Applying the ISAM Architecture for Genetic Alignment in a Grid Environment

Alberto E. Schaeffer Filho, Lincoln L. de Morais, Rodrigo A. Real, Luciano C. da Silva, Adenauer C. Yamin, Iara Augustin, Cláudio F. R. Geyer

[1]Informatics Institute
Federal University of Rio Grande do Sul
Bento Gonçalves Av. 9500, Porto Alegre, RS, Brazil

{egon, lincoln, rreal, lucc, adenauer, august, geyer}@inf.ufrgs.br

Abstract. *In this paper we present a practical experiment using an application to solve the genetic sequence alignment problem implemented in the grid environment provided by ISAM [1]. The ISAM architecture aims to provide an integrated solution, from development to execution, for general-purpose pervasive applications, combining techniques proceeding from context-aware, mobile and grid computing. Here, we present an overview of the genetic sequence alignment problem and the developed solution to solve it, as well as the obtained results in a multi-institutional execution of the application.*

1. Introduction

This paper outlines the design and execution of an application for genetic sequence alignment in the grid environment provided by ISAM. The designed application was named GeneAl (GeneAl stands for *Gene*tic *Al*ignment). There are several algorithms to perform genetic sequence alignment in different levels of sophistication, varying from the basic algorithm used in the implementation of the program BLAST [Altschul et al., 1990] to the Smith-Waterman algorithm [Smith and Waterman, 1981] that can detect weak similarities between sequences separated by a large evolutionary distance [Bundschuh, 2000]. The developed application implements a variant of the Smith-Waterman algorithm.

The ISAM project [ISAM, 2004] aims to explore alternatives to build and execute general-purpose pervasive computing applications, which to ISAM are distributed, mobile, and context-aware by nature. In our vision, the infrastructure to provide a pervasive environment encompasses the grid computing paradigm [Yamin et al, 2003], and so, our solution can also be employed in this research area. The components of the ISAM architecture were used to design and to run the GeneAl application.

This paper is organized as follows: Section 2 shows an overview of the ISAM architecture; Section 3 introduces the genetic sequence alignment problem; Section 4 presents our solution to the problem using the ISAM infrastructure; Section 5 describes some obtained results; finally, Section 6 presents the concluding remarks.

2. ISAM Architecture Overview

The ISAM software architecture provides an integrated solution, from language to execution environment. ISAM is organized in layers (Figure 1): the highest layer of the architecture corresponds to the distributed mobile adaptive application that is based on

[1]The ISAM project is partially supported by CNPq/SEPIN/FINEP foundations.

the ISAMadapt's abstractions [Augustin et al., 2002]. The intermediate layer represents the ISAM's middleware, which is responsible for supporting the execution of the application [Yamin et al, 2003]. Finally, the lowest layer corresponds to the hosting system, which should support the Java platform.

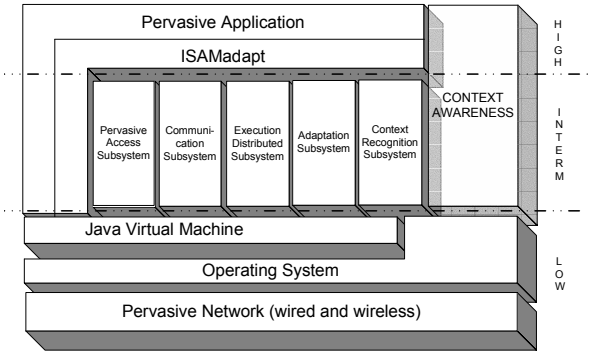

Figure 1: ISAM architecture

2.1. ISAM Cellular Organization

The ISAM architecture makes available a pervasive computing environment called ISAMpe (ISAM pervasive environment), which has a cellular organization. Hosts sharing an interconnection medium are grouped into network segments. Network segments are grouped into computing cells, which can be interpreted as institutional boundaries in a multi-institutional environment scope. Each cell has associated with itself a *Cell Information Base* (CIB), which controls all static and dynamic information originated from the internal hosts of the cell. This cellular organization is similar to virtual organizations in grid computing [Foster et al., 2001]. The management process of a cell is autonomous with respect to other cells in the system and each cell is responsible for managing and providing access to local computing resources. Moreover, a cell has a dynamic set of other known cells in the ISAMpe, which compose its neighborhood.

2.2. The Service-based Middleware

The ISAM's middleware is based on two perspectives: (i) physical management, by providing services to control the physical medium where the processing will take place; and (ii) support for execution of pervasive applications, by providing services and abstractions necessary for adapting the application to context changes along the execution path.

In this approach, a middleware minimum core has its functionality extended by pluggable services. Service loading is done on demand and, moreover, is adaptive. Thus, we can use implementations of a service that are better tuned to each device while we reduce resource consumption by loading only services that are effectively necessary. This functionality is customizable in a per-node basis, and an execution profile defines the set of services that should be activated in a node, assigning to each service an implementation.

3. The Genetic Sequence Alignment Problem

Genetic sequence alignment refers to the operation of nucleotide comparison that tries to find local similarities using biosequence databases. By a scoring scheme over compared sequences of nucleotides it is possible to identify which are the most similars. In this paper, we implemented a variant [Meidanis and Setúbal, 1994] of the traditional, and still broadly used, Smith-Waterman algorithm for global genetic sequence alignment [Smith and Waterman, 1981]. The idea of the algorithm is to find out the best alignment

between two sequences. The sequences to be compared need not have the same length, but in this case it is necessary to insert gaps (-) in arbitrary places along the sequences.

The scoring scheme is done as following (considering one sequence above another): *(a)* a column with the same nucleotides is given +1 point, *(b)* a column with distinct nucleotides is given -1 point and *(c)* a column that contains a gap is given -2 points. This set of values tries to benefit columns with the same nucleotides and penalize columns with distinct nucleotides or containing a gap. The best alignment between two sequences will be the one which has the best score considering all columns from the compared sequences. Figure 2 shows the best alignment between sequences S_1 and S_2. In this example, the differences between the two sequences are one additional T in S_2, and the unmatched column (the third from right to left). Thus, there are eight columns with the same nucleotides, one column with distinct nucleotides and one column containing a gap. The score of the alignment is $8 \times (1) + 1 \times (-1) + 1 \times (-2) = 5$.

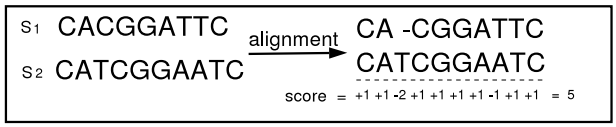

Figure 2: Alignment between two sequences

There are several instances of problems related to comparing genetic sequences, and in this paper the interest is, given some determined sequence of nucleotides, to compare it with thousands of sequences and to find the N most similar sequences in distributed databases of biosequences. As the algorithm is only able to calculate the score between two sequences, to find the N best alignments between the given sequence and those contained in a database will require that it must be applied upon all peers <*entry_sequence*, *database_sequence[i]*>, where *database_sequence[i]* is the i^{th} biosequence in the database. As biosequence databases can contain thousands of stored sequences, this task can be pretty hard and time consuming, and a distributed infrastructure to help solving this problem is very recommended.

4. GeneAl: A Grid Approach for Genetic Sequence Alignment

Using the ISAM infrastructure to support the execution of grid applications we built an application, named GeneAl, that addresses the problem of finding out the best N alignments among biosequences spread in distributed databases. The GeneAl application is described in the next sections.

4.1. Distribution Approach

In order to allow new resources to be dynamically added, the problem was modeled following a *master-worker* approach [Yamin et al, 2002], associating a master object along with each biosequence database. A master is responsible for managing a number of workers that will perform the searching of the best aligned sequences over a database. At top of individual databases, there is a management layer that gathers the partial results computed in each master to produce the final result. Figure 3 depicts such organization.

A master is responsible for splitting a biosequence database into a number of jobs, for distributing and keeping control of it and for generating the final result for a given execution cell. The workers get jobs from the master and process them. After the processing of a job has been finished, the worker will be responsible for sending the result back to the master and getting another job.

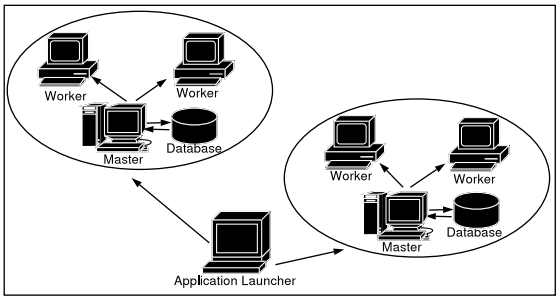

Figure 3: Distributed infrastructure with two execution cells

4.2. Perspectives of Adaptive Execution

GeneAl does not require that the databases must be located at a single site. This is the first degree of adaptation provided by the ISAM architecture to GeneAl: the Discoverer service of ISAM allows the user to find the available databases at run-time for a given subject. Someone can then select among which of them the processing should take place.

The second degree of adaptation provided by ISAM comes from its Scheduler service. The Scheduler is responsible for selecting the best nodes to host the workers.

Finally, the third degree of adaptation comes from the data partitioning strategy (presented in Section 4.3). ISAM allows the worker throughput constraints to be modeled as context elements. This abstraction is supported by the context recognition subsystem of the middleware, which notifies the application components whenever it is observed behavior changes, and allows them to adapt by acting over the size of the jobs.

4.3. Data Partitioning Strategies

The size of a job can be defined statically before the beginning of the execution, or it can be adjusted dynamically along the execution according to the capacity/load of the host where the work is being done. This last approach is named *time goal*, and the jobs are resized according to the accomplishment of a given goal in a determined period of time. The size of the first job is an estimated value, but the size of the next job is decided by the performance of the previous job, using the relation about the expected time and the real elapsed time of the previous execution (reducing or increasing the number of biosequences per job).

5. Experimental Results

The following research centers participated in the grid experiment using the GeneAl application: Federal University of Rio Grande do Sul (UFRGS), University of Passo Fundo (UPF), Federal University of Santa Maria (UFSM) and Catholic University of Pelotas (UCPEL). The computational resources made available are showed in Table 1.

Each research center, along with its computational power, constitutes an execution cell. Besides, each cell has a database of biosequences and its size varies between 36 MB and 46 MB, totalizing at about 174 MB of data to be processed (approximately 435.000 biosequences of 300 nucleotides in average). The genetic data used in the experiment was obtained from human sequenced chromosomes picked up from the Internet [National Center for Biotechnology Information, 2004].

We evaluated three strategies for data partitioning: *(a) static/small* jobs (200 biosequences sent to each worker a time), *(b) static/big* jobs (15.000 biosequences sent to each worker a time) and *(c) dynamic* jobs (resized dynamically according to the performance of the previous job). The *dynamic* approach adapts the size of the delivered jobs during

Research center	Processing nodes	Network (Ethernet)
UFRGS	2 Pentium IV 2.4 GHz 2 Pentium III 1 GHz 1 Athlon 1 GHz 1 UltraSparc IIi 333 MHz	100 Mb/s
UPF	6 Athlon 1 GHz	100 Mb/s
UFSM	6 Pentium IV 2.4 GHz	100 Mb/s
UCPEL	6 Pentium III 800 MHz	10 Mb/s

Table 1: Computational resources used in the grid experiment

the execution, based on the time goal parameter. Its objective is to increase the medium parallelism and to decrease the delay in the synchronization barrier, so reducing the total execution time.

Our experiment presents the execution of GeneAl in different scenarios simultaneously, considering the relation between the data partitioning and specific hardware and network characteristics inside each cell. The processing nodes were dedicated to the application, which means that there were no other users or applications using the nodes along the executions. Figure 4 shows the obtained results. The total execution time is determined by the slower execution cell.

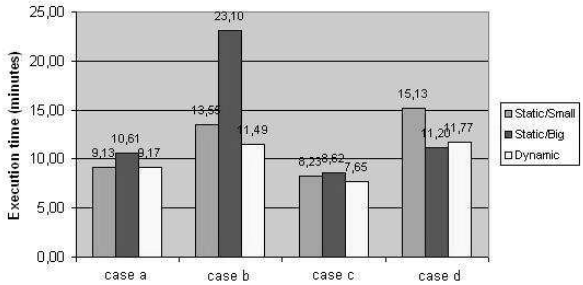

Figure 4: Data partitioning strategies in different scenarios

The UPF execution cell (case *a*), which is characterized by hardware homogeneity and a 100 Mb/s network bandwidth, showed that the *static/small* and *dynamic* data partitioning presented similar performances. In spite of hardware homogeneity, the synchronization barrier is observed due to the fact that the number of *static/big* jobs was not multiple of the number of processing nodes inside the cell.

The UFRGS execution cell (case *b*) is characterized by hardware heterogeneity and a 100 Mb/s network bandwidth; under these conditions, the *dynamic* data partitioning (using a *time goal*) obtained better performance over the static approaches. Here, the high network bandwidth did not injured drastically the *static/small* approach, but the potential synchronization barrier among the heterogeneous processing nodes caused a significant performance lost in the *static/big* strategy.

The UFSM execution cell (case *c*), which is characterized by hardware homogeneity and a 100 Mb/s network bandwidth, presents an environment very similar to case *a*. It showed that the *static/small* and *dynamic* data partitioning presented similar performances. In this setting, the high network bandwidth did not motivated the use of the *static/big* data partitioning that, associated with the potential synchronization barrier among the nodes, presented the worst performance (again, the number of *static/big* jobs was not multiple of the number of processing nodes inside the cell).

Finally, the UCPEL execution cell (case *d*), which is characterized by hardware

homogeneity and a 10 Mb/s network bandwidth, obtained a better performance using the *static/big* approach. Here, the number of *static/big* jobs was multiple of the number of processing nodes inside the cell. This fact, associated with the hardware homogeneity, eliminate the synchronization barrier inside the cell. The network characteristics make the *static/small* and the *dynamic* data partitioning strategies the worst options due to the high latency in the job distribution among the processing nodes.

6. Concluding Remarks

In this paper we briefly described some key features of the ISAM architecture and presented its use in a practical grid experiment. Motivated by the genetic sequence alignment problem we developed an application supported by the services that compose the ISAM's middleware.

About the obtained results, we analyzed different aspects associated with the distribution of work among the nodes and its relation to resource and network heterogeneity that compose the environment. The genome processing in a dynamic environment as the one presented here has a high degree of potential scalability and platform independence, enabling the use of more processing nodes, as they became available in the environment, to reduce the total execution time of the application. The several degrees of adaptation provided by the ISAM architecture allow the use of these resources in a more effective way. Specifically, the accomplished experiment demonstrated that the infrastructure provided by ISAM can be successfully applied to grid computing.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410.

Augustin, I., Yamin, A., Barbosa, J., and Geyer, C. (2002). ISAM - a software architecture for adaptive and distributed mobile applications. In *7th IEEE Symposium on Computers and Communications*, Taormina, Italy. IEEE Computer Society.

Bundschuh, R. (2000). An analytic approach to significance assessment in local sequence alignment with gaps. In *Proceedings of the fourth annual international conference on Computational molecular biology*, pages 86–95. ACM Press.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222.

ISAM (2004). ISAM project. http://www.inf.ufrgs.br/ isam.

Meidanis, J. and Setúbal, J. C. (1994). *An introduction to computational biology (portuguese)*, chapter 3, pages 15–54. Recife, Brazil.

National Center for Biotechnology Information (2004). Human reference sequence. http://www.ncbi.nlm.nih.gov/genome/seq/.

Smith and Waterman (1981). Comparison of biosequences. *Advances in Applied Mathematics*, 2:482–489.

Yamin et al, A. C. (2002). A framework for exploiting adaptation in high heterogeneous distributed processing. In *XIV Symp. on Computer Architecture and High Performance Computing*, Vitória, Brazil. IEEE.

Yamin et al, A. C. (2003). Towards merging context-aware, mobile and grid computing. *The International Journal of High Performance Computing Applications*, 17(2):191–203.