计算机科学
Computer Science
ISSN 1002-137X,CN 50-1075/TP

《计算机科学》网络首发论文

题目：基于十亿亿次国产超算系统的流体力学软件众核适应性研究
作者：李芳，李志辉，徐金秀，范昊，褚学森，李新亮
收稿日期：2018-11-26
网络首发日期：2019-09-19
引用格式：李芳，李志辉，徐金秀，范昊，褚学森，李新亮．基于十亿亿次国产超算系统的流体力学软件众核适应性研究．计算机科学.
http://kns.cnki.net/kcms/detail/50.1075.tp.20190919.1039.006.html

基于十亿亿次国产超算系统的流体力学软件众核适应性研究

李 芳[1] 李志辉[2] 徐金秀[1] 范 昊[1] 褚学森[3] 李新亮[4]

1 江南计算技术研究所 江苏 无锡 214083

2 国家计算流体力学实验室 北京 100191

3 中国船舶科学研究中心 江苏 无锡 214081

4 中国科学院力学研究所 北京 100190

(lifang56@163.com)

摘 要 国产众核处理器提供了两种移植难度相差较大的众核级并行编程语言。不同流体力学软件对众核架构适应性的不同，决定了它们在移植优化过程中适合于不同的编程语言。首先介绍了国产众核处理器的体系结构、编程模型和并行编程语言；然后分析了流体力学软件应用于国产众核处理器存在的挑战性问题，包括隐格式带来的数据相关性、大型稀疏矩阵线性代数方程组求解、多重网格方法和非结构网格等，这些问题限制了软件对众核架构的适应性。文中针对这些难题分别提出了创新的优化算法，并通过理论分析和实验得到了几种典型流体力学软件的众核适应性研究结论。实践证明，多数流体力学软件对国产众核处理器的适应性良好，能够采用 OpenACC 编译器自动移植，并扩展到百万核并行规模，保持较高的并行效率。

关键词: 国产；众核架构；流体力学软件；适应性；编程语言；并行算法

中图法分类号： TP311 **DOI** 10.11896/jsjkx.181102176

Research on Adaptation of CFD Software Based on Many-core Architecture of 100P Domestic Supercomputing System

LI Fang[1], LI Zhi-hui[2], XU Jin-xiu[1], FAN Hao[1], CHU Xue-sen[3] and LI Xin-liang[4]

1 Jiangnan Institute of Computing Technology, Wuxi, Jiangsu 214083, China

2 National Laboratory of Computational Fluid Dynamics, Beijing 100191, China

3 China Ship Scientific Research Center, Wuxi, Jiangsu 214081, China

4 Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China

Abstract Domestic many-core super computing system provides two program languages with different program

到稿日期： 2018-11-26 **返修日期：** 2019-04-23

基金项目： 国家重点研发计划基金 National Key Research and Development Program Fund（2017YFB0202702）；

作者：lifang, Female, Ph.D., Associate Researcher, main research direction for high performance parallel algorithms and applications,

通信作者：李志辉（zhli0097@x263.net），

difficulty. Adaptation to many-core architecture of CFD software decides which program language should be used. Firstly, the many-core architecture, program model and program languages are briefly introduced. And then challenges on the adaptation of CFD software are analyzed, including data relativity of implicit method, solving of big parse linear equations, many grid method and unstructured grids. For each challenge, corresponding countermeasure is provided. At last, the speedup ratio of some typical software of fluid dynamics is provided based on theory analysis and experiments. Facts prove that most CFD software adapt well to domestic many-core architecture and can use simple program language to get better parallel ration on million cores.
Keywords Domestic, Many-core architecture, Software of computer fluid dynamics, Adaptation, Program language, Parallel algorithm

无锡超算国产超级计算机系统[1]是世界上首台运算速度超过十亿亿次的超级计算机系统，系统全部采用众核异构处理器 SW26010 构建，运算速度连续 4 次在全球 TOP500 中排名第一[2]。由于受功耗等因素的制约，众核异构处理器逐渐成为高性能计算机的主流架构，传统的 MPI+OpenMP 并行编程模型无法满足新型计算模式的需求，现有程序毫无例外地需要进行众核级并行算法设计和优化才能有效地发挥高性能计算机系统的超强计算能力[3-7]。

计算流体力学是高性能计算的传统应用领域，在十亿亿次无锡超算系统上，借助高性能计算机硬件性能的大幅提升和并行算法的进步，一些高精度、跨流域、多尺度、多物理过程耦合的流体力学软件逐渐从科学研究阶段向工程应用阶段迈进。“航天飞行器全流域数值模拟”“三维水下航行体绕流数值模拟”“可压缩边界层湍流及转捩模拟”和“高超声速内外流数值模拟”等课题分别采用 GKUFS，SWLBM，OpenCFD 和 AHL3D 软件达到百万至千万核并行规模，并保持着较高的并行效率。这些具有自主知识产权的自研软件在国产高性能计算机系统上的成功部署和大规模运行，对新型流体力学数值方法的应用以及流体力学软件向快速选型、精细模拟等方面发展起到了巨大的推动作用[8-11]。

流体力学软件由于求解方法、网格类型、数值格式等差别，对众核架构的适应性不同，采用的编程语言和众核并行算法亦不尽相同，本文将对这些问题展开讨论。首先介绍无锡超算国产众核处理器体系结构、编程模型和两种众核级并行编程语言，这两种众核级编程语言的实现复杂度相差较大，选择不同编程语言对应用程序移植周期和优化效果有不同影响；然后分析无锡超算上运行的流体力学软件在众核处理器上遇到的适应性难题，这些难题的存在限制了软件对众核架构的适应性，需要研究创新性的优化算法，手动改写众核程序并反复优化；最后从求解方法、网格类型、数值格式等方面分析一些典型流体力学软件的众核适应性，给出数值实验结果并计算出大规模并行效率。本文的工作将为流体力学软件在国产众核处理器以及其他异构处理器上的移植、优化以及大规模并行提供参考。

1　众核处理器的体系结构及编程模型

1.1　众核处理器架构

无锡超算国产高性能计算机系统基于众核处理器构建（如图 1 所示），一个众核处理器由 4 个异构群（CG）构成，每个异构群包括一个主核（MPE）、64 个从核（CPE）构成的从核簇、异构群接口和存储控制器，整个芯片共有 260 个计算核心。主核即控制核心，除了具备计算、通信和 I/O 等常规多核 CPU 的功能外，还负责从核任务的加载与回收等控制操作。从核即计算核心，主要负责细粒度并行的计算任务。从核可以直接离散访问主存，也可以通过 DMA 方式批量访问主存，从核阵列内可以通过寄存器通信方式进行高效通信[12]。

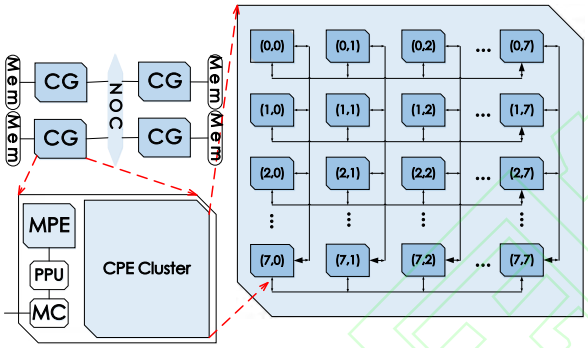


图 1 众核处理器架构

Fig. 1 Architecture of many-core

1.2 编程模型和编程语言

大部分应用课题都采用两级并行方式，即 MPI 进程级并行和众核线程级并行。众核线程级并行的主要实现方式是主从加速并行（如图 2 所示），应用课题通常按任务类型将各个计算模块划分为不同的核心段，计算集中的核心段加载到从核进行加速，不可众核加速的核心段（包括计算、通信、I/O 等）留给主核处理[13-14]。

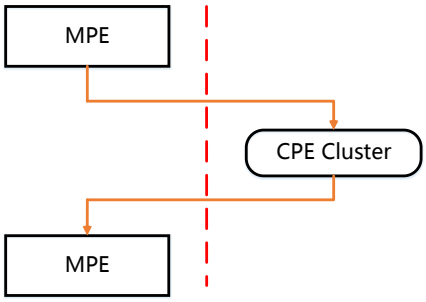


图 2 主从加速编程模型

Fig. 2 Master-slave acceleration model

无锡超算国产高性能计算机系统提供两种众核级并行编程语言：OpenACC 和加速线程库（Athread 接口）。OpenACC[15-16]是一种通用的基于编译指示的众核加速编程标准，它文本简单，语义清晰，用户只需在需要加速执行的代码区域添加 OpenACC 编译指示语句，OpenACC 编译器即可配合硬件系统自动完成从核计算任务的加载/回收及数据在主/从核之间的传输等工作，大大减小了用户编写众核并行程序的困难和工作量。加速线程库[8]是相较于 OpenACC 更底层的接口，采用加速线程库进行众核级并行需要对所有进行从核加速的核心函数手动实现主核与从核两部分代码，并且需要根据从核 id 计算每个从核应该分配的计算任务，手动进行任务划分。然而，利用加速线程库可以对任务进行更细致的划分和控制，进一步重构算法，从而提高其性能，也可以使用底层硬件接口（如寄存器通信、数据置换、指令重排等）对算法进行深度优化。

流体力学软件的多数核心段能够使用 OpenACC 编译器向众核平台移植，并取得较好的加速效果。本文着重讨论流体力学软件中遇到的特殊疑难问题。由于算法具有复杂性，针对这些问题，需要结合众核架构的特点设计高效的众核并行算法，采用加速线程库手工改写众核程序并反复优化，

才能充分发挥众核处理器的计算性能，从而获得较优的加速效果。

2 流体力学软件众核适应性的难点问题

2.1 隐格式带来的数据相关性问题

隐式算法由于具有稳定性好、收敛速度快等优点，在流体力学软件中被广泛应用。三维流体力学控制方程（纳维-斯托克斯方程）采用隐式算法对微分方程进行离散后，得到如下大型稀疏矩阵线性代数方程组：

$$\left[1+\Delta t\left(\rho(\tilde{A})+\rho(\tilde{B})+\rho(\tilde{C})\right)+\Delta t D^{n}_{i,j,k}\right]\delta\tilde{Q}^{n+1}_{i,j,k}$$
$$+\Delta t\left(\tilde{A}^{-}_{i+1,j,k}\delta\tilde{Q}^{n+1}_{i+1,j,k}+\tilde{B}^{-}_{i,j+1,k}\delta\tilde{Q}^{n+1}_{i,j+1,k}+\tilde{C}^{-}_{i,j,k+1}\delta\tilde{Q}^{n+1}_{i,j,k+1}\right)$$
$$-\Delta t\left(\tilde{A}^{+}_{i-1,j,k}\delta\tilde{Q}^{n+1}_{i-1,j,k}+\tilde{B}^{+}_{i,j-1,k}\delta\tilde{Q}^{n+1}_{i,j-1,k}+\tilde{C}^{+}_{i,j,k-1}\delta\tilde{Q}^{n+1}_{i,j,k-1}\right)$$
$$=\Delta t\cdot RHS \qquad (1)$$

LU-SGS 隐式格式被成功用于方程(1)的求解，其基本思想是运用通量线化假设和最大特征值方法进行雅可比矩阵分裂，把块对角矩阵分解为上、下两个三角矩阵，从而避免了繁杂的矩阵求逆运算，极大地提高了计算效率。LU-SGS 方法分为两步扫描。

第一步：

$$\left[1+\Delta t\left(\rho(\tilde{A})+\rho(\tilde{B})+\rho(\tilde{C})\right)+\Delta t D^{n}\right]_{i,j,k}\delta\bar{Q}^{n+1}_{i,j,k}$$
$$-\Delta t\left(\tilde{A}^{+}_{i-1,j,k}\delta\bar{Q}^{n+1}_{i-1,j,k}+\tilde{B}^{+}_{i,j-1,k}\delta\bar{Q}^{n+1}_{i,j-1,k}+\tilde{C}^{+}_{i,j,k-1}\delta\bar{Q}^{n+1}_{i,j,k-1}\right)$$
$$=\Delta t\cdot RHS \qquad (2)$$

第二步：

$$\left[1+\Delta t\left(\rho(\tilde{A})+\rho(\tilde{B})+\rho(\tilde{C})\right)\right]_{i,j,k}\delta\tilde{Q}^{n+1}_{i,j,k}$$
$$+\Delta t(\tilde{A}^{-}_{i+1,j,k}\delta\tilde{Q}^{n+1}_{i+1,j,k}+\tilde{B}^{-}_{i,j+1,k}\delta\tilde{Q}^{n+1}_{i,j+1,k}+\tilde{C}^{-}_{i,j,k+1}\delta\tilde{Q}^{n+1}_{i,j,k+1})$$
$$=\delta\bar{Q}^{n+1}_{i,j,k} \qquad (3)$$

式(2)和式(3)是递归方程，给定边界条件后，采用递推方法对其进行求解。递推方法本质上属于一种串行方法，计算过程须遵循严格的计算顺序，计算之间存在极强的数据相关性。流水线（Pipelining）并行是解决递推问题的常用并行技术。国产众核处理器在体系结构设计上有一项重大创新，即支持从核阵列内部寄存器通信，这就给从核阵列内流水线的并行提供了基础。

假设三维数据在内存中沿 *i* 方向连续，那么将 *j*，*k* 两维映射到从核阵列上，每个从核计算对应 *i* 维的全部数据，数据划分如图 3 所示。

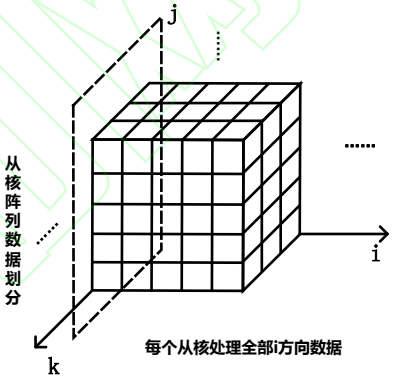


图 3 从核阵列的数据划分

Fig. 3 Data partition on CPE cluster

在三维流水线并行中，数据在从核上二维映射，其流水时从核间的通信关系如图 4 所示。不连续的两维数据映射到从核阵列，由于从核阵列是 8*8 的二维 mesh 网，阵列的行列与数组划分的两个维度之间的对应关系不影响计算结果与性能。一般地，在 8×8 的从核阵列中，水平方向映射最外层循环， 垂直方向映射次外层循环。在水平方向上，仅当当前从核计算数据的水平坐标小于总数据对应维度数据的大小时向右发送数据；在垂直方向上，仅当当前从核计算数据的垂直坐标小于总数据对应维度数据的大小时向下发送数据。对于第 7 行和第 7 列的从核，若不是第一次循环，需要把结果发送

给第 0 行和第 0 列。

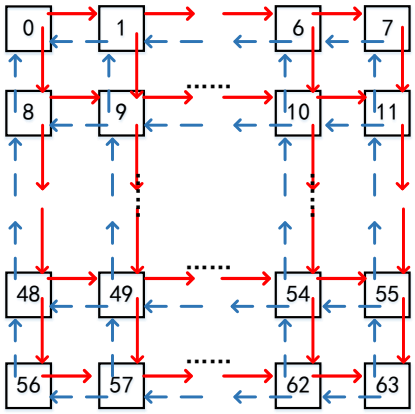


图 4 从核阵列的流水线并行

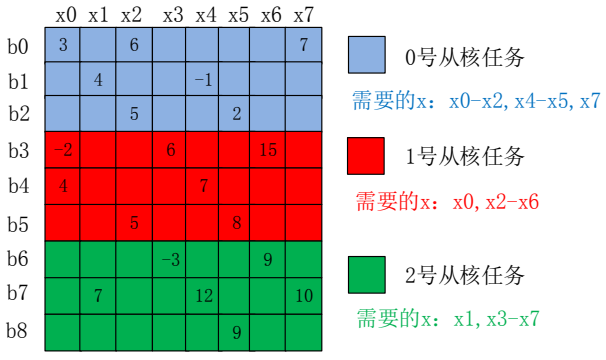
Fig. 4 Pipeline parallel on CPE cluster

2.2 大型稀疏矩阵线性代数方程组的求解

很多计算流体力学软件采用迭代法求解线性代数方程组，涉及到的操作包括大型稀疏矩阵矢量乘(SpMV)、向量点积(DOT)、标量向量乘(AXPY)、预条件子共轭梯度迭代（PCG）、对称 Gauss-Seidel（SymGS）等。这些操作具有较低的计算密度、不规则的访存模式和受限于访存带宽的缺陷。相比于传统的多核 CPU，众核处理器通过集成更多的轻量级的计算核心和更宽的向量处理单元获得高性能，但同时也加剧了带宽与计算能力的不匹配。因此，众核架构往往擅长处理稠密矩阵计算等计算密集型任务，大型稀疏矩阵操作对其来说具有挑战性[17-18]。

本文以稀疏矩阵矢量乘法（Sparse Matrix-Vector Multiplication，SpMv）为例来介绍一种众核优化算法。稀疏矩阵向量乘是数值计算中非常重要的一个核心函数，用于计算 $\boldsymbol{b}=\boldsymbol{Ax}$，其中 $\boldsymbol{A}$ 是稀疏矩阵，$\boldsymbol{x}$ 是已知向量，$\boldsymbol{b}$ 是结果向量。稀疏矩阵中的非零元极其稀少，稀疏行压缩（Compress Sparse Row，CSR）格式通过保存稀疏矩阵中每行包含的非零元数量对行坐标进行压缩，需要保存行索引（Row_p）、列坐标（Col_i）和非零元值（value）。

针对国产众核处理器，稀疏矩阵矢量乘众核并行的原理如图 5 所示。按向量 $\boldsymbol{b}$ 在从核阵列上进行任务划分（避免写冲突），由于矩阵 $\boldsymbol{A}$ 采用 CSR 格式存储，每个从核需要的矩阵元素在内存中连续存放，能够通过 DMA 操作连续访问，读效率较高；困难在于向量 $\boldsymbol{x}$ 不连续，但对于计算流体力学来说，这种不连续又呈现出局部集中性的特点[19]。网格生成算法有规律可循，编号上连续的网格单元(网格面）所能够影响的网格面（网格单元）在物理空间上是相邻的，这种相邻性反映到矩阵拓扑关系上，即计算某些 $\boldsymbol{b}$ 向量所需的 $\boldsymbol{x}$ 向量呈现局部集中性。因此，每个从核需要用到的 $\boldsymbol{x}$ 向量可按照局部集中性分组连续读取，这就需要在预处理阶段根据从核阵列任务划分的计算量对 $\boldsymbol{x}$ 向量分组，此过程在整个程序运行过程中仅需做一次，不影响众核加速效果。



Row_p=[0, 3, 5, 7, 10, 12, 14, 17, 19]

Cow_i=[0, 2, 7, 1, 4, 2, 5, 0, 3, 6, 0, 4, 2, 5, 3, 6, 1, 4, 7, 5]

value=[3, 6, 7, 4, −1, 5, 2, −2, 6, 15, 4, 7, 5, 8, −3, 9, 7, 12, 10, 9]

图 5 稀疏矩阵矢量乘众核并行的原理

Fig. 5 Data locate optimization of matrix

2.3 多重网格方法

多重网格（Multi Grid，MG）算法将细网格上频率变化较缓慢的低频分量映射到粗网格上来处

理，从而达到加速收敛的目的。MG 方法在流体力学软件，特别是不可压 NS 方程求解中得到广泛应用。V-cycle 多重网格的原理如图 6 所示，在原始网格的基础上生成一系列较粗网格，先从细网格逐层计算到粗网格，然后从粗网格返回细网格，在各层网格上分别求解各自的线性方程组，整个计算过程呈 V 字型。

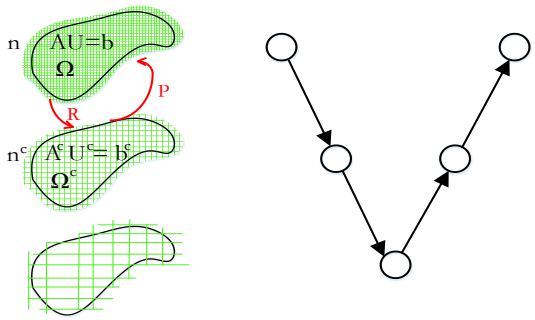


图 6 多重网格的原理

Fig. 6 Priciple of multi grid

算法 1 展示了 MG 算法的具体过程，包括前磨光（Pre-smothing）、后磨光（Post-smothing）、限制算子（Residual）、插值算子（Prolongation）和底部解法器（Bottom solver）。多重网格的内部操作包括向量数乘加、向量内积、稀疏矩阵向量乘、解方程等。

算法 1　MG 算法

输入：A_h, b^h

输出：x^h

1. if on the coarsest level then
2. $x^h \leftarrow SymGS(A_h, 0, b^h)$
3. else
4. $x^h \leftarrow SymGS(A_h, 0, b^h)$
5. $r^h \leftarrow (b^h - A_h x^h)$
6. $r^{2h} \leftarrow (I_h^{2h} r^h)$
7. $r^h \leftarrow MG(A_{2h}, 0, r^{2h})$
8. $x^h \leftarrow x^h + I_{2h}^h x^{2h}$
9. $x^h \leftarrow SymGS(A_h, x^h, b^h)$
10. end

无论在多核还是在众核架构上，实现高效的多重网格算法均充满挑战[20]。首先，多重网格意味着需要针对每一重网格分别建立各自的通信关系，因此通信结构会更加复杂；其次，随着网格逐渐变粗，通信和计算的比例增加，不利于大规模并行扩展；再者，并行任务分解根据原始网格划分，实际计算过程中网格规模呈 V 字型变化，这意味着核心计算的计算量、通信量和访存量也在不断做 V 字形变化，从而给并行程序调优带来困难。对于众核处理器，需要对不同的网格层次建立自适应并行算法。对于细网格层，网格规模较大，需将网格分组，并以组为单位通过 DMA 方式批量访问主存；对于粗网格层，网格规模较小，可将所需网格全部放入从核阵列，然后通过寄存器通信方式读取所需数据。

2.4 非结构网格

非结构网格没有结构网格的规则性限制，网格生成灵活方便。随着数值模拟问题复杂度的增加，越来越多的流体力学软件采用非结构网格对计算区域进行离散[21]。

非结构网格的大规模并行至少存在三方面困难。1）通信不均衡性。并行任务划分是基于网格数进行的，计算量可以达到负载均衡，但非结构网格的邻居进程数量可以很多（针对某复杂问题，一个进程与近百个邻居进程通信），也可以很少，进程间的通信量可能相差很大，并且随着并行规模的增长，这种通信不均衡性会更加严重。2）非结构网格间的拓扑关系复杂。并行计算需要建立各进程内网格单元、网格面和网格点信息以及进程间网格单元的通信关系，每个核心段的计算都要同时读取这些拓扑信息。对于众核架构来说，从核的局部存储空间非常有限，大量的拓扑信息占用了宝贵的存储资

源，这必然会减少从核每次加载的任务量，导致从核计算/访存比降低，加速效果变差。3）数据存放的无序性导致内存访问的随机性，使得从核很难直接通过 DMA 方式批量访问主存，不可避免地需要离散访存，从而导致从核性能下降。前两个问题比较复杂，本文不做阐述，下面为离散访存问题提供一种解决途径。

非结构网格通量计算是有限体积法离散微分方程的基本操作，属于典型的离散访存问题。流经某网格单元的物理量（如压力、速度等）通量由流动速率与该物理量的网格面矢量相乘得到，计算公式如下：

$$J_f^{\phi} \bullet S_f = FluxC_f \phi_C + FluxF_f \phi_F + FluxV_f \quad (4)$$

非结构网格通量计算的伪代码形式如下：

```
1.  for face←0 to n_Faces do
2.      Diag[Owner[face]] -= Lower[face]
3.      Diag[Neighbor[face]] -= Upper[face]
4.  endfor
```

对网格面 face 进行遍历，将该网格面的通量值分别累加到对应的 Owner 网格单元与 Neighbor 网格单元之中。网格单元号随网格面号变化呈现不连续性，造成读变量连续访存而写变量离散访存。

借鉴生产者-路由-消费者通信模式[22]，核组内一组从核负责连续读和计算，一组从核开辟连续空间来存储本核组的所有计算结果，二者之间基于一组路由从核，采用高效的寄存器通信方式转发计算结果。最终，记录结果的从核通过 DMA 将数据连续写回主存。从核阵列连续写的过程如图 7 所示。

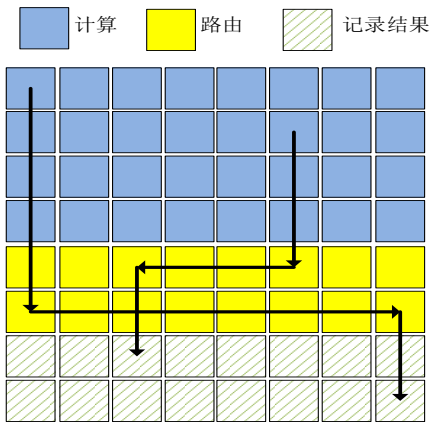
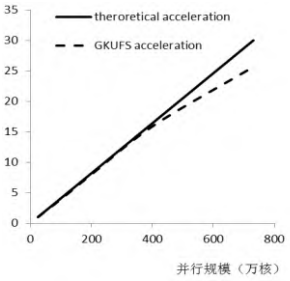


图 7 从核阵列连续写

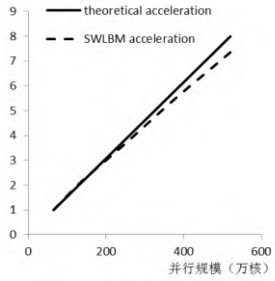
Fig.7 Continuous write on CPE cluster

3 典型流体力学软件的众核适应性

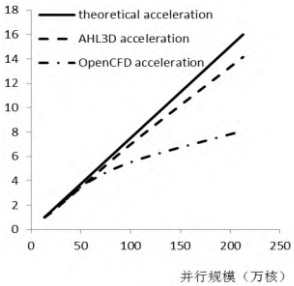
由于求解方程、适用物理过程、求解方法、网格类型、数值格式等不同，各流体力学软件对众核处理器架构的适应性有所差别，部分算法具有天然的适应性，部分算法存在一定的适应性难题。几款典型流体力学软件对国产高性能计算机众核的适应性如表 1 所列，实测大规模并行加速比如图 8 所示。



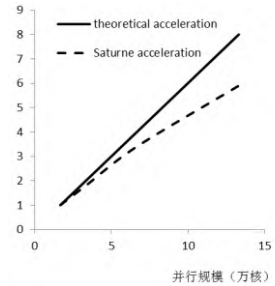
图（a）GKUFS



图（b）SWLBM



图（c）AHL3D 和 OpenCFD



图（d）Code_Saturne

图 8 大规模并行加速比

Fig.8 Acceleration ratio of massive parallel

表 1 CFD 软件的众核适应性

Table 1　Adaptation to Many-core of CFD Softwares

软件	求解方程	物理过程	离散方法	网格	解法	适应性
GKUFS	Boltzmann	全流域 超音速	有限体积	结构	显式	适应
SWLBM	Boltzmann	低速 弱可压流	碰撞/迁移法 （LBM）	结构	显式	适应
AHL3D	NS	超音速 可压流	有限体积	结构	隐式	适应
OpenCFD	NS	可压流	高精度差分	结构	显式	适应
Code_Saturne	NS	低速 弱可压流	有限体积	非结构	隐式 MG/PCG	较差

GKUFS[23-24]是国家计算流体力学实验室开发的一款基于气体动理论的计算流体软件，适用于从稀薄气体到连续稠密气体跨流域超声速流场的模拟。该软件基于速度空间和位置空间两相空间对计算区域进行离散，各离散速度坐标点之间具有非常好的并行独立性，各进程间几乎无数据通信；由于求解高超音速问题，能够在速度空间获得大量的任务并发，并行可扩展性很好；在众核级并行方面，在三维位置空间的最外层循环上划分任务，以保证每个计算核心计算任务的饱满，且线程调度、创建和结束的开销少；采用结构网格，数据在内存中的排列有序，从核访存效率较高。总之，GKUFS 在算法和求解问题上具有特殊优势，计算/访存比和计算/通信比均较高，对众核处理器的适应性很好。从图 8（a）可看出，即使在 731 万核规模下，GKUFS 的并行效率仍高达 85%。

SWLBM 是由中国船舶科学研究中心开发的一款基于格子玻尔兹曼方法（LBM）的计算流体软件，适用于弱可压非定常流动模拟。与 GKUFS 类似，SWLBM 也以粒子速度分布函数作为基本求解物理，求解 Boltzmann 方程；不同的是，SWLBM 通过碰撞迁移来模拟流体流动，计算更简单。SWLBM 采用直角笛卡尔网格（结构网格的特殊形式）显式求解，访存效率高，对众核处理器的适应性很好。经过通信与计算隐藏优化后，SWLBM 的并行可扩展性得到大幅提升。从图 9（b）可看出，在 520 万核规模下，SWLBM 的并行效率高达 91%。

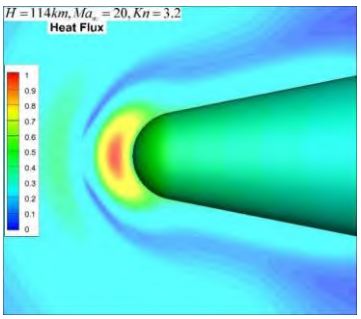
AHL3D 是中国空气动力研究发展中心开发的一款超燃冲压发动机模拟软件。该软件采用多块结构网格离散计算区域，由于自动分块算法先进，多数算例的MPI进程级并行的负载平衡率达到95%以上；采用有限体积法求解带化学反应和湍流的 N-S 方程，各变量统一求解，求解完成后所有变量一次边缘通信，计算/通信效率比较高；结构网格的数据结构简单，数据在内存中连续存储，众核级并行通过 DMA 方式批量访问主存，计算/访存效率比较高。AHL3D 采用隐格式进行时间推进，在 LU-SGS 模块中需要手动实现众核级流水线的并行。从图 9（c）可看出，AHL3D 在 213 万核规模下的并行效率高达 87%，该软件的并行可扩展性以及对众核处理器的适应性均较好。

OpenCFD 是中国科学院力学研究所开发的一款基于高阶有限差分格式的高精度计算流体软件。该软件采用结构网格离散计算区域，数据在内存中连续存储，有利于众核优化；采用显格式进行时间推进，整个计算只有在求 *x* 方向的导数时需要边缘通信，其他计算（包括流通矢量分裂、计算 y 方向和 z 方向的导数、时间推进）过程中各进程间无需交换数据，计算/通信效率比较高；采用直接数值模拟方法模拟湍流流动，网格量达到 5.52 亿，计算量大，有利于提高软件的并行规模。该课题的计算/访存比和计算/通信比较高，软件的并行可扩展性以及对众核处理器的适应性均较好。从图 9（c）可看

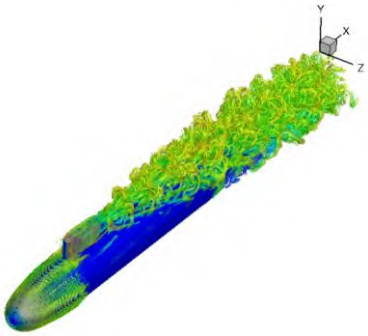
出，OpenCFD 在 213 万核规模下的并行效率达到 50%。

Code_Saturne 是法国电力集团开发的一款通用型开源 CFD 软件。测试例子采用的 Simple 算法是不可压 NS 方程求解的一种常用算法，在计算流体力学软件中具有代表性。该算法的计算热点在于压力修正方程的迭代收敛。由于压力修正方程离散后得到的稀疏矩阵收敛特性较差，Code_Saturne 采用算法复杂度较高的预条件子 CG 迭代（PCG）方法进行求解；为了加快收敛，使用多重网格算法；同时采用非结构网格离散计算区域。由此可见，前文提出的众核处理器适应性的难点问题在于“大型稀疏矩阵线性代数方程组求解”“多重网格算法”“非结构网格”在 Code_Saturne 中均存在，因此 Code_Saturne 算法本身对众核架构的适应性较差。从图 9（d）中可看出，目前 Code_Saturne 众核版本在 13 万核规模下的并行效率可达 75%。与本文分析的其他流体力学软件相比，Code_Saturne 受算法本身可扩展性和众核架构适应性的限制，无论 MPI 进程级并行还是众核线程级并行，尚有较大的优化和提升空间，需要继续改进并行算法，并反复优化。

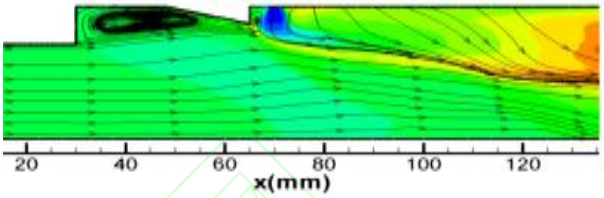
以上几款流体力学软件在国产高性能计算机上的大规模并行模拟结果如图 9 所示。



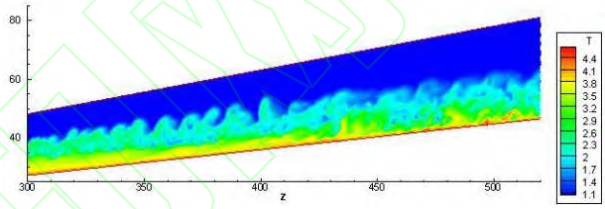
（a）再入飞行器绕流（GKUFS）



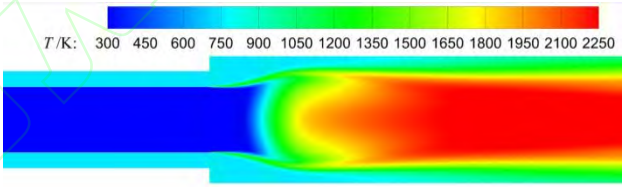
（b）Suboff 绕流（SWLBM）



（c）Taha 燃烧室内流场（AHL3D）



（d）球锥边界层（OpenCFD）



（e）同轴燃烧室内流场（Saturne）

图 9 大规模并行模拟结果

Fig.9 Simulation result of massive parallel

结束语近年来高性能计算机硬件性能大幅度提升，给科学与工程计算行业带来了巨大的机遇，但面对复杂的众核异构处理器架构，流体力学软件须进行众核级并行算法设计和优化才能发挥出硬件的高性能优势。本文结合无锡超算上运行的流体力学软件分析了它们在众核处理器上遇到的适应性难题，并分别针对这些问题提出了创新的优化算法。实践证明，多数流体力学软件对国产众核处理器的适应性良好，能够扩展到百万核并行规模，并保持较高的并行效率。

流体力学软件对众核架构的改造在持续进行，一方面高性能计算厂家会针对用户需求不断完善软硬件的生态环境，另一方面众核并行算法和流体力学应用算法也在不断进步，相信会有更多的流体力学软件加入国产众核高性能计算的行列，并取得越

来越多的科研成果。

参考文献

[1] ZHENG F, LI H L, LV H, et al. Cooperative computing techniques for a deeply fused and heterogeneous many-core processor architecture[J]. Journal of Computer Science and Technology, 2015, 30(1):145-162.

[2] FU H H, LIAO J F, YANG J Z, et al. The Sunway Taihulight supercomputer: system and applications[J]. Science China Information Sciences, 2016,59(7):72-91.

[3] YANG C, XUE W, FU H H, et al. 10m-core scalable fully- implicit solver for nonhydrostatic atmospheric dynamics[C]// Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE,2016: 6-15.

[4] ZHANG J, ZHOU C B, WANG Y G, et al. Extreme-Scale Phase Field Simulations of Coarsening Dynamics on the Sunway TaihuLight Supercomputer [C]// International Conference for High Performance Computing, Networking, Storage and Analysis.IEEE, 2016: 34–45.

[5] FU H H, XUE W, YANG C, et al. Redesigning CAM-SE for Peta-Scale Climate Modeling Performance on Sunway TaihuLight [C]// High Performance Computing, Networking, Storage and Analysis.IEEE, 2017: 4–12.

[6] FU H H, LIAO J F, YANG J Z, et al. 15-Pflops Nonlinear Earthquake Simulation on Sunway TaihuLight: Enabling Depiction of Realistic 10 Hz Scenarios[C]//High Performance Computing, Networking, Storage and Analysis.IEEE, 2017: 102–117.

[7] QIAO F L, ZHAO W, YIN X Q, et al. A highly effective global surface wave numerical simulation with ultra-high resolution[C]// High Performance Computing, Networking, Storage and Analysis.IEEE, 2016: 46–56.

[8] HOU C F, XU J, WANG P, et al. Efficient GPU-accelerated molecular dynamics simulation of solid covalent crystals[J]. MOLECULAR SIMULATION, 2012, 38(1):8-15.

[9] HOU C F, XU J, WANG P, et al. Petascale molecular dynamics simulation of crystalline silicon on Tianhe-1A[J]. International Journal of High Performance Computing Applications, 2013, 27(3):307-317.

[10] LI D, XU Z M, LI S, et al. A survey on information diffusion in online social networks [J]. Chinese Journal of Computers, 2014, 37(1):189-206 .(in Chinese)
李栋, 徐志明, 李生, 等. 在线社会网络中信息扩散[J]. 计算机学报, 2014, 37(1): 189-206.

[11] LIN H, TANG X C, YU B W,et al. Scalable Graph Traversal on Sunway TaihuLight with Ten Million Cores[C]// 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE Computer Society, 2017.

[12] LIN J, XU Z G, Nukada A. Optimizations of Two Compute-bound Scientific Kernels on SW26010 Many-core Processor[C]// Proceedings of the 46th International Conference on Parallel Processing.IEEE,2017.

[13] XU Z G, Lin J,Matsuoka S. Benchmarking Sunway SW26010 Manycore Processor[C]// Proceedings of The Seventh International Workshop on Accelerators and Hybrid Exascale Systems (AsHES) (IPDPS workshop). Orlando, USA,2017

[14] AN H. Pipelining Computation and Data Reuse Strategies for Scaling GROMACS on the Sunway Many-core Processor[C]// 18th International Conference on Algorithms and Architectures for Parallel Processing(ICA3PP-2018).IEEE,2018.

[15] YOU H T，ZHANG L B，MAO Z H. OpenACC2.0 VS OpenMP4.0 Comparation of Two Popular Programming Language Based on Compilation Instructions[J] .High Performance Computing, 2014, 227: 20-25.（in Chiniese）
尤洪涛，张立博，毛智辉. OpenACC2.0 VS OpenMP4.0:基于编译指示的两种主流众核编程语言的对比研究[J]. 高性能计算技术，2014，227:20-25.

[16] 何沧平. OpenACC 并行编程实战[M].北京：机械工业出版社,2016
[17] Liao J F. Redesigning CAM-SE for Peta-Scale Climate Modeling Performance on Sunway TaihuLight [D].Beijing: Tsinghua University,2017.（in Chinese）
廖俊峰，面向十亿亿次国产众核超级计算机的大气模式重构与优化 [D]. 北京：清华大学，2017.
[18] Ao Y L.Research on Key Optimizations of Sparse Matrix and Stencil Computation for the Domestic Large Many-core System[D].Beijing: University of Chinese Academy of Sciences,2017.（in Chinese）
敖玉龙，国产大型众核系统上稀疏矩阵和 Stencil 运算的性能优化关键技术研究[D]. 北京：中国科学院大学，2017
[19] Ni H. Research on Heterogeneous parallel computing technology of CFD in unstructured grids[D]. The 56th Institute of PLA,2018.（in Chinese）
倪鸿，非结构网格 CFD 异构并行计算技术研究[D] .战略支援部队第五十六研究所，2018
[20] Li Z Z. Research on parallel multi grid of unstructured grids[D]. Changsha: National University of Defense Technology ,2012（in Chinese）
李宗哲，非结构网格的并行多重网格算法研究[D].长沙：国防科学技术大学，2012
[21] Delong Meng, Minhua Wen, Jianwen Wei. Hybrid Implementation and Optimization of OpenFOAM on the SW26010 Many-core Processor. HPC China 2016
[22] Lin H. Extreme-scale graph analysis on heterogeneous architecture[D].Beijing: : Tsinghua University,2017（in Chinese）
林恒.基于超大规模异构体系结构的图计算系统研究[D]. 北京：清华大学，2017.
[23] Xu J X，You H T. Application of Many-core Programming Language OpenACC in Solving of Boltzmann Equations[J]. High Performance Computing, 2016(2): 7-12（in Chiniese）
徐金秀，尤洪涛. OpenACC*众核编程语言在求解 Boltzmann 模型方程的应用研究[J]. 高性能计算技术，2016(2)：7-12
[24] Li Z H，Zhang H X. Parallel Computing of Three Dimension Complex Gas Motion Flow[J]. Journal of Aerodynamics , 2010,28(1): 7-16.（in Chiniese）
李志辉，张涵信. 跨流域三维复杂绕流问题的气体运动论并行计算[J].空气动力学学报，2010,28(1): 7-16.