Basic Local Alignment Search Tool (BLASTn) for Hardware Acceleration

Alden Param, Alex Chan, Hmayak Apetyan, Jacob London, Simon Tutak, Siva Prabakar, Dr. Aly, Dr. Ellabaan
California State Polytechnic University, Pomona Department of Electrical and Computer Engineering

Introduction

Since the completion of the first draft of the human genome in 2001, genomic data has been far outpacing Moore's law and is expected to reach the Exabyte scale with the next year and surpass even YouTube and Twitter by 2025 [1]. However, even with the advent of third generation sequencing technologies, it is difficult to process the overwhelming amount of genomic data. These technologies have prohibitively high computational costs; over 1,300 CPU hours are required to analyze the data using a reference, and over 15,600 CPU hours are required to assemble the reads de novo (or without reference) [1]. One of the most popular methodologies created to speed up this process is BLASTn, or Basic Local Alignment Search Tool for nucleotides. For DNA subjects and DNA queries, BLASTn is used. Even with processes such as BLASTn, it can take a significant amount of time to process all of the subject. Another issue with processes like BLASTn is that it can be inaccurate, as it is a heuristic algorithm and thus takes shortcuts to lower the processing time.

Method

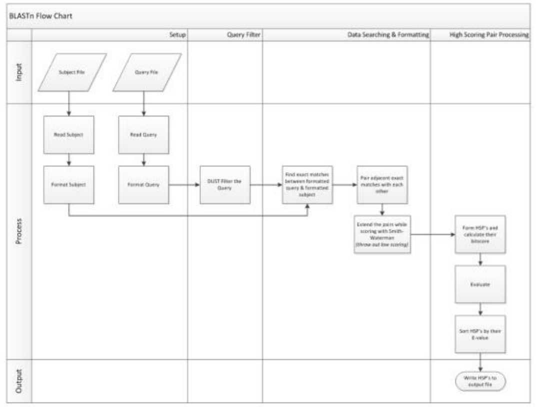
BLASTn Flow Chart
Setup
Query Filter
Data Searching & Formatting
High Scoring Pair Processing
Input
Process
Output
Subject File
Query File
Read Subject
Read Query
Format Subject
Format Query
DUST Filter the Query
Find exact matches between formatted query & formatted subject
Pair adjacent exact matches with each other
Extend the pairs while scoring with Smith-Waterman (throw out low scoring)
Form HSP's and calculate their bitscore
Evaluate
Sort HSP's by their E-value
Write HSP's to output file

Fig. 1. The BLASTn flowchart process.

Evaluation

With all of the test data, the Python implementation took around 35 minutes to run; however, it produced results that validated its design. From this, the C++ implementation is built from the building blocks set up in the Python design. The first version of the C++ implementation took approximately 31 minutes to fully execute. After using faster data structures, the program executed in approximately 1 minute and 5 seconds. Lastly, with GCC level 3 optimization, the runtime was reduced to 8.6 seconds with the small dataset and 8 minutes and 30 seconds with the large dataset.

Conclusion

This project was designed to provide an innovative solution for the large amount of data needing processing in the bioinformatics industry. The BLASTn algorithm implemented C++ to interop with an FPGA board with the BLASTn algorithm. Through evaluating the efficiency of the software solution, the Smith-Waterman algorithm was identified as the bottleneck in performance. As a result, the Smith-Waterman algorithm was implemented on an FPGA board with UART interfacing to score the words.

Acknowledgements

1. The ECE department at California Polytechnic State University, Pomona.
2. Mostafa M. Hashim Ellabaan for assistance through provided data and feedback regarding algorithm development and progress.
3. Mohamed El-Hadedy for the assistance in both consultancy and reference for the development of this project.