GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists

Ewa Deelman, Carl Kesselman, Gaurang Mehta, Leila Meshkat, Laura Pearlman
Information Sciences Institute
University of Southern California
Marina Del Rey, CA 90292
{deelman, carl, gmehta, meshkat, laura}@isi.edu

Kent Blackburn, Phil Ehrens, Albert Lazzarini, Roy Williams
California Institute of Technology
1200 East California Boulevard
Pasadena, California 91125
{kent, pehrens, lazz}@ligo.caltech.edu, roy@cacr.caltech.edu

Scott Koranda
University of Wisconsin Milwaukee
Department of Physics
1900 East Kenwood Blvd
Milwaukee, WI 53211
skoranda@gravity.phys.uwm.edu

Abstract

Many Physics experiments today generate large volumes of data. That data is then processed in a variety of ways in order to achieve the understanding of fundamental physical phenomena. The goal of the NSF-funded GriPhyN project (Grid Physics Network) is to enable scientists to seamlessly access data whether it is raw experimental data or a data product which is a result of further processing. GriPhyN provides a new degree of transparency in how data-handling and processing capabilities are integrated to deliver data products to end-users or applications, so that requests for such products are easily mapped into computation and/or data access at multiple locations. GriPhyN refers to the set of all data products available to the user as Virtual Data. Among the physics applications participating in the project is the Laser Interferometer Gravitational-wave Observatory (LIGO), which is being built to observe the gravitational waves predicted by general relativity. In this paper, we describe our initial design and prototype of a Virtual Data Grid for LIGO.

1. Introduction

GriPhyN (Grid Physics Network, www.griphyn.org) is a NSF-funded project which aims to support large-scale data management in physics experiments such as high-energy physics, astronomy and gravitational wave physics. GriPhyN puts data both raw and derived under the umbrella of Virtual Data[1]. A user or application can ask for data using application-specific metadata without needing to know whether the data is available on some storage system or if it needs to be computed. To satisfy the request, GriPhyN will schedule the necessary data movements and computations to produce the requested results. GriPhyN uses the Globus toolkit (www.globus.org) as the basic grid infrastructure and builds on top of it high-level services, which can support virtual data requests. Some of these services are request planning, request execution, replica selection, etc.

In this paper we describe the work we have done within the context of one of the experiments in

GriPhyN, namely the Laser Interferometer Gravitational-wave Observatory (LIGO), www.ligo.caltech.edu. Using LIGO, we explore the concepts of Virtual Data via a prototype. This prototype has been built within the larger framework of the GriPhyN architecture [2] and demonstrated at SC'01. Our contributions encompass three areas:

- Request planning.
- Transparency of Virtual Data requests with respect to data location and materialization.
- Integration of Globus security and resource management with the existing LIGO Data Analysis System (LDAS, ldas-sw.ligo.caltech.edu), which is the production environment for data processing in LIGO.

We describe the request management system, which takes the application-specific metadata and maps it to an abstract representation of the necessary transformations and the data required by them. That abstract representation, in the form of a directed acyclic graph (DAG), is taken by the request planner and transformed into a concrete DAG which includes the commands for necessary data movements, computations, etc. The plan also includes registering new data products in the replica catalog [3] to indicate the availability of the data.

Once the user makes the request, the user does not need to know whether the data is available on some storage system or whether it needs to be computed. The planner provides that transparency and places the desired data in a user-defined location.

In order to be able to execute requests, authentication and authorization need to be performed. All the operations set up by the planner (in the concrete DAG), are executed on the user's behalf by the request executor using the user's proxy credentials.

To frame this work in a context relevant to the LIGO application, we have used the LDAS system, developed by the LIGO Laboratory and being used widely within the LIGO Scientific Collaboration, as an execution platform. LDAS can perform a wide range of sophisticated and computationally intensive data analysis. To achieve integration of LDAS into the Grid environment, we have designed Globus [4] interfaces both for data staging (via GridFTP [5], a Globus Security Infrastructure-based data transfer protocol) and computation scheduling (via GRAM (Globus Resource Allocation Manager)) .

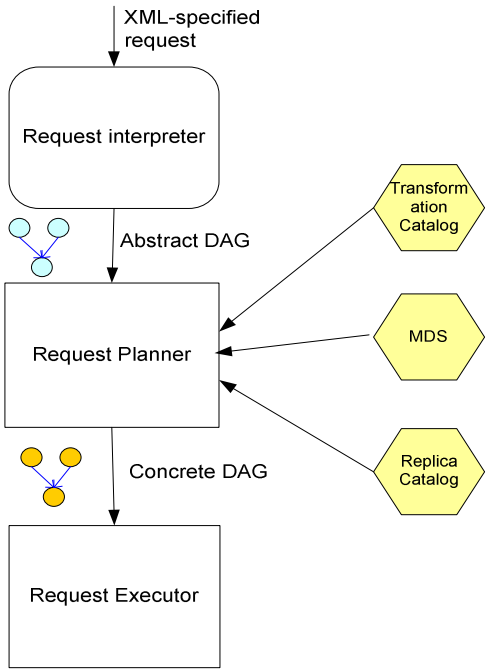

Figure 1: Virtual Data System

2. Related work

Many projects both in the US and in Europe are building and deploying data grids. The majority are focused on designing higher-level services on top of the basic Globus infrastructure. Some of the services being developed are: replica management, which combines replica cataloging with file transfer, replica

selection, which chooses the "best" replica with respect to network and storage performance, and broker services, which seek out available resources to schedule jobs. Among the data-grid-centric projects are the Particle Physics Data Grid (PPDG, www.ppdg.net), the European Data Grid (www.eu-datagrid.org), and the National Virtual Observatory (www.us-vo.org/). Another project, Gridlab (www.gridlab.org), is concerned with developing support for adaptive applications on the grid. The main difference between the projects mentioned above and GriPhyN is that the latter is the only one that supports the concept of Virtual Data and specifically provides transparency with respect to materialization.

3. Request Management System Design

Our system is designed to deliver data requested by the user/application at a user specified location. The user is provided with a customized GUI to input the desired metadata attribute values. This request is formatted into XML and sent to the application-specific request interpreter (see Figure 1).

The interpreter takes the XML request and translates it into an abstract directed acyclic, task graph (aDAG). The aDAG specifies the computations which need to take place (transformations) and the data needed for the computation *without specifying the physical location of the data*. Figure 2 (left) shows an aDAG which represents the sequence of operations needed to extract a channel of data from a frame file (a standard data structure in LIGO). The necessary operations are that the input file ("frame1.F") needs to be sent to the processing location "X", *yet to be determined*, then the transformation "extract", with the parameters, "channelA" (a specific channel requested by the user) needs to be performed at some location "X" and then the resulting data needs to be placed at the user-specified location, here "isi.edu"

3.1. Planning

The abstract DAG representation is sent to the planner, which combines information gathered about the data location(s), location(s) where the transformations can take place and constructs a concrete DAG.

The planner consults various services, described below, to determine the locations of the data, the locations of the transformations, and the state of the Grid—the availability of the resources.

The replica catalog [3, 6], which is a Globus service, allows users to register files as logical collections and provides mappings between logical names for files and collections and the storage system locations of file replicas. The catalog registers three types of entries: logical collections, locations, and logical files. A logical collection is a user-defined group of files. Collections allow users and applications to register and manipulate groups of files as a collection, rather than requiring that every file be registered and manipulated individually. Location entries in the replica catalog contain the information required for mapping a logical collection to a particular physical instance of that collection. Each location entry represents a complete or partial copy of a logical collection on a storage system. The planner consults the replica catalog to determine the locations of the data specified in the aDAG.

The transformation catalog [7] maps a logical transformation to a physical instance of the transformation. Logical transformations represent an abstract view of the transformation and can have one or more instances. A physical transformation is an instance of a logical transformation and refers to the physical location of the machine on which the transformation exists and the full path on that machine. The physical instantiations of the transformation can be installed, binary, or source. The latter two types of transformations can be moved to an appropriate system and executed.

Using the transformation catalog, the request manager can check for the presence of a requested transform, find the appropriate transformation for a given architecture and find the location of the transformation (URL). The transformation catalog can also provide necessary information to run the executable (for example, shared libraries) or compile from source (Makefile), as well as provide validation information for transformation (digital signature).

The Network Weather Service (NWS) [8] is a distributed system that periodically monitors and dynamically forecasts the performance that various network and computational resources can deliver over a given time interval. NWS forecasts process-to-process network performance (latency and bandwidth) and available CPU percentage for each machine that it monitors. This information can be used by the planner to evaluate which replicas should be accessed in the computations.

The Globus Metacomputing Directory Service (MDS) can provide information about the resources available in the Grid (within the scope of a Virtual Organization (VO)). It can also provide detailed information about the physical characteristics of a resource—the number of processors, memory, and disk space available, etc. This information can be

used by the planner to choose the most appropriate resources.

Since the input data files can be replicated at several locations and the transformations can be performed on a variety of systems, it is the planner's responsibility to find the "optimum" plan. The evaluation of what is optimal can be varied and can depend on a wide range of criteria. The simplest strategy is to find a feasible plan.

One possible solution to this planning problem is to use a general-purpose AI planner, such as Blackbox [9] or FF [10]. An AI planner uses sophisticated search techniques to find a path from the initial state of the system to the goal state, based on the domain knowledge—the planner's understanding of how various, predefined actions affect the state of the system. In the context of the GriPhyN project, the initial state of the system is determined by the resources available, both compute and storage, as well as the available transformations and data files.

The domain is composed of actions which are defined in the system. One such action is data transfer. The preconditions of the transfer of data "A" from location "X" to location "Y" is that "X" and "Y" are storage locations and that "A" is present at location "X" and not at location "Y". The result or post-condition of the transfer is that "A" is present at "Y".

The goal state of the system is determined by the user's request. In the simple example above, the goal state could be: "A" at location "Y". We have evaluated two planners: Blackbox and FF. The inputs to the Blackbox and FF planners are provided in PPDL (Planning Domain Definition Language), which describes the possible actions in a particular domain, their preconditions and effects.

The Blackbox planner finds all the paths between the initial state and goal state and then checks to see if they satisfy the constraints of the problem. For instance, it may be unfeasible to do computations on an LDAS system that has no memory available.

The FF planner uses heuristics to find a plan that satisfies the constraints without initially generating an exhaustive set of plans, thus making it faster than the Blackbox planner. We have been able to define our prototype domain in both planners and obtain feasible plans to satisfy the user's request.

We plan to incorporate performance estimates as well as reliability measures to guide the planning process. Incorporating modules within or outside of the planner that would allow us to find plans that are optimal for cost, performance, and reliability are subjects of our current research.

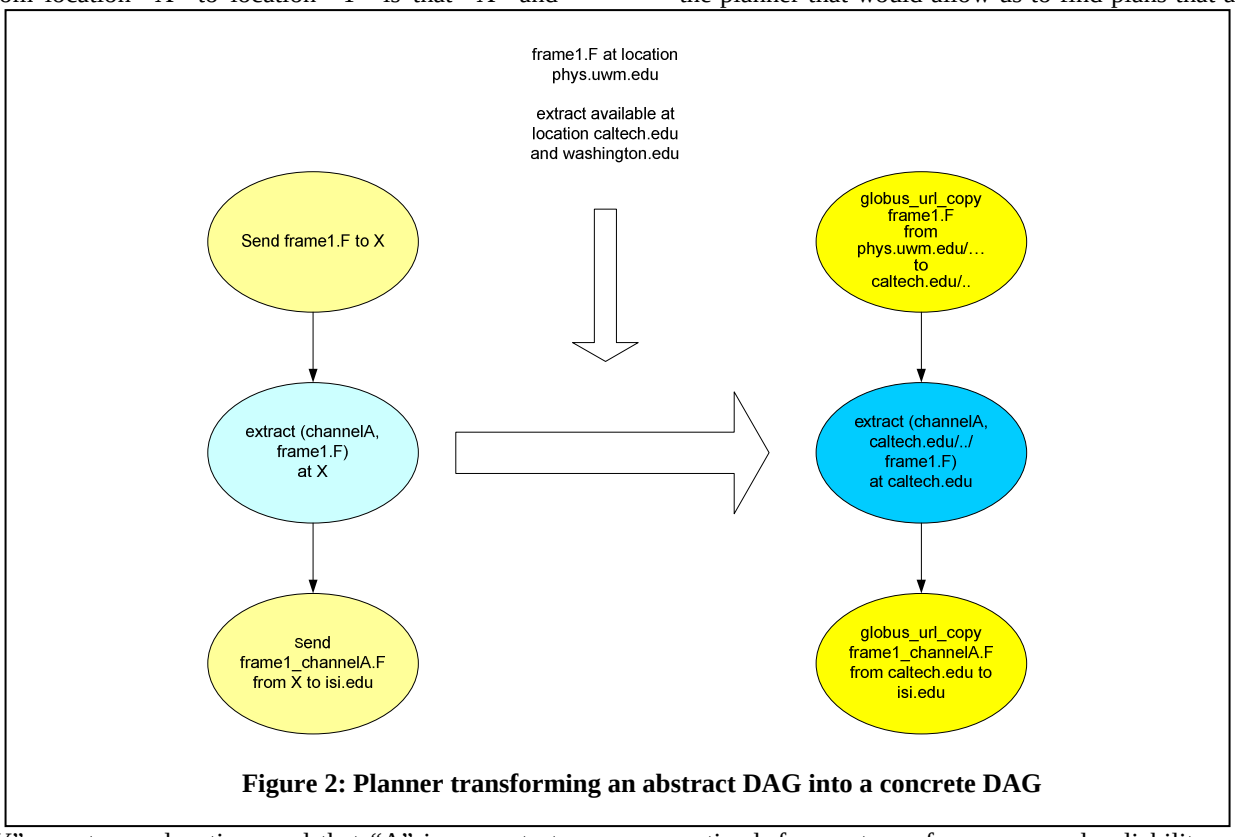

Figure 2: Planner transforming an abstract DAG into a concrete DAG

Evaluating the performance will take into account factors such as network, disk and compute resource performance. Basic information can be gathered from MDS and NWS. The expected performance of the plan can be obtained by evaluating the cost of each of the nodes in the concrete DAG.

Reliability measures will need to account not only for simple faults, such as lack of disk space or network partitioning, but also more complex events, such as "system is up and functioning, but your job is 999th in the queue and will not complete any time soon". The difficulty lies in capturing the latter effect, because, even though the system is not faulty, the application which relies on that system experiences unquantifiable delays.

Currently, the planner does not provide the user feedback regarding the expected execution time of the request. However, providing this type of information to the user is important. It is possible that the virtual data requested results in access to many data files and large amounts of computation. Given this information, the user might want to refine the request.

Another important feedback to the user is an indication of the amount of resource usage needed to satisfy the request. It is possible that the best performance, from the point of view of fulfilling the request, is to perform the computation on a massively parallel machine. However, if the user has a limited allocation on that machine, a network of workstations might be a more suitable alternative. Although the issues of accounting and policy-based resource usage are important, they are not addressed in the current work.

As the result of the planner, we have a concrete DAG (shown on the right in Figure 2), where each node contains a fully specified command—here a *globus_url_copy* of the input and output files for data staging in and out—and the extract transformation which will be performed at Caltech.

3.2. Execution

Once the concrete DAG is generated by the planner, it is submitted to a Condor-G server which starts a DAGMan . DAGMan analyses the DAG and then submits jobs in the manner in which they are defined in the DAG making sure that the dependencies between the nodes are observed. These jobs may run locally or on a Condor pool or via a GRAM interface to a resource, where Globus is

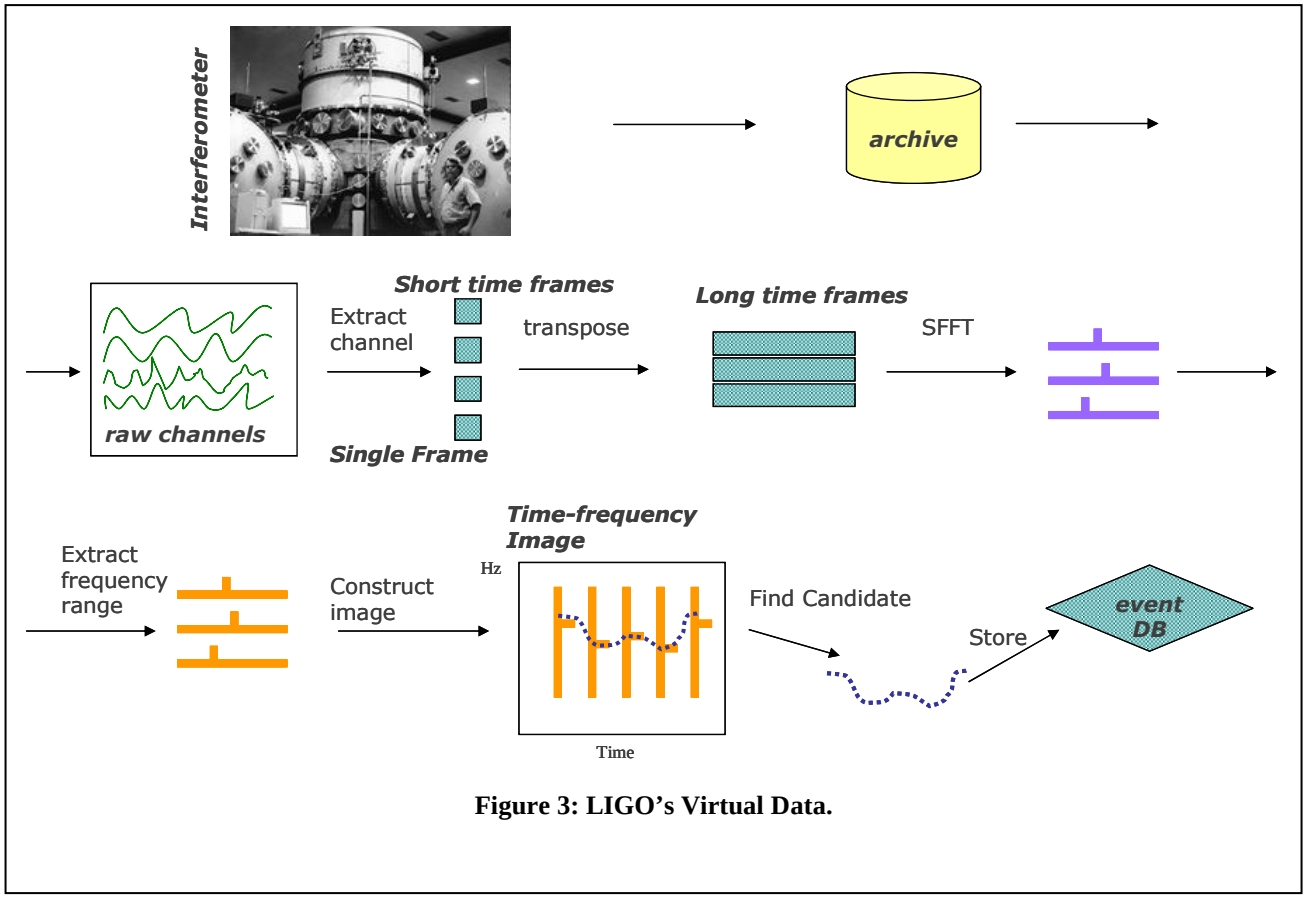

Figure 3: LIGO's Virtual Data.

available.

As the jobs in the plan finish they write the exit status of the job to a log file, which is parsed continuously by the request manager. If any of the tasks in the plan fail then the execution of the DAG is halted and the error is reported to the user via the web interface. Condor-G has a restart facility available by which a job can be resubmitted and it can continue from the point where it stopped, but this feature is currently not being used. If all the jobs in a plan succeed then the output file is generated and the replica catalog is updated to reflect the availability of the data at the new location (in the example in Figure 2, isi.edu).

If another request for the same data is made, the planner finds out (by consulting the replica catalog), that the data is already materialized and available at a given location (isi.edu, in Figure 2) and will only do a data movement if it is necessary to satisfy the request.

Clearly this type of system can improve the use of computational resources and provide a better response time to the users by identifying available virtual data and materializing data if necessary.

4. Realizing Virtual Data

Building on the basic request management infrastructure described above, we have prototyped a system that can support Virtual Data by providing the user with transparency with respect to data location and data materialization. First, we briefly describe LIGO and its Virtual Data[11-13].

4.1. LIGO

LIGO (Laser Interferometer Gravitational-Wave Observatory, www.ligo.caltech.edu) [14, 15] is a distributed network of three-km-scale interferometers occupying two sites in the U.S. The construction project was funded by NSF and jointly built by Caltech and MIT. The LIGO Scientific Collaboration (LSC, www.ligo.org) is the scientific body organized to use the instrumentation as a national resource. The observatory's mission is to detect and measure gravitational waves predicted by general relativity, Einstein's theory of gravity, in which gravity is described as due to the curvature of the fabric of time and space. One well-studied source of gravitational waves is the motion of dense, massive astrophysical objects such as neutron stars or black holes. Other signals may come from supernova explosions, quakes in neutron stars, and pulsars.

Gravitational waves interact extremely weakly with matter, and the measurable effects produced in terrestrial instruments by their passage is expected to be miniscule. In order to establish a confident detection or measurement, a large amount of auxiliary data will be acquired (including data from seismometers, microphones, etc.) and analyzed (for example, to eliminate noise) along with the strain signal that measures the passage of gravitational waves The raw data collected during experiments is a collection of continuous time series at various sample rates. The amount of data that will be acquired and cataloged each year is on the order of tens to hundreds of terabytes. The gravitational wave strain channel is less than 1% of all data collected. Analysis on the data is performed in both time and Fourier 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.

4.2. LIGO's Virtual Data

The goal of GriPhyN and the idea behind Virtual Data is to enable scientists to think of their data in terms of metadata and not in terms of file names and location or in terms of the processing that needs to be done on a concrete piece of data.

Thus, the first step in building a Virtual Data Grid for an experiment is to understand the problem domain and capture the data analysis processes. Initially, we focused on a specific LIGO problem, the pulsar search, which can be implemented as a pipeline depicted in Figure 3. The data needed to conduct the search is a long stretch (~4 months, $2x10^{11}$ points) of a single channel—the gravitational wave strain channel. The power spectra of the small segments are stacked to make a large frequency-time image, perhaps $4x10^5$ on each side. The pulsar search consists of searching for coherent signals in this image. A source would appear on the frequency-time image as a wavering line, whose frequency might be 1 kHz, but modulated by a few parts in 10^6 over periods of 1 day and a few parts in 10^4 over periods of 1 year. In addition, if the source exhibits any secular variations due to slowing down of its rotational period, these will be encoded in the data as well.

The first element in Figure 3 is data archiving as instrumental data is stored into an archive. Next, since, the raw data comes from the instrument as short (1 second duration) Frames (a data structure used in the gravitational wave community) with all the channels, some processing geared towards the removal (cleaning) of certain instrumental signatures needs to be done. For example, naturally occurring seismic vibration can be subtracted from the data using the channels from the sensitive seismometer that is part of the LIGO data stream. For the pulsar search, the gravitational wave strain channel is extracted. The short-duration, single channel frames are then combined into much longer frames in a transpose operation, and further data conditioning filters are applied.

The pulsar search is conducted in the frequency domain; thus, Fourier Transforms, in particular Short Fourier Transforms (SFTs), are performed on the long duration time frames. Since the pulsars are expected to be found in a small frequency range, the frequency interval of interest is extracted from the STFs. The resulting power spectra are used to build the time-frequency image, which is analyzed for the presence of pulsar signatures. If a candidate signal with a good signal to noise ratio is found, it is placed in LIGO's event database.

In Figure 3, each of the elements of the pipeline is a Virtual Data product. The user can address each of them using domain-specific metadata. For example, the short Fourier transform (SFT) can be described by the time interval of interest and the channel over which the transform is to be computed.

The SFTs and the resulting time-frequency images are standard data products for the LIGO community and can be used by many applications. In addition to being structures needed to find signals of pulsars, gravitational wave scientists can use them to locate gravitational wave bursts, which can be the result of certain supernovae explosions. Thus, building, based on user demands, a Grid of LIGO's Virtual Data products will be beneficial to the entire LIGO community and enable efficient sharing of computational and storage resources.

4.3. Prototype Demonstration

So far we have sketched the overall design of the system. Here, we describe our prototype, which implements many components of the design. The goal of this implementation is two-fold: first, to develop initial prototypes for the Virtual Data Toolkit in support of the Virtual Data concepts, and, second, to provide a scientifically meaningful, grid-enabled environment for the LIGO collaboration. This

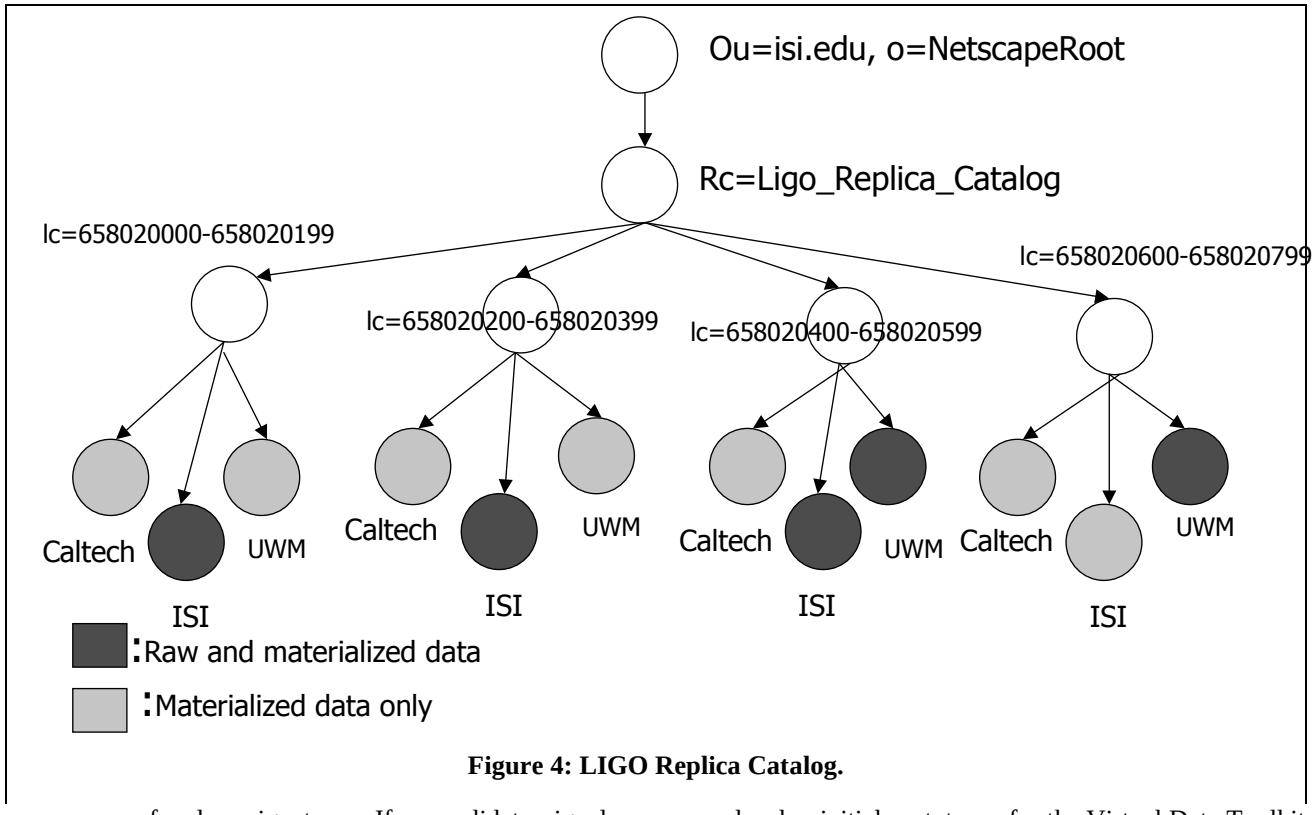

Figure 4: LIGO Replica Catalog.

prototype was shown at SC'2001 in November. At that time, our tested was composed of four storage systems at Caltech, ISI, UWM, and one on the showroom floor. The compute resources available were two compute nodes (at ISI and on the showroom floor) and two LDAS installations, one at UWM and one at Caltech.

Initially, we populated the replica catalog with raw LIGO data. Figure 4 depicts the organization of that replica catalog. Since LIGO's main index into the data is based on time, we have arranged the data into collections (lc) based on time intervals (expressed in GPS time, seconds since January 7 1980─a time format used by the LIGO community). Initially, only two locations (UWM and ISI) had raw data captured at the instrument. As users made requests for derived data, new data products were calculated and stored at various locations (UWM, ISI and Caltech).

Figure 5 depicts the overall logic and design of the prototype. The user makes a request for virtual data via a web portal. The user is able to specify the channel name, the start time, where the resulting data should be placed, and the format of the resulting data. The format of the data can be XML, Frame, or XSIL[16] (the Extensible Scientific Interchange Language, based on XML), which is a flexible, hierarchical, extensible, transport language for scientific data objects. The request is translated into XML and is passed to the Request Manager (RM).

The Request Manager interprets the requests and constructs a DAG that fully represents the computations and data movements necessary to satisfy the request. Some of the subtasks of the RM are:

- Authenticate user.
- Make an optimal plan for satisfying the user's request using the available resources.
- Submit the plan to Condor-G for execution.
- Return the status of the request and the location of the data to the user.

Security in the prototype is implemented using Globus-GSI [17] (Grid Security Infrastructure), an authentication system based on SSL/TLS and X.509 certificates, and MyProxy [18], a mechanism for delegating GSI credentials to applications such as web portals. GSI extends the traditional X.509 model by allowing an individual to generate short-lived *proxy certificates* that can be delegated to remote processes running on the user's behalf. A remote process can use a delegated credential (proxy certificate plus private key) to act as the original user when authenticating to third-party servers. For example, a simulation process may use a delegated proxy credential to fetch one of the user's files from a file server. The MyProxy server provides a mechanism to do this delegation indirectly: the user

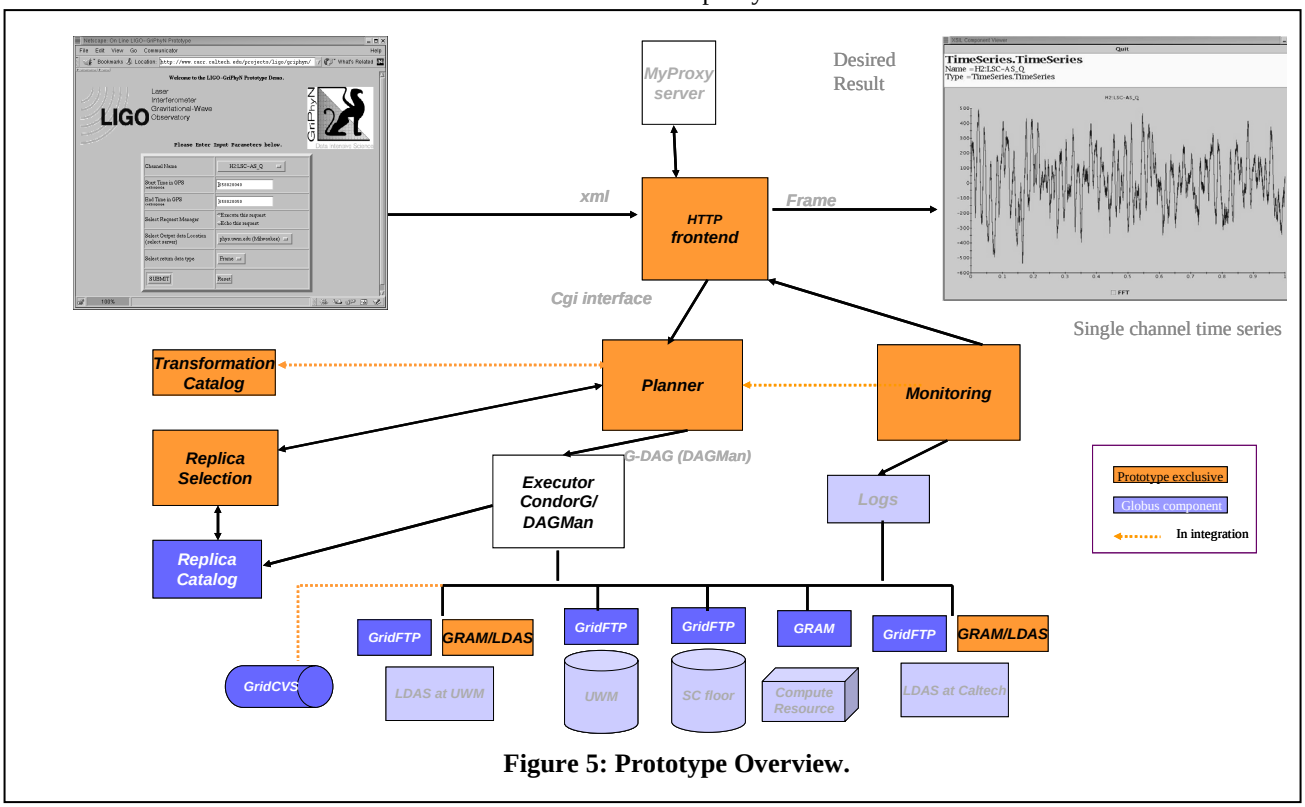

Figure 5: Prototype Overview.

delegates a proxy credential to the MyProxy server and associates a password with that credential. Later, an authorized application can request a delegated credential by presenting that same password to the MyProxy server (using an encrypted, mutually-authenticated GSI connection). The application, here the request manager, can then access Grid resources with the end-user's privileges. It asks the user for the username and password via an encrypted GSI connection and contacts the MyProxy server to obtain the user's proxy credential.

In order for the planner to determine if the requested data exists, or if the necessary input data exists (in case materialization needs to be performed), the planner consults the replica catalog. In the example shown in Figure 2, the planner has consulted the replica catalog, did not find the file needed to satisfy the user request, but found that logical file "Frame1.F" is available at "phys.uwm.edu/…". This file is a full frame file, a file with all the channels, and is needed to extract the channel specified by the user's request─channelA.

At the time of the demo, the transformation catalog was not yet fully integrated into the system. Transformations were assumed to be available at all the compute sites.

In our current implementation, if any processing on the data needs to be done, the planner will select a execution location which is "close" to the input data needed for the computation. Currently, "close" is defined as having a high bandwidth between the data and compute locations. In our example, caltech.edu was "close". To complete the user's request, the planner needs to move the result from caltech.edu to the user specified location—isi.edu. The concrete DAG on the right side of Figure 2 is the result of the planning process (although details of the commands are omitted here for clarity.)

The RM sends the concrete DAG to Condor-G. In our prototype the DAG can contain LDAS commands, GridFTP [5] commands, arbitrary analysis pipeline commands, and updates to the Replica Catalog. Condor-G executes the specified data movements, submits jobs to LDAS via a GRAM interface, and updates the replica catalog. LDAS or another analysis system performs computation. Data is moved to the user-specified location and the user is then notified of the request's completion and the updates performed to the catalog.

5. Globus/LDAS Interface

In the original LDAS system, users submitted jobs by sending commands to a port on a machine running the LDAS ManagerAPI. The Globus GRAM interface to LDAS was accomplished by adding a new Globus GRAM service to an existing Globus 1.1.4 resource manager on a machine on the same network as the LDAS ManagerAPI. The new service jobmanager-ldas translates the requests sent by Condor-G to appropriate LDAS commands and sends them to the LDAS ManagerAPI. Note that the actual Globus jobmanager program globus-jobmanager was not changed. Instead a new service was added by simply defining new submit, poll, queue, and rm scripts for the new service. The jobmanager-ldas also sends the name of the machine on which the Globus GRAM service is running along with the port on which an agent is listening. The agent listening on the Globus service machine receives all communications about the job that was submitted and other information from LDAS. The received information is parsed and written as appropriate so that the scripts globus-script-ldas-poll, -submit, -queue, and rm can read the information and communicate back via callbacks as necessary and standard as part of the GRAM protocol. Currently LDAS only reports when a job is finished, and it is not possible to query LDAS to determine the state of a submitted job, so the Globus jobmanager-ldas service only reports that jobs are either running or completed. Development is currently underway that will allow direct queries of job status, and we plan to incorporate this into the jobmanager-ldas service in the coming months.

One of the major accomplishments of the GRAM/LDAS interface is the integration of the GSI security within the LDAS framework. Previously, LDAS used an unencrypted username/password scheme for authentication.

6. Conclusions and Future Work

Some projects, such as the Particle Physics Data Grid and the European Data Grid, address issues of the Data Grid by providing transparent access to existing data (transparency with respect to location); however, GriPhyN is the only project that aims to provide transparency with respect to materialization.

Although we have taken the first steps in the area of Virtual Data by allowing the user to specify the data product in terms of the relevant metadata, these are only the initial steps in building the Virtual Data Grid. In the near future, we will focus on developing more sophisticated planners, which can take into account performance and reliability as well as provide feedback to the user (for example, about how long it will take to obtain the desired data products), so that the user can decide whether to go ahead with a request.

7. Acknowledgments

The LIGO Project and LIGO Laboratory were constructed by the National Science Foundation under cooperative agreement PHY-9210038. The Laboratory operates under cooperative agreement PHY-0107417. This work was also supported by NSF under contract ITR-0086044, "GriPhyN: Grid Physics Network," (www.griphyn.org). This paper has been assigned LIGO document number LIGO-P020004-00-E.

8. References

[1] E. Deelman, I. Foster, C. Kesselman, and M. Livny, "Representing Virtual Data: A Catalog Architecture for Location and Materialization Transparency," GriPhyN technical report 2001-13, 2001.

[2] I. Foster and C. Kesselman, "A Data Grid Reference Architecture," GriPhyN 2001-6, 2001.

[3] W. Allcock, A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke., "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets.," *Journal of Network and Computer Applications*, vol. 23, pp. 187-200, 2001.

[4] I. Foster and C. Kesselman, "Globus: A Toolkit-Based Grid Architecture," in *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds.: Morgan Kaufmann, 1999, pp. 259-278.

[5] W. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing," presented at Mass Storage Conference, 2001.

[6] W. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Leigh, A. Sim, A. Shoshani, B. Drach, and D. Williams, "High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies.," presented at SC 2001, 2001.

[7] E. Deelman, C. Kesselman, and G. Mehta, "Transformation Catalog Design for GriPhyN, Prototype of Transformation Catalog Schema.," GRIPHYN 2001-17, 2001.

[8] R. Wolski, "Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service," in *Proc. 6th IEEE Symp. on High Performance Distributed Computing*. Portland, Oregon, 1997.

[9] H. Kautz and B. Selman, "Pushing the envelope: Planning, propositional logic, and stochastic search.," presented at AAAI, 1996.

[10] J. Hoffmann and B. Nebel, " The FF Planning System: Fast Plan Generation Through Heuristic Search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253-302, 2001.

[11] E. Deelman, C. Kesselman, S. Koranda, A. Lazzarini, and R. Williams, "Applications of Virtual Data in the LIGO Experiment," presented at Fourth International Conference on Parallel Processing and Applied Mathematics (PPAM'2001), 2001.

[12] E. Deelman, C. Kesselman, R. Williams, A. Lazzarini, T. Prince, J. Romano, and B. Allen, "A Virtual Data Grid for LIGO," presented at HPCN 2001, Amsterdam, 2000.

[13] B. Allen, E. Deelman, C. Kesselman, A. Lazzarini, T. Prince, J. Romano, and R. Williams, "LIGO's Virtual Data Requirements," LIGO technical report T000135-00-D, 2001.

[14] A. Abramovici, W. E. Althouse, R. W. P. Drever, Y. Gursel, S. Kawamura, F. J. Raab, D. Shoemaker, L. Sievers, R. E. Spero, K. S. Thorne, R. E. Vogt, R. Weiss, S. E. Whitcomb, and M. E. Zucker, "LIGO: The Laser Interferometer Gravitational-Wave Observatory (in Large Scale Measurements)," *Science*, vol. 256, pp. 325-333, 1992.

[15] B. C. Barish and R. Weiss, "LIGO and the Detection of Gravitational Waves," *Physics Today*, vol. 52, pp. 44, 1999.

[16] R. Williams, "XSIL: Java/XML for Scientists," California Institute of Technology.

[17] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," in *ACM Conference on Computers and Security*, 1998, pp. 83-91.

[18] J. Novotny, S. Tuecke, and V. Welch, "An Online Credential Repository for the Grid: MyProxy.," presented at the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), 2001.