15618 Project: Parallel String Matching Algorithms

Milestone Report

Abhishek Kumar and Runze Wang

Project Progress

At this point, we have completed implementing sequential versions of Aho-Corasick, KMP, and Rabin Karp pattern matching algorithms. We have also completed implementing parallel versions of Aho-Corasick, KMP, and Rabin Karp via OpenMP. In particular, we parallelized the construction of AC automata by utilizing a lock-free Trie insertion algorithm and parallel BFS traversal for constructing fail links. We have also implemented an initial version of parallel failureless Aho-Corasick using CUDA as described in the project proposal. We have finished initial rounds of benchmarking to make sure the parallel implementations exhibit speedup against the sequential implementation. The current parallel implementations of string matching algorithms via OpenMP and parallel failureless Aho-Corasick via CUDA both show great speedups against the serial implementation.

Compared with the schedule in the project proposal, we believe that we are on the schedule if not being ahead of schedule. We believe that we are still able to finish the deliverables mentioned in the project proposal, namely (1) a CUDA implementation of KMP and Rabin Karp and (2) benchmarking different implementations and reasoning about the performance of different implementations under varying assumptions about the patterns and the text.

Updated Schedule

Time	Task	Who?
1 Dec - 5 Dec	Finish CUDA-based implementations of Rabin Karp and KMP.	Both
6 Dec - 10 Dec	Design benchmarks to measure speedup, memory access pattern, and bandwidth utilization.	Both
11 Dec - 14 Dec	Perform experiments and iteratively improve implementations.	Both
15 Dec - 17 Dec	Finish Report and Poster	Both

Plan for Poster Session

In the poster session, we will present our analysis using graphs and plots and we will show different benchmarks that we use to reason about the performance of different string search implementations on different kinds of queries (patterns).

Potential Issues

We don't foresee any potential problems as of now.

However, designing good benchmarks can be nontrivial. We need to analyze the algorithm theoretically, think about common applications of pattern searching algorithms, find relevant corpus for evaluation, conduct the experiment on GHC or even PSC machines, and iteratively improve our implementations based on the feedback of profilers. Our analysis will not be restricted to the speedup. We will also reason about why the data pattern makes a particular algorithm good or bad.