hw 3
jack ziesing
5/6/2014

Data Reorganization:

1. The general goal of data reorganization and the use of a pattern to reorganize it is that in the long run we want to get better performance. Often it is the transfer and communication of large pieces of data that restricts performance, so in parallel programming it is encouraged to organize your data in a way that best fits a pattern so you can maintain relatively good performance.

2.

a. It seems that this would run into some problems because of the programs data struct and thread concurrency. Even if its divided into a bunch of different threads (⅓ each) we still need to access it one at a time so latency occurs

b. This makes the performance closer to a serial implementation rather than a fast parallel implementation. The fact that memory needs to be accessed one at a time causes threads to wait which is more like a serial implementation.

c. To solve this in parallel you would want to use a stencil pattern mapped as much as possible with a radius of 3.

3. Since a stencil works with its neighbors, you would want to reorganize the data into chunks that include its neighbors. Maybe you break it down into a tree off chunks of neighbors or some other structure, then you compute and add the pieces together building the final data set in the end.

Stencil:

1.

a. The ghost cells refer to the outer region of each of the stencil blocks, the ones that define the boundaries of the stencil block. In this case it would be cells on the outer edges of a 500x500x500 area.

b. 250x250x250 ⇒ 1000/250=4 and 4x4x4=64 so 64 threads
125x125x125 ⇒ 1000/125=8 and 8x8x8=512 so 512 threads

c. As we compute with more threads and smaller stencil block sizes, our stencil creates more ghost cells or in turn smaller halo sets for each stencil block. This causes a couple things: data redundancy, and computation redundancy. It is nice because we are able to compute the chunks really fast because all the data is there for the computation, the issues lie with delivering copies of the data to each processor, and communicating and computing each processors computations back into a final solution. We also end up computing stuff more than once, which isn't the best. I think when the ration of ghost cells to non ghost cells gets over a

certain point (maybe around 2) then we start to lose performance, but if the ration is kept to a reasonable limit, I believe you probably will always gain performance.

2. A deep halo means redundancy, less communication, and independence. You are making more copies of the same data as you increase the halo depth which is harmful to the amount of space available and your speed (could use processor for other things). Though you may have lots of threads running a stencil with deep halos, you may get worse performance than serial or less threads due to all the redundancy.

3. For this I would use a recurrence pattern. In this practice the nested loop would be parallelized over n-1 dimensions with a hyperplane sweep. You also might be able to use a fork join in this instance.