计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(11):358−364 [doi: 10.15888/j.cnki.csa.008774]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

格点 QCD 基础求解器及其异构计算实现的性能优化①

杨子江[1,2], 张克龙[1], 刘　倩[1,2], 徐　顺[1], 孙　鹏[3]

[1](中国科学院 计算机网络信息中心, 北京 100190)
[2](中国科学院大学, 北京 100049)
[3](南京师范大学, 南京 210023)
通信作者: 张克龙, E-mail: klzhang@cnic.cn; 刘　倩, E-mail: liuqian@sccas.cn

摘　要: 格点量子色动力学 (格点 QCD) 是研究夸克、胶子等微观粒子间相互作用的重要理论和方法. 通过将时空离散化为四维结构网格, 并将量子色动力学的基本场量定义在网格上, 让研究人员可以使用数值模拟方法, 从第一性原理出发研究强子间相互作用和性质, 但这个过程中的计算量极大, 需要进行大规模并行计算. 格点 QCD 计算的核心基础为格点 QCD 求解器, 是程序运行主要的计算热点模块. 本文研究在国产异构计算平台下格点 QCD 求解器的实现与优化, 提出一套格点 QCD 求解器的设计实现, 实现了 BiCGSTAB 求解器, 显著降低了迭代次数; 通过对奇偶预处理技术, 降低了所求问题的计算规模; 针对国产异构加速卡的特点, 优化了 Dslash 模块的访存操作. 实验测试表明, 相比优化前的求解器获得了约 30 倍的加速比, 为国产异构超算下格点 QCD 软件性能优化提供了有益的参考价值.

关键词: 格点量子色动力学; 方程求解器; 并行计算; 异构计算

引用格式: 杨子江,张克龙,刘倩,徐顺,孙鹏.格点 QCD 基础求解器及其异构计算实现的性能优化.计算机系统应用,2022,31(11):358–364. http://www.c-s-a.org.cn/1003-3254/8774.html

Performance Optimization of Lattice QCD Solver and Its Heterogeneous Computation

YANG Zi-Jiang[1,2], ZHANG Ke-Long[1], LIU Qian[1,2], XU Shun[1], SUN Peng[3]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)
[3](Nanjing Normal University, Nanjing 210023, China)

Abstract: Lattice quantum chromodynamics (Lattice QCD) is an important theory and method to study the interaction between microscopic particles such as quarks and gluons. By discretizing the spacetime into a four-dimensional structural grid and defining the basic field quantity of QCD on the grid, researchers can use a numerical simulation method to study hadron interactions and properties from the first principle. However, the computation in this process is time-consuming, and large-scale parallel computing is required. The fundamental module of the Lattice QCD computation is the Lattice QCD solver which is the main hot spot of the program running. This work studies the realization and optimization of Lattice QCD solver from a domestic heterogeneous computing platform and proposes a design method of Lattice QCD solver, which realizes BiCGSTAB solver and significantly reduces the iteration numbers. With the odd/even pre-processing technology, the study reduces the computing scale of the problem and optimizes the Dslash module's memory access in terms of the characteristics of a domestic heterogeneous accelerator. Experimental tests show that the speedup ratio of the solver is about 30 times higher than that of the unoptimized one, which provides a useful reference for the performance optimization of Lattice QCD software of domestic heterogeneous supercomputers.

Key words: Lattice quantum chromodynamics (Lattice QCD); equation solver; parallel computing; heterogeneous computing

① 基金项目: 中国科学院 B 类先导培育项目 (XDPB25); 海光产业生态合作组织基金 (ghfund202107011598)
收稿时间: 2022-01-29; 修改时间: 2022-03-16; 采用时间: 2022-03-30; csa 在线出版时间: 2022-07-29

1 引言

强相互作用是自然界中 4 种 (强、弱、电磁、引力) 基本相互作用之一. 1974 年, Wilson 建立的格点量子色动力学 (Lattice quantum chromodynamics, 格点 QCD) 规范理论[1] 是从第一性原理出发研究强相互作用的基本理论和方法, 尤其是在高能物理的低能区, 格点 QCD 是当前已知的最为系统最为可靠的非微扰理论方法, 对高能物理实验和理论的研究都有着重要意义.

格点 QCD 算法与软件的研究是利用该方法开展强子谱学、核子结构、QCD 相变、标准模型精确检验和新物理寻等高能物理研究的基础支撑.

在国际上, 格点 QCD 作为重要的高性能计算应用, 并且形成诸多优秀的模拟计算软件. 例如 21 世纪初美国 USQCD 的 Chroma[2], 以及 2010 年发展出的 GPU 异构加速计算库 QUDA[3,4]; 日本从 2009 年开始研发适应日本格点计算研究需求的计算软件 Bridge++[5,6], 而后发展出面向日本超级计算机“富岳”的格点 QCD 的计算库 QWS[7,8]; 近年来英国的 Grid[9,10] 在数据处理的矢量化设计上表现突出, 对于不同架构的机器具有较好适应性.

目前, 随着国内高性能计算机的发展, 国内格点 QCD 研究团队积极开展面向国产超算平台的格点计算软件研发工作. 在申威、天河等不同超算平台上探索式地进行了软件移植和自主开发工作[11–16].

本团队致力于使用 HIP 框架开发针对国产 ROCm 异构平台的格点 QCD 异构计算程序库 Openchiral, 首次在 HIP 框架下自主实现了国产 ROCm 环境的格点 QCD 计算所需基本核心功能模块. 格点 QCD 计算的核心热点模块为有限差分的狄拉克 (Dirac) 方程求解, 也即 QCD 传播子的求解过程, 下称求解器, 其热点运算为 Dslash 运算 (参见第 2.1 节). 本文工作首先实现了格点 QCD 相关的矩阵向量操作、基本核函数 Dslash 和共轭梯度 (conjugate gradient, CG) 求解器出发, 而后逐步进行算法改进和计算优化, 实现了在 ROCm 国产异构平台上高效稳定的格点 QCD 传播子求解器.

2 格点 QCD 求解器及基本流程

格点 QCD 是定义在$N_x \times N_y \times N_z \times N_t$四维结构网格上的理论和方法. 在格点上定义费米子场$\varphi(\tilde{x})$, 格点之间链接上定义规范场$U_\mu(\tilde{x})$, 这里$\tilde{x}$表示四维时空坐标, μ表示四维时空方向. 每个网格点是一个张量点, 是由 QCD 色空间和旋量空间张成的 3⊗4 维空间. 计算的核心热点模块为有限差分的狄拉克方程求解, 也即 QCD 传播子的求解过程, 下称求解器. 其形式上类似于有限差分方程$Ax = b$的求解.

2.1 Dslash 运算

本文以基本的格点 QCD 的 Wilson 费米子作用量为例, 对应的求解方程形式如下:

$$\left[\gamma_\mu D_\mu + m\right]\varphi = M\varphi = \mathrm{b} \tag{1}$$

其中, M称为费米子矩阵, 是一个规模庞大的稀疏矩阵, 其维数规模为$N_x \times N_y \times N_z \times N_t \times 12$; 费米子矩阵$M$具体形式为:

$$M = (4+m)\delta_{\tilde{x},\tilde{y}} - \frac{1}{2}\sum_{\mu=0}^{3}\left[(1-\gamma_\mu)U_\mu(\tilde{x})\delta_{\tilde{x}+\mu,\tilde{y}} + (1+\gamma_\mu)U_\mu^\dagger(\tilde{x}-\mu)\,\delta_{\tilde{x}-\mu,\tilde{y}}\right] \tag{2}$$

其中, γ_μ是确定的四维狄拉克矩阵, $U_\mu(\tilde{x})$是$SU(3)$矩阵. 由式 (2) 可知, 该计算是四维时空结构网格上的四维 9 点的模板 (stencil) 计算, 点与点之间的运算涉及3⊗4 维张量计算. 根据如式 (2) 所示的稀疏矩阵M的数学形式, 为了增加数据复用、减少内存占用、降低访存压力, 实现中不用直接存储高维的M矩阵, 而是在计算过程中利用组态数据$U_\mu(x)$构建费米子矩阵乘向量操作$M\varphi$, 称为 Dslash 操作, 该运算是迭代求解中核心热点运算.

2.2 格点 QCD 求解器

稳定高效的 Dslash 运算和求解器将会显著提升格点 QCD 计算的整体性能. 通常使用的共轭梯度等迭代求解算法需调用第 2.1 节所述的 Dslash 操作, 现有的稀疏矩阵求解库难以满足需求, 通常需要为其专门实现一套求解器.

格点 QCD 求解器模块工作流程如下:

1) 初始化费米子数据.

2) 加载组态数据.

3) 开始迭代求解.

4) 对求出的解进行验算.

计算流程图如图 1 所示, 待求解矩阵M由组态数据与费米子场数据构建, 不直接存储.

程序中主要包含以下模块:

1) 数据初始化模块: 初始化费米子场、组态数据.

2) 迭代求解模块: 使用费米子场、组态数据构造所求的矩阵方程$Mx = b$, 并对其进行迭代求解.

3) Dslash 模块: 在异构加速卡完成迭代求解过程中的矩阵向量乘运算.

4) 验算模块: 对求解器给出的解进行验算, 确认正确性.

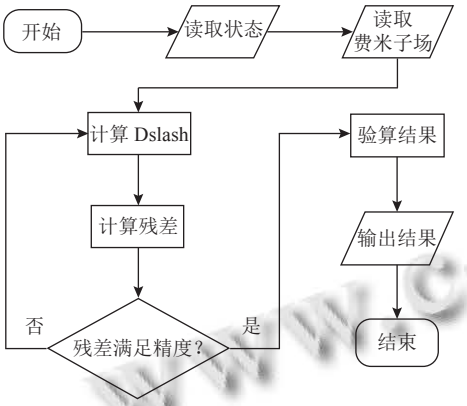


图 1 求解流程示意图

3 优化方案设计与实现

在基于 HIP 框架实现面向国产 ROCm 异构平实现核心运算 Dslash 基础上, 进一步通过算法改进和异构访存优化, 用以提高收敛速度、降低问题规模. 具体优化方案如下:

1) 在实现 CG (算法 1) 求解器的基础之上实现 BiCGSTAB (算法 2) 求解器, 相对于 CG 求解器迭代次数更少、收敛速度更快.

2) 采用奇偶预处理方法降低求解问题的规模, 进一步减少迭代次数并提高每轮迭代的速度.

3) 在异构访存优化方面, 通过 Dslash 合并异构访存与改用锁页内存方式, 减少每轮迭代时 CPU 与加速卡的通讯开销, 降低每轮迭代耗时.

算法 1. CG 算法

procedure CG ($M^{\dagger}M$, x, b)
  r_0:=$M^{\dagger}b-M^{\dagger}Mx_0$
  p_0:=r_0
  for i=0, 1, 2, ⋯, do
    $\alpha_i:=\frac{r_i\cdot r_i}{(M^{\dagger}Mp_i)\cdot p_i}$ /*计算步长*/
    $x_{i+1}:=x_i+\alpha_i p_i$ /*修正解向量*/
    $r_{i+1}:=r_i-\alpha_i M^{\dagger}Mp_i$ /*计算残差向量*/
    if $\|r\|<\varepsilon$ then break /*判断是否收敛*/
    end if
    $\beta_i:=\frac{r_{i+1}\cdot r_{i+1}}{r_i\cdot r_i}$
    $p_{i+1}:=r_{i+1}+\beta_i p_i$ /*计算梯度方向*/
  end for
  $x:=x_{i+1}$
  return x
end procedure

算法 2. BiCGSTAB 算法

procedure BiCGSTAB (M, x, b)
  $r_0:=b-Mx_0$
  r_0':=r_0
  p_0:=r_0
  for i=0, 1, 2, ⋯, do
  $a_i:=\frac{r_i\cdot r_0'}{(Mp_i)\cdot r_0'}$/*计算步长*/
    $s_i:=r_i-\alpha_i Mp_i$
    $\omega_i:=\frac{(Ms_i)\cdot s_i}{(Ms_i)\cdot(Ms_i)}$
    $x_{i+1}:=x_i+\alpha_i p_i+\omega_i s_i$ /*修正解向量*/
    $r_{i+1}:=s_i-\omega_i Ms_i$ /*计算残差向量*/
    if $\|r\|<\varepsilon$ then break /*判断是否收敛*/
    end if
    $\beta_i:=\frac{\alpha_i}{\omega_i}\times\frac{r_{i+1}\cdot r_0'}{r_i\cdot r_0}$
    $p_{i+1}:=r_{i+1}+\beta_i(p_i-\omega_i Mp_i)$ /*计算梯度方向*/
  end for
  $x:=x_{i+1}$
  return x /*返回结果*/
end procedure

3.1 提高收敛速度: 实现 BiCGSTAB 求解器

相对于 CG 算法 (算法 1), BiCGSTAB 算法 (算法 2) 能够求解非正定矩阵. 更重要的是, BiCGSTAB 可以减少迭代次数, 提升计算效率. 在第 i 轮迭代中, 记$\rho_i = (I-\omega_i M)$, r_i^B、r_i^G分别为 BiCGSTAB、CG 算法的残差, 则有:

$$\begin{aligned} r_i^B &= \rho_i(M)r_i^G = (I-\omega_i M)\rho_{i-1}(M)r_i^G \\ &= (I-\omega_i M)(I-\omega_{i-1}M)\rho_{i-2}(M)r_i^G \\ &= \cdots = \prod_{k=0}^{i}(I-\omega_k M)r_i^G \end{aligned} \tag{3}$$

由式 (3) 可知当$r_i^G = 0$时, $r_i^B = 0$, 表明二者具有相同的收敛性与精度, 额外的相乘操作$\rho_i(M)$使得 BiCGSTAB 每轮迭代更加平滑, 算法收敛到 0 的速度更快, 迭代次数下降.

在迭代求解过程中, 主要的热点是 Dslash 操作. BiCGSTAB 需要计算Mp_i、Ms_i, 故每轮迭代共需进行两次 Dslash 操作. CG 需计算Mp_i, 但由于格点 QCD 的

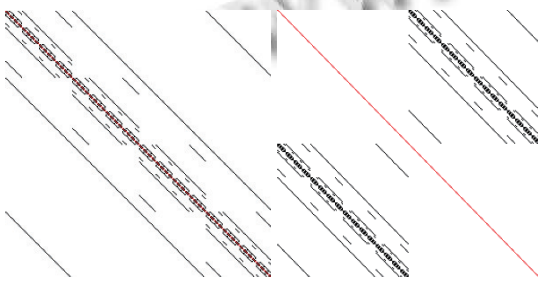
费米子矩阵M不保证正定, 使用 CG 求解时方程两侧需要左乘$M^{\dagger}$. 以保证待求矩阵的正定性, 即$M^{\dagger}Mx=M^{\dagger}b$, 所以 CG 和 BiCGSTAB 每轮迭代均需要进行两次 Dslash 操作. 由此可以得知, BiCGSTAB 每轮迭代开销与 CG 大致相同, 求解速度的提升主要来自更少的迭代次数.

3.2 降低问题规模: 奇偶预处理

根据格点 QCD 离散化的时空结构网格和费米子矩阵类差分形式, 对费米子矩阵M和对应的费米子场向量做奇偶预处理 (even-odd precondition), M矩阵会可以划分为 4 块子矩阵, 即:

$$M=\begin{bmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{bmatrix} \tag{4}$$

其中, M_{ee}和M_{oo}是实对角部分, M_{eo}和M_{oe}是非对角部分. 奇偶预处理前后矩阵元分布如图 2 所示.



(a) 处理前矩阵元分布 (b) 处理后矩阵元分布

图 2 费米子矩阵奇偶预处理示意图

经过奇偶预处理后的费米子矩阵 M, 可以做如下形式的分解:

$$M=L\tilde{M}U \tag{5}$$

其中, $\tilde{M}$为块对角矩阵.

$$\tilde{M}=\begin{bmatrix} M_{ee} & 0 \\ 0 & M_{oo}-M_{oe}M_{ee}^{-1}M_{eo} \end{bmatrix} \tag{6}$$

因此, 经过奇偶预处理后, 式 (1) 所示的线性求解问题$M\varphi=b$变为如下形式:

$$\begin{cases} L\tilde{M}U\varphi=b \\ \tilde{M}\varphi'=b' \end{cases} \tag{7}$$

其中, $b'=L^{-1}b$, $\varphi=U^{-1}\varphi'$. L 和 U 形式分别如下:

$$L=\begin{bmatrix} 1 & 0 \\ M_{oe}M_{ee}^{-1} & 1 \end{bmatrix},\ U=\begin{bmatrix} 1 & M_{ee}^{-1}M_{eo} \\ 0 & 1 \end{bmatrix}$$

其逆矩阵L^{-1}和U^{-1}很容易从理论上推导得到而不需要对 L 和 U 进行数值求逆, 其形式为:

$$L^{-1}=\begin{bmatrix} I & 0 \\ -M_{oe}M_{ee}^{-1} & I \end{bmatrix},\ U^{-1}=\begin{bmatrix} I & -M_{ee}^{-1}M_{eo} \\ 0 & I \end{bmatrix}$$

其中, M_{ee}和M_{oo}均为对角矩阵.

$$M_{oo}^{-1}=M_{ee}^{-1}=1/(4+m)$$

因此, 经过奇偶预处理后, 需要数值迭代求解仅为式 (6) 和式 (7) 下半部分, 仍记为$\tilde{M}\varphi'=b'$. $\tilde{M}$的维度是M的 1/2, 因此降低了问题规模, 可以有效减少迭代次数. 同时, 由于$\tilde{M}$减小, 每次 Dslash 操作的开销也大幅降低, 能够降低每轮迭代的耗时.

3.3 异构访存优化

3.3.1 Dslash 操作合并异构访存

在迭代求解过程中, Dslash 操作是计算热点, 提升这部分的计算效率可以显著提升迭代求解的速度. 我们基于 HIP 框架实现了 Dslash 的异构计算加速, 优化了异构计算访存.

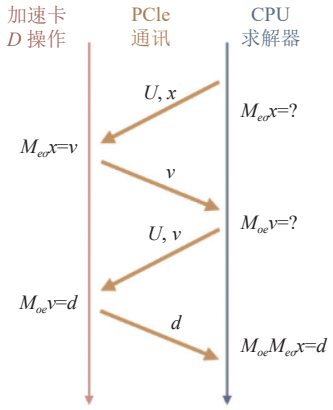
经过奇偶预处理后, 迭代求解中的 Dslash 操作的内容就变为了 $M_{oe}M_{eo}x$, 也就是连续两次 Dslash 操作, M_{oe}、M_{eo}分别是同一矩阵M的非对角区域. 每次 Dslash 操作会在加速卡显存、主存之间传输两次, 如图 3(a) 所示. 令$v=M_{eo}x$, $w=M_{oe}v$, 则第 1 次 Dslash 操作$v=M_{eo}x$需要从主存传输组态 U、向量x; 从显存传回计算结果向量v. 第 2 次 Dslash 操作需要从主存传输组态 U、向量v; 从显存传回计算结果向量w. 故连续两次 Dslash 操作就要在主存、显存间传输 4 次.

由于费米子矩阵元素数量是所依赖数据元素数量的 12 倍, 所以 Dslash 操作不直接构建和存储稀疏矩阵 M, 而是采用模板计算, 在计算时使用组态数据 U 生成待计算的矩阵元素, 故前后两次 Dslash 所需的组态数据 U 相同. 又因连续 Dslash 操作上一次运算的结果即为下一次运算的输入, 因此可以将 4 次访存合并为 2 次: 图 3 (b), 当前一次 Dslash 操作结束后, 将两次计算共同依赖的数据 U 和前一次 Dslash 操作$v=M_{eo}x$的计算结果存储在显存上, 作为下一次 Dslash 操作的输入数据, 节约了 2 次访存操作.

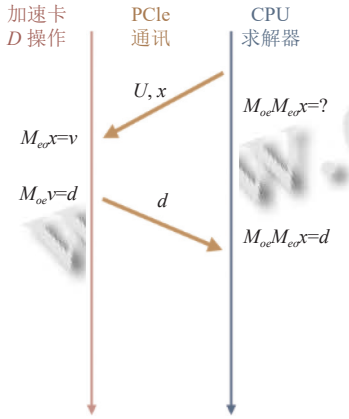
3.3.2 提高异构访存速度

迭代求解时, 每轮迭代进行 Dslash 操作时都需要通过 PCIe 接口向加速卡传输组态数据与费米子场数据, 这一操作是由 HIP 框架提供的 API 完成的.

默认情况下 C、C++中使用 new 或 malloc 所分配的内存空间是可分页内存. 在 HIP 框架中, 从主存向显存传输数据前必须先分配临时页锁定数据, 再传输到显存, 如图 4(a) 所示.

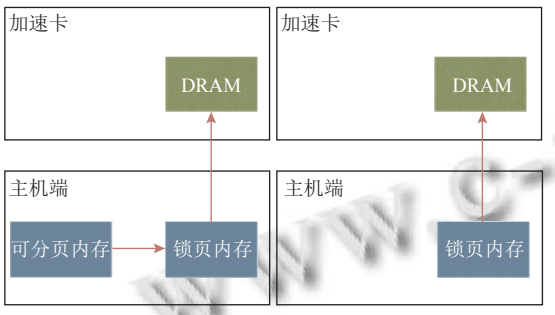


(a) 处理前矩阵元分布



(b) 处理后矩阵元分布

图 3 两次 Dslash 操作数据传输优化示意图



(a) 可分页内存 (b) 锁页内存

图 4 可分页内存与锁页内存通讯示意图

HIP 框架提供了分配锁页内存的功能, 即 hipHostMalloc() 函数. 相比可分页内存, 将待传输的数据存入锁页内存省去了锁页与拷贝的步骤, 可以加快数据传输速度, 如图 4(b) 所示. 在格点 QCD 计算迭代求解时, 需要传输的数据为费米子场向量与规范场数据. 将费米子与规范场的数据存储在锁页内存, 可以大幅提高加速卡与内存通讯的速度.

4 实验分析

本工作完成了以上几种优化手段的代码实现与性能测试. 在中科先导一号计算平台上, 优化后的求解程序相比最初的程序大体达到了 30 倍的加速比.

为适应不同问题规模, 本论文以下性能测试为弱扩展性测试, 具体测试规模如表 1 所示: 每节点的负载不变, 总计算量随节点数成比例扩展. 程序测试的规模以格子的x、y、z、t指标体现. 每张加速卡承担的计算任务规模为子格子大小, 设为16^4; 总格子规模为总计算任务规模, 随加速卡数增加. 令加速卡数为2^c, 总格子规模为16×2^x、16×2^y、16×2^z、16×2^t, 则有$c=x+y+z+t$.

表 1 测试规模

节点数	加速卡数	总格子尺寸
4	16	16×32×32×64
8	32	16×32×32×64
16	64	32×32×64×64
32	128	32×32×64×128
64	256	64×32×64×128

以32卡为例: $32=2^5$, 子格子为16^4, $5=0+1+2+2$, 则总格子各维度规模为16×2^0、16×2^1、16×2^2、16×2^2, 即$16\times32\times64\times64$.

4.1 求解速度优化效果

在表 2 中, 我们给出了不同规模下上述几种优化手段的求解耗时表现, 表格中从左到右优化手段顺次叠加, “锁页内存”即为最终优化后的程序. 可以看出, 充分优化后的程序在不同规模都达到了 30 倍左右的加速比. 图 5 直观地展示、对比了几种优化手段之间的耗时表现.

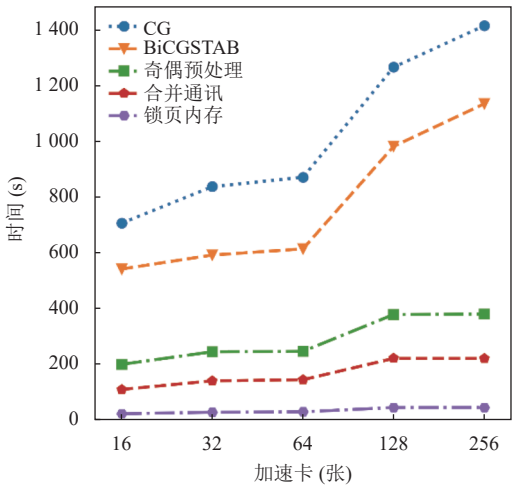


图 5 求解耗时优化示意图

在表 3 中, 给出各优化手段与对应减少的耗时, 从各个方法所优化耗时与最终减少的耗时的占比来看, 奇偶预处理降低了所求解问题的规模, 对耗时优化贡献最大, 平均贡献总优化耗时的 48%; BiCGSTAB 算法减少了迭代次数, 平均贡献了总优化耗时的 26%; 合并访存、锁页内存降低了加速卡访存的开销, 降低了每轮迭代的耗时, 对总优化耗时分别贡献了 12% 与 13%.

4.2 迭代次数优化效果

本文所采取的几种优化手段中, BiCGSTAB 与奇偶预处理有效减少了收敛迭代次数, 接下来将两者与原始程序的迭代收敛表现做对比分析.

在表 4 中, 我们在不同规模下对比了几种收敛优化手段的迭代收敛表现, 从左到右优化手段顺次叠加, 分别为原始程序 (CG)、BiCGSTAB 求解器与奇偶预处理的 BiCGSTAB 求解器. 从中我们可以看出, 原始程序中的 CG 求解器, BiCGSTAB 与奇偶预处理在不同规模下都展现出了大致相同的优化效果. 最终优化后的程序求解收敛所需的迭代次数在不同规模下平均减少为原始程序的 33.4%.

在图 6 中, 我们给出了 3 种算法在 128 卡下求解 $32\times32\times64\times128$时的收敛表现, 纵轴为求得残差, 当残差为 0 时即求解完毕. 曲线越陡峭、越靠左说明算法收敛得越快. 从图中可以直观地看出 BiCGSTAB 与奇偶预处理在迭代过程中的收敛情况.

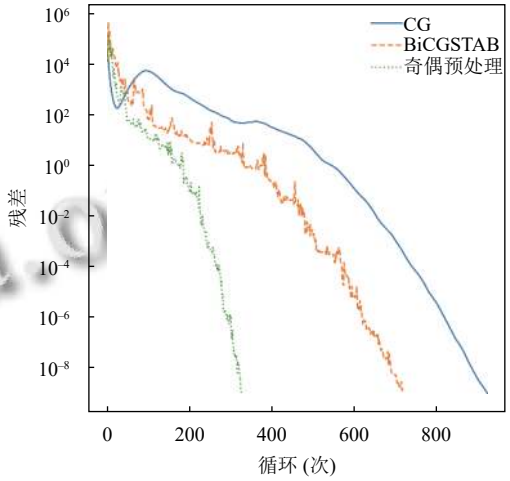


图 6 求解收敛优化示意图 (128 卡)

表 2 求解速度对比

加速卡 (张)	节点数	对比基准	收敛优化		异构访存优化		时间差=	加速比=
		CG (s)	BiCGSTAB (s)	奇偶预处理 (s)	合并访存 (s)	锁页内存 (s)	(CG–锁页内存)	(CG/锁页内存)
16	4	704.37	544.206	198.644	108.764	21.734	682.636	32.408 67
32	8	839.191	590.1	243.575	139.677	27.005	812.186	31.075 393
64	16	869.682	612.295	245.274	143.585	28.601	841.081	30.407 398
128	32	1 288.59	980.349	379.44	220.146	43.915	1 244.675	29.342 82
256	64	1 412.18	1 131.09	378.731	220.037	44.335	1 367.845	31.852 49

表 3 优化表现对比

加速卡 (张)	节点数	收敛优化		异构访存优化		总优化时间 (s)
		BiCGSTAB (s)	奇偶预处理 (s)	合并访存 (s)	锁页内存 (s)	
16	4	160.164	345.562	89.88	87.03	682.636
32	8	249.091	346.525	103.898	112.672	812.186
64	16	257.387	367.021	101.689	114.984	841.081
128	32	308.241	600.909	159.294	176.231	1 244.675
256	64	281.09	752.359	158.694	175.702	1 367.845

表 4 迭代次数对比

加速卡/张	节点数	对比基准	收敛优化		迭代次数差=(CG–奇偶预处理)	加速比=(CG/奇偶预处理)
		CG	BiCGSTAB	奇偶预处理		
16	4	516	396	171	345	3.02
32	8	612	432	209	403	2.93
64	16	634	447	210	424	3.02
128	32	917	713	323	594	2.84
256	64	1 015	817	322	693	3.15

5 结论与展望

本文针对格点 QCD 求解器设计实现, 采取了多项优化措施, 实现了 BiCGSTAB 求解器, 显著减少迭代次数; 采用了奇偶预处理技术, 降低了求解问题的规模;

针对 HIP/ROCM 异构计算平台采取了合并、加速访存等优化方式. 本文还开展了多种规模的并行测试, 最大规模为 64 节点、256 块加速卡, 相比原始程序实现了约 30 倍的加速比.

格点 QCD 是一种重要的高性能计算应用, 性能优化后的程序可以更好地支撑格点 QCD 研究, 充分发挥国产异构超算平台的算力.

参考文献

1 Wilson KG. Confinement of quarks. Physical Review D, 1974, 10(8): 2445–2459. [doi: 10.1103/PhysRevD.10.2445]

2 Edwards RG, Joó B. The chroma software system for Lattice QCD. Nuclear Physics B-Proceedings Supplements, 2004, 140: 832–834.

3 Clark MA, Babich R, Barros K, *et al*. Solving Lattice QCD systems of equations using mixed precision solvers on GPUs. Computer Physics Communications, 2010, 181(9): 1517–1528. [doi: 10.1016/j.cpc.2010.05.002]

4 Babich R, Clark MA, Joó B, *et al*. Scaling Lattice QCD beyond 100 GPUs. Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. Seattle: ACM, 2011. 70.

5 Akahoshi Y, Aoki S, Aoyama T, *et al*. General purpose Lattice QCD code set Bridge++ 2.0 for high performance computing. arXiv: 2111.04457, 2021.

6 Ueda S, Aoki S, Aoyama T, *et al*. Development of an object oriented Lattice QCD code “Bridge++”. Journal of Physics: Conference Series, 2014, 523: 012046. [doi: 10.1088/1742-6596/523/1/012046]

7 Kanamori I, Ishikawa KI, Matsufuru H. Object-oriented implementation of algebraic multi-grid solver for Lattice QCD on SIMD architectures and GPU clusters. Proceedings of the 21st International Conference on Computational Science and Its Applications. Cagliari: Springer, 2021. 218–233.

8 Ishikawa KI, Kanamori I, Matsufuru H, *et al*. 102 PFLOPS Lattice QCD quark solver on Fugaku. arXiv: 2109.10687, 2021.

9 Boyle P, Yamaguchi A, Portelli A, *et al*. Grid: A next generation data parallel C++ QCD library. Proceedings of the 33nd International Symposium on Lattice Field Theory. Kobe: Kobe International Conference Center, 2016.

10 毕玉江, 周超, 吴郁非, 等. 格点量子色动力学 Grid 数值模拟软件的并行计算特征分析. 计算机系统应用, 2020, 29(7): 199–204. [doi: 10.15888/j.cnki.csa.007498]

11 张淼, 周宇, 陈建海, 等. LQCD Dslash 在神威·太湖之光上的研究分析与 MPI 实现. 计算机科学与探索, 2019, 13(10): 1664–1676. [doi: 10.3778/j.issn.1673-9418.1811029]

12 田英齐, 毕玉江, 贺雨晴, 等. 格点量子色动力学组态产生和胶球测量的大规模并行及性能优化. 计算机系统应用, 2019, 28(9): 25–32. [doi: 10.15888/j.cnki.csa.007036]

13 栾钟治, 张增校, 杨海龙, 等. 基于异构众核处理器的格点量子色动力学并行加速方法: 中国, CN201910750655.3. 2019-08-14.

14 Bi YJ, Xiao Y, Guo WY, *et al*. Lattice QCD package GWU-code and QUDA with HIP. Proceedings of the 37th International Symposium on Lattice Field Theory. Wuhan: PoS, 2020.

15 Bi YJ, Xiao Y, Guo WY, *et al*. Lattice QCD GPU Inverters on ROCm Platform. EPJ Web of Conferences, 2020, 245: 09008. [doi: 10.1051/epjconf/202024509008]

16 宫明, 蒋翔宇, 陈莹, 等. 从格点量子色动力学应用看国产超算环境的基础软件. 大数据, 2021, 7(5): 31–39.

(校对责编: 牛欣悦)