计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(9):25−32 [doi: 10.15888/j.cnki.csa.007036]

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

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①

田英齐[1,2], 毕玉江[3], 贺雨晴[1,2], 马运恒[2,3], 刘朝峰[2,3], 徐　顺[1]

[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](中国科学院 高能物理研究所, 北京 100049)
通讯作者: 徐　顺, E-mail: xushun@sccas.cn

摘　要: 格点量子色动力学 (Lattice Quantum Chromo Dynamics, LQCD) 是目前已知能系统研究夸克及胶子间低能强相互作用的非微扰计算方法. 计算结果的统计和系统误差原则上都是可控的, 并能逐步减少. 基于格点 QCD 的基本原理, 更大的格子体积意味着可以计算更大空间的物理过程, 并且可以对空间进行更加精细的划分, 从而得到更加精确的结果. 因而大体系的格点计算对 QCD 理论研究有着重要意义, 但对程序计算性能提出了更高要求. 本文针对格点 QCD 组态生成和胶球测量的基本程序, 进行了其大规模并行分析和性能优化的研究. 基于格点 QCD 模拟采用的 blocking 和 even-odd 算法, 我们设计了基于 MPI 和 OpenMP 的并行化算法, 同时设计优化数据通信模块: 针对复矩阵的矩阵乘等数值计算, 提出了向量化的计算优化方法: 针对组态文件输出瓶颈, 提出了并行输出组态文件的实施方法. 模拟程序分别在 Intel KNL 和“天河 2 号”超级计算机 x86_64 队列进行了测试分析, 证实了相应的优化措施的有效性, 并进行了相应的并行计算效率分析, 最大测试规模达到了 1728 个节点 (即 41472 CPU 核).

关键词: 格点量子色动力学; 并行计算; 组态产生; 胶球测量; 性能优化

引用格式: 田英齐,毕玉江,贺雨晴,马运恒,刘朝峰,徐顺.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. http://www.c-s-a.org.cn/1003-3254/7036.html

Performance Optimizing for Large-Scale Lattice Quantum Chromodynamics of Configuration Generating and Glueball Measurement

TIAN Ying-Qi[1,2], BI Yu-Jiang[3], HE Yu-Qing[1,2], MA Yun-Heng[2,3], LIU Zhao-Feng[2,3], XU Shun[1]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)
[3](Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Lattice Quantum Chromo Dynamics (LQCD) is a non-perturbative method for the study of low-energy strong interactions between quarks and gluons. The statistical and systematic uncertainties of the results from LQCD are in principle all under control and can be reduced steadily. Based on LQCD theory, larger volume of lattice grids can calculate physical processes in larger space. And one can divide the space more meticulously to obtain more accurate results. Therefore, large system LQCD calculation is of great significance to the study of QCD theory, but is demanding for higher program computing performance. In this work, the large-scale parallel analysis and performance optimization of LQCD configuration generating and glueball measurement program are studied. Based on the blocking and even-odd algorithms used in LQCD simulation, we design a parallel algorithm based on MPI and OpenMP, and design an optimized data communication module. Aiming at the bottleneck of configuration file output, the solution of configuration file

① 基金项目: 国家重点研发计划 (2017YFB0203202); 国家自然科学基金面上项目 (11575197)
Foundation item: National Key Research and Development Program of China (2017YFB0203202); General Program of National Natural Science Foundation of China (11575197)
收稿时间: 2019-02-21; 修改时间: 2019-03-08; 采用时间: 2019-03-18; csa 在线出版时间: 2019-09-05

parallel output is put forward. The simulation programs are tested and analyzed on an Intel KNL platform and the x86_64 queues of “Tianhe 2” supercomputer. The results verify the effectiveness of the corresponding optimization measures, and the efficiency of parallel simulation is also analyzed. The maximum size of the test is 1728 nodes (i.e. 41 472 CPU cores).
Key words: lattice quantum chromodynamics; parallel computation; configuration generation; glueball measurement; performance optimization

格点量子色动力学 (Lattice Quantum Chromo Dynamics, LQCD) 是描述夸克和胶子之间的强相互作用的理论, 它将连续时空离散成有限大小的四维超立方晶格, 将 QCD 理论中的夸克场放在超立方晶格的格点上, 晶格点之间的连线上放置胶子场. 可测量的物理量是这两种场的函数, 夸克场和胶子场的分布情况反映了 QCD 的动力学性质, 所以只要在格子上给定了这两种场的分布, 原则上就能计算所有物理量 (的期望值), 因此格点 QCD 模拟[1]是一个重要的基础方法, 具体来说是目前已知能系统研究夸克及胶子间低能强相互作用的不可缺少的非微扰计算方法.

格点 QCD 的一个重要研究课题是胶球性质的研究. 胶球是传统夸克模型中没有的强子态, 由能发生自相互作用的胶子组成. 我国的 BESIII 合作组一直在寻找胶球, 并发现了一些可能的候选者. 但要确认它们, 需要将理论计算与实验测量结合起来, 因此胶球性质的理论研究是一个重要的前沿课题, 胶球的理论计算工作有助于基础物理科学的新发现.

格点 QCD 是一个典型的计算密集型同时也是通信密集型的计算工作. 常用马可夫链实现蒙特卡洛重点抽样, 并且采用各种高级算法如 pseudo-heat-bath 算法来加速组态生成过程的收敛; 高精度大规模组态抽样的文件输出需要高效的计算 IO; 格点化的场近程作用是四维 stencil 计算的一种具体形式, 其计算效率不仅受处理器频率影响也受内存缓存大小影响; 另外格距越小, 计算物理观测结果越精确, 但带来的计算量指数增加, 造成了很多物理研究中采用近似算法. 总之, 格点 QCD 对并行计算优化提出了很高的需求.

针对格点 QCD 的并行计算优化受到高度重视, 超级计算机上格点 QCD 计算优化及作业调度优化多次获得戈登贝尔 (Gordon Bell) 奖 (分别是 1987 年、1995 年、1998 年和 2006 年, 以及 2018 年入围决赛). 2005 年前后, IBM、 RBC/UKQCD 和 RIKEN 等设计了专门用于格点 QCD 计算的超级计算机 QCDOC (QCD on Chips), 取代了其更早的 QCDSP (QCD on DSPs) 机器, IBM 后来还打造了更强大的 Blue Gene/L 平台, 其 Pavlos Vranas 团队基于该平台实现了 3.2 万核 12 Tflops 的格点 QCD 应用高性能计算, 从而获得了 2006 年的戈登贝尔奖; 2018 年 Pavlos Vranas 再次采用 GPU 加速格点 QCD 计算, 在美国 Sierra 和 Summit 超算平台上以高效的计算性能晋级了当年的戈登贝尔决赛[2]. 格点 QCD 已逐渐成为美国、日本及欧洲超算平台上的重要应用[3].

当前以美国 USQCD 组织研发的格点 QCD 软件套件为领域内主流软件工具, 其为四层软件框架, 最上层应用层包括 Chroma、CPS 和 MILC 等, 下面三层为支撑库层, 分别实现消息通讯、数据结构定义和基本算法实现. MILC 软件工具也成为 SPEC CPU2006 基准测试程序[4]. QUDA 是基于 CUDA 的格点 QCD 计算库[5], 实现了较底层的计算加速, 可以和上层的 Chroma 和 MILC 等集成. Mikhail Smelyanskiy 等人基于多核处理器的 Cache 架构进行了格点 QCD 的线程化 MPI 优化, 相关工作发表在 2012 年 Supercomputing Conference 会议上[6], Issaku Kanamori 基于 Intel KNL 和 Oakforest-PACS 超算系统做了格点 QCD 设计实现的讨论, 涉及 SIMD 和 MPI 优化方面[7]. 日本高能加速器研究机构 (KEK) 也于 2009 年开始了用于格点 QCD模拟的 Bridge ++软件工具研发[8]. 我国存在一个“中国格点 QCD”的合作组, 当前正在努力推出国际上较有影响的格点 QCD 计算软件.

本文研究工作探讨格点 QCD 计算的瓶颈问题, 并提出有效的并行优化计算方案. 相关程序原始代码由中科院高能物理研究所研究员、“中国格点 QCD”成员陈莹提供. 为了实现其大规模并行计算、本文进行多方面的并行计算分析和性能优化研究.

1 格点 QCD 模拟的基本流程

本节给出在淬火近似下格点 QCD 生成规范场组态的基本流程. 一个具体的胶子场分布叫做一个组态, 产生上述的组态需要涉及名为马可夫链的随机行走过

程. 整个链条通常从一个最简单的或随机的胶子场分布开始起步. 每一次步进更新就意味着由当前的场分布更新到另一种物理上允许的分布.

组态生成与测量流程可以分为如下几步:

1) 从一个给定的场分布开始, 计算每个 link 的周遭环境, 使用 pseudo-heat-bath 算法计算 link 更新的几率, 据此决定是否将其更新, 对更新造成的 link 形变加以修正;

2) 重复以上过程将整个格子上的所有 link 更新若干次, 直到达到热平衡, 热平衡之后继续不断更新格子上的 link, 使之成为新的组态, 将新的组态导出保存;

3) 在生成好的组态上测量胶球关联函数, 继续生成组态, 保存和测量, 直到测量了足够多的组态样本.

更详细的程序流程见图 1.

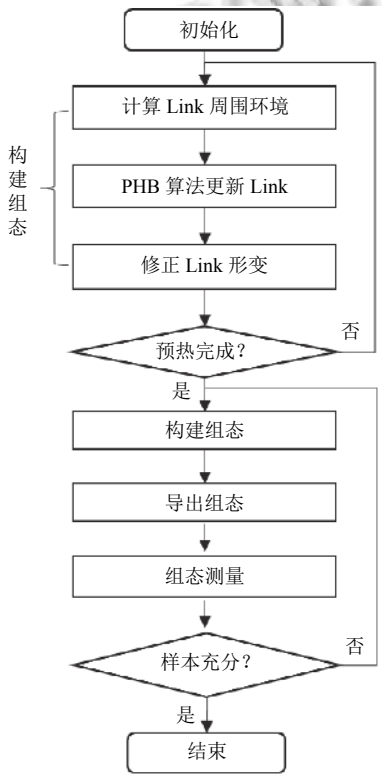

图 1 胶子场组态生成与测量的流程图

在我们的程序中使用的 pseudo-heat-bath 算法用于加快收敛. 当我们利用马可夫链抽样, 每个样本点对应的胶子场分布都是我们所要的组态. 在有夸克场时, 胶球与通常介子态之间的混合会给研究增加复杂度, 需要更大的计算量. 因此作为第一步, 忽略夸克场的所谓淬火近似下的模拟计算具有重要意义. 对淬火组态产生程序的优化, 目的在于给出更多的组态, 从而得到更大的统计量和更好的信号. 淬火近似下胶球谱的研究工作和 J/psi 粒子辐射衰变到标量胶球或张量胶球的淬火近似研究工作[9–11], 其结果都发表在物理学一流学术期刊上. 这些计算中用到组态产生程序正是本研究优化的对象.

程序实现中主要涉及的功能模块有:

1) 数据初始化模块: 体系的格点划分, 内存数组分配等;

2) 组态生成模块: 利用 pseudo-heat-bath 算法生成满足分布的组态;

3) 组态计算模块: 基于抽样的组态计算胶球的物理性质;

4) 辅助分析模块: 组态验证、统计计时、并行写组态等.

2 并行计算关键问题

QCD 理论是一个基于 SU(3) 规范对称性的理论. 在格子上, 每一条晶格连线上的胶子场的值由一个 3 乘 3 的 SU(3) 复矩阵描写: $U_\mu(x)$. 组态生成程序从一个给定的场分布开始, 计算每条连线 (也叫 link 变量) 的周遭环境, 这里涉及大量矩阵计算, 并且每个格点需要做独立性分析以适合并行化计算处理.

我们的程序按照 pseudo-heat-bath 算法来更新格子上的每个 link 变量的值, 按以下几率分布将一个 link 变量的值从 $U_\mu(x)$ 更新为 $U_\mu(x)'$:

$$dP(U_\mu(x)') = d(U_\mu(x)')\exp\left[\frac{\beta}{3}Retr(U_\mu(x)'A)\right]$$

其中, A 刻画该 link 周围的环境, 由其他 link 变量对应的矩阵值的乘积给出. 计算 A 的过程中会涉及到相应 link (link 构成订书针的形状, 被称为“staple”) 相乘的计算, 其示意图如图 2.

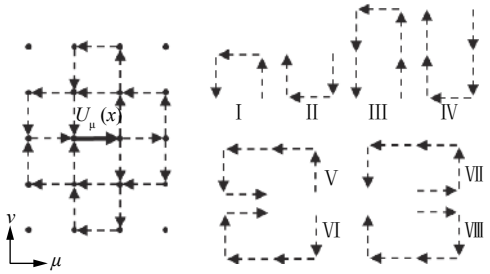

图 2 link 周围环境 (staple) 计算示意图

要计算图中的 link 变量对应的 A, 需要对其近邻以及次近邻的格点上相关的 link 变量的乘积 (即 3×3 复矩阵的矩阵乘积) 进行求和, 即对上图中的 8 个 “staple”进行求和:

$$A=\sum_{i=1}^{8}A_i$$

其中, A_i 为每个 staple 内涉及的 link 的乘积, 以图 2 中的 I 为例, 其计算公式如下:

$$A_1=U_\nu(x+\hat{\mu})U_\mu^\dagger(x+\hat{\nu})U_\nu^\dagger(x)$$

由此可以看出, 这里涉及大量 3×3 复矩阵的矩阵乘积计算优化问题.

由于要更新一个 link 变量, 只与其近邻和次近邻的相关 link 有关, 而与其他 link 无关. 因此这些相互无关的 link 之间可以并行处理. 传统的格点 QCD 程序一般采用 blocking 和 even-odd 算法进行并行. 该算法将相应的 link 分配到不同的 block 上, 根据 block 的 x、y、z、t 的坐标之和的奇偶分为两组, 每组内可以进行并行计算. 然而我们的胶球模拟中涉及到了近邻以及次近邻的相互作用, 因此需要首先将 24 个 block 组合为一个 hyper-block, 之后对 hyper-block 进行 even-odd 的并行计算, 来生成相应的组态. 一个二维结构格子的 Hyper-blocking 与 even-odd 算法示意如图 3.

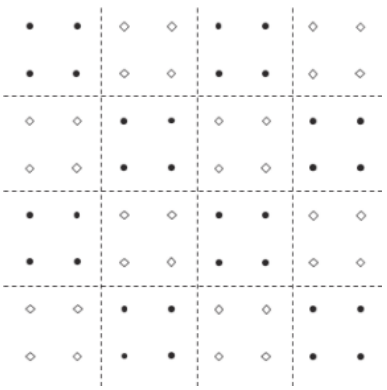

图 3 even-odd 算法示意如图

图 3 中同色的虚线框之间可以进行并行计算.

另外格点 QCD 需要重复以上将整个格子上的所有 link 的值更新若干次, 达到热平衡, 热平衡之后继续不断更新格子上的 link 的值, 使之成为新的组态,而组态的存储空间占用也比较大, 因此将新的组态导出保存也是并行化时需要考虑的问题.

3 并行分析与优化策略

3.1 瓶颈与热点分析

为了实现进行大规模的格点 QCD 计算, 我们首先需要分析程序运行可能的热点. 原型程序采用消息传递接口 (Message Passing Interface, MPI) 来实现多节点并行处理, 采用基于格点区域计算任务划分的方式来实现并行处理, 需要在各个 MPI 进程之间进行格点边界通信. 通信占比与格子的划分方式有关, 具体地说与子格子的面积体积比有关. 更小的面积体积比有更小的通信占比. 我们知道同样体积的情况下, 正方体比长方体有更小的面积体积比, 因此导致了在通常情况下, x、y、z 方向的格子规模相同, 且划分方式相同, t 方向格子规模与划分方式可以与 x、y、z 方向不同. 因而在格子规模 V 和计算节点规模 N 确定的情况下, 确定划分方式即为求以下方程的正整数解:

$$l_t\times l_{xyz}^3=V/N$$

其中, l_t 是 t 方向的子格子大小, l_{xyz} 是 x、y、z 方向的子格子大小. 该方程的正整数解一般是确定的, 因而不能灵活的协调通信与计算的比例, 来获取更优的性能. 为了更加灵活的分配计算任务, 获取通信与计算的平衡, 我们可以加入多线程 (如 OpenMP) 的并行处理, 更加充分的利用计算资源.

基于原型程序的初步测试, 我们发现在常用的计算规模下, 计算耗时占比 60% 以上. 此程序的计算中, 涉及大量 3×3 复矩阵的矩阵乘积计算, 是此程序的计算热点.

另外, 运行时输出组态文件无疑也是一个性能瓶颈, 尤其是在格子规模较大的情况下. 对于 964 大小的格子, 组态文件大小为 40 GB 以上. 然而原型程序采用了串行输出组态文件的方法, 耗时极大, 因此也需要采取优化措施.

因此通过分析与测试, 为了实现程序有效并行处理, 我们考虑以下方面:

1) 平衡通信与计算占比

2) 复矩阵矩阵乘等数值计算函数

3) 输出组态文件

以下分别介绍这几方面的计算瓶颈或计算热点的优化设计.

3.2 MPI+OpenMP 并行优化

针对胶球模拟的特点, 在 $xyzt$ 四个维度上均可以

进行并行. 程序可以设置各个方向分别的划分方式. 假设格子的体积设置为 (N_t, N_x, N_y, N_z), 在 t、x、y、z 方向分别划分为 m_t、m_x、m_y、m_z 份, 则每个 MPI 进程上的格子体积为 $(N_t/m_t, N_x/m_x, N_y/m_y, N_z/m_z)$, 划分方式记为 (m_t, m_x, m_y, m_z). 以 (2, 2, 2, 2) 划分方式为例, 4 个维度的并行分割方式如图 4 所示.

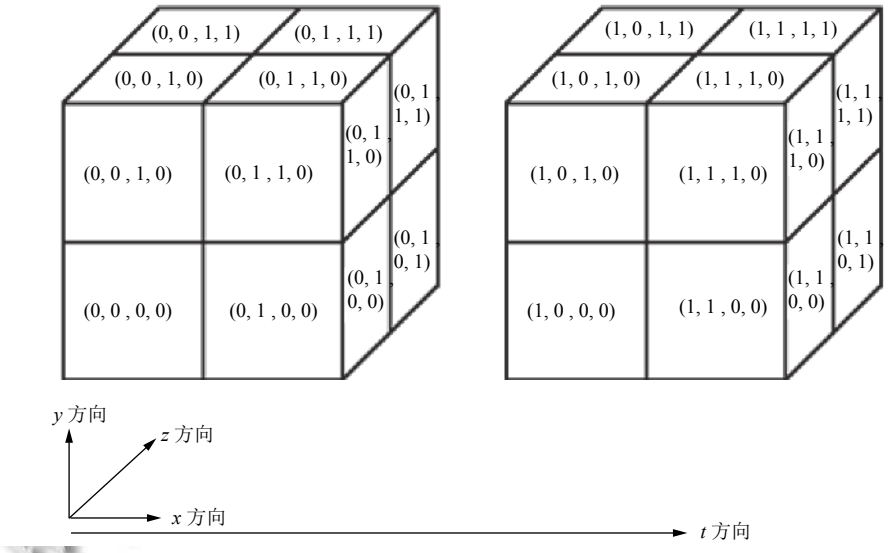

图 4 MPI 并行划分方式 (2, 2, 2, 2) 示意

基于前述的分析, 对于相同规模的格子, 使用 MPI+OpenMP 的并行方式, 可以有效增加单个 MPI 进程上格子的规模, 从而降低子格子的面积体积比, 从而减少通信的占比. 因此我们采用了动态负载平衡的方法, 在 MPI 进程内部进行 OpenMP 的并行. 图 5 以二维结构示意单个 MPI 进程内部 2 个 OpenMP 线程情况下的 OpenMP 并行情况:

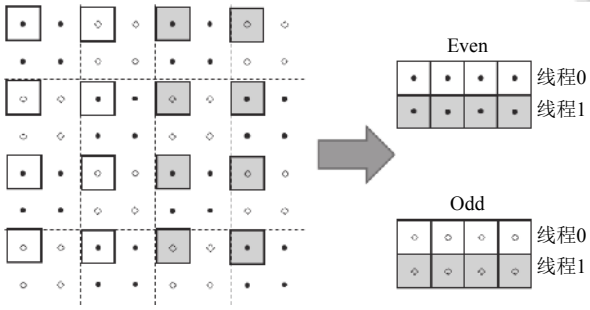

图 5 OpenMP 并行示意图

3.3 矩阵操作的向量化优化

在程序中, 涉及到的计算主要是 3×3 的复矩阵的矩阵乘计算, 即两个 3×3 的复矩阵相乘, 以及复矩阵与矩阵的厄密共轭矩阵相乘. 具体计算如下所示:

$$u\times v=\begin{bmatrix} a_1+b_1i & a_2+b_2i & a_3+b_3i \\ a_4+b_4i & a_5+b_5i & a_6+b_6i \\ a_7+b_7i & a_8+b_8i & a_9+b_9i \end{bmatrix}$$

$$\begin{bmatrix} a_1+b_1i & a_2+b_2i & a_3+b_3i \\ a_4+b_4i & a_5+b_5i & a_6+b_6i \\ a_7+b_7i & a_8+b_8i & a_9+b_9i \end{bmatrix}$$

$$u\times v^*=\begin{bmatrix} a_1+b_1i & a_2+b_2i & a_3+b_3i \\ a_4+b_4i & a_5+b_5i & a_6+b_6i \\ a_7+b_7i & a_8+b_8i & a_9+b_9i \end{bmatrix}$$

$$\begin{bmatrix} a_1-b_1i & a_4-b_4i & a_7-b_7i \\ a_2-b_2i & a_5-b_5i & a_8-b_8\mathbf{i} \\ a_3-b_3i & a_6-b_6i & a_9-b_9i \end{bmatrix}$$

我们针对复矩阵矩阵乘操作, 设计了多种不同的计算方法, 以下将分别介绍:

1) 我们基于结构体的数组转化为数组的结构体 (AoS to SoA) 的思想, 设计了数组长度可变的结构体数组, 用来存储中间数据, 提高了计算效率. 经过测试, 结构体内数组长度为 8 时的效果最好.

2) 我们将复矩阵矩阵乘的数学算式完全展开, 利用编译器的自动向量化功能进行相关代码的向量化.

3) 我们针对 AVX-512 指令集架构, 利用 Intel Intrinsics 设计了手动向量化的算法. 对于这样的矩阵计算, 我们可以将 v 矩阵的行向量实部和虚部[c1, d1,

c2, d2, c3, d3]组装为一个向量, 针对实部和虚部采用 _mm512_fmaddsub_pd() 函数进行计算.

3.4 输出组态文件的优化

原始的组态文件按照一般多维数组的存储方式, 以行主元的方式进行存储, 即依次按照 z、y、x、t 下标由小到大进行存储. 原程序依据该存储方法由 0 号进程进行数据收集及输出到文件的工作. 以 4^2 大小的格子, (2, 2) 划分方式为例, 4 个 MPI 进程的组态文件输出方式如图 6 所示.

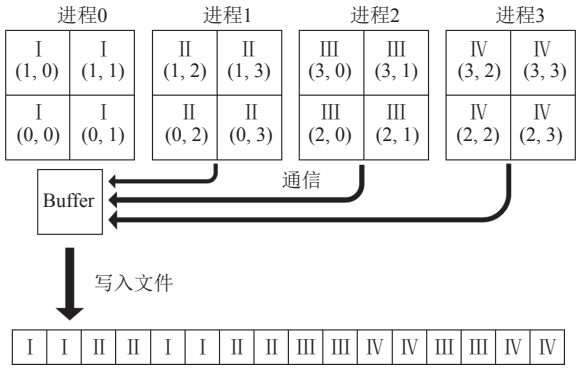

图 6 MPI 并行写入优化前算法示意图

对于 N 个进程的情况, 原始的输出方式需要进行的通讯量为 $(N–1)\times V_{sub}$, 其中 V_{sub} 为子格子的大小. 而且需要进行 $N\times m_t\times m_x\times m_y$ 次不连续的写操作.

我们修改了组态文件的数据排布方式, 将整个文件划分为 N 部分, 并在每个部分内按照数据连续的方式进行排布, 依次实现了多个 MPI 进程并行输出组态文件的算法. 采用此种方法后, 理论上没有通信操作, 也没有不连续的内存写操作, 可以极大的提高输出组态文件的性能. 并行写采用了 MPI-IO, MPI 中可以使用 mpi_file_open 让所有进程都打开同一个文件, 使用 mpi_file_write_at 函数独立互不干扰地写每个进程自己负责的数据. 图 7 展示了优化后的数据写入方式:

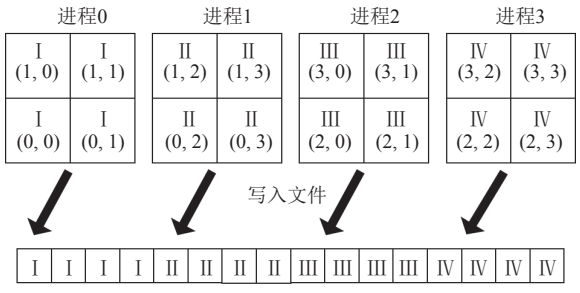

图 7 MPI 并行写入优化后算法示意图

4 实验结果与分析

4.1 实验准备

我们在“天河 2 号”超级计算平台以及 KNL 平台上测试了我们的代码, 测试平台的信息如表 1 所示.

表 1 软硬件环境清单

机器名称	CPU	内存	编译器
天河 2 号	Intel Xeon E5-2692 v2	64 GB	Intel C++ Compiler 2013
KNL	Intel Xeon Phi 7210	64 GB	Intel C++ Compiler 2018

其中“天河 2 号”超级计算机不支持 AVX 指令向量化, 而 Intel KNL 支持 AVX512 指令向量化.

4.2 OpenMP 多线程加速测试

基于前述 3.1 节的整体分析与 3.2 节提出的 OpenMP 多线程加速方案, 我们设定了两种格子规模和计算规模的情况, 考察 MPI 单节点上采用 OpenMP 多线程调整计算与通信占比以优化并行效率的效果. 同时我们也考察了优化前后程序强可扩展性和弱可扩展性的对比情况.

4.2.1 常规规模测试结果

我们首先针对常用的格子与计算规模, 进行了程序优化的测试. 我们设定格子大小为 96×48^3, 使用了 5184 个计算核心, 测试结果如表 2.

表 2 常规规模测试结果

划分方式 (t,x,y,z)	mpi	omp	计算时间 (s)	加速比
24,6,6,6	5184	1	51.04	1.00
12,6,6,6	2592	2	28.84	1.77
6,6,6,6	1296	4	15.37	3.32

从测试结果可以看出, 在规模不变的情况下, 原始的 MPI 并行并不是最优结果. 随着 OpenMP 线程数的增加, 性能逐渐提高, 达到了理想的计算效率. 较小规模的情况下最高达到了 3.32 倍的加速.

4.2.2 大规模测试结果

我们又选择了当前环境下可使用的最大计算规模进行了测试. 我们设定的格子大小为 96^4, 使用了 41 472 个计算核心, 测试结果如表 3.

表 3 大规模测试结果

划分方式 (t,x,y,z)	mpi	omp	计算时间 (s)	加速比
24,12,12,12	41 472	1	51.02	1.00
12,12,12,12	20 736	2	37.30	1.37
6,12,12,12	10 368	4	54.53	0.94
24,6,6,6	5184	8	71.02	0.72
12,6,6,6	2592	16	42.81	1.19

从测试结果可以看出, 在规模不变的情况下, 原始的 MPI 并行同样不是最优结果. 加入 OpenMP 并行后, 可以使得格子的划分更加灵活, 可以找到合理的划分方式, 达到理想的计算效率. 较大规模的情况下最高达到了 1.37 倍的加速.

4.2.3 划分方式对计算效率的影响分析

根据前述 3.1 与 3.2 节的分析, 我们可以知道减少通信占比的方法是增大子格子的面积体积比. 在常规规模的测试中, 我们可以看出, 通过加入 OpenMP 并行, 可以有效减小子格子的面积体积比, 取得了很好的加速效果. 对于大规模的测试结果, 我们发现单纯增大子格子面积体积比并没有导致计算性能的持续提高, 针对这一情况, 我们分析原因与计算机的网络结构有关. 计算任务在节点上的分配方式需要进一步的优化研究.

4.2.4 强可扩展性测试

针对 MPI+OpenMP 并行方案, 我们测试了程序的强扩展性. 我们对比了只有 MPI 并行的情况和 MPI 与 OpenMP 混合并行的情况. 强扩展性测试用例的格子大小为 64^4, 最高并行达到了 4096 核. 测试结果如图 8 所示.

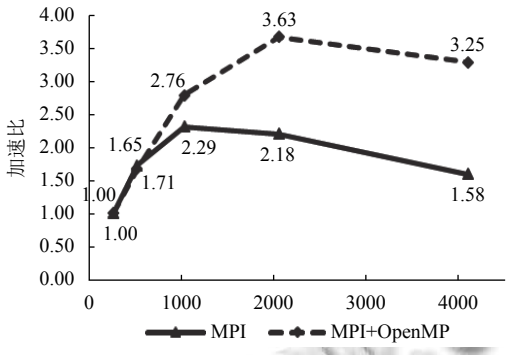

图 8 强扩展性测试结果

结果显示在应用了 OpenMP 多线程加速后, 强扩展性有了明显的提高.

4.2.5 弱可扩展性测试

我们针对 MPI+OpenMP 并行方案, 测试了程序的弱扩展性. 弱可扩展性测试的规模为: 每个核上分配有 8^4 大小的格子, 依次测试了 256 核~6144 核的计算结果. 测试结果如图 9.

图 9 显示了各次计算中单步生成组态的计算时间, 在理想情况下, 单步时间应该是保持不变的, 然而随着体系的增大, 实际会有所增加. 我们用最小二乘法线性拟合数据结果, 得到了两条拟合曲线的斜率, 以此来评价程序的弱扩展性. 结果显示在应用了 OpenMP 多线程加速后, 程序的弱可扩展性也得到了提高.

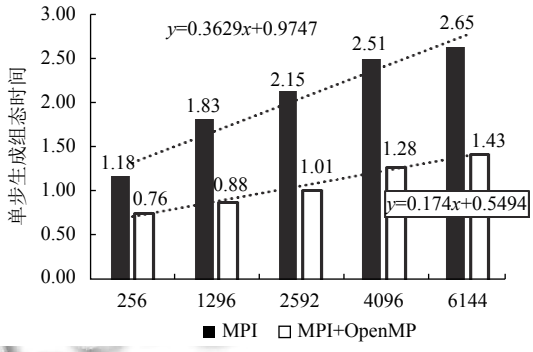

图 9 弱扩展性测试结果

4.3 向量化优化测试

针对 3.3 节提出的向量化优化方案, 我们在 KNL 平台上测试了 AVX-512 向量化优化的计算结果. 我们通过-xMIC_AVX512 编译选项控制向量化开关, 进行了向量化与不向量化的测试. 测试中使用了 32^4 大小的格子, 运行在 MPI+OpenMP 的模式下, MPI 绑定 CPU 采用 I_MPI_PIN_DOMAIN = omp:compact, OpenMP 设置亲缘性 KMP_AFFINITY = compact. 测试基准为未向量化的 Fortran 内置的 matmul 函数. 后续测试了采用-xMIC_AVX512 编译选项进行向量化的 matmul 函数; 采用 AoS to SoA 优化与-xMIC_AVX512 编译选项结合的算法; 采用公式展开和编译选项结合的算法; 以及采用了 Intel Intrinsics 手动向量化的算法. 几种算法在 16MPI + 1OpenMP 的情况下结果如图 10.

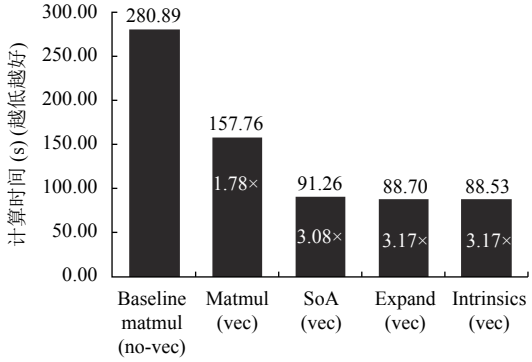

图 10 16MPI + 1OpenMP 各向量化优化测试结果及加速比

运行在 16MPI+16OpenMP 的情况下, 测试结果如图 11.

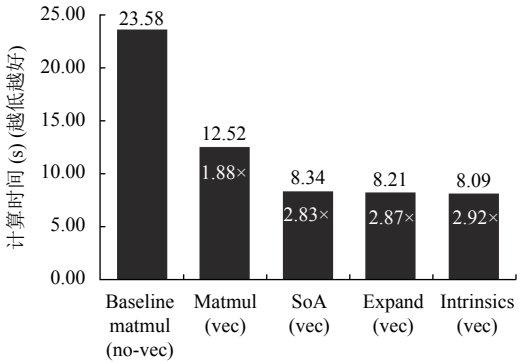

图 11 16MPI + 16OpenMP 各向量化优化测试结果及加速比

结果显示, 几种向量化算法均优于 Fortran 内置的 matmul 函数的效果, 而且都取得了 3 倍左右的加速效果. 其中基于 Intel Intrinsics 的手动向量化算法效果最好, 达到了 3.17 倍的加速.

4.4 输出组态文件优化测试

针对 3.4 节提出的组态输出优化方案, 我们测试了未改变数据顺序的 MPI 并行输出以及修改数据顺序后的 MPI 并行输出算法, 以单个 MPI 进程输出组态文件的结果为基准, 具体实验数据如下:

表 4 MPI 并行写组态文件测试结果

格子大小	串行写 (秒)	并行写 (秒)	加速比 (串行/并行)
$32^3\times32$	85.11	1.03	82.63
$32^3\times64$	143.19	0.93	153.97
$32^3\times128$	319.44	2.23	143.25

表 4 中“串行写”的数据为测试的基准, 并行写”为修改了数据顺序后的并行写算法测试结果, “加速比”为其对应的加速比. 通过表 4 我们可以看出, 优化了数据顺序的并行写算法取得了上百倍的加速, 最高达到了 153 倍.

5 总结与未来工作

格点量子色动力学一直是高性能计算领域的热门应用方向, 多年来一直伴随着超级计算机的发展, 对高性能计算的发展有强有力的驱动力. 本文基于典型的格点 QCD 胶球模拟计算程序, 采用了多项优化技术, 对程序进行了性能分析和并行优化, 并实现了大规模的并行测试, 并行规模超过 4 万核. 性能方面, 加入 OpenMP 多线程并行部分, 最高取得了 3.32 倍的加速比, 在向量化优化矩阵计算部分, 最高取得了 3.17 倍的加速比, 在并行输出组态文件部分取得了 153 倍的加速比.

优化后的程序可更有效地产生大量组态, 有利于得到更精确的胶球谱. 优化后的程序可较容易推广到夸克与胶子组成的混杂态的研究, 及普通介子和胶球的混合问题的研究, 为高能物理的相关研究提供更加便利的软件工具.

致谢

感谢中国科学院高能物理研究所理论物理室陈莹老师参与讨论和提供帮助, 感谢广州“天河 2 号”超算工作人员提供的测试支持.

参考文献

1 Wilson KG. Confinement of quarks. Physical Review D, 1974, 10: 2445–2459. [doi: 10.1103/PhysRevD.10.2445]
2 Five Gordon bell finalists credit summit for vanguard computational science. https://www.olcf.ornl.gov/2018/09/17/uncharted-territory/, 2018-09-17.
3 Brower R, Christ N, DeTar C, *et al*. Lattice QCD application development within the US DOE exascale computing project. Proceedings of the 35th International Symposium on Lattice Field Theory. Granada, Spain. 2018. 09010.
4 433.milc (su3imp), SPEC CPU2006 benchmark description. https://www.spec.org/cpu2006/Docs/433.milc.html.
5 QUDA: A library for QCD on GPUs. http://lattice.github.io/quda/.
6 Smelyanskiy M, Vaidyanathan K, Choi J, *et al*. High-performance lattice QCD for multi-core based parallel systems using a cache-friendly hybrid threaded-MPI approach. Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. Seatle, WA, USA. 2011. 1–10.
7 Kanamori I, Matsufuru H. Practical implementation of lattice QCD simulation on Intel Xeon Phi knights landing. arXiv: 1712.01505, 2017.
8 Lattice QCD code bridge ++. http://bridge.kek.jp/Lattice-code/.
9 Chen Y, Alexandru A, Dong SJ, *et al*. Glueball spectrum and matrix elements on anisotropic lattices. Physical Review D, 2006, 73(1): 014516. [doi: 10.1103/PhysRevD.73.014516]
10 Gui LC, Chen Y, Li G, *et al*. Scalar glueball in radiative J/ψ decay on the lattice. Physical Review Letters, 2013, 110(2): 021601. [doi: 10.1103/PhysRevLett.110.021601]
11 Yang YB, Gui LC, Chen Y, *et al*. Lattice study of radiative J/ψ decay to a tensor glueball. Physical Review Letters, 2013, 111(9): 091601. [doi: 10.1103/PhysRevLett.111.091601]