Matthew Holmes

Parallel Programming – Abstract

# Parallelized Davis-Putnam Algorithm

## Abstract

The purpose of this project is to analyze the effect of parallelizing the Davis-Putnam automated theorem prover algorithm. The Davis-Putnam algorithm is used on a set of clauses in conjunctive normal form. It then sequentially branches on literals until it is determined to be satisfiable, or if there is some set of variables that satisfy all the conditions. This branching factor, in addition to iterating over large sets suggests that speedup due to parallelism is possible. Thus, CUDA and OpenMP were used in order to achieve speedup to various degrees.

## Implementation

When first beginning this project, a serial version was made and benchmarked. It was run through various test sets to ensure validity. OpenMP was then used in various iterations until proper speedup was obtained. In CUDA, thrust was used in order to allow for easy operations on the large vector based datasets.

When it came to actually testing the amount of speed that was obtained through this new parallelism, several different test sets were made that tested different aspects of the program. A simple case, which shows a fairly normal problem was used as a baseline to compare the serial versions to. Then several large datasets that are worst-case were created using the following rules, with n values of 10,000,000 and 100,000,000:

{x0, x1, …, xn}

{x0, x1, …, xn-1}

Then a large valid set of data was created using n=1000:

{x0, x1, …, xn}

{!x0}

{!x1}

…

{!xn}

Finally, the last set of data that was made does a worst-case scenario with many sets with few elements, this time with n = 5000:

{x0}

{x1}

…

{xn}

## Results

Here is a chart with the results, with time in milliseconds

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Valid Small | Invalid  (10,000,000) | Invalid  (100,000,000) | Valid Large  (1000) | Many Set  (5000) |
| Serial | 0 | 195.8 | 1911.7 | 887.2 | 20971.6 |
| OpenMP | 1.8 | 157.3 | 142.18 | 1355.8 | 25989.2 |
| Cuda | 1.99 | 34.6 | 293.45 | 276.1 | Out of Memory! |

Overall I am pleased with the results. I wish the OpenMP versions had more speedup overall, and if given more time to work on this project that would certainly be something that I would look into. The CUDA versions are really fast, and I checked their validity and didn’t get a single question wrong so I’m extremely happy with that.

When it comes to things to work on, the many set data had just horrible, horrible runtimes. I spent quite a long time trying to boost the speed here but I just couldn’t find a way. This is another aspect that if given more time I would certainly take a longer look at it. The CUDA version runs great but has terrible large set times, and even runs out of memory for the many set invalid portion. I spent a long time looking over the code and attempting to debug where all this memory is being used and I couldn’t find a reasonable fix that didn’t break the other cases. I would say that given more time I would look into this too, but realistically I can’t find a better way other than taking out thrust and writing my own kernels in CUDA, where I can keep track of my memory allocation better.