Gridification of the SHMap Biocomputational Algorithm

P. Katsaloulis[1,2], E. Floros[1], A. Provata[2],
Y. Cotronis[1], T. Theoharis[1]

1. Department of Informatics and Telecommunications
National and Kapodistrian University of Athens,
15784 Athens, Greece

2. Institute of Physical Chemistry, National Center for the Scientific
Research "Demokritos", 15310 Athens, Greece

Abstract

In the current study we present a parallel statistical algorithm (SHMap), which distinguishes DNA regions which possibly carry protein coding information from non-coding regions. The code was parallelized using MPI and was deployed in a Grid testbed provided by the CrossGrid IST European Project. We tested the SHMap algorithm on mammalian chromosomes. The parallelization of the algorithm and the use of the Grid environment achieves significant speed-up compared to the sequential version of the algorithm, although the use of the Grid environment still presents some problems. Our algorithm is open source and freely distributed from our website.

1 Introduction

The need of biological problems for computational power has been steadily increasing over the past decade. New DNA sequences are decoded and stored in biological databases, while biocomputational tools are becoming more and more demanding and sophisticated.

Some of the most demanding and resource-consuming biological problems have to do with analyzing biosequences, in order to distinguish areas with specific biological meaning. These sequences may have DNA (nucleic acid) or protein (amino acid) origin and usually can be retrieved from databases like NCBI [9]. The annotation of DNA sequences requires hard and expensive experimental work. Alternatively, computational tools may be used based on similar statistics between different kinds of sequences. In the current study we will focus on the statistical analysis of DNA sequences aiming to recognize and annotate functional areas of the DNA from the degree of statistical homogeneity.

DNA chromosomes consist of areas with different composition. There are parts inside the chromosome (coding regions) which are able to code for proteins using the genetic code and their bases appear to be "statistically random", whereas other parts of the chromosome do not code for proteins (non-coding regions) and have more structural role[1, 2, 8, 7, 5]. In eucaryotic cells the coding areas are less common than the non-coding ones. Areas rich in coding regions are more homogeneous, in the sense that all possible combinations of nucleotides can be found, whereas areas poor in coding regions are less homogeneous, since some sequences (like poly-A) dominate these areas and some nucleotide combinations are missing. For this reason in homogeneous areas the distances between successive occurrences of the same oligonucleotide should be smaller than in less homogeneous areas.

Since coding regions appear more "random" than non-coding ones, if we consider all possible combinations of oligonucleotides of a fixed length and calculate their inter-distances, it should be possible to distinguish areas according to their homogeneity. Our method has been realized using a sequential algorithm (SHMap, Statistical Homogeneity Map Algorithm, [6]), by analyzing the homogeneity of long DNA sequences, or even entire chromosomes. Because of the length of a medium size eucaryotic chromosome, this algorithm is computationally demanding. However each oligonucleotide combination could be processed independently of the others; for this reason the performance of the SHMap algorithm can be drastically improved by parallelization. The code of our algorithm is open source under GNU GPL and can be freely downloaded from our website `http://www.bioinfo.gr`.

Bioinformatics is among the scientific disciplines that stand to benefit from Grid Computing capabilities [10, 3, 4]. Currently there does not exist a prevalent architecture for building Grid infrastructures. A trend is the agglomeration of various tools and APIs in order to provide what is considered a required set of Grid Services (job management, load balancing, security, data management, data replication etc). Such solutions are currently implemented in the context of various International Grid Projects like the European DataGrid (`http://www.edg.org`), CrossGrid (`http://www.crossgrid.org`) and EGEE (Enabling Technologies for E-sciencE, `http://www.cs.wisc.edu`). In this paper we have utilized the Grid Infrastructure of the CrossGrid IST Project Testbed in order to parallelize and gridify our application.

In the next section we outline the Grid infrastructure organization and implementation. The SHMap algorithm is described in section 3. Section 4 presents some applications of the algorithm and speed up tests, while the concluding section discusses the results and the problems we encountered while working with the Grid environment.

2 Grid Tools, Middleware and Testbed

2.1 Testbed Architecture

The CrossGrid testbed shares computing resources across sixteen European sites. The sites range from relatively small computing facilities in universities, to large computing centers. Sites are connected through the high-performance European network Géant. Each site includes at least the following computing and data storage resources: A *Computing Element (CE)* machine, which provides the entry point to the local processing farm, a farm with at least two *Worker Nodes (WNs)* which are responsible for the actual execution of a job, a *Storage Element (SE)* machine, which provides storage resources (disk space) as well as data and replica management services (GridFTP, RLS, etc), and a *User Interface (UI)* machine which provides an access point for local users to the testbed by including all the required client and development software to exploit the Grid capabilities.

The testbed provides a total of 104 CPUs and a storage volume above 4 TB. The integration of these resources is done by the following centralized services: The *Resource Broker (RB)* and the *Information Index (II)*. The RB is the heart of the workload management system. It receives job requests sent by users and finds computing resources (Computing Elements and respective Worker Nodes) suitable to run the jobs. Job management is performed by Condor-G (`http://www.cs.wisc.edu`). Every RB is paired with an II node which is a centralized LDAP repository of Grid resource information published by local CE and SE systems.

2.2 Middleware

The CrossGrid testbed utilizes the middleware that is being developed by CERN for the Large Hadron Collider (LHC) Particle Accelerator. This middleware known as LCG (LHC Computing Grid, `http://lcg.web.cern.ch`) is a collection of various Grid components, mainly the Globus Toolkit, Condor and PBS, deployed on top of a uniform Linux-based OS architecture (Red Hat 7.3). These components have initially been extended and enhanced in the context of European DataGrid project, have been adopted and further enhanced by CrossGrid and currently are also used by the EGEE (Enabling Grids for E-Science and Engineering) project. The main Grid capabilities that LCG provides are: job and workload management, replica and data management, information indexing and retrieval, job and testbed monitoring. Jobs are described using the Job Description Language (JDL) which is based on Condor ClassAdds.

2.3 Application Gridification

The main purpose of gridifying SHMap was to split the workload by parallelizing the execution of the algorithm, thus achieving execution speed-up. We have opted to parallelize the algorithm itself; this approach is the most efficient. The code is parallelized using the MPI library. The disadvantage of this solution is that the source code has to be altered in order to include the required MPI calls and apply the parallelization strategy. The advantage is that this solution can easily scale. All the required interprocess synchronization and communication is handled by MPI. For instance, the last step which produces the final result histogram file can be performed using MPI's collective communication or parallel I/O. Moreover the application is submitted as a single job irrespective of the number of processes, thus the Grid overhead is small and constant.

3 SHMap Biocomputation Algorithm under Grid environment

The SHMap algorithm distinguishes areas of the genome according to their statistical homogeneity. This is achieved by marking areas for which the distance between consecutive occurrences of any random oligonucleotide is larger than a given threshold . The algorithm consists of a repeatable procedure which marks areas where distances between two consecutive oligonucleotides are well above a statistically defined threshold. Computed areas of higher homogeneity has been found to be probable coding areas, when compared with experimental biological data. The structure of the SHMap algorithm has been described in previous work [6].

We have implemented the necessary changes to run the SHMap algorithm under MPI in data parallel mode. Each process is responsible for a different set of oligonucleotides. In SHMap searches the whole chromosome has to be traversed. Thus splitting the chromosome among multiple processes would not have been an efficient solution, as it would involve considerable 'border' checking and therefore communication between the parallel processes. The parallelization of SHMap follows a simple Single Program Multiple Data (SPMD) coarse grain strategy. All processes hold the entire chromosome. In the preparation phase each process computes the set of oligonucleotides that it will handle.

Synchronization is performed using the `MPI_Reduce` function on the master process, which is responsible for gathering the results and producing the unified statistical map. Upon completion of the reduction operation, the master process proceeds alone, performs the normalization of the map and stores the results in the output file.

The job is submitted to the Grid middleware using an appropriate JDL file. Using JDL we define the input sandbox comprised of the executable itself and the input genome file. As execution parameters we pass the name of the input file and the range of oligonucleotide sequences that will be collectively searched. Notice that the application can also be used with the first parallelization approach since the range of oligonucleotides is parametrically defined at execution time.

We define the total number of processes that will be created and give requirements regarding the execution environment of the application in JDL. Each process is given a complete copy of the genome and requires an equal amount of space for the result array. This requirement is passed in the JDL specification. Before submitting the job we can test for availability of a sufficient cluster using the *edg-job-list-match* LCG workload management command.

The job is submitted using *edg-job-submit.* The programmer then periodically checks the progress of the job execution using *edg-job-status* command. The parallel application is executed as a batch job since there is no requirement for interactivity. When the application has finished the user retrieves the result file using *edg-job-get-output* command.

m	n	t_{total}	t_{comp}	S_{total}	S_{comp}	pc_{total}	pc_{comp}
6	1	3982	3735	1	1	3982	3735
6	2	2701	2453	1.47	1.52	5402	4906
6	4	2231	1983	1.78	1.88	8924	7932
6	8	1554	1330	2.56	2.81	12432	10640

Table 1: Benchmark results of the SHMap algorithm when run in the Grid subsystem. m is the length of the oligonucleotide, n is the number of processors, t is run time in seconds. '*total*' stands for the parallel execution time (from submission of Grid request to run, up to the end of execution), '*comp*' is the execution time only. 'S' and 'pc' stands for speedup and parallel cost respectively.

4 Application to biological data

As already noted, the SHMap algorithm annotates areas of the chromosome depending on their degree of homogeneity. Genome areas where the output of the algorithm is high possess a low degree of homogeneighty and vice versa. Since coding regions are usually homogeneous, the SHMap output can be considered as an indicator for coding regions.

The parallel algorithm has been successfully tested on mammalian chromosomes of lengths ranging from 10M base pairs to 100M base pairs. As an example we present here the application of SHMap on chromosome 17 of *Mus musculus* (obtained from `ftp://ftp.ncbi.nih`). The oligonucleotide length chosen (m) is of critical importance to the results. Although the length of the chromosome (c) is large, the frequency of appearance of random oligonucleotides decreases dramatically as m increases. We have opted for oligonucleotides of sizes 6 (hexaplets), since this size produces clearly distinguishable areas with higher or lower homogeneity.

We have created the appropriate JDL files to perform the Grid tests for this system. The numbers of processors used were 1, 2, 4 and 8. Since timing was an important issue, we forced the resource broker to use only a specific cluster, so that the measurements are comparable to each other. We measured the total Grid execution time and the computational time (without the Grid overhead). The former corresponds to the duration between the submission of the job description to the resource broker to the successful completion of the application, while the latter measures only the pure application run time, without any Grid system overhead. The results appear in Table 1.

The speed-up corresponds to the sequential time over the corresponding parallel time:

$$S = \frac{t_{sequential}}{t_{parallel}} \tag{1}$$

We set $t_{sequential}$ equal to the execution time of the algorithm on one processor of the same system. The parallel cost was also computed:

$$pc = n * t_{parallel} \tag{2}$$

where n is the number of processors.

The speedup and the parallel cost increase monotonically. The reason for the increase in the parallel cost is the large amount of system overhead (relative to the computational cost) in order to execute parallel instances of the application. Moreover, there are parts of the algorithm which are executed sequentially and this cost (mainly gathering and combining the results) is proportional to the number of processors. We have to note that the unknown bases had to be removed from the DNA sequence (they were less than $1 : 10^3$ in this example), since these bases do not match with any given pattern and produce false high peaks.

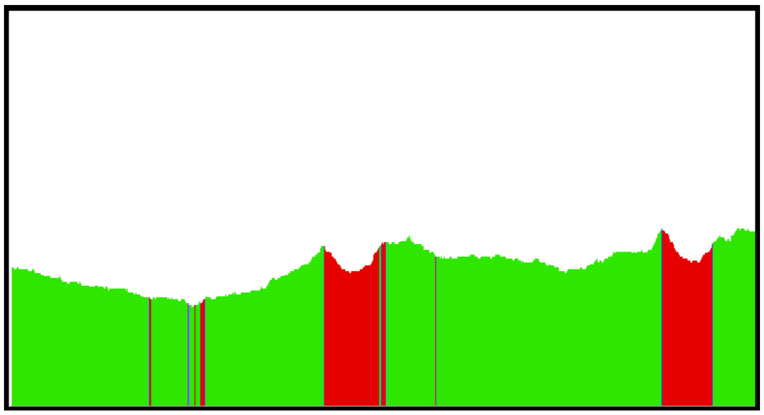

Figure 1: Statistical Homogeneity Map (SHMap) for the annotated chromosome 17 of *Mus musculus*. The visible region is between the bases $12040000 - 12070000$. The size of oligonucleotides is $m = 6$. The lower the value on the vertical axis, the higher the homogeneity (and thus high possibility for coding sequences) and vice versa. Red color marks "CDS" areas, as found in the GeneBank genome file, green marks non-coding areas and blue marks areas with both coding and non-coding parts.

4.1 Biological evaluation of the results

In order to test the validity of the produced results of *Mus musculus* chromosome 17, we compare them to the actual biological information gathered from a biological database. The annotated chromosome 17 of *Mus musculus* from NCBI database has been used. Figure 1 shows such a comparison for the area $12040000 - 12070000$. We show the produced SHMap of this chromosome together with the coding sequence areas (CDS) on the same graph. The higher the values on the vertical axis the lower the homogeneity of the corresponding area of the chromosome. Red color denotes CDS and green color the rest of the chromosome, as obtained from the GeneBank file.

We note that this approach does not perform a detailed distinction between coding and non-coding regions, but it can be used to focus the researcher on areas which have a high probability of being CDS. It is thus able to highlight areas of potential importance, since the structure and statistical characteristics of the DNA sequence are related to its functionality.

5 Discussion

The gridification of the SHMap algorithm achieved a practical speed-up on the application run time, compared to the sequential version. Moreover its scaling is rather good, especially when computationally intensive parameters are used in the algorithm. Since we used a small chromosome, compared to other mammalian chromosomes, it is expected that the algorithm will behave at least as well for longer eucaryotic chromosomes. The Grid system overhead is also rather small and represents an unnoticeable portion of the total execution time, for computationally intensive applications.

Apart from the benefits, we also faced some problems in the gridification of the system. Since the middleware is in principle optimized towards batch applications, the user did not have immediate feedback on the status of the requests. Sometimes a sample run crashed and the only way for the user to be informed was to constantly ask the resource broker for the status of the submitted job. Even worse, there were cases where the job had been terminated for no apparent reason and the feedback messages to the user were rather sparse and non-informative. Most of the time, only a simple message informed the user that the application terminated with a specific exit code, which was irrelevant to the application itself but was specific to the Grid middleware.

Another problem had to do with the functionality of the Grid testbed itself. Due to current middleware architecture restrictions, the Information Index cannot keep detailed information about all WNs in a farm. Site Administrators have to choose a ‘representative’ WN and publish its configuration information as if it applies to all WNs in a site. There are situations where sites do not properly advertise themselves, by providing outdated information about their status or their abilities. This had the side effect that the resource broker sent a job to a specific cluster which advertised more processors than the actual, with the result of constantly rejecting or aborting the job for reasons not obvious to the end-user. Since this application was memory demanding, we have used some memory constrains in order to send the job to the appropriate cluster. Unfortunately there were clusters where not all nodes had the same amount of memory and the system advertised the average or the maximum memory instead of the minimum. The effect was that the algorithm run well on some nodes of the cluster and broke on others, making it impossible to gather the data in the last phase of the algorithm.

Despite the current practical problems, we believe that as Grid technology matures, it will become a more attractive computational tool for demanding scientific applications.

6 Acknowledgments

The authors acknowledge financial support from the “HERAKLEITOS” research fellowships program 70/3/7156 with emphasis on basic research, Hellenic Ministry of Education.

References

[1] Y. Almirantis and A. Provata. Long- and short-range correlations in genome organization. *Stat. Phys.*, 97:233, 1999.

[2] A. Arneodo, E. Bacry, P.V. Graves, and J.F. Muzy. Characterizing long-range correlations in DNA sequences from wavelet analysis. *Phys. Rev. Lett.*, 74:3293, 1995.

[3] V. Breton, R. Medina, and J. Montagnat. DataGrid, prototype of a biomedical Grid. *Methods MIMST*, 42:143, 2003.

[4] M. Cannataro, C. Comito, F.L. Schiavo, and P. Veltr. Proteus, a grid based problem solving environment for bioinformatics: Architecture and experiments. *IEEE Computational Intelligence Bulletin*, 3:7, 2004.

[5] P. Katsaloulis, T. Theoharis, and A. Provata. Statistical distributions of oligonucleotide combinations: applications in human chromosomes 21 and 22. *Physica A*, 316:380, 2002.

[6] P. Katsaloulis, T. Theoharis, and A. Provata. Statistical algorithms for long dna sequences: oligonucleotide distributions and homogeneity maps. *Scientific Programming*, 13:177, 2005.

[7] R.N. Mantegna, S.V. Buldyrev, A.L. Goldberger, S. Havlin, C.K. Peng, M. Simons, and H.E. Stanley. Linguistic features of noncoding DNA sequences. *Phys. Rev. Letts.*, 73:3169, 1994.

[8] A. Provata and Y. Almirantis. Scaling properties of coding and non-coding DNA sequences. *Physica A*, 247:482, 1997.

[9] K. D. Pruitt and D. R. Maglott. RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Research*, 29:137, 2001.

[10] R. D. Stevens, A. J. Robinson, and C. A. Goble. myGrid: personalised bioinformatics on the information grid. *Bioinformatics*, 19:1302, 2003.