

Project #1: Parallel Programming Using Threads

Objectives:

- Reason about and design a parallel algorithm;
- Experiment with parallel programming using threads;
- Evaluate the performance of parallel programs.

Due date: March 3rd, via Digital Drop in BlackBoard.

For this project, you are invited to design and implement a program that uses multiple threads to count how many times each word appears in a large text file. Please use either the Pthreads library (with C or C++) or Java Threads for this project. (*Note: the next assignment will require you to repeat this exercise using the MPI library in C.*)

Deliverables:

- A document that explains the design of your parallel program and evaluates the performance of your implementation in terms of running time, scalability, and CPU efficiency. Notice that there are various ways to decompose the problem into parallel components:
 - Each of the k worker threads gets its $1/k$ chunk of the file. After all threads finish counting, the results are aggregated. For example, if threads 4, 12, and 13 encountered the word “the” for 15, 32 and 101 times, respectively, the aggregated result will output 148 times as the final count for that word.
 - Each thread is responsible for a fraction of the dictionary. Note that in this case the aggregation of results at the end is straightforward: basically, only one process counted the words starting with “pre”.
 - Threads are kept in a pool of predefined threads and whenever idle, they start another task. Tasks can be defined as segments of the file of sizes independent of the number of threads.
 - Other ideas may be possible, as well.For the performance evaluation you may use other machines than the ones in the C4 lab (if you have access to multi-core machines, for example). However, your code must be able to run on the C4 lab machines for me to test it.
- A README file that shows me how to compile, execute and understand the output of your code. In addition, a makefile would be highly appreciated.
- Your code.