Due Date: Friday, 24, 2017 - 5:00 pm
Your name:

INTRODUCTION

In this lab you will be working with OpenMP. There are three parts to the lab with increasing difficulty. If you are taking the class for two credits you only need to do the first part of the lab.

GETTING STARTED

Begin by logging into cees-rcf `ssh -Y cees-tool-7`. Begin by cloning your git repository. In this lab you will be parallelizing a code that resizes a regular cube (model) through linear interpolation. The basic algorithm assumes that you are wanting an output (data) space that is larger by an integer *rescale* factor. The `forward` does outer loops over the smaller input domain. Think of looping over a series of large boxes. To fill in the data for this portion of the model we can use linear interpolation using the current and next value along each axis.

PROBLEM 1: PARALLELIZING THE FORWARD [5 PTS]

Compile the code and record the time to execute the serial code. Introduce OpenMP parallelization to the forward. Begin by parallelizing over the i1 axis. Change the number of threads you are using from 1,2,4,8,16 using `setenv OMP_NUM_THREADS`. Calculate the speedup in each case.

Change the code to parallelize of over the i2 axis. Run the same scaling test. Did the choice of loop parallelization matter? Why or why not? How close did you come to perfect scaling? Speculate on why achieved your scaling?

Parallelizing the adjoint is more challenging problem. If you use the same approach as you did in the forward you are likely to get the incorrect result. Why?

PROBLEM 2: PARALLELIZING THE ADJOINT [5 PTS]

One way to parallelize the adjoint is by introducing locks and/or single statements. Choose one of the approaches to parallelize the adjoint. Try to at least beat the serial code's performance time. Perform the same scaling test and comment on the results.

PROBLEM 3: RED/BLACK ORDERING

One approach to avoid the race condition of the previous section is to use what is often referred to as red-black ordering, or more generally a colored ordering approach. The basic concept of these methods is to break a parallel problem which has a race conditions into multiple parallel problems each of which do not have a race condition.

Hint: To understand how a red-black ordering system might work imagine if we are attempting to do 1-D interpolation using the same algorithm. We could note that our first cell update model points 0 and 1, our second cell 1 and 2, and our third cell 2 and 3. We could modify our loop to parallelize over all odd number cells without introducing a race condition then parallelize over all of the even cells.