

Bioinformatics on a Heterogeneous Java Distributed System

Andrew J. Page, Thomas M. Keane and Thomas J. Naughton

Department of Computer Science,

National University of Ireland,

Maynooth, Ireland.

andrew.j.page@nuim.ie, tkeane@cs.nuim.ie, tom.naughton@nuim.ie

Abstract

A programmable Java distributed system, which utilises the free resources of a heterogeneous set of computers linked together by a network, has been developed. The system has been successfully deployed on over 200 computers, which were distributed over a number of locations, and has been successfully used to process bioinformatics, biomedical engineering, and cryptography applications. We present two bioinformatics applications, DSEARCH, which performs sensitive database and DPRml which performs distributed phylogeny reconstruction by maximum likelihood.

1. Introduction

There are many problems in modern scientific research which require large computational resources. The traditional way of providing these computational resources was to build large multi-processor supercomputers. However the costs of these dedicated systems can be prohibitive. A number of distributed computing systems have been built to process computationally intensive problems, without the traditional high costs associated with dedicated parallel hardware and clusters, by utilising the spare clock cycles of ordinary computers, which would otherwise have not been utilised to their full potential. Currently there are a few notable distributed computing platforms such as SETI@home [1], United Devices [17], and distributed.net [5]. These systems are generally referred to as web computing systems, where their resources are distributed across the Internet.

SETI@home [1] has harnessed the spare resources of over 4 million donor machines, and has demonstrated the potential of such heterogeneous distributed systems. There are however limitations with current distributed systems. Many are platform dependent [1, 5, 17], which incurs higher maintenance and development costs. Others are hard coded to perform only a single task [1, 13].

We have designed a programmable, platform-independent, heterogeneous, distributed system [12] using Java. Two bioinformatics applications have been successfully developed to run on the distributed system. The system has been deployed on over 200 PCs. The software we have developed is open source and available under the GNU GPL license free of charge.

The rest of the paper is organised as follows. In Sect. 2, we provide an overview of the distributed system. In Sect. 3 describe two bioinformatics applications, DSEARCH and DPRml, which have been developed for the system. We conclude in Sect. 4 and note future directions for our research.

2. Java distributed system

The foundations for the distributed system used in this paper, were laid in the Java distributed computation library [7] and its extension [12]. It arose out of a need for a scalable distributed system, that was easy to program and deploy.

2.1. Design

There are three distinct parts to the system. These are the client, the server, and the remote interface (fully described in [12]). The user is required to extend two classes to create a `Problem` to run on the system. The `DataManager` class (in the server) specifies how the problem is to be partitioned into units of work and the intermediate results put together, facilitating the computation of more generalisable problems, rather than being limited to trivially parallelisable problems [1, 13, 17]. The `Algorithm` class (in the client) specifies the actual computation.

The users of the system do not need any knowledge of the topology or workings of the system in order to submit problems and get their processed results back. They just provide a `DataManager`, an `Algorithm`, additional required classes, and data to be processed (if required), to cre-

ate a self contained `Problem` object. A `Problem` object provides a predefined interface to the functionality that the user has provided in their `DataManager`.

2.2. Communication

Communication within the system is based on a combination of Java RMI and ordinary Java sockets. RMI is a built-in facility in Java that allows one to interact with objects that are actually running in Java Virtual Machines (JVMs) on remote hosts on a network, and avoids the need for the designer to worry about low level communication issues. Data files, which may be large, are transmitted using ordinary sockets, which is more efficient than RMI.

3. Applications

We have deployed the distributed system in our university across 3 locations, by running it as a low priority background service in a number of computing laboratories, consisting of approximately 200 desktop PCs of various modest specifications (Pentium IIs up to Pentium IVs running assorted versions of Windows and Linux OSs) and on every node of an IBM Linux cluster (32 Dual PIII 1 GHz nodes) with all machines connecting via a 100 Mbit/s network to a single server (Pentium III 500 MHz). The distributed system has been running for over 3 year, in various different forms, and numerous different scientific applications have been create to run on the system.

The distributed system has so far been used to process problems from areas such as bioinformatics, biomedical engineering, and cryptography. The rest of this section will describe two bioinformatics applications and present performance results for each.

3.1. DSEARCH: sensitive database searching using distributed computing

Database searching for similar sequences is one of the fundamental tasks in bioinformatics. The two most rigorous search algorithms are the Needleman-Wunsch [10] and Smith-Waterman [14] algorithms. However for large databases it is not feasible to perform searches using these algorithms. Many heuristic search algorithms have been developed but these reduce the sensitivity of a search and can fail to detect certain matches. When given the choice, most biologists would prefer to use the more rigorous algorithms for their searches.

One way to significantly reduce the runtime of long searches is to parallelise the search process across multiple processors. Many approaches to parallelising database searching have been investigated because database searching is both computationally intensive and easily paral-

lised. However the overriding problem with many of these programs is that specialised parallel hardware and software is often required, making these programs either prohibitively expensive or simply too complicated to set up. DSEARCH is a fully cross-platform parallel database search program that does not require any specialised parallel hardware or software.

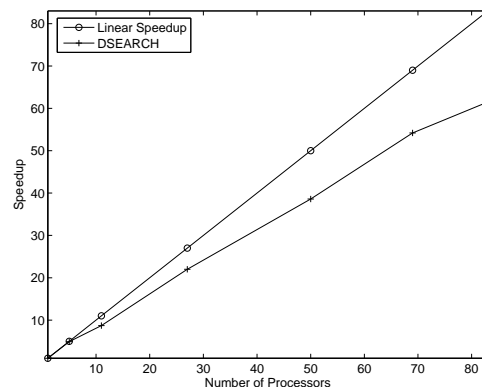


Figure 1. Speedup achieved by DSEARCH over a network of 83 semi-idle machines.

DSEARCH [8] operates over a client-server topology and the search is parallelised by splitting the database into dynamically sized units that are subsequently searched on the donor machines. The parallel granularity is dynamically controlled during each search to match the processing abilities of the current set of donor machines. The user edits a straightforward configuration file to tailor their computation and chooses one of the built-in search algorithms [4, 10, 14]. The inputs to the program are a FASTA database file, a FASTA query sequences file, a scoring scheme, and a configuration file.

Figure 1 shows how DSEARCH scales with increasing numbers of processors. A general comparative metric, like speedup for homogeneous systems, does not exist yet for heterogeneous systems, so for the speedup graph in Fig. 1 we used a laboratory of 83 homogeneous processors (Pentium III 1GHz).

3.2. DPRml: Distributed Phylogeny Reconstruction by Maximum Likelihood

One of the great challenges of molecular biology is the completion of the tree of life [3]. The massive accumulation of genomic data has led to increased interest in the production of large and accurate phylogenetic trees. However the decision problem associated with searching for the best

tree from a set of taxa is NP-hard [2]. Therefore it is not feasible to perform an exhaustive search of the tree space for trees of a non-trivial size. Maximum likelihood (ML) evaluation has been widely acknowledged as one of the most accurate techniques for reconstructing phylogenies.

In an effort to construct large and accurate phylogenetic trees, while still keeping overall processing times reasonable, a number of researchers have developed parallel ML programs that utilise the stepwise insertion approach [15, 16]. One of these programs [15] also employs some simple distance based heuristics to try to reduce the number of generated trees. These programs have been successful in speeding up phylogenetic computations but the overriding problem with these programs is that specialised parallel hardware and software is often required. For most researchers, this can make these programs either prohibitively expensive or simply too complicated to set up. Furthermore these programs are often implemented in a platform specific language which imposes a restrictive limit on the numbers and types of machines that can be used in a parallel computation. It should also be noted that some of these earlier parallel programs only allowed the user to choose from a very limited number of DNA substitution models, which often leads to a poor model fit resulting in sub-optimal trees.

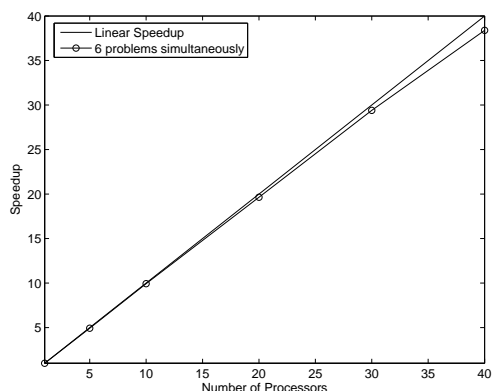


Figure 2. Speedup achieved over 50 taxa dataset with 6 problems running simultaneously.

We have identified the suitability of phylogenetic analysis to large-scale heterogeneous distributed computing and have developed a fully cross-platform distributed application, DPRml [9], which we believe to be one of the most general and powerful likelihood-based phylogenetic tree building programs currently available. DPRml is, to our knowledge, the first distributed phylogenetic tree building

program to satisfy each of the three requirements outlined above. The generality of our program is demonstrated by the fact that DPRml, written in Java, can run on virtually any architecture and operating system simultaneously while only using the spare clock cycles of donor machines. No specialised computer hardware is required, and no expense is incurred if idle computing resources are harnessed. This would not be as straightforward for a distributed application written in a native language because the application would have to be compiled for each particular architecture and operating system. We have demonstrated the ease of use and platform heterogeneity of DPRml with experiments that utilise the spare computing resources of several different architectures and operating systems simultaneously. The user has a very straightforward configuration file with which to tailor the computation and can choose from one of the most extensive ranges of DNA substitution models currently available.

Our performance analysis (see Fig. 2) demonstrates how effective DPRml can be for speeding up the process of constructing large phylogenetic trees. Biologists generally run stochastic algorithms, such as DPRml, a number of times, thus Fig. 2 shows the efficiency of running 6 instances of the application in parallel. DPRml is a staged computation so running a single instance of the application will result in clients becoming idle whilst waiting for stages to be completed.

DPRml implements an already proven tree building algorithm [11, 16] and uses the popular Phylogenetic Analysis Library (PAL) v1.4 [6] for all its likelihood calculations.

4. Conclusion

We have examined a variety of other existing distributed systems [1, 5, 13, 17] and have produced a system that combines the advantages offered by each of these existing systems and overcomes many of the disadvantages of each system. Central to our success was the use of Java to implement the distributed system. Our distributed system is capable of being deployed in a typical heterogeneous Internet/intranet environment. We presented two bioinformatics applications, DSEARCH and DPRml, which have successfully utilised the spare clock cycles of over 200 PCs.

Our future work will concentrate on enhancing the adaptive scheduling strategy in the distributed system and we will be creating more distributed bioinformatics applications.

The software is freely available under an open source GNU GPL licence from the system homepage located at <http://www.cs.may.ie/distributed/>

5. Acknowledgement

Support is acknowledged from the Irish Research Council for Science, Engineering, and Technology, funded by the National Development Plan.

References

- [1] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Massively distributed computing for SETI. *Computing in Science & Engineering*, 3(1):78–83, Feb 2001.
- [2] H. Bodlaender, M. Fellows, and T. Warnow. *Two strikes against perfect phylogeny*, volume 623, pages 273–283. Springer-Verlag, NY, USA, 1992.
- [3] K. Crandall and J. Buhay. Genomic databases and the tree of life. *Science*, 306(5699):1144–1145, 2004.
- [4] M. Crochemore, G. Landau, and M. Ziv-Ukelson. A subquadratic sequence alignment algorithm for unrestricted scoring matrices. *SIAM Journal of Computing*, 32(6):1654–1673, 2003.
- [5] Distributed.net. <http://www.distributed.net>.
- [6] A. Drummond and K. Strimmer. Pal: An object-oriented programming library for molecular evolution and phylogenetics. *Bioinformatics*, 17:662–663, 2001.
- [7] K. Fritsche, J. Power, and J. Waldron. A Java distributed computation library. In *2nd International Conference on Parallel and Distributed Computing Applications and Technologies*, pages 236–243, Taipei, Taiwan, July 2001.
- [8] T. M. Keane and T. J. Naughton. DSEARCH: sensitive database searching using distributed computing. *Bioinformatics*, page bti163, 2004.
- [9] T. M. Keane, T. J. Naughton, S. A. A. Travers, J. O. McInerney, and G. P. McCormack. DPRml: distributed phylogeny reconstruction by maximum likelihood. *Bioinformatics*, page bti100, 2004.
- [10] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.
- [11] G. J. Olsen, H. Matsuda, R. Hagstrom, and R. Overbeek. Fastdnaml: A tool for construction of phylogenetic trees of dna sequences using maximum likelihood. *Computer Applications in the Biosciences*, 10:41–48, 1994.
- [12] A. Page, T. Keane, and T. J. Naughton. Adaptive scheduling across a distributed computation platform. In *3rd International Symposium on Parallel and Distributed Computing*, Cork, Ireland, July 2004. In press.
- [13] T. Silvestre, E. Nugues, G. Perrière, M. Gouy, and L. Duret. Phylojava : a generic client-server tool for phylogenetic tree reconstruction - application to grid computing. In M.-F. Sagot and H.-P. Lenhof, editors, *European Conference on Computational Biology*, Paris, France, September 2003.
- [14] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
- [15] A. Stamatakis and T. Ludwig. Phylogenetic tree inference on pc architectures with axml/paxml. In *Proceedings of IPDPS2003 (High Performance Computational Biology workshop)*, Nice, France, 2003.
- [16] C. A. Stewart, D. Hart, D. K. Berry, G. J. Olsen, E. A. Wernert, , and W. Fischer. Parallel implementation and performance of fastdnaml a program for maximum likelihood phylogenetic inference. In *Proceedings of SC2001*, Denver, CO, USA, 2001.
- [17] United Devices. *Grid MP Platform Architecture*, 2003. White Paper.