EDITORS: Lorena A. Barba, labarba@gwu.edu
George K. Thiruvathukal, gkt@cs.luc.edu

SPECIAL TRACK: REPRODUCIBLE RESEARCH

Reproducing GW150914: The First Observation of Gravitational Waves From a Binary Black Hole Merger

Duncan A. Brown, *Syracuse University, Syracuse, NY, 13244, USA*
Karan Vahi, *University of Southern California, Los Angeles, CA, 90007, USA*
Michela Taufer, *University of Tennessee Knoxville, Knoxville, TN, 37996, USA*
Von Welch, *Indiana University, Bloomington, IN, 47405, USA*
Ewa Deelman, *University of Southern California, Los Angeles, CA, 90007, USA*

In 2016, LIGO and Virgo announced the first observation of gravitational waves from a binary black hole merger, known as GW150914. To establish the confidence of this detection, large-scale scientific workflows were used to measure the event's statistical significance. They used code written by the LIGO/Virgo and were executed on the LIGO Data Grid. The codes are publicly available, but there has not yet been an attempt to directly reproduce the results, although several analyses have replicated the analysis, confirming the detection. We attempt to reproduce the result presented in the GW150914 discovery paper using publicly available code on the Open Science Grid. We show that we can reproduce the main result but we cannot exactly reproduce the LIGO analysis as the original dataset used is not public. We discuss the challenges we encountered and make recommendations for scientists who wish to make their work reproducible.

For the scientific community to build on previous results, it must trust that these results are not accidental or transient, but rather that they can be reproduced to an acceptably high degree of similarity by subsequent analyses. This notion of reproducibility is magnified both in importance and challenges in the context of computational science workflows.[1] An increasingly large fraction of scientific results depend on computational elements, which in turn creates reproducibility challenges associated with the implementation of these computational elements. Being able to reason about the validity of published scientific results and reuse them in derivative works becomes an extremely challenging task. Publishers have made great strides in including relevant artifacts along with the manuscripts. However, data, methods, and results are still hard to find and harder still to reproduce (recreating the results from the original author's data and code), to replicate (arriving at the same conclusion from a study using new data or different methods), and to reuse in derivative works (using code or data from a previous study in a new analysis).[2]

Our work focuses on reproducing the computational analysis used to establish the significance of the first detection of gravitational waves created colliding binary black holes and observed by the Advanced Laser Interferometer Gravitational-wave Observatory (LIGO).[3] As part of its commitment to Open Data, LIGO made the data and scientific codes from its first observing run available to the scientific community. Previous analyses have *replicated* the results of the GW150914 discovery.[4,5] In these

1521-9615 © 2021 IEEE
Digital Object Identifier 10.1109/MCSE.2021.3059232
Date of publication 12 February 2021; date of current version 25 March 2021.

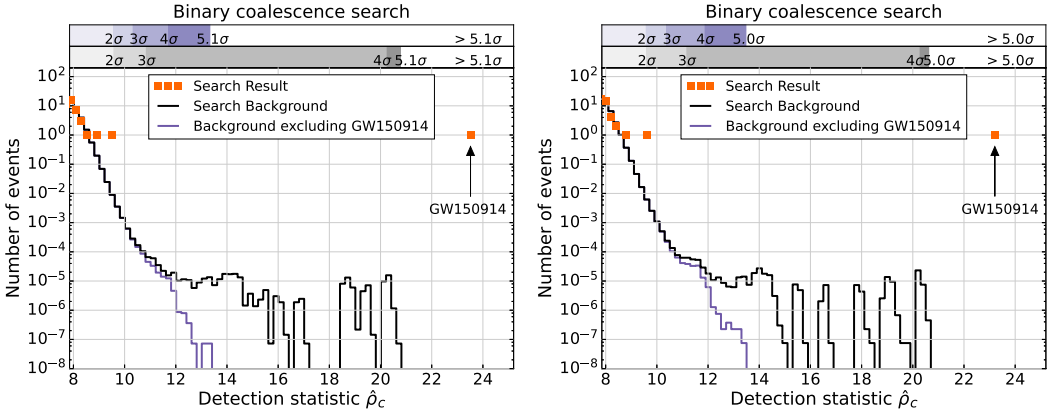

FIGURE 1. Results from the binary coalescence search presented in the GW150914 discovery paper from the paper by Abbott *et al.*[3] with permission (left) and our attempt to reproduce these results (right). These histograms show the number of candidate events (orange markers) and the mean number of background events (black lines) as a function of the search detection statistic and with a bin width of 0.2. The scales on the top give the significance of an event in Gaussian standard deviations based on the corresponding noise background. We were able to reproduce the search result for GW150914, but we were unable to exactly reproduce the search background. The differences between the two figures is likely due to differences in the gravitational-wave stain data used, as described in the text.

analyses, the data from LIGO's first observing run was reanalyzed either by independent teams of scientists with different codes, with different data, or by using different workflows to those used in the original GW150914 discovery.

In a previous work, we have performed a *posthoc* comparison of these results using the published papers and the PRIMAD reproducibility formalism.[6] Here, we attempt to reproduce *ab-initio* the original LIGO analysis used in the GW150914 discovery paper using public information. Specifically, we attempt to reproduce the results of the PyCBC search for gravitational waves[7,8] shown in Figure 4 of Abbott *et al.*[3]

Our effort is not completely separate from the original analysis, as one coauthor of this article was a member of the team involved in running the original LIGO analysis. However, our aim was to automate the production of the result in a way that other coauthors of this article who were not members of the LIGO or Virgo collaborations, as well as other scientists, could reproduce the result.

The original analysis workflows were executed on the LIGO Data Grid, a collection of computational resources that are not available to the wider community. Since non-LIGO scientists do not have access to these systems, we execute the analysis on the Open Science Grid (OSG)[9] and rely on a cyberinfrastructure software stack that has latest stable releases of key software packages such as HTCondor,[9] Pegasus,[10] and the CERN Virtual Machine Filesystem (CVMFS).[11] We have created a script that automates the setup and deployment of the LIGO workflows on a typical local compute cluster and from there Pegasus manages their execution on OSG.

Our main goal was to reproduce the results of the PyCBC search shown in Figure 4 of Abbott *et al.*,[3] shown on the left-hand side of our Figure 1, since this is the result used to make the statement that the signal is detected with a "significance greater than 5.1 σ" in the "Abstract" section of this article. Our reproduction of this plot, shown on the right-hand side of Figure 1 shows that we can reproduce the search result, but there are small, noticeable differences in the search background (explained later in this article). Based on the LIGO documentation, we believe that these differences are because the data used in the original analysis were different from that released by the Gravitational Wave Open Science Center (GWOSC)[12] and used in our analysis. Unfortunately, the original dataset is not public and so we are unable to confirm this hypothesis. However, we consider our ability to rerun a scientific workflow last executed in 2015 and largely reproduce the results to be a successful demonstration of reproducibility.

This article is structured as follows. First, we provide background on the first gravitational-wave discovery. We then describe our recent efforts on *ab-inito* analysis to reproduce the GW150914 result, followed by challenges we encountered. In the results section, we explain any difference observed in our reproduction of the result published by LIGO. We

perform an analysis of the workflow run-time provenance data and the compute resources required to execute the workflow. We conclude with recommendations for others who wish to reproduce the GW150914 result.

DISCOVERY OF GW150914

Gravitational-wave astronomy is an interesting case study for robust science because it has three main science phases: low-latency data analysis, offline analysis, and public and educational dissemination of results. The low-latency analysis processes instrumental data in near real time to identify astrophysical signals. Alerts are disseminated to the community to identify electromagnetic or neutrino counterparts to the gravitational-wave signal. Offline analyses validate the low-latency detections, identify signals missed in low-latency, and provide determination of source properties. When a detection is published, the data are released to the scientific community and the public. Since the analysis codes are also released, it should be possible for people outside the LIGO Scientific Collaboration and Virgo to reproduce the published results.

Our attempt to reproduce the first detection of gravitational waves from binary black holes, known as GW150914, starts from the data released by the GWOSC.[12] GW150914 was first detected by a low-latency search for gravitational-wave bursts that identifies interesting candidates but does not provide the final statistical significance of detected events. To establish the significance of events, data from the LIGO detectors are subsequently analyzed by scientific workflows that use longer stretches of data to provide a measure of the noise background in the detectors and use this to measure the significance of candidate events. Results from two offline analyses were presented in the GW150914 discovery paper: one that used a search technique that did not make assumptions about the shape of the gravitational waveform[3] and one using matched filtering (comparing the data to a known waveform) to search for the signals from merging black holes,[7] known as PyCBC. Here, we focus on reproducing the results of the PyCBC binary black hole search.

The PyCBC search uses matched filtering to compare the LIGO data with a bank of template binary black hole waveforms that model the target sources. If the noise in the LIGO detectors was stationary and Gaussian, the estimation of the statistical significance of candidate events that crossed a signal-to-noise ratio threshold would be straightforward. However, the LIGO detector data contain non-Gaussian noise transients and periods of nonstationary noise. As a result, additional signal-processing techniques are applied to the data that suppress non-Gaussian noise events. The search algorithms require that the same signal is seen in the detectors; the same waveform must be present both detectors and the signal's time of arrival must be consistent with the gravitational-wave travel time between the observatories. The map between the detection statistic (weighted signal-to-noise ratio) and the statistical significance of an event must be empirically measured by the workflow. This is done by time-shifting the data between the detectors and repeating the coincidence analysis many times. The most computationally intensive part of the PyCBC workflow are the matched filtering and the calculation of the detection statistics. Performing the coincidence and the time-shift analysis can require a large amount of memory to process the candidate events. Once these steps are complete, the workflow produces a measurement of the statistical significance of candidates. A separate script run after the workflow completes produces a histogram that compares candidate events to the noise background.

REPRODUCING THE ANALYSIS

Our work is the first attempt to *reproduce* the original LIGO analysis. Previous analyses, for example, 1-OGC result,[5] provide an example of *replication* in gravitational-wave science. In the 1-OGC analysis, a different team with different experimental setup recovered the discovery of GW150914. Here, "different experimental setup" means a modified data-analysis pipeline with a different configuration to that used in the original analysis. The 1-OGC result independently confirmed that GW150914 was a high significance discovery, but the event was recovered with slightly different parameters to the original discovery; these parameter differences can be explained by differences between the different algorithms used.

OUR WORK IS THE FIRST ATTEMPT TO REPRODUCE THE ORIGINAL LIGO ANALYSIS.

Here, we provide an example of *reproducibility* of the measurement of the statistical significance of GW150914 shown in Figure 4 of Abbott *et al.*[3] The paper by Abbott *et al.*[3] by its nature as a brief letter does not provide sufficient information to reproduce the result.

The paper by Abbott *et al.*[8] provides additional description of the analysis and the codes[a] and configuration files[b] are publicly released on GitHub. Although the codes and configuration are public, the LIGO/Virgo collaboration does not provide full instructions for running the workflow and reproducing the analysis. Our work provides a fully reproducible process.

Not all of the information needed to reproduce the GW150914 workflow was available in the public release accompanying the publications. The lack of a single, public repository of this knowledge is the most significant challenge for a group outside the LIGO and Virgo collaborations to reproducing the GW150914 result. However, one of the coauthors of our work was a member of the team who performed the original analysis. They were able to review their unpublished notes, which allowed us to successfully reproduce the LIGO analysis. To ensure that scientists who were not involved in the original analysis could reproduce the results, we created scripts that were run independently by another author of this article not involved in the original analysis. These scripts were created in a peer-programming style, which started from the original scripts used to run the LIGO workflow and created the result plot. We iteratively fixed problems encountered when trying to run the analysis using information entirely in the public domain, filling in missing public information with the original analysis notes where necessary.

PyCBC is a gravitational-wave data-analysis toolkit written primarily in Python with C extensions for numerically intensive computations. Rerunning old versions of interpreted Python code can be challenging if the underlying software stack has changed since the code was originally executed. Fortunately, LIGO packaged the PyCBC codes used in the original analysis as PyInstaller bundles. These bundles package the Python code with a Python interpreter and the Python library dependencies allowing us to run the original codes without needing to recreate the entire software stack. Our final version of the workflow execution script is provided in a data release that accompanies this article[c] and the GitHub commit history[d] documents the iterative process of addressing the issues encountered, which we describe below.

Software Versions

Software provenance is critical to the reproducibility of scientific workflows. However, neither the discovery paper published in Physical Review Letters, nor the technical paper published in Physical Review D documented the exact version of the PyCBC code used to produce the analysis. The notes from the original run recorded that PyCBC v1.3.2 was used, and recorded the git commit hash of the configuration files used (which are stored in a separate GitHub repository).

Open Data

The original analysis used data and metadata that are proprietary to the LIGO Scientific Collaboration and the workflow used tools that queried proprietary servers to locate and access these data. Our script modifies the workflow to use the public data and services provided by GWOSC. For the metadata, we created wrapper codes that have the same command-line API as the proprietary codes and translate these to queries against the public data repositories. The format of the data-quality metdadata provided by GWOSC is different to that used in the original analysis. Information from the public LIGO technical note T1600011-v3 was used to determine how to use the public metadata in way that is as close as possible to the original metadata. LIGO publishes its public data using CVMFS under the gwosc.osgstorage.org organization. Our script configures the workflow to use data from CVMFS, allowing us to rely on its distribution and caching capabilities when running jobs on the OSG. To allow the workflow generation script to find these data, we installed the LIGO Diskcache API to index the CVMFS files and the LIGO Datafind Server to resolve the workflow's metdatadata queries to file URLs for the CVMFS data. Configuration files for these tools are provided in our data release.

OUR SCRIPT CONFIGURES THE WORKFLOW TO USE DATA FROM CVMFS, ALLOWING US TO RELY ON ITS DISTRIBUTION AND CACHING CAPABILITIES WHEN RUNNING JOBS ON THE OSG.

Workflow Format

To provide sufficient resources to run the workflow, we executed the computationally intensive jobs on the OSG. This required a newer version of Pegasus workflow management system[10] than the version originally used to plan and execute the analysis. Our workflow

[a]https://github.com/gwastro/pycbc
[b]https://github.com/gwastro/pycbc-config
[c]https://doi.org/10.5281/zenodo.4085984
[d]https://github.com/gwastro/gw150914-fig4b/commits/1.1

generation script modifies the workflow written by PyCBC v1.3.2 to be compatible with Pegasus 4.9.3.

Access to Codes

Although all of the codes used to generate and run the analysis workflow were public, the script used to make the figure shown in Abbott *et al.*[3] was never released in the PyCBC software repository. Since one of the authors of this article helped create this script, we were able to obtain the original code used.

WORKFLOW EXECUTION

After modifying the original workflow generation script to address the challenges described in the previous section, we attempted to reproduce the analysis. LIGO did not provide estimates of the runtimes or the resource requirements of the analysis tasks, so we executed the workflow on a combination of local and OSG resources. We used USC-ISI computers to manage the workflow and run the postprocessing jobs and OSG resources to run the computationally intensive jobs. Several challenges were encountered during our attempt to execute the workflow, as described below.

Operating System and Hardware Mismatches

The PyCBC PyInstaller bundles are not true static executables nor are they packaged in a robust containerized environment like Singularity. The bundles require the appropriate C standard-library shared objects to be installed on the target machine and perform just-in-time compilation of bundled C code using the now-deprecated `scipy.weave` module. A standard set of OS libraries, the GNU C Compiler, and processor instructions was guaranteed for the original analysis as it was run on a single homogeneous LIGO Data Grid cluster. However, not all the OSG compute notes had the correct versions of C standard-library installed and some nodes lacked processor instructions (specifically, we encountered QEMU-emulated virtual machines that lacked the FMA4 instruction) that the PyCBC bundles required on the execute nodes. To address this, we used Pegasus and HTCondor matchmaking and fault tolerance functionalities and the ability to express requirements of the desired node characteristics to steer the PyCBC executables to compatible OSG compute nodes.

Nondeterministic Memory Use

The amount of memory that the matched-filtering jobs required is determined by the data that they analyze. If LIGO data contains more non-Gaussian noise than average, more memory is required to compute signal-based vetoes and to store the resulting triggers. Since the noise is random, it is not possible to determine in advance how much memory is required for a given job. To address this, we configured HTCondor to automatically request more memory on each retried filtering job that failed.

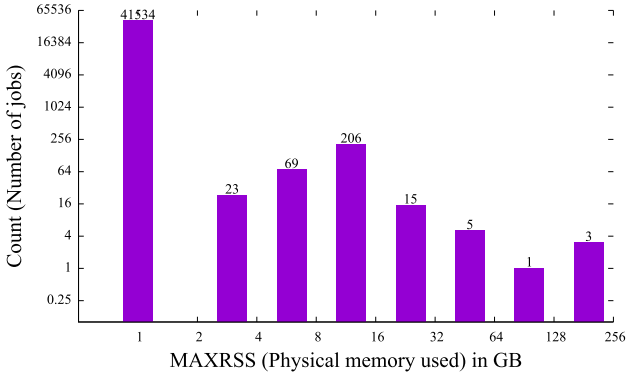

FIGURE 2. Frequency histogram showing maximum physical memory used by LIGO jobs as reported by *pegasus-kickstart* in range of 1–256 GB, with both *X*/*Y* axis on log scale.

Postprocessing Memory Requirements

Several of the workflow's postprocessing jobs require very large memory footprints (greater than 128 Gb). It was challenging to find machines with sufficient capacity for these jobs on OSG and so these jobs were executed on the local cluster at USC-ISI. This cluster is managed using HTCondor partitionable slots allowing a single job to request sufficient memory in our multicore machines. Coordination with the cluster administrators was required to ensure that these resources were available.

Long-Term Code Archival

During the preparation of this article, the LIGO Scientific Collaboration deleted the repository at `git.ligo.org` that stored the compiled PyInstaller PyCBC executables used in the original analysis. We had preserved a copy of the PyCBC v1.3.2 PyInstaller bundles prior to their deletion in an archive file on the IEEE DataPort server.[e] We have hosted an uncompressed version of this archive on a USC/ISI web server[f] and configured our workflow generation script to download the

[e]http://dx.doi.org/10.21227/c634-qh33
[f]https://pegasus.isi.edu/ligo/eager/pycbc-software/v1.3.2/

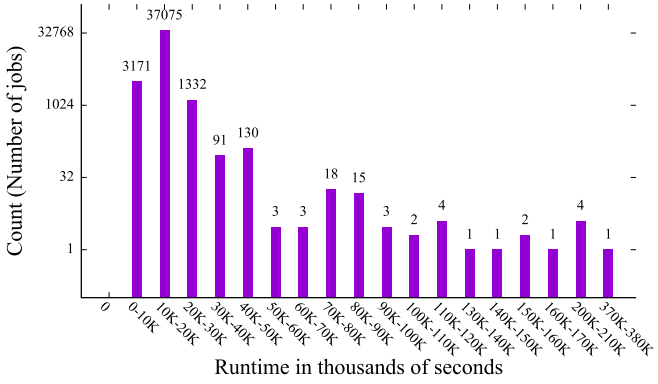

FIGURE 3. Frequency histogram showing runtime of LIGO jobs as reported by *pegasus-kickstart* in range of 0–380 000 s.

bundles from the USC/ISI server. Preserving these executables will allow others to run using the bundles rather than having to recreate the complex PyCBC software stack from the public source code available on GitHub.

RESULTS

Once the various issues described above had been addressed in our workflow-generation script `generate_workflow.sh`[g] and LIGO's PyCBC v1.3.2 PyInstaller bundles, we were able to reproduce the LIGO analysis workflow. The workflow contained almost 42 000 tasks). We observed 28 676 task failures as the workflow ran approximately 155 days of badput (amount of computation time used on failed jobs). The majority of job failures were caused by PyInstaller bundles landing on incompatible nodes and the majority of badput was due to compute-intensive jobs being evicted due to using too much memory. Failing jobs were rerun using the retry on failure semantics in Pegasus and HTCondor that then steered these jobs to compatible nodes. Listing 1 shows results retrieved from mining the Pegasus runtime provenance database. To execute the GW150914 workflow requires approximately 22 years of computing time (sum of duration of all jobs in the workflow with each job running on a single core).

The workflow generates a web page with a number of diagnostic plots that are used by LIGO scientists to understand the detector state, the properties of the noise, and the results of the search. For the purpose of reproducing Figure 4 of Abbott *et al.*,[3] the primary data product is a 2.5 Gb HDF5 file that contains the triggers found in coincidence between the LIGO detectors, and the search background estimated using the time-slide method.[7,8] We have archived a compressed version of this HDF5 file on the IEEE DataPort server.[h]

To allow future researchers to reproduce our work on their own resources, we show the distribution of physical memory used by LIGO jobs and their run-times as frequency histograms in Figures 2 and 3, respectively. In PyCBC workflows, each job type is associated with a transformation (executable). In Table 1, we describe the top 10 transformations ordered by the maximum physical memory used. Table 2 describes the top 10 transformations ordered by maximum runtime in seconds.

The workflow generates the data required to make Figure 4 of Abbott *et al.*[3] however, it does not generate the actual plot; a separate Python plotting script was used to create the histogram. As noted earlier, this script was not made public. Even though we had an internal version of the script, no PyInstaller bundle was created that captured the software stack used by that script. Running the plotting script against current versions of the libraries resulted in failures, so we needed to reproduce the original software stack. This was a considerable challenge and illustrates the importance of releasing containerized executables in addition to the source code for reproducibility in scientific analyses.

We obtained the version of PyCBC and LALsuite used by this code from notes made at the time of the original analysis (v1.3.4 and v6.36, respectively). We then determined the necessary and sufficient set of lower-level libraries required by these high-level libraries by examining the `setup.py` and `requirements.txt` in PyCBC v1.3.4. Using the PyCBC v1.3.4 install instructions, we create a Python virtual environment with the same version of `pip` and `virualenv` used in the original analysis. An iterative process of running the LALSuite `configure` script was performed until all the required dependencies of LALSuite were installed. The specific versions of 14 libraries (and their dependencies) were either installed using `pip` or compiled from source into the Python virtual environment. The iterative process of determining the required dependencies was complicated by the fact that `pip` caches previous software builds and so the install process is not necessarily idempotent.

[g]https://doi.org/10.5281/zenodo.4085984

[h]http://dx.doi.org/10.21227/c634-qh33

TABLE 1. Top 10 LIGO job types by maximum physical memory (maxrss) used in MB, where (FDFCC) expands to FULL_data_full_cumulative_cat and (FD_fb) to FULL_data_full_bin.

LIGO Job Transformation	Count	Mean (runtime) seconds	Min (mem) in MB	Max (mem) in MB	Mean (mem) in MB
distribute_background_bins-(FDFCC)_12H-H1L1_ID15	1	9673.17	194898.65	194898.65	194898.65
statmap-(FDFCC)_12H-H1L1_ID16	3	9698.37	16218.66	189332.06	103789.45
plot_snrifar-(FDFCC)_12H_(FD_FB)_2-H1L1_ID32	1	20459.47	150602.64	150602.64	150602.64
plot_snrifar-(FDFCC)_12H_(FD_FB)_2_IFAR-H1L1_ID34	1	2177.91	62365.84	62365.84	62365.84
plot_snrifar-(FDFCC)_12H_(FD_FB)_2_CLOSED-H1L1_ID21	1	1654.36	54009.43	54009.43	54009.43
plot_snrifar-(FDFCC)_12H_(FD_FB)_0_IFAR-H1L1_ID24	1	1473.62	40302.30	40302.30	40302.30
plot_snrifar-(FDFCC)_12H_(FD_FB)_0-H1L1_ID22	1	1484.56	40302.11	40302.11	40302.11
plot_snrifar-(FDFCC)_12H_(FD_FB)_0_CLOSED-H1L1_ID19	1	1167.86	34923.08	34923.08	34923.08
plot_singles-MTOTAL_EFFSPIN_NEWSNR_FULL_DATA-H1_ID47	1	1302.16	32334.82	32334.82	32334.82
plot_singles-ENDTIME_DURATION_NEWSNR_FULL_DATA-H1_ID45	1	1168.63	32334.81	32334.81	32334.81

Listing 1. Output of the pegasus-statistics tool showing runtime statistics from the OSG run

```
Type           Succeeded  Failed  Incomplete  Total  Retries
Tasks          41856      0       0           41856  28676
Jobs           46631      0       0           46631  28676
Sub-Workflows  8          0       0           8      104

Workflow wall time                       : 29 days, 0 hrs
Cumulative job wall time                 : 22 years, 54 days
Cumulative job badput wall time          : 155 days, 13 hrs
# Integrity Metrics
# Number of files for which checksums were compared/computed
  along
# with total time spent doing it.
94713 files checksums compared with total duration of 7 hrs, 55
mins
46200 files checksums generated with total duration of 4 hrs, 9
mins
# Integrity Errors
# Total:
#       Total number of integrity errors encountered across all
        job
#       executions(including retries) of a workflow.
# Failures:
#       Number of failed jobs where the last job instance had
integrity errors.
Total:    A total of 54 integrity errors encountered in the workflow
Failures: 0 job failures had integrity errors
```

After these libraries were installed, LALSuite and PyCBC were installed and the Python plotting script was executed. Our data release includes a script `make_pycbc_hist.sh`[i] that automates the installation and execution of the plotting code. Our reproduction of the LIGO result is shown in Figure 1, which includes the original LIGO/Virgo result for comparison.

We find that we were able to reproduce the search result. However, there are some small but noticeable differences in the search background (continuous black line) and the lower bound on the significance that the workflow reports for GW150914; We find the significance is greater than 5σ, rather than greater than 5.1σ (original plot). We attribute both of these differences to changes in the input data used by the workflow. The usage instructions for the GWOSC data state that the LIGO strain data in the public data set are based on the `C02` calibration of the LIGO detectors, whereas the original PyCBC configuration files state that `C01` data were used for the analysis of Abbott *et al.*[3] We hypothesize that the GWOSC `C02`-based data contains slightly less analysis time than the `C01` data originally used. This would result in a lower bound for the significance of the event and produce slight changes in the search background. However, we are not able to

[i]https://doi.org/10.5281/zenodo.4085984

TABLE 2. Top 10 LIGO job types by max runtime in seconds.

LIGO Job Transformation	Count	Mean (runtime) seconds	Min (mem) in MB	Max (mem) in MB	Mean (mem) in MB
hdf_trigger_merge-FULL_DATA-L1_ID12	1	376478.99	1026.20	1026.20	1026.20
calculate_psd-PART5-H1_ID75	1	205419.13	1027.13	1027.13	1027.13
calculate_psd-PART3-H1_ID73	1	204379.13	1024.27	1024.27	1024.27
calculate_psd-PART4-H1_ID74	1	202814.55	1024.95	1024.95	1024.95
calculate_psd-PART1-H1_ID71	1	202169.56	1024.81	1024.81	1024.81
calculate_psd-PART9-H1_ID79	1	167258.14	1356.44	1356.44	1356.44
calculate_psd-PART9-L1_ID68	1	116403.74	1408.61	1408.61	1408.61
calculate_psd-PART1-L1_ID60	1	115357.57	1386.77	1386.77	1386.77
calculate_psd-PART0-L1_ID59	1	110270.45	1025.48	1025.48	1025.48
calculate_psd-PART7-L1_ID66	1	109125.78	1398.72	1398.72	1398.72
calculate_psd-PART8-H1_ID78	1	71150.19	1344.01	1344.01	1344.01

verify this as we were unable to obtain access to the proprietary `C01` data.

CONCLUSIONS

We have described the process and challenges encountered in reproducing the measured statistical significance of GW150914. Our script is configured to execute compute-intensive jobs on the OSG. To allow scientists to run on other resources, our data release provides instructions for running all jobs on local resources. The memory and runtime profiling of the workflow tasks provided in this article will enable appropriate resource selection.

Our execution of the workflow used HTCondor as the job scheduler; this scheduler is also used by the LIGO Data Grid. Although one could modify our workflow generation script to use alternative job schedulers, we recommend against this because of the wide variance in the memory requirements of the jobs and need for a relatively homogeneous environment. We rely on HTCondor for job resubmission in case of failure and its mechanisms of custom-created HTCondor classAds to increase memory requested for a job in case of failures. When using a scheduler like SLURM, we recommend that one uses the upper memory bounds we provide in this article. A better practice would be to overlay HTCondor on the native scheduler using resource provisioning techniques such as HTCondor glideins. We also recommend the use of CVMFS to access GWOSC data. Although Pegasus can be configured to transfer the data at runtime, e.g., from the submit host, where the workflow system is located (USC/ISI in our case) or via http from the GWOSC web site, this requires movement of tens of thousands of input data files. It is more efficient to rely on the CVMFS storage and caching mechanism and to configure Pegasus to create symbolic links to the CVMFS locations, rather than performing true copies.

We have demonstrated that, although LIGO did not provide complete instructions for reproducing the GW150914 result, sufficient information exists either in the public domain or recorded as notes describing the original analysis to reproduce the PyCBC GW150914 workflow. Although we made substantial progress in reproducing the PyCBC result shown in Figure 4 of the paper by Abbott *et al.*,[3] we were unable to reproduce it exactly as we did not have access to the original input data and metadata. The LIGO data need to be calibrated based on the understanding of the characteristics of the instrument and its state. These calibrations may change over time as the knowledge about the detectors improves. Data providers often want to publish "best quality" data and not provide earlier outdated versions. This is the case with the data from LIGO's first observing run where only the final calibrated data are public. If the original data used in the GW150914 discovery paper are made public, it is straightforward to modify our workflow generation script to use these data. As part of this work, we have released scripts that allows other scientists to reproduce the LIGO analysis using publicly available data using their own compute resources or the OSG using the latest stable versions of Pegasus and HTCondor.

Our results show that, in principle, it is possible to release instructions and code that allow other scientists to reproduce a major scientific result. We encourage scientists who wish to do so to ensure that instructions include: access to the original data and codes used; documentation of software and configuration file versions; containerized executables that capture the complete software stack used in the original analysis; long-term archival of code and data

products used; and documentation about the computational resources needed to execute the analysis. Understanding how reproducibility is incorporated in astrophysics workflows in general and scientific workflows in particular through the sharing of practices in reproducible scientific software will help enable open science across disciplines. Codes, data, and workflows generated by this and similar efforts can ultimately enable researchers and students at various levels of education to regenerate the same findings, learn about the scientific methods, and engage in new science, technology, engineering, and mathematics (STEM) research.

ACKNOWLEDGMENTS

The scripts and supporting codes used to run the PyCBC workflow generation code using the GWOSC data and to make the result plot described in this article are available from GitHub at https://github.com/gwastro/gw150914-fig4b. The specific version used in this work was https://doi.org/10.5281/zenodo.4085984. The PyCBC v1.3.2 PyInstaller bundles used by the workflow and the HDF5 file created by the PyCBC workflow are available from http://dx.doi.org/10.21227/c634- qh33.

The authors would like to thank M. Rynge for providing input on configuring the pipeline to run on OSG, A. Nitz and M. Alessandra Papa for providing the script used to make the PyCBC result histogram, and S. Anderson for helpful discussions. This work was supported by the U.S. National Science Foundation under Grant OAC-1823378, Grant OAC-1823405, Grant OAC-1841399, and Grant OAC-1823385. Pegasus is supported by the U.S. National Science Foundation under Grant OAC-1664162. The Open Science Grid is supported in part by the U.S. National Science Foundation under Grant PHY-1148698, and the U.S. Department of Energy's Office of Science. This research has made use of data, software, and/or web tools obtained from the Gravitational Wave Open Science Center, a service of LIGO Laboratory, the LIGO Scientific Collaboration, and the Virgo Collaboration. LIGO is funded by the U.S. National Science Foundation. Virgo is funded, through the European Gravitational Observatory (EGO), by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale della Fisica Nucleare (INFN), and the Dutch Nikhef, with contributions by institutions from Belgium, Germany, Greece, Hungary, Ireland, Japan, Monaco, Poland, Portugal, and Spain.

REFERENCES

1. V. Stodden *et al.*, "Enhancing reproducibility for computational methods," *Science*, vol. 354, no. 6317, pp. 1240–1241, 2016, doi: 10.1126/science.aah6168.
2. M. A. Heroux, L. Barba, M. Parashar, V. Stodden, and M. Taufer, "Toward a compatible reproducibility taxonomy for computational and computing sciences," Sandia Nat. Lab., Albuquerque, NM, USA, Tech. Rep. SAND2018-11186 669580, 2018, doi: 10.2172/1481626.
3. B. P. Abbott *et al.*, "Observation of gravitational waves from a binary black hole merger," *Phys. Rev. Lett.*, vol. 116, Feb. 2016, Art. no. 061102, doi: 10.1103/PhysRevLett.116.061102.
4. T. Venumadhav, B. Zackay, J. Roulet, L. Dai, and M. Zaldarriaga, "New search pipeline for compact binary mergers: Results for binary black holes in the first observing run of advanced LIGO," *Phys. Rev. D*, vol. 100, no. 2, 2019, Art. no. 023011, doi: 10.1103/PhysRevD.100.023011.
5. A. H. Nitz *et al.*, "1-OGC: The first open gravitational-wave catalog of binary mergers from analysis of public advanced LIGO data," *Astrophys. J.*, vol. 872, no. 2, 2019, Art. no. 195, doi: 10.3847/1538-4357/ab0108.
6. D. Chapp *et al.*, "Applicability study of the PRIMAD model to LIGO gravitational wave search workflows," in *Proc. 2nd Int. Workshop Practical Reproducible Eval. Comput. Syst.*, 2019, pp. 1–6, doi: 10.1145/3322790.3330591.
7. S. A. Usman *et al.*, "The PyCBC search for gravitational waves from compact binary coalescence," *Classical Quantum Gravity*, vol. 33, no. 21, 2016, Art. no. 215004, doi: 10.1088/0264-9381/33/21/215004.
8. B. P. Abbott *et al.* "GW150914: First results from the search for binary black hole coalescence with advanced LIGO," *Phys. Rev. D*, vol. 93, no. 12, 2016, Art. no. 122003, doi: 10.1103/PhysRevD.93.122003.
9. B. Bockelman *et al.*, "Commissioning the HTCondor-CE for the open science grid," *J. Phys., Conf. Ser.*, vol. 664, no. 6, 2015, Art. no. 062003, doi: 10.1088/1742-6596/664/6/062003.
10. E. Deelman *et al.*, "The evolution of the pegasus workflow management software," *Comput. Sci. Eng.*, vol. 21, no. 4, pp. 22–36, 2019, doi: 10.1109/MCSE.2019.2919690.
11. D. Weitzel, B. Bockelman, D. A. Brown, P. Couvares, F. Würthwein, and E. F. Hernandez, "Data access for LIGO on the OSG," in *Proc. Pract. Experience Adv. Res. Comput. Sustainability, Success Impact* , 2017, pp. 1–6, doi: 10.1145/3093338.3093363.
12. M. Vallisneri, J. Kanner, R. Williams, A. Weinstein, and B. Stephens, "The LIGO open science center," *J. Phys. Conf. Ser.*, vol. 610, no. 1, 2015, Art. no. 012021, doi: 10.1088/1742-6596/610/1/012021.

DUNCAN A. BROWN is currently the Charles Brightman Professor of Physics with Syracuse University, Syracuse, NY, USA. He was a member of the LIGO Scientific Collaboration from 1999 to 2018 and is a Fellow of the American Physical Society. He is a coauthor of the paper "Data access for LIGO on the OSG," which won Best Software and Data Paper at PEARC17. His research includes gravitational-wave astronomy and astrophysics, and the use of large-scale scientific workflows. He received the Ph.D. degree in physics from the University of Wisconsin-Milwaukee, Milwaukee, WI, USA, in 2004. Contact him at dabrown@syr.edu.

KARAN VAHI is currently a senior computer scientist with the USC Information Sciences Institute, Marina Del Ray, CA, USA. He is a co-author of the paper "Integrity Protection for Scientific Workflow Data: Motivation and Initial Experiences," which won Best Paper in the Advanced Research Computing Software and Applications Track and also the "The Phil Andrews Most Transformative Contribution Award" at PEARC19. His research interests include scientific workflows and distributed computing systems. He received the M.S degree in computer science from the University of Southern California, Los Angeles, CA, USA, in 2003. Contact him at vahi@isi.edu.

MICHELA TAUFER holds the Jack Dongarra Professorship in high performance computing within the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, USA. She is an ACM Distinguished Scientist. Her interdisciplinary research is at the intersection of computational sciences, high permanence computing, and data analytics. She received the Ph.D. degree in computer science from the Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, in 2002. She is a Senior Member of IEEE. Contact her at taufer@acm.org.

VON WELCH is currently the Acting Associate Vice President for Information Security, Executive Director for the OmniSOC, Executive Director for Cybersecurity Innovation with Indiana University, Bloomington, IN, USA, and the Director of the IU's Center for Applied Cybersecurity Research (CACR). He specializes in cybersecurity for distributed systems, particularly scientific collaborations, and federated identity. Contact him at vwelch@iu.edu.

EWA DEELMAN is a research director with USC/ISI, Marina del Rey, CA, USA and a research professor with the USC Computer Science Department. Her research explores the interplay between automation and the management of scientific workflows that include resource provisioning and data management. Her group has lead the design and development of the Pegasus Workflow Management software (http://pegasus.isi.edu) and conducts research in job scheduling and resource provisioning in distributed systems, workflow performance modeling, provenance capture, and the use of cloud platforms for science. She is an AAAS and IEEE Fellow. She received the Ph.D. degree in computer science from the Rensselaer Polytechnic Institute, Troy, NY, USA. Contact her at deelman@isi.edu.