Tuning *HipGISAXS* on Multi and Many Core Supercomputers

Abhinav Sarje[1]

Xiaoye Sherry Li | Alexander Hexemer

[1]Computational Research | Advanced Light Source

Lawrence Berkeley National Laboratory

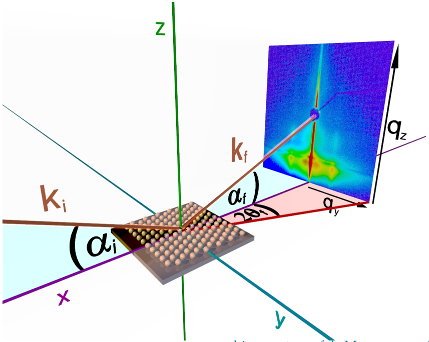
11.18.13

Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems

Supercomputing 2013, Denver

Background: What is *HipGISAXS*?

- Massively parallel **Hi**gh-**p**erformance **GISAXS** simulation code.
- **GISAXS:** Grazing Incidence Small-Angle X-ray Scattering.
- **X-ray scattering:** a tool to measure structural properties of materials; characterize macromolecules and nano-particle systems at micro and nano-scales.
- Expose sample to high-energy X-rays, producing scattering patterns.
- Scattering patterns contain structural information about the sample.



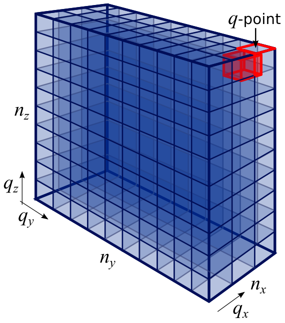
graphic: courtesy of A. Meyer, *www.gisaxs.de*

Outline

1. The main computational kernel.
2. Motivations and challenges.
3. Contributions.
4. Platforms' overview.
5. Code optimization and tuning on clusters of:
    - Graphics processors (GPU);
    - Intel Phi coprocessors (MIC);
    - Intel Sandy Bridge and AMD Magny Cours CPUs.
6. Performance analysis and comparisons.
7. Conclusions.

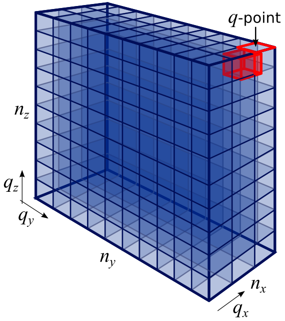
HipGISAXS Kernel: Numerical Form Factor Calculations

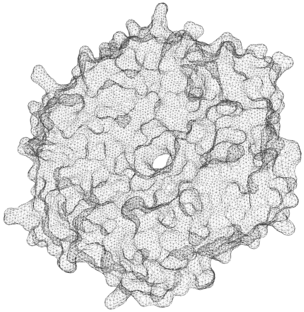
- $\mathcal{Q}$ is grid of size $n_x \times n_y \times n_z$
- Described by 3 vectors,
  $q_\alpha = \langle p_0 \cdots p_{n_\alpha - 1} \rangle, \alpha \in \{x, y, z\}$.



HipGISAXS Kernel: Numerical Form Factor Calculations

- $\mathcal{Q}$ is grid of size $n_x \times n_y \times n_z$
- Described by 3 vectors, $q_\alpha = \langle p_0 \cdots p_{n_\alpha - 1} \rangle, \alpha \in \{x, y, z\}$.
- $\mathcal{T}$ is set of n_t triangles describing a nano-structure model.
- $\mathcal{T} = \{t_0 \cdots t_{n_t - 1}\}, t_k = \langle \vec{r}_{t_k}, \vec{N}_{t_k}, s_{t_k} \rangle$





HipGISAXS Kernel: Numerical Form Factor Calculations

- Form factor computed over model surface at each q-point:

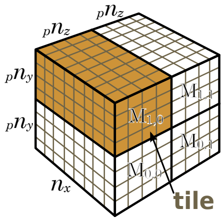
$$\mathcal{F} : f(\vec{q}) = -\frac{i}{|\vec{q}|^2} \sum_{t \in \mathcal{T}} e^{i\vec{q}\cdot\vec{r}_t} q_{N_t} s_t, \quad \forall \vec{q} \in \mathcal{Q}.$$

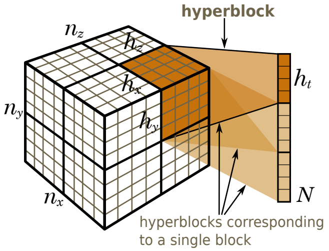
- Construct matrix $\mathcal{F}$ of size $n_x \times n_y \times n_z$.
- Hence, computational complexity = $O(n_x n_y n_z n_t)$.
- Kernel is *compute-bound*.

Generic Parallelization of the Form Factor Kernel

Covered in previous work [1]:

- "Embarrassingly parallel".
- *Tiles* across multiple nodes to decompose computations.
- *Hyperblocks* on a node to improve memory-based performance.
- Hyperblocks also helpful with accelerators.





[1] *"Massively Parallel X-ray Scattering Simulations"*, Supercomputing, 2012.

Motivations and Challenges

- This is a both *big data* and *big compute* problem.
- Data generation rate is continually increasing at light sources.
- Need for near real-time analysis.
- We utilize massive-parallelism to solve this.
- “Architecture-aware” algorithms and techniques necessary to extract raw computational power.
- Efficient mapping onto the processors’ multi-level memory and parallelism hierarchies.
- Make effective use of state-of-the-art compute facilities, reducing cost (time and power).

Motivations and Challenges

- This is a both *big data* and *big compute* problem.
- Data generation rate is continually increasing at light sources.
- Need for near real-time analysis.
- We utilize massive-parallelism to solve this.

- “Architecture-aware” algorithms and techniques necessary to extract raw computational power.
- Efficient mapping onto the processors’ multi-level memory and parallelism hierarchies.
- Make effective use of state-of-the-art compute facilities, reducing cost (time and power).

Contributions

- Optimizations on Nvidia Fermi and Kepler graphics processors.
- Parallelization and optimization on Intel MIC architecture.
- Optimization on the AMD Magny Cours and Intel Sandy Bridge processors.
- Implementation of auto-tuning of parallelization and optimization parameters.
- Detailed performance analysis and comparisons.

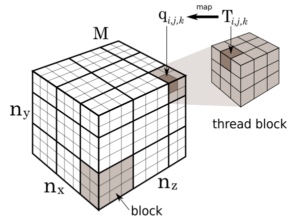
Experimental Platforms

System	**Titan** (Cray XK7)	**Stampede**	**Hopper** (Cray XE6)	**Edison-I** (Cray XC30)
Architecture	GPU	MIC	Multicore	Multicore
Processor	Nvidia K20X	Intel Phi SE10P	AMD Magny Cours	Intel Sandy Bridge
Cores	14×192 CUDA	$60 + 1$	2×12	2×8
HW Threads/Core	1	4	1	2
Theoretical peak	3,950 GFLOP/s	2,021 GFLOP/s	403 GFLOP/s	664 GFLOP/s

Performance Optimizations Highlights in HipGISAXS

- Guided by performance profiling:
  Nvidia visual profiler, Intel VTune, PAPI.

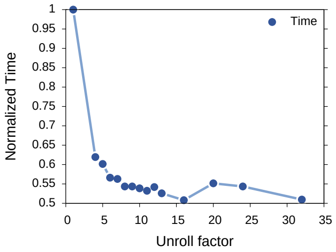
- Expose more parallelism.
  - *Loop interchange* and *loop collapsing*.
  - New optimal CUDA thread-block decomposition scheme on GPUs.



- Avoid memory transfers, read/writes.
  - *Kernel fusion*. E.g. 87% improvement on GPUs.
  - Use *persistent buffers* on MIC.
  - *Data reuse*. E.g. input data reuse through shared memory on GPUs.

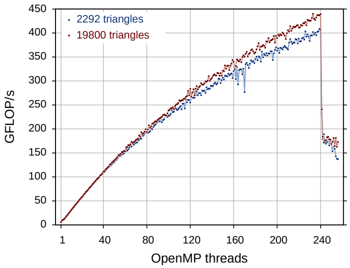
Performance Optimizations Highlights in HipGISAXS

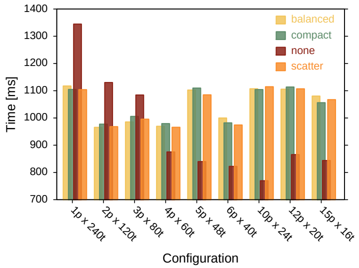
- Overlap computations with memory transfers.
  - *k*-buffering. E.g. triple-buffering > 5% better than double-buffering.
- Optimize the unavoidable memory transfers.
  - Input triangles data (7 reals) *padding* (1 real) for better alignment (32B).
  - Avoid expensive operations, like `vgatherdps` on MIC, by input *data reorganization*. E.g. 33% improvement.
- Expose instruction level parallelism.
  - Shape triangles *loop unrolling*. 50% improvement on GPUs.



Performance Optimizations Highlights in HipGISAXS

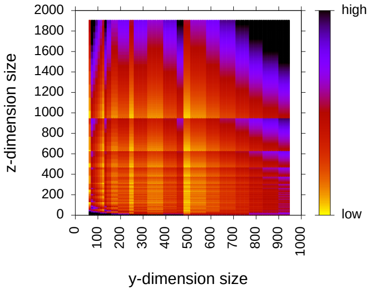
- Vectorize.
  - Reorganize data to facilitate better vectorization.
  - Hand vectorize and optimize complex number operations. E.g. 22% improvement on MIC over auto-vectorization.
  - E.g. use SSE2 on Magny Cours, AVX on Sandy Bridge.
  - Optimize *exp* and *sincos*. E.g. 25% improvement over SVML on MIC.
- Optimize execution environment configuration.
  - E.g. Thread configuration and thread affinities on MIC.

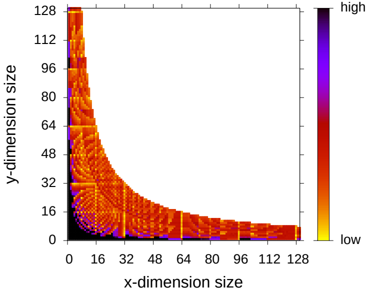




Performance Optimizations Highlights in HipGISAXS

- Optimize parameter values. E.g. On GPUs,
- auto-tuning hyperblock size.
- auto-tuning CUDA thread block size.





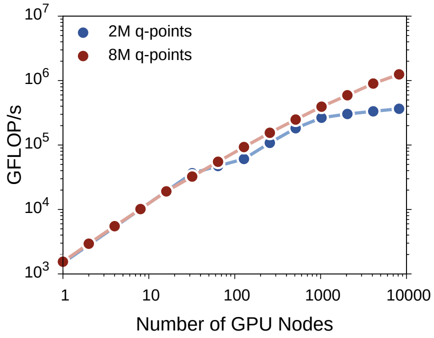
GPU Performance: Single Node

n_t	$\|\mathcal{Q}\|$	M2090	GFLOP/s	K20X	GFLOP/s
6,600	2M	0.68 s	813.6	0.25 s	2172.2
6,600	8M	2.73 s	813.2	1.02 s	2167.2
91,753	2M	9.47 s	813.4	3.56 s	2159.1
91,753	8M	37.9 s	813.4	14.25 s	2161.5

- Achieve **813 GFLOP/s** on M2090, **61.2%** of the theoretical peak.
- Achieve **2,172 GFLOP/s** on K20X, **55%** of the theoretical peak.

GPU Performance: Scaling Across Multiple Nodes

Strong Scaling on Titan (7.5M input triangles)



- Achieve **1.25 PetaFLOP/s** on **8,192** nodes.

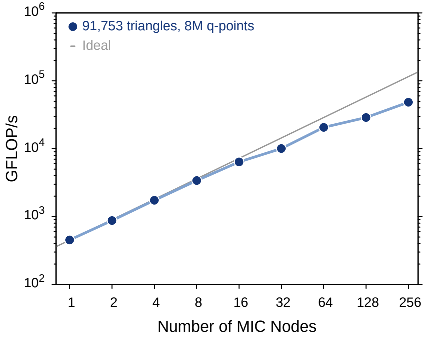
MIC Performance: Single Node

n_t	$\lvert\mathcal{Q}\rvert$	**Xeon Phi**	**GFLOP/s**
6,600	2M	2.27 s	453.58
6,600	8M	8.77 s	469.67
91,753	2M	30.77 s	465.22
91,753	8M	118.49 s	483.91
7,514,364	2M	2565.18 s	456.98

- Achieve **484 GFLOP/s**, about **24%** of the theoretical peak.

MIC Performance: Scaling Across Multiple Nodes

Strong Scaling on Stampede (91.7K input triangles)



- Achieve **0.1 PetaFLOP/s** on **1,024** nodes.

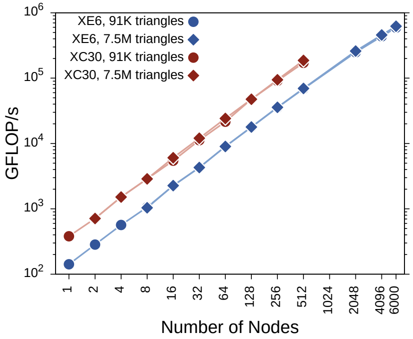
Performance on Multicores (XE6/XC30): Single Node

n_t	$\lvert\mathcal{Q}\rvert$	**XE6**	**GFLOP/s**	**XC30**	**GFLOP/s**
6,600	2M	6.34 s	141.7	2.97 s	378.2
6,600	8M	25.28 s	142.1	11.87 s	378.3
91,753	2M	88.16 s	141.7	41.0 s	379.0
91,753	8M	352.0 s	142.0	164.1 s	380.0

- Achieve **142 GFLOP/s** on XE6 node, **35.2%** of the theoretical peak.
- Achieve **380 GFLOP/s** on XC30 node, **57.2%** of the theoretical peak.

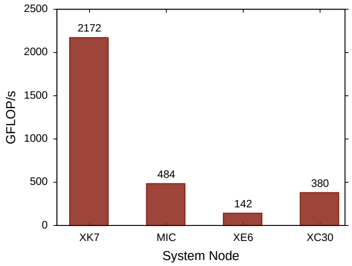
Performance on Multicores (XE6/XC30): Scaling

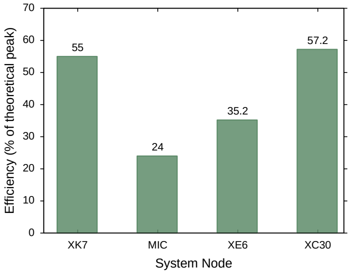
Strong Scaling on Hopper and Edison-I (91.7K and 7.5M input triangles)

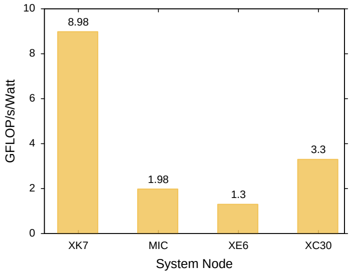


- Achieve **0.63 PetaFLOP/s** on **6,000** Hopper nodes (**144,000** cores.)
- Achieve **0.19 PetaFLOP/s** on **512** Edison-I nodes (**8,192** cores.)

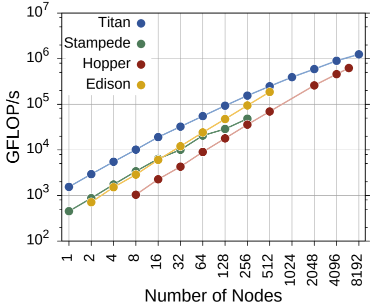
Performance Comparisons: HipGISAXS on Single Node

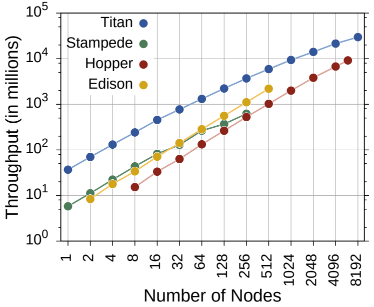






Performance Comparisons: HipGISAXS Strong Scaling





Conclusions and Future Work

- Developing a basic code with acceptable performance is easier on multicores like the Magny Cours and Sandy Bridge.
- **But**, extraction of high-performance from them is no easier than implementations on GPUs or MIC.
- Architecture-aware implementations are inevitable for high-performance on these platforms.
- Graphics processors in general perform higher, and have high power-efficiency.

- Explore more architectures such as IBM BG/Q.
- Multi-level distributed memory parallel framework for simulations.
- GISAXS simulation is just one component of the under-development HipGISAXS suite.

Conclusions and Future Work

- Developing a basic code with acceptable performance is easier on multicores like the Magny Cours and Sandy Bridge.
- **But**, extraction of high-performance from them is no easier than implementations on GPUs or MIC.
- Architecture-aware implementations are inevitable for high-performance on these platforms.
- Graphics processors in general perform higher, and have high power-efficiency.

- Explore more architectures such as IBM BG/Q.
- Multi-level distributed memory parallel framework for simulations.
- GISAXS simulation is just one component of the under-development HipGISAXS suite.

Acknowledgements

- *Samuel Williams* for the many helpful discussions.
- Supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and Alexander Hexemer's Early Career Award grant.
- Resource usage at *National Energy Research Scientific Computing Center* (NERSC) at the Lawrence Berkeley National Laboratory, supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02- 05CH11231.
- Resource usage at *Oak Ridge Leadership Computing Facility* (OLCF) at the Oak Ridge National Laboratory, supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- Resource usage at *Texas Advanced Computing Center* (TACC) at the University of Texas at Austin.

Thank you!

HipGISAXS Code and Documentation:

- *http://portal.nersc.gov/project/m1285/hipgisaxs.html*
- *http://github.com/hipgisaxs/hipgisaxs/wiki*

Additional Information:

- **Massively Parallel X-ray Scattering Simulations**, *Supercomputing (SC12), no. 46, pp. 46:1–46:11*, 2012.
- **HipGISAXS: A High Performance Computing Code for Simulating Grazing Incidence X-Ray Scattering Data**, *Journal of Applied Crystallography, vol. 46 (6), pp. 1781–1795*, 2013.