

Applications of Virtual Data in the LIGO Experiment

Ewa Deelman¹, Carl Kesselman¹, Scott Koranda², Albert Lazzarini³, and Roy Williams⁴

¹ USC Information Sciences Institute,
4676 Admiralty Way, Suite 1001, Marina Del Rey, CA 90092
{deelman, carl}@isi.edu

² UWM Department of Physics
1900 East Kenwood Blvd, Milwaukee, WI 53211
skoranda@gravity.phys.uwm.edu

³ LIGO Laboratory at Caltech
18-34 Caltech, Pasadena, CA 91125
lazz@ligo.caltech.edu

⁴ Center for Advanced Computing Research
158-79 Caltech, Pasadena, CA 91125
roy@caltech.edu

Abstract. Many Physics experiments today generate large volumes of data. That data is then processed in many ways in order to achieve the understanding of fundamental physical phenomena. Virtual Data is a concept that unifies the view of the data whether it is raw or derived. It 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 (Grid Physics Network) is a NSF-funded project, which aims to realize the concepts of 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. LIGO will produce large amounts of data, which are expected to reach hundreds of petabytes over the next decade. Large communities of scientists, distributed around the world, need to access parts of these datasets and perform efficient analysis on them. It is expected that the raw and processed data will be distributed among various national centers, university computing centers, and individual workstations. In this paper we describe some of the challenges associated with building Virtual Data Grids for experiments such as LIGO.

1 Introduction

Computational Grids [12] are emerging as an indispensable environment for wide area collaborative computing. They enable new problem-solving methodologies,

in which instruments, compute resources, and people interact over great distances. By providing seamless access to a range of distributed resources, Computational Grids permit the solution of ever greater and more complex problems. In 2001, a problem that appeared to be unsolvable, was solved using the power of the Computational Grid. NUG30 [16], which is a combinatorial optimization problem, was solved using Grid technologies by harnessing the power of more than 1,000 workstations distributed around the world.

Science today crosses conventional organizational boundaries and the Grid environment enables the creation of transient collaborations, also known as Virtual Organizations [14], where people and resources from diverse physical organizations are grouped together to pursue common goals.

Grid environments are very fluid: resources can come and go, and people can join or leave a Virtual Organization. Although the dynamic nature of the Grid is very desirable from the point of view of the ability to participate in it whenever possible, without making long-term commitments, it poses challenges to the users and applications. Further complicating issues, is the wide-area aspect of the Grid, where networks can often fail, effectively changing the set of resources available at any given time. In order to run on the grid, applications and users need to locate the resources available to them, schedule data movements and computations, monitor their progress and retrieve the results. These problems are made more complex because of the underlying assumptions of the Grid: lack of a single point of control, absence of central knowledge location, and lack of existing trust relationships between users and resources. In computational Grid, the focus has been mainly on the ability to schedule compute resources, to authenticate users and resources, to enable communications between Grid entities and support some level of fault tolerance. The greater computational power available through the harnessing of wide area resources enables the processing and analysis of ever greater quantities of data.

Data-intensive applications bring with them new challenges. Such applications currently deal with terabytes of data and will need to support petabytes of data in the near future. Clearly, new software systems are required to support such data-intensive applications. For example, finding the “fastest” machine to process the data might not be applicable in situations where data is so large that the cost of sending the data is significant. It might be easier to move the computation to the data, even if doing so means not having the greatest computing power available.

Another issue that needs to be addressed in Data Grids [1] is data replication, because of performance benefits as well as fault tolerance. As data sets grow, it becomes impractical to have a single copy of the data. Multiple replicas of the data are thus created and the issue of finding the “best” replica arises. Solving this problem might, for example, involve checking a *Replica catalog* [] to find all the possible locations of the desired data and then evaluating the cost of accessing the data at these locations by consulting the Network Weather Service (NWS) [17], which can provide network performance estimates. When dealing with large data sizes, the performance of data access protocols needs to be taken

into account as well, and as in the computational Grid, local and global policy constraints need to be respected.

The GriPhyN project, described in Section 2, aims to build a *Virtual Data Grid* (VDG) based on existing Grid technologies. VDG will not only provide users and applications with existing data products, but will also be able to materialize data products on demand. In this paper we describe the GriPhyN project and focus on how GriPhyN technologies will be used in large-scale physics projects such as LIGO.

Section 3 describes the LIGO experiment and its Virtual Data products. Obviously, realizing LIGO's Virtual Data brings many issues to light, and we look at some of them in Section 4. Many of these issues, such as security, resource discovery, and data replication can use currently available Grid components built as part of the Globus project[11, 10], the de-facto standard for Grid computing middleware. In that section, we also touch upon the new technologies that would enable LIGO to build a Virtual Data Grid. Finally, we conclude with some observations about the relationship between the Virtual Data and Data Grids.

2 GriPhyN's Virtual Data Concepts

The goal of the GriPhyN project (www.griphyn.org) is to enable a user and/or an application to ask for any domain-specific data without needing to know whether the data is available on some storage system or whether it needs to be computed. In the latter case, the system would, if possible, materialize the data by performing the necessary computations and data movements. The transparency in how the data is handled and processed is termed Virtual Data. The concept of Virtual Data is characterized by a large extent (national and worldwide). It layers new, sophisticated services on top of local policies, mechanisms and interfaces in order to use the grid resources in a coordinated fashion. In the extreme, the Virtual Data Grid recognizes that only raw data needs to exist, the other data products can then be derived from it.

GriPhyN (Grid Physics Network) is a multi-university and government laboratory project sponsored by NSF and tasked to realize the concept of virtual data, particularly in the context of four physics applications: the ATLAS (press.web.cern.ch/Atlas/Welcome.html) and CMS (Compact Muon Solenoid) (cmsinfo.cern.ch/Welcome.html/) experiments at the Large Hadron Collider at CERN, Switzerland, SDSS (Sloan Digital Sky Survey, web page), and LIGO (Laser Interferometer Gravitational-wave Observatory, web page). The first two experiments are in the area of high-energy physics. The scientific mission of these experiments is to find the origins of matter, discover new particles etc... The goal of SDSS is to construct a detailed map of a quarter of the sky, identifying millions of celestial objects and determining their characteristics such as brightness. Unlike the high-energy experiments which look at matter at its smallest scales, SDSS studies matter at its greatest scale. The goal for LIGO, which will be described further in Section 3, is to make the first (unambiguous) detection of gravitational waves and serve as a tool for astronomical observations.

The common thread among the experiments is that: a) The data is collected at a limited number of sites. b) The data generated is large (expected to reach petabytes in the near future.) c) The analysis of the data is very complex and requires the use of distributed resources. d) The community of scientists involved in the experiments is large and itself distributed around the world. e) Scientists need to access both raw data as well as processed data in an efficient way.

In that context, GriPhyN will provide access to data without the user needing to know whether the data is present on some storage system or on whether the data needs to be computed, which provides transparency with respect to materialization.

To support virtual data concepts, the GriPhyN project is conducting research in several areas:

- Data handling: how to identify the desired data, how to find it once it is identified, and finally in what form to deliver it.
- Request planning and scheduling: including finding the resources needed for the computation, locations of the data and scheduling the necessary data movements and computations.
- Request execution: once the plan has been constructed, fault-tolerant methods of plan execution need to be used.

As a result of the project, a Virtual Data Toolkit will be developed. This toolkit will encompass the basic elements needed to construct a Virtual Data Grid.

3 LIGO

LIGO (Laser Interferometer Gravitational-Wave Observatory) [9, 4] is a multi-university (Caltech, MIT, ...) project designed to build detectors for gravitational waves. Currently, there are two installations in the United States. More information on the project can be found at www.ligo.caltech.edu. The observatories aim to detect 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 source of gravitational waves is the motion of dense, massive astrophysical objects such as neutron stars or black holes. Waves generated at astrophysical distances from earth, however, are very weak by the time they reach the detectors, and thus far have not been directly detected. Another expected source of gravitational waves is the inspiral of one massive object, a neutron star say, into another massive object such as a black hole. Other signals may come from supernova explosions, quakes in neutron stars, and pulsars. It is even possible that relic gravitational waves left over from the big bang might one day be detected by earth-based or space-based interferometer gravitational wave detectors.

The existing LIGO interferometers both have two "arms", each being two or four kilometers in length, attached to a central station. Each of the arms contains test masses which move in response to the passing of a gravitational wave as it very slightly distorts the space in the region of the detector. A passing

gravitational wave is expected to reduce the distance between the test masses in one arm and increase it in the other (because of the L-shape of the arms and the tensor nature of gravitational waves), but only by less than the characteristic size of an atomic nucleus. To detect such a small change, a laser beam is generated at the central station (central point of the L), then split into two beams and sent into each arm of the interferometer. Later the two beams are recombined at the central station. The changes in the mass positions cause a misalignment or interference of the light in the two instrument arms and that interference is then detected after the beams are recombined. However, phenomena such as earthquakes, acoustic noise or laser fluctuations can also cause beam interference. In order to obtain a clean gravitational wave signal, significant amount of data needs to be collected (including data from seismometers, microphones, etc.) and analyzed (for example, to eliminate noise). The raw data collected during experiments is a collection of continuous time series at various frequencies. The amount of data expected to be generated and cataloged each year is in the order of tens of terabytes. The gravitational channel is less than 1% of all data collected.

3.1 Virtual Data Scenario in LIGO

A sample query to the LIGO Virtual Data Grid could be: *Conduct a pulsar search on the data collected from Oct 16 2000 to Jan 1 2001.*

The search is implemented as a pipeline of data transformations (Figure 1). As a first step, 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. The short-duration frames are then combined into much longer frames in a transpose operation, and further data-conditioning filters are applied.

This data is then used to conduct the pulsar search. Since the data needed to conduct the search is a long (~ 6 months, 2×10^{11} points) stretch of a single channel—the gravity-wave channel, the 1D time-series is broken into many small segments, and the power-spectra of these segments are stacked to make a large frequency-time image, perhaps 4×10^5 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 several Hz over periods of 1 day and 1 year, and also with a secular variation due to slowing of the source.

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 cataloged as an event data

structure, which describes the pulsar’s position in the sky, the signal-to-noise ratio, time etc...

3.2 Virtual Data Request

In the GriPhyN’s Virtual Data Grid, we assume that a request can be made for any of the data products along the data pipeline, without regard to whether the data actually exists or needs to be produced. Figure 2 depicts a prototype of the LIGO’s Virtual Data Grid components. We will explore the functionality of the components by concentrating on the various issues that need to be addressed in providing the users and applications with LIGO Virtual Data products. We note here, that to be able to access LIGO’s existing software resources, such as LDAS (the LIGO Data Analysis System), Globus interfaces for GridFTP and GRAM need to be implemented. GridFTP, described further in Section 4.2 allows for staging data in and out of LDAS using the GridFTP protocol. GRAM, the Globus Resource Allocation Manager [6] allows for remote scheduling of LDAS jobs, which perform data conditioning, filtering and many other types of data processing. In general, for each requested data value, the system needs to:

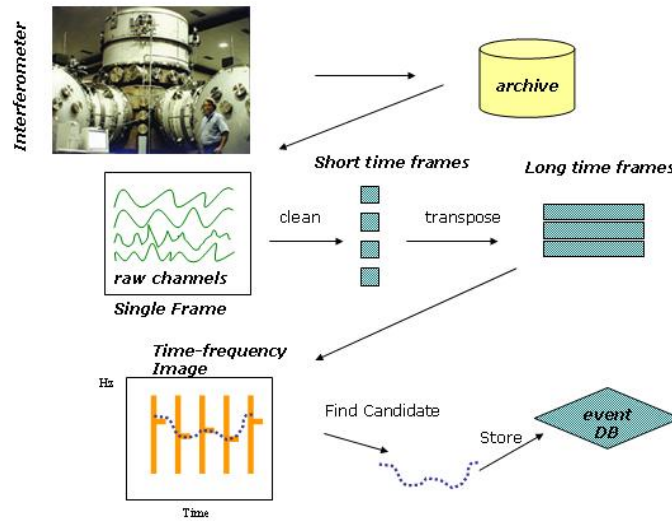


Fig. 1. A Pulsar Search in LIGO.

understand the request, determine if it is materialized; if so, where; if not, how to compute it. Then it needs to plan data movements and computations required to obtain all results and finally execute this plan.

Let’s assume that a request for a time-frequency image is made for a period of time from October 16, 2000 to January 1, 2001 for a frequency range of

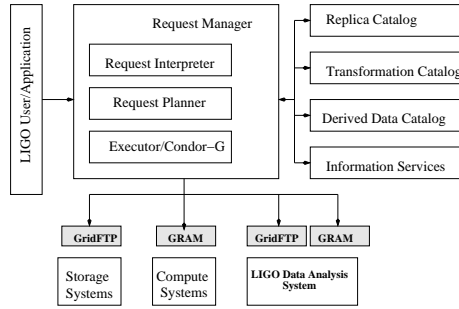


Fig. 2. LIGO's Virtual Data Grid Architecture Sketch.

0.5kHz-1.5kHz. Let's also assume that previously, a request for an image for September 20, 2000 to December 10, 2000 was made for that same frequency range. In our system, the request interpreter maps the request from the set of high-level, application-specific attributes to a set of specific data products (through the use of the application's Metadata Catalog). It is possible that the requested data product is already present, and can be delivered to the user. However, in our example, only a subset of the data has been materialized, so the remaining requested data (from December 21, 2000 to January 1, 2001) needs to be computed (if possible).

Using the *Derived Data Catalog* (currently in the design stage), which can capture the derived data materialization process, the system can determine that to materialize the frequency “fragments” composing the time-frequency image, Fast Fourier Transforms (FFTs) on specific long duration time intervals need to be performed. The system checks if the long duration time frames are available for the above time period. If so, FFTs need to be calculated over these time frames and the desired frequencies need to be extracted. The resulting data products need to be combined with the existing data and provided to the user. The resulting new data products also need to be made available to other users and applications by registering them in the Replica Catalog (described in Section 4.2).

Subsequent requests for any subset of the derived data will now be able to access the newly materialized data products. If the resulting data product is found to be accessed often from various locations, it might be advantageous to replicate it.

4 Virtual Data Issues

Clearly, there are many issues that need to be considered in GriPhyN's Virtual Data Grid. Here, we touch upon some of them and describe the available solutions. These are based mostly on the Globus software [11, 10], which is a project that provides the middleware necessary to build Grid infrastructures and appli-

cations. We also will rely on Condor-G [15], which provides a means of executing Globus jobs in a fault-tolerant way.

4.1 Security

Some issues such as security are relevant not only to the Virtual Data Grid, but any Data Grid. Users and applications need to be able to access only the data and compute resources they are authorized to access. Since scientific collaborations, such as LIGO, are composed of a variety of groups and individuals and often want to share their data with the general public, the system needs to differentiate between various types of users. For example, a few scientists in the LIGO collaboration have access to all the available data. When working with collaborators from other gravitational wave experiments around the world, possibly only a subset of the raw data need to be made available, such as environmental channels. The general public might only be allowed to see one or two raw data channels, or be limited to only derived data prepared in a particular way.

Issues get more complicated when Virtual Data products are materialized, since access now may involve not only reads of data, but possibly writes to storage systems and the use of compute resources. If the derived data are to be made available, Metadata Catalogs need to be updated. Obviously, since compute resources are used in the materialization process, only a few select users should be able to produce widely available data products.

The Globus Security Infrastructure, GSI [13], addresses mutual authentication issues, where users and resources can mutually authenticate using the public key infrastructure. Additional research is currently being conducted to be able to grant users capabilities based on their community membership.

4.2 Data Location and Access

The basic functionality that enables to deliver existing data to the user is data location and data access. In a large collaborative environment, such as the LIGO scientific collaboration, data will be replicated at various, geographically distributed, physical storage locations.

Replication serves various purposes. First it provides a basic level of fault-tolerance. When one storage system goes down, or its performance significantly degrades, one can still access replicas of the data. Second, it can improve data access performance. If, for example, a user can access a replica of the data at two different locations, where one provides a greater bandwidth and smaller latency than the other, for example a local storage system vs. a storage system accessible via the wide area, then it is advantageous to access the data from the system with the better network performance.

The Globus Replica Catalog [2, 3] provides a means of finding all the replicas of a desired logical file. Logical files are entities with globally unique names that may have one or more physical instances. The Replica Catalog provides simple mappings between *logical names* of files or *collections* of files and one or more copies of those objects on physical storage systems (*locations*). A logical

collection is a user-defined group of files. It is often convenient and intuitive to register and manipulate groups of files as a collection, rather than requiring that every file be registered and manipulated individually. This is especially the case in applications such as LIGO, where the data sets generated during the runs of the experiments are large and manipulating single files can become cumbersome. 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 Replica Catalog also includes optional entries that describe individual logical files.

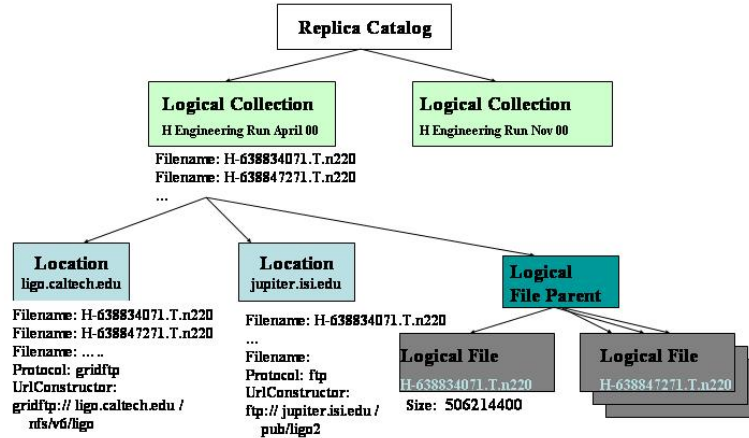


Fig. 3. LIGO's Replica Catalog.

Figure 3 shows an example of the LIGO replica catalog. We can see that the data is organized into collections based on the “runs” of the instrument. One of the collections is composed of files that were obtained during the run of the interferometer at Hanford, WA in April 2000. The logical files representing that data are *H-638834071.T.n220*, *H-638847271.T.n220* and others... Subsets of this collection are replicated at two locations: Caltech and ISI. The location information also indicates the URL constructor which is a string that provides a prefix, to which the logical filenames are appended to generate a complete URL needed to unambiguously access the data. Additionally, the locations specify the protocol for accessing the replicated data. The ISI location specifies the standard ftp protocol, whereas data at the Caltech server can be accessed via GridFTP [1, 2].

GridFTP is a data transport and access protocol built on top of the standard FTP protocol. It is integrated with the Grid Security Infrastructure, which enables data transfers to be done using the standard Grid security certificates, based on the public key infrastructure. In addition to the standard FTP function-

ality, GridFTP implements features that are invaluable in the Grid environment, such as: third-party control and data transfer, parallel data transfer, striped data transfer, partial file transfer and restartable data transfer.

4.3 Resource Discovery and Scheduling

Identifying which replicas are available can be done with the use of the Globus Replica Catalog. However, making decisions about which replica to access needs additional information, for example the status of the network connections, storage system performance, etc... This type of information can be obtained by using the existing Grid infrastructure such as the Network Weather Service (NWS) [17], a system that provides current network information and performs network performance predictions; and the Globus Metacomputing Directory Service (MDS) [7], which can provide information about compute and storage resources.

Information about compute resources is also needed to schedule the computation necessary to materialize the data. MDS can provide information about the load of the available machines, the amount of memory available, amount of free storage, type of CPU, OS, etc...

4.4 Request Planning and Execution

Decisions about how to materialize the data can be quite complex because of the many dimensions of the optimization problem. The data replicas and the software required for the materialization need to be selected. The *Transformation Catalog* [8] currently under design is analogous to the Replica Catalog in that it maintains a mapping between the logical transformations and the physical instances of the programs. These in turn are characterized by their location, the operating system that the program can be run under, compiler flags used for optimization, compiler used to generate the program, cost of running program and a logical expression specifying the resource requirements, such as the minimum number of processors, maximum number of processors. Another dimension of the planning problem is the scheduling of the compute resources. Currently, there are no planners geared specifically to the Grid environment and thus planning is a focus of research in GriPhyN.

Obviously, not all the plans that the Request Planner constructs would satisfy the user. One can imagine, that the user could make a request for a very large data set or data that would take very long to materialize. Therefore it is important that the planner consults the user and makes them aware of the expected duration of the request. This gives the user a chance to request a more reasonable data set. Another important communication between the planner and the user needs to be potentially made regarding the resources needed to satisfy the request. For example, the planner can schedule the materialization to be performed on a high-performance multi-processor system, where the user has only a limited allotment. In such situations, the user might prefer to have the request executed on a freely available cluster system, even though it might take longer to obtain the results.

Once a plan is constructed, it can be expressed in the form of a directed acyclic graph (DAG), suitable for submission to Condor-G DAGMan [15]. Condor-G DAGs are especially useful since task dependencies can be easily specified. These tasks can be composed of data movements, computations, updates to the various catalogs, and the like. Once the DAG is constructed it can be submitted to Condor-G DAGMan which will manage the execution of tasks in a fault-tolerant way. Condor-G DAGMan executes the tasks in the specified directed graph. If any particular task fails DAGMan marks that task as failed and does not execute any child tasks which depend on the failed parent. Condor-G DAGMan executes as much of the DAG as possible and if a failure of any task occurs a rescue DAG containing the failed parent and dependent child tasks is saved. This DAG can then be resubmitted as is, or it can be modified to take into account the observed system failure.

5 Conclusions

Although some projects, such as the Particle Physics Data Grid (www.ppdg.net) and the European Data Grid (www.eu-datagrid.org) address issues of the Data Grid, by providing transparent access to existing data, GriPhyN is the only project aims to provide transparency with respect to materialization.

In the GriPhyN's Virtual Data Grid, data suppliers, such as the LIGO collaboration, publish data to the Grid. Users can request raw or derived data from Grid, without needing to know where data is located, whether it is stored or computed. At the same time, the user can easily determine the cost of obtaining the data, the quality of the derived data product. The goal of VDG is to serve requests efficiently, subject to global and local policy constraints.

In this paper we examined the use of Virtual Data in the LIGO experiment. The goal of this work is to provide application scientist with enhanced data handling systems, geared towards large amounts of data, in the order of petabytes. LIGO's Virtual Data Grid builds on top of the standard Globus and Condor-G components, which form the Computational and Data Grids. In the future we will explore more complex data processing scenarios and research planning technologies applicable to Grid environments.

6 Acknowledgments

This work was supported by NSF under contract ITR-0086044, "GriPhyN:Grid Physics Network," (www.griphyn.org). Scott Koranda's work was also supported by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. We wish to thank all the members of the GriPhyN project for their valuable contributions.

References

1. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. "data management and trans-

- fer in high-performance computational grid environments.”. *Parallel Computing*, 2001.
2. W. Allcock, J. Bester, J. Bresnahan, A. 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.”. In *IEEE Mass Storage Conference*, 2001.
 3. W. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, A. Sim, A. Shoshani, B. Drach, and D. Williams. ”high-performance remote access to climate simulation data: A challenge problem for data grid technologies”. 2001. SC’2001.
 4. B. Allen, E. Deelman, C. Kesselman, A. Lazzarini, T. Prince, J. Romano, and R. Williams. ”ligo’s virtual data requirements.”. Technical Report 6, GriPhyN, 2001.
 5. O. Babaoglu and K. Marzullo. Consistent global states of distributed systems: Fundamental concepts and mechanisms. Technical Report UBLCS-93-1, University of Bologna, 1983.
 6. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82, 1998.
 7. Karl Czajkowski, Steven Fitzgerald, Ian Foster, and Carl Kesselman. Grid information services for distributed resource sharing. In *Proc. 10th IEEE Symp. on High Performance Distributed Computing*, 2001.
 8. E. Deelman, C. Kesselman, and G. Mehta. ”transformation catalog design for griphyn”. Technical Report 17, GriPhyN, 2001.
 9. E. Deelman, C. Kesselman, R. Williams, A. Lazzarini, T. A. Prince, J. Romano, and B. Allen. ”a virtual data grid for ligo”. volume 2110 of *Lecture Notes in Computer Science*, pages 3–12, 2001.
 10. I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl. Journal of Supercomputing Applications*, 11(2):115–128, 1997.
 11. I. Foster and C. Kesselman. The Globus project: A status report. In *Proceedings of the Heterogeneous Computing Workshop*, pages 4–18. IEEE Computer Society Press, 1998.
 12. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. 1999.
 13. I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *ACM Conference on Computers and Security*, pages 83–91. ACM Press, 1998.
 14. I. Foster, C. Kesselman, and S. Tuecke. ”the anatomy of the grid: Enabling scalable virtual organizations.”. *Intl. J. Supercomputer Applications*, 15(3), 2001.
 15. J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. ”condor-g: A computation management agent for multi-institutional grids”. In *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, 2001.
 16. ”MetaNEOS Project”. <http://www-unix.mcs.anl.gov/metaneos/nug30/pr.html>, 2001.
 17. 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. IEEE Press.