ASSIGNMENT-2

PNVSSK HAVISH,EE16BTECH11023

November 26, 2018

1 Design specifics of the Program

This assignment is a basic implementation of multi-threading and a comparison with multi-processing when it comes to parallel processing. The task was to calculate mean, standard deviation and median of a given data using 1.) threads and 2.) process. Below are the sub-sections related to the low-level implementation of thread and process codes.

1.1 Process approach

The given data is stored in *struct proc_args* along with the mean, standard deviation and median. A shared memory file descriptor is created using the function $shm_open()$. This memory is of read and write format. The function $ftruncate()$ is used to truncate this shared memory up to 4096 bytes. Using the $mmap()$ function, shared memory is created and *ptr* is used to point to that specific address. *ptr* is of the data type $proc\_args^*$. In the shared memory, we store the address to the *struct* of the given data. Now, the $fork()$ function is used to create a child process. This will return a pid of 0 for the child process and a non negative integer. In the child process another fork is called, which creates another child process for this child. The last created child calculates the mean of given data. The $wait(NULL)$ function makes the parent process wait until the child execution is completed. So, the first child process waits for its child to be completed and then calculates the standard deviation of the data. After this, the parent computes the median of the data by sorting the entire array. To calculate the execution time, the code from first assignment was used.

1.2 Threads approach

Even here, the given data is stored in *struct thread_args*. The mean, standard deviation and median are stored as global variables which are shared across all the threads. The *chrono* library was used to calculate the execution time of the code. The *pthread_create* command is used to create three threads each for calculating mean, standard deviation and median. First, the mean thread and median thread are executed parallely. The standard deviation thread waits for the completion of mean thread so that it can use the calculated mean. As for the median thread, it copies the contents of the array to another array so that this does not get intermixed with the mean thread(these two threads run parallely). *pthread_join* is used to get the results to the parent.

2 Analysis of the output

Here, the threads code seems to take nearly the same time as process code. This might be because we are actually copying the contents of the array to another temporary array in the threads code whereas we are not doing that in the process code. The plot is attached in the zip file. It was plotted for n values between $2.5e5$ and $20e5$ with a step of $2.5e5$.