Performance Portability Evaluation for OpenACC on Intel Knights Corner and Nvidia Kepler*

Yichao Wang [1], Qiang Qin [1], Simon Chong Wee SEE [1,2], James Lin [1,3]

[1] *Center for High Performance Computing, Shanghai Jiao Tong University, Shanghai 200240, China*

[2] *NVIDIA Corporation*

[3] *Tokyo Institute of Technoloy*

{wangyichao, charles.chin, simon.see, james}@sjtu.edu.cn

Abstract

OpenACC is a programming standard designed to simplify heterogeneous parallel programming by using directives. Since OpenACC can generate OpenCL and CUDA code, meanwhile running OpenCL on Intel Knight Corner is supported by CAPS HMPP compiler, it is attractive to using OpenACC on hardwares with different underlying micro-architectures. This paper studies how realistic it is to use a single OpenACC source code for a set of hardwares with different underlying micro-architectures. Intel Knight Corner and Nvidia Kepler products are the targets in the experiment, since they are with the latest architectures and have similar peak performance. Meanwhile CAPS OpenACC compiler is used to compile EPCC OpenACC benchmark suite, Stream and MaxFlops of SHOC benchmarks to access the peformance. To study the performance portability, roofline model and relative performance model are built by the data of experiments. This paper shows that at most 82% performance compared with peak performance on Kepler and Knight Corner is achieved by specific benchmarks, but as the rise of arithmetic intensity the average performance is approximately 10%. And there is a big performance gap between Intel Knight Corner and Nvidia Kepler on several benchmarks. This study confirms that performance portability of OpenACC is related to the arithmetic intensity and a big performance gap still exsits in specific benchmarks between different hardware platforms.

Keywords: OpenACC, Performance portability, GPU, MIC

1 Introduction

Heterogeneous parallel programming is widely used in scientific computing and engineering computing. Using directives such as OpenMP[3] can simplify programming on accelerators compared with low level programming such as CUDA[12] and OpenCL[5]. OpenACC is a new programming standard for parallel computing, whose Application Program Interface describes a collection of compiler directives to specify loops and regions of code in standard C, C++ and Fortran to be offloaded from a host CPU to an attached accelerator, providing portability across operating systems, host CPUs and accelerators.[6] Users can just add some directives such as "#pragma acc loop" before loop code to parallelize the loop in the program. Although it is lack of compute and data management in contrast to CUDA or OpenCL, high level and hybrid programming is the advantage of OpenACC. Users do not have to take much workload to maintain source code with different version to run on different platforms by OpenACC. So the most concerned problem about OpenACC is the performance portability. We focus on performance portability evaluation for OpenACC in this paper.

With the development of high performance computing, more devices are designed for heterogeneous parallel programming. Kepler is architecture of GPU released by Nvidia that consists of more than two thousands GPU cores designed for parallel computing. Nvidia K20c used in the experiment is the product of this architecture. Knight Corner is architecture that consists of up to 61 cores on one card, whose cores and architecture are absolutely different from GPU such as interconnect and memory. Knight Corner is called co-processor and Kepler is called accelerator, whose programming models are also different.

*Supported by NVIDIA Center of Excellence, HMPP CoC and JSPS RONPAKU Fellowship.

CAPS OpenACC Compiler[4] is used to compile source code to OpenCL target code so that benchmarks can run on cross-platform. EPCC OpenACC benchmark suite[8] consists of several algorithms widely used in scientific computing such as stencil, spmv and gaussian, which covers 5 problems of "Thirteen Dwarfs"[13]. In the experiment, at most more than 82% performance compared with peak performance on Kepler and Knight Corner is achieved by specific benchmarks, but average performance is about 10%. A big performance gap exists and lack of on-chip memory and cores control cause it in our opinion.

2 Evaluation Methodology

The goal of this paper is to make performance portability evaluation for OpenACC-based hybrid codes on two heterogeneous accelerators. We collect a series of benchmarks to measure the basic peak performance of devices by CUDA-based and OpenMP-based version and performance of applications ported on two devices by OpenACC-based version. Two evaluation models are built by the results of benchmarks, which describes the performance information by the image.

2.1 Benchmark

2.1.1 EPCC OpenACC Benchmark Suite

EPCC OpenACC Benchmark Suite is designed by EPCC at the University of Edinburgh. The suite is to test the performance of kernels and application commonly seen in scientific codes, which consists of gemm, gaussian, stencil and more. "Seven Dwarfs" is a well known role to evaluate whether the benchmark suite is good enough, which includes Dense Linear Algebra, Sparse Linear Algebra, Spectral Methods, N-Body Methods, Structured Grids, Unstructured Grids and Monte Carlo. According to this role, EPCC OpenACC benchmark suite covers three of them.
Since EPCC OpenACC Benchmark Suite was compiled by PGI compiler, but CAPS compiler is our choice. We have modified some details of the benchmark and added more output in order to print out more performance information.

2.1.2 Stream Benchmark

Memory bandwidth can be measured by Stream benchmark putting stress on memory transfer. All operation of the benchmark consists of the add, copy, scale and trial, which are basic operation of memory.[10] Stream benchmark was first designed by Prof. John McCalpin to get memory bandwidth of CPU, but as the growing of heterogeneous parallel computing CUDA and OpenMP implementation also appears. In the experiment, CUDA version is implemented to get memory bandwidth of GPU and OpenMP version is implemented to get MIC's memory bandwidth. According to the document by Intel, OpenMP version Stream benchmark can run under native mode.

2.1.3 SHOC Benchmark Suite

The Scalable Heterogeneous Computing Benchmark Suite is a collection of benchmark programs to test the stability of heterogeneous architecture systems.[11] SHOC benchmark suite supports CUDA, MPI, OpenCL and OpenMP. In the experiment, only MaxFlops benchmark is used to get the peak floating performance of two devices. MaxFlops benchmark consists of only addition, only multiplication, mix of addition and multiplication in equal proportions and mix of addition and multiplication in 1:2.

2.2 Evaluation Model

2.2.1 Roofline Model

One visual, intuitive way to compare potential floating-point performance of variations of SIMD architectures is the roofline model.[7] It ties floating-point performance, memory performance and arithmetic intensity in a two-dimensional graph. Arithmetic intensity is a concept introduced to show the ratio of floating point operations per byte of memory accessed. In the experiment, arithmetic intensity is calculated by all floating-point operation times divided by size of memory read and write, which is horizontal-axis of the model. Vertical-axis is the floating-point operation times per second and it can be double or single floating-point. Horizontal-axis and vertical-axis scale are both the logarithm of 2. In the model, roofline is a line that shows the potential floating-point performance, which is composed of two parts. A diagonal line at 45-degree angle is the first part of roofline. Since the vertical-axis is FLOP/sec and the horizontal-axis is FLOP/byte, slope of the first part is bytes/sec that gives the maximum of memory performance can be supported. The second part of roofline is a horizontal line which shows peak floating-point performance can be supported. So that the actual floating-point performance can not be higher than the roofline because of

hardware limit. A formula is built by the line: Attainable GFLOPs/sec = Min(Peak Memory Bandwidth × Arithmetic Intensity, Peak Floating-Point Performance).

2.2.2 Relative Performance Model

Simple model to compare the portability performance on two different platform by a couple of histograms on the same benchmark. So that the performance portability can be shown obviously from height difference between two histograms. In the experiment, horizontal-axis is the kind of benchmark and vertical-axis is the floating point performance and we just set the best performance as 100%.

3 Results and Discussion

3.1 Basic Peak Performance Results

Since the Roofline Model needs the peak performance of two devices to draw the roofline and theoretical peak performance is not absolutely accurate under the experiment environment, basic peak performance results were got by running Stream benchmark and MaxFlops of SHOC benchmark. Benchmarks of CUDA version ran on GPU and benchmarks of OpenMP version ran on Xeon Phi, because these two different version had deep optimization on two different devices so that the peak performance got by them is more accurate.

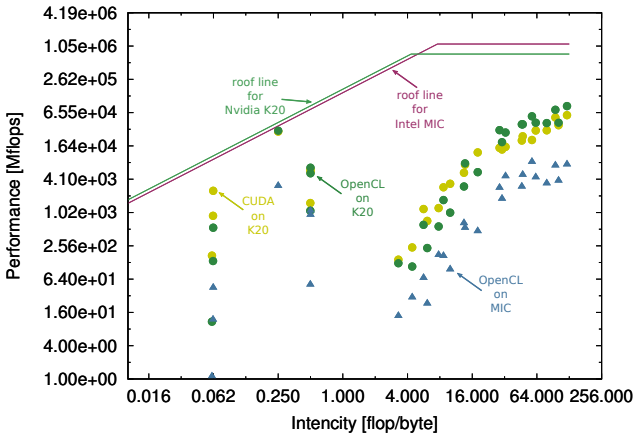
Table 1: Memory bandwidth results by Stream benchmark

Product	Theoretical Value (GB/s)	Measured Value (GB/s)	Percentage
Nvidia Tesla K20c	208	150	72.1%
Intel Xeon Phi 7110P	352	174	49.3%

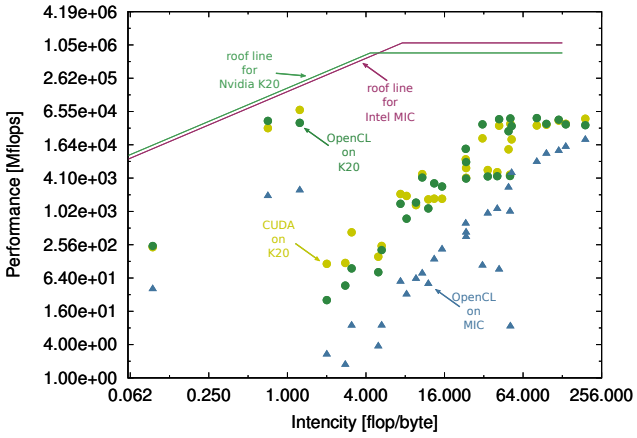
Table 2: Peak floating point performance results by MaxFlops of SHOC benchmark suite

Product	Precision	Theoretical Value (GFLOPS)	Measured Value (GFLOPS)	Percentage
Nvidia Tesla K20c	SP	3520	3006	85.4%
	DP	1170	1168	99.8%
Intel Xeon Phi 7110P	SP		1945	
	DP	1220	980	80.0%

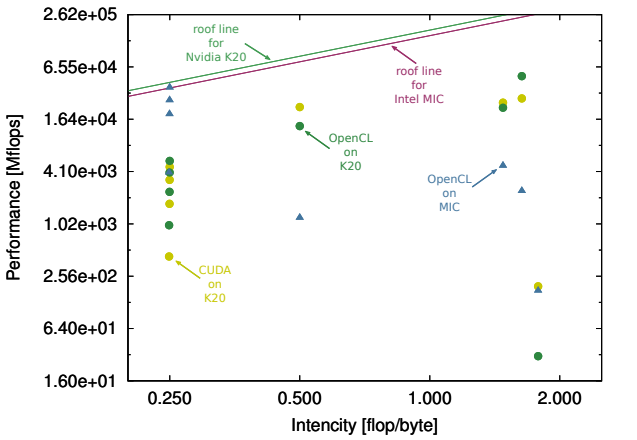
3.2 Result Models



(a)



(b)



(c)

Figure 1: Performance [Mflops] as a function of arithmetic intensity for benchmark method of level A (a) level B (b) and level C (c). Circles correspond to Nvidia K20 device, with CUDA coded data in yellow and OpenCL data in green. Blue triangles correspond to Intel MIC device using OpenCL. Roof line for K20 and MIC are represented by light green and purple lines respectively.

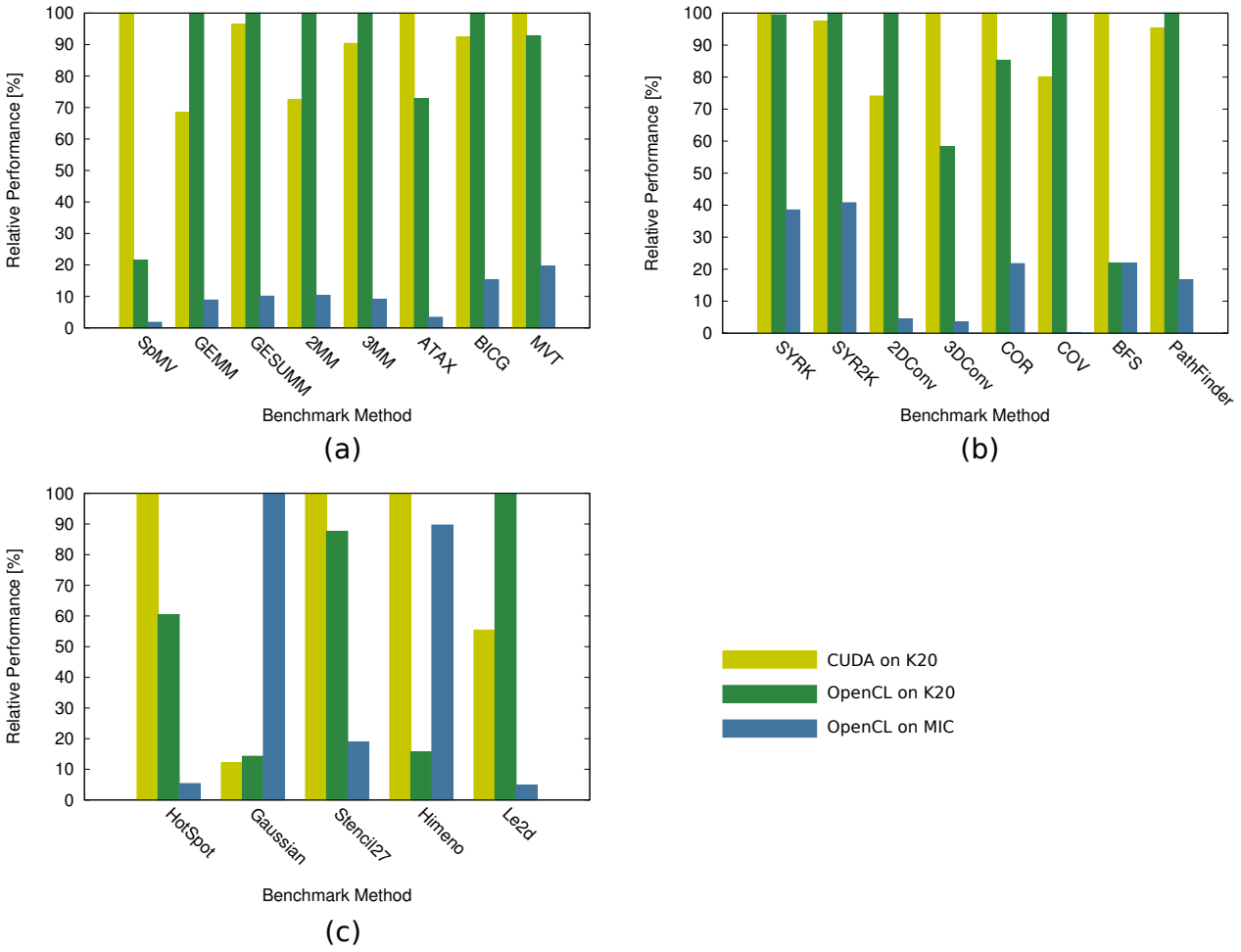


Figure 2: Maximum performance of CUDA on K20 and OpenCL on K20 and MIC for benchmark method of level A (a) level B (b) and level C (c). Yellow histograms correspond to CUDA on K20, green histograms to OpenCL on K20, and blue histograms to OpenCL on MIC.

Table 3: Maximum performance results of CUDA on K20 and OpenCL on K20 and MIC for benchmark of level A, B, and C methods.

		SpMV	GESUMM	ATAX	BICG	MVT	GEMM	2MM	3MM
Level-A	K20_CUDA	1.00	0.96	1.00	0.93	1.00	0.69	0.72	0.90
	K20_OpenCL	0.21	1.00	0.73	1.00	0.93	1.00	1.00	1.00
	MIC_OpenCL	0.02	0.10	0.03	0.15	0.20	0.09	0.10	0.09

		2DConv	3DConv	BFS	PathFinder	SYRK	SYR2K	COR	COV
Level-B	K20_CUDA	0.74	1.00	1.00	0.95	1.00	0.98	1.00	0.80
	K20_OpenCL	1.00	0.58	0.22	1.00	0.99	1.00	0.85	1.00
	MIC_OpenCL	0.04	0.04	0.22	0.17	0.39	0.41	0.22	0.02

		Gaussian	HotSpot	Stencil27	Le2d	Himeno	Maxflops
Level-C	K20_CUDA	0.12	1.00	1.00	0.55	1.00	1.00
	K20_OpenCL	0.14	0.60	0.88	1.00	0.16	1.00
	MIC_OpenCL	1.00	0.05	0.19	0.05	0.90	0.66

3.3 Discussion

3.3.1 Benchmark Suite Consistency

Benchmark suite is a collection of different benchmarks used to measure the performance of a device. If the consistency of a benchmark suite is better, the credibility of performance got by it will be higher. In this paper, the scope of arithmetic intensity in the roofline model and the number of problems corresponding to "Thirteen Dwarfs" are the two important standard concerned.

Arithmetic intensity is the horizontal-axis of roofline model, which shows the ratio of floating point operations per byte of memory accessed. Floating point operations and memory accessed are two key points to an algorithm implemented and that is different between each algorithm. So benchmark suite that covers more scope of arithmetic intensity can measure performance of more different algorithms. EPCC OpenACC benchmark consists of up to 21 different benchmarks and some of them can change their arithmetic intensity with input scale. In the experiment, this benchmark suite can cover more than 40 arithmetic intensity and scope is from 0.062 to 192.

Phillip Colella in his 2004 presentation "Defining Software Requirements for Scientific Computing" about DARPA's High Productivity Computing Systems (HPCS) program gave his list of the now-famous "Seven Dwarfs" of algorithms for high-end simulation in the physical sciences.[9] "Seven Dwarfs" consists of Dense Linear Algebra, Sparse Linear Algebra, Spectral Methods, N-Body Methods, Structured Grids, Unstructured Grids and Monte Carlo. Following Colella, David Patterson and Katherine Yelick upgraded it to "Thirteen Dwarfs", which consists of Dense Linear Algebra, Sparse Linear Algebra, Spectral Methods, N-Body Methods, Structured Grids, Unstructured Grids, Map Reduce, Combinational Logic, Graph Traversal, Dynamic Programming, Backtrack and Branch-and-Bound, Graphical Models and Finite State Machines. According to "Thirteen Dwarfs" in the benchmark suite, Gaussian corresponds to Dense Linear Algebra, SpMV corresponds to Sparse Linear Algebra, Stencil27 and HotSpot corresponds to Structured Grids, BFS corresponds to Graph Traversal, PathFinder corresponds to Dynamic Programming. From this corner, the benchmark suite has covered five important problems in scientific computing.

3.3.2 Performance Portability

In the roofline model, more than 82% performance compared with peak performance on Kepler and Knight Corner is achieved by specific benchmarks, but more benchmarks only achieve about 10% performance. GESUMM benchmark can achieve approximately 100% performance on K20c and Gaussian benchmark can do it on 7110P. Most of benchmarks can get good performance are before 2 flop/byte of arithmetic intensity. After 2 flop/byte, performance rises with the arithmetic intensity.

In the relative performance model, performance portability is shown by FLOPS and compared by histogram and we can find that CUDA target code on K20c is the best on most benchmarks. OpenCL target code on 7110P only gets the best performance on Gaussian benchmark and the second best performance on Himeno benchmark. In other benchmarks, the performance gap between OpenACC for K20c and 7110P is more than 50%.

Firstly, the results shows that performance portability for OpenACC does not arrive the theoretical peak performance except GESUMM and Gaussian. That is related to OpenACC and benchmark suite. Because OpenACC using directive is high level programming model and few hardware details can be controlled such as on-chip memory and cores. Meanwhile there is little optimization on the benchmarks of EPCC OpenACC benchmark suite so that the performance got is the basic one. The two reason causes the gap between their performance and peak performance.

Secondly, BFS, PathFinder and Himeno these three benchmarks' performance is obviously low, which is lower than 500MFLOPS. The reason is related to application itself. Since these three benchmarks have more memory operation and branches. As we know, global memory on GPU and Xeon Phi is both GDDR, which has big latency compared to DDR. So if there is no optimization on memory operation, performance will not be good. More branches also will bring big latency with the program.

Last but not least, almost all the benchmarks get the better performance from K20c, performance on 7110P seems poor. Not all the code under offloading mode is suitable for Knight Corner. In the report by Goldi Misra, we can find that performance of HotSpot by OpenMP on Intel Xeon Phi is lower than Intel Xeon.[1] Their conclusion is that scale-up of the codes can be enhanced by optimizing codes for Intel Knight Corner architecture with respect to memory/cache and SIMD optimizations. There can be also much other reason. On Knight Corner thread scalability is 244 threads and load balance is 60 cores. 61 cores are on the chip, but one core is the system core running a micro Linux on it and managing the threads. If workload is token on system core, the performance will be affected. So

workload balance is important since workload will be switched on different cores if no control in the code. Awaking a thread needs a big consumption. OpenCL code by OpenACC compiler can not control the workload balance well. Now better performance still need deep optimization.

4 Related Work

Application performance evaluation on Xeon Phi and Kepler GPU has been found in the report by Aoki[2]. They use a primitive stencil application to measure performance on Xeon Phi and Kepler GPU. The report indicates that the performance of Xeon Phi is obviously lower than Kepler GPU. But in the report made by You[15], they gave out a new method to evaluation, which is performance efficiency. The formula of it is Obtained Performance/Theoretical Performance. The performance efficiency gap between two devices is small in the report.

Performance evaluation for CUDA and OpenACC on GPU by two microbenchmarks and one real-world CFD application is reported by Hoshino.[14] In that paper, evaluation indicates that about 50% of performance of the CUDA versions can be achieved by OpenACC compilers and reaching up to 98% depends on the compiler. The performance gap between CUDA and OpenACC is caused by on-chip memory control. Since CUDA has low-level interfaces to control the on-chip memory and OpenACC lacks programmable control on that, no more optimization can be made on OpenACC.

5 Conclusion

This paper studied performance portability for OpenACC on Nvidia Kepler and Intel Knight Corner by two benchmarks and one benchmark suite. Firstly, we use Steam benchmark and MaxFlops benchmark of SHOC benchmark suite to get memory bandwidth and peak performance results to build the roofline model. From roofline model, evaluation is about performance portability for OpenACC on benchmarks of EPCC OpenACC benchmark suite. At most more than 82% performance compared with peak performance on Kepler and Knight Corner is achieved by specific benchmarks. After 2 flop/byte of arithmetic intensity, performance becomes better as the rise of arithmetic intensity and the average performance is 10%. From this corner, the performance is limited under 10% of the peak performance after arithmetic intensity is higher than 2 flop/byte.

Secondly, we use EPCC OpenACC benchmark suite to get couples of benchmark performance on two platforms. And CUDA and OpenCL target code is also the variable of it. The results we got can be used to build relative performance model. From the model, we find the obvious performance gap between OpenACC on K20c and 7110P, which is caused by many reasons. Lack of on-chip memory and cores control is the main reason of them. Introducing more low-level interface to control the on-chip memory and cores or more auto-tuning will be a good way for OpenACC to promote the performance portability.

6 Future Work

In the experiment, obvious performance gap between OpenACC on K20c and 7110P is found. PGI compiler will be used to search how much effect on performance the compiler makes. In this paper, we just used some benchmarks to get results. Some real applications will be used to get more results to show the performance portability in the future, which can consist of multi algorithm not only one benchmark corresponds to one algorithm. As the new version of OpenACC 2.0 is released, we will do performance portability evaluation for it after the compiler supports the new version. And more optimization will be made on the benchmark suite, since it still lacks that now. OpenACC is good standard about cross-platform programming on different accelerators and co-processors, so we will try to use it to do hybrid parallel programming on both of two different devices.

7 Acknowledgements

This research is supported by NVIDIA Center of Excellence, HMPP CoC and JSPS RONPAKU Fellowship.
Thanks Dr. Eric Germaneau of SJTU for his thoughtful comments and help.

References

[1] Nisha Kurkure Abhishek Das Manjunatha Valmiki Shweta Das Abhinav Gupta Goldi Misra. Evaluation of Rodinia Codes on Intel Xeon Phi. *International Conference on Intelligent Systems, Modelling and Simulation*, 2013.

[2] Takayuki Aoki. Application Performances on Many-core Processors Xeon Phi versus Kepler GPU. 2013.

[3] OpenMP Architecture Review Board. OpenMP Application Program Interface. 2013.

[4] CAPS entreprise. OpenACC Reference Manual CAPSCompilers 3.3. 2013.

[5] Khronos OpenCL Working Group. The OpenCL Specification, v1.2.15. 2011.

[6] OpenACC Group. The OpenACC Application Programming Interface, v1.0. 2011.

[7] David A. Patterson John L. Hennessy and et al. *Computer Architecture: A Quantitative Approach, Fifth Edition*. Proprietor, 2012.

[8] Nick Johnson. Epcc openacc benchmark suite. 2013.

[9] Erich L. Kaltofen. The "Seven Dwarfs" of Symbolic Computation. *Numerical and Symbolic Scientific Computing*, 2011.

[10] John D. McCalpin. Stream: Sustainable memory bandwidth in high performance computers. 2013.

[11] md-rezaur rahman. The scalable heterogeneous computing benchmark suite (shoc) for intel xeon phi. 2013.

[12] NVIDIA. CUDA C Programming Guide, v5.0. 2013.

[13] H. Lin T. Scogland J. Zhang W. Feng. OpenCL and the 13 Dwarfs: A Work in Progress. *International Conference on Performance Engineering (ICPE)*, 2012.

[14] Satoshi Matsuoka Tetsuya Hoshinom Naoya Maruyama Ryoji Takaki. CUDA vs OpenACC: Performance Case Studies with Kernel Benchmarks and a Memory-Bound CFD Application. *International Symposium on Cluster, Cloud, and Grid Computing*, 2013.

[15] Fu Haohuan You Yang. Accelerating Forward Modeling by CPU,GPU and MIC. *Accelerators and Hybrid Exascale Systems*, 2013.