GPU Clusters for Large-Scale Analysis of X-ray Scattering Data

Abhinav Sarje • Jack Pien • Xiaoye Li

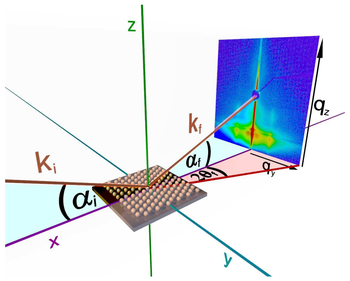
Lawrence Berkeley National Laboratory • Berkeley • CA

X-RAY SCATTERING is a valuable tool for measuring the structural properties of materials used in the design and fabrication of energy-relevant nanodevices that are key to the reduction of carbon emissions. Although today's X-ray scattering detectors can provide raw information on structural properties of nanoparticles, a primary challenge remains in the analysis of this data due to its size. We are developing novel high-performance computing algorithms and codes for such analyses. Here we present two such HPC advances:

1. **a flexible Grazing Incidence Small Angle Scattering (GISAXS) simulation code** based on the **Distorted Wave Born Approximation (DWBA)** theory. This code can compute the scattered light intensity from any given sample in all directions of space.
2. **an efficient inverse modeling code for structural fitting problems** using **Reverse Monte Carlo (RMC) simulation** algorithm.

Distorted Wave Born Approximation

- Studying complex nanostructures requires solving for **form factors** in a high-resolution Q-grid resulting in matrices with $\sim 10^7$ to 10^{10} q-points.
- This time-consuming and memory-demanding step is a major bottleneck in GISAXS simulations.



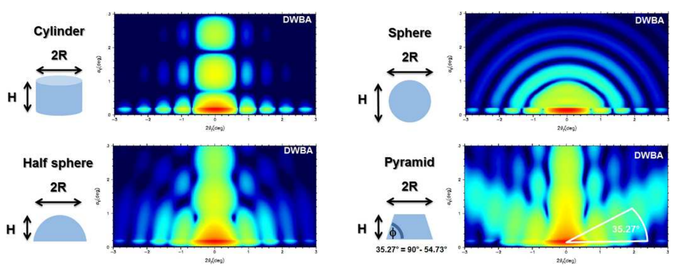
DWBA is the only theoretical framework which models the GISAXS process. It defines scattering intensity of the X-rays as:

$$I(\vec{\mathbf{q}}) \quad \propto \quad |F(\vec{\mathbf{q}})|^2$$

$F(\vec{\mathbf{q}})$ is the form factor of a q-point $\vec{\mathbf{q}}$, computed as an integral over the shape function $S(\vec{r})$ of the nanoparticles. For computational purposes, the shape surface is triangulated and the form factor is approximated as a summation over resulting N triangles, with surface area of triangle $t = s_t$:

$$F(\vec{\mathbf{q}}) \quad = \quad \int_{S(\vec{r})} e^{i\vec{\mathbf{q}}\cdot\vec{r}} d\vec{r} \quad \approx \quad \sum_{t=1}^{N} e^{i\vec{\mathbf{q}}\cdot\vec{r}} s_t$$

Below are analytically computed example form factor intensity images for simple shapes.

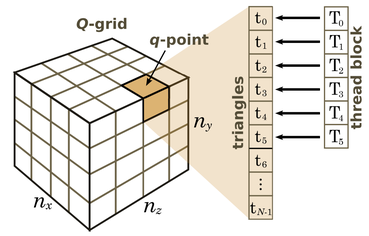


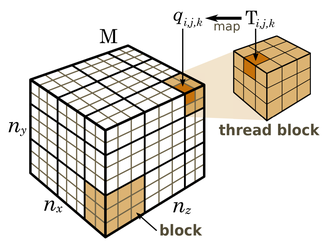
Computing Form Factors on GPUs

- **Given:** A 3-D Q-grid of resolution $n_x \times n_y \times n_z$, triangulated shape surface (set of N triangles.)
- **Task:** Generate 3-D matrix M of size $n_x \times n_y \times n_z$ where each $M(i,j,k) = F(q_i, q_j, q_k) = F(\vec{\mathbf{q}}_{i,j,k})$

Phase 1

- Compute inner term, $F_{t_l}(\vec{\mathbf{q}}) = e^{i\vec{q}\cdot\vec{r}} s_{t_l}$ for each triangle t_l.
- Store in a 4-D matrix M_I of size $n_x \times n_y \times n_z \times N$.
- Decompose computations along triangles: each thread mapped to a unique triangle.



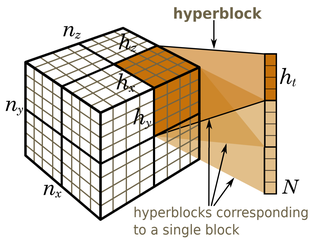


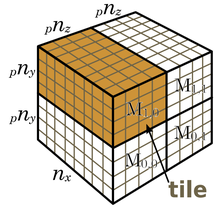
Phase 2

- Decompose M into a grid of 3-D **blocks**, M_b, each corresponding to a CUDA thread block.
- For each q-point $\vec{\mathbf{q}}_{i,j,k}$, a thread $T_{i,j,k}$ computes $M_b(i,j,k) = F(\vec{\mathbf{q}}) = \sum_{l=1}^{N} F_{t_l}(\vec{\mathbf{q}}_{i,j,k})$, generating M_b.

Handling Memory Limitations

- Decompose M_I along all dimensions into 4-D **hyperblocks**, M_h.
- Each hyperblock results in a partial sum in M. All partial sums for a block are reduced later.



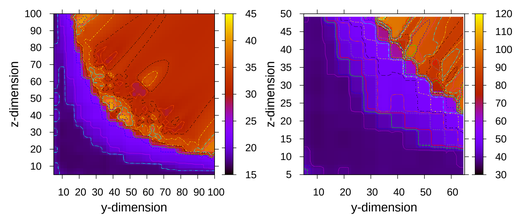


Utilizing Cluster of GPUs

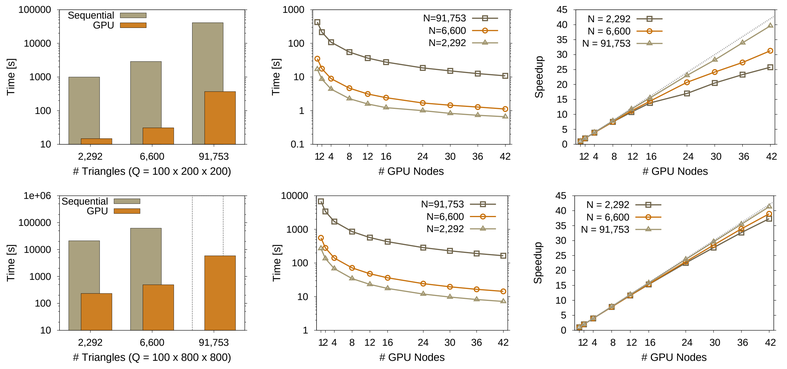
- Decompose M along y and z into **tiles**, as a higher level of parallelism.
- Use MPI across GPU nodes.
- Tile $M_{i,j}$ is assigned to processor $P_{i,j}$.

Choosing Hyperblock Size

- Crucial for performance: demonstrated by heat maps of runtimes (seconds) with varying sizes.
- Q-grid size is $100 \times 200 \times 200$, and $N_1 = 2,292$ and $N_2 = 91,753$, respectively.

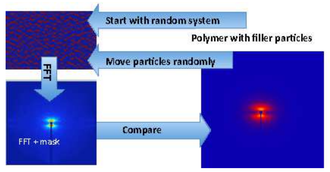


Performance Results



Reverse Monte Carlo Modeling on GPU

- Information on material structure from Small Angle X-ray Scattering (SAXS) data, consisting of 2-D images, is extracted using RMC.
- Involves $\sim 10^6$ iterations, each with $\sim 10^4$ Fourier transformations.

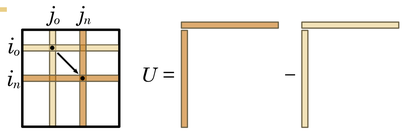


- **Given:** SAXS image as $N \times N$ array I, and random particle array A.
- **Task:** Compute structure factor F_n^2 fitting input image.

Initialization: Compute $F_0 = \text{FFT}(A)$, $F_0^2 = \texttt{cws}(F)$.

Simulation Steps

1. Randomly move a particle from (i_o, j_o) to (i_n, j_n).
2. Compute update to the Fourier transform, new structure factor, and χ^2 error (D = DFT matrix, cws = component-wise square):



$$U = \texttt{dft2}(i_o, j_o, i_n, j_n) = D(j, i_n) \cdot D(j_n, i) - D(j, i_o) \cdot D(j_o, i),$$
$$F_n = F_{n-1} + U, \qquad F_n^2 = \texttt{cws}(F_n), \qquad \chi_n^2 = \sum_{i,j} (I(i,j) - F_n^2(i,j))$$

- Decompose array computations into 1-D grid of thread blocks, each thread computing one element in the $N \times N$ matrices.
- Observed about 900x speedup over a sequential Matlab code.

Acknowledgements

This research has used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The GISAXS geometry diagram and scattering intensity images are from A. Meyer's *gisaxs.de*. The GISAXS & SAXS data are provided by Slim Chourou and Alexander Hexemer.

References

1. G. Renaud, R. Lazzari, and F. Leroy. Probing surface and interface morphology with grazing incidence small angle x-ray scattering. *Surface Science Reports*, 64:255–380, 2009.
2. R. McGreevy and L. Pusztai. Reverse monte carlo simulation: A new technique for the determination of disordered structures. *Molecular Simulation*, 1:6:359–367, 1998.
3. A. Scholz and A. Hexemer. Reverse Monte Carlo in Matlab/Jacket. *Private communication*, 2011.