Homework 2 Report

Group Members:

Name: Pranay Morye **UCINetID:** pmorye

Name: Junaid Magdum **UCINetID:** jmagdum

System Description:

1. Processor: Intel(R) Xeon(R) CPU X5680 @ 3.33GHz
2. RAM Memory: 96 GB RAM
3. Operating System: Linux (Ubuntu)

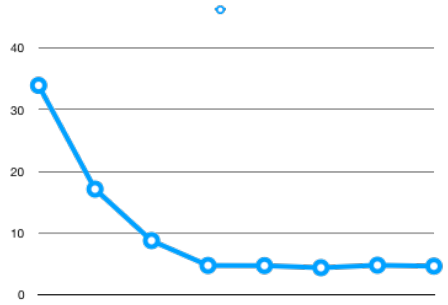
Execution Time:

1. Single Thread Program:

   Execution Time = 33.000000 seconds

   Multi-Thread Program:

Number of Threads	Execution Time
1	33.931065
2	17.114946
4	8.766892
8	4.741339
16	4.374492
32	4.374481
64	4.780748
128	4.629139



Analysis:

Speedup of program in Multi-threaded version: As we can see from the data, the performance of the program initially increases manifolds with the increase in the number of threads, however it gradually slows down to around 3.8 to 4.8 seconds seconds. What we can infer from this is that even though in theory, doubling the number of threads should halve the run-time, this does not factor in other tasks such as thread-management done by the program, thread creation, deletion, number of cores available, etc.

Thus, initially when we increase number of threads from 1 to 2, we can see halving of run-time because we have more than 2 cores, but eventually this run-time halving slows down with the doubling of the number of threads because of the above mentioned factors. In fact, we can also see some slowdown as we increase threads because this is where the thread-management overhead becomes more than the benefit of having more threads.

We can conclude that we need to figure out an optimum number of threads to run our program for it to run efficiently.

Speedup Formula:

Speedup = Sequential Program Execution Time / Best Multi-thread Program Execution Time

Speedup = 33.93165 / 4.374481

Speedup = 7.75672

Single-threaded VS Multi-threaded: As we can see from the data above, the single-threaded version runs a little faster than the multi-thread version with only 1 thread. The reason for this is that the multi-thread version has to do thread creation, management and destruction even for that 1 thread. So, even though both programs have technically the same run time complexity, we can see that the multi-thread program takes more clock-time.

Simultaneous Multi-threading: Simultaneous Multi-threading is basically creating virtual processors out of a single processor, as different instructions use different parts of the processor. So if we can split our program such that the instructions use different parts of the CPU, then enabling SMT will lead to decrease in runtime performance.