EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

Computer Science Department
Operating Systems Laboratory

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Parallelizing the blast applications on a network of Dec Alpha workstations

Alexandros Pappas

December 14, 1994

Abstract

The goal of this project, is to set up a service for blast requests on a network of Dec Alpha workstations. Blast (Basic Local Alignment Search Tool) is a family of programs used in genetics research, which try to find similarities between a given sequence of amino acids or nucleotides, and a database containing proteins or DNAs. The reason to do this on a distributed environment, is, to exploit effectively the computing power of a cluster of Dec Alpha workstations at the Computer Science Department at EPFL and at the same time, provide a usefull service to the research community.

1 Introduction

The blast programs compare a given sequence (of amino acids or nucleotides) with all the sequences of a database using the statistical methods of Karlin and Altschul [Kar Al 90].
There are six blast applications: blastp, blastn, tblastn, blastx, tblastx and blast3 which were implemented at the NCBI[1]. All these search in a database conatining sequences of amino acids or nucleotides and print the sequences, which "score" the highest (i.e. resamble most) to the query sequence. In addition, blast3, after printing the primary results, searches in the results for three way alignments.
One way to parallelize these applications, is, to cut the database into "chunks" and give each processor a chunk (i.e. a certain amount of sequences) in which to search for the query sequence. When it is finished with that chunk, send the next one, till no more chunks are available. With this method, blastp was parallelized by E. Masson [Masson94]. The other four applications where parallelized with the same method by the author.

[1]National Center for Biotechnology Information, USA

The problem was now, to decide which workstations of the cluster to choose for running a request. In their summer internship, C. Javet and X. Defago, implemented a programm for monitoring and scoring the load on the machines of the cluster. This point is discussed in detail in [Jav Def 94].
Also, a network server had to be written, to receive the requests, to forward them to the parallel blast application, to get the results, and send them back to the client. This was done by the author.
A problem still to be attacked, is what to do with requests, that can not be serviced right away, because the available resources are too busy. One has to decide, if they should wait, if they should be redirected to other servers, or if they should be executed on a smaller number of machines than by default.

2 Parallelization

The blast programs are written in C and the parallization was done with the Pvm library (Parallel Virtual Machine, described in [PVM 93]). Every parallel blast application has one master and a certain predetermined number of slaves, each running on a different processor. The master sends the command line arguments to the slaves, and every time a slave is finished with a chunk, the master gets the partial results from the slave, adds them to the results already received, and sends the next chunk. Note, not the portions of the database are sent, but just a chunk number. The database is accessed via the network with NFS, which with the type of network at our installation (FDDI) is pretty fast. But, this is still a point to investigate, if it would be better to send the actual portions of the database, so that each slave, makes just local accesses. At the end, when the master has collected the results from all the slaves, the master prints the results.
Blast3 is a little bit more complicated, because, after the first search, it searches for "three-way alignments" on the results. We did not parallelize blast3, because it is no more supported in the blast 1.4 version.
An effort was made, to parallelize the applications automatically, i.e. to translate the serial version to the parallel version with a program and not by hand. This is quite a simple program written in perl. It looks for some characteristic lines in the original program, and inserts some additional commands for the pvm version. If future serial versions are not too different from the current ones, the parallel version will be generated automatically. However, during this project we found out, that this is an illusion, because different serial versions can be very different.
In order to serve many requests at a time, things become a little more complicated. The parallel blast program (blast_pvm) calls a deamon, which reserves for that particular request some workstations according to their load. The deamon starts the master, and the master starts the slaves on the designated workstations. The deamon is now ready to accept other requests. When the master finishes, it notifies the deamon and tells blast_pvm to get the results. The deamon sets the workstations free again.

3 Server

We can provide three types of access to our service:

- By direct connection through a port.
  A client program requests a connection through a port with our network server. The network server, receives through a simple protocol the name of the blast program to execute, the command line arguments, and the query sequence. It starts the parallel blast application, collects the results, and writes them back to the same port. That required a lot of programming with

sockets and unix kernel calls. The server was implemented by the author, the client program, by NCBI.

- By mail.
  Users send a mail in a certain format, which is read by a program. Our mail server parses the message, starts the parallel blast application, collects and writes the results into a mail message, and sends them back. This was implemented by the author.

- Mosaic.
  On a mosaic page, the user can interactively give the query and choose the value of the parameters. The mosaic server, which may be anywhere, calls the network client program which makes a direct connection through a port to the blast program. The results, are displayed on a mosaic page. Kay Hoffmann at ISREC[2] wrote a mosaic page to interface with the network program.

4 Maintenance

There are some tasks which have to be done from time to time, in order to provide an efficient service.

- Database updates

  The protein and DNA databases must be updated. New information comes in every day. The NCBI and EMBL are the two major database maintainers. Database updates must be copied from these sites at least weekly, perhaps some of them, even daily.

- Program updates

  The original blast programs were developed and are maintained at NCBI. When a new version comes out, the pvm versions should also be updated. This might be a quite difficult task, for someone who does not know the structure of the blast programs. There may be also some minor changes to be done, like when a system upgrade is done, or a new version of PVM comes out.

- Statistics, Server restart

  It would be usefull to collect some data about the requests submitted, like start time, processing time, database used, rejected requests in order to ameliorate the service. That meens, the data must be analyzed and eventually corrections will be proposed and implemented. Also somebody should have an eye on the server and restart it when a crash occurs, or when the machine is rebooted.

References

[Kar Al 90] Karlin, Samuel and Stephen F. Altschul (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. USA 87:2264-68.

[PVM 93] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam. PVM 3 User's Guide and Reference Manual. Oak Ridge National Laboratory, May 1993.

[2] Swiss Institute for Experimental Cancer Research

[Masson94] Eric Masson, Portage d'une application de biologie moléculaire d'un environnement multiprocesseurs vers un réseau de stations", diploma thesis, Feb 1994, EPFL/LSE

[Jav Def 94] C. Javet, X. Defago, Distribution de la charge pour BLAST, internal report, Jul 1994, EPFL/LSE