MPBLAST: improved BLAST performance with multiplexed queries

Ian Korf and Warren Gish

Genome Sequencing Center and Department of Genetics, Washington University School of Medicine, St. Louis, MO 63108, USA

Received on June 8, 2000; revised on August 10, 2000; accepted on August 11, 2000

Abstract
Summary: We have developed a program, MPBLAST, that increases the throughput of batch BLASTN searches by multiplexing (concatenating) query sequences and thereby reducing the number of actual database searches performed. Throughput was observed to increase in reciprocal proportion to the component sequence length. For sequencing read-sized queries of 500 bp, an order of magnitude speed-up was seen.

Availability: Free (see http://blast.wustl.edu)
Contact: {ikorf, gish}@watson.wustl.edu

Main text
BLASTN (Altschul et al., 1990) searches are used in many applications that involve large batches of short query sequences. Examples include mapping, clustering, and annotating ESTs and genomic fragments. Typically, such batch jobs are performed by a script that executes one BLAST search for every sequence. We have developed a program, MPBLAST, that substantially improved the throughput of BLASTN, by multiplexing query sequences so that fewer actual database searches were performed. For short sequences, the benefit of performing fewer searches far outweighed the computational overhead of multiplexing.

In the strategy employed by MPBLAST, a preprocessor concatenates numerous short sequences into relatively few long sequences. These multiplex sequences are then used as queries in the BLASTN searches. A postprocessor parses and deconvolves the resultant BLAST reports, including conversion of multiplex query coordinates back to their component sequence origins. Importantly, gapped alignments against a multiplex query are prevented from crossing individual sequence boundaries. This is accomplished by inserting an appropriately constructed spacer or barrier in between each sequence (see Methods).

In Figure 1a, we characterized a common application of BLASTN. A small EST database was searched with 1–200 genomic query sequences, each 500 bp in length, and search times were compared for the serial versus multiplex strategies (‘Relative speed’). Improved throughput was apparent even at small multiplexes and reached a maximum at about 200 sequences (100 kbp total), where the relative speed was about 10-fold.

Throughput depended on several factors, from the version of BLASTN, to the parameters used, to the computer hardware, and the sequences themselves (data not shown). Perhaps the single most important factor, however, was the length of the component sequences. Large, BAC-sized component sequences actually ran slower using MPBLAST than with the traditional serial strategy, due to the increased memory demands of longer multiplexes and the cost of running BLASTN indirectly via MPBLAST (data not shown); while sequences shorter than 500 bp exhibited even greater benefit than that reported in Figure 1a.

To characterize the dependence of speed on length, we maintained a constant multiplex length of 64 kbp and successively halved the component length, thus doubling the number of sequences in the multiplex each time and halving the actual number of database searches performed. Figure 1b shows that a 2-fold change in sequence length resulted in roughly a 1.5-fold change in throughput over a range of query lengths from 62 bp to 4000 bp.

MPBLAST can be applied equally to other BLAST search modes—for example, TBLASTN (Altschul et al., 1990), BLASTX (Gish and States, 1993), and TBLASTX (W. Gish, unpublished)—but the efficiency gain was not as appreciable for all but the shortest of queries because database I/O for the other modes typically constitutes a much smaller fraction of the overall run time (data not shown). By using less sensitive (faster) search parameters, however, the benefit to using MPBLAST could be increased in all modes.

Aside from speed, multiplexing also impacts the estimation of statistical significance of the results. The search space with multiplex queries is larger than with single sequence searches and, therefore, the E- and P-values would ordinarily increase (Karlin and Altschul, 1990). Furthermore, blind application of joint statistical approaches, such as Sum statistics (Karlin and Altschul,
MPBLAST: improved BLAST performance in multiplexing

1053

Fig. 1. Comparisons of serial and multiplexed BLASTN searches. Performance is shown as relative speed, or the ratio of the time to complete serial over multiplex searches. Relative speed depended on: (a) number of sequences in the multiplex (constant component sequence length, variable multiplex length); and (b) length of the component sequences (variable component length, constant multiplex length). In (a) the data do not go through (1, 1) due to the overhead of running BLASTN via MPBLAST.

Methods

The 200 query sequences were non-overlapping fragments made from human genomic sequence RP1-130H15 (AC004997.2), whose repetitive elements were masked with RepeatMasker (A.Smit, unpublished, http://repeatmasker.genome.washington.edu; version dated 4-Apr-2000). The database was comprised of 130,000 human ESTs (46,707,689 bp). BLASTN searches were conducted with Washington University BLAST (Gish, 1996) dated (15-Jun-2000), using default parameters except ‘CPUs=1’. MPBLAST was developed under UNIX and is written in PERL.

With NCBI BLAST (Altschul et al., 1997) and older versions of WU-BLAST 2.0, the spacer inserted by MPBLAST between component sequences consisted of a run of negative-scoring letters (usually Ns or hyphens) sufficient in length as to guarantee gapped alignments would terminate before they crossed into adjacent sequences. The minimum length required for the spacer was dependent on the specific alignment parameters used. The WU-BLAST version used here required only a single hyphen between sequences, regardless of the alignment parameters.

Acknowledgements

This work was supported by NIH grant P50HG01458, with equipment support from Sun Microsystems Inc. and Compaq Computer Corporation.

References


