计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(7):199−204 [doi: 10.15888/j.cnki.csa.007498]

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 Grid 数值模拟软件的并行计算特征分析①

毕玉江[1], 周　超[2,3], 吴郁非[4], 黎睿翔[4], 刘朝峰[1,3], 陈建海[4], 徐　顺[2]

[1](中国科学院 高能物理研究所, 北京 100049)
[2](中国科学院 计算机网络信息中心, 北京 100190)
[3](中国科学院大学, 北京 100049)
[4](浙江大学, 杭州 310058)
通讯作者: 徐　顺, E-mail: xushun@sccas.cn

摘　要: 格点量子色动力学 (QCD) 是从第一原理出发求解 QCD 的非微扰方法, 通过在超立方格子上模拟胶子场和费米子场相互作用, 其计算结果被认为是对强相互作用现象的可靠描述, 格点计算对 QCD 理论研究意义重大. 但是, 格点 QCD 计算具有非常大的计算自由度导致计算效率难以提升, 通常对格子体系采用区域分解的方法实现并行计算的可扩展性, 但如何提升数据并行计算效率仍然是核心问题. 本文以格点 QCD 典型软件 Grid 为例, 研究格点 QCD 计算中的数据并行计算模式, 围绕格点 QCD 中的复杂张量计算和提升大规模并行计算效率的问题, 开展格点 QCD 方法中数据并行计算特征的理论分析, 之后针对 Grid 软件的 SIMD 和 OpenMP 等具体数据并行计算方式进行性能测试分析, 最后阐述数据并行计算模式对格点 QCD 计算应用的重要意义.

关键词: 数据并行; SIMD; 并行计算; 格点 QCD

引用格式: 毕玉江,周超,吴郁非,黎睿翔,刘朝峰,陈建海,徐顺.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. http://www.c-s-a.org.cn/1003-3254/7498.html

Parallel Computing Feature Analysis of Grid Numerical Simulation Software for Lattice Quantum Chromodynamics

BI Yu-Jiang[1], ZHOU Chao[2,3], WU Yu-Fei[4], LI Rui-Xiang[4], LIU Zhao-Feng[1,3], CHEN Jian-Hai[4], XU Shun[2]

[1](Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China)
[2](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)
[3](University of Chinese Academy of Sciences, Beijing 100049, China)
[4](Zhejiang University, Hangzhou 310058, China)

Abstract: Lattice Quantum Chromodynamics (QCD) is a non-perturbative method for solving QCD based on the first principles. By simulating the interaction between gluon field and fermion field on super-cubic lattice, the calculated results are considered to be a reliable description of the phenomenon of strong interaction. Lattice calculation is of great significance to the study of QCD theory. However, the lattice QCD computing has a very large degree of freedom, which makes it difficult to improve the computational efficiency. Usually, the domain decomposition method is used to realize the scalability of parallel computing, but how to improve the efficiency of data parallel computing is still the core problem. In this work, taking Grid, a typical lattice QCD software, as an example, the data parallel computing pattern in lattice QCD computing is studied. Focusing on the complex tensor computing in lattice QCD and improving the efficiency

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); 中国科学院“十三五”信息化建设专项 (XXH13506-404)
Foundation item: National Key Research and Development Program of China (2017YFB0203202); General Program of National Natural Science Foundation of China (11575197); CAS Special Fund for Informatization Construction in 13th Five-Year Plan (XXH13506-404)
收稿时间: 2019-12-12; 修改时间: 2020-01-03; 采用时间: 2020-01-14; csa 在线出版时间: 2020-07-03

of large-scale parallel computing, the theoretical analysis of data parallel computing features in lattice QCD method is carried out, and then the performance test and analysis are carried out for the specific data parallel computing methods such as SIMD and OpenMP of Grid software. Finally, the significance of data parallel computing pattern to the application of lattice QCD computing is explained.

Key words: data parallelism; SIMD; parallel computing; lattice QCD

在微观粒子世界中起主要作用的是强相互作用、电磁相互作用和弱相互作用, 描述强相互作用的量子色动力学 (QCD) 和描述电磁、弱相互作用的电弱统一理论 (EW) 统称为粒子物理中的标准模型. 就强相互作用而言, 其低能特性如夸克禁闭问题, 仍是需要解决的世纪难题. 实际上, 对于夸克禁闭以及与之密切相关的强子结构和强子性质等低能区强相互作用现象需要有效的非微扰方法. 格点 QCD[1]是目前公认有效的从第一原理出发的求解 QCD 的非微扰方法, 除标准模型基本参数之外没有任何额外模型参数, 因而其计算结果被认为是对强相互作用现象的可靠描述. 该格点 QCD 理论对不少重大物理研究有着不可替代的作用, 主要体现在用高能物理实验结果对现有理论的验证、用理论分析结果指导实验设计、以及进一步地寻找标准模型以外的新物理等方面.

格点 QCD 的研究手段是通过计算机进行大规模的矩阵运算和蒙特卡罗数值模拟, 其算法特点具有内在的高并行度和高可扩展性. 然而, 随着计算机硬件体系结构的日益复杂, 研发计算高效的格点 QCD 应用程序也变得愈加挑战性. 大规模格点 QCD 计算过程对高性能计算资源需求巨大, 其计算过程体现出计算密集型特征, 而其数据分析体现出数据密集型特征. 计算精度和数据分析效率直接反映了格点 QCD 计算方法的应用水平. 格点 QCD 已经逐渐成为与高能物理实验研究和理论解析研究并列的高能物理第三大分支, 研究范围已经拓展到强相互作用现象的各个方面. 美国、日本、欧洲等国家无一例外将格点 QCD 应用作为高性能计算的重要应用. 美国更是将其列为未来 E 级计算重要的应用之一, 凸显了格点 QCD 数值模拟内在的科学计算应用的价值[2]. 近年来, 格点 QCD 的数值模拟方法发展出了大量理论和算法, 形成了美国 USQCD[3]和 MILC[4]、英国 Grid[5]和日本 Bridge++软件[6]等诸多优秀的格点 QCD 模拟软件.

从格点 QCD 数值计算的特点来分析, 其主要过程为矩阵向量间的运算, 并行化时涉及格点体系的区域分解并行划分, 边界格子进程间通信还有组态存储分析等任务[7]. 如何实现高效的大规模格点 QCD 数值计算一直以来都是一个现实问题. 但我们研究发现, 格点 QCD 抽象分层软件架构的设计方法得到了重要应用, 软件框架的这种分层抽象的设计方法极大地提高了代码的开发效率和运行效率. 比如美国主导的 USQCD 软件框架分为 4 层: 最上层应用层包括 Chroma、CPS 和 MILC 等, 下面 3、4 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, 而其中的 QDP++支撑库是用来实现数据并行应用接口的, 提供逻辑上的数据并行操作, 隐藏底层数据依赖体系结构的操作, 提供上层统一的数据通信函数接口. 同样在这方面, 英国 Peter Boyle 等人 2015 年研发出了 Grid 软件[5], 它抽象出了一个数据并行计算模式, 针对格点 QCD 的热点计算抽象了统一的数据并行接口, 支持多种平台的 SIMD 矢量化计算, 如 AVX2、AVX512、NEON 等形式, 同时也支持组合 OpenMP 和 MPI 等多种并行计算编程模式. 在实现线程化、SIMD 操作可以实现比 QDP++更有效.

Grid 软件包近年来受到越来越多的应用, 利用它开展的物理研究包括 K 介子衰变中直接电荷共轭-宇称 (CP) 联合破坏参数的格点 QCD 计算[8], 这能帮助物理学家理解宇宙中观测到的正反物质不对称性. 文献[8]利用 Grid 软件改进了他们以前的物理测量代码, 性能得到显著提升. 另外核子结构的研究[9]以及超出标准模型新物理的研究[10]等也使用了 Grid 软件. 文献[11]中作者们初步尝试将 Grid 软件移植到 GPU 上, 他们比较了几种策略: OpenACC、OpenMP 4.5、Just-In-Time 编译和 CUDA 并讨论了各自的优缺点.

从现有工作来看, 格点 QCD 计算软件是以分层软件架构为主, 以数据并行计算模式为核心, 如何提升格点 QCD 中数据并行计算效率和传输效率成为关键点, 这是现有 USQCD 和 Grid 等软件包重要的研发目标.

本文基于格点 QCD 计算 Grid 软件包, 开展格点 QCD 数据并行计算的理论分析, 研究数据并行计算模式在实现上的关键问题, 分析典型应用软件 Grid 和 QDP++ 的软件架构实现, 最后结合具体的计算平台, 测试相关的应用软件性能, 针对实验数据开展分析, 验证相应的理论分析的有效性.

1 格点 QCD 模拟数据并行计算分析

1.1 格点 QCD 计算算法原理

格点 QCD 数值计算的基本思想为: 1) 将四维连续时空离散化为一个四维超立方时空格子 $L_x \times L_y \times L_z \times L_t$, 这样体系的自由度变为有限可数; 2) 在该时空格子上定义基本自由度夸克场和胶子场, 并在此基础上定义 QCD 的格点作用量形式; 3) 采用路径积分量子化的方式; 4) 物理观测量的测量类似统计物理中的系综平均值. 于是问题就转化为一个包含近邻相互作用 (因果律要求物理体系的相互作用必须是局域的)、多自由度体系的统计物理问题, 可以采用蒙特卡罗数值模拟方法对物理观测量进行计算. 格点 QCD 数值模拟计算的简单流程如下:

1) 按照 Boltzmann 分布因子进行蒙特卡罗随机抽样, 产生 QCD 物质状态系综 (规范场组态), 这个过程中会涉及大量的线性系统求解 **Dx** = **b** 来用于组态更新, 其中涉及到维数达到 $L_x \times L_y \times L_z \times L_t \times 4 \times 3$ (一般超过千万数量级) 的稀疏矩阵 $D^{f,ab}_{\alpha\beta}(n|m)$ (以典型的威尔逊格点作用量为例):

$$D^{f,ab}_{\alpha\beta}(n|m) = \left(m^{(f)} + \frac{4}{a}\right)\delta_{\alpha\beta}\delta_{ab}\delta_{n,m} - \frac{1}{2a}\sum_{\mu=\pm1}^{\pm4}\left(\mathbf{1}-\gamma_\mu\right)_{\alpha\beta}U_\mu(n)_{ab}\delta_{n+\hat{\mu},m} \quad (1)$$

这里 n 和 m 的取值为整个格子 $L_x \times L_y \times L_z \times L_t$, α 和 β 为夸克自旋指标分别有 4 个取值, 下标 a 和 b 为夸克的颜色自由度分别有 3 个取值.

2) 物理量测量, 这个过程涉及在第 1) 步生成的每一个组态上做大量的线性系统求解 **Dx** = **b**. 与 1) 的不同在于向量 **b** 的设置不一样以及解向量 **x** 这里是用于物理量的计算.

3) 对物理量做组态平均, 通过数据拟合与分析获得物理结果.

总体而言, 格点 QCD 的主要计算热点就是求解大规模线性方程组**Dx** = **b**. 人们一般采用 CG、BiCGstab、MinRes、GCR 等迭代算法, 并且配合 Even-odd、Multi-grid 等适当的预处理算法, 在{**b**, **Db**, **D2b**, **D3b**,…} 张开的 Krylov 子空间里迭代求解所需的大规模线性方程组**Dx** = **b**. 在求解过程中, 这些算法会涉及大规模数据处理操作, 因而“矩阵 **D** 乘以向量”操作是计算热点的热点. 另外, 在超大规模计算中, “计算两个向量内积”由于涉及到全局归约, 也会产生额外开销.

1.2 格点 QCD 计算的数据并行计算特征

格点 QCD 数值模拟时, 四维超立方格子上的点被分配到多个处理核做并行计算, 三维中的示意图如图 1 所示.

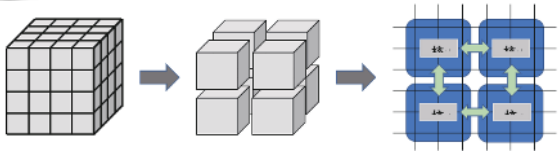

图 1 格点 QCD 计算中的空间分解并行划分方法

由于 QCD 具有近邻相互作用特征, 每个处理器核心在涉及小立方边界处就会向其他相关核心发起通信以获得相邻立方边界的数据. 因此切分格子的方法会对计算效率产生明显的影响, 需考虑计算与通信的平衡.

稀疏矩阵 $D^{f,ab}_{\alpha\beta}(n|m)$ 中包含的 **U** 矩阵为颜色空间 3×3 的复矩阵, **U** 和颜色空间向量的乘积涉及到矩阵元相乘然后求和. 单个这样的矩阵乘向量操作通常不适合直接做向量化, 因为矩阵元的个数和寄存器的宽度通常不会恰好配合, 而且最后的求和操作无法向量化. 在 Grid 软件中, 多个这样的矩阵乘向量操作被组合在一起 (数目根据硬件而定), 不同矩阵的矩阵元与向量元的乘积交织在一块并行, 之后的多个求和也可以同时进行, 从而实现高效的向量化计算.

1.3 数据并行计算的典型设计

Grid 最重要也是其突出的特点就体现在数据处理的矢量化设计上. 前面提到过, 在格点 QCD 计算中, 为了能够适应多节点并行计算, 通常是将格子在时空维度进行划分, 然后在单个节点上进行各种矩阵的计算, 这是很自然的想法. 在当时处理器没有向量化或向量化能力比较弱的情况下, 这种处理方法没有什么问题的. 早期的格点 QCD 程序如 QDP++ 在考虑处理器矢

量化时, 一般考虑的就是在单个矩阵计算时的矢量化 (如图 2, 其中等号左边列向量中实点的值等于等号右边矩阵中实点所在行与列向量的内积 (图引自文献[5]), 其并行方法是在单个矩阵计算内, 将多个元素的内积及规约进行并行), 这需要 MPI 进程之间进行很多的归约, 形成通讯上的瓶颈, 对性能造成影响.

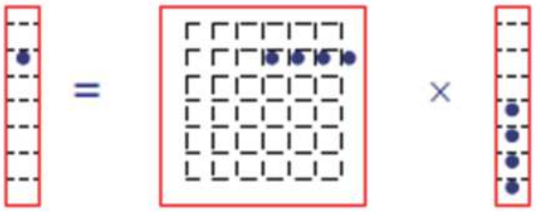

图 2 QDP++矩阵向量乘法示意图

但是数据的划分方式除了时空划分这种方式, 还可以通过每个格点上的自旋和洛仑兹指标进行划分, Grid 在设计时, 就部分采用了这种划分方式, 采用了与矩阵内矢量化不同的方式, 如图 3, 其考虑两个或 4 个矩阵一起对每个元素进行计算, 在 SIMD 线上, 没有 MPI 归约, 减少了通讯开销. 图 3 中, SIMD 可实现多个矩阵向量乘法并行操作, 多个向量对等位上的数存储在一个 SIMD 向量去计算, 避免了 SIMD 位长限制和规约操作 (图 3 引自文献[5]).

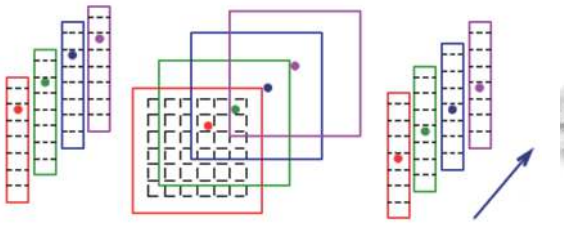

图 3 Grid 矩阵向量乘法的 SIMD 实现示意图

这种方式很适合格点 QCD 计算, 考虑到 AVX512 有 8 个 double 宽度, 由于要计算的矩阵有 4 维旋量指标, 自然就适合 4 个矩阵一起进行计算, 而且矢量化也非常简便. Grid 软件中采用的这种 SIMD 计算方式充分发挥了数据并行计算的潜在能力. 此外, 这种 SIMD 矢量化方式对各种数值运算都进行了矢量化抽象, 只需要完成相应的数据矢量化代码, Grid 就能够很容易地扩展到新的硬件架构中. 我们在开发格点 QCD 代码时, 也可以借鉴 Grid 优秀的设计思想.

2 实验结果与分析

2.1 AVX512 与 AVX2 性能对比

选择 Grid 软件 Wilson 实例作为测试对象, 我们比较了在 Intel Xeon Gold 5120 机器在采用 AVX2 和 AVX512 向量化版本的性能差异, 性能结果数据如表 1 和图 4.

表 1 AVX512 与 AVX2 性能对比

Threads	AVX512(MFlop/S)	AVX2(MFlop/S)	AVX512/AVX2
1	5662.76	4289.08	1.34
2	11 196.2	8634.17	1.29
4	18 347	15 207.4	1.21
8	25 396.8	21 470.6	1.18
14	30 105.2	26 470.8	1.14

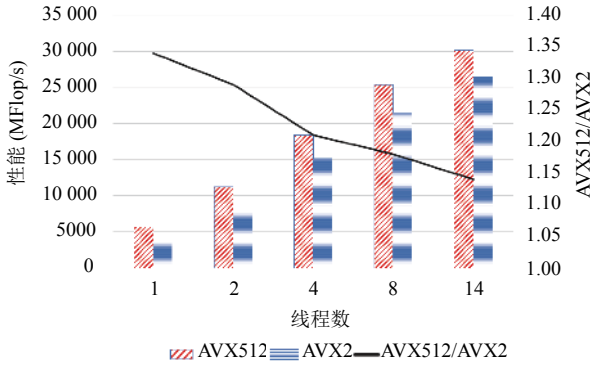

图 4 Intel Xeon Gold 5120 上 AVX512 与 AVX2 性能加速对比图

从图 4 中, 我们可以看出浮点计算性能在 AVX512 版本下优于 AVX2, 但随着线程数 (Threads) 增加, 越来越接近总核数 (28), 会发现其性能对比差异开始下降.

2.2 OpenMP 共享内存测试

接下来分析线程绑定对数据并行计算的影响. 我们选择在 2 路 2 核的 Intel Xeon Silver 4114 机器上做线程绑定测试, 采用 Intel 编译器来编译 Grid 软件, 通过设置环境变量 KMP_AFFINITY 实现线程绑定, 主要分为两种模式, 即 compcat 和 scatter. compact 模式下, 线程将优先集中在一个 CPU 上进行计算; scatter 模式下, 线程将均匀分布在多个 CPU 上进行计算, 访存变大.

具体测试方案: 分别在 scatter 模式下和 compact

模式下进行线程绑定的测试, 分别测试线程数为 1, 2, 3, 4, 8, 10, 12, 16, 20, 30, 32, 36 和 40 下的计算性能, 得到结果如表 2, 同时图示化对比结果如图 5 所示.

表 2 线程绑定 scatter 与 compact 模式下性能对比

Threads	KMP=scatter (MFlop/S)	KMP=compact (MFlop/S)	scatter/compact
1	2962.43	2927.32	1.01
2	5937.85	3556.29	1.67
3	8526.33	5288.4	1.61
4	11 198.6	6982.85	1.60
8	21 556.4	13 645.8	1.58
10	26 732.9	15 727.5	1.70
12	29 369.8	18 851.6	1.56
16	36 903.7	22 005.6	1.68
20	35 597.5	23 387	1.52
30	37 090.3	35 249.2	1.05
32	39 737	38 576.5	1.03
36	42 006.9	42 755.8	0.98
40	20 710.2	20 947.3	0.99

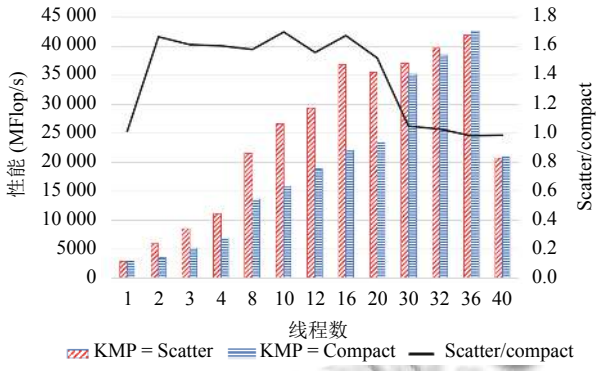

图 5 不同线程绑定方式性能对比图.

由于程序存在大量的 cache 访问, 在线程数分别为 2, 3, 4, 8, 10, 12, 16 和 20 的情况下, 线程绑定导致共享相同 cache, 造成冲突, 竞争使用 cache, 造成 cache 频繁读写, 降低访存速度, 相比分散线程, 性能要差. 如果应用程序需要大量的访存操作, 需要大量的 cache, 将 OpenMP 线程绑定核时, 尽量将线程分布在每个 CPU, 可以很好的提升性能, 在本次测试中, 性能较 compact 下最高提升 1.7 倍. 如果程序的每个线程需要共享大量数据, 则将所有线程绑定在一个 CPU 时, 性能更好.

2.3 ARM 架构向量化性能对比

如今越来越多的服务器开始采用 ARM 架构的处理器, 比如华为的鲲鹏处理器, 亚马逊的 Graviton 处理器, 飞腾的 FT 系列处理器等等, “天河三号”原型系统便使用了 64 位 ARM 架构的 64 核心处理器 FT-2000plus 和 Matrix-2000 加速卡, 其与 Intel 的 Xeon 架构类似. ARM 架构的 CPU 向量化操作 (asimd) 有 NEON 技术. 我们在“天河三号”原型系统上, 测试了向量化前后 Grid 的性能对比情况.

GCC 编译器可以通过-march 选项设置相关参数来启用向量化编译. 首先我们测试在没有开启向量化编译情况下 Grid 的性能. 在每个节点上, 我们使用 1 个 MPI 进程和 64 个 OpenMP 线程进行测试, 每个节点负责的子格子大小是固定的 16^4. 我们记录程序的计算性能和计算时间, 如表 3 所示, 其中, 第 2 列和第 3 列是计算 Dslash 和 Wilson fermion 时单个节点的总计算性能 (Dslash 是格点 QCD 中的最重要的算符, Wilson fermion 是格点 QCD 中的一种费米子), 第 4 列和第 5 列则是计算所消耗的总时间和通讯时间.

表 3 ARM 平台未开启矢量化优化测试

Node	Deo (MFlops)	Wilson (MFlops)	Time (ms)	Comm (ms)
16	3650.23	3716.88	189.579	75.325
32	4048.12	4122.05	170.945	77.6159
64	3627.63	3694.71	190.717	97.3861
128	3623.16	3689.66	190.978	90.566
256	3478.77	3543.19	198.872	110.190
512	3414.39	3476.81	202669	97.898

在开启向量化 (asimd) 后, Grid 的性能测试结果如表 4, 各列含义与表 3 相同.

表 4 ARM 平台下 Grid 开启 ASIMD 向量化测试结果

Node	Deo (MFlops)	Wilson (MFlops)	Time (ms)	Comm (ms)
32	4241.14	4318.61	163.164	63.9075
64	4254.38	4331.62	162.674	57.532
128	4276.96	4355.18	161.794	65.110
256	4030.78	4103.99	171.697	69.654

作为对比, 结合表 3 和表 4, 在图 6 中画出了表示性能的 Wilson 列数值随节点数变化的曲线图, 其中 w/o asmid 代表没有开启 asimd; 而 w asimd 表示开启了 asimd.

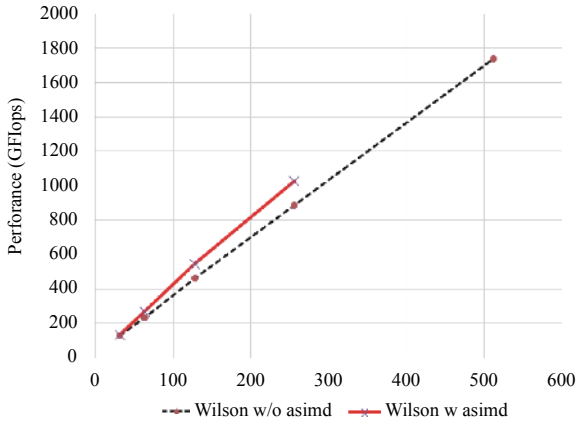

图 6 Grid 弱可扩展性测试

从图 6 可以看出, 一方面, 随着节点数增加, 单个节点计算能力慢慢下降, 但总计算能力呈线性增加, 这表明 Grid 有很好的弱扩展性; 另一方面, 目前我们只是采取了基本的自动化向量化编译, 可以看出矢量化后程序总体计算性能有一定提升, 而且加入矢量化之后, 同样具有非常好的弱扩展性, 可以预测在更大节点数时, 也能能够保持较好的计算效率.

矢量化前后的计算时间、通讯时间和总时间随节点数的变化情况如图 7 所示.

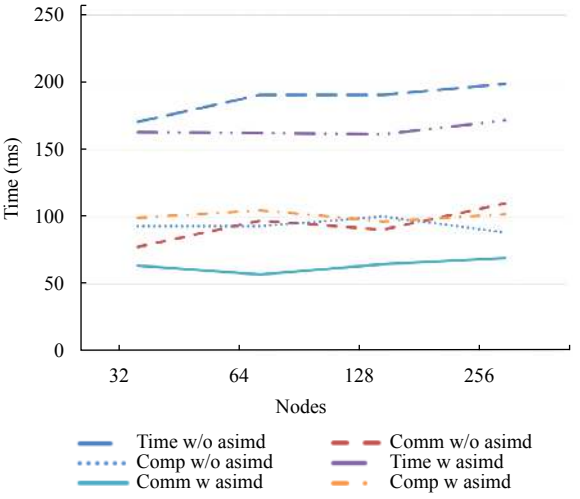

图 7 ARM 下 Grid 开启 ASIMD 向量化前后 Wilson 算符的计算、通讯时间和总时间对比图.

可以看到的是, 两种情况下计算时间基本一样, 减少明显的是通讯时间. 这是可以理解的, 在开启向量化之前, MPI 间进行同步和归约时会有瓶颈, 每次迭代计算时有大量的同步; 而在向量化后, MPI 会一次同步和归约多个元素, 因此同步和归约的次数会大大减少, 这一部分不再是瓶颈, 因此时间也相对减少了很多.

3 结论与展望

本文围绕格点 QCD 计算中采用数据并行模型提升上层应用计算效率的关键问题, 开展了计算算法的理论分析, 针对典型应用软件 Grid 开展了面向数据并行计算模型的软件架构分析, 并且结合具体计算平台, 测试了 Grid 应用软件的性能, 包括向量化特征和大规模可扩展性分析. 最终阐述了数据并行计算模式, 对格点 QCD 计算的有效性和重要性.

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 Brower R, Christ N, DeTar C, *et al*. Lattice QCD application development within the US DOE Exascale computing project. EPJ Web of Conferences, 2018, 175: 09010. [doi: 10.1051/epjconf/201817509010]
3 The USQCD collaboration. https://www.usqcd.org/collaboration.html. [2019-12-06].
4 MILC code. http://www.physics.utah.edu/~detar/milc/milc_qcd.html. [2019-12-06].
5 Boyle P, Yamaguchi A, Cossu G, *et al*. Grid: A next generation data parallel C++ QCD library. arXiv: 1512.03487, 2015.
6 Lattice QCD code bridge ++. http://bridge.kek.jp/Lattice-code/index_e.html. [2019-12-06].
7 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]
8 Kelly C. Progress in the calculation of epsilon on the lattice. Proceedings of the 34th annual International Symposium on Lattice Field Theory. UK. 2016. 308.
9 Green J, Jansen K, Steffens F. Nonperturbative renormalization of nonlocal quark bilinears for parton quasidistribution functions on the lattice using an auxiliary field. Physical Review Letters, 2018, 121(3): 022004.
10 Witzel O, Hasenfratz A, Rebbi C. Composite Higgs from mass-split models. arXiv: 1810.01850, 2018.
11 Boyle PA, Clark MA, DeTar C, *et al*. Performance portability strategies for Grid C++ expression templates. EPJ Web of Conferences, 2018, 175: 09006. [doi: 10.1051/epjconf/201817509006]