·技术 / TECHNOLOGY·

高精度湍流直接数值模拟程序的异构并行优化分析

张天文[1,2,3]，李新亮[2]，张鉴[1]，陆忠华[1]

1. 中国科学院计算机网络信息中心，北京 100190
2. 中国科学院力学研究所，北京 100190
3. 中国科学院大学，北京 100049

摘 要： 在众核处理器应用中，主要难点在于异构并行应用模式和负载均衡的策略，对于计算流体力学，需要针对相关应用设计相应的方案。我们针对湍流直接数值模拟中串行程序含有部分并行度较高的子程序或函数的特点，设计了一种新的并行计算模式，给出了一种异构平台优化方案，并在中科院超级计算系统“元”上进行了测试和分析，对领域内的典型算例进行了性能测试，着重讨论了不同规模下采用 offload 模式的 CPU 和 MIC 异构并行的扩展性能。

关键词： 高性能计算；优化分析；异构并行；MIC

doi: 10.11871/j.issn.1674-9480.2015.05.001

Optimization Analysis of the Heterogeneous Parallel Algorithm for High Accuracy Turbulence Direct Numerical Simulations

Zhang Tianwen[1,2,3], Li Xinliang[2], Zhang Jian[1], Lu Zhonghua[1]

1. Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
2. Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China
3. University of Chinese Academy of Sciences, Beijing 100049, China

Abstract: In the application of many-core processors, the main difficulties are to choose the heterogeneous parallel application mode and the load balancing strategy. It is important to design appropriate solutions for relevant applications in high-performance computing of computational fluid dynamics. In allusion to the feature of that the serial program contains the parts of subprograms or functions with higher parallal

基金项目：中国科学院信息化专项 (XXH12503-02-02-2(01))

performance in turbulence direct numerical simulations, we designed a new parallel computing model, and proposed a optimization scheme, and realized it on the supercomputing system, ERA, at CNIC, CAS. The performance test and analysis for typical computing examples were conducted, and the discussions are focused mainly on the scalability of the heterogeneous parallel algorithms with offload mode on the CPU and MIC heterogeneous architecture.

Keywords: high performance computing; optimization analysis; heterogeneous parallel; MIC

引言

计算流体力学 (Computational Fluid Dynamics，CFD) 已经广泛应用到航空航天，能源，环境，气象，海洋等众多工程中。其中，直接数值模拟已经成为辅助众多工程领域设计的重要工具。然而，由于其计算量极大，且随着所模拟流场的复杂度提高，流动机理的研究更加精细，以及数值模拟精度的提高，对计算机的性能也提出了更高的要求。在这种趋势下，CFD 也发展成为了典型的高性能数值计算应用学科。为了提高 CFD 问题的求解效率，需要对 CFD 程序进行优化，以充分发挥计算机系统硬件的潜力，而其中并行计算则是如今计算流体力学发展的重要研究方向。

从 1978 年 Intel 发布了 8086 处理器以来，Intel、AMD 等厂家推出的 CPU 性能不断提高，并且几乎以严格的摩尔定律所给出的方式增长着，CPU 提高单个核心性能的主要手段是提高处理器的工作频率和增加指令级并行。这两种手段在目前的发展已经遭遇瓶颈。众核处理器 (Many Integrated Core, MIC)是一种具有数十个精简的 x86 核心并将其整合在一起的新型处理器，作为计算机体系中的协处理器，对于高并行度的科学计算程序具有良好的加速效果。且在如今，异构并行计算凭借着低成本提供强大性能的能力逐渐受到开发者的亲睐。

在异构平台上进行三维流体应用的直接数值模拟时，需要充分利用计算机体系结构的特点，处理好计算任务的分布，把握住不同架构设备间的计算能力，并尽力消除浮点运算与节点间通信的矛盾关系。尽管在传统的高性能计算系统上已经实现过三维可压缩流体的 CFD 应用，并在科研及工程上取得了良好的应用，但是对于新兴的众核架构平台上的 CFD 并行计算与优化技术尚且较少。与之前较为流行的 CPU+GPU 的协同并行相比较而言，CPU+MIC 的协同并行在 CFD 的应用中的优化技术在各项研究中还很零散。由于 MIC 的出现时间较短，在大规模的并行计算平台，如：中科院超算中心“元”计算系统上有 40 个 MIC 节点，在此类计算系统上开展大规模异构协同并行 CFD 应用的研究就更为稀少。

在 2013 年 Subhash Saini 等人在 128 个节点的 CPU+ MIC 异构平台上对一个测试程序及两个 CFD 算例进行了初步的性能测试，而其中的算例测试仅限于使用单个计算节点的情况。而在同一年，王勇献等人基于天河 2 超级计算机将传统结构网格 CFD 应用移植到了 CPU+MIC 的异构平台上，最多使用了 3072 个节点进行了大规模数值模拟运算。本文中通过对 CFD 高精度直接数值模拟软件 Opencfd-sc 在 CPU+MIC 异构并行平台上的移植与优化，针对流体力学计算特点，从 N-S 方程组求解开始，重新设计并行消息交换策略，改写重叠交换区的数据结构，从而保证最大限度利用 MIC 的加速性能。并对可能出现性能瓶颈的部分进行了分析及优化，最终提出了完整的移植方案。发展了一系列针对硬件平台配置，以及针对 CFD 直接数值模拟的并行算法和优化方法。尤其是在优化方面进行了深入的探讨和分析，并在超级计算系统“元”上进行了单节点以及大规模测试，使得高精度 CFD 的应用具有良好的性能以及优异的可扩展性。

1 控制方程与求解流程

本文所使用的 Opencfd-sc 软件与国内外目前较为成熟的商业 CFD 软件相比，有着自主研发，精度高，完全开源等优点，在国内外已有数十家单位近百名用户持续使用，该软件的基本情况如下。该软件所使用的控制方程为曲线坐标系下的三维非定常可压缩 Navier-Stokes 方程，可表示为以下形式：

$$\frac{\partial Q}{\partial t}+\frac{\partial (F-F_v)}{\partial_\xi}+\frac{\partial (G-G_v)}{\partial_\eta}+\frac{\partial (H-H_v)}{\partial_\zeta}=0$$

其中，t 表示时间，Q 为守恒量，F、G 和 H 分别表示曲线坐标系的三个方向上的对流通量，F_v、G_v 和 H_v 则分别表示三个方向的粘性通量。求解过程中使用空间有限差分格式。对流项采用流通矢量分裂，对流通量采用七阶 WENOZ 格式进行离散，WENOZ 格式是一种加权本质无振荡格式，对于流场中的激波间断面附近可能出现的非物理振荡，具有良好的抑制效果。粘性通量采用八阶中心差分格式对其进行离散。按照以上方案可保证计算程序具有良好的分辨率和鲁棒性。选用了显格式进行时间推进，采用具有 TVD 性质的三阶 Runge-Kutta 时间离散格式如下：

$$\begin{cases} u^{(1)}=u^n+\Delta tL\left(u^n\right) \\ u^{(2)}=\frac{3}{4}u^n+\frac{1}{4}u^{(1)}+\frac{1}{4}\Delta tL\left(u^{(1)}\right) \\ u^{n+1}=\frac{1}{3}u^n+\frac{2}{3}u^{(2)}+\frac{2}{3}\Delta tL\left(u^{(2)}\right) \end{cases}$$

其中 $u^{(n)}$ 是第 n 层的守恒变量，L (u) 是空间离散的差分算子。这种 TVD 型 Runge-Kutta 方法是保 TVD 性质的，因此特别适用于具有强间断 (激波、接触间断等) 的问题，可以在一定程度上抑制数值振荡。

2 异构并行方案

对于异构平台编程而言，MIC 有着较为灵活的应用模式，由于 MIC 协处理器与 CPU 同样采用 x86 的指令集，MIC 在实际的应用过程中既可以作为一个协处理器，也可以作为一个独立的节点，目前 MIC 的应用模式主要有三种：(1) 以 CPU 为中心，MIC 协同计算；(2) 以 MIC 为中心，CPU 协同计算；(3) CPU 与 MIC 对等。第二种模式适用于并行程度极高的计算程序，程序中只含有少量的串行计算部分，此时程序的主函数可由 MIC 发起执行，CPU 作为协处理器计算串行部分，也可以只在 MIC 上执行计算任务，即 Native 模式，此时 CPU 处于空闲状态。第三种模式适用于多进程并行程序，程序的主函数在 CPU 和 MIC 端同时发起。此时将 MIC 设备看作与 CPU 对等的一个节点。对等模式要求跨机器节点的 MIC 设备进行通信，对于系统的网络硬件和协议的要求更高。

在本文中，主要采用了第一种应用模式，以 CPU 为中心，MIC 作为协处理器的 offload 分载模式，这种模式为 MIC 编程中最常用的模式，适合串行程序中含有部分并行度较高的子程序或函数。在该模式中，CPU 端作为控制端，负责主函数的发起执行，与 MIC 端数据的交互，以及少量的串行计算任务；MIC 端则作为计算端，完成控制端分载的高度并行化的计算任务 (图 1)。

在“元”超级计算机上每个 MIC 节点具有 2 个 CPU 计算设备和 2 块 MIC 卡，根据这种硬件配置模式，我们考虑了两种配置方案为：(1) 在每个计算结点中发起 2 个进程，在每个进程中分别分载 (offload) 计算任务至 MIC 中，利用 OpenMP 多线程编程的方式来利用所有的 CPU 核和 MIC 核。(2) 在每个计算结点发起 4 个进程，由前两个进程负责与 MIC 的通信，剩余进程则通过 OpenMP 利用所有的 CPU 核心，在 MIC 进行计算任务的同时进行计算。在方案一中，MIC 进行高并发度的计算时，若不采取异步 offload 特殊处理，CPU 核心将处于空闲状态，此时 CPU 的计算能力被闲置。而在方案二中，通过针对 CFD 应用的 MIC 版本加速能力测试后发现，在“元”平台上，通过一个进程来利用其余的所有计算核心可基本达到 MIC 卡的计算速度，CPU 进程可以通过 OpenMP 在 MIC 计算的时间段内最大限度的利用所有 CPU 的计算核心。本文对两种方案均进行了测试，并针对 OpenCFD 本来的程序结构选择了较为合适的配置方案进行计算。

在高精度 CFD 程序结构中，通过使用 gprof 和 Intel Vtune 等软件对程序进行统计，发现计算的热点函数主要存在于计算无粘项、粘性项和时间推进部分，这三部分合计达到总计算时间的 89.3%。其中在计算无粘项和粘性项时，多次循环调用差分内点格式函数，该函数部分的计算时间占总计算时间的 53.12%。因此主要将这三个模块以及其相关的程序移植到 MIC 上进行加速执行。而在程序划分计算区域时，我们采取平均划分的策略，这种方法的好处是无需在计算过程中考虑负载均衡的问题，各个进程分配到的流场数据块大小完全相等，而且各进程间的数据块通信效率较高。但是带来的问题也很明显，由于各进程间数据块大小的均相等，而在各个进程中，由于 MIC 设备只存在于部分进程，从而导致计算能力的差异无法通过分块机制来进行调节，即对具有 MIC 加速卡的进程分配较大的块，一般 CPU 进程分配较小的块在本软件中是无法实现的。如果没有特别的处理很容易造成在收集数据之前具有 MIC 设备加速的进程等待其他未完成计算任务的进程的情况产生。

由于“元”计算机上每个 MIC 节点上具有 2 个 CPU 设备和 2 个 MIC 设备，因而采用了之前方案二中提到的配置方案，即在每个结点发起 4 个进程，其中 2 个进程利用 MIC 设备进行加速，另外两个进程则仍利用 CPU 进行计算。在计算过程中，由于具有 MIC 加速的进程在将计算任务传输到 MIC 端后，其进程所控制的 CPU 会闲置下来，此时没有 MIC 加速的进程可通过 OpenMP 的线程控制充分利用所有的 CPU 核心 (图 2)。

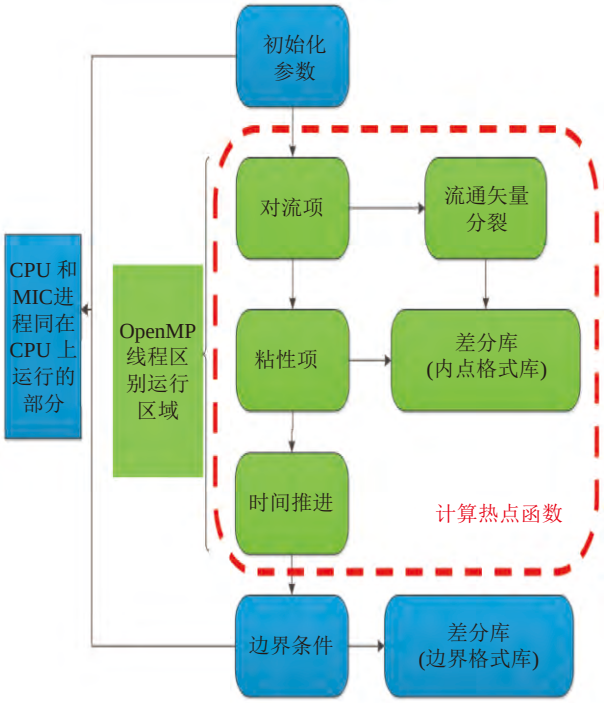

图1　软件计算流程示意图

Fig. 1　The computing process of software

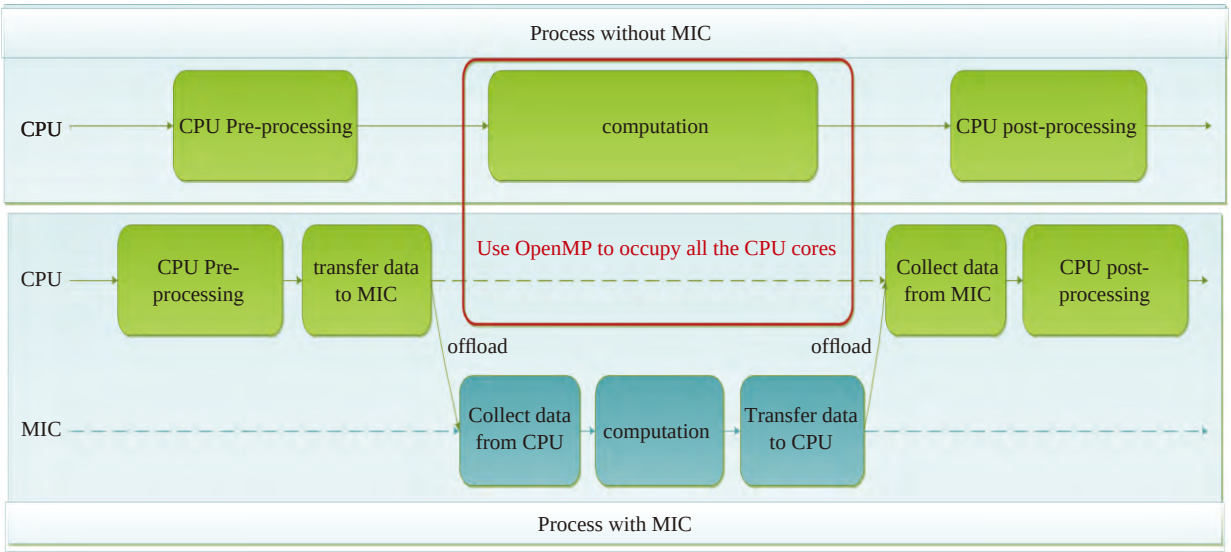

图2　异构并行优化示意图

Fig. 2　The process of heterogeneous parallel optimization

3 MIC 优化方案

为了充分发挥高精度 CFD 程序在“元”系统异构平台上的并行性能，针对第 2 节确定的移植方案采取了一系列的优化措施。在 CPU+MIC 异构平台下，程序性能的瓶颈主要来源于以下几个方面：(1) 多线程计算性能。由于在 CFD 程序中，流场区域中各网格点的循环迭代具有很高的并行度，优化好该部分的程序很大程度上决定了整个程序的性能基础。由于 OpenMP 编程模型的编译指导语句较为清晰，OpenMP 在进程内部成为优化该部分循环的主要手段之一。(2) MIC 上计算任务的整合。在将程序中的计算模块移植到 MIC 上时，不仅需要考虑模块的计算并行度，是否作为整个程序的计算热点存在，还要考虑移植后函数与 CPU 端的通信开销以及循环过程中数据的存储方式，这些都关系到整个程序的计算性能。(3) 向量化的利用。MIC 具有 512 位的宽向量指令，相对于 CPU 仅具有 256 位的向量指令而言，将循环内部代码充分向量化对于提高 MIC 上程序的性能是很有必要的。(4) MIC 和 CPU 之间的数据通信开销。MIC 协处理器是一种高密度集成大量英特尔集成众核架构内核的 PCI-E 接口形式的扩展卡，MIC 和 CPU 端传输数据的速度受到 PCI-E 接口的限制，因此，尽量减少传输的数据量，降低通信的开销，实现数据的重复利用对于性能提升的帮助同样明显。

3.1 多线程并行

异构平台节点内部的处理器核心主要通过 OpenMP 多线程模型加以利用，因此 OpenMP 多线程编程主要应用于单个流场分区内部的各网格点循环计算部分，而且这部分代码也是程序中并行度最高的部分，在 CPU 和 MIC 不同的设备上，通过调整 OpenMP 的编译指导语句对最外一层或几层循环进行多线程任务分配和调度。由于 MIC 卡上内存有限，因此不能求解过大的网格，而三个维度的变量数组基本满足内存的分块需求，如果存在最内层循环或最外层循环过小的情况，可利用 OpenMP 编译指导语句中的 collapse 选项以及展开次外层循环来提高并行度。

在使用 OpenMP 加速的循环中，应尽量减少循环内部程序的临时变量，否则会使得整个应用程序的内部占用内存过大。尽管会一定程度上降低程序的可读性，但是对于并行度较高或复用率极高的函数，改进临时变量后带来的性能提升是非常显著的。在本文所用到的高精度 CFD 软件的程序中，WENOZ 模块就符合上述的情况，该部分函数在对流项通量三个维度的计算中均被调用执行，且该模块本身也属于计算热点。在原始程序中存在大量的临时变量，且在将部分变量替代消除后，可进行一定程度上的公式化简，通过这些措施可将程序性能提高大约 10%。

3.2 并行任务

对于将程序中的计算热点函数移植到 MIC 卡上进行加速是必不可少的，但是在移植之后的性能优化的过程中，我们必须考虑计算热点前后的相关程序是否可能在 MIC 上运行，能否获得加速，以及 offload 模式下通信开销是否足够小。在本文所移植的 CFD 程序中，在 MIC 上运行的计算任务包括初始通量保存、粘性性系数的计算、对流项通量计算、粘性项通量计算以及时间推进迭代和计算原始变量。

初始通量保存函数的移植可提高通量数组的复用率，减少传输通量的数据量，在 R-K 三步时间推进中，通过判断语句只在第一步时进行通量保存即可。粘性系数计算部分的移植则主要考虑到 MIC 卡上的加速效果，且计算过程中所需要的其他变量已经在其他任务中用到，所以移植后不会带来额外的数据通信开销。时间推进部分的移植则显著可减少 CPU 和 MIC 端的通信频率，使得 Runge-Kutta 循环部分均在 MIC 上完成，只需输出最终的通量结果。

3.3 数据传输

在高并行度计算任务开始之前，进行一次初始数据传递及内存分配的 offload 加载，在这段 offload 模块中，使用 nocopy 语句，配合使用 alloc_if，free_if 语句，避免在 MIC 上频繁的重新分配与释放空间，从而降低 offload 加载可能带来的额外开销。

而在计算任务完成后，为了减少传输的数据量，

我们根据通过增大计算量来降低数据的通信量这一思想，设计了在 MIC 端和 CPU 端同时计算原始变量的方案，在时间步迭代的过程中，每次仅向循环内传输其它进程通过 MPI 通信交换得到的数据块，从而大幅减少了通信量。而且 MIC 端只向 CPU 端传递计算所得的通量 (flux)，而不是密度、三个方向的速度和温度这 5 个变量，同时在 CPU 端和 MIC 端利用通量来进行计算密度、三个方向的速度和温度这 5 个变量。这样在一次 MIC 加速过程之后，MIC 端可以保存有原始数据块所有的变量，而在下次计算任务开始之前，只需要从 CPU 端传入对应的重叠交换区的数据量即可，大大降低了运行时间。

3.4 SIMD 向量化

“元”超级计算系统上的 MIC 具有 512 位宽向量指令，在使用 Intel Fortran 编译器的编译选项 -vec-report3 可以输出向量化报告，对于未向量化的代码段手动优化。在本文的 CFD 程序中，在计算对流项通量时，每个维度的通量计算均需要调用 Steger-Warming 通量分裂函数，在原程序中函数通过传入参数 ns 判断所需要调用的维度的子函数，而三个维度子函数中的计算任务完全相同，三者的区别在于循环的范围不同。对于此类问题，可通过利用计算表达式来代替判断，从而减少了 2 倍的代码量，以及每次运行时的分支判断部分 (图 3)。

与此同时，循环过程中应保证内层循环步长为 1，如果内层循环范围较小，可将次外层循环展开，从而增大内层循环的粒度。对于 MIC 协处理器，当数据从位于 64 字节的边界地址开始传输时，效率是最佳的。因此，在进行 SIMD 计算之前，应尽量确保所调用的数组已经按照 64 字节对齐，数据对其的实现主要通过修改跟配合释放存储的代码、添加变量属性声明语句来实现。

4 测试结果及性能分析

为了全面的检测 OpenCFD 移植后的效果，我们对该程序在不同优化阶段的效果进行了测试和分析。测试的硬件平台为“元”高性能计算系统，每台 MIC 计算节点配置 2 个 Intel E5-2680 和 2 块 Intel Xeon Phi 5110P (8GB 内存) 卡，单节点具有 128 G 内存，我们在测试中程序主体采用 Fortran90 语言开发，编译器采用 Intel fortran，CPU 的编译选项采用 -O3 级别的优化，移植后的 MIC 版本程序采用 –xAVX 来生成向

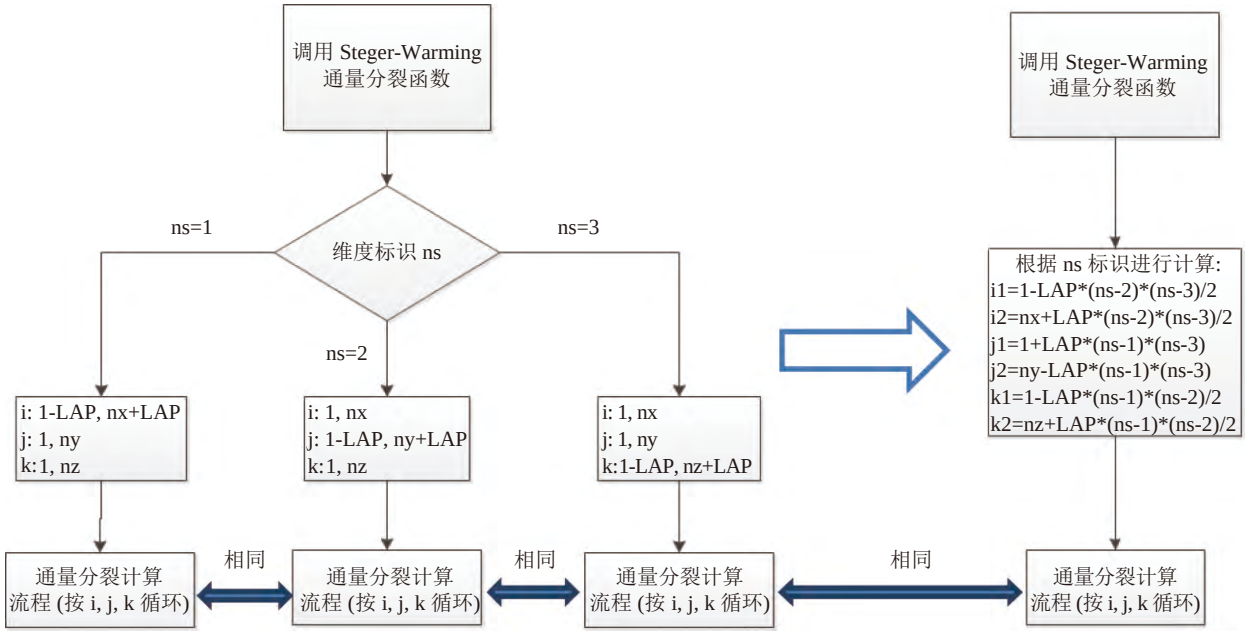

图3 向量化代码优化前后对比

Fig. 3 The optimization of vectorization

量化代码，对于评估性能时所记录的运行时间仅考虑时间步迭代过程所运行的时间，不考虑读取原始数据和初始化的时间。测试算例采用三维各向同性湍流的直接数值模拟。

在比较程序的各项性能时，均测量 300 步以上的时间迭代步的墙上时间，并最终取其中连续 100 步的平均时间作为相互比较的标准。

针对单节点测试，我们将 OpenCFD 的移植后的程序分为 4 个版本：OpenMP (1CPU) 表示采用 OpenMP 多线程在单个 CPU 实现程序并行的版本；MPI 则为经过优化后的 MPI 并行版本；双 MIC 为程序主体使用 2 块 MIC 卡进行加速的版本；双 MIC+CPU 为 CPU 与两块 MIC 卡协同并行的版本。

4.1 加速性能测试

在单节点测试中，为了便于比较 1 块 MIC 卡和 1 个 CPU 的计算能力，我们将 OpenMP 程序在单个 CPU 上的运行时间作为比较标准，即 OpenMP 版本程序运行时线程数设置为 10。测试规模为 300×168×224 的计算网格，共计约 1130 万个网格点。由图 4 可知，单块 MIC 卡即可相对于 1 个 CPU 而言产生较为明显的加速，而利用了两块 MIC 卡的加速效果达到 3 倍以上。

但是，在测试双 MIC 卡与 CPU 同时加速的异构并行的程序中，加速性能反而出现了一定的下降。经过分析发现，程序性能降低的主要原因在于，由于 OpenCFD 的应用采用了平均分快的策略，各进程间的计算数据量大小均相同，但是 CPUMICMIC 卡上超线程的优势较为明显，从而先于纯 CPU 进程完成计算。

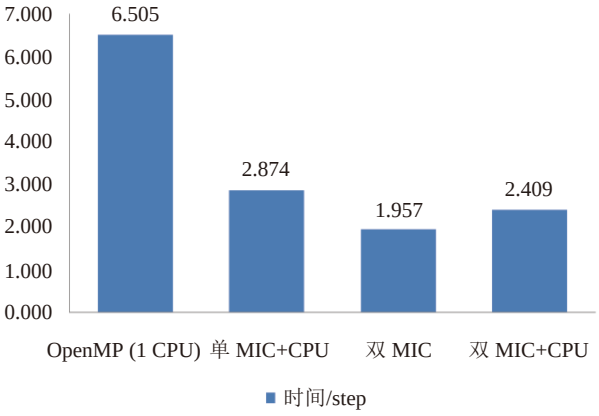

图4 单节点运行时间

Fig. 4 Computing time of single node

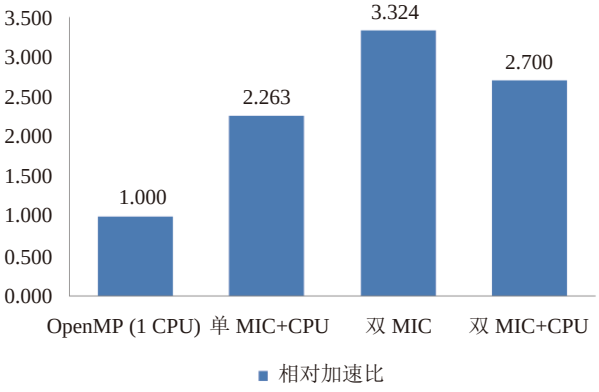

图5 单节点加速性能

Fig. 5 The speedup of single node

4.2 数据规模测试

随后，我们对数据量进行增大，尤其是针对循环中的最外层循环 (即 z 层)，z 方向的数据量大小进行了不同的测试。根据测试发现，当 z 层循环的数据大小逐渐增加，双 MIC 与 CPU 同时并行的程序性能逐渐优于单纯使用 2 块 MIC 卡加速的程序性能。且随着 z 层数据量的增大，这一现象有着则扩大的趋势 (图 6)。

表1 单节点测试性能 (规模：300×168×224)

Table 1 Performance test of single node (300×168×224)

	OpenMP(1CPU)	单MIC+CPU	双MIC	双MIC+CPU
100 step	650.480	287.444	195.691	240.946
时间/step	6.505	2.874	1.957	2.409
相对加速比	1.000	2.263	3.324	2.700

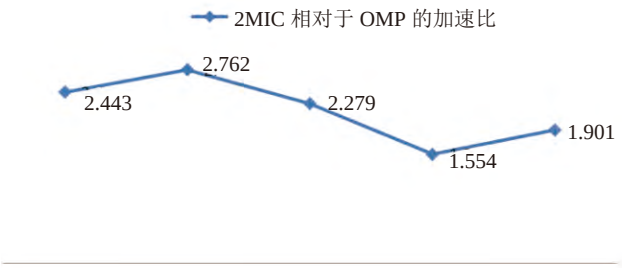

图6　2MIC 较 OpenMP 加速性能

Fig. 6　The performance comparison between OpenMP and one node with 2 MIC cooperating processors

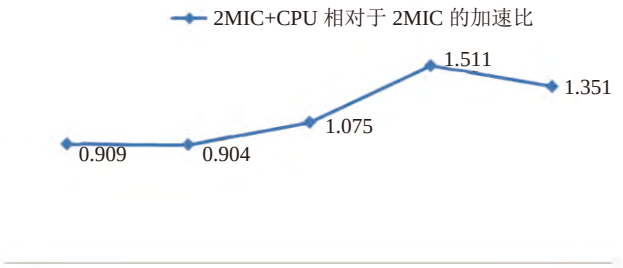

图7　异构并行较 2MIC 加速性能

Fig. 7　The performance comparison between heterogeneous parallelization and one node with 2 MIC cooperating processors

原因在于“元”的 MIC 节点所使用的 MIC 加速卡具有 61 个核心，除去一个控制核心外，最大的超线程数量为 240 个，当 z 所在的最外层循环数量超过 240 时，在进行 OpenMP 的线程调度时会产生较大的额外开销。而双 MIC+CPU 的异构并行模式，可以使得在分块过程中 z 方向的数据量降到最低，从而在一定程度上减少了线程调度的开销，提升了性能 (图 7)。

5　总结

本文在中国科学院超级计算系统“元”上对高精度直接数值模拟程序进行了众核版本的移植，从多线程、并行任务、数据传输和 SIMD 方向主要对该程序的优化手段及创新的优化算法进行了详细的说明和论证。对此类型应用的特点进行了深入的分析，针对其平均划分数据块的特点提出了具有创新性的 MIC 应用模式，并对移植后的程序进行了测试，得到了一些具有指导意义的结论。

本文得到了中国科学院信息化专项项目 XXH12503-02-02-2(01) (面向航空航天的十万核级计算流体力学软件开发及应用) 的资助，在此表示感谢。

参考文献

[1] Thomas, J. L., Krist, S. L., and Anderson, W. K., Navier-StokesComputations of Vortical Flows Over Low-Aspect-RatioWings, AIAA Journal, Vol. 28, No. 2, pp. 205-212, 1990.

[2] 李新亮, 傅德薰, 马延文. 可压缩均匀各向同性湍流的直接数值模拟. 中国科学, A 辑, 2002, 32(8): 716-724.

[3] 王恩东, 张清, 沈铂, 张广勇, 卢晓伟, 吴庆, 王娅娟. MIC 高性能计算编程指南 [M]. 北京: 中国水利水电出版社: 6-10.

[4] Jim Jeffers, James Reinders, Intel Xeon Phi Coprocessor High-Performance Programming[M], ELSEVIER, 2013: 104-134.

[5] 王勇献, 张理论, 车永刚, 等. 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 [J]. 计算机研究与发展, 2015, 52(4): 833-842.

[6] 傅德薰, 马延文. 计算流体力学 [J], 北京: 高等教育出版社: 8-20.

[7] Wei Xue, Chao Yang, Haohuan Fu, Xinliang Wang, Yangtong Xu, Lin Gan, Yutong Lu, Xiaoqian Zhu. Enabling and scaling a global shallow-water atmospheric model on Tianhe-2,2014 IEEE 28th International Parallel & Distributed Processing Symposium: 3-6.

[8] 傅德薰, 马延文, 李新亮. 可压缩湍流直接数值模拟 [M]. 科学出版社, 2010.

[9] 李新亮, 傅德薰, 马延文. 可压缩均匀各向同性湍流的直接数值模拟 [J]. 2002.

[10] 李新亮, 傅德薰, 马延文, 等. 高精度复杂流动数值模拟软件 Hoam-OpenCFD 的开发及应用 [J]. 高性能计算发展与应用上海超级计算中心编印, 2007: 52.

[11] 梁贤, 李新亮, 傅德薰, 等. 万核级可扩展 CFD 软件及应

用 [J]. 2011.

[12] Li Xinliang, Fu Dexun, Ma Yanwen. Direct numerical simulation of a spatially evolving supersonic turbulent boundary layer at Ma = 6 [J]. Chin PhysLett, 2006, 23(6): 1519-1522.

[13] 张林波, 迟学斌, 莫则尧, 等. 并行计算导论 [M]. 北京: 清华大学出版社, 2006: 210-216.

[14] 陈国良. 并行计算: 结构·算法·编程 [M]. 高等教育出版社, 2011.

[15] 张德良. 计算流体力学教程 [M]. 北京: 高等教育出版社, 2009: 62-406.

[16] Chi-Wang Shu, Essentially Non-oscillatory and WeightedEssentially Non-osicillatory Schemes for HyperbolicConservation Laws, B. Cockburn, C. Johnson, C.-W. Shuand E. Tadmor (Editor: A. Quarteroni), Lecture Notes inMathematics, volume 1697, Springer, 1998, pp. 325-432.

[17] A. Hasten, High resolution schemes for hyperbolic conservation laws[J], J. Comput. Phys, 19(1983), pp. 357-393.

[18] 陈作斌主编. 离散流体力学及应用 [M]. 国防工业出版社, 2003

[19] 孙天风. 流体力学词条, 中国大百科全书, 力学分卷. 中国大百科全书出版社, 1985, pp39-341

[20] 刘儒勋, 舒启望. 计算流体力学的若干新方法 [M]. 科学出版社, 2003.

[21] 阎超, 于剑, 徐晶磊, 范晶晶, 高瑞泽, 姜振华. CFD 模拟方法的发展成就与展望 [J]. 力学进展. Vol. 41, 2011.

[22] 张涵信, 沈孟育. 计算流体力学——差分方法的原理与应用 [M]. 北京: 国防工业出版社, 2003.

[23] C.-W. Shu and S. Osher. Efficient implementation capturing schemes II[J]. J. Comput. Phys, 83 (1989), of essentially non-oscillatory shock, pp. 32-78.

[24] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes[J]. J. Comput. Pbys., 77 (1988), pp. 439-471.

[25] A. Suresh and H.T. Huynh, Accurate monotonicity-preserving schemes with Runge- Kutta time stepping[J], J. Comput. Phys., 13G (1997), pp. 83-99.

收稿日期：2015 年 7 月 20 日

张天文：中国科学院计算机网络信息中心超级计算中心，硕士研究生，主要研究方向为高性能并行计算、计算流体力学。

E-mail: *zhangtw@sccas.cn*

李新亮：中国科学院力学研究所高温气体动力学重点实验室，研究员，博士生导师，主要研究方向为计算流体力学、湍流数值模拟和大规模并行计算。

E-mail: *lixl@imech.ac.cn*

张　鉴：中国科学院计算机网络信息中心，副研究员，硕士生导师，主要研究方向为科学计算、高性能计算和计算机可视化。

E-mail: *zhangjian@sccas.cn*

陆忠华：中国科学院计算机网络信息中心超级计算中心，研究员，博士生导师，主要研究方向为高性能计算及应用。

E-mail: *zhlu@sccas.cn*