Profiling the BLAST bioinformatics application for load balancing on high-performance computing clusters

Trinity Cheng1,2, Mike Mikailov, PhD2

1River Hill High School, Clarksville, Maryland

2U. S. Food and Drug Administration, Silver Spring, Maryland

Abstract

The NIH/NCBI BLAST software was profiled to identify the five most time-consuming functions that occupy more than 92% of the runtime. Runtime data collected with six databases and 15 queries on six different hardware configurations were used to create prediction models for finding the optimal data partitioning strategy for a high-performance computing cluster of thousands of nodes.

Introduction

The Basic Local Alignment Search Tool (BLAST) is probably the most widely used bioinformatics software for matching biological sequences between two user input files – database and query. The input files usually consist of millions of sequences, which makes BLAST computationally challenging and suitable for data parallelization on a high-performance computing (HPC) cluster. The "dual segmentation" method proposed by Mikailov et al. divides the database and query into *m* sub-databases and *n* sub-queries, respectively, to be concurrently processed by nodes. This study extends the dual segmentation work to find the optimal solution for *m* and *n*.

Methods

Three profiling techniques were used to analyze BLAST workload: shell-level profiling with the Unix “time” command, code-level profiling with the BLAST built-in module “profiler,” and system-level profiling with the Unix “gprof” program. The runtimes were measured with six different database files from 8,179 to 261,725 sequences and 15 query files from 17,848 to 1,142,220 sequences. The experiments were conducted on the FDA HPC cluster with more than 500 nodes. The runtimes for six different types of node were measured. The measurement data were fitted with bivariate quadratic functions to develop runtime prediction models. Finally, the models were used to guide the data parallelization for BLAST jobs.

Results

Profiling results (Table 1) show that BLAST contains more than 34,500 different functions, but a single function (RunMTBySplitDB) takes 99.12% of the total runtime. Among its 53 child functions, five core functions were identified to make up 92% of the runtime. Runtimes of these five core functions were individually measured against database size and query size (Figure 1) and fitted with bivariable quadratic functions (Table 2). The hardware performance of six different node types was characterized and normalized for each of the five core functions as a radar chart (Figure 2).

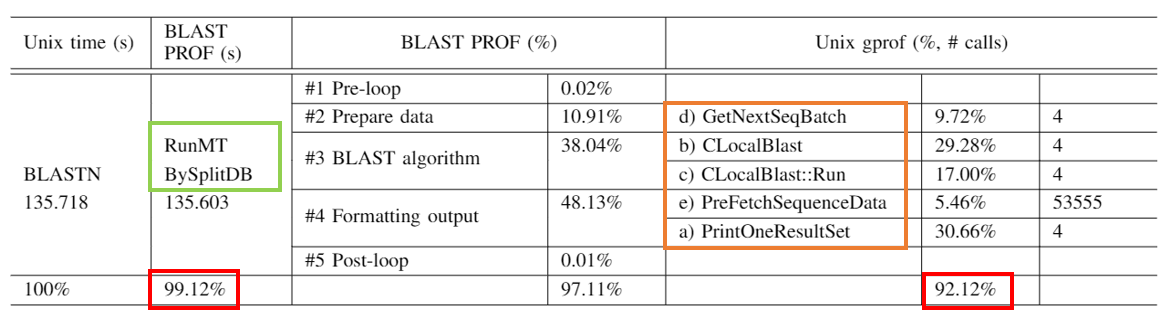
Finding the optimal data partitioning for BLAST

Consider a BLAST job of (*D,Q*), where *D* and *Q* are the database and query sequences, respectively. The BLAST job (*D,Q*) is concurrently processed by *P=m\*n* nodes, where each node processes a sub-BLAST job of size (*D/m, Q/n*). Our models provide a function *TH*(*d,q*) that estimates the runtime required by a BLAST job (*d,q*) on node type *H*. By factoring *P=m\*n*, the optimal solution can be found in the solution space constructed by all *TH*(*D/m*,*Q/n*).

As an example, when *D*=523,449, *Q*=73,102,023, and *P*=4,096, the optimal solution for node type five was found to be (*m,n*)=(2,2048), which had a runtime of 37.79 seconds. In comparison, a solution that evenly divides the database and query as (*m,n*)=(64,64) takes 204.15 seconds, 5.4 times slower.

Discussion and Conclusion

By using profiling techniques, five core functions were identified to cover 92% of BLAST runtime. Different database and query files were tested to measure runtimes of the five core functions. Simple bivariate quadratics were used to model the runtimes on different node types. The models can guide parallelization of BLAST jobs by recommending the optimal data partitioning on an HPC cluster. Experiment data show that the optimal data partitioning improves overall runtime 5.4-fold in comparison with dividing the database and query into the same number of fragments.



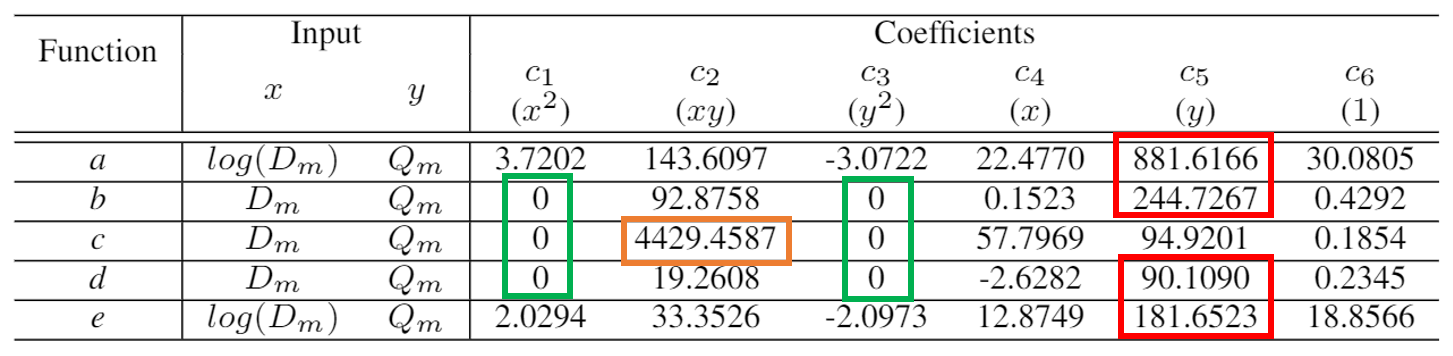
**Table 1**: Shell-, code-, and system-level profiling for BLASTN. Of the 34,500 functions contained in BLASTN, RunMTBySplitDB (boxed in green) takes 99.12% of the runtime. RunMTBySplitDB calls five core child functions (boxed in orange), which make up 92.12% of BLASTN’s runtime.

Chart, scatter chart

Description automatically generated

**Figure 1.** Runtimes of the five core functions vs. database size (*x*-axis) and query size (*y*-axis). The surfaces of functions *a* and *e* resemble saddles, while functions *b*, *c*, and *d* are planes. For functions *a*, *b*, *d*, and *e*, the surface descends faster in the query dimension, indicating that query size has a greater effect on runtime.

Chart, radar chart

Description automatically generated

**Table 2.** For Dm\*106 database sequences and Qm\*106 query sequences, the coefficients for bivariate quadratics modeling the runtime of each function in the form are shown. Functions *a, b, d,* and *e* are much more proportional to query size than database size (red boxes). Additionally, functions *b, c* and *d* are modeled as planes (green). Function *c* is proportional to both query and database size (orange).

**Chart

Description automatically generated**

**Figure 3.** BLASTN runtime on node type 5 (*z*-axis) vs. database size (*x*-axis) and query size (*y*-axis). The blue mesh represents the measured runtime. The red hyperbolic curve represents the possible runtimes of an HPC system with 4,096 type 5 nodes.

**Figure 2.** Normalized performance of six node types, with respect to node type 5, for each of the five functions.