Wentworth Institute of Technology

COMP3475 – Operating Systems

Final Project

The goal of the project is to use threads to get performance gain on a uniprocessor or on a multicore processor. The basic idea is to divide the problem into component parts, give each task or part to a separate thread and then combine the results to get the complete solution. The strategy is called the *divide and conquer* strategy.

A single task, sorting a file, is divided into subtasks which are delegated to separate threads. Merge-sort, in which the array to be sorted is divided into smaller arrays, is a divide and conquer algorithm. Each small array is sorted individually, and the individual sorted arrays are merged in pairs to yield larger sorted arrays. The pairwise merging continues until completion.

Generally, merge-sort starts with small arrays, which can be sorted efficiently with a simple algorithm such as qsort, as shown below.

Array to be sorted

qsort

Wait (thread 1)

Merge

Thread 0 wait (thread 2)

qsort

Merge

Merge

qsort

qsort

Thread 1

Wait (thread 3)

Thread 2

Thread 3

After the sort phase, the sorted subarrays are merged in pairs to generate larger sorted arrays. The merging continues until completion.

The environment and input for this experiment consists of:

1. Linux OS
2. pthreads
3. Memory mapped file so Merge-sort can sort and merge the file in memory
4. A test file consisting of 128, 64 byte records for testing and debugging. Each record has an 8 byte key and 56 byte data.
5. A file consisting of one million fixed size records of 64 bytes each to run the experiment

The program finds the number of cores, N, programmatically and displays it to the user.

Run the experiment with 2, 4, 8, …. N threads and display the elapsed time for each run. If the number of cores is <= 2, you can stop the test after 8 threads. The number of threads used must be a power of 2.

Running the experiment

Name your application sortMerge. The syntax of the command to run the program is

sortMerge <number of threads> <filename>

1. Run sortMerge with the test file and record the elapsed times. Save the sorted test file.
2. Run sortMerge with the one million record data file and record the elapsed times.

Submissions

1. All Source files (.c and .h)
2. Output file, showing the user interactions and the elapsed times for the various runs.
3. The sorted test file.