**Assignment 1**

Quan Zhou

1. Discussion

The CPU has 2 cores and can support 4 threads at most. Thus, maximum number of threads is 4. As for getting execution time, the value of *time(time\_t\* timer)* is in the order of seconds. So the length of vectors is set to 108 to make the timing more accurate. The final runtime was calculated as the average of 20 runs.

1. Runtime vs. Number of Threads

The program was run on a Win8 64 bit laptop with two cores 2.4GHz CPU and 8GB memory. With array size being 108, graph of computation time and number of threads is as below.

The result shows increasing number of threads does not reduce runtime, which is not what was expected. The expectation was after doubling number of threads, there would be a significant reduction in runtime, not only around 10%.

1. Code

#include <iostream>

#include <thread>

#include <vector>

#include <time.h>

#include "mingw.thread.h"

using namespace std;

int alpha = 2;

// compute X = X + alpha \* Y

void worker(double\* x, double\* y, int indexL, int indexR){

//cout << "This is thread " << indexL << " ~ " << indexR << endl;

for (int k = 0; k < 10; ++k){

for (int i = indexL; i < indexR; ++i){

x[i] += alpha \* y[i];

}

}

}

int main()

{

//configuration

int arraySize = 100000000;

int numThread = 4;

double\* x = new double[arraySize];

double\* y = new double[arraySize];

double runtime = 0;

time\_t t0 = time(NULL);

// run 20 times and take average runtime

for (int k = 0; k < 20; ++k){

// initialize arrays

fill(x, x+arraySize, 1);

fill(y, y+arraySize, 0.5);

// divide tasks for each thread

int bounds[numThread+1];

bounds[numThread] = arraySize - 1;

for (int i = 0; i < numThread; ++i){

bounds[i] = 1.0 \* arraySize \* i / numThread;

}

// start threads

thread t[numThread];

for (int i = 0; i < numThread; ++i){

t[i] = thread(worker, x, y, bounds[i], bounds[i+1]);

}

// join threads with the main thread

for (int i = 0; i < numThread; ++i){

t[i].join();

}

}

time\_t t1 = time(NULL);

runtime = t1 - t0;

cout << "Array Length: " << arraySize << endl;

cout << "Number of threads: " << numThread << endl;

cout << "Runtime: " << 1.0 \* runtime / 20<< endl;

delete[] x;

delete[] y;

return 0;

}