Performance optimization of WEST and Qbox on Intel Knights Landing

Huihuo Zheng\textsuperscript{1}, Christopher Knight\textsuperscript{1}, Giulia Galli\textsuperscript{1,2}, Marco Govoni\textsuperscript{1,2}, and Francois Gygi\textsuperscript{3}

\textsuperscript{1}Argonne National Laboratory
\textsuperscript{2}University of Chicago
\textsuperscript{3}University of California, Davis

September 27\textsuperscript{th}, 2017
ALCF Theta Early Science Project (Tier 1): First-Principles Simulations of Functional Materials for Energy Conversion

Embedded nanocrystal
T. Li, Phys. Rev. Lett. 107, 206805 (2011)

Matters at extreme conditions

Organic photovoltaics

Aqueous solution

Quantum information

Optimization focus

- Utilizing tuned math libraries (FFTW, MKL, ELPA, …)
- Vectorization: AVX512
- High Bandwidth Memory

Strong scaling limit

3,624 KNL nodes, 9.65 petaFLOPS

- Adding extra layers of parallelization -> increase intrinsic scaling limit
- Reducing communication overhead to reach the intrinsic limit
Outline

• WEST – additional layers of parallelization
  • Band parallelization of Sternheimer equation
  • Task group parallelization to fit 3D FFTs within single KNL node to reduce communication overheads and take advantage of HBM

• Qbox – reduce communication overheads of dense linear algebra with on-the-fly data redistribution
  • Gather & scatter remap
  • Transpose remap

• Conclusions and insights
Optoelectronic calculations using many-body perturbation theory (GW)

\[ \Delta \rho = \chi \Delta V_{\text{pert}} \]

Electronic density \rightarrow Response function \rightarrow Perturbation potential

\[ \Delta V_1 \rightarrow \Delta V_2 \rightarrow \Delta V_3 \rightarrow \Delta V_4 \rightarrow \cdots \rightarrow \Delta V_{N-1} \rightarrow \Delta V_N \]

\[ \Delta \rho_1 \rightarrow \Delta \rho_2 \rightarrow \Delta \rho_3 \rightarrow \Delta \rho_4 \rightarrow \cdots \rightarrow \Delta \rho_{N-1} \rightarrow \Delta \rho_N \]

Massively parallel by distributing perturbations

Parallelization scheme (image groups & plane wave)

Intrinsic strong scaling limit

\[ n_{\text{proc}} \sim N_{\text{pert}} \times N_z \]

$\text{Si}_{35} \text{H}_{36}$, 176 electrons, 256 perturbations

3D FFTs
Single perturbation runtime (4BG/Q vs 1KNL)

- 80% of runtime is spent in external libraries
- 3.7x speedup from BG/Q(ESSL) to KNL(MKL)
- High-bandwidth memory on Theta critical for performance (e.g. 3D FFTs): 3.1x speedup
Improvement of strong scaling by band parallelization

\[ n_{proc} = N_{pert} \times N_{band} \times N_z \]

Increased parallelism by arranging the MPI ranks in a 3D grid (perturbations & bands & FFT)
Improving performance of 3D FFTs using task group

Strong scaling of 3D FFT (plane and pencil decomposition) on Cetus (BG/Q) and Theta (KNL) using 256×256×256 FFT grid

Small 3D FFTs do not scale well across multiple KNL nodes because of internode communication overheads relative to shared-memory MPI. Task groups (tg) redistribute complete wave functions to separate nodes to simultaneously compute multiple 3D FFTs.
Outline

• WEST – additional layers of parallelization
  • Band parallelization of Sternheimer equation
  • Task group parallelization to fit 3D FFTs within single KNL node to reduce communication overheads and take advantage of HBM

• Qbox – reduce communication overheads of dense linear algebra with on-the-fly data redistribution
  • Gather & scatter remap
  • Transpose remap

• Conclusions and insights
Strong scaling of Qbox for hybrid-DFT calculations

Data layout: block distribution of wave functions to 2D process grid

\[ \psi_i(k), i = 1, 2, \ldots N_{\text{band}}, \]

\[ k = 0, 1, \ldots, n_{pw} - 1 \]

\[ \psi_i(k), i = 1, 2, \ldots N_{\text{band}}, \]

\[ k = 0, 1, \ldots, n_{pw} - 1 \]

SiC512: 140,288 × 1,024

140,288

nproc

nproc = N_{\text{band}}N_z

Good scaling for 3D FFTs up to intrinsic limit:

MPI_Alltoall(v)

MPI_Allreduce
Poor scalability of ScaLAPACK for tall-skinny matrices and small square matrices due to communication overheads.

Gram-Schmidt

Wave function matrix

Tall-skinny matrices

d(z)gemm
Reducing communication overheads from ScaLAPACK with “gather & scatter” remap

Solution: creating a context with fewer columns and on-the-fly data redistribution
- Compute 3D FFTs on original grid
- Gather data to smaller grid
- Run ScaLAPACK on smaller grid
- Scatter data back to original grid

The remap communication pattern only involves procs within same row or column.

Key: remap communication time needs to be small.
Improvement of strong scaling using “gather & scatter” remap

$hpsi + wf\_update$ time remains minimal relatively flat with remap, and the remap time (custom) is two orders of magnitude smaller than $hpsi + wf\_update$ time.

Custom remap function is 1000x faster than ScalAPACK’s pdgemr2d.

Improvement of Qbox’s strong scaling after optimizations; runtime of improves from ~400 to ~30s per SCF iteration (13x speedup) on 131,072 ranks for 2048 electrons.
Reducing communication overheads from ScaLAPACK by “transpose” remap

Problem of “gather & scatter”: Idle processes. How to utilize them? Assign idle processes to active columns.

Transpose remap:
• Perform 3D FFTs in the original context.
• Transfer data through a series of local regional transposes
• Run ScaLAPACK in the new context

Key concept for remap: creating different contexts that are optimal for different kernels redistributing the data on-the-fly

Process rearrangement and data movement of transpose remap

 Improvement of runtime by remap methods
(1) \( npcol' = \frac{npcol}{8}, nprow' = nprow \)
(2) \( npcol' = \frac{npcol}{8}, nprow' = 8 \times nprow \)
Conclusion and Insights

• Band parallelization reduces the internode communication overhead and improves strong scaling of WEST up to $N_{\text{FFT}}N_{\text{pert}}N_{\text{band}}$ cores.

• Optimal remapping of data for matrix operations in Qbox reduces ScaLAPACK communication overhead at large scale, and makes hybrid- DFT calculation scale to $N_{\text{FFT}}N_{\text{band}}$ cores.

• Given the increased computational performance relative to network bandwidths, it is crucial to reduce and/or hide inter-node communication costs.

• Guiding principles for developing codes in many-core architecture:
  1) Parallelizing independent, fine-grain units of work, reducing inter-node communication, and maximizing utilization of on-node resources.
  2) Optimizing communication patterns for performance critical kernels with on-the-fly data redistribution and process reconfiguration.
Acknowledgement

• This research is part of Theta Early Science Project at Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility under Contract DE-AC02-06CH11357.

• This work was supported by MICCoM, as part of Comp. Mats. Sci. Program funded by the U.S. DOE, Office of Science, BES, MSE Division.