Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 66 (2006) 1482–1488
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

# High performance cDNA sequence analysis using grid technology

G.A. Trombetti[a,b], I. Merelli[a,*], L. Milanesi[a]

[a]*Istituto di Tecnologie Biomediche, Consiglio Nazionale delle Ricerche, via F.lli Cervi 93, Segrate (Milano), Italy*
[b]*Università degli Studi di Bologna, Facoltà di Ingegneria, v.le del Risorgimento 2, Bologna, Italy*

Received 10 September 2005; received in revised form 22 July 2006; accepted 15 August 2006
Available online 16 October 2006

**Abstract**

Innovative DNA sequencers, relying on pyrosequencing, are now being produced, which cut down costs and speed up sequencing by an order of magnitude. Hence the capability of handling high throughput sequencing is becoming increasingly important for Bioinformatics.

This study concerns the development of a high performance pipeline for analyzing cDNA sequences produced by a high throughput pyrosequencer. Mainly, this analysis system has been developed by us to map the sequenced cDNA strands against a cDNA database for studying different mutations that can influence the genes functionality. The pipeline supports heterozygous organisms.

In order to use a high throughput pyrosequencer fruitfully, the related bioinformatics analysis requires high performance. Hence we implemented our analysis system leveraging the European EGEE project infrastructure: a network of several computational resources and storage facilities distributed at different sites.

The results of this high performance pipeline are stored into an output database directly from the grid sites using the Web Services technology. By querying this database it is possible to inspect the analysis results to detect different mutations in the cDNA sequences, as well as other meaningful biological parameters and information.


*Keywords:* Bioinformatics; Grid; High performance; Sequencing; cDNA; Distributed; Web services

## 1. Introduction

Bioinformatics studies complex biological processes in silico to understand how reactions allowing the survival of the cell take place. Particularly, genomics aims to study DNA characteristics for establishing the genes functionality, expression features and how their variations can influence biological parameters.

Once a genomic sequence of an organism is completely sequenced, and its peculiarities are washed out by averaging the sequences coming from different individuals (a process similar to that of creating a consensus sequence), it acts as a reference to understand how DNA modifications influence different physiological and pathological gene expressions [15]. Particularly, the sequencing of cDNA, that is DNA cloned from transcribed RNA, is used to determine how variations in the coding zones of DNA influence the gene expression.

This study concerns the development of a genomic analysis system working on the output data of a high throughput sequencer. This pipeline was designed to assemble the cDNA sequences starting from the sequencer output data, and using the human cDNA database as a reference, in order to identify punctual mutations in the expressed sequences.

Our main purpose is to detect punctual variations of the sequenced cDNAs in heterozygous organisms, in order to find either punctual mutations (genomic mutations [7] present in isolate biological samples) or SNPs (*Single Nucleotide Polymorphism*—genomic variations present in a statistically relevant percentage of the population [9]). These analyses are important to establish a relationship between mutations in the coding zone of DNA and genomic diseases.

## 2. Motivation

Problems in genomics and proteomics tend to have a quadratic or higher computational complexity [8]. For example, global/local genome pairwise alignment with general or affine gap penalty functions, genome assembly, inversion

* Corresponding author.
*E-mail addresses:* gabriele.trombetti@itb.cnr.it (G.A. Trombetti), ivan.merelli@itb.cnr.it (I. Merelli), luciano.milanesi@itb.cnr.it (L. Milanesi).

0743-7315/$ - see front matter © 2006 Published by Elsevier Inc.
doi:10.1016/j.jpdc.2006.08.009

distance computation, genome rearrangement analysis and molecular dynamics have all got a quadratic or higher complexity: small increases in the input data, due to the advancement in knowledge or improvement in machines providing the input, greatly increase the computation time. CPU speed increases also have been nonlinear (in fact exponential) for a long time, providing approximately a doubling in speed every two years; however, this is no longer the case, as the speed increases have almost stopped in recent years.

As far as genomic problems are concerned, we must also take into account that the development of sequencers has been far from linear in the last years, recently leading to high throughput pyrosequencers having a tenfold increase in throughput [13], and a similar decrease in operating costs, compared to the previous technology. Such high throughput pyrosequencer technologies create an enormous flow of genomic sequences that must be elaborated in minimum time to best exploit the sequencer capabilities.

In our case, in order to detect punctual mutations, comparing each of the sequences output of the sequencer, called reads, against the whole cDNA database was necessary. Having as reference a large database of over 39,000 cDNAs [14], and the output rate of our pyrosequencer as high as 10,000 reads per hour it was not possible to keep up reliably with a single machine. In addition, we wanted to allow repeatability of past calculations after variation of the algorithm or parameters, which meant a potentially very large dataset to be recomputed in a reasonable time.

Hence, for the implementation of this pipeline a high performance system needed to be designed to coordinate a large number of computation resources and to manage the flow of such large amount of information. This has been possible by leveraging a massively distributed environment such as the grid platform [5].

## 3. Implementation

First make it work, then make it fast. Our first approach was with a non-distributed pipeline (Fig. 1). The first stage of our pipeline leverages Blast [2] to match the reads against the

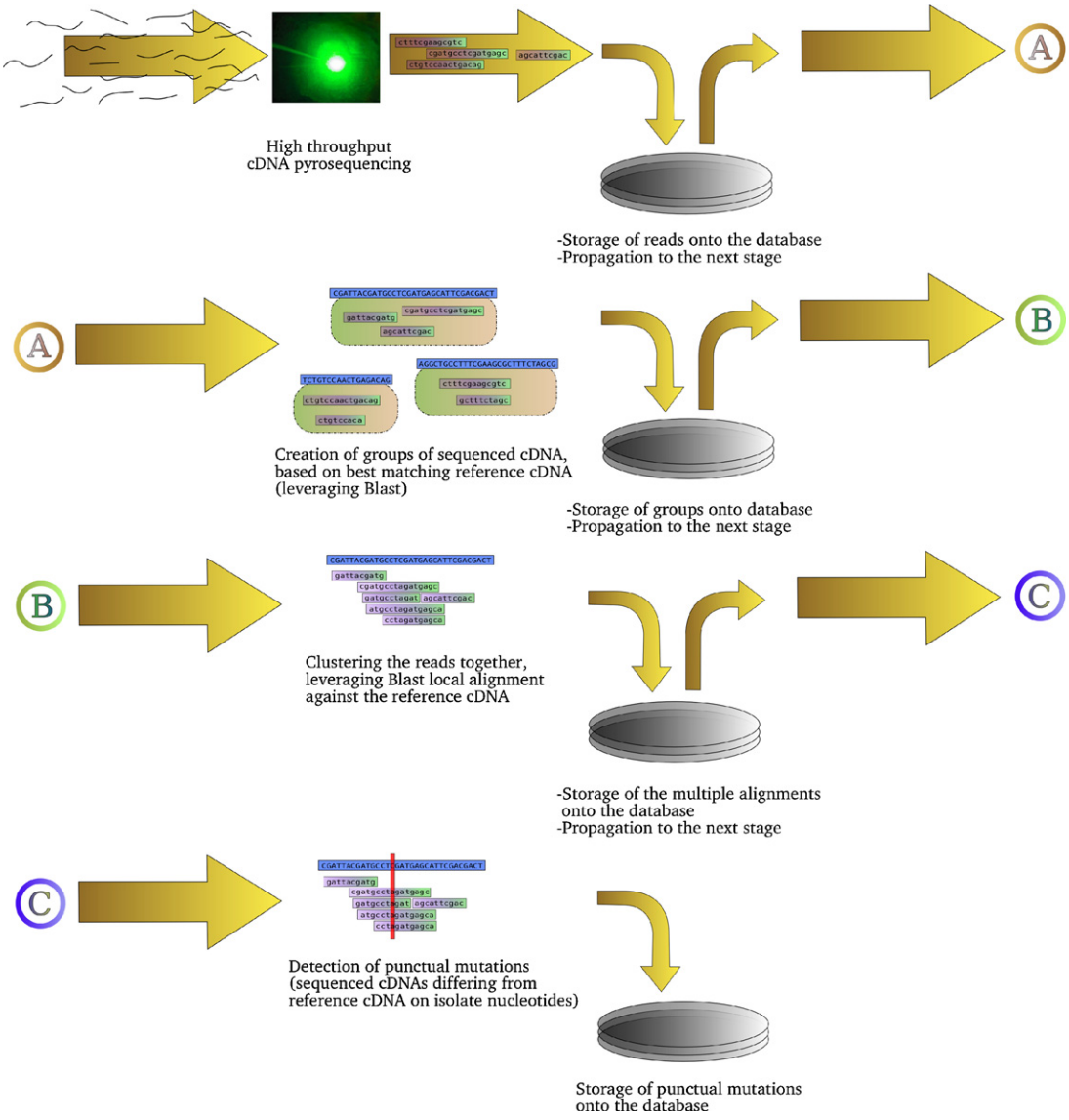


Fig. 1. The analysis pipeline.


Fig. 2. Clustered reads belonging to a cDNA. An A−− >G punctual mutation in one chromosome only (heterozygosis) is shown.

39,000 cDNA reference database. Starting from the Blast results groups of reads are then made, gathering together the reads which best match the same reference sequence. These associations are written onto a database. In case of alternative splicing, two or more reference cDNAs will have common parts, hence, when the Blast matching is attempted, a read matching a common part will match all such reference cDNAs. Since at this first stage it is not possible to determine, or even guess, which reference cDNA this read belongs to, our policy is to accept all. During the next (second) stage, it will be possible to heuristically filter out some reference cDNAs (and their associated groups) which were not in fact expressed, based on the coverage of the cDNA which would be covered by reads only on the common parts of the alternative splicing. In the meanwhile, during this first stage we already distinguish the cases in which the second (or further) Blast match of a read against another cDNA is caused by alternative splicing (for which our policy at this stage is to accept it) from the cases in which the match is determined by random similarity (which we clearly want to reject). An heuristic and partly adaptive algorithm solves this.

Once groups are made, a second computation stage clusters the reads together, using the cDNA as a reference [12]. For this second stage again we leveraged Blast, this time for anchoring the sequences (as Blast "subjects") against the reference cDNA (as Blast "query"), to obtain a multiple alignment of the reads referring to each cDNA [3].

A third computation stage analyzes each multi-alignment obtained at the previous stage looking for punctual mutations. We remark that we are working with gene expressions from heterozygous organisms, which can have any punctual mutations present on either homologous chromosome, or both of them. In the same way, gene expressions also will.

Hence, when looking at multiple alignments, it is to be kept in mind that, on average (even if with a significant variance), half of the aligned reads should come from a chromosome and the other half from the other homologous. In a column of aligned reads where more than one nucleotide is present, the most present nucleotide will most certainly be that of (at least) one chromosome, but the second most present nucleotide can either be a sequencing error or the expression of the gene on the homologous chromosome (Fig. 2).

After a statistical analysis we decided that a coverage of 10× (at least ten overlapping strands over each and every point of the whole multi-alignment) should be used to distinguish reliably between the two, and we put the threshold of >= 30% (of the coverage over that specific point, i.e. the column height) for calling the expression of the second allele.

## 4. The grid platform

The EGEE grid project infrastructure is a wide area grid platform for scientific applications composed of about 1000 CPUs [1]. This platform is a network of several *Computing Elements*, that are the gateways for the computer clusters on which jobs are performed, and an equal number of *Storage Elements*, that implement a distributed file system on which databases are stored. The grid core is a set of *Resource Brokers* delegated for controlling the execution of the different jobs (Fig. 3).

The EGEE grid project infrastructure is founded on the *Globus Toolkit* [4] that represents an ideal communication layer between the different grid components. The European EGEE grid platform middleware relies primarily on this open source software for building grid systems and applications. The major services are implemented using the *Globus Toolkit* software: the security service (GSI), the information service (GIS), the resource management service (GRAM) and the data access to storage facilities (GASS).

The computational resources are connected to a *Resource Broker* that routes each job on a specific *Computing Element* and takes into account the directives of the submitting script,

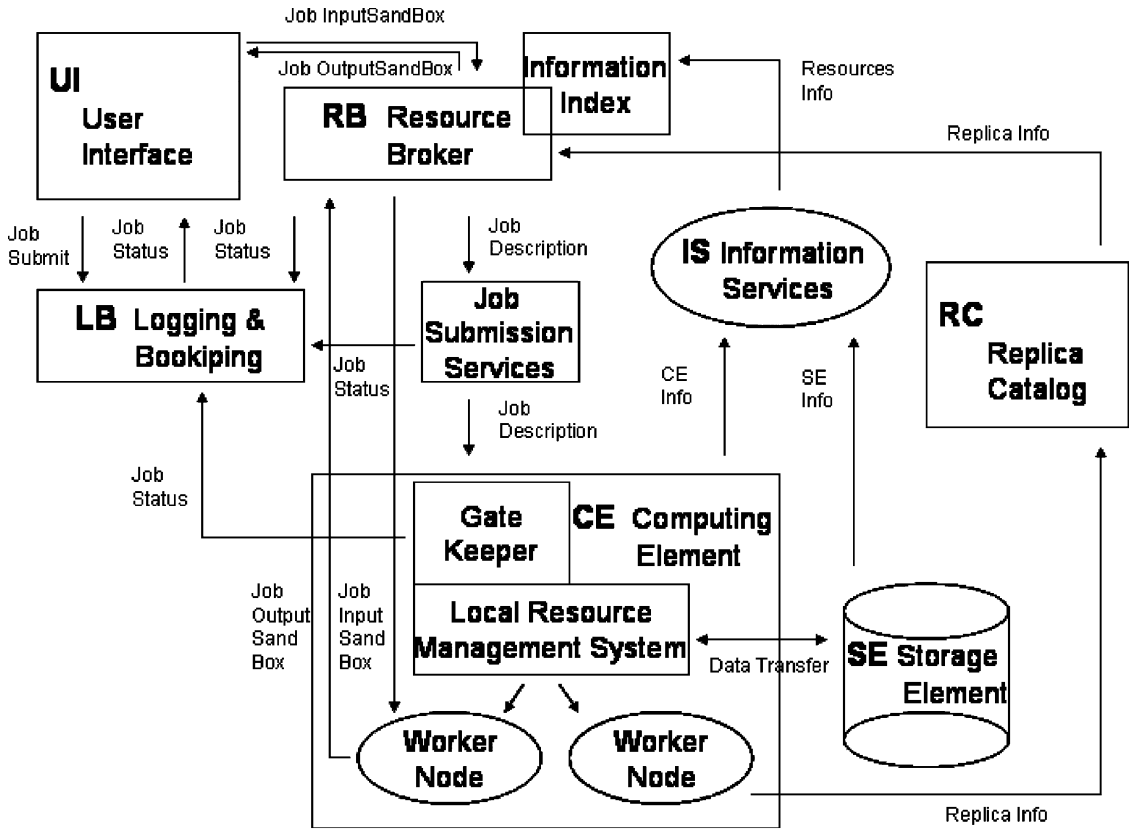


Fig. 3. Schema of the European EGEE project infrastructure with a detailed view on the information flow during a job elaboration.

called JDL because it is composed through the Job Description Language, and implements a load balancing policy. Each *Computing Element* has a *Gatekeeper* service that submits the incoming job to a batch system queue that hides a *Working Nodes* farm.

A set of tools is used to manage data as in a distributed file system. These tools allow the data to be allocated to different *Storage Elements*, maintaining the coherence between them through a *Replica Manager*, and help using this information in an efficient way. The *Resource Broker*, in fact, is able to redirect the execution of an application to a *Computing Element* as near as possible to the data location, minimizing the communication time.

The software that gives access to the distributed platform is made up of a set of tools, by which secure communications can be established between the grid infrastructure and *User Interface*. Through the grid *User Interface* it is possible to submit jobs, control the pipeline state of advancement and retrieve the outputs when the computations have a normal termination or resubmit the jobs in case of failure.

Due to the use of remote computational resources, the grid communication software must offer an efficient security system. The access to remote clusters is granted by a *Personal Certificate*, which accompanies each job to authenticate the user. Moreover, users must be authorized to job submission by a *Virtual Organization*, a grid community having similar tasks, that grants for him. This procedure is indispensable for maintaining a high security level.

## 5. Distributed implementation

Our distributed implementation for this pipeline shares the computational load over the grid nodes of EGEE grid. The whole pipeline is coordinated by a single central server, on which the grid *User Interface* software is installed. This creates a high performance and relatively scalable system according to the grid performance and the power of the central server (the central server can act as a bottleneck in the general case depending on the amount of work assigned to it, but such work was small enough in our case). The central server is both involved in the coordination of the parallel execution of the grid jobs through the *User Interface* and in the pre and post elaboration of sequences.

To obtain a high performance implementation of this analysis pipeline the most time-consuming steps have been implemented in a distributed way. The first step implemented on the grid platform is the blast that groups sequences according to cDNA similarity. The input data of this step are the raw sequences produced by the sequencer while the output is used as input of the second distributed step, anchoring each read to the related cDNA for creating a complete coverage.

Both of these steps are based on blast and their implementation is quite similar. Bioinformatics application that relies on the comparison of an input sequence against a database are usually implemented on a distributed platform by subdividing the input data set in small groups [11]. To manage the distributed implementation of these pipeline steps an efficient system has

been implemented for coordinating the execution of the jobs, controlling the completion status and retrieving the output in case of a successful termination.

For each job a JDL script is generated with the information about the input sequence, the job requirements and the information about the databases that have to be accessed. The jobs are routed by the Resource Broker to the best *Computing Element* that is available at the moment. From the *User Interface* the execution of the pipeline analysis is automatically monitored by our software and, in case of failure, is re-submitted to the grid infrastructure.

Porting onto the grid platform bioinformatics applications which rely on databases implies dealing with distributed database management. In the first analysis step of this pipeline, input sequences are clustered according to a database of cDNA. This is a flat file database of significant size which needs to be transferred to the used *Computing Elements* prior of invoking Blast. In order to minimize the transfer time we replicated the cDNA flat file database to various *Storage Elements*. In this way it is possible to use a high number of *Computing Elements* (each located near one of such replicas, i.e. local network) while keeping the database transfer overhead for the execution of step one minimal.

Grid technology does not support distributed RDBMS. This is a problem when the output data has to be collected in a database. Although it would be possible to retrieve data into the *User Interface* and parse them before storing results in the database, this solution would remarkably slow down the system performance. To obviate this problem a solution based on the technology of the Web Services has been implemented [6].

For each grid job, the blast output is parsed directly on the *Computing Element* on which it has been executed. In this way a temporary result set is created and, eventually, through a small Web Service client carried into the grid together with the input sequence, it is entirely stored into the results database on the *User Interface*. This is performed in one pass, hence minimizing the SOAP communication overhead [10]: the Web Service receives the incoming SOAP message from the client containing information about the blast results, and performs the SQL insert onto the results database.

## 6. Results database

The working data are passed from a computational stage of the pipeline to the next through the help of a database (Fig. 4). In the non-distributed version of the pipeline, the previous stage would store the result on a database which the next stage would read. This ensured complete separation of the stages, and made the repeatability of computations very easy, implying a simple deletion of some result rows of a certain stage from the database (or cloning of some input rows of the same stage) and re-run of the computational stage.

In the current, distributed version of the pipeline, at the end of a stage computation a Web Service is used to dump the results onto the database on the *User Interface*, which is then responsible for passing such data to the next stage. This does add some overhead, however it ensures that:

- the stages are fully separated,
- the computation is repeatable in part or whole,
- the intermediate results are held on a single and local database.

Results of intermediate calculations may hold important information for biological analyses which might not even be fully foreseen at this point. In order to allow inspired searches by the biologists, we kept all such database information meaningfully ordered and easily searchable with SQL queries and scripts.

Results of intermediate calculations include:

- grouping of the reads mapping onto any cDNAs, for each biological sample, potentially giving information for the relative amounts of gene expression for each cDNAs,
- multi-alignments within the groups,
- punctual mutations found (homozygous/heterozygous), whose frequency could be investigated to find new SNPs, or correlation between the presence of different SNPs,
- reads not matching reliably any known cDNA, which can be a clue for a new unknown cDNA that could be investigated further via multiple alignments among such reads, and could then be tested for "amount of expression" comparing it against genomic data collected in the past (e.g. unknown multiple splicing of a known cDNA).

## 7. Distributed implementation performance

Even though the performance of the grid pipeline is more than adequate for our situation, a numeric estimate of such performance is difficult due to the great variance in queue times for jobs sent on the grid. This depends mainly on the workload which is assigned to the grid throughout Europe at the specific time of submission. In addition, the higher the number of jobs which are submitted together to the grid, the more unfavorable (in terms of queue times) the computing resource the last of those will get. Needing to wait the execution of all jobs makes the pipeline wait for the worst queue time of the set of jobs, hence, the pipeline-perceived queue wait time for 40 jobs is significantly worse than the pipeline-perceived queue wait time for 10 jobs.

It is also obvious that, given a certain amount of computation to be performed, splitting such computation into a high number of grid jobs will reduce the size for each of them, and since the jobs are executed independently on the grid, raising the number of jobs increases the execution parallelism and reduces the computation time as it is perceived by the pipeline. Hence it is clear that splitting the workload in a too small number of grid jobs would also hinder the pipeline performance. In facts, in our executions we try to keep the length of every job comparable to the (estimated) queue wait time of the set with a lower limit of 20 min (in order to avoid excessive grid overhead).

Our performance is best described as follows: the non-distributed version of our pipeline needs 19 computation hours on a Xeon class CPU for the 100,000 reads that a high throughput pyrosequencer is ideally able to produce in 1 h. These are


Fig. 4. Entity–relationship diagram for our current database—10 tables. Biological *samples* are sequenced in pyrosequencer *runs* producing *reads*. Computation analysis *projects* are initiated on the reads of one or more runs, producing *groups* of reads according to Blast-detected most similar reference (*cdnas*) in pipeline step 1. These groups are clustered in multialignments (*multialig*) in pipeline step 2 and their punctual mutations (*punmuts*), either homozygous or heterozygous, are finally detected in pipeline step 3.

roughly 10 h for the first step, 6 for the second step and 3 for the third step. With the grid distributed version of our pipeline we can parallelize the three steps as much as we want by raising the number of grid jobs we submit. A good compromise for the number of jobs to complete such a computation in the shortest time considering the queue wait time is around: 20 jobs for the first step, 12 for the second and 6 for the last step, for a total of 36 half-an-hour-long jobs. A typical queue wait time is 30 min for the first step (a 20-job set) 15 min for the second step (a 12-job set) and 10 min for the third step (a 6-job set) making a total wait+execution time of two and half hour on average (but the variance can be significant, as we said, depending on the EGEE Grid load). In addition, our computation resources remain still substantially free and capable of submitting and handling more grid computation if this is needed (e.g. for a recomputation of older genomic data with altered pipeline parameters).

The benefit of the grid is hence very evident, at least for heavy computational loads, and the costs of joining the Grid are minimal (sometimes zero, depending on the load one intends to submit to the Grid) compared to those of a dedicated cluster.

## 8. Conclusion and future work

Bioinformatics computations are intensive, mostly nonlinear in complexity, and continuously asking for more processing power. On the other side, CPU speed improvements recently have nearly ceased and do not give much better hope for the future.

This need of high computation powers finds its solution in distributed and/or parallel environments such as clusters and the grid, the latter being the best solution in terms of lower costs due to its large scale combined with the innate ability of load distribution, which leverages unused machines, hence intrinsically taking into account the sporadic nature of most computations.

We have here shown how we were able to distribute most parts of our power-demanding pipeline for cDNA analysis on

the EGEE grid platform thus obtaining the needed processing power, while still maintaining the final and all the significant intermediate results stored on the *User Interface*.

The performance of the pipeline is difficult to evaluate, because the computation time depends primarily on the total computational load of the grid during the execution of the jobs. On heavy workloads of our pipeline though, the performance benefit of the distributed implementation is very evident compared to local single CPU execution, and the costs are many times lower than those of a dedicated cluster.

Our future work will be aimed at providing a web interface for the pipeline and the results database, to support and ease the work of the biologists.

## Acknowledgments

This work has been supported by the Italian FIRB-MIUR projects "Italian Laboratory for Bioinformatics Technologies-LITBIO" and by the European support action for "Bioinformatics in EGEE-BIOINFOGRID".

## References

[1] R. Alfieri, R. Barbera, A. Cavalli, R. Cecchini, A. Chierici, V. Ciaschini, L. Dell'Angelo, F. Donno, E. Ferro, A. Fortem L. Gaido, A. Gianoli, A. Ghiselli, A. Italiano, S. Lusso, P. Mastroserio, D. Mura, M. Serra, M. Reale, L. Salconi, F. Spataro, F. Taurino, G. Tortone, D. Vicinanza, G.V. Finzi, The INFN-GRID testbed, INFN Technical Report, November 2002.

[2] S. Altschul, W. Gish, W. Miller, E.W. Myers, D. Lipman, A basic local alignment search tool, J. Mol. Biol. 215 (1990) 403–410.

[3] R. Belshaw, A. Katzourakis, BlastAlign: a program that uses blast to align problematic nucleotide sequences, Bioinformatics 21 (2005) 122–123.

[4] I. Foster, C. Kesselman, Globus: a metacomputing infrastructure toolkit, Internat. J. Supercomputer Appl. 11 (2) (1997) 115–128.

[5] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, Internat. J. Supercomputer Appl. 15 (3) (2001).

[6] I. Foster, C. Kesselman, J. Nick, S. Tuecke, The physiology of the grid: an open grid services architecture for distributed systems integration, open grid service infrastructure WG, in: Proceedings Global Grid Forum, 2002.

[7] C.A. Garcia, A. Ahmadian, B. Gharizadeh, J. Lundeberg, M. Ronaghi, P. Nyren, Mutation detection by pyrosequencing: sequencing of exons 5-8 of the p53 tumor suppressor gene, Gene 253 (2) (2000) 249–257.

[8] S. Karlin, S. Altschul, Applications and statistics for multiple high-scoring segments in molecular sequences, Proc. Natl. Acad. Sci. USA 90 (1993) 5873–5877.

[9] S. Marsh, C.R. King, A.A. Garsa, H.L. McLeod, Pyrosequencing of clinically relevant polymorphisms, Methods Mol. Biol. 311 (2005) 97–114.

[10] I. Merelli, M. Landenna, L. Milanesi, Biological database access and integration using Web Services in GRID technology, in: Proceedings Bits, 2005, p. 60.

[11] I. Merelli, L. Milanesi, High performance implementation of BLAST using GRID technology, in: Proceedings Bits, 2005, p. 59.

[12] J.D. Parsons, S. Brenner, M.J. Bishop, Clustering cDNA sequences, Comput. Appl. Biosci. 8 (1992) 461–466.

[13] M. Ronaghi, M. Uhlen, P. Nyren, A sequencing method based on real-time pyrophosphate, Science 281 (1998) 363–365.

[14] S. Wiemann, B. Weil, R. Wellenreuther, J. Gassenhuber, S. Glassl, W. Ansorge, M. Bocher, H. Blocker, S. Bauersachs, H. Blum, J. Lauber, A. Dusterhoft, A. Beyer, K. Kohrer, N. Strack, H.W. Mewes, B. Ottenwalder, B. Obermaier, J. Tampe, D. Heubner, R. Wambutt, B. Korn, M. Klein, A. Poustka, Toward a catalog of human genes and proteins: sequencing and analysis of 500 novel complete protein coding human cDNAs, Genome Res. 11 (3) (2001) 422–435.

[15] J. Zhang, J. Dai, E. Zhao, Y. Lin, L. Zeng, J. Chen, H. Zheng, Y. Wang, X. Li, K. Ying, Y. Xie, Y. Mao, cDNA cloning, gene organization and expression analysis of human peptidylarginine deiminase type VI, Acta Biochim. Pol. 51 (4) (2004) 1051–1058.

**Gabriele A. Trombetti** received the Laurea (M.S.) degree with full marks (100/100 cum Laude) in Computer Science Engineering in 2004 at the University of Bologna, Italy with a thesis on Modelling and Validation Tools for QoS Enabled Distributed Systems and Applications. The thesis was developed at the DOC group at Vanderbilt University, Nashville, TN where he also worked as a Research Assistant in 2004 for researches on the same topics. In 1996 he graduated from the High school Liceo Scientifico Augusto Righi in Bologna, Italy with full marks (60/60). Since 2005 he attends the C.S. Engineering Ph.D program at the University of Bologna, Italy researching at the Institute of Biomedical Technology-National Research Council (Milan), currently in the scope of the European Specific Support Action project BioinfoGRID "Bioinformatics Grid Applications for Life Sciences".

**Ivan Merelli** received the Laurea degree in Biomedical Engineering from the Polytechnic University, Milan, Italy, in 2003 with a thesis about molecular surface modelling and analysis. His research activities concern the development of software for sequence based genomics and for structural Proteomics researches, with particular interest in protein–protein interaction. He works actively on the high performance implementation of Bioinformatics components using parallel programming and distributed platforms. Nowadays he works at Institute of Biomedical Technology of National Research Council at the FIRB-MIUR projects "Enabling Platforms for High-Performance Computational Grids Oriented Scalable Virtual Organizations-Grid.it" and "Laboratory of Interdisciplinary Bioinformatics Technologies-LITBIO".

**Luciano Milanesi** is currently researcher of the Italian National Research Council (CNR). Since 1988 he is head of the Bioinformatics Division of CNR-ITB. This division was established in 1989 to work in the framework of the Italian Human Genome Project. Initially, it focused on developing tools for the genome sequence analysis and prediction of gene structure in different organisms. More recently the work has shifted towards studying the functions of genes, promoter prediction, gene expression analysis and the development of databases integrated with sequence analysis tools. He has been principle investigator for the European Projects: TRADAT "TRAnscription Database and Analysis Tools", ORIEL "an Online Research Information Environment for the Life Sciences" and for the Italian CNR Finalized "Genetic Engineering" and FIRB Bioinformatics for Genomics and Proteomics and Enabling platforms for high-performance computational grids oriented scalable virtual organizations GRID-IT projects. He has published contributions in books and scientific publications in Bioinformatics.