软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(8):2341−2351 [doi: 10.13328/j.cnki.jos.006006]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

国产异构系统上的 HPCG 并行算法及高效实现*

刘芳芳[1,2,3], 王志军[1,2], 汪 荃[1,2], 吴丽鑫[1,2], 马文静[1,3], 杨 超[4], 孙家昶[1]

[1](中国科学院 软件研究所 并行软件与计算科学实验室,北京 100190)

[2](中国科学院大学,北京 100049)

[3](计算机科学国家重点实验室(中国科学院 软件研究所),北京 100190)

[4](北京大学 数学科学学院,北京 100871)

通讯作者: 马文静, E-mail: wenjing@iscas.ac.cn

摘 要: HPCG 基准测试程序是一种新的超级计算机排名度量标准.该测试基准主要用于衡量超级计算机解决大规模稀疏线性系统的能力,更贴近实际应用,近年来广受关注.基于国产超级计算机研究异构众核并行 HPCG 软件具有非常重要的意义,其不仅可以提升国产超级计算机 HPCG 的排名,还对很多应用提供了并行算法、优化技术等方面的参考.面向某国产复杂异构超级计算机开展研究,首先采用了分块图着色算法对 HPCG 进行并行,并提出一种适用于结构化网格的图着色算法.该算法并行性能高于传统的 JPL、CC 等算法,且着色质量高,运用于 HPCG 后,迭代次数减少了 3 次,整体性能提升了 6%.分析了复杂异构系统各个部件传输的开销,提出一套更适用于 HPCG 的任务划分方法,并从稀疏矩阵存储格式、稀疏矩阵重排、访存等角度开展了细粒度的优化.在多进程计算时,还采用内外区划分算法将核心函数 SpMV、SymGS 中的邻居通信操作进行了隐藏.最终整机测试时,性能达到了国产超级计算机峰值性能的 1.67%,与单节点相比,整机弱可扩展性并行效率达到了 92%.

关键词: HPCG;国产超级计算机;图着色;SpMV;SymGS

中图法分类号: TP303

中文引用格式: 刘芳芳,王志军,汪荃,吴丽鑫,马文静,杨超,孙家昶.国产异构系统上的 HPCG 并行算法及高效实现.软件学报, 2021,32(8):2341−2351. http://www.jos.org.cn/1000-9825/6006.htm

英文引用格式: Liu FF, Wang ZJ, Wang Q, Wu LX, Ma WJ, Yang C, Sun JC. Parallel algorithm and efficient implementation of HPCG on domestic heterogeneous systems. Ruan Jian Xue Bao/Journal of Software, 2021,32(8):2341−2351 (in Chinese). http://www.jos.org.cn/1000-9825/6006.htm

Parallel Algorithm and Efficient Implementation of HPCG on Domestic Heterogeneous Systems

LIU Fang-Fang[1,2,3], WANG Zhi-Jun[1,2], WANG Quan[1,2], WU Li-Xin[1,2], MA Wen-Jing[1,3], YANG Chao[4], SUN Jia-Chang[1]

[1](Laboratory of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

[3](State Key Laboratory of Computer Science (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

[4](School of Mathematical Sciences, Peking University, Beijing 100871, China)

* 基金项目: 中国科学院战略性先导科技专项(C 类)(XDC01030200); 国家重点研发计划(2018YFB0204404, 2016YFB0200603)
Foundation item: Strategic Priority Research Program of the Chinese Academy of Sciences (Category C) (XDC01030200); National Key Research and Development Program of China (2018YFB0204404, 2016YFB0200603)
本文由“国产复杂异构高性能数值软件的研制与测试”专题特约编辑孙家昶研究员、李会元研究员推荐.
收稿时间: 2019-08-22; 修改时间: 2019-12-05; 定稿时间: 2020-01-22

Abstract: HPCG benchmark is a new standard for supercomputer ranking. This benchmark is used mainly for evaluating how fast a supercomputer is able to solve a large scale sparse linear system, which is closer to real applications, and has attracted extensive attention recently. Research of parallel HPCG on domestic heterogeneous many-core supercomputers is very important, not only to improve the HPCG ranking of Chinese supercomputers, but also to provide reference of parallel algorithm and optimization techniques for many applications. This work studies parallelization and optimization of HPCG on a domestically produced complex heterogeneous supercomputer, leveraging blocked graph coloring algorithm for parallelism exploration for the first time on this system, and proposes a graph coloring algorithm for structured grids. The parallelism produced by this algorithm is higher than the traditional JPL and CC algorithm, with better coloring quality. With this algorithm, successfully reduced the iteration number of HPCG by 3 times, and the total performance is improved by 6%. This study also analyzes the data transfer cost of each component in the complex heterogeneous system, providing a task partitioning method, which is more suitable for HPCG, and the neighbor communication cost in SpMV and SymGS is hidden by inner-outer region partitioning. In the whole-system test, an HPCG performance of 1.67% of the peek GFLOPS of the system is achieved, compared to single-node performance, the weak-scaling efficiency on the whole system has reached 92%.

Key words: HPCG; domestic supercomputer; graph coloring; SpMV; SymGS

HPCG(high performance conjugate gradients)基准测试程序是一种新的超级计算机排名度量标准[1].该测试基准主要用于衡量超级计算机解决大规模稀疏线性系统的能力,相比于 HPL(high performance Linpack)[2],其计算、访存与通信模式与当前主流高性能计算机应用程序更为契合,能够更全面地反映系统的访存带宽和延迟的性能.HPL 和 HPCG 排名每年发布两次,一次在德国的 ISC[3]大会上,另一次在美国的 SC[4]大会上.位于世界前列的超级计算机均提交 HPL 和 HPCG 两个基准测试程序的结果,2019 年德国举办的 ISC 大会同时发布了 HPL 和 HPCG 结果.越来越多的应用研究人员开始关注 HPCG 的测试结果,期望从中可以了解超级计算机访存、通信等的特点,一些 HPCG 并行和优化方法也可能对某些应用有所启示.

目前超级计算机的体系结构日趋复杂,早已从同构转向异构,且同节点异构部件的运算性能差异越来越大,这对异构并行算法的设计将有很大的影响.本文主要面向一类国产复杂异构系统研究 HPCG 并行算法和高效实现方法.目前 Dongarra 教授课题组研究了 HPL-AI 混合精度算法[5,6],我们也正在关注该方向的研究.

本文工作是在国产超级计算机研制方大力支持下完成的,在此表示感谢.

1 HPCG 简介

HPCG 来源于半结构网格上的三维热传导应用,其核心是在三维规则区域上对 Poisson 方程进行有限差分离散化后,转换成一个稀疏线性方程组的求解问题.其方程如式(1)所示.采用如图 1 所示的 27 点 stencil 格式进行离散化.

$$coef \times x(i,j,k) - 1.0 \times \sum x(i',j',k') = b(i,j,k) \tag{1}$$

其中,每个网格点的更新都依赖其周围紧邻的点.由于参与计算的点只能在三维网格区域范围内,所以每个网格点最多依赖 26 个邻点,具体的邻点的个数,分别为 26(内部点,邻点都在域内)、17(边界面上的点,某一邻面上的 9 个点都在域外)、11(边界线上的点,比边界面上的点又少了 2/3 个面的点,即又少了 6 个点)和 7(边界顶点,仅有一个小立方体上的 8 个点在域内,即图 1 的 1/8,包括其自身).最终得到的稀疏矩阵 A 是由 27 条对角线组成,并且是一个对称正定矩阵.

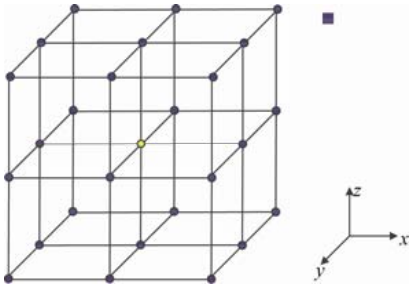


Fig.1 3D 27-point stencil in HPCG

图 1 HPCG 三维 27 点 stencil 格式

在大规模并行环境中,HPCG 使用三维区域分解策略,也就是按照 3 个维度将整个计算区域划分成若干个子区域,每个子区域被分配一个 MPI 进程,其规模由输入参数指定.每个 MPI 进程处理的计算区域中的内部点所依赖的 26 个邻点,全部来自于该计算区域,而对于计算区域边界上的点,其依赖的邻点一部分来自于该计算区域内部,一部分则来自于邻居进程所处理的子区域.不妨称计算区域为内区,其对应

的点为内点,计算区域所依赖的邻居进程的点构成的区域为外区,其对应的点为外点.由于我们使用的是 27 点 stencil,所以每个进程最多有 26 个邻居进程.

在 HPCG 中,该线性系统使用预处理共轭梯度(preconditioned conjugate gradient,简称 PCG)法来求解,计算流程如图 2 所示.

CG 算法中一次迭代计算包含 3 个向量点积操作(第 4、7、10 行)和 3 个向量更新操作(WAXPBY,第 6、8、9 行)以及 1 个稀疏矩阵向量乘操作(SpMV,第 7 行).

HPCG 中使用的预条件子 M^{-1} 是基于 V-cycle 几何多重网格,其计算流程如图 3 所示.多重网格通过将细网格上频率变化较缓慢的低频分量映射到粗网格上来处理,以达到加快收敛速度的目的.作为预条件子,多重网格的计算同样包含 4 个主要操作:前磨光操作(第 4 行)和后磨光操作(第 8 行)、限制算子(第 5 行)、插值算子(第 7 行)、底部解法器(第 2 行).在 HPCG 中,多重网格的层数被固定为 4 层,前后磨光操作以及底部解法器都是使用对称 Gauss-Seidel 迭代法的一次迭代来实现,并且磨光操作只执行一次,不可修改.使用 $SymGS(x,r)$ 表示一次对称 Gauss-Seidel 迭代,其中 r 是右端项,x 是解向量的初始猜测值.

Algorithm 1. CG for $AX=b$.

1: **Input**: A, b, x_0, $it_{\max}$, ε
2: $r_0=b-Ax_0$
3: **for** $i=0, 1, ..., it_{\max}$ **do**
4: $z_i=M^{-1}r_i$
5: $s_i=(r_i, z_i)$
6: **if** $(i=0)$ $p_i=z_i$
7: **else** $p_i=z_i+(s_i/s_{i-1})p_{i-1}$
8: $\alpha_i=s_i/(p_i, Ap_i)$
9: $x_{i+1}=x_i+\alpha_i p_i$
10: $r_{i+1}=r_i-\alpha_i Ap_i$
11: **if** $(\|r_{i+1}\|_2/\|r_0\|_2\leqslant\varepsilon)$ **break**;
12: **endfor**
13: **Output**: x_{i+1}

Fig.2 Conjugate gradient algorithm
图 2 共轭梯度法

Algorithm 2. V-cycle for $x^h=MG(r^h)$.

1: **Input**: r^h
2: If on the coarsest level **then**
3: $x^h=SymGS(0,r^h)$
4: **Else**
5: $x^h=SymGS(0,r^h)$
6: $r^{2h}=I_h^{2h}(r^h-A^hx^h)$
7: $x^{2h}=MG(r^{2h})$
8: $x^h=x^h+I_{2h}^h x^{2h}$
9: $x^h=\mathrm{SymGS}(x^h, r^h)$
10: **End**
11: **Output**: x^h

Fig.3 Multigrid pre-conditioner
图 3 多重网格预条件子

在 HPCG 中主要存在两种通信类型:全局通信和邻居通信.

1. 全局通信.主要为 MPI_Allreduce 通信,发生在向量点积操作中.在一次 CG 迭代中总共需要计算 3 次向量的点积,因此需要调用 3 次 MPI_Allreduce.

2. 邻居通信.主要发生在与稀疏矩阵相关的两个核心函数中:SpMV 和 SymGS,使用 MPI 点到点的通信实现 halo 区数据交换.HPCG 采用 27 点 stencil 格式,每个网格点的计算依赖周围紧邻的 26 个邻居.在 SpMV 和 SymGS 计算开始前,需要与周围的邻居进程交换边界数据(halo),以便将计算过程中所需要访问到的邻居数据传输到本地进程中.

3. 目前超级计算机的体系结构日趋复杂,早已从同构转向异构,且同节点异构部件的运算性能比差异越来越大,这对异构并行算法的设计将有很大的影响.本文主要面向一类国产复杂异构超级计算机研究 HPCG 异构众核并行算法.首先介绍了国内外进展,然后提出两种适用于该平台的 HPCG 异构众核并行算法.对于实际使用的第 2 种算法中,还存在图着色的问题,本文也开展了相关工作,并单独使用一节进行介绍.接下来介绍在该超级计算机上的实现问题,包括 CPU 和 GPU 之间的任务划分、为了隐藏邻居通信实现的内外区划分方法,以及一些稀疏矩阵重排等优化方法,最后介绍了实验结果.

2 国内外进展介绍

HPCG 的优化工作主要是针对热点函数 SpMV 和 SymGS 来开展.SpMV 的性能与存储格式以及计算方式有关.Bell 等人[7]在 GPU 平台上实现了常用的 CSR、COO、ELL 等基本格式,提出了一种新的 HYB 格式来提高 SpMV 的性能,并分析了不同矩阵适合使用的格式以及并行方法.另外,还有一些其他格式包括 ELLPACK-R[8]、BRC[9]、BCCOO[10]等被提出.还有一些学者研究稀疏矩阵的重排技术[11]及压缩格式[12],以减少访存开销.在计算方式方面,Williams 等人[13]在多种不同架构的多核平台上使用了线程分块、缓存分块、向量化等优化手段,并取得了很好的结果.另外还有学者研究自动调优技术[14],通过分析稀疏矩阵的特征来选择参数以提升性能.

SymGS 的求解过程类似于稀疏三角解法器,其关键是如何在这种强依赖的限制下获得并行度.研究人员已经提出了很多不同的并行方法,如 Park 等人[15]提出了 level scheduling 方法,这种方法使用层次遍历稀疏矩阵对应的有向无环图,保证每一层的节点间没有依赖关系,从而可以并行更新.但是这种方法获得的并行度不高,且各层之间的负载不均衡.Iwashita 等人[16]提出了适合结构化稀疏三角求解器问题的分块图着色技术,这种方法以块为单位对图进行着色,保证一个块与其相邻块颜色不同,相同颜色的块一起执行.这种方法虽然会破坏部分依赖,但是可以获得较大的并行度.

HPCG 的整体优化工作也受到了很多研究者们的关注.Kumahata 等人[17]在基于 CPU 的日本超级计算机 K 上的 HPCG 优化工作使用了分块图着色排序来挖掘 SymGS 中的并行度,并通过改变反向更新时的计算顺序来提高 cache 命中率,在 82 944 个 CPU 上取得了 0.461 PFLOPS 的成绩,达到整机峰值性能的 4.34%.Phillips 等人[18]在 NVIDIA GPU 平台上使用图着色方法对 SymGS 进行并行.刘益群等人[19,20]基于天河 2 号超级计算机的特征提出了混合的 CPU-MIC 协作的异构计算方法,通过区域划分充分利用 CPU 与 MIC 的计算资源,在整机 16 000 个节点上取得了 0.623 PFLOPS 的性能,位列 2014 年 11 月排行榜第一名.敖玉龙等人[21]设计了基于申威架构的并行算法,该算法使用分块着色的方式挖掘 SymGS 中的并行度,并使用神威特色的片上寄存器通信机制优化了核间的数据交换,在 163 840 个节点上取得了 0.481 PFLOPS 的性能.Ruiz 等人[22]基于 ARM 平台研究了多色重排和分块多色重排两个算法的性能,并从 cache miss 等角度进行了分析.目前世界排名第一的是 IBM 研制的超级计算机 Summit,其 HPCG 性能达到了 2.92 PFLOPS,是整机峰值性能的 1.5%,但目前还没有相关文献详细介绍其并行算法和优化技术.面向一类国产复杂异构系统研制众核并行版 HPCG 软件,且峰值性能和整机效率达到或超过 Summit 的水平,是一项具有巨大挑战的工作.

3 异构众核并行算法设计

GPU 是最近十几年被广泛使用的高性能计算机加速部件.一个 GPU 由多个计算单元(computing unit,简称 CU)组成,每个 CU 里有多个 SIMD,可以同时进行多路向量运算.CU 内部具有高速的软件可控缓存 LDS(local data share).所有 CU 共享 L2 cache 和设备内存.程序运行时组织成多个 workgroup,每个 workgroup 可包含一个或若干个 wavefront.每个 workgroup 只能在一个 CU 上运行,在我们使用的平台上,每个 wavefront 包含 64 个线程,一个 workgroup 内所包含的线程数不能超过 1 024 个.在这种高并行度的复杂架构上并行和优化 HPCG,是一个非常大的挑战.在并行算法层面,不仅要求高并行度,还要求整体迭代次数不能有较大增长.在优化方面,不仅要从算法层面尽量减少访存和通信,并将通信与计算重叠,还需要充分利用硬件的特性减少访存开销.

3.1 分块着色+level scheduling算法

3.1.1 基本算法

如第 2 节所述,HPCG 中 SymGS 模块的经典优化方法包括点着色方法、按层计算方法(level scheduling)以及分块着色算法等.考虑计算平台的特点及收敛速度的需求,我们选择了分块着色算法作为程序的基本框架.我们将 SymGS 和 SpMV 等基本算子在分块着色算法的基础上进行并行化,并提出了针对 GPU 异构平台的优化方法.

分块着色算法的基础是将三维网格分块,然后对每一个数据块进行着色.在预条件子中使用的四重网格里,每一层网格都需要分块及着色.参考文献[21]的工作,我们选取了类似的分块着色算法,既方便数据在 GPU 上的分配,又保证了较高的并行度,且不会大幅度增加迭代次数.如图 4 所示,三维空间中相邻的 8 个数据块使用 8 种不同的颜色,每次 SymGS 计算时,同样颜色的数据块被同时处理.

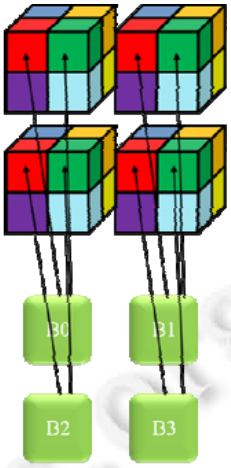


Fig.4 Block coloring algorithm

图 4 分块着色算法

3.1.2 数据及任务映射

块内 SymGS 需要严格保持依赖关系,否则迭代次数将大幅度上升.因此,本文采用 level scheduling 算法进行并行,方程 $4x+2y+z=k$ 所定义的平面上的点都可以算作一层(每个点的坐标为(x,y,z)),因为它们所依赖的数据都已经计算完毕.如图 5 所示,从上到下是时间轴,第 1 个点的计算不需要本次迭代的任何数据,可以在此数据块处理过程的第 1 次迭代进行更新;然后,下一个点仅依赖于第 1 个点,而第 1 个点已经有最新数据,因此第 2 个点可以进行计算,此为第 2 层;再然后,我们可以分析出第 3 层,第 4 层,...,直到整块数据被处理完.依照这种排序,每层数据的更新可以并行操作.通过实验不同的块大小,我们选出能够适应高速缓存 LDS 大小且保证一定的块间及块内并行度的分块方案.

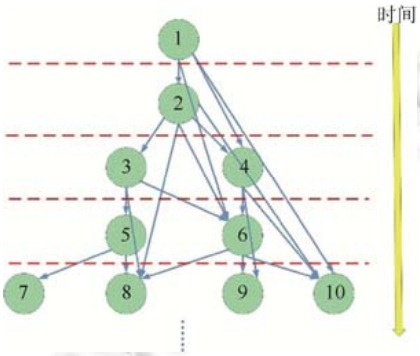


Fig.5 Data dependence and levels inside a block

图 5 块内数据依赖及分层

3.1.4 SymGS 内核优化

在分块着色+块内 level scheduling 算法的基础上,我们对 SpMV 和 SymGS 的内核函数进行了优化.SpMV 函数的优化参考文献[23]的方法,每线程计算一个双精度乘法,将中间结果保存在 LDS 中,然后将 LDS 中的乘积

进行相加.这种方法保证对矩阵 A 的读取都是连续访存(coalesced access),因而能够充分利用内存带宽.

对 SymGS 函数,我们对每一 level 采用类似的方式在一个 workgroup 内运行.这就需要对整个矩阵进行重排,使之在计算开始之前,就按分块之后所需要的顺序排列好.在 SymGS 函数被调用时,矩阵 A 也能实现连续访存.由于 level 内并行度较小,我们将 workgroup 大小定为一个 wavefront 的大小,即 64 线程.

在进行归约操作时,我们通过实验不同的归约方法,选出了一种尽量充分利用处理器单元的方式.

3.2 分块图着色算法

由于 HPCG 采用 27 点 stencil 格式进行离散,第 3.1 节分块方式使得每个子块内部网格点依赖关系比较强,只有严格保持其依赖关系整个问题才能收敛.而上一节提到,块内采用 level scheduling 的并行方式不能有效利用硬件资源,所以本节提出一种新的分块图着色算法,如图 6 所示.将网格 y 方向一条作为一个块进行处理,这样块内依赖较少,可以对其进行并行计算,同时块内向量 x 可以进行重用,以提升访存带宽利用率.将所有子块进行着色,每一个颜色的子块并行执行.着色方案需要精细设计,以确保整个算法可以以较少的迭代次数收敛,从而提升整体性能.

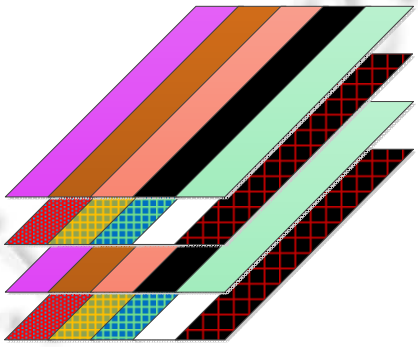


Fig.6 Blocked graph coloring algorithm

图 6 分块图着色算法

该方案可以实现两级并行:同色子块并行执行和子块内部网格点并行执行.这样可以更好的利用 GPU 的硬件资源,充分发挥硬件的性能.同时块内可以利用 LDS 对向量 x 进行重用,从而减少向量 x 从设备内存访存的次数,提升访存带宽利用率.

4 图着色算法研究

第 3.2 节的方案中需要对所有子块进行着色,且着色算法对整个 HPCG 的迭代次数有很大的影响.本文首先尝试了国际上著名的并行图着色算法 JPL[24]、CC[25]等,基于国产处理器进行了并行实现并将其运用于 HPCG 中.对 JPL 算法,本文尝试了贪心着色策略和随机着色策略.对 CC 算法,本文也进行了多种尝试,如每轮迭代的遍历次数选为 1,2,3,13,27 这 5 种.对于 256×256×256 的测试规模,迭代次数介于 58~67 之间,且最低迭代次数时颜色数为 70.这些算法的测试结果无论颜色数还是迭代次数都偏高.

经过分析可知,传统的 JPL 和 CC 算法只考虑一层依赖,即着色时每个网格点只考虑其周围最临近网格点的依赖关系,而实际上依赖关系是会传递的,如图 7 中,1 号点和 2 号点有依赖,2 号点和 3 号点有依赖,则 1 号点和 3 号点也有依赖.考虑这种依赖传递将有可能在并行算法中保持更多的依赖关系,从而降低迭代次数.为了降低迭代次数的同时减少颜色数(增加并行度),本文还考虑了 x,y,z 这 3 个方向的差异,设计了一些实验,验证了这 3 个方向有一个为强依赖方向,其他两个为弱依赖方向.对强依赖方向使用较多的颜色,对弱依赖方向使用较少的颜色,期望整体颜色数尽量减少.最终本文使用了 33 色对整个网格进行着色.经测试,对于 256×256×256 规模,单进程运行时迭代次数为 55 次,相比于 JPL 和 CC 算法的最少迭代次数降低了 3 次,即整体性能可提升 6%.

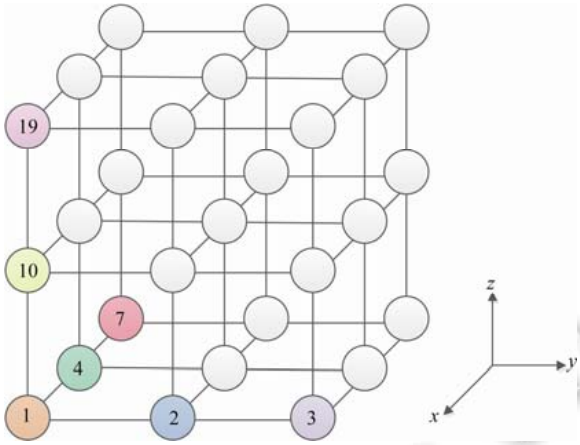


Fig.7 Propagation of dependence

图 7 依赖传递

5 异构部件任务划分

国产超级计算机每个节点包括 CPU、GPU 等部件,CPU、GPU 均拥有独立的内存.HPCG 的数据生成部分耗时较多,本文使用 GPU 进行矩阵、向量和其他信息生成,并拷贝回 CPU 进行参考版的计算.所有优化版需要的矩阵、向量等常驻 GPU 设备内存.整个任务划分时需考虑 CPU 和 GPU 的任务分工.由于 GPU 拥有比 CPU 更高的访存带宽,更多的计算资源,所以应该尽量把任务放到 GPU 上计算.这样就有两种可能的方案:

1. 将所有计算任务全部放到 GPU 上,CPU 不参与计算,只负责通信.
2. 将大部分计算任务放到 GPU 上,CPU 并行执行小部分计算任务同时负责通信.

第 2 种方案实际执行时需要 CPU 和 GPU 不断地传输新的计算数据,而这种传输开销较大,传输 256×256×256 规模的向量 x 约需 0.035s,而计算一次 SpMV 的时间仅为 0.008 78s,传输开销远大于计算开销,得不偿失.本文采用第 1 种方案,并且通过内外区划分将 CPU 通信与 GPU 内区的计算进行重叠,如图 8 所示,具体介绍见第 7 节.

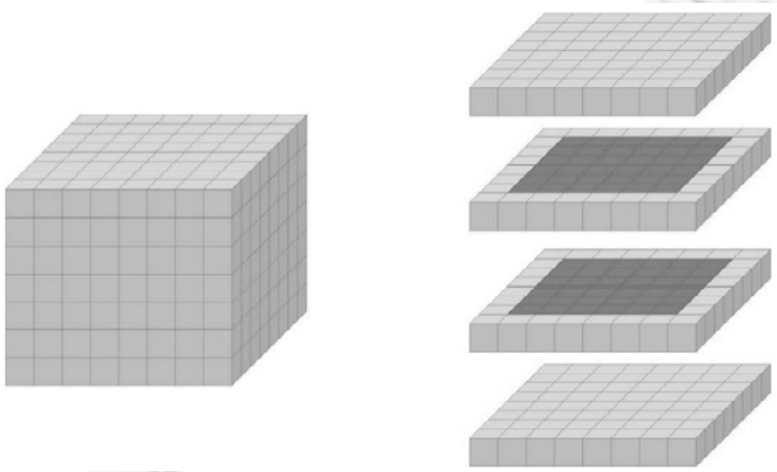


Fig.8 Partitioning of inner areas and outer areas

图 8 内外区划分

6 HPCG 优化方法

6.1 稀疏矩阵存储格式

HPCG 参考版本里矩阵存储采用的是 CSR 格式,如图 9 所示,矩阵的非零元素和其对应的列索引分别存放在各自的数组里,并用一个指针数组记录每一行的起始位置.这样数据的内存分配比较离散,局部性不高.为了解决这个问题,我们采用了如图 10 所示的 ELL 格式,按照矩阵中非零元最多的行对其他行进行填充,使每行的长度一样,这样数据在内存中占据了一块连续的空间,在知道这个长度和矩阵起始地址的情况下可以算出每一行的起始位置,节省了 CSR 格式中每行起始位置的索引数组.

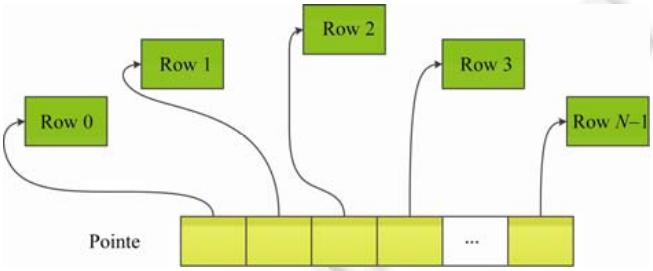


Fig.9 Storage format used by the reference version of HPCG

图 9 HPCG 参考版使用的存储格式

Fig.10 Optimized ELL format

图 10 优化 ELL 格式

对规模为 256×256×256 的矩阵而言,总非零元个数为 450 629 374,CSR 格式总存储量为 5.20 GB,ELL 格式总存储量为 5.06 GB,减少了 3%.

6.2 稀疏矩阵重排

由于使用了分块图着色算法,相同颜色块一起计算,而在原存储空间中,相同颜色的块都是不相邻的,这样无法充分利用 *L*2 Cache,因此,我们根据分块着色的结果对矩阵进行了重排,如图 11 所示,我们将相同颜色的块放在一起,进一步提高数据局部性.相应的,我们对解向量 *x* 和右端项 *y* 也做了同样的重排.

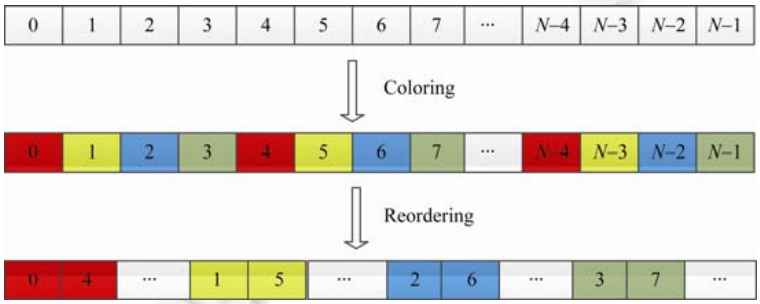


Fig.11 Blocked graph coloring and reordering

图 11 分块图着色以及重排

6.3 其他优化

分块后,块内的数据存在可能的复用,本文可以通过 GPU 卡上的程序员可控高速缓存 LDS 来完成这部分的复用,从而进一步提高访存的效率.

除此之外,本文还运用了多种优化手段,如分支消除、循环展开、数据预取、数据管理优化等.在分支消除方面,考虑到 GPU 的分支执行特性,本文通过逻辑计算来消除了一些分支指令.在循环展开方面,综合考虑寄存器的数量与性能,本文展开了一些热点函数中的循环.在数据预取方面,本文在计算时先统一预取不规则的内存区域中的数据.数据管理方面,本文采用了内存池的方式对内存进行管理,避免了很多不必要的内存申请与释放,并且在主存分配空间时都声明为 pinned memory 以加速数据拷贝.

7 多进程实现与优化

在多进程的实现中,本文采用了内外区划分的方式,将整个计算网格分成内区和外区两个部分,外区宽度为 1,如图 8 所示.将内区按照第 4 节中算法进行着色,将外区看成最后一个颜色.当内区部分所有的计算和外区所需的 halo 区数据通信都完成后开始进行外区的计算.具体流程如图 12 所示.这样整个 HPCG 中核心函数 SpMV、SymGS 中的邻居通信均可被内区计算掩盖,从而减少了整体运行时间.

由于矩阵、向量等常驻 GPU 设备内存,邻居通信前需要将 halo 区的数据先拷贝回 CPU 内存,待通信结束后再拷贝到 GPU 内存.第 1 次拷贝是无法重叠的,因为只有等完全拷贝结束后才能进行邻居通信.

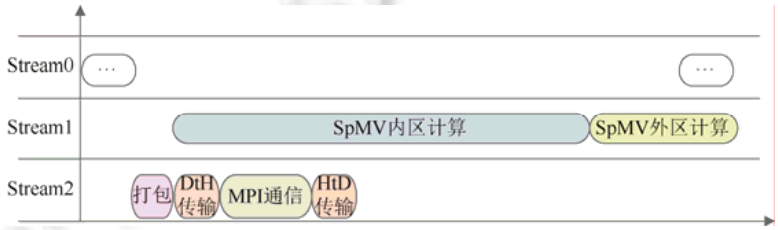


Fig.12 Overlap of computation and communication

图 12 计算通信重叠

8 HPCG 测试结果

本文对第 3.1 节和第 3.2 节的算法均进行了实现,并基于某国产复杂异构超级计算机单节点单进程进行了测试,测试结果如图 13 所示.图 13 中给出了 HPCG 参考版、两个并行方案的性能结果,对第 2 个方案,还给出了多个优化技术所带来的性能提升.从图中可以看出,第 3.2 节算法性能较高(见图中 para2),约是第 3.1 节算法(见图中 para1)的 2 倍多,因为第 3.2 节算法能更好的发挥 GPU 的计算能力,并行度更高.图中 opt1 是将 setup 部分和 OptimizeProblem 部分进行优化的结果,同时该版本还减少了不必要的同步操作,opt2 是对多重网格中每一层的着色方案进行自适应调整,以增加粗层的并行度,另外该版本中还包括对零元访存的优化.opt2 版本的性能是本文工作单节点可达到的最高性能,在做完内外区划分后,该性能有所下降,见图中 inner-outer 部分.

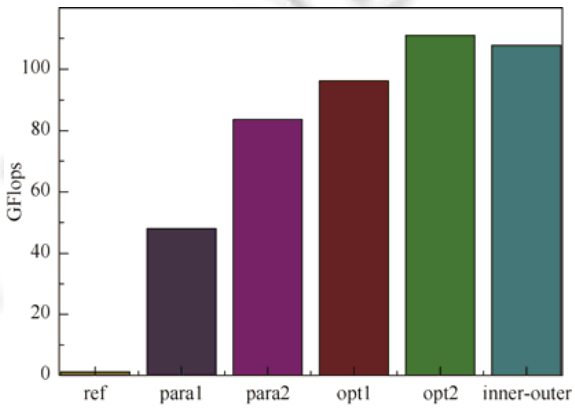


Fig.13 HPCG performance using single process

图 13 HPCG 单进程性能

本文基于某国产复杂异构超级计算机系统进行测试,分别测试了单节点计算效率和整机计算效率,并测试了整机的弱可扩展性,最终单节点计算效率达到了 1.82%,整机计算效率达到了 1.67%,整机弱可扩展性并行效率达到了 92%.

9 结论及下一步工作

本文面向国产复杂异构超级计算机研制了异构众核并行 HPCG 软件,从着色算法、异构任务划分、稀疏矩阵存储格式等角度展开研究,提出了一套适用于结构化网格的着色算法,用于 HPCG 后,着色质量与现有算法相比有明显的提升.通过分析异构部件的传输开销,提出了一套异构任务划分方法,并采用 ELL 格式存储稀疏矩阵,以减少访存开销.同时采用分支消除、循环展开、数据预取等多种优化方式进行了优化.在多进程实现时,为了尽可能地提升整机性能,本文还采用了内外区划分的算法,将邻居通信与内区计算进行重叠,以隐藏通信开销,提高并行效率.最终整机计算效率达到了峰值性能的 1.67%,弱可扩展性并行效率更是提高到了 92%.下一步将面向其他国产超级计算机开展 HPCG 的并行与优化工作,研究 HPCG 混合精度算法,并与相关应用进行合作,提升应用整体性能.

References:

[1] Dongarra J, Heroux MA. Toward a new metric for ranking high performance computing systems. Technical Report, SAND2013-4744, Sandia National Laboratories, 2013.

[2] Dongarra JJ, Luszczek P, Petitet A. The LINPACK benchmark: Past, present and future. Concurrency and Computation Practice & Experience, 2003,15(9):803−820.

[3] https://www.isc-hpc.com/

[4] http://www.supercomputing.org/

[5] Haidar A, Tomov S, Dongarra J, *et al*. Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers. In: Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC 2018). IEEE, 2018. 603−613.

[6] Haidar A, Wu P, Tomov S, *et al*. Investigating half precision arithmetic to accelerate dense linear system solvers. In: Proc. of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-scale Systems. ACM, 2017. 1−8.

[7] Bell N, Garland M. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In: Proc. of the Conf. on High Performance Computing Networking, Storage and Analysis. 2009. 1−11.

[8] Vazquez F, Fernandez J, Garzon E. A new approach for sparse matrix vector product on NVIDIA GPUs. Concurrency and Computation: Practice and Experience, 2011,23(8):815−826.

[9] Ashari A, Sedaghati N, Eisenlohr J, *et al*. An efficient two-dimensional blocking strategy for sparse matrix-vector multiplication on GPUs. In: Proc. of the 28th ACM Int'l Conf. on Supercomputing. ACM Press, 2014. 273−282.

[10] Yan S, Li C, Zhang Y, *et al*. yaSpMV: Yet another SpMV framework on GPUs. ACM SIGPLAN Notices, 2014,49(8):107−118.

[11] Pichel JC, Rivera FF, Fernández M, *et al*. Optimization of sparse matrix-vector multiplication using reordering techniques on GPUs. Microprocessors and Microsystems, 2012,36(2):65−77.

[12] Tang WT, Tan WJ, Ray R, *et al*. Accelerating sparse matrix-vector multiplication on GPUs using bit-representation-optimized schemes. In: Proc. of the Int'l Conf. on High Performance Computing, Networking, Storage and Analysis. 2013. 1−12.

[13] Williams S, Oliker L, Vuduc R, *et al*. Optimization of sparse matrix–vector multiplication on emerging multicore platforms. Parallel Computing, 2009,35(3):178−194.

[14] Guo D, Gropp W. Adaptive thread distributions for SpMV on a GPU. In: Proc. of the Extreme Scaling Workshop. 2012. 1−5.

[15] Park J, Smelyanskiy M, Vaidyanathan K, *et al*. Efficient shared-memory implementation of high-performance conjugate gradient benchmark and its application to unstructured matrices. In: Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC 2014). IEEE, 2014. 945−955.

[16] Iwashita T, Nakanishi Y, Shimasaki M. Comparison criteria for parallel orderings in ILU preconditioning. SIAM Journal on Scientific Computing, 2005,26(4):1234−1260.

[17] Kumahata K, Minami K, Maruyama N. High-performance conjugate gradient performance improvement on the K computer. The Int'l Journal of High Performance Computing Applications, 2016,30(1):55−70.

[18] Phillips E, Fatica M. A CUDA implementation of the high performance conjugate gradient benchmark. In: Proc. of the Int'l Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems. Cham: Springer-Verlag, 2014. 68−84.

[19] Liu YQ, Zhang XY, Yang C, *et al*. Accelerating HPCG on Tianhe-2: A hybrid CPU-MIC algorithm. In: Proc. of the 20th IEEE Int'l Conf. on Parallel and Distributed Systems (ICPADS). IEEE, 2014. 542−551.

[20] Liu YQ. Research on key technologies of communication intensive kernels for Intel MIC architecture [Ph.D. Thesis]. Beijing: Institute of Software, Chinese Academy of Sciences, 2015 (in Chinese with English abstract).

[21] Ao YL. Research on key optimizations of sparse matrix and stencil computation for the domestic large many-core system [Ph.D. Thesis]. Beijing: Institute of Software, Chinese Academy of Sciences, 2017 (in Chinese with English abstract).

[22] Ruiz D, Mantovani F, Casas M, *et al*. The HPCG benchmark: Analysis, shared memory preliminary improvements and evaluation on an Arm-based platform. 2018. https://upcommons.upc.edu/bitstream/handle/2117/116642/1HPCG_shared_mem_implementation_tech_report.pdf?sequence=8&isAllowed=y

[23] Greathouse JL, Daga M. Efficient sparse matrix-vector multiplication on GPUs using the CSR storage format. In: Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage and Analysis. IEEE Press, 2014. 769−780.

[24] Jones MT, Plassmann PE. A parallel graph coloring heuristic. SIAM Journal on Scientific Computing, 1993,14(3):654−669.

[25] Cohen J, Castonguay P. Efficient graph matching and coloring on the GPU. In: Proc. of the GPU Technology Conf. 2012. 1−10.

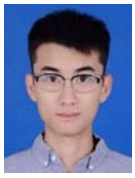
附中文参考文献:

[20] 刘益群.MIC 众核架构通信密集型函数的算法设计与性能优化研究[博士学位论文].北京:中国科学院软件研究所,2015.

[21] 敖玉龙.国产大型众核系统上稀疏矩阵和 Stencil 运算的性能优化关键技术研究[博士学位论文].北京:中国科学院软件研究所,2017.

刘芳芳(1982－),女,博士,正高级工程师,CCF 专业会员,主要研究领域为高性能扩展数学库,稀疏迭代解法器,异构众核并行.

马文静(1981－),女,博士,副研究员,CCF 专业会员,主要研究领域为高性能计算.



王志军(1995－),男,硕士,主要研究领域为高性能计算,并行计算.

杨超(1979－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为高性能计算,科学与工程计算.

汪荃(1996－),女,硕士,主要研究领域为并行计算.

孙家昶(1942－),男,研究员,博士生导师,主要研究领域为科学与工程计算的方法、理论与应用,并行计算.

吴丽鑫(1994－),女,硕士,主要研究领域为并行计算.