Improvements to Hardware-Accelerated 3D Single Particle Imaging Data Reconstruction

Niteya Shah, Christine Sweeney, and Vinay Ramakrishnaiah

I. Introduction

Fast analysis of scientific data from X-ray free electron laser (XFEL) experimental facilities is key for supporting real-time decisions that efficiently use these facilities to speed up scientific discovery. Our research shows gains obtained using graphics processing units (GPUs) to accelerate 3D reconstruction of Single Particle Imaging (SPI) X-ray diffraction data. We achieve a 4X speedup over the previous GPU implementation, 50% better image reconstruction resolution, and 485X speedup when calculating resolution compared to the existing implementation. We showcase techniques to optimize per-node computational efficiency, increase scalability and improve the accuracy of SPI by using better algorithms, improving data movement and accesses, reusing data structures and reducing memory fragmentation.

II. Background

SPI is a technique for determining the structure of biomacromolecules using diffraction patterns gathered from directing an X-ray beam at single particles. SPI via free electron lasers (FELs) is fundamental for understanding the time evolution of enzymatic reactions. SPI starts with a randomly generated electron density estimation and iteratively builds a closer estimation of the protein structure.

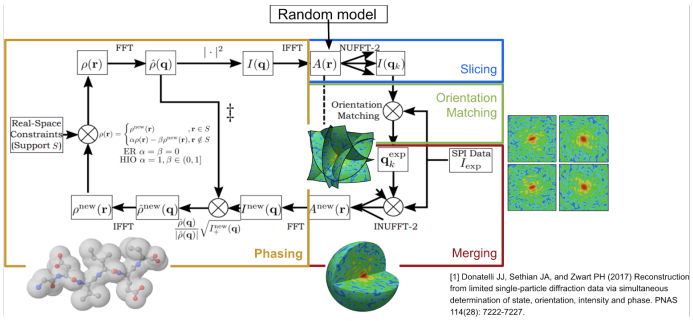


Fig. 1: SPI Workflow

SPI can be split into the following components, shown diagrammatically with Figure 1:

A. Slicing

The slicing step creates M 2D model images from the electron density estimation by applying non-uniform FFTs. SpiniFEL uses the cufinufft [4] library to efficiently calculate the forward (uniform to non-uniform) FFT.

B. Orientation Matching

The orientation matching step finds the closest reference image for each model image generated from slicing. Pairwise Euclidean distance is computed between the reference images and the model images and then the nearest matching reference image is found for each model image.

C. Merging

The merging step combines the reference orientations into a 3D diffraction volume. This is done by solving the linear system using the conjugate gradient (CG) algorithm.

D. Phasing

The phasing step converts the 3D diffraction volume into a molecular structure by applying real-space and reciprocal space constraints on the electron density estimate of the sample.

E. Resolution Calculation

When testing the algorithm, the achieved convergence is calculated by the comparing the normalised cross-correlation coefficient between the generated model and the base model using Fourier shell correlation (FSC).

III. Methodology

A. Improvements to Slicing and Orientation Matching

The slicing and orientation matching steps are combined to reduce data movement. The Einstein summation convention (einsum) operation for the rotation matrix is precomputed and stored in pinned memory. We then batch the slicing operation to reduce memory utilization, and for every batch, we slice it and then compute a partial orientation match using a GPU-optimized algorithm as follows:

The Euclidean distance of two vectors can be written as

$$(\vec{x} - \vec{y})^2 = ||\vec{x}||^2 + ||\vec{y}||^2 - 2 < \vec{x}, \vec{y} > \quad (1)$$

When implemented across multiple images, this transforms into

$$||(X_{A,n} - Y_{B,n})||^2 = \sum_{i=1}^{n} X_{A,i}^2 \oplus \sum_{i=1}^{n} Y_{B,i}^2 + X_{A,n} \cdot Y_{B,n}^T \quad (2)$$

The outer product is found by broadcasting the squared norm of the images. The matrix product can be computed using the GEMM BLAS routine $C := C - 2 * X * Y^T$. By using the CUBLAS GEMM routine to calculate the matrix product,

we can significantly speedup the computation by using tensor cores available on modern GPUs.

We calculate the row-wise minimum on the GPU using *cub::DeviceSegmentedReduce::ArgMin* [3], which uses a tree style reduction.

B. Improvements to Merging

We improve the performance of merging by moving the CG calculation to the GPU. Additionally, redundant computations are identified and such data is precomputed before the iteration. The CG algorithm used in merging is sensitive to small perturbations, affecting convergence [2]. We remedy this by adding support for multi-precision to achieve a balance of performance and stability.

C. Improvements to resolution computation

We identify hot spots in the FSC calculation and start processing those components on the GPU. This nets us a 40X speedup when the computation is done outside of Spinifel and a 485X speedup when done within the SpiniFEL iteration, storing intermediates and pre-loading libraries.

IV. Results

We run a variety of scaling studies on the NERSC Perlmutter supercomputer GPU nodes, each of which have a 3rd Gen AMD EPYC CPU, 4 A100 Nvidia GPUs and use the Slingshot10 interconnect for inter-node communication.

A. Weak Scaling Study

We perform a weak scaling study using the 3IYF protein by fixing the number of images per rank, the number of reference orientations used, and vary the number of nodes and measure the final execution time.

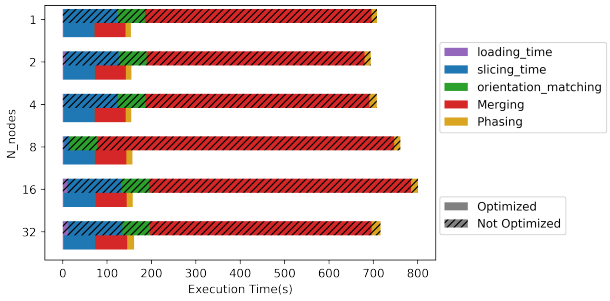


Fig. 2: Weak Scaling of Spinifel using the 3IYF dataset - 1500 images per rank, 4 ranks per node, 10K orientations

We see from Figure 2 that the execution time per node remains nearly constant during weak scaling. This shows that the communication overheads are small compared to the rest of the execution, and therefore, the algorithm scales well to a large number of nodes.

B. Varying Number of Orientations

We vary the number of orientations per node, keeping the number of nodes and images the same. Figure 3 shows the optimizations are not specific to a particular case and the performance gains remain consistent over the existing implementation as the work done per node increases.

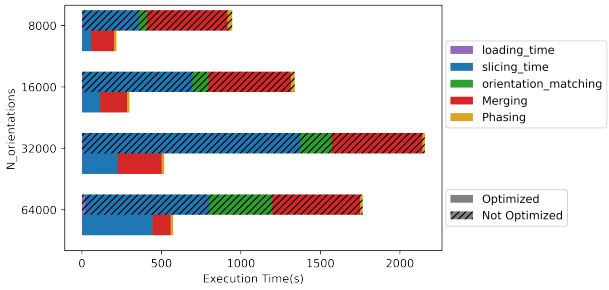
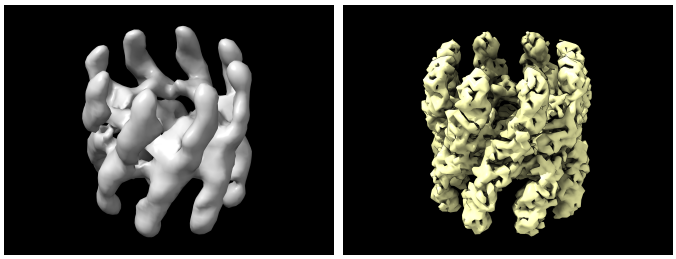


Fig. 3: Varying orientations in Spinifel - 3IYF dataset, 1500 images per rank, 4 ranks per node, 32 nodes

C. Visualization

Figure 4 shows the 3D structure of the 3IYF protein reconstructed using SpiniFEL, visualized in ChimeraX [1].



(a) Aligned structure (b) Reference structure

Fig. 4: Final result visualized using Chimera

V. Summary

We see significant gains in performance that scale up as the problem size grows. Additionally, the changes not only improve the time taken to measure convergence and but also the final resolution achieved. This work helps meet the Exascale Computing Project's goals for maximizing the use of GPUs on emerging exascale machines.

References

[1] Thomas D Goddard, Conrad C Huang, Elaine C Meng, Eric F Pettersen, Gregory S Couch, John H Morris, and Thomas E Ferrin. Ucsf chimerax: Meeting modern challenges in visualization and analysis. *Protein Science*, 27(1):14–25, 2018.

[2] William W Hager and Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[3] Péter Vingelmann NVIDIA and Frank HP Fitzek. Cuda, release: 10.2. 89, 2020.

[4] Yu-hsuan Shih, Garrett Wright, Joakim Andén, Johannes Blaschke, and Alex H Barnett. cufinufft: a load-balanced gpu library for general-purpose nonuniform ffts. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 688–697. IEEE, 2021.