Large-scale Electronic Structure Simulations with MVAPICH2 on Intel Knights Landing Manycore 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 (IPCC)

KISTI 한국과학기술정보연구원 Korea Institute of Science and Technology Information www.kisti.re.kr

국가슈퍼컴퓨팅연구소 National Institute of Supercomputing and Networking

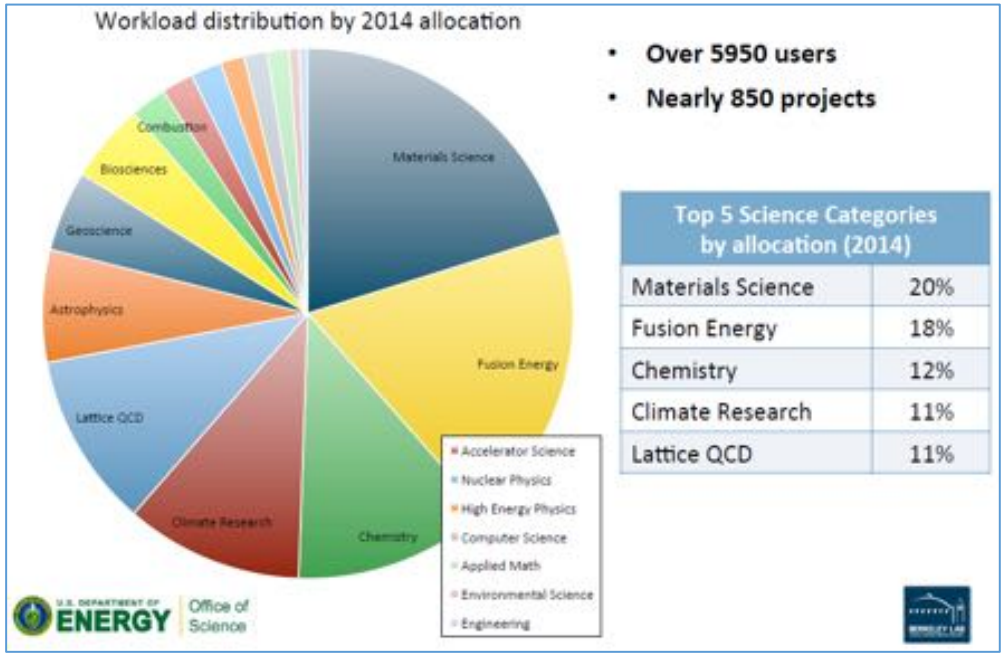
IPCC KISTI

Electronic Structure Calculations

An application that has customers in a a HUGE range

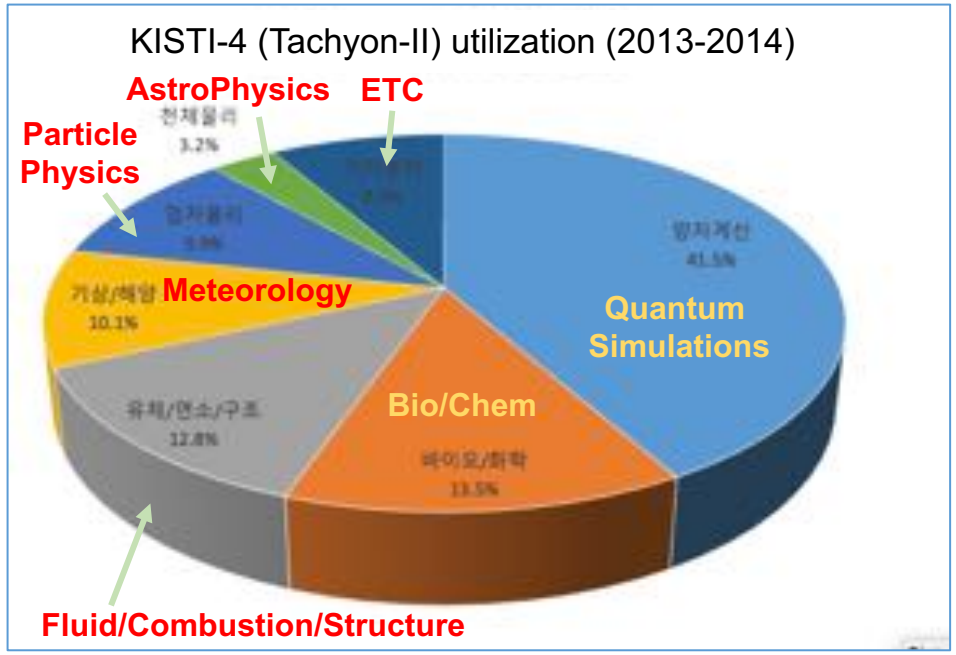
- **<u>The state of motion of electrons</u>** in an electrostatic field created by the stationary nuclei.
- Quantum Physics; Quantum Chemistry; Nanoscale Materials and Devices

  → Physics, Chemistry, Materials Science, Electrical Engineering and Mechanical Engineering ETC.
- Huge customers in the society of computational science



- Over 5950 users
- Nearly 850 projects

Top 5 Science Categories by allocation (2014)	
Materials Science	20%
Fusion Energy	18%
Chemistry	12%
Climate Research	11%
Lattice QCD	11%

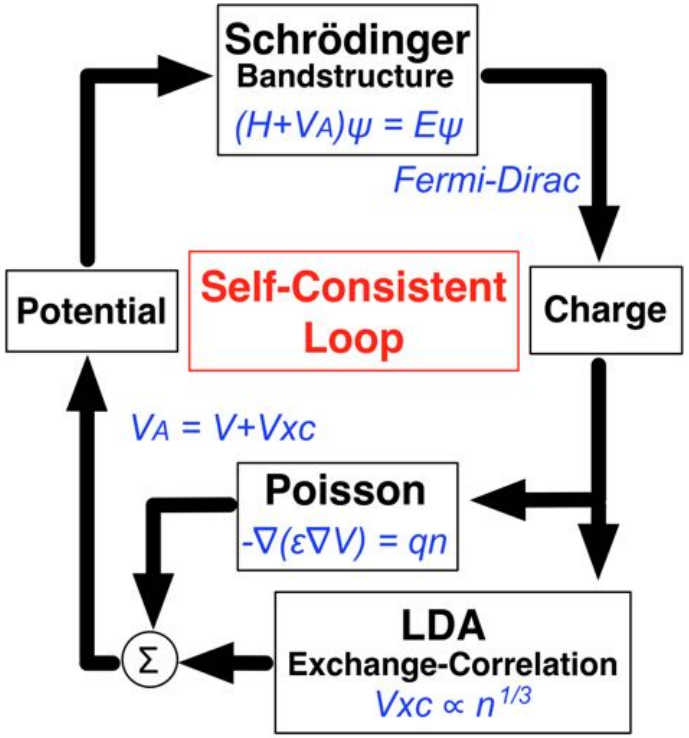


Electronic Structure Calculations

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

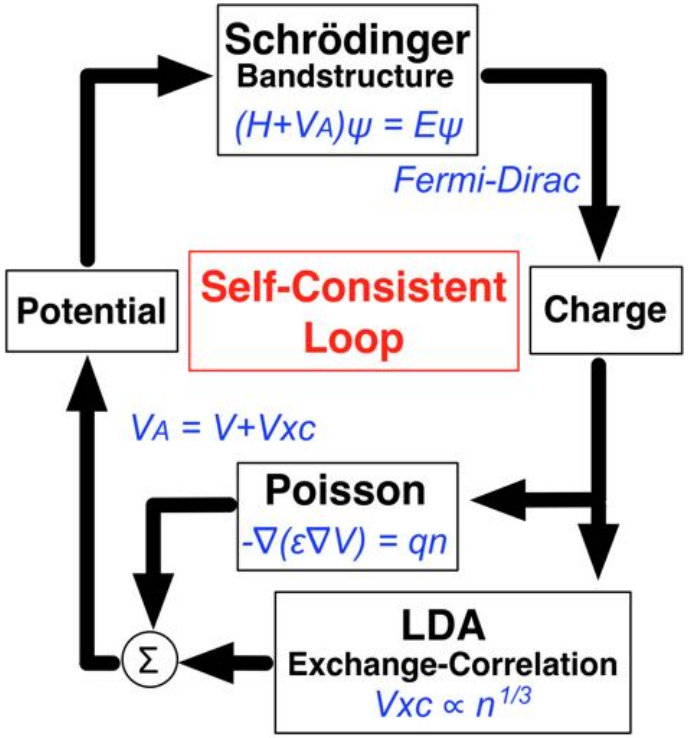
  → **Linear System Problem** $-\nabla(\varepsilon\nabla V) = \rho$ → $Ax = b$

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** $H\Psi = E\Psi$

- Poisson Equations

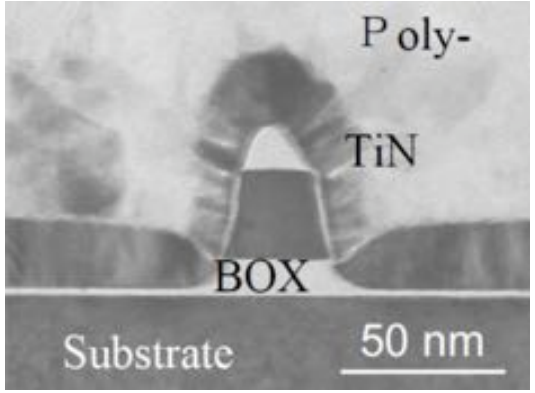
  → **Linear System Problem** $-\nabla(\varepsilon\nabla V) = \rho$ → $Ax = b$

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

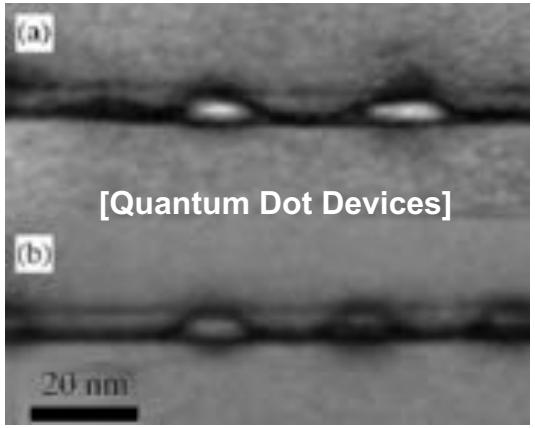
Needs for "Large" Electronic Structures

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)



[Logic Transistors (FinFET)]



[Quantum Dot Devices]

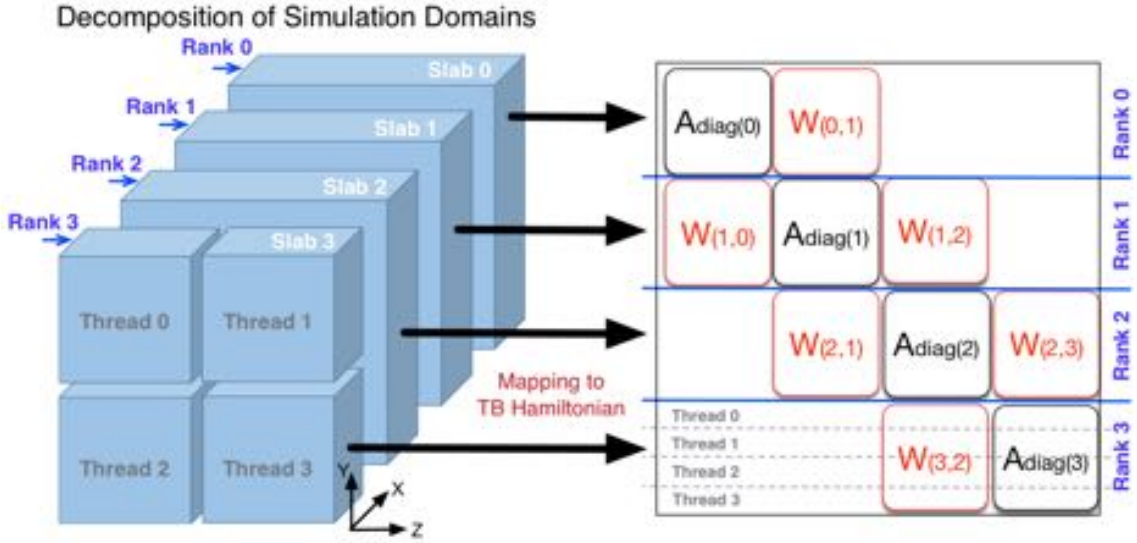
30nm³ Silicon Box? → About a million atoms

2. DOF of Matrices of Governing Equations
→ Linearly proportional to # of atoms (w/ some weight)

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

$$H\Psi = E\Psi$$

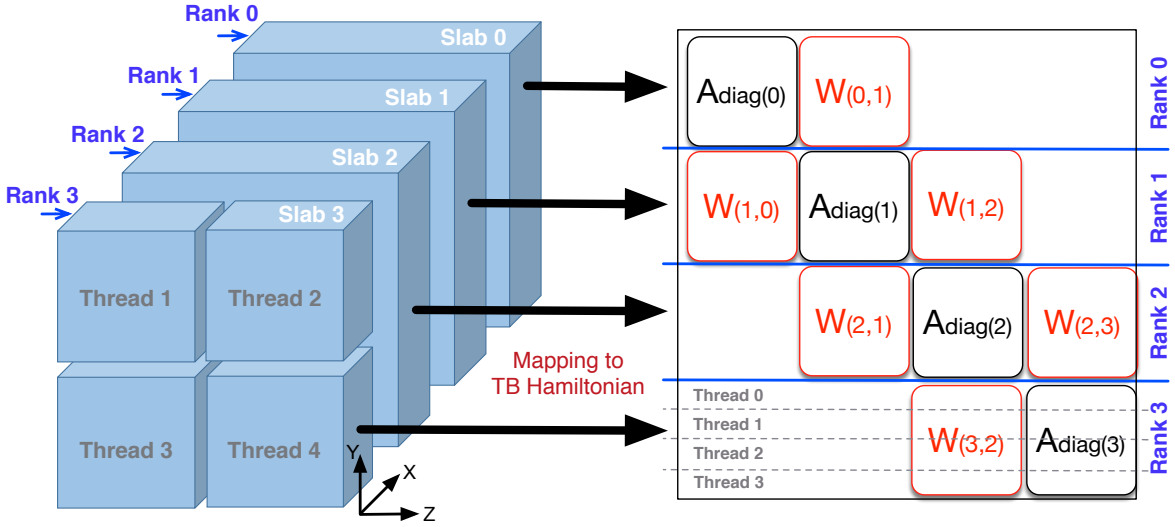
Poisson Equation

- Linear System Problem (Electrostatics: Q-V)
- Poisson matrix is always symmetric

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

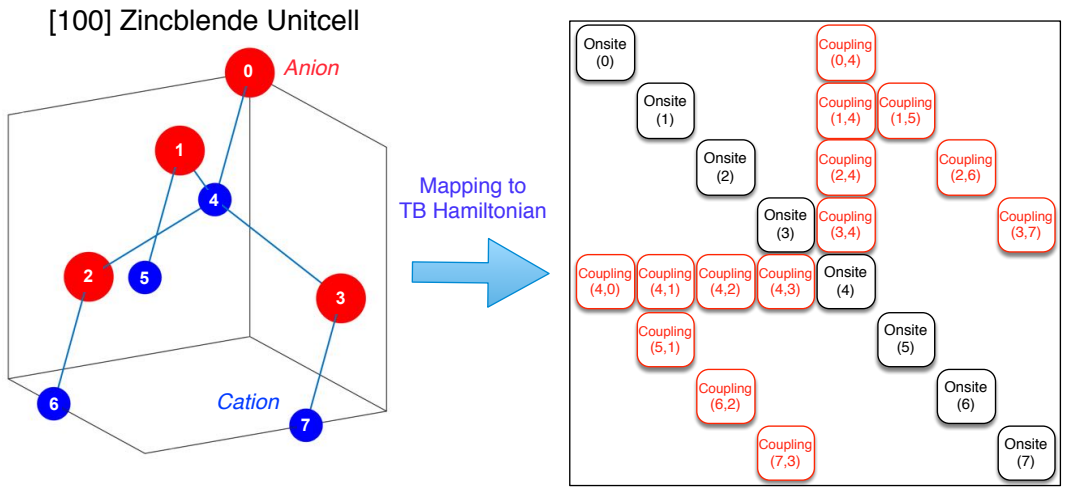
Domain Decomposition

- MPI + OpenMP
- Effectively multi-dimensional decomposition



Matrix Handling

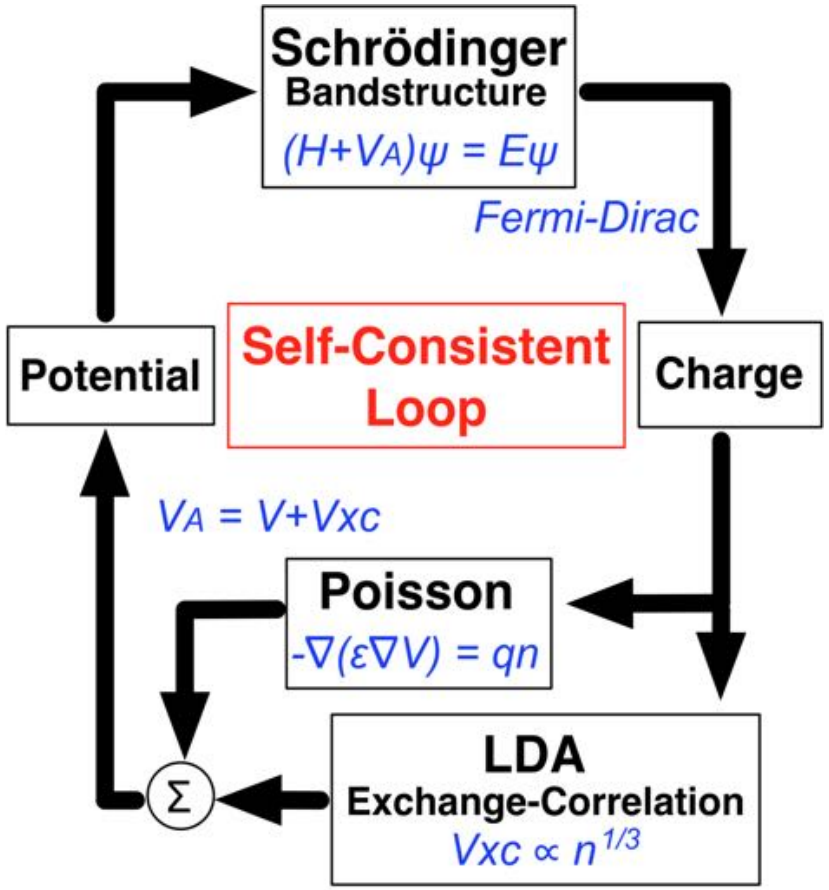
- Tight-binding Hamiltonian (Schrödinger Eq.)
- Finite Difference Method (Poisson Eq.)
- Nearest Neighbor Coupling: Highly Sparse → CSR



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.

$$H\Psi = E\Psi$$

$\mathbf{v}_i$: (Nx1) vectors (i = 0, …, **K**); $\mathbf{a}_i$ and $\mathbf{b}_i$: scalars (i = 1, …, **K**)

$\mathbf{v}_0 \leftarrow \mathbf{0}$, $\mathbf{v}_1$ = random vector with norm 1 ;

$\mathbf{b}_1 \leftarrow \mathbf{0}$;

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

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

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

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

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

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

construct T matrix;

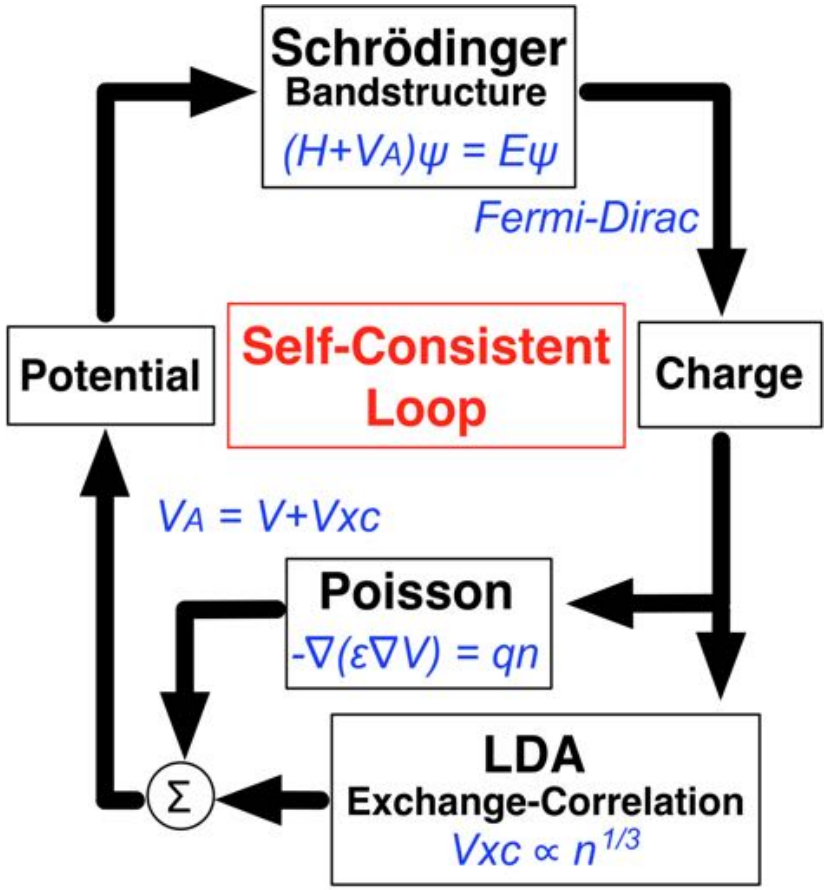
end loop

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

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 Algebraic Ops.

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

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++)

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

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

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

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

declare $\mathbf{r}_{j+1}$ is the solution of $\mathbf{Ax} = \mathbf{b}$ and break the loop

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

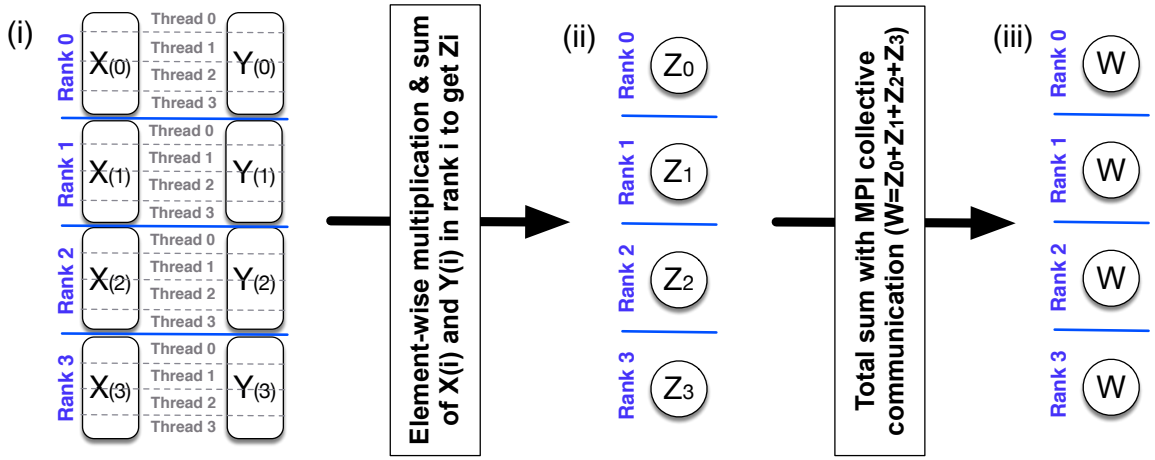
$\mathbf{p}_{j+1} \leftarrow \mathbf{r}_{j+1} + \mathbf{c}_j \mathbf{p}_j$;

end loop

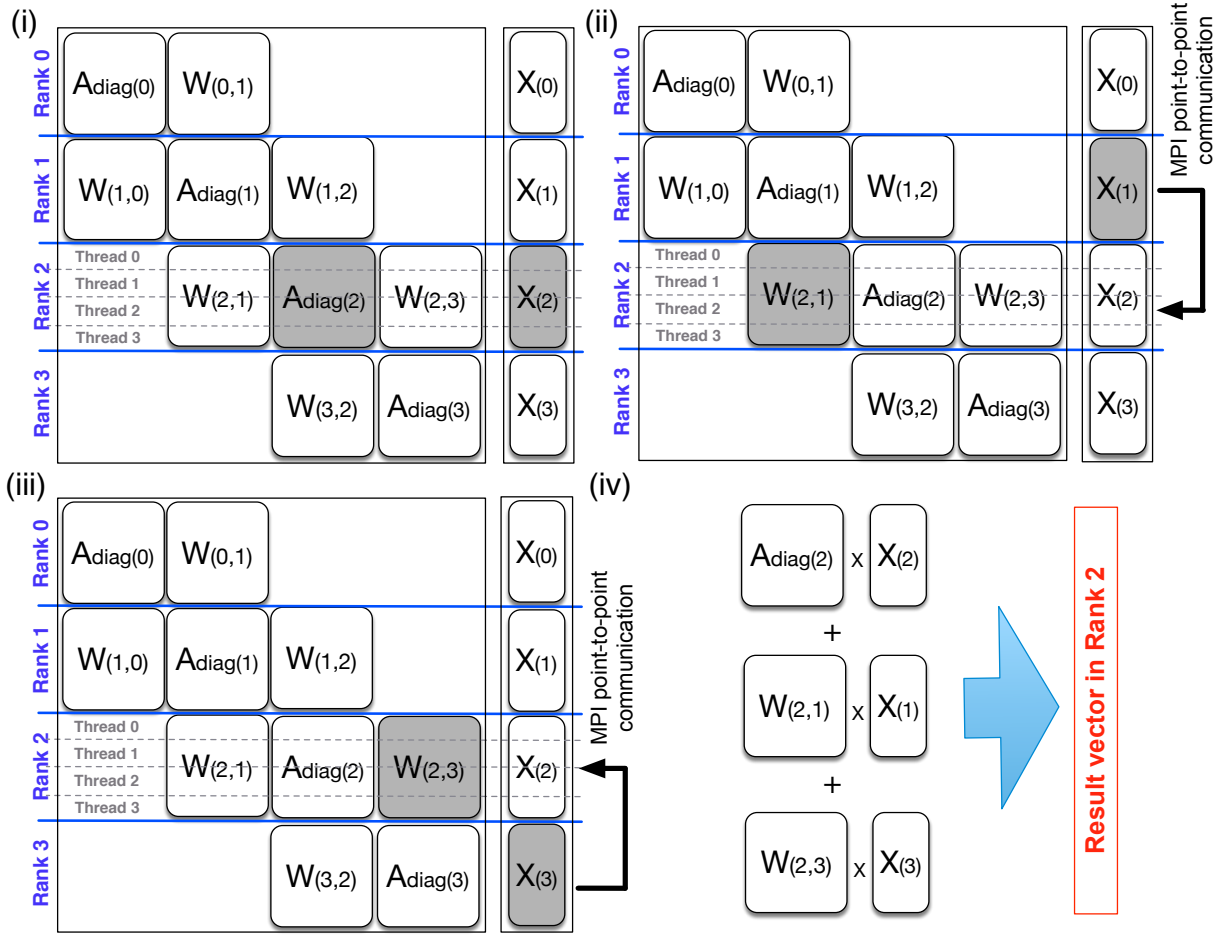
Performance Bottleneck?

Matrix-vector Multiplier: Sparse Matrices

Vector Dot-Product (VVDot)



(Sparse) Matrix-vector Multiplier (MVMul)

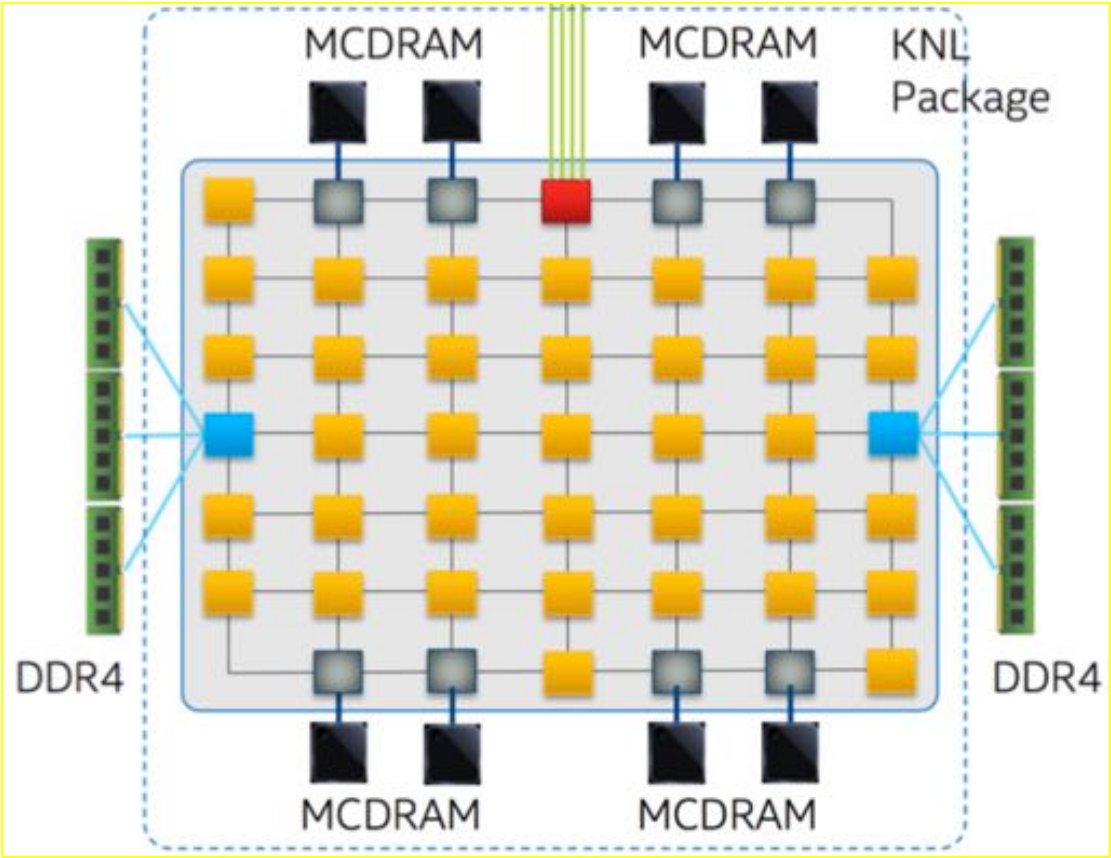


Main Concerns for Performance

- 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

Intel® Knights Landing (KNL) Processors

A Simple Overview of Architecture



A. Sodani, Intel® Xeon Phi™ Processor "Knights Landing" Architectural Overview, ISC (2015)

Summary of Spec & Issues

- Up to 72 CPU cores
  - → 36 Tiles in a 2D mesh (2 AVX-512 VPU per core)
  - → DDR4, 6 Channels
  - → Up to 16GB MCDRAM (HBM) Total
- Self-hosted, No issue in copying time
- Extracting performance means
  - → **Better memory access patterns**
  - → More threads & Vectorization

HBM Modes

- Cache Mode
  - → No source changes needed to use
- Flat mode
  - → MCDRAM mapper to physical address space
  - → Accessed through (memkind) library or numactl
- Hybrid
  - → Combination of the above two

Performance w/ Intel® KNL Processors

Single-node Performance: The power of MCDRAM

Description of BMT Target and Test Mode

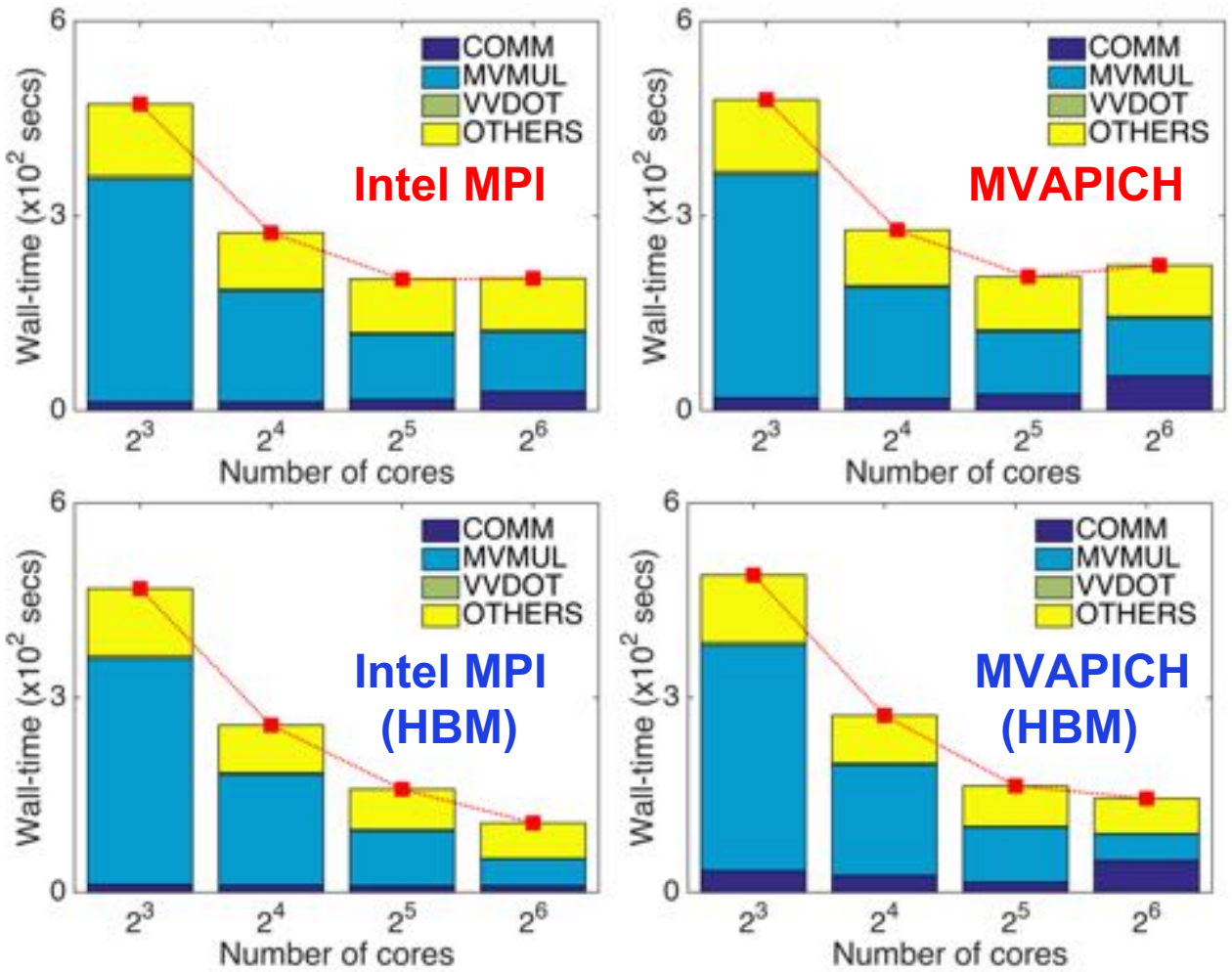
- 36x12x12(nm^3) [100] Si:P quantum dot
  → Material candidates for Si Quantum Info. Processors (Nature Nanotech. **9**, 430)
  → 2.1Mx2.1M Hamiltonian Matrix
- KNL (Xeon Phi 7210); Up to 256 (64x4) cores
- MCDRAM control w/ **numactrl**; Quad Mode; **MPI only**
- Intel Parallel Studio 2017 (Intel MPI)
- MVAPICH2-2.2 with Intel Composer XE 2017 (MVAPICH)

Results

- With no MCDRAM
  → No clear speed-up beyond 32 cores
- **With MCDRAM**
  → Full scalability up to 64 hardware cores

Points of Questions

- How is the performance of MVAPICH2 w.r.t. Intel?
  → Spot of comparison: Communications



Performance w/ Intel® KNL Processors

Single-node Performance: MVAPICH vs. Intel MPI

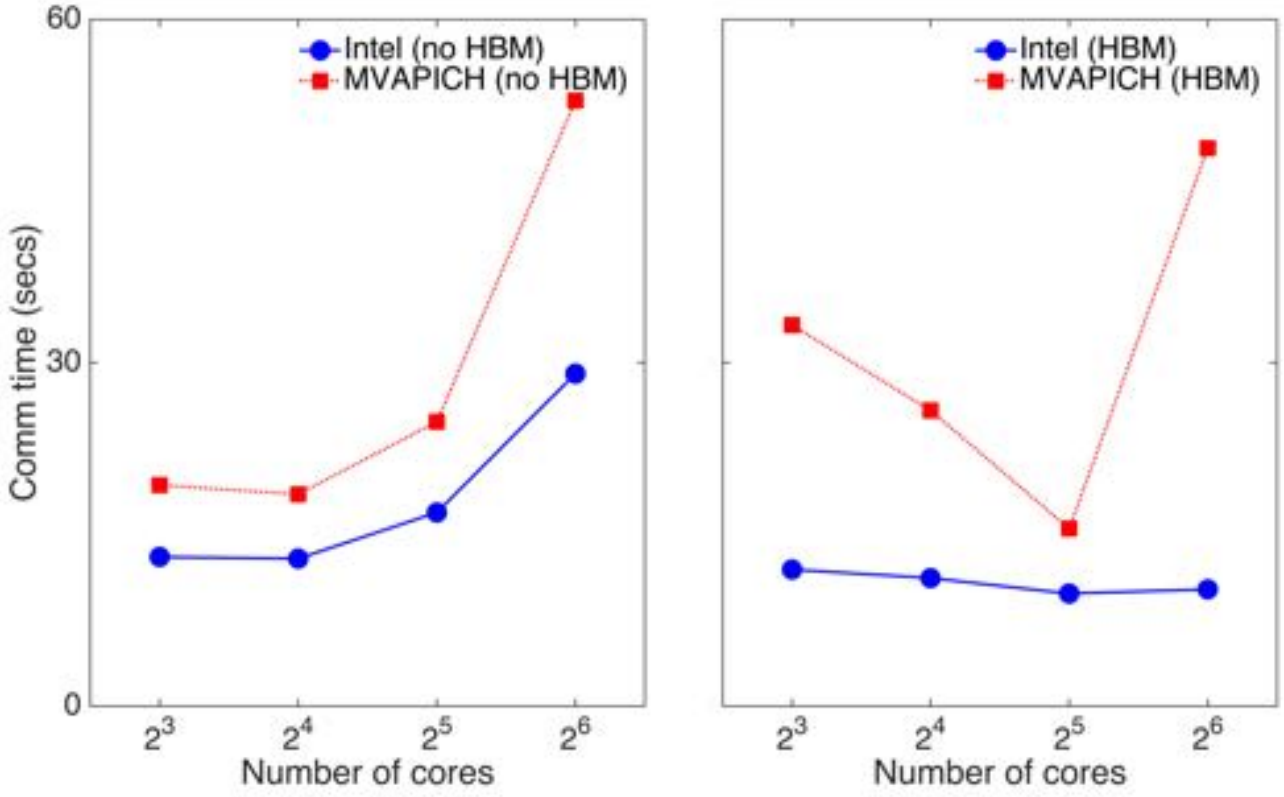
Description of BMT Target and Test Mode

- 36x12x12(nm^3) [100] Si:P quantum dot
  → Material candidates for Si Quantum Info. Processors (Nature Nanotech. **9**, 430)
  → 2.01Mx2.1M Hamiltonian Matrix
- KNL (Xeon Phi 7210); Up to 256 (64x4) cores
- MCDRAM control w/ **numactrl**; Quad Mode; **MPI only**
- Intel Parallel Studio 2017 (Intel MPI)
- MVAPICH2-2.2 with Intel Composer XE 2017 (MVAPICH)

Points of Questions

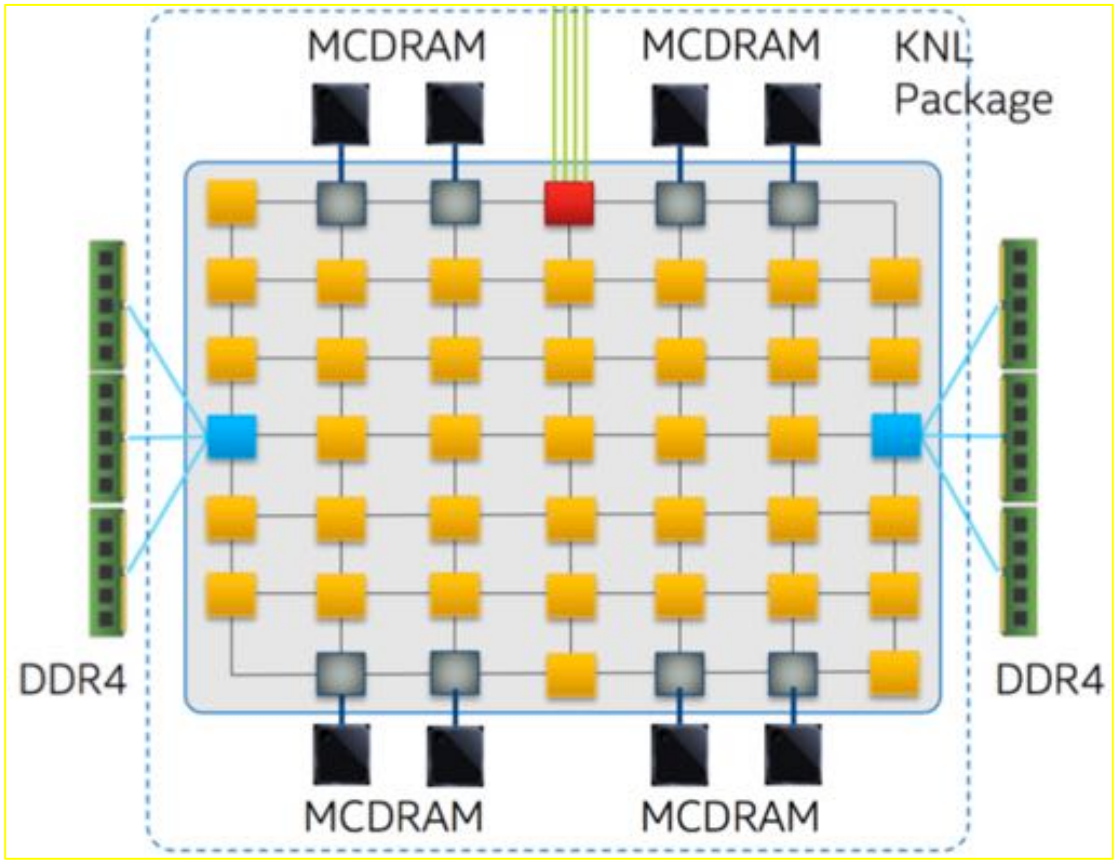
- How is the performance of MVAPICH2 w.r.t. Intel?
  → Where is the spot for comparison in this simple
- **A focus on communication time**
  → MVAPICH2 clearly shows longer communication times
  → MPI_Send/Recv + Allreduce function calls
- **Why?**
  → The reason may be related to the resource allocation

Communication Time



Performance w/ Intel® KNL Processors

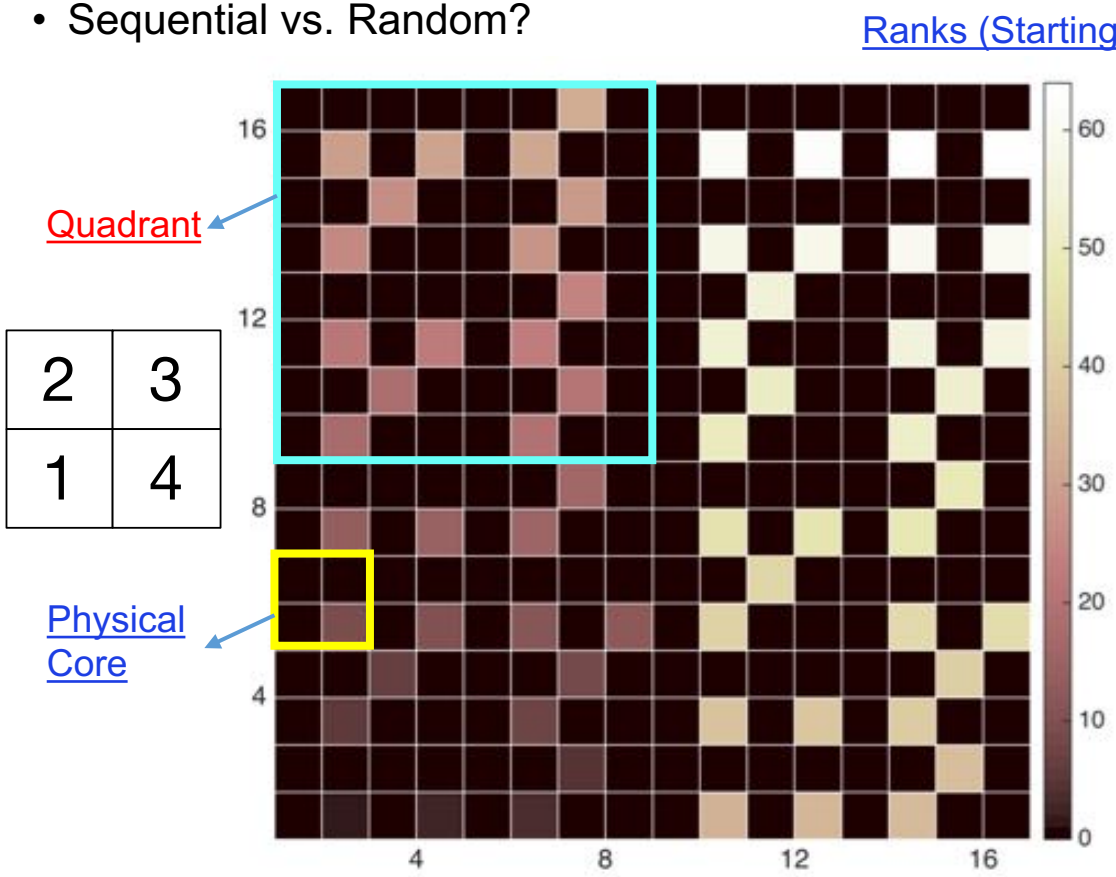
MVAPICH vs. Intel MPI: CPU allocation to Rank ID's



A. Sodani, Intel® Xeon Phi™ Processor "Knights Landing" Architectural Overview, ISC (2015)

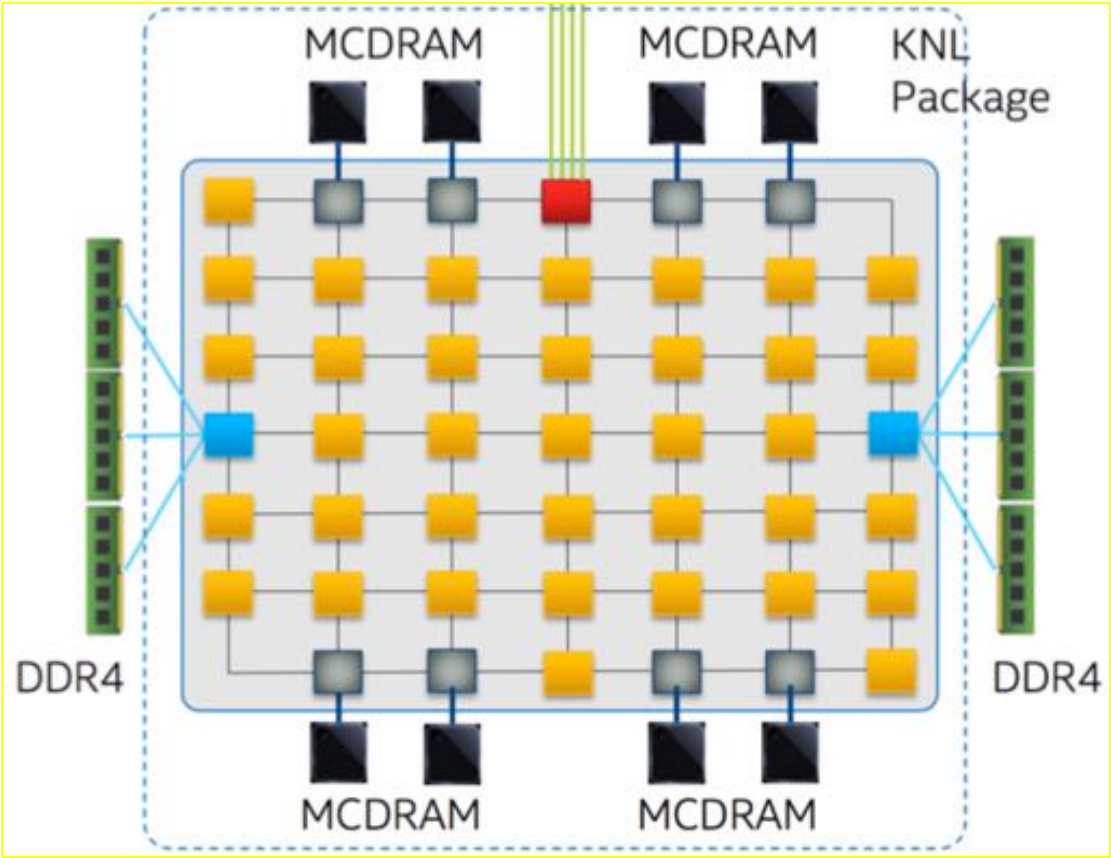
How are physical cores allocated to MPI processes?

- Cluster set to quadrant mode
- Sequential vs. Random?



Performance w/ Intel® KNL Processors

MVAPICH vs. Intel MPI: CPU allocation to Rank ID's



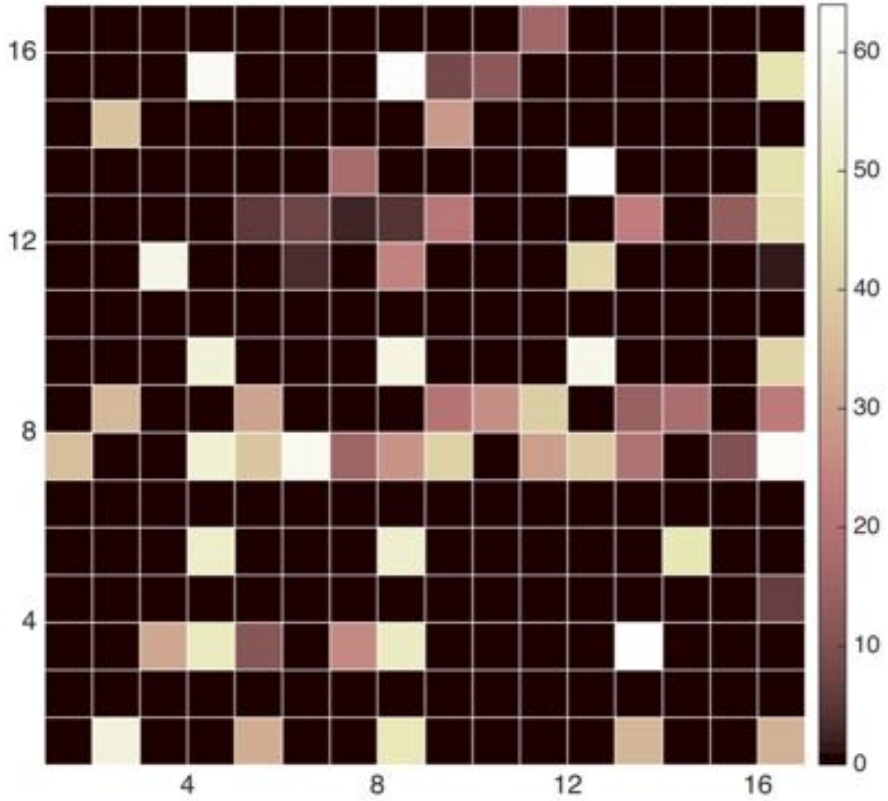
A. Sodani, Intel® Xeon Phi™ Processor "Knights Landing" Architectural Overview, ISC (2015)

How are physical cores allocated to MPI processes?

- Cluster set to quadrant mode
- Sequential vs. Random?

2	3
1	4

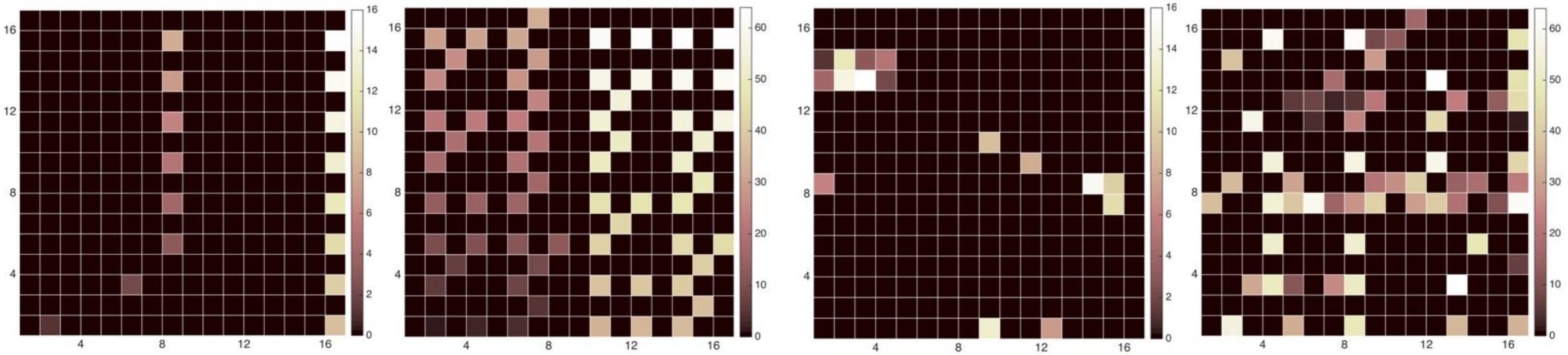
Ranks (Starting w/ 1)



Performance w/ Intel® KNL Processors

MVAPICH vs. Intel MPI: CPU allocation to Rank ID's

16 MPI | 64 MPI | 16 MPI | 64 MPI



← Intel MPI MVAPICH2 →

What about in terms of computing time?

- 1 physical core has 4 hardware threads
- If 1 physical core is mapped by multiple MPI processes, computing time would get less benefits

2	3
1	4

<u>Best affirnity for this workload: "clock-wise" and "inter-core" allocations of MPI processes</u>

Summary and Acknowledgements

Electronic Structure Simulations w/ Intel MPI and MVAPICH2

- Introduction to Code Functionality
- Main Numerical Problems and Performance Bottleneck
- Performance (speed and energy consumption) in a single KNL node
- Intel MPI vs MVAPICH 2: affinity of MPI processes

  → w/ initial analysis of its potential effects on performance

Members of KISTI Intel® Parallel Computing Center

Yosang Jeong

Seungmin Lee

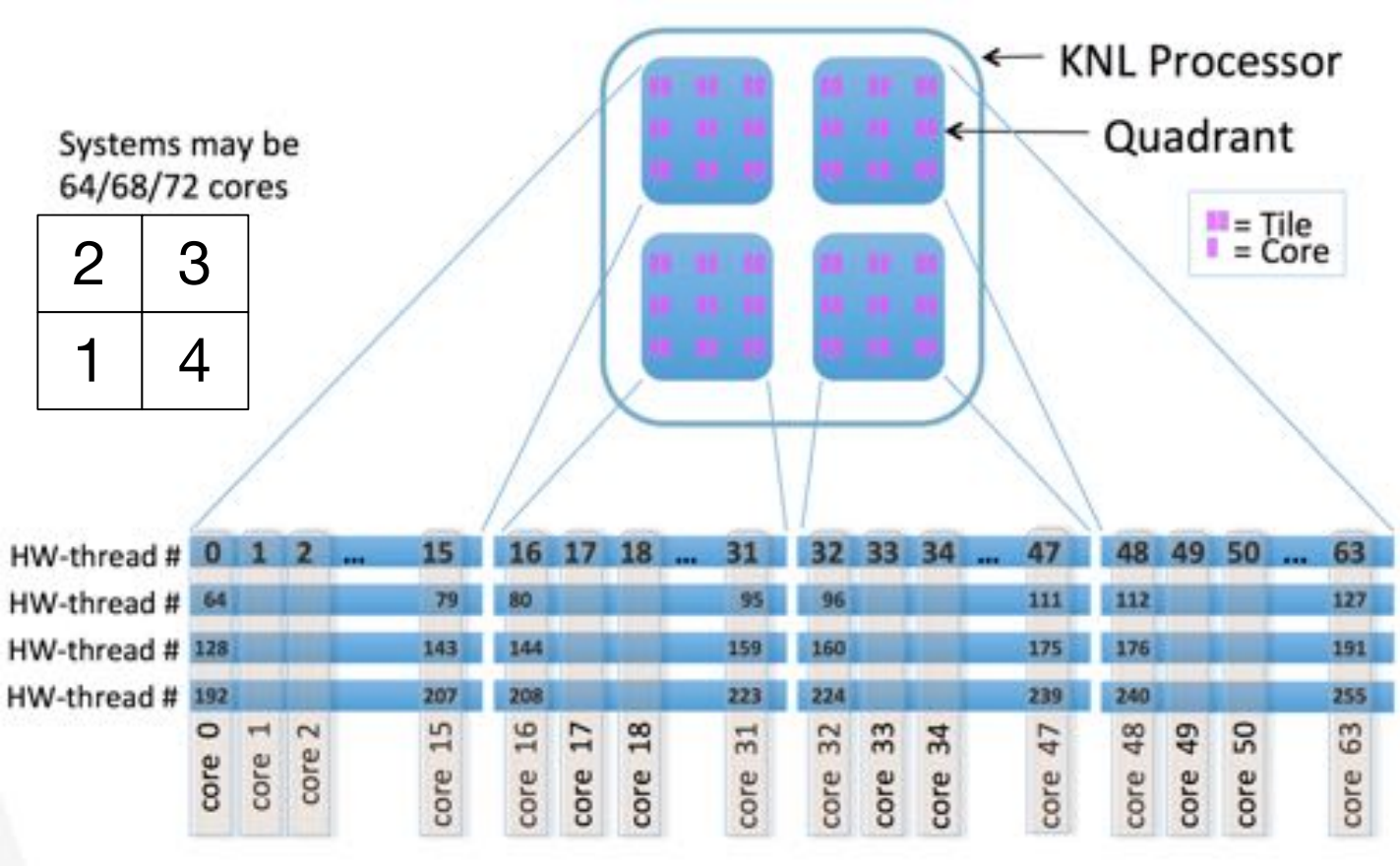
Geunchul Park

Dukyun Nam

Hoon Ryu

Mapping MPI processes to CPUs in KNL

Electronic Structure Simulations w/ Intel MPI and MVAPICH2



J. Cazes et al., Programming Intel's 2nd Generation Xeon Phi, IXPUG 2016 KNL Tutorial, ISC (2016)

Strategy

- We know the id's of cpus in each quadrant
  → (e.g) CPU 0~15 is in the 2nd quadrant
- Need approximations to describe the cpu map in a single quadrant
  → it is not clear which cpu has which id.
- The following approximation would not hurt the generality of the issue.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Quad 1

48	49	50	51
52	53	54	55
56	57	58	59
60	61	62	63

Quad 4

Performance w/ Intel® KNL Processors

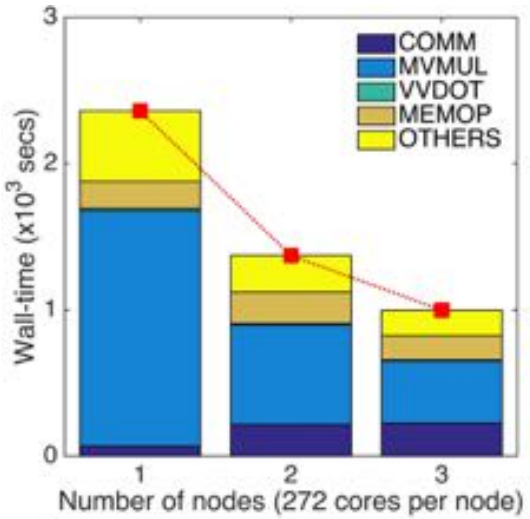
Multi-node Performance: Scalability

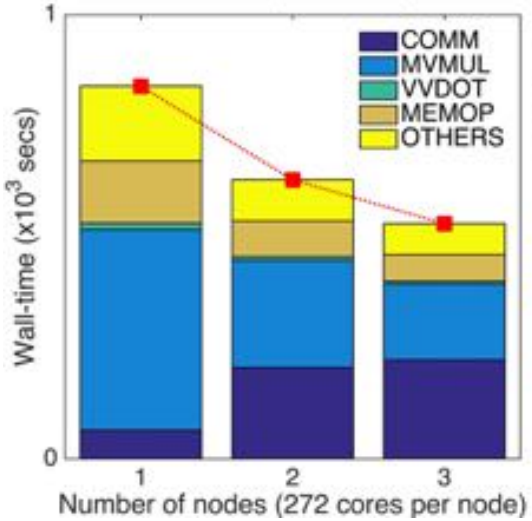
Description of BMT Target and Test Mode

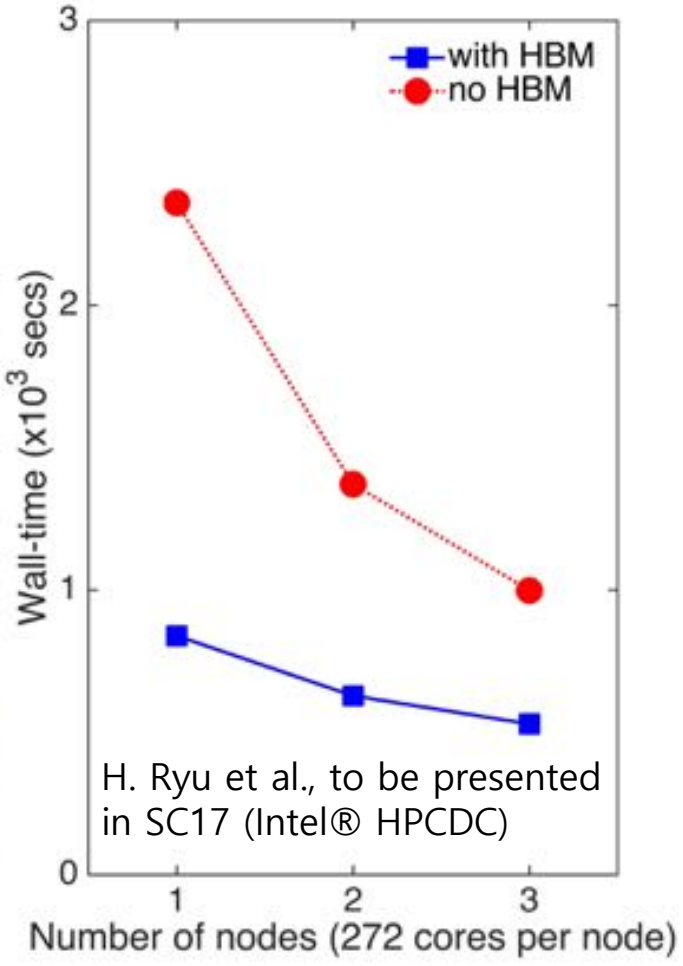
- 5 CB states in 27x33x33(nm^3) [100] Si:P quantum dot
  → Material candidates for Si Quantum Info. Processors (Nature Nanotech. **9**, 430)
  → 14.4Mx14.4M Hamiltonian Matrix
- KNL (Xeon Phi 7250) nodes; Up to 272 (68x4) cores/node
  → (4 MPI processes + 68 threads) per node
  → Quad / Flat mode, No OPA (10G network)
  → Strong scalability measured up to 3 nodes

Results

- 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)







H. Ryu et al., to be presented in SC17 (Intel® HPCDC)