PRESENTATION OUTLINE: XRAY Reconstruction on GPUs

Aditya Maheshwari
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
adityamaheshwari@scs.carleton.ca

November 19, 2019

1 Introduction (XCT Problem)

- XCT imaging widely applied in non-destructive testing and contact-free situations
- Medical imaging, age determination, and industrial materials testing
- X-ray source emits X-rays with an initial intensity which attenuates as it goes through the object
- The detector collects these rays on the "other side" as projection measurements. As the source and detector rotate around the object, we can get many views of the different "slices" of the objects
- For example, a detector can take 90 to 120 views of the slice (by rotating around the slice) and in each view take a tens or hundreds projection measurements

2 Data we are using for this talk

- Lotus Root with objects stuck in it - lotus root has texture of a potato and is composed of starch (sugar), which makes it interesting because these days humans eat a lot of starch...; it also has objects stuck in it like a pencil (carbon & wood), a chalk (calcium), ceramics (calcium; rectangular), and a match head (sulphur)
- Walnut Slice - contains a dense and layered shell enclosing which is roughly symmetric (like a skull) while the edible part is more scrambly
- (maybe one more)

3 Representations

- We will look at reconstructing one slice of the image (since what we finally see is in 2D

- In this case we can view all the projections for that slice as a matrix A_{ij} where one projection is represented by a row and the columns represent the voxels (pixels) of the image being reconstructed
- The whole problem then becomes $Af = g$ where g are the vectorized projection measurements, and f is the vector we want to reconstruct.
- The projection measurements should have one numeric value for each X-ray, and each X-ray is represented in one row of the matrix, so in a situation with m projections and n voxels total the dimensions are $A \in R^{m,n}$, $f \in R^n$, and $g \in R^m$

4 Optimization Problem

- We are essentially solving: $\mathbf{min}E(f)$

$$E(f) = D(g, Af) + \alpha R(f)$$

- $D(g, Af)$ is effectively a distance measure between the reconstruction and projection measurements and $R(f)$ is a regularizer where α is the weight
- This type of objective function can be iteratively solved with any iterative minimizer such as ICD or SGD!
- This method is known as Model Based Iterative Reconstruction (MBIR)

5 GPUs and Their Properties

- Several SMs connected by global memory; caches and shared memories too
- Each SM has many SIMD pipelines (INT, SP, DP)
- If a warp tries to access global memory, it coalesces memory accesses of threads within warp into one or more memory transactions
- Two types of irregular accesses result in memory collisions and non-coalesced memory access

6 Memory Issues in a GPU

- If more than one thread in a warp tries to write to the same memory address, only one thread can actually write to that address. That's a memory collision
- If different warps try to write to the same memory address, that's called a non-coalesced memory access
- While both memory problems cause issues, it's important to avoid collisions first and let the scheduler handle which warp writes to memory at what time

7 SGD and ICD

- To find the (local) minimum of the objective function, an iterative method usually involves taking a step proportional to the negative of the gradient of the gradient of the objective function
- Because our objective function is sparse (mainly because A is sparse), we can effectively parallelize using asynchronous parallel SGD/ICD
- SGD moves the entire vector being updated with a predetermined interval in the opposite direction of the gradient of the objective function per iteration, while ICD moves only along one dimension per iteration but with a step size proportional to the size of the gradient.
- This suggests that SGD will pick one projection measurement to update the image at each step, while ICD will pick a voxel to update the image at each step.

8 SGD & ICD Comparison

- SGD is effective because all voxels in a projection are updated for each projection and are immediately readable by all the other threads
- Furthermore, because we are updating multiple pixels in each iteration in theory, we can use a regularizer that enforces smoothness of the image to override any noise that can occur in the detection process
- ICD, on the other hand, can take longer because for each voxel we need to get which projection goes through it and then for each of those projections we once again need to see all the other voxels they go through to understand the impact on this one. This can potentially be fixed with parallelization
- Furthermore, even if we parallelize the searching process above, it's hard to guarantee smoothness between two consecutive voxels.

9 Memory Collisions

- We assign each projection to one thread in the GPU; usually the number of projections is extremely large
- Different projections will update different voxels; we do not want multiple threads updating the same voxel simultaneously
- It does not really matter here if we use atomic or non-atomic operations to make the update on each iteration; the goal is always to make sure we do not update the same memory address of the image we are reconstructing in the same iteration
- Therefore, we can view this as a graph partitioning (or even a clustering) problem where each projection is a vertex and each edge holds a weight that represents the number of shared voxels by the two projections in the same spot

10 Non-Coalesced Memory Accesses

- Once we are in a warp, we want to synchronize accesses to global memory as much as possible, meaning if two threads will need to access the same segment of memory (but not the same exact address), we want to them to require this at the same time as much as possible
- Those are known as non-coalesced memory accesses
- However, if we maximize this we will end up with a lot of memory collisions which are more degrading to performance; therefore we need a dual objective of first separating "very" similar X-rays into different warps but then regrouping "sort of" similar X-rays into the same warp
- Therefore the "easy" solution is to first separate our projections to reduce memory collisions as much as possible, and then within each subgroup once again create a separation that now groups similar memory accesses together
- Our final groups need a size equivalent to the size of the warp on the GPU (32), meaning we need to partition the nodes within each colour into sub-graphs
- This mapping in theory only needs to be calculated once and can be done offline (or in the case of using clustering maybe even online)

11 Summary of theory

12 Results - Single-Core

13 Results - Multi-Core

14 Results - GPU

15 Questions

References