BIOINFORMATICS APPLICATIONS NOTE Vol. 16 no. 6 2000
Pages 564–565

Disperse: a simple and efficient approach to parallel database searching

Raphaël Clifford[1] and Aaron J. Mackey[2]

[1]Department of Computing, Imperial College of Science, Technology and Medicine, London, UK and [2]Department of Microbiology, University of Virginia, Charlottesville, Virginia, USA

Received on October 18, 1999; revised on December 14, 1999; accepted on February 3, 2000

Abstract

Summary: *A general system for performing multiple independent database searches in parallel is presented. Run-time addition and removal of clients, robust failure and error trapping and near 100% efficiency with very large numbers of clients are achieved by a flexible asynchronous, client-driven approach.*

Availability: *Disperse software is freely available. Please visit http://www.doc.ic.ac.uk/~rc5/Disperse/ for further details.*

Contact: *r.clifford@ic.ac.uk, amackey@virginia.edu*

We present a customizable task-distribution system, denoted Disperse, which accomplishes parallelization by distributed serial execution of existing programs, which require no modification themselves.

Large-scale comparison of databases comprising the nucleotide or amino acid sequences of entire organisms is a common goal of current comparative genomics projects. These many-to-many comparisons are often computationally expensive and time-consuming, but the results obtained are extremely valuable, allowing, for instance, the clustering of evolutionarily related genes. Our method allows these kinds of analyses to be performed with greater ease and speed than previously possible.

Disperse requires any UNIX-like operating system, supporting Internet communication, and a modern Perl interpreter. Two Perl scripts, a 'server' and a 'client' are used. The server script is initiated through a command line which specifies search program parameters. If the database to be searched is not commonly available to all the clients, it may be copied to each client machine. The server then executes clients on each of the remote machines. These remote copy and execution steps are accomplished with the standard UNIX utilities **rcp** and **rsh**, or their secure counterparts **scp** and **ssh**. Each client then contacts the server using standard Internet communication to request a query sequence. Once the query is received, the connection is closed and the client provides the query to the specified search program.

The output of the executed search program is captured by the Disperse client and returned to the server via a new Internet connection, at which time a new query sequence is obtained. The client is able to recognize any errors generated by the search program, reporting them to the server for handling. If a client itself hangs or crashes, it does not affect the rest of system; the server keeps track of which queries have generated results, and resubmits any queries whose results are missing. If at least one client remains functional, complete results will be obtained. The ability to monitor the progress of the job via a server-maintained log file allows the addition of more clients and easy recovery from a server crash. When all the results have been obtained, or when no clients have connected after a specified timeout period, the server exits, writing a full description of its finishing state to the logfile.

This combination of Perl, standard UNIX utilities and internet protocols means that the system is able to run seamlessly across heterogeneous computing platforms in differing locations. For example, we performed multiple parallel database searches across the sites of the University of Virginia and Imperial College, London with Disperse. Although several different operating systems and architectures were involved, no extra configuration was needed except a UNIX account on each computer.

Performance did not differ significantly when clients searched a locally mounted (non-NFS) database, compared with searching an NFS-shared database. Disperse maintains an efficiency above 98% with 16 processors, above 90% with 64 and above 85% with 96. We are not aware of efficiencies of this size being published in similar work. The individual search times on these machines was about one query per 2 s, meaning that with 96 clients, the server was handling approximately 48 clients every second. At these rates, latency problems inevitably start to become significant. With greater than 64 processors, efficiency was increased slightly by using

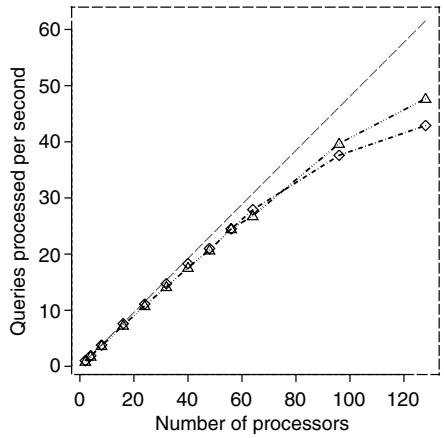

Fig. 1. Performance of the Disperse system using FASTA to search a client-local (non-NFS) database of 4289 *Escherichia coli* proteins, executing either one threaded client/machine (△) or two clients/machine (◇). The straight line at 45° corresponds to 100% efficiency. Ninety-six dual-processor Pentium II computers (450 MHz, 256 MB RAM), connected by a 100-baseT, fully switched, Ethernet network, were used for these timings.

threaded FASTA (Pearson and Lipman, 1988); this is a subject of further investigation. At 192 clients, Disperse exhibited only 50% efficiency, but an otherwise identical run using the SSEARCH (Smith–Waterman) program exhibited 70% efficiency, completing the entire database-to-database comparison in less than 15 min. It is expected that with larger computations, efficiency would be further increased.

Versions of FASTA and HMMER (Eddy, 1995) using parallel code development library PVM (Geist *et al.*, 1995) exist, allowing individual queries to be performed across distributed resources. Previous efforts to incorporate parallelism into FASTA and BLAST (Altschul *et al.*, 1990) have also been published (Miller *et al.*, 1991; Barton, 1991; Julich, 1995). These implementations divide the database to be searched amongst workers, and tally the results obtained when all the workers are finished. This approach requires the successful, synchronous completion of each worker to be maximally efficient. As a result, the efficiencies achieved are much lower than those shown here for Disperse. A comparison of the performance of Disperse with the PVM version of FASTA revealed that Disperse outperformed PVM-FASTA at all client numbers tested.

Acknowledgements

R.C. is supported by an EPSRC studentship. A.J.M. is supported by a grant from the National Library of Medicine (LM04969). We would like to thank Dr William R. Pearson for his insightful and challenging comments and questions. Also many thanks to Marek Sergot for his constant support.

References

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

Barton,G.J. (1991) Scanning protein sequence databanks using a distributed processing workstation network. *Comput. Appl. Biosci.*, **7**, 85–88.

Eddy,S. (1995) Multiple alignment using hidden Markov models. *Third International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, pp. 114–120.

Geist,A., Beguelin,A., Dongarra,J., Jiang,W., Manchek,R. and Sunderam,V. (1995) *PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Network Parallel Computing*. M.I.T. Press.

Julich,A. (1995) Implementations of BLAST for parallel computers. *Comput. Appl. Biosci.*, **11**, 3–6.

Miller,P.L., Nadkarni,P.M. and Carriero,N.M. (1991) Parallel computation and FASTA: confronting the problem of parallel database search for a fast sequence comparison algorithm. *Comput. Appl. Biosci.*, **7**, 71–78.

Pearson,W.R. and Lipman,D.J. (1988) Improved tools for biological sequence comparison. *Proc. Natl Acad. Sci. USA*, **85**, 2444–2448.