SoftwareX 14 (2021) 100679

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Adapting LIGO workflows to run in the Open Science Grid

Edgar Fajardo [a,*], Frank Wuerthwein [a], Brian Bockelman [b], Miron Livny [b], Greg Thain [e], James Alexander Clark [c], Peter Couvares [d], Josh Willis [d]

[a] University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093, USA
[b] Mortgridge Institute, 330 N Orchard St, Madison, WI 53715, USA
[c] School of Physics, Georgia Institute of Technology, Atlanta, GA 30332, USA
[d] LIGO, California Institute of Technology, Pasadena, CA 91125, USA
[e] University of Madison Wisconsin, USA

ARTICLE INFO

Article history:
Received 30 March 2020
Received in revised form 7 December 2020
Accepted 9 February 2021

Keywords:
LIGO
OSG
Grid
DHTC

ABSTRACT

During the first observation run the LIGO collaboration needed to offload some of its most, intense CPU workflows from its dedicated computing sites to opportunistic resources. Open Science Grid enabled LIGO to run PyCbC, RIFT and Bayeswave workflows to seamlessly run in a combination of owned and opportunistic resources. One of the challenges is enabling the workflows to use several heterogeneous resources in a coordinated and effective way.

Code metadata

Current code version	v3.6.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_104
Legal Code License	BSD License
Code versioning system used	git
Software code languages, tools, and services used	Python2
Compilation requirements, operating environments & dependencies	RedHat6 or RedHat7
If available Link to developer documentation/manual	glideinwms.fnal.gov
Support email for questions	glideinwms-support@fnal.gov

1. Motivation and significance

In order to reach the scientific and discovery goals of the LIGO Collaboration several pipelines of CPU intensive workflows are run. During certain times the pipelines compete for computing resources at the LIGO-owned computing laboratories. There is opportunity to migrate some of these pipelines from dedicated resources to a combination of owned and opportunistic resources.

The LIGO collaboration worked with the Open Science Grid (OSG) [1] to enable PyCBC [2], RIFT [3] and Bayeswave [4,5] workflows to run on the grid. These workflows share the same structure. They are made of several thousand individual tasks or jobs, which require no communication between them. The task runtime is in the order of hours. This intrinsically parallel formulation made them a candidate for the Distributed High Throughput Computing (DHTC) model in OSG. The distributed model generates a data distribution challenge: the LIGO experiment data is produced at the interferometer locations and then stored at a few computing centers. The problem lies then in distributing the data to all the participating computing centers around the world for the workflows to consume. The OSG solution to this data delivery problem is through Stashcache [6].

In a nutshell, Stashcache is a file block caching technology based on XRootD [7] that can deliver on-demand high volumes of data to the jobs. These jobs use GeoIP to retrieve data of the nearest cache from a set of caches conveniently located around the world.

[*] Corresponding author.
E-mail address: emfajard@ucsd.edu (Edgar Fajardo).

https://doi.org/10.1016/j.softx.2021.100679

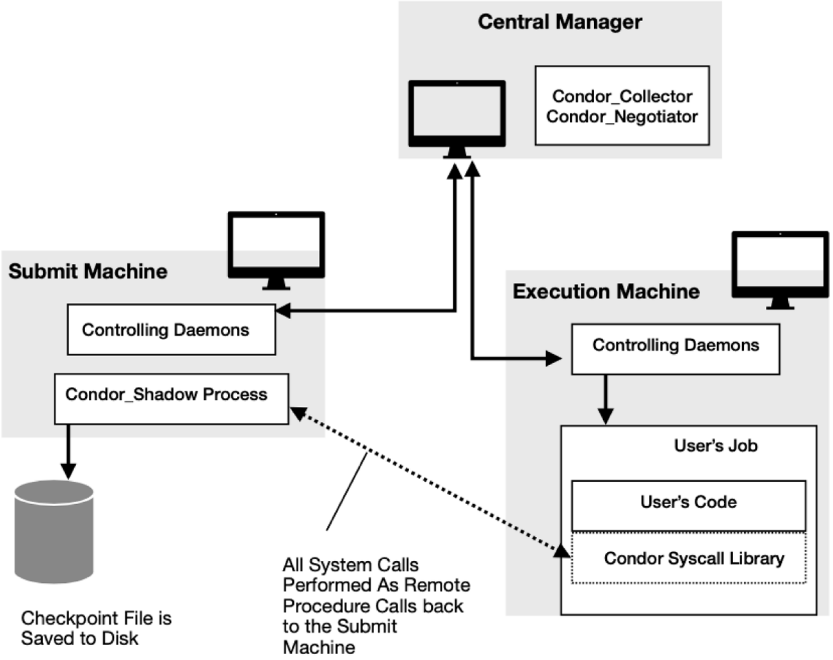

Fig. 1. Description of HTCondor Architecture [8].

2. Software description

The DHTC model in OSG is powered by the Glidein Workload Management System (GlideinWMS) system [9,10]. It is a pilot model system in which resources at heterogeneous sites are gathered and presented to the scientist as one single homogeneous pool of resources. GlideinWMS is based on the HTCondor [11] batch system and it is designed to create a changing pool of resources based on demand.

2.1. Software architecture

2.1.1. HTCondor architecture

The HTCondor batch system architecture is made of three components: scheduler (schedd), central manager and computing machines (see Fig. 1). The scheduler is the multi user client facing part of the architecture and takes care of submitting the jobs, maintaining the job queue, and transferring the input and output files needed for the job from itself to the compute nodes. In a usual set up (including the one used by LIGO collaboration) several schedd are deployed to serve a single pool for scalability and fault tolerance reasons.

The second component of the HTCondor architecture is the central manager. The central manager is made of two daemons: the collector and the negotiator. The collector daemon tracks all the information on the pool. This includes which computing machines are busy/idle and which users have idle jobs in the queues (schedds). The negotiator uses the information from the collector to decide which job matches which resources and among users who have the best priority to use those resources.

The final part of the HTCondor Architecture is the daemon that runs in the computing nodes called StartD. The StartD daemon runs on every compute node in an HTCondor pool and informs the collector of the machine usage, when idle, when busy. Once it is assigned a job it contacts the corresponding schedd to start the job.

2.1.2. GlideinWMS architecture

The GlideinWMS pilot system builds on top of the HTCondor architecture (see Fig. 2) and introduces two more pieces: factories and frontends. The latter is a set of Python daemons that continuously query the submit hosts (step 2 in Fig. 2) in a single HTCondor pool and calculate the demand for resources. Based on this demand, the frontend asks the factory to submit pilots on its behalf to a grid site (step 5 in Fig. 2).

The factory submits the pilots based on the pressure requested by the frontend (step 6 in Fig. 2). In order for the factory to submit to a site the site must have a Compute Element (CE) which "translates" grid submissions into local batch system submissions.

The frontend securely sends its credentials to the factory to be presented to the CE on the frontend's behalf. Once a pilot is submitted and is running at a site batch system it contacts a web server running in the factory to download configurations and the HTCondor binaries, then the pilot downloads frontend specific configurations from the frontend's web server. Finally it starts the HTCondor Daemons (as an unprivileged user) and connects back to the pool collector (step 7 in Fig. 2). From this point on a pilot looks just like any other resource in an HTCondor pool. The StartD advertises its capabilities to the Collector (step 8 in Fig. 2) and the schedd starts a job from its queue into the pilot (step 9 in Fig. 2.

The recommend usage is that each frontend is managed by a scientific community which we will call from now on a Virtual Organization (VO). The factory(s) are centrally operated services that can serve multiple organizations and hence reduce operational costs [12].

2.2. Software functionalities

A GlideinWMS pool can gather resources from several heterogeneous grid sites. The type of CE varies the most among sites. The Factory makes extensive use of Condor-G [11] capabilities to submit to CREAM [14], ARC-CE [15] and HTCondor-CE [16] as well as to several commercial cloud providers like Amazon Web Services and Google Cloud.

The strength of this architecture lies in both the breadth and scale of the resources that can be gathered. The scalability of a

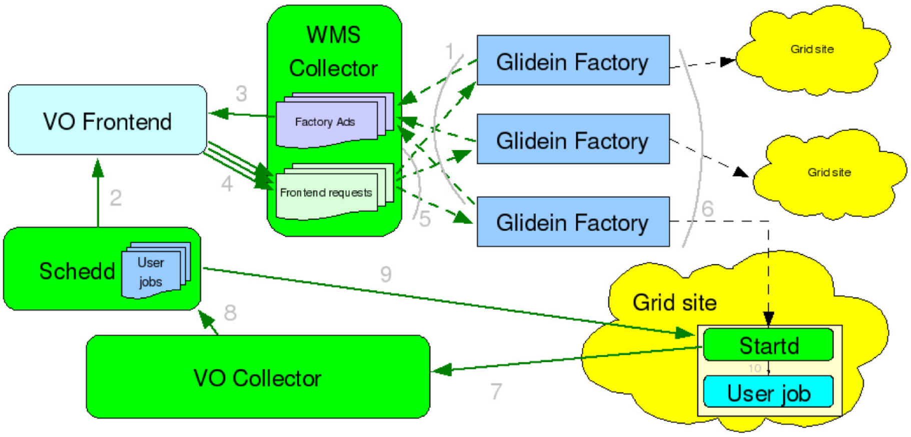

Fig. 2. Description of GlideinWMS architecture [13].

GlideinWMS pools has been measured to exceed more than 200k running jobs [17]. GlideinWMS handles mixed sets of GPU and CPU workloads efficiently among heterogeneous resources [18]. Moreover the schedds can move several GigaBytes/sec of traffic to and from the compute nodes [19]. Finally GlideinWMS, and HTCondor incorporate the ability for each individual user to run tasks in the container environment of their choice via integration with Singularity [20,21].

3. Illustrative examples

These capabilities are exercised by LIGO. It uses VIRGO resources on WLCG [22] thanks to the factory's ability to submit to different types of CEs. Moreover it uses several HPC sites, like Comet [23] and BlueWaters [24], in addition to traditional resources in OSG. Fig. 3 shows how LIGO can consume resources from several sites that are very heterogeneous and need not communicate among themselves to collaborate.

Since the RIFT pipeline was adapted to run in a distributed environment GlideinWMS was able to acquire over 240k GPU hours in the last year among several sites (see Fig. 4).

4. Impact

The advances of gravitational wave science and, multi-messenger astrophysics over the past four years have been absolutely contingent on the efficient and intelligent application of sophisticated, and computationally demanding, data analysis techniques. Some of the greatest drivers of this demand come from efforts to characterize gravitational wave signals and estimate the parameters of the progenitor systems. To meet the computational load these techniques demand, the gravitational wave community has significantly diversified its resource usage beyond dedicated sites to include more allocated and opportunistic resources. The two parameter estimation pipelines which have spearheaded the usage of opportunistic resources are BayesWave [4] and RIFT [3].

These pipelines are routinely run in dedicated LIGO owned HTCondor computing clusters. In these clusters the run time environment (Operative System, libraries, etc.) of a running job was highly controlled and uniform. Moreover the code was written to expect the input frame files to be in POSIX mounts at specific locations. Singularity provided the functionality to run a job inside a specific container, hence the solution to curate a distributed environment became to develop specific containers for each application. In its turn these containers would be distributed to all sites LIGO-owned or not using CVMFS [21]. Moreover CVMFS is used in tandem with stashcache to provide a single POSIX-like mount at all computing sites to deliver the input data for the running jobs hence solving the data delivery problem as well.

BayesWave is an algorithm designed for robust signal classification and waveform reconstruction. The ultimate goal here is to evaluate the respective Bayesian posterior probabilities that a given stretch of data contain a gravitational wave signal versus a transient "glitch" of terrestrial origin. Posterior probability density functions on the parameters of a wavelet decomposition of the data are then used to reconstruct, or de-noise, the underlying signal.

Under the hood, BayesWave utilizes a reversible-jump Markov chain Monte Carlo algorithm to explore a variable dimensional parameter space of gravitational wave signals, instrumental glitches and Gaussian noise. Even when using this efficient, stochastic sampling algorithm, the confident BayesWave classification of a single, sub-second duration putative gravitational wave candidate as astrophysical or terrestrial in origin, and the reconstruction of the underlying waveform, can take up to 48 h of computation on a single core.

Furthermore, full characterization of a putative signal requires large scale Monte-Carlo simulations: on the one hand, the statistical significance of a detection claim is ultimately determined by running the algorithm repeatedly on data which is believed to contain only noise; on the other hand, comparisons of waveform reconstructions with other analyses are quantified by running the BayesWave algorithm on thousands of simulations of the gravitational wave signal reported by those other analyses.

The properties of well-understood gravitational wave sources involving the coalescence of black holes and neutron stars are determined by comparing waveforms predicted by analytical or numerical models with data from the network of gravitational wave detectors. The ultimate goal here is to generate a posterior probability density function from which we may select point estimates and credible intervals for the progenitor system's parameters of interest, such as the mass, spin configuration, and so on.

While this is straightforward and well-posed in principle, a number of factors result in computationally costly analyses: a large and uncertain parameter space; evaluation of complex waveform models millions of times per source; and an often richly-structured, multi-modal likelihood function from which it is difficult to efficiently sample. Even using simplified waveform models for binary neutron star mergers, these analyses can take hours and even weeks, depending on the extent of the parameter space and complexity of the model. Sophisticated models for binary black hole mergers which include more exotic phenomenology may even take months to accurately determine

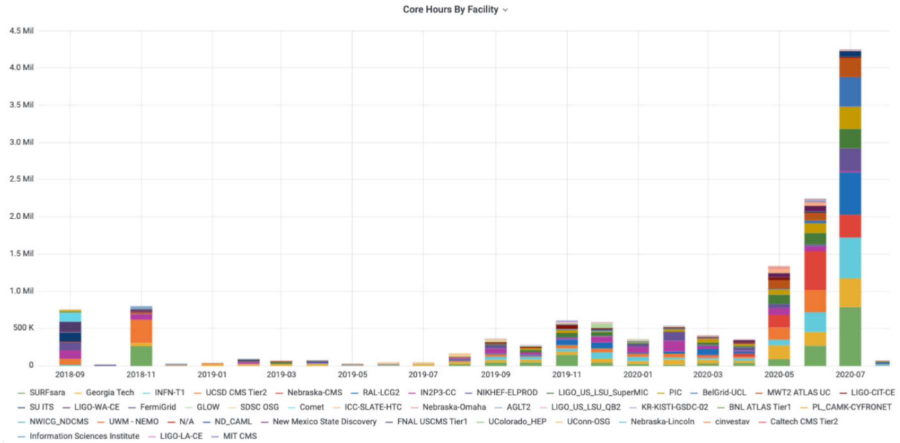

Fig. 3. Site usage distribution of LIGO on OSG for the last two years. Vertical axis is CPU core hours per month, peaking at 4.5 million hours per month, or an average of roughly 6000 cores.

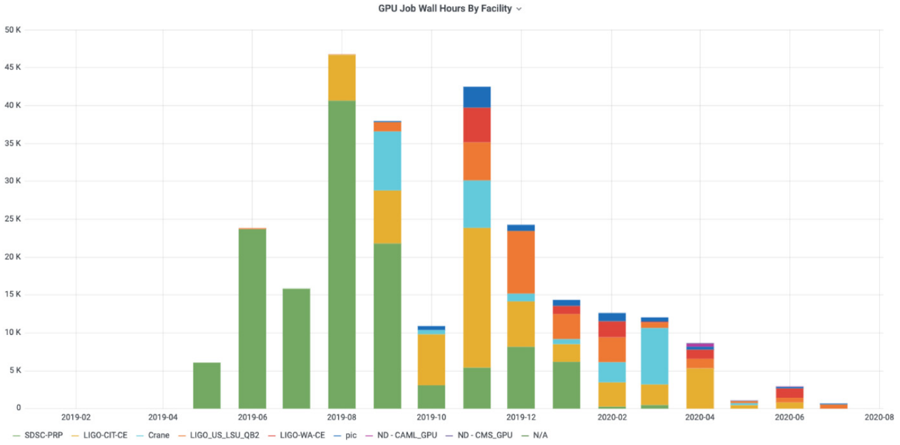

Fig. 4. Site usage distribution of LIGO GPU usage for the last year. Vertical axis is GPU hours per month peaking at 45,000, or an average of roughly 60 GPUs.

the parameters of the system with confidence in the convergence of the results. As with BayesWave, the cost is further compounded by the need to fully explore model systematics through large scale Monte Carlo simulations.

Rapid parameter Inference on gravitational wave sources via Iterative Fitting, i.e., RIFT, mitigates the costs inherent in sampling efficiency and waveform generation via a highly parallelizable grid-based algorithm. Rather than sampling directly from the joint posterior probability distribution on all of the system's parameters, RIFT constructs a grid over the intrinsic parameters (i.e., those which determine the system dynamics; typically the parameters of direct astrophysical interest) and employs Monte Carlo integration to marginalize over the extrinsic parameters (e.g., the spacetime coordinates for the event and its orientation with respect to Earth, which may be regarded as 'nuisance' parameters). The marginalized likelihood of the intrinsic parameters is efficiently evaluated through generation of an initial cache of all possible model values at each grid point. The grid of marginalized likelihood values then provides the seeds for Gaussian process interpolation to approximate the full, continuous likelihood function. Samples from the target posterior distribution are then obtained via adaptive Monte Carlo techniques.

The RIFT code has been significantly accelerated by leveraging GPUs. After an initial CPU-bound calculation to evaluate inner products of the waveform models with the data, the matrix operations which yield the likelihood from these inner products and the marginalization over extrinsic parameters are performed on a GPU. A typical single-threaded CPU-bound job running on an Intel(R) Xeon(R) Silver 4116 completes with a wall time of about 7 h 43 m. When the likelihood evaluation for the same job is performed on an Nvidia Quadro P2000, using the same CPU, the time to completion is just over 23 m, a factor $\sim$ 20× improvement. In terms of scientific analysis, a RIFT-based characterization of the GW170817 binary neutron star event requires 14 core-days, while a comparable analysis using more traditional sampling techniques requires 228 core-days. The RIFT code is written in python, using CUPY to implement the CUDA-based GPU analysis.

Finally, to ensure convergence to a robust result, this procedure is applied iteratively, with the posterior samples from each stage providing a new grid for the subsequent stage, resulting in an adaptive grid refinement which accurately captures the shape of the likelihood function. This algorithm lends itself naturally to a high-throughput computing approach, where each individual RIFT job independently explores a subset of the parameter space.

5. Conclusions

The functionalities of GlideinWMS and HTCondor have been sufficient to let the computing infrastructure help LIGO meet its scientific goals. PyCBC, BayesWaves and RIFT workflows have been successfully adapted to run in a Distributed High Throughput Computing model. This has led to an increase in the breadth and depth of the physics questions that can be answered by being able to consume hundreds of thousands of CPU and GPU hours in a worldwide distributed way. The near future still brings several short term challenges such as moving the infrastructure authentication from using X509 certificates to Scitokens and placing submit hosts at several institutes outside the US and closer integration of the data delivery systems and the submission infrastructure.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the funding agencies for this work, in particular the National Science Foundation, USA through the following grants: OAC-1541349, OAC-1826967, OAC-1841530, MPS-1148698.

References

[1] Pordes R, Petravick D, Kramer B, Olson D, Livny M, Roy A, Avery P, Blackburn K, Wenaus T, Würthwein F, Foster I, Gardner R, Wilde M, Blatecky A, McGee J, Quick R. The open science grid. J Phys Conf Ser 2007;78(1):012057, http://stacks.iop.org/1742-6596/78/i=1/a=012057.

[2] Nitz A, Harry I, Brown D, Biwer CM, Willis J, Canton TD, Capano C, Pekowsky L, Dent T, Williamson AR, De S, Davies GS, Cabero M, Machenschalk B, Reyes S, Kumar P, Macleod D, Pannarale F, Massinger T, dfinstad, Tápai M, Fairhurst S, Khan S, Kumar S, Singer L, Nielsen A, shasvath, idorrington92, Lenon A, Gabbard H. Gwastro/pycbc: Pycbc release v1.15.5. 2020, Zenodo, URL https://doi.org/10.5281/zenodo.3697109.

[3] Lange J, O'Shaughnessy R, Rizzo M. Rapid and accurate parameter inference for coalescing, precessing compact binaries. 2018, arXiv:1805.10457.

[4] Cornish NJ, Littenberg TB. BayesWave: Bayesian inference for gravitational wave bursts and instrument glitches. Classical Quantum Gravity 2015;32:135012. http://dx.doi.org/10.1088/0264-9381/32/13/135012, arXiv:1410.3835.

[5] Littenberg TB, Cornish NJ. Bayesian Inference for spectral estimation of gravitational wave detector noise. Phys Rev D 2015;91:084034. http://dx.doi.org/10.1103/PhysRevD.91.084034, https://link.aps.org/doi/10.1103/PhysRevD.91.084034.

[6] Weitzel D, Zvada M, Vukotic I, Gardner R, Bockelman B, Rynge M, Hernandez E, Lin B, Selmeci M. StashCache: A distributed caching federation for the open science grid. In: PEARC '19: Proceedings of the practice and experience in advanced research computing on rise of the machines (learning). 2019, p. 1–7. http://dx.doi.org/10.1145/3332186.3332212.

[7] Dorigo A, Elmer P, Furano F, Hanushevsky A. XROOTD-A highly scalable architecture for data access. WSEAS Trans Comput 2005;1(4.3).

[8] HTCondor Manual, http://web.archive.org/web/20200803183847/, https://research.cs.wisc.edu/htcondor/manual/v7.6/3_1Introduction.html [Accessed: 03 August 2020].

[9] Sfiligoi I, Bradley DC, Holzman B, Mhashilkar P, Padhi S, Wurthwein F. The pilot way to grid resources using glideinWMS. In: 2009 WRI world congress on computer science and information engineering, vol. 2, 2009. p. 428–32.

[10] Mhashilkar P, Mambelli M, Sfiligoi I, Holzman B, Larson K, Dost J, ddbox, Mascheroni M, Weigand J, Lobato L, Hein T, Lin B, Fajardo E, Weitzel D, Rynge M, Bockelman B, Selmeci M. Glideinwms/glideinwms: v3.4. 2018, https://doi.org/10.5281/zenodo.1309679.

[11] H T Condor Team. Htcondor 8.6.12. 2018, https://doi.org/10.5281/zenodo.1324567.

[12] Sfiligoi I, Dost JM, Zvada M, Butenas I, Holzman B, Wuerthwein F, Kreuzer P, Teige SW, Quick R, Hernández JM, Flix J. The benefits and challenges of sharing glidein factory operations across nine time zones between OSG and CMS. J Phys Conf Ser 2012;396(3):032103. http://dx.doi.org/10.1088/1742-6596/396/3/032103.

[13] GlideinWMS Official documentation [Accessed: 03 August 2020] http://web.archive.org/web/20200803183025/, http://glideinwms.fnal.gov/doc.prd/frontend/index.html.

[14] Lorenzo PM, Santinelli R, Sciaba A, Thackray N, Shiers J, Renshall H, Sgaravatto M, Padhi S. The CREAM-CE: First experiences, results and requirements of the four LHC experiments. J Phys Conf Ser 2010;219(6):062022. http://dx.doi.org/10.1088/1742-6596/219/6/062022.

[15] Ellert M, Konstantinov A, Kónya B, Smirnova O, Wäänänen A. The nordugrid project: using globus toolkit for building grid infrastructure. Nucl Instrum Methods Phys Res A 2003;502(2):407–10. http://dx.doi.org/10.1016/S0168-9002(03)00453-4, http://www.sciencedirect.com/science/article/pii/S0168900203004534, Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research.

[16] Bockelman B, Cartwright T, Frey J, Fajardo EM, Lin B, Selmeci M, Tannenbaum T, Zvada M. Commissioning the HTCondor-CE for the open science grid. J Phys Conf Ser 2015;664(6):062003. http://dx.doi.org/10.1088/1742-6596/664/6/062003.

[17] Fajardo EM, Dost JM, Holzman B, Tannenbaum T, Letts J, Tiradani A, Bockelman B, Frey J, Mason D. How much higher can HTCondor fly? 2015;664(6):062014. http://dx.doi.org/10.1088/1742-6596/664/6/062014.

[18] Fajardo E, Rynge M, Weitzel D, Merino G, Schultz D, Brik V, Skarlupka H. OSG And GPUs: A tale of two use cases. EPJ Web Conf 2019;214:03034, https://doi.org/10.1051/epjconf/201921403034.

[19] Fajardo E, Würthwein F, Jones R, Philpott S, Strosahl K. Limits of the htcondor transfer system. EPJ Web Conf 2019;214:03008. http://dx.doi.org/10.1051/epjconf/201921403008.

[20] Kurtzer GM. Singularity 2.5.2 - linux application and environment containers for science. 2018, http://dx.doi.org/10.5281/zenodo.1308868, https://doi.org/10.5281/zenodo.1308868.

[21] Rynge M, Bockelman BP, Weitzel D, jthiltges, Jones R, Downes T, Fajardo E, Blyth D, Skarlupka H, Riedel B, Diogo V, Bustamante J, Brown D, Desinghu B, Lukas, Kreczko L, drtmfigy, brichards64, Wasserman A, Devisetty UK, Roberts S, Mauri, Bryant L, Stark G. Opensciencegrid/cvmfs-singularity-sync: Singularity-sync first release. 2018, http://dx.doi.org/10.5281/zenodo.1469012, https://doi.org/10.5281/zenodo.1469012.

[22] Bird I. Computing for the large hadron collider. Annu Rev Nucl Part Sci 2011;61:99–118. http://dx.doi.org/10.1146/annurev-nucl-102010-130059, http://www.annualreviews.org/doi/abs/10.1146/annurev-nucl-102010-130059.

[23] Strande SM, Cai H, Cooper T, Flammer K, Irving C, von Laszewski G, Majumdar A, Mishin D, Papadopoulos P, Pfeiffer W, et al. Comet: Tales from the long tail: Two years in and 10,000 users later. In: Proceedings of the practice and experience in advanced research computing 2017 on sustainability, success and impact. PEARC17, New York, NY, USA: Association for Computing Machinery; 2017, http://dx.doi.org/10.1145/3093338.3093383, https://doi.org/10.1145/3093338.3093383.

[24] Huerta E, Haas R, Fajardo E, Katz DS, Anderson S, Couvares P, Willis J, Bouvet T, Enos J, Kramer WTC, Leong HW, Wheeler D. BOSS-LDG: A novel computational framework that brings together blue waters, open science grid, shifter and the LIGO data grid to accelerate gravitational wave discovery. In: 2017 IEEE 13th international conference on e-science (E-Science). 2017, p. 335–44. http://dx.doi.org/10.1109/eScience.2017.47.