A projection-based partitioning for tomographic reconstruction

Jan-Willem Buurlage (CWI)
Willem Jan Palenstijn (CWI)
Rob Bisseling (Utrecht University)
Joost Batenburg (CWI)

2020-02-13, SIAM PP20, Seattle
Outline

- Tomography, partitioning problems in imaging
- Previous work: GRCB algorithm
- Communication volume, shadows and overlaps
- Continuous model for load balancing
- Communication data structures
- Results and conclusion
Background
Tomography applications
Tomography
Reconstruction problem

- TODO big data sets, typical sizes, different acquisition geometries
- Distributed 3D volume over many GPUs, minimizing communication
Each combination source position and detector pixel defines a ray, in the solver each ray is traced through the discretized 3D volume.

Tomographic reconstruction problem deal with anywhere between $10^9$ and $10^{11}$ rays.
Partition 3D volume while minimizing the *line cut*

- The line cut is the number of additional parts a line crosses
- Assigning the entire volume to a single GPU is still a partitioning. Good for communication, but defeats the purpose.

- The load of a *voxel* is the *number of rays crossing it*. The load of a part is the sum over the loads of its voxels.
- A good partitioning ensures that each part has a similar load.
Problem (Tomographic partitioning)
Let $V$ be a cuboid, and $G$ a set of rays through $V$. Find a $p$-way partitioning of $V$, that minimizes the total line cut, while ensuring that the parts have a roughly equal load.

- **Recursive bisectioning strategy**: recursively split $V$ in two, somewhere along one of the three axes.
- It is possible to find the best partitioning of this kind in $O(p|G| \log |G|)$ time (GRCB algorithm).
- Communication reduced by between 60% and 90%
- Each GPU guaranteed to perform the same amount of work

Projection-based partitioning
• *Reducing the input size*: look at projections instead of rays.
- Communication volume is proportional to area of the shadow overlaps of parts.
Subroutine: communicationVolume
Input: $V_L$, $V_R$, projection set $\Pi$
Output: communication volume $\Theta$

$\Theta \leftarrow 0$

for all $\pi \in \Pi$ do

shadow$_L \leftarrow$ convexHull$\left(\text{project}(\pi, \text{corners}(V_L))\right)$

shadow$_R \leftarrow$ convexHull$\left(\text{project}(\pi, \text{corners}(V_R))\right)$

$\Theta \leftarrow \Theta + \text{area}(\text{shadow}_L \cap \text{shadow}_R)$

if consider gradient then

$\Theta \leftarrow \Theta + M \times \text{area}(V_L \cap V_R)$
Continuous load balance

- If we have a candidate partitioning, we can efficiently estimate the communication volume using the part shadows.
- Generating candidate partitionings involve finding a projection-based estimate for the load. (Number of rays crossing voxels).
- Estimate by integrating over ray densities for each source point. Find $c$ such that:

$$
\int_{x_1}^{x_2} \int_{y_1}^{y_2} \int_{z_1}^{z_2} \sum_{k=1}^{\text{|P|}} \frac{1}{||\vec{x} - \vec{s}_k||^2} \ dz \ dy \ dx = \int_{c}^{x_2} \int_{y_1}^{y_2} \int_{z_1}^{z_2} \sum_{k=1}^{\text{|P|}} \frac{1}{||\vec{x} - \vec{s}_k||^2} \ dz \ dy \ dx.
$$
Equal load

- We can reduce the integral to 2D, and then solve numerically:

\[
\int_{x_1}^{x_2} \int_{y_1}^{y_2} \prod_{k=1}^{n} \left( \frac{1}{a_k(x, y)} \left( \arctan \left( \frac{z_2 - s_{k,z}}{a_k(x, y)} \right) \right) \right.
\]
\[
\left. - \arctan \left( \frac{z_1 - s_{k,z}}{a_k(x, y)} \right) \right) \right) dy \ dx, \tag{1}
\]

where

\[
a_k(x, y) = \sqrt{(x - s_{k,x})^2 + (y - s_{k,y})^2}.\]

- For certain acquisition geometries, need to consider the cone instead of the entire cuboid. We reject samples outside cone.

- Most successful strategy we found so far is an adaption of a standard streaming median find algorithm.
Partitioning results

- **Partitioning method**: Use continuous load balance to find candidate splits in each direction, use shadow characterization of the communication volume to choose the best split. Recurse on the subvolumes.
Overlap structures: finding (possibly non-simple, non-convex) polygons for each set of contributors.
Overlap algorithm

Subroutine: FINDFACES
Input: $\pi = \{V_s\}, \pi_k$
Output: OVERLAY

OVERLAY ← EMPTYARRANGEMENT
for $0 \leq s < p$ do
    SHADOW$_s$ ← convexHull(project($\pi_k, \text{vertices}(V_s)$))
    ARRANGEMENT$_s$ ← FROMFACETAG(SHADOW$_s$, [s])
    MERGE(overlay, ARRANGEMENT$_s$, CONCATENATE)

- Subdivision merging algorithms: "find area on map with forests, low precipitation, high temperature".
- We rasterize the resulting faces, and perform aggregate reads from GPU textures containing image data for communication between nodes.
Reconstruction times

- CCBₙ (Pleiades)
- CCBₙ (ASTRA-MPI)
- HCB (Pleiades)
- CCBᵣ (Pleiades)
- CCBᵣ (ASTRA-MPI)
• TODO Faster, equal results