From Metadata to Execution on the Grid

Pegasus and the Pulsar Search

Ewa Deelman James Blythe Yolanda Gil Carl Kesselman Gaurang Mehta Karan Vahi

USC Information Sciences Institute, Marina Del Rey, CA 90292

Scott Koranda

University of Wisconsin Milwaukee, Milwaukee, WI 53211

Albert Lazzarini

Caltech, Pasadena, CA 91125

Maria Alessandra Papa

Albert Einstein Institute, Germany

1 Introduction

Grid computing has made great progress in the last few years. The basic mechanisms for accessing remote resources have been developed as part of the Globus Toolkit and are now widely deployed and used. Among such mechanisms are:

- Information services, which allow for the discovery and monitoring of resources. The information provided can be used to find the available resources and select the resources which are the most appropriate for the task.
- Security services, which allow users and resources to mutually authenticate and allows the resources to authorize users based on local policies.
- Resource management, which allows for the scheduling of jobs on particular resources.
- Data management services, which enable users and applications to manage large, distributed and replicated data sets. Some of the available services deal with locating particular data sets, others with efficiently moving large amounts of data across wide area networks.

With the use of the above mechanisms, one can manually find out about the available resources and schedule the desired computations and data movements. However, this process is time consuming and can potentially be complex. As a result it is becoming increasingly necessary to develop higher level services which can automate the process and provide an adequate level of performance and reliability.

The NSF-funded Grid Physics Network (GriPhyN, www.griphyn.org) project aims to develop just such services, in this paper we focus in particular on planners that can map complex workflows onto the Grid. In general, GriPhyN 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. 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.

The paper is organized as follows: first, we discuss in general the issues involved in mapping complex workflows onto the Grid. Then we describe Pegasus, the system we have developed to map domain

specific request onto the Grid. Section 4 describes in more detail the heart of Pegasus, which is an AI-based planner following by a description using Pegasus in the gravitational wave pulsar search. Section 6 gives a summary of our results and experiences demonstrating our system during the SC 2002 conference and we conclude with final remarks in Section 8.

2 Issues in Mapping Workflows onto the Grid

In general we can think of applications as being composed of application components. The process of application development (shown in Figure 1) can be described as follows. The application components are selected, their input and output file names are identified by their logical names (names that uniquely identify the content of the file, but not its location), and the order of the execution of the components is specified. As a result, we obtain an abstract workflow (AW), where the behavior of the application is specified at an abstract level.

Next this workflow needs to be mapped onto the available Grid resources, performing resource discovery and selection. Finally the resulting concrete workflow (CW) is sent to the executor for execution. In this section, we focus on the behavior of the concrete workflow generator (CWG) and its interaction with an executor, such as for example Condor-G/DAGMan [1]. Important issues are the relationship and interfaces between the planner and the executor, both from the standpoint of planning and fault tolerance. For this discussion, we assume that multiple requests to the system are handled independently.

One can imagine two extremes: on one hand, the planner can make an exact plan of computation based on the current information about the system. The planner would decide where the tasks need to execute, the exact location from where the data needs to be accessed for the computation etc… At the other extreme, the planner can leave many decisions up to the executor, it can for example give the executor a choice of compute platforms to use, a choice of replicas to access etc. At the time the executor is ready to perform the computation or data movements, the executor can consult the information services and make local planning decisions (in-time scheduling).

The benefit of the first approach (we term it full-plan-ahead), is that the planner can aim to optimize the plan based on the entire structure of the DAG, however, because the execution environment is very dynamic, by the time the tasks in the DAG are ready to execute, the environment might have changed so much that the execution is now far from optimal. Additionally, the data may no longer be available at the location assumed by the planner, leading to an execution-time error. If the planner constructs full plans, it must be able to adapt to the changing conditions and be able to quickly re-plan.

Faults due to the changing environment are far less likely to occur when the executor is given the freedom to make decisions as it processes the abstract workflow. At the time a task is to be scheduled, the executor can use the information services to find out about the state of the resources and the location of the data and make a locally optimal decision. However, because the executor does not have global information about the request it could make potentially expensive decisions.

Another approach is deferred scheduling, where the executor and the planner work together to come up with a plan. The planner provides an abstract workflow description to the executor, which, when it is ready to schedule a task, contacts the planner and asks for the execution location for the task. The planner can at that time make a decision, which would take into account the global information it has. Because the planner can make decisions at each time a task is scheduled, it can take many factors into consideration and use the most up-to-date information. The drawback however, is that the control might be too fine, and can result in high communication overheads and a large amount of computation due to re-planning.

Clearly, there is no single best solution for all applications, since these can have very different characteristics. For example, if we consider data-intensive applications, where the overall runtime is driven by data movement costs, then the full-plan-ahead planner can minimize the overall data movement by picking appropriate compute resources, resources “close” to the data. An in-time scheduler can also

schedule computation near the input data but without the knowledge of the overall data flow it will only try to achieve local optimization and might come up with an execution which is poor overall.

For compute-intensive applications, in-time-scheduling might be sufficient and optimal because the best compute resources can be found at a given moment in time and the time to stage the data is negligible.

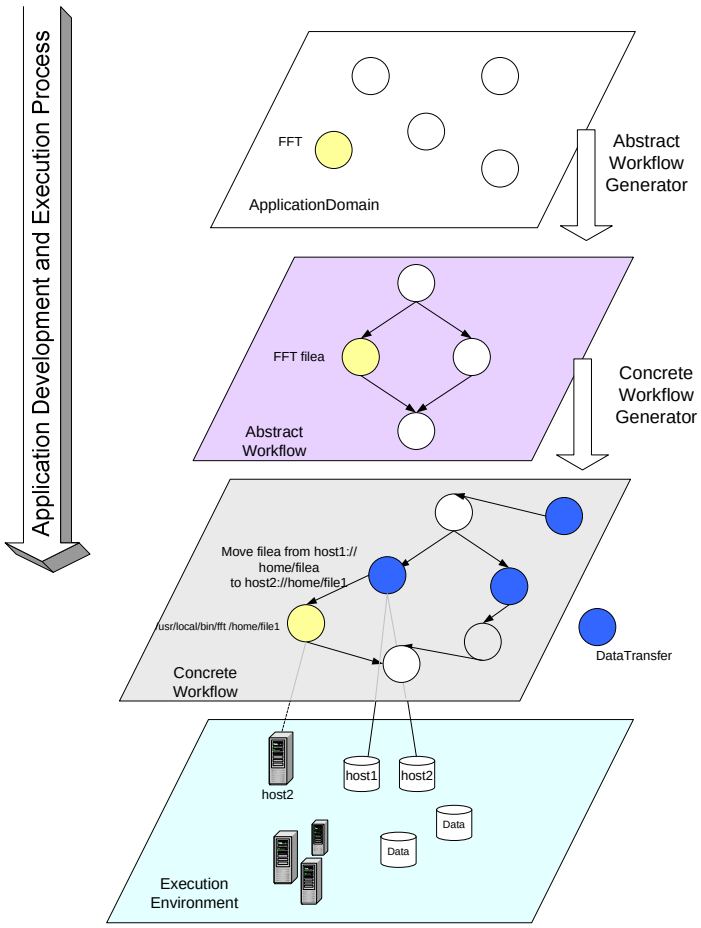

Figure 1: General view of application development in Grids.

Another factor in workflow management is the use of reservations for various resources such as compute hosts, storage systems and networks. As these technologies advance, we believe that the role of full-plan-ahead systems will increase.

Up to now, we have considered only the case where the workflow management system handles only one request at a time. The problem becomes more complex when the system is required to optimize across multiple requests and accommodate various usage policies and community and user priorities. In this case, full-plan-ahead planners have the advantage of being able to optimize the end-to-end workflows, however still facing the challenge of being able to react to the changing system state.

The nature of this problem seems to indicate that the workflow management system needs to be *flexible and adaptable* in order to accommodate various applications behavior and system conditions. Because the understanding of the application's behavior is crucial to the ability of planning and scheduling of the execution, application performance models are becoming ever more necessary.

In the next section, we describe the framework we have developed to explore the planning domain.

3 Pegasus

Pegasus, which stands for Planning for Execution in Grids, was developed at ISI as part of the GriPhyN project. Pegasus is a configurable system that can map and execute complex workflows on the Grid. Currently, Pegasus relies on a full-ahead-planning to map the workflows. As the system evolves, we will incorporate in-time scheduling and deferred-scheduling.

Chimera-Driven Pegasus

Pegasus was first integrated with the GriPhyN Chimera system [2]. In that configuration (see Figure 2), Pegasus receives an abstract workflow (AW) description from Chimera, produces a concrete workflow (CW), and submits it to DAGMan for execution. The workflows are represented as Directed Acyclic Graphs (DAGs). AW describes the transformations and data in terms of their logical names. CW, which specifies the location of the data and the execution platforms, is optimized by Pegasus from the point of view of Virtual Data. If data products described within AW are found to be already materialized (via queries to the Globus Replica Location Service (RLS)), Pegasus reuses them and thus reduces the complexity of CW. This optimization is performed in the "abstract DAG reduction" component. The "Concrete planner" component then consults the Transformation Catalog [3] to determine the locations where the computation can be executed. If there is more than one possible location, a location is chosen randomly. The Concrete Planner also adds transfer and registration nodes. The transfer nodes are used to stage data in or out. Registration nodes are used to publish the resulting data products in the Replica Location Service. They are added if the user requested that all the data be published and sent to a particular storage location.

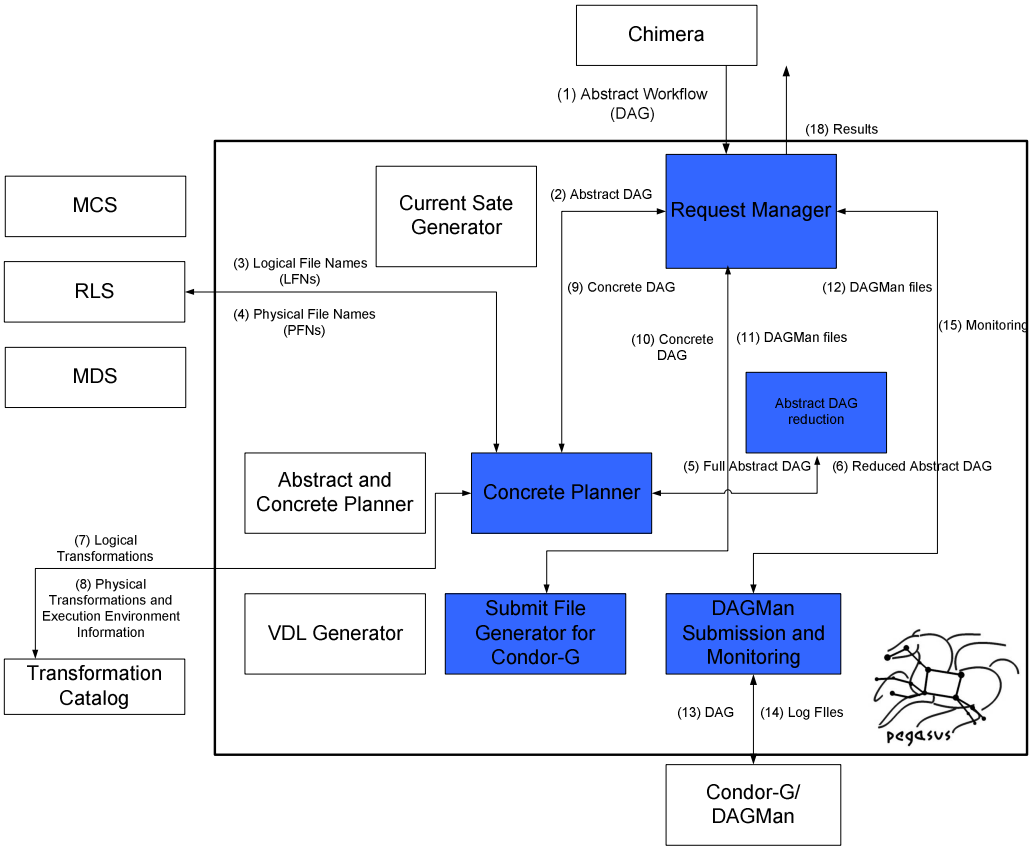

Figure 2: Configuration of Pegasus when driven by Chimera. Pegasus only generates the concrete workflow.

Once the resources are identified for each task, Pegasus generates the submit file for Condor-G. The resulting concrete DAG is sent to DAGMan for execution.

In that configuration, Pegasus has been shown to be successful in mapping workflows for very complex applications such as the Sloan Digital Sky Survey [4] and the Compact Muon Source [5].

Metadata Driven Pegasus

Pegasus can also be configured to perform the generation of the abstract workflow based on application-level metadata attributes. Given attributes such as time interval, frequency of interest, location in the sky, etc., Pegasus is currently able to produce any virtual data products present in the LIGO pulsar search, described in the Section 5. The figure below shows Pegasus configured for such a search.

Pegasus uses the Metadata Catalog Service (MCS) [6], newly developed at ISI, to perform the mapping between application-specific attributes and logical file names of existing data products. AI-based planning technologies, described in the next section, are used to construct both the abstract and concrete workflows.

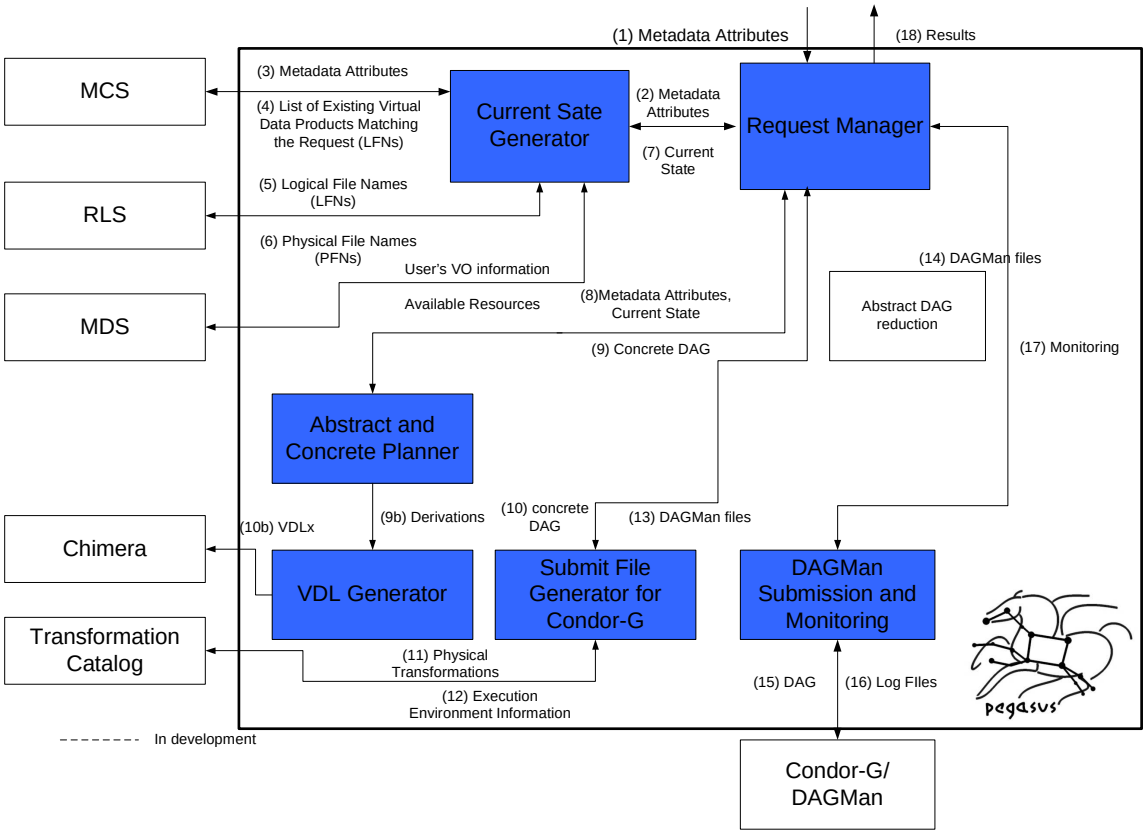

Figure 2: Configuration of Pegasus when used to construct both the abstract and concrete workflows.

The metadata for the final desired data product is specified to the Request Manager. For the SC2002 demo, this was specified through an http front end. The metadata specified was specific to the LIGO Pulsar search. Once the metadata for a product is specified, a Metadata Catalog Service (MCS) determines the corresponding logical file and determines the metadata and logical filenames for all other sub products which can be used to generate the data product (and returns them to the Current State Generator.) The Current State Generator then queries the Replica Location Service (RLS) [7] to find the physical locations of the logical files. We are also interfacing the Current State Generator to the Globus Monitoring and Discovery Service (MDS) [8] to find out about the available grid resources. Currently, this information is statically configured. The metadata and the current state information are then passed to

the Abstract and Concrete Planner through a XML interface. The planner generates the concrete workflow (in the form of a DAG) necessary to satisfy the user's request. The planner takes into account the state of the network, to come up with an optimal plan. This planner also reused existing data products where applicable. The plan generated also specifies the sites at which the job should be executed and refers to the data products in terms of metadata. This plan is then passed to the submit file generator for Condor-G. The submit file generator determines first the logical names for the data products by querying the MCS and then the physical names by querying the RLS. It in addition looks up the Transformation Catalog to get the complete paths for the transformations at the execution locations described in the concrete DAG.

Pegasus also contains a Virtual Data Language generator that can populate the Chimera catalog with newly constructed derivations. In this configuration, Pegasus also sends the concrete workflow to DAGMan for execution. We have configured Pegasus to support the LIGO and GEO pulsar searches. Details about the search can be found in [9].

As the result of execution of the workflow, the newly derived data products are registered both in the MCS and RLS and thus are made available to the following requests.

4 Planning Approach

The Abstract and Concrete Planner is implemented using the Prodigy planner [10]. The planner models the application components along with data transfer and data registration as operators. Each operator's parameters include the host where the component is to be run, so an output plan corresponds to a concrete workflow. In addition, some of the effects and preconditions of the operators capture the data produced by components and their input data dependencies. The state information used by the planner and gathered by the Current State Generator includes a description of the available resources and the relevant files that have already been created. The input goal description can include (1) a metadata specification of the information the user requires and the desired location for the output file, (2) specific components to be run or (3) intermediate data products. Several issues make this application domain challenging, we touch upon them as we describe the domain model in more detail.

In our initial work, we are using the Prodigy planner, because search heuristics play an important role in planning and Prodigy provides an expressive language. We also tested versions of the domain with the more recent planner FastForward [9, 11] and found Prodigy to be competitive for our purposes, as noted in another domain by [12].

State information

The planner's world state includes information about resources. Some state information changes slowly, such as the operating system or total disk space available on a resource, and some of the information can change in seconds or minutes, such as the available memory or queue length. In the long run the planner may need to reason about how the information can change over time, but in our initial implementation we only model the type of a host, network bandwidths and file information.

It is useful for the planning state to include metadata about the files for several reasons. As mentioned, the planner can assume the task of creating both the abstract and concrete workflows. It is also more appropriate to reason at the level of the metadata rather than at the level of the files that represent that data content. Rather than search for a file with appropriate characteristics, the components are linked to the characteristics themselves. This also avoids quantifying over the set of existing files, which may change during planning as objects are created and destroyed.

Goal statements

In most planning applications, goals refer to properties that should be true after the plan has been executed. For the planner, such goals include having a file described by the desired metadata information

on some host. However, it is also sometimes useful to specify goals that refer to intermediate components or data products, or for registering certain files. Thus the goal statement can specify a partial plan.

In principle, the goals given to the planning system may be those of a single user or the aggregated goals of a group of users, although we have not explored the latter case. In that case, the planner may be able to create a more efficient plan for the overall computations required by exploiting any synergy in the users' goals.

Operator descriptions

The operators themselves represent the concrete application of a component at a particular location to generate a particular file or a file movement across the network. Their preconditions represent both the data dependencies of the component, in terms of the input information required, and the feasible resources for running the component, including the type of resource. These operators capture information similar to that represented in Chimera's Virtual Data Language [2], such as the name of the component and its parameters. However, the operators also contain the additional information about the preconditions necessary for the use of the component, and provide the effect of the application of the component on the state of the system, such as the consumption of the resources. Further information about resource requirements, such as minimal physical memory or hard disk space, is a planned extension.

Plans generated in response to user requests may often involve hundreds or thousands of files and it is important to manage the process of searching for plans efficiently. If a component needs to be run many times on different input files, it is not useful for the planner to explicitly consider different orderings of those files. Instead the planner reasons about groups of files that will be treated identically. An auxiliary routine allocates the files to different groups, looking for a locally optimal allocation. Since the number of input files or groups may vary by component and even by invocation, the preconditions are modeled using quantification over possible files.

Solution space and plan generation strategy

Most planning systems are designed to produce a feasible plan given constraints on the possible actions, but do not attempt to optimize any measure of plan quality. In planning there may be many feasible plans and it is important to find a high-quality solution. The measure of a plan's quality may include several dimensions, including the performance in terms of the overall expected time to satisfy the user request, the reliability in terms of probability of failures and their impact on performance, and policy-related issues, for example not expending too much of a user's allowance on some precious resource if cheaper resources would be adequate. Helping users manage the tradeoff between these dimensions is a topic of future work. Our current system attempts to minimize the overall runtime of the plan. We can estimate the run-time of the plan based both on the expected runtime of individual components on the allocated resources and on the expected transfer time for files around the network.

In our initial approach, we seek high-quality plans with a combination of local search heuristics, aimed at preferring good choices for individual component assignments, and an exhaustive search for a plan that minimizes the global estimated run-time. Both aspects are necessary: without the global measure, several locally optimal choices can combine to make a poor overall plan because of conflicts between them. Without the local heuristics, the planner may have to generate many alternatives before finding a high quality plan.

5 LIGO and GEO pulsar search

LIGO (Laser Interferometer Gravitational-Wave Observatory, www.ligo.caltech.edu) [13, 14] 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. GEO 600 is a 600 meter interferometer installed in Hannover, Germany built by a British-German collaboration. The observatories' 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 will 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 in the order of tens to hundreds of terabytes. The gravitational 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.

Searching for pulsars, which may emit gravitational waves requires, among other things, a Fourier analysis of a particular set of frequencies over some time frame. To conduct a pulsar search, for example, the user must find a number of files of raw data output corresponding to this time frame, extract the required channel, concatenate the files and make a series of Fourier transforms (FT) on the result. The desired frequencies must then be extracted from the set of FT output files, and processed by a separate program that performs the pulsar search.

Depending on search parameters and the details of the search being conducted, a typical LIGO or GEO pulsar search may require thousands of Fourier transforms, some of which may have already been performed and stored at some location in the Grid. For good performance, this work must be divided between the suitable hosts that are available on the Grid, taking into account their different speeds and currently queued tasks. The results must be marshaled to one host for frequency extraction, and the final search must be executed on a different host because of the program requirements. In all, many gigabytes of data files may be generated, so a fast-running solution must take the bandwidth between hosts into account.

We have tailored the metadata-driven Pegasus to support LIGO and GEO pulsar searches. This involved developing application-specific operators for the planner and to provide a Globus interface to the LIGO data analysis facilities which are customized to the project needs. This included developing a new Globus jobmanager [15] to enable scheduling of jobs on the LIGO analysis system and providing a GridFTP [16] interface to stage data in and out of the system.

6 Results

The Metadata approach described in this paper was first demonstrated at the SC 2002 conference held in November at Baltimore. The Pegasus system was configured to generate both the abstract and the concrete work flows and run the LIGO and GEO Pulsar searches. For this demonstration the following resources were used:

- Caltech (Pasadena, CA): LIGO Data Analysis System (LDAS), Data Storage.
- ISI (Marina del Rey, CA) Condor Compute Pools, Data Storage, Replica Location Services, Metadata Catalog Services
- University of Wisconsin (Milwaukee): Condor Compute Pools and Data Storage.

The requests for pulsar searches were generated using an autogenerator which generated requests both for known pulsars (approximately 1300 known pulsars) as well as random point searches in the sky. A user

could also request a specific pulsar search by specifying the metadata of the required data product through a web based system. Both the submission interfaces as well as all the compute and data management resources were Globus GSI (Grid Security Infrastructure) enabled. Department of Energy (DOE) issued X509 certificates were used to authenticate to all the resources.

During the demonstration period and during a subsequent run of the system approximately 200 pulsar searches were conducted (both known as well as random) generating approximately 1000 data products involving around 1500 data transfers. The data used for this demonstration was obtained from the first scientific run of the LIGO instrument. The total compute time taken to do these searches was approximately 100 CPU hrs. All the generated results were transferred to the user and registered in the RLS, the metadata for the products in the MCS as well as into LIGO's own metadata Catalog. Pegasus also generated the corresponding provenance information using the Virtual Data Language and used it to populate in the Chimera Virtual Data Catalog.

The execution of the jobs was monitored by two means. For each dag, we had a start and end job added, which logged the start time and the end time for the dag into a mysql database. This information was then published via an http interface. We also implemented a shell script which parsed the condor log files at the submit host to determine the state of the execution and published this information onto the web.

7 Related Work

Central to scheduling large complex workflows is the issue of data placement, especially when the data sets involved are very large. In CWG we give preference to the resources where the input data set is already present. Others [17, 18] look at the data in the Grid as a tiered system and use dynamic replication strategies to improve data access. In [19] significant performance improvement is achieved when scheduling is performed according to data availability while also using a dynamic replication strategy.

While running a workflow on the Grid makes it possible to perform large computations that would not be possible on a single system, it leads to a certain loss of control over the execution of the jobs as they might be executed in different administrative domains. To counter this, there are other systems [20-23], which try to provide Quality of Service guarantees required by the user while submitting the workflow to the Grid. NimrodG uses the information from the MDS to determine the resource which meets the budget constraints specified by the user, while [23] monitors a job progress over time to ensure that guarantees are being met. If a guarantee is not being met schedules are recalculated.

Other work has focused on developing application specific schedulers, which maximize the performance of the individual application. In AppLeS [24], scheduling is done on the basis of a performance metric which varies from application to application. To schedule the jobs on the Grid, knowledge is required about resource usage. This leads to a customized scheduler for each application and not a general solution. Some schedulers have focused on parameter sweep applications, where a single application is run multiple times with different parameters [25]. Since there are no interdependencies between jobs, the scheduling process is far simpler from the one addressed here.

The European Data Grid (EDG) also has considered the automatic scheduling of jobs on the Grid which resulted in the development of their Workload Management Package [26]. The EDG Resource Broker [27] maps individual jobs based on information provided by various Grid Services such as MDS, Network Weather Service [28] and the Replica Location Services.

Each of the systems mentioned above are rigid because they use a fix set of optimization criteria. In this work we are developing a framework for a flexible system that can map from the abstract workflow description to its concrete form and can dynamically change the optimization criteria.

8 Conclusions and Future Work

The work presented here describes the Pegasus planning framework and its application to the LIGO and GEO gravitational wave physics experiments. The interface to the system was at the level of the application and AI planning techniques were used to map user requests to complex workflows targeted for execution on the Grid. The current planner employs full ahead planning, which as discussed in Section 2 might not be optimal. As part of our future work, we plan to investigate the planning space further.

In our demonstration we used scientifically meaningful data and used both generic Grid resources as well as LIGO specific resources enabled to work within the Grid. The results of our analysis were also fed back into the LIGO metadata catalogs for access by the LIGO scientists.

Although we were able to model the pulsar search within the planner, the issue of expanding this approach to other applications needs to be evaluated.

Acknowledgments

We gratefully acknowledge many helpful and stimulating discussions on these topics with our colleagues Ian Foster, Michael Wilde, Yong Zhao and Jens Voeckler. The Chimera system mentioned in this paper has been jointly developed by the University of Southern California Information Sciences Institute, the University of Chicago and Argonne National Laboratory. We would like to thank the following LIGO scientists for their contribution to the development of the grid-enabled LIGO software: Stuart Anderson, Marsha Barnes, Kent Blackburn, Philip Charlton, Phil Ehrens, Ed Maros, Greg Mendell, Mary Lei and Isaac Salzman. This research was supported in part by the National Science Foundation under grants ITR-0086044(GriPhyN) and EAR-0122464 (SCEC/ITR). LIGO Laboratory operates under NSF cooperative agreement PHY-0107417.

References:

[1] J. Frey, T. Tannenbaum, et al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids," presented at 10th International Symposium on High Performance Distributed Computing, 2001.

[2] I. Foster, J. Voeckler, et al., "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," presented at Scientific and Statistical Database Management, 2002.

[3] E. Deelman, C. Kesselman, et al., "Transformation Catalog Design for GriPhyN," Technical Report GriPhyN-2001-17, 2001.

[4] J. Annis, Z. Y, et al., "Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey.," presented at Supercomputing, Baltimore, MD, 2002.

[5] E. Deelman, J. Blythe, et al., "Mapping Abstract Complex Workflows onto Grid Environments," *to appear in the Journal of Grid Computing*, 2003.

[6] A. Chervenak, E. Deelman, et al., "A Metadata Catalog Service for Data Intensive Applications," GriPhyN technical report, 2002-11 2002.

[7] A. Chervenak, E. Deelman, et al., "Giggle: A Framework for Constructing Sclable Replica Location Services.," presented at Proceedings of Supercomputing 2002 (SC2002), 2002.

[8] K. Czajkowski, S. Fitzgerald, et al., "Grid Information Services for Distributed Resource Sharing," presented at 10th IEEE International Symposium on High Performance Distributed Computing, 2001.
[9] E. Deelman, K. Blackburn, et al., "GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists," presented at 11th Intl Symposium on High Performance Distributed Computing, 2002.
[10] M. Veloso, J. Carbonell, et al., "Integrating Planning and Learning: The PRODIGY Architecture," *Journal of Experimental and Theoretical AI*, vol. 7, pp. 81-120, 1995.
[11] 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.
[12] R. Aler and D. Borrajo, "On control knowledge acquisition by exploiting human-computer interaction," presented at Sixth International Conference on Artificial Intelligence Planning Systems,, 2002.
[13] A. Abramovici, W. E. Althouse, et al., "LIGO: The Laser Interferometer Gravitational-Wave Observatory (in Large Scale Measurements)," *Science*, vol. 256, pp. 325-333, 1992.
[14] B. C. Barish and R. Weiss, "LIGO and the Detection of Gravitational Waves," *Physics Today*, vol. 52, pp. 44, 1999.
[15] K. Czajkowski, I. Foster, et al., "A Resource Management Architecture for Metacomputing Systems," in *4th Workshop on Job Scheduling Strategies for Parallel Processing*: Springer-Verlag, 1998, pp. 62-82.
[16] W. Allcock, J. Bester, et al., "Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing," presented at Mass Storage Conference, 2001.
[17] K. Ranganathan and I. Foster, "Design and Evaluation of Dynamic Replication Strategies for a High Performance Data Grid," presented at International Conference on Computing in High Energy and Nuclear Physics, 2001.
[18] K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High Performance Data Grid," presented at International Workshop on Grid Computing, 2001.
[19] K. Ranganathan and I. Foster, "Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications.," presented at International Symposium for High Performance Distributed Computing (HPDC-11), Edinburgh, 2002.
[20] D. Abramson, R. Buyya, et al., "A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker.," *Future Generation Computer Systems*, vol. To appear.
[21] R. Buyya, D. Abramson, et al., "Nimrod-G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid," presented at HPC ASIA'2000, 2000.
[22] R. Buyya, D. Abramson, et al., "An Economy Driven Resource Management Architecture for Global Computational Power Grids," presented at The 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, USA,, 2000.
[23] P. Keyani, N. Sample, et al., "Scheduling Under Uncertainty: Planning for the Ubiquitous Grid.," Stanford Database Group.
[24] F. Berman and R. Wolski, "Scheduling From the Perspective of the Application," presented at High Performance Distributed Computing Conference, Syracuse, NY, 1996.
[25] H. Casanova, A. Legrand, et al., "Heuristics for Scheduling Parameter Sweep Applications in Grid environments," presented at 9th Heterogeneous Computing Workshop (HCW'2000), Cancun, Mexico, 2000.
[26] F. Giacomini and F. Prelz, "Definition of architecture, technical plan and evaluation criteria for scheduling, resource management, security and job description.," EDG Workload Management Draft. 2001.
[27] M. Ruda and e. al., "Integrating GRID tools to build a computing resource broker: activities of DataGrid WP1.," presented at CHEP 2001, Beijing, 2001.

[28] 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.