**Homework 3 - Data Reorganization Pattern & Stencil Pattern**

Data Reorganization

1. In general, what is the purpose of data reorganization and of having a parallel pattern to do it?

The performance bottleneck in many applications is due to data movement rather than computation. For data intensive application, it is a good idea to design the data movement first.

Data reorganization can minimize the data access time by increasing the locality so the application does not have to transfer data across memory layers/ networks.

Having a parallel pattern to do data reorganization can utilize the parallelization resources in order to achieve a better run time and hence optimized the application.

2.

(a) Do you see any issues that might arise concerning data access?

Each set of 1/3 of the threads are only updating the specific property and only concern about that property, the additional properties are not needed;

But when the Cells are cached, the additional properties are also cached, those memory space could have used on caching more of the specific property, which the threads are updating.

And when getting the surrounding Cells, the Cells might be in a different array. (e.g: atmosphere[4][4][4] and atmosphere[5][4][4]), then the application needs to read from multiple arrays.

(b) What affects could there be on performance?

When the Cells are cached, the additional properties (those properties not needed for updating a specific property) are also cached, those memory space could have used on caching more of the specific property.

The latency when reading in multiple arrays would affect the performance when the threads are waiting for I/O.

(c) How would you resolve the issues using a parallel data reorganization pattern? Explain.

We can gather the specific property the threads are working on, and put them into a 3-dimensional array Prop[1000][1000][1000].

After we obtain the array Prop, we can gather the needed area (the values needed for the computation) into one array, so that when we do the computation, we don’t have to access multiple arrays

3. Do you see an opportunity for using a data reorganization pattern in preparing for a stencil computation? Explain.

Yes, data reorganization can help with the stencil computation.

For instance, if we gather only the element needed for the current sub-problem to apply the stencil to minimize data accessing; Or

We can partition the data so then we can assign each subset of data to different threads that the problem can be solved in parallel. (to eliminate the problem when some thread is wait for another thread on the same data)

Stencil Pattern

1.

(a) Describe what the ghost cell regions will be like in this computation.

For a 500x500x500 chunk of data, there would be a number of:

500 \* 3 (sides with shared boundaries) = 1500 ghost cell

For 8 chunks of data:

1500 \* 8 = 12000 ghost cells.

The ghost cell regions would be the overlapping regions between each neighbors chunk (that are 1 cell away from the chunks).

(b) Suppose we would like to use more threads in this computation. How many threads would we need if we wanted each "chunk" to be 250 x 250 x 250? 125 x 125 x 125? (Show work)

64 threads = 4^3 = 1000^3 / 250^3; for making chunks to be 250 x 250 x 250 cubes

512 threads = 8^3 = 1000^3 / 125^3; for making chunks to be 125 x 125 x 125 cubes

(c) Using your answers from the previous questions, describe how the halo regions change relative to the rest of the data. Consider the ratio of ghost cells to non-ghost cells. How does this change effect our computation?

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of cell in halo | | Ratio |
| 250 x 250 x 250 | 8 \* (6 \* (250^2) + 12\*250) +8 (those that have all 6 neighboring sides) +  24 \* (5 \* (250^2) + 8\*250)+4 (those that have 5) +  24 \* (4 \* (250^2) + 5\*250)+2 (those that have 4) +  8 \* (3 \* (250^2) + 3\*250)+1 (those that have 3) | 18108216 | 18108216/(1000^3-18108216)  ~= .0184 |
| 125 x 125 x 125 | 216 with 6 neighboring sides  12 \* 6 with 5  36 \* 6 with 4  8 with 3 | 160070456 | 160070456  /(1000^3 – 160070456  )  ~= 0.191 |

The halo region increases as the chunk size decreases;

The ratio of ghost cells to non-ghost cells increases as the chunk size decreases;

As the ghost cell number increases, the memory demand is also increased; also the communication required to update the ghost cells between threats would create an overhead.

2. Given a halo region in a stencil computation, why might we want to increase the "depth" of our halo region? Why might making this "depth" too large be a bad thing?

We might want to increase the depth of our halo region to allow us to perform several iterations without stopping for a ghost cell update, and hence, reduce number of communications between threads. However, if the depth is too large, there would be more redundant computation and more memory used to compute the result.

3.

We can use a separating hyperplane

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| j  i | 0 | 1 | 2 | 3 | … |
| 0 | 0 - iteration | 1 | 2 | 3 | 4 |
| 1 | 1 | 2 | 3 | 4 | 5 |
| 2 | 2 | 3 | 4 | 5 | 6 |
| 3 | 3 | 4 | 5 | 6 | 7 |
| … | 4 | 5 | 6 | 7 | 8 |

By identifying the plane that cuts through grid of intermediate results, we can still perform parallelized computation with the dependencies hyperplane introduced.