Astronomy and Computing 42 (2023) 100664

Contents lists available at ScienceDirect

Astronomy and Computing

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

Full length article

RUNMON-RIFT: Adaptive configuration and healing for large-scale parameter inference

R. Udall [a,*], J. Brandt [b], G. Manchanda [b], A. Arulanandan [b], J. Clark [a,b], J. Lange [c], R. O'Shaughnessy [d], L. Cadonati [b]

[a] LIGO Laboratory, California Institute of Technology, United States of America
[b] Center for Relativistic Astrophysics, Georgia Institute of Technology, United States of America
[c] Center for Gravitational Physics, University of Texas at Austin, United States of America
[d] Center for Computational Relativity and Gravitation, Rochester Institute of Technology, United States of America

ARTICLE INFO

Article history:
Received 17 January 2022
Accepted 3 November 2022
Available online 12 November 2022

ABSTRACT

Gravitational wave parameter inference pipelines operate on data containing unknown sources, and run on distributed hardware with widely varying configurations and stochastic transient errors. For one specific analysis pipeline (RIFT), we have developed a flexible tool (RUNMON-RIFT) to mitigate the most common challenges introduced by uncertainties in source parameters and computational hardware. On the one hand, RUNMON provides mechanisms to adjust pipeline-specific run settings, including prior ranges, to ensure the analysis completes and encompasses the physical source parameters. On the other, it provides tools for identifying and adjusting to the realities of hardware uncertainties. We demonstrate both general features with controlled examples.

1. Introduction

Since the first gravitational wave detection GW150914 (Abbott et al., 2016a), the Advanced Laser Interferometer Gravitational Wave Observatory (LIGO) (Aasi et al., 2015) and Virgo (Accadia et al., 2012; Acernese et al., 2015) detectors have continued to discover gravitational waves (GW) from coalescing binary black holes (BBHs) and neutron stars (Abbott et al., 2019a, 2020b, 2021a,b, 2019b, 2020c,a; Abbott et al., 2021c). From the point at which data is collected, many computational analyses are required to render it into information of astrophysical interest, including detector characterization (Davis et al., 2021), calibration and data cleaning (Sun et al., 2021; Acernese et al., 2021), candidate identification (Usman et al., 2016; Cannon et al., 2021; Adams et al., 2015), noise estimation (Littenberg and Cornish, 2015), and parameter estimation (Wysocki et al., 2019; Veitch et al., 2015; Ashton et al., 2019). For the small number of observations reported through GWTC-3 (approximately 90 over 3 observing runs), these analyses could be monitored by individual humans to identify and remedy any problems that can occur. However, as detector sensitivity improves the number of observations and thus inferences increase (potentially hundreds in O4 alone (Abbott et al., 2016b)), saturating the ability of individual humans to carefully curate each analysis individually, such that automation will be necessary to correct common problems in future observing runs. This problem is especially acute for parameter inference, which this paper will focus on, though automation schemes have also been implemented for other types of analysis, see for example Brown et al. (2021), Vahi et al. (2019), Singer et al. (2021) and Essick et al. (2020). The most salient comparison to this software is Asimov (Williams et al., 2022), which has been developed for LVK analyses using RIFT and other parameter inference pipelines, and was used in GWTC-2.1 and GWTC-3 (Abbott et al., 2021a,b). This software has features in common with RUNMON-RIFT, most importantly monitoring software for project level management and automatic resubmission. However, notable features of our work – most importantly railing-correction and node exclusion – are not presently implemented in Asimov.

Parameter inference for gravitational waves is generally done within the Bayesian analysis framework. For many possible configurations of parameters which contribute to the gravitational wave (see Section 2.1 for details) likelihood values are computed – in this case using approximate models of waveform behavior (Ossokine et al., 2020; García-Quirós et al., 2020) – and are combined algorithmically with prior expectations to generate posterior distributions which describe the probability of various configurations. For this analysis to be robust, it generally requires at least $\mathcal{O}(10^6)$ likelihood evaluations, which may be computationally expensive. Various methods exist to sample these distributions efficiently, but all are of substantial complexity, and are run primarily on supercomputing clusters. In turn,

* Corresponding author.
E-mail address: rudall@caltech.edu (R. Udall).

https://doi.org/10.1016/j.ascom.2022.100664
2213-1337/

this complexity allows for many potential issues, both in the settings of the algorithm and the operation of the software, which may drastically reduce the pace of analysis.

In this paper, we discuss a newly developed Python package, RUNMON-RIFT,[1] which seeks to address a number of such problems in inferences performed using RIFT (Lange et al., 2018), one of the parameter estimation (PE) pipelines to interpret events in GWTC-1 (Abbott et al., 2019a), GWTC-2 (Abbott et al., 2020b), GWTC-2.1 (Abbott et al., 2021a), and GWTC-3 (Abbott et al., 2021b), as well as many individual events (Abbott et al., 2019b, 2020c,a; Abbott et al., 2021c). RUNMON-RIFT (and RIFT more generally) is geared primarily towards use on the LIGO Data Grid, a collection of independently operated computing clusters running HTCondor (Thain et al., 2005; Fajardo et al., 2015; Bockelman et al., 2020) with a common software environment and identity access management system, though RUNMON-RIFT also sees some use on the Open Science Grid (Pordes et al., 2007; Sfiligoi et al., 2009) via the LDG interface to it.

The challenges faced by software in scientific computing are highly context dependent, and parameter estimation software is no exception. However, some issues are common in many gravitational wave inference pipelines, and we shall focus on discussing these, with solutions tailored to the specific circumstances of RIFT. Large-scale parameter inference is frequently bedeviled by computing issues which are, as an individual matter, relatively straightforward to address, but which are, taken together, very difficult to resolve systematically. Notable examples of this behavior include misconfigured nodes (e.g. the node's hardware is incompatible with the system software distributions), transient issues (e.g. a node loses contact with the filesystem for a period while a job is attempting to run), and system wide configuration errors (e.g. expired authentication tokens, or filesystem errors). We introduce tools for managing such issues, both by immediately continuing the progress of a job, and by providing infrastructure to proactively avoid them and provide information about their origin to cluster administrators.

Another common issue with parameter inference is "railing": an artificially narrow prior range that constrains the extent of the posterior distribution in a physical parameter, significantly skewing the final result. For many practical reasons, PE inference pipelines adopt narrow prior ranges based on expectations informed both by experience and by any additional information, such as the output of a detection pipeline which identified the event originally. This is imperfect, however, especially when analysis is being done in bulk and available person hours to identify optimal settings are limited. We implement a mechanism for correcting this automatically; this algorithm works best with RIFT for reasons which will be described in more detail in Section 2, but could be broadly adapted for any PE software.

This paper is organized as follows. In Section 2 we review the RIFT parameter inference engine. We begin with the core functionalities of RUNMON-RIFT, including its logging and tools it implements which dramatically decrease the amount of person-hours required to ensure a workflow's completion. This includes a discussion of common error modes, and of a prototypical computing issue which RUNMON-RIFT helped overcome. We then describe how we identify potential 'railing' in our posterior distribution, associated with artificially narrow boundaries, and we introduce an adaptive method to extend these parameter-space boundaries. Finally, we discuss a toy model to demonstrate how RUNMON-RIFT can go beyond reactive workflow management, and proactively ensure that the computational pool used by a workflow is less likely to contain transient computing issues. Section 3 demonstrates the automated healing of these runs in both a stereotypical case of railing and for our computing issues toy model.

[1] Available at https://pypi.org/project/runmonitor-RIFT/.

2. Methods

2.1. RIFT review

A coalescing compact binary in a quasicircular orbit can be completely characterized by its intrinsic and extrinsic parameters. By intrinsic parameters we refer to the binary's masses m_i, spins, and any quantities characterizing matter in the system. For simplicity and reduced computational overhead, in this work we provide examples of parameter inference which assume all compact object spins are aligned with the orbital angular momentum; however, the techniques introduced in our study are not specific to any specific set of parameters or dimension. By extrinsic parameters we refer to the seven numbers needed to characterize its spacetime location and orientation. We will express masses in solar mass units ($M_\odot$), and dimensionless nonprecessing spins in terms of cartesian components aligned with the orbital angular momentum $S_{i,z}$. We will use $\boldsymbol{\lambda}, \boldsymbol{\theta}$ to refer to intrinsic and extrinsic parameters, respectively.

RIFT (Lange et al., 2018) consists of a two-stage iterative process to interpret gravitational wave data d via comparison to predicted gravitational wave signals $h(\boldsymbol{\lambda}, \boldsymbol{\theta})$. In the first stage, denoted by ILE (Integrate Likelihood over Extrinsic parameters), for each λ_α from some proposed "grid" $\alpha = 1, 2, \ldots, N$ of candidate parameters, RIFT computes a marginal likelihood

$$\mathcal{L}(\boldsymbol{\lambda}) \equiv \int \mathcal{L}_{\rm full}(\boldsymbol{\lambda}, \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{1}$$

from the likelihood $\mathcal{L}_{\rm full}(\boldsymbol{\lambda}, \boldsymbol{\theta})$ of the gravitational wave signal in the multi-detector network, accounting for detector response, and extrinsic parameters prior $\pi(\boldsymbol{\theta})$; see the RIFT paper for a more detailed specification. In the second stage, denoted by CIP (Construct Intrinsic Posterior), RIFT performs two tasks. First, it generates an approximation to $\mathcal{L}(\boldsymbol{\lambda})$ based on its accumulated archived knowledge of marginal likelihood evaluations $(\lambda_\alpha, \mathcal{L}_\alpha)$. This approximation can be generated by Gaussian processes, random forests, or other suitable approximation techniques. Second, using this approximation, it generates the (detector-frame) posterior distribution

$$p(\boldsymbol{\lambda}) = \frac{\mathcal{L}(\boldsymbol{\lambda})\pi(\boldsymbol{\lambda})}{\int d\boldsymbol{\lambda} \mathcal{L}(\boldsymbol{\lambda})\pi(\boldsymbol{\lambda})}. \tag{2}$$

where prior $\pi(\boldsymbol{\lambda})$ is the prior on intrinsic parameters like mass and spin. The posterior is produced by performing a Monte Carlo integral: the evaluation points and weights in that integral are weighted posterior samples, which are fairly resampled to generate conventional independent, identically-distributed "posterior samples." For further details on RIFT's technical underpinnings and performance, see Lange et al. (2018), Wysocki et al. (2019) and Lange (2020).

Parameter inference analyses generally require many configuration details, notably including prior assumptions and the amount of data to be analyzed. Most relevant to this work is the fact that, for computational efficiency, the priors adopted are generally targeted to cover a limited range of mass and luminosity distance most likely to enclose the true source parameters, with initial ranges chosen motivated by search results. A second critical setting is the starting frequency of the waveform's dominant quadrupole mode. For an inspiralling binary at early times, this frequency is twice the orbital frequency. Because the orbital frequency at the last stable orbit decreases with mass, for binaries with a large detector-frame mass a conventional starting frequency like $f_{lower} = 20$ Hz is too high: the waveform model does not permit it. Furthermore, generation of waveforms with higher modes must start at a reduced frequency ($f_{min} = 2f_{lower}/L_{max}$), in order that no mode's initial frequency is above

Table 1
Examples of common errors.

Error description	Recognition method	Machine excludable?	Fixed at origin?
CUDA compute incompatibility	Custom error code	Yes	Yes for IGWN clusters
Interpreter runtime error	Standard HTCondor error code	No	No
Interpreter not found error	Standard HTCondor error code	Yes	Yes
XLAL file transient	Output parsing	No	Yes

the fiducial starting frequency (i.e., no mode starts in band). A third critical setting is the amount of data to analyze, or "segment length". As the relevant starting frequency or mass decreases, the amount of data needed to be analyzed increases. Misidentification of any of these settings can cause a cascade of changes. For example, a mis-adapted mass prior might artificially exclude low masses, requiring a re-evaluation of the relevant segment length.

2.2. RUNMON-RIFT introduction

RUNMON-RIFT is a Python package to monitor and manage runs, including correcting common failure modes encountered. At its core, RUNMON-RIFT consists of tools to assemble and manage a lightweight run tracking log. In addition to monitoring the queuing system (HTCondor) logs, RUNMON-RIFT includes generic tools to parse, query, and even edit RIFT's internal files. This allows it to interpret log outputs for identification of pathological behaviors, and edit configuration files according to custom algorithms. A daemon will periodically use these tools to update status on each job under its purview, and, using the archived run logs, the RUNMON-RIFT suite can quickly assemble reports on run status, including measures of convergence. Moreover, being aware of the workflow's status and being able to edit the workflow and even RIFT settings, RUNMON-RIFT can adapt to issues arising with the host cluster, or individual machines upon it,[2] in a fashion that is minimally disruptive to ongoing analysis. RUNMON-RIFT's "healing" functions provide unique capability to handle ubiquitous challenges arising in large-scale parameter inference calculations. In this work, we will illustrate three such operations. First, we will consider healing parameter "railing", a generic issue associated with user mis-specified priors. Second, we will demonstrate how RUNMON-RIFT can respond to a transient cluster issue, here exemplified by problems with GPU use. Finally, we will show how RUNMON-RIFT can efficiently block use of undesired computing machines (e.g., identified by job failures or even slow past performance).

2.3. Managing jobs

In its simplest manifestation, RUNMON-RIFT implements a run index with operational metadata. LDG clusters feature a web-facing directory which may be accessed from a browser, and in this directory a file structure is generated, in which the user's run are organized by what event they are running on. For each run, there are a series of text files containing information about the run such as the name, location, number of completed iterations, and convergence details; RUNMON-RIFT includes a set of utilities which allow the user to parse these conveniently. A daemon is used to automatically analyze the workflows which are registered to its database, and updates the aforementioned metadata accordingly. Thus, we have operational information on all ongoing runs, allowing us to quickly identify potential problems and characterize overall progress, both critically important when working with many often heterogeneous analyses simultaneously.

RUNMON-RIFT can provide fixes for some of the many other issues which can prevent progress on a run. These issues can be conveniently flagged by the code, by the use of specific return values from the two key routines(ILE and CIP). Alternatively, RUNMON-RIFT can parse the codes' output and HTCondor logs, to identify and characterize issues that can cause the run to fail. Quite frequently, these issues are transient in the sense that they are not caused by the structure of the analysis itself, but rather by incompatibilities which occur only in certain parts of a heterogeneous computing pool.

A prototypical example is GPU utilization: RIFT uses GPUs to improve efficiency, but a given cluster may include many separate machines, often varying dramatically in age. Updates in some standard computing environments to the software library used by RIFT (`CUDA` Cook, 2012, called via the Python library `CUPY` Okuta et al., 2017) rendered it incompatible with some machines on a popular cluster, which in turn led to an extremely high failure rate, forcing the user to resubmit repeatedly until a job would be lucky enough to land on a compatible machine. This example motivated the introduction of automated resubmission within RUNMON-RIFT, so that up time for runs could be maintained with minimal user intervention, and during times when users would not be available. Furthermore, it inspired the machine exclusion algorithm described in Section 2.5. Ultimately, the root of the problem was identified after a number of months, and usage of software libraries was altered to remove the issue at its source for runs on shared IGWN filesystems, but the resubmission mechanics remain necessary for the highly heterogeneous OSG pool. Table 1 displays several additional errors which RUNMON solves in an analogous manner. Many result from instabilities in cluster filesystems which change frequently and are unavoidable for the end user. When a transient is sufficiently common and results in a consistent error message in the Python runtime, RIFT is edited to provide standard error codes for these errors, such that RUNMON-RIFT may more easily identify and cope with them.

2.4. Healing parameter railing

The priors $\pi(\boldsymbol{\theta}), \pi(\boldsymbol{\lambda})$ over extrinsic and intrinsic parameters are usually proportional to some a priori separable function. In each variable, the range and normalization of the prior is over some finite range. Sometimes the boundaries are physical and absolute, for example when integrating over phase or sky location. However, for variables like luminosity distance or mass, the user usually adopts upper and lower bounds for computational convenience, to bound the overall time to solution, centered on a weakly-informed guess. When performing large-scale inference, these arbitrary bounds are not-infrequently mis-specified, and the posterior is artificially constrained, "railing" against one or more boundaries.

Railing can be identified by having significant posterior support immediately adjacent to one of the arbitrary prior bounds. The blue curve in Fig. 1 shows an example of a railed posterior. Quantitatively we identify it as follows. Suppose for parameter x we have a sampled posterior $\hat{p}(x)$. We divide the *prior* range $\pi(x)$

[2] A given cluster may have many different machines, with varying behaviors due to hardware architecture, utilization protocols, and the like. We adopt the terminology "machine" to reflect HTCondor's internal attribute designation, but they may also be variously known as nodes or sites — machines on the primary LDG cluster, for example, have the naming convention node###.cluster.ldas.cit.

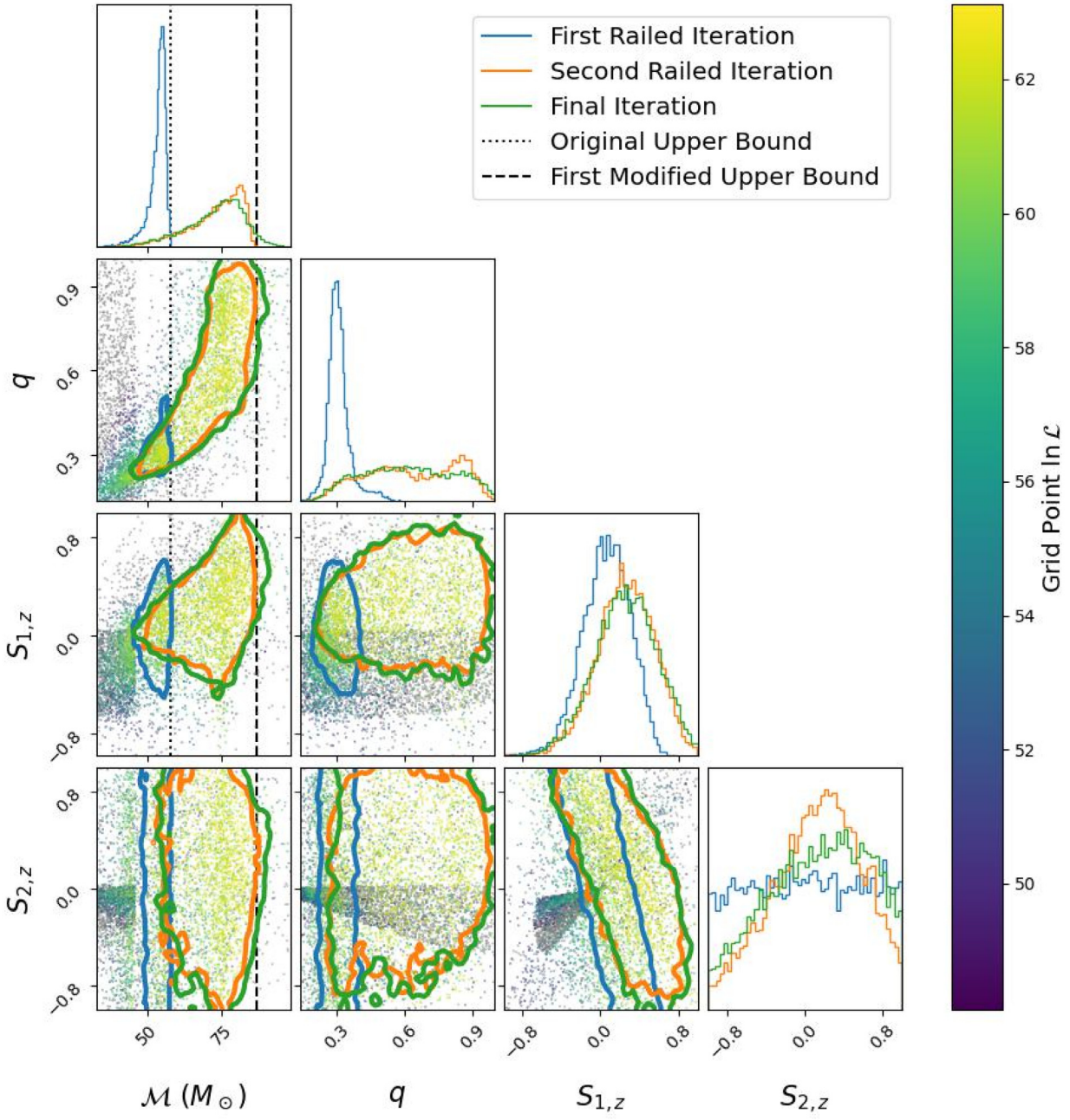

Fig. 1. Analysis for the GW190602_175927 (Abbott et al., 2021d), an event where standard parsing of internal low-latency estimates results in initially incorrect boundaries in $\mathcal{M}_c$. Contours are shown for iterations which triggered RUNMON-RIFT's railing test, as well as the final result, and vertical lines show the boundaries at the iterations where railing was found. The final boundary occurs substantially to the right of the plot's extent in $\mathcal{M}_c$. Colored points are individual points on the grid, with the heat map corresponding to likelihood.

into bins of equal width, with 20 bins as the default. We then determine the density of the posterior in the first and last bin. If this density exceeds some threshold – we use 0.03 by default – in either bin, we evaluate this bin to be railed, and extend the corresponding boundary accordingly. We emphasize this definition applies only to parameters with *user-specified* boundaries; parameters which have absolute limits, like the mass ratio $q = m_2/m_1$, do not rail against those limits, as more extreme values are unphysical.

RIFT's intrinsic boundaries only impact the second (CIP) phase [Eq. (2)] of the cycle, not the phase in which the likelihood values are computed [Eq. (1)]. RUNMON-RIFT's daemon reads the output of each CIP phase (these take the form of sampled posteriors over the intrinsic parameters $\hat{p}(\boldsymbol{\lambda})$), and applies our algorithm to identify railing if appropriate. If it is detected, the job is removed from the cluster, and RUNMON-RIFT changes the boundaries which are found to be railing. If lower bound railing is detected, R_{lower} is mapped to $(1-m)R_{lower}$ and if upper bound railing is detected R_{upper} is mapped to $(1+m)R_{upper}$, where m is 0.5 by default. The same job may then be resumed, without having to create a new workflow.[3] At least 2 iterations are required before the extended range can be fully explored, but since railing is normally identified early in the analysis, the job almost always has the ability to explore the extended range. In cases where a job does fail to

[3] In the language of HTCondor, we resubmit the dag.

explore this range, methods exist for creating new jobs which effectively continue the runs, though this does require manual intervention. Other parameter estimation methods – notably MCMC and nested sampling methods – require more complex methods of intervention and continuation to achieve similar results, such as the initialization of helper analyses, though it is also potentially feasible to automate these.

2.5. Problematic machine exclusion

Computing clusters frequently suffer from transient errors, usually triggered by some change in the computing environment, which take the form of everything from failed software dependencies to difficulties with file transfers. Since the specific conditions required to trigger these transients may only occur for certain computational tasks, on certain machines, or with specific settings, they may be difficult to track and address. Also problematic is the phenomena of "black hole" machines: a colloquial term referring to when a machine will accept a job, but that job will quickly fail due to something inherent to the machine (such as the aforementioned GPU incompatibility). Specifically, if large numbers of jobs are submitted in parallel (as is the case for high-throughput computing tasks, such as the ILE stage of RIFT), the scheduler will attempt to assign them in bulk. If the available computational resources are limited, then some fraction will be assigned and the rest will occupy the next spots in the queue. Accordingly, a single machine experiencing some transient error may fail immediately, then accept another job from this queue. In sufficiently low resource situations, this may result in the entire parallel content of a high-throughput computing job failing on a single machine.

To mitigate the impact of problematic machines, RUNMON-RIFT allows for tracking machines associated with known transient errors, and provides a tool for instructing the relevant HTCondor jobs to exclude these machines from matchmaking consideration. The information about which machines should be excluded is shared across all jobs managed by a given daemon, and thus is propagated quickly for all of a user's active jobs. Logging of these machines and their associated failure modes also offers a collated set of data to provide system administrators when troubleshooting issues, such that the root problem may be identified and addressed, at which point it is straightforward to remove the restrictions the daemon imposed upon the pool.

3. Results

3.1. Healing

Fig. 1 depicts a prototypical example of railing, along with the correction produced by RUNMON-RIFT. The pipeline constructor for the event in question, GW190602 (Abbott et al., 2021d), produced a railed prior boundary in chirp mass $\mathcal{M}_c$ when taking the metadata of the event's initial detection as input. Accordingly, it required careful and tedious human intervention, lest any run be completely ruined. The use of RUNMON-RIFT may be seen to alleviate this in the progression of the results seen in Fig. 1. The plot in question is an example of a corner plot — displaying both one dimensional histograms of individual parameters, as well as their two dimensional joint parameters, so that correlation may be understood and diagnosed. RIFT corner plots also include colored points to show the likelihood values of the underlying grid, with the yellow spectrum colors corresponding to the highest likelihood points, and the purple spectrum colors corresponding to the lowest likelihood points. Here the posterior after the first iteration of the workflow (the blue curve) may be seen to rail at the upper boundary in $\mathcal{M}_c$ which was set by the pipeline constructor (the dotted black line). Notably, this distribution also has an erroneous posterior distribution in mass ratio q, due to the correlation of this parameter with the erroneous chirp mass. RUNMON-RIFT then automatically increased the upper boundary to the value seen in the dashed black line, and continued the sampling process. After a number of iterations, the posterior had shifted to the orange curve, which may be seen to also rail (though to a lesser degree) against the modified upper bound, and so RUNMON-RIFT modified the upper bound once more, well past limits of the corner plot ($\mathcal{M}_c \approx 135 M_\odot$). Final sampling then brought the posterior to an unrailed distribution (the green curve), which agrees with the results presented for this event in Abbott et al. (2021a).

3.2. Runcrasher

To demonstrate the principle of machine exclusion, we construct an artificial scenario with known parameters and behavior which mimics the transient errors known to occur on computational clusters. In particular, we insert a step into the standard ILE portion of the workflow which tests the machine upon which the job lands, and produces a failure if that machine's name satisfies certain constraints (e.g. if the last digit is 5). RUNMON-RIFT included this failure code as one of the known transient values, and the exclusion system was triggered accordingly. We conducted tests under various constraints, reflecting the varying incidence rate of transients. This construct also naturally results in the aforementioned "black hole" machines when submission incidentally occurs at a time of high resource usage.

The results of this artificial scenario and corresponding intervention are shown in Fig. 2. The bar charts indicate the behavior of individual machines under high and low transient incidence rates respectively. Transients are separated into two types: those which are caused by the runcrasher, which behave in a predictable manner and are subject to machine exclusion, and those which are caused by miscellaneous other transients, which are not well characterized and not subject to machine exclusion (for the runs in question these transients primarily involve accessing certain public files). The scatter plots show how many machines are actively excluded for each of these corresponding submission batches.

A number of features may be noted in these plots. Firstly, the submission events for which the total number of jobs increase are those submissions which occur at the beginning of a new iteration. The total duration of iterations for which the same numbers of jobs are submitted decrease correspondingly in the high-incidence case (the number of jobs submitted per iteration varies over the duration of the underlying workflow to improve its efficiency). Similarly, the relative proportion of errors which are due to the unmodeled transients increases. The low incidence case shows somewhat similar behavior, though it is also more strongly subject to low number statistics, as it is relatively rare to hit a failure machine in the first place.

When including analysis of the number of machines submitted, one may also see the expected trend: initial jobs result in substantially more blocked machines, while later jobs run in a cleaner pool, and thus are less likely to simultaneously interact with many error-triggering machines. One may also note that there are submission events (submission event 20, for example) for which most jobs fail but very few machines are blocked — this is an example of the aforementioned black-hole machine phenomenon. These unfortunately take longer to root out, since it gets progressively harder to filter through the pool when it is already mostly successful, but integration of these error lists across multiple runs mean that in a high usage context (if one has 10 runs simultaneously, for example) it is very feasible to fully eliminate problematic machines.

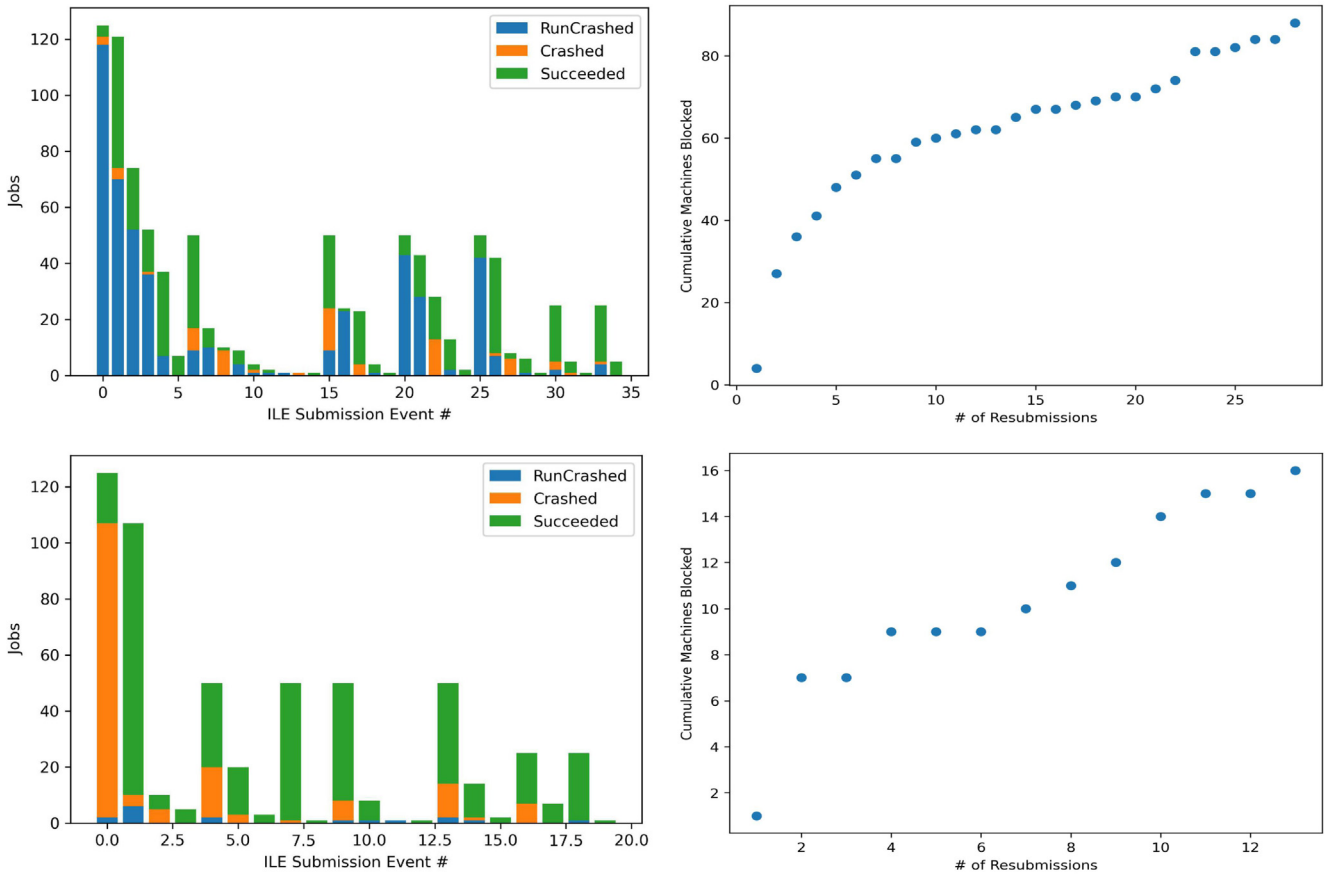

Fig. 2. The behavior of ILE jobs and number of machines blocked as a function of the associated ILE submission batch, for high and low error rate scenarios. *Left panels*: Histogram of the number of failed jobs versus submission attempt. Jobs labeled Succeeded complete normally; jobs noted as RunCrashed have intentionally failed, due to landing on a set of pre-selected target hosts; and jobs labeled Crashed fail for other reasons, not infrequently associated with problematic or misconfigured host machines. *Right panel*: Cumulative number of blocked as a function of rescue attempt.

4. Conclusion

We have presented our Runmonitor for RIFT (RUNMON-RIFT), a utility which greatly aids in the operation of this inference pipeline. Gravitational wave science is in an exciting time, with a rapid pace of discovery and exponentially increasing data to analyze. In this context, it is critical that the time required to complete a parameter estimation task, and the time the user spends actively monitoring and intervening in that task, be minimized. By introducing centralized diagnostic tools, RUNMON-RIFT makes it much easier for a user to check the status of their active jobs. Automated resubmission for known transient issues greatly decreases the amount of time a user spends actively engaging with the workflow (in particularly hostile computing environments this decrease may be up to an order of magnitude), and machine exclusion allows one to tailor the pool utilized towards the machines which will actually work consistently, decreasing restarts and improving efficiency for all cluster users. Monitoring of railing allows for aggressive (and hence efficient) initial settings, while also reducing the need for producing new workflows during exploratory phases of analysis.

CRediT authorship contribution statement

R. Udall: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project Administration. **J. Brandt:** Methodology, Software, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **G. Manchanda:** Methodology, Software, Writing – original draft, Writing – review & editing. **A. Arulanandan:** Software, Writing – original draft, Writing – review & editing. **J. Clark:** Validation, Resources, Writing – review & editing. **J. Lange:** Writing – review & editing. **R. O'Shaughnessy:** Conceptualization, Methodology, Resources, Writing – original draft, Writing – review & editing, Supervision, Project Administration, Funding acquisition. **L. Cadonati:** Resources, Supervision, Project Administration, Funding acquisition.

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.

Data availability

Data for repeating this inference is available at the Gravitational Wave Open Science Center (https://www.gw-openscience.org/)

Acknowledgments

The authors would like to thank Daniel Williams and Colm Talbot for insightful comments and review of the manuscript.

This material is based upon work supported by the National Science Foundation, United States of America under Grant Nos PHY-2012057, PHY-1841475, and PHY-2110481. The authors are grateful for computational resources provided by the LIGO-Laboratory, Leonard E Parker Center for Gravitation, Cosmology and Astrophysics at the University of Wisconsin-Milwaukee, and Inter-University Center for Astronomy & Astrophysics (IUCAA), Pune, India, and supported by National Science Foundation, United States of America Grants PHY-1626190, PHY-1700765, HY-0757058 and PHY-0823459. This research was done using resources provided by the Open Science Grid (Pordes et al., 2007; Sfiligoi et al., 2009), which is supported by the National Science Foundation, United States of America award #2030508. This research has made use of data, software and/or web tools obtained from the Gravitational Wave Open Science Center (https://www.gw-openscience.org/), a service of LIGO Laboratory, the LIGO Scientific Collaboration and the Virgo Collaboration. LIGO Laboratory and Advanced LIGO are funded by the United States National Science Foundation (NSF) as well as the Science and Technology Facilities Council (STFC) of the United Kingdom, the Max–Planck-Society (MPS), and the State of Niedersachsen/Germany for support of the construction of Advanced LIGO and construction and operation of the GEO600 detector. Additional support for Advanced LIGO was provided by the Australian Research Council. Virgo is funded, through the European Gravitational Observatory (EGO), by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale di Fisica Nucleare (INFN) and the Dutch Nikhef, with contributions by institutions from Belgium, Germany, Greece, Hungary, Ireland, Japan, Monaco, Poland, Portugal, Spain . This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation, United States of America. This work carries LIGO document number P2100347.

References

LIGO Scientific Collaboration, Aasi, J., Abbott, B.P., Abbott, R., Abbott, T., Abernathy, M.R., Ackley, K., Adams, C., Adams, T., Addesso, P., et al., 2015. Advanced LIGO. Classical Quantum Gravity 32 (7), 074001. doi:10.1088/0264-9381/32/7/074001, arXiv:1411.4547 [gr-qc].

The LIGO Scientific Collaboration, the Virgo Collaboration, Abbott, B.P., Abbott, R., Abbott, T.D., Abraham, S., Acernese, F., Ackley, K., Adams, C., Adya, V.B., et al., 2020a. GW190412: Observation of a binary-black-hole coalescence with asymmetric masses. Phys. Rev. D 102 (4), 043015.

The LIGO Scientific Collaboration, the Virgo Collaboration, Abbott, B.P., Abbott, R., Abbott, T.D., Abraham, S., Acernese, F., Ackley, K., Adams, C., Adya, V.B., et al., 2020b. GWTC-2: Compact binary coalescences observed by LIGO and virgo during the first half of the third observing run. Available As LIGO-P2000061.

The LIGO Scientific Collaboration, the Virgo Collaboration, Abbott, B.P., Abbott, R., Abbott, T.D., Abraham, S., Acernese, F., Ackley, K., Adams, C., Adya, V.B., et al., 2020c. Properties and astrophysical implications of the 150 Msun binary black hole merger GW190521. doi:10.3847/2041-8213/aba493, arXiv e-prints, arXiv:2009.01190 [astro-ph.HE].

The LIGO Scientific Collaboration, the Virgo Collaboration, Abbott, B.P., Abbott, R., Abbott, T.D., Abraham, S., Acernese, F., Ackley, K., Adams, C., Adya, V.B., et al., 2021a. GWTC-2.1: Deep extended catalog of compact binary coalescences observed by LIGO and virgo during the first half of the third observing run. Available As LIGO-P2100063.

The LIGO Scientific Collaboration, the Virgo Collaboration, Abbott, B.P., Abbott, R., Abbott, T.D., Abraham, S., Acernese, F., Ackley, K., Adams, C., Adya, V.B., et al., 2021b. GWTC-3: Compact binary coalescences observed by LIGO and virgo during the second half of the third observing run. arXiv:2111.03606, In Preparation Available As LIGO-P2000318.

The LIGO Scientific Collaboration, The Virgo Collaboration, Abbott, B.P., Abbott, R., Abbott, T.D., Acernese, F., Ackley, K., Adams, C., Adams, T., Addesso, P., et al., 2019a. GWTC-1: A gravitational-wave transient catalog of compact binary mergers observed by LIGO and virgo during the first and second observing runs. Phys. Rev. X 9 (3), 031040.

The LIGO Scientific Collaboration, the Virgo Collaboration, Abbott, B.P., Abbott, R., Abbott, T.D., Acernese, F., Ackley, K., Adams, C., Adams, T., Addesso, P., et al., 2019b. Properties of the binary neutron star merger GW170817. Phys. Rev. X 9 (1), 011001. doi:10.1103/PhysRevX.9.011001, arXiv:1805.11579 [gr-qc].

Abbott, B., et al., (The LIGO Scientific Collaboration and the Virgo Collaboration), 2016a. Observation of gravitational waves from a binary black hole merger. Phys. Rev. Lett. 116, 061102.

Abbott, B., et al., (LIGO Scientific Collaboration and the Virgo Collaboration), 2016b. Prospects for localization of gravitational wave transients by the advanced LIGO and advanced virgo observatories. Living Rev. Relativ. 19, doi:10.1007/lrr-2016-1, arXiv:1304.0670 [gr-qc].

Abbott, R., et al., LIGO Scientific, KAGRA, VIRGO Collaboration, 2021c. Observation of gravitational waves from two neutron star–black hole coalescences. Astrophys. J. Lett. 915 (1), L5. doi:10.3847/2041-8213/ac082e, arXiv:2106.15163.

Abbott, B., et al., (The LIGO Scientific Collaboration), 2021d. Open data from the first and second observing runs of Advanced LIGO and Advanced Virgo. SoftwareX 13, 100658.

Accadia, T., et al., 2012. Virgo: a laser interferometer to detect gravitational waves. J. Instrum. 7 (03), P03012.

Acernese, F., et al., VIRGO Collaboration Collaboration, 2015. Advanced Virgo: a second-generation interferometric gravitational wave detector. Classical Quantum Gravity 32 (2), 024001. doi:10.1088/0264-9381/32/2/024001, arXiv:1408.3978 [gr-qc].

Virgo Collaboration, Acernese, F., et al., 2021. Calibration of Advanced Virgo and reconstruction of detector strain h(t) during the Observing Run O3. doi:10.1088/1361-6382/ac3c8e, arXiv:2107.03294 [gr-qc].

Adams, T., Buskulic, D., Germain, V., Guidi, G., Marion, F., Montani, M., Mours, B., Piergiovanni, F., Wang, G., 2015. Low-latency analysis pipeline for compact binary coalescences in the advanced gravitational wave detector era. Classical Quantum Gravity 33.

Ashton, G., Hübner, M., Lasky, P.D., Talbot, C., Ackley, K., Biscoveanu, S., Chu, Q., Divakarla, A., Easter, P.J., Goncharov, B., Vivanco, F.H., Harms, J., Lower, M.E., Meadors, G.D., Melchor, D., Payne, E., Pitkin, M.D., Powell, J., Sarin, N., Smith, R.J.E., Thrane, E., 2019. Bilby: A user-friendly Bayesian inference library for gravitational-wave astronomy. Astrophys. J. Suppl. Ser. 241 (2), 27.

Bockelman, B., Livny, M., Lin, B., Prelz, F., 2020. Principles, technologies, and time: The translational journey of the HTCondor-CE. J. Comput. Sci. doi:10.1016/j.jocs.2020.101213.

Brown, D.A., Vahi, K., Taufer, M., Welch, V., Deelman, E., A. Barba, L., K. Thiruvathukal, G., 2021. Reproducing GW150914: The first observation of gravitational waves from a binary black hole merger. Comput. Sci. Eng. 23 (2), 73–82.

Cannon, K., Caudill, S., Chan, C., Cousins, B., Creighton, J.D., Ewing, B., Fong, H., Godwin, P., Hanna, C., Hooper, S., Huxford, R., Magee, R., Meacher, D., Messick, C., Morisaki, S., Mukherjee, D., Ohta, H., Pace, A., Privitera, S., de Ruiter, I., Sachdev, S., Singer, L., Singh, D., Tapia, R., Tsukada, L., Tsuna, D., Tsutsui, T., Ueno, K., Viets, A., Wade, L., Wade, M., 2021. GstLAL: A software framework for gravitational wave discovery. SoftwareX 14, 100680.

Cook, S., 2012. CUDA Programming: A Developer's Guide To Parallel Computing with GPUs, first ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Davis, D., Areeda, J.S., Berger, B.K., Bruntz, R., Effler, A., Essick, R.C., Fisher, R.P., Godwin, P., Goetz, E., Helmling-Cornell, A.F., et al., 2021. LIGO detector characterization in the second and third observing runs. Classical Quantum Gravity 38 (13), 135014.

Essick, R., Godwin, P., Hanna, C., Blackburn, L., Katsavounidis, E., 2020. iDQ: Statistical inference of non-gaussian noise with auxiliary degrees of freedom in gravitational-wave detectors. Mach. Learn.: Sci. Technol. 2 (1), 015004.

Fajardo, E.M., Dost, J.M., Holzman, B., Tannenbaum, T., Letts, J., Tiradani, A., Bockelman, B., Frey, J., Mason, D., 2015. How much higher can HTCondor fly? J. Phys. Conf. Ser. 664 (6), 062014.

García-Quirós, C., Colleoni, M., Husa, S., Estellés, H., Pratten, G., Ramos-Buades, A., Mateu-Lucena, M., Jaume, R., 2020. Multimode frequency-domain model for the gravitational wave signal from nonprecessing black-hole binaries. Phys. Rev. D 102 (6), 064002. doi:10.1103/PhysRevD.102.064002, arXiv:2001.10914 [gr-qc].

Lange, J., 2020. RIFT'ing the Wave: Developing and applying an algorithm to infer properties gravitational wave sources.

Lange, J., O'Shaughnessy, R., Rizzo, M., 2018. Rapid and accurate parameter inference for coalescing, precessing compact binaries. Submitted To PRD; available at arXiv:1805.10457.

Littenberg, T.B., Cornish, N.J., 2015. Bayesian inference for spectral estimation of gravitational wave detector noise. Phys. Rev. D 91, 084034.

Okuta, R., Unno, Y., Nishino, D., Hido, S., Loomis, C., 2017. CuPy: A numpy-compatible library for NVIDIA GPU calculations. In: Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS).

Ossokine, S., Buonanno, A., Marsat, S., Cotesta, R., Babak, S., Dietrich, T., Haas, R., Hinder, I., Pfeiffer, H.P., Pürrer, M., Woodford, C.J., Boyle, M., Kidder, L.E., Scheel, M.A., Szilágyi, B., 2020. Multipolar effective-one-body waveforms for precessing binary black holes: Construction and validation. Phys. Rev. D 102 (4), 044055. doi:10.1103/PhysRevD.102.044055, arXiv:2004.09442 [gr-qc].

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., 2007. The open science grid. In: J. Phys. Conf. Ser., Vol. 78. 012057, 78.

Sfiligoi, I., Bradley, D.C., Holzman, B., Mhashilkar, P., Padhi, S., Wurthwein, F., 2009. The pilot way to grid resources using glideinWMS. In: 2009 WRI World Congress on Computer Science and Information Engineering, Vol. 2. pp. 428–432. doi:10.1109/CSIE.2009.950, 2.

Singer, L., et al., 2021. Gwcelery.

Sun, L., Goetz, E., Kissel, J.S., Betzwieser, J., Karki, S., Bhattacharjee, D., Covas, P.B., Datrier, L.E.H., Kandhasamy, S., Lecoeuche, Y.K., Mendell, G., Mistry, T., Payne, E., Savage, R.L., Viets, A., Wade, M., Weinstein, A.J., Aston, S., Cahillane, C., Driggers, J.C., Dwyer, S.E., Urban, A., 2021. Characterization of systematic error in Advanced LIGO calibration in the second half of O3. arXiv:2107.00129 [astro-ph.IM].

Thain, D., Tannenbaum, T., Livny, M., 2005. Distributed computing in practice: the condor experience. Concurrency - Pract. Exp. 17 (2–4), 323–356.

Usman, S.A., Nitz, A.H., Harry, I.W., Biwer, C.M., Brown, D.A., Cabero, M., Capano, C.D., Dal Canton, T., Dent, T., Fairhurst, S., Kehl, M.S., Keppel, D., Krishnan, B., Lenon, A., Lundgren, A., Nielsen, A.B., Pekowsky, L.P., Pfeiffer, H.P., Saulson, P.R., West, M., Willis, J.L., 2016. The PyCBC search for gravitational waves from compact binary coalescence. Classical Quantum Gravity 33 (21), 215004. doi:10.1088/0264-9381/33/21/215004, arXiv:1508.02357 [gr-qc].

Vahi, K., Rynge, M., Papadimitriou, G., Brown, D.A., Mayani, R., da Silva, R.F., Deelman, E., Mandal, A., Lyons, E., Zink, M., 2019. Custom execution environments with containers in pegasus-enabled scientific workflows. arXiv:1905.08204 [cs.DC].

Veitch, J., Raymond, V., Farr, B., Farr, W., Graff, P., Vitale, S., Aylott, B., Blackburn, K., Christensen, N., Coughlin, M., Del Pozzo, W., Feroz, F., Gair, J., Haster, C.-J., Kalogera, V., Littenberg, T., Mandel, I., O'Shaughnessy, R., Pitkin, M., Rodriguez, C., Röver, C., Sidery, T., Smith, R., Van Der Sluys, M., Vecchio, A., Vousden, W., Wade, L., 2015. Parameter estimation for compact binaries with ground-based gravitational-wave observations using the LALInference software library. Phys. Rev. D 91, 042003.

Williams, D., Veitch, J., Chiofalo, M.L., Schmidt, P., Udall, R.P., Vajpeji, A., Hoy, C., 2022. Asimov: A framework for coordinating parameter estimation workflows. arXiv e-prints arXiv:2207.01468 [gr-qc].

Wysocki, D., O'Shaughnessy, R., Lange, J., Fang, Y.-L.L., 2019. Accelerating parameter inference with graphics processing units. Phys. Rev. D 99 (8), 084026. doi:10.1103/PhysRevD.99.084026, arXiv:1902.04934 [astro-ph.IM].