15618 Project: Parallel String Matching Algortihms

Abhishek Kumar and Runze Wang

URL of Project Webpage

https://abhishek763.github.io/618project/

Summary

We are going to work on Parallelelizing String Matching algorithms like Aho-Corasick, KMP, and Rabin Karp algorithms. These string search algorithms are sequential in nature, hence not efficient for large scale tasks like DNA Sequence Matching, Network Intrusion Detection (NID). We wish to explore their parallel implementations on GPUs.

Background

The Aho-Corasick algorithm is used to find matching substrings for a set of patterns in a large input string. For example, given dictionary of words *{a, ab, bab, bc, bca, c, caa}* and input text *abracadabra*, it must output *a* occurs at indexes 0, 3, 5, 7, 10, *ab* occurs at 0, 7, bab doesn't occur anywhere,.... and so on.

So it works on multiple patterns. It creates a Finite State Automaton which is similar to trie but with additional links which help in faster transition between states when matching fails. It was used in Unix command `fgrep`. It is used in various applications like network intrusion detection and bioinformatics for finding several input strings within a given large input string. If `n` is the length of the text and `m` be total number of characters in the search dictionary, the Aho-Corasick algorithm works in `O(n+m+z)` complexity where `z` is the total number of occurrences in text.

KMP algorithm is a linear time searching algorithm which given a pattern and text, finds all occurrences of the pattern in the text. If the length of the text is `n` and `m` is the length of the pattern, then KMP works in Time Complexity `O(n+m)`.

Rabin Karp is another string matching algorithm which uses hashing. It uses a rolling hash to eliminate positions of the text that cannot match the pattern, and then checks for the exact match at the positions where hash of the pattern matches the hash of the rolling window.

All these algorithms are very good for matching patterns sequentially. In this project, we will explore their adaptations in parallel domain especially over large input text size. Network Intrusion Detection Systems (NIDS) use string matching engine to identify network attacks by inspecting packet content against thousands of predefined patterns. Due to the increasing number of attacks, traditional string matching approaches on uni-processors may not be sufficient. Parallel String matching can be very helpful for other applications too like DNA Sequence matching, large scale text mining and digital forensics.

The Challenge

One of the biggest challenges for this project is that there isn't enough literature around it. The workload will be a large number of very large input text. We would look at different numbers of patterns (small, moderate, large). But, we will assume that size of pattern will be much less than the size of the text. Based on the implementation, there might be less locality in the data. We will get to know more about this once we do our experiments.

Resources

We will be using GPUs on GHC machines mostly for our work. We might consider PSC machines if we end up using MPI or OpenMP. But for now, the plan is to work with CUDA.
We found a parallel implementation of Aho-Corasick algorithm https://code.google.com/archive/p/pfac/ We will try to run and benchmark our implementation against it. We plan to have our own implementation for both parallel and sequential versions of the algorithm. We will be taking help from this paper regarding Parallel Aho-Corasick algorithms (PFAC) https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5683320

Goals and Deliverables

75% Goals: Sequential and CUDA Parallel implementation of Aho-Corasick, KMP, and Rabin Karp Algorithms

100% Goals: 75% Goals + A very good analysis of the performance of sequential algorithms vs parallel algorithms, analyzing bottlenecks in workload and comparable performance against the reference implementation of PFAC.

125% Goals: 100% Goals + Trying other parallelization techniques like SIMD/Cilk/OpenMP/MPI/Charm OR achieve significantly better performance than reference implementation of PFAC.

Platform Choice

We aim to use GPU in the GHC machines. General Purpose GPUs have been very successful at a lot of parallel tasks and we feel that we should explore their use-case on the parallel string matching problem. A good performance on GPU will mean that many systems like Network Intrusion Detection and DNA Sequence matching can use GPUs for improving performance.

Schedule

13 Nov - 19 Nov	Implement Sequential Versions of Algorithms
20 Nov - 26 Nov	Implement Parallel version of Aho-Corasick
27 Nov - 30 Nov	Work towards Project Milestone
1 Dec - 7 Dec	Work on Parallel version of KMP and Rabin Karp
8 Dec - 14 Dec	Do Analysis over different algorithms
14 Dec - 17 Dec	Work towards Final Project Report and Poster