

Energy-efficient, scalable computing of extremely large electronic structures with KNL processors

Hoon Ryu, Ph.D.

(E: elec1020@kisti.re.kr)

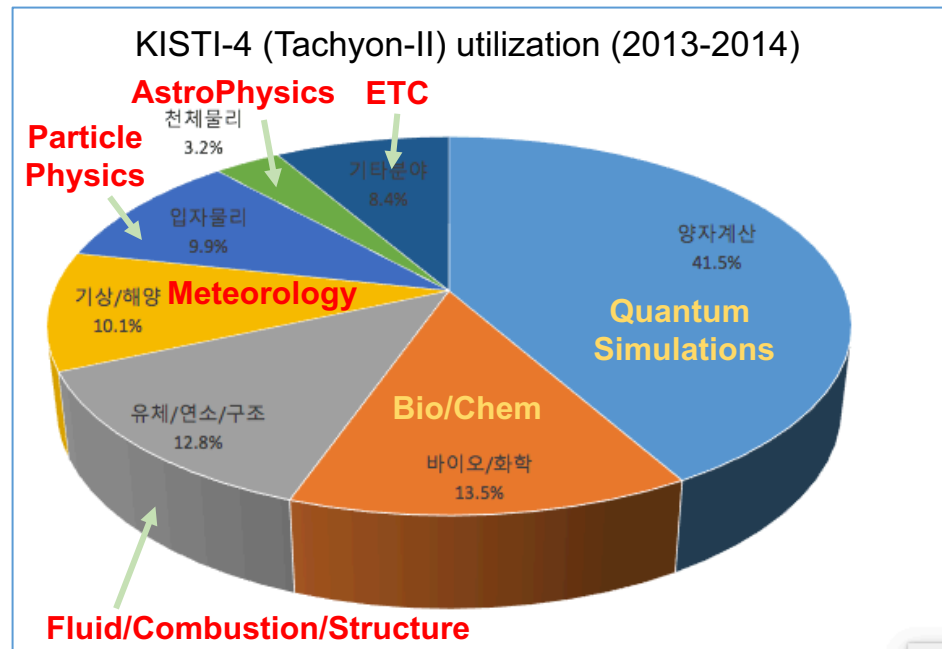
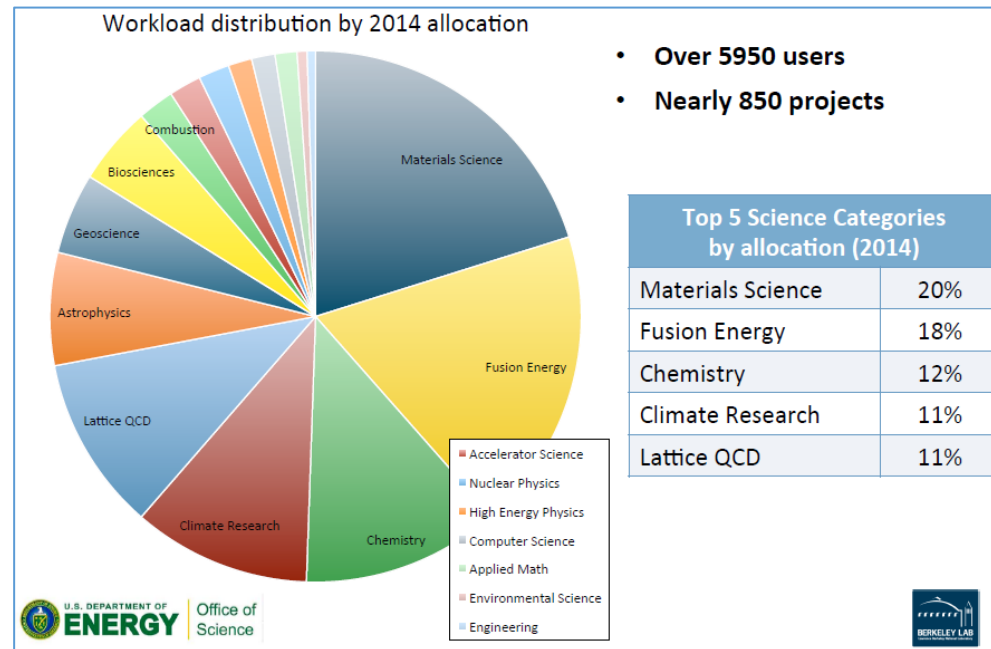
Principal Researcher / Korea Institute of Science and Technology Information (KISTI)
Principal Investigator / KISTI Intel® Parallel Computing Center

Electronic Structure Calculations

An application that has customers in a HUGE range



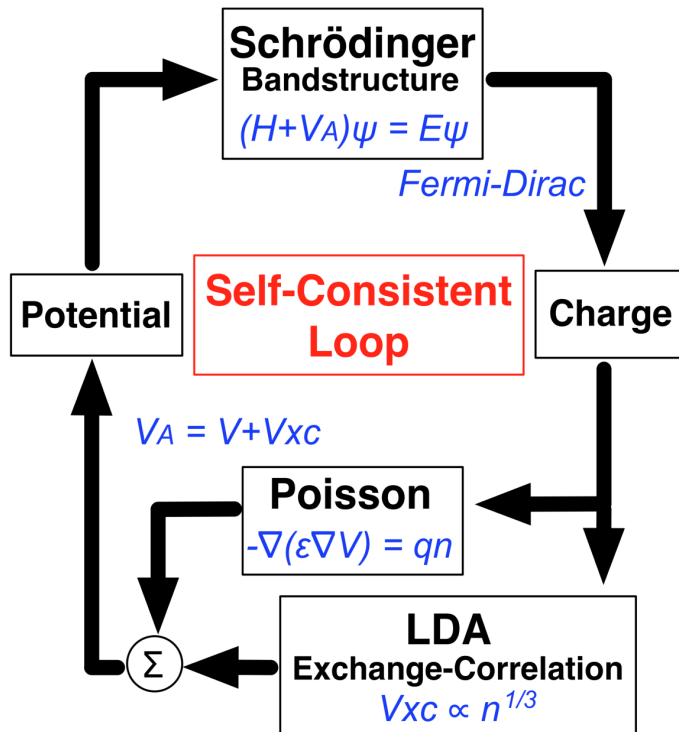
- The state of motion of electrons in an electrostatic field created by the stationary nuclei.
- Quantum Physics; Quantum Chemistry; Nanoscale Materials and Devices
→ Physics, Chemistry, Materials Science, Electrical Engineering and Mechanical Engineering ETC.
- Huge customers in the society of computational science



Electronic Structure Calculations

In a perspective of “numerical analysis”

- Two PDE-coupled Loop: Schrödinger Equation and Poisson Equation
- Both equation involve system matrices (Hamiltonian and Poisson)
 - DOFs of those matrices are proportional to the # of grids in the simulation domains



- Schrödinger Equations

→ Normal Eigenvalue Problem

$$H\Psi = E\Psi$$

- Poisson Equations

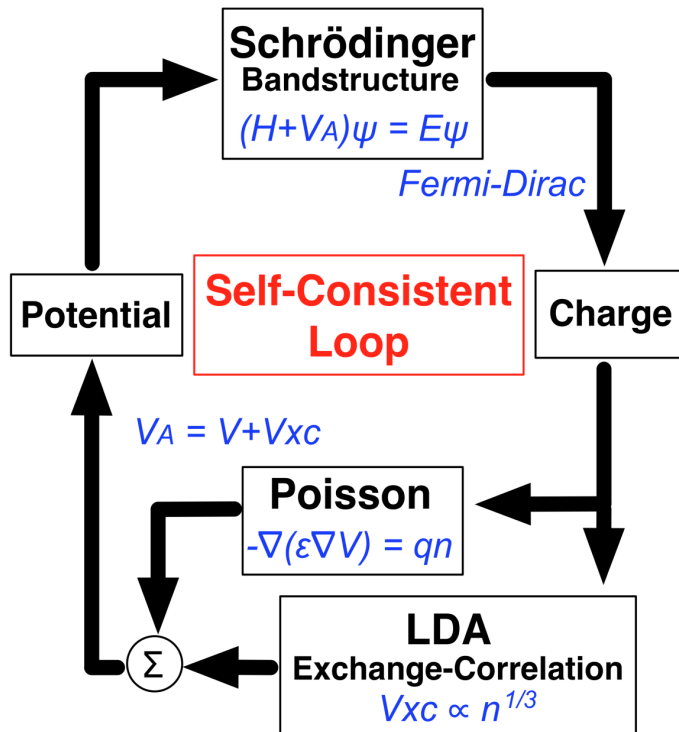
→ Linear System Problem

$$-\nabla(\epsilon\nabla V) = \rho \rightarrow Ax = b$$

Electronic Structure Calculations

In a perspective of “numerical analysis”

- Two PDE-coupled Loop: Schrödinger Equation and Poisson Equation
- Both equation involve system matrices (Hamiltonian and Poisson)
 - DOFs of those matrices are proportional to the # of grids in the simulation domains



- Schrödinger Equations

→ Normal Eigenvalue Problem

$$\textcircled{H}\Psi = E\Psi$$

- Poisson Equations

→ Linear System Problem

$$-\nabla(\epsilon\nabla V) = \rho \rightarrow \textcircled{A}x = b$$

How large are these system matrices?
Why do we need to handle those?

Needs for “Large” Electronic Structures

Needs for High Performance Computing

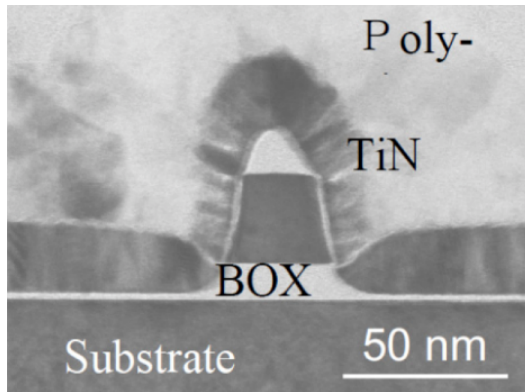
1. Quantum Simulations of “Realizable” Nanoscale Materials and Devices

→ Needs to handle large-scale atomic systems (~ A few tens of nms)

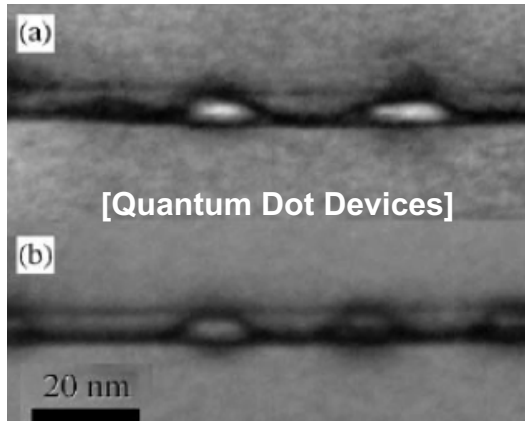
30nm³ Silicon Box? → About a million atoms

2. DOF of Matrices of Governing Equations

→ Linearly proportional to # of atoms (w/ some weight)



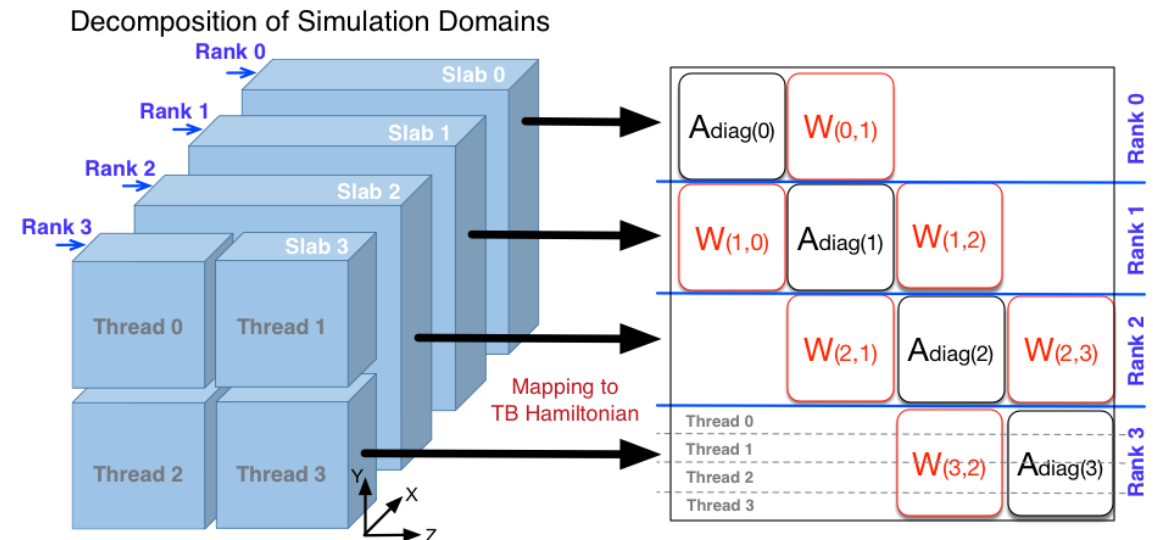
[Logic Transistors (FinFET)]



[Quantum Dot Devices]

3. Parallel Computing

$$Ax = b$$
$$H\Psi = E\Psi$$



Development Strategy: DD, Matrix Handling

Large-scale Schrödinger, Poisson Eqns.

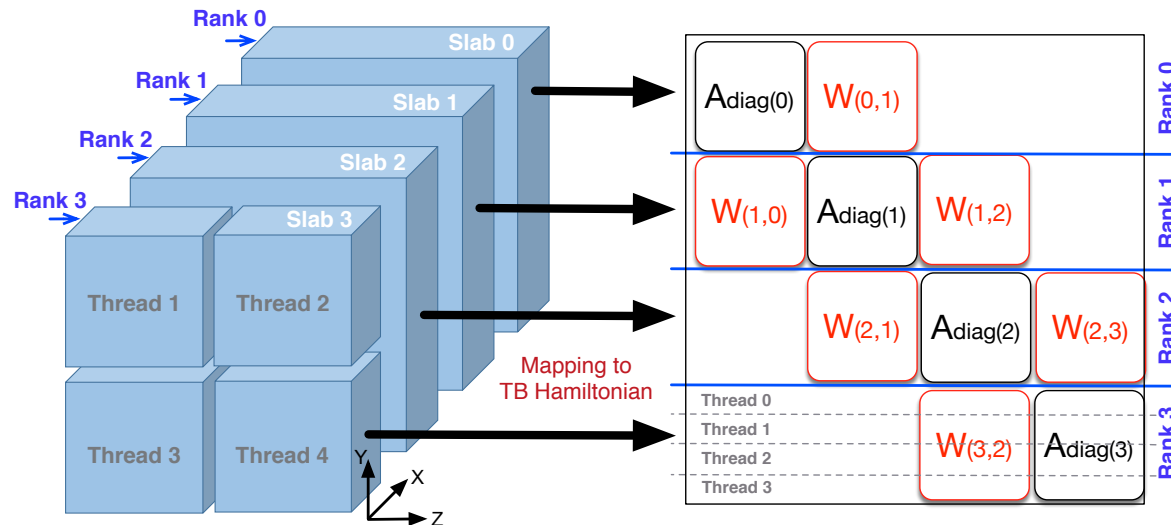
Schrödinger Equation

- Normal Eigenvalue Problem (Electronic Structure)
- Hamiltonian is always symmetric

$$H\Psi = E\Psi$$

Domain Decomposition

- MPI + OpenMP
- Effectively multi-dimensional decomposition



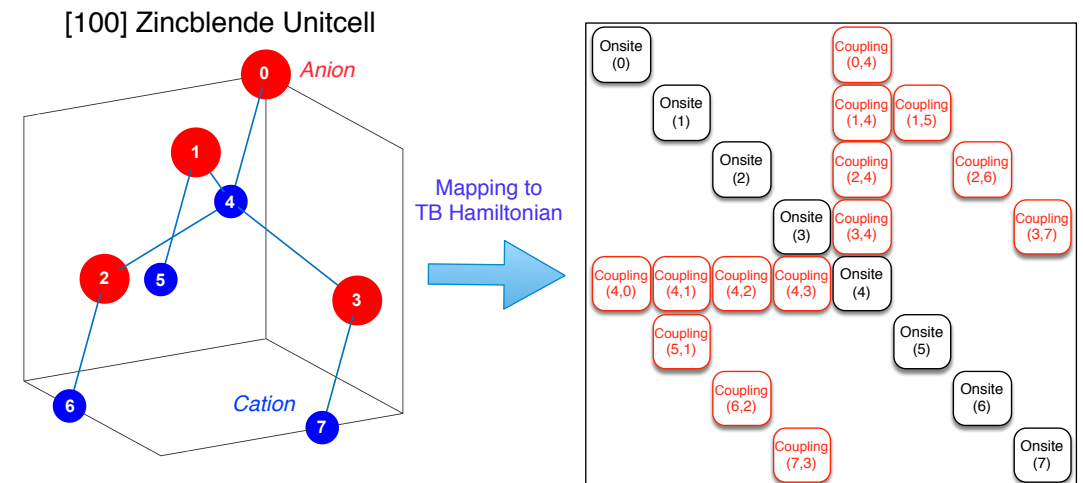
Poisson Equation

- Linear System Problem (Electrostatics: Q-V)

- Poisson matrix is always symmetric $-\nabla(\epsilon\nabla V) = \rho$

Matrix Handling

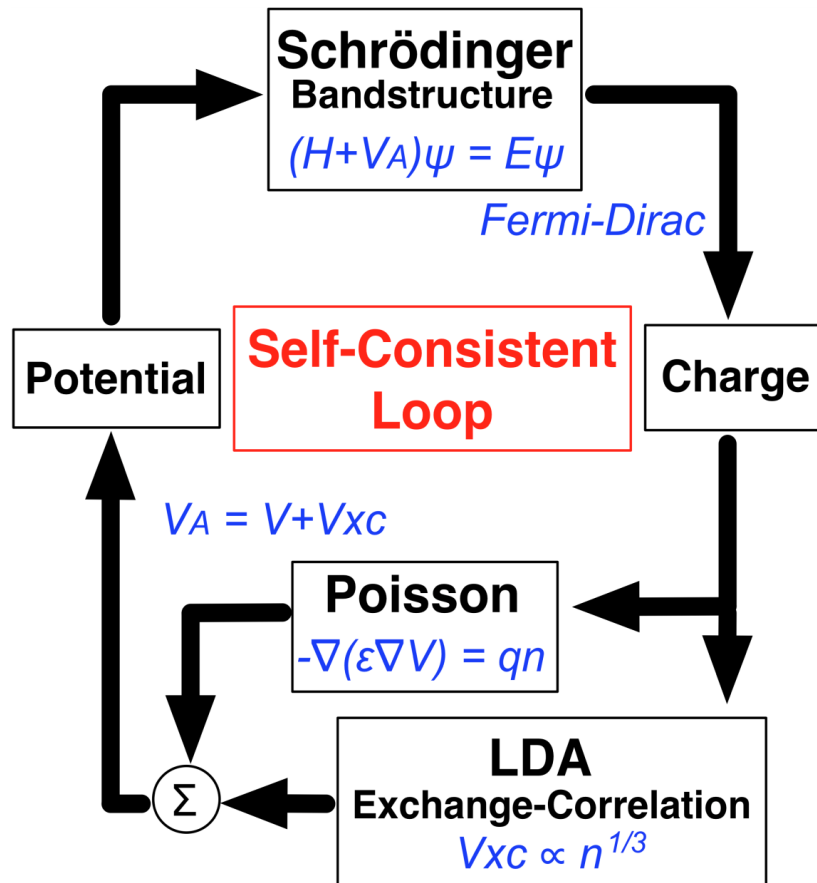
- Tight-binding Hamiltonian (Schrödinger Eq.)
- Finite Difference Method (Poisson Eq.)
- Nearest Neighbor Coupling: Highly Sparse \rightarrow CSR



Development Strategy: Numerical Algorithms

Schrödinger Equations

Self-consistent Loop for Device Simulations



Schrödinger Eqs. w/ LANCZOS Algorithm

→ C. Lanczos, *J. Res. Natl. Bur. Stand.* 45, 255

- Normal Eigenvalue Problem (Electronic Structure)
- Hamiltonian is always symmetric
- Original Matrix → T matrix-reduction
- Steps for Iteration: Purely Scalable Algebraic Ops.

$$H\Psi = E\Psi$$

v_i : ($N \times 1$) vectors ($i = 0, \dots, K$); a_i and b_i : scalars ($i = 1, \dots, K$)

$v_0 \leftarrow 0$, v_1 = random vector with norm 1 ;

$b_1 \leftarrow 0$;

loop for ($j=1$; $j \leq K$; $j++$)

$w_j \leftarrow A v_j$;

$a_j \leftarrow w_j \bullet v_j$;

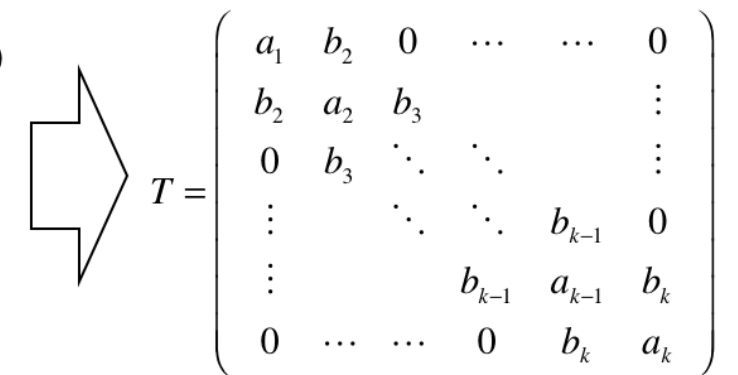
$w_j \leftarrow w_j - a_j v_j - b_j v_{j-1}$;

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

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

construct T matrix;

end loop



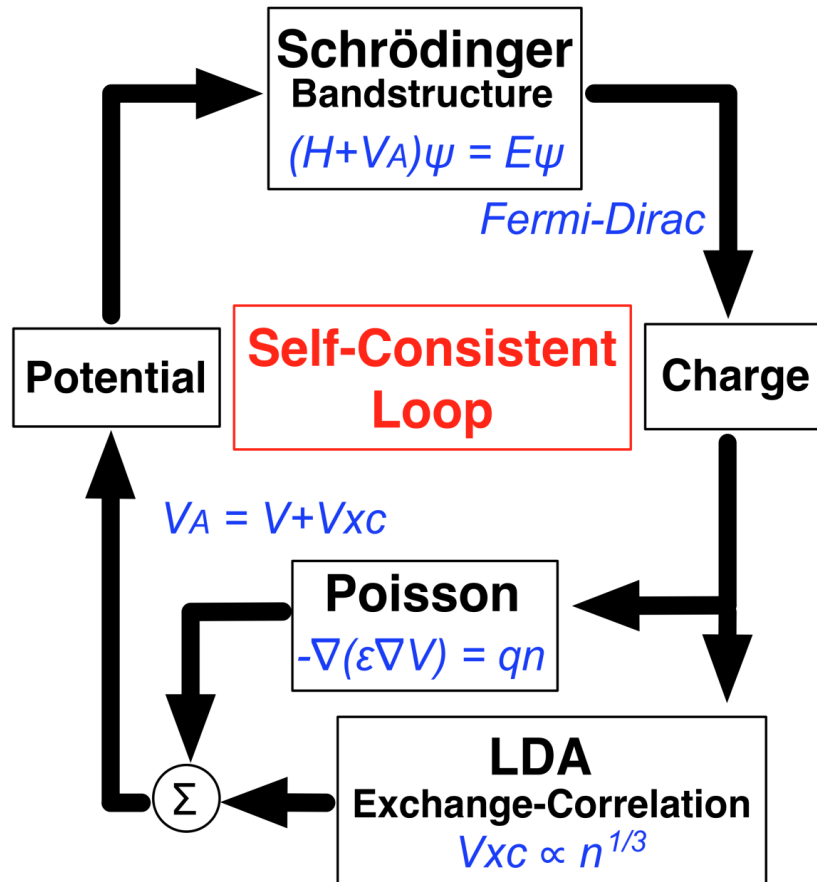
The diagram shows the construction of the T matrix, which is a tridiagonal matrix. The matrix is represented as:

$$T = \begin{pmatrix} a_1 & b_2 & 0 & \cdots & \cdots & 0 \\ b_2 & a_2 & b_3 & & & \vdots \\ 0 & b_3 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & b_{k-1} & 0 \\ \vdots & & & b_{k-1} & a_{k-1} & b_k \\ 0 & \cdots & \cdots & 0 & b_k & a_k \end{pmatrix}$$

Development Strategy: Numerical Algorithms

Poisson Equations

Self-consistent Loop for Device Simulations



Poisson Eqs. w/ CG Algorithm

→ A Problem of Solving Linear Systems

- Conv. Guaranteed: Symmetric & Positive Definite
- Poisson is always S & PD.
- Steps for Iteration: Purely Scalable

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

Algebraic Ops.

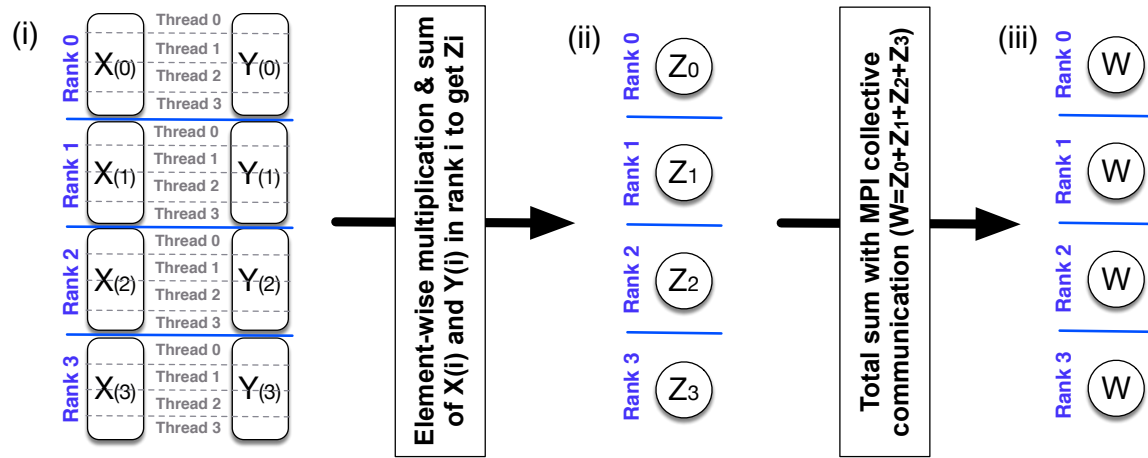
We want to solve $\mathbf{Ax} = \mathbf{b}$. First compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, $\mathbf{p}_0 = \mathbf{r}_0$

```
loop for (j=1; j<=K ; j++)
  a_j ← < r_j • r_j > / < A p_j • p_j >;
  x_{j+1} ← x_j + a_j p_j;
  r_{j+1} ← r_j - a_j A p_j;
  if (||r_{j+1}|| / ||r_0|| < e)
    declare r_{j+1} is the solution of Ax = b and break the loop
  c_j ← < r_{j+1} • r_{j+1} > / < r_j • r_j >;
  p_{j+1} ← r_{j+1} + c_j p_j;
end loop
```


Performance Bottleneck?

Matrix-vector Multiplier: Sparse Matrices

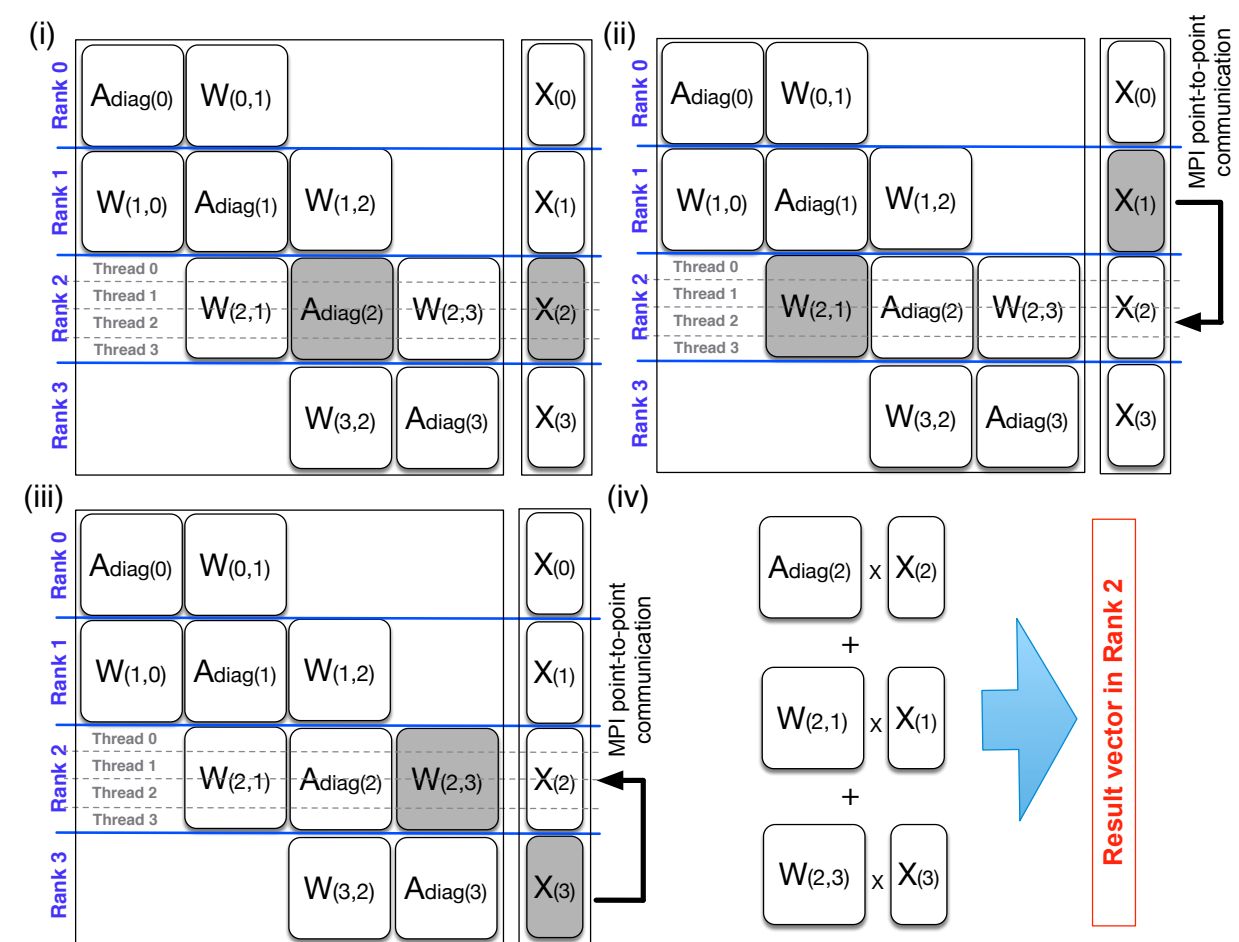
Vector Dot-Product (VVDot)



Main Concerns for Performance

- Collective Communication
 - May not be the main bottleneck as we only need to collect a single value from a single MPI process
- Matrix-vector Multiplier
 - Communication happens, but would not be a critical problem as it only happens between adjacent ranks
 - Cache-miss affects vectorization efficiency

(Sparse) Matrix-vector Multiplier (MVMul)



Performance w/ Intel® KNL Processors

Single-node Performance: The power of MCDRAM

Description of BMT Target and Test Mode

- 10 CB states in $16 \times 43 \times 43 (\text{nm}^3)$ [100] Si:P quantum dot
→ Material candidates for Si Quantum Info. Processors
(Nature Nanotech. **9**, 430)
→ $15.36\text{M} \times 15.36\text{M}$ Hamiltonian Matrix
- KNL (Xeon Phi 7210); Up to 256 (64×4) cores
- MCDRAM control w/ [numactrl](#); Quad Mode

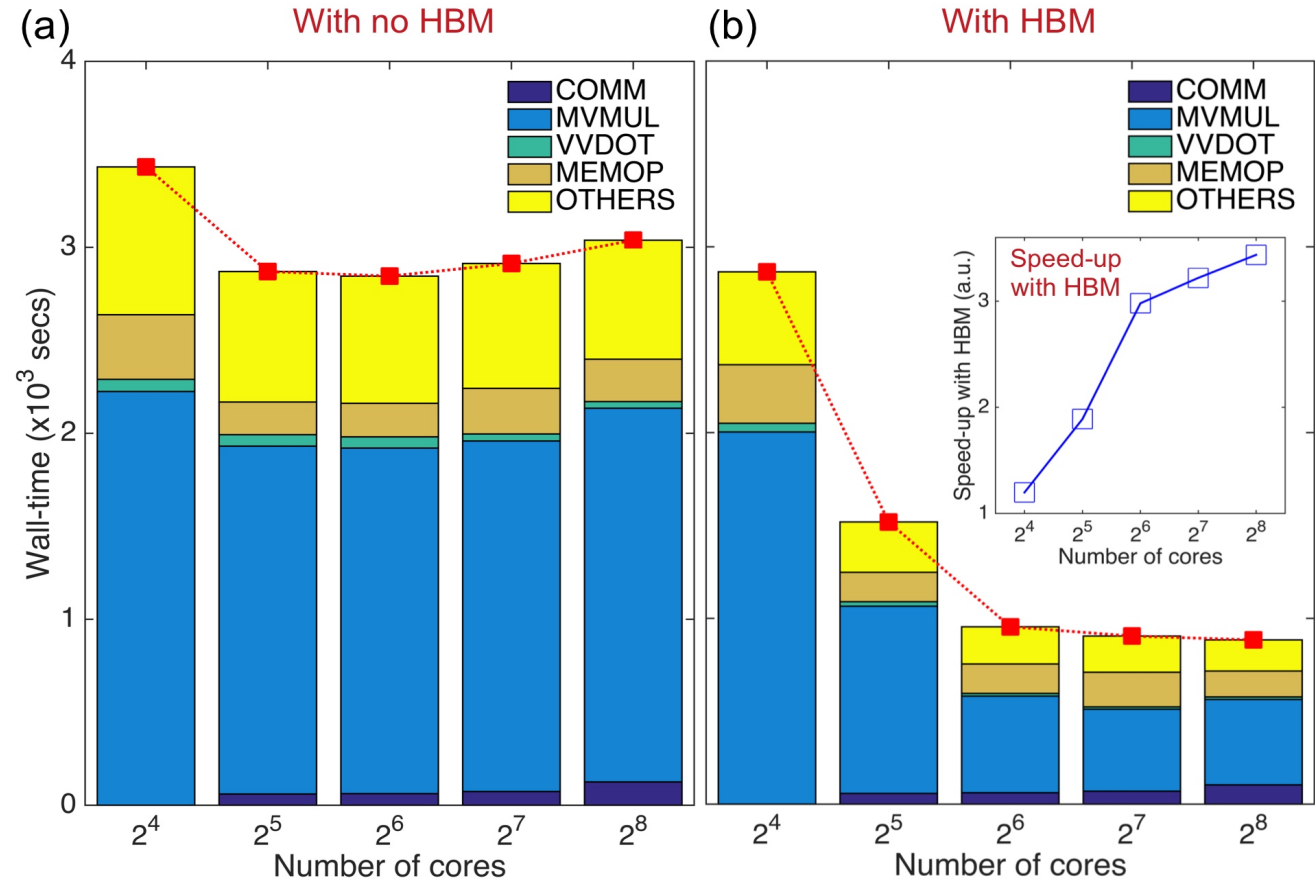
Results

[H. Ryu, Intel® HPC Developer Conference \(2017\)](#)

- With no MCDRAM
→ No clear speed-up beyond 64 cores
- **With MCDRAM**
→ Up to ~4x speed-up w.r.t. the case w/ no MCDRAM
→ Intra-node scalability up to 256 cores

Points of Questions

- How is the performance compared to the one under other computing environments? (GPU, CPU-only etc..)
→ In terms of speed and **energy consumption**



2^4 cores = (1 MPI proc(s), 16 threads), 2^7 cores = (2 MPI proc(s), 64 threads)
 2^5 cores = (2 MPI proc(s), 16 threads), 2^8 cores = (4 MPI proc(s), 64 threads)
 2^6 cores = (2 MPI proc(s), 32 threads)

Performance w/ Intel® KNL Processors

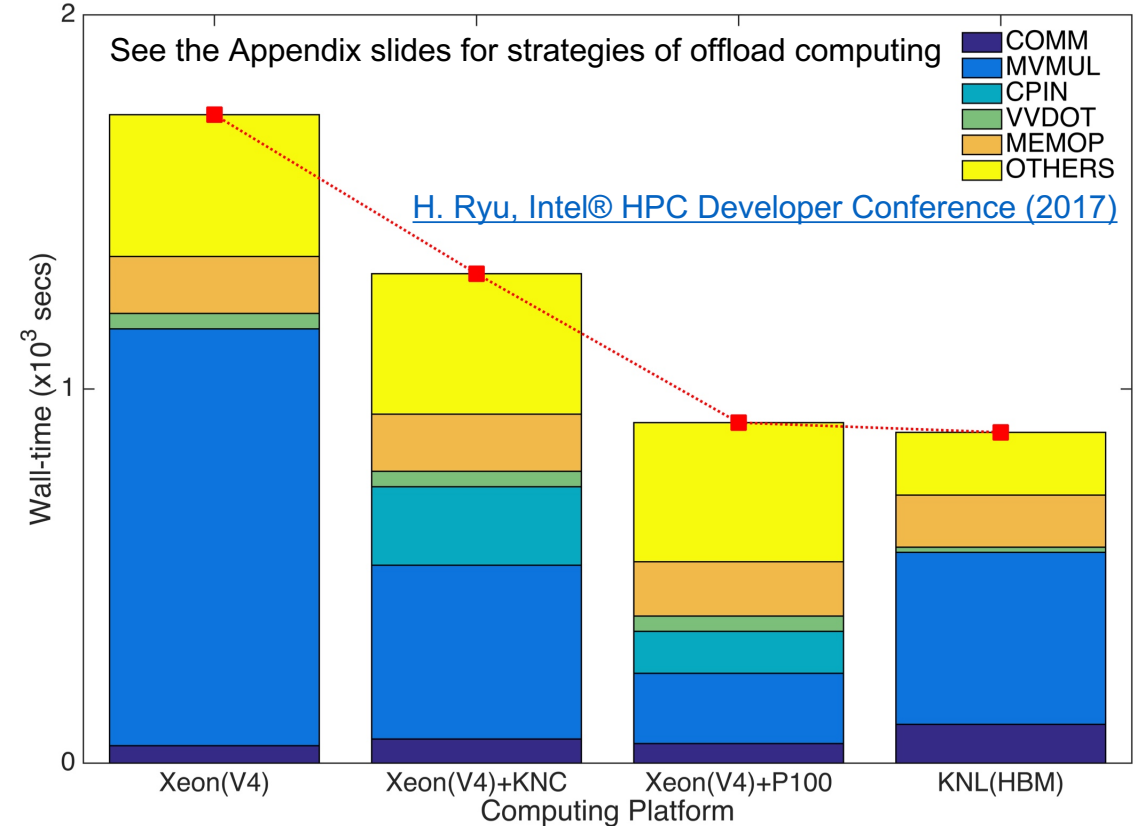
Single-node Performance: Speed

Description of BMT Target and Test Mode

- 10 CB states in $16 \times 43 \times 43 (\text{nm}^3)$ [100] Si:P quantum dot
→ Material candidates for Si Quantum Info. Processors (Nature Nanotech. **9**, 430)
→ $15.36 \text{M} \times 15.36 \text{M}$ Hamiltonian Matrix
- KNL (Xeon Phi 7210); Up to 256 (64×4) cores; Quad Mode
- Specs of Other Platforms
→ Xeon(V4): 24 cores of Broadwell (BW) 2.50GHz
→ Xeon(V4)+KNC: 24 cores BW + 2 KNC 7120 cards
→ Xeon(V4)+P100: 24 cores BW + 2 P100 cards
→ KNL(HBM): the one described so far

Results

- **KNL slightly beats Xeon(V4)+P100**
→ Copy-time (CPIN): a critical bottleneck of PCI-E devices
→ P100 shows better kernel speed, but the overall benefit reduces due to data-transfer between host and devices
→ CPIN would even increase if we consider periodic BCs
- **Another critical figure of merit: Energy-efficiency**



Xeon(V4) = (2 MPI proc(s), 12 threads)

Xeon(V4) + KNC = (2 MPI proc(s), 12 threads) + 2 KNC 7120 cards

Xeon(V4) + P100 = (2 MPI proc(s), 12 threads) + 2 P100 cards

KNL (HBM) = (4 MPI proc(s), 64 threads) with HBM

Performance w/ Intel® KNL Processors

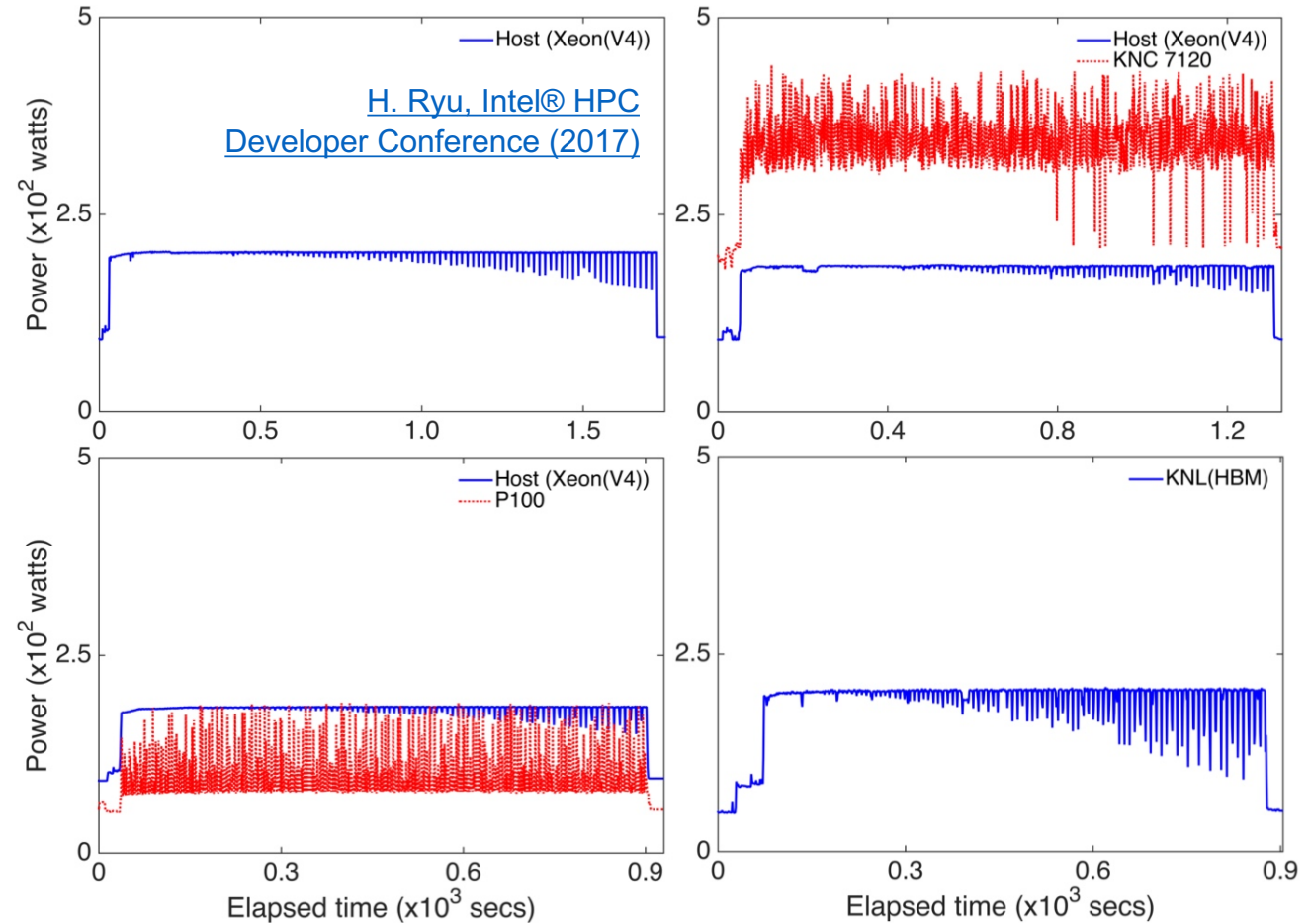
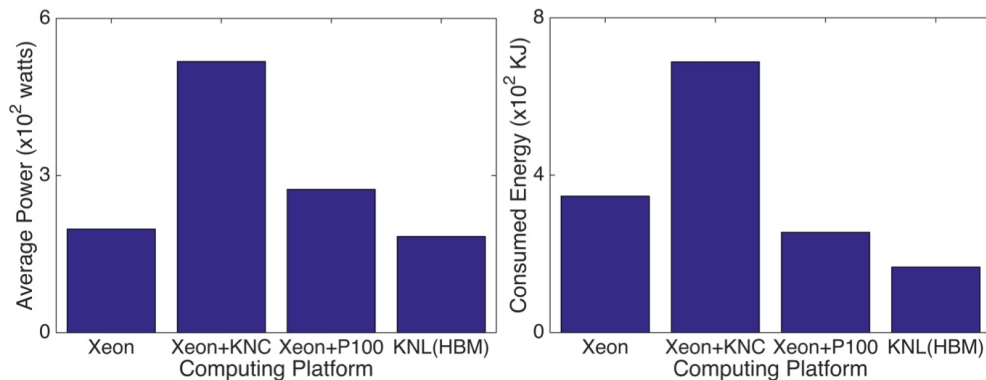
Single-node Performance: Energy Consumption

Description of BMT Target and Test Mode

- 10 CB states in $16 \times 43 \times 43 (\text{nm}^3)$ [100] Si:P quantum dot
→ Hamiltonian DOF: $15.36\text{M} \times 15.36\text{M}$
- Description of Device Categories
 - Xeon(V4): 24 cores of Broadwell (BW) 2.50GHz
 - Xeon(V4)+KNC: 24 cores BW + 2 KNC 7120 cards
 - Xeon(V4)+P100: 24 cores BW + 2 P100 cards
 - KNL(HBM): the one described so far

Power Measurement

- w/ RAPL (Running Ave. Power Limit) API
- Host (CPU+Memory), PCI-E Devices



Results: KNL consumes 2x less energy than Xeon(V4)+P100

Performance w/ Intel® KNL Processors

Multi-node Performance: Scalability

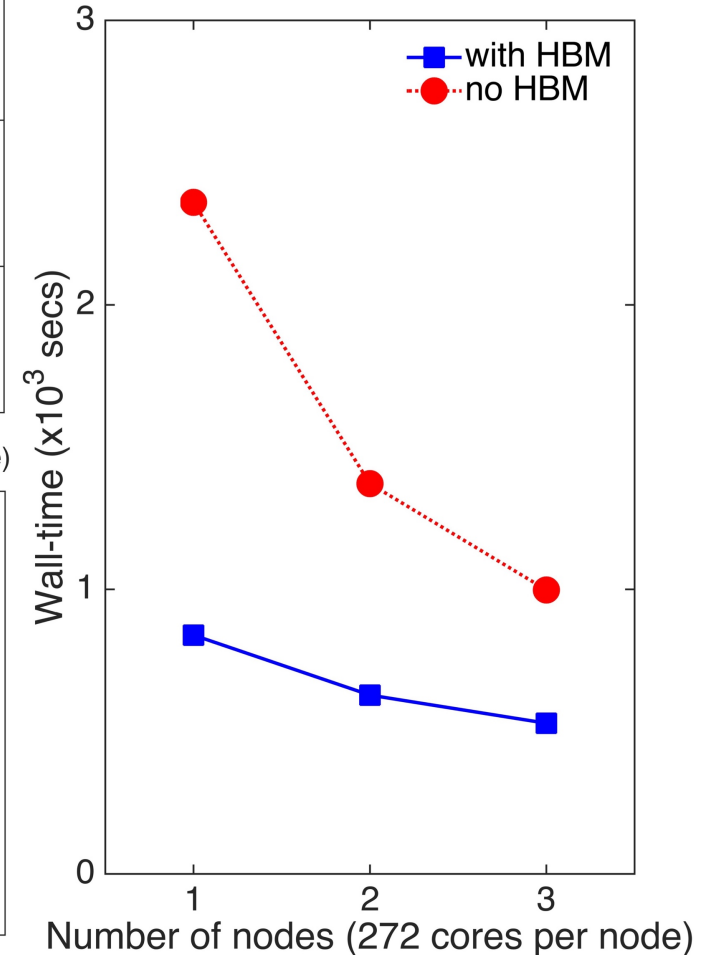
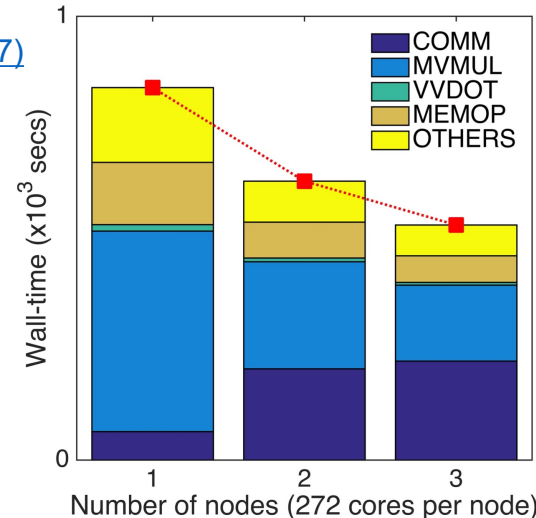
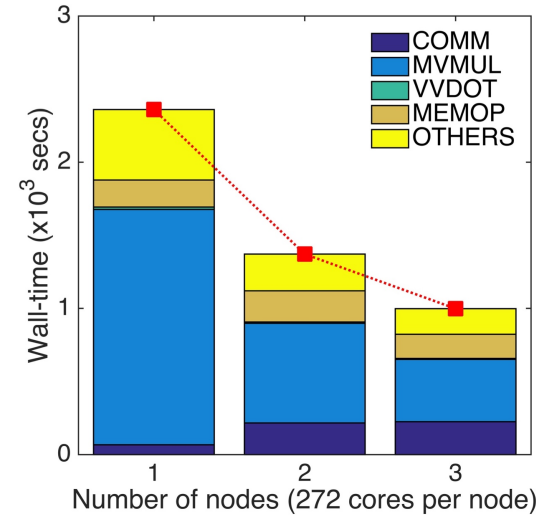
Description of BMT Target and Test Mode

- 5 CB states in $27 \times 33 \times 33 (\text{nm}^3)$ [100] Si:P quantum dot
→ Material candidates for Si Quantum Info. Processors (Nature Nanotech. **9**, 430)
→ 14.4Mx14.4M Hamiltonian Matrix
- KNL (Xeon Phi 7250) nodes; Up to 272 (68x4) cores/node
→ (4 MPI processes + 68 threads) per node
→ Quad / Flat mode, No omni-path (10G network)
→ Strong scalability measured up to 3 nodes

Results

[H. Ryu, Intel® HPC Developer Conference \(2017\)](#)

- Speed-enhancement with HBM becomes larger as a single node takes larger workload
- Inter-node strong scalability is quite nice with no HBM
→ **2.36x speed-up with 3x computing expense** (no HBM)
→ ~78% scalability ($2.36/3$) is what we usually get from multi-core base HPCs (Tachyon-II HPC in KISTI)
→ 1.58x speed-up with HBM (prob. size is not large enough)



Summary

KISTI Intel® Parallel Computing Center



- Introduction to Code Functionality
- Main Numerical Problems and Performance Bottleneck
- Performance (speed and energy consumption) in a single KNL node
- Strong scalability in multiple KNL nodes
- (Appendix) Strategies for offload computing (for PCI-E devices)

Thanks for your attention!!

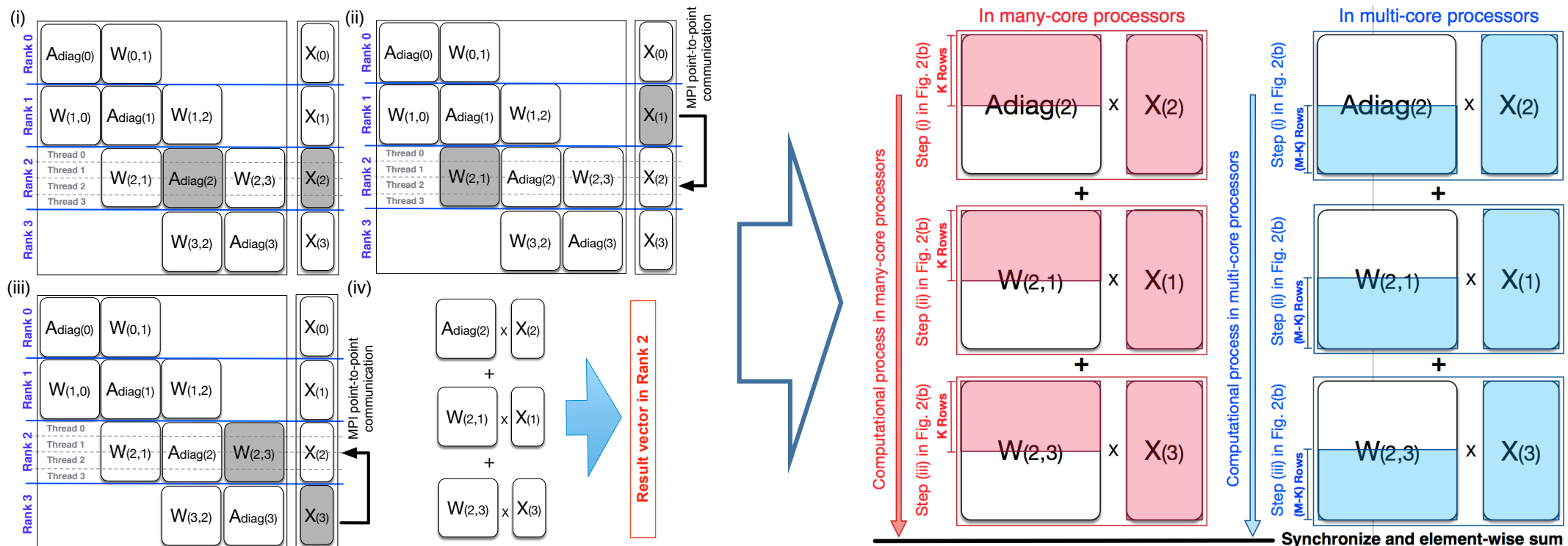
Appendix: Strategy for offload-computing

Asynchronous Offload (for Xeon(V4) + KNC, Xeon(V4) + GPU)

The real bottleneck of computing: Overcome with asynchronous offload

H. Ryu et al., Comp. Phys. Commun. (2016)
(<http://dx.doi.org/10.1016/j.cpc.2016.08.015>)

- 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 is a big deal: Host and PCI-E device shares computing load



Appendix: Strategy for offload-computing

Data-transfer and Kernel Functions for GPU Computing

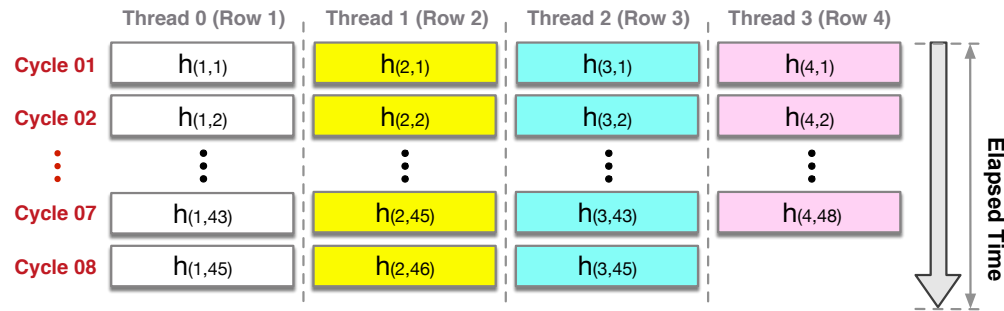
Data-transfer between host and GPU Devices

- 3x increased bandwidth with **pinned memory**
- Overlap of computation and data-transfer with **asynchronous streams**

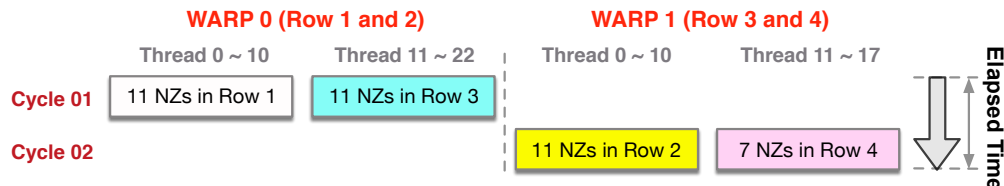
Speed-up of GPU Kernel Function (MVMul)

- Treating several rows at one time with WARPs

Data-Access with a thread-base (no WARPs)

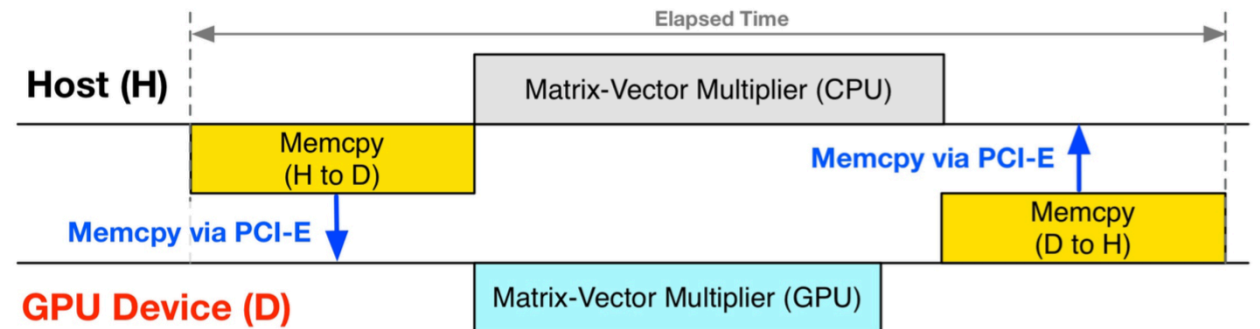


Data-Access with a WARP-base



H. Ryu et al., J. Comp. Elec. (2018)
(<http://dx.doi.org/10.1007/s10825-018-1138-4>)

[Synchronous Data Transfer with Pageable Memory]



[Asynchronous Data Transfer with Pinned Memory]

