

TeraGrid Open Life Science Gateway

Wenjun Wu¹, Rob Edwards^{2,4}, Ivan R. Judson³, Michael E. Papka^{1,4}, Mary Thomas², Rick Stevens^{1,4}

¹Computation Institute, University of Chicago & Argonne National Laboratory, USA

²San Diego State University, USA

³Montana State University, USA

⁴Math & Computer Science Division, Argonne National Laboratory, USA

Abstract

Most tools and algorithms of high-throughput bioinformatics data analysis are CPU-intensive, requiring high-end computing resources such as the TeraGrid. To facilitate the usage of the TeraGrid resources by the Life Science community for computing and data management, we have developed an integrated cyber computational environment named Open Life Science Gateway (OLSGW). OLSGW hides the complexity of accessing the distributed TeraGrid resources from biologists and bio-informaticians, and provides them with a user friendly web-service based interface. Since the user interface of OLSGW has the similar look and feel to bioinformatics web environments that have been available to the Life Science community for years, biologists can easily use this gateway to execute their analysis programs and compose computational workflow scripts without extensive knowledge about Grid technology. This paper describes the service-oriented framework of OLSGW and how it integrates a group of bio-informatics applications and data collections into a GridSphere[1] portal by customizing OGCE[2] portlets.

1 INTRODUCTION

Software tools and algorithms for high-throughput bioinformatics data analysis, such as sequence search, alignment and protein structure analysis, and so on, are CPU-intensive, requiring tremendous computing power which is usually beyond the capability of clusters at a single biomedical research institute. Instead, institutes can take advantage of available high-end computing resources – TeraGrid to run their computing tasks on a much larger scale.

However, the complexity of Grid middleware makes it a challenge for biologists and bioinformaticians without extensive knowledge of Grid technologies to utilize the TeraGrid resources. In order to enable the Life Science community to make full use of the TeraGrid resources for computing and data management, we have developed an integrated cyber computational environment named Open Life Science Gateway (OLSGW). It provides the Life Science community with an easy-to-use web-service interface, which has the similar look and feel to bioinformatics web environments that have been available to the community for years. Therefore biologists with no prior experience of Grid computing can easily use this gateway environment to run their analysis programs and compose computational workflow scripts without the challenges of a sharp learning curve.

Open Life Science Gateway (OLSGW) integrates a group of bio-informatics applications and data collections into a portal so that life scientists could easily access the resources of the TeraGrid to submit their Genome-related analysis programs, manage the Petta-bytes of datasets, and visualize protein structures. This science gateway established a solid foundation for high throughput genome and protein analysis workflows, helping scientists make great strides in solving the pressing problems faced by bioinformatics groups.

Although early bioinformatics tools used command line

programs extensively, a lot of web-based interfaces including CGI web applications and web-services have been developed to integrate the command-line bioinformatics tools for the life science communities. For example, PISE (Pasteur Institute Software Environment) [3] is a web interface (HTML and CGI) generator for more than 150 molecular biology command-line driven programs, including phylogeny, gene prediction, alignment, RNA, DNA and protein analysis, motif discovery, structure analysis and database searching algorithms. Another example is EBI (European Bioinformatics Institute) web-service interface [4] to bio-informatics application, which has bioinformatics applications including data retrieval, analysis tools, homology searches, and multiple sequence alignment.

These tools are designed to support bio-informatics applications on local resources within their own institutions. Open life science gateway is motivated to leverage these community-specific software toolkits and build a service-oriented web interface on the TeraGrid infrastructure. It supports both interactive web portlets and web-services interface to job execution and data management services.

This paper is organized as follows. In section 2, we introduce the general framework in the life science gateway. Details about the design and implementation are described in section 3. Finally, section 4 summarizes our work and points the further development of the gateway.

2 GENERAL SERVICES FRAMEWORK FOR LIFE SCIENCE RESEARCHERS

The framework of Open Life Science Gateway is designed based on Service Oriented Architecture:

Each bio-informatics command-line tool is considered a “Service”, which can be described in a XML file in the community-specific format. The gateway generates necessary stubs from the XML description for a bio-application and deploys the stubs as a web-service in the

system. Users are allowed to send requests to invoke the service through web pages or web-service interface. The gateway will create service instances upon request, and execute the service instances on the computing and storage resources of the TeraGrid.

Figure 1 illustrates the primary components in this framework. There are two layers in the diagram: the application service layer and the run-time layer. The application service layer has the meta-information that describes the command line format and run-time requirement of a legacy bio-application. The command line description includes argument lists, input files, and output files for this application. The application registry keeps the community-specific application description package -- PISE XML and the run-time requirements for applications such as architecture preference and QoS related attributes. The application service factory builds portlets including web pages and portlet codes from the application descriptions. The run-time layer supports the execution of deployed web services. The gateway offers a generic web-service which generates the command line wrapper script and input files based on the PISE XML description.

In this framework, A Job is viewed as a running service instance. Both portlets and web-services can call the generic Job Factory to create a job instance after they receive requests from users, and deposit the job instance into a persistent job storage queue. The engine for job execution – job manager pulls the pending job instance from the job storage, and submits the job into the allocated TeraGrid sites through Globus GRAM.

provides a generic web-service for all legacy applications and a static WSDL for every service deployed using Opal.

The difference between our effort and other web-service wrappers is that we focus on the customization of the generic service framework for the life science community. Similar effort can be found in BioPortal [7], which synthesizes the HTML forms generated by PISE into Velocity templates and import them into its portal. We leverage the same PISE package not only to create portlet user interface but also to build our web-service wrapper for those command-line bioinformatics applications. The other feature is the adoption of the intermediate job storage, which is used to queue all the job submissions before they are actually passed on to Grid resources. It enables more sophisticated schedule policies such as a job batch combination which could optimize the usage of the TeraGrid resources and improve the throughput of genome analysis.

In the following section, we will discuss the primary components of the gateway in detail.

3 CURRENT DESIGN AND IMPLEMENTATION

3.1 Application Registry and Job Storage

The Application registry holds the metadata for every registered application. It includes: the Application Name, the PISE XML description of this application, the location of the installed binary package, and the run-time requirement for the application.

The run-time requirement is defined to optimize the job submission and scheduling. For example, if a non-open source software package can only run on 32-bits architecture, we have to run this application on the TeraGrid clusters with 32-bit architecture instead of 64-bit. In this case, it is necessary to specify the architecture preference in the application registry. Applications also need to set their timeframes for jobs running in the application registry. The OLSGW gateway relies on the timeframe attribute to define the value of MaxWallTime which is mandatory for job RSL descriptions for every application.

Besides the application registry, OLSGW has to keep the host information for the allocated TeraGrid resources because there are multiple resource sites in the TeraGrid infrastructure. Every resource site assigns a different path for the home directory of the same user, has its own GRAM URL and Gridftp address, architecture (32bits/64bits) and different job schedulers. In the current OLSGW implementation, they are put in a property configuration file.

The job persistence storage, a MySQL database, keeps all the job instances created by the gateway to ensure the reliability and optimization of job submission. And Hibernate ORM library is used to map the Job objects to the relational tables in the database.

3.2 Creating application command scripts and job objects

Two steps are needed for generating the command line script, and input files for a particular application.

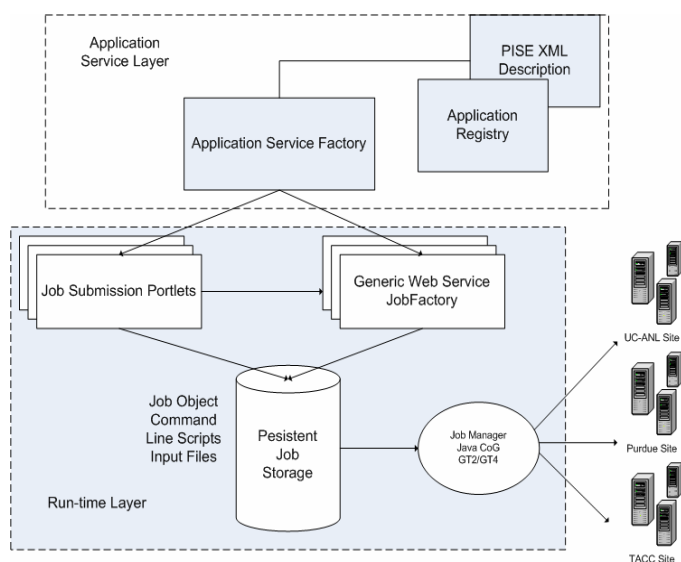


Figure 1 Open Life Science Gateway Framework

Many researches have been conducted to support the wrapping of legacy applications as web-service. The Generic Factory Service [5] is developed at Indiana University to wrap scientific applications as web-services and create new instances of these services on the Grid. The Opal toolkit [6] is another service wrapping toolkit which uses standard commodity SOAP toolkit -apache AXIS. It

- (1) In the initialization stage, OLSGW need to parse the PISE XML description and generate client-side stubs.
- (2) When a request comes from the user side, whether as a web form action or as a SOAP request, a job object is created in the database along with the job execution script and input files.

In a XML description file, every command line parameter has a name and a given type, followed by a list of attributes which describes their general behavior and properties. The attributes include the prompted text, the default value, and the code segment to be evaluated while the command line is being generated.

```

<parameter iscommand="1" ishidden="1" type="String">
  <name>clustalw</name>
  <attributes>
    <format>
      <language>perl</language>
      <code>"clustalw"</code>
    </format>
  </attributes>
</parameter>
<parameter ismandatory="1" issimple="1"
type="Sequence">
  <name>infile</name>
  <attributes>
    <precond>
      <language>perl</language>
      <code>${actions ne "-profile" and $actions ne "-
sequences"</code>
    </precond>
    <prompt>Sequences File (or Alignment File for
Bootstrap and Tree actions) (-infile)</prompt>
    <format>
      <language>perl</language>
      <code> "-infile=$value"</code>
      <language>seqlab</language>
      <code> "-infile=value"</code>
    </format>
  </seqfmt>
</attributes>
</parameter>

```

Table 1 PISE XML description for clustalw

Table 1 illustrates a segment in the PISE description for clustalw – a multiple alignment tool. The -infile argument is mandatory and needs to be specified as an input files with multiple DNA or protein sequences. Its code attribute helps the gateway to generate the command line.

Such a PISE XML description is treated as a job template that can be initialized as a command line script with specific parameters to run as a job in the TeraGrid resources. In OLSGW, Job.cmd is defined as a shell script file wrapping the actual executable bio-application command. An example of such a Job.cmd for Clustalw application is listed as follows:

```

#!/bin/sh
#
# This is a clustalw job script
#

PROGRAM="$HOME/tools/clustalw/bin/ clustalw "
INPUT="JOBDIR/input"
OUTPUT="JOBDIR/results"

# export necessary environment variable

cd $HOME

$PROGRAM -quicktree -input= $INPUT -
outfile=$OUTPUT

```

Table 2 Job.cmd for clustalw application

In the above example, there is another file – the input sequence, to be created along with the Job.cmd script. The input file contains the properly formatted input sequences to be processed by clustalw application. Besides the Job.cmd and input files, a persistent job object has to be created for the purpose of job submission and job history tracking. Such an object has three attributes – JobID, CmdFile and Status. Each job object has a unique JobID generated by concatenation of application name, current date and UUID string. The CmdFile property is the file path in which the generated command script resides. The Status property represents the possible status for the jobs: <CREATED, SUBMITTED, ACTIVE, CANCELED, ERROR, COMPLETE>.

There are three components required in OLSGW to complete the above steps – an Application Service Factory, generic web-service and Job Factory. The Application Service Factory is responsible for parsing an application description, generating the internal serialized Java Object, and formatting application specific web-pages and portlet codes. The factory creates JSP pages with the forms like PISE web pages and the portlet skeleton that invokes the generic web-service.

The internal serialized Java Object is loaded by the generic web-service to build a real command-line script based on the requests from users. This generic Job web-service is designed to handle SOAP requests for job initiation, job status query and job result retrieval as well as job destroy. It offers four major methods which can be invoked through SOAP.

Generic Web-Service

```

{
String      JobID      runJob(String      application,
HashMap<String,String> params);
String checkStatus(String jobId);
String getResults(String jobId);
String destoryJob(String jobId);
}

```

The runJob method actually calls a JobFactory to construct

a job instance that consists of a persistent job object and a command-line wrapper script. When the JobFactory is invoked to create a job instance, the factory will load the serialized command template object for this application, and generate the Job.cmd by replacing the predefined variables in the application description with the parameters specified by users. The input files for the program are also created with the input sequences in the right format.

Currently, the generic web-service is deployed in a AXIS container. User can use any SOAP library to invoke the generic web-service. Since Perl is widely used for bioinformatics programming, in Table 3 we give an example of a Perl script for submitting a clustalw job to explain how to access the web-services of OLSGW. Via SOAP::Lite library, users only need to specify the parameters for the clustalw command, and send the SOAP request to the generic web-service.

```

my $WSDL =
'http://lsgw.uc.teragrid.org/webservices/wsd/JobService.wsd';
my $soap = SOAP::Lite->service($WSDL);

my ($sequence, $outformat, $job, %params);
if (! defined($job)) { # Submit a job
  open(FH, $sequence) or die "Can't find file\n";
  ...
# Fill in the appropriate values
$params{'web-services-key'}='';
$params{'program'} = 'clustalw';
$params{'quicktree'} = 'fast'; #Parameter options
$params{'infile'} = $seq; # the query sequence
$job = $soap->run(SOAP::Data->name('params')
->type(map=>\%params));
print "Run Job Result($job)\n";
} else {
  print "No job submitted\n";
}

```

Table3. Perl example for accessing clustalw web-service

3.3 Job Manager

After a job instance is created, its status in the database is set to "Created". It waits for the Job Manager, which is a running daemon following a FIFO service policy to perform the data staging, job submission and the result retrieval tasks. The whole life cycle for this job instance includes four stages:

Created: The job instance has been created and is waiting to be processed in the database.

Created->Submitted: The job instance has been submitted to a real queue in the TeraGrid resources at this step. At first, the command line script and input files are staged in one of the TeraGrid host which is selected by the Job Manager. And a GRAM job is created for this job instance and submitted to the gatekeeper of the host through Java CoG library. The job manager then launches a listening thread to monitor the status of the job instance in the TeraGrid host and updates the status information in the database according to GRAM notifications.

Submitted->Active: After the pending GRAM Job obtains a computing node in the host and starts to run, the job thread receives the notification from the gatekeeper and sets the status of the job to "Active".

Active->Completed: When the execution of the job instance is finished, the gatekeeper sends a new notification to the job thread. Consequently, the job thread starts the data staging out, and retrieves the report (or error message) back to the gateway.

Sophisticated optimization strategy can be applied in the Job Manager to make further improvements on job submission. Job Manager can ensure the reliability of job submission. When the job is submitted to a TeraGrid host which is down at that time, Job Manager can keep pushing the job onto the same or different resource until it gets done. Job Manager can make high-level job scheduling to optimize the execution of tasks according to the workload of the TeraGrid resources. Based on the batch queue predictions and file transfer estimates offered by QBets [8] web-services, the job manager can choose the best TeraGrid compute resource to perform the task.

Job Manager can also help to optimize the running performance of bio-applications over large scale of data. Many life scientists tend to split a large DNA or protein sequences into small pieces and then use these pieces as inputs for individual jobs. If the length of the sequence segments is too short, the jobs will only run a very short time in a computing node. It results in a low throughput of the task execution on a cluster with a lot of workload since the small jobs of the tasks have the extra cost caused by job submission and queue time. To make full use of the resource, the Job Manager can combine the scripts from those small jobs into a large one, execute it as a single job on the TeraGrid host, and retrieve the results back separately.

Preliminary performance results show the effectiveness of the approach. In running an bio-application tool named AACPro [9] for the prediction of protein second structure, we have been given 4400 test sequences, each of which has only 100 amino acids. In an IA-32 machine (Dual Xeon, 2.4GHz), an AACPro job handling such a short protein sequence needs only about 2 minutes to finish. Without the combination optimization, it takes 12340 seconds to run through all the 4400 sequences on the TeraGrid UC/ANL IA-32 cluster. If 100 sequences are combined into a single job, the whole execution time is shortened to 8372 seconds, which results in 32% improvement in the throughput of the sequence analysis.

3.4 Deployed Bio-Application Tool

Four bio-applications have been integrated into the gateway, including BLAST [10], Hmmer [11], InterProScan [12], and ClustalW [13]. HMMER is an implementation of profile hidden Markov models for biological sequence analysis. InterProScan is a tool that integrates the search algorithms and protein signature recognition methods from the InterPro member databases into one resource, and provides the corresponding InterPro accession numbers and Gene Ontology (GO) annotation in the results. ClustalW is a widely used Multiple sequence alignment

computer program. Besides of the executable programs, both BLAST and InterProScan's installation have sequence databases, which are approximately 30 Gigabytes. We also have to maintain some customized blast databases (SEED [14]) for our user community. All the software packages including the databases are placed in the community software area.

Since the OLSGW is built upon a service-oriented framework and supports generic web-service based on the package of PISE XML description, it is easy to deploy bio-application packages with a correspondent PISE description. For other applications without PISE XML description, we can either add new description files for them or build dedicated web-service classes by extending the generic web-service. We are deploying more tools on the allocated TeraGrid resources for LSGW community account. For example, we have installed AACPro - a protein second structure tool and PHYLIP (PHYLogeny Inference Package) [15].

3.5 User Interface

OLSGW presents users with a group of portlets for job submission, proxy and data management. Figure 2 and 3 show two examples of the portlets currently deployed in the GridSphere container of OLSGW. The clustalw portlet is actually created on the basis of the clustalw-simple html form generated by PISE XML package. The form allows users to fill the input multiple-sequence files and set a few basic options. The portlet receives the form parameters and invokes the generic web-service to submit a clustalw job.

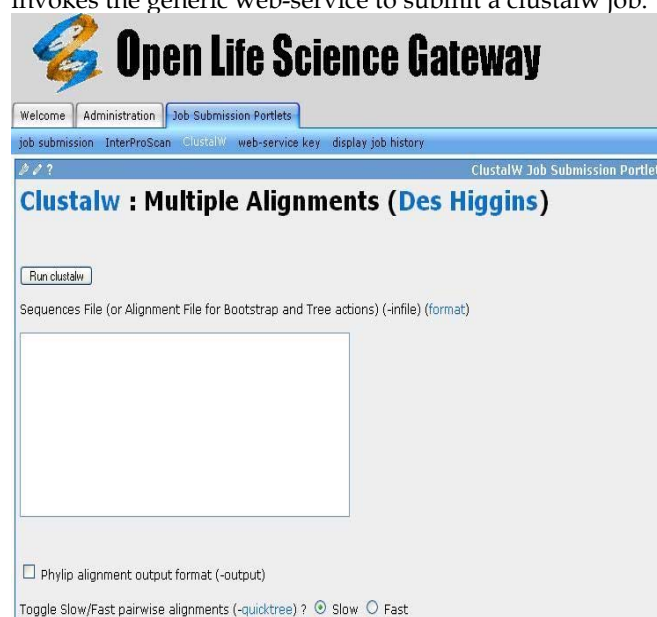


Figure 2 Clustalw Job Submission Portlet

Figure 3 display the job history portlet which lists the table of all the submitted job instances for the logged-in user. The user can check a job's information including the status, the input, the command script and the final result in this portlet. When he clicks the links of the Input, Cmd and Result, the JavaScript of the job history page fetches the information

from the portlet through JSON RPC, and presents the data in the area of the Job Detail in the page. In the Figure 3, the generated command-line script and the final result for the latest clustalw job is displayed.

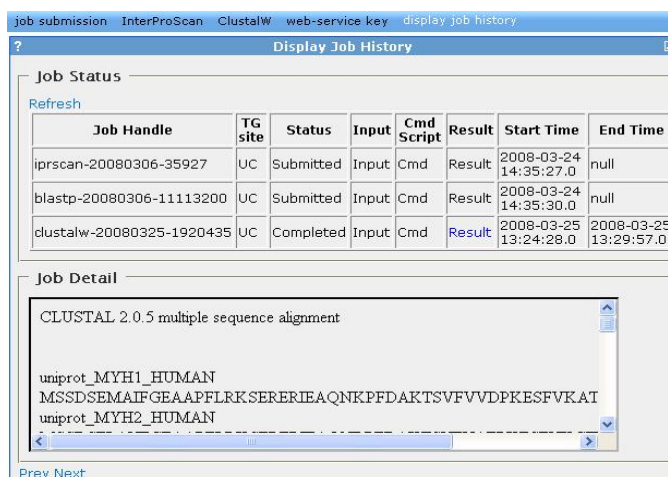
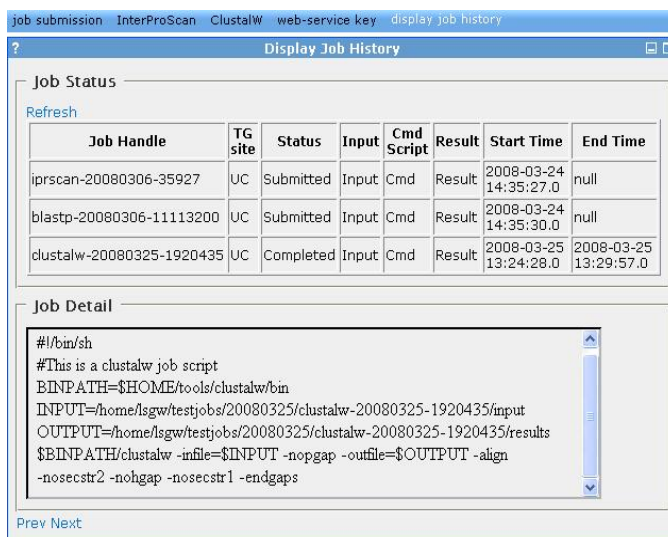


Figure 3 Job History Portlet

3.6 Security and Account Management

Security mechanisms are introduced in multiple layers of the gateway. Since the gateway is deployed as a GridSphere portal in Tomcat, it is effective to use built-in Tomcat security mechanisms for securing the gateway application. Tomcat can be configured to use SSL to encrypt the communication between web clients and the portal server. Moreover, JDBC security Realms in Tomcat supports authentication and authorization through checking a relational database that defines user name, password and role. Based on this security Realm, a simple web-service authentication solution is set up for OLSGW web-services. Each user is required to generate a web-service key which is regarded as a password for the authentication of accessing the AXIS web-services. The AXIS application is configured to use JDBC security and the database with the columns of the user name and web-service key.

For the general account management, the basic login

module in the GridSphere is customized to verify his/her email account when a user fills the registration form. If the user has a valid email address from an education or research institution, the GridSphere will send a grant notification email to the user.

OLSGW follows the transitive mode mentioned in the "AAA model to support science gateways with community accounts" [16] to enforce the access to the remote TeraGrid resources. The TeraGrid community account for the OLSGW project is used by the gateway to retrieve the temporary MyProxy credential from the MyProxy server of the TeraGrid, and use that credential for job submission and GridFtp data management.

4. Conclusion and Future Work

Open Life Science Gateway provides services to the Life Science community to access the TeraGrid resources for computing and data management. The extensible and service-oriented framework is introduced in developing the OLSGW prototype to enable the integration of many command line applications tools for biology researches. The generic web-service wraps any command line tool defined in the package of PISE XML description, therefore greatly leverages the capability of the OLSGW portal. Standard Grid tools such as OGCE and Java CoG are utilized as building blocks for the prototype portal.

As the OLSGW prototype becomes more stable, we are adding new tools in response to user feedback to promote the growth of the LSGW user community. We are also working with Globus gRavi group to implement WSRF web-service based on the current AXIS based implementation. And we plan to integrate light-weight and advanced cross-site resource managers into our job manager as well.

REFERENCES

- [1] GridSphere, www.gridsphere.org
- [2] Open Grid Computing Environment, http://www.collab-ogce.org/ogce/index.php/Main_Page
- [3] C. Letondal, "A Web Interface Generator for Molecular Biology Programs in Unix," *Bioinformatics*, 17(1), pp 73-82, 2001
- [4] Alberto Labarga, Franck Valentin, Mikael Anderson and Rodrigo Lopez, Web Services at the European Bioinformatics Institute, *Nucleic Acids Research Advance Access* published June 18, 2007
- [5] Gopi Kandaswamy, Liang Fang, Yi Huang, Satoshi Shirasuna, Suresh Marru, and Dennis Gannon. Building Web Services For Scientific Grid Applications. In *IBM Journal of Research and Development*, 2005.

[6] Sriram Krishnan, Brent Stearn, Karan Bhatia, Kim K. Baldrige, Wilfred W. Li, and Peter Arzberger, *Opal: Simple Web Services Wrappers for Scientific Applications*, 2006 IEEE International Conference on Web Services (ICWS 2006), Sep 18-22, Chicago

[7] BioPortal, <http://www.tgbiportal.org/>

[8] QBets,

<http://nws.cs.ucsb.edu/ewiki/nws.php?id=Batch+Queue+Prediction>

[9] AACPro (<http://download.igb.uci.edu/>)

[10] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", *Nucleic Acids Research*. 25:3389-3402, 1997.

[11] Eddy, S. R. , HMMER: profile HMMs for protein sequence analysis, *Bioinformatics* Vol. 14 , Page 775-763, 1998.

[12] Evgeni M. Zdobnov , Rolf Apweiler, InterProScan - - an integration platform for the signature-recognition methods in InterPro, *Bioinformatics* Vol. 17 no. 9 2001 Pages 847-848

[13] J.D. Thompson, D.G. Higgins, and T.J.

Gibson, CLUSTALW: Improving the Sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673-4680 , 1994.

[14] SEED project, http://www.theseed.org/wiki/Main_Page

[15] J. Felsenstein, PHYLIP Phylogeny Inference Package (Version 3.2). *Cladistics* 5: 164-166. , 1989.

[16] Von Welch1, Jim Barlow, James Basney, Doru Marcusiu , "AAA model to support science gateways with community accounts", *Concurrency and Computation: Practice and Experience*, Volume 19, Issue 6 ,2006, Pages 893 - 904