Backfilling the Grid with Containerized BOINC in the ATLAS computing

Wenjing Wu, David Cameron, on behalf of the ATLAS collaboration

1 Institute of High Energy Physics, CAS, 19B Yuquan Road, Beijing, 100049, China
2 Department of Physics, University of Oslo, P.b. 1048 Blindern, N-0316 Oslo, Norway

E-mail: wuwj@ihep.ac.cn

Abstract. Virtualization is a commonly used solution for utilizing the opportunistic computing resources in the HEP field, as it provides a unified software and OS layer that the HEP computing tasks require over the heterogeneous opportunistic computing resources. However there is always performance penalty with virtualization, especially for short jobs which are always the case for volunteer computing tasks, the overhead of virtualization becomes a big portion in the wall time, hence it leads to low CPU efficiency of the jobs. With the wide usage of containers in HEP computing, we explore the possibility of adopting the container technology into the ATLAS BOINC project, hence we implemented a Native version in BOINC, which uses the singularity container or direct usage of the target OS to replace VirtualBox. In this paper, we will discuss 1) the implementation and workflow of the Native version in the ATLAS BOINC; 2) the performance measurement of the Native version comparing to the previous Virtualization version. 3) the limits and shortcomings of the Native version; 4) The use case and harvest of the Native version which includes using it in backfilling the ATLAS Grid Tier2 sites and other clusters, and using it to utilize the idle computers from the CERN computing centre.

ATL-SOFT-PROC-2018-060
06 December 2018

1. Introduction

Volunteer computing started to become popular in the late 1990s. Its goal is to use the free CPU cycles of worldwide volunteers personal computers for scientific computing. BOINC[1][2] is a middleware developed and commonly used for volunteer computing. Some projects, such as SETI@home[3] and Eintein@home[4], became very successful, and they not only gained a big amount computing resources, but also attracted wide public interest and attention to their research projects. Volunteer computing started to be applied to the HEP computing in 2004 with the launch of the LHC@home[5] project which was to run a small simulation of the accelerator.

Utilizing all these volunteer computers with heterogeneous operating systems and software environment requires the application to be platform and software environment independent. The application for LHC@home is a small FORTRAN program, so it is possible to port the application with some modification and recompilation of the source code on different operating systems.Big HEP experiments offer a different use case, as a single release of the software can be as large as 10GB and heavily depends on various platforms libraries. Hence it is very difficult to migrate the software to more operating platforms other than a few customized versions of Linux. This became a big obstacle to applying volunteer computing to the LHC experiments.

The combination of CernVM[6] and CVMFS[7] made it possible to run their computing tasks inside a 1GB size image on top of various operating systems. However a certain amount of

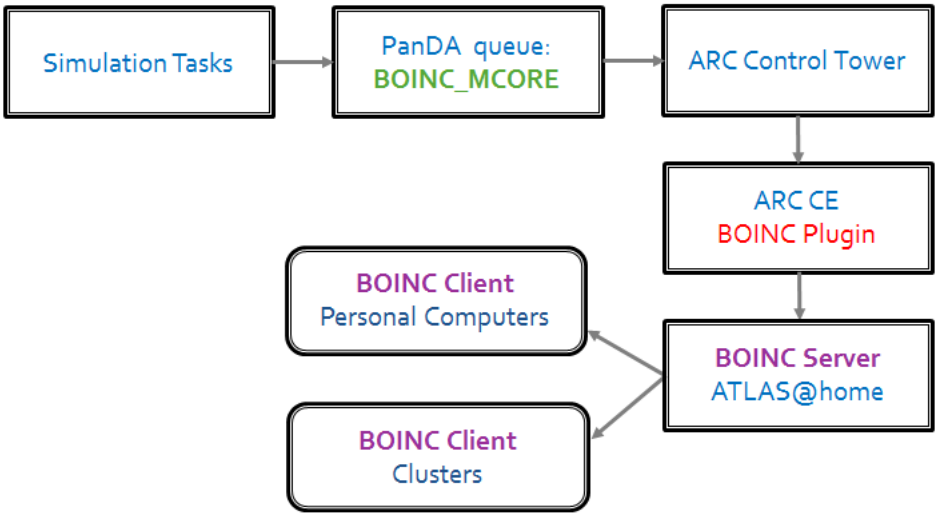

Figure 1. ATLAS@home architecture and workflow

performance penalty is unavoidable with virtualization and also it requires the pre-installation of VirtualBox on the volunteer computers. The containerization technology which can also provide the required software environment for application, started to be developed and be used in the HEP computing field in recent years as a more lightweight and performant solution comparing to virtualization. In this paper, we explore the advantages and use cases of containerization in the context of the volunteer computing project ATLAS@home.

2. ATLAS@home

ATLAS@home[8] [9] is a BOINC based volunteer computing project started in 2013 to use the free CPU cycles from worldwide volunteer computers and apply them to the simulation computation of the ATLAS experiment[10][11]. ATLAS@home was the first volunteer computing project in the field of HEP experiments.

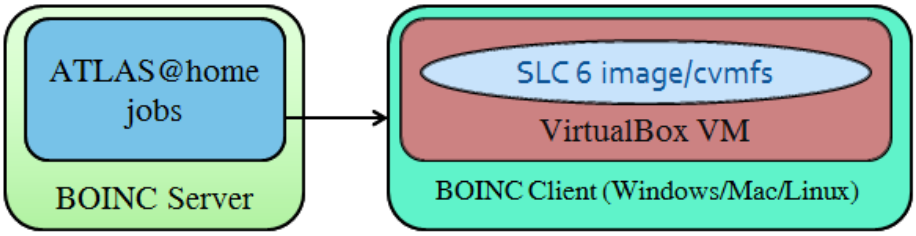

Figure 2. The virtualization of ATLAS@home

As shown in Figure 1, in its workflow and architecture, ATLAS@home is fully integrated into the grid computing infrastructure of the ATLAS experiment, through the common interface PanDA[12][13], the production manager can submit their simulation tasks[14][15] to ATLAS@home which appears as a PanDA queue; Jobs from the simulation tasks are pulled to the ACT(ARC Control Tower), then ACT sends the jobs to the ARC CE that forwards the jobs to the BOINC server; The BOINC clients request and fetch the jobs from the BOINC server whenever they have idle CPU cycles.

As of now, ATLAS@home has become a very reliable computing resource for the ATLAS experiment, important simulation tasks are run via ATLAS@home, and on average it contributes about 15KHS06 computing power to the ATLAS computing on a daily basis.

3. Virtualization

3.1. Virtualization for ATLAS@home

When the project was first setup, the target hosts were the worldwide volunteer computers which were very heterogeneous in terms of operating systems and software environment. According to the ATLAS@home project monitoring, over 85% of the volunteer computers use various Microsoft Windows operating systems, and about 10% use various Linux operating systems, and the other use various Mac OS. So in the initial workflow, ATLAS@home adopted VirtualBox to virtualize the target hosts, they could have the software environment that the ATLAS simulation tasks require.

As shown in Figure 2, when a BOINC client attaches to the ATLAS@home project, it first downloads a program called Vbox-wrapper, which controls VirtualBox to create, run and destroy a virtual machine based on a virtual machine image. This image caches the basic ATLAS software that the simulation tasks require through the CVMFS client, and can be cached on the client for jobs to reuse until there is a new release of image from the BOINC server. The size of image is about 1.3GB.

When the BOINC client gets a job, the Vbox-wrapper starts a virtual machine based on the cached image, then the simulation job is launched from inside the virtual machine. The virtual machine is shut down and destroyed when the job is finished.

3.2. The disadvantages of Virtualization

There are a few obvious disadvantages of using virtualization, mainly from the performance point of view:

- The process of creating the virtual machine image is tedious.
- Whenever there is a new software release, the virtual machine image needs to be updated on both the server and client sides.
- Every client needs to download and cache the full image, and the image is usually between 1 and 2 GB in size.
- For every new job, virtualization also introduces overhead in order to clone the image, and create and start the virtual machine. These extra work can significantly reduce the CPU efficiency of the short wall time jobs, which is a common case for the ATLAS multicore simulation jobs[16].
- Virtualization has performance penalty in IO and CPU in general.
- Most of the ATLAS software is cached in the image, but extra files, including conditional database files, might need to be downloaded on the fly, and they cant be cached and shared by different jobs on the same host.

All the above factors can add up, and impact the CPU efficiency of the ATLAS simulation jobs.

4. Containerization

4.1. Singularity and its performance test

Singularity[17] is an implementation of container which enables users to have full control of their environment, and can be used to package the entire scientific workflows, software and libraries, and even data. Singularity is being commonly used by the ATLAS computing in the scenarios

of both grid sites and HPC sites. ATLAS distributes standard image files through CVMFS for sites to launch their Singularity containers.

Compared to virtualisation, the performance loss is reduced with Singularity. This was measured by comparing a few factors on the native host and the singularity container, such as IO performance, CPU performance and the average wall time of running real ATLAS simulation jobs. The tests were done on a host with 8GB RAM, and CentOS as its native OS, and CentOS was also used as the Singularity image.

As shown in Table 1, there is no performance loss in both IO and CPU Performance by using Singularity, and in terms of the average wall time of running ATLAS simulation jobs, using Singularity introduces 1.17% loss.

Table 1. Local performance test between native host and Singularity

	READ	WRITE	CPU	ATLAS JOB
Singularity	113 MB/s	116MB/s	26.97s	587s
Native Host	113 MB/s	114MB/s	27.74s	594s
Performance Loss	0	-1.7%	-2.7%	1.17%

4.2. Implementation of the Containerized ATLAS@home

As Singularity only runs on Linux OS systems, the targeted hosts for containerization are various Linux OS machines. As shown in Figure 3 depending on the target OS, there are two different approaches of OS provision, namely for Linux OS such as Scientific Linux and CentOS, the native OS is being used, and for all other Linux OS, Singularity with CentOS 6 is being used to run the ATLAS@home jobs.

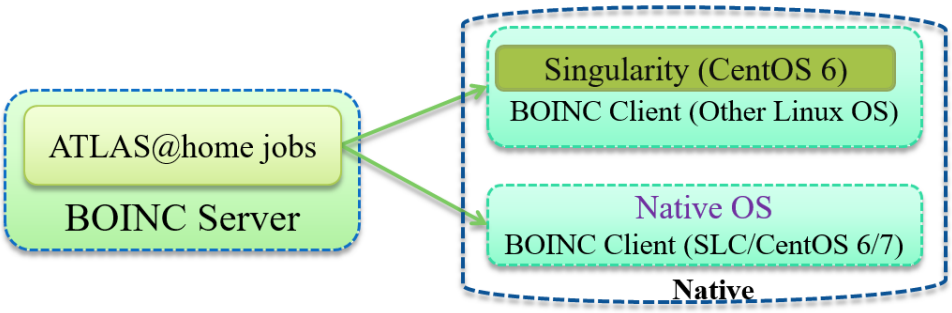

Figure 3. The containerization of ATLAS@home

A wrapper is being used to control the workflow of running ATLAS jobs on the BOINC client host. As shown in Figure 4, before launching the ATLAS job, the wrapper needs to do a few checks include the sanity of CVMFS, the target OS, and the sanity of Singularity, then decides how to provide the OS environment for running the job.

4.3. Advantages of containerization

There are some obvious advantages of using containerization comparing to virtualization in the context of ATLAS@home:

- There is no need to maintain the VM image on the server side.

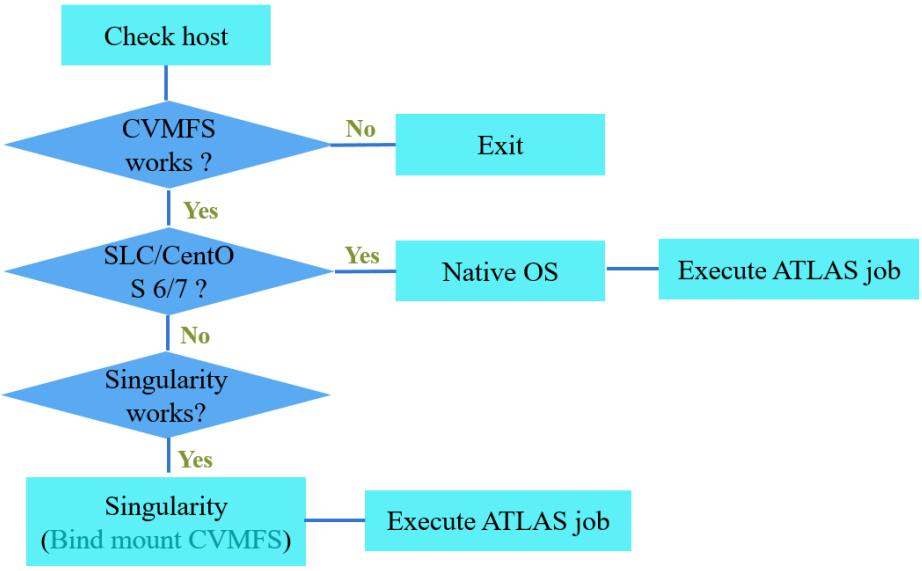

Figure 4. The workflow of containerized ATLAS@home

- It significantly saves the downloading time and the disk space on the client side for two reasons: 1) The software is cached in the CVMFS of the target host, and can be shared among different jobs on this host; 2) There is no need to clone and store an image for each job running on the host.
- It eliminates the overhead time for each job, which includes the time for clone the image and the creation and destroy of the virtual machine.
- There is no performance penalty in both IO and CPU.

4.4. Performance measurement

In order to have a precise comparison of the performance between virtualization and containerization, we run the same set of test jobs in both implementations of ATLAS@home. The test jobs we use are of the same size in terms of the total CPU time, and they are all multi-core jobs, so the more cores each job uses, the shorter the wall time it uses. We compare the CPU efficiency of both ATLAS and BOINC jobs. The ATLAS job CPU efficiency is defined by the ratio between the CPU time and wall time of the ATLAS job. And the wall time includes the IO time, CPU time, and extra time for downloading software and database files. In the context of containerization, both the IO and CPU time are improved, and the downloading time is completely eliminated.

As shown in Table 2, the ATLAS job CPU efficiency gets improved by 4% to 10% by using containerization depending on the number of cores used per job. The BOINC job CPU efficiency is defined by the ratio between the BOINC job CPU time and BOINC job wall time. And the BOINC job wall time includes the ATLAS job wall time and the VM overhead time. In the context of containerization, the VM overhead time is completely eliminated. As shown in Table 3, the BOINC job CPU efficiency get improved by 1% to 12% by using containerization depending on the number of cores used per job.

5. Use cases of the containerized ATLAS@home

Before using containerization, it is very tedious and inefficient to run the virtualized ATLAS@home on the cluster with physical nodes, as it requires the pre-installation of VirtualBox

Table 2. ATLAS job cpu efficiency: VM vs. Containerization

Core per job	CPU Eff. (VM)	CPU Eff. (Containerization)	CPU Eff. offset
2	86.0%	91.0%	5.0%
4	72.0%	82.0%	10.0%
8	71.0%	75.0%	4.0%

Table 3. BOINC job cpu efficiency: VM vs. Containerization

Core per job	CPU Eff. (VM)	CPU Eff. (Containerization)	CPU Eff. offset
2	90.4%	91.6%	1.2%
4	75.1%	81.3%	6.2%
8	66.3%	75.2%	11.9%

on the cluster nodes and also causes performance loss. And for the cluster with cloud nodes, it is impossible to run the virtualized version as another layer of virtualization can not be implemented on the cloud nodes.

The containerized ATLAS@home provides a lightweight solution to run the ATLAS@home jobs on clusters, including clusters with both physical nodes and cloud nodes. In the cluster, we use containerized ATLAS@home jobs to either backfill the busy ATLAS grid sites, or fulfill the idle computer nodes from CERN IT department.

5.1. Backfilling the grid sites

A study we conducted on the ATLAS grid sites shows that the average CPU utilization is below 70%.The concept of using ATLAS@home jobs to backfill grid sites implies running more than one process on each CPU core, and let the backfilling job have lower priority than the grid job, hence the backfilling job only uses the fragmental CPU cycles which cant be fully used by the grid jobs.

The backfilling model was tested on two ATLAS grid site, the BEIJING Tier 2 site and the TRIUMF Tier 1 site. The BEIJING Tier 2 site is a small scale site with 464 CPU cores, and the TRIUMF Tier 1 site is a medium scale site with 4816 CPU cores.

Table 4. CPU Utilization of TRIUMF site before and after backfilling

Before backfilling	CPU Util.	wall Util.
BOINC	0	0
Grid	0.69	0.88
Overall	0.69	0.88

After backfilling	CPU Util.	wall Util.
BOINC	0.27	0.91
Grid	0.65	0.97
Overall	0.92	1.88

As shown in both Table 4 and 5, for the TRIUMF site, the overall CPU utilization, which is the sum of both BOINC and Grid CPU utilization, is improved by 23% with adding the backfilling jobs, and for the BEIJING site, the average overall CPU utilization improvement over the period of 6 months is 26%, with the average CPU utilization of 90% and with the CPU utilization of 95% at its peak.

Table 5. CPU Utilization of BEIJING site during a busy and an idle week

Busy week	CPU Util.	wall Util.
BOINC	0.15	0.88
Grid	0.80	0.93
Overall	0.95	1.81

Idle week	CPU Util.	wall Util.
BOINC	0.42	0.88
Grid	0.48	0.62
Overalll	0.90	1.50

5.2. Fullfilling CERN cluster

Any large computer center such as the CERN IT department has a non negligible amount of idle computer nodes. These are either physical nodes near retiring, or the cloud nodes before they are being delivered to applications and users. And for the cloud nodes, they usually require running the CPU intensive benchmark before being delivered to some applications, so running the CPU intensive ATLAS@home jobs fit into such requirement. As the ATLAS@home client is provided by the system provision software puppet[18], the job running environment can be easily deployed and configured on all these nodes.

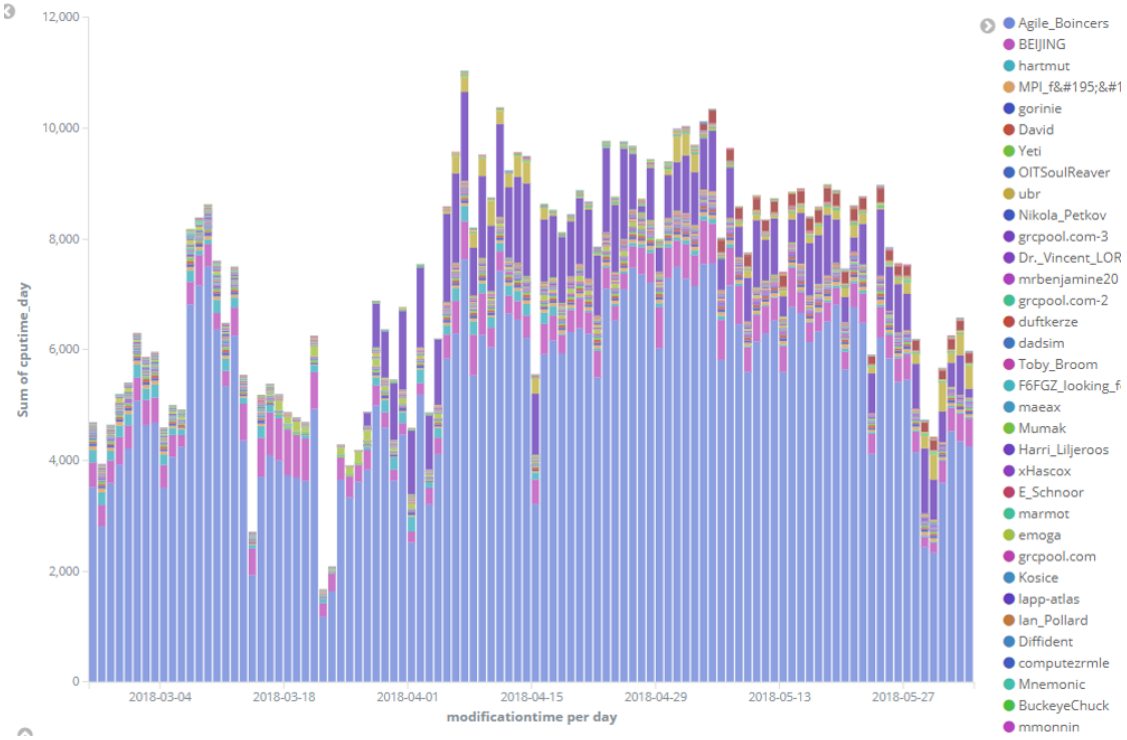

Figure 5. CPU time delivered by different ATLAS@home users

In Figure 5, Agile_Boincers stands for all the nodes from CERN IT, and since March 2018, CERN IT has been continuously proving CPU time, which on average is 6000 CPU days per day, and accounts for 60% of the CPU time for the ATLAS@home project. ATLAS@home jobs successfully utilize these idle CPU cycles which can not be used by other experiments otherwise.

6. Conclusion

Containerized ATLAS@home provides a lightweight solution comparing to the virtualization version. It reduces the overhead time caused by virtualization and improves the ATLAS job

CPU efficiency by 5% to 10%; it also reduces the usage of disk space and network bandwidth from the clients, and lightened the maintenance work on the server side; it makes it possible for the two usages of running ATLAS@home jobs on the clusters, which significantly increases the available resource for ATLAS@home. And as of now, over 85% of the computing power of ATLAS@home is delivered by its containerized version.

Acknowledgments
This project is supported by the National Natural Science Foundation of China grants "Research on fine grained Event Service for the BESIII offline software and its scheduling mechanism (No.11675201)" and "Research on BESIII offline software and scheduling mechanism on desktop grid (No.11405195)". We'd also like to thank all the volunteers of ATLAS@home who made this project possible.

References

[1] David Anderson, Boinc: A system for public-resource computing and storage, proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, 4-10 (2004)
[2] Myers, Daniel S and Bazinet, Adam L and Cummings, Michael P, Expanding the reach of Grid computing: combining Globus and BOINC-based systems, Grid computing for bioinformatics and computational biology, 71-84 (2007)
[3] Anderson, David P., et al. "SETI@ home: an experiment in public-resource computing." Communications of the ACM 45.11 (2002): 56-61.
[4] Abbott, B. P., et al. "Einstein@ Home search for periodic gravitational waves in early S5 LIGO data." Physical review d 80.4 (2009): 042003.
[5] Herr, Werner, D. I. Kaltchev, F. Schmidt, and E. McIntosh. Large Scale Beam-beam Simulations for the CERN LHC using distributed computing. No. LHC-PROJECT-Report-927. 2006.
[6] Buncic, Predrag, et al. "CernVMa virtual software appliance for LHC applications." Journal of Physics: Conference Series. Vol. 219. No. 4. IOP Publishing, 2010.
[7] Aguado Sanchez, Carlos, et al. "CVMFS-a file system for the CernVM virtual appliance." Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research. 2008.
[8] C Adam-Bourdarios, D Cameron,A Filipcic, E Lancon and Wenjing Wu for the ATLAS Collaboration, ATLAS@Home:Harnessing Volunteer Computing for HEP,21st International Conference on Computing in High Energy and Nuclear Physics, 664, 022009 2 (2015)
[9] Adam-Bourdarios, C., R. Bianchi, D. Cameron, A. Filipi, G. Isacchini, E. Lanon, Wenjing. Wu, and ATLAS Collaboration, Volunteer Computing Experience with ATLAS@Home, Journal of Physics: Conference Series, 898, 052009 5 (2017)
[10] Simone Campana, ATLAS Distributed Computing in LHC Run2, Journal, 664, 032004 3 (2015)
[11] Filipcic A, ATLAS Collaboration, ATLAS Distributed Computing Experience and Performance During the LHC Run-2, Journal of Physics: Conference Series, 895, 052015 5 (2017)
[12] Maeno T, PanDA: distributed production and distributed analysis system for ATLAS, Journal of Physics: Conference Series, 119, 062036 5 (2008)
[13] De, Kaushik and Klimentov, A and Maeno, T and Nilsson, P and Oleynik, D and Panitkin, S and Petrosyan, Artem and Schovancova, J and Vaniachine, A and Wenaus, T, The future of PanDA in ATLAS distributed computing, Journal of Physics: Conference Series, 664, 062035 6(2015)
[14] Rimoldi, A and Dell'Acqua, A and Gallas, M and Nairz, A and Boudreau, J and Tsulaia, V and Costanzo, D, The simulation for the ATLAS experiment: Present status and outlook, Nuclear Science Symposium Conference Record, 2004 IEEE, 3, 1886–1890 (2004)
[15] Yamamoto S, Shapiro M, on behalf of the ATLAS Collaboration, The simulation principle and performance of the ATLAS fast calorimeter simulation FastCaloSim, ATL-COM-PHYS-2010-838 (2010)
[16] Calafiura, Paolo and Leggett, Charles and Seuster, Rolf and Tsulaia, Vakhtang and Van Gemmeren, Peter, Running ATLAS workloads within massively parallel distributed applications using Athena Multi-Process framework (AthenaMP), Journal of Physics: Conference Series, 664, 072050 7 (2015)
[17] Kurtzer GM, Sochat V, Bauer MW (2017) Singularity: Scientific containers for mobility of compute. PLoS ONE 12(5): e0177459.
[18] Puppet: https://puppet.com/ [accessed 2018-11-12]