Hao-Hsiang Liao (haoli436)
Santiago Pagola (sanpa993)

TDDC78 Lab 3

Stationary Heat Conduction Using OpenMP

Table of Contents

Hao-Hsiang Liao (haoli436)
Santiago Pagola (sanpa993)

Introduction

The goal of this assignment is to solve a stationary heat conduction problem on a shared memory computer (a single compute node on triolith.nsc.liu.se), using OpenMP. A serial code for solving the problem is given in the file laplsolv.f90, which is found on the course home page4, and should be used as a starting point for the implementation.

The final parallel implementation, named laplsolv-parallel.f90, should produce the same results as the sequential one. Let's describe our solution.

Solution

In order to implement this program in parallel, we split the workload among a number of threads. The number of threads is read from the environment variable OMP_NUM_THREADS, which is parsed into nr_threads in our program. Every process is implemented by a thread. In other words, this program has a higher efficiency if we have a higher nr_threads.

First of all, we set a variable, ratio, to express how many columns should be processed by a thread. And then we set start_pos and stop_pos to every thread so that every thread can know which column should start and stop. For example, assume n = 60, nr_threads = 3, then we can get start_pos(0) = 1, stop_pos(0) = 20, start_pos(1) = 21, stop_pos(1) = 40, start_pos(2) = 41, and stop_pos(2) = 60.

Secondly, we set the matrices tmp1 and tmp3 to store previous column's data of start_pos and next column's data of stop_pos so that we can do the calculation correctly. Besides, it is worth mentioning that when every thread is processing its final column, or stop_pos(my_id), it should be separated because T(1:n,j+1) has been changed by next thread. Thus, we have to use tmp3 to describe the equation

T(1:n,j)=(T(0:n-1,j) + T(2:n+1,j) + tmp3(0:n-1,my_id) + tmp1(0:n-1,my_id)) / 4 .

Furthermore, we set tmp2 and my_id as private valuable and T, tmp1, and tmp3 as shared variable. We also reduce the global variable error modified by every thread to the maximum value among all threads via the reduction (max:error) directive.

Results

Fig.1 shows the elapsed execution time when simulating a different number of threads. As expected, the more threads we use, the smaller the execution time becomes. Note that in order to achieve this, a dedicated node on Triolith must be requested in order not to interfere with other running jobs.

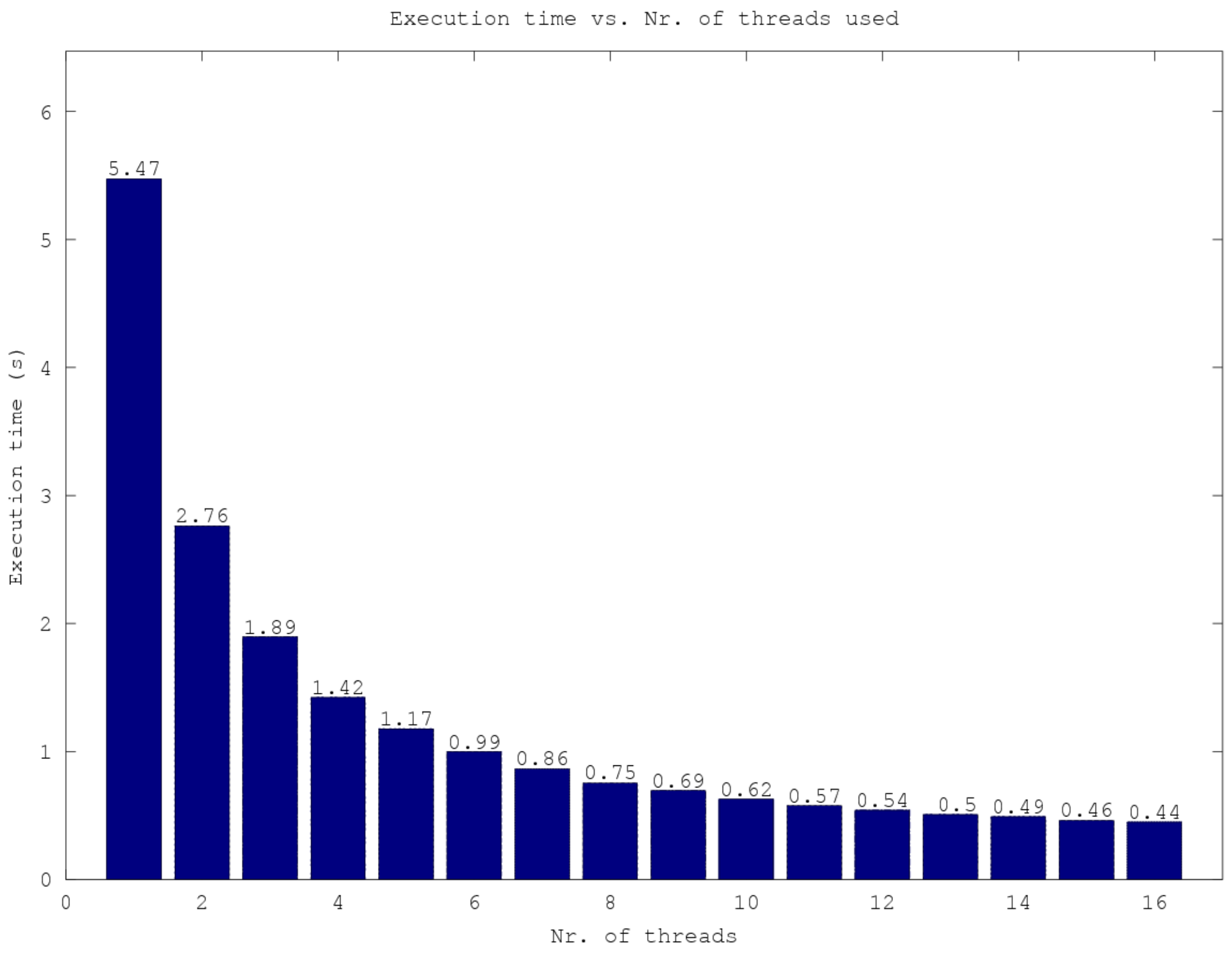

Fig. 1: Execution times vs. number of threads used in the simulation.