Performance Improvement of Electronic Structure Calculations using MIC programming: A Sparse-matrix-coupled Normal Eigenvalue Problem

Hoon Ryu, Ph.D.

Senior Researcher and PI of IPCC@KISTI

Korea Institute of Science and Technology Information (KISTI)

Republic of Korea

Introduction: Ongoing Project

- **A High-fidelity Schrödinger-Poisson Solver**
  - In-house code
  - The code solves large-scale electronic structures using an empirical tight-binding approach. The calculations are self-consistent as we solve a tight-binding Schrödinger equation coupled to a FDM Poisson equation. The code would be useful to help experimentalists by presenting a guideline for design of realistically sized semiconductor devices.
- **Application Domain:** Electrical Engineering and Solid State Physics
- **Execution mode:** offloaded
- **Tools used for development, analysis and debugging**
  - Intel C/C++/Fortran v. 15.0.2
  - Intel MPI 5.0 Update 2 for Linux and Intel MPSS v. 3.4.2
- **Sparse-matrix-involved operation**
  - Schrödinger Equations: LANCZOS Iteration
  - Poisson Equations: CG Iteration

Performance

- **Compelling Performance**

  - with and without MIC

- **Performance Improvement**

  - 1.5x~3x with xeon+MIC against the performance with xeon only

- **List of optimizations that yield performance improvment**

  - Control of computing load in Xeon and MIC for Asynchronous offload

    → the best performance is obtained when MIC takes 65~80% of computing load, depending on the specs of xeon server.

  - Minimization of data transfer

    → Schrödinger matrix (Hamiltonian) can be copied only one time.

    → Minimization of Matrix Data Size: Sparse Matrix (CSR)

    → Copy of Lanczos vectors can be minimized (in terms of size): nearest neighbor coupling

Insights

- **What we have learned?**

  - (Synchronous) offload can be still a good solution if the target computing can be efficiently vectorized, i.e. dense-matrix operations. But, the sparse-matrix-involved operations are generally hard to be efficiently vectorized in MIC.

- **What we recommend and how we would have done it differently?**

  - Adopted the Asynchronous offloading to make CPUs busy with MIC

- **Biggest surprises**

  - Optimal load balancing for MV multipler

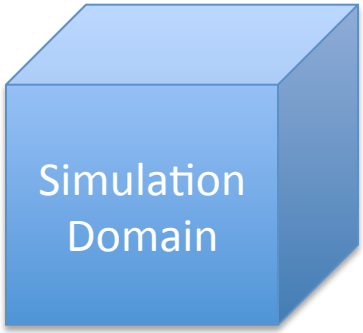
    → the best performance is obtained when MIC takes 65~80% of computing load, depending on the specs of xeon server. (Expected under 50% as CPUs have better performance)

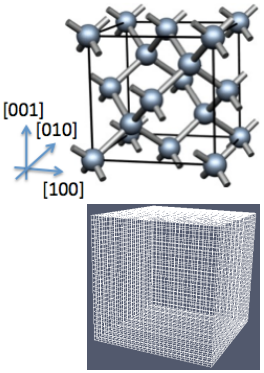
- Key remaining challenges

  - Large-scale BMT: in > 10 nodes
  - Performance improvement: Parallelization of T matrix handler in traditional Lanczos Loop
  - Expansion: solving degenerated eigenvalues

Intel® PCC Project: A Two-year Milestone

Mesh Generator





Core PDE Solvers using MIC Programming

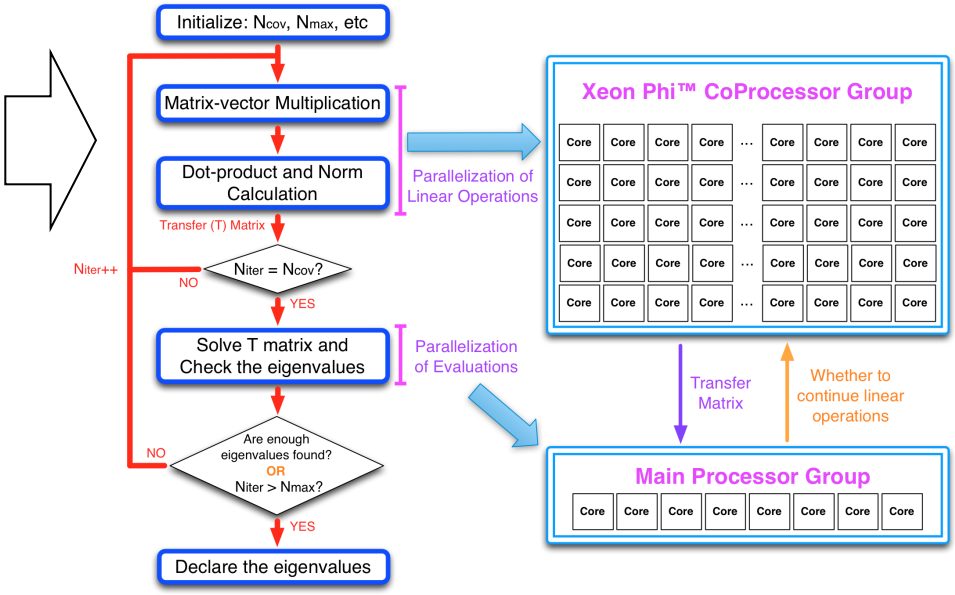
Schrödinger Solver (LANCZOS)

$$H\Psi = E\Psi$$

Poisson Solver (Conjugate Gradient)

$$-\nabla(\varepsilon\nabla V) = \rho$$

Research Articles



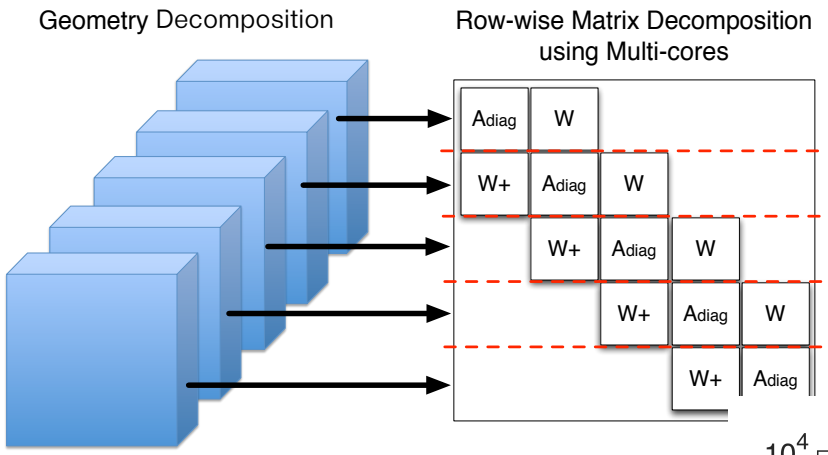
- Inhomogeneous P dopant-distributions in Si Nanowires
- Performance Improvement of PDE solvers using Xeon-Phi™

Training for Scientific Computing

- Sparse Matrix Storage Algorithm
- MIC Programming with Application Examples
- Finite Difference Methods

Performance on a CPU-level

Domain Decomposition



Schrödinger (LANCZOS)

loop for (j=1; j<=**K** ; j++)

$\mathbf{w}_j \leftarrow \mathbf{A}\mathbf{v}_j$;

$\mathbf{a}_j \leftarrow \mathbf{w}_j \bullet \mathbf{v}_j$;

$\mathbf{w}_j \leftarrow \mathbf{w}_j - \mathbf{a}_j\mathbf{v}_j - \mathbf{b}_j\mathbf{v}_{j-1}$;

$\mathbf{b}_{j+1} \leftarrow \|\mathbf{w}_j\|$;

$\mathbf{v}_{j+1} \leftarrow \mathbf{w}_j / \mathbf{b}_{j+1}$;

Poisson (CG)

loop for (j=1; j<=**K** ; j++)

$\mathbf{a}_j \leftarrow \langle \mathbf{r}_j \bullet \mathbf{r}_j \rangle / \langle \mathbf{A}\mathbf{p}_j \bullet \mathbf{p}_j \rangle$;

$\mathbf{x}_{j+1} \leftarrow \mathbf{x}_j + \mathbf{a}_j\mathbf{p}_j$;

$\mathbf{r}_{j+1} \leftarrow \mathbf{r}_j - \mathbf{a}_j\mathbf{A}\mathbf{p}_j$;

if ($\|\mathbf{r}_{j+1}\|/\|\mathbf{r}_0\| < \mathbf{e}$)

declare $\mathbf{r}_{j+1}$ is the solution

$\mathbf{c}_j \leftarrow \langle \mathbf{r}_{j+1} \bullet \mathbf{r}_{j+1} \rangle / \langle \mathbf{r}_j \bullet \mathbf{r}_j \rangle$;

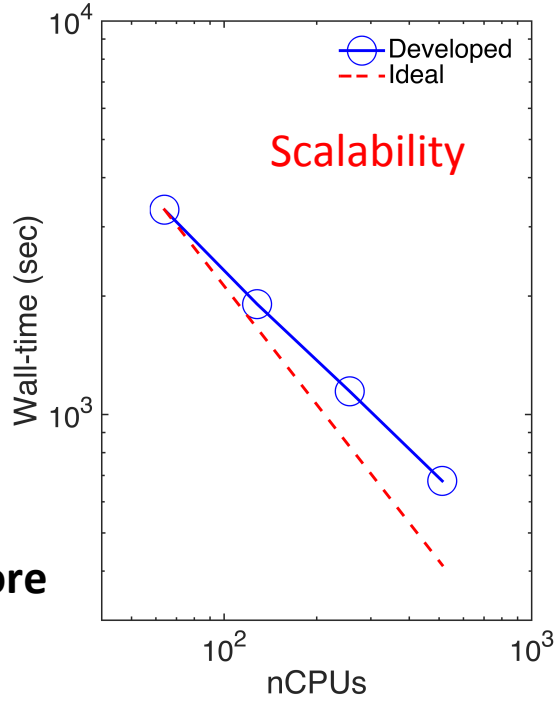
$\mathbf{p}_{j+1} \leftarrow \mathbf{r}_{j+1} + \mathbf{c}_j\mathbf{p}_j$;

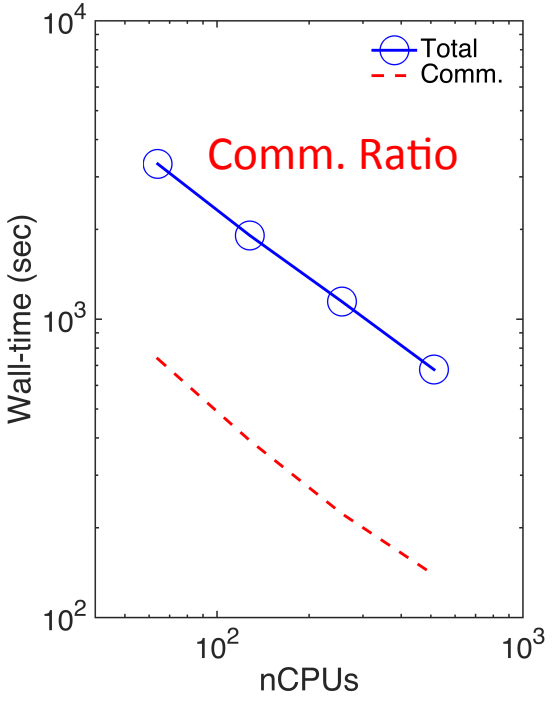
Performance Benchmarking @ KISTI Tachyon-II

- 278x16x16nm [100] Si nanowires
- 4194304 atoms
- Find 5 lowest energy-levels

- Scalability upto ~65%

- Computing Intensive:
How to reduce computing time more with less # of nodes?





Performance Improvement: "Asynchronous" Offload

The real bottleneck of computing

- Vector dot-product is not expensive: All-reduce, but small communication loads.
- Vector communication is not a big deal: only communicates between adjacent layers.
- Sparse-matrix-vector multiplication itself is indeed a big deal.

Schrödinger (LANCZOS)

loop for (j=1; j<=**K** ; j++)

$\mathbf{w_j} \leftarrow \mathbf{A v_j}$;

$\mathbf{a_j} \leftarrow \mathbf{w_j} \bullet \mathbf{v_j}$;

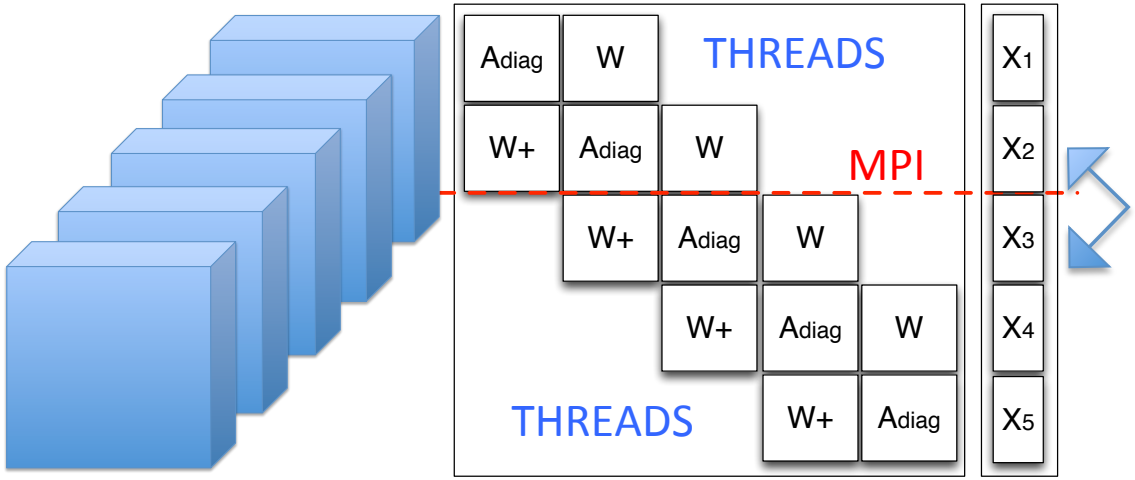
$\mathbf{w_j} \leftarrow \mathbf{w_j} - \mathbf{a_j v_j} - \mathbf{b_j v_{j-1}}$;

$\mathbf{b_{j+1}} \leftarrow \|\mathbf{w_j}\|$;

$\mathbf{v_{j+1}} \leftarrow \mathbf{w_j} / \mathbf{b_{j+1}}$;

- Matrix-vector multiplier (MVMul)
  : Nearest neighbor comm.
- Vector dot product (VVDot)
  : Reduction
- Others: No communication

Communication Pattern for MVmultiplier



Food for Thoughts

- Sparse MV multiplier
  → cash miss: full performance of vectorization?
- Synchronous offload: CPU should be idle
- **Why not asynchronous offload?**

Performance Improvement: Case 1

Benchmark in a testbed 1 @ KISTI

- 1 node: 4 Xeon E5-2603 v2 CPUs (1.8GHz), 16GB M w/ a 3120 card (6GB M)
- Problem size: A P-atom integrated in 22nmx22nmx22nm Si box (DOF: 5.12M, M ~ 3.8GB)
- Target: Run 5000 Lanczos iterations to find as many eigenvalues as possible
- Cond: 1 MPI process with 4 threads per CPU, 224 threads in MIC (56*4)

Schrödinger (LANCZOS)

loop for (j=1; j<=**K** ; j++)

$\mathbf{w}_j \leftarrow \mathbf{A}\mathbf{v}_j$;

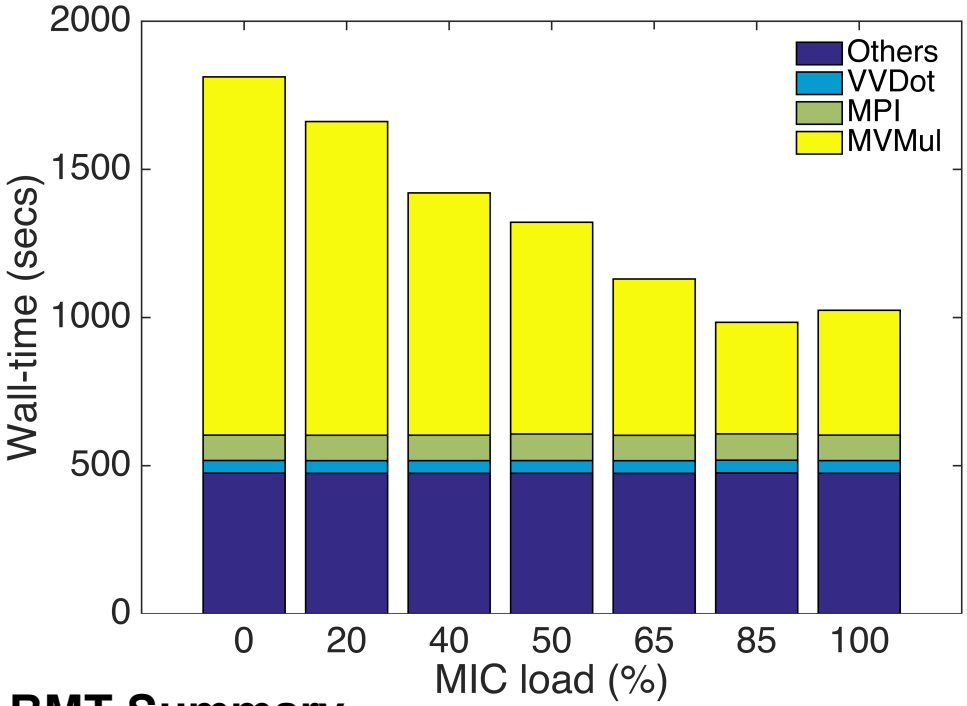
$\mathbf{a}_j \leftarrow \mathbf{w}_j \bullet \mathbf{v}_j$;

$\mathbf{w}_j \leftarrow \mathbf{w}_j - \mathbf{a}_j\mathbf{v}_j - \mathbf{b}_j\mathbf{v}_{j-1}$;

$\mathbf{b}_{j+1} \leftarrow \|\mathbf{w}_j\|$;

$\mathbf{v}_{j+1} \leftarrow \mathbf{w}_j / \mathbf{b}_{j+1}$;

- Matrix-vector multiplier (MVMul)
  : Nearest neighbor comm.
- Vector dot product (VVDot)
  : Reduction
- Others: No communication



BMT Summary

- 85% load in MIC for MVMultiplier: Best
- ~3x Improvement in MVMul, ~2x in Total

Performance Improvement: Case 2

Benchmark in a testbed 2 @ KISTI – super-excellent spec

- 2 nodes: 20 Xeon E5-2680 v2 CPUs (2.8GHz), 256GB M w/ two 7120 card (32GB M total)
- Problem size: A P-atom integrated in 22nmx71nmx71nm Si box (DOF: 54M, M ~ 41GB)
- Target: Run 5000 Lanczos iterations to find as many eigenvalues as possible
- Cond: 4 MPI process with 10 threads per CPU, 240 threads in MIC (60*4)

Schrödinger (LANCZOS)

loop for (j=1; j<=**K** ; j++)

$\mathbf{w}_j \leftarrow \mathbf{A}\mathbf{v}_j$;

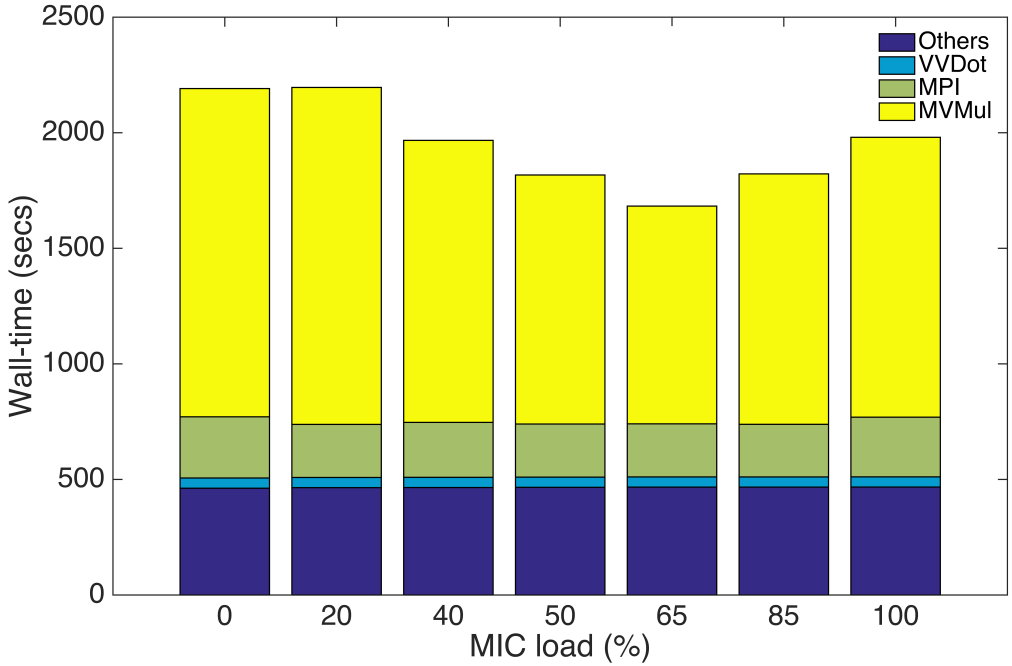
$a_j \leftarrow \mathbf{w}_j \bullet \mathbf{v}_j$;

$\mathbf{w}_j \leftarrow \mathbf{w}_j - a_j\mathbf{v}_j - b_j\mathbf{v}_{j-1}$;

$b_{j+1} \leftarrow \|\mathbf{w}_j\|$;

$\mathbf{v}_{j+1} \leftarrow \mathbf{w}_j / b_{j+1}$;

- Matrix-vector multiplier (MVMul)
  : Nearest neighbor comm.
- Vector dot product (VVDot)
  : Reduction
- Others: No communication



BMT Summary

- 65% load in MIC for MVMultiplier: Best
- ~1.5x Improvement in MVMul, ~1.3x in Total

Other Progress, Near-future Plans

Research Outcomes (Objective: 2 journal articles, 2 proceedings by May-2016)

- 1 Journal Article (Nano Letters, IF 13.7), 1 Proceeding (IEEE SISPAD) published so far.
- 1 Proceeding Accepted (MRS), 1 Journal Article under Review

Knowledge Disseminations: Education (Objective: 100 students by May-2016)

- Intel(Korea)-KISTI Collaborative Workshop: 25 students (Jun-15)
- HPC summer school in SNU: 30 students (Aug-15)
- Korea Supercomputing Conference: 30 developers (Oct-2015)

Near-future Plans for R/D

- Benchmark for end-to-end simulations in more realistic conditions in a large # of clusters
- Another Possibility for Performance Improvement: “T-matrix computing” in LANCZOS
- Research Papers