Homework 4.5: Report

Dante Buhl

May 16, 2024

Question 1: PCAM for Finite Difference Stencil Problem

P - Smallest task is computing the derivative / "other physics" for one point in the domain. Different processors can work on updating different points concurrently.

C - Tasks will have to communicate with adjacent tasks after every update in order to get information needed in the stencil for the next update.

A - In order to decrease the communication versus computation ratio, we can agglomerate the tasks of updating multiple points in the domain into one processor. We can minimize the ratio of comm v comp, using some basic calculus. This minimization will depend on just how long the "extra physics" of the problem is. For example, if the computations is rather expensive, this might affect the optimal agglomeration size.

M - If for example, the agglomeration size is too large or too small in order to partition the domain across the number of processors evenly, there are several ways to handle this. Some examples, would be for some processors to take a slightly larger domain (though this might break symmetry), or some processors may take an entire extra agglomerate compared to the other processors (bad if each agglomerate takes a considerable amount of time), or each processor can take a slightly suboptimal agglomeration in order to load balance the tasks evenly.

Question 2: PCAM for Node Search Problem

P - Smallest task is to search down to the bottom of a branch, i.e. a leaf and checking the area needed by that orientation. Different processors can investigate different leaves.

C - Tasks will have to communicate with other tasks in order to see which leaf minimizes the area.

A - If a particular branch doesn't have too many leaves/branches originating from it, then it might make sense for a single processor to search that branch.

M - In order to load balance on processors, it might make sense to probe 1 or two layers at each iteration to see how many leaves that node has. Overall, I think the best way to do this problem, is to have the root cpu to a maximum depth search, at each level picking the left most leaf. Then going back to the top, partitioning nodes to search into agglomerates for some arbitrary number of processors to compute. Then to go down 1 layer at a time, communicating at each layer, to see if anybody is above the current minimum in order to prune branches. Then tasks and agglomerates might be dynamically allocated to processors so that the load balancing would be updated and constant as the search progresses. If ever a new minimum is found at a certain layer search, then this should be broadcasted to the other processors.