中国科学院战略性先导科技专项(C)

国产安全可控先进计算系统研制
PyQCU 软件测试报告

测试单位：中国科学近代物理研究所

测试日期：2025 年 8 月 29 日

PyQCU 性能测试

1.1 软件 PyQCU 简介

格点量子色动力学（Lattice Quantum Chromodynamics，简称格点 QCD）作为目前较为成熟的研究量子色动力学非微扰问题的基准性数值方法，在强相互作用物理研究中具有不可忽略的作用。本软件聚焦于该方法的核心应用场景——部分子分布函数的数值研究，并针对其中的关键计算瓶颈即传播子的高效求解展开攻关。从数学形式上看，该问题可归结为具有特殊结构的大规模稀疏复系数矩阵线性方程组的求解，其维度规模在传统通用计算框架下已超出可处理范围，亟需结合其数学特性设计专用的高性能并行求解算法，这正是本软件研究的核心目标与技术突破点。目前研究工作已取得阶段性成果：首先完成了格点 QCD 中两类典型作用量对应的稀疏矩阵构建器开发——基于 Wilson 作用量的 Dslash 算符（格点 QCD 中描述夸克场传播的特殊微分算符）及其经 Clover 作用量优化的改进型版本；其次针对上述矩阵特性，成功实现了共轭梯度法（Conjugate Gradient Method, CG）与稳定双共轭梯度法（Biconjugate Gradient Stabilized Method, BICGSTAB）两类核心线性求解算法；更进一步，实现了进阶线性求解算法——多重网格算法（MultiGrid Method，MG）的前期验证版本，不久将投入实用；最后构建了面向实际应用的 Python 计算接口，为后续大规模物理问题求解奠定了完整的软件基础。（访问地址：https://gitee.com/zhangxin8069/PyQCU）

1.2 测试环境简介

419 项目测试平台软硬件环境信息：

硬件配置：

处理器：1* Hygon C86 7185 32-core Processor

内存：8*16GB DDR4 2666MHz；

高速网络：200Gb 高速网络；

DCU 加速卡：4*Pre-Wukong；

共享存储：ParaStor300 分布式文件系统；

软件环境：

OS: Red Hat Enterprise Linux Server release 7.4 (Maipo)

Kernel 版本：3.10.0-693.el7.x86_64

编译器: GCC 7.2.1 20170829 (Red Hat 7.2.1-1)

UCX: 1.6.0

MPI: HPC-X 2.4.0

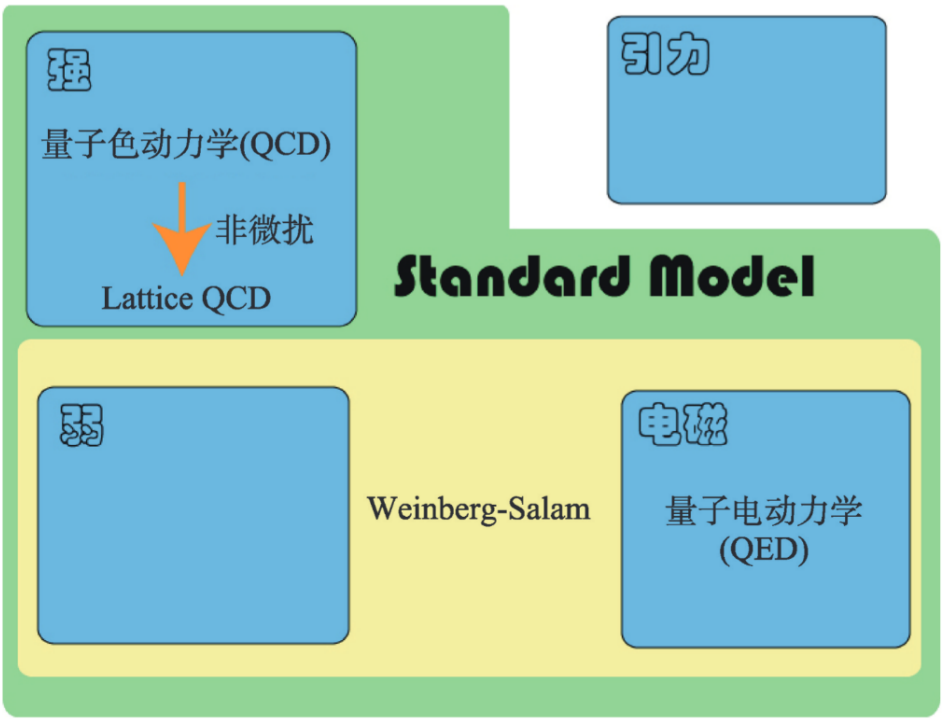
监控与作业调度系统：Gridview4.6

1.3 软件算法介绍

1.3.1 格点量子色动力学

在QCD中拉氏量为：

$$\mathcal{L}_{QCD} = \overline{\psi_i}\left(i\gamma^{\mu}\left(D_{\mu}\right)_{ij} - m\delta_{ij}\right)\psi_j - \frac{1}{4}F^{a}_{\mu\nu}F^{\mu\nu}_{a} \quad (1)$$



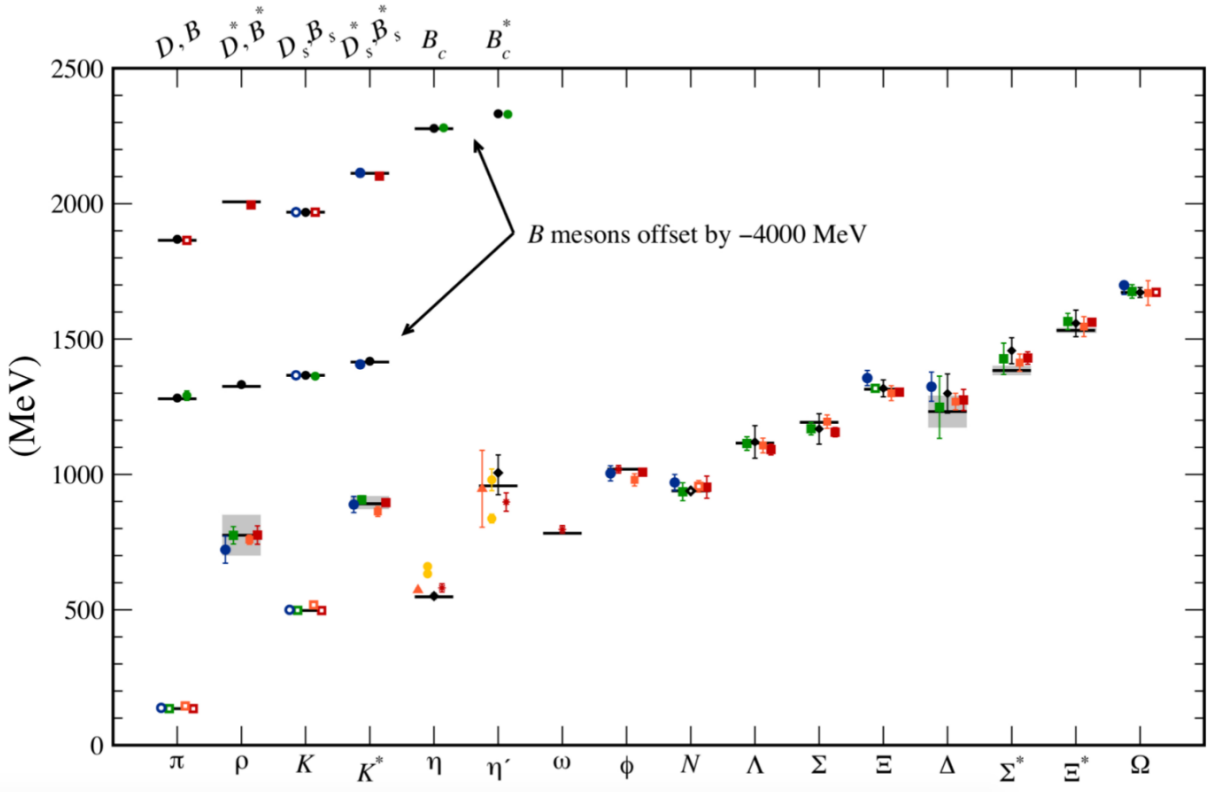
图表 1 格点量子色动力学在标准模型中的相对位置的示意图[1]

其中ψ_i为夸克场，γ^{μ}为狄拉克矩阵，D_{μ}为协变导数，$F^{a}_{\mu\nu}$为规范胶子场张量。

其中QCD是一种强相互作用的规范场论，而LQCD的计算方法就是在不破坏规范不变性的情况下，将连续的时空离散化为欧几里得时空下的四维超立方格子

包含特定的信息，包括其位置，自旋和颜色等。将其位置矢量定义为$x_\mu = n_\mu a$，其中n_μ是两格子之间相邻的格子数，a被称为格距[1]。

格点的计算方法在非微扰QCD中得到了广泛应用，但其适用范围不仅仅局限于非微扰QCD。其核心思想是将连续时空离散化为分立时空，这才是格点方法最重要的部分。此外，它具有天然的优势：在构建量子场论时，由于非平凡的量子场存在紫外发散的困难，导致量子场论在构建过程中缺乏严格的数学定义，因此必须要有一个内禀的紫外截断来解决紫外发散的问题，而在LQCD中因为格距a的存在所以天然的存在一个紫外截断$\Lambda = \pi/a$，而又因为整个空间的体积被限制在一个特定大小的四维格子内部，所以也就存在一个红外的截断$\Lambda' = 2\pi/L$，L为空间的长度[1]。



图表 2 格点 QCD 计算得到的强子谱[2]

现在，通过格点 QCD 进行的强子谱计算已经在国际上被广泛认可，如上图，可以看到各大格点合作组确定的强子谱基本都与实验相符[2]。

1.3.2 格点中的部分子分布函数

根据费曼的部分子模型，接近光速运动的核子可以看作是由高速运动的自由点粒子构成，但是在这些高速运动的粒子中，夸克所携带的质子动量的份额x是

不确定的，这是因为夸克之间会有相互作用导致动量发生变化。但夸克的动量具有一定的概率分布，因此为了定量描述夸克的动量分布，引入部分子分布函数（PDFs）$\delta f(\mathrm{x},\mu)$，x为夸克携带的动量分数，$\delta f(\mathrm{x},\mu)\mathrm{dx}$表示为动量在$\mathrm{x}\to\mathrm{x}+\mathrm{dx}$之间的夸克数目。

但是当夸克的动量非常大时，其会遇到紫外发散的困难，所以可以在天然具有紫外截断的 LQCD 中计算有限大动量的准 PDFs（quasi PDFs），然后利用大动量外推计算无穷大动量时的 PDFs[3]。（这里仅考虑价夸克的部分子分布函数，而不考虑胶子的部分子分布函数）

准 PDFs 被定义为：

$$\delta\tilde{f}(x,P_z,1/a)=\int\frac{dz}{4\pi}e^{ixzP_z}\langle P|\bar{\psi}(z)\gamma^t U(z,0)\psi(0)\lfloor P\rangle \quad (2)$$

P_z是沿着Z方向的动量，P为夸克携带的动量，$\psi(0)$表示为位置在 0 点的夸克场，$U(z,0)$为位置 0 点与 z 点的规范链接。

以π^+介子为例，为了提取 PDFs 的矩阵元，将其两点和三点关联函数进行分解可以得到：

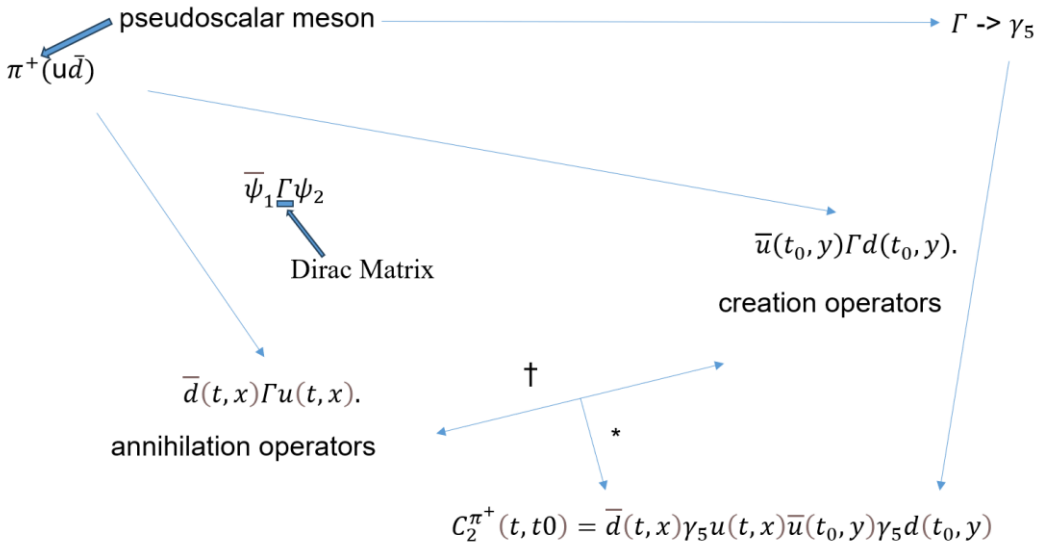
$$C_2^{\pi^+}(P_z,t,t_{sep})=|\mathcal{A}_0|^2\frac{e^{-E_\pi\vec{P}\frac{Lt_t}{2}}}{E_\pi\vec{P}}cosh\left[-E_\pi\vec{P}\left(t-\frac{L_t}{2}\right)\right]+\cdots$$

$$C_3^{\pi^+}(P_z,t,t_{sep})=|\mathcal{A}_0|^2\frac{e^{-E_\pi\vec{P}\frac{Lt_t}{2}}}{E_\pi\vec{P}}\langle 0|\mathcal{O}_\Gamma|0\rangle cosh\left[-E_\pi\vec{P}\left(t-\frac{L_t}{2}\right)\right]+\cdots \quad (3)$$

其中$\left\langle 0\middle|\mathcal{O}_\Gamma\middle|0\right\rangle$是准 PDFs 的矩阵元，$\mathrm{t_{sep}}$表示源和汇的时间间隔，t表示两点关联函数所处的时间片和算符$\mathcal{O}_\Gamma$插入时的时间，$\mathrm{E_0}$表示基态的能量，省略号表示在强子中衰减速度相较于基态快很多的高激发态的贡献[4]。

1.3.3 格点中的传播子

继续以π⁺介子的两点关联函数为例，其还可以表示为：

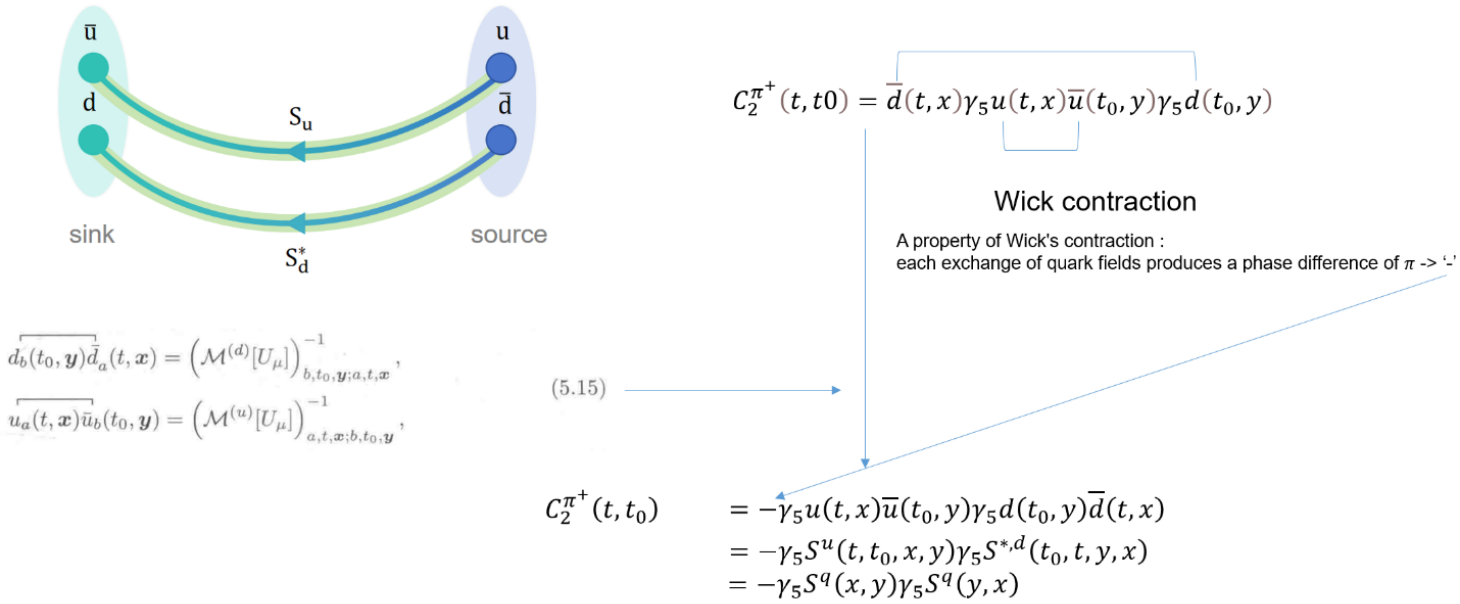


$$C_2^{\pi^+}(t,t0) = \overline{d}(t,x)\gamma_5 u(t,x)\overline{u}(t_0,y)\gamma_5 d(t_0,y) \quad (4)$$

图表 3 π⁺介子的两点关联函数的推导示意图

对其进行维克收缩后表示为：

$$C_2^{\pi^+}(t,t_0) = -\gamma_5 S^q(x,y)\gamma_5 S^q(y,x) \quad (5)$$



图表 4 对π⁺介子的两点关联函数维克收缩示意图

其中S被称为传播子，其具体表现为费米子矩阵的逆$\mathcal{M}^{-1}$[1]。

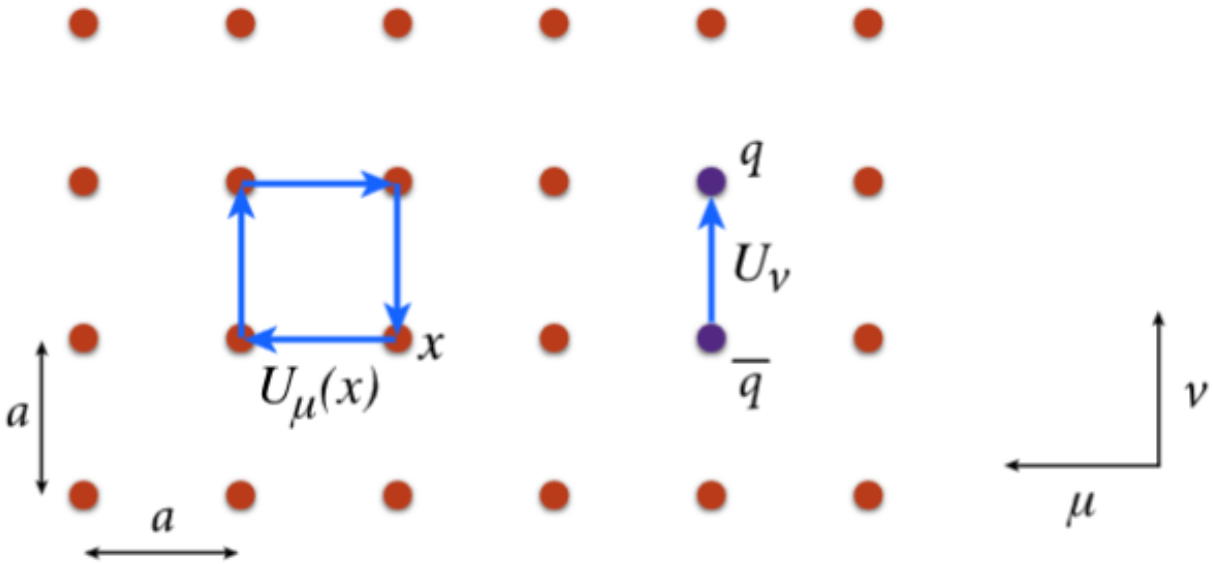
结合（3）式与（5）式，我们可以通过求解传播子S来给出两点关联函数$C_2^{\pi^+}$，再结合其与准 PDFs 的矩阵元的关系来研究部分子分布函数。

1.3.4 格点中的 Dslash

上述的费米子矩阵$\mathcal{M}$或者说其与向量进行矩阵乘的操作，我们一般称为 Dslash。依据选用的格点上的作用量的不同，我们常用的 Dslash 有 Wilson Dslash 与 Clover Dslash。

1.3.4.1 Wilson Dslash

当使用 Wilson 作用量时，给出 Wilson Dslash[2]：



$$M^{W}_{x,y}[U]a = \delta_{xy} - \kappa\sum_{\mu}\left[(r-\gamma_\mu)U_{x,\mu}\delta_{x,y-\mu} + (r+\gamma_\mu)U^{\dagger}_{x-\mu,\mu}\delta_{x,y+\mu}\right] \quad (6)$$

图表 5 二维格点上的物质场和规范链接[2]

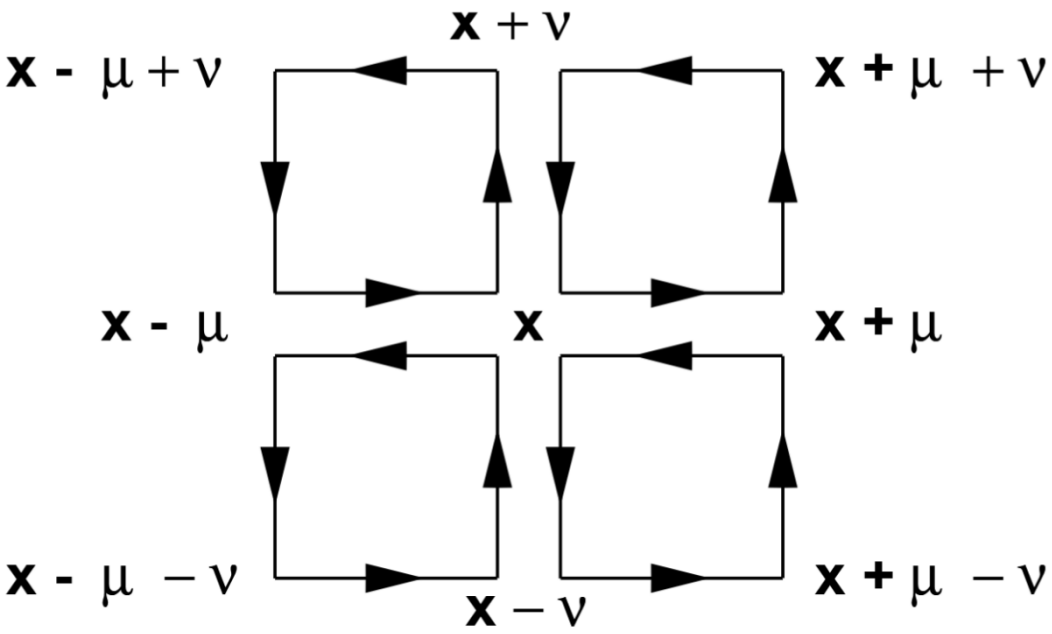
其中κ与a为实际使用时给出的系数，在此不做讨论，U_μ为格点上的规范链接，后续将作为输入之一。

1.3.4.2 Clover Dslash

当使用 Clover 作用量时，给出 Clover Dslash[2]：

$M_{x,y}^{W}[U]a =$

$$\delta_{xy} - \kappa\sum_{\mu}\left[(r-\gamma_{\mu})U_{x,\mu}\delta_{x,y-\mu} + (r+\gamma_{\mu})U_{x-\mu,\mu}^{\dagger}\delta_{x,y+\mu}\right] - \frac{iac_{SW}\kappa r}{4}\sigma_{\mu\nu}F_{\mu\nu}\delta_{x,y} \quad (7)$$



图表 6 Clover(四叶草)项示意图[2]

其中，$F_{\mu,\nu} = \frac{1}{4}\sum_{p}\frac{1}{2}\left[U_p(x) - U_p^{\dagger}(x)\right]$，$U_p$展开为：

$U_1 = u(x,\mu)u(x+\mu,\nu)u^{\dagger}(x+\nu,\mu)u^{\dagger}(x,\nu)$

$U_2 = u(x,\nu)u^{\dagger}(x-\mu+\nu,\mu)u^{\dagger}(x-\mu,\nu)u(x-\mu,\mu)$

$U_3 = u^{\dagger}(x-\mu,\mu)u^{\dagger}(x-\mu-\nu,\nu)u(x-\mu-\nu,\mu)u(x-\nu,\nu)$

$U_4 = u^{\dagger}(x-\nu,\nu)u(x-\nu,\mu)u(x-\nu+\mu,\nu)u^{\dagger}(x,\mu)$

且$\sigma_{\mu,\nu} = \gamma_{\mu}\gamma_{\nu} - \gamma_{\nu}\gamma_{\mu} = 2\gamma_{\mu}\gamma_{\nu}$，则，$\sigma_{\mu\nu}F_{\mu\nu}$可展开为：

$$\begin{aligned}\sigma_{\mu\nu}F_{\mu\nu} &= 2\gamma_{\mu}\gamma_{\nu}\frac{1}{4}\sum_{p}\frac{1}{2}\left[U_p(x) - U_p^{\dagger}(x)\right] \\ &= \frac{1}{4}\gamma_{\mu}\gamma_{\nu}[u(x,\mu)u(x+\mu,\nu)u^{\dagger}(x+\nu,\mu)u^{\dagger}(x,\nu) \\ &+ u(x,\nu)u^{\dagger}(x-\mu+\nu,\mu)u^{\dagger}(x-\mu,\nu)u(x-\mu,\mu) \\ &+ u^{\dagger}(x-\mu,\mu)u^{\dagger}(x-\mu-\nu,\nu)u(x-\mu-\nu,\mu)u(x-\nu,\nu) \\ &+ u^{\dagger}(x-\nu,\nu)u(x-\nu,\mu)u(x-\nu+\mu,\nu)u^{\dagger}(x,\mu) \\ &- BEFORE^{\dagger}] \quad (8)\end{aligned}$$

1.3.5 求解算法

综合 1.3.3 与 1.3.4 所述，传播子S为费米子矩阵的逆$\mathcal{M}^{-1}$，即 Dslash 的逆。在实际应用中我们会给定一个向量作为输入，即求解传播子的问题可以简化为求解$Ax = b$。求解此类问题有 CG 算法与 BISTABCG 算法两种较为成熟的简单方法（实际运用时使用进阶的 MultiGrid 算法）。

1.3.5.1 CG 算法

CG算法一般作为最为基础的求解器，常用作进阶算法（例如MultiGrid算法）的内嵌求解器，

算法 1 Conjugate gradient

1: **procedure** CG(A, b) ▷ Solve $Ax = b$ by CG
2: $x \leftarrow 0$
3: $r \leftarrow b - Ax$
4: If converge return x
5: $p \leftarrow r$
6: **while** not reach max iteration **do**
7: $\alpha \leftarrow \frac{\langle r,r\rangle}{\langle p,Ap\rangle}$
8: $x \leftarrow x + \alpha p$
9: $r' \leftarrow r - \alpha Ap$
10: If converge return x
11: $\beta \leftarrow \frac{\langle r',r'\rangle}{\langle r,r\rangle}$
12: $p \leftarrow r' + \beta p$
13: $r \leftarrow r'$
14: **end while**
15: **end procedure**

图表 7 CG 算法的伪代码示意图[2]

但其要求A为正定矩阵，我们常常通过求解$A^{\dagger}Ax = A^{\dagger}b$来满足这个要求。

1.3.5.2 BISTABCG 算法

BISTABCG算法不要求A为正定矩阵，也可以作为进阶算法的内嵌求解器，且相较于CG算法更加稳定可靠。当实现进阶算法复杂度过大时，通过实现BISTABCG算法来解决问题具有相当的“性价比”。

算法 3 Biconjugate gradient stablized

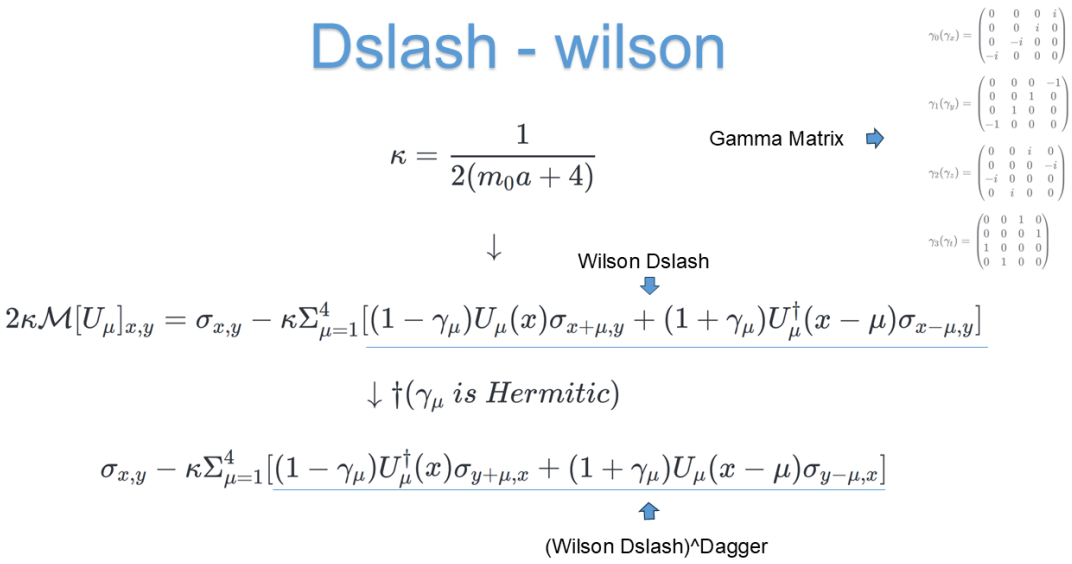
procedure BiCGSTAB(A, b) ▷ Solve $Ax = b$ by BiCGSTAB
  $x \leftarrow 0$
  $r \leftarrow r_0 \leftarrow b - Ax$
  If converge return x
  $\leftarrow r$
  while not reach max iteration **do**
    $\alpha \leftarrow \frac{\langle r_0, r\rangle}{\langle r_0, Ap\rangle}$
    $x \leftarrow x + \alpha p$
    $r' \leftarrow r - \alpha Ap$
    If converge return x
    $t \leftarrow Ar$
    $\omega \leftarrow \frac{\langle t, r\rangle}{\langle t, t\rangle}$
    $x \leftarrow x + \omega r'$
    $r' \leftarrow r' - \omega Ar'$
    If converge return x
    $\beta \leftarrow \frac{\langle r', r'\rangle}{\langle r, r\rangle}$
    $p \leftarrow r' + (\alpha\beta/\omega)p - \alpha\beta Ap$
    $r \leftarrow r'$
  end while
end procedure

图表8 BISTABCG算法的伪代码示意图[2]

1.3.6 格点算符的实现

1.3.6.1 Wilson Dslash 的实现

实现理论参考 1.3.4.1 以及下图：

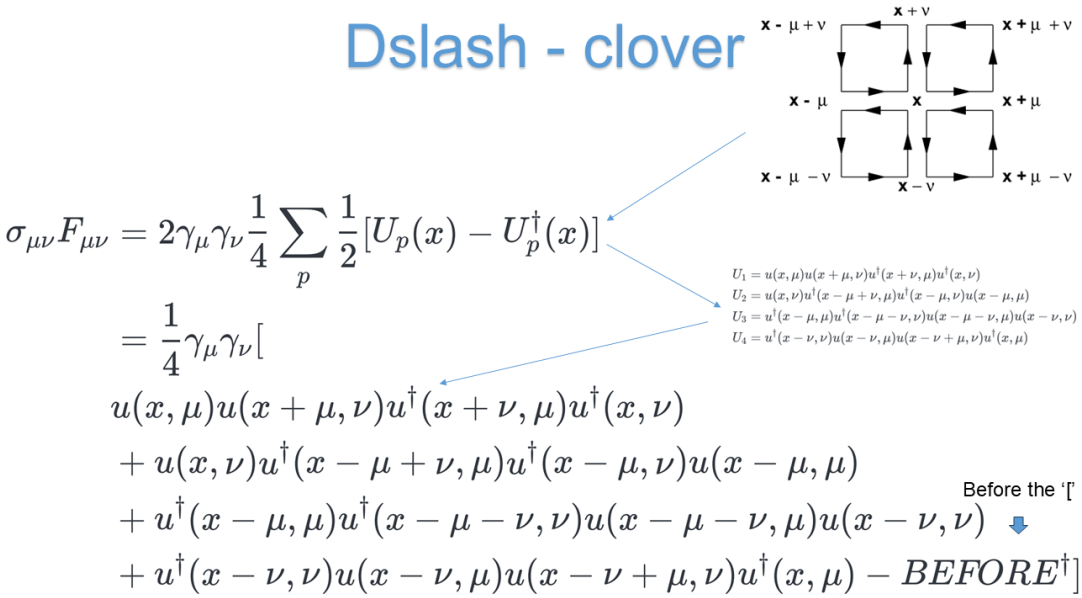


图表 9 WilsonDslash 实现示意图

核心代码参考“*wilson_dslash*.cu”。

1.3.6.2 Clover Dslash 的实现

实现理论参考 1.3.4.2 以及下图：



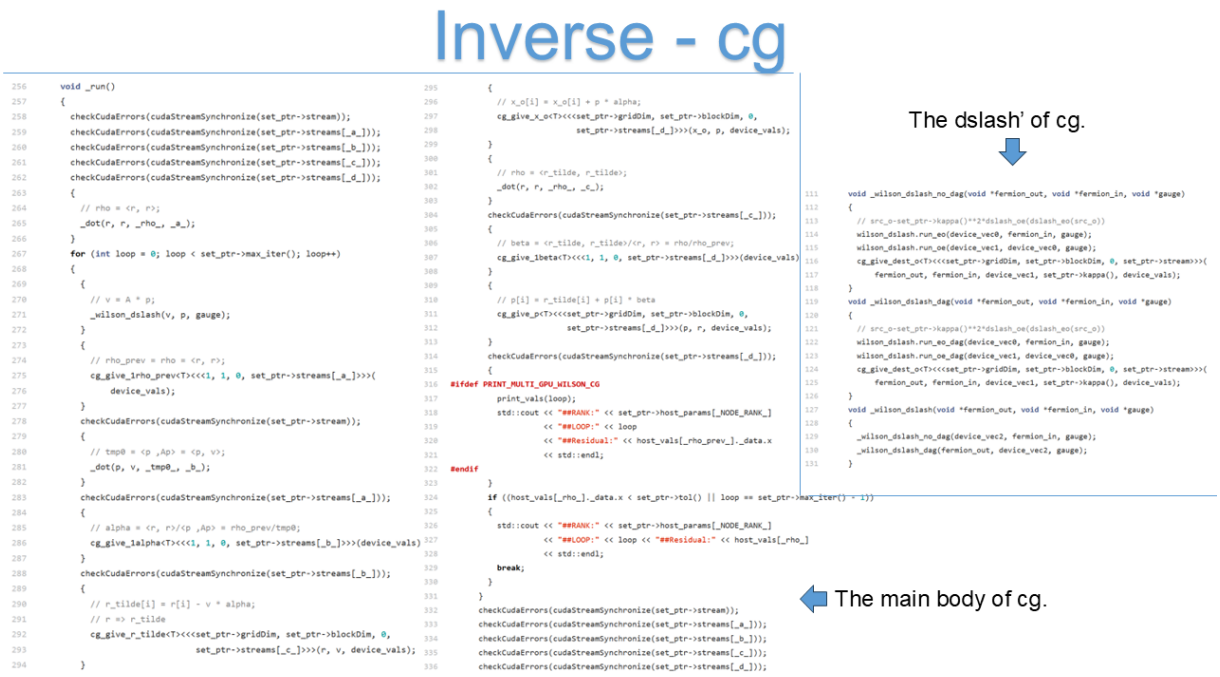
图表 10 CloverDslash 实现示意图

核心代码参考“*clover_dslash*.cu”。

1.3.7 求解算法的实现

1.3.7.1 CG 算法的实现

实现理论参考 1.3.5.1 以及下图：

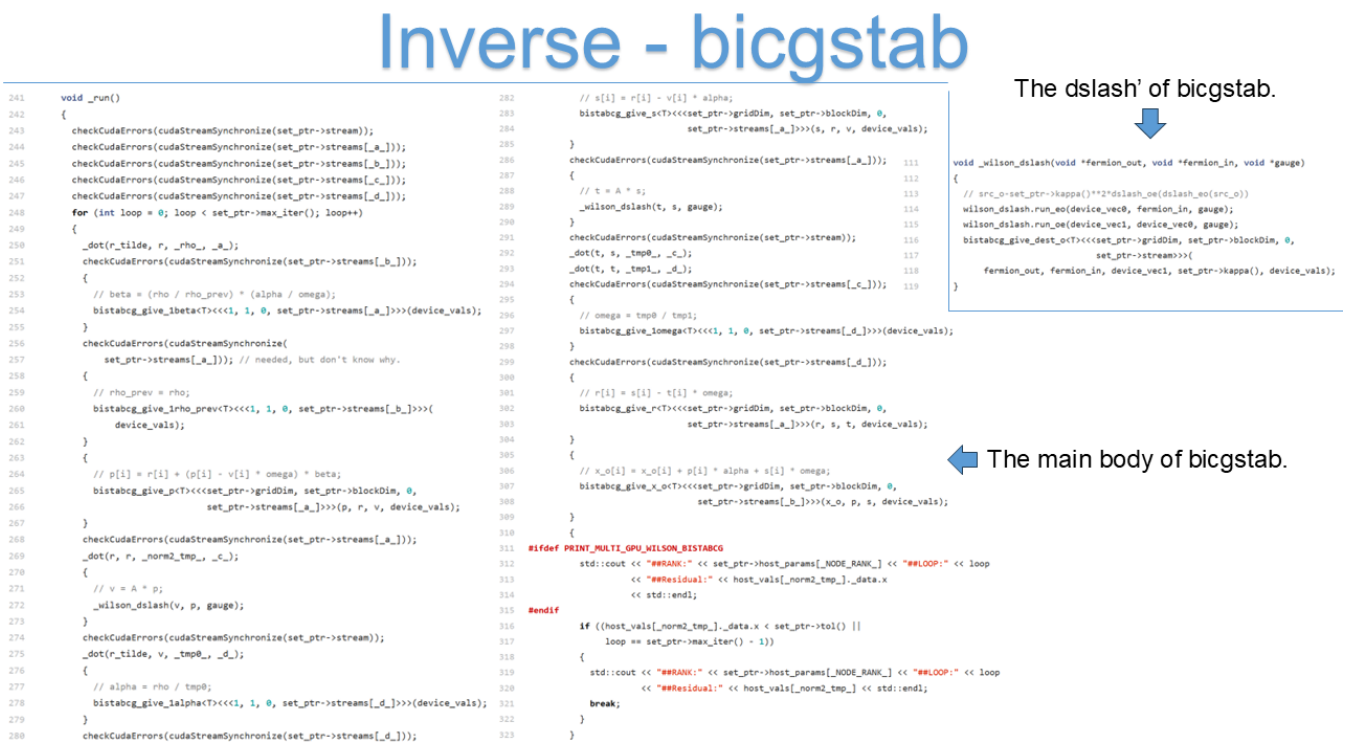


图表 11 CG 算法实现示意图

核心代码参考“*cg.h”。

1.3.7.2 BISTABCG 算法的实现

实现理论参考 1.3.5.2 以及下图：



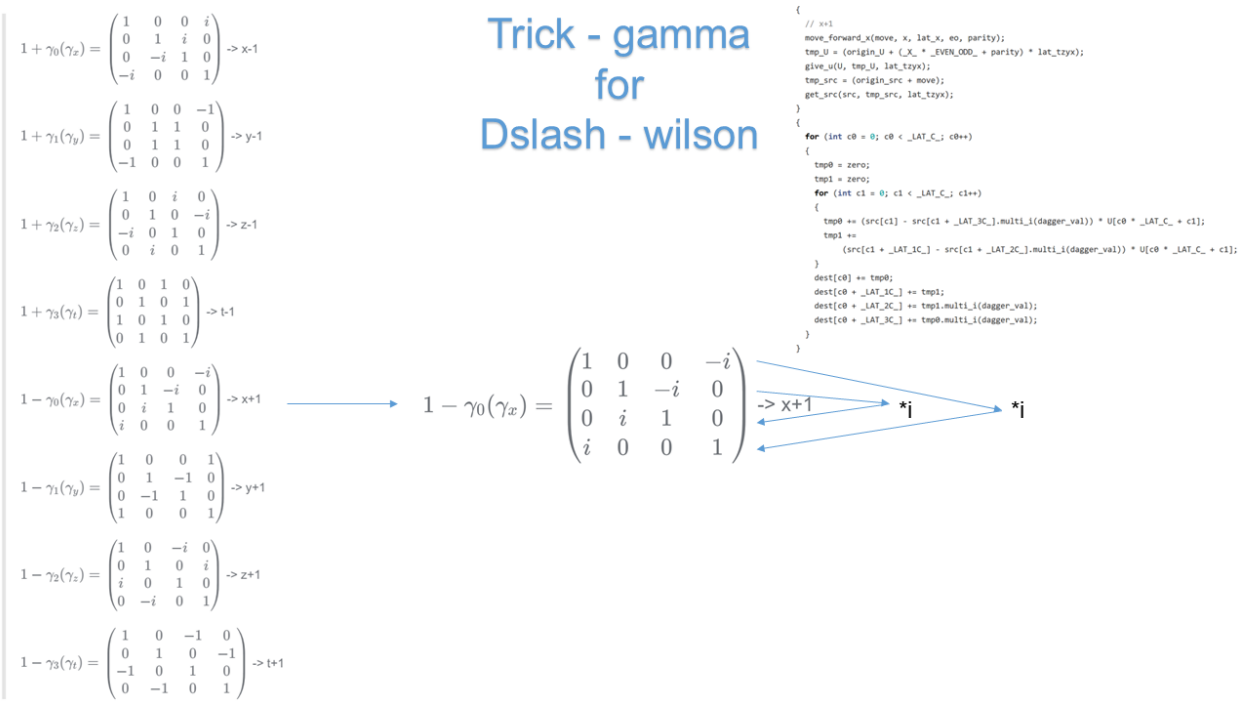
图表 12 BISTABCG 算法实现示意图

核心代码参考“*bistabcg.h”。

1.3.8 性能优化

1.3.8.1 优化 1

在进行 Dslash 操作时，其自旋维度体现为γ矩阵，其形式如下图左半部分所示：

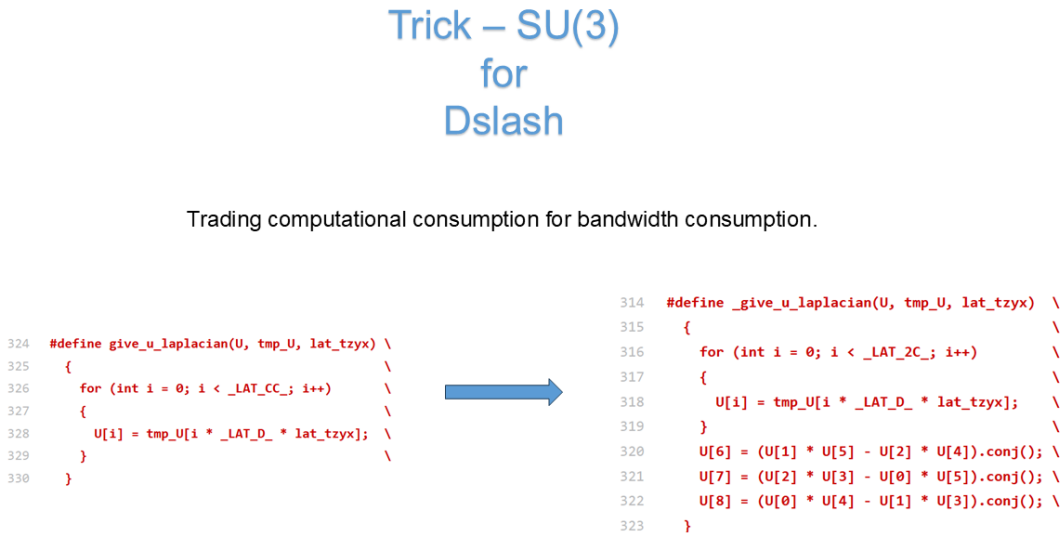


图表 13 优化 1 示意图

以其中的$1-\gamma_0$为例，其矩阵形状为 4*4，可见第一行乘以i的结果与第四行相同，同理第二行乘以i的结果与第三行相同。在实际编写代码时需要对这四行进行遍历，这时可以将第一行矩阵乘的结果乘以i后作为第四行的结果，第二行与第三行同理，以此可以减少几乎一半的计算消耗。

1.3.8.2 优化 2

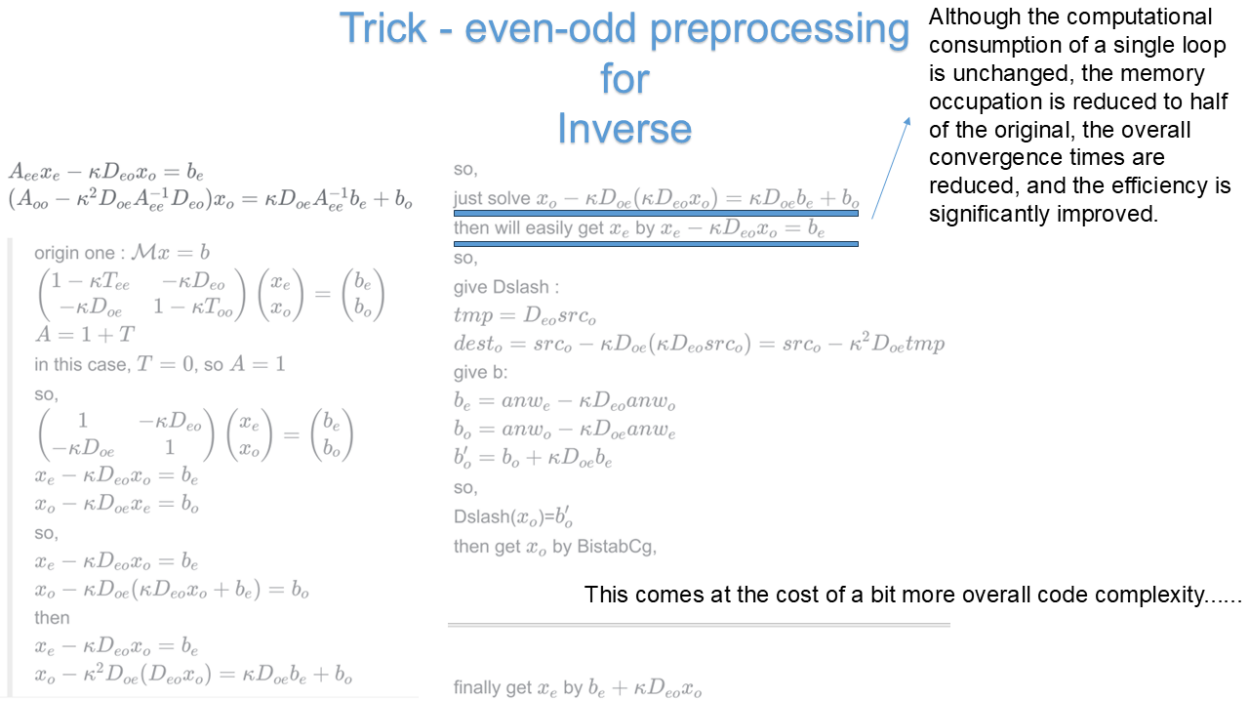
在进行 Dslash 操作时，其颜色维度体现为$SU(3)$矩阵元，其形式满足如下图右半部分所示的关系：



图表 14 优化 2 示意图

通过如上图所示的操作，可以将 GPU 上三分之一的全局内存到设备内存读写过程转换为计算过程，对于以硬件带宽为瓶颈的读写密集型程序有着相当不错的性能提升。

1.3.8.3 优化 3



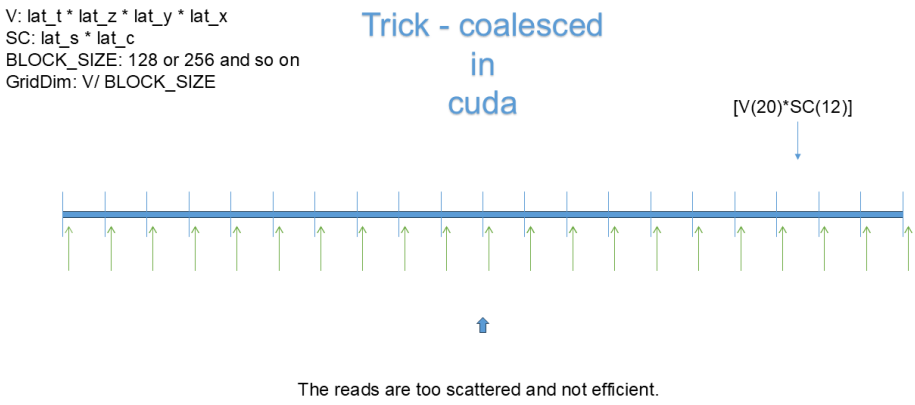
图表 15 优化 3 的示例图

对于求解线性方程组的通用算法（例如上述的 CG 算法与 BISTABCG 算法）有一种巧妙的优化手段——奇偶预处理。其大体思路为将求解一个线性方程组问题中的矩阵分为奇偶两部分，通过分块后的性质得到两个互相关的线性方程组，再求解其中一个线性方程组来得到一个部分的解，代入另外一个线性方程组得到另外一个解，最后合并两个部分解得到完整解。较为详细的推导如下图所示：

实际运行按上述优化过的代码，我们发现单次循环的Dslash计算量并没有减少，但是由于矩阵大小变为原来的一半即求解线性方程组的规模变为原来的一半导致算法迭代收敛变快，所需的总的迭代次数变少，总体来说花费的时间减少。

1.3.8.4 优化 4

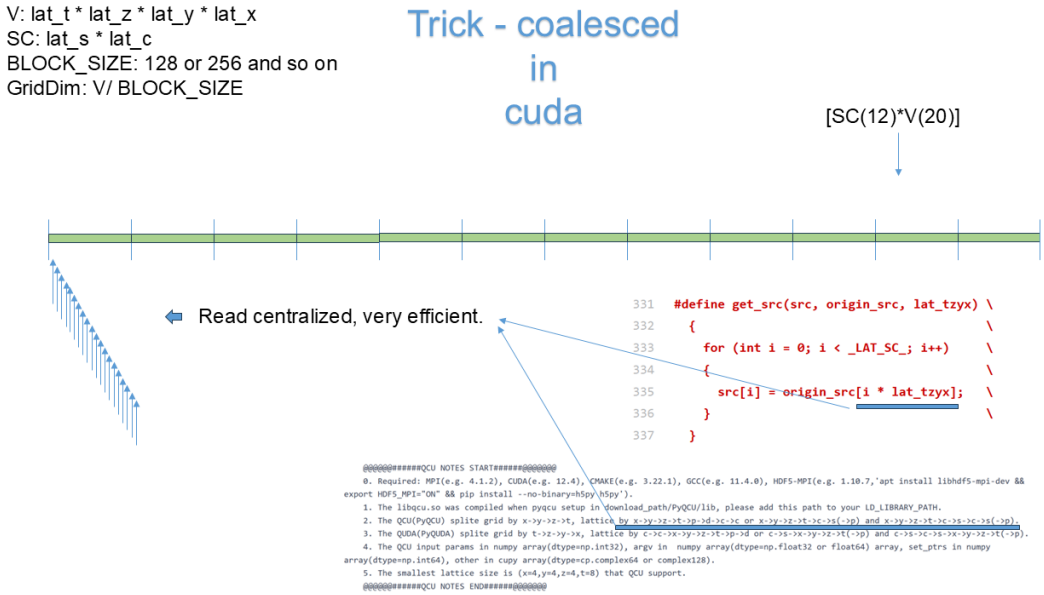
对于CUDA程序有一种与硬件架构直接联系的优化方案——访存合并。如下图所示是没有经过访存合并处理过的读写过程：



图表16 优化4示意图（未优化）

可见硬件层面需要分多次读写数据，但实际上现在的GPU基本都支持一次读写多个相邻的数据，即访存合并，这要求编写程序时的多线程分配逻辑要与数据排列逻辑相对应。然而，一般来说多线程分配逻辑参考的是时空维度（例如32*32*32*64的格子，其大小与常用GPU的CUDA核心数目相匹配，而其他维度显然太小了，无法充分发挥所有CUDA核心的性能），数据排列逻辑则按照朴实的物理认知将自旋维度（大小为4）与颜色维度（大小为3）排列在最后，这导致了两种逻辑不匹配，即不能访存合并，由此本软件将时空维度排列在最后。

访存合并时的读写过程如下图所示：



图表17 优化4示意图（已优化）

1.4 测试算例介绍

本次测试对象为 WilsonBistabCg（将 WilsonDslash 算符视为线性矩阵 Ax=b 中的 A）与 CloverBistabCg（将 CloverDslash 算符视为线性矩阵 Ax=b 中的 A），运行模板参考“test.clover.bistabcg-test6.py”，后续的“test.clover.bistabcg-test7.py”、“test.clover.bistabcg-test8.py”、“test.clover.bistabcg-test9.py”仅改变数据类型与格子大小（‘params[define._DATA_TYPE_]’，‘ params[define._LAT_*_]’）。

本算例在通过 MPI 启动后会根据‘-np’指定的线程数自动配置最佳的格点分割方案（为多 GPU 运行准备）。默认的最大的迭代次数为 1000，收敛容忍值为 1e-9（‘params[define._MAX_ITER_] = 1000’， argv[define._TOL_] = 1e-12），物理相关量 Mass 为 0.05（‘argv[define._MASS_] = 0.05’）。

本算例预留了读取已有 guage 与 fermion_in 与比对先前求解的 fermion_out 的功能（‘if os.path.exists("_gauge-test8.h5")……’），但在之前的预热测试中发现存储 guage 的时间成本太大（大约 309.238GB），于是本算例不读取 guage，而是每次都采用随机生成的 gauge（满足 SU(3)对称性，随机系数符合高斯分布，参考‘give_gauss_SU3’，但实际调用的为更加复杂的满足多线程的 CUDA C++程序），进而也无法使用 fermion_out 比对功能。

本算例通过 linux bash 脚本运行多次参数不同的程序，同时指定生成 log 文件，参考“test.clover.bistabcg-npX-test6.sh”：

```
mpirun -x UCX_TLS=self,sm,rc -np 32 '/public/home/zhangxin80699/openmpi_bind_mlnx.sh' python -u ./test.clover.bistabcg-test6.py > test.clover.bistabcg-np32-test6.log 2>&1
```

“test.clover.bistabcg-npX-test7.sh”、“test.clover.bistabcg-npX-test8.sh”、“test.clover.bistabcg-npX-test9.sh”与上述类似。

在此也给出环境配置脚本“env.sh”、节点绑定脚本“openmpi_bind_mlnx.sh”、任务提交脚本“_sbatch.sh”，在此衷心感谢本次测试中曙光老师们在环境配置以及后续 DEBUG 中的大力支持！希望国产曙光越来越好！

1.5 测试结果介绍及性能分析

1.5.1 测试结果介绍

本次测试的多线程任务结果输出为自定义的纯文本文件“*-np*-test[6-9].log”，经整理后得到附件中的“张鑫_2025 曙光新机器测试-PyQCU.xlsx”

在此列出其中的一个子表作为示例：

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex64，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试，其中（X:32,Y:64,Z:64,T:64）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test6）

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	1	1,1,1,1	8388608	152	1.40E-11	6.709556736	44.14182063	1.000	157	1.76E-11	24.66405581	157.0958969	1.000
2	2	1,1,1,2	4194304	152	1.42E-11	3.477103872	22.87568337	1.930	154	1.63E-11	12.98704896	84.33148675	1.863
3	4	1,1,2,2	2097152	144	1.42E-11	1.703268864	11.828256	3.732	151	1.71E-11	7.138976768	47.27799184	3.323
4	8	1,2,2,2	1048576	145	1.34E-11	1.244662784	8.583881269	5.142	146	1.54E-11	3.86076928	26.44362521	5.941
5	16	2,2,2,2	524288	140	1.27E-11	0.815100352	5.822145371	7.582	143	1.49E-11	2.485253888	17.37939782	9.039
6	32	2,2,2,4	262144	130	1.23E-11	0.80795648	6.215049846	7.102	999	nan	12.45687706	12.4693464	12.599

显存峰值占用大概为64GBi*8卡*15%/2=38.4GBi

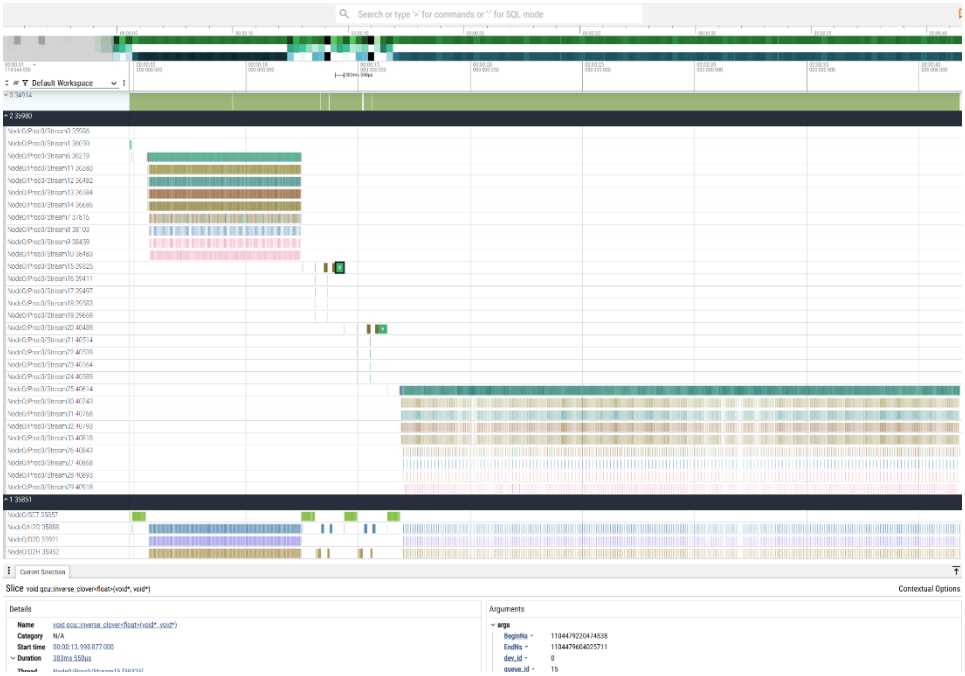
图表 18 “张鑫_2025 曙光新机器测试-PyQCU.xlsx”示例子表

表头为给定的各种参数，‘tol(x_o)为 1e-12’中的 x_o 为实际迭代中的 x，tol(x_o)表示对残差（b’-Ax_o）的范数的平方的容忍值，‘最终残差范数的平方(x_e)’表示迭代结束后通过奇偶运算还原的另外一半的解（参考 1.3.8.3）构成的残差的范数的平方。由于此过程相当于对迭代求解出的解做进一步的验证，所以此数据更能反应结果的正确性。

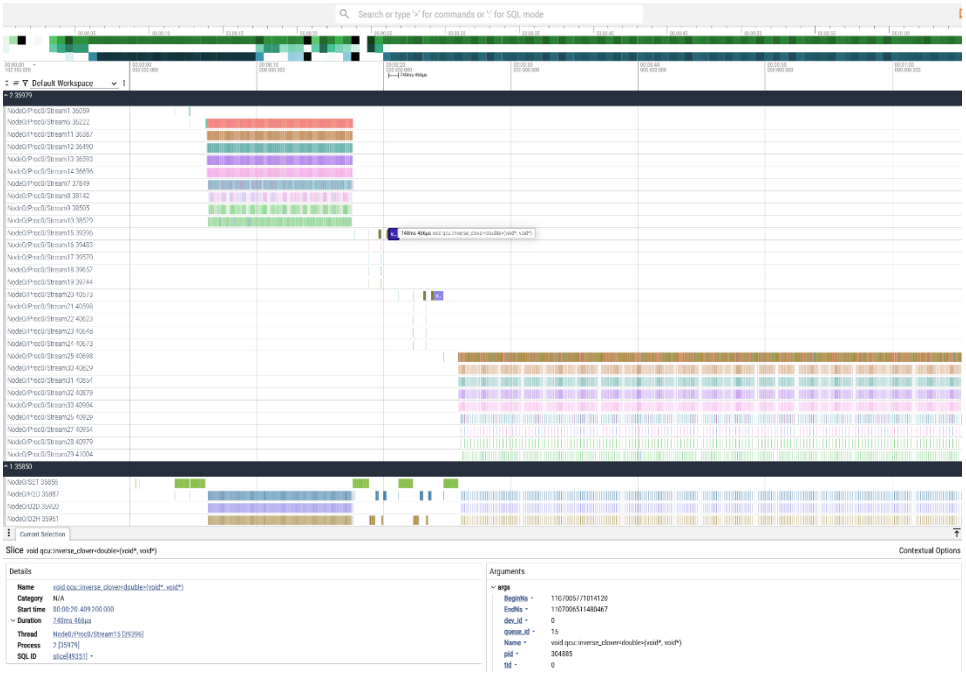
本次测试的单线程任务结果输出为 nvprof 的输出文件“*-test6.log.*”，建议用“https://ui.perfetto.dev”打开其中的 json 文件做性能分析。

1.5.2 性能分析

多线程任务的性能分析合并到 1.6，在此分析 nvprof 在运行单线程任务时输出的 json 文件，给出部分性能分析示意图：



图表19 单精度部分性能示意图，参考“test.clover.bistabcg-test6.log.json”



图表20 双精度部分性能示意图，参考“test.clover.bistabcg-test7.log.json”

1.6 应用扩展性分析

1.6.1 强扩展性分析

给出“张鑫 2025 曙光新机器测试-PyQCU.xlsx”的‘TEST6’、‘TEST7’、‘TEST8’、‘TEST9’子表：

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex64，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试，其中（X:32,Y:64,Z:64.T:64）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test6）

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	1	1,1,1,1	8388608	152	1.40E-11	6.709556736	44.14182063	1.000	157	1.76E-11	24.66405581	157.0958969	1.000
2	2	1,1,1,2	4194304	152	1.42E-11	3.477103872	22.87568337	1.930	154	1.63E-11	12.98704896	84.33148675	1.863
3	4	1,1,2,2	2097152	144	1.42E-11	1.703268864	11.828256	3.732	151	1.71E-11	7.138976768	47.27799184	3.323
4	8	1,2,2,2	1048576	145	1.34E-11	1.244662784	8.583881269	5.142	146	1.54E-11	3.86076928	26.44362521	5.941
5	16	2,2,2,2	524288	140	1.27E-11	0.815100352	5.822145371	7.582	143	1.49E-11	2.485253888	17.37939782	9.039
6	32	2,2,2,4	262144	130	1.23E-11	0.80795648	6.215049846	7.102	999	nan	12.45687706	12.4693464	12.599

显存峰值占用大概为64GBi*8卡*15%/2=38.4GBi

图表 21 “张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘TEST6’子表

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex128，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试，其中（X:32,Y:64,Z:64.T:64）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test7）

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	2	1,1,1,2	4194304	146	3.04E-21	5.659320891	38.76247186	1.000	154	1.56E-21	21.33112007	138.5137667	1.000
2	4	1,1,2,2	2097152	144	2.74E-21	2.843802679	19.74862972	1.963	146	1.28E-20	10.56981468	72.39599097	1.913
3	8	1,2,2,2	1048576	143	2.79E-21	1.909812752	13.35533393	2.902	145	3.04E-21	5.647270017	38.94668977	3.556
4	16	2,2,2,2	524288	136	3.00E-21	1.206956023	8.87467664	4.368	143	3.21E-21	3.423593426	23.94121277	5.786
5	32	2,2,2,4	262144	131	3.70E-21	0.839245233	6.40645216	6.051	144	2.45E-21	2.150249049	14.93228506	9.276
6	64	2,2,4,4	131072	136	1.29E-21	0.722651008	5.313610353	7.295	142	4.37E-21	1.306913047	9.203613007	15.050

显存峰值占用大概为64GBi*8卡*15%=76.8GBi

图表 22 “张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘TEST7’子表

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex64，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试，其中（X:128,Y:128,Z:128.T:256）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test8）

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	64	2,2,4,4	8388608	163	1.48E-11	7.912702976	48.54419004	1.000	168	1.63E-11	30.1419049	179.4161006	1.000
2	128	2,4,4,4	4194304	163	1.42E-11	5.454500352	33.46319234	1.451	170	2.07E-11	18.40790323	108.2817837	1.657
3	256	4,4,4,4	2097152	161	1.39E-11	3.90646144	24.26373565	2.001	160	1.63E-11	10.68423066	66.7764416	2.687
4	512	4,4,4,8	1048576	157	1.33E-11	1.608097152	10.24265702	4.739	156	1.53E-11	5.99338752	38.41915077	4.670
5	1024	4,4,8,8	524288	153	1.24E-11	2.381117696	15.56286076	3.119	147	1.48E-11	3.666724864	24.94370656	7.193

显存峰值占用大概为64GBi*64卡*50%=2048GBi

图表 23 “张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘TEST8’子表

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex128，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试，其中（X:128,Y:128,Z:128.T:256）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test9）

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	128	2,4,4,4	4194304	162	4.36466E-23	9.078154613	56.03799144	1.000	170	6.14E-23	28.5984921	168.2264241	1.000
2	256	4,4,4,4	2097152	156	9.40601E-23	6.953258005	44.5721667	1.257	159	6.36E-23	16.31604647	102.6166444	1.639
3	512	4,4,4,8	1048576	157	2.17205E-23	5.462116936	34.79055373	1.611	157	5.33213E-23	9.582966548	61.03800349	2.756
4	1024	4,4,8,8	524288	157	1.14308E-23	3.979191492	25.34516874	2.211	149	3.79539E-23	5.88710933	39.51080087	4.258

显存峰值占用大概为64GBi*64卡*50%*2=4096GBi

图表 24 “张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘TEST9’子表

上述图表中的 Wilson 算子与 Clover 算子在实际代码中相差一次‘give_clover’操作（经过奇偶处理后相当于两次一半格子大小‘give_clover’操作，在此看作一次完整格子大小的‘give_clover’操作）。在 BistabCG 中相当于单次迭代相差两次‘give_clover’操作，然而在单卡相同格子大小下的‘give_clover’操作（此操作不与 Host 和其他 Devic 通信）的运行效率相当依赖 GPU 的 DeviceToDevice 带宽（主要为读取 Clover 矩阵，其数据数量为格点数量乘以 144）。

因此，在相同的总格子大小下，随着线程数的增多，‘等效单卡格点数量’（X*Y*Z*T/线程数）减小，然而 GPU 的 DeviceToDevice 带宽不变，‘give_clover’操作耗时减少明显，‘CloverBistabCg 单次迭代耗时/ms’与‘WilsonBistabCg 单次迭代耗时/ms’越来越接近。

但是‘give_clover’操作以外的各种操作十分复杂，Wilson 算子与 BistabCg 迭代本身就涉及相当多的 D2H、H2H，D2D 通信与向量操作……在此限于篇幅，不做深入分析。

综上所述（‘总格点数：X*Y*Z*T’固定，‘线程数’变化），本软件的强扩展性良好，适合对特定规模的格点任务做更多线程（GPU）的加速计算。

补充，本次测试每次随机生成 guge，可能无解，“张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘TEST6’子表最后一行的‘nan’为正常现象。

1.6.2 弱扩展性分析

给出“张鑫 2025 曙光新机器测试-PyQCU.xlsx”的‘TEST6-8’、‘TEST7-9’、‘单精度总表’、‘双精度总表’子表：

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex64，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试，其中（X:32,Y:64,Z:64.T:64）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test6）

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	1	1,1,1,1	8388608	152	1.40E-11	6.709556736	44.14182063	1.000	157	1.76E-11	24.66405581	157.0958969	1.000
2	2	1,1,1,2	4194304	152	1.42E-11	3.477103872	22.87568337	1.930	154	1.63E-11	12.98704896	84.33148675	1.863
3	4	1,1,2,2	2097152	144	1.42E-11	1.703268864	11.828256	3.732	151	1.71E-11	7.138976768	47.27799184	3.323
4	8	1,2,2,2	1048576	145	1.34E-11	1.244662784	8.583881269	5.142	146	1.54E-11	3.86076928	26.44362521	5.941
5	16	2,2,2,2	524288	140	1.27E-11	0.815100352	5.822145371	7.582	143	1.49E-11	2.485253888	17.37939782	9.039
6	32	2,2,2,4	262144	130	1.23E-11	0.80795648	6.215049846	7.102	999	nan	12.45687706	12.4693464	12.599

显存峰值占用大概为64GBi*8卡*15%/2=38.4GBi

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex64，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试，其中（X:128,Y:128,Z:128.T:256）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test8）

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	64	2,2,4,4	8388608	163	1.48E-11	7.912702976	48.54419004	1.000	168	1.63E-11	30.1419049	179.4161006	1.000
2	128	2,4,4,4	4194304	163	1.42E-11	5.454500352	33.46319234	1.451	170	2.07E-11	18.40790323	108.2817837	1.657
3	256	4,4,4,4	2097152	161	1.39E-11	3.90646144	24.26373565	2.001	160	1.63E-11	10.68423066	66.7764416	2.687
4	512	4,4,4,8	1048576	157	1.33E-11	1.608097152	10.24265702	4.739	156	1.53E-11	5.99338752	38.41915077	4.670
5	1024	4,4,8,8	524288	153	1.24E-11	2.381117696	15.56286076	3.119	147	1.48E-11	3.666724864	24.94370656	7.193

显存峰值占用大概为64GBi*64卡*50%=2048GBi

图表 25 “张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘TEST6-8’子表

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	2	1,1,1,2	4194304	146	3.04E-21	5.659320891	38.76247186	1.000	154	1.56E-21	21.33112007	138.5137667	1.000
2	4	1,1,2,2	2097152	144	2.74E-21	2.843802679	19.74862972	1.963	146	1.28E-20	10.56981468	72.39599097	1.913
3	8	1,2,2,2	1048576	143	2.79E-21	1.909812752	13.35533393	2.902	145	3.04E-21	5.647270017	38.94668977	3.556
4	16	2,2,2,2	524288	136	3.00E-21	1.206956023	8.87467664	4.368	143	3.21E-21	3.423593426	23.94121277	5.786
5	32	2,2,2,4	262144	131	3.70E-21	0.839245233	6.40645216	6.051	144	2.45E-21	2.150249049	14.93228586	9.276
6	64	2,2,4,4	131072	136	1.29E-21	0.722651008	5.313610353	7.295	142	4.37E-21	1.306913047	9.203613007	15.050

显存峰值占用大概为64GBi*8卡*15%=76.8GBi

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex128，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试，其中（X:128,Y:128,Z:128.T:256）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test9）

序号	线程数	切分方式	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	WilsonBistabCg加速比	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg加速比
1	128	2,4,4,4	4194304	162	4.36466E-23	9.078154613	56.03799144	1.000	170	6.14E-23	28.5984921	168.2264241	1.000
2	256	4,4,4,4	2097152	156	9.40601E-23	6.953258005	44.5721667	1.257	159	6.36E-23	16.31604647	102.6166444	1.639
3	512	4,4,4,8	1048576	157	2.17205E-23	5.462116936	34.79055373	1.611	157	5.33213E-23	9.582966548	61.03800349	2.756
4	1024	4,4,8,8	524288	157	1.14308E-23	3.979191492	25.34516874	2.211	149	3.79539E-23	5.88710933	39.51080087	4.258

显存峰值占用大概为64GBi*64卡*50%*2=4096GBi

图表 26 “张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘TEST7-9’子表

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex64，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test6，https://gitee.com/zhangxin8069/PyQCU/tree/test8）

序号	总格点数量	线程数	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg单次迭代耗时-WilsonBistabCg单次迭代耗时/ms
1	8388608	1	8388608	152	1.40E-11	6.709556736	44.14182063	157	1.76E-11	24.66405581	157.0958969	112.9540762
2	536870912	64	8388608	163	1.48E-11	7.912702976	48.54419004	168	1.63065E-11	30.1419049	179.4161006	130.8719105
3	8388608	2	4194304	152	1.42E-11	3.477103872	22.87568337	154	1.63E-11	12.98704896	84.33148675	61.45580338
4	536870912	128	4194304	163	1.42E-11	5.454500352	33.46319234	170	2.07051E-11	18.40790323	108.2817837	74.81859137
5	8388608	4	2097152	144	1.42E-11	1.703268864	11.828256	151	1.71E-11	7.138976768	47.27799184	35.44973584
6	536870912	256	2097152	161	1.39E-11	3.90646144	24.26373565	160	1.62992E-11	10.68423066	66.7764416	42.51270595
7	8388608	8	1048576	145	1.34E-11	1.244662784	8.583881269	146	1.54E-11	3.86076928	26.44362521	17.85974394
8	536870912	512	1048576	157	1.33E-11	1.608097152	10.24265702	156	1.52817E-11	5.99338752	38.41915077	28.17649375
9	8388608	16	524288	140	1.27E-11	0.815100352	5.822145371	143	1.49E-11	2.485253888	17.37939782	11.55725245
10	536870912	1024	524288	153	1.24E-11	2.381117696	15.56286076	147	1.47887E-11	3.666724864	24.94370656	9.3808458

平均等效单卡格点数量相等的数据后得到下表：

序号	等效单卡格点数量	CloverBistabCg单次迭代耗时-WilsonBistabCg单次迭代平均耗时/ms
1	8388608	121.9129934
2	4194304	68.13719738
3	2097152	38.98122089
4	1048576	23.01811884
5	524288	10.46904912

图表 27 “张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘单精度总表’子表

PyQCU：Mass为0.05，tol(x_o)为1e-12，数据类型为complex128，_BLOCK_SIZE_=256，使用节点绑定脚本，使用Wilson与Clover算子时，用面源求解传播子的DCU性能测试）（具体数据参考https://gitee.com/zhangxin8069/PyQCU/tree/test7，https://gitee.com/zhangxin8069/PyQCU/tree/test9）

序号	总格点数量	线程数	等效单卡格点数量	WilsonBistabCg迭代数	WilsonBistabCg最终残差范数的平方(x_e)	WilsonBistabCg耗时/s	WilsonBistabCg单次迭代耗时/ms	CloverBistabCg迭代数	CloverBistabCg最终残差范数的平方(x_e)	CloverBistabCg耗时/s	CloverBistabCg单次迭代耗时/ms	CloverBistabCg单次迭代耗时-WilsonBistabCg单次迭代耗时/ms
1	536870912	128	4194304	162	4.36466E-23	9.078154613	56.03799144	170	6.14E-23	28.5984921	168.2264241	112.1884327
2	8388608	2	4194304	146	3.04E-21	5.659320891	38.76247186	154	1.56E-21	21.33112007	138.5137667	99.75129484
3	536870912	256	2097152	156	9.40601E-23	6.953258005	44.5721667	159	6.36E-23	16.31604647	102.6166444	58.04447775
4	8388608	4	2097152	144	2.74E-21	2.843802679	19.74862972	146	1.28E-20	10.56981468	72.39599097	52.64736126
5	536870912	512	1048576	157	2.17205E-23	5.462116936	34.79055373	157	5.33213E-23	9.582966548	61.03800349	26.24744976
6	8388608	8	1048576	143	2.79E-21	1.909812752	13.35533393	145	3.04E-21	5.647270017	38.94668977	25.59135584
7	8388608	16	524288	136	3.00E-21	1.206956023	8.87467664	143	3.21E-21	3.423593426	23.94121277	15.06653613
8	536870912	1024	524288	157	1.14308E-23	3.979191492	25.34516874	149	3.79539E-23	5.88710933	39.51080087	14.16563213

平均等效单卡格点数量相等的数据后得到下表：

序号	等效单卡格点数量	CloverBistabCg单次迭代耗时-WilsonBistabCg单次迭代平均耗时/ms
1	4194304	105.9698638
2	2097152	55.3459195
3	1048576	25.9194028
4	524288	14.61608413

图表 28 “张鑫 2025 曙光新机器测试-PyQCU.xlsx”‘双精度总表’子表

当单卡（GPU）与其对应 CPU 没有性能瓶颈时，弱扩展性主要影响因素为单节点内的通信与跨节点间的通信。

综上所述（‘等效单卡格点数量：X*Y*Z*T/线程数’固定，‘线程数’变化），本软件弱扩展性良好，适合使用特定数量的线程（GPU）运行更大规模的格点任务。

1.7 附录

张鑫
2025曙光新机器测

1.8 引用

[1] 刘川. 格点量子色动力学导论. 北京大学出版社, 1 edition, 6 2017.

[2] 蒋翔宇. 轻强子性质的格点 QCD 研究. PhD thesis, 中国科学院高能物理研究所, 2023.

[3] Xiangdong Ji. Parton physics from large-momentum effective field theory. Science China Physics, Mechanics & Astronomy, 57:1407–1412, 2014.

[4] Fei Yao, Lisa Walter, Jiunn-Wei Chen, Jun Hua, Xiangdong Ji, Luchang Jin, Sebastian Lahrtz, Lingquan Ma, Protick Mohanta, Andreas Scha¨fer, et al. Nucleon transversity distribution in the con- tinuum and physical mass limit from lattice qcd. Physical Review Letters, 131(26):261901, 2023.

测试单位：中国科学近代物理研究所

测试负责人：张鑫

测试日期：2025 年 8 月 29 日

盖章：