory. In the next paragraph, we explain how this directory is shared between the ARC-CE node and the HPC nodes. ARC-CE solution guarantees that the job's life cycle on the remote HPC system is integrated with in the ATLAS distributed grid production system.

3.3 SSH-FS

ARC-CE by design works with a shared file system. All the files related to the job life-cycle, such as the job script, job input/output and the related errors and status files, are stored in the session-specific directory. This directory corresponds to a semi-persistence disk area, which is shared over the network (e.g like NFS) between the ARC-CE gateway and all the compute nodes of the farm. In order to emulate a shared file system between the ARC-CE node and the nodes of the HPC machine, we use the ssh-fs

[11]. The ssh-fs is an application that can mount a remote directory (via the FUSE module) over ssh and sftp. We create a proper directory in the $WORKDIR on Ada GPFS file-system and mount this directory on the ARC-CE node at CC-IN2P3. The ARC-CE daemons can write and read files on this directory and share those files with both the Ada front-end node and the Ada compute nodes. Ssh-fs supports POSIX ACL compatibility and UID/GID translation between the local users and the HPC user. Although there are limitations related to synchronization, latency and performance, this solution seems adequate for a small number and volume of files. Ssh-fs permits to create an interface of data stage-in and stage-out from HPC machines in a non-invasive way.

3.4 Software Provisioning for Ada machine

During the past few years, LHC experiments use an innovative and an effective solution to distribute software across the grid computer farms around the world: CernVM-FS. CernVM-FS is implemented as a POSIX read-only file system in user space [9, 18]. The files and directories are hosted on standard web servers and the directory tree structure can be mounted in the client in the universal namespace /cvmfs.

The ATLAS software is publicly available in pre-compiled releases for x86_64 based Linux systems from the CernVM-FS file-system. In HPC machines the full deployment of CernVM-FS solution is not always applicable due to various constrains (i.e. lack of FUSE module in the compute nodes, external connectivity of compute nodes, lack of local disk volumes, etc). Consequently, we need another non-intrusive way to deploy the ATLAS software on Ada. With appropriate CernVM-FS tools (i.e. cvmfs_preload), we can select particular directories from the ATLAS software CernVM-FS repository and can create a custom cache. This custom cache contains a collection of encoded files (cvmfs chunks), which are organized in an appropriate directory structure. We selected about 1.7 Tbytes over 1.5 million of files which correspond to ten (10) ATLAS Monte Carlo software releases plus the pool condition data. As a consequence, Ada can run only predefined jobs according to the software release present in the local cache collection.

We stored this selection in the local CC-IN2P3 GPFS instance. We replicated this cache collection (in multiple archive files) to the network shared file-system of the Ada machine via sftp and we used particular nodes (ADAPP[1]) in order to extract the custom cache from those tar files. The initial population of the ATLAS software in the network shared file-system of the Ada machine have been done manually.

The synchronization of the custom cache have been done in two steps. First, we run the cvmfs_preload on CC-IN2P3 systems in order to synchronize the CernVM-FS external repository with the local GPFS instance. Later, we employed the rsync tool in order to synchronize the local CC-IN2P3 GPFS instance with remote IDRIS GPFS instance [2].

Using the Parrot client, we can read (decoded) this cache collection and mount in the user space our software selection under the universal name space $/cvmfs$ without requiring the FUSE Linux kernel module. It works by trapping a program's system calls through the ptrace debugging interface, and replacing them with remote I/O operations, as desired. This permits deploying the software and the related dependencies into the HPC machine in the least possible intrusive way and without performing modifications on the software. In contrast, the Parrot tool is very tightly coupled with the OS kernel and so it is only available on Linux based operating systems [12]. The Parrot client method is a straightforward method of software replication from the CernVM-FS repository to the HPC system shared files-system.

Another option for the ATLAS software provisioning is to replicate (fully or partially) the ATLAS cvmfs repositories on top of the GPFS filesystem and do proper modification of the installation paths of the software from $/cvmfs$ to $/gfs/area/user/mydir$ [16]. The issue of this method that the initial population and synchronization (i.e. with rsync) of the repository involves a large number of files (in the millions). In the future, HPC architectures where the usage of the Parrot client could be not applicable, we will consider this method.

The ATLAS software provisioning on HPC environment is real a chal-

[1]Adapp is the IDRIS computer dedicated to intensive pre- and post-processing and to managing great masses of data

[2]The update proccess is not so heavy, we need to update 1000-2000 of files

lenge. There are many different scenarios for copying and synchronizing the software and we would like to investigate them. We should find a technical solution for the automatic and seamless replication of ATLAS software, that will respect the limitations of the experiment without disturbing the smooth operations of the HPC shared storage system.

3.5 ARC in ATLAS Grid System

We setup three different queues in the level of ATLAS Global Workload System (Panda):

- IN2P3-CC_HPC_DEBUG
- IN2P3-CC_HPC_IDRIS_MCORE
- IN2P3-CC_HPC_IDRIS_MCORE1

The IN2P3-CC_HPC_DEBUG queue corresponds to a single core class of Ada (named t2 class), with max wall-clock time 1.5 hours. We use this queue for short testing propose.

The IN2P3-CC_HPC_IDRIS_MCORE queue corresponds to a multi-core class (named mt32t3), with 32 CPUs and a maximum wall-clock time of 18 hours. We use this queue for testing propose.

The IN2P3-CC_HPC_IDRIS_MCORE1 queue corresponds to the dedicated atmt32 class, with a max wall-clock time of 1.5 hours.

In brief, ATLAS Production Operators subscribe tasks in the ATLAS Production System. ATLAS Production system submits the tasks in Panda which creates ready to be process jobs. The ATLAS jobs ready to be processed on a HPC Panda queue (i.e. IN2P3-CC_HPC_IDRIS_MCORE) are sent via the arcControlTower to the corresponding ARC-CE queue at CC-I2NP3. The arcControlTower service [15] take care of all the communications with the Panda (WorkLoad Management System of ATLAS) :

- The ARC-CE, using the information contained inside the job description file (JDL), downloads the input data into the network shared area (sessiondir) via ssh-fs.

- The ARC-CE translate the information inside to the JDL into a proper job script file and submit this job script to the Load-leveler via ssh on Ada front-end node.
- Periodically, arcControlTower interrogates the ARC-CE (and later the ARC-CE the batch system of Ada) and provides the job status to the Panda system.
- When some job is successfully finished, the ARC-CE transfers the results directly to the predefined ATLAS storage, in our case on CC-IN2P3 dCache storage.

4 Tests and Results

4.1 Hammer-Cloud tests

We run ATLAS Hammer-Cloud (HC) tests, in order to test and debug our installation. Hammer-Cloud is a testing system developed by CERN IT initially for ATLAS and then adapted by CMS to run custom tests on grid sites [19]. Hammer-Cloud jobs are based on well-defined templates which can use predefined datasets or random selected datasets. Our tests are based on template 914, which corresponds to a Monte Carlo job based on a particular dataset and runs for two (2) events only. In those tests, we are interested in testing the stability of the short job of two (2) events (30min wall-clock time), which is adequate for this purpose. We run our test for several days and for numerous cycles. In all our tests, the job efficiency (i.e. the number of successful jobs divided by the number of all submitted jobs) was more than 95%. We kept the number of submitted and running jobs small (1-3 per 3 hours) and we did not perpetuate the system. Hammer-Cloud single core jobs are very important work-flows for checking and verifying configurations. We need access to a single queue in order to avoid wasting multi-core resources and to profit from the fast starting of the jobs. In figure 3, we present time evolution for an indicative test period of single HC jobs which we run on the Ada machine.

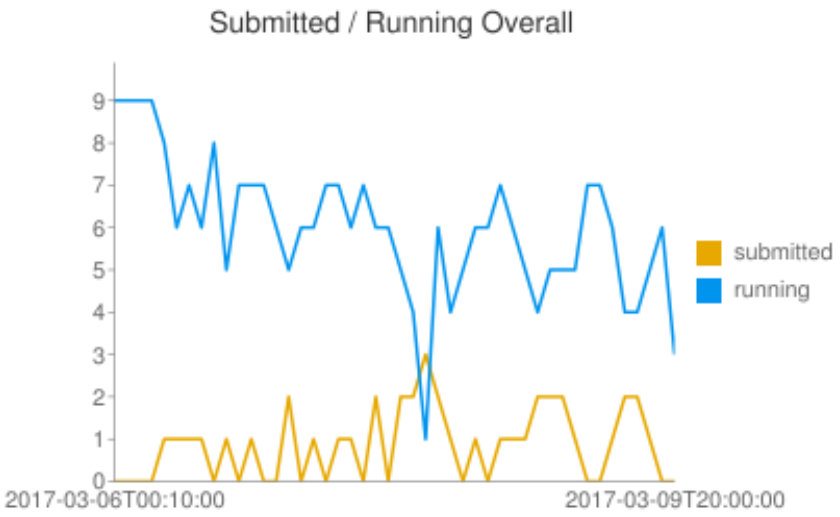

Figure 3: Evolution of used slots for HammerCloud tests. The Blue line corresponds to the number of running jobs and the orange line corresponds to the number of summitted jobs.

4.2 ATLAS MC tests

In order to test our interface platform with ARC-CE and ssh-fs, we run real ATLAS Monte Carlo jobs (MC). Those jobs were subscribed automatically to queues of IDRIS by the ATLAS production system. We allowed the system run up to 400 jobs, during four (4) days of testing. The wall-clock time of those jobs ranged from 3h to 5h. The system ran up to 78 concurrent jobs which correspond to 2,500 CPU slots. In addition, the success rate of real MC jobs was more than 95%, while the jobs exhibited CPU efficiency of more than 80%. In figure 4, we present the time evolution of used slots for IN2P3-CC_HPC_IDRIS_MCORE queue. For this type of test, we use the mt32t3 class, in order to have some flexibility in the wall-clock limit. The total amount of processed data (stage-in) was 30GB and the total amount of produced (stage-out) data was 300GB over the duration of the tests. Those data volumes create very low traffic from/to the ADA front-end machine on top of ssh-fs application (short peaks of 5 to 10 MB/sec). The total wall-clock consumption for those tests was 40,000 hours, while the total wall-clock ATLAS consumption of Tier1 CC-IN2P3 for the same

period was 1,642,088 hours (within week 18). We need to repeat those tests (even with a lower number of CPU slots), in order to understand better the limits of our solution. Also, we need to verify our integration solution over a longer duration of the tests with a steady rate of submitted jobs and recheck the network traffic. Unfortunately, it was not possible to run real jobs on atmt32 class due to a wall-clock limit of to 1.5 hours. Even if the workload system tried to find and create suitable jobs with respect to the time limit, this limit is too low for this class and all jobs would fail to run. The origin of this behavior needs further investigation. We suppose that the origin of this behavior resides in the insufficient accuracy of the system to predict the wall-clock time of the jobs below a certain limit. For the regular future use of Ada (or similar HPC system), we would need to extend the wall-clock time of atmt32 to higher limits (i.e. to 6 hours).

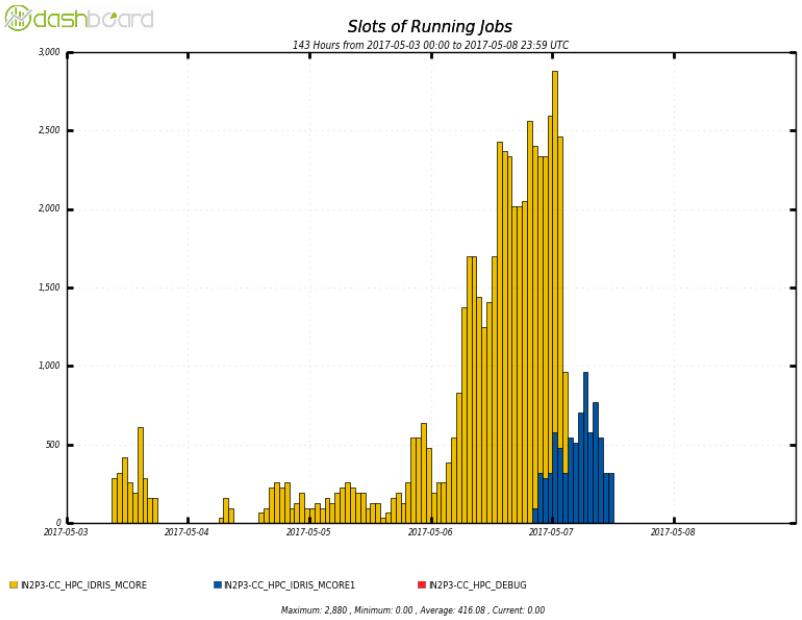

Figure 4: Evolution of used slots for IN2P3-CC_HPC_IDRIS_MCORE and IN2P3-CC_HPC_IDRIS_MCORE1 queues for the duration of the test (week 18).

5 Conclusions

We installed and configured an ARC-CE instance at CC-IN2P3, in order to interface the IDRIS Ada machine within the ATLAS grid production system. We ran multiple cycles of Hammer-Cloud test jobs and one cycle of real ATLAS multi-core jobs in order to test and validate our solution. For software deployment, we used the parrot client method as a straightforward solution and avoided any modification of the ATLAS software. The software replication can be further automated and further investigation of the software provisioning solutions in architectures where the parrot client usage is not applicable is required. The automatic replication of the experiment's software (and/or data) from a central grid repository to HPC storage area is an important aspect which gives a room for improvements. Moreover, ATLAS Monte Carlo jobs run with job efficiency more than 95% and CPU efficiency more than 80% for wall-clock times between 3 hours and 5 hours on Ada machine. Those results are promising for future systematic usage of the Ada machine (or similar one based on x86_64 architecture). Unfortunately, the dedicated class atmt32 has a limited wall-clock time of 1.5 hours which is not adequate for our working conditions. Further investigation is needed in order to run real ATLAS jobs on this kind of short duration class. The opportunistic usage of a significant amount of resources for short-time is an issue that should be addressed within the ATLAS computing collaboration. In the meantime, we are looking forward for the relaxation of wall-clock limit in atmt32 class (up to 6 hours) in order to continue our test at limited scale. This requirement doesn't seem possible to obtain given it would not fit with the back fill mode and would compete with the Genci jobs.

Acknowledgments

Many thanks go to all CC and IDRIS experts for their kind technical support and to ATLAS HPC experts for fruitful discussions and for providing guidelines regarding the details of the ARC-CE solution.

Acronyms

ACL Acccess Control List
ARC Advanced Resource Connector
ATLAS One of the four major experiments at the Large Hadron Collider at CERN
FUSE Filesystem in Userspace
GID Group ID
GPFS General Parallel File System
HPC High Performance Computing
JDL Job Description Language
LHC The Large Hadron Collider
LRMS Local Resource Management System
NFS Network File System
POSIX Portable Operating System Interface
RUN3 corresponds to 2020-2023 operation period of LHC
UID User ID
WAN Wide Area Network
X509 is a standard that defines the format of public key certificates

References

[1] ATLAS High Performance Computing Initiative Wins Award, http://ATLAS.cern/updates/ATLAS-news/ATLAS-high-performance-computing-initiative-wins-award

[2] ATLAS Accounting for the first quarter of 2017, https://tinyurl.com/ATLAS-hpc-wallclock-Q1-2017

[3] Opportunistic resources HPC, Alexei Klimentov, Brookhaven National Laboratory, ATLAS Jamborre 2017, https://indico.cern.ch/event/440821/contributions/1931075

[4] Private comminication with ATLAS HPC experts

[5] Diskless Sites, E.Vamvakopoulos, CC-IN2P3, WLCG WORKSHOP 2017, https://indico.cern.ch/event/609911/contributions/2605033/

[6] https://operations-portal.egi.eu/vo/view/voname/ATLAS

[7] https://twiki.cern.ch/twiki/bin/view/LCG/DeployMultiCore

[8] The ATLAS Event Service: A new approach to event processing, P. Calafiura, K. De, W. Guan, T. Maeno, P. Nilsson, D. Oleynik, S. Panitkin, V. Tsulaia, P.V. Gemmeren, T. Wenaus on behalf of the ATLAS collaboration, , Journal of Physics: Conference Series 664 (2015) 062065

[9] CVMFS Technical Document, http://cvmfs.readthedocs.io/en/stable/cpt-hpc.html?highlight=parrot

[10] http://www.nordugrid.org/arc/

[11] https://github.com/libfuse/sshfs

[12] http://ccl.cse.nd.edu/software/manuals/parrot.html

[13] The ATLAS ARC backend to HPC, S. Haug, M. Hostettler, F.G. Sciacca, M. Weber on behalf of the ATLAS collaboration, Journal of Physics: Conference Series 664 (2015) 062057

[14] Bringing ATLAS production to HPC resources - use case with the Hydra supercomputer of the Max Planck Society, A. Kennedy, S. Kluth, L. Mazzaferro and R. Walker on behalf of the ATLAS Collaboration, Journal of Physics: Conference Series 664 (2015) 092019

[15] ARC Control Tower A flexible generic distributed job management framework, J.K. Nilsen, D. Cameron and A. Filipcic, Journal of Physics: Conference Series 664 (2015) 062042

[16] Stratum-R service, https://indico.cern.ch/event/579473/contributions/2429450/

[17] Scaling HEP to Web Size with RESTful Protocols: The Frontier Example, D. Dykstra, Journal of Physics: Conference Series 331 (2011) 042008

[18] Security in the CernVM File System and the Frontier Distributed Database Caching System, D. Dykstra and J. Blomer, Journal of Physics: Conference Series 513 (2014) 042015

[19] Experience in Grid Site Testing for ATLAS, CMS and LHCb with HammerCloud, J. Elmsheuser, R.M. Llamas, F.Legger, A. Sciaba, G. Sciacca, M.U. García and Daniel van der Ster, Journal of Physics: Conference Series 396 (2012) 032111