Design and Implementation of Project 4

This project aims to realize a merge sort by using multiple threads and synchronization.The program designed for this project contain two major part, merge function and main function.

The merge function is used to sort and merge the subset, the arguments of this function contain 3 parameters which represent the left, mid and right points of the subset. Each merge function will be put in a thread. Thus, the battier is implemented in this function.

The main function will read in the file, and use loop to execute the merge sort in different rounds. In each round, the position parameters will be calculate and assigned to merge functions in threads. Once all threads in single round are finished, the barrier will release the program to the next loop, which is the next round. After the sort finishes, print out the sorted array.

In addition, the important global variables in this program include an integer array for the input numbers; an array for containing threads' tids; a pthread mutex; a pthread condition and a counter named "barrier".

(1) How you implemented the barrier?

Set a global variable "int barrier" as a condition variable to represent the number of thread in current round. Once a thread finishes sort, the barrier will pause one, and then use a while loop to check whether all threads in this round are completed. If this is not 0,the thread will be blocked on the condition variable. If this is 0, call the cond.broadcast function to release all threads. Since the change of global variable happens, this section should be protected by a mutex lock. Also, the barrier should be used in main thread at the end of a round.

(2) How you tested your barrier?

To test the barrier, I print the number of the global variable “barrier” and the tid of current thread. By observing the output, I can found that when the barrier is not 0, the current thread is blocked, and in a single round, the number of blocked thread is our desire, which means the barrier is successful.

(3) Whether or not your main thread participates in comparisons?

No, the main thread is waiting the completion of all thread in this round. It actually doesn't participates in comparisons.

(4) How you organized (in memory) the intermediate results?

A dynamic array is created for storing the intermediate results in the merge function. Once the merge is completed and this part memory will not be used, call the free() function to release the memory.