The LIBI Grid Platform for Bioinformatics

Maria Mirto®, Italo Epicoco®, Sandro Fiore®, Massimo Cafaro®, Alessandro Negro®, Daniele Tartarini®, Daniele Lezzi®, Osvaldo Marra®, Antonio Turi®, Alessandra Ferramosca®, Vincenzo Zara®, Giovanni Aloisio®

⊗Università del Salento, Lecce & SPACI Consortium, Italy

²DISTEBA, Università del Salento, Lecce

Giacinto Donvito*, Luciana Carota†, Guido Cuscela*, Giorgio Pietro Maggi*, Giuseppe La Rocca*, Mirco Mazzucato‡, Salvatore My*, Giovanni Selvaggi*

*INFN, Sezione di Bari, ‡INFN-CNAF, Bologna, †INFN, Sezione di Padova, ¸INFN, Sezione di Catania,

ßDipartimento Interateneo di Fisica di Bari

Gaetano Scioscia%, Pietro Leo%, Luigi Di Pace%

Graziano Pappada%, Vincenzo Quinto%, Margherita Berardi°

%IBM Italy S.p.A. - Innovation Lab, Bari

Exhicon srl, Bari

Francesco Falciano°, Andrew Emerson°, Elda Rossi°, Giovanni Lavorgna°, Andrea Vanni°

°CINECA, Bologna, DIBIT-HSR, Milano

Lisa Bartoli†, Pietro Di Lena†, Piero Fariselli†, Raffaele Fronza†, Luciano Margara†, Ludovica Montanucci†, Pier Luigi Martelli†, Ivan Rossi†, Marco Vassura†, and Rita Casadi°

*Biocomputing Group, CIRB/CIG-Department of Biology, University of Bologna and Bioinformatics Group-Department of Computer Science

Tiziana Castrignanò
CASPUR, Roma

Domenica D’EliaΩ, Giorgio GrilloΩ, Flavio LicciulliΩ, Sabino LiuniΩ, Andreas GiselΩ, Monica SantamariaΩ, Saverio VicarioΩ, Cecilia SacconeΩ

ΩITB-CNR, Bari, §Dipartimento di Biochimica e Biologia Molecolare “E. Quagliariello”, Università di Bari

Anna Anselmo&, David Horner&, Flavio Mignone&, Giulio Pavesi&, Ernesto Picardi&, Viviana Piccolo&, Matteo Re&, Federico Zambelli&, Graziano PesoleΩ&

& Dipartimento di Chimica Strutturale e Stereochemica Inorganica, Università di Milano, Ω Dipartimento di Scienze Biomolecolari e Biotecnologie, Università di Milano

&Correspondence author: giovanni.aloisio@unile.it

ABSTRACT

The LIBI project (International Laboratory of BioInformatics), which started in 2005 and will end in 2009, was initiated with the aim of setting up an advanced bioinformatics and computational biology laboratory, focusing on basic and applied research in modern biology and biotechnologies. One of the
goals of this project has been the development of a Grid Problem Solving Environment, built on top of EGEE, DEISA and SPACI infrastructures, to allow the submission and monitoring of jobs mapped to complex experiments in bioinformatics. In this work we describe the architecture of this environment and describe several case studies and related results which have been obtained using it.

**INTRODUCTION**

Modern biology, and particularly the molecular aspects of genomics, proteomics and transcriptomics, require multidisciplinary skills and the use of high-level technologies. Bioinformatics has been developed to meet these needs; it encompasses biological knowledge and expertise in advanced information technology. Public and private interests and investments have definitely boosted this discipline in many European and non-European countries. In Italy in particular many important initiatives have been funded to accelerate the development of bioinformatics. Among them, a project named LIBI (www.libi.it) (International Laboratory of BioInformatics, starting date September 2005), aims at setting up an advanced bioinformatics and computational biology laboratory, focusing on basic and applied research in modern biology and biotechnologies.

LIBI has its headquarters in Bari, Italy. It has been conceived as a bioinformatics laboratory “without walls” and carries out a number of activities, including: i) multidisciplinary collaborations to encourage interactions between researchers in public institutions and industry working in the biomedical and biotechnological field; ii) technological transfer and dissemination to promote training and provision of bioinformatic services; iii) the implementation of a high performance distributed infrastructure supporting the access to large data sets and high computational power; iv) the promotion of bioinformatics research at the (inter)national level maintaining and creating new specialized databases, developing new algorithms, developing and implementing new and more efficient processes and mechanisms for automatic bioinformatics analysis.

Even though this project features only Italian partners the idea is to open the laboratory to other international research groups in order to test and therefore improve the offered services.

To achieve a high level of transparency we designed and developed the LIBI Grid portal which represents the entry point to the LIBI platform, allowing easy and persistent access to distributed and heterogeneous data and software.

The LIBI project will terminate in 2009 and for four years more it will guarantee a virtual working space for the academic and industrial partners belonging to public and private institutions which shall provide their infrastructures, use the platform tools and equipment, and benefit from the same capacity and expertise made available by the other partners. The activities, technological platforms and applications will be integrated within the same environment, aiming at producing and increasing scientific knowledge, providing services and training as well as enhancing international competitiveness.

In this chapter a description of the project will be presented, proposing the architecture of the Grid PSE and the main services grouped in computational and data grids; finally, several case studies implemented with this infrastructure will be discussed as well.

**BACKGROUND**

A key requirement, considered during the design phase of the LIBI laboratory, has been the consideration that bioinformatics applications are naturally distributed, because experimental data and biological databases are themselves usually distributed. Also many experiments require huge computing power, owing to the large size of data sets and the complexity of processing, and may need to access heterogeneous data, where heterogeneity is multifaceted (data format, access policy, distribution, etc.),
and require a secure infrastructure to protect and secure the access to private data owned by different organizations.

The Problem Solving Environment (PSE) (Houstis, 1997) is an approach and a technology that can fulfill such bioinformatics requirements. The PSE can be used for the definition and composition of complex applications, hiding programming and configuration details to the user that can concentrate instead only on the specific biological problem. Moreover, computational grids (Foster, 1999) can be used for building geographically distributed collaborative problem solving environments and grid-aware PSEs (Laszewski, 2001) can search and use dispersed high performance computing, networking and data resources. In this work, the PSE solution has been chosen as the integration platform for bioinformatic tools and data sources.

THE LIBI - INTERNATIONAL LAB FOR BIOINFORMATICS

The main goal of this project is to set up and test an advanced bioinformatics and computational biology laboratory, focusing mainly on the following activities:
- the construction and the maintenance of general genomic, proteomic and transcriptomic databases (e.g. ENSEMBL) as well as specialized databases developed by LIBI partners (e.g. MitoRes, UTRdb, UTRsite, ASPicDB, CSTdb, etc.);
- the design and implementation of new algorithms and software for the analysis of genomes and their expression products and for the prediction of the structure of proteins.

It is a public and private distributed laboratory targeting the following requirements of the bioinformatic community:
- access to shared tools (analysis, data and displaying tools) in the form of simple or composed (workflow) services and their optimization;
- distribution of data and their transparent access (database federation);
- development of a collaborative environment allowing users to solve complex biological problems;
- the possibility to customize the user environment (content management) and manage or share/exchange personal data according to well defined authorization policies.

To satisfy these requirements the LIBI strategy was defined considering the following points: i) choice of high level and interoperable technologies based on open standards; ii) applications and services following the SOA (Service Oriented Architecture) approach; iii) priority of the end-user in defining functionalities (user driven approach); iv) easy access to the platform; v) dissemination and opening to public services.

The Grid Problem Solving Environment (Grid PSE) meets these requirements providing an integrated software platform for solving complex experiments in bioinformatics. Moreover, the Grid offers the computational power needed for solving large scale simulations, through easy-to-use interfaces that hide implementation details.

Each component (application or data) is seen as an atomic service that the user can use “as is” or compose into a graph (workflow).

In the design phase of the laboratory, these applications have been classified considering the computing needs for the following three classes of experiments:
- **High Throughput Computing (HTC)** solves problems that can be split up into many smaller tasks that are essentially independent (coarse-grained problems). They require large amounts of weakly coupled resources;
- **High Performance Computing (HPC)** solves problems that can be split up into small tasks that need to exchange information and often require synchronization (fine-grained problems). They require large amounts of highly coupled resources such as memory, CPU, disk, etc.

According to this classification, a re-engineering of the applications has been needed for their porting in the grid environment. Major details will be given in the Section “Porting Bioinformatic Applications On The Grid” which focuses on several case studies implemented within LIBI.

The partners involved in the laboratory are grouped into 8 research units (RUs), located in Italy (Fig. 1): 4 technological units, whose specific task is to design and implement the technological platform of the Laboratory and 4 scientific-bioinformatic units which, besides dictating the contents and function of the laboratory, are the main users and exploit the platform for developing their research activities.

The technological RUs and their roles in the project are:

- **SPACI**, University of Salento: it offers a HPC/HTC platform and collaborates in the porting of multiple sequence alignment (HTC), protein tertiary structure prediction (HTC) and molecular dynamics simulations (HPC), sequence pattern search (HTC) and Bayesian phylogenies (HPC) applications;

- **INFN** (Sections from Bari, Catania, Padova and CNAF-Bologna): it provides the HTC infrastructure, monitors and gives access to the INFN production GRID and to the EGEE infrastructure (http://www.eu-egee.org/) and collaborates in the porting of applications for sequence similarity search, sequence pattern search and Bayesian phylogenies;

- **IBM Italia S.p.A (IBM Innovation Lab – Bari)**: is working on the data federation platform by integrating several structured and unstructured bioinformatic data sources. This RU includes also Exhicon S.r.l which collaborates in the realization of the data federation platform, the University of Bari, Dept. of Computing Science which is involved in providing a number of text analysis services, and the University of Bari, Dept. of Biochemistry and Molecular Biology which is setting-up a specialized a human mitochondrial genomic resource based on variability studies for supporting population genetics and biomedical Research;

- **CINECA** (the Interuniversity consortium for supercomputing in North-East Italy) provides an HPC platform and collaborates in the porting of applications for the molecular dynamics simulations and Bayesian phylogenies. As a sub-contractor, **CASPUR** (the Interuniversity consortium for supercomputing applications for universities and research) offers a HPC/HTC bioinformatic platform for comparative genomic and gene expression studies. It also works with scientific RUs in developing new applications within a Service-Oriented Architecture, where application integration is facilitated by using bioinformatic web services as the building blocks.

The scientific RUs are:

- **CNRBA** (CNR Institute for Biomedical Technologies, Bari): its research activity mainly focuses on genomic and proteomic databases;

- **UNIBO** (University of Bologna) proposes data and services related to the bioinformatics characterization of the proteome. The activities include: the development of new algorithms, software and web resources for the study and characterization of human and other organism proteomes and the development of proteomic databases;
UNIMI (University of Milan): it proposes the development of new algorithms for the functional analysis and annotation of genomes, particularly focused on the identification and characterization of regulatory elements regulating gene expression at the transcriptional and post-transcriptional level;

CBMTS (Centre of Molecular Biomedicine, Trieste): proposes the development of new algorithms for the characterization of human and other organism transcriptomes and contributes to the development of transcriptomic databases.

LIBI Architecture

In practical terms, the LIBI project aims to set-up a virtual bioinformatics research workplace by providing a high performance infrastructure consisting of a grid computing facility that includes a number of data and computational nodes (both HPC and standard nodes) as well as bioinformatic application services and resources. Figure 2 provides a high-level description of the LIBI functional environment. The architecture of the system (Figure 3) is Service-Oriented (SO), and is built on a network infrastructure for the exchange of information.

On the top layer there are the application services that can be run separately or composed into a graph by means of a workflow.

Applications in the Laboratory include, for example:
- **BLAST** (Basic Local Alignment and Search Tool) and **PSI-BLAST** (Position-Specific Iterated Blast) (Altschul, 1997): both programs carry out the similarity searches in a data bank, with the latter using a position-specific scoring matrix in the search; these tools are both HTC;
- **PatSearch** (Grillo, 2003) searches complex sequence motifs (i.e. any assortment of regular expression patterns, secondary structures and position weight matrices) and is HTC;
- **DNAFan** (Gisel, 2004) searches specific genomic regions (DNA) defined by a Feature ANalyser where specific patterns can be matched by using PatSearch, hence it is HTC.
- **FT-COMAR** (Vassura, 2007; Vassura, 2008) performs the **reconstruction of 3D protein structure** from its contact map. It is HTC;
- **AntiHunter** (Lavorgna, 2005): tool for the identification of expressed sequence tag (EST) antisense transcripts from BLAST output. It is HPC;
- **GROMACS** (Lindahl, 2001) / **NAMD** (James, 2005) molecular dynamics software. It is HPC;
- **MrBayes** (Ronquist and Huelsenbeck, 2003) performs Bayesian inference of phylogeny and is HPC/HTC;
- **CSTminer** (Castrignanò, 2004) identifies conserved tracts in the comparison of two sequences and classifies them as coding (likely belonging to protein coding genes) or non-coding (potential regulatory regions).

These are described in detail in the “Porting bioinformatic applications on the Grid” section in several case studies deployed in the Laboratory.

Access to the applications is obtained via high level **Resource Management** and **Data Management** services.
The Resource Management service contains the execution logic of the applications and exploits the so-called “Meta Scheduler” which is based on several highly efficient C libraries implementing functionality for job submission, job monitoring, file transfer and management of users’ credentials (Aloisio, 2007). These include technologies for the development of Web Services based PSE through the use of grid portals that contain the logic for accessing the distributed system by means of easy-to-use interfaces such as the Web. Web Service technology and the PSE allow a secure and ubiquitous access to computing resources and implemented services. The Meta Scheduler is built on top of basic services offered by different grid middlewares such as gLite for HTC applications, Unicore for HPC and Globus for HTC/HPC applications. These grid middlewares are described in the “The LIBI Resources” section. The functionalities provided by the Resource Management service include scheduling and monitoring of the user requests, the optimal allocation of resources in a distributed environment, and so on.

The Data Management service incorporates the grid logic for advanced data management, allowing a transparent and dynamic access to heterogeneous and geographically distributed data. Such functionalities are offered within the LIBI platform by the GReIC Data Access Service (DAS) (Aloisio, 2005a). This grid service includes in addition secure mechanisms for data transfer on the Grid using advanced protocols to guarantee a high level of efficiency. The Security component handles the management and access to both services and resources within the LIBI distributed environment. It provides mechanisms for authentication, authorization, data protection, etc.

THE GRID IMPLEMENTATION OF THE VIRTUAL LAB

This section presents the Resource and Data Management services implemented for these platforms. For the Data Management service, the implementation of the Federated DB is also described.

Resource Management: interoperability in the grid
One of the key aspects in life science applications is the execution of a high number of jobs for large-scale analysis. The Grid is a powerful tool for solving this kind of problem, but in a distributed environment, bioinformatic applications need to be re-engineered in order to provide better performance. In addition, a tool for orchestrating the execution of different kinds of applications and the access to the needed data is fundamental, because different and heterogeneous grid infrastructures have to coexist. The interoperability issue must be addressed analyzing and taking into account the differences between the adopted approaches in the following areas: job submission and monitoring, file transfer, authentication and authorization, and information services.

The solution adopted within the LIBI infrastructure consists of the development of an easily extensible Meta Scheduler able to support different middleware. The development leverages on the GRB technology, developed by the University of Salento, Lecce & the SPACI Consortium, Italy and includes a set of grid libraries, the Meta Scheduler and the LIBI grid portal. The LIBI grid portal provides an integrated approach to grid resource management, through a user-friendly web GUI and back-end GSI-enabled Meta scheduler Web Service, developed using gSOAP (Van Engelen, 2003) and our GSI plugin for gSOAP (Aloisio, 2003). The Meta Scheduler provides the following features:

- it can be user-centric or VO-centric. Each user is allowed to create his or her own personal grid on-the-fly by adding, removing or updating his or her computer configuration. Alternatively, the Meta Scheduler allows a user to be associated to a GRB managed VO; in this case the user will automatically benefit from all the computational resources belonging to that VO;
- it allows the use of multiple credentials (and thus the usage of multiple machines belonging to different VO) in the same job submission;
- it provides the automatic, transparent refresh of job credentials to support long running jobs;
- it notifies automatically the user via email each time the status of a job changes;
- it is based on the Open Grid Forum (OGF) Job Submission Description Language, vastly improved to support the addition of VO description, credentials utilized, software installed on remote machines, parameter sweep and workflow job submissions;
- workflow description based on WF management coalition WorkFlow Description Language (WFDL); the Meta Scheduler is currently partially compliant with WFDL.

The Resource Management service includes a workflow management system for supporting complex “in silico” experiments. It has an editor, implemented in the Java programming language, which allows the definition of an experiment using a graphical formalism for modeling the experiment and describing it in detail.

In a first version we chose UML (Unified Modeling Language) since this provides a visual modeling notation valuable for designing and understanding complex systems. It has been extended for supporting specific grid requirements such as data stage-in or stage-out, parameter sweep jobs and so on. The application components are chosen in the editor and the constructed graph is checked before execution by using rules derived from metadata and ontology (Aloisio, 2005b).

We classified workflow components as data banks, bioinformatic algorithms, graphics tools and input data types. Such a classification is done with a simple ontology written in DAML+OIL.

The workflow engine is embedded in the Meta Scheduler (Cafaro, 2008). The Meta Scheduler supports batch job submission, supporting X-Windows applications, parameter sweep jobs and workflow jobs. Resource brokering is done using the resource related information gathered from the iGrid Information Service (developed within the GridLab Project), but Meta Scheduler supports also LDAP based information services such as Globus-MDS and gLite BDII.
Within the LIBI project the GRB framework has been extensively extended in order to support the interoperability among different middlewares, particularly Globus, gLite and Unicore. The design of the Meta Scheduler was driven mainly by the definition of a modular architecture which can be extended easily by a plug-in based approach. In order for the Meta Scheduler to support different grid middlewares the specific “adaptor” for the target middleware must be developed and plugged in the Meta Scheduler architecture. The adaptor must include the mechanisms, specific for the target grid middleware, to address job submission and monitoring, authentication and authorization, file transfer and the gathering of resource related information. The Meta Scheduler is then able to use gLite grid resources through its “grb_gLite adaptor”. Figure 4 illustrates the interchange mechanism between gLite Workload Management System (WMS) and Meta Scheduler. The Meta Scheduler can access the WMS using the user’s credentials with the VOMS extension obtained from the VOMS server. The job is described according to the JDL language and submitted to the WMProxy web service using the GSI enabled web service interface; the job monitoring is performed contacting the Logging and Bookkeeping (LB) web service while the staging in and out of files is explicitly done via GridFTP contacting the appropriate Storage (SE).

In addition to the standard case, for those applications which require the submission to the gLite grid of a large number of jobs, the Meta Scheduler uses a tool developed by the INFN- Bari section for monitoring and bookkeeping called the Job Submission Tool (JST). A description of the tool can be found later in this book.

Within the LIBI project the “grb_unicore adaptor” has also been developed, in order to support a Unicore based grid infrastructure. The Meta Scheduler can contact the Network Job Supervisor (NJS) service, through the Unicore Gateway service, by using the user’s credentials through a GSI connection in order to submit batch jobs. The job is submitted to the Gateway using the AJO description file format; the Gateway exposes also functionalities for job monitoring and output retrieval; the file transfer mechanism is performed through a GridFTP connection with the remote NJS.

![Figure 4: Meta Scheduler support for gLite WMS](image)

**Data Management: federated approaches in the grid**
One of the general challenges in life science research is the heterogeneous nature of both bioinformatic data sources and the nature of the content hosted in them. It is not uncommon to find in every laboratory or even for a specific life science project the adoption of a variety of data source formats, such as relational databases (such as Oracle, DB2, and MySQL) mixed with non-relational sources. Such non-relational sources include XML documents, spreadsheets, ASCII text files and BLAST/FAST/HMMR files. Additionally, independently of the data source format, data sets may include various kinds of structured and unstructured data, such as collections of texts, notes, images, etc. The ability to manage, integrate and analyse related structured and unstructured information, possibly hosted in heterogeneous data source repositories, is fundamental for accelerating bioinformatics research.

Two key elements that must be addressed by the data management framework concern:

1. the need to federate several and heterogeneous data sources within the LIBI context;
2. the need to securely and transparently access the federated LIBI data bank in the grid.

In the following sections we describe the LIBI solution to data management, access and integration of biological data sources.

Data Federation issues in the bioinformatics domain

Within the LIBI platform both bioinformatic databases and algorithms that access database data can coexist. In such an environment an extensive integration of data and information is highly desirable, both from the point of view of information technicians (developers), who would be able to manage a repository with just a single interface, and for final users, who would benefit from this integration to perform, in a much simpler way, powerful queries that combine a number of data sources. According to the LIBI strategy, there are two main approaches to obtain data integration: i) data federation, and ii) a data-warehouse (consolidation) of various data sources.

In the first case, federation is the ability to transparently access relevant information from a variety of sources and platforms as though it were a single resource. A federated approach to both data and information integration provides the ability to synchronize distributed data without requiring that it be moved to a central repository, as shown in Figure 5.

On the other hand, consolidating data into a single physical data store has been, for many problems, the best way to achieve fast, highly available, and integrated access to related information. The drawbacks of this solution include: latency, data aging, and lack of a unique access interface to heterogeneous data.

For the LIBI platform and, in general, for the class of IT bioinformatics problems this platform is devoted to, the solution we designed and experimented is a hybrid approach: in our case it seems to give the best results.
In the following section we provide additional details about the LIBI data federation server we implemented which is based on the IBM WebSphere Information Integrator (Alur, 2005). This kind of data federation uses SQL as the only language to access all data sources. This enables all data sources to be accessed in a standardized format, whether they are an application program, tool or program product. Such architectures can also be efficiently implemented in a grid architecture.

**The LIBI data federator server**

Figure 6 depicts the realization of the architecture in Figure 5 in the LIBI environment. This picture shows the four wrappers which have been used so far in the LIBI data federation architecture:

1. the Relational Wrapper allows the access of relational databases of some LIBI partners such as MitoRes, UTRsite, UTRdb and HmtDB;
2. the Entrez Wrapper is the gateway for accessing the well-known remote databases hosted by the NCBI: e.g. PubMed, GenBank, OMIM;
3. the EMBL/FASTA Wrapper (Leo, 2008), developed within the LIBI project by using the Information Integrator SDK, allows data sources stored in EMBL format flat-files to be federated. It has been used to access local replicas of the EMBL, UniProt and EMBL CDS data sources;
4. finally, the Web services Wrapper has been applied with the aim of integrating the remote database Species 2000.
A simplified environment to data mash-up, query and discover data from the structured world

The basic interface provided by the LIBI federator server is SQL, thus directly targeted by specialized bioinformatic applications. In order to allow also non-technical people access and directly benefit from the federated database layer a generic system to retrieve information as well as to allow data mash-up is also provided. To do this a new, specialized tool has been introduced in the LIBI platform: the IBM Data Discovery and Query Builder for Healthcare and Life Sciences (DDQB) (DDQB, 2006). It is a powerful search capability tool with a graphical interface that enables users with various levels of expertise to easily configure queries and leverage the full spectrum of information assets.

Figure 7 shows the operation flow of DDQB. The key concepts of the tool are the Data Abstraction Model (DAM) and the Query Abstraction Model (QAM). The former consists of an XML repository that stores all the information concerning the DB physical model, the abstract entities the end-user can handle, and their mapping (this logical to physical mapping is designed just once by the db administrator). Most DDQB behaviour is driven by this component. On the other hand the Query Abstraction Model provides a XML representation of an abstract query in terms of logical fields, and is formulated by considering both the authorization level of the user (with a granularity at the field level) and the selection logic imposed by the query criteria. The Transformation and Mapping block, in Figure 6, is the component devoted to concrete query translation and execution.

The end user perspective and a portion of the DDQB interface are shown in Figure 8. End-users construct queries by operating on DAM and QAM in a graphical way by navigating and acting on a generic taxonomy pointing to the LIBI federated database biological concepts. Figure 8 reports a portion of the DAM the user works with (on the left hand side) and a step of the query building phase. The end-user, in this case, is asking with one query for H. Sapiens DNA sequences, extracted from GenBank, for which the annotation field includes the globin term and the abstracts of the PubMed reference linked to these sequences include the anemia term.
Grid data Access to LIBI federator server

In the LIBI project, the grid access to the LIBI Federated DB is carried out through the **GRelC Data Access Service** (GRelC DAS) (Aloisio, 2005a). The two layers (GRelC DAS and LIBI federator server) are complementary and work at different levels: while the LIBI federator server performs the data federation task (discussed in detail in the previous Section), the GRelC DAS exposes the federated DB in the grid making it available to all the LIBI grid users (see Figure 9).

The GRelC DAS is a GSI/VOMS enabled web service providing extreme performance, interoperability and security. It efficiently and transparently manages databases on the grid across VOs, complying with emerging and consolidated grid standards (W3C and OASIS) and specifications (OGF) as well as production grid middleware (gLite & Globus). It currently provides a uniform grid access interface to a wide range of relational (for instance IBM DB2, on which the LIBI federator server is based, as well as MySQL, Oracle, PostgreSQL, SQLite, Microsoft SQL Server, etc.) and non-relational data sources (XML DB engines such as eXist, Xindice, etc.).

**Figure 7**: Data Discovery & Query Builder operation flow.
**Figure 8:** An example of Data Abstraction Model (on the left) and the query building interface (on the right).

**Figure 9:** GRelC DAS and DB2 Federated DB.
THE LIBI RESOURCES

In this section, the grid resources involved in the Laboratory are described.

The SPACI grid

The Italian Southern Partnership for Advanced Computational Infrastructures (SPACI) is a consortium consisting of the University of Salento, Lecce, the University of Calabria, Hewlett-Packard Italy and an ICT company based on spinoff companies belonging to the universities involved in the consortium. The consortium is the follow-up of the SPACI project, a partnership between the University of Calabria, the University of Lecce and the University of Naples “Federico II”, based on three geographically distributed High Performance Computing centers located in southern Italy, namely, the MIUR/HPC Center of Excellence of University of Calabria, the CACT (Center for Advanced Computational Technologies) of University of Salento and the Naples/DMA (Dept. of Mathematics and Applications) of University of Naples “Federico II”.

The SPACI Grid comprises more than two hundred dual processor nodes (Itanium2-based HP Integrity) running Linux, two clusters Compaq AlphaServer SC Tru64/Unix (4+16 nodes), plus services and applications.

SPACI is also an associate center of the Euromediterranean Center for Climate Change (CMCC) which provides more than thirty Tflops of aggregated computing power. The middleware deployed includes Globus and gLite and it supports HPC/HTC applications.

The EGEE grid Infrastructure

The LIBI High Throughput applications, which can be decomposed into many smaller independent tasks, can exploit the unique features of the EGEE Grid infrastructure (http://www.eu-egee.org/). The EGEE grid is in fact the largest multi-disciplinary grid infrastructure in Europe exploiting the gLite middleware (http://glite.web.cern.ch/glite/), which brings together more than 140 institutions to produce a reliable and scalable computing resource available to the European and global research community. At present, it consists of approximately 300 sites in 50 countries and gives its roughly 10,000 users access to 80,000 CPU cores around-the-clock. Of these resources, more than 17 thousand CPUs are accessible to the bioinformatic community using the “biomed” VO. The LIBI partners contribute to the EGEE infrastructure with a) the INFN Production Grid (http://grid-it.cnaf.infn.it/) including more than 20 sites with a total of 4500 CPUs, of which up to 1,500 CPUs and 5 TB of disk space can be used by the Italian bioinformatic community using the “bio” or “libi” VO; b) the ITB-Bari farm with 20 CPUs and c) the SPACI-LECCE farm with 30 CPUs and 0.5 TB of disk space. GridICE (Aifitimiei, 2007), a grid-monitoring tool developed by INFN, was used to monitor the infrastructure and its usage. GridICE provides a complete monitoring of the grid resources giving users the possibility of retrieving data according to their “Grid role”, such as VO managers, site administrators or standard users, providing a secure and authenticated direct access to the data which interest them most.

The INFN Certification Authority (INFN CA) issues the user certificates necessary to access the grid in collaboration with the local Registration Authority (RA). Two of the local RAs, those of ITB-Bari and IBM, have been set up explicitly for the LIBI project.
The DEISA grid

DEISA is a consortium of leading national supercomputing centres that currently deploys and operates a persistent, production quality, distributed supercomputing environment with continental scope. The purpose of this infrastructure is to enable scientific discovery across a broad spectrum of science and technology, by enhancing and reinforcing European capabilities in the area of high performance computing. This becomes possible through a deep integration of existing national high-end platforms, tightly coupled by a dedicated network and supported by innovative system and grid software.

It is difficult to quote the computing power of this network because the clusters involved are being constantly updated but at the beginning of 2007 the aggregated computing power of the DEISA supercomputers was estimated to be equivalent to approximately 35,000 CPUs and 270 Tflops. Each DEISA site is interconnected by a dedicated 10 Gb/s network which is used to mount a Global Parallel Filesystem. User data stored on the logical DEISA filesystem can be accessed anywhere in the DEISA network.

DEISA provides users with many facilities for job and data management and also allows the construction and submission of workflows via Unicore. The utilization of the DEISA infrastructure is facilitated by the so-called Common Production Environment which allows the user to deploy software modules to customize the working environment and launch applications in an identical way regardless of site location.

Currently the LIBI project can take advantage of the CINECA resources in DEISA, which amounts to about 30 Tflops of processing power. Further resources are available by applying through CINECA for the DEISA Extreme Computing Initiative (DECI) projects.

A summary of the resources involved in the project is reported in the Table 1.

<table>
<thead>
<tr>
<th>Infrastructure</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACI</td>
<td>More than two hundred bi-processor nodes (Itanium2-based HP Integrity)</td>
</tr>
<tr>
<td>EGEE</td>
<td>More than 17,000 CPUs (biomed VO); more than 1,500 CPUs (bio and libi VOs)</td>
</tr>
<tr>
<td>DEISA</td>
<td>Aggregated computing power is given by 35,000 CPUs and 270 TFlop/s</td>
</tr>
</tbody>
</table>

PORTING BIOINFORMATIC APPLICATIONS ON THE GRID

This section summarises a number of case studies that have been performed with LIBI. They were carried out not only to test the platform design itself but also to experiment the environment in a number of real bioinformatics application scenarios. Involved applications in the case studies are available, by using the LIBI platform, at the following url: http://www.libi.it/biotools.

CASE STUDY 1: “Genome wide identification of conserved coding and non-coding regions”
One of the major tasks of genomic analysis is the identification of genes and regulatory elements of gene expression. Among the different methods used to solve this task, comparative analysis has shown to be both efficient and reliable.

CSTminer (Castrignanò, 2004) is a comparative analysis tool that compares two sequences to identify conserved tracts and classifies them as coding (likely belonging to protein coding genes) or non-coding (potential regulatory regions). This classification is based upon the computation of a coding potential score (CPS) that takes into account the evolutionary dynamics of coding sequences.

Unfortunately given the size of the genomes (Giga base pairs – Gbp – in higher eukaryotes) it becomes difficult to run genome scale analysis.

For this reason we exploited the possibility to use a grid based approach performing human-mouse whole genome comparison.

**Method**

We split genomes in 100Kbp long slices and compared all human slices against the mouse ones. Clearly the number of comparisons needed to complete the analysis is considerable but this kind of problem fits very well the distributed approach offered by the EGEE grid platform. To execute on the grid the huge number of jobs required by this application we used the JST tool described previously.

In order to optimize the ratio between the time needed to exchange data with the work and the time required to perform the analysis we ran groups of 1,000 comparisons (a “task”) for each job. This also provides something similar to a checkpoint: whenever a job crashes only a small part of the calculation is lost.

The procedure required to run the analysis is controlled by a shell script that performs the following operations: it downloads the input sequences, decompresses them, starts the executable with the correct parameters, checks if the output was correct and, in case of success, stores the output on a Storage Element randomly chosen from a list of alternatives (granting the possibility to store the output also if some of the nodes are not working at that moment and contributing to reduce the load on each machine).

Because of the high number of jobs needed to complete all the comparisons, it is difficult track every job that must be submitted or that are already completed. Moreover in a distributed environment it is foreseen to experience failures that must be tracked in order to detect and resubmit failed jobs.

To handle this problem we setup a RDBMS procedure to submit each task, and retrieve results. By using this approach it is also possible to save information about the Storage Element used to store the output file and to monitor the status of each tasks. It is also possible to save information related to grid usage (like the job that is running or that has executed that task).

**Results**

From the CSTminer analysis we obtained a collection of nearly 1,500,000 CSTs with an average length of 190 nt. More than 400,000 CSTs were labelled as coding, 550,000 CSTs as non-coding, while 500,000 CSTs remained undefined as the CPS fell in a twilight zone between the coding and the non-coding thresholds used for the correct classification. Finally nearly 20,000 CSTs displayed more than 95% of identity and were labeled as ultra-conserved (Figure 10).
CASE STUDY 2: “Identification of regulatory elements in mRNA untranslated regions”

The detection of regulatory elements controlling gene expression at the transcriptional and post-transcriptional level, generally embedded in the non-coding part of the genome, represents a major challenge in post-genomic biology. Transcriptional or post-transcriptional control of gene expression generally involves short DNA or RNA tracts, respectively interacting with transcription factors or RNA-binding proteins.

A large number of oligonucleotide patterns involved in transcriptional and post-transcriptional regulation have been experimentally characterized and this information collected in a number of specialized databases such as UTRsite (Mignone, 2005) for UTR-specific functional motifs. UTRsite is a collection of regulatory elements located in 5’ and 3’ untranslated sequences (UTRs) whose function and structure have been experimentally determined and annotated in UTRdb (Mignone, 2005), a curated database of 5’ and 3’ UTRs of eukaryotic mRNAs, derived from several sources of primary data.

Thus, it is of utmost importance to develop specific software tools that are able to identify these functional elements in genomic or cDNA sequences in order to aid their functional characterization.

Method

The PatSearch (Grillo, 2003) software belongs to this family and is currently used to identify the presence of UTRsite patterns in UTRdb entries for their automatic annotation and localisation. It can be used as well for the annotation of newly produced or previously unannotated sequences for UTRsite or other user-defined patterns.

PatSearch is a flexible and fast pattern matcher able to search user submitted sequences for any combination of Position Weight Matrices (PWMs), primary sequence patterns and structural motifs also allowing mismatches and/or mispairings below a user fixed threshold.
It is also a pattern-matching tool able to find a well-defined pattern against sequences or complete primary or specialized biosequence databases.

The pattern description has been inspired by ‘regular expression’ rules, although both the syntax and the semantics are different, especially for the inclusion of specific operators for finding PWMs, complementary helices and palindromes.

A pattern is made by a combination of pattern units, defined in syntax rules as: String pattern; Range pattern; Palindrome pattern; Hairpin loop pattern; Position weight matrix (PWM) pattern; Repeat pattern; Length constraints.

The program is also able to do a post-processing of the already matched pattern in order to refine the search. The computational complexity of the PatSearch software is strictly influenced by the structure of pattern composition, since there are some pattern units of syntax which require a backtracking process and a massive comparison; this affects the whole computation process heavily.

Grid application

To detect the regulatory motifs in the UTRdb sequence collection, each UTRsite element is defined using the syntax in a format suitable for PatSearch analysis. Generally, the motifs are defined by several and complex pattern units which cannot be separately processed because they are closely inter-dependent.

To accomplish the annotation of these regulatory motifs in the UTRdb database (actually about 1,300,000 sequences) it is necessary to run a PatSearch analysis against each UTR sequence.

A typical high throughput computing approach in a grid infrastructure represents an effective solution to this huge analytical effort. In fact it is advisable to split the UTRdb database into separated subsets in order to run different PatSearch processes against them.

A typical architecture for this task involves a massive data transfer among the main server (that submits the jobs on grid), the data repository and the computational nodes of the grid.

A solution to this bottleneck is solved by using GRelC, a database middleware for the grid. Using this layer is possible to access the data partition directly from the grid node avoiding transfers between main and data servers.

In our test case we have successfully used GRelC with the database federator (IBM Information Integrator) to access the UTRdb.

The porting of this strategy solution on the grid infrastructure has yielded a drastic reduction of the total CPU time through the distribution of processes and database subsets on the computational grid nodes.

CASE STUDY 3: “FT-COMAR: a large scale application for reconstructing protein folding”

To perform their biological function most proteins fold into unique three-dimensional (3D) structures. The prediction of the 3D protein structure from its residue sequence is one of the most important and still unsolved problems in bioinformatics (Lesk, 2006). A promising approach to this challenging problem is based on the prediction of the protein contact map, a reduced in plane 2D representation of the 3D structure, a sort of intermediate step between the primary and tertiary structure. In this representation, two residues are in contact if their distance is lower than a given threshold, usually 1.2 nm in the protein space, when the Calpha trace of the backbone is adopted. A contact map is a binary symmetric matrix containing 1 for each pair of residues in contact, and 0 otherwise. The problem of predicting the protein folding from its residue sequence may be sub-divided into two steps: 1) the prediction of the contact map, 2) the reconstruction of the 3D structure from the contact map. Currently several efforts are addressing
the problem of predicting the protein contact map but the methods developed so far are still far from satisfactory. In this large scale effort, we address the second step of the protein folding prediction problem: the reconstruction of the 3D protein structure from its contact map. The general version of this problem has been proved to be NP-hard (Breu et al., 1998). Various heuristic algorithms have been proposed to solve the problem (Galaktionov, 1994; Vendruscolo, 1997; Bohr, 1993; Pollastri, 2006) but most of these methods have been tested only on the relative predictions and none of them are available to the scientific community. We have designed and implemented a new heuristic algorithm, called FT-COMAR, which can reconstruct the 3D structure of a protein from its contact map and which outperforms all existing algorithms (Vassura, 2007 a, b; Vassura, 2008). Grid technology is used to provide the processing power required to cope with the enormous computational effort that results when the effect of noise on the protein reconstruction is analyzed.

The method: How to reconstruct the protein 3D structure from the contact map

In any prediction problem usually a blurred putative solution is given and this is the case when the prediction of the contact map is considered. The algorithm consists of two phases. The first phase allows the prediction of a partially random set of starting coordinates. This initial phase relies essentially on statistical values of inter-residues distances which are randomized and then used to generate a starting set of 3D coordinates through the metric matrix embedding algorithm (Havel, 1998). The second phase of the algorithm iteratively applies a correction/perturbation procedure to the randomly generated set of coordinates. The correction procedure moves each pair of residues whose relative positions are not satisfying the corresponding contact map values according to the contact map value itself. The perturbation procedure slightly moves residues improving their mobility for the next correction. This iteration is performed in order to obtain a new set of coordinates as consistent as possible with the given contact map. The refinement applies until the set of coordinates is consistent with the given contact map or until a stopping condition, based on a control parameter, is reached.

The “in GRID” extension

FT-COMAR has been previously tested on a data set of 1,760 exact contact maps (Vassura, 2007a; Vassura, 2008) and on a subset of 120 erroneous contact maps (Vassura, 2007b). However a statistical validation should use a much larger data set. To extend our results to a consistent set of proteins, we downloaded all the protein structures from the Protein Data Bank and we retained only those files with coordinates obtained from X-ray experiments, with resolution <2.5 Å, and without missed internal residues. Using BLAST (Altschul et al., 1997) we removed sequence redundancies. We thus obtained a dataset of 3,613 proteins with sequence similarity lower than 30% and with protein size ranging from 39 to 1,392 residues. For each protein we computed its contact map with a fixed threshold set at 12 Å and, for each percentage of error, we generated 100 random noisy contact maps. Each of these maps was then reconstructed with FT-COMAR. The same experiment was replicated for random errors, random missing contacts, random added contacts, random unknown entries, random unknown entries together with random errors, and random errors with a pre-filtering procedure. The total number of running jobs was 8,467,577 and the experiment was run on the SPACI grid. The parallel time is shown in Figure 11 as a function of the number of processors.

The parallel time (Tp) is given by sequential time (Ts) on processors number (nproc) i.e. Tp = Ts/nproc, so the Tp values are showed in the Table 2.
Figure 11. Estimated computing time in days for 100 runs of FT-COMAR for each type of test on 3,613 proteins as function of the number of processors used.

Table 2: Parallel Time (in days).

<table>
<thead>
<tr>
<th>Nproc</th>
<th>Tparallel (Days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>26</td>
</tr>
<tr>
<td>512</td>
<td>6.50</td>
</tr>
<tr>
<td>1024</td>
<td>3.25</td>
</tr>
</tbody>
</table>

Results

We have shown on a small protein set that FT-COMAR reconstruction is more tolerant to missing contacts than to added contacts in the contact map (Vassura, 2007b). We also introduced a pre-filtering procedure on noisy contact maps based on common neighbors. The filter is based on statistical values of a set of 120 proteins (Vassura, 2007b): the contact map values are not considered if the number of common neighbors between two non-contacting residues is above a given threshold or if the number of common neighbors between two contacting residues is below a given threshold. As shown in Figure 12, when a large testing set is considered (3,613 proteins) the introduction of this filter still improves reconstruction quality for percentages of random errors ranging between 1% and 10%. All together these results indicate that the robustness of FT-COMAR in reconstructing 3D protein structure even in the presence of random errors is good enough to be consistent when a large set of non redundant examples is tested.
Large scale genome sequencing projects provide biological data that are available for addressing basic biological problems at the molecular level. An exciting problem that emerges as a consequence of the availability of complete genome sequences is the comparison of all the available putative/annotated proteins in order to highlight basic aspects of life at the molecular level (Marsden, 2007; Bateman, 2006; Lee, 2007). The enormous computational effort in performing such an experiment is clear when the size of the dataset is considered. A total of 599 complete genomes, comprising 551 Prokaryotes and 48 Eukaryotes, can be downloaded from public databases. Data were extracted from NCBI, National Center for Biotechnology Information (http://www.ncbi.nlm.nih.gov/) and from Ensembl genome browser (http://www.ensembl.org/) updated on 27/07/2007. The overall database consists of 2,624,555 protein sequences in FASTA format with the goal of comparing each of the protein sequences to all the others in the dataset. 

Method

Proteome comparison was performed on the EGEE Grid infrastructure through the BLAST program (Altshul, 1990) (http://www.ncbi.nlm.nih.gov/blast/, version 2.2.15) which aligns biological sequences to find regions of similarity; in our case each protein sequence was aligned against all the others. For each sequence, BLAST gives an output file that contains all the sequences in the dataset that have a match with the query and scores these matches on the basis of statistical indices. The E-value accounts for the number of alignments with scores better or equal than the actual expected to occur by chance in a database search. The lower is the E value, the more reliable is the alignment; we fixed a 10^{-10} threshold value. We also fixed the effective length of the database to 100,000 sequences in order to make the BLAST searches independent from the database size. The input and output files were stored and replicated on two different storage devices (Storage Elements, SE) in order to balance the data traffic on the grid and to have a robust fault tolerance. We developed an automated job submission procedure that used 4 different Resource
Brokers (RB), distributed all around Europe. The overall production run consisted of approximately 2.6 million pairwise comparisons that were grouped in independent tasks of about 500 comparisons each. Even in this case the jobs were submitted to the grid using the JST tool. The complete comparison of all the protein sequences in the dataset, carried out on the grid, took about 1 week, which should be compared with the about 130 weeks estimated for the run of the same job performed on a single medium power CPU. The duration of the overall task was optimized employing the result of a previous study of the grid job dynamics on a sample of BLAST comparisons (Carota, 2008).

The fast and high-throughput comparison performed through the grid middleware, allowed us to fine tune our analysis and all the 2,624,555 outputs obtained from the BLAST all-against-all comparison were then processed. We searched for clusters with at least one sequence from each genome, to assess the most conserved domain through different species. We developed a clustering algorithm based on a 40% sequence identity threshold. Indeed, several analyses have suggested that a 40% pairwise sequence identity can be used as a significant threshold for grouping sequences in order to obtain common functional and structural biological relevant information. The second constraint used was a 90% coverage. Given two sequences of a BLAST hit, we defined the coverage as: COV=I/U, where I is the length of the intersection of the aligned regions on the two sequences and U is the union of the length of the alignment, namely the length of the two sequences minus their intersection. Both constraints were adopted to detect groups of related protein sequences. BLAST outputs were represented by means of a graph structure. The nodes are the protein sequences and two nodes are connected by an edge if the corresponding proteins share a BLAST hit with the constraints on sequence identity and coverage. We thus computed the connected components of the graph, where every connected component consists of proteins that are connected to each other by a BLAST match in a direct or indirect way.

**Results**

After clustering, we ended up with 187,594 clusters, containing from 2 up to 15,875 protein sequences (see Figure 13).

![Figure 13. Distribution of the number of elements across all the clusters. Only clusters with less than 200 elements are represented: they cover 99% of the overall number of clusters (186,482 over 187,594 clusters). 1,112 clusters contain more than 200 sequences and are not represented in the above distribution.](image)

Among the clusters, 122,686 contain only prokaryotic organisms while 63,230 are specific for eukaryotes. We focused on the most populated cluster (15,875 sequences) that is also highly specific for prokaryotic organisms, since it has at least one representative sequence for each of them. It turns out that the ABC
transporter protein ATP-binding domain (Figure 14) belonging to the ATP-Binding Cassette (ABC) superfamily is the common protein structure. This protein sequence analysis may help in further speculation on possible original functions at the basis of molecular evolution.

Figure 14. Visualization of an ABC transporter protein structure (PDB ID 1l7v). The darkest chain is the conserved domain represented in the most populated cluster.

CASE STUDY 5: “Analysis of the distribution of introns in mtDNA genes of Ascomycota”

The case-study presented here highlights how a new query approach developed within the LIBI database federation system integrating public and private molecular and taxonomic databases allows a data retrieval for specific information usually present in the entries of a biological database, but which is hard to extract using the classical query systems. The analysis is based essentially on the use of GenBank (the NIH genetic sequence database), an annotated primary collection of all publicly available DNA sequences (Benson, 2007).

The database query systems commonly used such as SRS (Etzold, 2003) or Entrez (http://www.ncbi.nlm.nih.gov/sites/gquery) appear rather deficient when it is necessary to retrieve selected information contained in particular lines of a database entry. This problem is particularly evident when we wish to extract the data present in the so-called Feature Tables that, in a typical entry of a primary database, contain a detailed description of the various functional portions of the sequence. The described case study demonstrated that the query system developed within the federation of biological databases produced in the LIBI platform can constitute a really effective alternative solution.

The biological case

The biological case under analysis is part of an extensive project which aims at the development of an effective and standardized species discrimination molecular system in Ascomycota fungi. Because of the peculiar and favourable features of the mitochondrial genome, mitochondrial DNA has been considered for use as a “species molecular marker” or “barcode”. However, a possible problem in the PCR amplification of mitochondrial fungi genes and in their bioinformatic surveys may result from the presence of mobile introns (Hamari et al., 2002; Seifert et al., 2007), non coding sequences which interrupt genes. The aim of our study case is to explore 11 protein coding genes and 2 rRNA coding genes.
currently available in public databases to find one or more regions free from introns in all Ascomycota and large enough to be proposed as species molecular marker candidates. The most effective and accurate way to obtain the distribution, conceived as the number and position, of the introns in each gene and inside a specific taxonomic class, is to extract this information directly from the features tables of GenBank. An approach based on the LIBI database federation query system seemed immediately desirable to reach this goal. It can effectively avoid a series of alignment and retrieval stages based on the similarity calculation, intrinsic in a Blast approach, which inevitably produce false positives and negatives in the final results.

Results

The results of our study reveal the pervasiveness of the introns across almost all the mtDNA genes, even if the density of distribution and the size can vary remarkably between them. CO1 and CYTB genes show the maximum density of introns suggesting that the COI gene, designated as a core barcode region for animals (Hebert, 2003), has to be reconsidered in fungi. The analysis suggests that at least three genes could be considered the best barcode candidates for their scarcity of introns, namely ND2, ND4 and ND6. In conclusion, the higher efficiency of the query-based system in revealing introns seemed to be evident compared with a Blast-based approach, reducing significantly false positives and negatives.

Technological aspects

This section reports some performance parameters evaluated during the activities of data retrieval of intron locations from the LIBI federated DB, in particular from its GenBank subsection. It is important to bear in mind that the queries against the GenBank subsection of the LIBI DB are propagated to the NCBI DB in real time and so they experience internet-network latency.

Performances

<table>
<thead>
<tr>
<th>Query sent against the federated LIBI DB</th>
<th>Number of queries</th>
<th>Approximate response-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries against the GenBank subsection to extract the IDs of the entries to be evaluated</td>
<td>1</td>
<td>9 min</td>
</tr>
<tr>
<td>Query against the GenBank subsection to extract features related to the previous IDs</td>
<td>7,089</td>
<td>2 h</td>
</tr>
</tbody>
</table>

CASE STUDY 6: “The Use of Bayesian Phylogenetic Inference and Complex Phylogenetic Workflow on the LIBI platform, both in a Distributed and a Parallel Environment”

MrBayes is a program for Bayesian inference of molecular evolution parameters and topologies from a set of molecular sequences. Molecular evolution is generally studied only by inference: a pattern (in this
case a set of biological sequences) is observed and different possible processes are evaluated to infer what process produced this pattern.

The inference of the values of parameters in the program is Bayesian. The program uses a markovian integration to obtain the sample from the posterior distribution of the parameters. The posterior distribution is the probabilities of the parameter values given the data and assuming that the true process is included in the model of evolution taken under consideration. The markovian integration implemented in MrBayes is a Metropolis-coupled Markov Chains Monte Carlo (MCMC) (Geyer, 1991; Altekar, 2004).

**The overview of the test cases**

**Test on the MrBayes Parallel implementation**

The MCMC implementation of the markovian integration allows MrBayes to be a natural candidate for execution in a parallel environment. The availability of a large number of CPUs on the platform allows us to deal with biological problems that require a great number of chains to adequately explore the parameter space and to achieve convergence.

The test case for this kind of implementation was simply a series of large sequence data sets that represent the nascent barcode reference database for Lepidopterans. The goal was to inform the query system of the database of the phylogenetic structure of the data in order to facilitate the species diagnosis of unknown sequences that would query the database. Because of the large number of sequences that typically make up these databases, the query system uses simplified phylogenetic inference as the overall similarity approach, neighbor-joining, or parsimony in order to inform the query (Geiser, 2004; Hibbett, 2005; Ratnasingham, 2007). The LIBI platform allowed us to test the possibility of more realistic inference, that cyclically updated, would inform the query system. The 3,523 sequences were reduced to 2,080 non redundant sequences and were subdivided in 13 groups based on a priori phylogenetic information. The 12 groups ranged from 635 to 1 with 4 groups with more than 200 taxa. All the data set had the approximately the same length. We chose a realistic but generic nucleotide model composed of a GTR substitution matrix (Lanave, 1984) and a gamma distribution to model site variability (Yang, 1994), branch lengths without molecular clock constraints and unconstrained topology.

The execution time of the 13 phylogenies are listed in Table 3 together with the number of taxa (that represent the difficulties of the problem), the number of chains and CPUs used to perform the calculation. For this experiment, the SPACI infrastructure has been employed. The last column shows the temperature used.

**Table 3.** The comparison between computational effort and execution time for the 13 phylogenetic inferences. The computational effort is function of number of taxa and number of phylogenies evaluated. The number of phylogenies is the result of the product of number of generation, runs and chains.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Nº of Taxa</th>
<th>Nº of phylogenies evaluated</th>
<th>Nº of Generation</th>
<th>Nº of runs</th>
<th>Nº of chains</th>
<th>Nº of CPUs</th>
<th>Temperature</th>
<th>Time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arsenurinae</td>
<td>77</td>
<td>8,000,000</td>
<td>1,000,000</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0.20</td>
<td>5.232</td>
</tr>
<tr>
<td>Ceratocampinae</td>
<td>150</td>
<td>24,000,000</td>
<td>2,000,000</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0.20</td>
<td>38.522</td>
</tr>
<tr>
<td>Hemileucinae</td>
<td>215</td>
<td>100,000,000</td>
<td>5,000,000</td>
<td>2</td>
<td>10</td>
<td>20</td>
<td>0.05</td>
<td>86.760</td>
</tr>
<tr>
<td>Hesperiinae</td>
<td>208</td>
<td>36,000,000</td>
<td>3,000,000</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0.20</td>
<td>35.853</td>
</tr>
<tr>
<td>Limenitidinae</td>
<td>29</td>
<td>8,000,000</td>
<td>1,000,000</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0.20</td>
<td>1.163</td>
</tr>
</tbody>
</table>
Test on the distributed MrBayes implementation and workflow

The use of a distributed version of MrBayes on the LIBI platform is more problematic given the nature of the markovian integration. Typically each MrBayes run, although starting from a fairly small input data, has a quite long execution time. This does not fit with a typical distributed application where generally each job needs to be short. We therefore decided to restrict the distribution of jobs, consisting of single complete analyses, to high performance nodes of the EGEE grid configured to accept MPI jobs. This ensured that each job would run in a sufficiently efficient manner and they would arrive at completion before the maximum run time allowed by the chosen nodes of the EGEE grid.

The case study for the distributed implementation of MrBayes, in reality involves different capabilities of the LIBI platform from the multiple database querying, blast runs, alignment and phylogenetic inference components. The goal was to evaluate the ability to discriminate among species of the 13 mitochondrial genes in vertebrates using all vertebrate mitochondrial sequences available on EMBL. We were interested to record how often the mitochondrial data were able, using realistic phylogenetic inference, to discriminate between species belonging to the same genus, and if there are differences in discrimination power among the different genes that make up the mitochondrial genome. The goal of the study was to check the validity of the choice of cytochrome oxydase for vertebrates as barcode markers by the CBOL.

The full workflow included at first a database query to collect all vertebrate mitochondrial coding sequences sorted by gene name. The query system involved first the use of blastP on UniProt and then cross-linked the results with EMBL CDS to obtain the nucleotide sequences. The performances of the queries against the federated LIBI DB are shown in the Table 4.

The data sets divided per gene were then grouped per genera (the systematic grouping that include the more related species). On the LIBI platform each genera data set was aligned as protein in muscle (Edgar, 2004) and the nucleotide alignment was built using protein alignment information as a guide. The platform transformed the fasta alignment in nexus files with a MrBayes command block and submitted them to the distributed version of MrBayes. The model of evolution chosen is the same as before: GTR substitution matrix, gamma distribution for variability among site, and unconstrained branch length and topology.

The complete results were downloaded locally and the burn-in for each analysis was determined using the routine of the coda library within the R environment. Steady state was tested on all parameters of the model and not only on likelihood scores.

The results were then summarized choosing a focal species (the one with more sequences) per genus data set and recording the number of times that the species was reported to be monophyletic in each analysis. The output of the query results and subsequent check of quality are summarized in Table 5.

<table>
<thead>
<tr>
<th>Family</th>
<th>Species</th>
<th>Total</th>
<th>Nucleotides</th>
<th>Genes</th>
<th>Trees (10)</th>
<th>Branches (20)</th>
<th>Gamma (0.03)</th>
<th>Sites (109)</th>
<th>Phylogenic Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macroglossinae</td>
<td>515</td>
<td>60,000,000</td>
<td>3,000,000</td>
<td>2</td>
<td>10</td>
<td>20</td>
<td>0.03</td>
<td>109.310</td>
<td></td>
</tr>
<tr>
<td>Pyrginae</td>
<td>635</td>
<td>60,000,000</td>
<td>3,000,000</td>
<td>2</td>
<td>10</td>
<td>20</td>
<td>0.05</td>
<td>380.600</td>
<td></td>
</tr>
<tr>
<td>Pyrrhopyginae</td>
<td>49</td>
<td>8,000,000</td>
<td>1,000,000</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>0.20</td>
<td>3.816</td>
<td></td>
</tr>
<tr>
<td>Saturniinae</td>
<td>55</td>
<td>8,000,000</td>
<td>1,000,000</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0.20</td>
<td>4.095</td>
<td></td>
</tr>
<tr>
<td>Sphinginae</td>
<td>119</td>
<td>36,000,000</td>
<td>3,000,000</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>0.20</td>
<td>25.215</td>
<td></td>
</tr>
<tr>
<td>Tortricinae</td>
<td>27</td>
<td>8,000,000</td>
<td>1,000,000</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>0.20</td>
<td>2.233</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Performances of queries sent against the federated LIBI DB
Table 5. Query result. The table shows for each gene the total sequences found, the number of correct data sets and the number of shared haplotypes across species that invalidated part of the data set.

<table>
<thead>
<tr>
<th></th>
<th>atp6</th>
<th>atp8</th>
<th>co1</th>
<th>Co2</th>
<th>co3</th>
<th>Cytb</th>
<th>nd1</th>
<th>nd2</th>
<th>nd3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequences</td>
<td>8,017</td>
<td>6,635</td>
<td>10,904</td>
<td>3,289</td>
<td>3,251</td>
<td>69,384</td>
<td>5,340</td>
<td>17,398</td>
<td>4,293</td>
</tr>
<tr>
<td>Clean Data set</td>
<td>112</td>
<td>62</td>
<td>376</td>
<td>42</td>
<td>26</td>
<td>46</td>
<td>52</td>
<td>76</td>
<td>34</td>
</tr>
<tr>
<td>same haplotype</td>
<td>37</td>
<td>48</td>
<td>26</td>
<td>9</td>
<td>3</td>
<td>238</td>
<td>2</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>cross_species</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>nd4</th>
<th>nd4L</th>
<th>nd5</th>
<th>nd6</th>
<th>Total</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequences</td>
<td>9,447</td>
<td>2,482</td>
<td>4,128</td>
<td>3,198</td>
<td>139,749</td>
<td>2,662</td>
</tr>
<tr>
<td>Clean Data set</td>
<td>164</td>
<td>-</td>
<td>54</td>
<td>34</td>
<td>1,044</td>
<td></td>
</tr>
<tr>
<td>same haplotype</td>
<td>15</td>
<td>-</td>
<td>10</td>
<td>37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cross_species</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CASE STUDY 7: “Large scale molecular dynamics on high performance platforms”

The structural and dynamic properties of proteins and other large biomolecules can be investigated experimentally by X-ray crystallography or Nuclear Magnetic Resonance. The information available by these methods is however generally limited and experimental difficulties, such as the need to crystallize or dissolve the protein for example, restrict their applicability to a subset of molecules of interest. Therefore, very often researchers use computational approaches such as molecular dynamics (MD) simulation to obtain data which is not readily available from experiment alone. But realistic MD simulations are computationally expensive, so in order to achieve the necessary performance, the program must be able to distribute the workload over many processors. In this case study, we analyse the parallel performance of two common MD programs, GROMACS (Lindahl, 2001) and NAMD (James, 2005), on two different architectures: the LINUX CLX cluster (Intel Xeon processors and Myrinet interconnects) at the CINECA supercomputer centre and the Itanium XC6000 (Intel Itanium 2 processors and Quadrics interconnects) at the SPACI consortium, University of Salento.

We have performed short simulations of a small protein, bovine β-lactoglobulin, in a solvent (water/urea + NaCl) at 300K and at constant pressure (1bar). Other common features include a time step of 1 fs, the Particle Mesh Ewald method for the electrostatics and SHAKE for constraint dynamics. We note that while it is virtually impossible to do exactly equivalent simulations between any two molecular dynamics programs, we have endeavoured to use equivalent parameters where possible.

Figure 15 shows the performance, measured as the number of picoseconds that can be simulated in 1 day, as a function of the number of processors. We see that the XC6000 is generally faster for both programs: this is probably due to the faster processors and also the faster internal network of this machine. It can
also be seen that for a small number of processors, GROMACS is faster than NAMD on both architectures. This is to be expected since GROMACS has been optimised for serial performance, particularly for Intel processors where some sections have been implemented with assembly language instructions. NAMD, on the other hand, has been designed for high parallel scalability and by the time we get to 8 processors, it is performing better than GROMACS on the CLX. But for the XC6000 GROMACS is always faster than NAMD, although the small number of processors used provides only a limited test.

![Figure 15. Performance as a function of number of processors.](image)

The often-reported superior parallel scalability of NAMD is highlighted by Figure 16 for CLX which shows the speedup, defined as the ratio of the performance for N processors with the performance for 1 processor. On CLX the NAMD parallel scaling is almost linear. GROMACS, however, scales quite poorly and reaches a maximum at 16 processors. After this limit the program actually runs more slowly with the addition of new processors, presumably because communication overheads start to dominate the overall execution time. For the XC6000, however, GROMACS scales well compared to NAMD, presumably because of the faster Quadrics network, although we repeat the number of processors utilised is quite small.

To summarise, in this study we have shown that just based on speed considerations GROMACS is the program of choice when only four or fewer processors are available, regardless of hardware architecture. For medium-sized clusters (8-32 processors), either could be chosen, depending on the characteristics of the cluster such as the internal network. Based on the CLX results, NAMD is generally more suited for simulations with higher numbers of processors.
CASE STUDY 8: “AntiHunter 3.0: enhanced sensitivity in large scale identification of antisense transcripts”

The functional role of mRNA antisense (AS) transcripts in prokaryotes and viruses has been documented in several works. Further studies in eukaryotes suggest that AS transcripts are involved in gene regulation including transcriptional interference, genomic imprinting, RNA interference, translational regulation, alternative splicing, X-inactivation, RNA editing and promoter methylation (Wagner, 2002; Lavorgna, 2004).

Methods:

AntiHunter’s goal is to identify potential AS transcripts within a genomic region of interest (Lavorgna, 2005).

AntiHunter version 3.0 web interface uses the LIBI Meta Scheduler for the submission of jobs on Unicore. The Meta Scheduler manages the needed logic for submitting and monitoring the jobs on Unicore as described in a previous section.

AntiHunter takes in input a genomic sequence and a list of annotated transcripts of the same region and some details or, alternatively, a preloaded and annotated genome stored in a resident database. In both cases a filtering of the repeated sequences has to be performed because these can falsify the results of the following BLAST step (Altschul, 1990).

The next step compares the filtered sequences against the whole dbEST database (Boguski, 1993) or a restricted selection of ESTs, using the NCBI BLASTN tool or MEGABLAST (Megablast). The resultant output is the list of the anti-sense ESTs that can interact with the genomic region given in the input. Other information, such as the length of the spanned genomic region, the length of the EST, the actual splicing sites plus some flanking sequences, etc. and a quality control check are added to the output as well. The result is available immediately when the job is finished or a link where it can be downloaded is sent by email. In any case the relevant output data are stored for a week.
The accuracy of AntiHunter was tested using genes, which had been shown previously to possess AS transcripts. Fifteen genomic regions, containing overlapping transcriptional units in mammalian genomes described previously in the literature, were used as input to the program.

Results:

The program correctly determined the presence of EST AS transcripts in 14 out of 15 cases. In the missing case, given by the human distal-less homeobox protein 1 (DLX1) gene, the presence of AS transcripts was not detected because of the lack of double-checked EST entries in the dbEST.

An excerpt from AntiHunter output is shown on Figure 17. It refers to a 96,700 bp sequence from the human chromosome 4, containing the FGF-2 gene. Several AS ESTs, are detected and the accession numbers are shown in the first column of Figure 17, corresponding to the exons of NUDT6, a known AS transcript to the FGF-2 gene (Li, 2000). Intriguingly, AntiHunter was also able to identify, despite the fact that it was used to query a human sequence, the presence of EST antisense transcripts from species other than human (i.e. Rattus norvegicus, Mus musculus and Bos taurus), unravelling the possible evolutionary conservation of the phenomenon (data not shown).

Figure 17: An AntiHunter output.

In conclusion, AntiHunter is a new tool capable of performing an in-silico search for putative EST AS transcripts. It can effectively use relatively raw, but frequently updated, material such as EST sequences and provide useful preliminary results for guiding the design of further experimental analyses.

Furthermore it has the additional advantage of providing the user with an HPC tool capable of tackling complex problems with a friendly web-based user interface.

FUTURE TRENDS

In the last year of the project, the goals of the laboratory will be to integrate the services developed in order to test the entire platform and provide a production infrastructure. Several improvements are planned in different areas: as regards the resource management, we intend to test the workflow management system in order to compose different applications and run them on different grid middlewares. In addition, we plan to develop a client library for the Web Service applications. For the data management, tools for content-based retrieval, especially those based on visual properties, will be integrated in the platform, in order to provide enhanced analysis for the produced data. Finally, other applications such as the R package will be integrated into the PSE.

BENEFITS OF THE LIBI PLATFORM AND CONCLUSIONS
This chapter describes the LIBI virtual laboratory for bioinformatics based on a high performance and distributed infrastructure supporting access to large datasets and the execution of single or complex jobs. One of the main goals of the LIBI project has been the development of a Grid Problem Solving Environment, built on top of EGEE, DEISA and SPACI infrastructures, to allow the submission and monitoring of jobs mapped to complex experiments in bioinformatics. At the same time we have developed several case studies and successfully verified the effectiveness of the Grid Problem Solving Environment in supporting experiments in a number of real world bioinformatics application scenarios. As a summary, in Table 6, we compare the main issues (from the biological and computation perspectives) that characterize each case study and highlight the solutions and benefits that the LIBI platform provided.

<table>
<thead>
<tr>
<th>Applications</th>
<th>Issue</th>
<th>Grid/HPC solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Case Study 1: CSTMiner</strong></td>
<td>Identification of genes and regulatory elements of gene expression.</td>
<td>HTC. The genomes have been split in 100Kbp long slices; all human slices are compared against the mouse ones.</td>
</tr>
<tr>
<td><strong>Case Study 2: PatSearch</strong></td>
<td>Detection of regulatory elements controlling gene expression at the transcriptional and post-transcriptional level.</td>
<td>HTC. Data partition directly from the grid node avoiding transfers between main and data servers.</td>
</tr>
<tr>
<td><strong>Case Study 3: FT-COMAR</strong></td>
<td>Reconstruction of 3D protein structure from protein contact maps</td>
<td>HTC. The protein contact map reconstructions have been split in chunks of jobs.</td>
</tr>
<tr>
<td><strong>Case Study 4: BLAST</strong></td>
<td>Comparison of 599 genomes in order to highlight basic aspects of life at the molecular level.</td>
<td>HTC. ~2,6 million of pairwise comparisons that were grouped in independent tasks of about 500 comparisons each.</td>
</tr>
<tr>
<td><strong>Case Study 5: DB Federated</strong></td>
<td>Exploring mtDNA coding sequences of all Ascomycota entries contained in GenBank in order to search for regions free from introns and large enough to be proposed as species molecular marker</td>
<td>The database federation system integrating public and private molecular and taxonomic databases developed within the LIBI platform constituted a really effective alternative solution to classical query methods.</td>
</tr>
</tbody>
</table>
candidates (barcodes).

| Case Study 6: MrBayes | Bayesian inference of molecular evolution parameters and topologies from sets of molecular sequences. | Biological problems that require a great number of chains to adequately explore the parameter space and reach convergence need the availability of a large number of CPUs. | HPC/HTC. 13 phylogenesis with a huge amount of taxa were run by using at least 8 and no more than 40 CPUs with i between 70 and 620 input sequences. |
| Case Study 7: Gromacs/NAMD | Molecular dynamics (MD) simulation for determining structural and dynamic properties of proteins. | MD simulations are computationally expensive; there is the need to distribute the workload over many processors. | HPC. Short simulations of a small protein were run on two clusters and the performance as a function of the number of processors was recorded. |
| Case Study 8: AntiHunter | AntiHunter searches for potential AntiSense ESTs within a given genomic sequence. | It performs blast search over many sequences. | HPC. The blast search can run over more than 1 processor. |

Concluding, the LIBI platform involves a large set of resources belonging to three different grid middlewares: gLite, Unicore and Globus. These toolkits provide basic services for managing the resources. Built on top of these services, a set of enhanced and novel services has been implemented related to resource and data management. For the resource management, a Meta Scheduler service has been implemented. It embeds several plug-ins for accessing different grid middlewares. The Meta Scheduler acts as an engine not only for batch, MPI and parameter sweep jobs, but also for complex jobs described by a workflow. Finally, a workflow editor has been implemented in order to compose different application runs on different middlewares. For data management, the LIBI federator server performs the data federation task enabling federated databases to access ten heterogeneous and distributed data sources. It provides a unified data-management interface (both for query and insert statements) based on SQL, whereas the GRelC DAS exposes the federated DB in grid making it available to all the LIBI grid users. It currently provides a uniform grid access interface to a wide range of relational (Mysql, Oracle, Postgresql, SQLite, Microsoft SQL Server, etc.) and non-relational data sources (XML DB engines such as eXist, Xindice, etc.). Finally, several case studies with related results have been presented.

ACKNOWLEDGEMENTS

This work was supported by the MIUR (Italian Ministry for Education, University and Research) in the LIBI “International Laboratory of BioInformatics” project, F.I.R.B. 2003, under grant RBLA039M7M.


gLite middleware from http://glite.web.cern.ch/glite/.


INFN Production Grid from http://grid-it.cnaf.infn.it/.


UTREF: untranslated regions database from RefSeq, from http://www.ba.itb.cnr.it/BIG/UTRHome/.

UTRsite: collection of functional sequence patterns located in 5' or 3' UTR sequences, from http://www2.ba.itb.cnr.it/UTRsite/.


KEY TERMS

Computational Grid - By providing scalable, secure, high-performance mechanisms for discovering and negotiating access to remote resources, the Grid promises to make it possible for scientific collaborations to share resources on an unprecedented scale, and for geographically distributed groups to work together in ways that were previously impossible (Foster, 1999).

Grid Problem Solving Environment (Grid-PSE) - A PSE is a computer system that provides all the computational facilities needed to solve a target class of problems. These features include advanced solution methods, automatic and semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware or software. By exploiting modern technologies such as interactive color graphics, powerful processors, and networks of specialized services, PSEs can track extended problem solving tasks and allow users to review them easily (Houstis, 1997). A Grid PSE is a grid-based PSE that integrates heterogeneous components into an environment providing transparent access to distributed resources, collaborative modeling and simulation, and grid portals.

Grid Workflow Management System - A system that defines, creates and manages the execution of workflows through the use of grid computing technologies, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, integrate distributed resources and legacy software modules.

Interoperability – is a property referring to the ability of diverse systems and organizations to work together (inter-operate). Mainly referred to the basic grid services like resource management, information management, data management, security

High Performance Computing (HPC) – It uses supercomputers and computer clusters to solve advanced computing problems characterized by intensive computing requirements and consist of tightly-coupled tasks.

High Throughput Computing (HTC) – It is a computer science term to describe the use of many computing resources over long periods of time to accomplish a computational problem. A typical HTC problem consists of many loosely-coupled tasks that can be executed in parallel.

Grid-DBMS – A distributed system which automatically, transparently and dynamically manages Data Resources, according to the Grid state, in order to maintain a desired performance level. It must offer an efficient, robust, intelligent, transparent, uniform access to Grid-Databases by means of a Grid Data Access Service (Grid-DAS) interface.

Grid Data Access Service – The Grid-DBMS virtualized access interface to Grid-Databases.

Data Federation – It refers to platforms able to provide virtualized federation integration of multiple disparate heterogeneous information sources enabling applications to access and integrate diverse data and content sources as if they were a single resource, regardless of where the information resides, while retaining the autonomy and integrity of the data and content sources.

Grid Resource Management – It refers to all of the actions to be considered in order to efficiently use the available computational resources in a distributed environment. The grid resource management must take into account the heterogeneity of the resource in terms of processors, storage and network performance.