ENCE360 Assignment - Http Downloader Algorithm Analysis

Name: Jiahao Guo
Student ID: 53974451
Lecturer: Richard Green
Date: 10 / Oct / 2017

Terms of Reference

A report submitted in fulfilment of the requirements for Course ENCE360, College of Engineering, University of Canterbury.

Summary

The algorithm that is used in the downloader program is recognized as cigarettes smoker algorithm. The best thread number to be chosen for the performance test is 4 since the computer is equipped with 4 cores.

Introduction

This report is aimed at recognizing the algorithm that has been used on the 'downloader' program and analysing the performance by changing the thread number / testing url number.

Methods

The algorithm can be identified by reviewing the lecture notes. To do a performance test, a modified version of the 'perf' script from lab 5 can be used.

Results

Algorithm

The algorithm is most likely to be cigarette smoker algorithm. Program calls 'spawn_workers' at the start to produce threads and at the mean time calls the routine function 'worker_thread' to get hosts and paths ('cigarettes ingredients') from *todo queue, calls 'http_url' to get the header info and put the header info back to *done queue. At the same time, program in 'main' keeps reading URLs and put them in *todo queue. A counter called work is used to keep track of the current capacity of the queue. If the queue is full the program will call 'wait_task' which start downloading contents with the header info provided earlier and with the directory (final ingredient) from input. After this, one queue slot is freed and counter is decremented. The second while loop is just checking if the program has emptied the working queue since the program was using pre-increment. On finishing, the program closes file descriptor and frees the queues.

Performance

On first run, threads number 1, 3, 5, 9, 12, 16, 22 and 32 are used. Time had decreased trend from thread number 9 to 16 and between thread number 3 and 5.

Commands	Real	User	Sys
time ./downloader test.txt 1 download > perf_test/test1.txt	0m0.563s	0m0.008 s	0m0.016s
time ./downloader test.txt 3 download > perf_test/test1.txt	0m0.188	0m0.000 s	0m0.020s
time ./downloader test.txt 5 download > perf_test/test1.txt	0m0.174s	0m0.000 s	0m0.020s
time ./downloader test.txt 9 download > perf_test/test1.txt	0m0.177s	0m0.004 s	0m0.016s
time ./downloader test.txt 12 download > perf_test/test1.txt	0m0.176s	0m0.000 s	0m0.020s
time ./downloader test.txt 16 download > perf_test/test1.txt	0m0.169s	0m0.000 s	0m0.020s
time ./downloader test.txt 22 download > perf_test/test1.txt	0m0.181s	0m0.000 s	0m0.020s
time ./downloader test.txt 32 download > perf_test/test1.txt	0m1.045s	0m0.000 s	0m0.020s

*Real --- wall clock timer, time from start to finish of the call

*User --- the amount of CPU time spent in user-mode code (outside the kernel) within the process

*Sys --- the amount of CPU time spent in the kernel within the process

Files of different sizes were tested with the program and results are shown below. As clearly as it is indicated in the table. The larger the file size the slower the performance goes.

Commands	Real	User	Sys
time ./downloader small.txt 4 download > perf_test/small.txt	0m10.370 s	0m0.164 s	0m1.060 s
time ./downloader large.txt 4 download > perf_test/large.txt	0m16.334 s	0m0.360 s	0m3.412 s

Discussion

Further investigation showed that the program performs the best when the thread number is 4. The reason of this is that the CPU got only 4 cores when it processing the worker threads. Thread numbers that exceeded 4 will cause the inefficient threading processes.

After the same performance test technique applied on the two different downloader with increasing thread number, no significant difference was observed. It is likely that there is nothing wrong with the concurrent queue.

Conclusion

The result of this report shows that choosing an appropriate algorithm and thread numbers optimizes the performance of a program.