物理学报 Acta Physica Sinica
Chinese Physical Society
Institute of Physics, CAS

密度矩阵重正化群的异构并行优化

陈富州 程晨 罗洪刚

Hybrid parallel optimization of density matrix renormalization group method

Chen Fu-Zhou Cheng Chen Luo Hong-Gang

引用信息 Citation: Acta Physica Sinica, 68, 120202 (2019) DOI: 10.7498/aps.68.20190586
在线阅读 View online: https://doi.org/10.7498/aps.68.20190586
当期内容 View table of contents: http://wulixb.iphy.ac.cn

密度矩阵重正化群的异构并行优化*

陈富州[1)] 程晨[1)2)] 罗洪刚[1)2)†]

1) (兰州大学物理科学与技术学院, 兰州 730000)

2) (北京计算科学研究中心, 北京 100084)

(2019 年 4 月 22 日收到; 2019 年 5 月 16 日收到修改稿)

密度矩阵重正化群方法 (DMRG) 在求解一维强关联格点模型的基态时可以获得较高的精度, 在应用于二维或准二维问题时, 要达到类似的精度通常需要较大的计算量与存储空间. 本文提出一种新的 DMRG 异构并行策略, 可以同时发挥计算机中央处理器 (CPU) 和图形处理器 (GPU) 的计算性能. 针对最耗时的哈密顿量对角化部分, 实现了数据的分布式存储, 并且给出了 CPU 和 GPU 之间的负载平衡策略. 以费米 Hubbard 模型为例, 测试了异构并行程序在不同 DMRG 保留状态数下的运行表现, 并给出了相应的性能基准. 应用于 4 腿梯子时, 观测到了高温超导中常见的电荷密度条纹, 此时保留状态数达到 10^4, 使用的 GPU 显存小于 12 GB.

关键词: 密度矩阵重正化群, 强关联格点模型, 异构并行

PACS: 02.70.–c, 71.10.Fd, 71.27.+a, 05.10.Cc **DOI:** 10.7498/aps.68.20190586

1 引 言

密度矩阵重正化群方法 (DMRG)[1,2] 是研究相互作用量子系统最重要的多体数值方法之一. 众所周知, 多体量子系统的复杂度随着系统尺寸指数增长, 给数值计算带来了极大的困难. 而 DMRG 给出了一种非常有效的截断希尔伯特空间的方法, 保留有限个状态并通过扫描即可得到变分收敛的基态或低激发性质. 应用于自旋 1 的海森伯链时, 该方法仅保留数百状态并通过数次扫描即可得到相对误差 10^{-9} 左右的基态能量, 而所需的计算量仅随格点尺寸线性增长 [3,4]. 作为求解一维格点模型基态的成熟方法, DMRG 也不断被应用于其他各类问题并取得了一定的成功, 如动量空间中的哈密顿量 [5] 与量子化学问题 [6–8]、量子系统的时间演化问题 [9–11]、准二维以及二维量子格点模型 [12–14] 等. 这些尝试极大扩展了 DMRG 的应用, 但同时也对该数值算法的优化提出了更高的要求. 以二维相互作用电子系统为例, 该系统包含非常丰富的物理, 例如高温超导条纹相 [15–17]、量子自旋液体 [18,19] 等. 然而, 在面对二维或者准二维格点模型时, DMRG 所需的保留状态数大致随格点宽度指数增长, 得到收敛结果所需的扫描次数也大大增加. 此时, DMRG 所需的计算量和存储量较大, 使用各种方法优化算法显得十分必要.

在实际应用中, 对 DMRG 算法的优化与加速主要体现在两个方面. 一方面, 人们不断改进 DMRG 算法本身以减少计算量, 包括使用各种好量子数对角化小的希尔伯特子空间 [20,21], 使用动态保留状态数节约计算资源 [22,23], 使用好的初始预测波函数减少对角化方法迭代次数 [24], 使用单格点 DMRG 方法减少计算量与内存 [25,26] 等. 另一方面, 人们结合 DMRG 算法的特性, 发挥高性能计算机的并行计算能力进一步缩短计算时间, 包括实空间并行 [27]、共享存储的多核行 [28]、分布式存储的多节点行 [29] 以及 CPU-GPU 异构并行 [30] 等.

相比 CPU, GPU 具有较高的浮点计算能力以

* 国家自然科学基金 (批准号: 11674139, 11834005) 和长江学者和创新团队发展计划 (批准号: IRT-16R35) 资助的课题.

† 通信作者. E-mail: luohg@lzu.edu.cn

及较大的存储带宽, 因此在通用计算中得到了大量的应用. 当前很多求解强关联格点模型的数值方法 (比如: 精确对角化方法[31]、量子蒙特卡罗方法[32]以及张量乘积态方法[33] 等) 都实现了基于 GPU 的并行优化. 在 CPU-GPU 异构并行环境中, 异构并行算法可以同时发挥 CPU 和 GPU 的计算性能, 并且可以实现内存和 GPU 显存的分布式存储, 在一定程度上减小了 GPU 显存容量对计算规模的限制. 最近, Nemes 等[30] 提出了 DMRG 方法的一种 CPU-GPU 异构并行实现, 并将其应用到一维格点模型的基态能量计算. 在保留状态较多时, 最耗时的哈密顿量对角化部分在 GPU 中获得了接近峰值的运算性能. 在对角化过程中, 对占用存储空间较多的 Davidson 方法实现了内存和 GPU 显存的分布式存储, 减小了 GPU 显存的容量限制. 然而与此同时, Davidson 方法在计算中内存和 GPU 显存之间需要通信向量数据, 其性能会受到通信带宽的限制. 另一方面, 该方案中哈密顿量和波函数基于两子块表示, 其子块算符需要的存储量大致相当于目前流行的四子块表示的 d^2 倍 (d 为单格点希尔伯特空间维数), 在实际应用中存在很大的局限性.

在前人工作的基础上, 本文提出了一种新的 CPU-GPU 异构并行优化算法, 并使用四子块表示的 DMRG 超块与波函数. 为了说明该方法的有效性, 将其应用到准二维模型的求解, 获得了不同保留状态数下的基态能量及其性能基准. 在 Hubbard 梯子模型的例子中, 得到了非均匀的电荷密度分布, 与高温超导铜氧面中观测到的条纹相定性一致. 我们希望 CPU-GPU 异构并行算法能进一步推进 DMRG 在求解二维与准二维模型、时间演化、量子化学等问题中的应用, 同时引起强关联领域对 GPU 算法的关注和重视.

本文的内容包括以下几部分: 第 2 节回顾了有限 DMRG 算法, 基于四个子块表示的 DMRG 实现, 介绍了该工作采用的基准模型, 并得到了在 CPU 执行时各个部分的计算时间占总时间的比例; 第 3 节针对 DMRG 中最耗时的哈密顿量对角化部分给出了异构并行优化方法; 第 4 节以计算 4 腿 Hubbard 模型基态为例, 对比了异构并行优化方法与单个 CPU 中 MKL 并行的性能; 最后给出本文的总结.

2 有限 DMRG 方法和基准模型

有限尺寸格点模型的 DMRG 计算分为两部分: 首先执行无限 DMRG 方法, 这可为后面的计算提供较好的初始波函数; 然后执行有限 DMRG 扫描, 其中每一步有限 DMRG 扫描会优化系统块的状态, 直到收敛至基态. 在准二维梯子模型的计算中, 有限 DMRG 部分需要多次扫描, 且保留很多的状态才能收敛, 几乎占用整个 DMRG 计算的全部时间, 因此本文主要考虑该部分的并行优化. 有限 DMRG 计算中向左扫描和向右扫描非常相似, 本文以向右扫描为例介绍有限 DMRG 方法. 每一步优化的过程中整个格点系统由四个部分构成 (图 1): S, $s1$, $e1$和 E, 其中 S 和 $s1$构成系统块, E 和 $e1$构成环境块, 系统块和环境块构成超块. 基于四个子块, 波函数有如下形式:

$$|\psi\rangle = \sum_{s,\sigma_{s1},\sigma_{e1},e} \psi_{s,\sigma_{s1},\sigma_{e1},e} |s\rangle |\sigma_{s1}\rangle |\sigma_{e1}\rangle |e\rangle ,$$

其中$|s\rangle$, $|e\rangle$, $|\sigma_{s1}\rangle$和$|\sigma_{e1}\rangle$分别为子块 S, E, $s1$和$e1$上的状态. 同样, 基于四个子块, 一般形式的哈密顿量 H可以表示为

$$H = \sum_{i,j,k,l} W_{ijkl} O_i^S O_k^{s1} O_l^{e1} O_j^E ,$$

其中O^S, O^E, O^{s1}和O^{e1}分别为子块 S, E, $s1$和$e1$上的算符. 以求解系统的基态为例, 给出基于四子块表示的 DMRG 一步优化过程, 如算法 1 所示. 重复执行 DMRG 的一步优化过程, 进行多次实空间扫描, 即可得到收敛的结果. 在算法 1 中, 对角化超块哈密顿量最为耗时, 通常人们采用稀疏矩阵对角化方法 (Lanczos 方法、Davidson 方法等). 此时, 不需要超块哈密顿量的矩阵表示, 仅需要算符在四个子块中的矩阵表示, 并迭代执行 $|\phi\rangle = H|\psi\rangle$, 文献 [4] 中给出了该操作的高效算法, 具体过程如算法 2. 在后面的描述中, 我们称算法 2

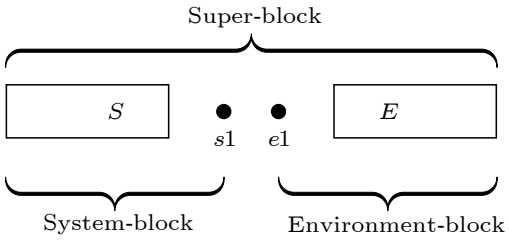

图 1 超块中的四个子块

Fig. 1. 4 Sub-blocks of super-block.

中第 2, 5 和 8 行对应的循环分别为 step1, step2 和 step3. 可以看到该算法 step1 和 step3 中为矩阵乘法, 对应的计算量为$O\left(D^3\right)$, 其中 D 为 DMRG 保留状态数. step2 中仅包含向量操作, 对应的计算量为$O\left(D^2\right)$. 在保留状态较多时, 哈密顿量作用在波函数的计算量主要由 step1 和 step3 决定. 而对角化哈密顿量的总时间线性依赖于对角化方法的迭代次数, 本文实现的有限 DMRG 中通过上一步优化波函数给出一个较好的初始迭代波函数, 可以有效地加快对角化方法的收敛[24].

算法 1 有限 DMRG 向右扫描一步

1: 求解 $H\left|\psi\right\rangle = E\left|\psi\right\rangle$, 得到当前表示下哈密顿量的最小本征值 E 和相应的本征矢 $\left|\psi\right\rangle = \sum_{s,\sigma_{s1},\sigma_{e1},e} \psi_{s\sigma_{s1},\sigma_{e1}e} \cdot \left|s\right\rangle\left|\sigma_{s1}\right\rangle\left|\sigma_{e1}\right\rangle\left|e\right\rangle$;

2: 对角化系统块的约化密度矩阵 $\boldsymbol{\rho}^{S,s1}$, 其中 $\boldsymbol{\rho}^{S,s1}_{s\sigma_{s1},s'\sigma'_{s1}} = \sum_{\sigma_{e1},e} \psi_{s\sigma_{s1},\sigma_{e1}e}\psi^{\dagger}_{\sigma_{e1}e,s'\sigma'_{s1}}$;

3: 保留系统块约化密度矩阵 $\boldsymbol{\rho}^{S,s1}$ 较大本征值对应的态;

4: 投影系统块算符到新的状态空间.

算法 2 $\left|\phi\right\rangle = H\left|\psi\right\rangle$

Input: $\left|\psi\right\rangle$
Output: $\left|\phi\right\rangle$

1: **for** s in system-block **do**
2: **for** σ_{s1}, σ_{e1}, e and j **do**
3: $\left|\tilde{\psi}_j^{[s,\sigma_{s1},\sigma_{e1},e']}\right\rangle = O_j^E\left|\psi^{[s,\sigma_{s1},\sigma_{e1},e]}\right\rangle$
4: **end for**
5: **for** σ_{s1},σ_{e1}, e' and i **do**
6: $\left|\hat{\psi}_i^{[s,\sigma'_{s1},\sigma'_{e1},e']}\right\rangle = \left|\hat{\psi}_i^{[s,\sigma'_{s1},\sigma'_{e1},e']}\right\rangle + \sum_{j,k,l} W_{ijkl}O_k^{s1}O_l^{e1}\left|\tilde{\psi}_j^{[s,\sigma_{s1},\sigma_{e1},e']}\right\rangle$
7: **end for**
8: **for** $\sigma'_{s1},\sigma'_{e1}$, e' and i **do**
9: $\left|\phi^{[s',\sigma'_{s1},\sigma'_{e1},e']}\right\rangle = \left|\phi^{[s',\sigma'_{s1},\sigma'_{e1},e']}\right\rangle + O_i^S\left|\hat{\psi}^{[s,\sigma'_{s1},\sigma'_{e1},e']}\right\rangle$
10: **end for**
11: **end for**

为了获得异构并行优化方法的性能基准, 选取 4 腿梯子上的 Hubbard 模型为例子, 并利用 DMRG 求解其基态. 具体地, 系统哈密顿量为

$$H_{\text{Hubbard}} = -t\sum_{\langle ij\rangle,\sigma}(\hat{c}^{\dagger}_{i,\sigma}\hat{c}_{j,\sigma}+\text{h.c.})+U\sum_i \hat{n}_{i,\uparrow}\hat{n}_{i,\downarrow}, \quad (1)$$

这里 t 为两个格点之间的电子跃迁能, U 为单个格点上的电子在位库仑排斥能, $\hat{c}^{\dagger}_{i,\sigma}(\hat{c}_{i,\sigma})$ 为格点 i 上自旋 σ 的费米子产生 (湮灭) 算符, $\hat{n}_{i,\sigma}$ 为自旋 σ 的粒子数算符, 其中 $\sigma\in\{\uparrow,\downarrow\}$. 在梯子模型中, 格点 i 包含 x,y 两个方向的分量, 以 x 标记梯子长边坐标, y 标记梯子短边坐标. 在后面的测试计算中, 选取 $t=1$ 为能量单位, 沿着长边和短边方向分别使用开边界和周期边界条件, 梯子长度为 16. 对于其他参数, 取相互作用 $U=8$, 总电荷密度 0.875, 即 1/8 空穴掺杂, 这是在其他数值工作中观测到条纹相的典型参数[13,34]. 另外本文的实现利用了该模型的总粒子数和总自旋在 z 方向投影两个好量子数, 每个子块的希尔伯特空间可以被划分为多个子空间, 每个子空间中的状态对应于相同的量子数. 此时模型 (1) 中算符的矩阵表示为分块矩阵, 相应地, 超块的希尔伯特空间可以用其四个子块的好量子数划分为多个子空间.

模型 (1) 哈密顿量的矩阵表示为实对称矩阵, 本文使用 Davidson 方法[35,36] 对角化该哈密顿量. Davidson 方法是一种使用预条件技术的子空间迭代方法, 算法 3 给出了该方法每一步迭代的具体操作. 可以看到, 每一步迭代都需要作用哈密顿量到波函数, 并且包含多个向量操作. 其中向量操作的计算量和存储量均线性依赖于 Davidson 方法子空间中向量的个数和向量的维数. 在性能测试中, 使用 7 次有限 DMRG 扫描收敛到基态, Davidson 方法子空间中向量个数最大为 11.

算法 3 一步 Davidson 迭代

Input: $\hat{\boldsymbol{H}}$ 为哈密顿算符的矩阵表示; $\boldsymbol{\psi}^{(0)}$, $\boldsymbol{\psi}^{(1)},\cdots,\boldsymbol{\psi}^{(i)}$ 和 $\boldsymbol{\phi}^{(0)},\boldsymbol{\phi}^{(1)},\cdots,\boldsymbol{\phi}^{(i-1)}$ 为之前迭代得到的向量

1: $\boldsymbol{\phi}^{(i)} = \hat{\boldsymbol{H}}\boldsymbol{\psi}^{(i)}$
2: $B_{kl} = {\boldsymbol{\phi}^{(k)}}^{\mathrm{T}}\boldsymbol{\psi}^{(l)}$, $0\leqslant k,l\leqslant i$
3: 计算 $\boldsymbol{B}$ 的最小本征值 λ 和相应的本征矢 $\boldsymbol{x}$
4: $\boldsymbol{v} = \sum_{k=0}^{i} x_k\boldsymbol{\psi}^{(k)}$
5: $\boldsymbol{r} = \hat{\boldsymbol{H}}\boldsymbol{v} - \lambda\boldsymbol{v}$
6: 如果收敛, λ 和 $\boldsymbol{v}$ 分别为 $\hat{\boldsymbol{H}}$ 的最小本征值和本征矢, 然后退出
7: 修正 $\boldsymbol{r}$ 得到 $\boldsymbol{u}$
8: $t_k = {\boldsymbol{\psi}^{(k)}}^{\mathrm{T}}\boldsymbol{u}$, $0\leqslant k\leqslant i$
9: $\boldsymbol{u} = \boldsymbol{u} - \sum_{k=0}^{i} t_k\boldsymbol{\psi}^{(k)}$
10: $\boldsymbol{\psi}^{(i+1)} = \dfrac{\boldsymbol{u}}{\sqrt{\boldsymbol{u}^T\boldsymbol{u}}}$

本文所有计算使用的异构并行环境包含 1 个 CPU (Intel Xeon E5-2650 v3, 10 核, 主频 2.3 GHz) 和 Nvidia Tesla K80 卡中的 1 个 GPU (含 12 GB 显存). 程序的实现基于 CUDA 环境, CPU 和 GPU 中的矩阵向量操作分别调用了 Intel MKL 和 CUBLAS 中的子程序. 为了描述矩阵操作的性能, 估计了单位时间内处理器执行的浮点操作的次数, 其中总的浮点操作次数在保留状态数较大时可以由所有矩阵乘法中浮点操作次数之和近似. 为获取异构并行的优化表现, 需要给出 CPU 上的并行性能作为基准. 哈密顿量对角化中包含大量矩阵乘法, 这里首先测试了随机方阵乘法的性能, 结果见图 2(a). 可以看出, 当矩阵尺寸较大 (大于 400) 时, 矩阵乘法的性能较高, 并随着矩阵尺寸增大而增大. 对于 DMRG 算法, 图 2(b) 中的结果表明, 保留状态数越大并行计算性能越高, 并逐渐接近峰值性能. 另外也测试了对角化哈密顿量以及哈密顿量作用于波函数部分占总计算时间的比例, 如图 3 所示. 在我们所关心的保留状态数范围内, 对角化哈密顿量的耗时比例超过总计算时间的 90%, 其中作用哈密顿量到波函数占总时间比例超过 80%. 因此, 我们的工作主要针对该部分进行异构并行优化.

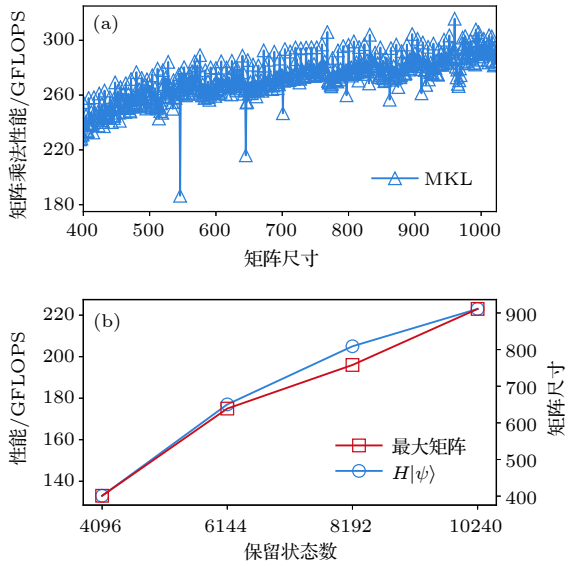

图 2 CPU 中作用哈密顿量在波函数上的性能 (a) 矩阵乘法的浮点性能; (b) 作用哈密顿量于波函数的性能, 及矩阵乘法中的最大矩阵尺寸

Fig. 2. Performance of acting the Hamiltonian on the wave function in CPU: (a) The matrix multiplication performance; (b) the performance of acting the Hamiltonian on the wave function, and the maximum matrix size of the matrix multiplications.

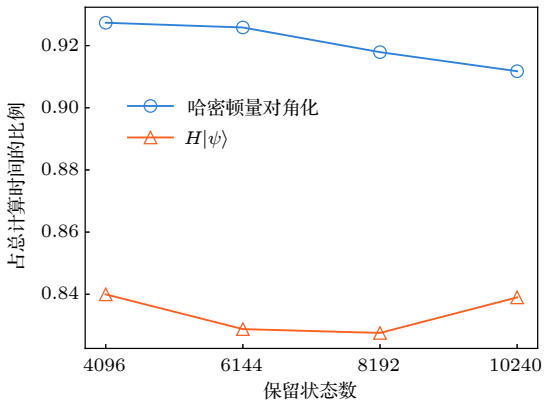

图 3 对角化哈密顿量和作用哈密顿量到波函数操作占总计算时间的比例

Fig. 3. Time ratio of diagonalization of the Hamiltonian and acting the Hamiltonian on the wave function to the total time cost.

3 DMRG 的异构并行实现

主要考虑 DMRG 方法在准二维模型中的应用, 并且针对有限 DMRG 中计算量较大的哈密顿量对角化部分进行并行优化. 对于准二维问题, DMRG 方法达到较高精度通常需要保留较多的状态, 相应地, 对角化哈密顿量时会执行一些大尺寸矩阵操作. 类似于 CPU, 矩阵乘法的性能在 GPU 中随着矩阵尺寸增大而增大; 同时, GPU 的浮点运算能力一般远大于单个 CPU. 因此, 在异构并行优化中, 我们倾向于尽可能在 GPU 中执行大尺寸的矩阵操作. 从存储方面考虑, 为了避免 GPU 显存和内存之间频繁的数据通信, 多次参与 GPU 计算的数据需要存储在 GPU 显存中, 主要包括 Davidson方法中的向量、算符数据和临时数据 (算法 2 中$\left|\tilde{\psi}\right\rangle$和$\left|\hat{\psi}\right\rangle$). 但同时, 保留较多的状态也导致计算中需要的存储容量较大; 考虑到当前 GPU 显存容量较小, 在异构并行方法中需要合理分配各个部分的存储利用.

首先考虑 Davidson 方法的异构并行实现, 如算法 3 所示, Davidson 方法中主要执行各种向量操作, 相对而言运算量较小, 因此应尽可能充分发挥内存和 GPU 显存的存储带宽. 具体地, 我们将所有向量以相同的方式按行划分为两部分, 使得一部分波函数基矢对应的分量存储在 GPU 显存中, 另一部分存储在内存中. 这样任一向量的操作将由 CPU 和 GPU 共同完成, 并且不需要向量数据的通信. 当 GPU 显存足够时 (通常内存容量远大

于 GPU 显存), 内存和 GPU 显存中向量行数的比值为两者存储带宽的比值, 理论上此时可以获得最高的性能. 由于 Davidson 方法的存储量线性依赖于子空间中向量的个数和超块希尔伯特空间的维数, 当向量个数或者 DMRG 保留状态较大时, GPU 显存中的向量数据会受到 GPU 显存容量的限制. 这会导致较多向量操作在 CPU 中执行, 此时 Davidson 方法获得的性能较低.

接下来给出哈密顿量作用于波函数 (即算法 2) 部分的异构并行实现, 首先将其中的操作合理划分到 CPU 和 GPU 中. 将算法 2 中所有操作按照子块 S 中的好量子数分为多个组, 此时各个组中的运算可以独立进行, 仅在求和计算$|\phi\rangle$(即算法 2 中 step3) 时需要通信. 此时波函数$|\psi\rangle$也按 S 中好量子数划分为两个部分, 其中一部分仅在 GPU 中计算 (记为$|\psi_{\rm GPU}\rangle$), 另一部分仅在 CPU 中计算 (记为$|\psi_{\rm CPU}\rangle$).

算法 2 中耗时较多的运算为一系列矩阵乘法, 且 S 中每个好量子数对应分组的计算量在执行运算前可以比较准确地估计. 通常, GPU 的浮点运算能力强于 CPU, 适合处理大尺寸矩阵乘法运算, 因此在这一步尽量将大矩阵运算分配至 GPU, 将相对较小的矩阵运算分配至 CPU 执行. 为了实现这一目标, 在具体操作中, 我们将矩阵乘法平均运算量较大的组分配到 GPU 中执行, 这里平均运算量为组内矩阵乘法总计算量与矩阵乘法个数的比值. 进一步, 可根据 GPU 中计算量占的比例 $P_{\rm GPU}$ 将相互独立的分组计算分配给 GPU, 剩余组的计算分配给 CPU 执行.

为了获得较高的异构并行效率, 在执行 Davidson 对角化方法时动态调整 $P_{\rm GPU}$ 以尽可能实现负载均衡. 在具体操作中, 将每一步迭代优化比例所处的区间记为 $(P^0,\ P^1)$. 设定初始区间 $P^0=0,\ P^1=1$, 然后进入 Davidson 方法迭代过程. 令 GPU 中计算量比例 $P_{\rm GPU}=\left(P^0+P^1\right)/2$, 执行一步 Davidison 迭代, 可以获得此时作用哈密顿量到波函数的 CPU 和 GPU 计算时间, 分别记为 $T_{\rm CPU}$ 和 $T_{\rm GPU}$. 以此为依据更新下一步迭代区间, 使得

$$\begin{cases} P^0=P_{\rm GPU}, & T_{\rm CPU}>T_{\rm GPU};\\ P^1=P_{\rm GPU}, & T_{\rm CPU}<T_{\rm GPU}, \end{cases}$$

并开始下一步 Davidson 迭代. 如此按照类似二分法的收敛思路, 通过少数几步迭代就可以收敛到优化比例, 之后的大部分迭代运算都趋于负载均衡.

本文主要针对保留状态数较大的问题进行优化, 此时涉及到的矩阵较大, GPU 中矩阵运算的并行效率较高. 同时, 考虑到 GPU 显存的限制, 为了减小临时数据的存储量, 根据子块 S 分组后各个组的操作依次执行. 这种情况下, 仅需要分配一段存储空间, 使其同时满足任意一个组中的所有操作即可. 在图 4 中, 给出了对角化哈密顿量部分运算的存储需求, 并给出了与两子块表示所需存储的对比. 可以明显看出其总体的显存需求远远小于两子块表示. 因此, 相比于参考文献 [30] 中的异构并行, 本文方案可以处理需要更大 DMRG 保留状态数的问题.

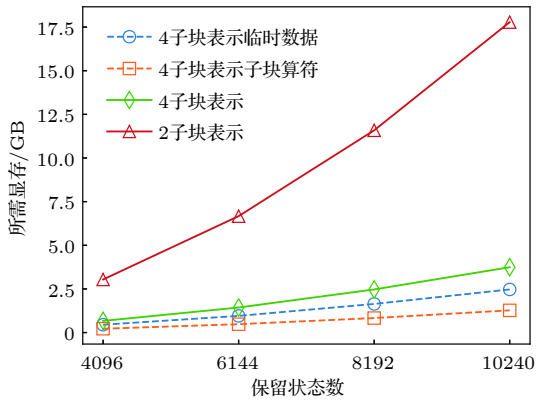

图 4 存储临时数据, 子块算符需要的 GPU 显存

Fig. 4. The GPU memory cost of temporary data and sub-block operators.

分别在 GPU 和 CPU 中执行$|\phi_{\rm GPU}\rangle=H\,|\psi_{\rm GPU}\rangle$和$|\phi_{\rm CPU}\rangle=H\,|\psi_{\rm CPU}\rangle$时, 首先需要将$|\psi_{\rm GPU}\rangle$和$|\psi_{\rm CPU}\rangle$分别拷贝到 GPU 显存和内存中. 由于 GPU 中各个组依次计算, 为了获得较高的性能, 我们在执行其中一个组对应矩阵乘法操作时, 同时进行另一个组的数据通信. 对于 CPU 部分, $|\phi_{\rm CPU}\rangle=H\,|\psi_{\rm CPU}\rangle$包含的的矩阵向量操作基于 IntelMKL 库中的矩阵向量操作子程序并行执行; 而对于 GPU 部分, $|\phi_{\rm GPU}\rangle=H\,|\psi_{\rm GPU}\rangle$中的矩阵向量操作基于 CUBLAS. 在算法 2 中, 由于每个组的操作中 step2 的计算依赖于 step1 的结果, 而 step3 的计算依赖于 step2 的结果, 因此本文依次执行 step1, step2和 step3. 进一步可以看到 step1 中所有矩阵乘法操作相互独立, 因此被分配到多个 CUDA 流 (stream)中, 这样使得多个矩阵乘法可以在 GPU 中同时计算, 较

充分地利用 GPU 的并行计算能力. step2 和 step3 中输出结果为多个操作求和得到, 因此相同好量子数标记的输出结果相关的矩阵向量操作被分配到同一个 CUDA 流执行, 这样使得多个不同好量子数标记的输出结果相关的操作尽可能同时被 GPU 执行. 可以看到算法 3 中$|\phi\rangle$进一步将参与 Davidson 方法中的向量操作, 因此需要被分布式存储在内存和 GPU 显存中, 为了实现异构并行计算$|\phi\rangle = |\phi_{\rm CPU}\rangle + |\phi_{\rm GPU}\rangle$, 需要将$|\phi_{\rm GPU}\rangle$中参与 CPU 计算的数据从 GPU 显存中拷贝到内存中, 并将$|\phi_{\rm CPU}\rangle$中参与 GPU 计算的数据从内存中拷贝到 GPU 显存中. 每个由好量子数标记的输出计算完成则可以开始内存和 GPU 显存之间的数据通信, 因此该部分的实现中数据通信与计算是并行执行的, 有利于实现较高的总性能.

4 数值结果

为了说明本文优化方法的有效性, 我们分别保留 4096, 6144, 8192, 10240 个状态计算 4 腿 Hubbard 梯子的基态能量, 并得到了相应的性能基准. 图 5(a) 给出了 DMRG 优化中各个部分相对于单个 CPU 的加速比 (单个 CPU 计算时间与单个 GPU 计算时间的比值), 可以看到作用哈密顿量在波函数部分获得加速比最大, 其加速比在保留状态数大于 4096 后较为接近 (不低于 3.8). 当保留状态数较大时, 由于 GPU 显存总量的限制 (如图 5(b) 所示), 大部分向量操作由 CPU 完成, 这导致 Davidson 方法的加速比在保留状态数较大时有所下降. 然而, Davidson 方法中向量操作占哈密顿量对角化的时间比例较少, 因此对哈密顿量对角化部分加速比影响较小 (不低于 3.5). 本文哈密顿量对角化之外的其他操作计算时间占总时间比例约 10% (如图 3), 该部分的 GPU 并行优化还没有被考虑, 因此总并行加速比低于哈密顿量对角化部分的加速比, 这里保留最大 10240 个状态获得的加速比为 2.85. 图 5(c) 中给出了异构并行实现中 CPU 和 GPU 分别贡献的浮点性能, 可以看到随着保留状态数的增大, CPU 和 GPU 中执行的大尺寸矩阵增多, 两者贡献的浮点性能随之增大. 虽然本文数值计算保留状态数较大, 但是其中矩阵尺寸为两子块表示时的 $1/d$ (Hubbard 模型中 $d = 4$), 目前获得的性能仍明显小于 GPU 可以达到的峰值性能 (1200 GFLOPS). 对 16×4 的梯子, 根据不同保留状态数 (4096, 6144, 8192, 10240) 得到的基态能量外推得到模型 (1) 在 $U = 8.0$ 时的格点平均能量 $E_g = -0.75114(2)$, 与文献 [34] 结果一致, 见图 6. 进一步给出基态的电荷密度分布, 如图 7 所示, 可以观察到明显的电荷密度条纹, 这是铜氧化物高温超导体中经常被观测到的现象之一[13,14,37,38].

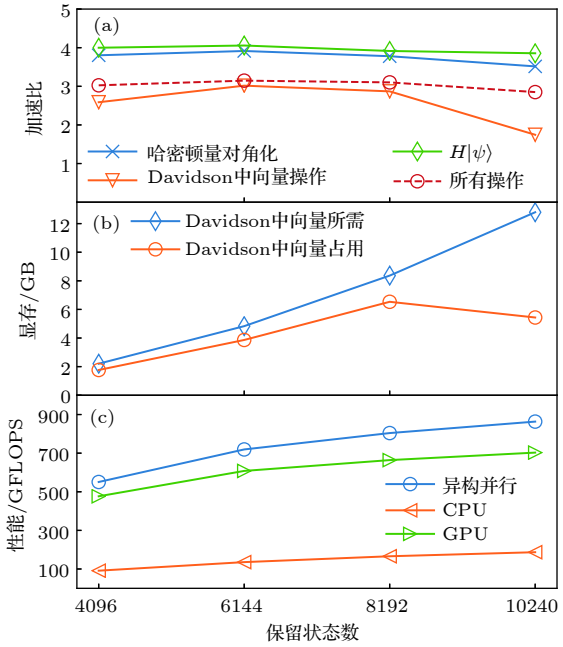

图 5 异构并行的性能 (a) 加速比; (b) Davidson 方法中的向量占用 GPU 显存; (c) 作用哈密顿量到波函数部分的性能

Fig. 5. Performance of hybrid parallel strategy: (a) The speedup; (b) the GPU memory cost of vectors in Davidson; (c) the performance of $H|\psi\rangle$.

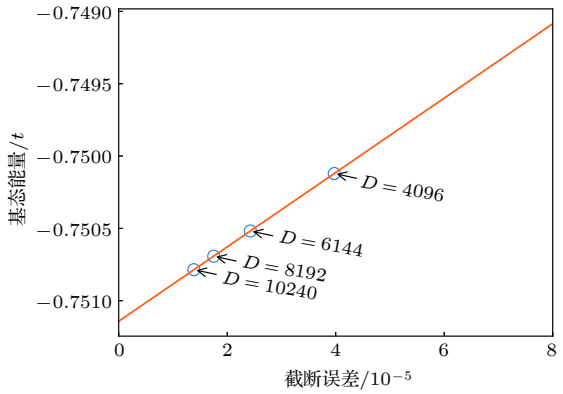

图 6 基态能量关于截断误差的函数 (直线表示对基态能量的线性外推, 直至截断误差为 0)

Fig. 6. Groundstate energy as a function of truncation error. The straight line gives a linear extrapolation of the ground energy until 0 truncation-error.

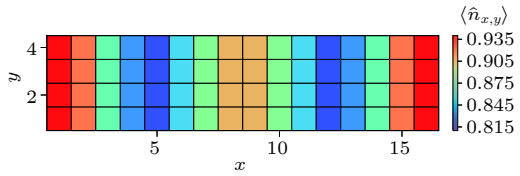

图 7 对于 16 × 4 Hubbard 模型, $U = 8.0$ 时的基态电荷密度分布 (可以观察到明显的电荷密度条纹)

Fig. 7. Ground state density profile for the 16 × 4 Hubbard ladder with $U = 8.0$. Charge density stripes can be clearly observed.

5 结 论

本文主要考虑 DMRG 方法在准二维格点模型中的应用, 针对其中最耗时的哈密顿量对角化部分实现了 CPU-GPU 异构并行优化, 并且给出了负载平衡方法. 为了减小准二维格点模型计算中 GPU 显存的限制, 本文的实现中哈密顿量与波函数基于四子块表示, 其对角化时需要的 GPU 显存占用远小于两子块表示, 使得本文的异构并行方法可以应用于更多模型、更多问题的研究. 将该方法应用到 4 腿 Hubbard 梯子模型的求解中, 得到了不同保留状态数时 DMRG 中各个部分的加速比. 数值结果表明, 本文的异构并行方法适用于保留状态数较大的准二维模型计算, 并且总性能随着保留状态数增大而增大. 目前, 强关联物理问题很大程度上依赖于多体数值计算, 一些复杂问题通常进一步受制于计算方法的计算量与计算时间. 在多体算法本身出现革命性发展之前, 合理利用计算机技术的发展提升算法的效率能为研究强关联系统提供很大的帮助. 我们希望该并行方法可以在更多的复杂格点模型、更多问题中得到应用, 并能够进一步引起强关联领域对于以 GPU 为代表的新技术的关注和重视.

参考文献

[1] White S R 1992 *Phys. Rev. Lett.* **69** 2863
[2] White S R 1993 *Phys. Rev. B* **48** 10345
[3] Schollwöck U 2005 *Rev. Mod. Phys.* **77** 259
[4] Schollwöck U 2011 *Annals of Physics* **326** 96
[5] Xiang T 1996 *Phys. Rev. B* **53** R10445
[6] White S R, Martin R L 1999 *J. Chem. Phys.* **110** 4127
[7] Luo H G, Qin M P, Xiang T 2010 *Phys. Rev. B* **81** 235129
[8] Yang J, Hu W, Usvyat D, Matthews D, Schütz M, Chan G K L 2014 *Science* **345** 640
[9] Cazalilla M A, Marston J B 2002 *Phys. Rev. Lett.* **88** 256403
[10] Luo H G, Xiang T, Wang X Q 2003 *Phys. Rev. Lett.* **91** 049701
[11] White S R, Feiguin A E 2004 *Phys. Rev. Lett.* **93** 076401
[12] Cheng C, Mondaini R, Rigol M 2018 *Phys. Rev. B* **98** 121112
[13] Zheng B X, Chung C M, Corboz P, Ehlers G, Qin M P, Noack R M, Shi H, White S R, Zhang S, Chan G K L 2017 *Science* **358** 1155
[14] Huang E W, Mendl C B, Liu S, Johnston S, Jiang H C, Moritz B, Devereaux T P 2017 *Science* **358** 1161
[15] Dagotto E 1994 *Rev. Mod. Phys.* **66** 763
[16] Keimer B, Kivelson S A, Norman M R, Uchida S, Zaanen J 2015 *Nature* **518** 179
[17] Fradkin E, Kivelson S A, Tranquada J M 2015 *Rev. Mod. Phys.* **87** 457
[18] Yan S, Huse D A, White S R 2011 *Science* **332** 1173
[19] Savary L, Balents L 2017 *Rep. Prog. Phys.* **80** 016502
[20] Alvarez G 2012 *Comput. Phys. Commun.* **183** 2226
[21] Tzeng Y C 2012 *Phys. Rev. B* **86** 024403
[22] Legeza O, Röder J, Hess B A 2003 *Phys. Rev. B* **67** 125114
[23] Legeza O, Sólyom J 2003 *Phys. Rev. B* **68** 195116
[24] White S R 1996 *Phys. Rev. Lett.* **77** 3633
[25] Hubig C, McCulloch I P, Schollwöck U, Wolf F A 2015 *Phys. Rev. B* **91** 155115
[26] White S R 2005 *Phys. Rev. B* **72** 180403
[27] Stoudenmire E M, White S R 2013 *Phys. Rev. B* **87** 155137
[28] Hager G, Jeckelmann E, Fehske H, Wellein G 2004 *J. Comput. Phys.* **194** 795
[29] Chan G K L 2004 *J. Chem. Phys.* **120** 3172
[30] Nemes C, Barcza G, Nagy Z, Örs Legeza, Szolgay P 2014 *Comput. Phys. Commun.* **185** 1570
[31] Siro T, Harju A 2012 *Comput. Phys. Commun.* **183** 1884
[32] Lutsyshyn Y 2015 *Comput. Phys. Commun.* **187** 162
[33] Yu J, Hsiao H C, Kao Y J 2011 *Comput. Fluids* **45** 55
[34] Ehlers G, White S R, Noack R M 2017 *Phys. Rev. B* **95** 125125
[35] Davidson E R 1975 *J. Comput. Phys.* **17** 87
[36] Sadkane M, Sidje R B 1999 *Numer. Algorithms* **20** 217
[37] Tranquada J M, Sternlieb B J, Axe J D, Nakamura Y, Uchida S 1995 *Nature* **375** 561
[38] Comin R, Damascelli A 2016 *Annu. Rev. Condens. Matter Phys.* **7** 369

Hybrid parallel optimization of density matrix renormalization group method*

Chen Fu-Zhou [1)] Cheng Chen [1)2)] Luo Hong-Gang [1)2)†]

1) (*School of Physical Science and Technology, Lanzhou University, Lanzhou* 730000, *China*)

2) (*Beijing Computational Science Research Center, Beijing* 100084, *China*)

(Received 22 April 2019; revised manuscript received 16 May 2019)

Abstract

Density matrix renormalization group (DMRG), as a numerical method of solving the ground state of one-dimensional strongly-correlated lattice model with very high accuracy, requires expensive computational and memory cost when applied to two- and quasi-two-dimensional problems. The number of DMRG kept states is generally very large to achieve a reliable accuracy for these applications, which results in numerous matrix and vector operations and unbearably consuming time in the absence of the proper parallelization. However, due to its sequential nature, the parallelization of DMRG algorithm is usually not straightforward. In this work, we propose a new hybrid parallelization strategy for the DMRG method. It takes advantage of the computing capability of both central processing unit (CPU) and graphics processing unit (GPU) of the computer. In order to achieve as many as DMRG kept states within a limited GPU memory, we adopt the four-block formulation of the Hamiltonian rather than the two-block formulation. The later consumes much more memories, which has been used in another pioneer work on the hybrid parallelization of the DMRG algorithm, and only a small number of DMRG kept states are available. Our parallel strategy focuses on the diagonalization of the Hamiltonian, which is the most time-consuming part of the whole DMRG procedure. A hybrid parallelization strategy of diagonalization method is implemented, in which the required data for diagonalization are distributed on both the host and GPU memory, and the data exchange between them is negligible in our data partitioning scheme. The matrix operations are also shared on both CPU and GPU when the Hamiltonian acts on a wave function, while the distribution of these operations is determined by a load balancing strategy. Taking fermionic Hubbard model for example, we examine the running performance of the hybrid parallelization strategy with different DMRG kept states and provide corresponding performance benchmark. On a 4-leg ladder, we employ the conserved quantities with $U(1)$ symmetry of the model and a good-quantum number based task scheduling to further reduce the GPU memory cost. We manage to obtain a moderate speedup of the hybrid parallelization for a wide range of DMRG kept states. In our example, the ground state energy with high accuracy is obtained by the extrapolation of the results, with different numbers of states kept, and we show charge stripes which are usually experimentally observed in high-temperature superconductors. In this case, we keep 10^4 DMRG states and the GPU memory cost is less than 12 Gigabytes.

Keywords: density matrix renormalization group, strongly correlated lattice model, hybrid parallelization

PACS: 02.70.–c, 71.10.Fd, 71.27.+a, 05.10.Cc **DOI:** 10.7498/aps.68.20190586

* Project supported by the National Natural Science Foundation of China (Grant Nos. 11674139, 11834005) and the Program for Changjiang Scholars and Innovative Research Team in University, China (Grant No. IRT-16R35).

† Corresponding author. E-mail: luohg@lzu.edu.cn