A Virtual Data Grid for LIGO

Ewa Deelman, Carl Kesselman
Information Sciences Institute, University of Southern California

Roy Williams
Center for Advanced Computing Research, California Institute of Technology

Albert Lazzarini, Thomas A. Prince
LIGO Experiment, California Institute of Technology

Joe Romano
Physics, University of Texas, Brownsville

Bruce Allen
Physics, University of Wisconsin

Abstract. GriPhyN (Grid Physics Network) is a large US collaboration to build grid services for large physics experiments, one of which is LIGO, a gravitational-wave observatory. This paper explains the physics and computing challenges of LIGO, and the tools that GriPhyN will build to address them. A key component needed to implement the data pipeline is a virtual data service; a system to dynamically create data products requested during the various stages. The data could possibly be already processed in a certain way, it may be in a file on a storage system, it may be cached, or it may need to be created through computation. The full elaboration of this system will allow complex data pipelines to be set up as virtual data objects, with existing data being transformed in diverse ways.

This document is a data-computing view of a large physics observatory (LIGO), with plans for implementing a concept (Virtual Data) by the GriPhyN collaboration in its support. There are three sections:

- Physics: what data is collected and what kind of analysis is performed on the data.
- The Virtual Data Grid, which describes and elaborates the concept and how it is used in support of LIGO.
- The GriPhyN Layer, which describes how the virtual data will be stored and handled with the use of the Globus Replica Catalog and the Metadata Catalog.

1 Physics

LIGO (Laser Interferometer Gravitational-Wave Observatory) [1] is a joint Caltech/ MIT project to directly detect the gravitational waves predicted by Einstein's theory of relativity: oscillations in the fabric of space-time. Currently, there are two observatories in the United States, (Washington state and Louisiana), one of which recently went through the “first lock” phase, in which initial calibration data was collected. Theoretically, gravitational waves are generated by accelerating masses, however

B. Hertberger et al. (Eds.): HPCN Europe 2001, LNCS 2110, pp. 3 - 12, 2001.

they are so weak that so far they have not been directly detected (although their indirect influence has been inferred [2]). Because of the extreme weakness of the signals, large data-computing resources are needed to extract them from noise and differentiate astrophysical gravitational waves from locally-generated interference.

Some phenomena expected to produce gravitational waves are:

- Coalescence of pairs of compact objects such as black-holes and neutron stars. As they orbit, they lose energy and spiral inwards, with a characteristic "chirp" over several minutes. Estimates predict of order one observation per year for the current generation of detectors.
- Continuous-wave signals over many years, from compact objects rotating asymmetrically. Their weakness implies that deep computation will be necessary.
- Supernova explosions — the search may be triggered by observation from traditional astronomical observatories.
- "Starquakes" in neutron stars.
- Primordial signals from the very birth of the Universe.

LIGO's instruments are laser interferometers, operating in a 4km high-vacuum cavity, and can measure very small changes in length of the cavity. Because of the high sensitivity, phenomena such as seismic and acoustic noise, magnetic fields, or laser fluctuations can swamp the astrophysical signal, and must themselves be instrumented. Computing is crucial to digitally remove the spurious signals and search for significant patterns in the resulting multi-channel time-series.

The raw data is a collection of time series sampled at various frequencies (e.g., 16kHz, 16Hz, 1Hz, etc.) with the amount of data expected to be generated and catalogued each year is in the order of tens of terabytes. The data collected represents a gravitational channel (less than 1% of all data collected) and other instrumental channels produced by seismographs, acoustic detectors etc. Analysis on the data is performed in both time and frequency domains. Requirements are to be able to perform single channel analysis over a long period of time as well as multi-channel analysis over a short time period.

1.1 Data in LIGO

Each scalar time series is represented by a sequence of 2-byte integers, though some are 4-byte integers. Time is represented by GPS time, the number of seconds since an epoch in 1981, and it is therefore a 9-digit number, possibly followed by 9 more digits for nanosecond accuracy (it will become 10 digits in Sept. 2011). Data is stored in Frame files, a standard format accepted throughout the gravitational wave community. Such a file can hold a set of time-series with different frequencies, together with metadata about channel names, time intervals, frequencies, file provenance, etc. In LIGO, the Frames containing the raw data encompass an interval of time of one second and result in about 3Mb of data.

In addition to the raw time series, there are many derived data products. Channels can be combined, filtered and processed in many ways, not just in the time domain, but also in the Fourier basis, or others, such as wavelets. Knowledge is finally extracted from the data through pattern matching algorithms; these create collections of candidate events, for example, inspiral events or candidate pulsars.

1.2 LIGO Data Objects

The Ligo Data Analysis System (LDAS) [3] is a set of component services for processing and archiving LIGO data. Services deliver, clean, filter, and store data. There is high-performance parallel computing for real-time inspiral search, storage in a distributed hierarchical way, a relational database, as well as a user interface based on the Tcl language.

The LIGO data model splits data from metadata explicitly. Bulk data is stored in Frame files, as explained above, and metadata is stored in a relational database, IBM DB2. There is also an XML format called LIGO-LW (an extension of XSIL [4]) for representing annotated, structured scientific data, that is used for communication between the distributed services of LDAS.

In general, a file may contain more than one Frame, so we define the word *FrameFile*, for a file that may contain many frames. Raw data files contain only one frame, and they are named by the interferometer that produced the data (H: Hanford, L: Livingston), then the 9-digit GPS time corresponding to the beginning of the data. There is a one or two letter indication of what kind of data is in the file, (F: full, R: reduced, T: trend, etc.). So an example of a raw data frame might be `H-276354635.F.`

For long-term archiving, rather larger files are wanted than the 3-megabyte, one second raw frames, so there are collection-based files, generally as multi-frame FrameFiles. In either case, an additional attribute is in the file name saying how many frames there are, for example `H-276354635.F.n200` would be expected to contain 200 frames.

One table of the metadatabase contains FrameSets, which is an abstraction of the FrameFile concept, recognizing that a FrameFile may be stored in many places: perhaps on tape at the observatory, on disk in several places, in deep archive.

Each frame file records the names of all of the approximately one thousand channels that constitute that frame. In general, however, the name set does not change for thousands or tens of thousands of one-second frames. Therefore, we keep a ChannelSet object, which is a list of channel names together with an ID number. Thus the catalog of frames need only store the ChannelSet ID rather than the whole set of names.

The metadatabase also keeps collections of Events. An event may be a candidate for an astrophysical event such as a black-hole merger or pulsar, or it may refer to a condition of the LIGO instrument, the breaking of a feedback loop or the RF signal from a nearby lightning strike. The generic Event really has only two variables: type and significance (also called Signal to Noise Ratio, or SNR). Very significant events are examined closely, and insignificant events used for generating histograms and other statistical reports.

1.3 Computational Aspects, Pulsar Search

LDAS is designed primarily to analyze the data stream in real time to find inspiral events, and secondarily to make a long-term archive of a suitable subset of the full LIGO data stream. A primary focus of the GriPhyN effort is to use this archive for a full-scale search for continuous-wave sources. This search can use unlimited computing resources, since it can be done at essentially arbitrary depth and detail. A major objective and testbed of the GriPhyN involvement is to do this search with *back-*

fill computation (the "SETI@home" paradigm), with high-performance computers in the physics community.

If a massive, rotating ellipsoid does not have coincident rotational and inertial axes (a time-dependent quadrupole moment), then it emits gravitational radiation. However, the radiation is very weak unless the object is extremely dense, rotating quickly, and has a large quadrupole moment. While the estimates of such parameters in astrophysically-significant situations are vague, it is expected that such sources will be very faint. The search is computationally intensive primarily because it must search a large parameter space. The principle dimensions of the search space are position in the sky, frequency, and rate of change of frequency.

The search is implemented as a pipeline of data transformations. First steps are cleaning and reshaping of the data, followed by a careful removal of instrumental artifacts to make a best estimate of the actual deformation of space-time geometry at the LIGO site. From this, increasingly specialized data products are made, and new ways to calibrate and filter the raw and refined data.

The pulsar search problem in particular can be thought of as finding features in a large, very noisy image. The image is in time-frequency space, and the features are curves of almost-constant frequency — the base frequency of the pulsar modulated by the doppler shifts caused by the motion of the Earth and the pulsar itself.

The pulsar search can be parallelized by splitting the possible frequencies into bins, and each processor searching a given bin. The search involves selecting sky position and frequency slowing, and searching for statistically-significant signals. Once a pulsar source has been detected, the result is catalogued as an event data structure, which describes the pulsar's position in the sky, the signal-to-noise ratio, time etc.

1.4 Event Identification Computation Pipeline

During the search for astrophysical events a long duration, one dimensional time series is processed by a variety of filters. These filters then produce a new time series which may represent the signal to noise ratio in the data. A threshold is applied to each of the new time series in order to extract possible events, which are catalogued in the LIGO database. In order to determine if the event is significant, the raw data containing instrumentation channels needs to be re-examined. It is possible that the occurrence of the event was triggered by some phenomena such as lightning strikes, acoustic noise, seismic activity, etc. These are recorded by various instruments present in the LIGO system and can be found in the raw data channels. To eliminate their influence, the instrumental and environmental monitor channels must be examined and compared to the occurrence of the event. The location of the raw data channels can be found in the LIGO database. Since the event is pinpointed in time, only small portions of the many channels (that possibly needs to be processed) need to be examined. This computational pipeline clearly demonstrates the need for efficient indexing and processing of data in various views:

- a long time interval single channel data, such as the initial data being filtered, and
- the many channel, short time interval such as the instrument data needed to add confidence to the observation of events.

2 GriPhyN/LIGO Virtual Data Grid

GriPhyN [5] (Grid Physics Network) is a collaboration funded by the US National Science Foundation to build tools that can handle the very large (petabyte) scale data requirements of cutting-edge physics experiments. In addition to the LIGO observatory, GriPhyN works with the CMS and Atlas experiments at CERN's Large Hadron Collider [6] and the Sloan Digital Sky Survey [7].

A key component needed to implement the data pipeline is a *virtual data service*; a system to dynamically create data products requested during the various stages. The data could possibly be already processed in a certain way, it may be in a file on a storage system, it may be cached, or it may need to be created through computation. The full elaboration of this system will allow complex data pipelines to be set up as virtual data objects, with existing data being transformed in diverse ways.

2.1 Virtual Data

An example of Virtual Data is this: *"The first 1000 digits of Pi"*. It defines a data object without computing it. If this request comes in, we might already have the result in deep archive: should we get it from there, or just rerun the computation? Another example is: *"Pi to 1000 places"*. How can we decide if these are the same request even though the words are different? If someone asks for *"Pi to 30 digits"*, but we already have the first two, how can we decide that the latter can be derived easily from the former? These questions lie at the heart of the GriPhyN Virtual Data concept.

In the extreme, there is only raw data. Other requests for data, such as obtaining a single channel of data ranging over a large time interval, can be derived from the original data set. At the other extreme, every single data product that has been created (even if it represents an intermediate step not referred to again) can be archived. Clearly, neither extreme is an efficient solution; however with the use of the Virtual Data Grid (VDG) technology, one can bridge the two extremes. The raw data is of course kept and some of the derived data products are archived as well. Additionally, data can be distributed among various storage systems, providing opportunities for intelligent data retrieval and replication.

VDG will provide transparent access to virtual data products. To efficiently satisfy requests for data, the VDG needs to make decisions about the instantiation of the various objects. The following are some examples of VDG support for LIGO data:

- Raw data vs. cleaned data channels. Most likely, only the virtual data representing the most interesting clean channels should be instantiated.
- Data composed from smaller pieces of data, such as long duration frames that could have been already processed from many short duration frames.
- Time-frequency image, such as the one constructed during the pulsar search. Most likely the entire frequency-time image will not be archived. However, all its components (short power spectra) might be instantiated. The VDG can then compose the desired frequency-time images on demand.
- Interesting events. Given a strong signal representing a particularly promising event, the engineering data related to the time period of the occurrence of the

event will most likely be accessed and filtered often. In this case, the VDG might instantiate preprocessed instrumental and environmental data channels, data that might otherwise exist only in its raw form.

2.2 Simple Virtual Data Request Scenario

A VDG is defined by its Virtual Data Domain, meaning the (possibly infinite) set of all possible data objects that can be produced, together with a naming scheme ("coordinate system") for the Domain. There are functions on the domain that map a name in the domain to a data object. The VDG software is responsible for caching and replicating instantiated data objects. In our development of the LIGO VDG, we begin with a simple Cartesian product domain, and then add complexity.

Let D be the Cartesian product of a time interval with a set of channels, and we think of the LIGO data as a map from D to the value of the channel at a given time. In reality, of course, it is complicated by missing data, multiple interferometers, data recorded in different formats, and so on.

In the following, we consider requests for the data from a subdomain of the full domain. Each request is for the data from a subset of the channels for a subinterval of the full time interval. Thus, a request might be written in as:

```
T0=700004893, T1=700007847; IFO_DCDM_1, IFO_Seis_*
```

where `IFO_DCDM_1` is a channel, and `IFO_Seis_*` is a regular expression on channel names. Our first task is to create a naming scheme for the virtual data object, each name being a combination of the name of a time interval and the name of a set of channels.

This could be satisfied if there is a suitable superset file in the replica database, for example this one:

```
T0=70004000,T1=700008000;IFO_*
```

We need to be able to decide if a given Virtual Data Object (VDO) contains another, or what set of VDO's can be used to create the requested VDO. If C_i is a subset of the channels, and I_i is a subinterval, then tools could be used to combine multiple files (C_1,I_1), (C_2,I_2), ... perhaps as:

- The new file could be (C, I), where C = *union* C_i and I = *intersect* I_i, a channel union tool, or
- The new file could be (C, I), where C = *intersect* C_i and I = *union* I_i, an interval union tool.

We could thus respond to requests by composing existing files from the distributed storage to form the requested file.

To be effective, we need to develop a knowledge base of the various transformations (i.e., how they are performed, which results are temporary, and which need to be persistent). We need to know the nature of the context in which these transformations occur: something simple is the Fourier transform, but more difficult would a transformation that uses a certain code, compiled in a certain way, running on a specific machine. This description of context is needed in order to be able to execute the transformations on data sets as well as to determine how to describe them.

2.3 Generalized Virtual Data Description

The goal of the GriPhyN Virtual Data Grid system is to make it easy for an application or user to access data. As explained above, as a starting point we will service requests

consisting of a range of time t_0 to t_1 (specified in GPS seconds), followed by a list of channels. However, we now extend the idea of channel to "virtual channel".

Virtual Channels

A virtual channel is a time series, like a real channel, but it may be derived from actual channels, and not correspond to a channel in the raw data. Some examples of virtual channels are:

- An actual recorded channel, "raw".
- An actual recorded channel, but downsampled or resampled to a different sampling rate.
- An arithmetic combination of channels, for example $2C_1 + 3C_2$, where C_1 and C_2 are existing channels.
- The actual channel, convolved with a particular calibration response function, and scaled. For example, the X component of the acceleration.
- The virtual channel might be computed from the actual data channels in different ways depending upon what time interval is requested (e.g., the calibrations changed, the channels were hooked up differently, etc.).
- A virtual channel could be defined in terms of transformations applied to other virtual channels.

In short, the virtual channels are a set of transformations applied to the raw data.

The set of virtual channels would be extendable by the user. As the project progresses, one may want to extend the set of virtual channels to include additional, useful transformations. Thus, if a user is willing to define a virtual channel by specifying all the necessary transformations, it will be entered in the catalog and will be available to all users, programs, and services. New channels can be created from the raw data channels by parameterized filters, for example decimation, heterodyning, whitening, principle components, autocorrelation, and so forth.

Data Naming

A crucial step in the creation of the GriPhyN Virtual Data model is the naming scheme for virtual data objects. Semantically, we might think of names as a set of keyword-value pairs, extended by transformations, perhaps something like `(T0=123,T1=456,Chan=[A,B*,C?]).pca().decimate(500)`. The first part in parentheses is the keyword-value set, the rest is a sequence of (perhaps parameterized) filters. We could also think of using names that contain an SQL query, or even names that include an entire program to be executed. The syntax could also be expressed in other ways, as XML, or with metacharacters escaped to build a posix-like file name. We could use a syntax like `protocol://virtual-data-name` to express these different syntax in one extensible syntax. However, decisions on naming Virtual Data must be premised on existing schemes described in Section 1.2.

3 GriPhyN Support

The goal of GriPhyN is to satisfy user data requests transparently. When a request for a set of virtual channels spanning a given time interval is made, the application (user program) does not need to have any knowledge about the actual data location, or even if the data has been pre-computed. GriPhyN will deliver the requested virtual chan-

nels by either retrieving existing data from long term storage, data caches containing previously requested virtual channels, or by calculating the desired channels from the available data (if possible).

When satisfying requests from users, data may be in slow or fast storage, on tape, at great distance, or on nearby spinning disk. The data may be in small pieces (~1 second) or in long contiguous intervals (~1 day), and conversion from one to another requires computational and network resources. A given request for the data from a given time interval can thus be constructed by joining many local, small files, by fetching a distant file that contains the entire interval, or by a combination of these techniques. The heart of this project is the understanding and solution of this optimization problem, and an implementation of a real data server using GriPhyN tools.

3.1 User Requests

The initial implementation of the GriPhyN system will accept requests in the semantic form:

```
t0,t1; A,B,C...,
```

where `t0`, `t1` is a time interval, and `A,B,C,...` are virtual channels. We assume that the order in which the channels are listed does not affect the outcome of the request. The syntax of the virtual channel is yet to be determined. In the simplest form, a virtual channel resulting from a transformation Tr_x on a channel C_y would be $Tr_x(C_y)$; additional attributes (such as transformation parameters or other input channels) can be specified as additional parameters in the list: $Tr_x(C_y, C_z; \alpha, \beta,)$, depending on the transformation. The transformation specific information will be stored in the Transformation Catalog described below.

The semantic content of the request is defined above, but not the syntax. The actual formulation for the GriPhyN-LIGO VDG will be an XML document, though the precise schema is not yet known. We are intending to implement requests as a very small document (for efficiency), but with links to context, so that a machine can create the correct environment for executing the request. We expect much of the schema development to be implemented with nested XML namespaces [8].

3.2 Data Access, Cost Performance Estimation

When a user or computer issues a request to the Virtual Data Grid, it is initially received by a *Request Manager* service and sent for processing to the *Metadata Catalog*, which provides the set of logical files that satisfies the request, if such exists. The files names are retrieved from the Metadata Catalog based on a set of attributes.

The logical files found in the Metadata Catalog are sent to the *Replica Catalog*, which maps them to a unique file, perhaps the closest in some sense of many such replicas. The information about the actual file existence and location (provided by the Replica Catalog) are passed to the Request Manager, which makes a determination about how to deliver the data.

If the requested data is present, the Request Manager still needs to determine whether it is cheaper to recalculate the data or access it. When considering the cost of referencing data, the cost of accessing various replicas of the data (if present) needs to be estimated. If the data is not present, the possibility and cost of data calculation needs to be evaluated. In order to make these decisions, the Request Manager queries the *Informa-*

tion Catalog. The latter can provide information about the available computational resources, network latencies, bandwidth, etc.

The next major service is the *Request Planner*. It is in charge of creating a plan for the execution of the transformations on a given data set and/or creating a plan for the retrieval of data from a storage system. The Request Planner has access to the system information retrieved by the Request Manager. To evaluate the cost of re-computation, the cost of the transformations needs to be known. This information, as well as the input and parameters required by a given transformation, code location, etc. are stored in the *Transformation Catalog*. The Request Planner uses information about the transformations that have been requested, to estimate the relative costs of computation vs. caching, and so on. The system would possibly keep a record of how long the various transformations took, and could use this performance history to estimate costs. This record and the analytical performance estimates will be maintained in the Transformation Catalog.

The performance data needed in the evaluation of re-computation and replica access costs (such as network performance, availability of computational resources, etc.) will be provided by other information services, which are part of the Globus toolkit [9], a software environment designed for the Grid infrastructure.

Once the request planner decides on a course of action the *Request Executor* is put in charge of carrying out the plan which involves the allocation of resources, data movement, fault monitoring, etc. The Request Executor will use the existing Globus infrastructure to access the data and the computational Grid, and run large jobs reliably with systems controlled by systems such as Condor [10]. As a result of the completion of the request, the various catalogs might need to be updated.

3.3 Proactive Data Replication

Simply, just retrieving data from the replica catalog is not sufficient. The system must take a proactive approach to creating replica files and decide whether the results of transformations will be needed again. For example, if there is a request for a single channel spanning a long time interval and the replica catalog contains only files which are multi-channel spanning short time periods, then a significant amount of processing is needed to create the requested file (many files need to be opened and a small amount of data needs to be retrieved from each of them). However, once this transformation is performed, the resulting data can be placed in the replica catalog for future use, thus reducing the cost of subsequent requests. New replicas should also be created for frequently accessed data, if accessing the data from the available locations is too costly. For example, it may be useful to replicate data that initially resides on tape to a local file system. Since the Request Manager has information about data existence, location and computation costs, it will also be responsible for making decisions about replica creation.

4 Conclusions

Although the GriPhyN project is only in its initial phase, its potential to enhance the research of individual physicists and enable their wide-spread collaboration is great. This paper presents the first step in bridging the understanding between the needs of

the physics community and the research focus of computer science to further the use and deployment of grid technologies [11] on a wide scale, in the form of a Virtual Data Grid.

References

[1] Barish, B. C. and Weiss, R., Physics Today, Oct 1999, pp. 44-50; also http://www.ligo.caltech.edu/

[2] Taylor, G., and Weisberg, J. M., *Astrophys. J.* **345,** 434 (1989).

[3] Anderson, S., Blackburn, K., Lazzarini, A., Majid, W., Prince, T., Williams, R., The LIGO Data Analysis System, Proc. of the XXXIVth Recontres de Moriond, January 23-30, 1999, Les Arcs, France, also http://www.ligo.caltech.edu/docs/P/P990020-00.pdf

[4] Blackburn, K., Lazzarini, A., Prince, T., Williams, R., XSIL: Extensible Scientific Interchange Language, Lect. Notes Comput. Sci. **1593** (1999) 513-524.

[5] GriPhyN, Grid Physics Net, http://www.griphyn.org/

[6] CERN Large Hadron Collider, http://lhc.web.cern.ch/lhc/

[7] Sloan Digital Sky Survey, http://www.sdss.org/

[8] For information of XML and schemas, see http://www.xml.com/schemas

[9] Globus: http://www.globus.org

[10] Condor: http://www.cs.wisc.edu/condor/.

[11] Gridforum: http://www.gridforum.org/