

Systems Programming Coursework 2: Concurrency

1. Status

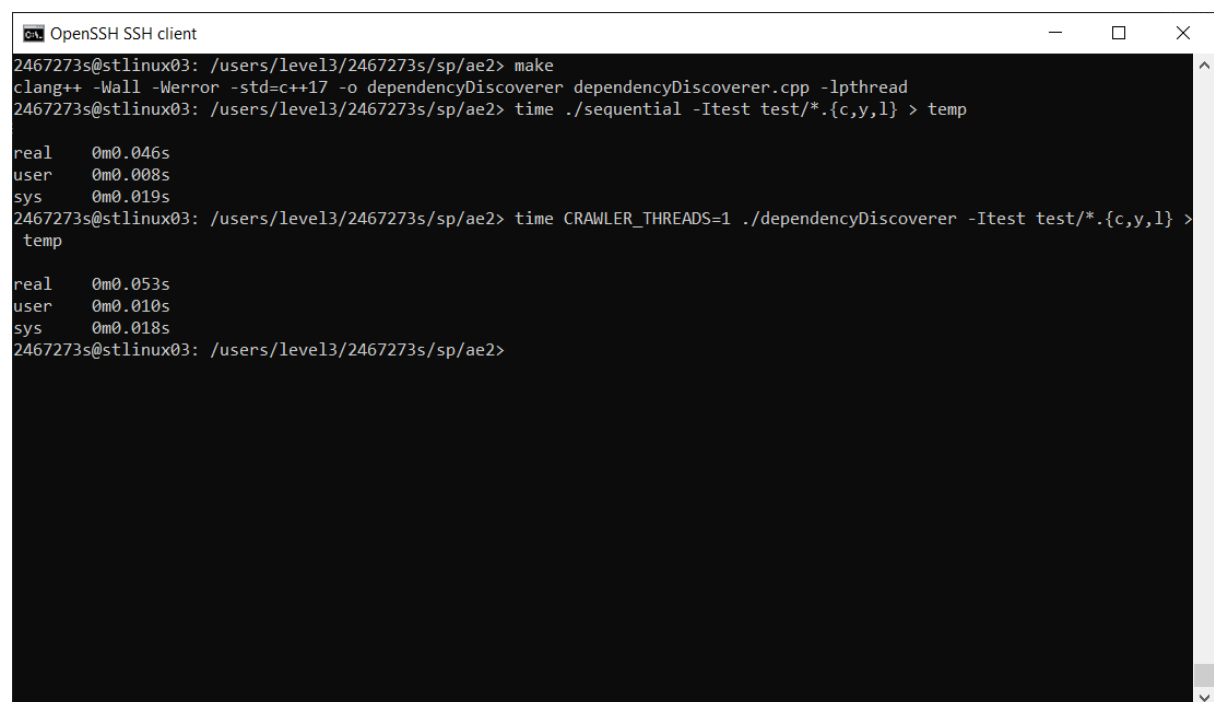
The program compiles and runs successfully, giving the exact output expected. It is a full solution to the specification problem, using the number of threads specified in the environment variable `CRAWLER_THREADS`. If it is given as an integer-less string, the program exits with an error.

The only problem I could foresee is checking the vector of threads and how those threads are stored, caused by my lack of understanding of C++ and C++ vectors.

A further optimisation could use condition variables to notify waiting threads of the safe data structures being available, because currently the only thread-safe mechanism implemented is a mutex for each structure. Thus, some threads might exit early when the work queue is empty but not all work is done yet.

Furthermore, since a queue is used for the work, the program could be optimised by accessing the queue both while one thread pops and another thread pushes, which would increase efficiency because currently the whole data structure is locked with a mutex every time a thread accesses it, and the same goes for the unordered map previously called *theTable*.

2. Build and Runtimes (Sequential, 1-thread)



```
OpenSSH SSH client
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> make
clang++ -Wall -Werror -std=c++17 -o dependencyDiscoverer dependencyDiscoverer.cpp -lpthread
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> time ./sequential -Itest test/*.{c,y,l} > temp
real    0m0.046s
user    0m0.008s
sys     0m0.019s
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> time CRAWLER_THREADS=1 ./dependencyDiscoverer -Itest test/*.{c,y,l} >
temp
real    0m0.053s
user    0m0.010s
sys     0m0.018s
2467273s@stlinux03: /users/level3/2467273s/sp/ae2>
```

3. Runtime of Multiple Threads

```
OpenSSH SSH client
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> time CRAWLER_THREADS=1 ./dependencyDiscoverer -Itest test/*.{c,y,l} > ^
temp
real    0m0.055s
user    0m0.011s
sys     0m0.017s
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> time CRAWLER_THREADS=2 ./dependencyDiscoverer -Itest test/*.{c,y,l} > ^
temp
real    0m0.030s
user    0m0.009s
sys     0m0.018s
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> time CRAWLER_THREADS=3 ./dependencyDiscoverer -Itest test/*.{c,y,l} > ^
temp
real    0m0.025s
user    0m0.006s
sys     0m0.023s
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> time CRAWLER_THREADS=4 ./dependencyDiscoverer -Itest test/*.{c,y,l} > ^
temp
real    0m0.024s
user    0m0.012s
sys     0m0.021s
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> time CRAWLER_THREADS=6 ./dependencyDiscoverer -Itest test/*.{c,y,l} > ^
temp
real    0m0.020s
user    0m0.012s
sys     0m0.020s
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> time CRAWLER_THREADS=8 ./dependencyDiscoverer -Itest test/*.{c,y,l} > ^
temp
real    0m0.020s
user    0m0.014s
sys     0m0.022s
2467273s@stlinux03: /users/level3/2467273s/sp/ae2> _
```

CRAWLER_THREADS	1	2	3	4	6	8
	Elapsed Time, ms	Elapsed Time, ms	Elapsed Time, ms	Elapsed Time, ms	Elapsed Time, ms	Elapsed Time, ms
Execution 1	55	30	25	24	20	20
Execution 2	48	31	28	22	20	20
Execution 3	46	31	25	23	21	20
Median	48	31	25	23	20	20

a) Additional cores and threads

By adding additional threads, the program is able to utilise concurrency to look up multiple dependencies simultaneously, making the simple program run much quicker than its sequential or single-threaded alternatives.

b) Variability of elapsed times

As more threads are added, however, the marginal efficiency gained decreases, analogous to the economical law of diminished returns, caused by threads waiting to access the thread-safe data structures and unable to do any work in the meanwhile. By adding even more than 8 threads it might start to become slower because of the additional computing power needed to make more threads without the benefit of added efficiency.