Pegasus and the Pulsar Search: From Metadata to Execution on the Grid

Ewa Deelman[3], James Blythe[3] Yolanda Gil[3], Carl Kesselman[3], Scott Koranda[4], Albert Lazzarini[2], Gaurang Mehta[3], Maria Alessandra Papa[1], and Karan Vahi[3]

[1] Albert Einstein Institute, Golm, Germany
papa@aei.mpg.de
[2] Caltech, Pasadena, CA 91125
lazz@ligo.caltech.edu
[3] USC Information Sciences Institute, Marina Del Rey, CA 90292
{deelman,blythe,gil,carl,gmehta,vahi@isi.edu}
[4] University of Wisconsin Milwaukee, Milwaukee, WI 53211
skoranda@gravity.phys.uwm.edu

Abstract. This paper describes the Pegasus workflow mapping and planning system that can map complex workflows onto the Grid. In particular, Pegasus can be configured to generate an executable workflow based on application-specific attributes. In that configuration, Pegasus uses and AI-based planner to perform the mapping from high-level metadata descriptions to a workflow that can be executed on the Grid. This configuration of Pegasus was used in the context of the Laser Interferometer Gravitational Wave Observatory (LIGO) pulsar search. We conducted a successful demonstration of the system at SC 2002 during which time we ran approximately 200 pulsar searches.

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 that allow for the discovery and monitoring of resources. The information provided can be used to find the available resources and select the resources that are the most appropriate for a given task.
- Security services that allow users and resources to mutually authenticate and allows the resources to authorize users based on local and global policies.
- Resource management that allow for the remote scheduling of jobs on particular resources.
- Data management services that 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 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 that can automate the process and provide an adequate level of performance and reliability.

The NSF-funded Grid Physics Network (GriPhyN [1]) project aims to develop just such services. In this paper we focus in particular on a workflow management system 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 describe Pegasus, the system we developed to map both domain-independent and domain-specific requests onto the Grid. Section 3 describes in more detail the heart of domain-specific Pegasus that includes an AI-based planner. Following is a description of applying Pegasus to the gravitational wave pulsar search. Section 5 gives a summary of our results and experiences demonstrating our system during the SC 2002 conference, and discusses the related work. We conclude with final remarks in Section 6.

2 Pegasus, a Workflow Mapping System

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. First, 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 (without identifying the resources needed for execution.) 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, such as Condor-G/DAGMan [21] for execution. Pegasus [10, 9, 18, 16], 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. Pegasus exists in two configurations: Abstract workflow-driven and metadata-driven.

2.1 Abstract Workflow-Driven Pegasus

In the first configuration, Pegasus is an application-independent software that takes an abstract workflow as input and produces a concrete workflow that

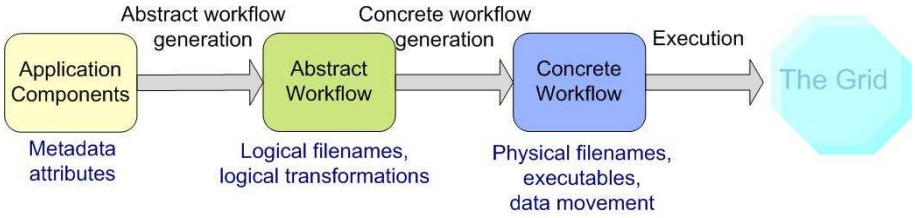

Fig. 1. Generating and Mapping Workflows onto the Grid.

can be executed on the Grid. This version of Pegasus is available as part of the GriPhyN Virtual Data Toolkit [3]. In that configuration, Pegasus receives an abstract workflow (AW) description from Chimera[20], 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)) [13], Pegasus reuses them and thus reduces the complexity of CW. Pegasus also consults the Transformation Catalog (TC) [19] to determine the locations where the computations can be executed. If there is more than one possible location, a location is chosen at random. Pegasus also adds data transfer and data registration nodes. Transfer nodes are used to stage data in or out. Registration nodes are used to publish the resulting data products in the RLS. They are added if the user requested that all the data be published and sent to a particular storage location. Once the resources are identified for each task, Pegasus generates the submit files for DAGMan. In that configuration, Pegasus has been shown to be successful in mapping workflows for very complex applications such as the Sloan Digital Sky Survey [6] and the Compact Muon Source [18].

2.2 Metadata-Driven Pegasus

Pegasus can also be configured to perform the generation of the abstract workflow based on application-level metadata attributes (See Figure 2.) Given attributes such as time interval, frequency of interest, location in the sky, etc., Pegasus is able to produce any virtual data products present in the LIGO Pulsar search, described in the Section 4.

Pegasus uses the Metadata Catalog Service (MCS) [26] 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. MCS is also used to determine the metadata and logical file names for all other sub products that can be used to generate the data product. Pegasus then queries the RLS to find

the physical locations of the logical files. The Globus Monitoring and Discovery Service (MDS) [14] is used to find the available Grid resources. The metadata and the current information about the Grid are used by the Pegasus planner to generate the concrete workflow (in the form of a DAG) necessary to satisfy the user's request. The planner reuses existing data products where applicable. The generated plan specifies the sites where the job should be executed and refers to the data products in terms of metadata. This metadata-defined plan needs to mapped to particular file instances. Pegasus determines the logical names for the input data in the plan by querying the MCS and then the physical names by querying the RLS. In addition it queries the Transformation Catalog to get the complete paths for the transformations at the execution locations described in the plan. Finally, the submit files for Condor-G/DAGMan are generated.

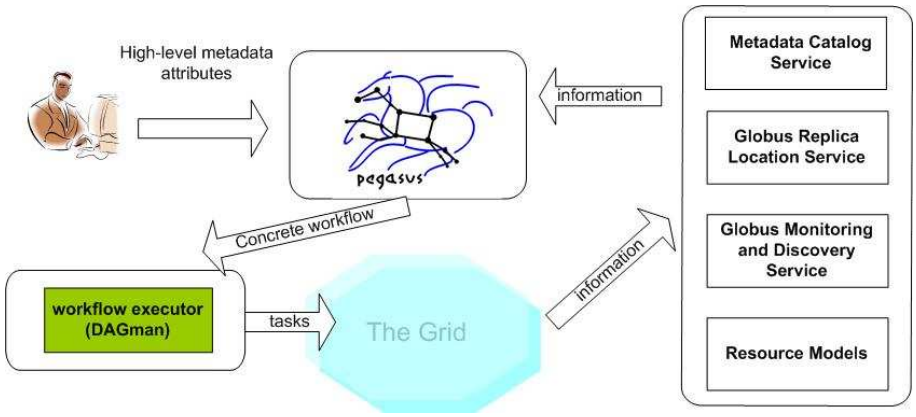

Fig. 2. Configuration of Pegasus when used to construct both the abstract and concrete workflows.

Pegasus contains a Virtual Data Language generator that can populate the Chimera catalog with newly constructed derivations. This information can be used for provenance tracking of the derived data products. Finally, Pegasus also sends the concrete workflow to DAGMan for execution. 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. We have configured Pegasus to support the LIGO and GEO pulsar searches. Details about the search can be found in Section 4 and [17].

3 Planning Approach

The Abstract and Concrete Workflow are generated using the Prodigy planner [27]. 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 and as a result 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 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 final 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.

3.1 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 job 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. First, the planner can assume the task of creating both the abstract and concrete workflows. Second, 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 by the characteristics themselves. This also avoids quantifying over the set of existing files that may change during planning as objects are created and destroyed.

3.2 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 data 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.

3.3 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 [20], 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.

3.4 Solution space and plan generation strategy

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 runtime. 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.

4 LIGO and GEO Pulsar Search

LIGO (Laser Interferometer Gravitational-Wave Observatory,) [2, 4, 7] 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 and analyzed along with the strain signal that measures the passage of gravitational waves. The amount of data that will be

acquired and cataloged each year is in the order of tens to hundreds of terabytes. Analysis on the data is performed in both time and Fourier domains.

Searching for pulsars that may emit gravitational waves requires, involves among others, 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 that may have already been performed and stored at some location in the Grid. 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 providing a Globus interface to the LIGO data analysis facilities that are customized to the LIGO project needs. This included developing a new Globus jobmanager [15] to enable scheduling of jobs on the LIGO analysis system and providing a GridFTP [5] interface to stage data in and out of the system.

5 Results and Related Work

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: 1) *Caltech (Pasadena, CA):* LIGO Data Analysis System (LDAS) and Data Storage. 2) *ISI (Marina del Rey, CA):* Condor Compute Pools, Data Storage, Replica Location Services, and Metadata Catalog Services. 3) *University of Wisconsin (Milwaukee):* Condor Compute Pools and Data Storage.

The requests for pulsar searches were generated using an auto generator that produced 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 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 in the order of 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 was registered 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 executable workflow, a start and end job were added. They logged the start time and the end time for the workflow into a MySQL database. This information was then published via an http interface. We also implemented a shell script that parsed the condor log files at the submit host to determine the state of the execution and published this information to the web interface.

5.1 Related Work

Central to scheduling large complex workflows is the issue of data placement, especially when the data sets involved are very large. In Pegasus we give preference to the compute resources where the input data set is already present. Others [23, 24] look at the data in the Grid as a tiered system and use dynamic replication strategies to improve data access. In [25] significant performance improvement are 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 [11, 22] that try to provide QoS guarantees required by the user while submitting the workflow to the Grid. NimrodG uses the information from the MDS to determine the resource that meets the budget constraints specified by the user, while [22] 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 that maximize the performance of the individual application. In AppLeS [8], scheduling is done on the basis of a performance metric that varies from application to application. 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 [12]. Since there are no interdependencies between jobs, the scheduling process is far simpler from the one addressed here.

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.

6 Conclusions and Future Work

The work presented in this paper 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. As part of our future work, we plan to investigate the planning space further, explore issues of planning for only parts of the workflow at a time, using dynamic system information to make more reactive plans, etc.

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 fed back to 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.

7 Acknowledgments

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. *GriPhyN.* http://www.griphyn.org.
2. *LIGO.* http://www.ligo.caltech.edu.
3. *The Virtual Data Toolkit.* http://www.lsc-group.phys.uwm.edu/vdt/home.html.
4. A. Abramovici et al. *LIGO: The Laser Interferometer Gravitational-Wave Observatory (in Large Scale Measurements).* Science, 1992. 256(5055): p. 325-333., 1992.
5. W. Allcock et al. *Data management and transfer in high performance computational grid environments,.* Parallel Computing Journal, 28, 5, 2002a, 749-771., 2002.
6. J. Annis et al. *Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey.* in Supercomputing. 2002. Baltimore, MD., 2002.
7. B. C. Barish and R. Weiss. *LIGO and the Detection of Gravitational Waves. Physics Today, 1999. 52(10): p. 44.* 1999.
8. F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. *Application-Level Scheduling on Distributed Heterogeneous Networks.* In Proceedings of Supercomputing'96 (Pittsburgh, 1996)., 1996.
9. J. Blythe, E. Deelman, Y. Gil, and C. Kesselman. Transparent grid computing: a knowledge-based approach. In *Innovative Applications of Artificial Intelligence Conference*, 2003.

10. J. Blythe, E. Deelman, Y. Gil, C. Kesselman, A. Agarwal, G. Mehta, and K. Vahi. The role of planning in grid computing. In *International Conference on Automated Planning and Scheduling*, 2003.
11. R. Buyya, D. Abramson, and J. Giddy. *An Economy Driven Resource Management Architecture for Global Computational Power Grids.* in The 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000). 2000. Las Vegas, USA,., 2000.
12. H. Casanova et al. *Heuristics for Scheduling Parameter Sweep Applications in Grid environments.* in 9th Heterogeneous Computing Workshop (HCW'2000). 2000. Cancun, Mexico., 2000.
13. A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney, and Giggle. *A framework for constructing scalable replica location services,.* Proceedings of Supercomputing 2002 (SC2002), November 2002., 2002.
14. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. *Grid information services for distributed resource sharing.* In Proceedings of the 10th IEEE 5'gmposium on High-Performance Distributed Computing, August 2001., 2001.
15. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. *A Resource Management Architecture for Metasystems.* Lecture Notes on Computer Science, 1998., 1998.
16. E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. *Grid Resource Management*, chapter Workflow Management in GriPhyN. Kluwer, 2003.
17. E. Deelman et al. *GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists.* in 11th Intl Symposium on High Performance Distributed Computing. 2002., 2002.
18. E. Deelman et al. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing, 2003.*, 1(1), 2003.
19. E. Deelman, C. Kesselman, and G. Mehta. *Transformation Catalog Design for GriPhyN.* GriPhyN technical report 2001-17, 2001.
20. I. Foster et al. *Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation.* in Scientific and Statistical Database Management. 2002., 2002.
21. J. Frey et al. *Condor-G: A Computation Management Agent for Multi-Institutional Grids.* in 10th International Symposium on High Performance Distributed Computing. 2001: IEEE Press., 2001.
22. P. Keyani, N. Sample, and G. Wiederhold. *Scheduling Under Uncertainty: Planning for the Ubiquitous Grid.,.* Stanford Database Group.
23. K. Ranganathan and I. Foster. *Design and Evaluation of Dynamic Replication Strategies for a High Performance Data Grid.* in International Conference on Computing in High Energy and Nuclear Physics. 2001., 2001.
24. K. Ranganathan and I. Foster. *Identifying Dynamic Replication Strategies for a High-Performance Data Grid.* In Proceedings of the Second International Workshop on Grid Computing (2001)., 2001.
25. K. Ranganathan and I. Foster. *Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications.* in International Symposium for High Performance Distributed Computing (HPDC-11). 2002. Edinburgh., 2002.
26. G. Singh et al. *A Metadata Catalog Service for Data Intensive Applications.* SC03, 2003.
27. M. Veloso, J. Carbonell, et al. Integrating planning and learning: The prodigy architecture. *Journal of Experimental and Theoretical AI*, 7:81–120, 1995.