计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2015.20131922
52(4):833-842,2015

高阶精度CFD应用在天河2系统上的异构并行模拟与性能优化

王勇献[1,2] 张理论[1,2] 车永刚[1,2] 徐传福[1] 刘 巍[1] 程兴华[1]
[1](国防科学技术大学计算机学院 长沙 410073)
[2](国防科学技术大学并行与分布处理重点实验室 长沙 410073)
(yxwang@nudt.edu.cn)

Heterogeneous Computing and Optimization on Tianhe-2 Supercomputer System for High-Order Accurate CFD Applications

Wang Yongxian[1,2], Zhang Lilun[1,2], Che Yonggang[1,2], Xu Chuanfu[1], Liu Wei[1], and Cheng Xinghua[1]
[1](*College of Computer, National University of Defense Technology, Changsha* 410073)
[2](*Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha* 410073)

Abstract There still exist great challenges when simulating the large-scale computational fluid dynamics (CFD) applications on the contemporary supercomputer systems with many-core heterogeneous architecture like Tianhe-2, which is also one of the research hotspots in this field. In this paper, we focus on exploring the techniques of efficient parallel simulations on the heterogeneous high-performance computing (HPC) platform for large-scale CFD applications with high-order accurate scheme. Some approaches and strategies of performance optimization matched with both the characteristic of CFD application and the architectures of heterogeneous HPC platform are proposed from the perspective of task decomposition, exploration of parallelism, optimization for multi-threaded running, vectorization by employing single-instruction multiple-data (SIMD), optimization for the cooperation of both CPUs and co-processors, and so on. To evaluate the performance of these techniques, some numerical experiments are performed on Tianhe-2 supercomputer system with the maximum number of grid points achieving 1.228×10^{11}, and the total amount of processors and/or co-processors being 590000. Such a large-scale CFD simulation with high-order accurate scheme has to our best knowledge never been attempted before. It shows that the optimized code can get the speedup of 2.6X on CPU and co-processor hybrid platform than that on the CPU platform only, and perfect scalability is also observed from the test results. The present work redefines the frontier of high performance computing for fluid dynamics simulations on heterogeneous platform.

Key words computational fluid dynamics (CFD); high-order accurate scheme; parallel computing; CPU+MIC heterogeneous computing; performance optimization; Tianhe-2 supercompter

摘 要 在当前主流的众核异构高性能计算机平台上开展超大规模计算流体力学(computational fluid dynamics, CFD)应用的高效并行数值模拟仍然面临着一系列挑战性技术问题,也是该领域的热点研究问题之一.面向天河2高性能异构并行计算平台,针对高阶精度CFD流场数值模拟程序的高效并行进行了探索,重点讨论了CFD应用特点与众核异构高性能计算机平台特征相适应的性能优化策略,从任务

收稿日期:2013-12-18;修回日期:2014-04-29
基金项目:国家自然科学基金项目(61379056, 11272352);国家"九七三"重点基础研究发展计划基金项目(2009CB723803)

分解、并行度挖掘、多线程优化、SIMD 向量化、CPU 与加速器协同优化等方面，提出一系列性能提升技术. 通过在天河 2 高性能异构并行计算平台上进行了多个算例的数值模拟，模拟的最大 CFD 规模达到 1 228 亿个网格点，共使用约 59 万 CPU+MIC 处理器核，测试结果表明移植优化后的程序性能提高 2.6 倍左右，且具有良好的可扩展性.

关键词 计算流体动力学；高阶精度格式；并行计算；CPU+MIC 异构协同并行；性能优化；天河 2 超级计算机

中图法分类号 TP301

近年来，随着计算流体力学(computational fluid dynamics, CFD)方法的不断突破和计算机技术的快速发展，基于 CFD 的数值模拟方法开始越来越多地被应用到复杂外形飞行器的研究和设计当中，成为航空航天飞行器研制的有力工具. 为了提高 CFD 数值模拟的计算规模、计算效率，更好地满足工程设计和科学研究，CFD 数值模拟代码通常需要并行计算，以便充分利用高性能计算机的强大并行处理能力. 当前，众核、异构等新型处理器逐渐成为高性能计算的主流平台，它们能以较低成本提供极大的性能，以 Intel 的新一代至强融合(Xeon Phi，简称 MIC)加速器为例，作为一类新型的众核计算加速部件，一个 MIC 设备往往拥有 50 以上的处理器核，通过采用宽向量设计可以提供 1 TFLOPS 双精度浮点运算的性能. 如何将现有 CFD 数值模拟程序向新型众核异构的高性能计算机系统上移植，以获得更高的模拟性能，将对应用软件开发人员提出严峻挑战. 本文面向天河 2 高性能异构并行计算平台，探索高精度 CFD 程序高效并行及性能优化的技术，为航空航天 CFD 应用提供技术积累.

1 高阶精度 CFD 应用背景及其并行计算

1.1 控制方程与 CFD 求解流程

本文使用经典三维非定常 Navier-Stokes 控制方程对飞行器流场进行数值计算，以来流速度、来流密度以及翼的平均弦长 L 为特征量进行无量纲化，在曲线坐标系 (ξ,η,ζ) 下微分形式的控制方程组可写为

$$\frac{\partial \boldsymbol{Q}}{\partial t}+\frac{\partial(\boldsymbol{E}-\boldsymbol{E}_v)}{\partial \xi}+\frac{\partial(\boldsymbol{F}-\boldsymbol{F}_v)}{\partial \eta}+\frac{\partial(\boldsymbol{G}-\boldsymbol{G}_v)}{\partial \zeta}=\boldsymbol{0}, \quad (1)$$

其中，$\boldsymbol{Q}$ 为基本物理量(原始量或守恒量)；$\boldsymbol{E},\boldsymbol{F},\boldsymbol{G}$ 分别为 ξ,η,ζ 这 3 个方向上的对流通量；$\boldsymbol{E}_v,\boldsymbol{F}_v,\boldsymbol{G}_v$ 分别为 3 个方向上的粘性通量. 为方便叙述，式(1)中等号左边的 4 项分别称为时间导数项和 3 个方向上的空间通量导数项，其中每个方向上的空间通量导数项又进一步分成对流(通量导数)项和粘性(通量导数)项. $\boldsymbol{Q},\boldsymbol{E},\boldsymbol{F},\boldsymbol{G},\boldsymbol{E}_v,\boldsymbol{F}_v,\boldsymbol{G}_v$ 的具体形式参见文献[1-5]，对于三维流场而言，它们都是具有 5 个分量的向量.

为了数值求解式(1)的连续控制方程，需要利用有限差分法、有限体积法和有限元素法等对其数值离散，从而在每个时间步内形成大型线性代数方程组，并利用显式 Runge-Kutta 方法、Jacobi 迭代、LUSGS 迭代等各类经典方法进行求解. 典型的 CFD 求解流程如图 1 所示. CFD 数值模拟的一个重要研究课题是如何获得更为精细准确的模拟结果，为此在空间离散方法、时间离散方法、求解器的选用等方面都发展了一系列成熟的理论与方法.

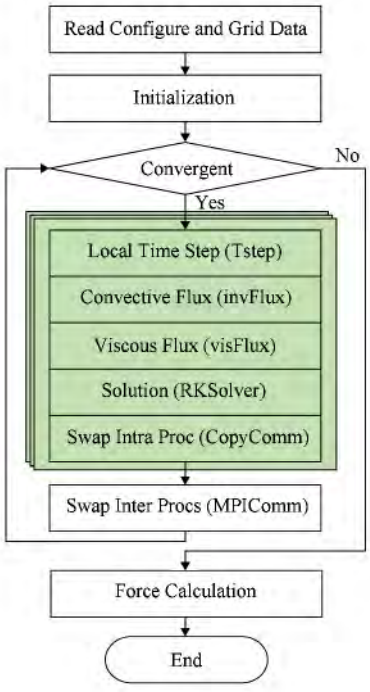

Fig. 1 Flowchart of parallel CFD simulations.

图 1 高阶精度 CFD 模拟流程

1.2 高阶精度格式 WCNS-E-5 的空间离散方法

本文的高阶精度模拟方法中，空间离散使用有限差分方法，对流项采用低耗散、高分辨率特征的

5 阶精度加权紧致非线性格式(fifth-order explicit weighted compact nonlinear scheme, WCNS-E-5)单元中心型差分离散格式,粘性项采用 4 阶中心差分离散格式[2-4]. 为了保证高阶精度的需求,除了内部网格单元的高精度格式外,还需要构造与之相匹配的高阶精度边界格式和高阶精度网格导数计算格式,详细方法的介绍可参见文献[6].

空间项离散完成后,式(1)转化为式(2)的半离散形式,需要进一步对时间项进行离散.

$$\frac{\partial \boldsymbol{Q}}{\partial t} = -R(\boldsymbol{Q}), \tag{2}$$

其中,右端项$-R(\boldsymbol{Q})$表示已经完成的空间离散数值.

1.3 时间离散与推进求解

通过对时间项离散后得到了多时间步的离散方程组,整个流场采用时间推进的方式进行求解,从第 n 个时间步的流场基本量 $\boldsymbol{Q}^{(n)}$ 出发,通过显式方法与隐式方法获得第 $n+1$ 个时间步的流场量 $\boldsymbol{Q}^{(n+1)}$. 本文基于 3 步 Runge-Kutta 法用于非定常流场的数值模拟,时间推进公式如下:

$$\boldsymbol{Q}^{(n,1)} = \boldsymbol{Q}^{(n)} + \lambda \times R(\boldsymbol{Q}^{(n)}); \tag{3}$$

$$\boldsymbol{Q}^{(n,2)} = \frac{3}{4}\boldsymbol{Q}^{(n)} + \frac{1}{4}[\boldsymbol{Q}^{(n,1)} + \lambda \times R(\boldsymbol{Q}^{(n,1)})]; \tag{4}$$

$$\boldsymbol{Q}^{(n+1)} = \frac{1}{3}\boldsymbol{Q}^{(n)} + \frac{2}{3}[\boldsymbol{Q}^{(n,2)} + \lambda \times R(\boldsymbol{Q}^{(n,2)})]. \tag{5}$$

其中,λ 是与时间步长相关的系数.

1.4 并行计算策略

在典型多区结构网格 CFD 应用的大规模并行模拟中,根据高性能计算平台的不同,分别采用分布存储系统上的 MPI 多进程并行编程模型和共享存储系统的 OpenMP 多线程并行编程模型. 本文的高阶精度 CFD 并行模拟程序在向天河 2 高性能异构并行计算平台移植之前主要采用分布存储 MPI 并行编程模型加以实现,并针对天河 1 高性能计算系统进行了并行优化[7],主要包括:1)优化了进程间通信,提前发送消息,推迟接收消息,计算与通信充分重叠,减少等待时间等;2)由于高精度并行程序采用有限差分法,为了提高计算精度其插值模板会较宽,从而导致计算过程中需要传递的数据量增加,为了缩短通信时间,采用了非阻塞通信方式,实现计算与通信重叠;3)以各进程上的计算量是否平衡作为度量准则,优化实现了多进程计算的负载平衡.

尽管对 MPI 并行编程实施了上述优化,然而由于天河 1 高性能计算系统的 CPU+GPU 异构混合架构的特点,再加上 CPU 结点本身也是由共享存储的多 CPU 核组成,因此原始的高精度程序既没有有效利用天河 1 计算系统中的加速器、在共享存储平台上使用消息传递编程模型也潜在地增加了通信开销,最终导致整个并行程序的可扩展性受到限制. 测试结果表明,原始 MPI 并行程序在天河 1 高性能计算系统上扩展到 5 000 万网格规模时性能尚可,但当扩展到 1 亿以上网格规模时并行效率下降较大,且此时使用最多的核数仅能达到 144 个. 受限于此,当模拟外形更为复杂、网格更为精细的真实飞行器流场时,现有 CFD 程序无法适应其对网格规模急剧膨胀的需求. 为此,本文中我们面向天河 2 高性能计算平台对原有高精度 CFD 程序进行了并行移植与优化,使其可完全利用天河 2 高性能异构并行计算平台的新型异构计算部件,并极大地提高其可扩展性,为下一步进行真实飞行器周围流场的高精度模拟奠定基础.

2 高阶精度 CFD 应用的异构并行与优化方法

2.1 天河 2 高性能异构并行计算平台

高精度 CFD 程序的并行移植必须与高性能计算平台的体系结构特点紧密结合. 天河 2 高性能异构并行计算平台(以下简称天河 2 系统)的计算性能主要由 16 000 个计算结点提供,每个计算结点呈异构体系结构,由 2 块 Intel Xeon E5-2692(至强 CPU)处理器以及 3 块 Intel Xeon Phi 31S1P(或简称为 MIC)协处理器组成. 每个至强 CPU 采用了 12 核 24 线程,默认频率为 2.2 GHz;每个 MIC 加速器含 57 个核,时钟频率为 1.1 GHz. 每个计算结点上为 CPU 配置 64 GB 内存,每块 MIC 上拥有 8 GB 的内存. 天河 2 系统总处理器核数达到 312 万个,双精度浮点运算的总峰值性能达到 54PFLOPS,在 2013 年 6 月全球超级计算机 500 强排名中位列第一.

为了在天河 2 系统上获得较好的性能,我们分 2 步完成高精度 CFD 程序的移植.

2.2 面向众核 CPU 平台的多线程并行实现

充分利用高性能计算环境的结点内共享主存特点进行多线程并行模拟,是有效扩展原有 CFD 程序的重要途径. 为此,我们采用开放的 OpenMP 编程模型对原有程序进行多线程并行化改造. 通过在单个机器结点上对中等规模的 CFD 应用的数值模拟进行性能测试,找到 3 个性能热点的计算核心模块:对流项计算模块 *invFlux*()、粘性项计算模块 *visFlux*()和显式的 Runge-Kutta 求解器模块 *RKSolver*(),分别约占总迭代计算时间的 48.4%,23.6%,6.1%,

合计达到总迭代计算开销的 78.1%. 权衡并行粒度、代码实现的简洁性与可移植性等因素后,我们选择对 3 个计算模块内部的网格点迭代计算进行数据划分这种并行策略,通过适当的代码变换(包括调整计算次序、循环融合与分裂、循环仿射变换等),使用 OpenMP 编译指导语句完成每个热点计算模块的多线程并行,形成了初步的 MPI+OpenMP 两层混合并行实现(简记此版本为 MPIOMP).

2.3 面向 CPU+MIC 异构计算平台的并行优化

进一步利用天河 2 系统中的加速器是挖掘高性能计算系统处理潜力、有效扩展 CFD 数值模拟问题规模的必要措施. CPU 多核和 MIC 协同编程有许多模式,比较实用的模式有:1) native 模式. 程序只在 MIC 上执行计算任务,CPU 处理空闲状态. 2) offload 加载模式. 该模式即 CPU 端发起主函数,将部分计算核心通过 offload 模式加载到 MIC 上执行. 3) symmetric 对等模式. 该模式即 CPU 多核和 MIC 各自发起主函数,各自执行自己所负责的计算任务. native 模式实现最为简单,但空置了 CPU 的计算能力;加载及对等模式都较好地使用了 CPU 和 MIC 两者的计算资源,但都需要对 CPU 及 MIC 的计算负载进行精细调度,以达到最佳的负载均衡;此外,对等模式要求跨机器结点的 MIC 设备之间进行通信,对系统的网络通信硬件与协议有更高要求. 本文主要采用 offload 加载模式实现 CPU 与 MIC 的异构协同并行.

2.3.1 总体异构协同并行及负载均衡方案

CPU+MIC 异构混合并行的并行方案如图 2 所示,其任务调度与负载均衡采用静态方式加以实现,共分成 3 个层次,分别从应用问题、软件组织、硬件平台 3 方面描述了多区结构网格 CFD 应用的计算任务(应用任务层)、并行数值模拟中的多个进程与线程(运行实体层)以及高性能并行模拟系统的硬件平台结构(硬件平台层)等特点与组织方式. 通过在相邻 2 层之间建立静态的映射,便可实现任务分配与调度. 为此,首先对 CFD 应用按照区域分解原则划分成多个网格区块,不同网格区块可以独立并行地执行数值模拟;然后按照负载平衡性原则将这些网格区块分组、映射到第 2 层的各个 MPI 进程上,每个进程上的计算负载应当尽可能均衡;最后再根据高性能计算平台所能提供的硬件资源状态将进程映射到各个处理器结点上.

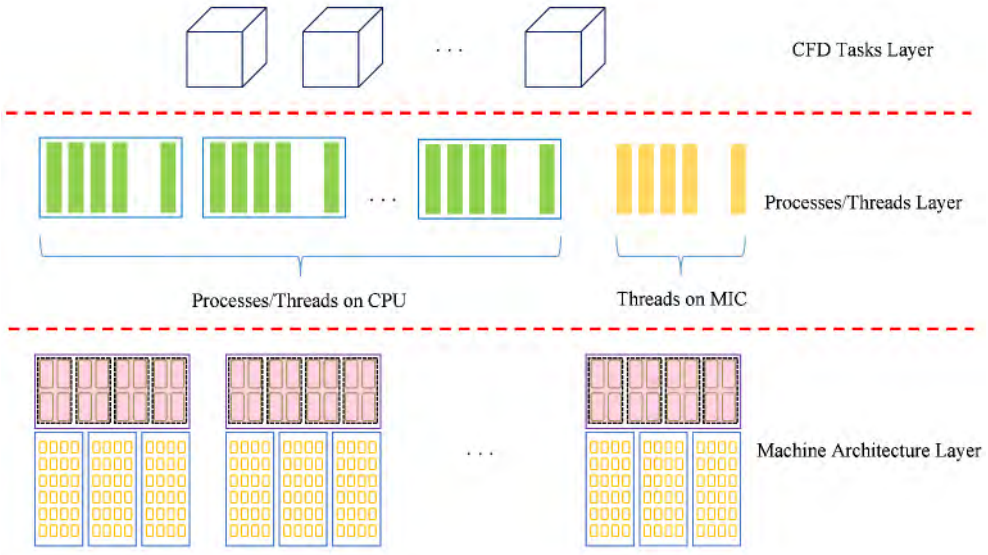

Fig. 2 Strategy of CPU+MIC hybrid parallel computing for CFD applications.

图 2 CPU+MIC 异构混合并行的并行方案

考虑到天河 2 系统中每个处理器结点包含了 2 个 CPU 计算设备和 3 个 MIC 计算设备,为了便于均衡负载,我们设计了 2 种运行配置方案:1)在每个结点上运行 1 个 MPI 进程,结点内使用 OpenMP 多线程方式利用所有的 CPU 核,其中 1 个线程负责将部分计算任务加载到 3 个 MIC 设备上;2)每个结点上运行多个 MPI 进程,每个进程使用 OpenMP 多线程方式利用结点内的 1 组 CPU 核,其中 1 个线程负责将部分计算任务加载到 3 个 MIC 设备上. 第 1 种方案易于达成负载平衡,仅需要关注结点内 CPU 设备与 MIC 设备间的负载比例即可,难点是加载到 MIC 设备上的计算需要挖掘足够大的线程级并行性才能提高加速部件利用率;而第 2 种方案中,由于多个进程可同时将各自的部分计算任务加载到同一

MIC 设备上,因此较容易发挥加速器利用率,缺点是需要显式为各进程指定计算与通信任务,程序可移植性差.

同一进程内的 CPU 与 MIC 协同执行,主要通过 offload 加载模式将部分计算任务从 CPU 交由 MIC 设备执行,完成后将结果收集回 CPU. CPU 端通过 OpenMP 编程模式创建多个线程,由主线程负责与 MIC 设备交互,其余线程负责利用 CPU 的众核完成各自的计算任务. 主线程承担的角色也有 2 种选择:一种是主线程只负责向 MIC 分发计算任务及回收数据,本身不承担 CPU 端的计算任务;另一种则不同,主线程除完成上述与 MIC 的交互外,它自身也如同其余线程一样,承担了 CPU 端的部分计算任务. 为了充分利用 CPU,我们采用了第 2 种方案,并结合下文介绍的 CPU 与 MIC 间异步计算方法,使整个计算不会因为主线程负荷过重而减慢速度.

同一进程内 CPU 与 MIC 间还面临如何分配任务的问题,为使负载平衡更易于实现,我们采用一种"对等任务"的思想,将分配给每个进程的多个网格区块分成 2 类规格的 5 个分组{$G0$,$G1$;$G2$,$G3$,$G4$}. 其中,$G0$ 和 $G1$ 的计算任务交由进程内的 2 个 CPU 去完成;$G2$,$G3$ 和 $G4$ 规模相同或相近,其计算任务分别交由 3 个 MIC 设备各自完成. 这种对等任务分解所带来的一个好处是:不论 CPU 上的计算任务还是 MIC 上的计算任务,尽管其硬件具有异构性,但在构造其更细粒度的并行(线程级、向量化指令级等)策略方面可以采用基本相似的思路.

上述静态任务调度与负载均衡策略对原始 CFD 应用的网格分布提出了严苛的要求,为此我们提出并实现了一个针对多区结构网格 CFD 应用、适用于异构并行模拟的负载平衡算法与工具[8-9],用于事先对网格剖分及组合,形成符合上述原则的 2 层分组结构.

2.3.2 任务分解与多层次并行性的开发

为使高精度 CFD 程序在具有异构众核的天河 2 系统上最大限度地发挥并行性能,我们对其进行了一系列的优化. 不论是 CPU 还是 MIC,处理器核数众多是其重要的体系结构特点,提高众核利用率是达到高性能的关键之一,为此首先需要对 CFD 应用问题进行充分的任务分解,并通过必要的流程重构,以发掘其多层次的并行性[10]. 按粒度从粗至细的次序,第 1 层采用区域分解对网格和流场分区,每个区域间可使用数据并行策略进行并发模拟;在每一流场区域内我们按照任务分解原则形成第 2 层次的并行性,例如对流项计算与粘性项计算可并行进行,每一项通量计算模块的内部 3 个方向间彼此无依赖也可并发计算;若在每个流场区内对主要的空间迭代计算使用数据分块技术往往还可进一步形成更多层次的并发性,例如在简单的一层数据分块中,形成的外层数据块循环可用于线程级并行计算,内层的数据块内循环可用于细粒度的向量化并行,等等. 一旦将任务分解成多层次的并发性子任务,便可进一步建立多层次的并行实现方法. 为此,我们将最粗粒度的并发性映射到 MPI 多进程实现层,将流场区域内的并发性采用 OpenMP 多线程加以实现,将最细粒度的数据分块内并发性通过单指令多数据(single instruction multiple data, SIMD)的向量化指令加以实现.

2.3.3 针对 OpenMP 多线程的优化

处理器结点内的众多处理器核主要是通过 OpenMP 多线程模型加以利用的,因此我们对多核多线程并行进行了优化,主要包括:

1) 多线程并行的粒度与并发度. 如 2.3.2 节所述,OpenMP 多线程主要针对单个流场区域内的各层并行性,其中流场各计算模块间的任务并行是粒度最大、满足条件的并行性,但其并发度太小(不超过 6),为此我们选择更下层的数据分块作为多线程并行的对象,为进一步增大并行度,往往还对 3 个坐标轴方向的循环同时实施数据分块. 视运行设备是 CPU 还是 MIC 的不同,利用 OpenMP 编译指导语句对最外一层或数层的块循环进行多线程任务分配与调度. 如图 3 所示:

```
!$ omp parallel do collapse(N_LEVEL)  !N_LEVEL=1,2,3
do kb=1,nk,kblksize
do jb=1,nj,jblksize
do ib=1,ni,iblksize
   do k=kb,kb+kblksize-1
   do j=jb,jb+jblksize-1
      !dir $ simd
   do i=ib,ib+iblksize-1
      !update (i,j,k)
   enddo
   enddo
   enddo
enddo
enddo
enddo
```

Fig. 3 Multithreaded execution and data blocking.

图 3 多线程并行与数据分块

2) 减少线程动态创建与销毁的开销. 在代码结构上,循环形式的多线程任务往往并不是程序的最外层循环,而是内嵌在更外层循环结构内. 为了减少因反复动态创建与销毁线程带来的额外开销,将创建线程并行区的编译指导语句提前到外层循环中,并重新仔细调整线程并行区内变量的共享属性.

3) 减小每个线程的内在占用(memory footprint). 出于对计算性能的考虑,在 OpenMP 多线程实现中,应当尽可能将数据设为线程私有变量,但这样会导致整个应用程序内存占用过大;为此我们采用共享数据分块私有化的处理,尽可能使每个线程只分配、访问和释放自身使用的数据以获取最大的性能收益.

2.3.4 SIMD 向量化优化

天河 2 系统的至强 CPU 具有 256 b 宽的向量指令,而 MIC 则具有 512 b 宽的向量指令,充分利用好向量指令集和向量计算部件,以 SIMD 方式并行可获得最多 4 倍(对 CPU 而言)或 8 倍(对 MIC 而言)的双精度浮点计算性能收益. 使用 Intel Fortran 编译器的编译选项-vec、并配合用户指定的编译指导语句,编译器可实现大部分简单代码的自动向量化. 对于编译器报告出的未能自动向量化的代码段,有针对性地进行手工优化,主要包括:1)尽可能消除分支语句,必要时将分支转化为计算;2)循环交换以确保步长为 1 的循环放到内层;3)进行循环展开,将次外层循环转化为最内层循环,增大 SIMD 粒度;4)确保数据对齐,并在待向量化代码前添加编译指导语句辅助编译器完成向量化.

2.3.5 存储使用的优化

如 2.2 节所述,高阶精度 CFD 程序每个进程的主要计算集中在最内层迭代的数个计算核心模块,特别是对流项计算 *invFlux*()、粘性项计算 *visFlux*()和求解器 *RKSolver*()模块,这 3 个模块的共同特点是计算与访存比很低、计算密度很小,特别是对流项与粘性项计算均是典型的模板计算(stencil computing),因此优化其存储使用及访存性能对提高整体性能至关重要. 我们主要采取了以下 6 种优化手段:

1) 避免在线程内反复分配与释放大块存储. 为了简化编程接口及节约存储,原程序在一些底层调用过程中大量使用了动态分配与释放内存的操作,对程序性能造成不利影响. 优化后,我们将这类对大块存储的动态分配提前到线程区的入口处,释放操作推后到线程区的出口处,保证只需要分配与释放一次,降低了时间开销.

2) 对齐动态分配存储的变量地址. 程序中多数流场变量通过动态分配存储方式使用,确保它们在存储分配时地址对齐到 16,32 或 64,不仅能对前述的向量化提供便利,更能高效利用 cache,提高存储访问性能. 数据对齐对访存的改善作用同样也适用于多数全局静态变量及过程内自动变量. 数据对齐主要通过修改分配与释放存储的语句,并添加变量属性声明语句实现.

3) 多级循环的数据分块. 如 2.3.2 节所述,主要的模板计算中普遍采用多级数据分块技术,这不仅对开发程序并行性带来好处,更为改善模板计算的时间局部性、提高 cache 命中率提供了便利.

4) 面向 cache 的优化. 为改善程序代码的访存局部性、减小 cache 失效率,我们还专门展开面向 cache 的优化. 例如:修改了程序中主要的流场变量的数据结构,尽可能用数组组成的结构体(structure of array, SOA)结构代替元素为结构体的数组(array of structure, AOS)结构,并调整了多维数组变量的下标次序,使其内存布局与访问次序尽可能匹配,以提高空间局部性;进行了必要的循环交换、消除冗余的中间变量、重组执行流程等程序变换,优化了访存的时间局部性.

5) 针对多核处理器非一致 cache 体系结构(NUCA)的优化. 不论是多核 CPU 还是众核 MIC 上都存在因 NUCA 结构而导致访存延迟的问题,我们充分利用主流操作系统普遍采用的 first-touch 存储映射机制,采用"预热"法(warm-up)将各线程中的存储动态分配、初始化等操作提前到正式迭代计算之前. 为了确保所有变量存储都得到预热,一种简易的做法是正式迭代前额外增加一次完整的多线程并行迭代过程.

6) MIC 上使用较大的存储页面. 在 MIC 设备上为大型数组分配空间时,使用大内存页面设置环境变量 MIC_USE_2MB_BUFFERS,可以有效提高 TLB 缓存命中率、降低内存页面失效率,从而减少存储分配的时间开销.

2.3.6 CPU+MIC 协同并行的优化

现实的高阶精度 CFD 数值模拟应用中,并行模拟的各进程之间、CPU 与 MIC 设备之间需要进行必要的通信以完成数据交换,这些都会对整体并行性能造成不利影响,必须仔细加以分析优化,降低其开销,具体的优化手段如下:

1) CPU 与 MIC 间数据传输的优化. 采用加载模式实现 CPU+MIC 协同并行的主要不利因素是需要在 2 种设备之间传输大量数据,主要通过以下

方法加以优化:①采用异步传输方式尽早启动数据传输,这样当 CPU 将准备好的数据向 MIC 传入的同时,CPU 本身还可进行后续的额外计算,然后再通过编译指导语句"!dir $ offload"启动 MIC 上加载的任务;反之,MIC 设备计算完毕后,同样以异步方式传出结果,传输的同时 CPU 仍可进行后续的计算,直到需要使用输出结果时再进行同步. 通过 CPU 计算与数据传输的有效重叠,可以部分或全部地隐藏数据传递的时间开销. ②通过程序流程重构,尽可能将可加载到 MIC 的计算核心组合成粒度更大的一个模块,以便通过一次 offload 完成所有任务,这样可避免因多次向 MIC 传入输出数据而带来的过大时间开销. ③使用编译指导语句"!dir $ offload"进行数据传递时,通过 alloc_if,free_if 等修饰从句避免每次加载时在 MIC 设备上重新分配及释放空间,而将空间分配及初始化放在迭代之前的"预热"阶段. ④数据传递最小化. 若前一次加载到 MIC 上计算得到的结果数据能在下次加载时使用,则避免重新传送这些数据;若一个变量仅有一段需要传递,则可利用编译指导语句"!dir $ offload"仅传递必要的数据节段;若一些传递数据可由另一些传递的数据计算得到时,为减少数据传输量,必要时用重新计算代替直接传递这些数据.

2) 不同层次设备间通信的重叠. 如图 1 的流程所示,在每次 Runge-Kutta 求解完毕后,需要进行数据交换与同步,这些通信模块进一步可区分为 2 类:跨进程的数据通信(通过 MPI 消息传递)、进程内 CPU 与 MIC 的通信(通过加载编译指导语句). 在多数应用问题中,两者之间并无依赖关系,为了进一步隐藏通信,我们将这 2 种通信异步实现并加以重叠,可以有效改善总体性能.

3) 不同设备间计算任务的重叠. 在 CPU 与 MIC 设备以及不同的 MIC 设备之间,计算任务的加载均采用异步方式,以便最大限度地实现多设备计算任务的并行与重叠,只在后续必要的位置进行同步.

除了上述主要优化外,我们还在程序中采用了指针别名消除、变量局部化、异步 I/O 操作、数据双缓冲等优化措施,不再一一赘述.

3 数值实验结果与讨论

3.1 测试平台与环境

为了考查并行化及向 CPU+MIC 异构平台的移植效果,我们对程序不同优化阶段的效果进行了测试与评估. 测试的硬件平台为天河 2 系统(见 2.1 节),每个机器结点含 2 个 12 核的至强 E5-2692 的 CPU 和 3 块 57 核的 Xeon Phi 31S1P(即 MIC)加速器,单结点为 CPU 配置 64 GB 内存,单 MIC 卡共 8 GB 内存. 我们的测试中最大规模使用了 3 072 个结点. 高阶精度 CFD 程序为 in-house 的实现代码,使用 Fortran 90 语言开发,编译器采用 Intel Fortran v13,CPU 上的编译选项取-O3 级别的优化,CPU+MIC 协同并行版本中对 MIC 端的优化选项取为-O3-xAVX 以便生成向量化代码. 本节中所有报告的结果均取 5 次以上独立测试中的最好结果,性能计时仅针对图 1 流程中的时间步迭代,不计之前的读取数据及初始化时间. 测试算例共取 3 种配置:1) DeltaWing 三角翼周围流场,共 32 个网格分区,合计 353 万个网格点;2) NACA0012 机翼外形流场,单个网格分区,共 256 万个网格点;3) CompCorner 可压缩拐角流场[11],主要用于各种网格规模下的性能,为此专门设计了可控制网格分布结构及规模的三维网格生成器. 所有算例使用非定常模拟并采用显式 Runge-Kutta 法求解,在比较程序运行性能时,时间步迭代仅取 50 步,并对测得的墙上时间归一化以便进行相互比较.

3.2 测试结果与讨论分析

由于并行移植及采用的优化方法较多,为了避免冗余的讨论、方便厘清优化效果,这里我们仅针对几个优化过程中的里程碑节点进行对比分析,为此将高精度 CFD 程序区分为 4 个版本:1) MPI_ORG 表示实现了 MPI 并行的原始版本;2) MPI_OMP 表示初步加入 OpenMP 多线程,实现了 MPI+OpenMP 混合并行的版本;3) CPU_OPT 表示上述版本经过优化运行于 CPU 平台的版本;4) CPU+MIC 表示运行于 CPU+MIC 混合平台的最终优化版本.

3.2.1 CPU 上的优化结果

原始版本 MPI_ORG 实现了 OpenMP 多线程并行后得到了 MPI_OMP 版本,在 CPU 平台上分别取单结点的 DeltaWing 算例、8 个结点的 DeltaWing 算例以及单结点的 NACA0012 这 3 个算例进行测试比较,如表 1 所示.

表 1 中计时取为平均单个时间步迭代的墙上时间,MPI_OMP 版本运行时取线程数为 24,符号 np 表示机器结点的数目(每个结点上运行 1 个进程). 由表 1 可知,由于 OpenMP 多线程只能针对网格块内的部分计算核模块,并行粒度比较小,因此尽管

线程越多加速比越高(表 1 中未给出其余线程数目下的性能),但并行效率却没有同步增长.

Table 1 Performance Speedup on CPU by Using OpenMP Threading

表 1 OpenMP 多线程并行带来的性能提升效果

np	Test Cases	Wall Time/s		*Speedup*
		MPI_ORG	MPI_OMP	
8	DeltaWing	8.3	0.9	9.53
	DeltaWing	67.2	7.2	9.37
1	NACA0012	75.7	7.9	9.63

在 CPU 平台上经过使用第 2 节介绍的各种优化后得到 CPU_OPT 版本,我们将其与 MPI_OMP 进行对比. 算例为 CompCorner,我们在 16 个机器结点上使用 16 个 MPI 进程进行测试,共取了 4 种不同的网格规模,网格点数分别为 28×10^6,34×10^6,40×10^6 和 46×10^6,测试中每个进程仍使用 24 个线程,测试结果如图 4 所示. 其中,图 4(a)显示了每种规模下优化版本较 MPI_OMP 版本的加速比情况;图 4(b)的纵轴为绝对性能,即以每秒更新的网格单元数表示;图 4 横坐标的标值含义是 CPU 上分配的网格+MIC 上分配的网格,比如 16M+12M 表示 28M 的总网络点数中 CPU 上分配了 16M、MIC 上分配了 12M,M 为 10^6,其余标值以此类推. 图 4 的结果表明,与未经优化的 MPI_OMP 版本相比,优化后的版本性能平均提升 1.7 倍左右.

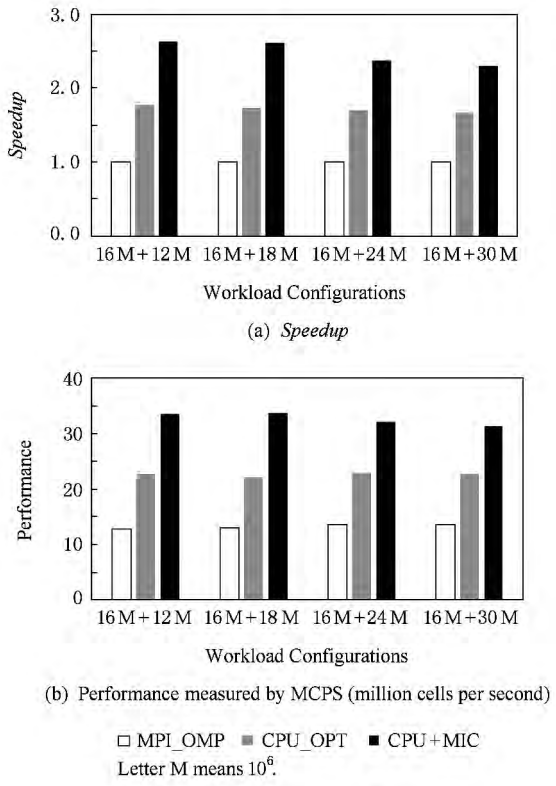

Fig. 4 Performance comparison of three code versions.

图 4 异构协同并行优化 3 个版本的性能比较

3.2.2 CPU+MIC 异构优化及负载分配效果

CPU 上的优化版本在向 CPU+MIC 混合平台上移植时必需解决静态负载平衡问题,为了测试方便,我们采用 2.3.1 节介绍的第 1 种协同并行方案,即每个结点运行 1 个 MPI 进程,结点内使用 OpenMP 多线程细粒度并行,其中主线程负责将部分计算任务加载到 3 个 MIC 设备上. 在任务分配上,每个进程分配 5 个网格块,其中 2 个块取相同的规模,分别交由 2 个 CPU 计算;另 3 个块取另一种规模,分别交给 3 个 MIC 设备进行计算. 为了达到最佳的负载平衡效果,我们固定 CPU 负责的网格块规模为 $8\times10^6\times2=16\times10^6$,每个 MIC 设备负责的网格块规模取 4×10^6,6×10^6,8×10^6 和 10×10^6 共 4 种情形,图 4 报告了 4 个不同规模的 CompCorner 算例分别在 MPI_OMP,CPU_OPT 和 CPU+MIC 这 3 种版本下的性能数据. 图 4 结果表明,当 MIC 设备网格块规模取 4×10^6 或 6×10^6 时,协同并行达到最佳的加速效果,由于 MIC 的参与计算性能较移植之前提高约 1.5 倍,CPU 优化及 MIC 协同并行的总加速达到 2.6 倍. 考虑到高阶精度 CFD 是一个实用的业务程序,流程中包含有各种复杂的逻辑分支,达到这样的加速效果是令人满意的.

通过上述较小机器规模下的测试,可以确定 CPU 工作负载与 MIC 负载的最佳比例. 我们取每进程上的工作负载为 2 个 8×10^6 规模的网格块和 3 个 4×10^6 规模的网格块,然后测试了更多进程并行下的性能,如图 5 所示. 由图 5 可知,由于实现了 MPI 通信与 CPU/MIC 间数据传输的重叠,增加进程数目并不会带来总时间开销的增大.

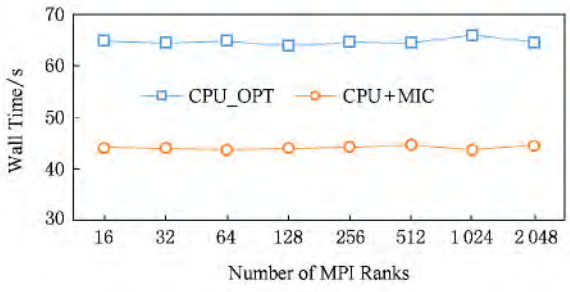

Fig. 5 Parallel weak scaling performance of the optimized codes.

图 5 异构协同并行优化代码的可扩展性

我们进一步对各种负载比例下 CPU+MIC 协同并行实现的可扩展性进行了测试,在天河 2 系统上,最大规模的测试使用 3 072 个结点,单结点处理的网格量最大达到 1 亿,最大算例总网格规模达到 1 228 亿,每结点含 24 个 CPU 核及 168 个 MIC 核,使用 CPU+MIC 核数达到 589 824 个. 测试结果如图 6 所示,同样地,图 6(a)显示了问题规模随进程数同比例增加时墙上时间的变化情况,图 6(b)的纵轴则显示了绝对性能数据(仍用每秒更新的网格单元数量表示). 结果表明,在大规模的测试中,MIC 上的工作负载比例越小,程序的运行时间越短,这一方面是由于 CPU 向 MIC 加载任务时总会带来数据传输开销,另一方面是单个 MIC 设备的主频低于单个 CPU,当 MIC 上的多线程加速无法超过 CPU 上的多线程加速时,这种主频效应就会导致 MIC 上总时间开销大于 CPU 的情形. 图 6 还显示,在每种 CPU+MIC 协同并行配置中,随着使用的处理器结点数与问题规模的同比增长,总模拟时间基本维持不变,表现了很好的可扩展性能.

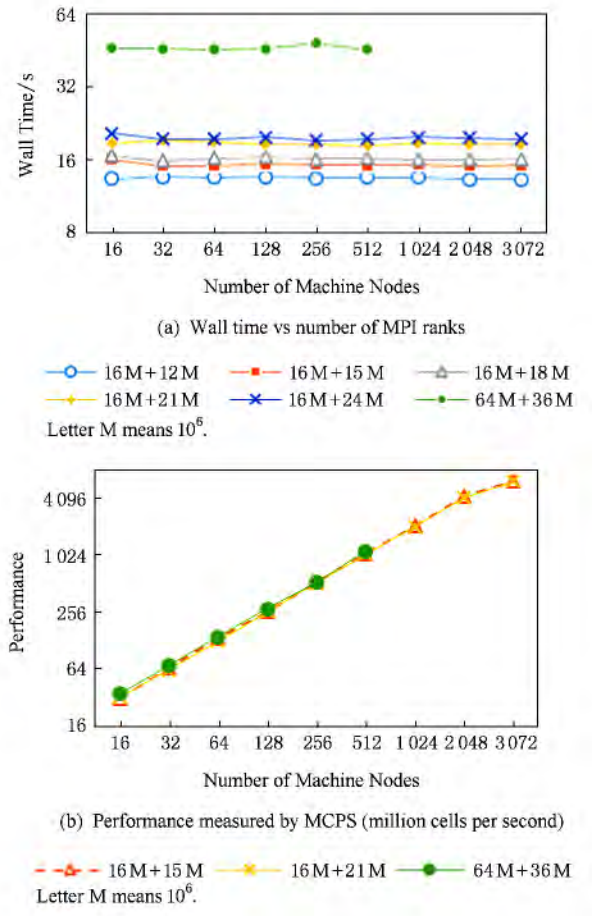

Fig. 6 Weak scaling on Tianhe-2 CPU+MIC platform.

图 6 天河 2 CPU+MIC 平台上的并行可扩展性

4 结束语

复杂飞行器外形流场的高阶精度 CFD 数值模拟对高性能计算提出了越来越高的要求,本文面向天河 2 系统,针对现有的高阶精度 CFD 程序进行多线程并行优化及向 CPU+MIC 协同异构平台进行移植,将体系结构特点与应用问题相结合,展开多层次的并行优化. 大规模数值实验表明,移植优化后的程序模拟性能提高 2.6 倍;在天河 2 系统上模拟的最大规模 CFD 应用达到 1 228 亿个网格点,共使用约 59 万个 CPU+MIC 处理器核,结果显示优化后的程序具有良好的可扩展性.

参 考 文 献

[1] Deng Xiaogang, Maekawa H, Shen C A. Class of high order dissipative compact schemes [C] //Proc of the 27th AIAA Fluid Dynamics Meeting. Reston, VA: American Institute of Aeronautics and Astronautics, 1996: A9636452

[2] Deng Xiaogang, Maekawa H. Compact high-order accurate nonlinear schemes [J]. Journal of Computational Physics, 1997, 130(1): 77-91

[3] Deng Xiaogang, Mao Meiliang. Weighted compact high-order nonlinear schemes for the Euler equations [C] //Proc of the 13th AIAA Computational Fluid Dynamics Conf. Reston, VA: American Institute of Aeronautics and Astronautics, 1997: 539-551

[4] Deng Xiaogang. High-order accurate dissipative weighted compact nonlinear schemes [J]. Science China Mathematics, 2002, 45(3): 356-370

[5] Deng Xiaogang, Mao Meiliang, Tu Guohua, et al. Geometric conservation law and applications to high-order finite difference schemes with stationary grids [J]. Journal of Computational Physics, 2011, 230(4): 1100-1115

[6] Wang Guangxue, Zhang Yulun, Li Song, et al. A study on massively parallel computation [J]. Computer Engineering & Science, 2012, 34(8): 125-130 (in Chinese)

(王光学, 张玉伦, 李松, 等. WCNS 高精度并行软件的大规模计算研究[J]. 计算机工程与科学, 2012, 34(8): 125-130)

[7] Wang Yongxian, Zhang Lilun, Liu Wei, et al. Efficient parallel implementation of large scale 3D structured grid CFD applications on the Tianhe-1A supercomputer [J]. Computers & Fluids, 2013, 80: 244-250

[8] Wang Yongxian, Zhang Lilun, Liu Wei, et al. Grid repartitioning method of multi-block structured grid for parallel CFD simulation [J]. Journal of Computer Research and Development, 2013, 50(8): 1762-1768 (in Chinese)

(王勇献, 张理论, 刘巍, 等. CFD并行计算中的多区结构网格二次剖分方法与实现[J]. 计算机研究与发展, 2013, 50(8): 1762-1768)

[9] Tang Bo, Wang Yongxian. The task load balancing algorithm in the large-scale CFD with multi-zone structured grids [C] //Proc of 2013 High Performance Computing of China. Guilin, Guangxi: Guilin University of Electronic Technology, 2013: 665-672 (in Chinese)
(唐波, 王勇献. 大规模CFD多区结构网格任务负载平衡算法[C] //2013全国高性能计算学术年会论文集. 桂林, 广西: 桂林电子科技大学, 2013: 665-672)

[10] Cai yong, Li Guangyao, Wang Hu. Parallel computing of central difference explicit finite element based on GPU general computing platform [J]. Journal of Computer Research and Development, 2013, 50(2): 412-419 (in Chinese)
(蔡勇, 李光耀, 王琥. GPU通用计算平台上中心差分格式显式有限元并行计算[J]. 计算机研究与发展, 2013, 50(2): 412-419)

[11] Li Bangming, Bao Lin, Tong Binggang. Theoretical modeling for the prediction of the location of peak heat flux for hypersonic compression ramp flow [J]. Chinese Journal of Theoretical and Applied Mechanics, 2012, 44(5): 869-875 (in Chinese)
(李邦明, 鲍麟, 童秉纲. 高超声速压缩拐角峰值热流位置预测模型研究[J]. 力学学报, 2012, 44(5): 869-875)

Wang Yongxian, born in 1975. PhD and associate professor. Member of China Computer Federation. His main research interests include applications of high performance computing, parallel computing, etc.

Zhang Lilun, born in 1975. PhD and professor. Member of China Computer Federation. His main research interests include the applications of high performance computing.

Che Yonggang, born in 1973. PhD and associate professor. Member of China Computer Federation. His main research interests include high performance computing applications, program tuning and performance evaluation of computers.

Xu Chuanfu, born in 1980. PhD and assistant professor. Member of China Computer Federation. His main research interests include high performance computing and application.

Liu Wei, born in 1980. PhD and assistant professor. His current research interests include computational fluid dynamics.

Cheng Xinghua, born in 1982. PhD and assistant professor. Member of China Computer Federation. His current research interests include computational fluid dynamics.