Managing Large-Scale Scientific Workflows in Distributed Environments: Experiences and Challenges

Ewa Deelman, Yolanda Gil
USC Information Sciences Institute, Marina Del Rey, CA 90292,
deelman@isi.edu, gil@isi.edu

Abstract

In this paper we discuss several challenges associated scientific workflow design and management in distributed, heterogeneous environments. Based on our prior work with a number of scientific applications, we describe the workflow lifecycle and examine our experiences and the challenges ahead as they pertain to the user experience, planning the workflow execution and managing the execution itself.

1. Introduction

Scientific workflows are becoming a vehicle for enabling science at a large scale [1]. The scale can be measured in terms of the scale and scope of the scientific analysis itself and its complexity as well as in terms of the number of scientists and the number of organizations that collaborate in the process of scientific discovery. Workflows provide a representation of complex analyses composed of heterogeneous models designed by various individuals. At the same time, workflows have also become a useful representation that is used to manage the execution of large-scale computations. This representation not only facilitates overall creation and management of the computation but also builds a foundation upon which results can be analyzed and validated. Since workflows formally describe the sequence of computational and data management tasks, it is easy to trace back how particular data were derived.

Workflows can also become a tool capable of bringing sophisticated analysis to a broad range of users. Experts can: formulate workflows, set parameters of individual components, annotate the workflow

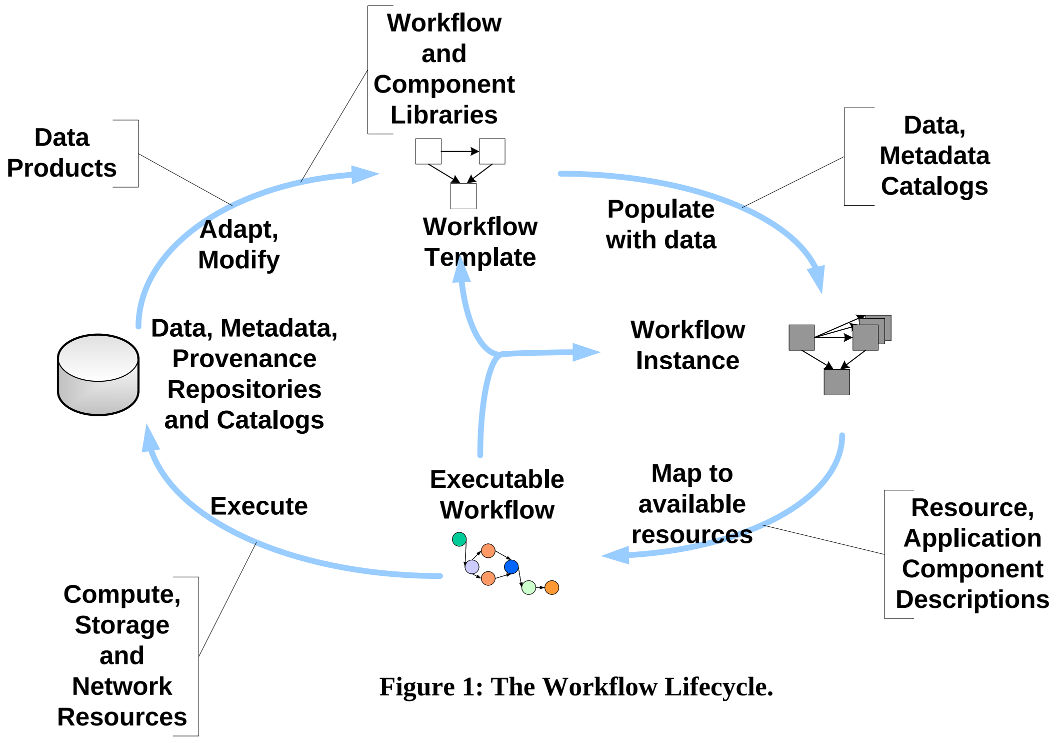
Workflow and Component Libraries
Data Products
Data, Metadata Catalogs
Adapt, Modify
Workflow Template
Populate with data
Data, Metadata, Provenance Repositories and Catalogs
Workflow Instance
Execute
Executable Workflow
Map to available resources
Resource, Application Component Descriptions
Compute, Storage and Network Resources

Figure 1: The Workflow Lifecycle.

components or the overall workflow, and validate the results. Once this process is completed, the newly developed workflows can be shared with other members of the community, other experts or even researchers and students that are not familiar with all the details of the analysis to the point where they can set all the necessary parameters themselves, but are fully able to make use of the workflow for their own work.

In this paper we describe the workflow lifecycle in terms of how workflows are defined, managed and executed, and illustrate several existing challenges and the solutions we are exploring for each of the lifecycle stages.

2. The Workflow Lifecycle

Figure 1 depicts the workflow lifecycle. Starting at the top of the figure, we can begin the process by composing a **workflow template**. A template describes the steps of analysis to be done by indentifying individual application components (at an abstract level—without identifying particular codes) and their order. An example template could indicate that first component would decimate a signal and then perform an FFT. This would be a workflow template composed of two application components and there would be a dependency between them (see Figure 2).

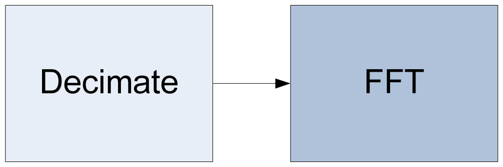

Figure 2: A Workflow Template.

When composing templates, users can access and search workflow and component libraries. Users can find "similar" templates that can be modified as well as components that can be added to existing templates. We assume that application developers would previously input the component descriptions into the component library. Workflow templates can be very useful within a collaboration because they allow for the sharing of the definition of an analysis. Experts can help develop and validate the templates, checking whether the overall logic of the analysis makes sense, and whether the parameters are set correctly. Users can also be assisted in validating the correctness of templates they create, either with syntactic checks or more complex semantic checks that take into account the constraints of the components. Once the templates are validated they can be re-used "as is" and referenced widely. It is also often the case that the templates are being developed and validated by a group of experts, where each individual brings their own knowledge about subparts of the problem.

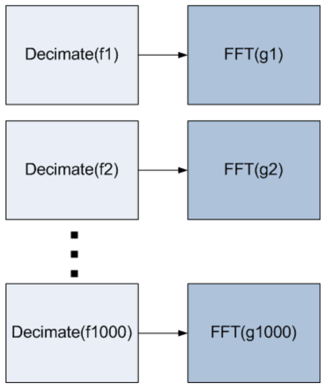

Figure 3: A Workflow Instance. Identifies the Workflow Steps and the Data Used in Each Step (f1, g1,f2,g2...).

Once the template is constructed, it needs to be populated with data, we call this form of the workflow a **workflow instance**. The instance describes the analysis exactly at the application level without indicating the resources needed to execute the analysis. Users can create a workflow instance by finding appropriate data using data and metadata catalogs. For example, a user could decide to run the two step workflow mentioned above on the data collected from experiment *X* during the time interval T. The user would query a metadata catalog (such as MCS [2] for example) to retrieve the identifiers of the data of interest (in our case these are often file names). These identifiers are then used to populate the template. In our example, if there were a thousand of identifiers of interest, then the workflow instance would be a two-level workflow composed of 2,000 nodes. At the first level we would have 1000 "decimate" nodes, each of which having a dependency to one of the 1000 "fft" nodes as illustrated in Figure 3. The workflow instance can also be represented more compactly through the use of the iteration construct, but for illustration purposes, we show the full instance. In our work we use Wings [3] to assist with the workflow template composition and workflow instantiation.

The workflow instance is important because it uniquely identifies the analysis to be conducted at the application level without including operational details of the execution environment. The instance can thus be published along with the results to describe how a particular data product was obtained. As such the workflow instance is key to reproducibility of scientific results.

The next step in the workflow lifecycle is to create the **executable workflow**. It is created by mapping the workflow instance onto the available execution resources. We assume that the execution environment is composed of distributed, heterogeneous resources. The mapping includes finding the appropriate software and computational resources where the execution can take place as well as finding copies of the data indicated in the workflow instance. This mapping usually involves adding nodes to the workflow to perform the desired functions. The mapping process can also involve workflow restructuring geared towards optimizing the overall workflow performance as well as workflow transformation geared towards data management and provenance information generation. The mapping process is usually automated (in our work we use Pegasus [4]) and involves the discovery of data replicas and the discovery of the available resources and their characteristics. Information about the application components and their locations are also necessary for the process of executable workflow generation.

Once the executable workflow is defined, it can be executed by a workflow engine which follows the dependencies defined in the workflow and executes the activities defined in the workflow nodes. The workflow engine (in our work we use Condor's DAGMan [5]) relies on the resources (compute, storage and network) defined in the workflow to perform the necessary actions. As part of the execution, the data is generated along with its associated metadata and any provenance information that is collected. This information can then be used to reuse the workflow in future analyses, either as is or modifying and adapting it, thus completing the workflow lifecycle.

We note that there are also other paths through the stages of the workflow and that a user can enter the cycle at different points. For example, the mapping from the workflow instance to the executable workflow may be done incrementally instead of in one shot. The user may develop a workflow instance or executable workflow directly. Also, based on the possible failures in the execution, the executable workflows, workflow instances or even workflow templates may need to be revised and the process of workflow refinement needs to be repeated.

We can broadly classify the process of workflow template and instance creation as user experiences, the mapping process as pre-planning and the execution as dynamic scheduling and monitoring. Based on these three general categories, we explore several challenges associated with them and describe our work in these areas.

3. Challenges in User Experiences

Challenges in providing users with satisfying experiences and enabling them to conduct their science efficiently and effortlessly stem from the fact that user expectations vary greatly. Users typically want to be able to focus solely on the scientific aspects of the problem, choosing appropriate components, structuring the overall analysis, selecting the necessary data, etc. However, some users may want to look deeper into the workflow lifecycle and guide the mapping process, perhaps indicating preferred compute resources and data sources.

At the same time, users often do not have a fixed analysis in mind, rather they converge on the final workflow through the process of exploration, perhaps developing different versions of the same workflow or modifying portions of the workflow as the computation progresses.

Finding information about the state of the workflow and being advised of any fatal failures or long delays are important feedback to provide to the users. The challenge is to supply the information at the right level of detail in a form that is easily understandable by the user (again the differences in user expectations make this challenging).

In our work, we explored the issues of user experiences from two angles: one is to adopt widely used portal technologies to provide users with high-level interfaces, and the second is to use semantic technologies to assist users with workflow composition.

Portals are being used today to deliver sophisticated capabilities customized for a wide range of users within a given community. They can also be used to provide an interface to the development and execution of workflow-based applications. One example of this work is the Montage portal [6], where the Montage application [7] which delivers science-grade mosaics of the sky is made available to the astronomy community. Figure 4 shows a generic view of portal that supports workflow-based applications. The user interface is customized to a given application domain and to a given community. Based on the user requests, a workflow instance is generated. In some portals, such as Montage, there exists support for only one application, but the user is able to select from a variety of data sources, for example, choosing images from different sky surveys. In other portals, such as the Earthworks portal [8] developed by the earthquake science community, users are able to configure a template by choosing amongst different application components (although the overall structure of the template is pre-defined).

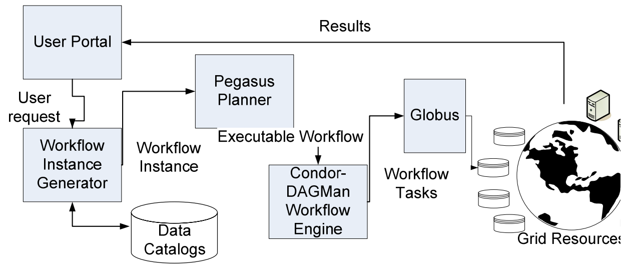

Figure 4: A Schematic of a Portal for Workflow-based Applications.

After the workflow instance is defined, it is passed to the Pegasus planner for mapping. The resulting executable workflow is then executed by Condor's DAGMan workflow engine. Finally the results are displayed to the user inside the portal or links to the data are presented.

However, in these portals the user has a limited choice of workflow templates. Many scientists desire more flexibility and also want to be able to explore alternative analysis, which means creating alternative workflow templates. Also, as we mentioned before, users may not know ahead of time the exact structure of the workflow and may need guidance in the way the workflows are constructed. In our work [3] we use semantic technologies to describe the application components and input data and use this information to guide the user in developing semantically correct workflows.

In some cases, it is impossible to develop and execute the entire workflow at once, sometimes because the user does not have enough information about the intermediate data to fully specify the workflow, sometimes because the instantiation of the workflow (populating the workflow template with data) can only be done once this data is generated. In such cases we can use Wings and Pegasus to iteratively instantiate and map the workflow.

Figure 5 shows the interaction between Wings and Pegasus when instantiating a workflow in an earthquake science application CyberShake[9]. Initially the first portion of the workflow is instantiated by Wings and sent to Pegasus for mapping. Once that portion is executed, the resulting data files are sent back to Wings and are used to instantiate the remaining portions of the workflow. The latter portion of the workflow can then be mapped and executed and the final results can be sent back to the user.

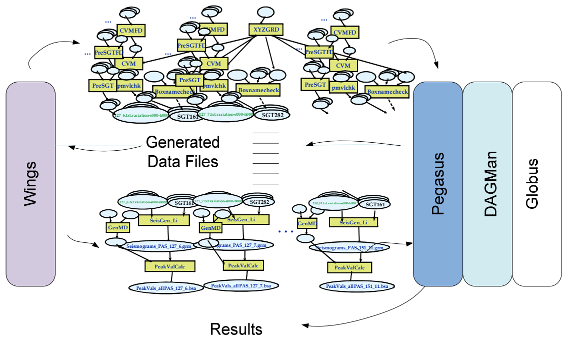

Figure 5: Iterative Workflow Instantiation, Planning and Execution.

4. Challenges in Workflow Planning

The challenges in workflow planning can be divided into two parts: feasibility and performance. In the first case we need to find a mapping of the workflow instance to an executable workflow that correctly identifies the needed resources and data and that manages the resulting data by staging them out to appropriate storage locations and by registering them in a catalog where they can be subsequently found. Obviously, faults in the environment can still occur, but the executable workflow generated by the planning process needs to be correct in order to minimize errors during execution.

Workflow planners also can optimize the workflow from the point of view of performance by scheduling individual jobs and the entire workflow onto resources in a way that optimizes the overall workflow performance [10]. One can sometimes increase the performance of the overall workflow when one can carefully schedule critical portions of the workflow. In data-intensive workflows making sure to schedule the computation close to the data is also important.

There can be cases where we could benefit from combining workflow-level with component-level optimizations. In particular, we are exploring issues of using search techniques to select the most appropriate implementations of both components and workflows. In terms of component optimization, we are relying on compiler and search technologies to select among implementation variants of the same computation [11]. We are also relying on knowledge-rich representations of components and workflow properties to guide the search for an optimal overall workflow.

Another form of optimization we explore within Pegasus is the clustering of jobs to increase the computational granularity of the workflows, and reduce the remote scheduling overhead of the individual jobs [12]. Clustering is a technique which we can also apply to data transfer jobs present in the workflow. In this case the optimizations can be also in terms of management of large number of jobs but also in terms of optimizing the file transfers by performing several transfers in parallel.

5. Challenges in Workflow Execution

One of the main challenges in workflow execution is the ability of documenting and dealing with failures. Failures can happen because resources go down, data becomes unavailable, networks go down, bugs in the system software or in the application components appear, and many other causes.

One approach to dealing with failures is tightening the interaction between the workflow planning and workflow execution processes, where only small portions of the workflow are planned ahead of time and then executed before progressing further. Although this approach does not avoid failures, it may provide more resilient mappings, by allowing for the inclusion of the latest information about the resources into the planning process. When failures occur, planning can also be redone. In Pegasus, we optimize the re-mapping process by keeping track of the intermediate data products [13] and then reducing the workflow based on the data produced so far. For example if a workflow composed of application components: *A*, *B* and *C*, each producing files f_1, f_2, and f_3 respectively, fails while C is executing, we can remap the original workflow (possibly scheduling C to another resource) without recomputing components A and B. Clearly, the intermediate data reuse can be very beneficial in terms of optimizing performance.

Another challenge in workflow execution is the collection of the information necessary to identify causes of failures so they can be avoided in future executions when possible. This would include placing monitoring components within all levels of workflow execution, from the workflow engine down to the application and system software components. Since the workflows are often running in very heterogeneous environments and rely on multi-level software stacks, collecting and interpreting this information is very challenging.

Instrumentation of the execution process can also be useful for providing provenance and performance information about the workflow components and entire workflows. The provenance information, which specifies how particular data was produced (which software was used, which input data) is necessary for scientists to be able to interpret, validate and reproduce the results. The performance information is useful for application component developers to identify problems with particular algorithms.

Workflow execution systems may also improve the performance of the workflow and support the scheduling performed by the workflow planner by providing resource provisioning capabilities [14]. Reserving compute resources ahead of the execution, removes some of the uncertainties associated with the scheduling of workflow components onto the resources. In the case of provisioned resources, the workflow tasks incur much reduced delays in the queues of the remote execution systems. These delays are also independent of the jobs submitted by other users of the resource.

6. Conclusions

In this paper we described several challenges associated with creating and managing large-scale workflows in distributed environments. We examined the problem from the point of view of user interactions, workflow planning and workflow execution. Although we are addressing some of these issues in our work, many challenges still remain.

In a recent NSF Workshop on the “Challenges of Scientific Workflows” [15], it was recognized that workflows are critical to capture complex scientific analysis processes in today’s computationally intensive and highly distributed science. In order to realize their full potential, the workshop report indicates that future workflow approaches will need to support the exploratory and dynamic nature of science and provide the provenance information necessary for result reproducibility.

7. Acknowledgments

We would like to thank the many collaborators who made various aspects of this work possible: Bruce Berriman, John Good, Mary Hall, Carl Kesselman, Jihie Kim, Phillip Maechling, Gaurang Mehta, David Meyers, Joanna Muench, Varun Ratnakar, Gurmeet Singh, and Karan Vahi.

We would also like to thank the National Science Foundation for the support of this work under grants: SCI-0455361, CNS- 0509517 and CNS- 0615412.

8. References

[1] Workflows for e-Science, Scientific Workflows for Grids," Springer, I. J. Taylor, et al. (eds), 2006, to appear.

[2] E. Deelman, et al., "Grid-Based Metadata Services," Proceedings of Statistical and Scientific Database Management (SSDBM), Santorini, Greece, 2004.

[3] J. Kim, et al., "Semantic Metadata Generation for Large Scientific Workflows," Proceedings of International Semantic Web Conference, 2006 (to appear).

[4] E. Deelman, et al., "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, pp. 219 - 237, 2005.

[5] J. Frey, et al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids." *Cluster Computing*, vol. 5, pp. 237-246, 2002.

[6] J. C. Jacob, et al., "Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking," *IJCSE*, 2006, to appear.

[7] B. Berriman, et al., "Montage: A Grid-Enabled Image Mosaic Service for the NVO," Proceedings of Astronomical Data Analysis Software & Systems (ADASS) XIII, 2003.

[8] J. Muench, et al., "SCEC Earthworks Science Gateway: Widening SCEC Community Access to the TeraGrid." TeraGrid 2006 Conference, 2006.

[9] E. Deelman, et al., "Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance tracking: The CyberShake Example," Proceedings of e-Science, Amsterdam, 2006 (to appear).

[10] J. Blythe, et al., "Task Scheduling Strategies for Workflow-based Applications in Grids," Proceedings of IEEE International Symposium on Cluster Computing and Grid (CCGrid), 2005.

[11] E. Deelman, et al., "A Systematic Approach to Composing and Optimizing Application Workflows," Proceedings of Workshop on Patterns in High Performance Computing, 2005.

[12] G. Singh, et al., "Optimizing Grid-Based Workflow Execution," *Journal of Grid Computing*, vol. 3, pp. 201-219, 2005

[13] E. Deelman, et al., "Pegasus : Mapping Scientific Workflows onto the Grid," Proceedings of 2nd EUROPEAN ACROSS GRIDS CONFERENCE, Nicosia, Cyprus, 2004.

[14] G. Singh, et al., "Application-level Resource Provisioning," Proceedings of eScience, Amsterdam, 2006 (to appear).

[15] "Workshop on the Challenges of Scientific Workflows," Arlington, VA 2006.http://vtcpc.isi.edu/wiki/, E. Deelman and Y.Gil (eds)