Performance Evaluation of Large Scale Electron Dynamics Simulation under Many-core Cluster based on Knights Landing

Yuta Hirokawa
Graduate School of Systems and Information Engineering, University of Tsukuba
Ibaraki, Japan
hirokawa@hpcs.cs.tsukuba.ac.jp

Taisuke Boku
Center for Computational Sciences, University of Tsukuba
Ibaraki, Japan

Shunsuke A. Sato
Max Planck Institute for the Structure and Dynamics of Matter
Hamburg, Germany

Kazuhiro Yabana
Center for Computational Sciences, University of Tsukuba
Ibaraki, Japan

ABSTRACT

We have been developing an advanced scientific code called "ARTED" for an electron dynamics simulation using the first-order computation of materials to be ported to various large-scale parallel systems including the "K" Computer, which was previously Japan's fastest supercomputer. In this paper, the implementation and performance evaluation of the ARTED code used in Intel's latest many-core processor, the Knights Landing (KNL) stand-alone cluster, are described based on past research on porting the code to the Knights Corner (KNC) accelerator. Our target system is Oakforest-PACS, which is currently the fastest supercomputer in Japan. For performance tuning on KNL, the largest issue is how to utilize multiple levels of parallelism, such as the instruction level (512-bit SIMD instruction), hardware thread (4 threads/core), and large number of cores. We focus on the dominant computation part of the code, where 25 points of a 3D stencil computation are required.

We successfully optimize this part to achieve 758.4 GFLOPS per node, which corresponds to 24.8% of the theoretical peak on the node of Oakforest-PACS using an Intel Xeon Phi 7250 (3046 GFLOPS peak). It is also shown that the KNL sustained performance is better than that of the two KNC accelerator cards. The entire ARTED code implies other time step computing, and was designed for a large-scale parallel execution using MPI, whereas single-node parallelization is achieved using OpenMP. We finally evaluate the entire parallel execution performance with up to 128 nodes.

CCS CONCEPTS

• **Computing methodologies** → **Multiscale systems**; **Massively parallel and high-performance simulations**; **Simulation evaluation**;

KEYWORDS

Intel Xeon Phi, Knights Landing, Electron Dynamics Simulation

HPC Asia 2018, January 28–31, 2018, Chiyoda, Tokyo, Japan

ACM ISBN 978-1-4503-5372-4/18/01.
https://doi.org/10.1145/3149457.3149465

ACM Reference Format:
Yuta Hirokawa, Taisuke Boku, Shunsuke A. Sato, and Kazuhiro Yabana. 2018. Performance Evaluation of Large Scale Electron Dynamics Simulation under Many-core Cluster based on Knights Landing. In *HPC Asia 2018: International Conference on High Performance Computing in Asia-Pacific Region, January 28–31, 2018, Chiyoda, Tokyo, Japan.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3149457.3149465

1 INTRODUCTION

The many-core architecture is promising to be a new generation of high performance processors, providing a large number of cores in a chip to achieve an aggregated performance with a relatively simple control of each core rather than increasing the CPU clock frequency, and providing highly functional multiple cores to save the total power consumption of the chip for a higher performance-watt ratio. Intel's Xeon Phi processor series is today's representative many-core processor, and was recently promoted to a standalone self-bootable CPU in the new generation Knights Landing (KNL) architecture based on the previous Knights Corner (KNC) architecture, which is implemented as an accelerator card. The performance of a KNL CPU is not provided solely based on the number of cores, but also by highly enhanced SIMD instructions and the high bandwidth on-chip memory of MCDRAM. Therefore, when we port any scientific code, it is necessary to tune the code originally developed for an ordinary Xeon CPU to fit the architectural characteristics of KNL.

Studies have been conducted on the development of an electron dynamics simulation code called "ARTED" for use in various large-scale parallel systems including the "K" computer, which was Japan's previous fastest supercomputer with a theoretical peak performance of 10 PFLOPS. To achieve a much higher performance required for the next generation of optical material science, the code has been optimized particularly for large-scale accelerated clusters using a KNC processor [9]. The many-core architecture systems have entered a new era by utilizing a KNL CPU to provide a standalone system to replace ordinary commodity CPU clusters, especially for highly computational and memory bound applications. In this study, we ported the application to KNL based on the optimized code for KNC.

In December 2016, the Joint Center for Advanced HPC (JCAHPC) [21], under cooperation with University of Tsukuba and the University of Tokyo, started the full-operation of a large-scale KNL cluster system called "Oakforest-PACS" with 8208 nodes and a theoretical peak performance of 25 PFLOPS. The system ranked sixth on the TOP500 List in November 2016, achieving a Linpack performance of 13.55 PFLOPS, and has been recognized as Japan's fastest supercomputer [22][1]. Moreover, the HPCG benchmark achieved 0.3855 PFLOPS, ranking it third in the world [23]. We implemented the ARTED code to target the Oakforest-PACS system for large-scale many-core cluster computing.

For performance tuning on KNC or KNL, the largest issue is how to utilize the multiple levels of parallelism, such as the instruction level (512-bit SIMD called AVX-512), a huge number of hardware threads (each core runs up to four threads), and the physical cores (up to 72 cores) on each node. In this paper, we describe how to enhance the performance of ARTED on a single core to achieve the highest performance when considering the various architectural characteristics of KNL as well as a many-thread operation on the shared memory architecture of a single node. A parallel execution on MPI with up to 128 nodes of the KNL cluster is then shown.

The rest of this paper is constructed as follows. First, we introduce the overview of our target code, ARTED, in Section II, and then briefly introduce the past reported work for performance tuning on KNC in Section III. After describing the porting to KNL in Section IV, its performance evaluated on the Oakforest-PACS system is detailed in Section V. Finally, we conclude the paper in Section VI.

2 ARTED: ELECTRON DYNAMICS SIMULATOR

2.1 Overview

The Ab-initio Real-time Electron Dynamics (ARTED) simulator is a multi-scale electron dynamics simulator based on the real-time real-space density functional theory (RTRSDFT), developed at the Center for Computational Sciences, University of Tsukuba [1]. The application simulates the interaction between pulse light and matter to solve RTRSDFT using coupled electron dynamics, namely, the Time-Dependent Kohn-Sham (TDKS) equation, and electromagnetic fields, that is, the Maxwell equation, through a current, as shown in Fig. 1. The purpose of the application is to achieve research progress in optoelectronics, which is at the frontier in computational science. The application contributes to the material sciences under international collaboration [7, 8]. In addition, the performance of ARTED is evaluated on the "K" Computer [19] using up to 11,520 nodes (92,160 cores). As shown here, the application is sufficiently parallelized for large-scale computing facilities. ARTED has already been released as an open-source software, and our entire implementation has been made available [27].

Here, a time-development loop computes the wave function using a time-dependent Kohn-Sham equation. The time-development loop is a type of 3D stencil computation, where the physical domain is decomposed and computed, reflecting the variables in the surrounding external area. This 3D stencil computation is applied

[1]It ranked seventh on the latest TOP500 list in June 2017.

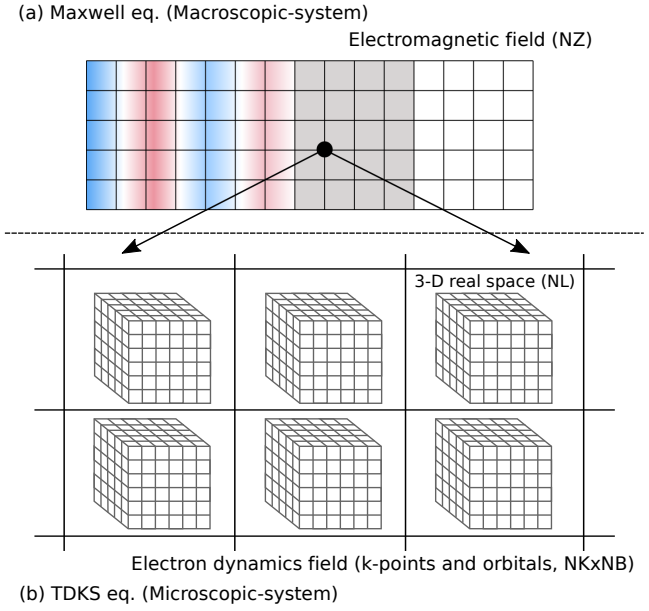

Figure 1: ARTED computation fields (Maxwell + TDKS equation)

to every wave band, and we obtain a considerable degree of parallelism in total. Actually, we do not apply a domain decomposition for a 3D stencil computation to avoid communication overhead in the halo region. The time-development part with the wave function of electrons uses a fourth-order Taylor expansion [2]. A stencil computation arises when operating a Hamiltonian of the wave function. The number of time steps in the time-development loop is quite large, within the range of 10,000–100,000 steps [3], and the optimization for the stencil computation is the most crucial in this code.

As a related work, an open-source software used for a real-time real-space DFT code, called "OCTOPUS," has been widely utilized [4]. "GCEED," which is another real-time real-space DFT code has been reported using large-scale computations of up to 7,920 compute nodes on "K" computer [5]. As another challenge for a large-scale simulation, Sequoia BlueGene/Q at LLNL recorded an actual performance of 8.75 PFLOPS using full 98,304 nodes [6]. As a novel contribution to the computing frontier, our challenge is to achieve a coupled simulation with electron dynamics and electromagnetic fields. Such a coupled application has yet to be achieved, and we are hoping to discover new theories based on ARTED.

2.2 Implementation details

ARTED has four parameters related to computational complexity: a macroscopic grid point, Bloch wave number $\boldsymbol{k}$, band number, and space lattice for representing the wave function of electrons. A wave function is represented by an array of (NZ, NK, NB, NLx, NLy, NLz). NZ denotes the number of macroscopic grid points, NK denotes the number of $\boldsymbol{k}$, NB denotes the size of the band, and (NLx, NLy, NLz) denotes the size of the three-dimensional space lattice. The total size of a 3D space is also denoted as NL (= NLx

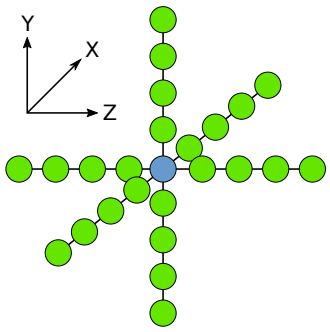

Figure 2: Memory access pattern of stencil computation.

× NLy × NLz). In this paper, we target the performance evaluation of the TDKS equation, and the wave function array size is denoted as (NK, NB, NL). In other words, we evaluate the strong scaling performance of electron dynamics fields only. NZ indicates the size of electromagnetic fields, and relates only to a weak scaling performance. For the evaluation, we targeted these parameters for two materials, Si and SiO_2. The typical wave function array parameter is set as (NK, NB, NL) = (24^3, 16, 4096 = (16, 16, 16)) for Si, and (4^3, 48, 37440 = (20, 36, 52)) for SiO_2. In this paper, we focus on a fixed size problem to evaluate the strong scalability. The communication between macroscopic grid points is very small in time-development loops.

We consider how to parallelize the computation in a hybrid manner using MPI and OpenMP. Only the Bloch wave number $\boldsymbol{k}$ (NK) is distributed to processes using MPI because this is the largest parameter for the computation domain. The domain with NK/NP×NB is then parallelized using OpenMP threads, where NP denotes the number of MPI processes. Each MPI process computes the space domain (NL) with the amount of NK/NP×NB. The time-development on a stencil computation is then executed iteratively. Here, the most important aspect is that we do not apply domain decomposition on the space, which is the most popular parallelization on an ordinary stencil computation. The space domain size (NL) is relatively small compared with the wave counts (NK and NB), and thus it is inefficient to distribute it to multiple threads or processes. As a result, there is no data exchange for a 3D space lattice in the halo region. Each OpenMP thread computes a space lattice (NL) independently and sequentially. Hence, the degree of parallelism of stencil computation in an OpenMP thread is NK/NP×NB. This hybrid parallelization minimizes the communication and synchronization overhead because we can compute the Bloch wave number and wave bands independently in a time step. Therefore, we can focus on the optimization of the many-core processors for a sequential computation and OpenMP task scheduling. A space lattice of the computation field is represented by a double-precision complex (double-complex) value, and computes a 25-point stencil with a periodic boundary condition on the Hamiltonian computation. Fig. 2 shows a 25-point stencil access pattern of a 3D stencil computation in ARTED. Here, the memory bandwidth requested in this part is 2.68 bytes/Flop without cache utilization. Therefore, it is strongly bound by the memory bandwidth, and the cache hit ratio is critical for the performance improvement. Through these

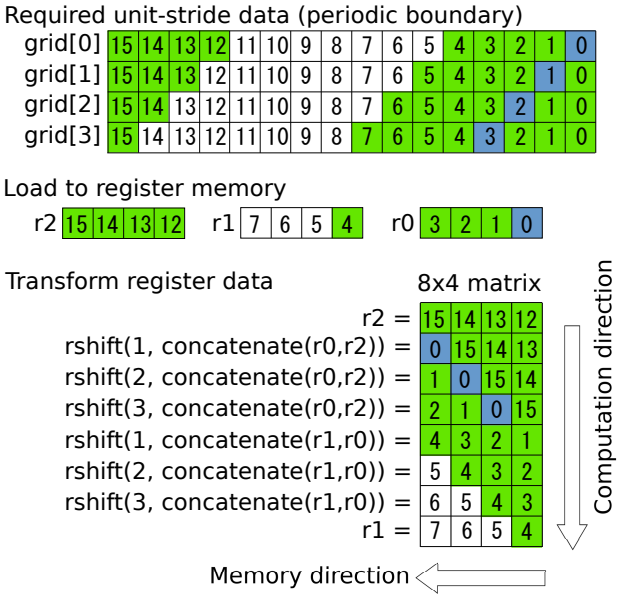

Figure 3: Schematic image of unit-stride memory access optimization

features, we evaluate the performance of the stencil computation as the main target for optimization.

3 PREVIOUS STUDIES ON INTEL KNIGHTS CORNER

A study on porting the ARTED code to the first-generation Intel Xeon Phi many-core processor, KNC [9], was conducted. In this section, we introduce this work briefly to help in understanding our study presented in this paper.

3.1 Optimization for KNC

A 25-point stencil computation was implemented using the explicit vectorization of C-language (Explicit vec.) for the KNC accelerator processor after optimizing the automatic vectorization of the Fortran90 compiler (Compiler vec.). In particular, the Z-dimension (unit-stride) memory access should be optimized to decrease the memory pressure. An `alignr` instruction (concatenated shift) is used for implementation of unit-stride memory access without `gather` because the instruction performance is very low with each processor [10]. A memory access pattern and schematic of the implementation is shown in Figs. 2 and 3. This implementation updates four grid points to fit to the SIMD operation length. In Fig. 3, the required data are loaded into the vector register through a `load` instruction. The ranges of ±4 neighbor points are transformed into a 8×4 matrix through a `alignr` instruction. The computation direction is column-wise, and is a very simple computation for SIMD instructions. Their approach is optimized to a periodic boundary condition based on Intel's report [11]. Whereas that work avoids a periodic boundary condition, their work covers both situations with and without such condition.

The index calculation for a periodic boundary condition requires an integer remainder operation, and the performance is degraded through this operation because the instruction on Xeon Phi processors is extremely slow compared with that of the Xeon CPU. When a field size has a power of 2, we can replace this operation with a logical AND instruction. The KNC performance of a stencil computation with a remainder operation is only 65% of that with a logical AND instruction for a masking operation. With explicit vectorization, they apply a table lookup instead of a remainder operation to avoid limiting the size to a power of 2. This method is slower than a logical AND, but is still faster than using a remainder operation.

The multiplication of complex values can be simplified within the stencil computation using intrinsics because one of the operands is a constant value of $(0, -i)$. The vectorization performance on the KNC is degraded when the multiplication of complex variables is needed because it must be separately executed for the real and imaginary parts. The multiplication can be replaced with a two-step operation.

(1) Swap a real part for an imaginary part using a `_mm512_shuffle_epi32` instruction.
(2) Flip the sign bit of an imaginary part with a `_mm512_xor_si512` instruction.

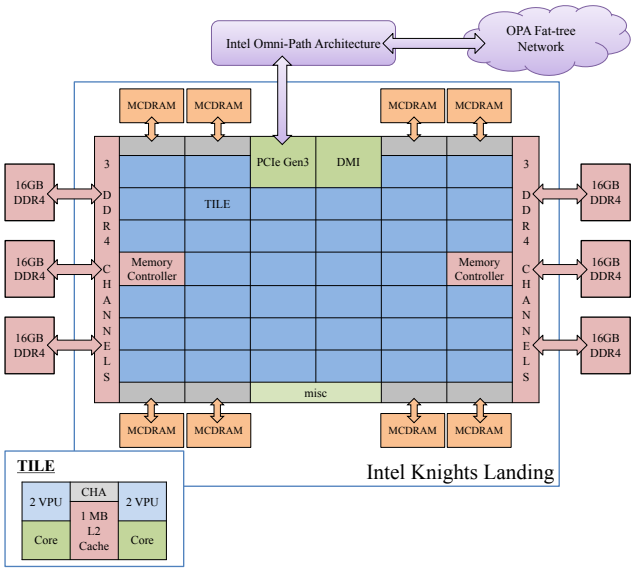

Figure 4: Example of Knights Landing cluster configuration (Oakforest-PACS)

3.2 KNC performance results

In a KNC accelerator card, the stencil code optimization with explicit vectorization using a 512-bit SIMD instruction achieves a performance approximately 7.3 times better (212.2 GFLOPS) than the original code (29.0 GFLOPS), and the KNC was approximately two times faster than a socket CPU with an Ivy-Bridge architecture (106.9 GFLOPS). For the entire code, we can apply the Symmetric mode to combine the CPU and KNC resources together for a higher aggregated performance on each node, where the performance of the system is greatly enhanced under strong scaling. When compared with the same number of computing nodes, the KNC performed approximately 1.45 times better when compared to the CPU, and the performance of the Symmetric mode was approximately 2.16 times better than the CPU-only execution when maintaining the load balance between the CPU and KNC.

Based on these results, we enhanced the code and evaluated the performance on KNL.

4 INTEL KNIGHTS LANDING

4.1 Architecture

KNL is Intel's latest many-core processor, known as Xeon Phi (Many-Integrated Core architecture, or MIC), following KNC. This section explains the important change from KNC. [12] provides more detailed information.

- A socket-type processor is provided along with a PCIe card version.
- A chip is equipped with high-bandwidth memory called MCDRAM.
- The intercore network is changed into a 2D mesh from the ring.
- Each core provides an AVX-512 instruction.
- The aggregated performance is three times faster than KNC because of a larger number of cores, enhanced instruction set, and Turbo Boost technology.

Using a socket-type processor, we can build a KNL-standalone system similar to a traditional Xeon CPU cluster. Fig. 4 shows the chip configuration and an example cluster configuration. In addition to MCDRAM, the processor can be equipped with up to six channels of traditional DDR4 memory and communication devices such as an InfiniBand or Intel Omni Path Architecture. The system is much simpler than accelerated systems such as GPU clusters.

The intercore network of KNL is changed into a 2D mesh instead of KNC's ring network. A pair of two cores called a "Tile" shares 1 MB of L2 cache. If the cache is used by two cores in a Tile evenly, the capacity is equivalent to a core of KNC. Each core is equipped with two vector processing units (VPU), each of which provides AVX-512 instructions and four hardware threads similar to KNC. A DDR4 memory controller and three memory channels are provided on the two sides of a chip. As the most important change for the application, MCDRAM with up to 480 GB/s of actual bandwidth is implemented with 16 GB of chip capacity.

There are several configuration combinations in a 2D mesh network and MCDRAM memory utilization mode, which should be fixed at the boot time of a KNL chip. A 2D mesh network provides three modes:

- All-to-All
- Quadrant/Hemisphere
- Sub-NUMA Clustering (SNC)

All-to-All treats all on a chip as a single network, which takes a long pass among the core and memory. Quadrant/Hemisphere divides the network into four or two virtual sub-networks to minimize the longest pass. SNC can be divided into four (SNC4) or

```
#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
```

Figure 5: Schematic code of conversion into AVX-512 from IMCI with preprocessor

two (SNC2) virtual NUMA networks. We describe SNC4 as an example because both modes have the same behavior. SNC4 divides the network into four virtual NUMA nodes, which act as a sort of closed memory access with a 4-socket Xeon CPU NUMA node. KNL provides the following configuration for a use case of MCDRAM (high bandwidth, small capacity) and DDR4 memory (low bandwidth, large capacity).

Flat mode
MCDRAM and DDR4 memory are separated as NUMA memory.

Cache mode
MCDRAM is used as a last-level (L3) cache.

Hybrid mode
MCDRAM is divided into NUMA memory and L3 cache (*Flat + Cache*).

If the data size per node is smaller than the MCDRAM capacity, it is best to use MCDRAM only for an application. To change the memory and NUMA mode, we need to reboot the processor under different BIOS settings. A preliminary evaluation is needed because the performance is influenced based on the configuration [13] [15].

4.2 ARTED porting to KNL

This section describes the conversion of the IMCI vectorized code into AVX-512 instructions, which is provided under KNL, Skylake-SP, and later CPUs. The instruction format, such as four arithmetic operations, is basically the same in both IMCI and AVX-512 with the exception of shuffle, permute, and other special instructions. In an unaligned load instruction, whereas IMCI requires that two instructions be sent between cache lines, AVX-512 only issues one instruction, similar to AVX. Our implementation requires that the following instructions and operations be replaced for execution under AVX-512:

(1) 128-bit unit shuffle instruction.
(2) Non-temporal storage (stream store) instruction.
(3) Unaligned load instruction.
(4) A double precision value to double the complex value conversion.

Table 1: Summary of differences between KNC and KNL versions

Code conversion	IMCI specifically routines to AVX-512 should be replaced.
Compiler options	Architecture specific options should be replaced.
Execution mode	We can select the mesh and memory modes under application characteristics.
OpenMP affinity	KNL provides 2-D mesh network chip topology. We should change the affinity for memory locality.

Instructions 1 and 2 have the same parameters in both instruction sets except for the name of the instruction or argument sequence. Instructions 3 and 4 are very different, such as in the number of required instructions; however, it can be replaced with 5–10 lines of inline functions. These modifications can be taken easily through preprocessor directives, as shown in Fig. 5. A `__AVX512__` or `__MIC__` macro symbol is defined using the compiler when compiling for AVX-512 provided processors or KNC. Our explicit vectorization code for KNL is implemented based on a KNC-optimized code, which applies preprocessor directives, and we additionally optimized it for KNL. An AVX-512 instruction set consists of several separate subsets, and some generation of processors do not support all subsets. Our implementation requires a common subset only, called an AVX-512F (foundation) instruction set, which is supported by all types of processors. In other words, our implementation can be applied to all processor families equipped with AVX-512.

On Oakforest-PACS equipped with a Xeon Phi 7250, it is recommended to use 64 cores per node because the CPU enables 34 out of 36 Tiles to make the total core count 68. It is desirable to maintain some cores as assistants to support the operating system, interrupt handling, and other functions. In addition, Oakforest-PACS is equipped with a tickless-kernel, which has no OS interruption to the cores except core 0. We believe that an execution with up to 66 cores is the best practice owing to the pollution of L2 cache, which is shared in each Tile (pair of two cores).

We evaluate the stencil computation using 64 cores and 256 OpenMP threads (four threads per core) because this is the appropriate number as a power of 2. Core-Thread affinity refers to two environment variables under Intel OpenMP. `KMP_HW_SUBSETS` controls the affinity in OpenMP, such as the number of cores and number of threads per core. For example, `KMP_HW_SUBSETS=64c@2,4t` means that 64 cores and 256 OpenMP threads except cores 0 and 1 are assigned to an application. `KMP_AFFINITY` operates the orchestration of a CPU ID and OpenMP thread ID. The application excepts that the thread ID is continuous within a core because two cores and eight threads compose a Tile and they share L2 cache. Therefore, we apply `KMP_AFFINITY=balanced,granularity=fine`.

Finally, Table 1 shows a summary of the difference in optimization between KNC and KNL.

Table 2: Evaluation Environment of Knights Landing cluster (Oakforest-PACS)

# of Node	128 (system total: 8208)
Processor	Intel Xeon Phi 7250 1.4 GHz
# of Cores / Node	68 (We use up to 64 cores)
Memory	16GB (MCDRAM) + 96GB (DDR4)
Interconnect	Intel Omni-Path Architecture
Compiler & MPI	Intel compiler 17.0.1 & Intel MPI 2017 update 1
Peak perf. / Node	3046 GFLOPS

Table 3: Evaluation Environment of Knights Corner cluster (COMA)

# of Node	128 (system total: 393)
Processor	Intel E5-2670v2 2.5 GHz ×2 + Intel Xeon Phi 7110P ×2
# of Cores / Node	20 (10 × 2) + 120 (60 × 2)
Memory	64GB (CPU, DDR3) + 8GB ×2 (Xeon Phi, GDDR5)
Interconnect	InfiniBand FDR (Mellanox Connect-X3)
Compiler & MPI	Intel compiler 16.0.2 & Intel MPI 5.1.3
Peak perf. / Node	400 GFLOPS (200 × 2) + 2148 GFLOPS (1074 × 2)

5 PERFORMANCE EVALUATION

In this study, we evaluate the performance of ARTED on the KNL cluster "Oakforest-PACS" at JCAHPC [21] in comparison with the KNC cluster "COMA" at University of Tsukuba [20]. The environment of each system is shown in Tables 2 and 3. Oakforest-PACS and COMA are equipped with an interconnection network with a full bisection bandwidth of a fat-tree topology by Intel's Omni-Path architecture (OPA) and InfiniBand FDR, respectively. COMA is a heterogeneous accelerated cluster with a KNC card and Ivy-Bridge Xeon CPU (IVB), whereas Oakforest-PACS is a homogeneous system with KNL only. In this paper, we compare the KNL results and their previously published performances under KNC [9]. Here, we additionally apply cache-prefetching to the original KNC code to make a fair comparison between these two systems.

In the performance evaluation on Oakforest-PACS, we apply a Quadrant NUMA configuration and Flat mode, where the MCDRAM is explicitly used to hold all application data[16]. This combination of configurations is called a Flat-Quadrant, which we hereafter call a "Quadrant" for short because we continuously apply Flat mode for MCDRAM handling. A result with a different NUMA mode with SNC4 will also be examined later. Our application performance is dominated with stencil computations, which are highly memory bandwidth bound. [16] shows that the DDR4 performance is slower than that of MCDRAM because of a memory bandwidth bottleneck when the data size fits the capacity of MCDRAM. If the application data size is larger than the MCDRAM capacity, we should still use MCDRAM only by increasing the number of computation nodes rather than using DDR4 memory.

```
#define L1PREFETCH_DISTANCE 64
inline
__m512d _mm512_load_prefetch_pd(void const* c) {
  __m512d a = _mm512_load_pd(c);
  _mm_prefetch(
    ((char const*)c) + L1PREFETCH_DISTANCE,
    _MM_HINT_T0
  );
  return a;
}
```

Figure 6: Insertion of software prefetch to load instruction

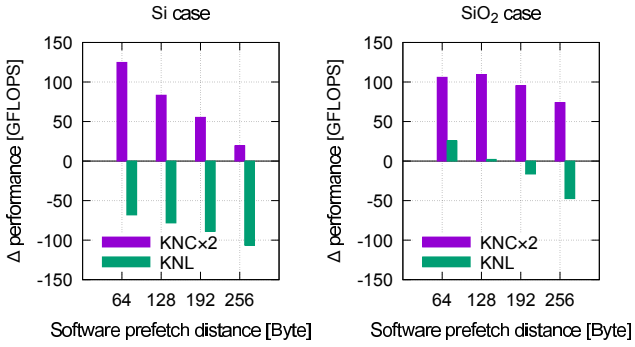

Figure 7: Software prefetch impact [GFLOPS]

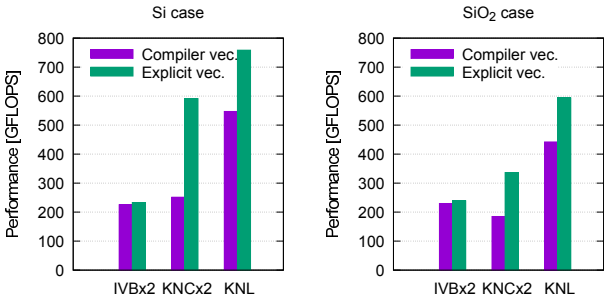

Figure 8: The stencil computation performance with various processors [GFLOPS]

5.1 25-point stencil computation

For an advanced optimization, we apply a software prefetch instruction on KNL following the optimization for KNC. Our implementation is optimized by a spatial locality, and thus the register data are fully utilized in the computation. Here, the software prefetch instruction is issued for the required data at the next iteration after issuing a load instruction during the current iteration. A stencil domain is aligned to a 64-byte block, which is identical to a cache line size, and thus the code always prefetches the

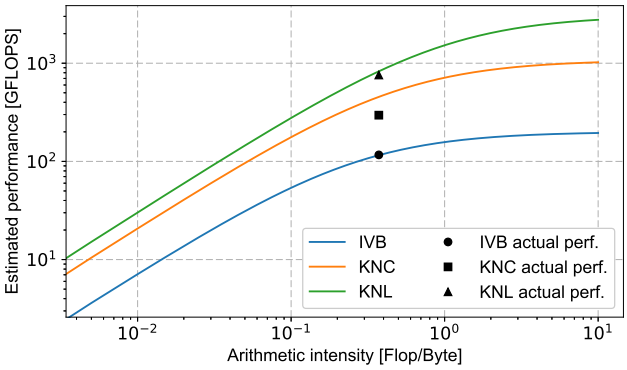

Figure 9: Roofline analysis with L2 cache bandwidth

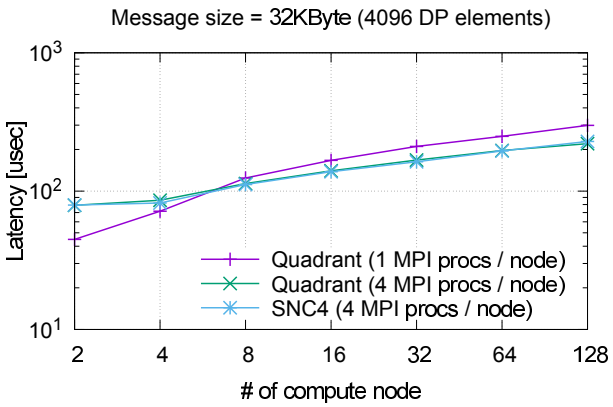

Figure 10: The `MPI_Allreduce` latency of KNL + OPA

next cache line. Fig. 6 shows the load operation of a double precision value with software prefetching. The function used here is `_mm512_load_pd` under AVX-512 or IMCI. Fig. 7 shows the software prefetch impact described in the delta performance for the two materials. In the SiO_2 case, the performance of both processors is improved, particularly on KNC by up to 50 GFLOPS per processor when the prefetch distance is set to 64 bytes. In the Si case, however, the performance in KNL is degraded by a software prefetch, whereas the KNC performance is improved by up to 60 GFLOPS per processor. In several cases, KNL does not require the manual insertion of a software prefetch owing to the improved hardware prefetcher as an advanced feature from KNC [14]. The computation data size of SiO_2 is 585 KB per wave domain, and may spill out from L2 cache on each Tile. An investigation into the impact of a detailed software prefetch will be one of our future areas of study.

We compared the performance of the case with two KNC cards and one KNL processor, corresponding to the single-node performance of COMA and Oakforest-PACS, respectively. Fig. 8 shows the stencil computation performance with various processors: two KNCs, two IVBs, and one KNL. For IVB, the code is explicitly vectorized with 256-bit SIMD instructions based on the KNC implementation, by converting AVX-512 (512-bit) instructions into AVX (256-bit). For KNC, the performance of the SiO_2 case is simply one-half of Si because the domain size is greater than the L2 cache capacity causing a cache spill. On the other hand, KNL achieves 1.77 times the performance of two KNC cards, which is almost comparable theoretically with three KNC cards. The Turbo Boost technology on KNL improves the performance of the serial execution part, which is particularly important in the SiO_2 case. Finally, KNL achieves a performance of 758.4 GFLOPS with a 24.8 % theoretical peak in Si, which implies a large degree of thread level parallelism and a relatively small part for sequential execution. Because we optimized the KNL version based on the KNC-optimized code, it can be shown that the optimization of KNC strongly helps with further optimization on KNL.

We estimated the achievable performance using various processors based on a memory bandwidth bottleneck. Fig. 9 shows a roofline model analysis with L2 cache bandwidth [17]. Our stencil computation is strongly impacted with a L2 cache bandwidth because the domain size per two cores fits the L2 cache capacity, whereas that of Si is 64 KB per domain. KNL and IVB nearly follows the roofline model, whereas KNC has only an estimated performance of approximately 70%. In the case of SiO_2, the data size for a single domain slightly exceeds the L2 cache capacity, and the performance is then degraded as described above. We believe that the MCDRAM and DDR4 bandwidth have an influence when caching the domain to L2 from a global array. Each thread computes a Hamiltonian under thread-local working memories. A thread copies a domain to working memory from a global array at MCDRAM or DDR4 before computing the Hamiltonian, and the domain is then cached to L2.

5.2 Preliminary: Collective communication

Oakforest-PACS is the world's largest Omni-Path Architecture (OPA) system, as of June 2017. The OPA interconnection is not equipped with a hardware offloading feature, which is available on InfiniBand EDR, although these network systems share a 100 Gbps link performance. We are aware of a collective communication performance on it because the system code execution performance on KNL is lower than with an ordinary Xeon CPU. Thus far, this has not been well studied with regard to the combination of OPA and KNL for collective communication, and we evaluated the performance of `MPI_Allreduce`, which is the dominant interprocess communication in ARTED. [18] shows the basic performance and characteristics. We evaluated the collective communication performance using Quadrant and SNC4. Intel MPI with OPA provides "Tag Matching Interface (TMI)," "OpenFabrics capable device (OFA)," and "OpenFabrics Interface (Libfabric, OFI)" [24] [25] as the variations in a communication stack. TMI and OFI use the Intel Performance Scaled Messaging 2 (PSM2) API internally [26], which is implemented in OPA. Oakforest-PACS uses TMI stack as a default.

Fig. 10 shows `MPI_Allreduce` the latency for the Si case, which is required in the time-development part. The latency with SNC4 is smaller than that of Quadrant in the Si case simulation, which requires `MPI_Allreduce` with a size of 32 KB at maximum. For the best use of MCDRAM on an SNC4 mode, we run four MPI processes per node (with a quarter number of OpenMP threads for

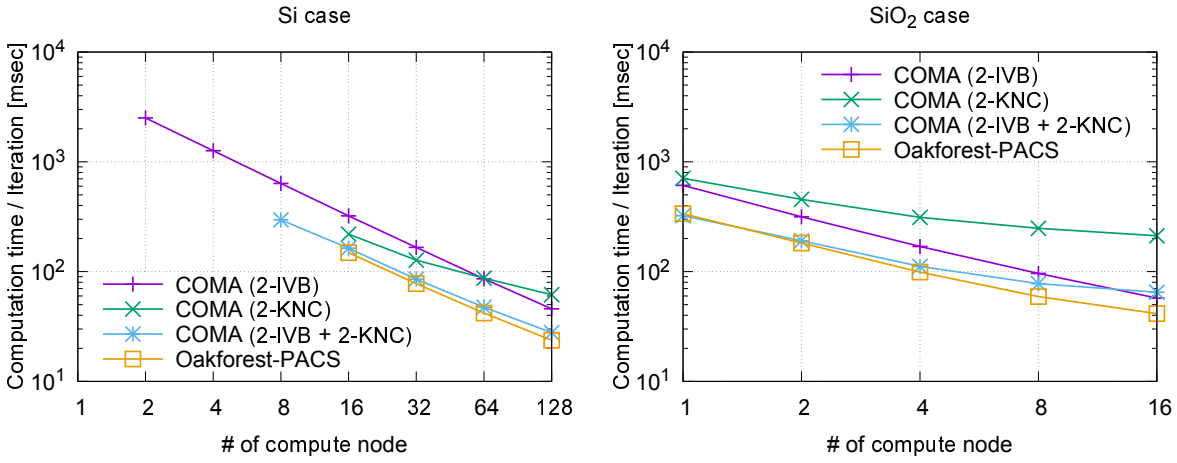

Figure 11: The overall performance with Si and SiO_2

each), whereas we run a process per node in Quadrant mode. Naturally, the total message count of `MPI_Allreduce` in the SNC4 case is much larger than that of the Quadrant case because of four times the number of MPI processes. Regardless, this result suggests that OPA utilization with multiple processes enhances the aggregated throughput, particularly for a small message size. Here, the communication time is impacted simply by the number of MPI processes regardless of the NUMA model. However, with our application, the communication time in a time step is within the range of microseconds to milliseconds, whereas the entire computation time is within the range of milliseconds to seconds, as shown in the next section. Thus, we can still neglect the communication overhead for large-scale problems, and we apply only Quadrant mode for the code execution.

5.3 Time-development part

For the NUMA modes, Quadrant and SNC4 achieved the same performance through our evaluation. In ARTED, we parallelize the computation using a wave space in which we can flexibly control the process-thread affinity.

Fig. 11 shows the overall performance under strong scaling for the Si and SiO_2 cases. Using COMA, we evaluated IVB-only (Xeon only), KNC-only (through Private mode execution of Xeon Phi), and Symmetric mode execution, where IVB and KNC cooperate using MPI. The Si case achieves a linear scaling except a KNC-only execution. Oakforest-PACS (KNL) achieves a comparable or higher performance of the Symmetric mode of two KNCs. In the SiO_2 case, the performance of each system is saturated, particularly in Symmetric mode where the parallelism of KNC is insufficient. Because this mode is based on a heterogeneous execution using a Xeon CPU and Xeon Phi card, we applied the load balancing to change the number of wave function assignments to each type of processor. However, this load balancing does not work when the performance between processors is extremely different.

The performance degradation compared with the theoretical peak can be explained using the actual memory bandwidth. The actual memory bandwidth in each compute node is 480 GB/s (Oakforest-PACS, MCDRAM) and 450 GB/s (aggregated in COMA using two IVBs + two KNCs). Because the basic part of the application is a 25-point stencil computation, the final performance bottleneck is the actual memory bandwidth. Therefore, the result is reasonable.

6 CONCLUSIONS

In this study, we presented the actual KNL performance using an electron dynamics simulation code named "ARTED" as a real-world application, and how to port and optimize it for KNL from KNC-based optimization. We focus on the dominant computation part of the code, which is a 25-point 3D stencil computation. The code was successfully implemented on KNL with a minor change regarding the KNC-optimized version with 512-bit SIMD instructions. We achieved a performance of 758.4 GFLOPS per node, which corresponds to a theoretical peak of 24.8 %. This result indicates that the code tuning for KNC has a positive impact on KNL. In the entire code, the KNL cluster is faster than a cluster with two KNC cards per node with an ordinary Xeon CPU.

The KNL cluster requires the following consideration of the code tuning:

- Instruction conversion from KNC into KNL (with intrinsics);
- MPI and OpenMP affinity set with difference in mesh network mode such as Quadrant and SNC4;
- Performance impact of software pre-fetch instruction;
- Communication and computation performance based on the network fabric.

We are planning to extend the simulation size to the full system of Oakforest-PACS. Our future work includes a higher optimization of KNL and a performance evaluation of weak scaling using a very large number of KNL processors.

ACKNOWLEDGMENT

A part of this research was based on the Oakforest-PACS system operated at The Joint Center for Advanced HPC (JCAHPC) under cooperation with the Information Technology Center at the

University of Tokyo and the Center for Computational Sciences at the University of Tsukuba. The performance evaluation applied in this paper using the COMA system was supported in part by the interdisciplinary collaborative research program at the Center for Computational Sciences, University of Tsukuba. This work was also supported by CREST, JST (Grant No. JPMJCR16N5).

REFERENCES

[1] S. A. Sato, and K. Yabana: Maxwell + TDDFT multi-scale simulation for laser-matter interactions, *J. Adv. Simulat. Sci. Eng.*, Vol. 1, No. 1 (2014).
[2] G.F. Bertsch, J.-I. Iwata, A. Rubio, and K. Yabana: Real-space, real-time method for the dielectric function, *Phys. Rev. B*, Vol. 62, No. 12 (2000).
[3] K. Yabana, T. Sugiyama, Y. Shinohara, T. Otobe, and G. F. Bertsch: Time-dependent density functional theory for strong electromagnetic fields in crystalline solids, *Phys. Rev. B*, Vol. 85, No. 4 (2012).
[4] X. Andrade et.al: "Time-dependent density-functional theory in massively parallel computer architectures: the OCTOPUS project", *Journal of Physics: Condensed Matter*, pp. 233202, Vol. 24 (2012).
[5] M. Noda, K. Ishimura, K. Nobusada, K. Yabana and T. Boku: "Massively-parallel electron dynamics calculations in real-time and real-space: Toward applications to nanostructures of more than ten-nanometers in size", *Journal of Computational Physics*, pp. 145–155, Vol. 265, No. 14, (2014).
[6] E. W. Draeger, X. Andrade, J. A. Gunnels, A. Bhatele, A. Schleife and A. A. Correa: "Massively parallel first-principles simulation of electron dynamics in materials", *2016 IEEE International Parallel and Distributed Processing Symposium*, pp. 832 (2016).
[7] M. Schultze, K. Ramasesha, C. Pemmaraju, *et al.*: Attosecond band-gap dynamics in silicon, *Science 12 Dec 2014*, Vol. 346, No. 6215, pp. 1348–1352.
[8] M. Lucchini, S. A. Sato, A. Ludwig, *et al.*: Attosecond dynamical Franz-Keldysh effect in polycrystalline diamond, *Science 26 Aug 2016*, Vol. 353, No. 6302, pp. 916–919.
[9] Y. Hirokawa, T. Boku, S. A. Sato, and K. Yabana: Electron Dynamics Simulation with Time-Dependent Density Functional Theory on Large Scale Symmetric Mode Xeon Phi Cluster, *The 17th IEEE International Workshop on PDSEC'16* (2016).
[10] J. Hofmann, J. Treibig, G. Hager and G. Wellein: Comparing the Performance of Different x86 SIMD Instruction Sets for a Medical Imaging Application on Modern Multi- and Manycore Chips, *Proceedings of WPMVP'14*, pp. 55–64 (2014).
[11] C. Andreolli: Eight Optimizations for 3-Dimensional Finite Difference (3DFD) Code with an Isotropic (ISO), https://software.intel.com/en-us/articles/eight-optimizations-for-3-dimensional-finite-difference-3dfd-code-with-an-isotropic-iso
[12] A. Sodani: Knights Landing (KNL): 2nd Generation Intel Xeon Phi processor, *IEEE Hot Chips 27*, pp. 1–24 (2015).
[13] C. Rosales, J. Cazes, K. Milfeld, *et al.*: A Comparative Study of Application Performance and Scalability on the Intel Knights Landing Processor, *ISC '16 Proceedings*, pp. 307–318 (2016).
[14] B. Joó, D. D. Kalamkar, T. Kurth, *et al.*: Optimizing Wilson-Dirac Operator and Linear Solvers for Intel KNL, *ISC '16 Proceedings*, pp. 415–427 (2016).
[15] T. Barnes, B. Cook, J. Deslippe, *et al.*: Evaluating and optimizing the NERSC workload on Knights Landing, *Proceedings of the 7th International Workshop on PMBS '16*, pp. 43–53 (2016).
[16] C. Yount, and A. Duran: Effective use of large high-bandwidth memory caches in HPC stencil computation via temporal wave-front tiling, *Proceedings of the 7th International Workshop on PMBS '16*, pp. 65–75 (2016).
[17] S. Williams, A. Waterman and D. Patterson: "Roofline: An Insightful Visual Performance Model for Multicore Architectures" *Commun. ACM*, Vol. 52, No. 4, pp. 65–76 (2009).
[18] M. S. Birrittella, M. Debbage, R. Huggahalli, *et al.*: Enabling Scalable High-Performance Systems with the Intel Omni-Path Architecture, *in IEEE Micro*, Vol. 36, No. 4, pp. 38–47, July–Aug. 2016.
[19] RIKEN AICS, http://www.aics.riken.jp/en/
[20] CCS, University of Tsukuba, http://www.ccs.tsukuba.ac.jp/eng/
[21] Joint Center for Advanced HPC, http://jcahpc.jp/eng/
[22] TOP500, http://www.top500.org/
[23] HPCG Benchmark, http://www.hpcg-benchmark.org/
[24] Intel MPI, https://software.intel.com/en-us/intel-mpi-library
[25] OpenFabrics Interfaces: Libfabric, https://ofiwg.github.io/libfabric/
[26] Intel: opa-psm2, https://github.com/01org/opa-psm2
[27] Github: ARTED, https://github.com/ARTED/ARTED