在 Intel Knights Corner 和 NVIDIA Kepler 架构上 OpenACC 的性能可移植性分析

王一超 秦 强 施忠伟 林新华
(上海交通大学 上海 200240)

摘 要 OpenACC 是一套基于指导语句方式的并行编程语言标准。编程者可以通过在代码中添加符合该标准的指导语句,经 OpenACC 编译器的编译,将串行代码并行化地移植到加速器或者协处理器上,进而获得异构加速器所带来的加速效果。OpenACC 与 CUDA 和 OpenCL 这类异构并行编程技术的不同之处在于,它的目的是使编程者在应用移植过程中不需要考虑加速器或协处理器的底层硬件架构,从而降低编程难度。同时它也具有仅需维护一套代码便可在不同硬件平台上运行的优良跨平台性。因此,OpenACC 是一个值得研究的并行编程标准。如今的异构加速硬件设备呈现出多元化趋势。在 2013 年 11 月的 Top500 榜单上排名第一的"天河二号"使用了 48000 块构建在 Intel Knights Corner 架构之上的协处理器。与此同时,发布不久的 NVIDIA 公司最新的 Kepler 架构 GPU 产品由于多年来的 GPU 市场积累也迅速形成了可观的用户群体。对于并非追求性能极限的应用移植者而言,寻求应用性能和移植简易性之间的平衡是相当重要的议题。只需要编写一套代码便可运行在这两种硬件平台上的 OpenACC 正迎合了用户在移植简易性上的需求。解决了移植的简易性之后,同一个应用在不同硬件平台上的性能表现便成了用户最想了解的问题。通过实验和构建性能模型向读者展示使用 OpenACC 移植的应用在 Intel Knights Corner 和 NVIDIA Kepler 架构硬件上的性能可移植性。

关键词 OpenACC,性能可移植性,高性能计算
中图法分类号 TP338.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.017

Performance Portability Evaluation for OpenACC on Intel Knights Corner and NVIDIA Kepler

WANG Yi-chao QIN Qiang SEE Simon LIN Xin-hua
(Shanghai Jiao Tong University, Shanghai 200240, China)

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 studied 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 have the latest architectures and similar peak performance. Meanwhile CAPS OpenACC compiler is used to compile EPCC OpenACC benchmark suite, Stream and MaxFlops of SHOC benchmarks to access the performance. To study the performance portability, roofline model and relative performance model were built by the data of experiments. It 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 confirmed 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 portabilty, High performance computing

1 引言

随着高性能计算技术的发展,异构加速硬件设备正一次次地帮助超级计算机打破运算性能的极限。在最新一期的 Top500 榜单上,位列前十的超级计算机中就有 4 台使用了异构加速硬件设备。由此可见,异构加速技术已经成为了走向 Exascale 超级计算机道路上的重要一环。与此同时,统一计算架构(Compute Unified Device Architecture,CUDA)和开放计算语言(Open Computing Language,OpenCL)的出现提高了异构加速设备的可编程性,卓有成效的推广工作也使得异

到稿日期:2013-12-26 返修日期:2014-03-15
王一超 男,硕士生,主要研究方向为高性能计算;秦 强 男,硕士生,主要研究方向为高性能计算;施忠伟 教授,主要研究方向为高性能计算;林新华 主要研究方向为高性能计算。

构并行编程现如今已广泛地应用于科学计算及工程计算领域。但是,基于 CUDA[12] 和 OpenCL[5] 的异构并行编程依旧存在着可编程性的问题。在用户对应用的移植过程中,对于异构加速设备的硬件架构的认识将直接影响到应用性能,这增加了编程的难度。出于进一步提高可编程性的考虑,基于指导语句的异构并行编程诞生了。

OpenACC[6] 是一套基于指导语句方式的并行编程语言标准,它旨在使用户能在高层级对于处理器和异构并行设备(例如 APU、GPU 和协处理器)进行并行编程。用户需要在需要并行化的代码段前添加 OpenACC 编程标准中提供的指导语句,经由各厂商开发的对应编译器的编译,就可以自动生成相应的并行化的中间代码(包括 CUDA 和 OpenCL),最后即可生成并行化的可执行程序。它可以替用户省去与底层硬件架构相关的设备初始化、数据管理和传输等来降低可编程性的操作,从而进一步降低异构并行编程的使用门槛。于 1997 年 10 月发布的 OpenMP[3](Open Multi-Processing)便是基于指导语句并行技术的最具代表性的编程标准。正如我们所看到的,OpenMP 提高了在 CPU 上进行同构并行编程的可编程性,而且随着编译器技术的发展,OpenMP 的性能也在提高。而版本更新更快的 OpenACC 标准的发展就更令人期待了,这也使得它成为了本文的研究对象。

现如今的异构加速硬件设备呈现出多元化趋势。在 2013 年 11 月的 Top500 榜单上排名第一的"天河二号"使用了 48000 块构建在 Intel Knights Corner 架构[1] 上的协处理器。与此同时,发布不久的 NVIDIA 公司最新的 Kepler 架构 GPU 产品由于多年来的 GPU 市场积累,也迅速形成了可观的用户群体。由于 OpenACC 是针对异构并行编程的标准且已具备在多设备上的优良跨平台性,因此针对 OpenACC 在 Intel Knights Corner 和 NVIDIA Kepler 架构上的性能可移植性分析将会是一项极具价值的工作,这也正是不同于同构并行编程标准的 OpenACC 最值得被考量的一点。

2 实验方法

2.1 编译器

本次实验使用 CAPS 公司开发的 OpenACC 编译器版本 3.3.3[4],以 OpenCL 作为编译的中间代码来实现测试算例在 Intel Knights Corner 和 NVIDIA Kepler 架构下的跨平台运行。

2.2 测试集

EPCC OpenACC benchmark suite[8] 是由英国爱丁堡大学高性能计算中心发布的一套 OpenACC 的测试集。其中,测试算例涵盖了 21 种科学计算中常见的算法实现,包括稀疏矩阵乘法、广度遍历等。"十三个小人"[13] 是由弗吉尼亚理工大学的冯吴春教授提出的针对将在未来科学和工程计算领域中起到重要影响的 13 类数值计算方法的一个归纳准则。它基于 Phillip Colella 教授提出的"七个小矮人"——密集线性代数、稀疏线性代数、谱方法、N-body 方法、结构化网格、非机构化网格和蒙特卡罗方法,添加了分布式算法、组合逻辑、图遍历、动态规划、回溯及分支合并、图像建模和有限状态机,去掉了蒙特卡洛方法。该测试集中的算例对应了"十三个小矮人"中的 5 个,它们分别是密集线性代数、稀疏线性代数、结构化网格、图的遍历以及动态规划。再从计算密集程度的角度来考量这个测试集。使用算法强度这一概念,来衡量一个算法的计算密集程度。算法强度是指算法每进行一个字节的内存读写,浮点数运算的次数。在具体实验中,我们会事先分析算法的代码从而计算出该算法的算法强度,即总的浮点数运算次数/总的内存读写大小。由此计算了所有 20 个算例在不同问题规模下的算法强度,发现其分布于 0.062 到 192 之间。

Stream[10] 测试程序最早是由 John McCalpin 教授设计的,通过大量密集的数据传输操作测试 CPU 的内存读写带宽。近些年来,随着加速器的出现,该测试程序也有了相应的版本。为了获得 Intel Knights Corner 和 Nvidia Kepler 架构下的极限内存传输带宽,本实验针对 GPU 使用的是 CUDA 版本代码,针对 MIC 使用的是 OpenMP 版本代码。

SHOC[11](Scalable Heterogeneous Computing)是由一系列测试异构系统稳定性的测试程序组成的测试集。该测试集中的 MaxFlops 测试程序通过大量密集计算可以获得加速卡或者协处理器的极限浮点数运算能力。同时正如现在的异构系统由不同的硬件架构组成一样,该测试集也包括了 MPI、OpenMP、OpenCL 和 CUDA 多种版本。

3 性能模型

3.1 屋顶线模型

该性能模型将浮点数运行性能、内存传输带宽以及算法强度结合在了一张二维图里[7],旨在通过图形化方式展示硬件架构的浮点数运算潜在性能的模型。通过它应用移植者可以了解到当前的应用性能与该硬件架构的极限性能之间的差距。本实验采用该模型旨在针对单个算例进行性能的评估,从而衡量该算例的特性。

模型的 X 轴表示算法强度,算法强度的单位是 flop/byte,也即表示每个字节的内存访问所进行的浮点数运算。计算密集型和非计算密集型的算例对于测试集的完备性具有重要意义,该变量与算例的计算密集程度相关,可以帮助我们了解测试集的完备性。模型的 Y 轴表示浮点数的运算性能,单位是 Mflops。

"屋顶线"具体由两部分组成——斜线与水平线。斜线部分以实测的内存传输带宽作为斜率,表示由于内存传输带宽造成性能极限。水平线部分直接采用实测的浮点数性能极限,表示该硬件架构下的浮点数运算性能极限。

3.2 相对性能模型

该模型旨在帮助我们进行性能可移植性的分析,即相同代码的算例在两种硬件平台上的性能差值。差值越大说明此算例针对于这两种硬件平台的性能可移植性越差。本文使用成组的柱状图来表现性能可移植性。每个算例都使用同样的 OpenACC 代码,通过 CAPS 的 OpenACC 编译器分别编译成 CUDA 和 OpenCL 两套中间代码。其中 CUDA 版本可以运行在 NVIDIA 的 Kepler 架构加速器上,OpenCL 代码既可以运行在加速器上,也可以运行在 Intel 的 Knights Corner 架构协处理器上。由此得到了同一算例在两个不同硬件平台上的三元组数据,以三元组中性能最佳的数据作为 100%,计算出另两个数据相对于它的百分比。实验结果的呈现即把 3 个数据作为一组构成柱状图,并将所有算例的数据并列显示在同一张图上。

4 实验结果及分析

本次实验使用的 Intel Knights Corner 架构的产品为 In-

tel Xeon Phi 7110P，NVIDIA Kepler 架构的产品为 NVIDIA Tesla K20c。

4.1 屋顶线模型测试结果

由于屋顶线模型需要针对硬件的基本性能参数进行实测，因此本次试验使用 Stream 算例对内存传输带宽进行测试，使用 SHOC 测试集中的 MaxFlops 算例针对浮点数的峰值性能进行测试，结果如表 1 和表 2 所列。

表 1 基于 Stream 测试算例的内存带宽测试结果

设备	理论值(GB/s)	实测值(GB/s)	百分比
NVIDIA Tesla K20c	208	150	72.1%
Intel Xeon Phi 7110P	352	174	49.3%

表 2 基于 SHOC 测试集中 MaxFlops 测试算例的浮点数峰值性能测试结果

设备	计算精度	理论值(Gflops)	实测值(Gflops)	百分比
NVIDIA Tesla K20c	单精度	3520	3006	85.4%
	双精度	1170	1168	99.8%
Intel Xeon Phi 7110P	单精度		1945	
	双精度	1220	980	80.0%

基于以上的测试数据，针对 Knights Corner(即图中的 MIC)和 Kepler(即图中的 K20)两种架构分别构建了屋顶线模型。然后在这两个平台上使用 EPCC 的 OpenACC 测试集分别进行测试，在 Knights Corner 上以 OpenCL 作为中间代码，而在 Kepler 上分别以 OpenCL 和 CUDA 作为中间代码。将所有的测试结果呈现在屋顶线模型上后，结果如图 1—图 3 所示。

通过该模型可以发现两个显著的特征。第一，所有的算例距离“屋顶线”所表示的峰值性能还有比较大的差距，这表明 EPCC 测试集中的算例还未调优至其所在平台的最佳性能，不论是 Knights Corner 还是 Kepler。所以，EPCC 的 OpenACC 测试集是未经代码优化的版本。第二，EPCC 测试集中的算例的算法强度分布从 0.062 递增到 192，且分布基本均匀，增强了本次测试的全面性和可靠性。再结合所有算例已涵盖了“十三个小矮人”中的 5 个，这些都表明 EPCC 的 OpenACC 测试集具备优良的测试全面性。

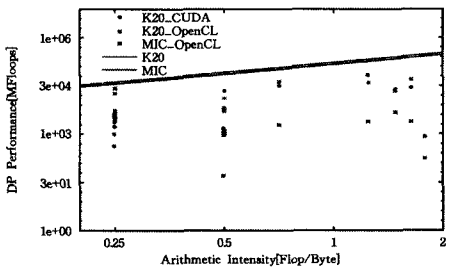

图 1 Level-A 算例的 roofline 模型

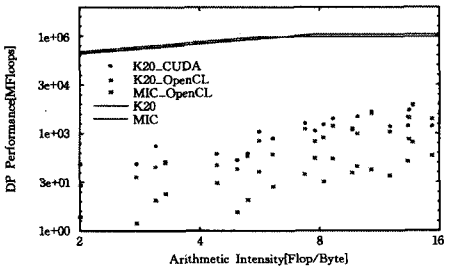

图 2 Level-B 算例的 roofline 模型

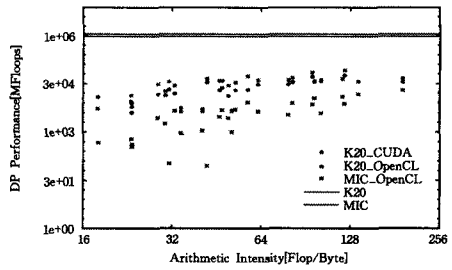

图 3 Level-C 算例的 roofline 模型

4.2 相对性能模型

EPCC 的 OpenACC 测试集中的 20 个测试算例及 SHOC 测试集中的 MaxFlops 算例分别在 7110P 和 K20c 上进行了测试，测试的结果构建成的相对性能模型如图 4 所示。

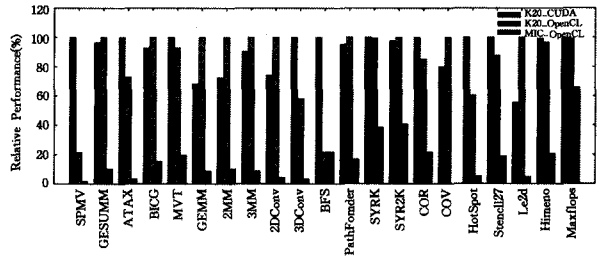

图 4 EPCC 测试集的性能可移植性分析图

仅 MaxFlops 在 Knights Corner 上的性能达到了 Kepler 上的约 70%，其余所有的算例性能在 Knights Corner 上的均未达到 Kepler 上的 40%。由于 MaxFlops 是高度计算密集型的应用且经过一定的优化，这也就表明了未经优化的采用 OpenACC 移植到 Intel Knights Corner 和 NVIDIA Kepler 架构下的应用之间性能依旧存在着很大的差距，这也说明了未经优化的 OpenACC 应用性能可移植性不佳。

5 相关工作

东京工业大学的 Aoki 在文献[2]中使用一个未经优化的应用，针对 Intel Knights Corner 与 NVIDIA Kepler 这两个平台进行了应用的性能评估工作，研究表明该应用在 Knights Corner 上的性能明显低于 Kepler 上的。但是 You 在文献[14]中针对性能效率(即实测性能/理论性能)对这两个平台进行评估，结果是这两个平台下性能效率之间的差异并不大。

日本东京工业大学的 Hoshino 在文献[15]中针对两个小型测试算例以及一个现实的 CFD 应用，通过 CUDA 和 OpenACC移植后运行在 GPU 平台上的性能做了评估，他指出使用 OpenACC 移植的应用经由不同的编译器编译能达到 CUDA 版本 50%至 98%的峰值性能。文章的结论是 CUDA 和 OpenACC 在性能上的差值是由于 OpenACC 缺乏对于底层内存管理接口支持造成的，所以提供基于内存控制的优化将在 OpenACC 的进一步性能提高方面具有重要意义。

结束语 本文通过使用 EPCC 测试集针对 OpenACC 在 Intel Knights Corner 和 NVIDIA Kepler 架构下的性能可移植性进行了分析。首先，通过 Stream 和 MaxFlops 测试算例获得内存传输带宽和浮点数运算的峰值性能，从而构建针对两个不同硬件架构的屋顶线性能模型。将所有测试数据通过屋顶线模型呈现后能发现两个特点。第一，所有的算例距离峰值性能还有比较大的差距，这表明 EPCC 测试集中的算例还

未调优至最佳性能。第二，EPCC 测试集中的算例的算法强度分布均匀，再结合所有算例已涵盖了“十三个小矮人”中的5个，这些都表明该测试集具备优良的测试性。其次，通过对比同一个算例在 Intel Knights Corner 和 NIVIDIA Kepler 架构下的性能，构建了针对这两个平台的相对性能模型。本实验中，仅一个算例在 Knights Corner 上的性能达到了 Kepler 上的约 70%，其余所有的算例性能在 Knights Corner 上的均未达到 Kepler 上的 40%。这也就表明了未经优化的 OpenACC 代码在 Intel Knights Corner 和 NVIDIA Kepler 架构下的性能可移植性不佳。

参 考 文 献

[1] Kurkure N, Das A, Valmiki M, et al. Evaluation of Rodinia Codes on Intel Xeon Phi[C]// 4th International Conference on International Conference on Intelligent Systems, Modelling and Simulation, 2013. Bangkok: IEEE, 2013: 415-419

[2] Aoki T. Application Performances on Many-core Processors Xeon Phi versus Kepler GPU[OL]. 2013-12[2014-3]. http://www.ocw.titech.ac.jp/index.php? module=General&action=DownLoad&file=20131226717065-477-1-45.pdf&type=cal&JWC=20131226717065

[3] OpenMP Architecture Review Board. OpenMP Application Program Interface[OL]. 2013-7[2014-4]. http://www.openmp.org/mp-documents/spec30.pdf

[4] CAPS entreprise. OpenACC Reference Manual CAPSCompilers 3.3[OL]. 2012-12[2014-4]. http://www.caps-entreprise.com/products/caps-compilers/

[5] Khronos OpenCL Working Group. The OpenCL Specification [OL]. 2008-12[2014-4]. https://www.khronos.org/registry/cl/specs/opencl-1.0.29.pdf

[6] OpenACC Group. The OpenACC Application Programming Interface_v1.0[OL]. 2011-11[2014-4]. http://www.openacc.org/sites/default/files/OpenACC.1.0_0.pdf

[7] David A. Patterson John L. Hennessy and et al. Computer Architecture: A Quantitative Approach(第5版)[M]. 北京：机械工业出版社，2012：285-288

[8] Johnson N. EPCC OpenACC benchmark suite [OL]. 2013-5 [2014-4]. https://www.epcc.ed.ac.uk/research/computing/performance-characterisation-and-benchmarking/epcc-openacc-benchmark-suite

[9] Kaltofen E L. The "Seven Dwarfs" of Symbolic Computation [C]// Numerical and Symbolic Scientific Computing, 2012. Wien: Springer Vienna, 2012: 95-104

[10] McCalpin J D. Stream: Sustainable memory bandwidth in high performance computers[OL]. 2013-2[2014-4]. http://www.cs.virginia.edu/stream/ref.html

[11] md rezaur rahman. The scalable heterogeneous computing benchmark suite (shoc) for intel xeon phi[OL]. 2013-4[2014-4]. https://software.intel.com/en-us/blogs/2013/03/20/the-scalable-heterogeneous-computing-benchmark-suite-shoc-for-intelr-xeon-phitm

[12] NVIDIA. CUDA C Programming Guide [OL]. 2014-2(5.5) [2014-4]. http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz342yBEw4Q

[13] Lin H, Scogland T, Zhang J, et al. OpenCL and the 13 Dwarfs: A Work in Progress[C]// ICPE'12 Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, 2012. New York: ACM, 2012: 291-294

[14] Hoshinom T, Maruyama N, Takaki R. CUDA vs OpenACC: Performance Case Studies with Kernel Benchmarks and a Memory-Bound CFD Application[C]// 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2013. Delft: IEEE, 2013: 136-143

[15] Yang You, Fu Hao-huan, Huang Xiao-meng, et al. Accelerating the 3D Elastic Wave Forward Modeling on GPU and MIC[C]// the 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum. Washington, 2013: 1088-1096

(上接第 66 页)

这类方法可以推广到不是基于结构网格的其他并行应用。希望将来在各种应用中都能使用这类方法。

参 考 文 献

[1] Luitjens J, Berzins M. Improving the performance of Uintah: A large-scale adaptive meshing computational framework[C]// Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS10). 2010: 1-10

[2] Sheng Di, Kondo D, Cirne W. Host Load Prediction in a Google Compute Cloud with a Bayesian Model[C]// International Conference for High Performance Computing, Networking, Storage and Analysis (SC12). 2012

[3] Ishiyama T, Nitadori K, Makino J. 4.45 Pflops Astrophysical N-Body Simulation on K Computer -The Gravitational Trillion-Body Problem[C]// International Conference for High Performance Computing, Networking, Storage and Analysis (SC12). 2012

[4] 张林波，迟学斌，莫则尧，等. 并行计算导论[M]. 北京：清华大学出版社，2006

[5] Frenkel D, Berend S. Understanding Molecular Simulation[M]. Academic Press Inc., 2001

[6] Hess B, Kutzner C, Spoel D V D, et al. GROMACS 4: Algorithms for highly efficient, load balanced, and scalable molecular simulation[J]. Journal of Chemical Theory and Computation, 2008, 4(3): 435-447

[7] 曹小林，刘旭. 基于 JASMIN 框架的粒子模拟并行计算[J]. 科研信息化技术与应用，2010，1(2)：28-33

[8] 莫则尧，张爱清. 并行自适应结构网格应用支撑软件框架 JASMIN 用户指南[R]. T09-JMJL-01. 北京应用物理与计算数学研究所，2009

[9] Mo Z Y, Zhang A Q, Cao X L, et al. JASMIN: a parallel software infrastructure for scientific computing[J]. Front. Comput. Sci. China, 2010, 4(4): 480-488

[10] 刘旭，张爱清. 一种空间矩形剖分的负载平衡算法[C]// 高性能计算年会(HPCC12). 2012

[11] 刘旭，莫则尧，曹小林. 基于内存约束的一维负载平衡方法及其应用[J]. 计算物理，2009，26(2)：184-190