Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 23 (2007) 371–381
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

A high performance grid-web service framework for the identification of 'conserved sequence tags'

Paolo D'Onorio De Meo[a], Danilo Carrabino[a], Nico Sanna[a], Tiziana Castrignano[a,*], Giorgio Grillo[b], Flavio Licciulli[b], Sabino Liuni[b], Matteo Re[c], Flavio Mignone[c], Graziano Pesole[b,c,1]

[a] *CASPUR: Supercomputing Center for University and Research, Via dei Tizii, 6/b - 00185 Rome, Italy*
[b] *Istituto Tecnologie Biomediche - Sezione di Bari, C.N.R., Bari, Italy*
[c] *University of Milan, Dipartimento di Scienze Biomolecolari e Biotecnologie, via Celoria 26, Milan 20133, Italy*

Received 15 December 2005; received in revised form 18 May 2006; accepted 2 July 2006
Available online 27 October 2006

Abstract

The continuous increasing of computing power in biological research places a threshold to the single host use and suggests an approach based on distributed computing. An emerging solution is grid technology, which allows organization to make better use of existing computing resources by providing them with a single, transparent, aggregated source of computing power. Equally, bioinformatics analysis often involves many web services, allowing shared access to information and helping the biologist to design, describe, record complex experiments. A new generation of grid infrastructure, where web services are building blocks, allow managent of a web services workflow.

This work shows a tool for the identification and functional annotation of 'Conserved Sequence Tags' (CSTs) through cross-species genome comparisons, deployed on a Grid System Architecture, based on Web Services concepts and technologies.

1. Introduction

The explosive growth of biological data, stimulated by genome projects, has generated a parallel development of efficient computational approaches suitable for several biological research projects. In this area the need of High Performance Computing (HPC) is growing, though usually not affordably by computational resources of a single research laboratory. Grid computing addresses this problem by coordinating and unifying several computational resources [1], allowing the evaluation and mining of a large amount of data in the terabyte and petabyte range. Unfortunately, present-day versions of grid middleware provide only a small part of the functionality required from a bioinformatics community. On the other hand, web services are the distributed computing technology that offers powerful capabilities for scalable interoperation of heterogeneous software across a wide variety of networked platforms [2]. They give the opportunity to create a framework in which applications distributed across local and wide area networks can interoperate through a set of standard protocols. The crucial difference with the past is that most of the systems consisted of ad hoc solutions (e.g. CGI programs) whereas the web services (WS) should lower the barrier to application integration. To increase individual and collective scientific productivity by making powerful information tools available to everyone, a service-oriented strategy is necessary. New projects on service-oriented grids [3] have the assets of both grid and web service technology and help researchers to obtain high performance web services. Complex applications exchanging huge amounts of data, using several web services, have to be managed to gain high performance and high avalability systems, encouraging convergence of grid and web services.

Among those classes of applications, to face the problem of identifying and assessing the coding or noncoding nature of conserved sequence tags (CSTs) through cross-species genome comparisons [4–6], we present a grid-web service

* Corresponding author. Tel.: +39 644486704.
E-mail address: tiziana.castrignano@caspur.it (T. Castrignano).
URL: http://www.caspur.it/CSTgrid (T. Castrignano).
[1] Present address: Dipartimento di Biochimica e Biologia Molecolare "E. Quagliariello", Università di Bari, Italy.

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.07.012

framework, CSTgrid, whose core is implemented as web services. It is composed of one grid daemon module and by seven web services, three for grid components and four for resource components. CSTgrid web tool, available at http://www.caspur.it/CSTgrid, has been developed as an Open Grid Service Architecture, in which services act as a building block of the grid system, allowing a biology community to use all services without any knowledge of the underlying infrastructure. It can provide high performance, high availability and can fairly handle hundreds of concurrent requests. The grid infrastructure has an *ad hoc* library, implemented as a set of web services, developed while the grid community is working on a standard toolkit for a service-oriented grid [3]. Furthermore our grid-web service prototype built to minimize the overhead of standard grid toolkit (e.g. Globus toolkit [7]), is based on grid source components developed compliant to Gtk standards, thus permitting an easy migration path to future grid service-oriented standards.

2. The problem of the identification of conserved sequence tags (CSTs)

The annotation of sequence features in genome tracts is a fundamental task in genome analysis. Although the complete genomes of several eukaryotic organisms have been sequenced, we are not yet able to detect their complete gene inventory, including their regulatory elements [8–10]. The identification and assessment of the coding or noncoding nature of conserved sequence tags (CSTs) through cross-species genome comparisons may contribute significantly to functional annotation of whole genome sequences with the discovery of novel genes or gene expression isoforms. It is well known that both sequences regulating gene expression and genes are conserved among species, therefore the problem of discriminating between potentially coding and noncoding conserved sequence tags (CSTs) arises. The computation of a coding potential score (CPS) for each CST identified in a pairwise genome comparison has been introduced [4] that provides a reliable classification of CSTs in coding (high CPS) and noncoding (low CPS), these latter being candidates of some regulatory activity. An improved version of the method described in [4] has been developed as a web tool [5]. It provides, through gene name or chromosomal coordinates, direct access to Ensembl genomes [11] as well as the possibility for the user to submit data directly. The graphical output of detected CSTs shows known gene features (mRNA, exons, etc.) of the genomic region under investigation and allows researchers to infer easily the functional annotation of CSTs. The described algorithms are not able to perform a search for CSTs between a query sequence and the huge amount of nucleotide sequence data produced by large scale sequencing projects. GenoMiner algorithm [6] is a first experiment in this direction. A GenoMiner analysis is partitioned into three steps: (1) the best local alignments between the query sequence and the target genomes identified by BLAT [12] define one or more homologous regions in the selected target genomes; (2) CSTs are detected though a BLAST-like alignment with sensitive parameters of the query sequence and the homologous regions delimited previously using sensitive parameters; (3) a CPS is assigned to each detected CST as described in [5].

The GenoMiner web tool allows the user to paste or upload a query sequence in Fasta format with no length limitation and select one or more target genomes. Presently, eleven vertebrate genomes collected in the Ensembl database are available for GenoMiner analysis. In order to increase computational efficiency, the searches against genome assemblies are spread across multiple distributed BLAT servers. This tool has been developed using distributed computing technology without the use of any standard developed platforms such as web services or grid technology. CSTgrid has been implemented with the aim of searching CSTs among several couples of nucleotide sequences allowing the user to analyze huge tracts of whole genomes.

2.1. Description of basic tools involved

A set of four web services (*GeneInfo*, *Features*, *Seq_ret*, *CSTfinder*) has been developed allowing the user to perform a CST search in four different ways: (1) pasting the sequences (in FASTA format), (2) uploading a text file containing one query sequence and one target sequence, (3) submitting the Ensembl gene ID and selecting the corresponding organism and (4) selecting the organism and choosing the chromosomal range. The first two selection cases involve the use *CSTfinder* of WS only whereas the last two involve the other three WSs needed to compose the CTSminer output. In Table 1 we list each of four WS with a short description and the input and output streams. A summary of the interaction among these resources is reported in Section 3.2.

Both *GeneInfo* and *Features* WSs query liteDB, a home-made database of features and genes annotated on genomes. Gene Info takes a Ensembl gene name and queries liteDB for the chromosomic coordinates of the gene. Features takes the chromosomic coordinate and queries liteDB for the list of annotated features. Data to populate liteDB tables are mainly extracted from UCSC and Ensembl databases, but other sources can be used. The advantage of using liteDB is that information taken from different sources is parsed and stored with homogeneous structure. Moreover, liteDB has been designed with a very simple structure so that direct queries to the database can be performed avoiding the need for complex API.

Seq_ret WS is based on a custom C program designed to extract efficiently genomic sequences given the organism name, the absolute genome coordinates and strand orientation (forward or reverse) of the required region. If reverse orientation is selected *Seq_ret* outputs the reverse-complement of the sequence. *Seq_ret* works extracting relevant sub-sequence from fasta files containg the assembled sequences of chromosomes. It is also possible to extract masked sequences where TRF and RepeatMasker repeats are substituted by 'N' if chromosomes used have been soft-masked (i.e. repeated nulceotides are shown in lower-cases). *Seq_ret* has been designed keeping performance in mind; it is able to extract

Table 1
Short description and the input and output streams of web services resources

ResourcesWS	Description	Input	Output
GeneInfo	Gives information about a gene (chromosome number, coordinates, strand).	An ENSEMBL identifier.	A chromosome name, chromosomal range, a strand.
Features	Gives a list of annotated features in a chromosomal range.	An organism, a chromosome name, a chromosomal range.	A list of features.
SeqRet	Extract DNA sequence from a chromosomal range of an organism.	An organism, a chromosome name, a chromosomal range, a strand, a mask option.	A DNA sequence.
CSTfinder	Performs the search of CSTs between two DNA sequences.	Two DNA sequences.	A list of CSTs and their associated features (Coding Potential Score, Alignment, % Identity, etc.).

the selected region much faster than similar programs, such as extractseq from EMBOSS package.

CSTfinder represents the core of the resources and essentially implements the new version of the algorithm described in [4] with default parameters i.e. word size of 7 and maximum E-value of 10^{-5} for Blast analysis and minimum CST length of 60 nt [5]. A couple of sequences is needed to run a job. CSTfinder results are displayed by scanning each detected CST with the highest-scoring triplet window (default minimum length of 60 nt). This approach facilitates the detection of potential coding regions located in longer CSTs which might contain both coding and noncoding tracts (through the presence of untranslated mRNA or intronic regions). The resources worklow is explained in detail in Section 3.2.

3. The software architecture of CSTgrid

The system is developed in multi-layered components to allow a Rapid Application Development (RAD) infrastructure and minimal administration efforts. CSTgrid is logically composed by three tiers (Fig. 1):

(1) An *interface tier* responsible for communicating with end-user agents such as web browsers and command line clients.

(2) A generic (not oriented to search CSTs) *grid tier* composed by a grid daemon responsible for the management of the grid resources.

(3) A *resource tier* composed by a set of Resources WS, specific to search CSTs.

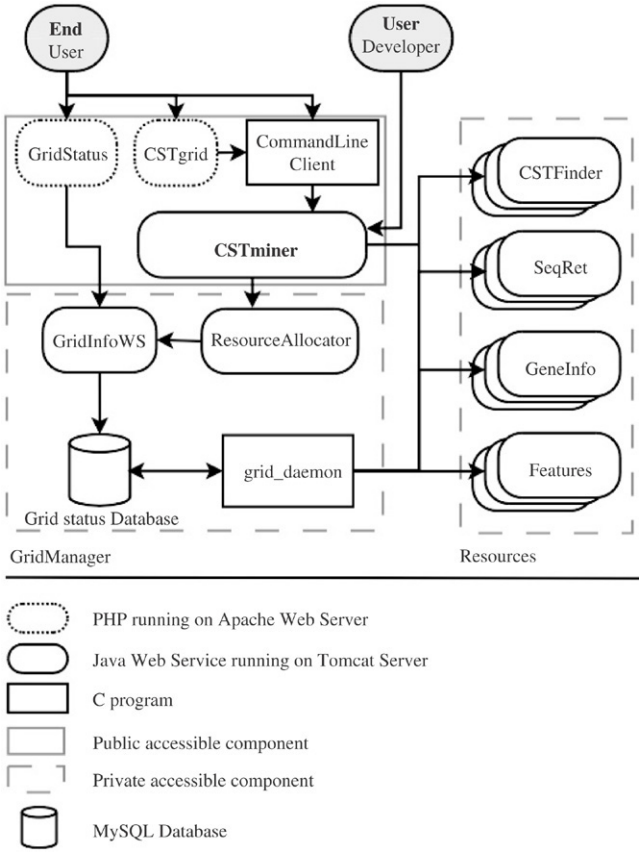

Fig. 1. The CSTgrid distributed software architecture.

3.1. The interface tier

This tier is responsible for communicating with end-user agents such as web browsers and command line clients. PHP scripts (GridStatus and CSTgrid), running under Apache, allow the user both to obtain information about the status of the grid and to launch a CST search job through a command line client. More in detail CSTgrid script inserts new requests into and fetches results from the CSTminer web service, the specific web service for managing jobs to search CSTs. CSTminer performs continuously the following steps:

* receives a request from a client;
* obtains information about free resources from the ResourceAllocator web service;
* uses several resources depending on the input request to perform CSTs search;
* sends CST results to the end-user agent.

CSTminer is a public WS available to an end-user developer through the standard service description layer, Web Service Description Language (WSDL), an XML grammar for specifying a public interface for a web service. Using

CSTminer WSDL the end-user developer can locate the WS and invoke any of the publicly available functions from home-made applications. As with any WS, CSTminer can let users create new, more complex software that makes use of CSTs data through the standard web service.

3.2. The grid manager tier

The grid manager tier is based on four components: two web services (*GridInfo* and *ResourceAllocator*), one database to store information about the grid status and one grid daemon. The database contains all the information about the hosts taking part to the grid, the services available on that hosts and the history of the availability of these services. The history data are managed by the grid daemon, a C program running in the background, which periodically queries services to know their actual status and then stores this information in the database. The detecting time interval for a given WS is calculated by the system and thus configured and stored in the database.

GridInfo is a private web service responsible for giving access to information about grid status vis vis the external world via the web. *GridInfo* sends its data to two components:

(1) the GridStatus PHP page;

(2) the ResourceAllocator web service for the managing of the resources.

ResourceAllocator is a web service responsible for taking resource requests and providing access to them according to a load-balancing fail-safe policy. It takes up-to-date information about the grid by the *GridInfo* web service. Traditional control algorithms for load balancing include Random, Round-Robin (RR) [13], Weighted Round-Robin (WRR) [14], Least Load, Least Connections and Fastest Response algorithms [15]. For CSTgrid platform, in *ResourceAllocator*, we implemented, as failure-safe policy, the Dynamic Weighted Round-Robin (DWRR) [16] for load balancing. DWRR is a variant of WRR, in which the main merit of the algorithm is to minimize the frequency of detection. *ResourceAllocator*, calling the method to perform a DWRR, detects each host's load in the system at intervals and, following the detection of loads, a set of weights (the inverse ratio of host loads) is given to each host. The system allocates new jobs to each host according to the set of weights.

3.3. The resources tier

The resource tier is actually composed of a set of four web services, working together to compose the CTSminer output. In Table 1 we list them with a short description and the input and output streams. All the resources are replicated on several server hosts to obtain the load-balancing of resources. They are the building blocks of the CSTgrid service-oriented architecture. A summary of the interaction of these resources is reported in Section 4.2.

Two of them, *GeneInfo* and *Seq_ret* WS, are necessary for a pre-processing phase. CSTfinder represents the core of resources and Features runs for a post-processing phase. GeneInfo WS is useful to obtain, from an Ensembl gene name, the corresponding organism, chromosome name, chromosomal range and strand. Features and SeqRet WSs accept similar input strings (composed by organism, chromosome name and chromosomal range for both WSs, plus mask option for SeqRet): the former gives a list of features as result, whereas the latter returns the corresponding DNA sequence. CSTfinder WS essentially implements the new version of the algorithm described in [4] with default parameters i.e. word size of 7 and maximum E-value of 10^{-5} for Blast analysis and minimum CST length of 60 nt [5]. A couple of sequences is needed to run a job. CSTfinder results are displayed by scanning each detected CST with the highest-scoring triplet window (default minimum length of 60 nt). This approach facilitates the detection of potential coding regions located in longer CSTs which might contain both coding and non-coding tracts (through the presence of untranslated mRNA or intronic regions). The resources worklow is explained in detail in Section 3.2.

4. Implementing the CSTgrid web services

4.1. The grid enabled CSTminer

CSTminer is a web tool for the identification and characterization of genome tracts which are highly conserved across species during evolution. It is available at http://www.caspur.it/CSTminer. Such a tool make use of local executables to perform CSTs search and is dynamically interconnected to Ensembl genomes. The system was adequate for few concurrent requests, but in case of multiple concurrent requests the server performance dropped. Furthermore, in case of a failure of some part of the distributed system, the entire application was unable to give any output. These facts gave us the idea of developing a grid version of the software where each component of the system was replicated to gain better performances in case of many concurrent requests and to manage component failures. In fact when an incoming search request is submitted according to the input selection the *ResourceAllocator* web service assigns the corresponding resources to different jobs depending on predefined policies. The *CSTminer WS* performs the search using the *Resources WS*, located on remote machines and replicated to obtain the fault-tolerant property.

4.2. Fault tolerance

In the event of a *Resource WS* failure, searches are simply rescheduled on other available servers. Queuing information is stored in the grid-status database possibly to preserve the trace of failure jobs. The end-user agent is also able to show the route and the history of each job. The system also offers an interface to view the status of the grid showing a map with the distributed resources that can be selected to control their state, history, load, etc. The grid daemon is the managing component of failures. It periodically queries servers and stores information about their status in the database. Therefore when the CSTgrid server asks for free resources the *ResourceAllocator* web service, through the information stored in the database, will exclude

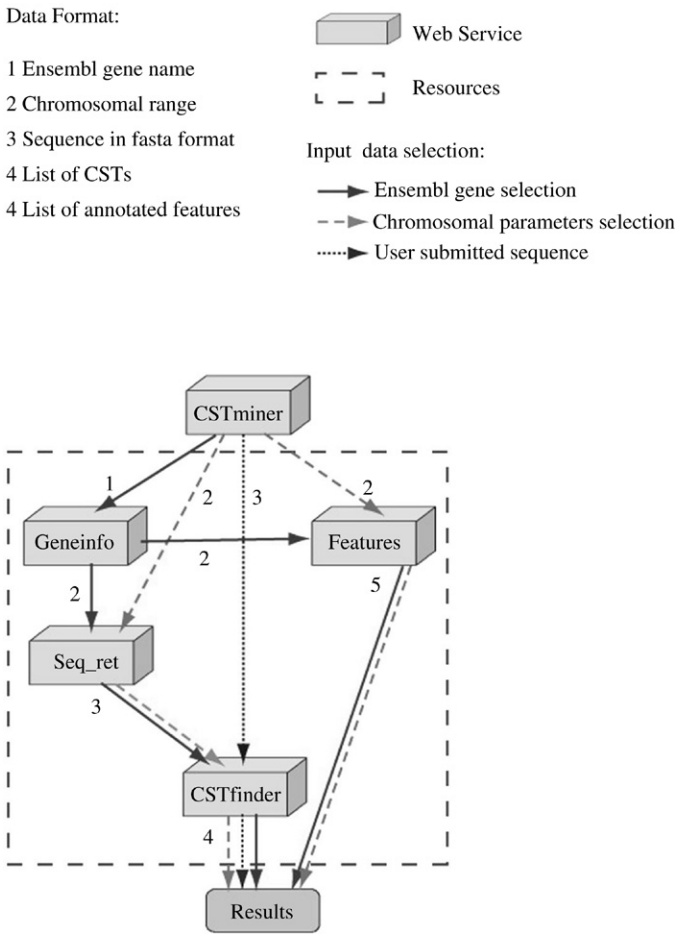

Fig. 2. Workflow of CSTgrid jobs.

those unavailable. If suddenly a resource becomes unavailable while the CSTgrid server is using it, the CSTgrid server notifies the failure to the grid daemon and requests a new resource.

4.3. Resources dataflow

As seen above, the web service CSTminer directly manages each CSTgrid job. The use of Resources web services depends on the input data selection and in the submission form of CSTgrid the end-user has to provide two sequences to start the search. This can be done in four different ways for both sequences:

(1) pasting the sequences (in Fasta format);

(2) uploading a text file containing the sequence;

(3) submitting the Ensembl gene name and selecting the corresponding organism;

(4) selecting the organism and choosing the chromosomal range.

If one of the selected inputs is a user submitted sequence (cases a and b) *CSTminer* sends at once a sequence to *CSTfinder* WS. If one of the selected inputs is an Ensembl Gene (case c) *CSTminer* uses *GeneInfo* WS to obtain relative chromosomal range. This chromosomal range is used as input of both *Features* and *seq_ret* WS. If one of the selected inputs is a Chromosomal parameter selection (case d) *CSTminer* WS uses directly *Features* and *seq_ret* WS. In all cases, as soon as each of the two sequences are available, *CSTminer* WS submits them to *CSTfinder WS* to search for CSTs. A workflow of CSTgrid jobs is depicted in Fig. 2.

For each completed CST search, the results, made of CSTs and related annotated features in case 3 or 4 input selection, is returned and displayed on the end-user agent. When the user is adopting a web-based agent the CSTs are displayed on the web page. Further details about each plotted CSTs (CPS values vs. input sequence for each CST) are available by clicking on a specific CST.

4.4. The web interface and the command-line client

CSTgrid provides an easy-to-use graphical user interface (http://www.caspur.it/CSTgrid) that allow the users to control grid services even though they may have little or no knowledge of grid computing and Web Services. The details of underlying technologies are completely trasparent to the end-user within the CSTgrid system, while a scheme of the architecture (Fig. 1) is also available on the web ('Grid Architecture' button in the menu bar). By clicking on the 'Use it!' button the end-user can load data, run the jobs and visualize the results.

By clicking on the 'Grid Status' button the end-user can obtain the following information (Fig. 3):

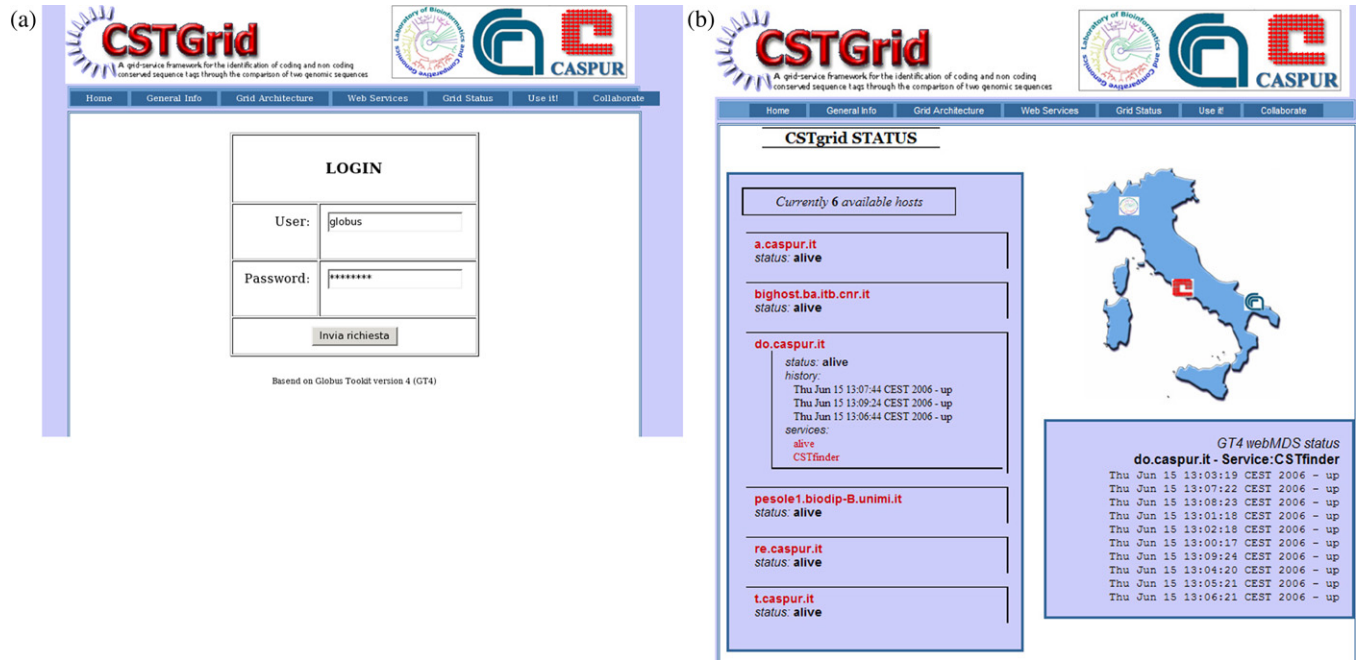

Fig. 3. Screenshot of the web interface showing the grid status.

- current status and status history of each host taking part to the grid
- current status and status history of each resource
- resources available on each host of the grid.

CSTgrid provides a command-line client to let the end-user interact with the system. This client has been developed as a java application with options to select different inputs to submit a CSTgrid job. This tool can be used to perform a very large amount of sequential or parallel CSTs searches and dump the results. Since the results are written in files, the user can write code to analyze the results. The ideal target of this tool are researchers that want to perform CSTs searches on many sequences and to analize them through other tools without using the web interface, that, although providing a friendly interface, is not suitable for batch jobs and very large searches.

4.5. Implementing CSTgrid on the Globus Grid framework

The compliance of a given grid-enabled application with respect to widely accepted standards is a crucial step in its development phase. The introduction of the Open Grid Service Architecture (OGSA) as implemented in the last release of the Globus Toolkit (GT4) offered an open source fundamental technology for the sharing of computing resources and a common reference to the developers of grid applications. Although still in its development stage, the GT4 offers a a set of OGSA capabilities based on the WS Resource Framework (WSRF), which has been used in our project to extend the CSTgrid compliance with respect to standard grid-oriented service infrastructure. At this stage of the CSTgrid development phase, we decided to interface our application at the WS level of the GT4 components in order to create the interface at the upper level of the GT framework. In particular, we devoted much of the efforts toward the implementation within the CSTgrid of (1) grid authentication and credential certification, (2) Grid Resource Allocation and Management, and (3) Grid Data Replication mechanisms.

The mechanism of authentication/certification of user/job over the grid can be generally obtained in a two stage phase with local credential and successive validation of the certificate within a scheme contained in a gridmap file. In our experiment we provided a WS (GT-Auth) for a single point of authentication over the grid of the generic (distributed) 'globus' user with its CA certificate in a LAN shared homedir at CASPUR as reported in Fig. 3(a). By following the suggestions from a larger experiment of the GT infrastructure [17] the extension over WAN of the authentication/authorization phase will be deployed as soon as we implement the data replication mechanism for the user's data, that is, when the CSTgrid will be finally released in production. However, this is still in development phase; the study of the authentication mechanism (local or server-based) to be adapted for grid users in respect of the local users in particular for the sharing of the computing resources, but even at this preliminary stage, it is the opinion of the authors that the flexibility of GT4 components that will permit an easy extension of the GT-Auth WS.

Concerning the integration of the resource allocator within the GT4 framework of reference, we interfaced the GridInfo WS with the WebMDS component in order to retrieve grid status for the job submission. As reported in Fig. 3(b), the CSTgrid is able to get grid information from the GridInfo WS, which can be configured to interface with the GT4 component and/or with our grid status database. At the present stage of development, the CSTminer WS dispatches the resource WSs following the scheduling mechanism reported in Section 3.2,

but a porting using also the globus-*job-submit*() function will be released by the time the CSTgrid will be deployed in production.

The configuration of data replication over the grid nodes participating in the CSTgrid environment is an issue which we are approaching as a two layer problem: (1) the user and (2) the resource data sets. First, the mechanism implemented for the replication of the user data should be compliant with the authentication scheme adopted (see above). While the computational experiments reported into the next benchmark section were carried out using either LAN-based NFS or WAN based AFS filesystems, the replication of local homedirs using the GT4 components could be as well implemented. However, the size of the user datasets to be moved over the grid and the synchronization of the data flows is still a matter to be assessed depending on the typical usage of the CSTgrid tool from the community of users. On the other hand, the FASTA datasets used by most of the WS resources (mainly *Seq_ret*) are easier to manage and a data replication scheme among the nodes which synchronize the local (equally replicated) databases were implemented using the Data Replication components of GT4. The benchmarks reported in the next section were carried out among nodes all with the same replicated FASTA sequences and the overhead due to the data movement is clearly reported as a function of the data size (number of sequences = number of organisms), this becoming less important as the number of WS resource fetched over the grid increases.

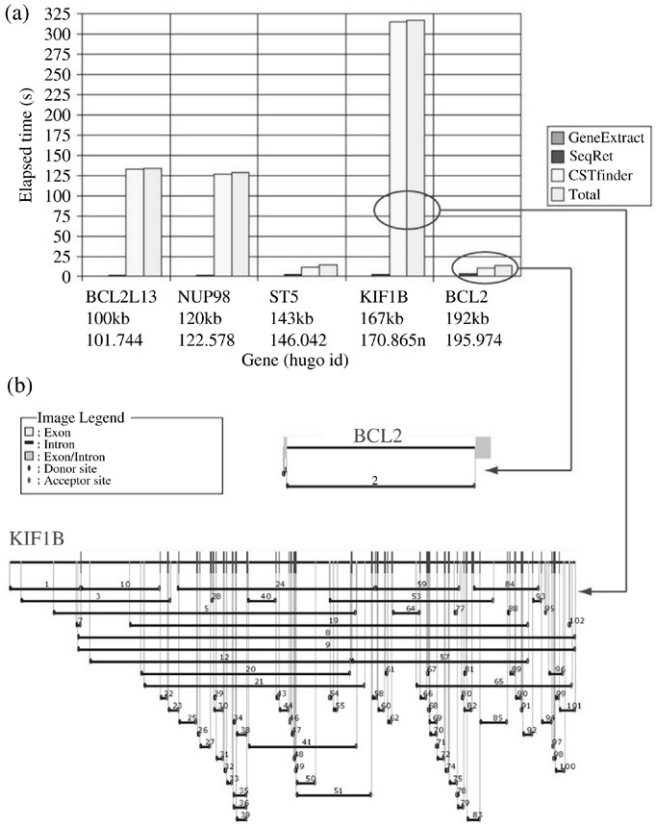

Fig. 4. (a) Elapsed time of CSTminer resources vs. different genes. (b) Analysis of constitutive and alternative splice sites for KIF1B and BCL2 genes.

5. Benchmarks

In order to study the code scalability as the complexity of the run increases, we setup a computing environment with two DL385 (2x AMD Opteron @ 2.6 GHz, B1 and B2) and two DL585 (4x AMD Opteron @ 2.4 GHz, Q1 and Q2) interconnected via Cu–Gb ethernet (one *op* between B1–B2/Q1–Q2, 3 *ops* between B1–Q1/B2–Q2). Although this testbed system was implemented over LAN, the particular configuration we setup for the above multiprocessor SMP machines, has been able to clarify the performance of the *CSTgrid* WS and its components as reported below in detail.

A first test of *CSTgrid* system has been performed over the B1 machine to measure the elapsed time proper of each resource using five genes. The aim of this benchmark was to monitor the weight of each resource growing the nucleotide length of the submitted genes. In Fig. 4(a) the results are shown. The elapsed time of *GeneExtract* was very low and negligible with respect the CPU time needed from the other resources. As expected, there was a linear dependence between the elapsed time and the length of genes for *Seq_ret* WS which extracts a given sequence from the FASTA chromosomes. On the other hand, a direct correlation between the elapsed time and the length of genes for *CSTfinder* WS has not been observed. A cause of the lack of correlation could be due to the order of gene complexity (i.e. how many exons and introns constitute the gene). For istance KIF1B and BCL2 genes are different in length for about 15 Kb length (only 12% of whole gene length), but the corresponding *CSTfinder* elapsed time differs greatly. To verify the hypothesis that this effect could be due to the complexity of the gene, using the Aspic prediction server [18] we detected the intron–exon structure of the two genes: KIF1B gene has a very complex structure, 102 predicted introns, while BCL2 has only few splicing sites, 3 predicted introns (Fig. 4).

A second test of *CSTgrid* system has been performed on the two WSs and was more time comsuming: *CSTfinder Seq_ret*. The benchmark was made on the same B1 machine, adding 15–30–45 Kb nucleotides to ST5 gene. The added sequences were generated randomly to avoid growing the complexity of ST5 sequence. The results of Fig. 5 show that the elapsed time is proportional to the sequence length (see Fig. 4(b) for detail) for both WSs and in particular, the computing time is linearly dependent to the ST5 randomly added chunks of nucleotides. From these results, one is able to forecast the computing time when the dataset (the total number of nt where to perform the CSTs search) will increase or when several of these jobs have to be carried out at the same time, as expected in real production runs of the *CSTgrid* application.

From the results of the last tests, we selected the ST5+30 Kb as target sequence for the complete *CSTfinder* benchmarks over the whole SMP cluster described above. The aim of these series of tests was to measure the *CSTfinder* code scalability and efficiency with respect to the number of jobs submitted to the grid. Furthermore, having several SMP architectures at the same time, the overhead due to multiple fetches of the

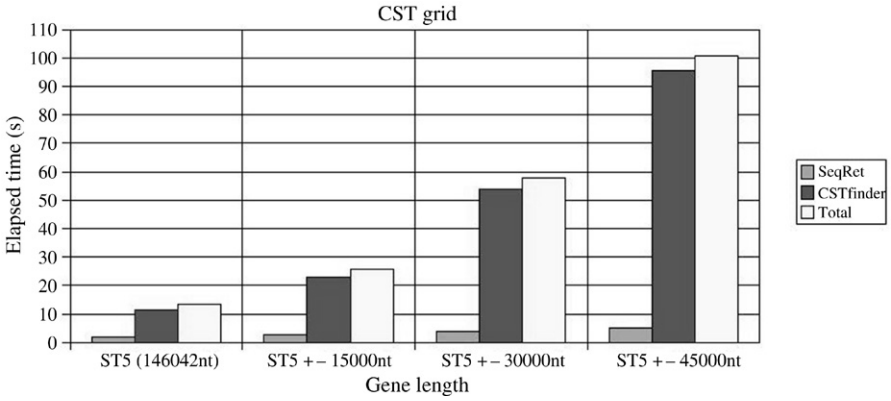

Fig. 5. Elapsed time of CSTminer resources vs. different gene length.

Table 2
CSTFinder WS elapsed time (ET) in seconds of the ST5+30 Kb job (see text for details)

Node	elapsed time (ET) (s)	overhead (s)	code efficiency (%)
B1_1	26.6		
B1_2	26.8	0.2	99.2
B2_1	26.4		
B2_2	27.0	0.5	98.1
Q1_1	31.3		
Q1_2	32.0	0.7	97.8
Q1_3	33.1	1.8	94.5
Q1_4	33.8	2.5	92.6
Q2_1	32.2		
Q2_2	33.4	1.2	96.4
Q2_3	33.7	1.5	95.5
Q2_4	34.0	1.8	94.7

Timings refer to multiple spawning of WS where each multiprocessor machine acts as master node for a complete run (XML-IO+CPU).

Table 3
CSTFinder WS elapsed time (ET) in seconds of the ST5+30 Kb job (see text for details)

Node	Elapsed time (ET) (s)	Overhead (s)	Code efficiency (%)
B1_1	27.7		
B1_2	28.5	0.9	97.5
B2_1	28.7	1.1	96.1
B2_2	31.5	3.8	87.9

Timings refer to multiple spawning of WS where the B1 biprocessor machine acts as master node for a complete run (XML-IO+CPU).

Table 4
CSTFinder WS elapsed time (ET) in seconds of the ST5+30 Kb job (see text for details)

Node	Elapsed time (ET) (s)	Overhead (s)	Code efficiency (%)
Q1_1	32.1		
Q1_2	32.7	0.5	98.4
Q1_3	34.3	2.2	93.5
Q1_4	36.2	4.1	88.6
Q2_1	37.7	5.6	85.1
Q2_2	37.8	5.6	85.1
Q2_3	37.9	5.8	84.6
Q2_4	39.4	7.2	81.6

Timings refer to multiple spawning of WS where the Q1 quadriprocessor machine acts as master node for a complete run (XML-IO+CPU).

CSTfinder WS has been evaluated, either over local or remote nodes of the grid, thus including any bottleneck arising from multi-*ops* network paths.

The first test results are given in Table 2, where the elapsed time of the *CSTfinder* WS are reported when each machine of the grid acts as master node for the job submission. In Tables 3 and 4 we report the same data and in Table 2, but when the B1–B2 and Q1–Q2 sets of machines were clustered together with the first acting as master node. In the Tables 2–4 we also report the overhead due to multiple instances of the *CSTfinder* WS over the grid and the corresponding code efficency given by the ratio of the measured parallel execution time with respect to the elapsed time of a single WS instance. We should note however, that the jobs we carried out included the whole workflow of the *CSTfinder* WS, that is (1) XML input data stream; (2) computing phase; and (3) XML outuput data stream (XML-IO+CPU). Taking this fact into account, the results showed in Table 2 are fairly good with a single node spawn ovehead well below 10% of the execution time and with a computational efficiency of 93% and above. Concerning the multi-*ops* benchmarks, of Table 3 (B1-2) and Table 4 (Q1-2) we found a a larger drop in efficiency than expected (down to 82% for the quadriprocessor cluster) with a corresponding increase of the WS multi instances overheads.

In order to investigate further this unusual feature of the *CSTfinder* WS, we setup a second set of benchmarks without the input/output IO via XML, thus focusing the measurements on the CPU bound section of the workflow (CPU only). The results of the tests carried out on the same ST5+30 Kb target system described above are reported in Tables 5 and 6. The data reported clearly show the bottleneck due to the XML-IO stage of the workflow: once this phase is skipped from the computation the *CSTfinder* code efficiency return to very good

Table 5
CSTFinder WS elapsed time (ET) in seconds of the ST5+30 Kb job (see text for details)

Node	Elapsed time (ET) (s)	Overhead (s)	Code efficiency (%)
B1_1	24.4		
B1_2	24.4	0.0	99.8
B2_1	24.9	0.4	98.1
B2_2	25.0	0.6	97.7

Timings refer to multiple spawning of WS where the B1 biprocessor machine acts as master node for a partial run (CPU only).

Table 6
CSTFinder WS elapsed time (ET) in seconds of the ST5+30 Kb job (see text for details)

Node	CST elapsed time (s)	Delta	Efficiency (%)
Q1_1	28.8		
Q1_2	29.1	0.4	98.7
Q1_3	29.6	0.9	97.0
Q1_4	30.0	1.3	95.8
Q2_1	30.1	1.4	95.6
Q2_2	30.4	1.7	94.7
Q2_3	30.5	1.8	94.4
Q2_4	30.8	2.2	93.4

Timings refer to multiple spawning of WS where the Q1 quadriprocessor machine acts as master node for a partial run (CPU only).

efficiency (93% and above) and the overhead is reduced on both bi- and quadri-processor clusters. The reason why this effect of reduced performance take place is probably due to the serialization of the XML I/O streaming from the apache servers running on each nodes, and on very large computational grids composed of many different SMP architectures; this factor should be tuned with care in order to reach the expected parallel performance of the CSTfinder WS.

In Table 7 we report the last set of measuments regarding the data replication of the FASTA databases used by our *CSTgrid* package with the aim of understanding the impact of transferring the most relevant datasets over the grid nodes. We carried out several node-to-node copy instances between couples of bi- and quadri-processors machines with one or three network *ops* using the *scp* and *rsync* tools. We found the latter tool to be more efficient in any test we performed independently of the network *ops* between the grid nodes; taking into account the extremely good perfomance over LAN via TCP/IP we obtained using *rsync*, we expect to implement a data replication mechanism over the WAN grid of the *CSTgrid* databases in the production phase. Furthermore, even taking into account the expected reduced performance of the data replication mechanism over the WAN network among the *CSTgrid* centres, we are confident that these results confirmed the feasibility of sharing over the grid a common database of FASTA sequences and efficient maintenance of the upgrades.

6. Conclusion

CSTgrid architecture is highly modular allowing an easier development and debugging process. The system has been developed as a Service-Oriented Architecture based on a collection of web services distributed over a geographical grid. It deploys an interface layer, completely independent of an underneath grid-layer. The system has been designed in a user-centric way providing two points of access: the first one is for an end-user to perform high-performance CST serches; the second one is for the developer user to build new large scale WS applications. The access point for the end-user is composed of two tools: a web interface giving an easy-to-use connection to CSTgrid system both in the submission of data and in the visualization of results, and a client to perform batch searches. The access point for the developer user is essentially the CSTminer web service. It provides several methods allowing CST searches based on different inputs parameter selection. It gives functionalities similar to the web interface and the command-line client, but has to be used as a component of more complex applications.

The system has been developed as a grid-aware application to deploy fault tolerance, high availability and high performance. The fault tolerance and high availability have been obtained by replicating all resouces on many hosts and creating the grid manager tier responsible of dynamically allocating resources and taking care of their status. Of course, the whole system still has two single points of failure: the grid manager layer and the web server component. For the improvement of the web server availability, an easy solution could be a cluster of web servers accessed through a load balancer. On the other hand, the problem of the grid manager layer is more complex, since the simple replication of the grid managers on several hosts does not give the solution. The release of a standard architecture for a service-oriented

Table 7
Data transfer rates of selected chromosomes and whole genome (human and mouse FASTA format databases) using the scp and rsync tools

Organism	Chromosome (size)	Scp timing (s)	Scp bandwidth (MB/s)	Rsync timing (s)	Rsync bandwidth (MB/s)
Human	*Chr1* (238.83 Mb)	9.7	24.6	8.3	28.8
	Chr14 (103.46 Mb)	4.7	22.0	4.5	23.0
	Whole genome (2992.94 Mb)	86.0	34.8	78.3	38.2
Mouse	*Chr1* (189.88 Mb)	6.5	29.3	6.3	30.1
	Chr14 (113.88 Mb)	5.2	21.9	4.8	23.7
	Whole genome (2562.39 Mb)	67.2	38.1	62.1	41.3

grid will give the best solution to the high availability of this component.

CSTgrid high performance is based on a native concurrent architecture where several distributed tasks are managed across the network. Future efforts will be dedicated to the optimization of the parallel execution of exact CST resource, in order to best fit the guest host architecture and real-time environment, by gathering information from the CST grid manager database.

References

[1] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, Internationl Journal of Supercomputer Applications 15 (2001) 3.

[2] Etham Cerami, 2002. Web Services. O'Really.

[3] I. Foster, C. Kesselman, J.M. Nick, S. Tuecke, The physiology of the Grid: An open Grid services architecture for distributed system integration. 2002. http://www.globus.org/alliance/publications/papers/ogsa.pdf.

[4] F. Mignone, G. Grillo, S. Liuni, G. Pesole, Computational identification of protein coding potential of conserved sequence tags through cross-species evolutionary analysis, Nucleic Acids Research 31 (15) (2003) 4639–4645.

[5] T. Castrignano, A. Canali, G. Grillo, S. Liuni, F. Mignone, G. Pesole, CSTminer: A web tool for the identification of coding and noncoding conserved sequence tags through cross-species genome comparison, Nucleic Acids Research 32 (Web Server issue) (2004) W624-7.

[6] T. Castrignano, P. D'Onorio De Meo, G. Grillo, S. Liuni, F. Mignone, I.G. Talamo, G. Pesole, GenoMiner: A tool for genome wide search of coding and noncoding conserved sequence tags, Bioinformatics (2) (2005).

[7] I. Foster, C. Kesselman, Globus, A metacomputing infrastructure toolkit, International Journal of Supercomputer Applications 11 (1997) 115–128.

[8] M.A. Nobrega, L.A. Pennacchio, Comparative genomic analysis as a tool for biological discovery, Journal of Physiology 554 (2004) 31–39.

[9] L.A. Pennacchio, E.M. Rubin, Genomic strategies to identify mammalian regulatory sequences, Nature Reviews Genetics 2 (2001) 100–109.

[10] E.T. Dermitzakis, A. Reymond, R. Lyle, N. Scamuffa, C. Ucla, S. Deutsch, B.J. Stevenson, V. Flegel, P. Bucher, C.V. Jongeneel, et al., Numerous potentially functional but nongenic conserved sequences on human chromosome 21, Nature 420 (2002) 578–582.

[11] E. Birney, D. Andrews, P. Bevan, M. Caccamo, G. Cameron, Y. Chen, L. Clarke, G. Coates, T. Cox, J. Cuff, et al., Ensembl 2004, Nucleic Acids Research 32 (Database issue) (2004) D468–D470.

[12] W.J. Kent, BLAT—the BLAST-like alignment tool, Genome Research 12 (4) (2002) 656–664.

[13] J. Nagle, On packet switches with inBnite storage, IEEE Transactions on Communications 35 (4) (1987) 435–438.

[14] M. Katevenis, C. Sidiropoulos, C. Courcoubetis, Weighted Round-Robin Cell multiplexing in a general-purpose ATM switch chip, IEEE Journal on Selected Areas in Communications 9 (8) (1991) 1265–1279.

[15] River Stone Networks, Server load balancing: the key to a consistent customer experience, Technology white paper, No. 101. http://www.riverstonenet.co.jp/pdf/technology/whitepapers/SLB˜v5.PDF, December 2002. p. 1–6.

[16] D.-C. Li, C. Wu, F.M. Chang, Determination of the parameters in the dynamic weighted Round-Robin method for network load balancing, Computers and Operation Research 32 (2005) 2129–2145.

[17] Computing in High Energy and Nuclear Physics, 24–28 March 2003, La Jolla, California. SLAC-PUB-9983.

[18] T. Castrignano, R. Rizzi, I.G. Talamo, P. D'Onorio De Meo, A. Anselmo, P. Bonizzoni, G. Pesole, ASPIC: A web resource for alternative splicing prediction and transcript isoforms characterization, Nucleic Acid Research 2006 (in press). To be published in Web-Server Issue 2006.

Paolo D'Onorio De Meo graduated with a Bachelors degree in Computer Science from the University of Rome 'La Sapienza' in 2004. He has been a bioinformatics developer at the Caspur since 2004.

Danilo Carrabino graduated with a Bachelors degree in Computer Science from the University of Rome 'La Sapienza' in 2003. He has been a bioinformatics developer at the Caspur since 2005.

Nico Sanna is chief of the Computational Biology and Chemistry Group at Caspur (Consorsio per le Aplicazioni di Supercalcolo per l'Universita' e Ricerca). He received his Master in Chemistry in 1990 from the University of Rome 'La Sapienza'. His primary research interest is in the development of high performance computing applications.

Tiziana Castrignano is bioinformatics specialist at Caspur (Consorsio per le Aplicazioni di Supercalcolo per l'Universita' e Ricerca). She received her Ph.D. in Biophysics in 1999 from the University of Rome 'La Sapienza'. Her primary research interest is in the development of high performance bioinformatics services.

Giorgio Grillo is a researcher at the Institute Biomedical Technologies CNR of Bari, Italy. He obtained a degree in Computer Science at University of Bari and since 1994 he has been involved in Bioinformatics, particularly in the development of software for genomics and proteomics. He takes part in many national and international research projects for his expertise in the analysis of biosequences and their functional characterization, in the linguistic and computational analysis in nucleotide sequences. Projects include design, development and implementation of bioinformatic databases largely used by international researchers and in the management of query systems.

Flavio Licciulli obtained a degree in Computer Science at University of Bari in 1992. Since 1994 he has been involved in Bioinformatics with the Bari BioInformatics Group. He is also a researcher at the C.N.R.- Istitute for Biomedical Technologies. His main research activity is design, development, implementation and maintenance of biological databases, development of bioinformatics software and WEB interfaces; he has published over 20 scientific publications in these fields. He is responsible for the database and operating systems management at his institute.

Sabino Liuni is a researcher at the Institute of Biomedical Technology CNR-Bari, Italy. He obtained a degree in Biology in 1984 with a thesis on Bioinformatics. He has focused his interest on bioinformatics. In 1989 he was involved in the management and development of the Italian National EMBnet (European Molecular Biology Network) node. His current interests are mainly on the design and implementation of specialized databases of nucleotide sequences and on the development of algorithms for the analysis of biosequence. He has over 50 publications and is a co-author of widely used computational biology software.

Matteo Re is currently a doctoral student in the Department of Molecular Biosciences and Biotechnology at the University of Milan where he is working on methods for the detection and in-silico characterization of evolutionarily conserved sequences, novel genes and gene isoforms. He was awarded his degree (Molecular Biology, University of Milan) in 2004.

Flavio Mignone received his Degree in Biological Sciences from the University of Milan in 1998. Since 2000 he has worked with the Bioinformatics and Comparative Genomics group headed by Professor G. Pesole at the University of Milan. He received his PhD in Bioinformatics from the University of Milan in 2003; currently he is Computer Science researcher at the same University. Main work area includes study of mRNA untranslated regions (UTRs), identification of regulatory elements and identification of unannotated genes using computational approaches.

Graziano Pesole is full professor of Molecular Biology in the University of Bari. He has since long carried out research in the fields of bioinformatics, comparative genomics and molecular evolution. In particular, his interests are computational approaches for the identification of regulatory elements in noncoding genome regions, alternative splicing and functional analysis of untranslated regions of eukaryotic mRNAs. He has developed a specialized database (UTRdb/UTRsite), largely used by the scientific community, collecting mRNA untranslated sequences and related regulatory motifs involved in the post-transcriptional regulation of gene expression. He has also developed analysis software and several algorithms largely used by the scientific community and available also through web browsers. Within his studies on molecular evolution, he has contributed to the development of new analysis methodologies and has carried out several studies on the evolution of mitochondrial genome at the intraspecies level, in order to clarify some aspects of the origin of modern man, and at the interspecies level to reconstruct mammal phylogeny and to study the evolutionary dynamics of the mitochondrial genome of Tunicata. He leads an interdisciplinary research group including molecular biologists, computer scientists and mathematicians. He coordinated research units in several research project funded by national (MIUR, CNR, Telethon, AIRC) and international (EU, NIH) agencies, and has filed an international patent for the selection of primers for RNA fingerprinting. He is a member of the editorial Board of international journals (GENE, BMC Bioinformatics, BMC Genomics, Computational Biology and Chemistry, Briefings in Bioinformatics), author of over 120 papers published in international journals and co-author of books on Bioinformatics and Genomics published by Italian (Zanichelli, Gnocchi) and international (Wiley) publishers. For further information see http://www.pesolelab.it.