An Agent Approach for Protein Function Analysis in a Grid Infrastructure

Alessandro ORRO [a,b,1] and Luciano MILANESI [a]
[a] *Institute for Biomedical Technologies - Italian National Research Council*
Via Fratelli Cervi 93, I-20090 Segrate (MI) – Italy
[b] *Consorzio Interuniversitario Lombardo per L'Elaborazione Automatica*
Via R. Sanzio, 4 - Segrate (MI) – Italy

Abstract. Many tasks in bioinformatics can be faced only using a combination of computational tools. In particular, functional annotation of gene products can be a very expensive task that may require the application of many analysis together with a manual intervention of biologists. In this area, the phylogenomics inference is one of the most accurate analysis methodologies for functional annotation that is not yet widely used due to the computational cost of some steps in its protocol. This paper discusses the implementation and deployment of such analysis protocol in a distributed grid environment using an agent architecture in order to simplify the interaction between users and the grid.

Keywords. bioinformatics, protein function, grid computing, agent systems.

Introduction

Nowadays, the huge amount of biological sequences produced in sequencing projects increases the need for automatic tools for structural and functional annotation. Manual annotation supported by experimental analysis is often very accurate, but it is not able to keep up with the flow of data in high throughput sequencing experiments.

The simplest method for functional characterization of proteins is the sequence similarity analysis. Unfortunately, when an evident similarity is not observed, it is difficult to adopt a transitive assignment. On the other hand there are many known proteins that share the same function but are dissimilar in their amino acid sequences. Many methods have been developed in order to improve the similarity-based function assignment. Let us recall OntoBlast [1], GOBlet [2] and GOtcha [3] that weights the functions represented in similar sequences using the E-value derived from a blast search. The GOFigure [4] software uses a similar approach but the amount of possible functions is reduced by minimizing the graph of represented nodes. GOAnno [5] annotates a sequence by propagating the GO terms through subfamilies in a hierarchical multiple alignment.

[1] Corresponding Author: Alessandro Orro, Institute for Biomedical Technologies – National Research Council, via Fratelli Cervi 93, 20090 Segrate (MI), Italy; E-mail: alessandro.orro@itb.cnr.it

However, as for many other bioinformatics tasks, protein function assignment usually requires the use of several programs organized in computational pipelines. In this context the manual intervention of a biologist is often needed in order to choose, in each step of the pipeline, the best strategies to improve the quality of classification.

In this paper we describe the implementation, in the EGEE Grid environment, of a protein function classification method based on the phylogenomics pipeline. The implementation makes use of software agents in order to hide some of the complexity of the grid and exhibit to the users only a simplified user oriented interface.

Materials and Methods

Methods based on assignment from highly similar sequences perform quickly and therefore are suitable especially for a preliminary annotation of a whole genomes but the underlying model is known to have significant drawbacks mainly due to evolutionary events (for example gene duplication and shuffling). Phylogenomics protocol [6] [7] has been conceived to remove these limitations by using evolutionary information contained in a phylogenetic tree built from the set of sequences related to the target. Functional annotations are transferred only if they are consistent with the tree structure and the genetic events found in the tree.

The general schema of the phylogenomics analysis is shown in Figure 1. The target sequence is queried against a database by similarity in order to find out potential related sequences that are globally aligned to highlight similar regions. The alignment is used to produce a phylogenetic tree that is annotated with functional and evolutionary information. Finally an inference rule is applied in order to transfer functions from the nodes in the tree to the target sequence.

Although phylogenomic analysis is deemed to be one of the most accurate in the inference of sequence functions, it is not easy to adopt on large scale. In fact some computation in the analysis, like the construction of the multiple alignment and the phylogenetic tree can be very expensive. Moreover, to take full advantage of such methodology, all steps of the pipeline need to be accurate enough to exclude false positives in the set of homologous sequences and not leave out important ones.

1.1. Grid Infrastructure

The EGEE Grid Infrastructure [8] adopted in this work is a network of computing and storage resources connected together with the gLite middleware [9] that provides a framework for the management of grid nodes and a complete command line API for access the distributed resources (job and data).

Figure 2 shows the main components of the EGEE Grid and their interactions. A job, described in the JDL format, is submitted by the user from a *User Interface* (UI) to the *Resource Broker* (RB) that processes the job in order to find a *Computing Element* (CE) matching user requirements. When the Computing Element receive a job from the Resource Broker, executes it in a cluster of *Worker Nodes* (WN).

Security is handled through *Personal Certificates* that allow users to be recognized in the infrastructure and to execute their jobs for a limited authentication time. Usually users belong to a *Virtual Organization* (VO) that represents a logical group of users that share common research interests and projects.

1.2. Execution Framework

The Grid Infrastructure is not an integrated system in which the correct overall behaviour is guaranteed but it is a network that connects independent machines each one providing a particular service. Therefore, due to synchronization or overload problems, there are a number of aborting jobs or staying scheduled for a long time. Moreover, the execution time of a complex pipeline often takes a time longer than the personal certificate time life.

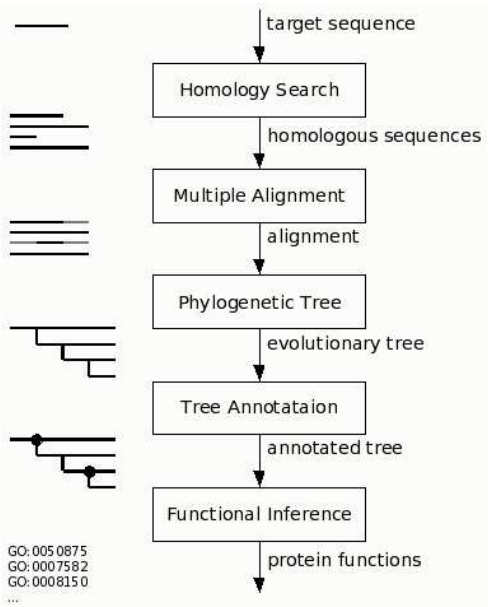

Figure 1. Phylogenomics pipeline from the target sequence to the functional annotation of GO terms.

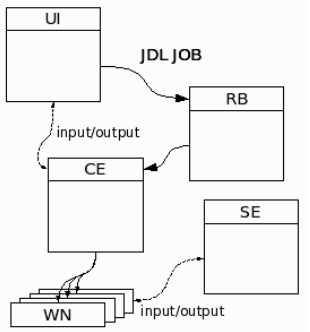

Figure 2: Main components of the EGEE Grid Infrastructure.

The proposed agent framework makes the execution of job more robust hiding the complexity of the underline grid infrastructure. In the agent paradigm [10] a program (the agent) is not directly invoked by the user or other program, but it is able to decide how and when to perform its action. In fact the actions performed by agents are mainly goal oriented i.e. based on an assigned goal rather than enabled by a function call.

The use of agent systems in the bioinformatics field has been focused on several aspects: genome annotation [11] [12], text mining [13], resource integration [14] [15] and the management of webservices-based grid environment (MyGrid) [16].

The proposed agent system is mainly devoted to enable users to easily and effectively interact with the grid environment and to manage job execution. All tasks related to the authentication, data management and low level job scheduling are left to the underline grid environment while the agents wrap the grid services in order to export to the user a more application oriented view of the computational resources. Four types of agents have been implemented in the system:

- *UserAgent* is devoted to the user interaction. Its main duty is to take care of all the communications needed to ask the user about some choice during the execution of the pipeline. In particular interactions occur when the user is alerted when a job finishes or when a partial output has to be checked and approved.
- *GridAgent* is devoted to all the operations concerning the Grid Environment (execution of jobs, proxy management, read/write files in storage element). In particular it interacts with the Grid through primitive commands of the gLite framework.
- *NodeAgent* is the node that wraps a single application in the pipeline and takes care of the job distribution.
- *ExecuteAgent* executes the whole pipeline interacting with the *NodeAgents* and updating the execution environment.

Figure 3 depicts the agent infrastructure. Each node of the pipeline is wrapped by a NodeAgent able to execute local or remote programs. The most part of NodeAgents performing computational intensive jobs execute programs through the grid distributed environment. In this case the NodeAgent executes jobs by requesting one or more services to the GridAgent. It also takes care of distributing the jobs between the nodes in grid by splitting application jobs in JDL jobs and reconstructing the output of each job from the outputs of the related JDLs.

ExecuteAgent manage the execution of a pipeline of application jobs each one represented by a NodeAgent. Its main duty is to coordinate the execution of the NodeAgents by implementing some strategies in order to better exploit the features of the grid and automate some routine work. In particular

1. enforcing resubmission of failed jobs
2. changing CE destination in case of repeated failures
3. periodic refreshing of status of the jdl jobs
4. retrieving outputs and cleaning temporary folders and files

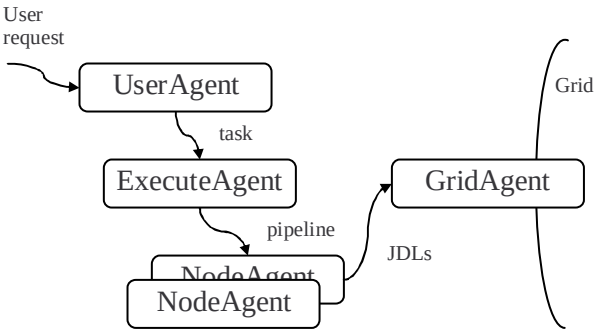

Figure 3. Agent infrastructure.

Results

The infrastructure has been tested in a particular implementation in which the pipeline analysis is composed by the following steps

5. the target sequence is searched with the blast program in order to find out all sequences similar to the target one. Sequences gathered at this step are stored in a relational database allowing user to manually remove entries from the set and providing to the next analysis step possibly only “true” homologs
6. homologs sequences are aligned with the MAFFT multiple alignemt program [17] that allow a good compromise between efficency and quality. Like the first step the resulting alignment can be edit by the user and orphan sequences removed or realigned.
7. phylip program [18] has been chosen for building the phylogenetic tree from the multiple alignment.
8. sequences in the resulting tree are annotated locally by quering external database (depending on the data source used in the blast search), running the Forester program [19] for labelling duplication/speciation nodes.

Outputs of all the programs are collected in files as partial outputs and stored in grid storage in order to reduce network load between the user interface and the grid environment. When a user intervention is required on the data, they are also parsed and stored in a relational database.

Steps 1, 2 and 3 generally associated with different jdl jobs. In particular distribution strategy enforce grouping of single jobs in sets of jdl jobs that are executed sequentially in a single worker node. This allows to make the overall time of each job longer enough and to better exploit the overhead due to the grid middleware.

Steps 4 concerns above all the visualization and integration of the results produced in previous steps and the functional annotation of the sequences in the tree (especially the target one). This is an operation that often requires user intervention and it difficult to automate. Some inference rules have been implemented (like the N-best-hits, and other function weighing methods [3]) in order to make automatic inference and other rules can be added by users. Threshold values are used for alert user that the prediction

could be not accurate and another intervention is required. All these operation are performed locally.

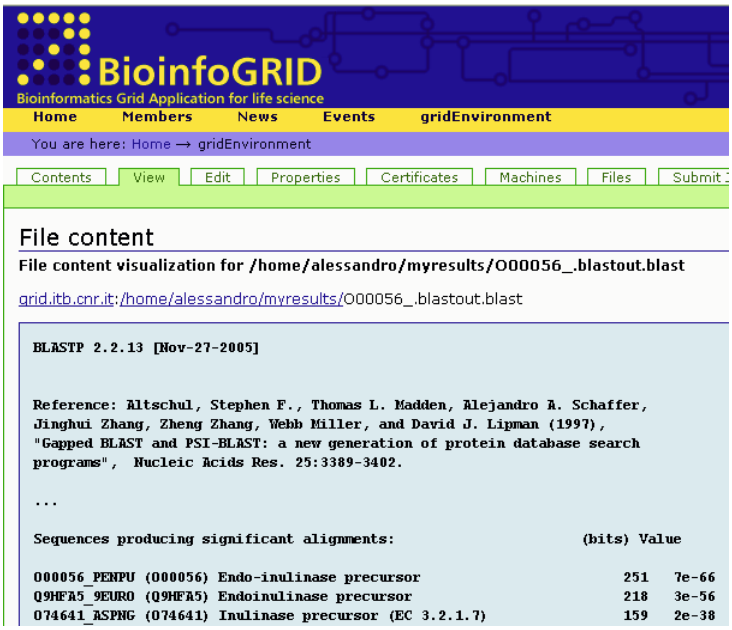

Figure 4. A screenshots of the Web portal for the management of grid-oriented application.

All the agents are implemented as active objects in a web portal that follow user actions or modification in the grid (for example change of the job status).

The UserAgent takes input from the users through a web portal built with the Zope Framework [20]. Users can navigate in storage elements and user interface and visualize both local and remote files that represent partial of final results of the submitted jobs. In particular outputs can be shown in different view modes (ascii text, xml, graphical) depending of the meta type of the file. For example, figure 4 shows the visualization of an ascii file in the user interface containing the result of a blast search.

The ExecutionAgent acts like a cron job that periodically refresh the status of all JDL jobs and automatically download the output of finished jobs. All these tasks are done by holding some information about each submitted job JDL in a relational database. These information include the grid identifier, the start time, the computing element executing the job, the job status and the name of input and output files.

Each time a user runs an application, the corresponding NodeAgent generates a set of job JDLs and puts them in the database of jobs managed by the ExecutionAgent. Figure 5 show an example of creation of a blast job that will be splitted in 5 independently jdls jobs each one running in a (potentially) different computing elements and processing a different portion of the input file.

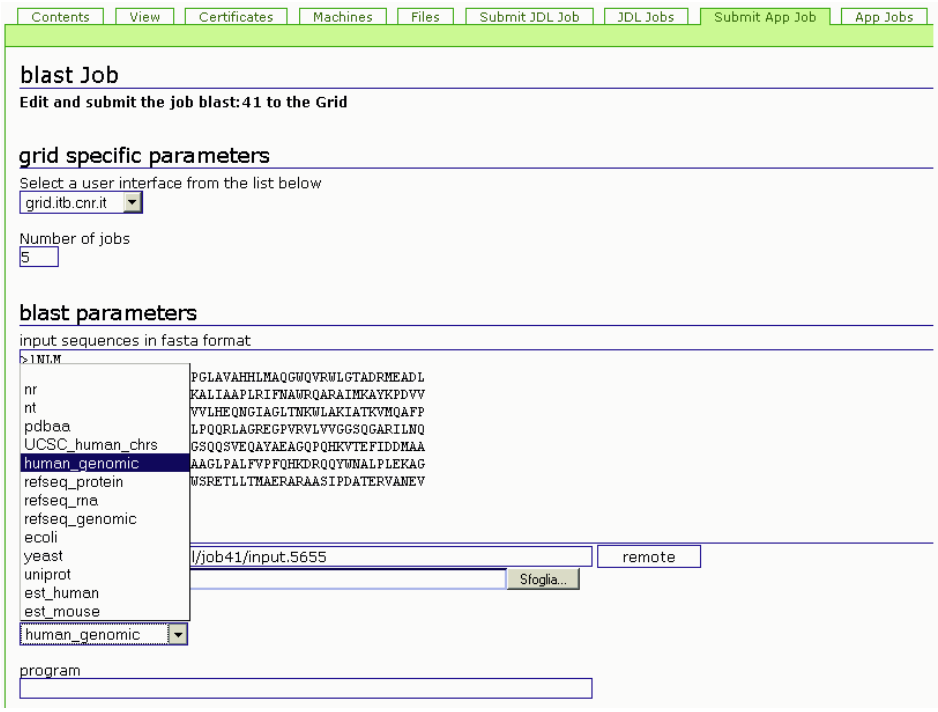

Figure 5. Parameters for the blast application job splitted in 5 jdls jobs

Conclusion

In this paper an agent system for the management of a distributed grid pipeline has been presented. It allow the user to access the computational resources exposed by the EGEE Grid in a application oriented way. This is done by hiding to the users some details that usually are performed in command line mode in the grid user interface.

Acknowledgements

This work has been supported by the Italian FIRB-MIUR project “LITBIO - Italian Laboratory for Bioinformatics Technologies” and by the European Specific Support Action BioinfoGRID and EGEE projects.

References

[1] G. Zehetner, OntoBlast Function: From Sequence Similarities Directly to Potential Functional Annotations by Ontology Terms, *Nucleic Acids Research* **31** (2003), 3799–3803.

[2] S. Hennig, D. Groth and H. Lehrach, Automated Gene Ontology Annotation for Anonymous Sequence Data, *Nucleic Acids Research* **31** (2003), 3712–3715.
[3] D.M. Martin, M. Berriman and G.J. Barton, GOtcha: a new method for prediction of protein function assessed by the annotation of seven genomes, *BMC Bioinformatics* **5** (2004) 178.
[4] S. Khan, G. Situ, K. Decker and C.J. Schmidt, GoFigure: automated Gene Ontology annotation, *Bioinformatics* **19** (2003), 2484–2485.
[5] F. Chalmel, A. Lardenois, J.D. Thompson, J. Muller, J.A. Sahel, T. Leveillard and O. Poch, GOAnno: GO annotation based on multiple alignment, *Bioinformatics* **21** (2005), 2095–2096.
[6] J.A. Eisen, C.M. Fraser: Phylogenomics: intersection of evolution and genomics. *Science* **300** (2003),1706–1707.
[7] K. Sjolander: Phylogenomic inference of protein molecular function: advances and challenges, *Bioinformatics* **20** (2004), 170–179.
[8] F. Gagliardi, B. Jones, F. Grey, M.E. Begin, M. Heikkurinen, Building an infrastructure for scientific Grid computing: status and goals of the EGEE project, *Philosophical Transactions: Mathematical, Physical and Engineering Sciences* **363** (2005), 1729–1742.
[9] gLite: Lightweight Middleware for Grid Computing, http://glite.web.cern.ch/glite
[10] S.J. Russel and P Norvig, Artificial Intelligence: A Modern Approach, *Prentice Hall*, 2002.
[11] K. Bryson, M. Luck, M. Joy, D Jones: Applying agents to bioinformatics in Geneweaver, In Cooperative Information Agents IV, Lecture Notes in Artificial Intelligence, (2000), 60–71.
[12] K. Decker, S. Khan, C. Schmidt, G. Situ, R. Makkena, D. Michaud: BioMAS: A Multi-Agent System for Genomic Annotation, *International Journal Cooperative Information Systems* **11** (2002), 3:265–292.
[13] A. Doms and M. Schroeder: GoPubMed: exploring PubMed with the Gene Ontology, *Nucleic Acids Research* **33** (2005) 783–786.
[14] K.A. Karasavvas, R. Baldock, A. Burger: A criticality-based framework for task composition in multi-agent bioinformatics integration systems, *Bioinformatics* **21** (2005), 3155–3163.
[15] K. A. Karasavvas, R. Baldock, A. Burger: Bioinformatics integration and agent technology. *Journal of Biomedical Informatics* **37** (2004) 3:205–219.
[16] L. Moreau et al.: On the Use of Agents in BioInformatics Grid. CCGRID, Proceedings of the 3st International Symposium on Cluster Computing and the Grid, 2003.
[17] K. Katoh, K. Misawa, K. Kuma, T. Miyata: MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform, *Nucleic Acids Research*, **30** (2002) 14:3059–3066.
[18] J. Felsenstein: PHYLIP (Phylogeny Inference Package) version 3.6 (Department of Genetics, University of Washington, Seattle, WA). 2002.
[19] C.M. Zmasek, S.R. Eddy. A Simple Algorithm to Infer Gene Duplication and Speciation Events on a Gene Tree. *Bioinformatics* **17** (2001), 821–826.
[20] www.zope.org