Computer Science and Application 计算机科学与应用, 2021, 11(1), 84-94
Published Online January 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.111010

Hans汉斯

OpenMPI环境下的VASP软件的并行与进程

潘　勇[1]，史海洪[2]，赵琉涛[1]，刘　彤[1]

[1]北京市计算中心，北京
[2]北京北科融智云计算科技有限公司，北京
Email: shihh@bcc.ac.cn, zhaolt@bcc.ac.cn, panyong@bcc.ac.cn, liutong@bcc.ac.cn

收稿日期：2020年12月24日；录用日期：2021年1月18日；发布日期：2021年1月26日

摘　要

VASP是材料微观物性模拟的重要软件，并行性能优良，其中矩阵对角化在其运算过程中占据大量时间。在HPC作业调度下，VASP并行的进程分配表现单节点和跨节点的差异。通过对MPI并行的节点内和节点间进程、EDDIAG和RMM-DIIS运行时间分析，明确了VASP程序并行中，矩阵对角化的重要性和OpenMPI对矩阵对角化并行的作用。并编写了一个通讯优化的矩阵对角化程序，通过测试并行程序托比串行的马斯算法表现出强大的可扩展性，基于OpenMPI的并行程序能够极大提高对角化的并行效率，但其并行度、通讯代价与计算规模之间的关系需要进一步研究。

关键词

VASP，矩阵对角化，MPI，跨节点并行

The Parallel and Progress of VASP in OpenMPI Environment

Yong Pan[1], Haihong Shi[2], Liutao Zhao[1], Tong Liu[1]

[1]Beijing Computing Center, Beijing
[2]Beijing Beike Rongzhi Cloud Computing Science and Technology Ltd., Beijing
Email: shihh@bcc.ac.cn, zhaolt@bcc.ac.cn, panyong@bcc.ac.cn, liutong@bcc.ac.cn

Received: Dec. 24th, 2020; accepted: Jan. 18th, 2021; published: Jan. 26th, 2021

Abstract

VASP is an important software for material microscopic physical properties simulation, with excellent parallel performance, among which matrix diagonalization takes up a lot of time in its cal-

文章引用: 潘勇, 史海洪, 赵琉涛, 刘彤. OpenMPI 环境下的 VASP 软件的并行与进程[J]. 计算机科学与应用, 2021, 11(1): 84-94. DOI: 10.12677/csa.2021.111010

culation process. Under HPC job scheduling, the parallel process allocation of VASP shows the difference between the single node and across multiple nodes. Through the analysis of the intra-node and inter-node processes of MPI parallel, the running time of EDDIAG and RMM-DIIS, we clarified the importance of matrix diagonalization in VASP program parallelism and the role of OpenMPI on matrix diagonalization parallelism. Furthermore, we write a communication-optimized matrix diagonalization program. By test, the parallel program using the Thomas algorithm shows strong scalability than the serial algorithm. The OpenMPI-based parallel program can significantly improve diagonalization's parallel efficiency, but the relationship between its parallelism, communication cost, and computing scale needs further study.

Keywords

VASP, Matrix Diagonalization, MPI, Inter-Node Parallelism

1. 简介

近年来，材料计算与模拟在新材料研发和材料物性分析中发挥越来越重要的作用，特别是基于量子力学的第一原理(ab initio)计算，构成材料计算与模拟的微观基础[1]-[8]，由于实验技术和条件的限制，也是对计算需求特别大的阶段。描述微观粒子运动需要求解偏微分方程，对计算能力提出了巨大的挑战。随着高性能计算的能力提升，材料计算软件的并行性能也不断提高，出现了越来越多的适应高性能计算的第一原理并行计算软件，如 VASP (Vienna Ab-initio Simulation Package) [9]、QE (Quantum Espresso) [10]和 PWmat [11]等。考虑到构成宏观物质的微观粒子的数量高达 1023 量级，现有的高性能计算能力面对材料设计的计算需求，还有不小的距离。

第一原理材料计算软件是基于密度泛函理论(Density Functional Theory) [12] [13]和周期势条件下电子运动的偏微分方程求解器。计算过程中主要涉及的高性能计算包括：1) 周期势条件下的平面波基组和 FFT 变换所需的网格分布；2) 矩阵对角化求解 DFT 基态方程本征值的迭代优化算法。以 VASP 为例，在网格划分方面，采取了“双网格技术(dual grid)”[14]，即在倒空间用粗网格计算基础上，将网格加密，然后实施 FFT 到正空间(密网格)，计算有关物理量后，变换回到倒空间(密网格)，最后得到粗网格上的物理量，确保计算精度。而矩阵迭代优化算法则采用了 Davison 方法、共轭梯度(CG)和 RMM-DIIS [15] [16]等多种优化算法实现。

对称特征值问题作为数值计算的重要问题，存在于很多学科领域中，尤其是材料计算第一性原理，该问题的求解中，矩阵三对角化和三对角矩阵特征值的求解是两个重要的步骤，鉴于问题规模问题，本文中使用分治策略和 OpenMPI 进行矩阵并行求解，并测试了在分布式模式下的并行求解效率。

本文按以下顺序组织：第二部分介绍 HPC 并行环境，OpenMPI 和 LSF 集群管理软件，第三部分介绍 VASP 在 HPC 作业下的调度、跨节点并行测试和结果进行分析，分别从 VASP 并行、MPI 的体系结构和作业管理系统的资源分配讨论；第四部分为矩阵对角化程序并行算法及测试效果；最后为本文结论。

2. 使用须知

作为以理论计算为基础的计算材料科学来说，计算研究是第一要务。通过模拟计算，可以在不实际制备的前提下深入理解材料的微观结构性质，从而节约研发成本；同时还可以模拟特殊/极限环境下的实

验结果，比如对人体健康有害，或者处在压、超低温、强磁场等某些极端条件下，实验测量很难实现或者耗费巨大，可以使用模拟计算来代替或指导实验。

随着 HPC 能力飞速提高，材料科学应用领域大幅度扩展，更大尺度的体系、更长时间的动力学演化、更精确的理论计算描述成为可能。高性能计算在材料科学中的应用也更加广泛。高性能计算给材料科学研究带来极大便利的同时，材料科学的发展也推动了高性能计算的进步。

2.1. OpenMPI 的体系结构

适用于机群、MPP 等分布式内存结构的并行编程环境，通常由“并行虚拟机”(Parallel Virtual Machine, PVM)或“消息传递接口”(Message Passing Interface, MPI)来实现。利用 PVM 工具，可以把互连的各种计算机虚拟为一台并行机，从而为编程人员提供了一个便于管理和使用的编程环境，而由 PVM 的编译库对程序进行转换，将程序的计算任务分解为若干子任务后合理分配到各个节点机进行并行处理。MPI 是一种基于消息传递的并行计算规范，消息(Message)一般包括数据、指令或其它各种控制信号等，MPI 提供了一套消息传递库，基于消息传递的并行编程实际上就是通过调用 MPI 的消息传递库函数实现节点机之间的数据交换，并提供并行处理任务之间的同步等。目前，基于 PVM 和 MPI 并行编程环境，都可以支持 C、C++和 FORTRAN 等的并行编程。

Open MPI 的基于构件的体系结构不仅为第三方研究提供了稳定的平台，也使得独立软件附件能够在运行时组合。Open MPI 的运行时环境提供了启动和管理并行应用的基本服务。Open MPI 的设计以 MPI 构件架构(MPI Component Architecture, MCA)为中心，为所有其它层次提供管理服务的基础组件结构。对 Open MPI 的架构分析表明，OpenMPI 并不会开启额外进程，因此跨节点时的新增进程的唯一来源是作业管理系统对计算资源的分配与监控过程。

2.2. 作业管理系统的资源分配与监控

高性能计算是以大型集群和计算队列为硬件基础。作业管理系统包括集群管理和作业管理两个方面，集群管理系统将一组独立的计算机组合成资源池，共同协作以完成任务。集群广泛应用于高性能计算领域，提供低成本，可扩展及高性能的计算能力，作业管理系统是构成集群的重要软件系统，它的主要任务是对集群的资源进行集中的监控和管理，为用户提交的任务分配可用的计算资源，并监控和管理作业的执行及结果的返回；同时，还提供系统容错和错误恢复能力，可以在异常或故障发生时最大程度的减少任务中断的代价。

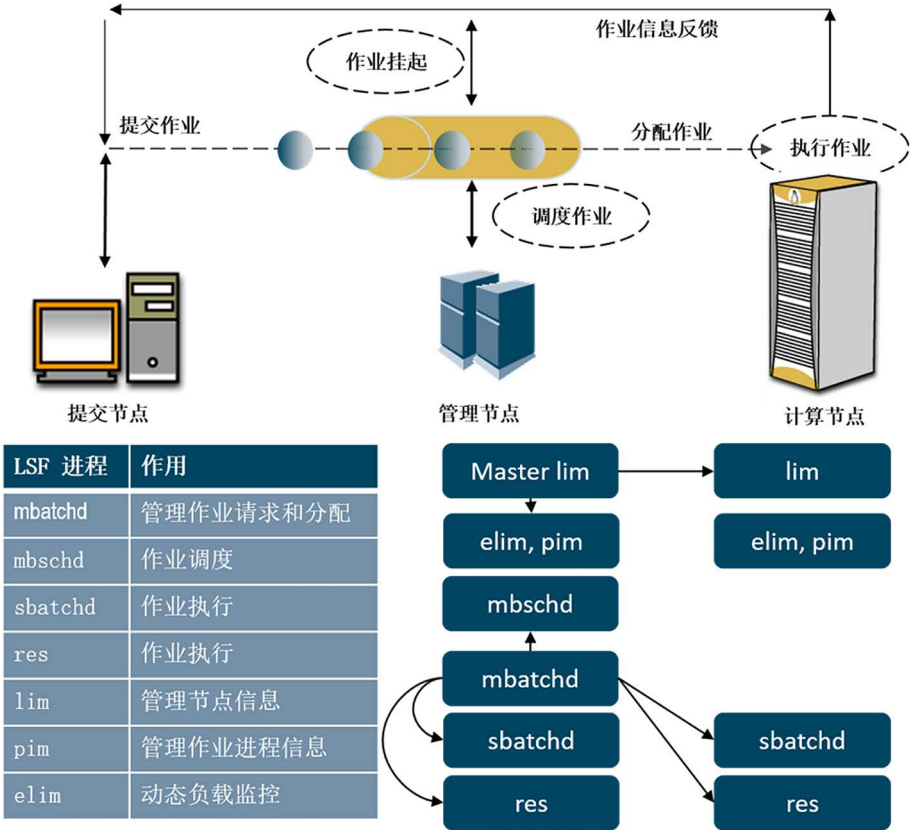
随着数据中心在规模和复杂性上的快速增加，使得对集群工作负载和应用的管理更加困难，同时也难以确保计算硬件和软件等资源的正常使用。用户希望在任何地方都能灵活使用应用程序，并自动操控数据流，管理者希望能够监督集群的资源和负载，管理软件使用权，确定瓶颈问题，监督面向用户的服务协议是否满足，并计划系统的扩容数量。

北京市计算中心采用 IBM Spectrum LSF (Load Sharing Facility)平台系列软件管理集群和作业。对于要求高的分布式任务型高性能计算环境，该软件能够通过综合的基于智能的，策略驱动的调度策略高效地管理负载，方便客户使用所有计算基础设施资源，保障最佳的应用性能。软件将集群内服务器分为两类，分别是管理节点和计算节点。管理节点负责监控集群中所有节点状态，并分配任务到计算节点，计算节点负责运行用户提交的任务，图 1 是 LSF 平台在集群中的系统环境下的任务提交执行过程。

作业管理系统展示了任务提交的完整过程：

1) 作业提交

用户通过 LSF 客户端，或者可以执行 bsub 命令的服务器上提交一个作业，当提交这份作业时，如果不指定哪个队列，这份作业就会被提交到系统默认的队列中，作业在队列中等待安排，这些作业处于等待状态。



LSF 进程	作用
mbatchd	管理作业请求和分配
mbschd	作业调度
sbatchd	作业执行
res	作业执行
lim	管理节点信息
pim	管理作业进程信息
elim	动态负载监控

Figure 1. The submit progress when using LSF in Cluster environment
图 1. LSF 平台在集群中的系统环境下的任务提交执行过程

2) 作业调度

后台的主进程 mbatchd 将处理队列中的作业，在一个预定的时间间隔里将这些作业按设定的计划，传递给主调度进程 mbschd。

主调度进程 mbschd 评估这份工作时，根据作业的优先权制定调度决策、调度机制和可利用资源。主调度进程选择可以运行作业的最佳计算节点，并将它的决策返回给后台主进程 mbatchd。主负载信息管理进程(lim)收集资源信息，主 lim 与 mbatchd 主进程交流这些信息，反过来 mbatchd 主进程使用之前交流信息支持调度决定。

3) 作业分配

mbatchd 主进程收到 mbschd 发过来的决定后，立即分配作业到计算节点。

4) 作业运行

从批处理进程(sbatchd)，从 mbatchd 主进程接到要求，为作业创建子 sbatchd 和一个执行环境，通过使用一个远程执行服务器开始这个作业。对于使用 MPI 执行的作业，LSF 会在 MPI 和应用中间创建一个 TaskStarter 进程，用于管理和监控每个作业的执行。

5) 返回输出

当一个作业完成时，如果这个作业没有出现任何问题，它处于一个完成状态。如果有错误作业无法完成，这份作业处于退出状态。sbatchd 传达作业信息，包括错误提示和给 mbatchd 的输出信息。

6) 信息反馈

mbatchd 通过预定方式给提交者反馈作业输出信息、作业错误、提示信息、作业信息。

通过上述作业提交和管理、监控过程，不难看出，在跨节点计算任务中，作业管理系统将在 MPI 通信启动时，在计算节点上开启两类进程，一类用于监督作业运行状况(节点数−1)*CPU 数，另一类是管理进程，即检查跨节点计算资源的运行和稳定状况。因此是(节点数−1)。如果计算中的资源不跨节点，作业管理系统启动 MPI 的进程同时充当监督和守护进程，因此作业管理系统没有启动额外的进程。

3. VASP 并行效果测试

VASP 是维也纳大学开发 PAW 方法[17] [18]的第一原理材料物性模拟的主要软件，VASP 的 MPI 并行效率较高。在北京市计算中心的服务器的计算环境为 Intel® Xeon (R) CPU E5-2680 V3 2.50 GHz，内存 DDR4 64 G，2133 MHz，数据网络连接为 Infiniband 56 Gb/s。每个节点有 24 核，每核 4 线程，通过作业管理系统，分别分配的计算资源包括单节点 8 核、16 核、24 核和跨节点 24 核(N = 2)、48 核(N = 3)、72 核(N = 4)等几种情况，VASP 正常结束后，作业管理系统将统计并输出的最大进程数和线程数。由此确定 VASP 并行过程中节点、进程等资源的使用情况，为进一步优化计算资源作准备。

3.1. VASP 的并行实现

本文实验的运行环境中，VASP 的并行计算，是由 Open MPI 支持的。Open MPI 是在 LAM/MPI，LA-MPI 和 FT-MPI 的基础上的一种全新的基于构件概念的 MPI 实现，VASP 的 MPI 实现只关注高性能计算的部分方面，即倒空间布点的分配和能带并行快速求解的问题。Open MPI 并不是 LAM/MPI，LA-MPI 和 FT-MPI 的简单组合，而是一种全新的 MPI 实现，完全实现了 MPI-1.2 和 MPI-2 规约，并且完全支持并发和多线程应用(也就是 MPI_THREAD_MULTIPLE)。

VASP 运行过程中，并行计算部分主要集中在主要倒空间网格点的并行分配、k 点能带本征值求解(矩阵对角化)和 DFT 总能量计算这几部分。VASP 的主程序将计算资源(分配到的 CPU)按控制参数 NPAR 和 NCORE 分解(要求满足 NPAR*NCORE = NCPU)，在实际并行计算过程中，各部分对计算资源的利用方式不尽相同：

1) 倒空间网格点并行分配和 FFT 变换，在该阶段，NCPU 个计算资源是作为整理对待的，即空间划分的网格被均分到 NCPU 个进程上，形成网格点和 NCPU 个进程的拓扑映射，在此基础上，根据 NCPU 个进程的网络关系完成 MPI 支持的 FFT 变换，实现最高效的并发式 FFT 变换。

2) k 点能带本征值求解时，全部 CPU 分解为 NPAR 个组，每组 NCORE 个 CPU 上实现 MPI 通信和数据共享(通过共享内存)来求解一个能带的本征值，每次可同时求解 NPAR 个能带本征值。

3) DFT 总能量计算，与 k 点能带本征值求解时相似，将 NPAR 组计算的结果汇总，得到体系的总能量。只是简单的 MPI 进程。

3.2. VASP 并行测试

VASP 并行计算的节点、进程统计结果列于表 1，计算结果表明，单节点(N = 1)时，进程数即 CPU 数目，当出现跨节点(N > 1)计算时，进程数显著增加，并且进程数随着每个节点上的 CPU 数目增加呈现出有规律的变化，线程数则始终保持 CPU 数目的 4 倍递增。为讨论跨节点计算额外新增进程的可能起源。我们从 VASP 软件的并行实现、MPI 的体系结构和作业管理系统的资源分配和监控三个方面讨论计算过程中启动的进程数目。

从图 2 中可以看出总的计算时间(紫色线，单位小时)随着并行核数的增加和而降低。当全部核被占据时，并行效率明显降低(计算时间增加)。VASP 程序的并行执行主要是自洽迭代(SCF)中矩阵对角化，可以看出，每次自洽迭代的对角化过程 RM-DIIS 的耗时(蓝色线，单位秒)也是类似规律。在单机最大节点 24 核的情况下，并行核数达到 20 核时，RM-DIIS 达到并行峰值。

Table 1. The number of threads when using different number of nodes and CPUs
表 1. VASP 并行计算时不同节点产生进程和线程

N_{node}	N_{CPU}	$N_{Process}$	N_{Thread}	N_{node}
N = 1	8	8	32	N = 1
N = 1	16	16	64	N = 1
N = 1	24	24	96	N = 1
N = 2	24	25	96	N = 2
N = 2	48	49	192	N = 2
N = 3	48	66	192	N = 3
N = 4	72	111	288	N = 4
N = 4	96	147	384	N = 4

跨节点并行时(图 3)：满负荷(24 核)并行时，随着并行节点数的增加，单次自洽迭代的矩阵对角化时间逐渐降低，但降低的梯度逐渐变小(计算效率有所下降)。半负荷(12 核)并行时，计算效率比满负荷并行效率显著提高，由此可以推断，LSF 作业调度系统下，每个阶段满负荷运行计算效率不高，考虑到单节点并行的情况，建议 20 核作为跨节点并行的推荐值。

综上所述，VASP 计算过程中，无论实际计算中是否需要跨节点，VASP 将 NCPU 个计算资源作为总体对待，虽然有一定的控制参数将计算资源进行划分，但是 VASP 运行中并未开启额外的进程，总进程保持为 NCPU，因此额外的进程可能由 MPI 跨节点并行和作业管理系统对作业和计算资源监控产生的。

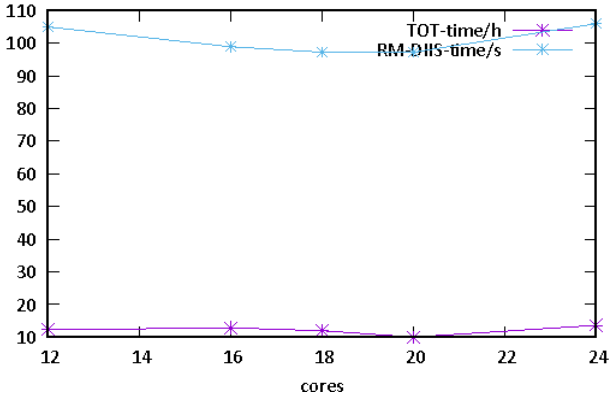


Figure 2. VASP parallel test single-node
图 2. 单节点 VASP 并行测试

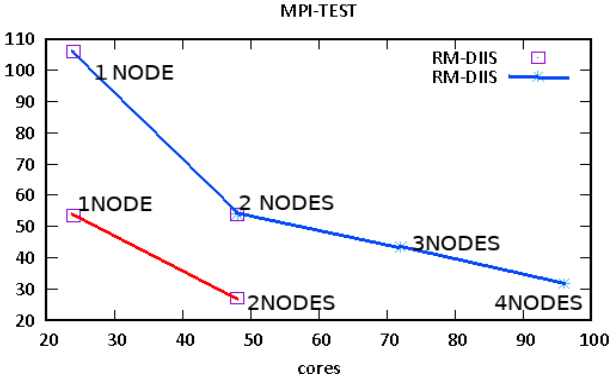


Figure 3. VASP parallel test inter-nodes
图 3. 跨节点的 VASP 并行测试

根据上述讨论，我们可以得到以下结论：在高性能集群中的 VASP 计算，从作业管理的角度考虑，当指定的计算资源是跨节点计算时，除了 VASP 作业自身需开启的流程，作业管理系统会开启一部分进程，实现对作业和计算资源的管理、监控和维护。这些额外的进程通过主进程 mbatchd 监控。一般情况下，这些进程占用的系统资源不大。因此，对于 VASP 这样的计算作业，由于作业管理产生的进程，对计算资源和计算效率的影响可以忽略不计。但是，由于这些进程伴随作业任务的全过程，因此 Wall-time 的影响，则必须考虑。

4. 矩阵并行对角化及实验效果

VASP 软件中的矩阵对角化计算采用两个阶段，分别采用 EDDIAG 和 RMM-DIIS 算法，为了降低问题复杂度并体现 MPI 并行对矩阵对角化的效率提升，我们采用分治法对矩阵进行分块，每个块独立求解后再用 Thomas 算法进行归约计算。

4.1. 算法描述

对角矩阵是指只有主对角线上含有非零元素的矩阵，即，已知一个 n × n 矩阵 M，如果对于 $i \neq j$，$M_{ij} = 0$，则该矩阵为对角矩阵。如果存在一个矩阵 A，使 $A^{-1}MA$ 的结果为对角矩阵，则称矩阵 A 将矩阵 M 对角化。对于一个矩阵来说，不一定存在将其对角化的矩阵，但是任意一个 n × n 矩阵如果存在 n 个线性不相关的特征向量，则该矩阵可被对角化。

```
1:
2: Begin the Reduction phase of the algorithm
3:
4: for (i = 1; i <= log2(n+1); i++) do              ▷ Loop through number of stages to solve
5:     for j = 0 to numactivep do                    ▷ Loop through active processors
6:         if this processor is active then
7:             Reduce system of 3 equations to 1 equation
8:             if not reduced then
9:                 send/receive information to/from neighbours
10:            end if
11:        end if
12:    end for
13:    Reduce active processor list
14: end for
15:
16: Begin the Back substitution phase of the algorithm
17:
18: for (i = log2(n+1)-1; i >= 1; i−−) do        ▷ Loop back through number of stages to solve
19:    MPI::Allgather(result)                          ▷ send result to all processors
20:    refill active processors
21:    for j = 0 to numactivep do                ▷ Loop through processors without solutions
22:        if this processor is active then
23:            Solve for local solution on this processor
24:        end if
25:    end for
26: end for
27: MPI::Allgather(local result)                       ▷ send local result to all processors
28:
29: if processor 0 then          ▷ Processor 0 loops through even rows and computes final values
30:    for all even rows do
31:        find solution
32:    end for
33:    Print solutions
34: end if
```

Figure 4. Pseudo code of algorithm
图 4. 矩阵并行对角化程序伪代码

算法描速：

1) 矩阵分块，假设矩阵规模为 n，并行数量为 p，令 k = n/p，则矩阵划分为 k*k 的矩阵块；

2) 在每个进程中对矩阵进行独立求解；

3) 用 Thomas 算法对 2)中的结果进行归约。

求解算法的伪代码见(图 4 矩阵并行对角化程序伪代码)

4.2. 实验结果

实验以串行的 Thomas 算法作为参考，求解规模分为 1,000,000，5,000,000，10,000,000，50,000,000 四种，分别在 2、4、10、16、20、32 核的并行度下完成计算。计算时间、加速比和并行效率如表 2 所示。

Table 2. Parameters of matrix diagonalization
表 2. 矩阵对角化参数

求解规模	进程数	运行时间	加速比	效率
	串行：0.057035			
1,000,000	2	0.110394	0.88302806	0.44151403
	4	0.045438	1.25522414	0.31380604
	10	0.018939	3.00200644	0.30020064
	16	0.012024	4.85033974	0.30314623
	20	0.009271	6.10399584	0.30519979
	32	0.005903	9.60506589	0.30015831
	串行：0.28042			
5,000,000	2	0.327978	0.82309179	0.4115459
	4	0.219792	1.27037836	0.31759459
	10	0.095686	2.92946722	0.29294672
	16	0.06291	4.88983486	0.30561468
	20	0.035386	7.87165017	0.39358251
	32	0.021872	12.8209583	0.40065495
	串行：0.561426			
10,000,000	2	0.644677	0.876874776	0.438437388
	4	0.309206	1.729714171	0.432428543
	10	0.183201	3.048149301	0.30481493
	16	0.121082	4.61117259	0.28819828
	20	0.068256	8.225310926	0.411265546
	32	0.042963	12.96180434	0.405056386
	串行：2.79557			
50,000,000	2	3.07504	0.898105391	0.449052695
	4	2.12244	1.254122614	0.313530653
	10	0.895866	3.09856608	0.309856608
	16	0.569611	4.899396255	0.306212266
	20	0.321817	8.706314458	0.435315723
	32	0.202909	13.77745689	0.430545528

实验结果表明利用分治算法，在 OpenMPI 和分布式并行环境下，矩阵对角化计算具有较强的扩展能力，在文中的规模上并行效率基本稳定，运行时间随着核数增加线性减少，直观效果见图 5~8。

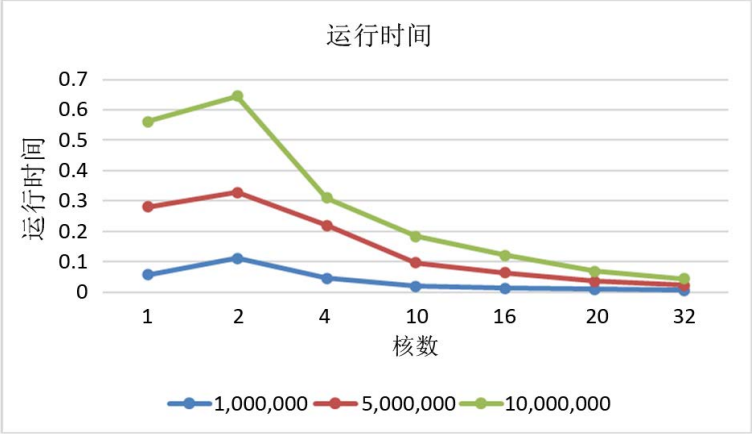


Figure 5. Processing time under 1 M, 5 M, 10 M size
图 5. 1 M，5 M，10 M 矩阵规模下执行时间

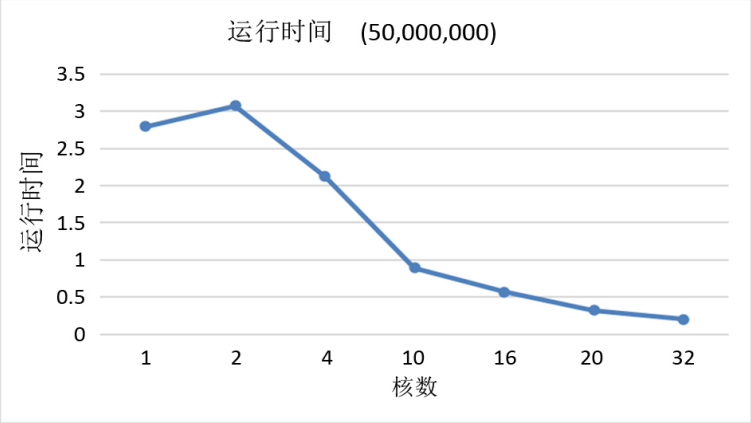


Figure 6. Processing time under 50 M size
图 6. 50 M 矩阵规模下的执行时间

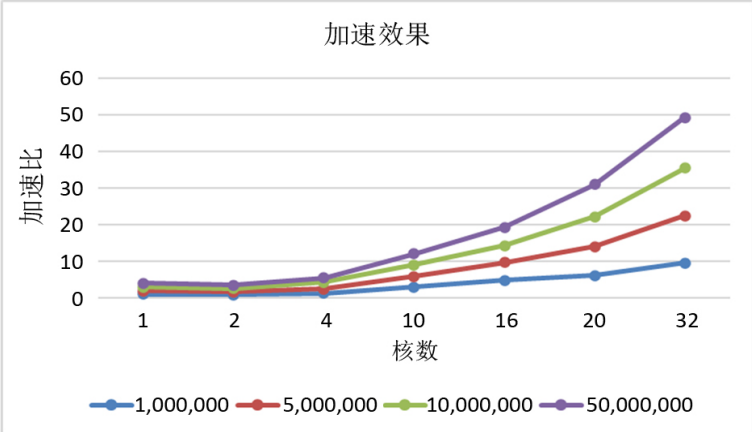


Figure 7. Speedups in different core count
图 7. 不同核数的并行加速比

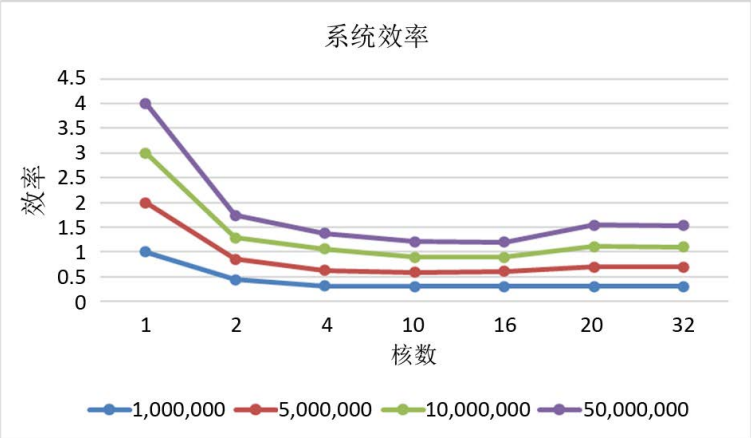


Figure 8. System utilized in different size of problem and core count
图 8. 不同矩阵规模和并行核数下系统的使用效率

5. 结论

在 HPC 环境中的材料计算的核心问题是计算分解、通讯和归约。应用在实际运行中需要兼顾计算资源、通讯效率等多方面问题。在指定的计算资源是跨节点计算时，除了应用软件自身需的资源，操作系统、存储系统和管理软件也会占用一定资源。虽然，这些程序占用的系统资源不大，但对于 VASP 之类的 CPU 密集型应用，由于辅助程序产生的进程，对作业任务的全过程均会产生影响，在系统满负荷运行时，会导致计算任务的不均衡问题，不如空置少量核作为预留资源。

矩阵对角化在科学计算领域占有重要地位，如何更加合理地解构问题，处理分块求解通讯和同步问题，保证加速比随着并行度提升线性增加是矩阵优化的重要研究方向。

致　谢

本项目由国家重点研发计划项目“产学研用协同的高通量材料计算融合服务平台”(2018YFB0703900)，课题 3 高通量材料计算的工作流设计与交互图形化(2018YFB0703903)支持。

参考文献

[1] Thijssen, J.M. (2007) Computational Physics. 2nd Edition, Cambridge University Press, Cambridge, England.

[2] Ong, S.P., Chlia, S., Jain, A., Brafman, M., Gunter, D., Ceder, G. and Persson, K.A. (2015) The Materials Application Programming Interface (API): A Simple, Flexible and Efficient API for Materials Data Based on REpresentational State Transfer (REST) Principles. *Computational Materials Science*, **97**, 209R. https://doi.org/10.1016/j.commatsci.2014.10.037

[3] Potyrailo, A. Chisholm, B.J., Morris, W.G., Cawse, J.N., Flanagan, W.P., Hassib, L., Molaison, C.A., Ezbiansky, K., Medford, G. and Reitz, H. (2003) Development of Combinatorial Chemistry Methods for Coatings: High-Throughput Adhesion Evaluation and Scale-Up of Combinatorial Leads. *Journal of Computational Chemistry*, **5**, 472. https://doi.org/10.1021/cc030022s

[4] Potyrailo, R.A. and Takeuchi, I. (2005) Role of High-Throughput Characterization Tools in Combinatorial Materials Science. *Materials Science and Technology*, **16**, 1. https://doi.org/10.1088/0957-0233/16/1/E01

[5] Persson, K. (2020) Materials Project. http://www.materialsgenome.org

[6] Setyawan, W. and Curtarolo, S. (2010) High-Throughput Electronic Band Structure Calculations: Challenges and Tools. *Computational Materials Science*, **49**, 299. https://doi.org/10.1016/j.commatsci.2010.05.010

[7] Jain, A. and Hautier, G., Moore, C.J., Ong, S.P., Fischer, C.C., Mueller, T., Persson, K.A. and Ceder, G. (2011) A High-Throughput Infrastructure for Density Functional Theory Calculations. *Computational Materials Science*, **50**, 2295. https://doi.org/10.1016/j.commatsci.2011.02.023

[8] Yang, X., Wang, Z., Zhao, X., Song, J., Zhang, M. and Liu, H. (2018) MatCloud: A High-Throughput Computational Infrastructure for Integrated Management of Materials Simulation, Data and Resources. *Computational Materials Science*, **146**, 319. https://doi.org/10.1016/j.commatsci.2018.01.039

[9] Kresse, G., Marsman, M. and Furthmüller, J. (2016) VASP the Guide. http://cms.mpi.univie.ac.at/VASP/

[10] Giannozzi, P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G.L., Cococcioni, M., Dabo, I., Dal Corso, A., Fabris, S., Fratesi, G., de Gironcoli, S., Gebauer, R., Gerstmann, U., Gougoussis, C., Kokalj, A., Lazzeri, M., Martin-Samos, L., Marzari, N., Mauri, F., Mazzarello, R., Paolini, S., Pasquarello, A., Paulatto, L., Sbraccia, C., Scandolo, S., Sclauzero, G., Seitsonen, A.P., Smogunov, A., Umari P. and Wentzcovitch, R.M. (2009) QUANTUM ESPRESSO: A Modular and Open-Source Software Project for Quantum Simulations of Materials. *Journal of Physics*: *Condensed Matter*,**21**, Article ID: 395502. https://doi.org/10.1088/0953-8984/21/39/395502

[11] Jia, W., Fu, J., Cao, Z., Wang, L., Chi, X., Gao, W. and Wang, L.-W. (2013) Fast Plane Wave Density Functional Theory Molecular Dynamics Calculations on Multi-GPU Machines. *Journal of Computational Physics*, **251**, 102. https://doi.org/10.1016/j.jcp.2013.05.005

[12] Hohenberg, P. and Kohn, W. (1964) Inhomogeneous Electron Gas. *Physical Review*, **136**, B864. https://doi.org/10.1103/PhysRev.136.B864

[13] Kohn, W. and Sham, L.J. (1965) Self-Consistent Equations Including Exchange and Correlation Effect. *Physical Review*, **140**, A1133. https://doi.org/10.1103/PhysRev.140.A1133

[14] Kresse, G. and Hafner, J. (1994) Norm-Conserving and Ultrasoft Pseudopotentials for First-Row and Transition Elements. *Journal of Physics*: *Condensed Matter*, **6**, 8245. https://doi.org/10.1088/0953-8984/6/40/015

[15] Kresse, G. and Furthmüller, J. (1996) Efficiency of Ab-Initio Tottal Energy Calculations for Metals and Semiconductors Using a Plane-Wave Basis Set. *Computational Materials Science*, **6**, 15. https://doi.org/10.1016/0927-0256(96)00008-0

[16] Kresse, G. and Furthmüller, J. (1996) Efficient Iterative Schemes for Ab Initio Total-Energy Calculations Using a Plane-Wave Basis Set. *Physical Review*, **54**, Article ID: 11169. https://doi.org/10.1103/PhysRevB.54.11169

[17] Blöchl, P.E. (1994) Projector Augmented-Wave Method. *Physical Review B*, **50**, Article ID: 17953. https://doi.org/10.1103/PhysRevB.50.17953

[18] Kresse, G. and Joubert, D. (1999) From Ultrasoft Pseudopotentials to the Projector Augmented-Wave Method. *Physical Review B*, **59**, 1758. https://doi.org/10.1103/PhysRevB.59.1758