

# HPC Application Performance & Characteristics on Knights Landing Architecture

Nisha Kurkure

Indian Institute of Science Education and Research Pune,  
India  
nishakurkure@iiserpune.ac.in

Goldi Misra

Indian Institute of Science Education and Research Pune,  
India  
goldi@iiserpune.ac.in

**Abstract**—High Performance Computing (HPC), serves as a catalyst for cutting edge scientific research. In the research and scientific arena, many scientific applications require immense computational power to run. To strike an appropriate balance between computation and power requirements, Intel has launched the second generation Xeon Phi called as Knights Landing(Knl). Knl is a many core technology which is designed for high degree of parallel processing. First generation Intel Xeon Phi is used only as a coprocessor, but the second generation Xeon Phi i.e. Knl can be used both as a self boot processor and a coprocessor. It runs on a very low frequency thereby minimizing the power consumption of the system. Due to paradigm shift in technology of many core architecture, it is imperative to test the performance of applications. The paper elaborates the architecture of Knl and its various modes of operation in addition to the performance of scientific molecular dynamics application Gromacs and Monte Carlo method for directed loop algorithm.

**Keywords**—High Performance Computing; Knl; Many Core Technology; Intel Xeon Phi; Processor; Coprocessor

## I. INTRODUCTION

The cost of moving data is increasing day by day as compared to the cost of computing. Earlier, High Performance Computing essentially focused on a single large scale system. Design of such mammoth systems led to development of architectures that were varied and extremely proprietary in comparison to other contemporary architectures. HPC is a tool, which not only solves grand challenge problems, but is now the essence of all possible commoditized services available to human beings. The demand for computing is increasing, and particularly the scientific domain is the largest consumer of compute cycles to carry out applied research. Initially, a processor was developed with a single cpu or core executing one thread at a time. Multi core and many core processors are used to run parallel applications for optimal speedup. Intel Xeon Phi family provides outstanding parallel performance for highly parallel workloads. Applications run best on Knl, if they have a high degree of vectorization and predictable high bandwidth communication with memory[7]. Specifically, in the following case scenarios:

- If memory utilization is less compared to the compute process, the application is compute intensive and may run well on Knl due to its high computational requirements.
- If the application requires predictable memory access pattern, then it may run well on Knl due to its on chip high-bandwidth memory (HBM)

- Finally, if an application is neither compute intensive nor memory intensive, it shall not run well on Knl architecture due to the inherent latency.

This paper is organized as: section II describes the architecture of Knl, various clustering modes and memory organization in terms of platform and on package memory. Section III describes a parallel and a serial application for this test. Section IV describes the system configuration and experiments conducted, followed by conclusion and future work.

## II. KNL ARCHITECTURE

Knl is a second generation Xeon phi processor with many new features. It is a first self-boot Intel xeon phi processor. There is significant improvement in its scalar and vector performance. It has on package memory and integration of fabric. Fig 1 shows[2] the architecture block diagram of Knight Landing processor. The Knl chip set consists of 36 tiles interconnected by 2D mesh. Each tile has 2 cores and 2 vector processing unit (VPU) per core with 1 MB L2 cache which is shared between the two cores. Each core has four threads, deeper out-of-order buffers, higher cache bandwidth, new instructions, better reliability, larger translation look-aside buffers (TLBs), and larger caches. In Knl processor, advanced vector extension instruction set AVX-512 have been added which provides 512 bit wide vector instruction. The instruction set is fully binary compatible with earlier version of Xeon processors. Knl consists of two types of memory, MCDRAM 16 GB on-package high bandwidth memory and other is DDR4 which scales up to 384GB capacity[2]. MCDRAM can be configured in different modes. The cache organizations in Knl come with increased complexity of the chip hardware. In order to handle complex cache organization and to ensure optimal performance for a user application, the user has to access the clustering mode[7].

### A. High Bandwidth Memory (MCDRAM)

In a computational intensive application, one of the performance bottlenecks is memory bandwidth. Bandwidth-limited applications are characterized by algorithms that have few floating point operations per memory access. In order to take care of such memory bandwidth applications, Knl has an on-package high bandwidth memory, which is based on multi-channel dynamic random access memory (MCDRAM)[8].

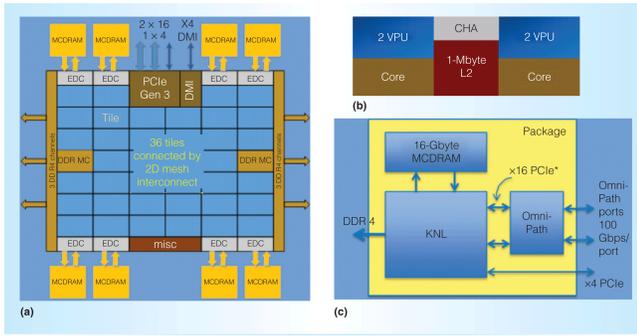


Fig. 1. Knights Landing block diagram (a) the CPU, (b) an example tile and (c) Knl with omni-path fabric integrated on the CPU package

HBM is used to achieve the highest performance on such bandwidth sensitive applications. HBM exists on the CPU chip, next to the processing cores. The MCDRAM memory modules in the Knl have upto 16GiB of HBM, which is fixed and not removable such as a normal memory system. This memory can be configured in three modes namely Cache mode, Flat mode and hybrid mode by setting up the BIOS configuration[8].

- Cache Mode: In this mode, the whole HBM can be used as a cache. This mode is not used frequently, but may have a lower performance than flat mode in case of frequent misses in the HBM.

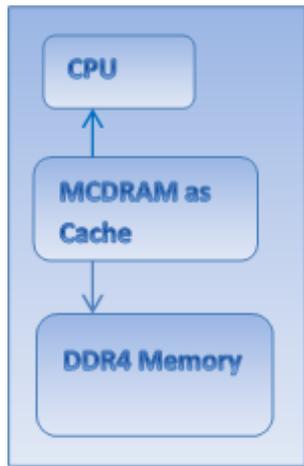


Fig. 2. MCDRAM in Cache mode

- Flat mode: The Flat mode uses the whole HBM as an addressable memory. It may offer better performance than cache mode, but requires modification of the code and/or execution environment.
- Hybrid Mode: In this mode, some portion of HBM is used as an addressable memory and remaining is used as a cache memory. It derives benefit of both flat and cache mode, but has smaller sizes of each.

### B. DDR4 Memory

Knights Landing chips have a 6 Channel DDR4 memory controller capable of supporting upto 384 GB of memory with

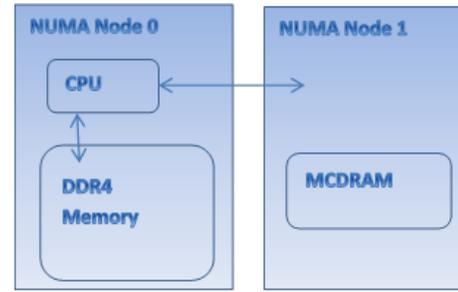


Fig. 3. MCDRAM in flat mode

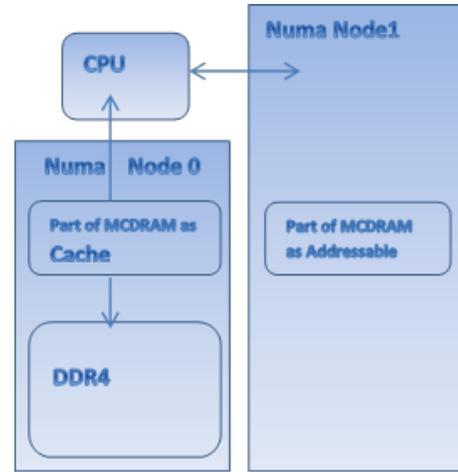


Fig. 4. MCDRAM in hybrid mode

speeds of 2400 MHz. This memory provides a speed of about 90GB/sec of bandwidth.

### C. Clustering Mode

The programmer has to maintain the locality of messages to achieve the lowest latency and greatest bandwidth of communication with the caches. Knl supports all-to-all, quadrant/hemisphere and sub-NUMA cluster SNC-4/SNC-2 modes of cache operation[7].

- All to All : In all-to-all clustering mode, memory addresses are uniformly distributed across all tag directories on the chip. This mode should not be used for day-to-day operations, but used for troubleshooting when other clustering modes cannot operate; for example, a missing memory module or faulty MCDRAM.
- Quadrant /Hemisphere: In the quadrant clustering mode, the tiles are divided into four sections called quadrants, which are spatially local to four groups of memory controllers. Memory addresses served by a memory controller in a quadrant are mapped to tag directories in the same quadrant. In Hemisphere mode, the die is divided into two hemispheres.
- SNC-4/SNC-2: The sub-NUMA cluster modes, SNC-4 and SNC-2 which divide the chip into four quadrants or two hemispheres as NUMA nodes. In this mode, NUMA-aware software can pin software threads to the same quadrant (hemisphere) that contains the tag directory and

accesses NUMA-local memory. If in SNC-4 or SNC-2 mode, cache traffic crosses NUMA boundaries, then this path is more expensive than in the quadrant mode. Knl in SNC-4 (-2) mode is similar to a 4-way (2-way) Intel Xeon processor. Sub-NUMA clustering is the recommended mode of operation for NUMA-aware applications.

### III. ABOUT APPLICATIONS

In this experiment, a parallel and a serial application have been used to test the system.

- **Parallel Application:** Gromacs is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with lots of particles. It is designed for biochemical molecules like proteins, lipids and nucleic acids. Gromacs has closely bound interaction and is extremely fast at calculating the non-bonded interactions[13]. Gromacs is a compute bound application. Gromacs takes the advantage of vectorization in most of their central kernels. However; these vectorized kernels usually do not include multi body potentials. Implementation methods for the kernels vary between hand written assembly, intrinsic and annotations that guide the compilers optimization. Furthermore, the majority of these kernels are not readily portable to different architectures. On the contrary, for each target architecture, separate optimizations are required[1]. Gromacs uses the Multiple-Program, Multiple-Data approach. Here, some ranks are selected to do only the Particle Mesh Ewald(PME) mesh calculation, while the other ranks, called particle-particle (PP) ranks, do rest of the work.
- **Serial Application :** Monte Carlo Algorithm : The directed loop algorithm [11] allows Monte Carlo sampling over complex configuration spaces, in terms of a large number of strongly constrained local degree of freedom and often encountered in the study of frustrated quantum and classical systems. The algorithm, in effect, filters the physical configuration samples, after sampling from an enlarged but computationally simpler space of configurations that violate the physical constraints. The specific implementation tested here implements the directed loop algorithm in the background of a finite density with constraint violations [10].

### IV. EXPERIMENTAL DETAILS

In this paper, we have run the molecular dynamic application Gromacs as well as Monte Carlo code on xeon processor and second generation xeon phi (Knl) processor. Table I shows the Knl System configuration on which the test cases has been conducted.

**Test Case I:** Gromacs has been run on Platform memory i.e. DDR4 with different combination of pp and pme node. In this test case, firstly Gromacs was compiled with higher optimization flag -O3 and secondly with -XMIC-AVX512 Advanced Vector Extension 512 bit vector processing unit especially designed for Knl architecture[6]. From the Table II,

TABLE I  
KNL SYSTEM CONFIGURATION

System Elements	Parameters
Core/Socket	68
Threads/core	4
Frequency	1.4GHz
DDR4	116GB
MCDRAM	16GB

it is observed that there is some improvement in the performance with AVX512 flag. Additionally, the best combination of threads for pp and pme nodes have been used.

TABLE II  
GROMACS RUN ON DDR4

No of Processes	Decomposition	Flags	Time in Min
272	pp=168 and pme=104	-O3	90
272	pp=168 and pme=104	-O3, -xmic-avx512	88
272	pp=200 and pme=72	-O3, -xmic-avx512	88
136	pp=84 and pme=52	-O3, -xmic-avx512	87
136	pp=100 and pme=36	-O3, -xmic-avx512	84
136	pp=90 and pme=46	-O3, -xmic-avx512	84

**Test Case II:** Run Gromacs on package memory i.e. on MCDRAM. There are three cases to determine how to run the application on HBM[8].

- If the application is able to fit into 16GB, then directly run it on HBM.
- If the application is too large to fit into 16GB, then partition it and put the bandwidth critical portion on HBM while the rest is on DDR4.
- If the Application is too large and difficult to partition, use the default cache mode so that the application at least takes some advantages of HBM.

Gromacs is able to fit into 16GB HBM. Table III shows the results. We can utilize HBM as an addressable memory using two methods by setting affinity policy with the numactl tool and through the usage of special allocators in the memkind library. In this case, we have used numactl tool to run the application on HBM. E.g numactl -m 1 mpirun -np 272 gromacs-4.5.5/bin/mdrun -npme 72 -deffnm 455\_500ps. In

TABLE III  
GROMACS RUN ON MCDRAM

No of Processes	Decomposition	Affinity	Time in Min
272	pp=200 and pme=72	Default	76
136	pp=90 and pme=46	Default	82
272	pp=200 and pme=72	Scatter	76
272	pp=200 and pme=72	Compact	75
272	pp=200 and pme=72	Balanced	76

this example 455\_500ps is the input file and mdrun is the gromacs executable. We can observe, from Table II and Table III that, gromacs is performing better in HBM memory using all possible threads with compact thread affinity. Affinity is nothing, but placing threads such that there are minimum communication overheads.

**Test Case III:** Gromacs run on different clustering modes. In this case, we have changed the clustering mode to All2All,

TABLE IV  
GROMACS RUN ON DIFFERENT CLUSTERING MODE WITH MPI

No of Processes	Decomposition	clustering mode	Memory mode	Time in Min
136	pp=100 and pme=36	All2All	DDR4	84
272	pp=200 and pme=72	All2All	MCDRAM	75
272	pp=200 and pme=72	Quadrant	DDR4	87
272	pp=200 and pme=72	Quadrant	MCDRAM	74
272	pp=200 and pme=72	Hemisphere	DDR4	87
272	pp=200 and pme=72	Hemisphere	MCDRAM	75
272	pp=200 and pme=72	SNC-4	DDR4	87
272	pp=200 and pme=72	SNC-4	MCDRAM	197
272	pp=168 and pme=104	SNC-2	DDR4	75
272	pp=168 and pme=104	SNC-2	MCDRAM	134

Quadrant, Hemisphere, SNC-4 and SNC-2 while keeping memory mode constant as flat mode and taken the readings as shown in the Table IV. From this table, it is observed that Gromacs performs well in Quadrant clustering mode with High Bandwidth Memory in flat mode. In test case II, we have used different thread affinity i.e. compact, balanced and scatter and observe better results in compact affinity. Hence compact affinity in each clustering mode has been used. Table IV shows

TABLE V  
GROMACS RUN ON DIFFERENT CLUSTERING MODE WITH OPENMP

No of Processes	clustering mode	Memory mode	Time in Min
272	All2All	DDR4	91
272	All2All	MCDRAM	86
272	Quadrant	DDR4	90
272	Quadrant	MCDRAM	86
272	Hemisphere	DDR4	90
272	Hemisphere	MCDRAM	86
272	SNC-4	DDR4	88
272	SNC-4	MCDRAM	86
272	SNC-2	DDR4	90
272	SNC-2	MCDRAM	85

the reading when Gromacs is run in MPI mode and Table V shows the reading when Gromacs is run in OpenMP mode. If we compare both the tables, it is observed that in some clustering mode MPI implementation performs better and in others, OpenMP implementation performs better.

**Test Case IV:** Gromacs has been run on a single server of Xeon processors namely Haswell and Broadwell. Haswell server is a 18 core dual socket with 2.30Ghz frequency. Similarly Broadwell server is a 14 core dual socket with 2.40Ghz frequency.

TABLE VI  
GROMACS RUN ON HASWELL & BROADWELL

No of Processes	Decomposition	Time in Min
36	pp=28 and pme=8	55
28	pp=21 and pme=7	60

From the Table VI, it is observed that Gromacs performs better on both the Haswell and Broadwell Xeon processors, though the frequency and numbers of cores are different. Fig 5 shows the performance of Gromacs on Haswell, Broadwell and Knl server. Without any optimization and vectorization the performance of Gromacs on Knl is comparable with Haswell or Broadwell, although there is a difference in the frequencies

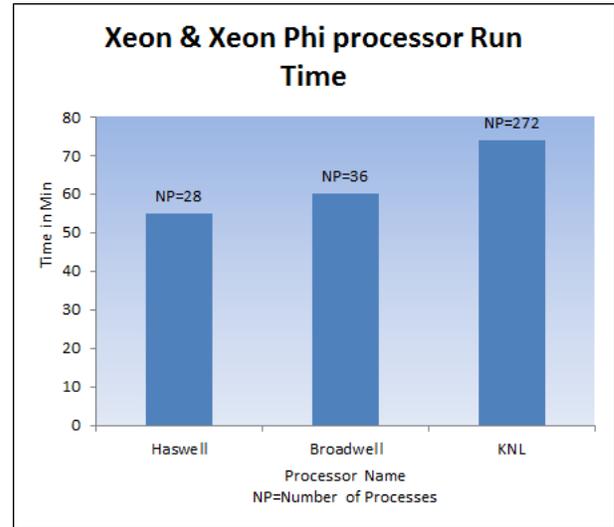


Fig. 5. Gromacs Performance on Xeon and Knl Processor

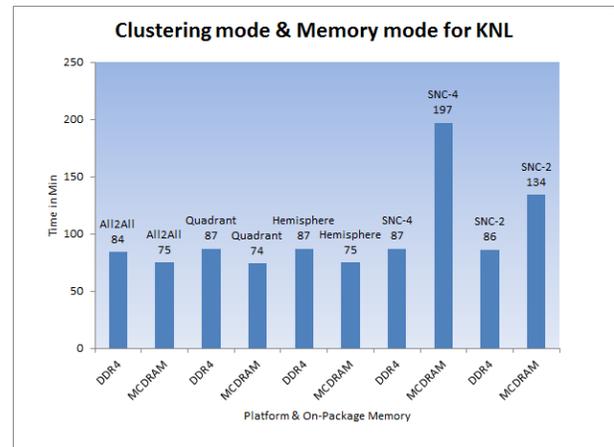


Fig. 6. Gromacs Performance on Knl Clustering mode

of both the architectures. Fig. 6 depicts the Gromacs performance on Knl using different clustering modes. Since Gromacs is able to fit into 16GB of HBM, the system is configured in flat memory mode. Best results were achieved in Quadrant clustering mode with 272 cores in HBM. From the above test cases, it is seen that many core performance may not be better always, but is definitely comparable to the multi core alternatives[12].

TABLE VII  
MONTE CARLO SIMULATION ON HASWELL AND KNL

Processor name (number of threads)	Number of iterations per sec per threads	Total number of iterations of all threads per sec
Haswell(36)	78	2750
Knl(68)	23	1600
knl(272)	11	3250

**Test Case V :** Monte Carlo Algorithm has been executed on Knl and Haswell machine with a single thread. Multiple

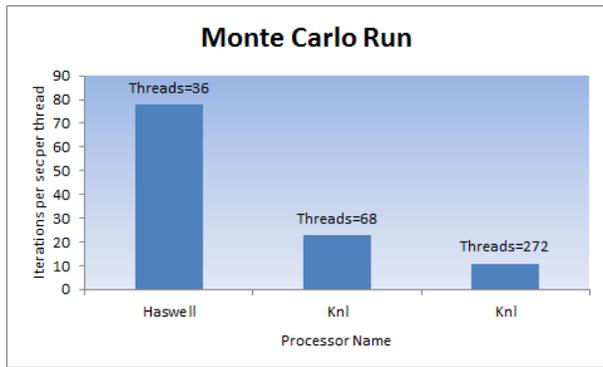


Fig. 7. Monte carlo run for iterations per sec on each thread

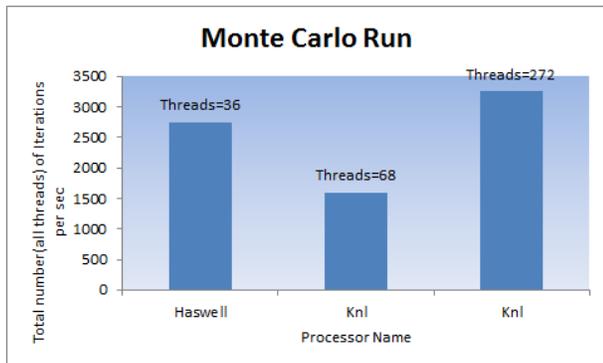


Fig. 8. Monte carlo run for total number of iterations for all threads per sec

copies were run on available threads to arrive at an average. Table VII shows the iterations performed per threads and total number of iterations of all the threads per sec. From Fig. 7 and Fig. 8, it is observed that, the number of iterations executed per thread on Haswell is more as compared to Knl because of the frequency. The total number of iterations of all threads per sec are more in knl as compared to Haswell, due to more number of threads in Knl.

## V. CONCLUSION

In these experiments, no specific optimization to Knl has been incorporated for Gromacs. The speedup was achieved by compiling with AVX512 and utilizing high bandwidth memory. Technological advancements in cpu architecture are evolving rapidly and it is becoming a challenge to ensure minimal power consumption. Many core architecture such as Knl delivers absolute performance due to generic programming models for majority of the scientific codes. From the above experiments, it has been observed that Gromacs application is memory bound and performs well on high bandwidth memory. There is a lot of scope to optimize, vectorize and rewrite the algorithm for the Knl architecture to extract best of performance. In case of the Monte Carlo algorithm, the performance is excellent for a small data set i.e. iterations per second per thread. In the case of larger repetitive runs i.e. number of iterations for all the threads, the Knl outperforms due to high vectorization traits of the processor. Finally, we get the speedup if we achieve a reasonable degree

of computational load with the advantages of cross platform code reuse and optimizations.

## VI. FUTURE WORK

The entire experiment has been carried out on a single server having a Knl processor. Hence the performance within a box has been elucidated. However for large scale clusters, it would be interesting to observe the the behavior of Gromacs as well as Monte Carlo when the vectorizable portion is executed on a large chunk of Knl nodes.

## VII. ACKNOWLEDGEMENT

The simulations were performed on the Knl machine provided by Intel.

## REFERENCES

- [1] M. Hohnerbach, Ahmed E, P. Bientinesi, The Vectorization of the Tersoff Multi-Body Potential: An Exercise in Performance Portability, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2016; arXiv:1607.02904 [Cs.CE]
- [2] A. Sodani, R. Gramunt, J. Corbal, H.-S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y.-C. Liu, Knights Landing: Secondgeneration Intel Xeon PhiTM product, IEEE Micro, vol. 36, no. 2, pp. 3446, 2016.
- [3] T. Barnes, B. Cook, J. Deslippe, D. Doerer, B. Friesen, Y. (Helen) He, T. Kurth, T. Koskela, M. Lobet, T. Malas, L. Oliker, A. Ovsyannikov, A. Sarje, J. Vay, H. Vincenti, S. Williams, P. Carrier, N. Wichmann, M. Wagner, P. Kent, C. Kerr and J. Dennis, Evaluating and Optimizing the NERSC Workload on Knights Landing, 2016 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems
- [4] A. Malhanov, M. Chuvelev, A. Biller, Optimizing PARSEC for Knights Landing, EuroMPI16, September 2528, 2016, Edinburgh, the United Kingdom.
- [5] A. Heinecke, A. Breuer, M. Bader, and P. Dubey, High Order Seismic Simulations on the Intel Xeon Phi Processor (Knights Landing), High Performance Computing, Volume 9697 of the series Lecture Notes in Computer Science pp 343-362, 15 June 2016
- [6] B. Zhang, Guide to Automatic Vectorization with Intel AVX-512 Instructions in Knight Landing Processors, Colfax International May 11, 2016, .
- [7] A. Vladimirov and R. Asai, Clustering Mode in Knight Landing Processors: Developer's guide, Colfax International May 11, 2016
- [8] R. Asai, MCDRAM As High Bandwidth Memory(HBM) in Knight Landing Processor: Developer's guide, Colfax International May 11, 2016,
- [9] D. Doerfler, J. Deslippe, S. Williams, L. Oliker, B. Cook, T. Kurth, M. Lobet, T. Malas, J.-L. Vay, and H. Vincenti, Applying the Roofline Performance Model to the Intel Xeon Phi Knights Landing Processor, in Proceedings of the ISC 2016 IXPUG Workshop, June 2016
- [10] GJ. Sreejith and S. Powell, Critical behavior in the cubic dimer model at nonzero monomer density, Phys. Rev. B 89, 014404 (2014)
- [11] O.F. Syljuasen and A.W. Sandvik, Quantum Monte Carlo with directed loops, Phys. Rev. E66, 046701 (2002)
- [12] A. Gomez-Iglesias, F. Chen, L. Huang, H. Liu, S. Liu, A. Lamas-Linares, J. Cazes, C. Rosales, Performance of Popular HPC Applications on the Intel Knights Landing Platform, Super Computing 2016.
- [13] <http://www.gromacs.org/>