Contents

A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis

Duncan A. Brown[1] Patrick R. Brady[2], Alexander Dietz[3], Junwei Cao[4], Ben Johnson[5], and John McNabb[6]

[1] LIGO Laboratory, California Institute of Technology, Pasadena, CA 91125
dbrown@ligo.caltech.edu
[2] Department of Physics, University of Wisconsin–Milwaukee, P.O. Box 413, Milwaukee, WI 53201 patrick@gravity.phys.uwm.edu
[3] Department of Physics, Louisiana State University, Baton Rouge, LA 70803
dietz@phys.lsu.edu
[4] LIGO Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139
jcao@ligo.mit.edu
[5] LIGO Hanford Observatory, Richland, WA 99352
bjohnson@ligo-wa.caltech.edu
[6] The Pennsylvania State University, University Park, PA 16802
mcnabb@gravity.psu.edu

Keywords: Gravitational wave data analysis, signal processing, data access, grid computing.

1.1 Introduction

Modern scientific experiments acquire large amounts of data which must be analyzed in subtle and complicated ways to extract the best results from the data. The Laser Interferometer Gravitational Wave Observatory (LIGO) is an ambitious effort to detect gravitational waves produced by violent events in the universe, such as the collision of two black holes, or the explosion of supernovae [12, 25]. The experiment records approximately 1 TB of data per day which is analyzed by scientists in a collaboration which spans four continents. LIGO and distributed computing have grown-up side by side over the past decade, and the analysis strategies adopted by LIGO scientists have been strongly influenced by the increasing power of tools to manage distributed computing resources and the workflows to run on them. In this chapter, we use LIGO as an application case-study in workflow design and implementation. The software architecture outlined here has been used with great efficacy to analyze LIGO data [18–21] using dedicated computing facilities operated by the LIGO

Scientific Collaboration, i.e. the LIGO Data Grid. It is just the first step, however. Workflow design and implementation lies at the interface between computing and traditional scientific activities. In the conclusion, we outline a few directions for future development and provide some long term vision for applications related to gravitational-wave data analysis.

1.2 Gravitational waves

Although Einstein predicted the existence of gravitational waves in 1916, the challenge in directly observing them is immense because of the extremely weak coupling between matter and gravitation. Small amounts of slowly moving electric charge can easily produce detectable radio waves, but the generation of detectable amounts of gravitational radiation requires extremely massive, compact objects, such as black holes, to be moving at speeds close to the speed of light. The technology to detect the waves on Earth only became practical in the last decade of the twentieth century. The detection of gravitational waves will open a new window on the universe and allow us to perform unprecedented tests of general relativity. Almost all of our current knowledge about the distant universe comes from observations of electromagnetic waves, such as light, radio and X-ray. Gravitational waves, unlike electromagnetic waves, travel through matter and dust in the universe unimpeded. They can be used to see deep into the cores of galaxies or probe the moment when space and time came into being in the Big Bang.

Gravitational waves are ripples in the fabric of spacetime; their effect on matter is to stretch it in one direction and squeeze it in the perpendicular direction. To detect these waves, LIGO uses three laser interferometers located in the Unites States. Two interferometers are at the Hanford Observatory in south eastern Washington and one is at the Livingston Observatory in southern Louisiana. The purpose of the multiple detectors is to better discriminate signal from noise, as a gravitational wave signal should be detectable in all three interferometers. Each interferometer consists of a vacuum pipe arranged in the shape of an L with 4 kilometer arms. At the vertex of the L and at the end of each of its arms are mirrors that hang from wires. Laser beams traversing the vacuum pipes accurately measure the distance between the mirrors in the perpendicular arms. By measuring the relative length of the two arms, LIGO can measure the effect of gravitational waves. These changes in length are minute, typically 10^{-19} meters over the 4 kilometer arm; musch less than the size of a proton. To measure such small distances requires ultra-stable lasers and isolation of the mirrors from any environmental disturbances. Any difference in the lengths of the arms, due to detector noise or gravitational waves, is detected as a change in the amount of light falling on a photo-detector at the vertex of the L. Figure 1.1 shows a schematic diagram of a LIGO detector. In a perfect detector and in the absence of a gravitational wave, no light would fall on the photo-detector. In practice, however, random

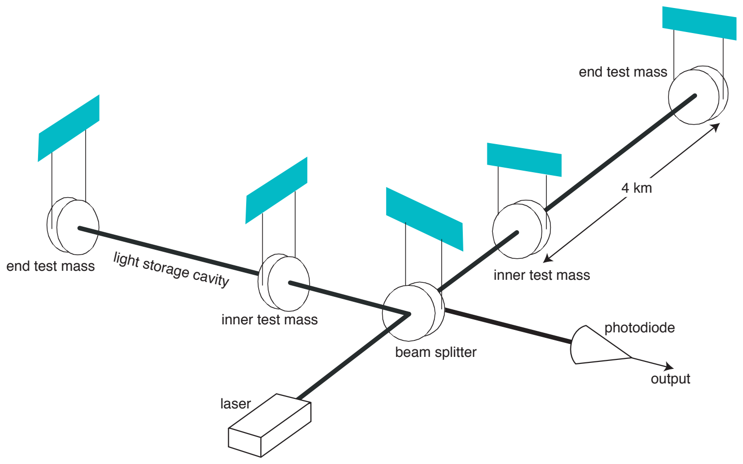

Fig. 1.1: Schematic diagram of a LIGO detector. Laser light is incident on a partially reflective mirror or beam-splitter. Half the light is transmitted into one arm of the interferometer and half is reflected into the other arm. The light in each arm resonates between two mirrors which act as test masses and change position in response to a gravitational wave. The light is recombined at the beam-splitter and the light incident on the photodiode contains information about the position of the mirrors, and hence about any gravitational waves incident on the detector.

fluctuations in the interferometer cause some light to fall on the detector. Among other sources, these fluctuations come from seismic noise from ground motion coupling into the mirrors, thermal noise from vibrations in the mirrors and their suspensions, and shot noise due to fluctuations in the photons detected by the photo-detector. LIGO data analysis is therefore a classic problem in signal processing: determining if a gravitational wave signal is present in detector noise.

Data from the LIGO detectors is analyzed by the LIGO Scientific Collaboration (LSC), an international collaboration scientists. The searches for gravitational waves in LIGO data fall broadly into four classes: compact binary inspiral, continuous waves from rotating neutron stars, unmodeled burst sources, and stochastic gravitational waves backgrounds. In this chapter we focus on the workflows used in the search for gravitational waves from compact binary inspirals. For details on the other searches we refer the reader to [25].

The gravitational waves arising from coalescing compact binary systems consisting of binary neutron stars and black holes are one of the best understood sources for gravitational wave detectors such as LIGO [41]. Neutron stars and black holes are the remnants produced by the collapse of massive stars when they reach the end of their life. If two stars are in a binary sys-

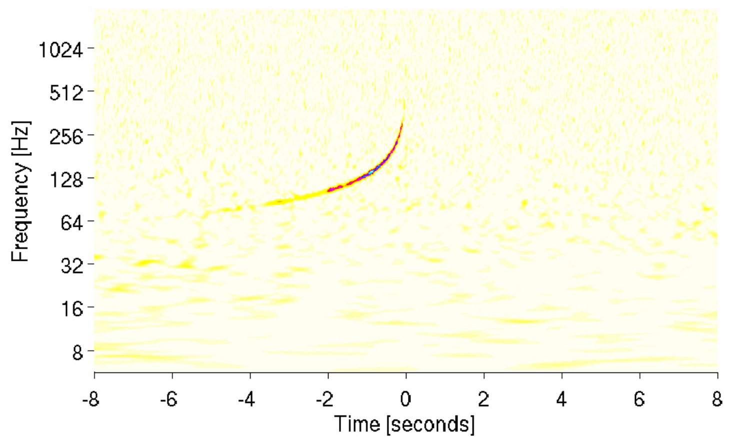

Fig. 1.2: A time-frequency spectrogram of a simulated binary inspiral signal. The waveform increases in amplitude and frequency as time increases. The well defined shape of the waveform makes matched filtering a suitable data analysis technique.

tem, the compact bodies orbit around each other and lose energy in the form of gravitational waves. The loss of energy causes their orbit to shrink and their velocities to increase. The characteristic "inspiral" signal emitted increases in frequency and amplitude, until the bodies finally plunge toward each other and coalesce, terminating the waveform. Fig. 1.2 shows a time-frequency spectrogram of a simulated inspiral signal. It is expected that there will be approximately one binary neutron star coalescence every three years in the volume of the universe accessible to LIGO [38].

The shape of the inspiral waveform depends on the masses of the binary components. When both components are below approximately three solar masses, the waveform is well modeled by theoretical calculations and we can use matched filtering to find the signals in detector noise. For higher mass waveforms, such as black hole binaries, uncertainties in the waveforms grow, but in practice we may continue to use matched filtering, albeit with a modified template family [27, 28]. These templates are not exact representations of the signals, but are designed to capture the essential features of the waveforms. The first science run of LIGO focused attention on the search for binary neutron stars [18]. The second science run refined the binary neutron star search [19] and extended the analysis to include searches for binary black hole systems with higher masses [21] and sub-solar-mass binary black hole systems which may be components of the Milky Way Halo [20].

Analysis of the LIGO data for binary inspirals is performed using the LIGO Data Grid (LDG) [10]. In this chapter, we describe the LDG infrastructure, the software used to construct data analysis workflows for the LDG, and the components and execution of the inspiral analysis pipeline. Finally we discuss the use of these tools by other gravitational wave searches, and the extension of the workflows to other Grids, such as the Open Science Grid (OSG) [13].

1.3 The LIGO Data Grid Infrastructure

LSC scientists conducting gravitational wave data analysis need to analyze many terabytes of data. The scientists have access to a large number of distributed computing resources, including resources external to the collaboration. To fully leverage the distributed resources in an integrated and seamless way, infrastructure and middleware have been deployed to structure the resources as a Grid. The LIGO Data Grid infrastructure includes the LSC Linux clusters, the networks that interconnect them to each other, grid services running on the LSC Linux clusters, a system for replicating LIGO data to LSC computing centers, DOE Grids certificate authority authentication [3], and a package of client tools and libraries that allow LSC scientists to leverage the LIGO Data Grid services.

The LDG hardware consists of Linux clusters for data analysis and Linux and Sun SPARC Solaris servers, used for data replication and metadata services. The hardware is distributed between the LIGO observatories, the LIGO Laboratories at the California Institute of Technology (Caltech), the Massachusetts Institute of Technology (MIT), and various LSC member institutions, as detailed below. The middleware software that supports Grid services and users is known as the LDG server package. The LDG server package itself is built on top of the Virtual Data Toolkit (VDT) [42] as provided by the iVDGL [9] and OSG [13] projects. A subset of the LDG server software is distributed as the LDG client package, and contains only the tools needed to access the computing clusters and discover LIGO data across the LDG. The LDG also uses some proprietary software, such as the Sun StorEdge SAM-QFS [16] software and the IBM DB2 [6] database. In this section, we describe the LDG hardware and software infrastructures in more detail.

1.3.1 Management of the raw detector data

The LIGO detectors are sensitive to gravitational waves with frequencies between approximately 40 Hz and 4 kHz. The output signal from each of the three detectors is digitized as a 16 bit signal at sample rate of $16,384$ Hz. In addition to the output photo diode signal, many other detector data channels are recorded at various sample rates between 8 Hz and $16,384$ Hz. These channels monitor the performance of the detector and its environment. The total output data rate of the observatories is 8 MByte per second for Hanford and 4 MByte per second for Livingston. The many channels are written to a high-performance file system, each individual file or *frame* containing 32 seconds of data. Approximately $10,000$ frame files are written per day at each observatory.

Distribution of this data is managed by the LIGO Data Replicator (LDR) [11], which provides robust replication and data discovery services. The LDR service is built on top of the Globus Replica Location Service (RLS) [30] , Globus GridFTP [22], and a metadata catalog service. Each of these services is deployed separately from the other services in the LDG server package. Together these services are used for replicating data. Data at the observatories are published into LDR and then replicated to the LIGO Laboratory at Caltech, which is responsible for permanent data storage and archival of data. Other LDG sites deploy LDR to replicate particular subsets of LIGO data to the local site for data analysis. The subsets of LIGO data that are replicated can be configured by each site's local policy and each site stores the data in accordance with its own local policies in terms of the directory structure. Note that the LDR service replicates data in bulk to sites, independently of the demands of any particular data analysis job. In order to execute analysis workflows LSC scientists need to be able to discover the location of specific LIGO data files across the LIGO Data Grid. The *LSCdataFind* tool included in the LDG client package allows LSC scientists to discover LIGO data based on metadata about the LIGO data rather then based on file names. Typical metadata attributes used for finding LIGO data include a start and end time describing the epoch of data to be analyzed, the observatory at which the data was collected, and the class of LIGO data files (different classes or frame types contain different sets of data channels from the detectors).

The LSCdataFind tool by default returns a list of physical file names (PFNs) or URLs for the location of LIGO data files at a particular LDG site. These PFNs can then be used directly by tools building a LIGO workflow, tailoring it for use at that particular site. In order to support the more sophisticated planning of the LIGO workflows detailed below, LSCdataFind also supports returning only the logical file names (LFNs) of the data files meeting the user's metadata constraints. The LFNs are just the simple file names and do not contain any location information.

1.3.2 Management of detector metadata

In addition to the access to the raw detector data, LSC scientists need additional metadata, known as *data quality information* which describe the state of the interferometers, when the data is suitable for analysis, and records information about periods of unusual behavior. This metadata is stored in the LSC segment database which allows storage, retrieval and replication of the data. The segment database uses the IBM DB2 database to provide the underlying relational database engine. The publication scripts used to publish the data into LDR also publish detector state information into the segment database.

The segment databases at Caltech, and the observatories are connected together by low latency peer-to-peer database replication, using the "Q-replication" service provided by DB2. Any metadata inserted at one of the three databases will be replicated to the other two databases with a latency of a few seconds to a couple of minutes. Replication time varies depending on the load on the databases. IBM WebSphere MQ [7] is used as the transport layer for replication between the databases. Message queues are set up between each of the servers that take part in the replication and these are used by the replication programs to send and receive data and control messages.

Client and server tools written on top of the LDG server middleware allow scientists to connect to the database, query information and insert new metadata based on detector characterization investigations. Segment discovery services are provided by the *LSCsegFind* server which runs at each site and responds to user requests for segment and data quality information. It constructs the SQL needed to service the user's request, executes the query on the database and returns the results to the user. The client and server communicate over a Globus GSI [29] authenticated connection. The server runs on the same machine as the DB2 database, and queries can be issued by remote clients, which are distributed as part of the LDG client bundle.

Metadata is exchanged in the LSC as XML data, with the LSC-specific schema called LIGO lightweight XML. The Lightweight Database Dumper (LDBD) provides a generic interface between the segment database and LIGO lightweight XML representations of table data in the database. The LDBD server can parse the contents of a LIGO lightweight XML document containing table data and insert it into the database. It can also execute SQL queries from a client and return the results as LIGO lightweight XML data. Data quality information is generated as LIGO lightweight XML by various data monitoring tools and inserted via the LDBD server. This generic framework allows construction of metadata services specific to the various requirements of gravitational wave data analysis. Again, communication between the client and server is performed over a GSI-authenticated socket connection. The server runs on the same machine as the DB2 database, and queries can be issued by remote clients. The LDBD server is also capable of inserting LFN to PFN maps into an RLS server, if desired, to allow metadata to be associated with specific files.

1.3.3 Computing resources

LSC scientists have access to a number of computing resources on which to analyze LIGO data. Some resources are dedicated Linux clusters at LSC sites, others are Linux clusters available via LSC partnership in large Grid collaborations such as the international Virtual Data Grid Laboratory (iVDGL) [9] and its successor the Open Science Grid [13], and still other resources are available via more general arrangements with the host institution. The vast majority of available computing resources are Intel [8] or AMD [1] based clusters running some version of the Linux operating system.

LSC Linux clusters

The LSC itself has available as dedicated computing resources Linux clusters hosted at the LIGO observatories at Hanford and Livingston, at the LIGO host institutions Caltech and MIT [12], and at LSC computing sites hosted at the Pennsylvania State University (PSU) [14] and the University of Wisconsin–Milwaukee (UWM) [17]. In addition there are Linux clusters dedicated for gravitational wave data analysis made available by the British–German GEO-600 [43] gravitational wave detector, which is also a member of the LSC.

Each dedicated LSC Linux cluster and its related data storage hardware is categorized as a Tier 1, 2, or 3 site depending (in a rough way) on the amount of computing power and data storage capacity available at the site. The LIGO Caltech Linux cluster, with over 1.2 teraflops (TFlop) of CPU and 1500 terabytes (TB) of data storage, serves as the Tier 1 site for the collaboration. All LIGO data is archived and available at the Tier 1 site. The detector sites at Hanford and Livingston, although the LIGO data originates there, are considered to be Tier 2 sites. The Hanford site has available 750 gigaflops (GFlop) of CPU and 160 TB of data storage while the Livingston site has available 400 GFlops of CPU and 150 TB of data storage. The LIGO MIT site is also considered a Tier 2 site with 250 GFlops of CPU and 20 TB of data storage. The PSU and UWM sites are operated as Tier 2 sites. The PSU site includes 1 TFlop of CPU and 35 TB of storage. The UWM site has operated in the past with 300 GFlops of CPU and 60 TB of storage, although it is currently being upgraded to 3 TFlops and 350 TB of storage.

Each of the Linux clusters within the LIGO Data Grid deploy a set of standard Grid services including Globus GRAM [37] for submitting jobs and resource management, a Globus GridFTP server for access to storage, and a GSI-enabled OpenSSH server [5] for login access via digital certificate credentials. The middleware software that supports these and other Grid services is deployed using the LDG server package.

Other computing resources

Through LSC membership in large Grid computing projects and organizations, LSC scientists have access to a large number of computing resources outside of the dedicated LSC computing resources. The LSC was a founding contributor to iVDGL and much of the development and prototyping of the effort described here was done as part of an effort to allow LSC scientists to leverage iVDGL resources not owned by the LSC. In particular the initial prototyping of the LIGO inspiral workflow management that leverages the use of Condor DAGMan (see Chapter **??** and reference [31]) and Pegasus (see chapter **??** and references [33] [34] [35]) was driven by the desire to leverage the

Grid3+ [4] resources made available by the iVDGL collaboration. The more recent work done to run LIGO inspiral workflows on non-LSC resources is targeted at running on the Open Science Grid. In addition LSC scientists (in particular those running inspiral workflows) have access to the large computing resources from the Center for Computation and Technology at Louisiana State University [2].

1.3.4 Batch processing

All of the LSC Linux clusters, with the exception of the cluster at PSU, use Condor (see Chapter **??**) as the local batch scheduler. As discussed in detail below, this has allowed LSC scientists to begin developing complex workflows that run on a single cluster and which are managed by Condor DAGMan. To run workflows across LSC clusters running Condor and leverage geographically distinct resources as part of a single workflow the LSC has investigated using Condor-only solutions such as Condor Flocking [36].

The Linux clusters at PSU and LSU, however, use the Portable Batch System (PBS) [15] for managing batch jobs, and since these resources represent a significant fraction of the resources available to LSC scientists it is important that the workflows be able to also leverage those resources. In addition a majority of the resources available outside the LDG use a tool other then Condor for managing compute jobs. While recent development work from the Condor group involves providing access to non-Condor managed resources directly from a Condor-only environment, the workflow management work described here has focused on using a blended approach that involves tools beyond Condor and Condor DAGMan.

1.3.5 LIGO Data Grid client package

LIGO Data Grid users install the LDG client package on their workstation. The LDG client package is also built on top of the VDT but only includes a subset of the client tools and libraries. No Grid services are deployed as part of the client package. In addition to the client tools from the VDT a number of client tools specific for use in creating and managing LIGO workflows are included in the client package. The most significant of these tools are the LSCdataFind and LSCsegFind tools used for data discovery across the LIGO Data Grid.

1.4 Constructing Workflows with the Grid/LSC User Environment

In the previous section we described the hardware and middleware infrastructure available to LSC scientists to analyze LIGO data. In this section, we describe the Grid/LSC User Environment (Glue), a toolkit developed to allow construction of gravitational-wave data analysis workflows. These workflows can be executed on LSC Linux clusters using the Condor DAGMan workflow execution tool or planned and executed on wider grids, such as the OSG, using the Pegasus workflow planner, Condor DAGMan and Globus GRAM.

1.4.1 Overview of LIGO workflows

LIGO data analysis is often referred to as "embarrassingly parallel" meaning that, although huge quantities of data must be analyzed over a vast parameter space of possible signals, parallel analysis does not require interprocess communication. Analysis can be broken down into units that perform specific tasks which are implemented as individual programs, usually written in the C programming language or the Matlab processing language/environment. Workflows may be parallelized by splitting the full parameter space into smaller blocks, or parallelizing over the time intervals being analyzed. The individual units are chained together to form a data analysis pipeline. The pipeline starts with raw data from the detectors, executes all stages of the analysis and returns the results to the scientist. The key requirements that the software used to construct and execute the pipelines must satisfy are:

1. Ensure that all data is analyzed and the various steps of the workflow are executed in the correct sequence.
2. Automate the execution of the workflow as much as possible.
3. Provide a flexible pipeline construction toolkit for testing and tuning workflows.
4. Allow easy, automated construction of complex workflows to analyze large amounts of data.
5. Have a simple reusable infrastructure which is easy to debug.

In order to satisfy the first two requirements, we implement a data analysis pipeline as a directed acyclic graph (DAG) that describes the workflow (the order that the programs must be called to perform the analysis from beginning to end). A DAG description of the workflow can then be submitted to a batch processing system on a computing resource, or to a workflow planner. The pipeline construction software must maintain an internal representation of the DAG which can then be written out in the language which a batch processing system or a workflow planner can understand. By abstracting the representation of the workflow internally, the workflow may be written out using different syntaxes, such as a Condor DAGMan input file or the XML syntax (known as DAX) used by the Pegasus workflow planner. To simplify the construction of DAGs for gravitational wave data analysis, the LSC has developed the Grid/LSC User Environment or Glue; a collection of modules, written in the Python language, developed especially for LSC scientists to help build workflows.

The components of a DAG are its *nodes* and *edges*. The nodes are the individual analysis units and the edges are the relations between the nodes that determine the execution order. Each node is assumed to be an instance of a *job* that performs a specific task in the workflow. Glue contains three basic abstract classes that represent DAGs, jobs and nodes. The DAG class provides methods to add nodes and write out the workflow in various formats. The job class provides methods to set the name of the executable and any options or arguments common to all instances of this job in the DAG. The node class, which inherits from the job class, provides methods to set arguments specific to a node, such the start and stop time to be analyzed, or the required input files. The node class also has a method to add parent nodes to itself. The edges of the DAG are constructed by sucessive calls to `add_parent` for the nodes in the workflow. The executables to be run in the DAG read their arguments from the command line, and read and write their input from the directory they are executed in. This constraint is enforced to allow portability to grid environments, discussed below. Glue also knows about other LIGO-specific concepts, such as *science segments* (time epochs of LIGO data suitable for analysis), and the methods that are used to split these segments into *blocks*, or sub-units of science segments used to parallelize workflows. By providing iterators for these classes, it is simple to loop over segments and blocks in the construction of a workflow.

To address the specific needs of different analysis tasks, the user writes a pair of classes that describe the task to glue: a job class and a node class that inherit from the base classes. The user may extend or override the base methods to allow the pipeline construction scripts to set options particular to the task being described. In this way, the components of the workflow are abstracted, and it is straightforward to write pipeline scripts that construct complex workflows. The Glue method of constructing data analysis pipelines has been used in the binary inspiral analysis, the search for gravitational wave bursts from cosmic strings, the excess power burst analysis, and in the stochastic gravitational wave background analysis. Fig 1.3 shows how Glue is used in workflow construction, with metadata and analysis parameters taken as input, and different workflow styles written as output. Below we give an example of a script to construct a simple workflow, and Sec. 1.5 describes how this is used in practice for the inspiral analysis pipeline.

1.4.2 Constructing a workflow with Glue

In this example, an LSC scientist wishes to analyze data from a single LIGO detector through a program called *GWSearch*, which analyzes data in blocks of duration 2048 seconds. Fig. 1.4 shows the Python code necessary to construct this workflow using Glue. The user has written a pair of classes which describe the job and nodes for the GWSearch program, as described in the previous section, and the script imports them along with the pipeline generation module from Glue. The user has requested a list of times from the

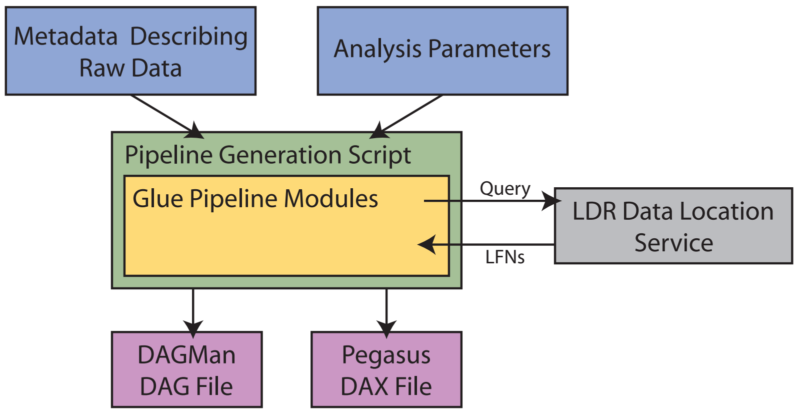

Fig. 1.3: The Glue pipeline modules are used by LSC scientists use to write pipeline generation scripts. Pipeline scripts take as input analysis parameters and metadata describing the raw data, and output workflows as DAGMan DAG files or Pegasus DAX files which can be used to execute the pipeline. If Glue is generating a Pegasus DAX, the pipeline modules can query the LDR data location service to obtain LFNs for the input data, as described in Sec. 1.4.4.

segment database that are suitable for analysis and stored them in a text file named `segments.txt`. This file contains a list of start and stop times in GPS seconds, which may vary in length between several seconds and many hours. The user's pipeline script creates a representation of these intervals using the Glue `ScienceData` class. The segments are parsed from the file by the `read` method, which is told to discard any segments shorter than 2048 seconds. The segments are then split into blocks of length 2048 seconds by the `make_chunks` method.

To construct a workflow, the script first creates a representation of the workflow itself using the `CondorDAG` class. Instances of the `LSCDataFindJob` and `GWSearchJob` classes are then created to describe the programs that will be used in the workflow. Next the script iterates over all segments in the `data` class and constructs a node in the workflow that performs an LSCdataFind job to find the data for each segment. There is then a second loop over the 2048 second blocks within each segment, and a node to execute the GWSearch program on each block. A dependency is created between the LSCdataFind and the GWSearch jobs by using the `add_parent` method of the GWSearch nodes. This ensures that the GWSearch jobs do not execute until the LSCdataFind job is complete. Finally a relation is created between the LSCdataFind jobs, so that only one job executes at a time; this is a technique used in real workflows to reduce the load on the server. The final workflow constructed by this example is shown in Fig. 1.5 for a segment file which contains segments of lengths 6144, 4192 and 4192 seconds.

```
from glue import pipeline
import gwsearch

data = pipeline.ScienceData()
data.read('segments.txt',2048)
data.make_chunks(2048)
dag = pipeline.CondorDAG('myworkflow')

datafind_job = pipeline.LSCDataFindJob()
datafind_job.add_option('data-type','raw')
previous_df = None

gwsearch_job = analysis.GWSearchJob()

for seg in data:
      df = pipeline.LSCDataFindNode()
      df.set_start(seg.start())
      df.set_end(seg.end())
      for chunk in seg:
            insp = gwsearch.GWSearchNode()
            insp.set_start(chunk.start())
            insp.set_end(chunk.end())
            insp.add_parent(df)
      if previous_df:
            df.add_parent(previous_df)
      previous_df = df

dag.write_dag()
```

Fig. 1.4: Example code showing the construction of a workflow using Glue. The input data times are read from the file `segments.txt`. For each interval in the file, an LSCdataFind job is run to discover the data and also a sequence of inspiral jobs are run to analyze the data. The workflow is written to a Condor DAG file called `myworkflow.dag` which can be executed using DAGMan.

1.4.3 Direct execution using Condor DAGMan

Once the script to generate an analysis pipeline has been written, the resulting workflow must be executed on an LSC computing cluster. As described previously, most of the LSC clusters run the Condor batch processing system. The `write_dag` method of the Glue DAG class creates a DAG in Condor DAGMan format, as well as the necessary Condor submit files to execute the jobs. DAGs for LSC data analysis range in size from a few tens of nodes to over $100,000$ nodes. The DAG written by the pipeline script is submitted to Condor, which ensures that all the nodes are executed in the correct sequence. If any node fails, for example due to transient errors on cluster nodes, a res-

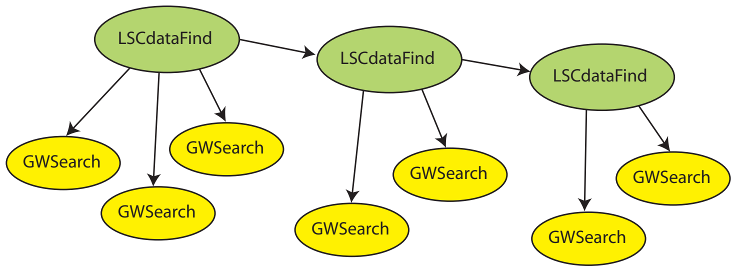

Fig. 1.5: The workflow constructed by the sample code shown in Fig 1.4. In this case there are three segments used as input, the first of which contains three 2048 second blocks, and the second and third contain two 2048 second blocks. The resulting workflow has ten nodes.

cue DAG is created containing only the nodes that failed, or were unable to execute due to failures. This rescue DAG can be resubmitted to Condor and in this was LSC scientists can ensure that all data has been correctly and completely analyzed.

1.4.4 Planning for grids with Pegasus

To complete a search for gravitational waves, it is necessary to run many large-scale Monte Carlo simulations with simulated signals added to the data. The results of these simulations are used to measure the efficiency and tune the parameters of the search. This requires a great deal of computing power and Glue has been extended to write workflows in the *abstract DAG* (DAX) format so they can be planned for grid execution with Pegasus.

When running data on the Grid, it is no longer guaranteed that the LIGO data is present on the computing cluster that the job will execute on. Glue has been modified so that when it is instructed to write a DAX it does not add any requested LSCdataFind nodes to the workflow. Instead it queries the LDR data discovery service to find the logical file names (LFNs) of the input data needed by each node and adds this information to the DAX. When the workflow is planned by Pegasus on a given list of potential grid sites, it queries the Globus RLS servers deployed on the LIGO Data Grid to determine the physical file names or URLs of the input data. Pegasus then adds transfer nodes to the workflow to stage data to sites that do not have the input data, and uses local replicas of the data on those sites that do already have the necessary input data available. In addition to the LFNs of the input data, Glue also writes the LFNs of all intermediate data products in the DAX so that Pegasus may plan the workflow across multiple sites. One of the key features of Glue is that this is transparent to the user. Once they have written their workflow generation script, they may simply add a command line switch that calls the `write_dax` method, rather than `write_dag`, and Glue will produce a DAX description of the workflow suitable for use with Pegasus.

1.5 The Inspiral Analysis Workflow

In the previous sections we have described the infrastructure of the LIGO Data Grid and the construction of workflows using Glue. In this section, we describe the use of these tools to implement the search for compact binary inspiral in LIGO data, with practical examples of the workflow.

The signal from a true gravitational wave should be present in all the LIGO detectors. It should occur at the same time in the two detectors at the Hanford observatory, and no later than the light travel time of 10 ms at the Livingston observatory. The actual time delay between observatories varies, depending where on the sky the signal originates. Triggers are said to be *coincident* if they have consistent start times. The triggers must also be in the same waveform template and may be required to pass additional tests, such as amplitude consistency. The triggers that survive all coincidence tests are the output of the inspiral analysis pipeline, and are known as *event candidates*. Further manual follow-up analysis is used to determine if the triggers are truly due to gravitational waves.

If one detector is more sensitive than the other two detectors, as was the case in the second LIGO science run, we may only wish to analyze data from the less sensitive detectors when there is a trigger in the most sensitive detector. If the detectors are equally sensitive, as is presently the case, we may wish to demand that a trigger from the matched filter is present in all three detectors before computing computationally-expensive signal-based vetoes.

1.5.1 Components of the inspiral analysis

The inspiral workflow is divided into blocks that perform specific tasks, which are summarized in Table 1.1. Each task is implemented as a separate program written in the C programming language. The core of the workflow, and the most computationally-intensive task, is the computation of the matched filter signal-to-noise ratio and a time-frequency test, known as the χ^2 veto [23, 24]. There are several other components of the workflow, however, which we describe briefly here. A detailed description of the components may be found in [26].

Data from the three LIGO detectors must first be discovered, and then split into blocks of length 2048 seconds for analysis by the inspiral program. The workflow uses the LSCdataFind program to discover the data, and the methods of the Glue pipeline module described above to subdivide the data into blocks. For each block, and for each detector, a *template bank* must be generated for the matched filtering code. The template bank is a discrete subset of the continuous family of waveforms that belong to the parameter space.

Component	Description
tmpltbank	Produces a bank of waveform parameters for use by the matched filtering code. The bank is chosen so that the loss of signal-to-noise ratio between a signal anywhere in the desired parameter space and the nearest point in the bank is less than some specified value, which is typically 3%.
inspiral	For each template in a bank, compute the matched filter and χ^2 veto algorithms on a given block of data. Generates a list of inspiral triggers, which are times when the matched filter signal-to-noise ratio and the value of the χ^2 veto exceed user-defined thresholds.
trigbank	Converts a list of triggers coming from the inspiral program into a template bank that is optimized to minimize the computational cost in a follow-up stage.
inca	Performs several tests for consistency between triggers produced by the inspiral program from analyzing data from two detectors.

Table 1.1: The components of the inspiral analysis workflow.

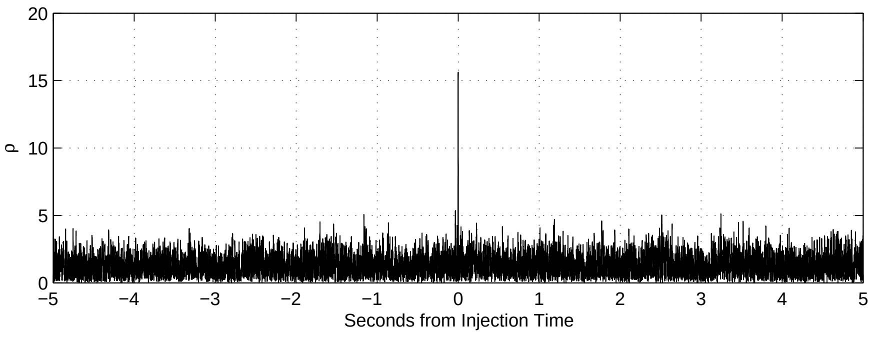

Fig. 1.6: The output of the matched filter in the presence of a simulated signal. The signal in injected into the data at time $t = 0$. The signal-to-noise ratio generated by the filter peaks at the time of the injected signal.

The placement of the templates in the bank is determined by the *mismatch* of the bank which is the maximum fractional loss of signal-to-noise ratio that can occur by filtering a true signal with component masses m_1, m_2 with the "nearest" template waveform for a system with component masses m_1', m_2'. The construction of an appropriate template bank is discussed in [39, 40].

The bank is then read in by the inspiral program which reads in the detector data and computes the output of the matched filter for each template in the bank. In the presence of a binary inspiral, the signal-to-noise ratio ρ of the matched filter will peak, as shown in Fig. 1.6. The inspiral program may also compute the χ^2 time-frequency veto, which tests that the signal-to-noise ratio has been accumulated in a manner consistent with an inspiral signal, and not the result of a "glitch" or other transient in the detector data. If the value of the signal-to-noise and χ^2 veto pass defined thresholds at any given time, the inspiral code outputs a *trigger* for this time with the parameter of the template and filter output. These triggers must then be confronted with triggers from other detectors to look for coincidences.

The trigbank program can convert a list of triggers from the inspiral program into a template bank that is optimized to minimize the computational cost of a follow-up stage. We describe the optimization in detail in section 1.5.2. The *in*spiral *c*oincidence *a*nalysis program, or inca, performs several tests for consistency between triggers produced by inspiral output from analyzing data from two or more detectors and generates event candidates.

1.5.2 Inspiral workflow applications

The second LIGO science run

In LIGO's second science run (S2) we performed a *triggered* search for primordial binary black holes and neutron stars [19, 20]. Since we require that a trigger occur simultaneously and consistently in at least two detectors located at different sites in order for it to be considered as a detection candidate, we save computational effort by analyzing data from the Livingston detector (the most sensitive detector at the time) first and then performing follow-up analyses of Hanford data only when specific triggers are found. We describe the tasks and their order of execution in this triggered search as our detection pipeline (workflow).

Figure 1.7 shows the workflow in terms of these basic tasks. Epochs of simultaneous Livingston-Hanford operation are processed differently depending on which interferometer combination is operating. Thus, there are several different sets of data: $\mathrm{L1} \cap (\mathrm{H1} \cup \mathrm{H2})$ is when the Livingston detector L1 is operating simultaneously with either the 4 km Hanford detector H1 or the 2 km Hanford detector H2 (or both)—this is all the data analyzed by the S2 inspiral analysis—while $\mathrm{L1} \cap \mathrm{H1}$ is when L1 and H1 are both operating operating, $\mathrm{L1} \cap (\mathrm{H2} - \mathrm{H1})$ is when L1 and H2 but not H1 are operating, and $\mathrm{L1} \cap \mathrm{H1} \cap \mathrm{H2}$ is when all three detectors are operating. A full L1 template bank is generated for the $\mathrm{L1} \cap (\mathrm{H1} \cup \mathrm{H2})$ data and the L1 data is filtered with inspiral. Triggers resulting from these filter operations are then used to produce triggered banks for followup filtering of H1 and/or H2 data. However, if both H1 and H2 are operating then filtering of H2 is suspended until coincident L1-H1 triggers are identified by inca. The workflow used to execute this pipeline is generated by a script called *inspiral_pipe*, which is written using the Glue library described in the previous section. The script is given the list of times suitable for analysis and generates a Condor DAG which is used to execute the pipeline. Fig. 1.8 shows a small subset of the workflow created by the pipeline generation script.

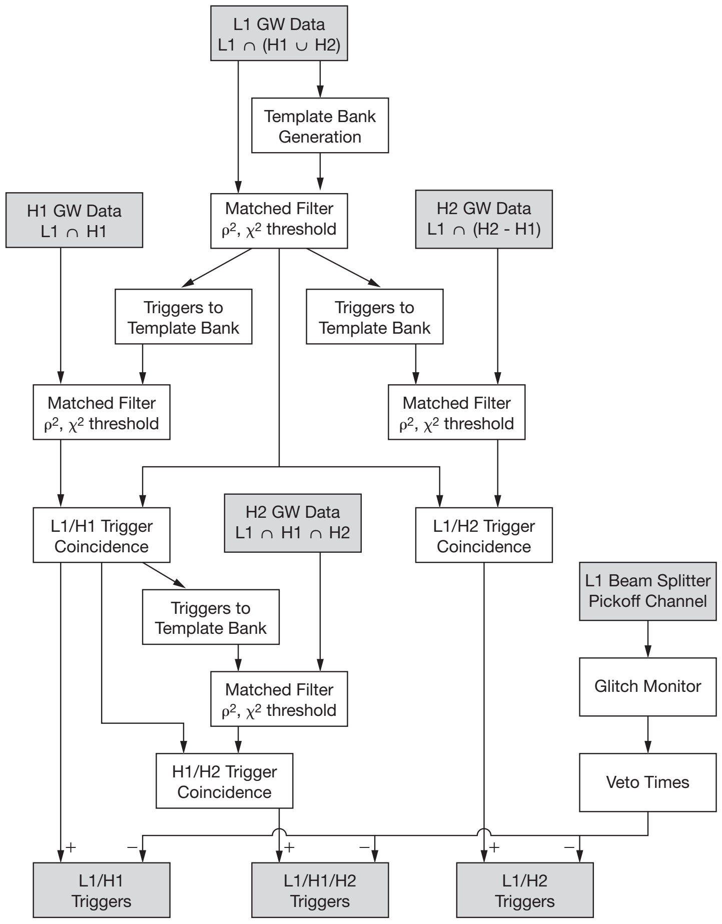

Fig. 1.7: Structure of the S2 Triggered Search Pipeline

The fifth LIGO science run

As the complexity of the analysis pipeline increases, and the amount of data to be analyzed grows, so does the size of the inspiral workflow. To illustrate this, we give a brief description of the binary neutron star search in the fifth LIGO science run (S5). The S5 run is presently under way (as of April 2006) and will record a year of coincindent data from the LIGO detectors. We will not describe the S5 inspiral pipeline in detail here, suffice it to say that the analysis uses a different workflow topology to the second science run. To analyze a small subset of S5 consisting of 1564 hours of data for binary neutron star inspirals requires a workflow with 44,537 nodes. To execute this workflow required 3000 CPU days on the LIGO Caltech cluster, which consists of 1000 2.2GHz Dual

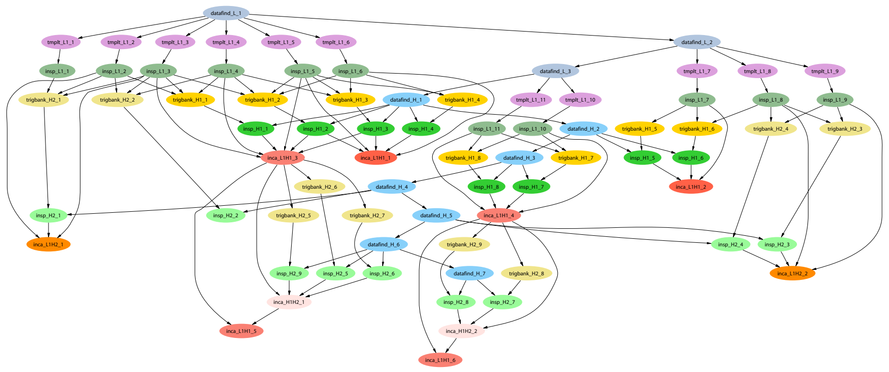

Fig. 1.8: A subset of the workflow used to analyze data from the second LIGO science run for binary inspirals. The full workflow has 6986 nodes.

Core AMD Opteron Processors. A complete analysis of this data will require approximately 3–6 additions executions of the workflow.

1.5.3 Using Pegasus to plan inspiral workflows

Since the inspiral pipeline workflows are produced using Glue, it is trivial to create Pegasus abstract DAX descriptions of the workflow (see Chapter **??**). To run the inspiral analysis on the Penn State LSC cluster, which uses PBS as the scheduler rather than Condor, a DAX is created which describes the workflow. Using this method we conducted a Monte Carlo-based computation that analyzed 10% of the data from the fourth LIGO science run (S4), a total of 62 hours of data. The DAX created by the inspiral pipeline script contained $8,040$ nodes with $24,082$ LFNs listed as input files, $7,582$ LFNs listed as intermediate data products generated by the workflow and 458 final data products. Once the DAX is planned by Pegasus, the executable concrete DAG used to execute the workflow had $12,728$ nodes, which included the jobs necessary to stage the input data to the remote cluster and transfer the output back to the users local system. Execution of the workflow took 31 hours on the PSU cluster, described in section 1.3.3.

Pegasus has also been used to parallelize inspiral workflows across multiple grid sites. For a demonstration at the SC 2004 conference a typical LIGO inspiral analysis workflow was planned using Pegasus to run across the LSC Linux clusters at Caltech and UWM as well as a Linux cluster operated by the CCT at LSU. The effort demonstrated:

1. Running a LIGO inspiral analysis workflow internally within the LIGO Data Grid.
2. Running a LIGO inspiral analysis workflow externalally to the LIGO Data Grid on the LSU resource.

3. Running across multiple types of cluster batch systems (Condor at Caltech and UWM and PBS at LSU).
4. Running at sites where LIGO data was pre-staged using the LIGO Data Replicator (the LSC sites).
5. Running at sites where LIGO data needed to be staged to the compute resource as part of the workflow (the LSU Linux cluster).

All of the work planned by Pegasus and executed across the grid sites ran to completion and all of the output was staged back to the machine from which the workflow was launched.

1.6 Concluding remarks

The workflow tools described in this chapter provide an extensible architecture for rapid workflow development and deployment and continue to be used and extended by the LIGO Scientific Collaboration. There are areas of the current framework which need to be strengthened, however, which we discuss in this section.

A key challenge is the better integration of the pipeline development tools and workflow planning middleware. The LSC has successfully used the Pegasus workflow planner to leverage computing power at remote grid sites, but there is still a substantial burden on the scientific end-user to integrate this into the execution of a workflow. There is a need to develop the interfaces between data management, planning and batch processing tools so that the use of large, distributed, grid computing resources appears to be as simple to the end-user as submitting a DAG to a single LDG cluster running Condor.

Gravitational wave detectors generate large data sets which are need to be accessed by various elements of the analysis workflows. In order to transparently execute jobs at remote locations, it is important to have seamless management of jobs and data transfer. In the work described above, Pegasus has been used to provide data staging to remote sites using GridFTP. Additional development will be needed to take advantage of grid storage management technologies, such as dCache [32], and to accomodate any storage constraints that may be placed by non-LDG computing centers.

LIGO workflows also typically consist of a mixture of computationally intensive and short running jobs. This information is not presently taken into account when planning a workflow. The Glue environment could be extended to provide additional job metadata to the workflow planner to allow it to make better use of available resources. For example, the user may only wish to run long running jobs on remote grid sites, and execute short follow-up jobs locally. Furthermore, only minimal information about the grid on which the workflow is to executed is presently incorporated at the workflow planning stage. Metadata services need to be better integrated into the workflow design and implementation to allow efficient planning and execution.

Finally, the user interfaces to all of these computing resources must be simplified if they are to become truly powerful scientific tools. Users must easily be able to monitor the activity of their jobs using simple tools like the Unix command `top`, they must be easily able access their data products or input data sets, and they must be able to prototype and deploy application workflows with ease. From the perspective of the user–an application scientist–quick and easy access to this information is of paramount importance.

Acknowledgments

We would like to thank Stuart Anderson, Kent Blackburn, Ewa Deelman, Stephen Fairhurst, Gaurang Mehta, Adam Mercer, David Meyers and Karan Vahi for comments and suggestions. This work has been supported in part by the National Science Foundation grants 0086044, 0122557, 0200852, and 0326281, and by the LIGO Laboratory cooperative agreement 0107417. Patrick Brady is also grateful to the Alfred P. Sloan Foundation and the Research Corporation Cotrell Scholars Program for Support.

References

1. AMD. `http://www.amd.com`.
2. Center for computation and technology at louisiana state university. See `http://www.cct.lsu.edu/`.
3. Doe grids certificate authority. See `http://www.doegrids.org/`.
4. Grid3+. See `http://www.ivdgl.org/grid2003`.
5. Gsi-enabled openssh. See `http://grid.ncsa.uiuc.edu/ssh/`.
6. Ibm db2. See `http://www-306.ibm.com/software/data/db2`.
7. Ibm websphere. See web site at `http://www-306.ibm.com/software/websphere/`.
8. Intel. `http://www.intel.com`.
9. international virtual data grid laboratory. See project web site at: `http://www.ivdgl.org`.
10. Ligo data grid (ldg). `http://www.lsc-group.phys.uwm.edu/lscdatagrid`.
11. Ligo data replicator. LIGO Data Replicator. See `http://www.lsc-group.phys.uwm.edu/ldr`.
12. LIGO Home Page. `http://www.ligo.caltech.edu`.
13. The open science grid consortium. http://www.opensciencegrid.org/.
14. Penn State LIGO Data Processing Center. `http://ligo.aset.psu.edu`.
15. Portable batch system. See `http://www.openpbs.org/`.
16. Sun storedge sam-qfs. Sun StorEdge SAM-QFS. See `http://www.sun.com`.
17. UWM LSC Group Medusa Cluster. `http://www.lsc-group.phys.uwm.edu/beowulf/medusa`.
18. B. Abbott et al. Analysis of ligo data for gravitational waves from binary neutron stars. *Phys. Rev.*, D69:122001, 2004.
19. B. Abbott et al. Search for gravitational waves from galactic and extra- galactic binary neutron stars. *Phys. Rev.*, D72:082001, 2005.
20. B. Abbott et al. Search for gravitational waves from primordial black hole binary coalescences in the galactic halo. *Phys. Rev.*, D72:082002, 2005.
21. B. Abbott et al. Search for gravitational waves from binary black hole inspirals in ligo data. *Phys. Rev.*, D73:062001, 2006.
22. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Data management and transfer in high-performance computational grid environments. *Parallel Computing*, 28(5):749–771, May 2002.
23. B. Allen. A chi^2 time-frequency discriminator for gravitational wave detection. *Phys. Rev.*, D71:062001, 2005.

24. B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown, and J. D. E. Creighton. Findchirp: An algorithm for detection of gravitational waves from inspiraling compact binaries. *Submitted to Phys. Rev. D.*, 2005.
25. B. C. Barish and R. Weiss. Ligo and the detection of gravitational waves. *Phys. Today*, 52 (Oct)(10):44–50, 1999.
26. D. A. Brown. Using the inspiral program to search for gravitational waves from low-mass binary inspiral. *Class. Quant. Grav.*, 22:S1097–S1108, 2005.
27. A. Buonanno, Y. Chen, and M. Vallisneri. Detecting gravitational waves from precessing binaries of spinning compact objects: Adiabatic limit. *Phys. Rev. D*, 67:104025, 2003.
28. A. Buonanno, Y. Chen, and M. Vallisneri. Detection template families for gravitational waves from the final stages of binary-black-hole inspirals: Nonspinning case. *Phys. Rev. D*, 67:024016, 2003.
29. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch. *IEEE Computer*, 33(12):60–66, 2000.
30. A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney. Giggle: A Framework for Constructing Scalable Replica Location Services. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–17. IEEE Computer Society Press, November 2002.
31. Condor Team. DAGMan: A Directed Acyclic Graph Manager, July 2005. `http://www.cs.wisc.edu/condor/dagman/`.
32. dCache.org. `http://www.dcache.org/`, 2006.
33. E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus : Mapping Scientific Workflows onto the Grid. In *2nd European Across Grids Conference*, Nicosia, Cyprus, 2004.
34. E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda. Mapping Abstract Complex Workflows onto Grid Environments. *Journal of Grid Computing*, 1(1):25–39, 2003.
35. E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. Katz. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13(2), November 2005.
36. D. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A worldwide flock of Condors: Load sharing among workstation clusters. *Future Generation Computer Systems*, 12:53–65, 1996.
37. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, Fall 2001.
38. V. Kalogera et al. The cosmic coalescence rates for double neutron star binaries. *Ap. J.*, 601:L179–L182, 2004.
39. B. J. Owen. Search templates for gravitational waves from inspiraling binaries: Choice of template spacing. *Phys. Rev. D*, 53:6749–6761, 1996.
40. B. J. Owen and B. S. Sathyaprakash. Matched filtering of gravitational waves from inspiraling compact binaries: Computational cost and template placement. *Phys. Rev. D*, 60:022002, 1999.

41. K. S. Thorne. Gravitational radiation. In S. W. Hawking and W. Israel, editors, *Three hundred years of gravitation*, chapter 9, pages 330–458. Cambridge University Press, Cambridge, 1987.
42. The virtual data toolkit. http://www.vdt.org.
43. B. Willke et al. The geo 600 gravitational wave detector. *Class. Quant. Grav.*, 19:1377–1387, 2002.