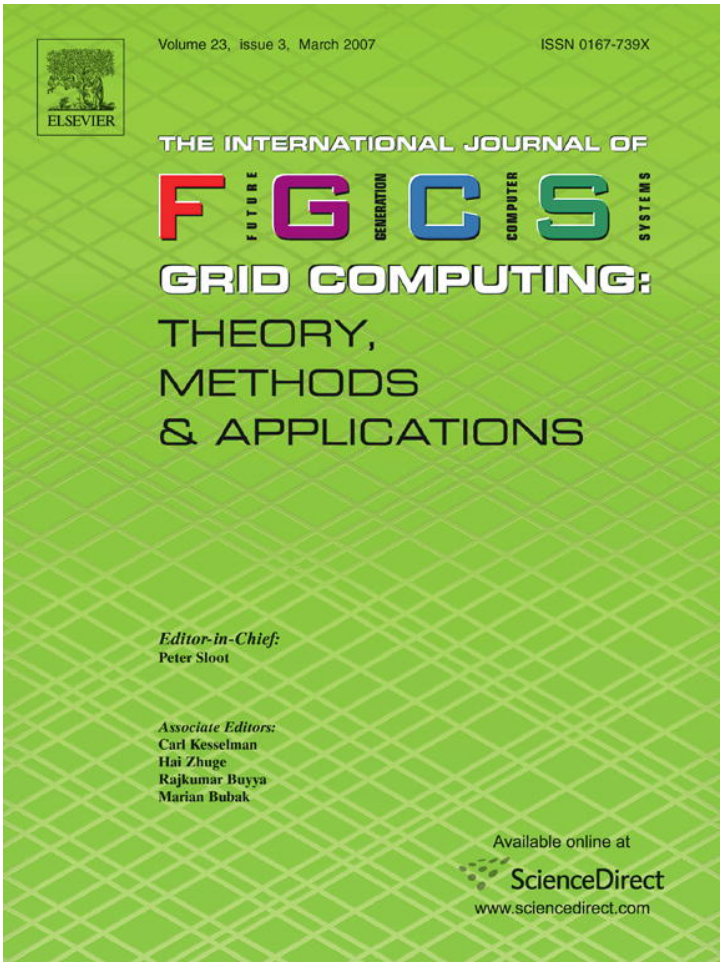


Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 23 (2007) 497–509
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

Bio-STEER: A Semantic Web workflow tool for Grid computing in the life sciences

Sung Lee[a,*], Taowei David Wang[b], Nada Hashmi[c], Michael P. Cummings[d]

[a] *Fujitsu Laboratories of America, Inc., 8400 Baltimore Ave., Suite 302, College Park, MD 20740-2496, USA*
[b] *Maryland Information and Network Dynamics Laboratory, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*
[c] *Management Information Systems, College of Business Administration, Jeddah, Saudi Arabia*
[d] *Center for Bioinformatics and Computational Biology, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*

Received 15 December 2005; received in revised form 26 April 2006; accepted 2 July 2006
Available online 22 August 2006

Abstract

Life science research is becoming evermore computationally intensive. Hence, from a computational resource perspective, Grid computing provides a logical approach to meeting many of the computational needs of life science research. However, there are several barriers to the widespread use of Grid computing in life sciences. In this paper, we attempt to address one particular barrier: the difficulty of using Grid computing by life scientists. Life science research often involves connecting multiple applications together to form a workflow. This process of constructing a workflow is complex. When combined with the difficulty of using Grid services, composing a meaningful workflow using Grid services can present a challenge to life scientists. Our proposed solution is a Semantic Web-enabled computing environment, called Bio-STEER. In Bio-STEER, bioinformatics Grid services are mapped to Semantic Web services, described in OWL-S. We also defined an ontology in OWL to model bioinformatics applications. A graphical user interface helps to construct a scientific workflow by showing a list of services that are semantically sound; that is, the output of one service is semantically compatible with the input of the connecting service. Bio-STEER can help users take full advantage of Grid services through a user-friendly graphical user interface (GUI), which allows them to easily construct the workflows they need.

Keywords: Client/server; Distributed systems; Integrated environments; Semantics; Web-base services; Workflow management; User interface

1. Introduction

Life science research is becoming evermore computationally intensive, involving increasingly larger data sets, particularly genomic data (e.g., complete genome sequences, large-scale microarray results), and increasingly demanding analysis procedures (e.g., stochastic simulation, Bayesian analysis, Markov-chain Monte Carlo sampling). Hence from a computational resource perspective, Grid computing provides a logical approach to meeting many of the computational needs of life science research, much as it does for academic computing needs in general [16]. However, there are several barriers to widespread use of Grid computing in the life sciences, including lack of Grid-enabled applications, difficulty in creating Grid-enabled applications, insufficient Grid computing resources available for life science research, and difficulty of using Grid computing. The first several of these barriers to the use of Grid computing in the life sciences are being addressed [3]. The objective of this paper is to describe a means to address the last barrier mentioned above: difficulty of using Grid computing by life scientists. In general, life scientists might be appropriately characterized as computer literate, but not exceptionally computer-sophisticated. Life scientists should not have to be exceptionally computer-sophisticated to make use of Grid computing, and the tools we present here help make Grid computing more accessible, as well as make bioinformatics workflows more intuitive and in keeping with how workflows are conceptualized [30].

This paper describes a specialized Grid client environment, which utilizes emerging Web and Semantic Web technologies built on the Task Computing Environment (TCE) [38,39].

* Corresponding author. Tel.: +1 301 486 0398; fax: +1 301 441 9676.
E-mail addresses: Sung.Lee@us.fujitsu.com (S. Lee), tw7@cs.umd.edu (T.D. Wang), nada@cba.edu.sa (N. Hashmi), mike@umiacs.umd.edu (M.P. Cummings).

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.07.011

We first provide context and background information, and then describe Bio-STEER [33], an application of the Task Computing Environment for research in the life sciences.

2. The Grid computing environment

Ad hoc Grid systems have been successfully used in life science research [41], but the state of general Grid middleware has improved considerably to where use of standard middleware tools is a more efficient way to develop Grid systems for life science research. We have been working to integrate and deploy computing resources, Grid middleware, and specialized scientific application software presented as Semantic Web services in a Grid system for scientific analysis. Our system is based on a novel Grid infrastructure that combines different Grid computing models, and encompasses resources from high-end clusters and multiprocessors to individual desktop computers.

We have also developed the Grid Service Base Library (GSBL) [3], a Java application programming interface (API) library that aims to reduce the complexity of writing Grid services under the Globus Toolkit [28]. GSBL provides base classes from which clients and services can extend; additionally, it provides features for executing and managing remote jobs and file transfers. As a complement to the GSBL library, the Grid Service Generator speeds the development of Grid services by automatically creating the template files required by a GSBL service [3].

3. Bioinformatics applications

For illustrative purposes, we focus our presentation on a very common bioinformatics workflow that has as its desired end state a phylogenetic tree, which represents the evolutionary history of organisms or genes. The inferred phylogenetic relationships are often used for addressing specific hypotheses, making inferences, providing an evolutionary context, or as a starting point for subsequent study. Therefore phylogenetic analysis constitutes a very important basic bioinformatics workflow in the life sciences. The importance of this workflow is demonstrated by the very large number of papers based on phylogenetic analysis (e.g., [8,12,14,15,17–20,25–27,36,37,42,46]), and entire journals devoted to phylogenetics.

A typical study using phylogenetic analysis includes some fundamental steps in the workflow; excluding file parsing and data formatting. These steps are as follows:

(1) Identify homologous sequences: using a query composed of a DNA or protein sequence of interest to search a database such as GenBank [5].
(2) Retrieve homologous sequences: sequences are retrieved from the database, which may or may not be combined with newly generated sequences.
(3) Align sequences: homologous sequences are aligned position by position to form a multiple sequence alignment.
(4) Infer phylogenetic relationships: with the multiple sequence alignment as input, a phylogenetic analysis is done using any of several methods.

We developed Semantic Web-based Grid services for bioinformatics applications corresponding to the primary phylogenetic analysis workflow steps described above. Here we provide a very brief description of the relevant underlying applications.

(1) BLAST: Basic Local Alignment Search Tool (BLAST) is a package of applications used to compare one or more nucleotide or amino acid query sequences to a database of sequences, and provides measures of similarity and associated statistics [1].
(2) Clustal W: a program for multiple DNA or protein sequence alignment based on a progressive alignment algorithm [60].
(3) EFetch: a Web service for retrieving records from databases at the National Center for Biotechnology Information, part of the National Library of Medicine (USA) [23].
(4) Muscle: another program for multiple DNA or protein sequence alignment [22].
(5) Modeltest: a program for evaluating different hypotheses about the process of DNA substitution [47].
(6) MrBayes: a program for phylogenetic analysis of nucleotide or amino acid sequence data using Bayesian methods [49].
(7) PAUP*: Phylogenetic Analysis Using Parsimony (*and Other Methods) is a program for phylogenetic analysis using parsimony, as well as maximum likelihood, and distance methods [57].
(8) Phyml: another program for phylogenetic analysis of sequence data using maximum likelihood methods [29].

These applications are among the most frequently used in phylogenetic analysis. We specifically included multiple applications for some workflow steps (some using different algorithms), because some users prefer one application over others, and providing options gives flexibility in workflow composition. All services developed from these applications run on Globus resources.

4. Task computing

Researchers at Fujitsu Laboratories of America, jointly with the MINDSWAP research group at University of Maryland, applied the Semantic Web technologies to pervasive computing and created a new environment, called Task Computing, which makes it easier and more efficient for users to accomplish their goals [38,39]. Task Computing is defined as computation to fill the gap between tasks (what a user wants to be done) and services (functionalities that are available to the user). The goals of Task Computing are to present tasks that are possible in the user's current context, to assist the user in creating more complex tasks by using simpler tasks as building blocks, and to guide the user through the execution of the complex tasks. Ultimately, Task Computing brings the user considerably closer to their goals by presenting an abstract view of the resources (devices and services) that can be used in an ad hoc manner by the user, in order to execute tasks of arbitrary complexity. Task Computing achieves these goals through exposing the functionality in such environments (device functionality or

third-party functionality) as Semantic Web services using Semantic Web (Resource Description Framework [RDF] [65], Web Ontology Language [OWL] [66]), Web Service (Simple Object Access Protocol [SOAP] [52], Web Services Definition Language [WSDL] [67]), and pervasive computing (Universal Plug and Play [UPnP] [61]) technologies. The users, in turn, can compose these Semantic Web services to perform complex tasks.

Task Computing successfully demonstrated the following benefits in pervasive environments such as e-office and e-home [39].

- Resources/services abstraction
  - Users need not be concerned with low level details such as platform.
  - Remote services (services available on the Internet) and pervasive services (services available in the user's environments) can be treated just as local services (services available on the user's machine).
- Semantic services
  - Guide the user to compose a complex service by presenting likely compositions of available services based on semantic input and output descriptions.
  - Utilize existing services and enable construction of more complex services using the existing services as building blocks.
- Fast and easy new service composition and integration capability.
- User friendly environment, in which non-computing experts can take full advantage of available resources and services just as computing experts would.

In order to support Task Computing, an application called the Task Computing Environment has been created. The Task Computing Environment includes a client interface, a service discovery mechanism, a dynamic service management environment, and various sample Semantic Web services. STEER, the Task Computing client, first dynamically discovers the services that are advertised via Universal Plug and Play (UPnP) and other discovery mechanisms and categorizes achievable two-step service compositions based on their semantic inputs and outputs on a *Compose* page. We use the word STEER to mean the ability to pursue a course of action by easily connecting disparate applications. More complex compositions can be realized through a *Construct* button. The Web service semantic descriptions enable STEER to automatically combine services into likely compositions for the user to consider, select, extend, and perform.

Ordinary Web service documents, described by the Web Service Definition Language (WSDL), lack the necessary extensive, structured documentation of services. In the Task Computing Environment, an additional semantic description document is written in Web Ontology Language-Service (OWL-S) [44]. OWL-S is a set of language features arranged in ontologies to establish a framework within which the Web services may be described in the Semantic Web context. The main contribution of OWL-S is the ability to express entities using concepts defined in Semantic Web ontologies, which provide expressive constructs that are suitable for the automatic discovery and composition of services [53].

In order to provide the extra OWL-S description in the Task Computing Environment, the developers of services are equipped with a new tool called OntoLink [40], which generates OWL-S descriptions from WSDL annotations. OntoLink grounds the semantic service described in an OWL-S description to its corresponding Web service implementation. Additionally, OntoLink can be used to map between ontology elements, thus allowing composition of two services that use independent but related ontologies.

5. Bio-STEER

Recognizing the benefits of semantic-enabled computing environments for life sciences research, we have applied the experience and lessons learned from implementing Task Computing to bioinformatics through Bio-STEER [33]. Bioinformatics applications are presented as semantic services, and Bio-STEER provides an environment in which the user can compose custom workflows. At each step, Bio-STEER utilizes the application semantics (particularly the input and output types) in order to assist the user in narrowing down the list of potential services that can go into the workflow. Bioinformatics applications differ in several regards from applications found in e-office or e-home environments. For example, bioinformatics services are typically long running processes, and there are opportunities to run multiple services in parallel. Often in e-home or e-office environments, it is more intuitive to run services in series as each service completes within a short period of time. Bio-STEER supports both long running processes and parallel processing. Similarly, Med-STEER [32] is an application of Task Computing in medical informatics that enables efficient data and biomedical application service composition and execution for the better care of patients. In such environments, the ability to dynamically compose services based on what is available within the current context is important, as care givers often move from one environment (e.g., diagnostic facility) to another (e.g., patient care facility) while requiring seamless exchange of information.

5.1. Architecture

Song et al. defined a four-layered architecture to describe a semantic rich environment of Task Computing [55]. Because Bio-STEER is an application of the Task Computing Environment in the life sciences domain, the basic architecture remains the same. Devices, applications, e-services, and data reside in the realization layer, as do Bioinformatics applications and Grid services. This basic architectural design can be modified to describe phylogenetic analysis applications (Fig. 1).

Note that in the Task Computing Environment, devices can be handled just in the way applications or e-services are handled. Although for simplicity our current example does not include integration of biological laboratory devices, integration of such devices can be easily accomplished. For example, we

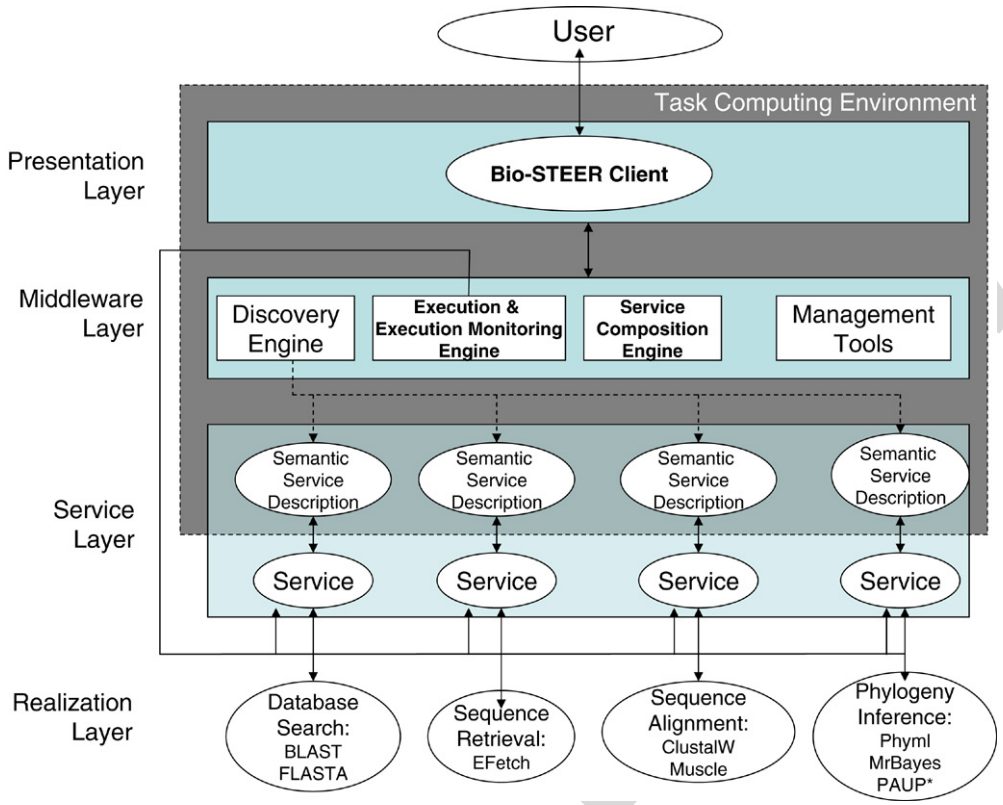


Fig. 1. A four-layered architecture for the Task Computing Environment with Bio-STEER and Grid/Web services for phylogenetic analysis functions.

are in the process of introducing a microarray data analysis capability into Bio-STEER, in which a microarray reader device can be directly integrated into workflows involving microarray data analysis.

In the service layer, we map the realization layer to Web services. Among the changes from Globus Toolkit 3 to Globus Toolkit 4 was the presentation of Grid services as conventional Web services. However, our work on Semantic Web/Grid services and workflow composition using Bio-STEER predates the release of Globus Toolkit 4. Therefore we had to map Grid services into Web services. Our Grid system is in the process of migration to Globus Toolkit 4, at which point these Grid services will no longer be mapped to intermediate Web services. Also in the service layer are semantic service descriptions for each of the Web services.

Although much of the canonical Task Computing Environment middleware layer and presentation layer remains the same, the Task Computing Environment has been modified to support bioinformatics applications. In computational biology research, many bioinformatics applications may run for long periods of time, and Bio-STEER accommodates this by allowing for unlimited application run times. Another feature of Bio-STEER is the addition of parallel execution that occurs at a fork in a workflow where two Grid service instances can execute at the same time (parallel computation within a single service is handled at the application and/or Grid service level). Additionally, Bio-STEER has a *save* workflow option to include the details of all the services involved in the workflow. Most of the middleware layer components (Execution & Execution Monitoring Engine, Service Composition Engine, and Management Tools) are implemented in Java on Microsoft Windows. The Discovery Engine relies on standards such as UPnP.

We currently provide a four-pane, GUI-based client environment, Bio-STEER client, implemented in C#. However, for lightweight client environments such as hand held devices, a Web-based client can be used. In this case, the client's browser is the interface. Fig. 2 shows the client displaying four panes. The upper left corner is the Service Explorer pane, which shows the list of available services. In the lower left, the Properties pane is shown with details of a selected service including where its service description file can be found. In the lower right corner is the output from the execution of the workflow. The construction pane is in the upper right corner, which is mainly used to construct a new workflow or modify an existing workflow.

We created the BioGridService ontology that the reasoner, Pellet [58], uses in assisting users to construct a workflow. The BioGridService ontology [6] is simple in structure, with several top concepts as shown in Fig. 3. CommandOption concept represents the command argument fed into a particular Grid service, and has subclasses that represent the command arguments for each different Grid service. The GridClientLocalFile concept represents the remote files that reside on the Grid client, and has subclasses that represent different types of files.

There are several properties in BioGridService ontology. The property *hasJobName* describes the name of a particular service (e.g., the BLAST service would have 'blast' as the value of its *hasJobName*). The property *hasParameter* denotes the input parameters for that particular Grid service. These

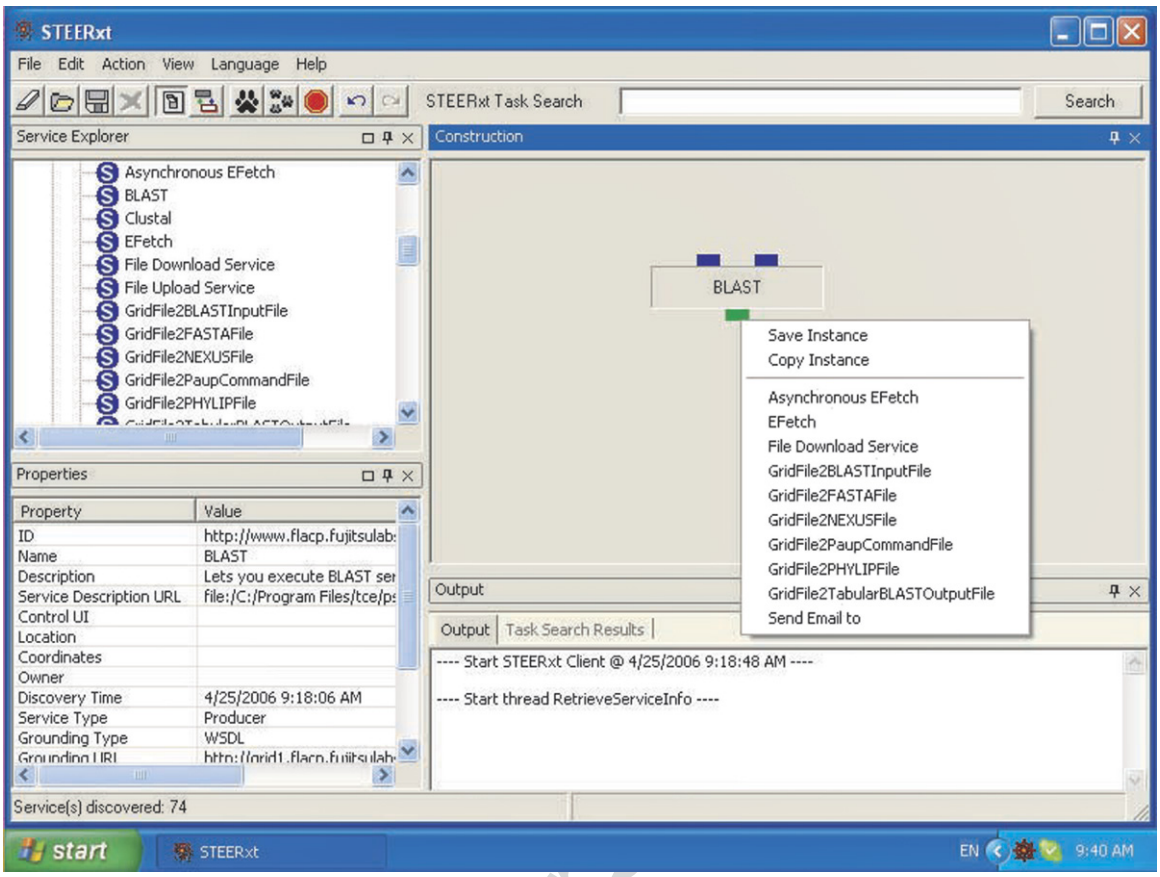


Fig. 2. Workflow construction. After dragging-and-dropping the BLAST service into the construction pane, right clicking on the BLAST service shows compatible services that can be combined with the output of the BLAST service. Inputs depicted as blue rectangles at top of service, and output depicted as green rectangle at bottom of service. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

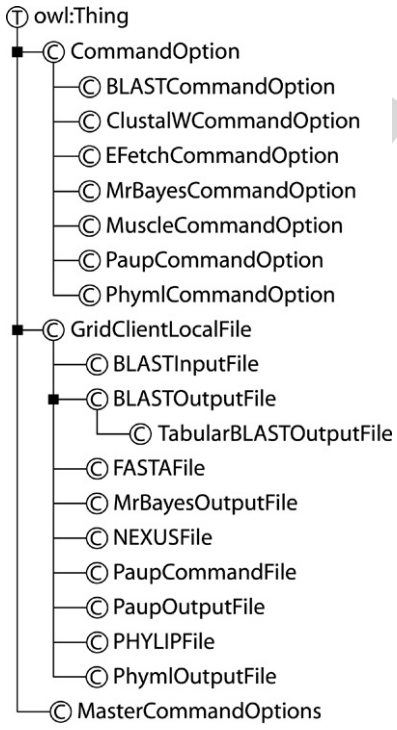


Fig. 3. The class tree of BioGridService ontology.

two properties are required in any CommandOption concepts. The *hasFileName* property is used in any GridClientLocalFile concept. The BioGridService ontology can be further expanded to include additional reasoning necessary to assist the end users.

5.2. Web services and Semantic Web services

The Grid computing environment provides the infrastructure for job submission, monitoring, retrieval, and a framework that includes a Grid client to invoke these capabilities. For each application, our Grid client has a script to submit a job of that particular type to the Grid, and another script to retrieve the computational result once the job is finished. Devices, applications, and our Grid client are mapped to Web services (Fig. 1). We create a Semantic Web service description for each Web service so the Task Computing Environment can facilitate users in task composition using the semantic constraints. In this section, we describe how the end-user interacts with the Grid, and how we create these mappings.

There are three layers of communication in the execution of a workflow: user desktop, Web service host/Grid Client, and the Grid System. The execution model is as follows. An end-user uses her desktop to construct workflows using the Task Computing Environment (TCE), which executes each step of the workflow by invoking the associated Web service. Since our Grid client hosts these Web services, as each service is

invoked, it runs the appropriate script to submit the job to the Grid. When a Grid job is finished, the Grid client retrieves the result, returns the Web service call to the TCE, and the TCE can execute the next step in the workflow. The TCE acts upon the Semantic Web service layer, and the Web services to which Semantic Web services are grounded operate on the Extensible Markup Language (XML) layer. Finally the submission to the Grid involves raw data files and a list of parameters. A complete execution of a workflow requires the conversion of data representation through the three different layers.

Each Web service consists of two parts: the code that implements the application logic for the computational task, and the interface that allows agents to invoke the service. The application logic, written in Java, invokes the appropriate script to submit to or retrieve from the Grid jobs. We have one generic Web service that handles all job submissions and retrievals with parameters specifying the Grid service invoked and the input data. Aside from this service, there are other auxiliary services, which we describe in detail at the end of this section. For each Web service, we create an interface in WSDL, which the Grid client hosts publicly so any machine with access to the Internet can see and invoke these Web services. For the operations defined in the WSDL file, we create a corresponding Semantic service description file that details the operation semantically, and can be used by the TCE for workflow construction. The semantic service description for BLAST is available [9].

A Semantic service description file in OWL-S contains three parts: profile, process, and grounding. Profile describes what the service does, process tells how the service works, and grounding specifies how an agent can invoke the service. The Semantic service description files are backed by an OWL ontology [6] that models the inputs and outputs for the Semantic Web services. Each call to the Semantic Web service is accompanied by an input of the appropriate Semantic type (a concept in the ontology). In the grounding portion of the Semantic service description file, each semantic input is converted to an appropriate XML type as input for the grounded Web service. These Extensible Stylesheet Language Transformations (XSLT) are necessary because Web services accepts XML data types, not OWL concepts. When the Web service returns with its XML-typed output, another XSLT defined in the Semantic service description file is used to convert it back to a Semantic instance so that can be utilized by the TCE. In this semantic layer, we create a Semantic service description file for each corresponding application. All these Semantic service description files are grounded to the same generic Web service that submits or retrieves jobs, but they contain parameters that specify the Grid service to run, and inputs and options to be used.

The TCE uses information in the Semantic service description files to evaluate if the output of a service matches the input of another, and therefore if the two services can be joined in series. Because this matching can be automatically determined, the TCE can quickly filter out the irrelevant services, and present to the user a much smaller list of compatible services.

Two inputs to the Grid applications are required: the data input file, and a list of parameters. The ontology for Bio-STEER models these two inputs for each service explicitly so we can represent them in the Semantic layer in the TCE. There is one filetype class for each possible data format (e.g. NEXUS [35]) that the application accepts. There is one parameter type for each application, since each application has different options. For each execution of a Semantic Web service that results in a bioinformatic Grid service submission the input data file is wrapped in the filetype semantic instance, and the parameter instance contains options for the Grid service.

There are several auxiliary Web services that make our system work, of which the most important are FileTransfer, FileConversion, and EFetch. The FileTransfer service handles file upload and download between the user TCE to the Grid client. The FileConversion service converts a semantically untyped file to a semantically typed one. This allows files that have been uploaded or downloaded to have a proper type so the TCE can work with it. The EFetch service, though it does not initiate a job submission to the Grid, is considered part of the analysis workflow. Given the outputs of the BLAST service in tabular form, the EFetch service gathers the accession numbers that identify nucleotide or protein sequences, and queries the NCBI database and retrieves the sequence(s) in FASTA format.

In this section we have described the layers that make up the underlying steps from the user's desktop to the submission of a job to the Grid. In the next section, we give the details on how a workflow is constructed using the TCE.

5.3. Workflow construction

A user can drag and drop an available service from the Service Explorer pane to the Construction pane. Based on the information found in the service description, Bio-STEER shows if the service requires input and produces output. For instance, the BLAST service shows two inputs and one output (Fig. 2). The user can then right click on any one of the input or output boxes, and Bio-STEER assists users by showing compatible services. In the BLAST case, the semantic description of BLAST indicates that the output of BLAST can be combined with any of the services shown in Fig. 2.

Similarly, Bio-STEER can help users construct the desired workflow by showing services whose output can be passed as an input to the BLAST service (Fig. 4). A workflow is completed when there is no service with open input or open output (Fig. 5). The workflow then can be saved for later use or for further modification, and Bio-STEER offers an option to save the details of constituent services. This *Save* option makes it easy to share workflows with others, especially ones who do not have access to all the services.

5.4. Workflow execution

For many computational biology problems, the bioinformatics analyses may require a relatively long time to complete due to the size of the data set and/or solution space, or the computational complexity of the analysis algorithms. These are among

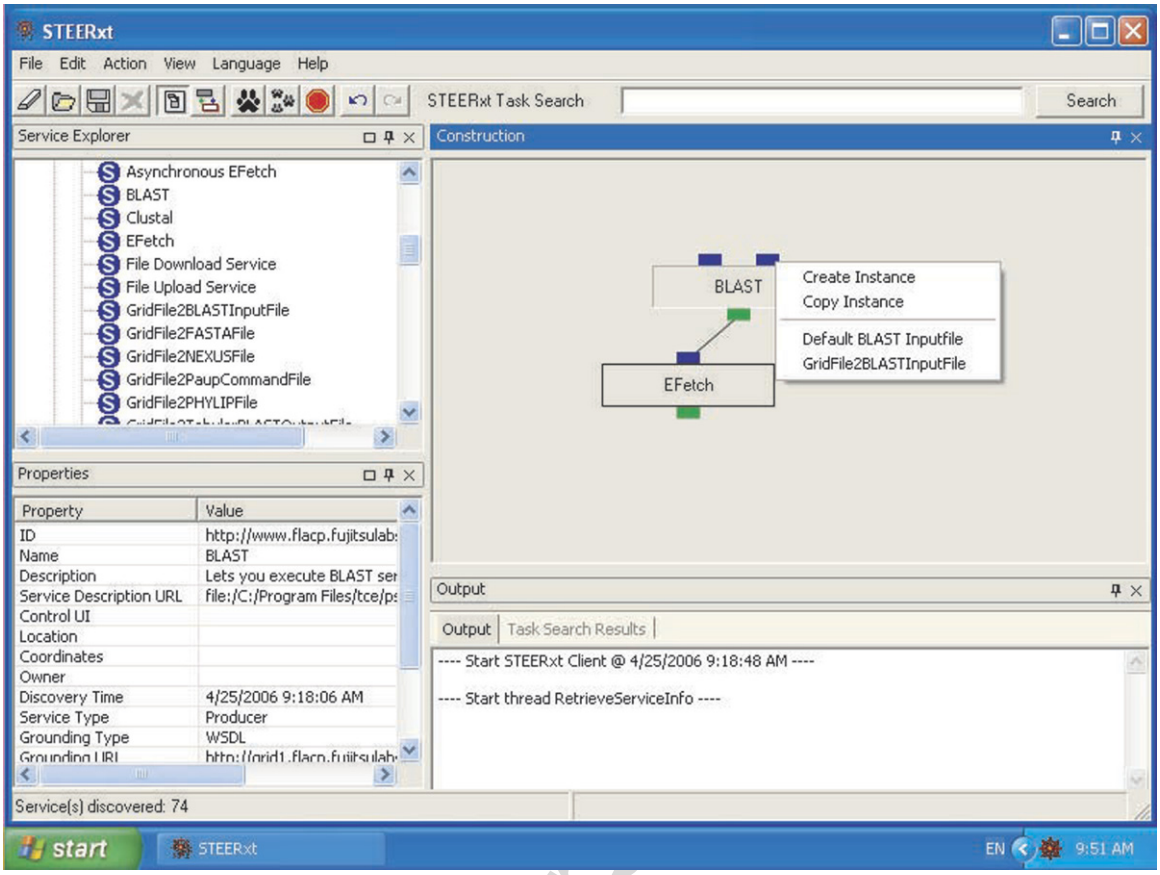


Fig. 4. Workflow construction. Compatible services whose output can be the input to one of the BLAST inputs.

the very reasons motivating the use of Grid computing in the life sciences. Bio-STEER facilitates the efficiency of bioinformatics workflows by accommodating parallel execution of multiple Semantic Web/Grid services described as forks in a workflow composition.

Bio-STEER provides the user with process status information during execution through the use of color: pink denotes services that have completed whereas blue denotes services that are currently being executed, and gray denotes services still to be executed (Fig. 6). Thus the progress through the workflow can be monitored easily and quickly. In the example presented here Bio-STEER executes both Muscle service and Clustal service in parallel once the EFetch service completes (Fig. 6).

The execution of a workflow can be completely automated and the users can be notified via emails with the results attached as shown in Fig. 6. However, it may be desirable to examine intermediate results before proceeding to the next step. For example often it is common to examine the multiple sequence alignment, and perhaps change the alignment, before proceeding to the phylogenetic inference step. Bio-STEER offers pause or breakpoint functionality in these cases through any one of the three services: *Modify Instance*, *Copy Instance*, or *Save Instance* services. The workflow execution would be halted until the appropriate user input is entered for those three services. Fig. 7 shows a modified workflow (detailed in Fig. 6) where the user has a chance to examine and modify the result of the EFetch service before proceeding to the Clustal service.

5.5. Availability

Bio-STEER software and semantic service descriptions are available to researchers from academic and non-profit research institutes on a non-commercial basis. Additional information is available on the Task Computing site [59].

6. Related work

Jia Yu and Rajkumar Buyya [68] provide a detailed study and define a taxonomy for workflow systems for Grid computing. We briefly describe salient properties of this taxonomy, extend this taxonomy by considering additional features of workflow systems, and apply the extended taxonomy to workflow systems designed for Grid computing in the life sciences.

A workflow tool can be described by the operations permitted in a workflow (structure), whether it's bound to its resources (model), and how the users assemble the workflow (composition). A workflow that allows directed acyclic graphs (DAG) will inherently permit sequential tasks, parallel tasks, and tasks that will only execute if their conditions are met at run-time. A non-DAG implementation permits an additional powerful operation, *iteration*, the repetition of tasks or set of tasks. There are two types of models that workflows systems employ: *abstract*, in which the user defines the workflow without specific reference to the Grid resources; and *concrete*, in which the workflow binds the tasks to specific resources.

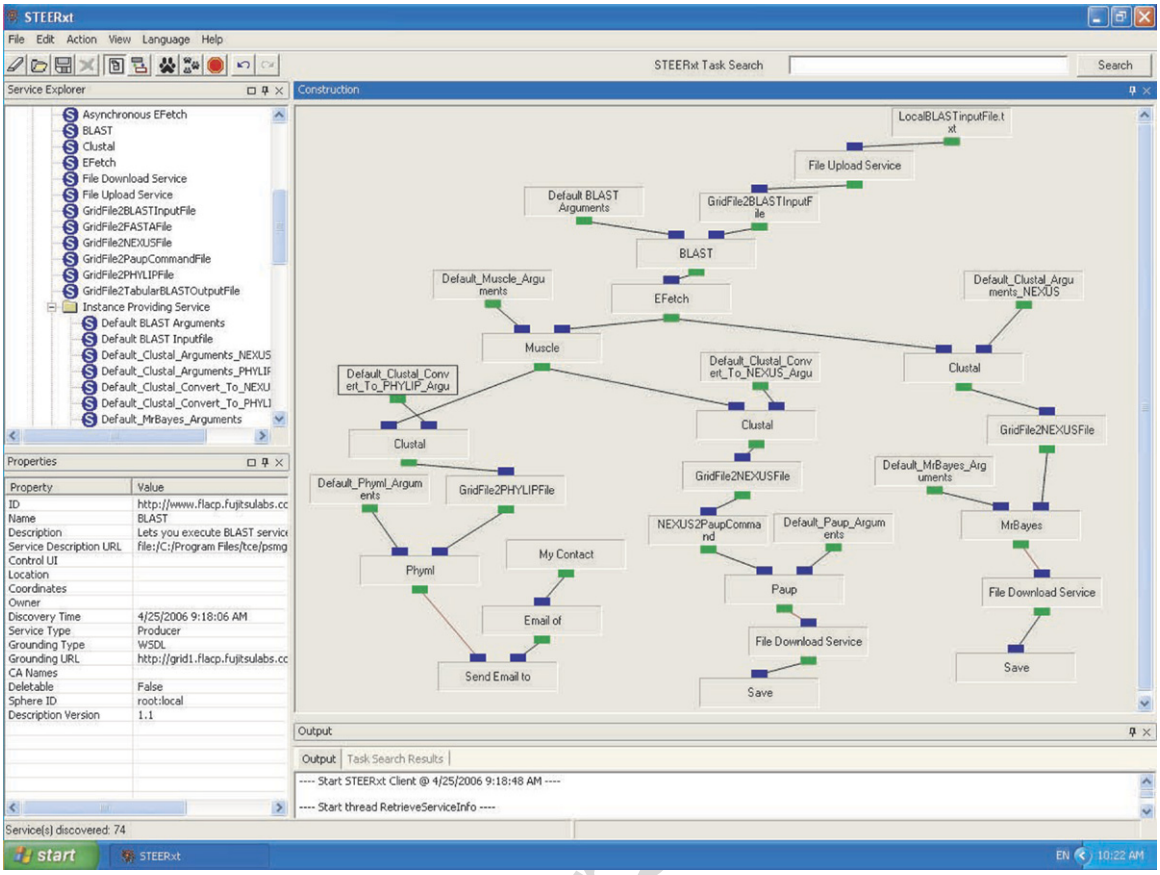


Fig. 5. A fully complete workflow using local files, Web services and Grid services. It is also an example of forked execution (EFetch to Muscle and ClustalW).

The composition of the workflow can be *user-directed* where the user manually specifies the workflow (either using a graphical representation or a language) or *automatic* where the workflow is automatically generated based on the high level requirements. Although data format compatibility between services may exist, this is not sufficient to insure services should be linked in a workflow. Therefore, we consider *intelligence*, a system to help compose the workflows, which we define here as guided assistance based on the context, a pseudo-automatic composition method. Finally, we also include, for comparison in the workflow design, whether the system is *extensible*, having the capability to add or remove additional tasks, and the control of the workflow (*flow control*) of the system.

Workflow systems for bioinformatics depend heavily on the input of data either initially or at various points during execution. We define this as *data integration*. Additionally, another aspect which is important for a Grid-based architecture is the ability to integrate devices at will. The findings are summarized in Table 1, and short descriptions of each workflow systems are provided below.

6.1. BioWBI

BioWBI [7] is a portal system developed by IBM alphaworks. Users log into the portal and can access workflows created by other users or can create custom workflows. Workflows are created by first selecting the tools that are available in a particular order. The user then specifies operations that should be performed on the tool; it uses a non-DAG model thereby permitting iterations, conditionals, parallel and sequential operations in workflows. User can upload their input data or directly enter the data. The workflow is executed using a custom made workflow execution engine. The results are collected and can be viewed immediately or saved for comparison at a later date. The portal system allows users to create systems specific to a particular type of research (e.g., pharmaceutical companies targeting particular drugs).

BioWBI relies heavily on the users sustaining a reliable Internet connection, and the use of the BioWBI system server for computational power. While IBM does support a strong infrastructure in this regard, users become restricted to what BioWBI offers and supports for computational power. Furthermore, while other systems rely on application processing at a local level, the users of BioWBI only have access to the presentation layer at a local level; application processing, data storage and access are controlled, and processed, by BioWBI.

6.2. Pegasys

Pegasys [51], developed by the University of British Columbia Bioinformatics Centre, facilitates the construction and execution of workflows consisting of biological sequence

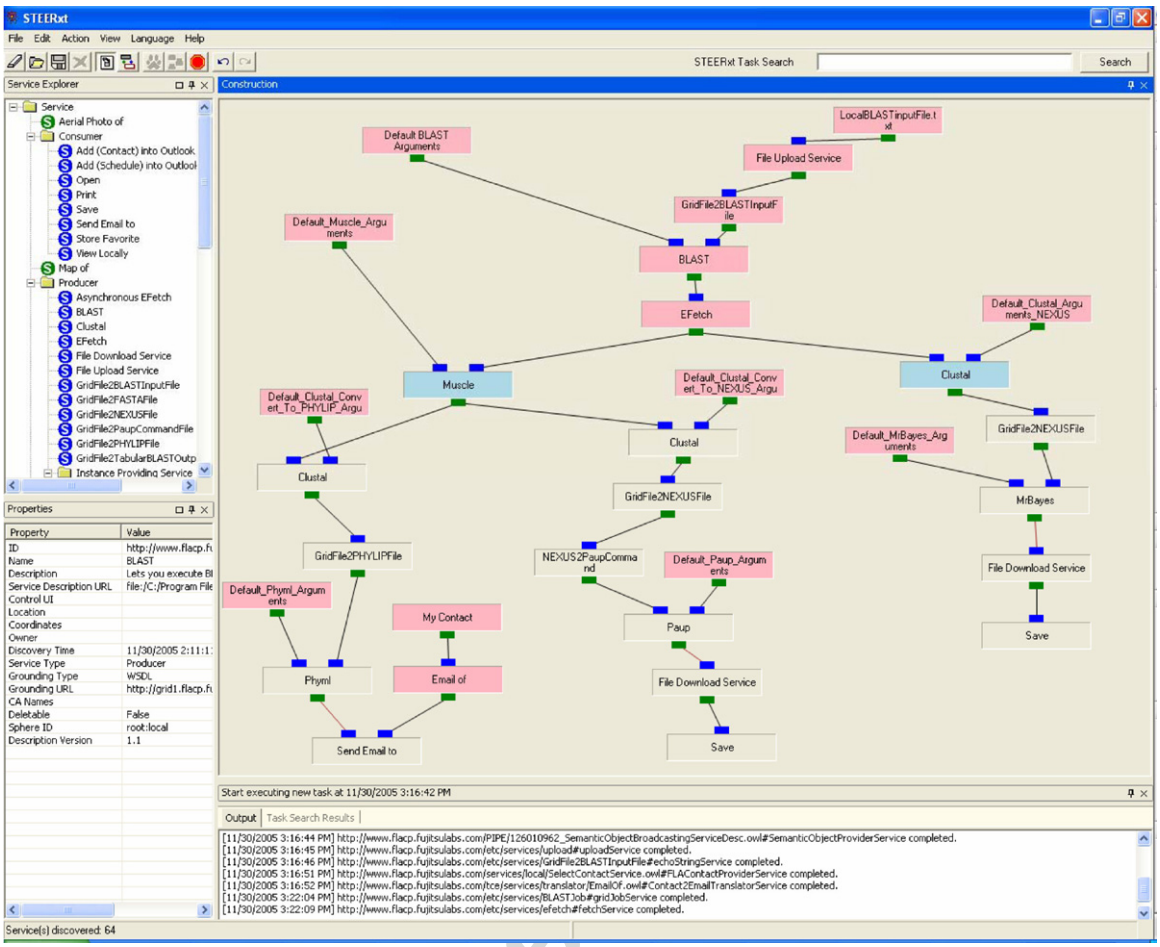


Fig. 6. Workflow execution can be monitored using the Bio-STEER user interface. Process status information is depicted through the use of color: pink denotes services that have completed whereas blue denotes services that are currently being executed, and gray denotes services still to be executed. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Comparison of workflow composition tools for bioinformatics

Workflow design features	BioWBI	Pegasys	Taverna	Wildfire	Bio-STEER
Structure	Non-DAG[a]	DAG	DAG	Non-DAG	DAG
Model	Concrete	Abstract	Abstract Concrete	Concrete	Concrete
Composition	User-directed	User-directed Automatic	User-directed	User-directed	User-directed
Composition Interface	Language Graph	Language	Language Graph	Graph	Graph
Intelligence	Yes	No	Yes	Yes	Yes
Flow Control	Elaborate	Elaborate	Elaborate	Elaborate	Medium
Extensible	Yes	Yes	Yes	Yes	Yes
Pause (breakpoint)	No	No	No	No	Yes
Data integration	Yes	No	No	No	Yes
Device integration	No	No	No	No	Yes

[a] DAG (directed acyclic graph).

analysis tools. The Pegasys client, implemented in C++ using QT graphical libraries [50] and tested on Linux, Solaris, Mac and Windows platforms, allows users to create workflows by dropping Pegasys programs (nodes) and connecting two nodes through directional edges. Users must provide input/output types of each edge during construction of workflows. The client validates syntax through a ProgramList.xml file which lists all the programs and their associated parameters.

Pegasys by default provides RepeatMasker [4], BLAST (blastn, blastp, blastx, tblastn, tblastx) [1,2], WU BLAST [62],

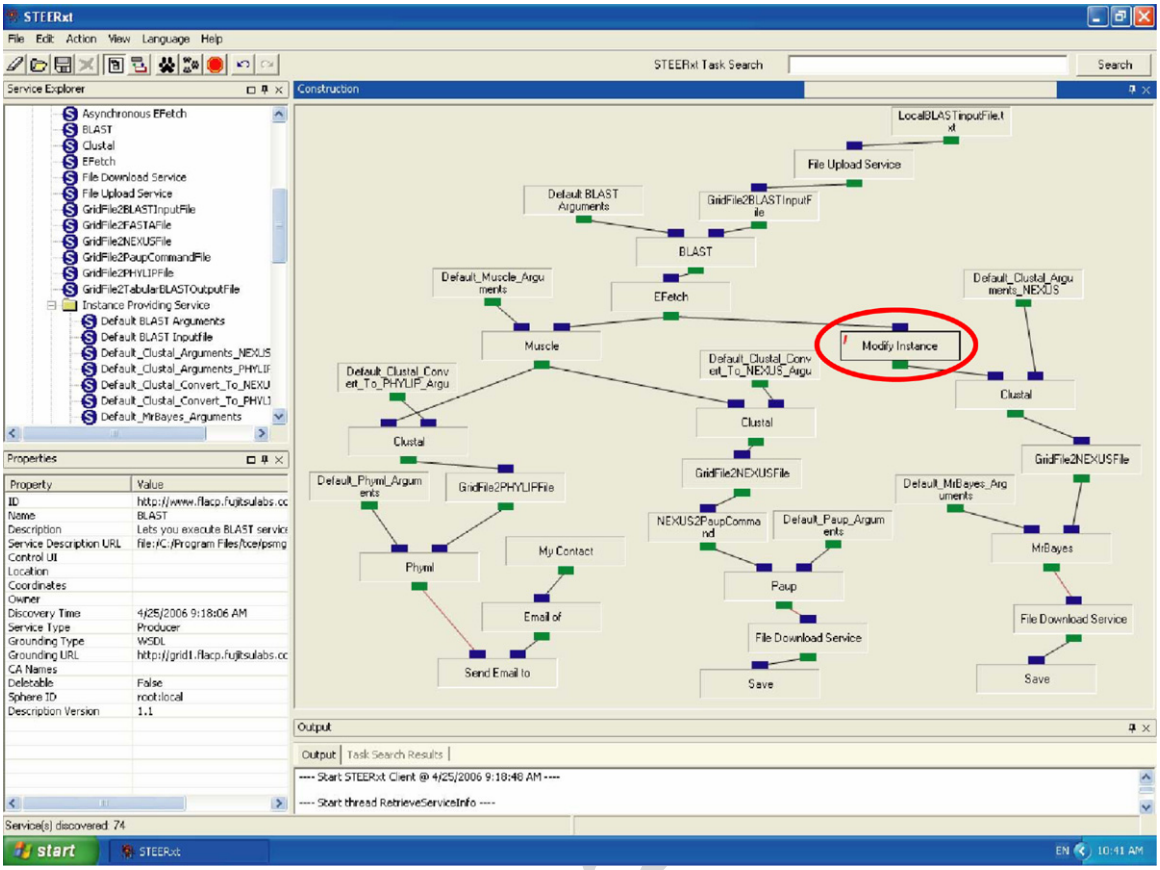


Fig. 7. Workflow with breakpoint (using the Modify Instance service) for user interaction inserted between EFetch and ClustalW services.

the EMBOSS [48] implementation of Smith–Waterman [54], Genscan [11], HMMgene [31], Mlagan [10], Sim4 [24], tRNAscan-SE [34], and GeneSplicer [45]. However, additional programs can be easily added.

Once the workflow is constructed, it is dispatched to the server for execution. The Pegasys server, implemented in Java for UNIX platforms and tested on a distributed memory cluster, analyzes the workflow and schedules each program on a distributed computing cluster. The result of each Pegasys program is stored in the back-end server. The back-end database system is provided to integrate data from heterogeneous sources of analysis tools, with an advantage that results can be analyzed collectively.

6.3. Taverna

Taverna [43] is the result of collaboration by the European Bioinformatics Institute, the Human Genome Mapping Project and the myGrid Project [56] to serve as the workflow composition tool. Users build workflows by linking the data input and output of different types of processes (computational methods and tools) either manually or with guided assistance from the Taverna system. The guided assistance is based on the type-matches of the data format where lists are generated after each step of the possible next steps that the user can select from. The processes and methods that are available can be locally installed on their machines, external Web services or Web services supported by myGrid. After the composition, the user must further specify the format of the files from one process to the other. No form of automatic translation or format recognition is provided.

Taverna implements a DAG workflow design; thereby restricting the operations that processes can perform to be either in parallel, sequential or conditional. Iterations are not supported. Taverna does however, provide sophisticated error-checking capabilities during runtime and execution time. It provides alternatives for Web services that may be unavailable or broken at the time of execution. It further provides error-reporting; where the type of error and place of error is reported back to the user. However, the workflow composition process workflows is complex, and requires up to two or three days for a workflow to be composed.

6.4. Wildfire

Wildfire [63], developed by the Bioinformatics Institute (Singapore), is an integrated environment for constructing and executing bioinformatics workflows. Implemented in Java for Windows and Linux platforms, Wildfire can run workflows locally, remotely on clusters, and on Grid-enabled resources. Workflows are constructed on a graphical canvas by dropping workflow components, which include an atomic component, a sub-workflow, or a loop. The atomic components are pre-configured with EMBOSS [48] applications, and

have corresponding Ajax Command Definition (ACD) [21] descriptions of parameters and options. Users can expand the atomic components to include custom applications with corresponding ACD descriptions. Flow-control includes loops (for-each, for, and while) and conditionals (if-else). The workflow is exported in a script language called Grid Execution Language (GEL) [13,64] and executed on any resources with a GEL interpreter.

Unlike other workflow management tools, Wildfire works directly with executables. One direct benefit is that Wildfire can run as a standalone system. However, the users cannot take advantage of many bioinformatics applications and tools available on the Web. They are limited to either using the default EMBOSS suite or custom applications they must add into the system. When using the default EMBOSS suite, the composition of their workflow will include only compatible tasks in terms of data formats. Due to this, error-checking is performed on the type of workflow; that is, to ensure the user restricts operations each task can perform to the non-DAG model.

7. Future work

Bio-STEER is already a very useful workflow system for bioinformatics, but we continue to increase its functionality. Here we list some of current and planned development objectives.

- Additional workflow control including conditional control support such as if-else-then, and loop control support (e.g., while, for).
- Additional support for logging for debugging purposes, parameter settings, when and where the execution of the service occurred.
- Improved support for error handling with automatic continuation of workflow from point of failure following error resolution.
- Access control mechanisms so that the use of devices or applications can be prioritized.

8. Conclusion

Grid computing offers a computing infrastructure that helps advance life sciences research. We applied Task Computing to this field in order to reduce the barriers to effectively using Grid computing by researchers in the life sciences.

Workflow tools in bioinformatics are gaining popularity among life scientists. Recognizing the potential benefit of Semantic Web technologies, some of the tools have made use of semantics in helping users. Our approach of using standard technologies such as WSDL and OWL differs from other approaches. Also, the use of semantics to filter and suggest only semantically compatible services will reduce the time it takes to construct a meaningful workflow. In Bio-STEER, devices are also treated as any other applications, and they too can be mapped to services and semantic services. This capability can offer the option of directly including devices as part of the workflows life scientists need. Bio-STEER is currently production ready, although we are actively developing many enhanced capabilities.

Acknowledgments

We thank Adam Bazinet, Matthew Conte, Ryusuke Masuoka, and Zhexuan Song for their help.

References

[1] S. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, Basic local alignment search tool, J. Mol. Biol. 215 (1990) 403–410.

[2] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, D.J. Lipman, Gapped BLAST and PSI-BLAST: A new generation of protein database search programs, Nucleic Acids Res. 25 (1997) 3389–3402.

[3] A.L. Bazinet, D.S. Myers, J. Fuetsch, M.P. Cummings, Grid services base library: A high-level, procedural application programming interface for writing Globus-based Grid services, Future Gener. Comput. Syst. (in press). doi:10.1016/j.future.2006.07.009.

[4] J. Bedell, I. Korf, W. Gish, Masker Aid: A performance enhancement to RepeatMasker, Bioinformatics 16 (2000) 1040–1041.

[5] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, D.L. Wheeler, GenBank, Nucleic Acids Res. 33 (2005) D34–D38.

[6] Bio Grid Service Ontology, http://www.flacp.fujitsulabs.com/tce/ontologies/2005/08/BioGridService.owl.

[7] Bioinformatic Workflow Builder Interface, http://www.alphaworks.ibm.com/tech/biowbi.

[8] D. Blair, A. Campos, M.P. Cummings, J.P. Laclette, Evolutionary biology of platyhelminths comes of age: The role of molecular phylogenetics, Parasitol. Today 12 (1996) 66–71.

[9] BLAST Semantic Web service description, http://www.flacp.fujitsulabs.com/sylee/BLAST.owl.

[10] M. Brudno, C. Do, G. Cooper, M. Kim, E. Davydov, E. Green, A. Sidow, S. Batzoglou, LAGAN and Multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA, Genome Res. 13 (2003) 721–731.

[11] C. Burge, S. Karlin, Prediction of complete gene structures in human genome DNA, J. Mol. Biol. 268 (1997) 78–94.

[12] A. Campos, M.P. Cummings, J.L. Reyes, J.P. Laclette, Phylogenetic relationships of Platyhelminthes based on 18S ribosomal gene sequences, Mol. Phylogenet. Evol. 10 (1998) 1–10.

[13] C. Chua, F. Tang, P. Issac, A. Krishnan, GEL: Grid execution language, J. Parallel. Distr. Com. 65 (2005) 857–869.

[14] M.T. Clegg, M.P. Cummings, M.L. Durbin, The evolution of plant nuclear genes, Proc. Natl. Acad. Sci. USA 94 (1997) 7791–7798.

[15] M.P. Cummings, S.A. Handley, D.S. Myers, D.L. Reed, A. Rokas, K. Winka, Comparing bootstrap and posterior probability values in the four-taxon case, Syst. Biol. 52 (2003) 477–487.

[16] M.P. Cummings, J.C. Huskamp, Grid computing, EDUCAUSE Rev. 40 (2005) 116–117.

[17] M.P. Cummings, L.M. King, E.A. Kellogg, Slipped-strand mispairing in a plastid gene: *rpo*C2 in grasses (Poaceae), Mol. Biol. Evol. 11 (1994) 1–8.

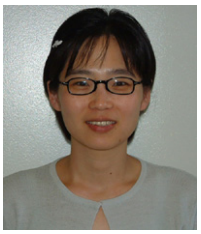
[18] M.P. Cummings, A. Meyer, Magic bullets and golden rules: Data sampling in molecular phylogenetics, Zoology 108 (2005) 329–336.

[19] M.P. Cummings, J.M. Nugent, R.G. Olmstead, J.D. Palmer, Phylogenetic analysis reveals five independent transfers of the chloroplast gene *rbc*L to the mitochondrial genome in angiosperms, Curr. Genet. 43 (2003) 131–138.

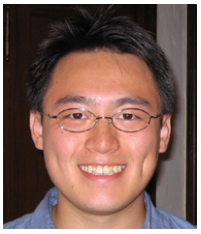
[20] M.P. Cummings, S.P. Otto, J. Wakeley, Genes and other samples of DNA sequence data for phylogenetic inference, Biol. Bull. 196 (1999) 345–350.

[21] T. de Boer, AJAX Command Definition (ACD files). http://www.rfcgr.mrc.ac.uk/Software/EMBOSS/Acd/.

[22] R.C. Edgar, MUSCLE: Multiple sequence alignment with high accuracy and high throughput, Nucleic Acids Res. 32 (2004) 1792–1797.
[23] EFetch, http://eutils.ncbi.nlm.nih.gov/entrez/query/static/efetchseq_help.html.
[24] L. Florea, G. Hartzell, Z. Zhang, G. Rubin, W. Miller, A computer program for aligning a cDNA sequence with a genomic DNA sequence, Genome Res. 8 (1998) 967–974.
[25] H. Förster, M.P. Cummings, M.D. Coffey, Phylogenetic relationships of *Phytophthora* species based on ribosomal ITS I DNA sequence analysis with emphasis on Waterhouse groups V and VI, Mycol. Res. 104 (2000) 1055–1061.
[26] M. García-Verela, M.P. Cummings, G. Pérez-Ponce de León, S.L. Gardner, J.P. Laclette, Phylogenetic analysis based on 18S ribosomal RNA gene sequences supports the existence of class Polyacanthocephala (Acanthocephala), Mol. Phylogenet. Evol. 23 (2002) 288–292.
[27] M. García-Verela, G. Pérez-Ponce de León, P. de la Torre, M.P. Cummings, S.S.S. Sarma, J.P. Laclette, Phylogenetic analysis of Acanthocephala based on 18S ribosomal gene sequences, J. Mol. Evol. 50 (2000) 532–540.
[28] Globus Toolkit, http://www.globus.org/toolkit/.
[29] S. Guindon, O. Gascuel, A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood, Syst. Biol. 52 (2003) 696–704.
[30] N. Hashmi, S. Lee, M.P. Cummings, Abstracting workflows: Unifying bioinformatics task conceptualization and specification through semantic Web services, in: W3C Workshop on Semantic Web for Life Sciences, Cambridge, MA, USA, 2004.
[31] A. Krogh, Two methods for improving performance of an HMM and their application for gene finding, in: Proc. Int. Conf. Intell. Syst. Mol. Biol., vol. 5, 1997, pp. 179–186.
[32] S. Lee, M. Galdzicki, R. Masuoka, Y. Labrou, J. Agre, Med-STEER: Enabling composition and execution of semantically described medical informatics services for mobile caregivers, in: Biomedical Informatics for Clinical Decision Support, A vision for the 21st Century, BECON/BISTIC Symposium, Bethesda, MD, USA, 2004.
[33] S. Lee, N. Hashmi, J. Hendler, B. Parsia, Bio-STEER: An application of Task Computing — the Semantic Web meets Grid computing, Technical Report FLA-PCR-TM-3, Pervasive Computing Research, Fujitsu Laboratories of America, Inc., 2004.
[34] T. Lowe, S. Eddy, tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence, Nucleic Acids Res. 25 (1997) 955–964.
[35] D.R. Maddison, D.L. Swofford, W.P. Maddison, NEXUS: An extensible file format for systematic information, Syst. Biol. 46 (1997) 590–621.
[36] D.B. Mark Welch, M.P. Cummings, D.M. Hillis, M. Meselson, Divergent gene copies in the asexual class Bdelloidea (Rotifera) separated before the bdelloid radiation or within bdelloid families, Proc. Natl. Acad. Sci. USA 101 (2004) 1622–1625.
[37] J.C. Marks, M.P. Cummings, DNA sequence variation in the ribosomal internal transcribed spacer region of freshwater *Cladophora* (Chlorophyta), J. Phycol. 32 (1996) 1035–1042.
[38] R. Masuoka, Y. Labrou, B. Parsia, E. Sirin, Ontology-enabled pervasive computing applications, IEEE Intell. Syst. 18 (2003) 68–72.
[39] R. Masuoka, B. Parsia, Y. Labrou, Task Computing — the Semantic Web meets pervasive computing, in: Proceedings of the 2nd International Semantic Web Conference, Sundial Resort, Sanibel Island, FL, USA, 2003.
[40] Mindswap, http://www.mindswap.org/2004/ontolink/.
[41] D.S. Myers, M.P. Cummings, Necessity is the mother of invention: A simple Grid computing system using commodity tools, J. Parallel. Distr. Com. 63 (2003) 578–589.
[42] M.C. Neel, M.P. Cummings, Section-level relationships of North American *Agalinis* (Orobanchaceae) based on DNA sequence analysis of three chloroplast gene regions, BMC Evol. Biol. 4 (2004) 15.
[43] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, T. Carver, M.R. Pocock, A. Wipat, P. Li, Taverna: A tool for the composition and enactment of bioinformatics workflows, Bioinformatics 20 (2004) 3045–3054.
[44] OWL-S, http://www.daml.org/services/owl-s/1.0/.
[45] M. Pertea, X. Lin, S. Salzberg, GeneSplicer: A new computational method for splice site prediction, Nucleic Acids Res. 29 (2001) 1185–1190.
[46] D.D. Pollock, J.A. Eisen, N.A. Doggett, M.P. Cummings, A case for evolutionary genomics and the comprehensive examination of sequence biodiversity, Mol. Biol. Evol. 17 (2000) 1776–1788.
[47] D. Posada, K.A. Crandall, Modeltest: Testing the model of DNA substitution, Bioinformatics 14 (1998) 817–818.
[48] P. Rice, I. Longden, A. Bleasby, EMBOSS: The European Molecular Biology Open Software Suite, Trends Genet. 16 (2000) 276–277.
[49] F. Ronquist, J.P. Huelsenbeck, MrBayes 3: Bayesian phylogenetic inference under mixed models, Bioinformatics 19 (2003) 1572–1574.
[50] QT, http://www.trolltech.com/products/qt/index.html.
[51] S.P. Shah, D.Y.M. He, J.N. Sawkins, J.C. Druce, G. Quon, D. Lett, G.X.Y. Zheng, T. Xu, B.F.F. Quellette, Pegasys: Software for executing and integrating analyses of biological sequences, BMC Bioinformatics 5 (2004) 40.
[52] Simple Object Access Protocol, http://www.w3.org/tr/soap.
[53] E. Sirin, J. Hendler, B. Parsia, Semi-automatic composition of Web services using semantic descriptions, in: Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS, Angers, France, 2003.
[54] T.F. Smith, M.S. Waterman, Identification of common molecular subsequences, J. Mol. Biol. 147 (1981) 195–197.
[55] Z. Song, Y. Labrou, R. Masuoka, Dynamic service discovery and management in Task Computing, in: First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous'04, Boston, MA, USA, 2003.
[56] R. Stevens, A. Robinson, C.A. Goble, myGrid: Personalised Bioinformatics on the Information Grid, in: Proceedings of 11th International Conference on Intelligent Systems in Molecular Biology, 2003, Brisbane, Australia, Bioinformatics 19 (Suppl. 1) (2003) i302–i304.
[57] D.L. Swofford, PAUP*: Phylogenetic analysis using parsimony (*and other methods), version 4, Sinauer Associates. Sunderland, MA, USA.
[58] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, Y. Katz, Pellet: A Practical OWL-DL Reasoner, http://www.mindswap.org/papers/PelletJWS.pdf.
[59] Task Computing, http://taskcomputing.org.
[60] J.D. Thompson, D.G. Higgins, T.J. Gibson, CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position–specific gap penalties and weight matrix choice, Nucleic Acids Res. 22 (1994) 4673–4680.
[61] Universal Plug and Play, http://www.upnp.org.
[62] Washington University BLAST 2.0, http://blast.wustl.edu/blast/README.html.
[63] Wildfire, http://wildfire.bii.a-star.edu.sg/wildfire/.
[64] Wildfire/GEL, http://Web.bii.a-star.edu.sg/~francis/wildfiregel/.
[65] W3C Resource Description Framework, http://www.w3.org/rdf.
[66] W3C Web-Ontology (WebOnt) Working Group, http://www.w3.org/2001/sw/Webont.
[67] W3C Web Services Description Working Group, http://www.w3.org/2002/ws/desc.
[68] J. Yu, R. Buyya, A taxonomy of scientific workflow systems for Grid computing, SIGMOD Rec. 34 (2005) 44–49.



Sung Lee received her B.S., M.S., and Ph.D. degrees in computer science from the University of Maryland. She is a researcher at the Fujitsu Laboratories of America in College Park, Maryland. Her research interests include application of Semantic Web technologies in biomedical and life sciences, and Grid computing. Her previous experience includes network security, protocols, and Quality of Service support. Throughout her career, she has been involved with various aspects of software engineering at many commercial and non-commercial companies.



Taowei David Wang received his B.Sc. degree in mathematics and computer science from Duke University, and his M.S. degree in computer science from the University of Maryland, College Park. He is currently a Ph.D. student under the supervision of Dr. Jim Hendler in the Department of Computer Science at the University of Maryland. He is doing research in Semantic Web. More specifically, he is interested in effective user-end presentation of Web ontologies, including ontology visualization.

Nada Hashmi received her B.Sc. degree in computer science and mathematics from Washington College, and her M.S. degree in computer science from the University of Maryland, College Park. She is currently lecturing at the Management Informations Systems Department for the women's section at the College of Business Administration in Jeddah, Saudi Arabia. She also directs the Quality Assurance initiative at the College. Her past experience included directing the development of Web services for 10blade, inc; a medical informatics company based in Boston, MA. While at the University of Maryland, she conducted research with Dr. Jim Hendler and Fujitsu focusing on the application of the Semantic Web in the biomedical and life sciences.

Michael P. Cummings received a B.Sc. degree in botany from the University of California, Davis, and a Ph.D. degree in biology from Harvard University. He received an Alfred P. Sloan Foundation Postdoctoral Fellowship in Molecular Studies of Evolution and has done postdoctoral research at the University of California, Berkeley, and the University of California, Riverside. He is an Associate Professor in the Center for Bioinformatics and Computational Biology, University of Maryland, with appointments in the Department of Biology and the Institute for Advanced Computer Studies, and an affiliate appointment in the Department of Computer Science. He is also Director of the Workshop on Molecular Evolution, Marine Biological Laboratory, Woods Hole, Massachusetts. His research interests are in the areas of molecular evolutionary genetics, and computer science related to bioinformatics, computational biology and Grid computing.