doi:10.3772/j.issn.1006-6748.2019.02.001

PhiBench 2.0: characterizing data analytics workloads on Intel Knights Landing①

Xie Biwei(解壁伟)[*,**], Zhan Jianfeng②[*,**], Wang Lei[*], Zhang Lixin[*]
([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P. R. China)
([**] University of Chinese Academy of Sciences, Beijing 100049, P. R. China)

Abstract

With high computational capacity, e.g. many-core and wide floating point SIMD units, Intel Xeon Phi shows promising prospect to accelerate high-performance computing (HPC) applications. But the application of Intel Xeon Phi on data analytics workloads in data center is still an open question. Phibench 2.0 is built for the latest generation of Intel Xeon Phi (KNL, Knights Landing), based on the prior work PhiBench (also named BigDataBench-Phi), which is designed for the former generation of Intel Xeon Phi (KNC, Knights Corner). Workloads of PhiBench 2.0 are delicately chosen based on BigdataBench 4.0 and PhiBench 1.0. Other than that, these workloads are well optimized on KNL, and run on real-world datasets to evaluate their performance and scalability. Further, the microarchitecture-level characteristics including CPI, cache behavior, vectorization intensity, and branch prediction efficiency are analyzed and the impact of affinity and scheduling policy on performance are investigated. It is believed that the observations would help other researchers working on Intel Xeon Phi and data analytics workloads.

Key words: Intel Xeon Phi, data analytics workloads, characterization, Knights Landing (KNL), many core, x86 processors

0 Introduction

As the volume of data explodes[1,2], there are new challenges emerging for the speed of data processing of many data analytics workloads, like graph computing[3] and machine learning[4]. On one side, these data analytics workloads show much different characteristics from each other. On the other side, many new architectures have been proposed to accelerate workloads from different perspectives, such as APU[5], GPGPU[6] and Intel Xeon Phi[7]. Among them, Intel Xeon Phi is well known for its high parallelization and vectorization potential, as well as the backward compatibility with the traditional x86 platforms.

Some prior work has shown that Intel Xeon Phi is very useful for high-performance computing (HPC) applications[8]. But is it suitable to the data analytics workloads too? Considering the difference between the HPC applications and data analytics applications[2], the answer is not straightforward. Characterization of data analytics workloads on Intel Xeon Phi is motivated.

Current benchmarks, for example, BigDataBench[1] and DCBench[2], which consist of many big data workloads, are popularly used for benchmarking data analytics workloads. But, their workloads are built upon many big data frameworks, like Hadoop, Spark, etc. These frameworks affect the architecture behavior of these workloads a lot and prohibit us from understanding the intrinsic characteristics of these workloads. There is also some benchmark work targeting heterogeneous platforms based on openCL, but it is believed that these openCL workloads cannot exploit the best potential of the platform.

Though there is already some work that optimizes or evaluates some data analytics workloads on Intel Xeon Phi, to the best of our knowledge, there is still no systematic work to fill this gap. The prior work PhiBench[9] performs the characterization of data analytics workloads on the former generation of Intel Xeon Phi (KNC, Knights Corner)[10]. But, the latest generation of Intel Xeon Phi (KNL, Knights Landing) is much dif-

① Supported by the National High Technology Research and Development Program of China (No. 2015AA015308), the National Key Research and Development Plan of China (No. 2016YFB1000600, 2016YFB1000601) and the Major Program of National Natural Science Foundation of China (No. 61432006).

② To whom correspondence should be addressed. E-mail: zhanjianfeng@ict.ac.cn

Received on Apr. 27, 2018

ferent from KNC on many different perspectives, e.g. core architecture, ISA, and MCDRAM[11]. PhiBench needs improvement and should be extended to KNL to satisfy new research requirements. In this work, PhiBench 2.0 is built based on PhiBench[9] and a comprehensive characterization of data analytics workloads on Knights Landing is performed.

The following contributions are claimed.

1) PhiBench 2.0 is built on KNL, the latest generation of Intel Xeon Phi. PhiBench 2.0 covers seven workloads from six application domains and is fully optimized to take good advantage of KNL.

2) A comprehensive analysis of architectural behaviors of data analytics workloads on KNL, including CPI, cache behavior, vectorization intensity, and branch prediction efficiency are conducted.

3) PhiBench 2.0 is evaluated to investigate the impact of affinities and scheduling policies on performance and scalability.

Many-core processors were evolved from multi-core CPU designs to provide applications with higher parallel power. Intel Xeon Phi is one of the widely used many-core processors. The rationale for Xeon Phi is to allow certain applications that run well on existing multi-core CPUs, like Xeon, could gain performance directly with little or no modification. But the unique architecture of Xeon Phi determines that it still takes much effort to optimize the code to take the full advantage of this many-core processor.

The latest generation of Intel Xeon Phi (code-named KNL, or Knights Landing) evolves from KNC. KNL includes 681.4 GHz cores connected by a 2D mesh. It has a small cache hierarchy, which consists of only L1 and L2. Every two cores share the same 1MB L2 cache. Each KNL core has one 512-bit wide vector processing unit (VPU) and four hardware threads with up to 32 registers per thread context. There are two types of memory on KNL: multi-channel DRAM (MCDRAM) and DDR4 memory. MCDRAM can be configured to three modes: cache mode, flat mode and hybrid mode. KNL introduces AVX-512, which provides 512-bit-wide vector instructions. Moreover, it supports non-continuous memory read/write with gather/scatter instructions. The large number of hardware threads and wide SIMD unit is the main source of high computational power of KNL.

1 Related work

In this section, some related characterization work is investigated, and existing benchmark suites are compared with PhiBench in Table 1. Some benchmarks have been proposed targeting big data workloads, for example, BigDataBench and DCBench. But they are built upon complicated frameworks, such as Hadoop and Spark, which would prohibit us from understanding the intrinsic characteristics of these workloads.

Table 1 Benchmark comparison

Benchmark Effort	Workload Variety	Optimized on Phi	Software Stack	Platform
BigDataBench CloudSuite	Data center workloads	No	Hadoop Spark	Cluster
DCBech	Data analytics workloads	No	Hadoop Spark	Cluster
MineBench	Data mining algorithms	No	openMP	CMP
Parsec	Mainly HPC workloads	No	Pthreads and openMP	CMP
Rodinia SHOC Parboil Valar	Mainly HPC workloads	No	openCL and CUDA	GPGPU
SHOC-MIC	Mainly HPC workloads	Yes	openMP	Xeon Phi(KNC)
PhiBench	Data analytics workloads	Yes, on KNC	openMP	Xeon Phi(KNC)
PhiBench 2.0	Data analytics workloads	Yes, KNC & KNL	openMP	Xeon Phi(KNL)

Parsec[12] and NAS Parallel Benchmark (NPB)[13] are designed for CMP systems without complicated software stack. But they mainly cover HPC workloads instead of data analytics workloads, and they can not take the good advantage of Xeon Phi, due to the lack of appropriate optimization on Intel Xeon Phi. Motivated by research of heterogeneous systems, some benchmark suites have been proposed based on OpenCL and CUDA, such as Parboil[14], Rodinia[15], Valar[16], and SHOC[17,18]. These benchmarks also can not represent data analytics workloads and exploit the best potential of Xeon Phi.

There are many other efforts to optimize and evaluate specific systems or applications on KNL. The work pays particular attention on a specific problem[19,20] and optimize them in details to exploit the best potential of Xeon Phi (KNL). To the best knowledge, there is still no suitable benchmark for characterizing data analytics workloads on KNL. A new benchmark suite that can represent data analytics work-

loads and fully optimized on KNL is motivated.

2 Benchmark methodology

PhiBench 2.0 is built based on the prior work, PhiBench, and also the well-known benchmark BigdataBench[1] is referred to choose the workloads, which cover six application domains: Graph Calculation, Clustering, Classification, Recommendation, Sorting and Irregular Kernel. The main difference between PhiBench 2.0 and PhiBench is the platform, where PhiBench 2.0 works on KNL and PhiBench works on KNC. Given the architecture difference of KNL and KNC, the optimization method and according workload implementation are quite different. Additionally, the latest version of BigdataBench is referred and SpMV is added in PhiBench 2.0

Moreover, to get the real-world characteristics of these workloads, real-world dataset is used as long as possible. For some workloads, MovieLens[21], Google Web Graph[22], livejournal[22], are used. For other workloads, the real-world dataset is enlarged to fulfill their requirements.

2.1 Chosen workloads

In this section, a brief introduction of the workloads included in the PhiBench 2.0 will be given.

Graph Algorithms PageRank is a representative graph algorithm, which is widely adopted as benchmark workloads in much graph computing work. Based on iterative computation, PageRank converges only when the PageRank value of the current iteration is less than a specific threshold.

Clustering Clustering algorithm, like K-means, is widely used in image analysis, bioinformatics, and pattern recognition. K-means iteratively calculates the distance between the object and the centroids and always makes sure that the object is clustered to the nearest centroid. When the cluster of the points becomes stable, K-means converges.

Classification Classification is one of the most important algorithm domains in machine learning and pattern recognition. Naive Bayes and support vector machine (SVM) are widely adopted in real world applications. SVM is a supervised learning model that can map its inputs into high-dimensional feature spaces, while Naive Bayes is a simple probabilistic classifier using naive independence assumptions based on the conditional probability model.

Recommendation In electronic business and social network, recommendation algorithm, like collaborative filtering is often used to recommend items to customs according to the purchase history and favorite items of a large number of users. The kernel of collaborative filtering is to compute the similarity of different items or users, and sort them to get the top-k favorite items for a specific user.

Sorting Sorting is a basic operation of many applications in data center. It is generally I/O intensive. As too many sort algorithms exist, the well-known merge-sort is chosen in our benchmark.

Irregular Kernel Sparse matrix-vector multiplication (SpMV)[23] is an important computational kernel in sparse linear system solvers and real-world applications from both data center and HPC area. The characteristics of SpMV represent many other real-world applications. There is many existing work on SpMV based on different storage formats. CSR5[24] based SpMV algorithm is used for characterization.

In total, seven workloads are chosen to build PhiBench 2.0. these workloads and their respective datasets are summarized in Table 2.

Table 2 Chosen Workloads of PhiBench 2.0

Workload	Application Domain	Application Scenario	Datasets	Data Size
PageRank	Graph Computing	Search Engine	Web-Google	Size: 87M 875K nodes, 5.1M Edges
K-means	Clustering	Data Mining Image Processing	Generated	Size: 2.2G
SVM	Classification	Image Processing, Pattern Recorgnition	Generated	Size: 1.2G
Naive Bayes				Size: 1.8G
Collaborative filtering (CF)	Recommendation	Electronic Commerce Social Network	MovieLens-100K	Size: 17M 1k users, 1.7k movies
Sort	Sorting	Basic Operation	Randomized	Size: 1G
SpMV	Data Center and HPC	Basic Kernel	LiveJournal	Size: 1.1G 4.8M nodes, 69M edges

2.2 Optimization

Intel Xeon Phi is a quite different architecture. KNL has up to 272 hardware threads, much more than the existing x86 based platform and 512-bit-wide SIMD unit. Workloads in PhiBench 2.0 need to be well optimized on KNL, so that they can exploit good advantage of the platform.

Some of the workloads are obtained from the existing work, like Sort[25], SVM[26] and SpMV[24], or open-source projects, like CF and PageRank. Others are parallelized with openMP and vectorized with SIMD intrinsic statements. Considering the poor serial processing capability of KNL, delicate efforts are dedicated to improve the parallel performance as much as possible. Moreover, the memory region is aligned for better memory reference efficiency.

3 Experiment setup

PhiBench2.0 is characterized on the latest Intel Xeon Phi platform, Knights Landing (KNL)7250. Table 3 summarizes the detail configurations of KNL. There are two kinds of memory on KNL: MCDRAM and DDR4, supporting three modes to configure them: flat mode, cache mode, and hybrid mode. In the experiments, flat mode and 'numactl-membind = 1' are used to bind the workload to MCDRAM. '-O3-qopenmp' is used to compile the workloads. Thread numbers are set through "OMP _ NUM _ THREADS", and "KMP _ AFFINITY, OMP _ SCHEDULE" are used to set the affinities (scatter, compact and scatter) and schedule policies (dynamic, static and guided).

Table 3 Description of hardware platform

CPU Type	Knights Landing 7250
Number of cores	68
Number of threads/core	4
Basic frequency (GHz)	1.40
Vector instruction set	AVX2 and AVX512
SIMD vector width	512
L1 cache size/core	32 kB(I) +32 kB (D)
L2 cache size	1 024 kB(shared by two cores)
Memory type	MCDRAM and DDR4
Memory size	96GB and 16GB
Peak bandwidth	400GB/s and 115GB/s
OS	CentOS 7.2 (Kernel 3.10.0)
Compiler	Intel ICC 17.0

To collect the running time, the wall-time is used to run the workloads multiple times to take the average value. The profiling tool used is Intel VTune (Amplifier XE 2017). All the metrics used in the experiments are inspired from the developer's manual of Xeon Phi listed in Table 4.

Table 4 List of metrics

Metrics	Fomula
IPC	CPU _ CLK _ UNHALTED. THREAD/ INST _ RETIRED. ANY
L1 miss ratio	MEM _ UOPS _ RETIRED. L1 _ MISS _ LOADS /MEM _ UOPS _ RETIRED. ALL _ LOADS
L2 miss ratio	MEM _ UOPS _ RETIRED. L2 _ HIT _ LOADS /(MEM _ UOPS _ RETIRED. L2 _ HIT _ LOADS + MEM _ UOPS _ RETIRED. L2 _ MISS _ LOADS)
Vectorization Intensity	UOPS _ RETIRED. PACKED _ SIMD / (UOPS _ RETIRED. PACKED _ SIMD + UOPS _ RETIRED. SCALAR _ SIMD)
Branch miss ratio	BR _ MISP _ RETIRED. ALL _ BRANCHES/ BR _ INST _ RETIRED. ALL _ BRANCHESs

4 Experiment results

In this section, performance and the architectural characteristics of data analytics workloads on Intel Xeon Phi are presented in detail. Impact of SMT on performance is analyzed firstly. Then, the CPI, cache behavior, vectorization efficiency and branch prediction efficiency are characterized. At last, PhiBench 2.0 under different openMP configurations is evaluated (thread numbers, affinities, and schedule policies) to see their impact on the performance.

4.1 SMT

The thread of each core to 1, 2, 3 and 4 respectively is set to evaluate the impact of SMT on performance. In Fig. 1, it can be seen that most workloads could gain good scalability on more hardware threads,

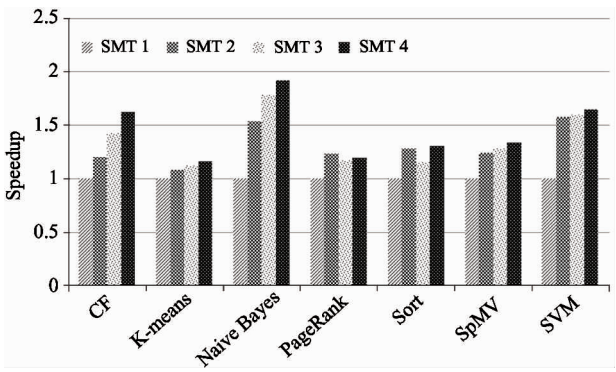

Fig. 1 SMT performance (normalized to 1HT)

especially CF, Naive Bayes and SVM. Some others show modest performance gain, like K-means, PageRank, Sort and SpMV. So turning on the SMT on Intel Xeon Phi (KNL) would benefit the workloads effectively.

4.2 CPI

Cycles per instruction (CPI) is an important metric to measure the utilization of hardware. Fig. 2 shows the CPI of each workload in Phi Bench 2.0. CF. PageRank and SpMV has high CPI, while K-means, Naive Bayes, Sort and SVM shows much lower CPI. For all workloads, the CPI is higher when number of threads is larger. The higher CPI could be caused by the high memory access latency. Section 4.3 shows that the workloads with high CPI also shows high L2 cache miss ratio, which means that these worklaods need to load data from memory frequently.

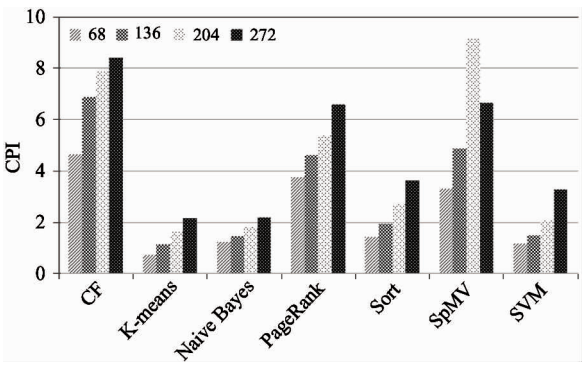

Fig. 2 CPI (cycles per instruction)

4.3 Cache locality

To characterize the data reference efficiency of these workloads, their behaviors on L1 and L2 cache are collected. Fig. 3 figures out that, most workloads achieve quite good L1 cache locality, with less than 5% L1 cache miss ratio.

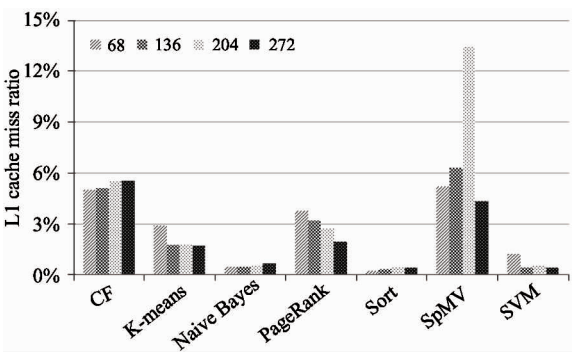

Fig. 3 L1 cache miss ratios

LLC cache miss is more expensive than L1 cache miss, since it needs to be filled by loading data from memory, which has much higher latency. So here the cache miss ratio of L2 cache is considered which is also the last level cache (LLC) on Intel Xeon Phi (KNL). Fig. 4 shows that CF, PageRank, and SpMV have large L2 cache miss ratio, which makes the memory latency bound. On the other hand, the high memory access latency is the main reason of higher CPI, for more cycles are needed to prepare the data. The reason of the poor L2 cache locality of CF, PageRank and SpMV is that they need to deal with sparse matrices. More efficient data layout may improve their performance profoundly.

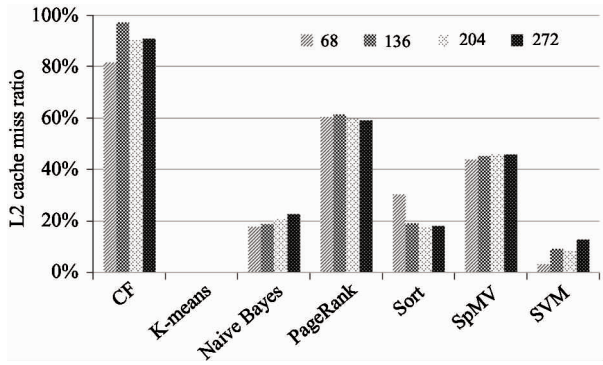

Fig. 4 L2 cache miss ratio

4.4 Vectorization

Vectorization is the main power of KNL, which introuduces AVX512 for better vectorization support. The vectorization intensity of these workloads is characterized to see whether they could take good utilization of the vectorization processing units.

As Fig. 5 shows, K-means, Naive Bayes, Sort and SpMV achieve nearly ideal vectorization intensity. It is found out that these workloads are well optimized by intrinsic statements manually or auto-vectorized by the compiler. As for other workloads, like CF and Pagerank, some reduction statements prohibit them from better vectorization. SpMV has similar algorithm pattern to CF and Pagerank, but has much higher vectorization intensity. Because the SpMV workload (CSR5)

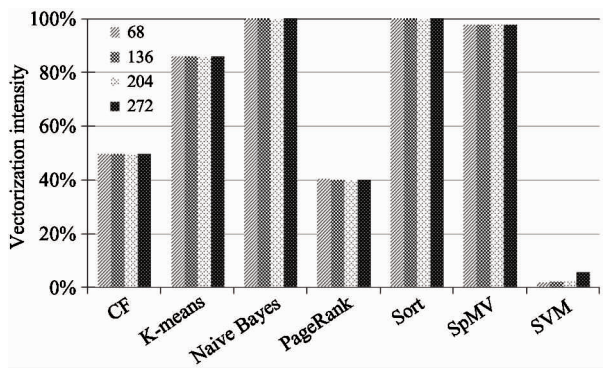

Fig. 5 Vectorization intensity

used is delicately optimized for vectorization. SVM has the lowest vectorization intensity. Though it is well optimized, there is still lots of scalar instructions existing. Further investigation is made to the workloads with high vectorization intensity, like Naive Bayes, Sort, and SpMV, to find that they are well auto-optimized by the compiler or hand-optimized.

4.5 Branch prediction efficiency

Further look is taken into the branch prediction efficiency by examining the branch miss ratio. Fig. 6 shows that PageRank and SpMV suffer from high branch miss ratio. It is worth noting that branch prediction design on KNL is much different from KNC. On KNC, Naive Bayes and K-means both suffer from high branch misses[9]. But on KNL, these two workloads work well.

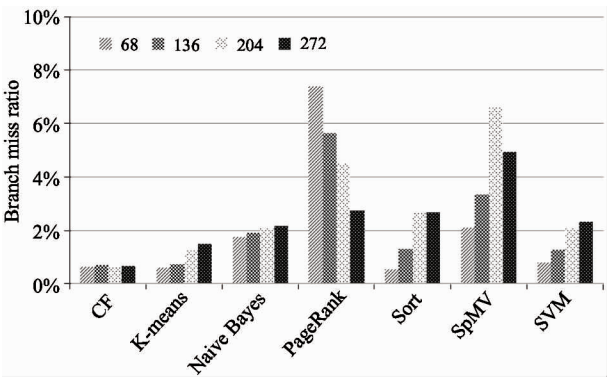

Fig. 6 Branch miss ratio

4.6 Parallel policies

To further analyze the impact of affnity and schedule policy on perfomance and scalability, experiments with different parallel policies are conducted. The thread counts are set to 68, 136, 204 and 272, and three schedule polices: dynamic, guided and static. As for the affinity, only balanced and scatter are used, since compact is not so suitable in the proposed expeirments (with 'compact', all threads would be compacted into a small set of cores, while other cores are idle).

The execution time of these workloads with different thread counts, affinities, and schedule polices is listed in Table 5. It can be figured out that affinities and scheudle polices do not affact the performance heavily on KNL, a similar conclusion is got in Ref. [9]. But it is still recommanded to run these workloads with optimal configuration. It is an open problem to find the best configuration automatically, without running them one by one.

Table 5 Execution time (in seconds) with different affinities and scheduling policies (Entries in read show the best performance for each workload. B stands for balanced, while S stands for scatter)

Workload	Thread Counts	B-Dynamic	B-Guided	B-Static	S-Dynamic	S-Guided	S-Static
CF	68	252.18	249.40	251.54	258.57	248.98	249.99
	136	190.11	191.39	192.92	192.19	195.68	191.71
	204	160.96	160.47	162.02	161.67	161.72	161.59
	272	147.87	148.33	146.21	146.61	147.66	148.24
K-means	68	7.70	7.81	7.73	8.57	7.66	7.71
	136	7.02	7.08	7.12	7.06	7.07	7.02
	204	6.81	6.80	6.82	6.78	6.76	6.81
	272	6.89	6.77	6.82	6.74	6.85	6.83
Naive Bayes	68	13.60	13.68	13.68	13.64	13.66	13.93
	136	8.82	8.82	8.77	8.79	8.78	8.81
	204	7.61	7.59	7.60	7.55	7.66	7.65
	272	7.15	7.24	7.19	7.18	7.22	7.30
PageRank	68	5.60	5.45	5.58	5.49	5.46	5.26
	136	4.61	4.69	4.83	4.46	4.36	4.74
	204	3.94	4.09	4.02	4.17	4.48	4.15
	272	4.23	4.34	4.17	4.63	4.24	4.47
Sort	68	168.15	168.35	169.00	168.44	168.46	168.09
	136	130.79	131.10	130.77	131.85	131.33	131.86
	204	168.69	167.63	168.39	146.51	147.68	149.26
	272	151.17	151.32	150.51	130.21	130.60	126.36

Table 5 Contiuned

SpMV	68	22.28	22.12	21.95	22.11	22.34	22.02
	136	17.83	17.79	17.97	17.90	17.97	18.19
	204	16.67	16.54	16.69	17.21	17.03	17.26
	272	18.65	17.97	18.41	19.22	18.99	19.10
SVM	68	115.12	115.15	115.15	115.35	115.35	115.21
	136	75.52	75.76	75.20	74.98	75.30	75.20
	204	72.41	72.54	72.55	72.59	72.66	72.66
	272	74.79	75.79	74.53	72.33	74.76	74.60

5 Conclusions

This paper presents a new benchmark on KNL, PhiBench 2.0, based on PhiBench, which is designed for KNC, the former generation of Intel Xeon Phi. Workloads in PhiBench 2.0 are optimized to take good advange of the parallelizaton and vectorization power on KNL. Moreover, a comprehensive characterization of workloads in PhiBench 2.0 is conducted to investigate their performance and micro-achitecture behaviors, e.g. CPI, cache locality, vectorization intensity, branch prediction efficiency, and the impact of parallel policies on performance. The expeirment results and according observations will help researchers to understand the characteristics of data analytics workloads on KNL.

References

[1] Wang L, Zhan J F, Luo C J, et al. BigDataBench: a big data benchmark suite from internet services[C]. In: Proceedings of the 20th IEEE International Symposium on High Performance Computer Architecture (HPCA), Orlando, USA, 2014. 488-499

[2] Jia Z, Wang L, Zhan J F, et. al. Characterizing data analysis workloads in data centers[C]. In: Proceedings of the IEEE International Symposium on Workload Characterization(IISWC), Portland, USA, 2013. 66-76

[3] Zhou Y, Liu L, Lee K, et. al. GraphTwist: fast iterative graph computation with two-tier optimizations[C]. In: Proceedings of the VLDB Endowment, Hawaii, USA, 2015. 1262-1273

[4] Yavits L and Ginosar R. Accelerator for sparse machine learning. *IEEE Computer Architecture Letters*, 2017, 17(1):21-24

[5] Kyriazis G. Heterogeneous System Architecture: A technical review[EB/OL]. http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/hsa10.pdf, 2012

[6] Luebke D, Harris M, Govindaraju N, et al. GPGPU: general- purpose computation on graphics hardware[C]. In: Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques, Los Angeles, USA, 2004. 33-34

[7] Sodani A, Gramunt R, Corbal J, et al. Knights Landing: second-generation Intel Xeon Phi Product[J]. *IEEE Micro*, 2016,36(2): 34-46

[8] Kindratenko V, Trancoso P. Trends in high-performance computing[J]. *Computing in Science and Engineering*, 2011,13(3): 92-95

[9] Xie B, Liu X, McKee S, et al. Understanding data analytics workloads on Intel Xeon Phi[C]. In: Proceedings of IEEE 18th International Conference on High Performance Computing and Communications (HPCC), Sydney, Australia, 2016. 206-215

[10] Chrysos G. Intel Xeon Phi X100 family coprocessor—the architecture. white paper, Intel[EB/OL]. https://software.intel.com/en-us/artcles/intel-xeon-phi-coprocessor-codename-knights-corner: Intel, 2012

[11] Asai R. MCDRAM as high-bandwidth memory (HBM) in Kights Landing processors: developer's guide[EB/OL]. https://colfaxresearch.com/knl-mcdram: Colfaxresearch, 2016.

[12] Bienia C, Kumar S, Singh J P, et al. The PARSEC benchmark suite: characterization and architectural implications[C]. In: Proceedings of the ACM/IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT), Toronto, Canada, 2008. 72-81

[13] Bailey D H, Barszcz E, Barton J T, et al. The NAS parallel benchmarks — summary and preliminary results [C]. In: Proceedings of the ACM/IEEE International Conference on Supercomputing, New Mexico, USA, 1991. 158-165

[14] Stratton J A, Rodrigrues C, Sung I J, et al. Parboil: a revised benchmark suite for scientific and commercial throughput computing[R]. Technical Report IMPACT-12-01, University of Illinois at Urbana-Champaign, 2012

[15] Che S, Boyer M, Meng J, et al. Rodinia: a benchmark suite for heterogeneous computing[C]. In: Proceedings of the IEEE International Symposium on Workload Characterization (IISWC), Washington, USA, 2009. 44-54

[16] Mistry P, Ukidave Y, Schaa D, et al. Valar: a benchmark suite to study the dynamic behavior of heterogeneous systems[C]. In: Proceedings of the ACM Workshop on General Purpose Processor Using Graphics Processing Units, Houston, USA, 2013. 54-65

[17] Danalis A, Marin G, McCurdy C, et al. The scalable heterogeneous computing (SHOC) benchmark suite[C]. In: Proceedings of the ACM Workshop on General-Purpose Computation on Graphics Processing Units, Pitts-

burgh, USA, 2010. 63-74
[18] Vetter J S. The scalable heterogeneous computing benchmark suite (SHOC) for Intel Xeon Phi[EB/OL]. https://github.com/vetter/shoc-mic: Github, 2013
[19] You Y, Buluc A, Demmel J. Scaling deep learning on GPU and Knights Landing clusters[C]. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Denver, USA, 2017. 1-12
[20] Doerfler D, Deslippe J, Williams S. Applying the roofline performance model to the intel xeon phi knights landing processor[J]. *Lecture Notes in Computer Science*, 2016, 9945: 339-353
[21] Harper F, Konstan J. The movieLens datasets: history and context[J]. *ACM Transactions on Interactive Intelligent Systems*, 2016, 5(4): 1-19
[22] Leskovec J, Lang K J, Dasgupta A, et al. Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters[R]. CoRR, abs/0810.1355, 2008
[21] Xie B, Zhan J, Liu X, et al. CVR: efficient vectorization of SpMV on x86 processors[C]. In: Proceedings of the 2018 International Symposium on Code Generation and Optimization(CGO), Vienna, Austria, 2018. 149-162
[24] Liu W, Vinter B. CSR5: an efficient storage format for cross-platform sparse matrix-vector multiplication[C]. In: Proceedings of the 29th ACM International Conference on Supercomputing (ICS), Newport Beach, USA, 2015. 339-350
[25] Tian X, Rocki K, Suda R. Register level sort algorithm on multi-core SIMD processors[C]. In: Proceedings of the ACM Workshop on Irregular Applications: Architectures and Algorithms, Denver, USA, 2013. 1-8
[26] You Y, Song S, Fu H, et al. MIC-SVM: designing a highly efficient support vector machine for advanced modern multi-core and many-core architectures[C]. In: Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), Washington, USA, 2014. 809-818

Xie Biwei, born in 1987. He is a Ph. D candidate in computer science at Institute of Computing Technology, Chinese Academy of Sciences (University of Chinese Academy of Sciences). He received his M. S. degree in 2012 from Beijing University of Technology (BJUT) and B. S. degree in 2009 from Hebei University of Science and Technology. His research interest lies in benchmarking, parallel computing, and performance optimization.