CATEGORY: **SUPERCOMPUTING - SC04**

POSTER **P4255**

CONTACT NAME **Abhinav Sarje: asarje@lbl.gov**

GPU TECHNOLOGY CONFERENCE

Inverse Modeling of X-ray Scattering Data with Reverse Monte Carlo Simulations

Abhinav Sarje **Xiaoye S Li** **Alexander Hexemer**

Computational Research Division Advanced Light Source

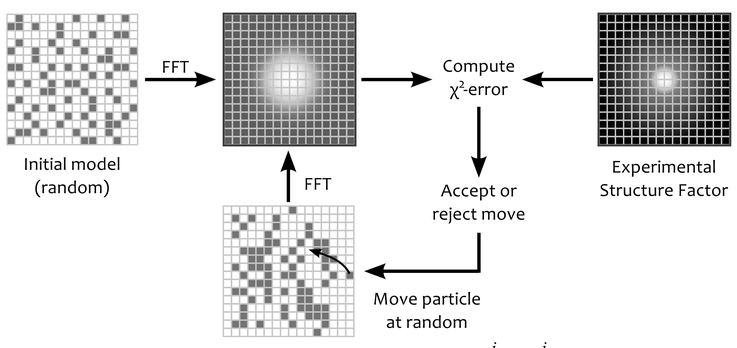
Lawrence Berkeley National Laboratory

abstract

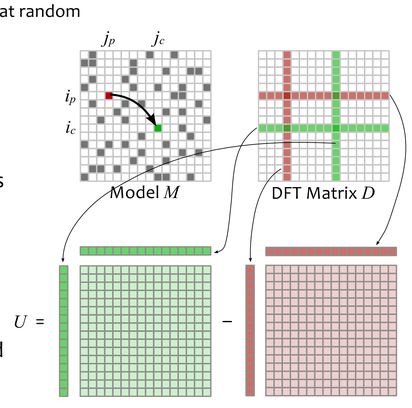
We are developing high-performance parallel algorithms and codes for analysis of X-ray scattering data for methods, such as the Small Angle X-ray Scattering (SAXS). X-ray scattering techniques are widely used for the characterization of macromolecules and nanoparticle systems based on their structural properties and are applicable to numerous applications, including design and fabrication of energy-relevant nanodevices such as photovoltaic cells and energy storage devices. Our high-performance codes address the primary bottleneck in such characterization processes: analysis of large amounts of raw data obtained through present ultra-fast light beamlines and scattering detectors. As an inverse modeling problem, structural fitting of the data obtained through SAXS experiments is used for extracting information on structural properties of materials. Such fitting processes involve a large number of variable parameters and, hence, require high computational power. In the following we present a scalable parallel solution based on the Reverse Monte Carlo simulation algorithm, and accelerated with graphics processors.

reverse monte carlo

Reverse Monte Carlo (RMC) modeling is a variation of the standard Monte Carlo procedures, and is a popular general method of structural modeling based on experimental data [1]. We start with a randomized initial configuration of *n* particles within a matrix *M* with periodic boundary conditions. RMC is an iterative procedure where in each iteration a new model configuration is generated by moving one particle in *M* randomly. The RMC process is depicted below:



Between the two successive steps, since only one particle is moved, the change to the structure factor is rather small, making it more efficient to compute this change instead of a full Fourier transformation. This amounts to adding the new contribution due to particle at $M[i_c, j_c]$ and subtracting the contribution due to its previous position at $M[i_p, j_p]$. See adjacent figure for an illustration. These can be directly computed from two rank-1 matrices derived from the outer products of the two involved columns and rows of the DFT matrix. This update U is then simply added to the structure factor.

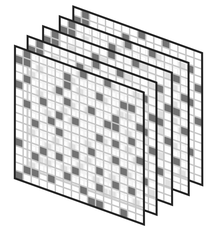


Computing the update matrix from outer products of participating rows and columns, to be added to the structure factor.

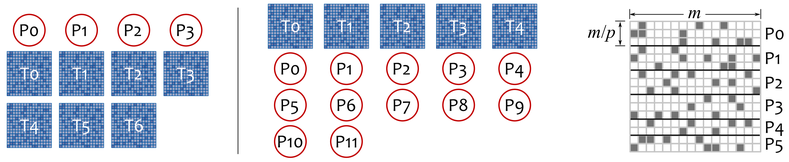
parallelization

An RMC simulation run comprises simulation over multiple ***tiles***, each of size *s* x *s*. A typical run would have about $O(10^2)$ to $O(10^3)$ tiles. A tile contains its own model configuration matrix and simulation over each tile is completely independent of other tiles, and hence exposes parallelism. We make this as the highest parallel decomposition level. Let there be a total of *t* tiles and *p* processors/nodes. Two scenarios arise:

1. p < t: We distribute tiles to processors in a round-robin fashion as visualized below, and each processor then performs simulation on the tiles assigned to it independent of other processors.
2. p ≥ t: We first do a round robin distribution of processors to tiles as depicted in figure below, resulting in the assignment of more than one processor to each tile. The second level in the our parallel hierarchy scheme is distributed memory parallelization of each tile across multiple processes. We perform a one-dimensional decomposition of the tile along rows among the assigned processors. See illustration below.



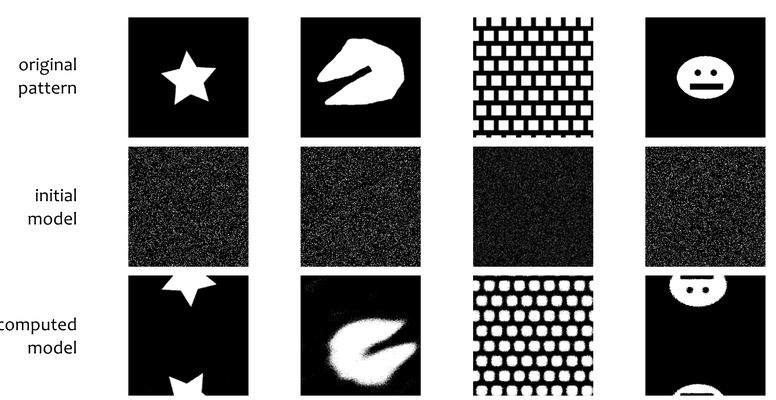
Tiles in a simulation. Each tile represents a particle configuration and is independent of all other tiles.



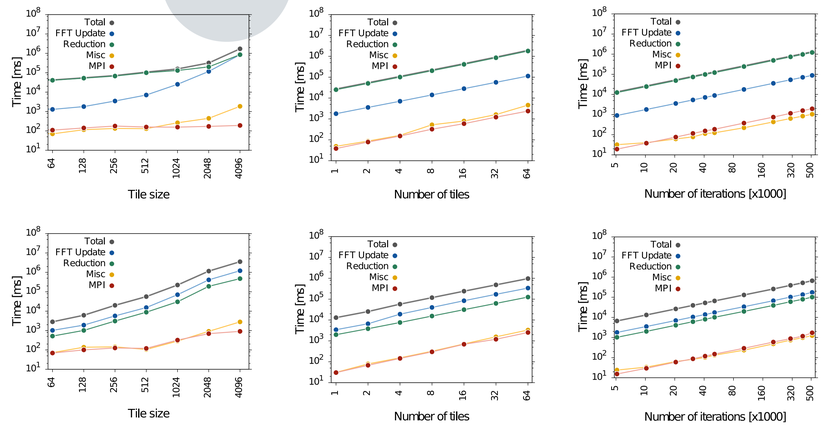
(Left) Round-robin distribution of *t* tiles among *p* processors in the simple case when $p < t$. *(Center)* Round-robin distribution of *p* processors among *t* tiles when $p > t$. In this case multiple processors perform computations on a single tile. *(Right)* Row-wise decomposition of a tile among its assigned processors.

simulations on graphics processors

We further parallelize the simulation on graphics processors by off-loading the compute intensive kernels in the algorithm. In the following we show results of several inverse modeling examples with our codes. In most cases the code is able to reconstruct the original pattern with high degree of similarity.



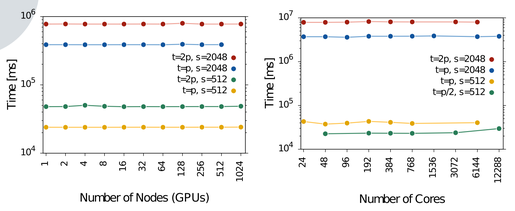
single-node results



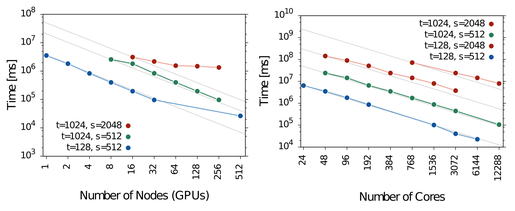
Code behavior with varying input parameters and sizes. The plots are shown for GPU (*top row*) and for comparison multicore CPU (*bottom row*). In the case of multicore CPU, we have a total of 32 OpenMP threads.
(*Left*) Execution time with varying input tile sizes, number of tiles = 1 and number of iterations = 40k.
(*Middle*) Execution time with varying number of tiles, tile size = 512 x 512 and number of iterations = 10k.
(*Right*) Execution time with varying number of iterations, tile size = 512 x 512 and number of tiles = 1.

multi-node results

Performance on Cray XK7 (Titan) and Cray XE6 (Hopper) are shown on the right. Note that the GPU execution is more than a degree of magnitude faster than the CPU execution for same number of nodes. The obtained weak scaling is almost perfect as shown. We also obtain super-linear speedups for certain cases. On average we obtain speedups of 222x and 230x with 256 nodes on the GPU and CPU cluster, respectively.



Weak scaling on GPUs (*left*) and CPUs (*right*) with respect to the number of tiles t and number of compute nodes. There are 24 cores per node on the CPU cluster.



Strong scaling on GPUs (left) and CPUs (right) with respect to the number of nodes on the two clusters. There are 24 cores per node on the CPU cluster.

acknowledgements

We used resources of the National Energy Research Scientific Computing Center, which along with this work is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. We also used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No.DE-AC05-00OR22725. This work was supported by the Director, Office of Science, of theU.S. Department of Energy under Contract No. DE-AC02-05CH11231. In addition, A. Hexemer was supported by a DOE Early Career Research Program grant.

references

1. R. McGreevy. Reverse Monte Carlo modeling, *J. Phys.: Condens. Matter*, vol. 13 R877–R913, 2001.
2. A. K. Scholz. Investigations of deformation and fatigue properties of filled elastomers using small angle x-ray scattering, *Ph.D. dissertation, University of California at Santa Barbara*, Dec. 2012.