POSIX Threads- Project Assignment II

Siddhartha Srinadhuni Enugurthi Madhukar

9508286854 9506077537

sisr16@student.bth.se maen16@student.bth.se

Pthreads:

Threads can be used to implement parallelism in shared memory multiprocessor architectures. POSIX threads or Pthreads is a standard and benchmarking Application Programming Interface (API) specified by IEEE POSIX 1003.1c. Implementations of the standards are usually referred to as the POSIX threads[1], [2] . <pthread.h> is the library that has all the pthreads functions. Out of which pthread_exit(), pthread_create(), pthread_join() are a few that are part of this library [1] . The project task is to implement Gaussian elimination using Pthreads.

Gaussian Elimination:

- Gaussian elimination is used to solve a system of linear equations. The start of the Gaussian elimination is the conversion of the given linear equations into matrices wherein the coefficients of the equations represent the elements of the matrices.
- In this task, after obtaining the upper triangular matrix and comparing the values obtained to the system of linear equations, Gaussian elimination is implemented.
- Parallel threads having shared address space require certain help from pthreads as the threads involved have to be synchronised, hence pthreads are used in this task to apply Gaussian elimination

Implementation of Gaussian Elimination using POSIX threads:

- *Init_Default()* function has the size of the matrix and initialization type declared in it while *rand()* initializes the elements in the matrix randomly.
- *pthread_barrier_init()* function is used to initialize the barrier [1].
- *pthread_create(&cal_thrs[i], NULL, (void *)&cal, (void *)i)* is used to create the threads.
- The start routine *void cal(void *thread_id)* is executed by the thread when it is created.
- *pthread_mutex_init(&mutex_variable, NULL)* is used to initialize the mutex variable while *pthread_mutex_lock(&mutex_variable)* is used to lock. Unlocking can be achieved by *pthread_mutex_unlock(&mutex_variable)*[1].
- *pthread_once(&once_control, Syn_Init)* is used to call the initialization routine and only after this the mutex variables can be unlocked.
- The rows in the considered matrix will now undergo assigning to a thread and will undergo the division step.
- *pthread_barrier_wait(&barrier)* function will result in waiting of the section of code until the threads get terminated. The threads will get terminated after subjecting to division step.
- *pthread_barrier_destroy(&barrier)* function is used to terminate the barrier that has been created and is done after all the computations have been performed [1].
- *Print_Matrix()* function is used to print the matrix.

Implementaion of Gaussian elimination through pthreads speeds up the execution by a huge margin. This also yields an enormous change of improvement on the CPU and hence takes less time than the sequential counter-part.

References:

[1] Blaise Barney, Lawrence Livermore National Laboratory, “POSIX Threads Programming.” [Online]. Available: https://bth.itslearning.com/ContentArea/ContentArea.aspx?LocationID=6423&LocationType=1. [Accessed: 26-Dec-2016].

[2] “POSIX Threads,” *Wikipedia*. 16-Oct-2016.