Highly Parallel Image Reconstruction for Positron Emission Tomography (PET)

Jingyu Cui[1] Guillem Pratx[2] Sven Prevrhal[5] Lingxiong Shao[5] Craig S. Levin[1,3,4]

[1]Department of Electrical Engineering, [2]Radiation Oncology, [3]Departments of Radiology, [4]MIPS, Stanford University, CA, [5]Philips Healthcare, CA.

Introduction

Positron Emission Tomography (PET) is a nuclear medicine imaging technique which produces a 3D image of **functional** processes in the body. Unlike X-ray CT, which mainly reveals **anatomical** detail, PET imaging can detect biological activities at the molecular level, for instance heightened sugar metabolism in cancerous tissue. PET is used extensively to measure blood flow or glucose metabolic activity in brain and heart of human and animals.

Typical steps in PET imaging include:

- Patient is injected with a tracer substance tagged by positron emitting radioactive atoms;
- Positron annihilates with electron, generating two photons leaving the event location in random but opposite directions;
- A ring of detectors detect photon pairs in coincidence;
- A few hundred millions of line pairs pin down the volumetric distribution of the tracer.

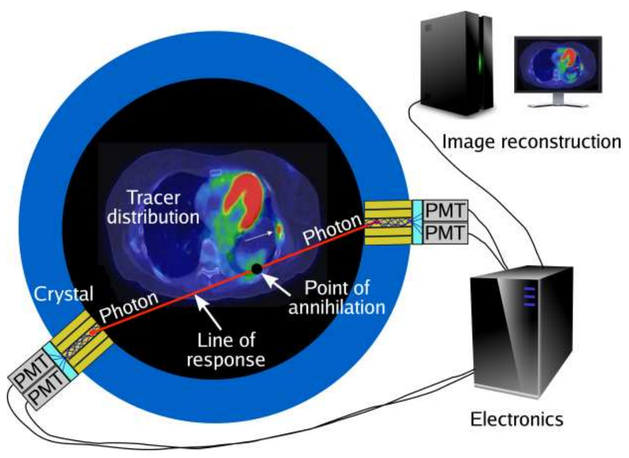

Fig. 1: Typical PET System.

Image Reconstruction

The list-mode Ordered Subset Expectation Maximization (OSEM) iterative algorithm involves a series of forward and back- projection operations for each voxel-line pair, and computes the tomographic image that has maximal likelihood given the measurements.

$$x^{(k+1)} = x^{(k)} \frac{1}{A^T \mathbf{1}} A^T \frac{y}{Ax^{(k)}}$$

2. Compare

4. Multiplicative Update 3. Backprojection 1. Forward Projection

Workload Distribution

We implemented the traversal of the projective line through image space on a slice-by-slice basis. Line-slice interactions are handled by parallel threads. To avoid data race, we try to partition the computation according to the output, so that the threads are **independent** in term of write collision, yet they can **collaborate** in reading data since they share the input.
This is easy for forward projection, where only one thread writes to a line. However, it's hard for backprojeciton since line-slice interactions are sparse.

Typical problem size:

- 100 million lines
- 10 million voxels
- Iterate for 20 times

Design considerations:

- Keep all cores busy;
- Ensure thread coherency;
- Hide memory latency by using concurrent threads;
- Make use of locality to minimize bandwidth usage.

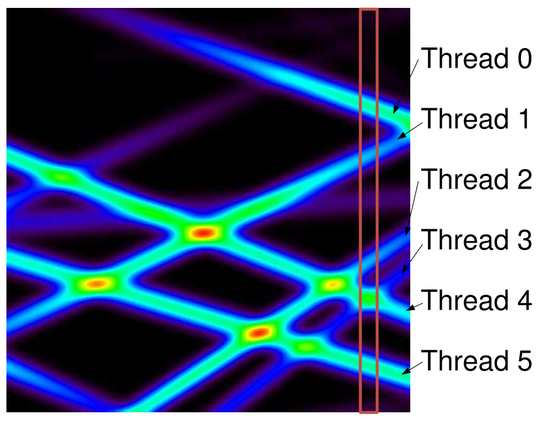

Fig. 2: Demonstration of the workload distribution.

Implementation and Results

Forward projection: Lines are partitioned into groups, each assigned to a thread block. Each thread in the thread block processes a line independently.
Backprojection: Sparsity is not exploited if job is partitioned by voxels. Instead, the same partition scheme as the forward projection was used with atomic operations to ensure correctness.

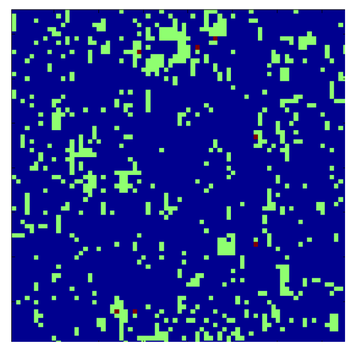

Fig. 3: Retries in spin-lock atomics. Blue: 0 retry; Green: 1 retry; Red: 2 retries.

Optimization techniques used:

- Loop unrolling. Trade off between unrolling and register usage.
- Fast math.
- Linear texture interpolation.

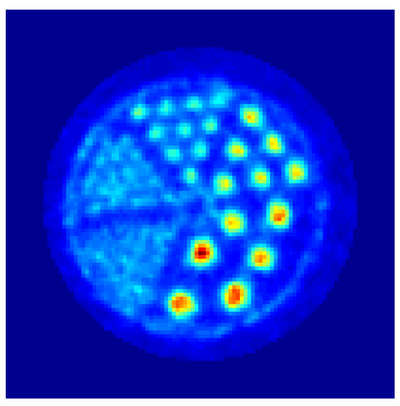

Fig. 4: A hot rod phantom reconstructed on the GPU with 5 subsets and 20 iterations. Rods sizes are 1.2, 1.6, 2.4, 3.2, 4.0, and 4.8 mm.

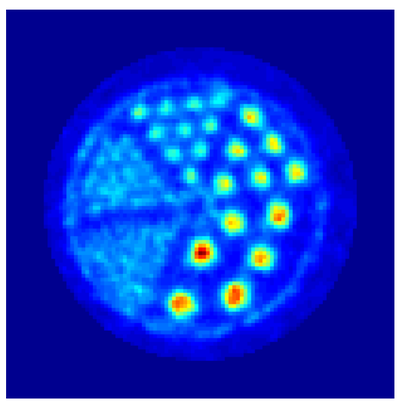

Fig. 5: The same phantom in Fig. 4 reconstructed on the CPU with the same parameters.

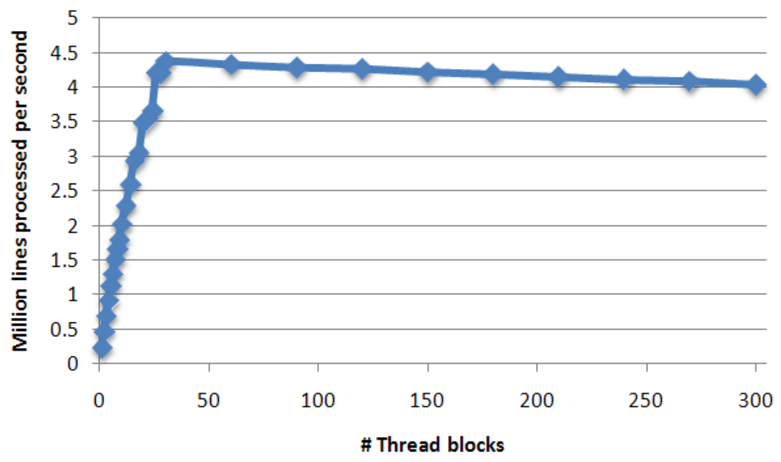

Fig. 6: Number of lines processed per second v.s. varying thread blocks.

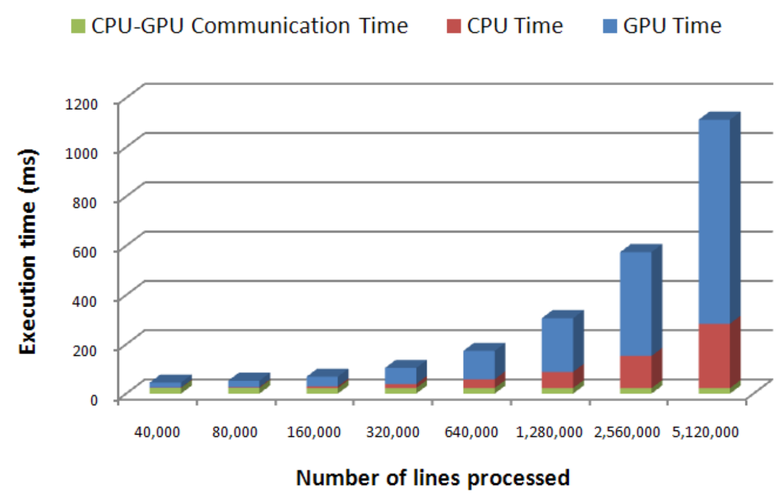

Fig. 7: Execution time for processing varying numbers of lines.

120x speedup compared with optimized CPU implementation.

Discussion

Future Optimizations:

- Better culling to skip non-intersecting slices and lines;
- More accurate physics modeling.

This work was supported in part by a grant from Philips Healthcare and NIH grants R01 CA110956, ARRA R01CA119056-04S1, and R01 CA120474.