SLICEE: A Service oriented middleware for intensive scientific computation

Jonathan Piat, Francois Moreews, Olivier Collin, Dominique Lavenier, Alexandre Cornu

SLICEE: A Service oriented middleware for intensive scientific computation

A bioinformatics case study

Jonathan Piat, François Moreews, Olivier Collin,
Dominique Lavenier, Alexandre Cornu
Symbiose team
INRIA
Rennes, France
fmoreews@irisa.fr

***Abstract*—*With the growth of computational power and storage demand from bioinformatics applications, cluster computing has emerged as a good answer. Bioinformatics applications are often structured as workflows that are composed of a set of operations to perform on large data sets. These workflows are deployed as complex script that handles the sequence of program calls with their relevant inputs and try to take advantage of a computer cluster using a scheduler. Their performance rely on the user ability to analyze the potential parallelism in the workflow. SLICEE (Service Layer for Intensive Computation Execution Environment) proposes to abstract the calls to the cluster scheduler by handling command submission and data management. A workflow client is used to orchestrate the SLICEE services, that takes care of exploiting the data parallelism, and data routing between tasks to take advantage of the cluster architecture. A storage management layer takes care of optimizing data transfer with a reference passing mechanism. More-over SLICEE handles multi site command submission and data storage in a way that makes it transparent to the end-user. This tool enables an easy to maintain and to share implementation of bioinformatics workflows using intensive computation resources.*

Orchestration, SOA, service, HPC, parallelism extraction, middleware

I. Introduction

Bioinformatics applications such as tools for NGS (Next Generation Sequencing) data processing challenges today's computation resources by raising the amount of data to process and computation complexity to a new level. The common answer to such challenge is to use a computer grid or computer cluster with a large storage facility. Computer clusters combine the processing power of multiples machines that can be addressed as a single computing resource. A scheduler such as SGE (Sun Grid Engine), PBS (Portable Batch System), TORQUE or Condor is then in charge of dispatching the processing demand onto the available processing resources. Such architecture allows to take advantage of coarse grain parallelism to speed-up computation of scientific applications. This parallelism can either be used by running multiple applications in parallel or by designing the application in such way that it can be divided into smaller grain tasks to be achieved in parallel.

Bioinformatics applications can usually be described as a sequence of operation to perform on an input data-set. Such sequence of operations can be modeled using a workflow/dataflow model. This class of model represents the application as a network of operations connected through their data-dependencies. Such representation allows to retrieve the available parallelism in the application by analyzing data dependencies. Bioinformatics workflow are often deployed as a set of complex hand written scripts that represent the sequence of operations and their submission to the cluster scheduler with manual management of data dependencies between calls. These scripts are hard to maintain and their performance rely on the users ability to take advantage of the available data parallelism within the application.

Graphical workflow engines such as Kepler [1] propose to ease workflow specification through a graph editing UI. This tool also propose to schedule and run the workflow even addressing clusters through scheduler (SGE, Torque, PBS...) or Opal web service. Tasks are then achieved by uploading inputs, performing the task and download the outputs thus doubling the data transfer overhead. This makes it hard to deal with distributed data intensive applications, usually dealing with Gbytes of data. Kepler does not propose automatic data parallelism extraction from the workflow structure, but only rely on the user ability to represent it.

The SLICEE (Service Layer for Intensive Computation Execution Environment) tool proposes to address the parallelism available on a computer cluster with minimum user intervention. Its service oriented approach handles job submission with automated data parallelism extraction. Moreover this service permits multi site job submission and distant data storage on a non-shared file system with no user intervention. When running a workflow through SLICEE, input data are uploaded once if local and then final output can be retrieved without partial result downloading, thus limiting the required bandwith and data transfer overhead. SLICEE also features a user authentication that allow one user to manage its own data sets and execute applications with its access rights thus preserving existing access and quota policy.

This tool can be supported by most workflow engines. In this paper we demonstrate the use of SLICEE using the

Kepler workflow framework for graphical workflow edition, and tasks orchestration. In this approach Kepler is used to orchestrate the workflow but tasks execution is delegated to SLICEE with only references on data being exchanged between workflow tasks.

First we start with a describtion of the application context by giving a brief overview of the bioinformatics context and its challenging applications. A second part presents existing workflow execution systems while a third part presents SLICEE followed by an application example. In a final part we conclude and give directions for future work and improvements.

II. BIOINFORMATICS APPLICATIONS

Bioinformatics applies computational approaches to study molecular biological entities like genes and proteins. Many algorithms found in the bioinformatics field are computationally intensive and process a huge amount of data. To illustrate, the recent technological breakthrough in sequencing technology allows to decipher a genome sequence for an affordable price but has drastically increased the requirement of high performance computing resources and data storage. In an academic context, bioinformatics analysis usually consists in a script calling heterogeneous specialized software provided by the research community. Even if more and more tools like mpiBlast [2] use distributed computational resources like cluster-nodes or CPU-cores, the data parallelism pattern should often be manually coded. Many bioinformatics programs are command line based and can easily be wrapped as web services with dedicated toolkit like Opal [3]. Although the orchestration of such bioinformatics web services has proved to be realistic in a intensive computation production environment [4], the designer still needs to integrate in his workflow many components for data transfer, I/O [5] and data parallelism.

Bionformatics intensive computing platform commonly offer access to many users simultaneously. Each of them can run parallel jobs on the cluster-nodes. In this context, a strong authentication mechanism is important to monitor the jobs of each user and preserve user data confidentiality.

III. STATE OF THE ART ON SERVICE ORIENTED WORKFLOW APPLICATIONS FOR BIOINFORMATICS

Building a highly maintainable shared repository of bioinformatics workflows is challenging and can really speed-up genomics research. Multiple tools have emerged to propose service oriented workflow bioinformatics design and execution environment.

Conveyor [6], a workflow engine for bioinformatic analyzes, propose a library to design workflow components constructed with a strongly typed, interface driven, object oriented design. These components can wraps existing classes for functionality or integrate existing applications. A custom GUI allows the designer to instantiate these components into a workflow and monitor its execution. The workflow execution is performed through the Conveyor web service that orchestrate the components and execute them on a single machine, but supports inter-components parallelism by making use of multi-threading. This tool does not support access to data on a non shared file-system and currently only support applications execution on a single machine making it improper for use in a intensive computation environment.

The MeDICi Integration Framework (MIF) [7] propose to design components by encapsulating code into MIF components that can then be distributed across a cluster or a network of processing nodes. Components can exchange data either as reference or data copy to minimize data transfers. The components executes in a MIF container that manages a local data cache for inter components communication and allow to connect external MIF resources for remote execution. A Kepler integration was proposed in [8] that allow one user to design a Kepler workflow that makes call to MIF components to execute actors processing. This approach allows to exploit fine grain parallelism but impose to write applications using the MIF framework. This makes it unsuitable for bioinformatics legacy applications integration.

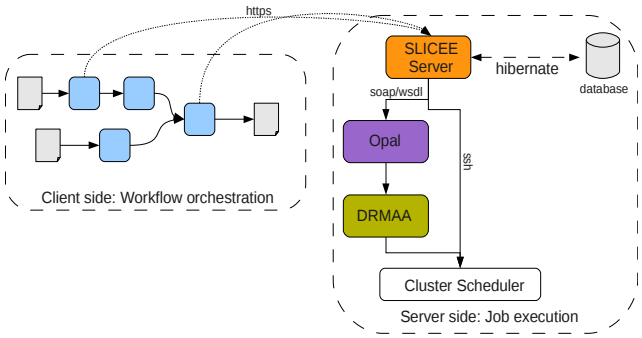
Taverna [9] are graphical workflow authoring and execution tools able to orchestrate remote web services or local component. Taverna's strong point is its ability to integrate a large set of Domain specific (BIOMOBY [10], BIOMART [11]), resource specific (OPAL) or generic (WSDL, REST) clients API.

The OPAL client API integrated in Taverna and Kepler is designed for bioinformatics intensive computation. It uses WSDL services with an asynchronous call pattern and proposes job scheduler submission capabilities. During the worfklow execution, submitted jobs are anonymized, without personal resource allocation policy. I/O data are passed by values or local file path. In such workflows, the instantiation of many glue component for I/O and data transfer is still necessary. Web based tools like Ergatis [12], Mobyle [13] and Galaxy [14] have been successfully adopted to make an intensive production environment available to the end user, the biologist. They are suitable to integrate intensive computation tools and support a large collection of bioinformatics components, and they have orchestration capabilities.

IV. THE SLICEE MIDDLEWARE

The SLICEE aims at providing the user with an easy way to exploit the available parallelism on a cluster to run bioinformatics applications at minimal development cost. A client allow the user to graphically design its application pipeline, and offload tasks execution and data management to the SLICEE service through web-services calls. Executed tasks are command line tools which allows to benefit of a large repository of applications. Data management helps at optimizing data transfer to limit inter-tasks communications overhead. The workflow orchestration is performed by the

calling client thus preserving the confidentiality of the workflow structure. This approach makes this tool relevant for both academic and industrial use.



Drawing 1: Overall tool structure. The client orchestrate a workflow that delegates tasks execution to the SLICEE service. This service can either submit the command through a web service (OPAL) or using a direct job submission in a ssh session.

A. SLICEE Features

1) Session Management

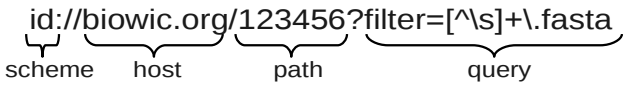
A workflow execution always start by opening a session in the application. Users can either be authenticated using an LDAP directory or using a locally stored user account. Once the user is authenticated, he obtains a session token that he can use to call SLICEE services such as job posting, data uploading/downloading. Moreover, when authenticated a user can access the data generated by its previous activities for use in the current session. These sessions can also helps at tracking the user activity for better error recovery and/or application debugging.

A user can also register identities that are domain specific authentication data. These identities can then be used to post job on distant clusters with an ssh authentication or retrieve data from hosts that requires authentication. This allow to execute applications and use data with limited access rights. Identities are not saved on persistent storage for security reason but only stored in ram for the session lifetime.

2) Storage service layer

SLICEE has its own data management system for better performance. This data management system allows the use of distant data repository (through scp, http and soon to come S3) with transparent data downloading uploading from the user perspective. Moreover this data management system maximize the data locality while posting job on distant cluster.

SLICEE references data through data sets. A data set is a set of files and directory that result from a job execution, a data uploading or a data transformation. A data set is identified by a URI with the “id” scheme, the server name as host and a long unique identifier. This URI can be associated to a query field that can contain a “filter=” query to extract specific data from the data set.



Drawing 2: SLICEE data set URI specification

A data references a file or a directory through a URI. These URI supports several VFS like scheme, that specify data access protocol and host. This allow SLICEE to access datas in a non-shared file system by using several protocol for copy, read and write in a VFS fashion. Thus protocol handling remains hidden to the user.

SLICEE also optimizes data locality to minimize bandwidth used by data transfer by avoiding data copy when possible.

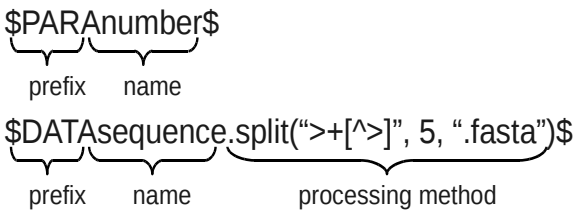
3) Job posting

The main purpose of the SLICEE is to execute commands using the processing power of a computer cluster. The cluster can either be addressed by a scheduler like SGE, Torque, PBS (only SGE supported at the time) or through a web service wrapping job schedulers, using OPAL. A command submitted to SLICEE consists in a command pattern with variable attributes. These variable inputs are surrounded by “$”. Two kind of variable are currently supported:

- A parameter variable is identified by the prefix “PARA” (see Figure [4]) and is replace in the command at execution time by the variable value.
- A data denotes a variable that references a data set. It is identified by the prefix “DATA”. This variable can be associated to a method that specify a processing to be applied on a data set by the server. Two processing methods are currently supported : splitting and joining (see Figure [4]) . The join method, groups the input data as a single input, while the split method cuts the input data into several chunks to be processed in parallel. This split method can take up to three arguments: *expr* that specify the regular expression to match on cut, *size* to specify data chunks size, and *extension* that specify the file extension for data chunks files.

```
blastall $PARA1$ -i  $DATA1:F$ -d\"$PARA2$\" $PARA3$
```

Drawing 3: Command pattern example. This pattern execute the command blastall with three different variable parameter and one input dataset that is the file containing the sequences.



Drawing 4: Variables syntax description. The parameter variable is composed of the prefix "PARA" followed by the parameter name and surrounded by $. The data variable is composed of the prefix DATA followed by the data name. An additional optional field can be used to use a processing method on the data set.

The command submission also contains a parameter values list and a data values list containing lists of data sets URIs. Other attributes specify the job posting url (either http for opal, or ssh for SGE) and the queue to post (only supported in SGE) to allows posting on accelerated queues running on FPGA or GPU.

In a first step SLICEE generates a set of command from the command submission. It starts by creating new data sets that contains the result of applying the processing method defined in the command pattern to the data sets pointed by the lists of URI contained in the command submission.

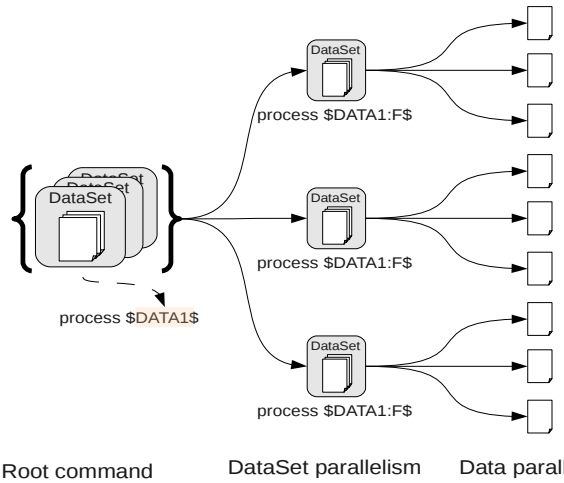
Then it generates a command for each of the data sets pointed by the $DATA..$ variables. At last a command is generated for each of the data contained in the data set that matches the URI filter (if present).

In a second step generated commands are submitted either through opal or directly to the job scheduler. For the opal submission the opal java API is used to create the submission. For the job scheduler submission a custom client creates job posting scripts that are then uploaded to the submission url with input files (if not already available locally) and launched using a ssh session (opened using user identities). Once a job is posted, the server returns a list of URIs that references the data sets resulting of the jobs execution. This URI can be used to track the jobs status, access jobs results when the job has ended or used as command input for later command submission.

4) Parallelism extraction

One of the main feature of the SLICEE job submission service is that it extracts the available data parallelism to execute one command submission as a set of parallel commands. For single data input commands, parallelizing is straightforward, as as much command as the data set datas can be launched. For a command taking N data inputs and P data sets for each input, N^P commands are generated, covering all the possible input combination.

Moreover as the result of a job submission is a list of data sets, following tasks may take advantage of the parallelism revealed by the previous command. For a command that generates N files, the following tasks may run in N parallel command executions.



Drawing 5: Parallel processing of a single command submission with multiple input datasets. A command submission takes a list of datasets as input. A command is created for each element of the list. Moreover a dataset is composed of multiple files that are processed in parallel.

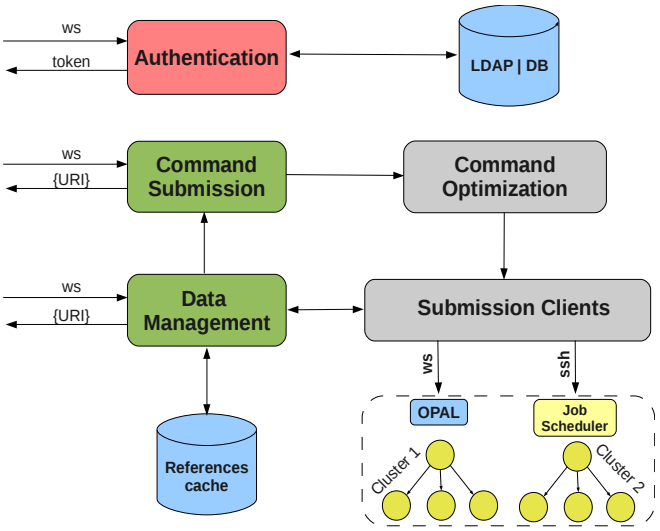
	Taverna	*MIF*	*Conveyor*	*SLICEE*	*Galaxy*	*Mobyle*
Multi-site	++	++	-	++	-	-
Authentication	-	-	-	++	+	+
Applications repository	++++	-	-	++	++++	+++
Data management	-	++	+	++	-	-
Web-based GUI	-	-	-	-	+	+
SOA	+++	++	+	++	-	-
Job scheduler support	-	-	-	+++	++	++

Table 1: Comparison of the workflows execution tools. MIF an SLICEE can be compared due to their common integration with Kepler. Galaxy and Mobyle propose a full web approach while Taverna only focus on discovery and orchestration. Taverna is also an interesting front-end to call SLICEE services.

B. Kepler SLICEE client

We implemented a Kepler based client to interface with SLICEE. Kepler is a scientific workflow tool based on Ptolemy. Kepler provides the graphical user interface for workflow edition, and workflow tasks orchestration. Our Kepler plugin implements several actors for SLICEE command submission, data sets filtering and data preview. The command submission actor (called Easy Remote

Execution) is a configurable actors that takes the SLICEE formatted command pattern as an attributes and automatically generates actor ports for variables elements. When executed, the actor first upload to the server the input data if local, and then submits the command to the server. When the command ends, the actor only outputs the URIs of the resulting data sets. These data sets can then be used as actors input without downloading their content or previewed (limited in size) within the UI. This client also implements a logging feature to log the workflow execution for testing and recovery purpose. A recovery actor allows to perform error recovery on the client side by using a log file to restore the workflow to a given state. This allow to recover error that may result from a service or connection failure.Multiple example of bioinformatics applications were developed to show the tool capabilities.



Drawing 6: SLICEE propose three services layer to manage the different workflow execution aspects. The authentication layer ensure the user authentication consistency along the workflow execution. The command submission makes use of the available data parallelism to generate the command set to execute. The Data management service, allows the user to upload data, download results and manages the multi-site data reference cache.

V. Application Case Study

In this example we demonstrate the use of SLICEE through a phylogeny workflow designed using Kepler. This workflow shows a simple phylogenomics application for predicting the function of a protein. Input is a protein sequence which is scanned against the Swissprot protein knowledge base to retrieve sequences with homologies. The resulting set of sequences is thus globally aligned, processed to remove ambiguously aligned regions, then a phylogenetic tree is built.

To accelerate the workflow execution the Blastp and ClustalW operations are accelerated using a FPGA implementation that can be used through a special job scheduler queue. The idfetch, Gblocks [15], Phylip [16] tasks runs in general purpose queue. As FPGAs are more efficient when processing large volume of data, input datas of Blastp and ClustalW uses the method “join()” on the data inputs to ensure these tasks runs a single command with a large amount of data. In contrary tasks idfetch, Gblocks and Phylip uses the “split()” method on their inputs with relevant arguments to process multiple data chunks in parallel for better performance.

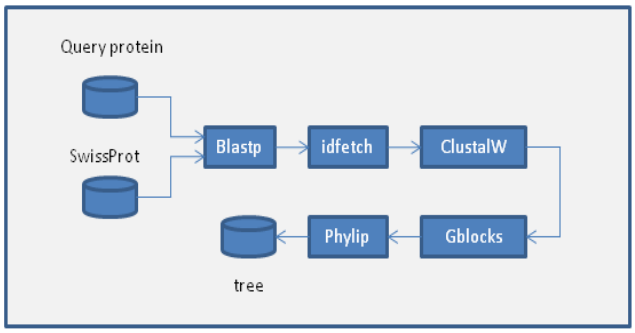


Illustration 1: Structure of the phylogeny workflow. This example workflow exposes multiple levels of parallelism. Fine grain parallelism is exploited on Blastp and ClustalW using a FPGA implementation. IdFetch Gblocks and Blastp benefits of the coarse grain data parallelism extracted by SLICEE.

This example was tested on the *GenOuest* cluster that provide computing resources like FPGA, GPU and general purpose high performance cores. All these resources benefits from a shared file-system. This approach permits to exploit both the intrinsic parallelism of an FPGA by feeding it with large volume of data and still take advantage of the coarse grain parallelism available on the cluster for applications executing on general purpose cores. Moreover the data management system of SLICEE using only references on data, minimize the data transfer between tasks. Only two data transfers occurs in this example, one for input sequence uploading and one for final result downloading. In the case of a multi-site execution, transfer would only have occurred on purpose.

VI. Conclusion and Future Work

The SLICEE middleware is designed to be used on a single cluster but multiple instances deployed on different remote sites can be called by a client within the same workflow execution. It provides a partial lightweight distributed execution environment, far from classical grid computing environment with mechanisms like certificate based authentication and resource sharing but much easier to deploy. Furthermore, the fast increase of requirements for

bioinformatics treatment leads to the adoption by bioinformatics computing platforms of an hybrid or private cloud model. Service oriented middlewares like SLICEE will enable a smooth migration of bioinformatics workflows execution from clusters to cloud.

Cloud Storage systems have proved to be a valuable option for scientific workflows [17]. Thus we plan to integrate the support of cloud storage services like amazon S3 in the storage service layer of SLICEE. The design paradigm of Service-oriented architecture provides important benefits to the bioinformatics field driven by an academic scientific community where shared infrastructures and collaborative development efforts are mainly adopted.

Today bioinformatics web services consists in database content search tools and analysis tools but even if bioinformatics services orchestration editors are attracting interest, their usage remains limited to educational or prototyping purposes but dedicated service layers can bring the power of intensive computation to the end user.

VII. Acknowledgments

SLICEE is developed within the BioWIC project (Bioinformatics Workflows for Intensive Computation) , funded by the ANR (french National Research Agency) ANR-08-SEGI-005. We also thank the *GenOuest* and the *Migale* bioinformatics platform for hosting the BioWIC tools.

[1] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the Kepler system: Research Articles," *Concurrency and Computation: Practice & Experience*, vol. 18, Aug. 2006, pp. 1039–1065.

[2] A. Darling, L. Carey, and W. Feng, "The Design, Implementation, and Evaluation of mpiBLAST."

[3] S. Krishnan, B. Stearn, K. Bhatia, K.K. Baldridge, W.W. Li, and P. Arzberger, "Opal: SimpleWeb Services Wrappers for Scientific Applications," *Web Services, IEEE International Conference on*, Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 823-832.

[4] T.L. Bailey, M. Boden, F.A. Buske, M. Frith, C.E. Grant, L. Clementi, J. Ren, W.W. Li, and W.S. Noble, "MEME Suite: tools for motif discovery and searching," vol. 37, Jul. 2009, pp. W202-W208.

[5] W. Li, S. Krishnan, K. Mueller, K. Ichikawa, S. Date, S. Dallakyan, M. Sanner, C. Misleh, Zhaohui Ding, Xiaohui Wei, O. Tatebe, and P. Arzberger, "Building cyberinfrastructure for bioinformatics using service oriented architecture," 2006, pp. 8 pp.-39.

[6] B. Linke, R. Giegerich, and A. Goesmann, "Conveyor: a workflow engine for bioinformatic analyses," *Bioinformatics (Oxford, England)*, Jan. 2011.

[7] I. Gorton, A. Wynne, J. Almquist, and J. Chatterton, "The MeDICi Integration Framework: A Platform for High Performance Data Streaming Applications," *Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference on*, 2008, pp. 95-104.

[8] J. Chase, I. Gorton, C. Sivaramakrishnan, J. Almquist, A. Wynne, G. Chin, and T. Critchlow, "Kepler + MeDICi," *Services, IEEE Congress on*, Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 275-282.

[1] T. Oinn, M. Greenwood, M. Addis, M.N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M.R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, ?Taverna: lessons in creating a workflow environment for the life sciences: Research Articles,? *Concurrency and Computation: Practice & Experience*, vol. 18, Aug. 2006, pp. 1067?1100.

[10] E. Kawas, M. Senger, and M.D. Wilkinson, "BioMoby extensions to the Taverna workflow management and enactment software," *BMC Bioinformatics*, vol. 7, 2006, p. 523.

[11] D. Smedley, S. Haider, B. Ballester, R. Holland, D. London, G. Thorisson, and A. Kasprzyk, "BioMart--biological queries made easy," *BMC Genomics*, vol. 10, 2009, p. 22.

[12] J. Orvis, J. Crabtree, K. Galens, A. Gussman, J.M. Inman, E. Lee, S. Nampally, D. Riley, J.P. Sundaram, V. Felix, B. Whitty, A. Mahurkar, J. Wortman, O. White, and S.V. Angiuoli, "Ergatis: a web interface and scalable software system for bioinformatics workflows," *Bioinformatics (Oxford, England)*, vol. 26, Jun. 2010, pp. 1488-1492.

[13] B. Néron, H. Ménager, C. Maufrais, N. Joly, J. Maupetit, S. Letort, S. Carrere, P. Tuffery, and C. Letondal, "Mobyle: a new full web bioinformatics framework," *Bioinformatics (Oxford, England)*, vol. 25, Nov. 2009, pp. 3005-3011.

[14] J. Goecks, A. Nekrutenko, J. Taylor, and T. Galaxy Team, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biology*, vol. 11, 2010, p. R86.

[15] J. Castresana, "Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis," *Molecular Biology and Evolution*, vol. 17, Apr. 2000, pp. 540-552.

[16] J. Felsenstein, "PHYLIP - Phylogeny Inference Package (Version 3.2)," *Cladistics*, vol. 5, 1989, pp. 164-166.

[17] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B.P. Berman, and P. Maechling, "Data Sharing Options for Scientific Workflows on Amazon EC2," *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–9.