Performance evaluation of scalable optoelectronics application on large-scale Knights Landing cluster

Yuta Hirokawa

Graduate School of Systems and Information Engineering, University of Tsukuba

hirokawa@hpcs.cs.tsukuba.ac.jp

Agenda

- Background
- ARTED and SALMON: Electron Dynamics Simulation with TDDFT
- Porting to Knights Landing
- Performance comparison of KNC and KNL
  - Single node performance: stencil computation (dominant part)
  - Multi node performance: time-development part (entire performance)
- Performance evaluation under full-system of Oakforest-PACS
- Summary

Background

- Intel Xeon Phi (a.k.a. MIC, Many Integrated Core Architecture)
  - *Current* architecture: Knights Landing (KNL)
  - *Previous* architecture: Knights Corner (KNC)
- Many-core system actively operated in the world
  - Intel Xeon Phi (KNL): Cori (NERSC, USA), Oakforest-PACS (JCAHPC, Japan)
  - Proprietary processor: Sunway Taihulight (NSCC, China)
- Performance gain of real scientific applications on many-core processor is very difficult
  - *Optimization* is strongly requested for the applications on many-core processors

Purpose of this study

- Recent supercomputers provide standalone many-core-based clusters
  - To replace ordinary commodity CPU clusters (Recent Xeon is a many-core CPU)
  - WHY: Achieve a high watt performance, Keep a lid on building costs...
- Our code has been optimized particularly for KNC many-core
  - Y. Hirokawa, *et. al.*: "Electron Dynamics Simulation with Time-Dependent Density Functional Theory on Large Scale Symmetric Mode Xeon Phi Cluster", *PDSEC16*
- We implement and evaluate the application to KNL based on the optimized code for KNC
  - The optimization for KNC strongly helps with migration to KNL clusters

ARTED and SALMON

Electron Dynamics Simulation with TDDFT

ARTED: **A**b-initio **R**eal-**T**ime **E**lectron **D**ynamics simulator
SALMON: **S**calable **A**b-initio **L**ight-**M**atter simulator for **O**ptics and **N**anoscience

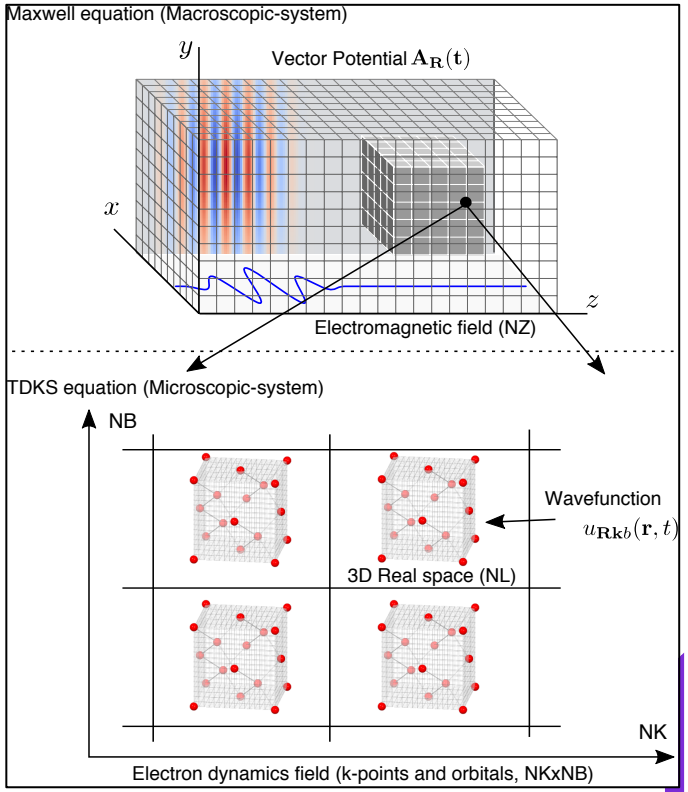
- Electron dynamics simulation with Time-Dependent DFT
  - Developed at Center for Computational Sciences (CCS), U. Tsukuba
  - Collaborative research with CCS (since Dec. 2014, in the Master program)
  - Available: K-computer, FX100, Xeon CPU, Xeon Phi, GPU (OpenACC + CUDA)
  - 25-points stencil computation should be very optimized
  - Written by Fortran (2003) + C (for hand-coding vectorization), MPI+OpenMP

- Currently, we develop an advanced scientific application based on ARTED, names *SALMON*: http://salmon-tddft.jp/
  - Open-source software: it is developed at GitHub
  - Supported by JST-CREST and Post-K priority issue (7)
  - Please contact our mailing-lists: salmon-users@salmon-tddft.jp

Overview

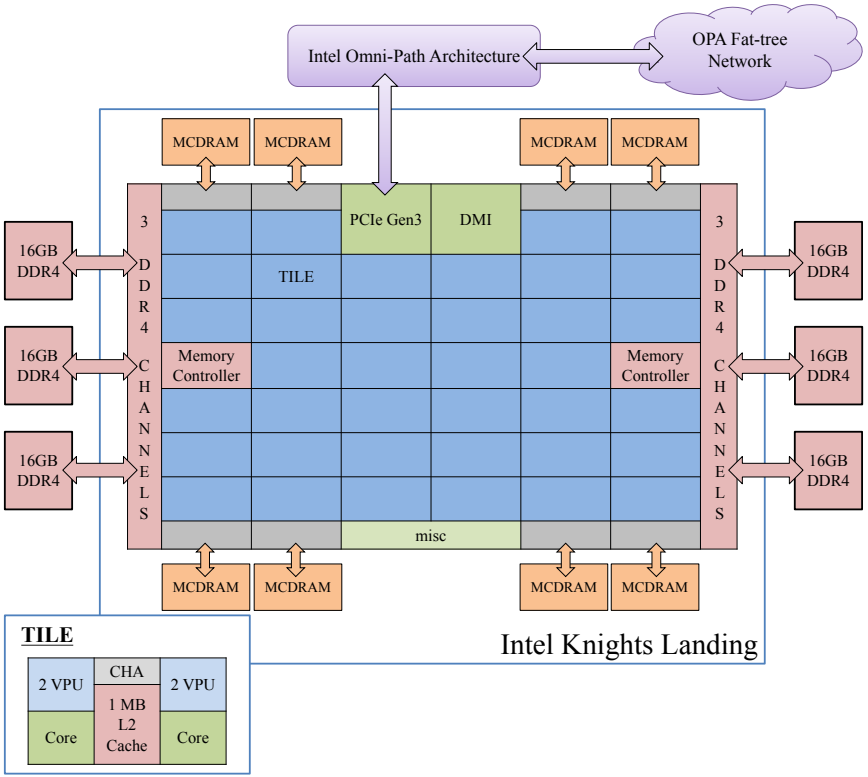
[1] Y. Hirokawa, *et al.*: "Performance Evaluation of Large Scale Electron Dynamics Simulation under Many-core Cluster based on Knights Landing", *HPCAsia2018*, Tokyo. OA: https://doi.org/10.1145/3149457.3149465

- First order quantum calculation based simulation
  - Electromagnetic (Maxwell) + Electron dynamics (Time-Dependent Kohn-Sham)
  - Huge wave-space, Small real-space 3-D grid
  - The communication is negligible (not bottleneck)
  - Please refer to the our paper for the simulation and implementation details [1]
- 25-points stencil for the 3-D domain is dominant
  - Periodic boundary condition, 158 FLOPS / grid
  - The stencil computation is not decomposition
  - Each thread computes a 3-D domain with sequentially
  - Single-thread/core level optimization problem



Porting to Knights Landing

Oakforest-PACS (OFP) at JCAHPC (U. Tsukuba and U. Tokyo)



Japan's 2nd supercomputer system (TOP500 Nov. 2017)

Total 8208 compute nodes
Use up to 8192 nodes (Full system)
Peak: 24.91 PFLOPS
HPL: 13.55 PFLOPS (54.4 %)
HPCG: 0.3855 PFLOPS (1.54%)

Fundamental tuning for stencil computation

```
real(8),   intent(in)  :: B(0:NLz-1,0:NLy-1,0:NLx-1)
complex(8),intent(in)  :: E(0:NLz-1,0:NLy-1,0:NLx-1)
complex(8),intent(out) :: F(0:NLz-1,0:NLy-1,0:NLx-1)

#define IDX(dt) iz,iy,modx(ix+(dt)+NLx)
#define IDY(dt) iz,mody(iy+(dt)+NLy),ix
#define IDZ(dt) modz(iz+(dt)+NLz),iy,ix

do ix=0,NLx-1
do iy=0,NLy-1
!dir$ vector nontemporal(F)
do iz=0,NLz-1
  v=0; w=0
  ! z-computation
  v=v+Cz(1)*(E(IDZ(1))+E(IDZ(-1))) ...
  w=w+Dz(1)*(E(IDZ(1))-E(IDZ(-1))) ...
  ! y-computation
  ! x-computation
  F(iz,iy,ix) = B(iz,iy,ix)*E(iz,iy,ix) &
  &           + A          *E(iz,iy,ix) &
  &           - 0.5d0*v - zI*w
end do
end do
end do
```

Index calculate every time with remainder table

Remainder op. is very slow

Use non-temporal store (Intel Compiler only)

Calculation order awareness of continuous address on memory

[C-language]

Explicit (hand-coding) vectorization with 512-bit SIMD on KNC

Single processor performance: KNC vs KNL

Y. Hirokawa, *et. al.*: "Electron Dynamics Simulation with Time-Dependent Density Functional Theory on Large Scale Symmetric Mode Xeon Phi Cluster", *PDSEC16*, 2016

For AVX-512 processors

- We demand the conversion of KNC SIMD code into AVX-512
  - Preprocessor directive can be solved a minor difference of instructions

```
#ifdef __AVX512F__
/* Intrinsics for KNL and AVX-512 processors */
#define _mm512_loadu_epi32   _mm512_loadu_si512
#define _mm512_storenrngo_pd _mm512_stream_pd
#elif __MIC__
/* Intrinsics for KNC */
inline __m512i _mm512_loadu_epi32(int const* v)
{
  __m512i w = _mm512_loadunpacklo_epi32(w, v + 0);
  return      _mm512_loadunpackhi_epi32(w, v + 16);
}
#endif
```

- Our code requires a common AVX-512F subset only (F: Foundation)
  - Our implementation can be applied to all AVX-512 processor families

- For the performance evaluation
  - KNL uses a MCDRAM only for computation

Performance comparison of KNC and KNL

Evaluation environment

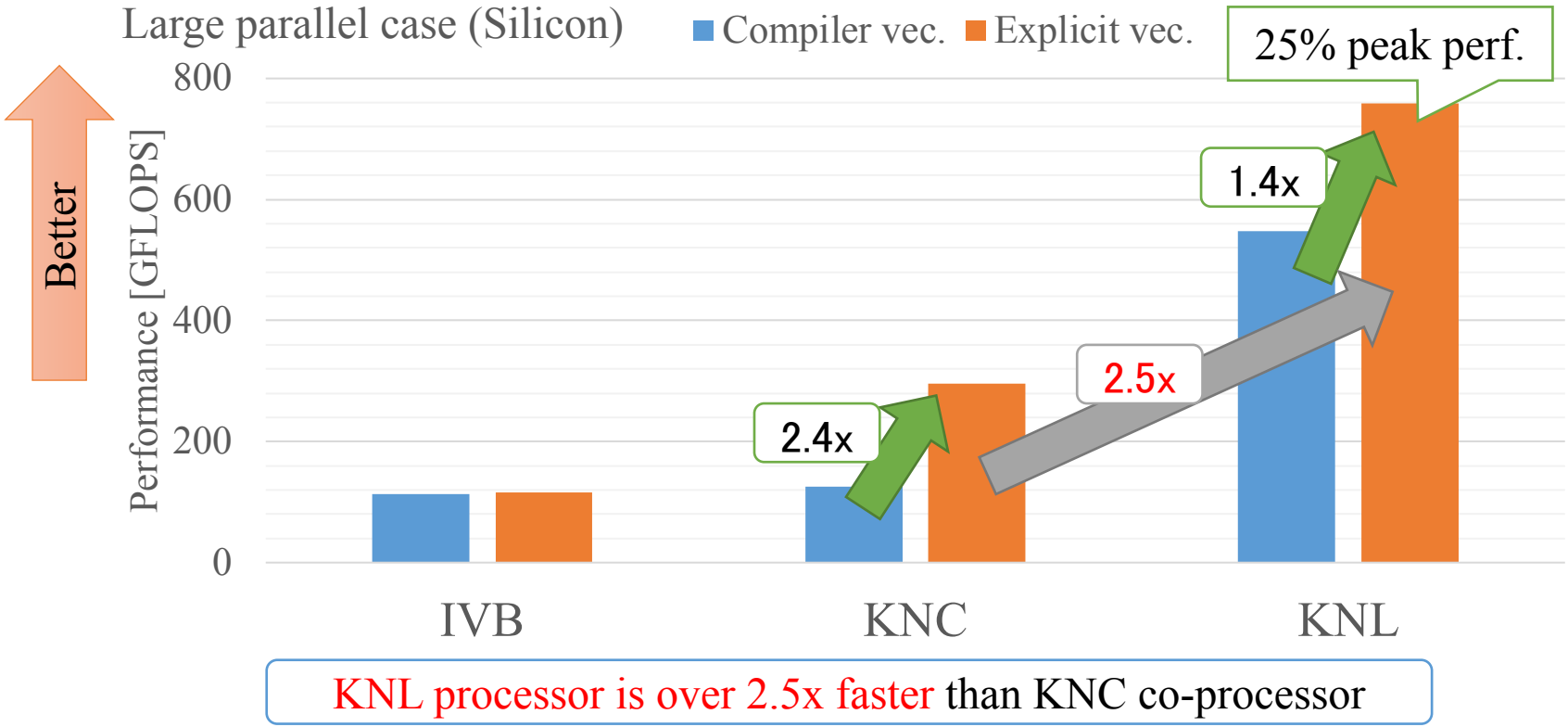
1-MPI process attached to each processor

4-threads per core is fast on the Intel Xeon Phi

	COMA (U. Tsukuba)	Oakforest-PACS (JCAHPC)
# of Node	393	8208
CPU	Intel E5-2670v2 (Ivy-Bridge)	Intel Xeon Phi 7250 (KNL)
	Xeon Phi 7110P (KNC)	
# of Cores / Node	20 (10 cores x2, IVB) + 120 (60 cores x2, KNC)	68 cores (Quadrant) We use 64 cores (4 cores are used with OS)
Memory / Node	64 GB (IVB, DDR3) + 8 GB x2 (KNC, GDDR5)	16 GB (MCDRAM) + 96 GB (DDR4)
Interconnect	Mellanox InfiniBand FDR Connect-X3	Intel Omni-Path Architecture
Compiler and MPI	Intel 16.0.2 and Intel MPI 5.1.3	Intel 17.0.1 and Intel MPI 2017 update 1
Peak Perf. / Node	2548 GFLOPS (400 GFLOPS CPU + 2148 GFLOPS KNC)	3046 GFLOPS

Stencil computation

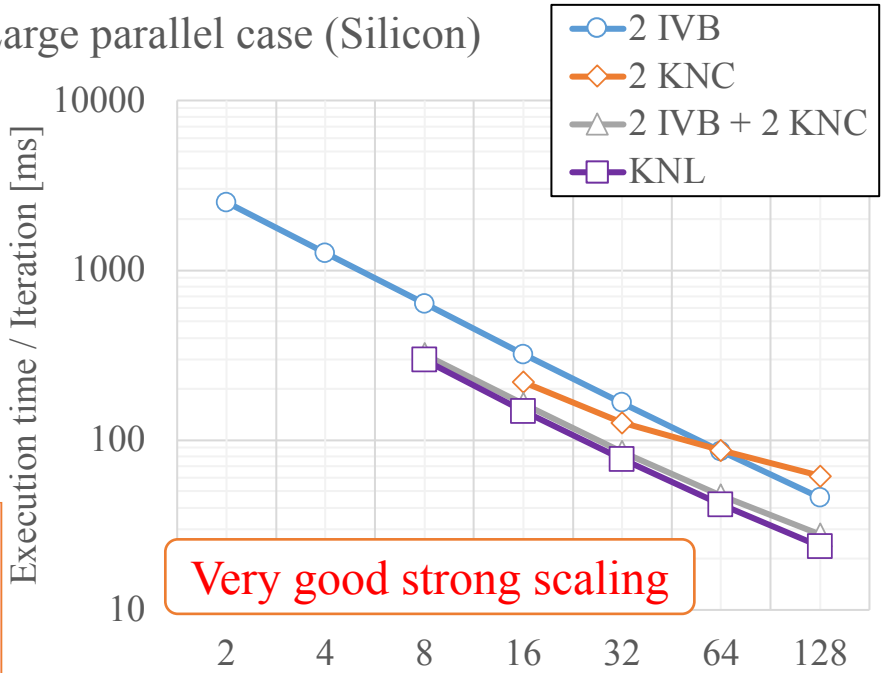
Single processor performance



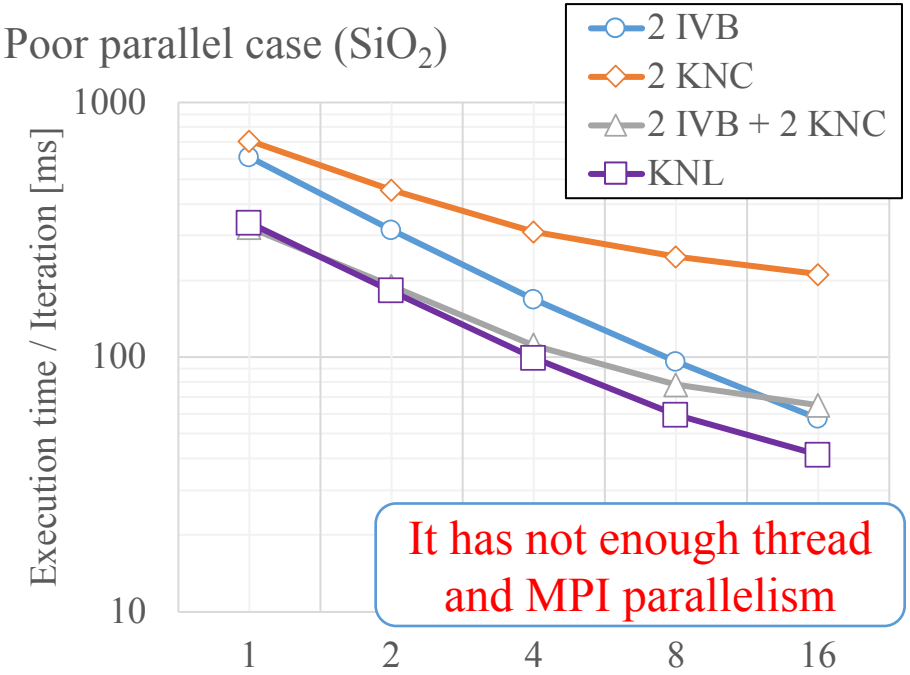
KNL processor is over 2.5x faster than KNC co-processor

Entire computation

Large parallel case (Silicon)



Poor parallel case (SiO_2)



Faster

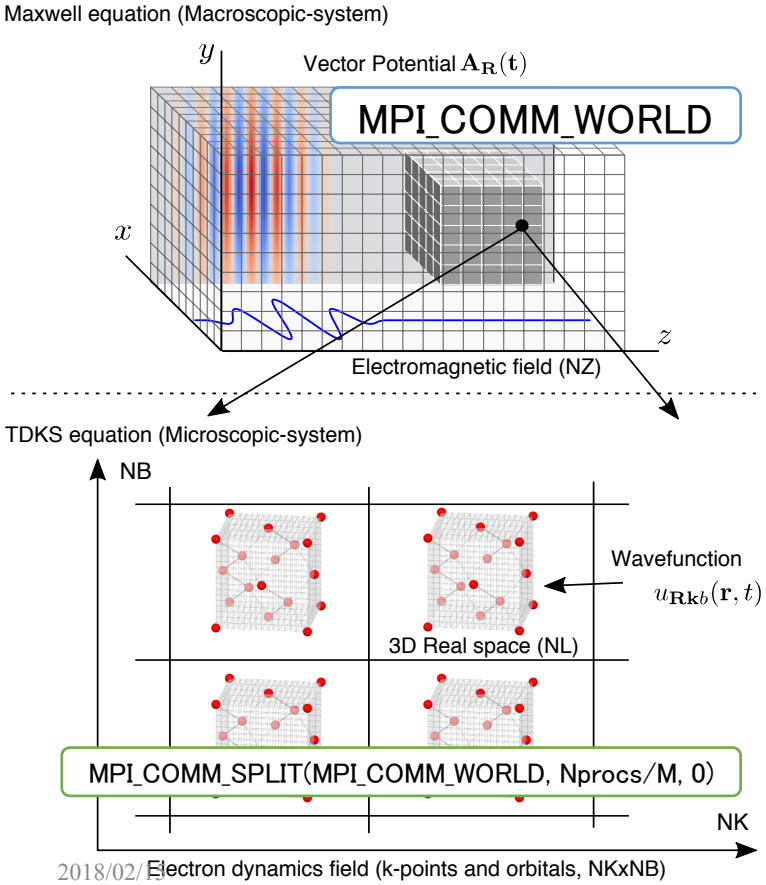
# of compute nodes

# of compute nodes

	Peak Perf. [GFLOPS]	Actual Memory Bandwidth [GB/s]
Oakforest-PACS	3046	486.99 (64 cores)
COMA	400 + 2148 = 2548	46.55×2 + 171.73×2 = 436.56

Performance evaluation under full-system of Oakforest-PACS

Distributed multi-scale simulation



SALMON takes two MPI distribution scheme:

1. Simple distribution
   - Distribute Maxwell equation only
   - Use MPI_COMM_WORLD for Maxwell eq.
   - TDKS (Time-Dependent Kohn-Sham) eq. does not distributed
   - Entire collective only

2. Two-phase distribution
   - Distribute Maxwell&TDKS equations
   - Use MPI_COMM_WORLD for Maxwell eq.
   - Use sub MPI communicator for TDKS eq.
   - Entire collective + Subgroup collective

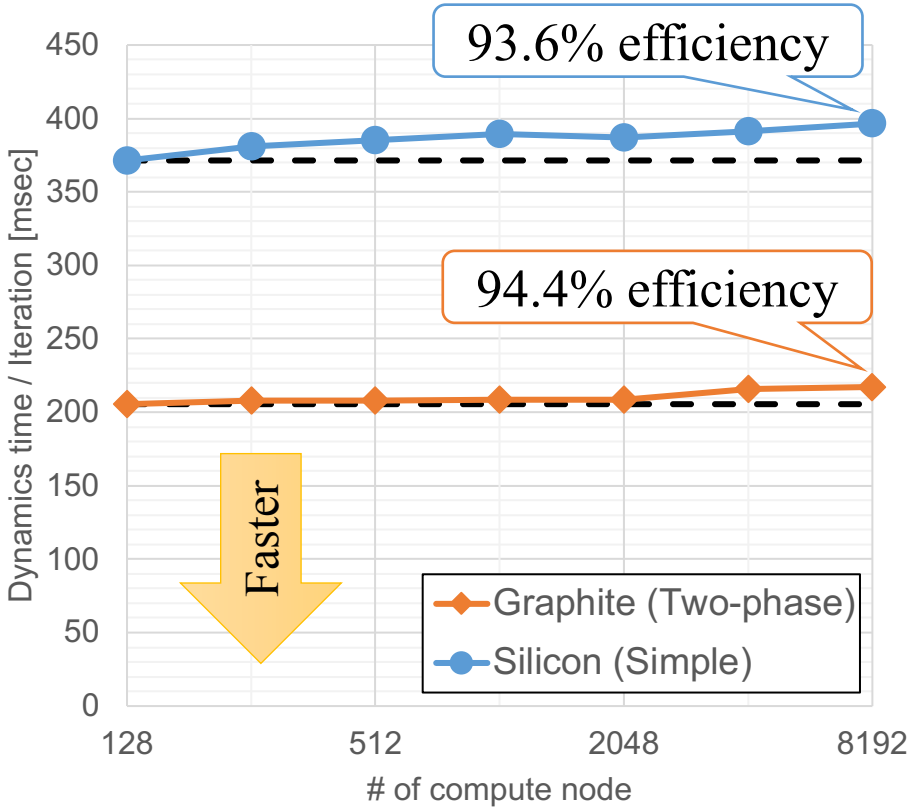
Halo comm. does not required

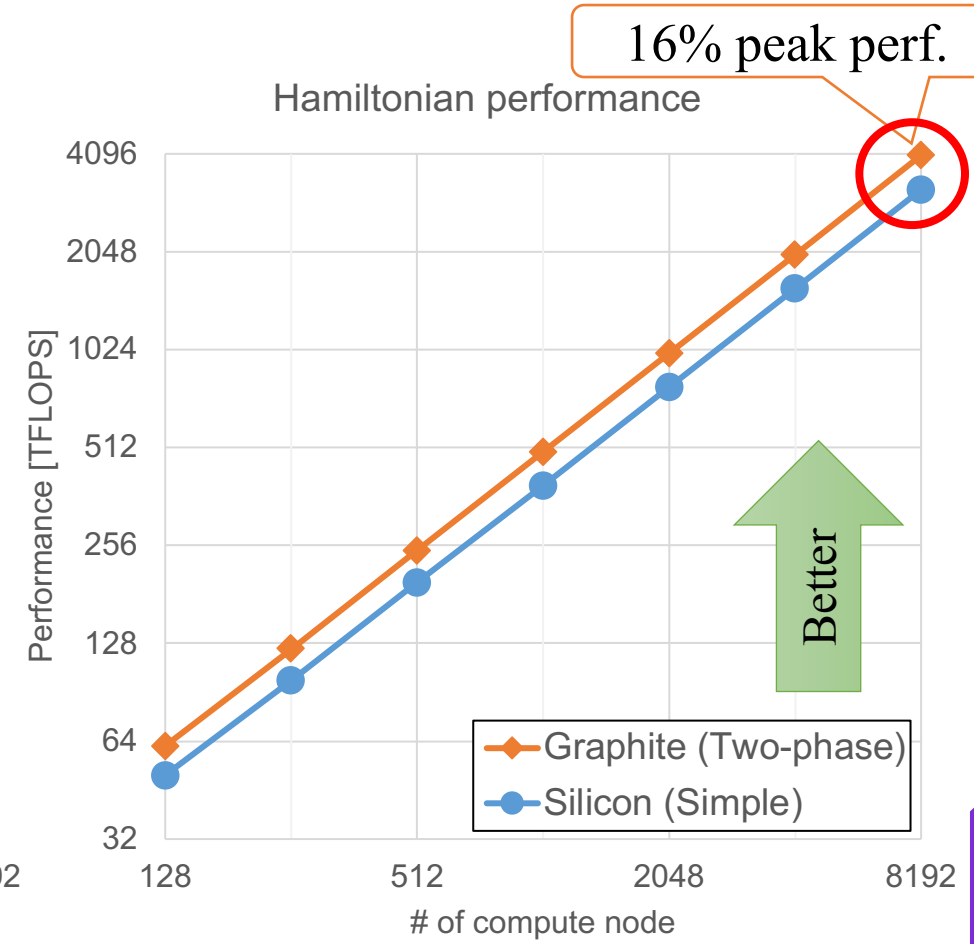
Full-system evaluation of OFP

- Laser-interaction problem to apply material processing with laser cutter

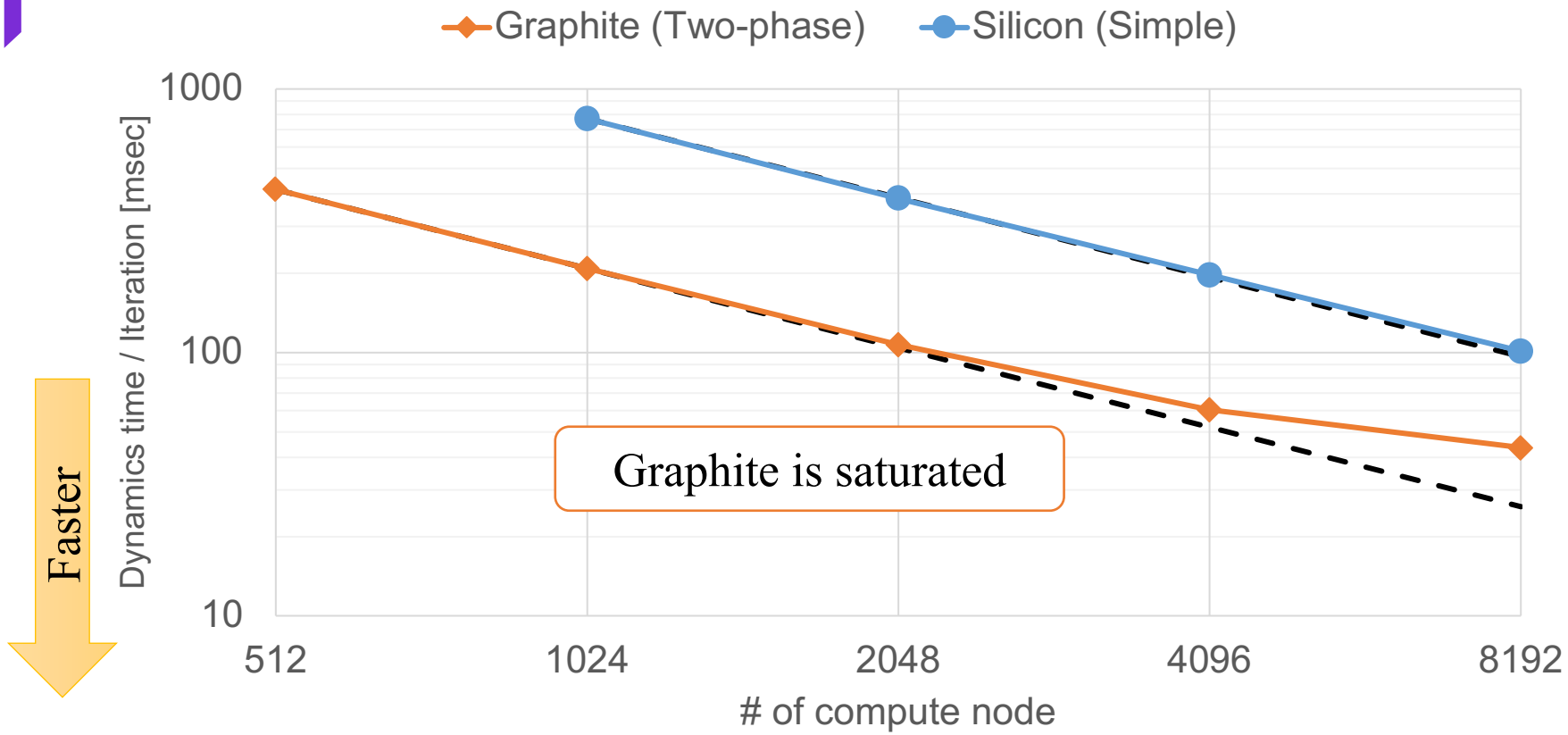
	Graphite (Graphene)	Silicon
Distribution scheme	Two-phase	Simple
MPI procs. / Macro-grid	8	–
Macro-grids / MPI proc.	–	1, 2, 4
Total # of macro-grid (max)	1024	32768
# of wave space	7928×16	8^3×16
Size of 3-D real scape	26×16×16	16^3
Data size / MPI proc.	$(7928 \div 8)\times 16\times NL$ ≈ 1.6 [GB]	$8^3\times 16\times NL$ ≈ 0.5~2.0 [GB]
Actual calculation time (1 case with 8192 KNL)	5–6 hours	8–9 hours

Weak scaling

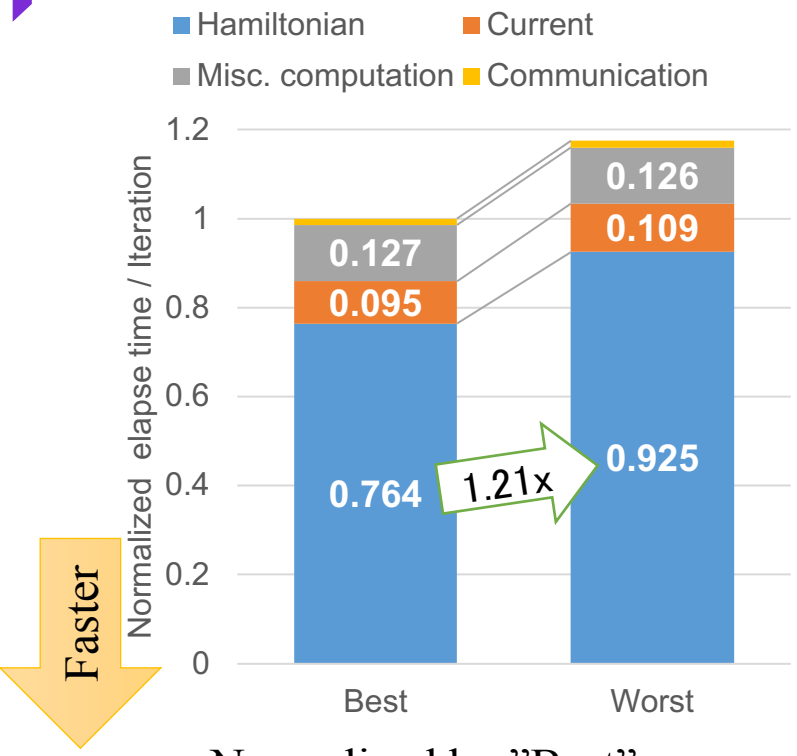




Strong scaling



Why degraded performance



Normalized by ”Best” case

- Blue box shows the computation-only part
  1. Communication does not included
  2. Problem size per node is even
- Graphite case uses two-phase distribution
  - Requires two-phase synchronization in both sub MPI communicator and all MPI processes
  - It is sensitive for the load-imbalancing
- Non-algorithmic load-imbalancing
  - Intel Turbo Boost mechanism

Conclusion

- We implemented and evaluated the application on many-core system
  - To Intel Knights Landing from Intel Knights Corner
- 25-point stencil computation
  - KNL is 2.5x faster than that of a KNC
- Time-development part
  - Performance: 1-KNL > 2-KNC
- Full-system evaluation of Oakforest-PACS
  - Our code is scalable on world-class KNL many-core system
  - Future work: solve load-imbalancing problem between compute nodes
- Current: we develop a SALMON towards ”Post-K” and beyond