<u>Homework 3</u>

Data Reorganization

1. Performance is often constrained by data transfer rates as processors can easily outpace data movement on systems. Memory hierarchies and caching techniques improve locality and try to minimize costly data transfers but reorganizing data can reduce risk of a memory bottleneck especially in data intensive applications. Furthermore, programs computing in parallel will need to retrieve data for needed by each thread of execution so it makes sense to retrieve the data in parallel.
2.
    a. 3 groups of threads writing to the same cell which might cause collisions with different copies of the struct in their thread-local cache
    b. To make the update deterministic, locks would probably needed to modifying the same struct at the same time which could cause idle threads waiting for locks. It could be possible to make each group of thread never collide starting at different indices but it may be difficult to find a balance of indices close enough to benefit shared caching and far enough to avoid collisions and cache coherence.
    c. Make seperate arrays for velocity, temp and ozone. No collisions and higher chance for data to be in cache as threads increment down their arrays.
3. Yes, since computations require multiple inputs it is important to be able to quickly gather the needed neighborhood. Inputs also overlap and can be reused so the data should also be reorganized in a way that inputs transferred into cache by a different neighborhood computation is used by subsequent computations.

Stencil Pattern

1.
   a. The ghost cell region would be the surface cells 502 x 502 x 502 cube surrounding each of the 8 partitions which would be 1506068 cells per partition minus the cells out of bound.

   b. 1000 x 1000 x 1000 = 1000000000

      250 x 250 x 250 = 15625000

      1000000000/15625000=64 threads

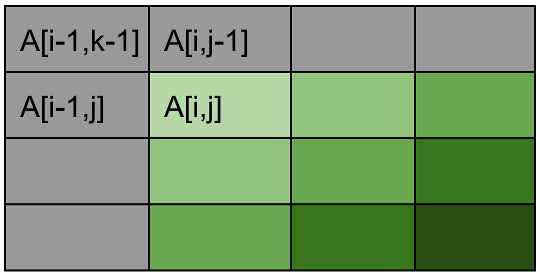
      125 x 125 x 125 = 1953125

      1000000000/1953125=512 threads

   c. Increasing the threads increases the size of the halo. With 64 threads, each partition has 378008 ghost cells (approx ~42 cells per ghost) with a total of 24192512 in the halo minus the cells out of bounds. With 512 threads, each partition has 95258 ghost cells (approx ~20 cells per ghost) with a total of 48772096 in the hall minus the cells out of bounds. 8 threads had approximately 82 cells per ghost cell so the ratio of non-ghost cells to ghost cells decreases. Increasing the thread count adds parallelism but lowers efficiency since more communication is required for ghost cells and there is a greater demand on memory transfers. Depending on these factors, performance for the computation could improve or decline.

2. Increasing the depth should be used when the limiting factor is latency of communication, rather than bandwidth or computation since it would decrease communication and allow for more independence. Making the depth too large could lead to redundant computations amongst threads and increased memory use.

3. Use a separating hyperplane the cuts through intermediate results such that computed values on a one side of the plane. For this particular operation, the computation of A[i][j] depends on array entries to the left(i-1,j), above(i,j-1) and upper-left (j-1,i-1). So the separating hyperplane is a diagonal line, up and to the right 1 cell. Pictured below, after the computation of A[i,j], A[i,j+1] and A[i+1,j] have their dependencies ready.



. Grey is outside iteration