A Framework for the Design and Reuse of Grid Workflows

Ilkay Altintas[1], Adam Birnbaum[1], Kim K. Baldridge[1,2], Wibke Sudholt[2], Mark Miller[1], Celine Amoreira[2], Yohann Potier[2], and Bertram Ludaescher[1,3]

[1] San Diego Supercomputer Center, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA
{altintas, birnbaum, kimb, miller, ludaesch}@sdsc.edu
[2] Institute of Organic Chemistry, University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland
{kimb, wibke, ypotier, amoreira}@oci.unizh.ch
[3] Dept. of Computer Science & Genome Center, University of California at Davis, One Shields Ave, Davis, CA 95616, USA
ludaesch@ucdavis.edu

Abstract. Grid workflows can be seen as special scientific workflows involving high performance and/or high throughput computational tasks. Much work in grid workflows has focused on improving application performance through schedulers that optimize the use of computational resources and bandwidth. As high-end computing resources are becoming more of a commodity that is available to new scientific communities, there is an increasing need to also improve the design and reusability "performance" of scientific workflow systems. To this end, we are developing a framework that supports the design and reuse of grid workflows. Individual workflow components (e.g., for data movement, database querying, job scheduling, remote execution etc.) are abstracted into a set of generic, reusable tasks. Instantiations of these common tasks can be functionally equivalent atomic components (called *actors*) or composite components (so-called *composite actors* or *subworkflows*). In this way, a grid workflow designer does not have to commit to a particular Grid technology when developing a scientific workflow; instead different technologies (e.g. GridFTP, SRB, and *scp*) can be used interchangeably and in concert. We illustrate the application of our framework using two real-world Grid workflows from different scientific domains, i.e., cheminformatics and bioinformatics, respectively.

1 Introduction

With the increase in the volume of scientific data and knowledge, the demand to utilize the largest portion thereof in an efficient and simple way has become one of the main challenges in today's science. Many scientific domains need computing methods and resources for continued improvement of the quality of their research. Important examples include computational problems in bio- and cheminformatics. Technical challenges also arise through the introduction of different, heterogeneous distributed network computing systems that make up the Grid [1,2]. While an increasing number of computational tools for the Grid become available, they are generally difficult to

P. Herrero, M.S. Pérez, and V. Robles (Eds.): SAG 2004, LNCS 3458, pp. 120 – 133, 2005.

use for the domain scientist. Scientific workflow user environments, e.g., Kepler [7], Taverna [8], and Triana [9], aim at improving this situation by "wrapping" Grid tools and making them available in a user-friendly visual programming environment.

Grid Workflows can be seen as special scientific workflows that exhibit features of high-performance computing (HPC) workflows and/or high-throughput computing (HTC) workflows. While the focus of the former is on maximal peak performance, e.g., in terms of floating point operations per second (FLOPs), the latter can deliver large amounts of processing capacity over long periods of time [3]. HTC systems are effective for problems that deal with the management and tracking of data movements and the efficient assignment of tasks to resources.

We first discuss the practice of and the challenges in assembling HTC Grid workflows, and describe a Grid workflow framework that can help scientists develop HTC workflows for their research problems (Section 2). This is followed by a discussion of two real-world use cases from the cheminformatics and bioinformatics domains, respectively (Section 3). We conclude in Section 4 with a brief outlook on future work.

2 Grid-Workflow Framework

With the existing Grid infrastructure, building scientific applications for large-scale collaborative Grid workflows is complicated. Many scientists do not have the technical expertise to use the existing Grid components, so they need to recruit additional Grid expertise to assist them with their applications. One of the main reasons for these difficulties is that the basic Grid services to authenticate, access, manage, and discover remote resources are not easily obtained, nor easy to utilize once they are obtained. The goal of our Grid framework is to design abstract components and templates that facilitate Grid-based workflow construction, and to integrate multiple such Grid components into a single system with an intuitive graphical user interface (GUI).

Such a Grid workflow framework can be useful at several levels, e.g. as a modeling environment to capture the scientists' high-level ideas as a model of a scientific process, to design application-specific data analysis pipelines, or even to control the actual computational experiments, track the provenance of derived data, etc. The workflows generated by the system can be saved and reused in other studies. Another function of such a framework is to interface multiple technologies in one composition infrastructure, and use them interchangeably (e.g., GridFTP get vs. SRB get vs. *scp* etc.) To the best of our knowledge, ours is the first such workflow framework and system with this capability.

2.1 Grid Workflows: The Ingredients

We summarize below some common Grid service functions and then describe the abstract components that correspond to these functions in our framework.

Authentication. The Grid community has generated tools for authentication and authorization via generated proxy certificates. As summarized in [4], certificate management tools are developed for generating credentials for users and services, for getting users "signed up" to use a Grid, and for getting users' Grid credentials to wherever they are needed in a system. The Globus Toolkit [5] provides software de-

velopment kits for the core security software in Globus-based Grid systems and applications. These Software Development Kits (SDKs) include libraries and Java APIs for a certificate-based authentication system that conforms to the Grid Security Infrastructure (GSI) and that can be used to generate proxy components. The toolkit also provides a web services implementation of the same package.

As straightforward as it sounds, these tools still require programming in order to be used in an end-user application. An abstract component in a visual workflow programming environment simplifies such programming (see Section 3).

Data Movement. Access and management of remote data are basic functions in distributed Grid computing. There are several methods for moving data from one location to another, e.g., GridFTP, SRB put/get, *scp* and others. GridFTP is a secure data transfer protocol optimized for wide-area networks. The SDSC Storage Resource Broker (SRB) is a client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network and for accessing replicated data sets, e.g., based on metadata attributes [6]. While the former two are designed and optimized for file sharing of very large data over the Grid, sometimes a simple *scp* (secure copy) Unix command may be sufficient and easier to use for the scientist. *scp* is a shell command that allows users to copy files between systems quickly and securely, without the need for expertise in Grid systems. Such a tool can be as helpful in some workflows as any of the other file transfer mechanisms, even for data that will be used by a Grid job. Most systems provide interfaces to one or more of these tools. Ideally these methods should be usable interchangeably, depending on the user's needs, preferences, and abilities.

Remote Service Execution. Most scientists today are familiar with the use of web-based resources, and can make their work available through such distributed systems. However, manual copy/pasting or programming is usually required when using multiple of these resources in a data analysis or transformation pipeline. Often software developers are needed to write custom workflows that automate large-scale scientific workflows and processes. The use of generic tools for service-based execution can simplify the problem somewhat. A service is a component within the Internet computing model that provides a particular function through a simple remote invocation mechanism. Through the introduction of Web and Grid services, many new resources for different scientific domains are becoming available. However, services do not necessarily "fit together" (in the sense that they can be composed into a chain or pipeline of services) unless they have been designed to do so. Hence service composition (e.g., via "shim services") is an active field of research and development in scientific workflow systems [7, 8, 9].

The *ssh* (secure shell) Unix-command provides a simple way of connecting to a remote resource and executing a program there. A special actor component for this command is available in Kepler [7] as an easy way of executing a function on a remote machine, and getting back the results of the execution.

Grid Job Submission. A Grid job is an executable or command that runs on a (typically remote) Grid resource. The remote resource, also referred to as a 'contact' or 'gatekeeper', must have a Grid environment such as the Globus toolkit [5] installed to recognize this submission. Once submitted, a job can run in *batch* mode or *non-batch*

mode. The jobs submitted in batch mode are assigned a job-id, which is returned immediately and can be used for subsequent monitoring of the submitted job. The non-batch jobs return the result of the computation once they are finished. Batch mode submission is useful for jobs that take a long time, such as process-intensive computations [10]. The jobs to be submitted can be described using the Resource Specification Language (RSL), a common interchange language to describe resources.

Job Scheduling and Resource Management. In order for a high-throughput application to make use of distributed resources, a solution must exist for the scheduling problem, i.e. there must be a mapping between tasks and resources. Solving this problem in an ideal system has been shown to be NP-hard [39], and research has largely focused on the development of scheduling heuristics, which have been built into the commonly-used high-throughput systems such as Condor [3], Nimrod/G [11,14], and the AppLeS Parameter Sweep Template (APST) [12]. In building practical systems, it is difficult to isolate the issues of scheduling from those of managing the heterogeneity and instability of component subsystems. All three of the aforementioned high-throughput scheduling tools have the ability to constantly monitor and adjust to changing load and performance. In addition, they all provide some ability to monitor the state of the running application, as they all maintain job databases that may be polled and updated during experiment execution. It is one of our goals in this work to show how these systems can be leveraged in the construction of Grid workflows.

Fault Tolerance. Because Grid workflows depend on distributed computational resources under diverse controlling authorities, they are exposed to high risk of component failure, including failures in computational platforms, network, application services or the workflow system itself. Many of these issues fall into the domain of system and network administrators, who must design infrastructure to provide redundant components. In this work, we address only the parts that are under our control; in particular, the workflow system can retry to connect to failed resource or service after a certain amount of time. Redundant resources may be either found in a service registry, or may be hard-coded into workflows. The decision on which approach has to be taken depends on the fail-over policy/strategy of a particular workflow system.

Logging and Provenance. In scientific applications it is often necessary to keep track of data and processes that were used to produce the results of a computational experiment or scientific workflow, in particular to facilitate reproducibility. This *provenance* information can be associated with a result data set or workflow run, effectively providing an execution trace of certain crucial provenance information. Logging services, e.g., of the Globus Toolkit [13], can be customized and integrated into a scientific workflow system for this purpose. Such services provide interfaces to modify log filters and monitor and create views over previous logs.

User Interaction and Reporting. Scientific workflows may require user interaction at runtime, e.g., to determine which data subsets should be routed through which of several alternate paths of the workflow, or for computational steering. Workflow engines already maintain information of intermediate steps and execution details of processes. The challenge is to display that information in a way that it will satisfy the

needs of different users, with different detail levels and provisions for a variety of different publishing methods.

2.2 Component Composition and Interaction

Kepler [16] is an active cross-project collaboration bringing together several large-scale NSF/ITR projects (including SEEK [18], GEON [17], and ROADNet [19]), the DOE/SciDAC SDM project [20], and several other projects including Research Surge Enabled by Cyberinfrastructure (Resurgence) [21] and Encylopedia of Life (EOL) [22], to develop an open source scientific workflow system. The emerging Kepler system allows scientists from different domains (bioinformatics, cheminformatics, ecoinformatics, geoinformatics, astrophysics etc.) to design and execute scientific workflows. Scientific workflows can be used to combine data integration, analysis, and visualization steps into larger, automated "knowledge discovery pipelines" and "grid workflows" [23, 33].

Kepler is build on top of the mature Ptolemy II system developed at UC Berkley [24]. Ptolemy II is a system along with a set of APIs for heterogeneous hierarchical modeling. Not unlike the electrical circuit design, the focus of the Ptolemy II system is to build models of systems based on the assembly of predesigned components. These components are called *actor*s [25]:

> *"An actor is an encapsulation of parameterized actions performed on input data to produce output data. An actor may be state-less or state-full, depending on whether it has internal state. Input and output data are communicated through well-defined ports. Ports and parameters are the interfaces of an actor. A port, unlike methods in Object-Oriented designs, does not have to have a call-return semantics. The behaviors of a set of actors are not well-defined without a coordination model. A framework is an environment that actors reside in, and defines the interaction among actors."*

The interaction styles of actors are captured by *models of computation* (MoC). A MoC defines the communication semantics among ports and the flow of control and data among actors.

A framework implements a model of computation. Frameworks and actors together define a system [25]. In our Grid Workflow framework, we define a set of grid actors in Kepler that work in dataflow-based computation models such as Process Network (PN) and Synchronous Data Flow (SDF). *Director*s are responsible for implementing particular MoCs, and thus they define the "orchestration semantics" workflows. Simply by changing the director of a workflow, one can change the scheduling and overall execution semantics of a workflow, without changing any of the components or the network topology of the workflow graph.

The theoretical basis for the PN director are *Kahn Process Networks*. A process network is a directed graph, comprising a set of nodes (processes) connected by a set of directed arcs (representing FIFO queues). Each process executes a sequential program and is wrapped as a Ptolemy II actor. The one-way FIFO channels are used for the communication of processes and, in Kahn's process networks, have unbounded capacity, i.e., each channel can carry a possibly infinite sequence (a *stream*) of atomic data objects (*tokens*). Since channels have in principle unbounded capacity, writes to channels are non-blocking, while reads are blocking [26]. The PN domain in Ptolemy and the director implementing it in Ptolemy (and thus in Kepler) employ an extended

model due to Lee and Parks [27, 28]. The SDF domain is a special variant of PN in which a sequential execution order of actors can be statically determined prior to execution. This results in execution with minimal overhead, as well as bounded memory usage and a guarantee that deadlock will never occur.

2.3 Abstract Grid Workflow Actors

Grid workflows often exhibit similar flow patterns [29, 30], including the basic workflow patterns [31]. A very common scenario is the following: a user needs to copy (or *stage*) a set of files from one resource (e.g., the local environment) to a remote resource, run a computational experiment on that remote resource, and then fetch the results back to the local environment or copy them to another resource/database. We call these types of workflows *stage-execute-fetch* workflows. A script can implement a workflow that conforms to this pattern. However, a script does not specify the details of the scheduling of tasks and communication between the resources while the workflow is running. Also, scripts are often platform dependent and specific to a scenario, despite the fact that the pattern can be parameterized and used in many workflows. Users could more easily specify their own workflows via GUIs or a well-defined set of reusable components (actors) that can be connected to each other through some interfaces. The Kepler scientific workflow system, through its modeling foundation inherited from Ptolemy, provides an environment with such reuseable building blocks for Grid workflows. Motivated by the need to develop a simple, extensible, platform independent, and client-controllable grid workflow framework, we propose the following set of abstract actors that can be used as building blocks for the construction of Grid workflows.

Authenticate Actor. This component acts as a certificate source for other actors. All actors that use the same certificate can use the output of this actor. For the Globus Grid authentication, the actor initializes a proxy that creates a Globus proxy certificate from an X.509 key and certificate pair, when provided a pass-phrase. For SRB and remote database actors, this actor is generating the connection and can forward a connection token to the following steps in the workflow.

Copy Actor. This fundamental actor copies sets of files from one resource to another resource during workflow execution. The abstract copy actor can be instantiated to a simple FTP actor, a secure copy (*scp*) actor, a GridFTP actor, or an SRB-based put/get actor. For example, a GridFTP actor involves a Globus-grid proxy certificate, source and destination resources including directories, and a set of file names to be transferred. Similarly, SRBPut and SRBGet can be used to instantiate the abstract copy actor. Special variants include:

- **Stage Actor.** This variant copies files from the local host to a remote host.
- **Fetch Actor.** This variant retrieves files from the remote host to the local host.

Job Execution Actor. The purpose of this actor is to submit and run a remote job. Submission methods and clients can include special wrappers for *ssh*-based execution, web service-clients, Grid job runner proxies, and actors for Nimrod- and APST-based

submission. Kepler provides a variety of these instantiations, which have proven to be useful for remote job execution in different scientific application domains [32, 33].

Monitoring Actor. Monitoring actors and tools of our framework are designed to be scalable depending on user needs. We propose three different levels of monitoring, namely, *light*, *standard*, and *heavy*. In the standard monitoring level, the user is notified only if an actor fails to execute. Polling the job database of Nimrod/G or APST is an example mechanism for checking the state of execution of an actor. The overall workflow execution monitoring is done via a monitoring subsystem that interacts with the director. The light monitoring system is one that watches the execution but does not notify the user about failures until the workflow has finished or stalled. The heavy monitoring verbosely reports every communication between the workflow entities and also notifies the user about failures immediately.

Reporting Actor. The reporting actors work in coordination with the other actors to report regular intermediate results or exceptional conditions such as actor failures. This actor can also be implemented as a separate utility rather than as a Kepler actor. It talks to the monitoring unit and director, and allows users to report information wherever they would like, e.g., at a remote Grid resource, in a provenance database, or directly on a website.

Filter Actor. Filtering and subsetting data is a very common function. For example on a tuple stream, or a stream of XML elements, filtering corresponds to a selection operation σ. In contrast, cutting a certain region of interest from a map image can be seen as a data subsetting operation. A common requirement in Grid applications is to filter or subset data at the (usually remote) site of origin before passing it on to subsequent processing steps of the workflow.

Storage Actor. Once results are produced, they need to be stored on different resources, file systems, or databases. Sometime this step is preceded by a filtering step so that only interesting data will be saved. Stored information can include the primary data flowing through a workflow as well as process and provenance related metadata. Different incarnations of this actor can be used to save data on a number of storage devices, e.g. directly to a file system or databases, or indirectly to SRB.

Data Discovery Actor. Previously stored results should be searchable in various ways, e.g., through simple keyword based search, or more advanced ontology-based search mechanisms that "understand" how to expand a given search term (or a metadata annotation of a dataset). Since discovery of relevant datasets is very common tasks, the data discovery component is being integrated into the Kepler graphical user interface (i.e., Vergil, which is Ptolemy's GUI).

Service Discovery Actor. Kepler provides a web service harvester component for importing web services (or, more precisely, their interfaces) from a service repository or website. For the latter, the harvester can search text/html pages or repositories for appropriate links to WSDL web service descriptions. After parsing and analyzing these descriptions, the harvested web services appear in Kepler as any other actors; in particular, their input and output parameters and types are inferred from the WSDL

descriptions. Different operations from a single web service "package" are grouped together via the web service name and stored in the Kepler actor library. Once imported, web service actors are given a local LSID and can be annotated using a Kepler actor classification ontology. In this way, different dynamic view can be created on the actor library, depending on the chosen "view ontology" and the given search terms (concept names). Annotations can refer to a (web service) actor as a whole, or to the specific inputs and outputs of the actor. After the web service import is completed, actors representing the different operations can be searched, dragged and dropped onto Kepler's Vergil design interface, etc. like any other predefined actor.

Transformation and Querying Actors. When chaining together actor components to form larger workflows, consecutive actors or services do not necessarily "fit together". Data transformation actors and query actors can be used as "shims" to bridge structural and/or format mismatches between the output of a data producing actor and the input of a subsequent data consuming actor. Kepler provides various data transformation and querying actors, e.g., XSLT and Perl actors for data transformations, and XQuery and SQL actors for querying.

2.4 A Grid Workflow Pattern: *Stage-Execute-Fetch*

The abovementioned set of abstract Grid-related actors and their concrete instantiations allow a Kepler user to design and execute Grid workflows using a number of different tools, e.g., SRB for data handling including replica management, and Globus, Condor, and Nimrod, for remote execution and scheduling, respectively. In this way, the most suitable of a number of Grid tools become available to the scientist in a uniform manner. In addition to employing existing concrete Grid actors or their abstract counterparts[1], our framework for Grid workflows also includes *patterns* of Grid workflows. Such patterns correspond to *abstract workflows*, i.e. which might not be immediately executable and which involve abstract actors like the ones discussed above. An *abstract actor* can be seen as a "stub" or placeholder for a yet to be specified function, but whose input and output ports have already been pre-configured to capture the essential arguments of the operation. For example, the abstract copy actor will contain as inputs at least descriptions of the files/objects to be copied and their source and target locations. Concrete instantiations might require additional information, e.g., one or more authentication or connection tokens to access the various involved resources.

Using Ptolemy's hierarchical modeling capabilities, combined with the notion of abstract actors, larger templates of workflow patterns can be represented as abstract workflows. A very common pattern involves only three core abstract actors, i.e., *stage, execute,* and *fetch*, and is described as a linear chain of these actors. This Grid workflow pattern or template can be retrieved from the workflow repository (which is identical to the service/actor repository, modulo the fact that workflows are composite actors) and instantiated using suitable concrete actors to make the abstract workflow executable. The next section discusses in more detail two instantiations of this pattern, i.e., two real-world scientific workflows from very different domains.

[1] At design-time; they have to be substituted/instantiated with concrete ones at runtime.

3 Instantiating the Framework Using the Kepler Workflow System

The proposed Grid framework and its incarnation in Kepler have proven useful in different application domains, including those from computational chemistry and biology described below. Thanks to its generality, the approach and framework are applicable in other scientific domains as well.

3.1 Use Case 1: GAMESS Workflow for Quantum Chemistry

RESearch sURGe ENabled by CyberinfrastructurE (RESURGENCE) [21] is a project to develop a general workflow infrastructure for computational chemistry that allows high-throughput calculations distributed on computational grids. The project was initiated by the need to make the evolving technologies, such as computational grids and web services, available to scientists. In addition, such infrastructure provides a mechanism for researchers to couple different scientific codes within one overall calculation pipeline, spanning across domain sub-fields, input and output formats, and computational resources. The goal is thus to build a tool that provides a common user interface so that users do not have to be concerned with the particulars of grid computing, web services nor their associated underlying code, computational platforms, or with data file formats. However, the focus is not to generate complete predefined workflows, but large enough workflow chunks so that scientists can string them together according to their individual interests. With this purpose in mind, the Resurgence project became a part of the Kepler collaboration for developing common scientific workflow systems for a variety of disciplines [30, 34, 35].

The first target of the project is to build a pipeline from the base of Kepler composite actors, which automatically prepares and executes quantum chemical calculations for a number of molecules, with the individual input files generated on the fly (see Figure 1). For this, the *General Atomic and Molecular Electronic Structure System* (GAMESS) [36, 37] is employed, a program for ab initio molecular quantum chemistry. The program is an important internationally used software tool for the study of molecular and biomolecular research problems. Using this software, one can make reliable predictions of the structure, molecular properties and reactivity of molecules, which are useful for understanding complex problems in the real world. There are many standardized methods that can be invoked within the software package, and a very large variety of options and capabilities exist. Results of these computations can be compared to experimentally determined properties of the same type, used for predictions of properties before an experiment is performed, or, in some cases provide information that can not be obtained experimentally. As such, results of these computations can fill important gaps in our scientific knowledge. The software can be run on a variety of computer platform types, and many enhancements have been made to GAMESS both scientifically, as well as in terms of the latest middleware technology developments. Therefore, the software serves as an excellent testbed and driving application for further development of the Kepler system.

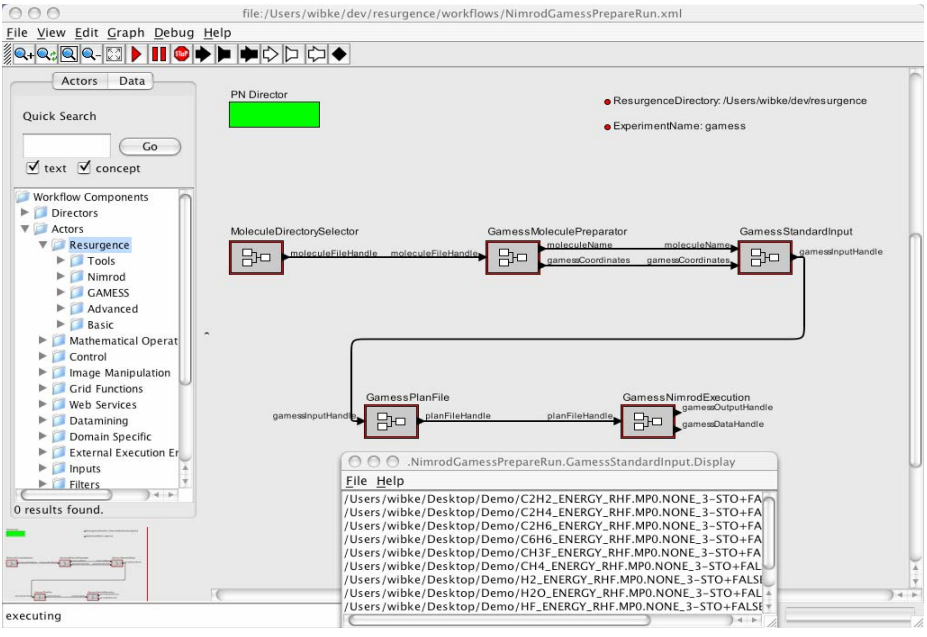

Fig. 1. Development version of the Resurgence GAMESS pipeline during execution

One principle of workflows in the Resurgence project is that the complex file preparation, transformation and analysis pipeline components should be mainly executed on the same machine where Kepler is running, while the highly compute-intensive molecular calculation pipeline components should be executed on dedicated, typically remote, compute servers, if possible. This allows access to helper tools that are often only installed on the central machine, to safeguard intermediate files after each workflow step and collect all outcomes in one place, but also means that input and output files have to be transferred back and forth between local and grid machines. This is of course an instantiation of the general stage-execute-fetch pattern, refined by other steps, including data transformations. For the preparation of GAMESS input files, the Open Babel program [38] is used to convert between different molecular file formats. For the execution of GAMESS jobs, the Nimrod/G toolkit [11, 14] is applied. For the future, there are plans to extend the Resurgence interface to additional molecular modeling software, particularly for the treatment of large biomolecules by classical mechanics. In addition, input and output data are planned to be directly read from and stored into molecular databases using concrete instances of the abstract storage actor.

3.2 Use Case 2: The Encyclopedia of Life/iGAP Workflow for Protein Sequence Annotation

It is hard to think of a better example of the explosion of data than computational molecular biology. Biologists are currently hard at work in digesting an over-abundance of

DNA and protein sequence data. One such effort is the Encyclopedia of Life Project (EOL) [22], the goal of which is to predict the three-dimensional protein structures for all of the genomes that have been sequenced to date. This is a calculation of such a huge scale that it requires the use of bleeding-edge grid technology and massively-parallel computation to access the requisite computational power. In previous work [40], we built a Workflow Management System daemon (WMSD) to manage the logistics of this large calculation. WMSD selects sequences from an input database, and continuously feeds many thousands of tasks to APST. APST manages the low-level complexities of job submission, heterogeneous resource management, and scheduling.

In the present work, we have integrated this workflow system with Kepler. Our ultimate aim is to provide biologists with the ability to set up a flexible pipeline of analysis tasks, which are then executed on a large scale for a huge number of input sequences. Since this is a long-running system, a key requirement is the ability to recover from major system failure – an instance of the monitoring actor. This is partially addressed by the fact that WMSD stores its state in an Oracle database, making it possible to recover from a failure of APST. In such a case, Kepler enables the automation of higher-level error recovery mechanisms. For example, after correcting the problem that caused the jobs to fail, it is easy for a scientist to insert actors to reset jobs with a "failed" state to "new", which would cause the WMSD actor to resubmit these jobs to APST at the next update.

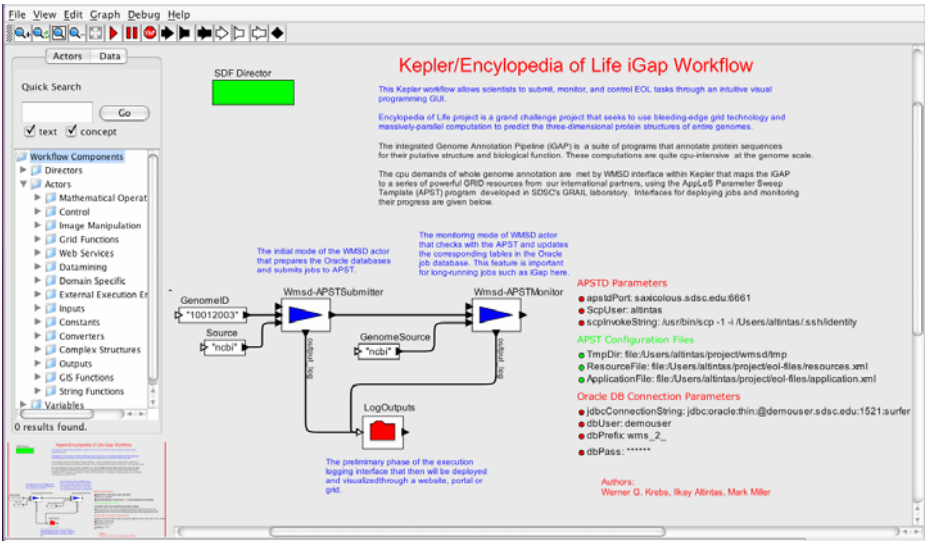

Fig. 2. The Encyclopedia of Life iGAP workflow

In the first implementation of this system, the workflow of tasks to be executed for each genome sequence is hard-coded into logic in the WMSD. The benefit of Kepler to date has been primarily in the areas of error recovery and resource management. In the future, we would like to allow the scientist to specify these workflows using the full power of the Kepler system. In essence, this would allow users to assemble work-

flows consisting of "pseudo-actors", whose sole behavior would be to emit WMSD configuration files specifying the workflow to be executed in high-throughput mode.

4 Discussion and Outlook

We have described a framework for Grid workflows based on abstractions of common Grid workflow components such as authentication, data movement, and remote execution, and monitoring. Abstract workflows, consisting of abstract and possibly concrete actors provide the workflow designer with common components and workflow patterns that can be reused and instantiated to create executable Grid workflows. A main advantage of this approach is that (a) it frees the designer from making technology decisions early on in the design process, and (b) at instantiation time, it allows the user to chose and even combine different concrete technologies such as Globus, SRB, and Nimrod. In future work we plan to automate the instantiation process of our framework using a reasoning approach that aims at automating the "wiring" of different actor instances, based on their semantic port types [41].

References

1. Berman, F., Wolski, R., Casanova, H., Cirne, W.,Dail, H., Faerman, M., Figueira, S., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S., Spring, N., Su, A., Zagorodnov, D. : Adaptive computing on the Grid using AppLeS. Parallel and Distributed Systems, IEEE Transactions on, Vol. 14, Issue 4, 369-382, April 2003.
2. F. Berman, G. Fox, and A. Hey, editors. Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, 2003.
3. The Condor Project Homepage: http://www.cs.wisc.edu/condor/
4. GRIDS: Grid Research Integration Deployment and Support Center, The Grid Ecosystem: Software Components for Grid Systems And Applications: http://www-unix.grids-center.org/r6/ecosystem
5. The Globus Toolkit: http://www-unix.globus.org/toolkit/
6. Storage Resource Broker: http://www.npaci.edu/DICE/SRB/
7. Kepler Project: http://kepler-project.org
8. Taverna Project: http://taverna.sourceforge.net
9. Triana Project: http://www.trianacode.org/
10. Vladimir, S.: Grid Job submission using the Java CoG Kit, IBM Developer Works
11. Nimrod/G Project: http://www.csse.monash.edu.au/~nimrod/nimrodg/
12. AppLeS Parameter Sweep Template (APST) Project: http://grail.sdsc.edu/projects/apst/
13. Configuring Globus Toolkit Logging Services: http://www-unix.globus.org/toolkit/docs/3.2/core/admin/configuringlogging.html
14. Abramson, D., Giddy, J., Kotler, L.: High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?, IPDPS'2000, Mexico, IEEE CS Press, 520-528, USA, 2000.
15. Schwiegelshohn, U., Yahyapour, R..: Attributes for Communication Between Scheduling Instances, in Global Grid Forum (GGF), December 2001.
16. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S.: Kepler: An Extensible System for Design and Execution of Scientific Workflows, In the 16th Intl. Conference on Scientific and Statistical Database Management(SSDBM), Santorini Island, Greece, June 2004.

17. NSF/ITR: GEON: A Research Project to Create Cyberinfrastructure for the Geosciences, http://www.geongrid.org
18. NSF/ITR: Enabling the Science Environment for Ecological Knowledge (SEEK), http://seek.ecoinformatics.org
19. ROADNet: Real-time Observatories, Applications and Data Management Network, http://roadnet.ucsd.edu
20. Scientific Data Management (SDM) Center, http://sdm.lbl.gov/sdmcenter
21. Resurgence Project Home Page: http://www.resurgence.unizh.ch/~resurgence/
22. EOL Project: http://eol.sdsc.edu
23. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S.: Kepler: Towards a Grid-Enabled System for Scientific Workflows, In the Workflow in Grid Systems Workshop in GGF10 - The Tenth Global Grid Forum, Berlin, Germany, March 2004.
24. 24.E.A. Lee et al., Ptolemy II project and system, Department of EECS, UC Berkeley, http://ptolemy.eecs.berkeley.edu/ptolemyII
25. Liu, X., Liu, J., Eker, J., and Lee, E. A.: Heterogeneous Modeling and Design of Control Systems, in Software-Enabled Control: Information Technology for Dynamical Systems, Tariq Samad and Gary Balas (eds.), Wiley-IEEE Press, April 2003.
26. G. Kahn, “The Semantics of a Simple Language for Parallel Programming”, Proceedings of International Federation for Information Processing Congress 74, pp. 471-475, North Holland Publishing Co., Aug 1974.
27. E.A. Lee and T.M. Parks, “Dataflow Process Networks”, Proceedings of the IEEE, Vol. 83 No. 5, pp. 773-801, May 1995.
28. Hylands, C., Lee, E. A., Liu, J., Liu, X., Neuendorffer, S., Xiong, Y., Zheng, H. (eds.): Heterogeneous Concurrent Modeling and Design in Java (Volume 3: Ptolemy II Domains), Technical Memorandum UCB/ERL M03/29, University of California, Berkeley, CA USA 94720, July 16, 2003.
29. van Laszewski, G., Amin, K., Hategan, M., Zaluzec, N., J., Hampton, S., Rossi, A.,: GridAnt: A Client-Controllable Grid Workflow System, 37th Hawaii International Conference on System Sciences (HICSS-37), Hilton Waikoloa Village, Island of Hawaii, January 2004.
30. K. K. Baldridge, W. Sudholt, J. P. Greenberg, C. Amoreira, Y. Potier, I. Altintas, A. Birnbaum, D. Abramson, C. Enticott, S. Garic, "Cluster and Grid Infrastructure for Computational Chemistry and Biochemistry", in "Parallel Computing for Bioinformatics", A. Y. Zomaya (Ed.), John Wiley & Sons, submitted for publication
31. van der Aalst, W., M., P. , Barros, A., P., ter Hofstede, A., H., M., and Kiepuszewski, B.: Advanced Workflow Patterns, in Conference on Cooperative Information Systems, pp. 18–29, 2000.
32. I. Altintas, E. Jaeger, K. Lin, B. Ludaescher, A. Memon, A Web Service Composition and Deployment Framework for Scientific Workflows, In the 2nd Intl. Conference on Web Services (ICWS), San Diego, California, July 2004.
33. B. Ludaescher, I. Altintas, C. Berkely, D. Higgins, E. Jaeger, M. Jones, E.A. Lee., J. Tao, Y. Zhao, Scientific Workflow Management and the KEPLER System, special issue of Distributed and Parallel Systems, to appear, 2005.
34. K. K. Baldridge, J. P. Greenberg, W. Sudholt, S. Mock, I. Altintas, C. Amoreira, Y. Potier, A. Birnbaum, K. Bhatia, M. Taufer, "The Computational Chemistry Prototyping Environment", Proceedings of the IEEE Special Issue on Grid Computing, in print
35. W. Sudholt, K. K. Baldridge, D. Abramson, C. Enticott, S. Garic, C. Kondric, D. Nguyen, "Application of grid computing to parameter sweeps and optimizations in molecular modeling", Future Generation Computer Systems 21 (2005) 27-35

36. Schmidt, M. W., Baldridge, K. K., Boatz, J. A., Elbert, S. T., Gordon, M. S., Jensen, J. H., Koseki, S., Matsunaga, N., Nguyen, K. A., Su, S. J., Windus, T. L., Dupuis, M., Montgomery, J. A., J. Comput. Chem. 1993, 14, 1347-1363
37. GAMESS Home Page: http://www.msg.ameslab.gov/GAMESS/
38. Open Babel: A Package to Decypher Computational Chemistry: http://openbabel.sourceforge.net/
39. O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonindentical processors," *Journal of the ACM*, 24(2): 280-289, Apr. 1977.
40. A. Birnbaum, J. Hayes, W. W. Li, M. A. Miller, P. W. Arzberger, P. E. Bourne, H. Casanova. To appear in *Proceedings of LNCS, Springer Lecture Notes in Computer Science, 2005*.
41. S. Bowers and B. Ludäscher, An Ontology-Driven Framework for Data Transformation in Scientific Workflows, Intl. Workshop on Data Integration in the Life Sciences (DILS'04), March 25-26, 2004 Leipzig, Germany, LNCS 2994.