Parallel Computing Logbook

# INTRODUCTION

<< Provide any introductory information regarding the study, and the details of the report, e.g. what is the report all about, the structure of the report etc. >>

Logs of Progress

## Session One - October 31st, 2019

The goals of this session was to create working brute force algorithm and to demonstrate it working by having it crack the target serially. Once it is known that the program works serially then reconfiguring it utilising OpenMP and OpenMPI would be simpler, as the development wouldn’t involve testing both the algorithm and the parallelisation simultaneously. If it is certain that the algorithm works serially, then any issues that arise in the later stages would be solely due to the newer additions.

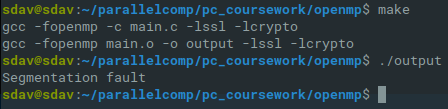
The brute force algorithm is a modified version of the provided algorithms, being extended from checking three levels of permutations to five, this stage is much easier than previously anticipated due to the knowledge that the key we are attempting to find will certainly be five characters in length including “a-z, A-Z, 0-9”. This fact removes the need to dynamically resize the string we are generating and instead just create a string of size 5 with permutations of the aforementioned character sets.

Resultantly, instead of using the method within the provided algorithm, it was instead opted to use a hard coded character set and arrange the string using that instead of calculating the value of an ASCII character like it was previous, this created the basic brute forcing algorithm.

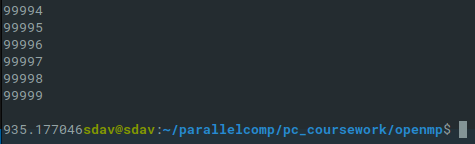


This was then tested as previously mentioned, however it was not as simple as previously thought.

At first the brute force algorithm began segmentation faulting, usually this would be due to a data structure being overflowed or possibly an incorrect input being passed somewhere. The latter is out of the question, as the program has hard coded inputs and there are no varying type functionalities. The former is difficult to judge, as the program utilises few data structures and those that are used do not see the massive amount of data being generated by the brute force algorithm.



In order to test what was causing the segmentation fault I separated the AES decrypting code from the brute force code I had developed and ran the latter on its own to see if it was the one causing the fault, and it was not.



The brute force itself is capable of running to completion even when printing to the command line, which usually uses more resources. The only thing excluded from this sequence was the AES OpenSSL decrypting code, meaning the segmentation fault occurs there.

METHOD OF PARALLELISATION

<< Provide details of parallelisation approach, referring to renown taxonomies and approaches; e.g. SIMD/MIMD etc, Data Parallelism, Task parallelism. A good reason should also be provided for why the identified parallelism is adopted. This section should not exceed 500 words. >>

Performance analysis

<< Experimental and test results should be collated in this section in tables(s). Comparative analysis and discussions can be provided with help of plotted graphs and other possible visual materials, wherever appropriate. The analysis can be developed referring to theoretical lows, rules, and models, such as Amdahl’s Law, Gustafson Law etc. This section should not exceed 2000 words. >>

CONCLUSIONS

<< Draw upon the evidence and analysis presented in the previous section, together with any relevant and useful sources, to derive conclusion(s) >>

REFERENCES

<< List references using Harvard style >>