CIS 520 - Project 4 Design Document

Pthreads

Software Architecture

We designed our pthreads implementation for this project as a single C program called `scorecard-pthread` that takes two arguments: first, the filename to read lines of text from, and second, the number of threads to use. It can be called by: `scorecard-pthread <filename> <threads>`. The first thing the program does after checking for correct arguments is get the size of the passed file and read the file into a buffer array of the same size. After that the program calculates what section of the buffer each thread should read from and starts the number of requested threads. Each thread then reads the highest ASCII value from every complete line in its section of the buffer and stores its results in another buffer. The program waits for each thread to finish its work and, when each one does, it gets the results from that thread and stores them in a new string buffer for all the threads' results, before finally printing all the results.

Each thread gets information about the buffer and the work it needs to do through a custom `thread_info_t` struct. When each thread starts, it is given a start and end index in the struct. First it changes these indices to be at the next newline after each initial given index. Then the thread starts finding the highest ASCII value for each line between the new start and end index in the buffer and stores the value of each line in a results buffer. The results buffer is reallocated during runtime based on the number of lines being read by the thread.

The program was compiled and run on Beocat running the Linux 5.14.0 kernel with the Red Hat GCC 11.4.1 compiler.

Performance Analysis

We tested the program on Beocat with a 1.7GB text file of Wikipedia articles which stored one article per line. We ran it with 1, 2, 4, 8, 16, and 20 cores and threads, each with the same Wikipedia text file. We used hyperfine (a cli benchmarking tool) to benchmark the program and calculate average runtime by running the benchmark 40 times at each core/thread count. We also measured the program's memory use from 5 runs at each core/thread count and averaged it. Here are the results of the benchmarks and memory measurements.

-	1 Thread: Time (mean ± σ)	6.466 s ± 0.253 s	Avg. 1.683 GB
-	2 Threads: Time (mean ± σ)	4.058 s ± 0.294 s	Avg. 1.683 GB
-	4 Threads: Time (mean ± σ)	2.480 s ± 0.038 s	Avg. 1.683 GB
-	8 Threads: Time (mean ± σ):	1.779 s ± 0.030 s	Avg. 1.683 GB
-	16 Threads: Time (mean ± σ):	1.288 s ± 0.034 s	Avg. 1.683 GB
-	20 Threads: Time (mean ± σ):	1.298 s ± 0.047 s	Avg. 1.683 GB