An Investigation into Implementations of DNA Sequence Pattern Matching Algorithms

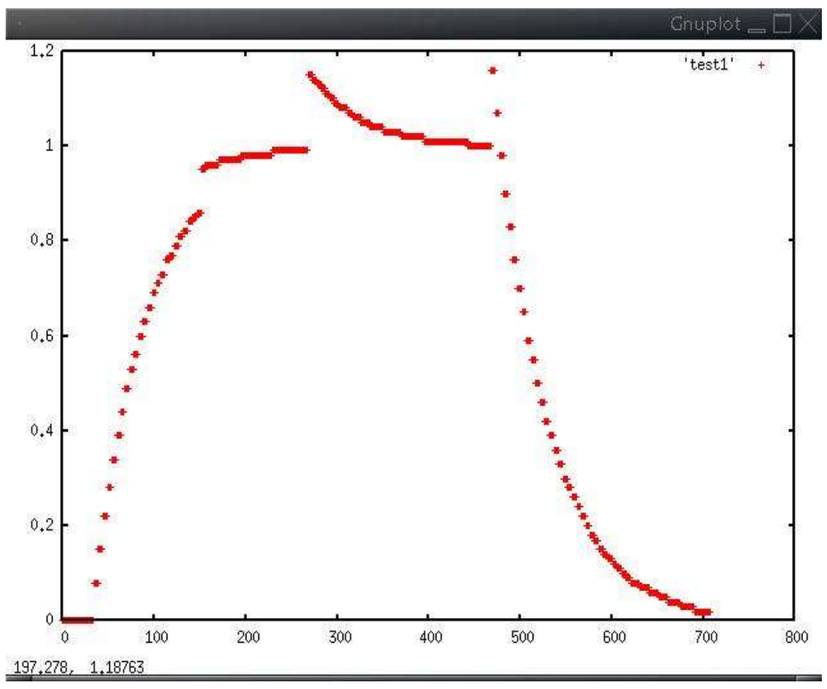
Peden Nichols

Abstract

The BLAST (Basic Local Alignment Search Tool) algorithm of genetic comparison is the main tool used in the Bioinformatics community for interpreting genetic data. Existing implementations of this algorithm (in the form of programs or web interfaces) are widely available and free. Therefore, the most significant limiting factor in BLAST implementations is not accessibility but computing power. My project deals with possible methods of alleviating this limiting factor by harnessing computer resources which go unused in long periods of idle time. The main methods used are grid computing, dynamic load balancing, and backgrounding.

Background

There is an immense amount of genetic data generated by government efforts such as the human genome project and by organization efforts such as The Institute for Genomic Research (TIGR). The task of extracting useful information from this data requires such processing power that it overwhelms current computational resources. However, there exist large amounts of unused processing power in schools and labs across the country; most computers are never being used all of the time, and most of the time that computers are being used their processors are nowhere near 100% load. Harnessing some of this unused power is a useful problem not just for the specific application in Bioinformatics of DNA sequence pattern matching, but for many computationally intensive problems which could be solved more accurately and faster with increased resources.



Development

The first step in harnessing unused processor power is to clearly establish and document the existence and magnitude of that unused power. Accomplishing this task requires that we establish some metrics for describing computer load and develop a way to keep a record of those metrics over time. Perl is an ideal language with which to write a program which could perform this task because of its text manipulation capabilities and high speed. The program “cpuload” uses the Linux “uptime” command every second, parses the output, and writes the results to a file which is then plotted using gnuplot. The graph shows the results over one execution of the BLAST algorithm comparing two strains of e-coli bacteria.

The use of grid computing to optimize BLAST implementations is not an original idea; a program called mpiblast has already been written and made available to the public. However, implementing mpiblast in any given environment is not a trivial task. For example, our systems lab, although it has mpi installed on several computers, has not maintained a list of which computers are available to run parallel programs. My next task was to compile this list using essentially trial and error and running a test mpi program, mpihello.c. The original, obsolete mpihosts file and the updated file are shown below.

Initial (obsolete) machines list

Updated machines list