有限元网格积分算法在MIC众核平台上的并行实现

寇大治[1] 孔大力[2]
(上海超级计算中心 上海 201203)[1] (埃克塞特大学数学系 埃克塞特 EX4 4QF)[2]

摘 要 基于英特尔集成众核(Many Integrated Core,MIC)架构,将有限元网格积分算法在至强融核(Xeon Phi)协处理器做了移植和性能分析。该应用全面测试了有限元分析的核心计算过程在MIC上的加速效果,实现了卸载模式(offload)[1]下利用OpenMP在MIC上的线程并行化。计算性能测试结果显示集成众核平台可以有效地加速有限元网格积分算法:1)一块被充分利用的MIC设备卡(3115A)的计算能力超过两路16核Intel Xeon™ E5-2670 CPU;2)MIC并发的物理线程可能由于公共缓存访问存在竞争而降低程序的扩展性。测试结果还显示了在多CPU多MIC平台上进一步移植完整的MPI并行有限元模拟软件的可行性。这项工作有助于推动与有限元网格相关的科学和工程高性能计算的研究。

关键词 集成众核,卸载模式,并行,多线程,有限元
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.11.010

Parallel Implementation of Finite-element Mesh Integration Algorithm on Many Integrated Core

KOU Da-zhi[1] KONG Da-li[2]
(Shanghai Supercomputer Center,Shanghai 201203,China)[1]
(Department of Mathematics,University of Exeter,Exeter EX4 4QF,UK)[2]

Abstract A C++ 3-D finite-element mesh integration algorithm was implemented and profiled on a heterogeneous Intel CPU/MIC architecture. By virtually programing in the offload mode[1] with explicit copies, a sequence of key element-wise operations are fully parallelized utilizing massive concurrency of OpenMP threads on MIC devices. It is remarkably demonstrated that, in the sense of overall run-time efficiency, one fully employed 3115A MIC card outweighs two 8-core Intel Xeon™ E5-2670 CPUs. However, possibly owing to cache contention among physical threads on individual MIC core, scalability is somehow below an ideal level. Current test unveils a good chance of transplanting a full finite-element analysis code onto a multi-CPU nodes/multi-MIC devices platform based on this single-process multi-thread building block presented here.

Keywords Many integrated core, Offload mode, Parallel, Multi-threads, Finite element

1 引言

从计算的角度来说,大规模三维有限元模拟[2]的实现包含两个核心过程:1)大型稀疏线性系统的并行构造、存储与组装;2)大规模预条件线性方程组的高效并行求解。在两个过程中,都存在有限元局部与全局运算交替的现象。局部运算是指在网格中每个单元内部进行的小规模几何与代数数据计算不参与全局进程之间的通信且在单元之间彼此独立;而全局运算则是指在整个网格和全局线性系统层面的有关计算需要大量进程间的数据交换。当程序完全部署在多CPU核心的同构架构中时,整个有限元网格按照并发进程数被划分为多个子网格,然而子网格内的局部运算一般得不到线程级别的再并行,而只能以串行方式在各个进程中运行。这种粗粒度的并行无法使得整个程序得到最充分的并行化;而若引进GPU异构架构[3,4],协处理器的工作方式造成所要求的并行粒度过细,导致在CPU端产生大量的额外数据准备与PCI-E数据传输开销,这对于浮点数运算中并不密集的局部运算来说是得不偿失的。因此,GPU主要在全局线性方程组并行求解的过程中提供部分加速[5]。由于GPU的特殊硬件特性对编程的限制,数值计算界仍然对这种加速方式持谨慎态度。对于Element-by-Element(EBE)技术[6]来说,由于不需显式地构造全局线性系统,所有代数运算都在单元内局部进行,因此GPU无法发挥加速作用。Intel集成众核架构的推出使得我们第一次拥有了与有限元计算相适应的并行粒度层次。单元级别的运算可以通过MIC平台整体实现线程并行,这得益于MIC核对x86指令集的兼容。无论是否采用EBE技术,预期MIC都将有助于实现最经济适用的有限元计算加速方案。本文将借助一个具有行星物理背景的有限元网格数值积

到稿日期:2014-10-23 返修日期:2014-12-15 本文受国家高技术研究发展计划(863)(2012AA01A308),国家自然科学基金(11473014),上海市科学技术委员会科研计划项目(13DZ2294500)资助。
寇大治(1980-),男,硕士,工程师,主要研究方向为高性能计算集群系统、高性能计算在科学计算中的应用,E-mail:dzkou@ssc.net.cn;孔大力(1985-),男,博士,副教授,主要研究方向为计算与理论流体力学、行星科学的高性能计算。

分应用测试 MIC 众核计算平台对单元级别运算的并行加速效果。

从代码组织的角度来说，C++语言在传统科学计算领域内并未被广泛使用，这在很大程度上是因为科学计算程序往往主要重视运行效率而不像商业应用软件对接口、封装、隔离和扩展的高性能要求高。然而正如面向对象编程思想伴随着代码规模的扩大而变得重要，在大规模高性能计算领域，越来越多的科学计算平台开始转向 C++语言。在近期的工作[7,8]中，我们基于 C++平台实现了一个具有清晰继承层次和完善接口设计的有限元分析软件，本文的测试应用以该软件中的部分功能模块为实现基础。由于 C++有许多不同于 C 的特性，众核架构对充分的面向对象编程的支持程度需要小心地检验。这里说的“支持”不仅仅指 MIC 计算平台自身对 C++语言标准的支持，还包括整个 CPU/MIC 异构架构对 C++编程思想的支持。C++语言在 CPU/MIC 异构架构下的行为多是标准中未定义的，Intel 编译器因此拥有完全的决定权，这使得编译器能否完美地实现异构平台面向对象编程成为一个严肃的问题。例如，一般的 C++类对象在 MIC 卸载模式下存在整体卸载困难[9]，这就会一定程度地破坏 C++语言的封装性。目前的 CPU/MIC 混合编程还存在其它一些违背 C++编程思想的行为，这将在下文中得到讨论。

本文第 2 节讨论必要的有限元方法基础知识；第 3 节详细讨论代码的结构和功能；第 4 节给出测试结果与性能分析；最后总结目前的工作并对未来的工作进行展望。

2 有限元网格积分

在有限元方程组的构造、求解和后处理中，都需要用到有限元网格和单元上的形函数插值与高斯积分。这些运算要求一系列几何与代数数据结构的准备。这个测试并不是实现完整的有限元求解过程，而是专注于上述插值和积分运算。我们选择一个引力势积分问题系统来考察众核平台对一系列典型的有限元计算步骤的加速性能。

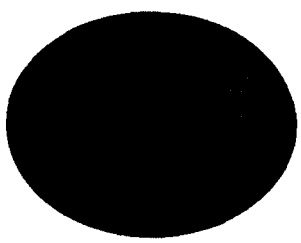

图 1 扁椭球区域有限元网格整体空间剖分

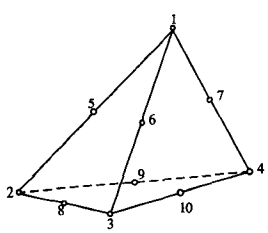

图 2 剖分网格中一个十点二阶四面体单元及单元内节点编号方式

从几何直观的角度，图 1 展示了一个扁椭球区域有限元网格整体空间剖分，在此例的网格中包含约 1 万个四面体单元，图 2 展示了剖分网格中一个十点二阶四面体单元及单元内节点编号方式。在扁椭球区域内的四面体有限元网格剖分，其应用背景是与地球类似的旋转行星。扁椭球形状是旋转离心力与行星自身引力达到平衡时的流体静力学状态[10]。行星准静态的形状和内部物质分布状态可以通过对其外部引力场的测量得到反演[7]。在经典牛顿引力理论中，已知一个孤立区域 $\mathcal{D}$ 中的物质质量密度分布 $\rho(\boldsymbol{r}')$，则可以计算任意场点 $\boldsymbol{r}$ 处的引力势：

$$V_g(\boldsymbol{r})=-G\int_{\mathcal{D}}\frac{\rho(\boldsymbol{r}')\mathrm{d}\tau'}{|\boldsymbol{r}-\boldsymbol{r}'|} \tag{1}$$

其中，$G=6.67\times10^{-11}\ \mathrm{m^3kg^{-1}s^{-2}}$ 是万有引力常数。积分区域的 N 单元网格剖分 $\mathcal{D}=\bigcup_{i=1}^{N}\mathcal{D}_i$，其中 $\mathcal{D}_i$ 是第 i 个四面体单元区域，则引力势积分可以采用单元积分求和归约的方式计算：

$$V_g(\boldsymbol{r})=-G\sum_{i=1}^{N}\int_{\mathcal{D}_i}\frac{\rho(\boldsymbol{r}')\mathrm{d}\tau'}{|\boldsymbol{r}-\boldsymbol{r}'|} \tag{2}$$

可以看到，每个单元内进行的积分操作是完全彼此独立的，整个归约过程可以被充分地并行。对单元积分而言，一般采用体积坐标 $L_i(i=1,2,3,4)$ 下的二阶四点高斯积分。通过 10 个节点上的二阶插值函数：

$$\begin{aligned}&\phi_i=L_i(2L_i-1),i=1,2,3,4,\\&\phi_5=4L_1L_2,\phi_6=4L_1L_3,\phi_7=4L_1L_4,\\&\phi_8=4L_2L_3,\phi_9=4L_2L_4,\phi_{10}=4L_3L_4\end{aligned} \tag{3}$$

可以将节点上的函数值 $\rho_i(i=1,2,3,\cdots,10)$ 内插到高斯积分点上的值 $\rho_n(n=1,2,3,4)$。而高斯点上的 Jacobi 矩阵在已知单元节点直角坐标：

$$\boldsymbol{X}=\begin{bmatrix}x_1&y_1&z_1\\x_2&y_2&z_2\\\vdots&\vdots&\vdots\\x_{10}&y_{10}&z_{10}\end{bmatrix} \tag{4}$$

和高斯点上的插值函数梯度矩阵：

$$\boldsymbol{D}_n=\begin{bmatrix}\frac{\partial\phi_1}{\partial L_1}&\frac{\partial\phi_2}{\partial L_1}&\cdots&\frac{\partial\phi_{10}}{\partial L_1}\\\frac{\partial\phi_1}{\partial L_2}&\frac{\partial\phi_2}{\partial L_2}&\cdots&\frac{\partial\phi_{10}}{\partial L_2}\\\frac{\partial\phi_1}{\partial L_3}&\frac{\partial\phi_2}{\partial L_3}&\cdots&\frac{\partial\phi_{10}}{\partial L_3}\end{bmatrix}_n \tag{5}$$

的情况下，可以通过线性代数计算得到：

$$\boldsymbol{J}_n=\boldsymbol{D}_n\boldsymbol{X}=\begin{bmatrix}\frac{\partial x}{\partial L_1}&\frac{\partial y}{\partial L_1}&\frac{\partial z}{\partial L_1}\\\frac{\partial x}{\partial L_2}&\frac{\partial y}{\partial L_2}&\frac{\partial z}{\partial L_2}\\\frac{\partial x}{\partial L_3}&\frac{\partial y}{\partial L_3}&\frac{\partial z}{\partial L_3}\end{bmatrix}_n \tag{6}$$

这样，第 i 个单元内的高斯积分过程可以写为：

$$\int_{\mathcal{D}_i}\frac{\rho(\boldsymbol{r}')\mathrm{d}\tau'}{|\boldsymbol{r}-\boldsymbol{r}'|}=\sum_{n=1}^{4}w_n|\boldsymbol{J}_n|\frac{\rho_n}{|\boldsymbol{r}-\boldsymbol{r}'_n|} \tag{7}$$

其中，w_n 是给定的积分权重，$\boldsymbol{r}_n'$ 是给定的高斯积分点。

在测试中，利用了 MIC 设备的并发多线程能力来并行计算网格中每个单元上的积分，改变了以往在 CPU 进程中逐个计算的串行局面。所有的插值、积分和矩阵构造都是完整的有限元模拟中所必需的。对这些核心计算过程的移植和优化可以为未来的有限元计算工作提供参考。

3 实现与分析

程序中包含两个主要的数据抽象：一个是网格类(class Mesh)，另一个是单元类(class ElementBase 和 class Element-

PostProcess:public ElementBase)。网格类包含与全局网格有关的几何与物理数据,例如节点和单元编号、节点的坐标、节点上的密度值等。这些数据通过公共接口为类对象的用户提供只读访问。网格类的对象将在 CPU 端使用。而单元类是每个单元上数据与操作的封装。单元类提供两个主要接口,一个是外部传入所需要的数据,用以构造单元内的相关数据;另一个是计算积分的功能接口,为类对象的用户计算场点 r 处引力势的值。单元类的对象将在 MIC 端被构造并使用。需要指出的是,从代码设计的角度来说,网格类应当封装单元类类型的数据成员,从而使得用户(这里指 main 函数)只需面对全局层次的接口。但测试表明这种封装并不适宜 CPU/MIC 架构。由于尚不明确的原因,MIC 的数据卸载方式不能完全正确支持 C++类作用域内的卸载。类的数据成员无法在成员函数局部域内被直接卸载到 MIC 设备端,因此应当避免将一个类作用域横跨于 CPU 和 MIC 两端。为此,单元类对象独立于网格类,通过编译选项"-offload-attribute-target=mic"能确保其正确地在 MIC 端构造、运算。

并行实现采用单元级别的并行方式:CPU 进行全局性的计算、I/O 等,而在 MIC 上对单元级别的计算进行并行化。这正是体现了 MIC 提供的是介于 MPI 并行和 GPU 加速之间的并行粒度。MIC 具有这样的能力,源于它对处理器核数与指令集复杂度的平衡兼顾:众核的特性使得它可以提供大量的并发并行能力;而兼容 x86 处理器架构的指令集使得它可以像 CPU 一样处理完整的复杂指令。

4 结果与讨论

本文测试 CPU 平台的硬件环境采用了 Intel Xeon™ E5-2670,2.6GHz 处理器,属于至强 E5-2600 系列,Sandy Bridge-E 核心,制作工艺 32 纳米,插槽类型为 LGA2011,核心主频为 2.6GHz,三级缓存容量为 20MB,工作功率为 115W,单处理器含八核心十六线程。计算节点为两路处理器,内存为 DDR3\1600MHz 8GB * 8 根,共计 64GB。软件环境包括:SUSE Linux Enterprise Server 11 (x86_64)操作系统,VERSION=11,PATCHLEVEL=2 (kernel:3.0.13—0.27),编译运行环境采用了 Intel 集群组件 Intel 64 Compiler XE Version 13.0.1.117 Build 20121010。测试使用的 MIC 设备硬件和软件相关的信息如表 1 所列。

表 1 MIC 软硬件信息

	3115A
Total No. of Active Cores	57
Frequency	1.1GHz
GDDR Technology	GDDR5
GDDR Size	5952 MB
GDDR Speed	5GT/s
GDDR Vendor	Hynix
MPSS Version	2.1.5889-16
Flash Version	2.1.01.0385
SMC Firmware Version	1.13.4570
SMC Boot Loader Version	1.4.3581
uOS Version	2.6.38.8-g9b2c036
ECC Mode	Enabled
SMC HW Revision	Product 300W Active CS

有限元计算使用了规模足够大的网格(包含 17301504 个单元),使得运行时机器状态变化带来的影响可以基本忽略不计。测试结果如表 2 和图 3 所示。需要指出的是,计时仅针对 MIC 上运行的代码段,不包括 CPU 端的网格文件 I/O 部分。

表 2 测试计算中不同线程数时程序的运行时长

MIC 线程数	积分计算时间(s)
1	571.842
4	153.714
8	72.983
20	31.593
40	16.252
60	12.552
80	9.413
100	9.129
120	9.106
140	8.905
160	8.421
180	8.385
200	8.247
220	7.557

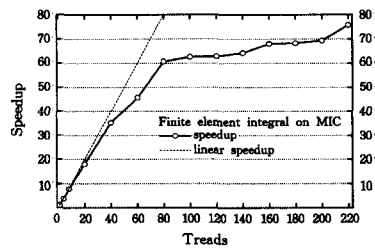

图 3 MIC 上有限元积分程序的加速曲线

可以看到,在线程数达到大约 60 之前,加速曲线基本可以认为是线性的,这对应每个 MIC 核上运行一个物理线程;而在此之后,每个核上有不止一个物理线程并发,加速比大大低于线性。

由于在算法层面,此应用在 MIC 上的并行部分可以做到完全并行化和均衡负载,因此我们期待可以看到优秀的扩展性结果。实测结果在一定程度上支持这个预期。然而,在单 MIC 核中出现多物理线程并发后,程序的加速性能出现下降,这一点在其它应用的测试中也同样如此[9]。我们猜想这个问题是由并发线程出现缓存竞争所致。尽管如此,当使用全部 228 个线程后,最终程序的加速比达到约 80,积分耗时仅约 8s;作为比较,使用双路 16 核 Xeon E5-2670 CPU(主频 2.6GHz)运行相应的纯 CPU 版本的应用,在 16 个并发线程的情况下,耗时约 24s。可见此应用在 MIC 上的加速效果是非常优秀的。

我们初步对此应用的运行时性能进行了分析。结果表明程序的热点集中在单元引力势积分部分,这符合我们的应用的特点;同时,程序运行时的负载均衡性非常好,这也同样符合我们的预期;然而性能分析显示,程序运行时存在较多的 L1 缓存未中和向量化程度不高的缺点。我们认为这仍然是符合本应用特点的。首先,由于有限元网格节点全局编号与单元内局部编号的不一致性,访问全局节点坐标数组(Mesh::Coord)时必然频繁出现不规律和大跨度的指标跳跃,会使缓存未中的情况大大增多。其次,单元内局部计算均是小规模零散的算术或线性代数运算,不具备充分向量化的条件。这也正是 GPU 设备无法适用此类应用的原因。从这两个角度来说,性能分析中指出的问题也是可以理解和接受的。造成该问题的原因不在于算法设计和代码编写质量,而是植根于问题本身的特性。

结束语 本文讨论了有限元网格积分算法应用于 MIC 架构上的移植与性能分析。测试结果基本符合工作的各方面预计。从一方面说,MIC 独特的处理器架构为有限元计算方

(下转第 62 页)

到来时，交换机中无该流相关的转发信息而导致流表不命中造成的。通过分析发现，产生流表不命中的原因与交换机在网络中的位置有关；数据流建立开销对包延时的影响是控制器到沿途交换机的链路延迟及交换机间链路延迟的多元函数；是否会发生 table-miss、发生 table-miss 对时延产生的影响与具体的网络环境配置有关。同时，网络产生的延时与以下因素相关：1)第一跳交换机到控制器的延时；2)控制器及交换机的处理能力；3)交换机在路由路径中的位置。最后，我们通过实验评估了流建立开销对传输延迟造成的影响：数据流的建立致使包传输延迟至少增加 2 倍，对网络性能产生显著影响。

参 考 文 献

[1] Open vSwitch[OL]. http://openvswitch.org/

[2] Curtis A R, Mogul J C, Tourrilhes J, et al. DevoFlow: Scaling flow management for high-performance networks[J]. ACM SIGCOMM Computer Communication Review. ACM, 2011, 41(4): 254-265

[3] Zarek A, Ganjali Y, Lie D. Openflow timeouts demystified[D]. Toronto, Ontario, Canada: Univ. of Toronto, 2012

[4] Kim E D, Lee S I, Choi Y, et al. A flow entry management scheme for reducing controller overhead[C]// 2014 16th International Conference on Advanced Communication Technology (ICACT). IEEE, 2014: 754-757

[5] Benson T, Akella A, Maltz D A. Network traffic characteristics of data centers in the wild[C]// Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. ACM, 2010: 267-280

[6] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74

[7] Beacon controller [OL]. https://openflow.stanford.edu/display/Beacon/Home

[8] Huang D Y, Yocum K, Snoeren A C. High-fidelity switch models for software-defined network emulation[C]// Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, 2013: 43-48

[9] Vishnoi A, Poddar R, Mann V, et al. Effective switch memory management in OpenFlow networks[C]// Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems. ACM, 2014: 177-188

[10] Fundation O N. Software-defined networking: The new norm for networks[R]. ONF White Paper, 2012

[11] Wang A, Guo Y, Hao F, et al. Scotch: Elastically Scaling up SDN Control-Plane using vSwitch based Overlay[C]// Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies. ACM, 2014: 403-414

[12] Yu M, Rexford J, Freedman M J, et al. Scalable flow-based networking with DIFANE[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 351-362

[13] Yan B, Xu Y, Xing H, et al. CAB: a reactive wildcard rule caching system for software-defined networks[C]// Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, 2014: 163-168

(上接第 58 页)

面的应用提供了十分契合的细化并行方案，且对代码修改量和修改难度的要求较低，最大限度地保证了代码移植效率。然而在具体的技术细节方面，特别是对 C++语言的支持方面，Intel 编译器对 CPU/MIC 混合架构尚未做到软件上的完美无缺。除了在原理上无法保证 C++原有的良好的封装性外，甚至还存在不少的漏洞。目前完整的大规模并行有限元模拟程序一般仅采用 MPI 进程并行模型。正如引言中指出的那样，每个进程包含的单元计算可以并且应当得到进一步细化的并行。当计算节点采用 CPU+MIC 构型时，应当考虑利用 MIC 设备将每个进程中的单元操作并行化。本文中的应用相当于多进程任务中的某一个进程与 MIC 进行结合的测试。后续我们还将考虑将一个成熟的有限元计算流体程序[6]完整地移植到大规模多 CPU 多 MIC 的硬件平台上。

参 考 文 献

[1] 沈铂，张广勇，吴韶华，等. 基于 MIC 平台的 offload 并行方法研究[J]. 计算机科学，2014，41(6A)：477-480
Shen Bo, Zhang Guang-yong, Wu Shao-hua, et al. Research of Offload Parallel Method Based on MIC Platform[J]. Computer Science, 2014, 41(6A): 477-480

[2] 刘跃进，薛孟君. LDLT 分块求解计算方法在有限元分析中的编程实现[J]. 计算机科学，2014，41(11A)：408-409
Liu Yue-jin, Xue Meng-jun. Program of Blocks Combining with LDLT Method for Finite Element Analysis[J]. Computer Science, 2014, 41(11A): 408-409

[3] 刘建华，王朝尉，任江勇，等. 面向异构架构的混合精度有限元算法及其 CUDA 实现[J]. 计算机科学，2012，39(6)：293-296
Liu Jian-hua, Wang Chao-wei, Ren Jiang-yong, et al. Mixed Precision Finite Element Algorithm on Heterogeneous Architecture [J]. Computer Science, 2012, 39(6): 293-296

[4] 王迎瑞，任江勇，田荣. 基于 GPU 的高性能稀疏矩阵向量乘及 CG 求解器优化[J]. 计算机科学，2013，40(3)：46-49
Wang Ying-rui, Ren Jiang-yong, Tian Rong. Efficient Sparse Matrix-vector Multiplication and CG Solver Optimization on GPU[J]. Computer Science, 2013, 40(3): 46-49

[5] Luo Li, Yang Chao, Zhao Yu-bo, et al. A scalable hybrid algorithm based on domain decomposition and algebraic multigrid for solving partial differential equations on a cluster of CPU/GPUs [J/OL]. http://www.cs.colorado.edu/~cai/papers/lyzc2011.pdf

[6] Chan K, Zhang Ke-ke, Li Li-gang, et al. A new generation of convection-driven spherical dynamos using EBE finite element method[J]. Physics of the Earth and Planetary Interiors, 2007, 163(1-4): 251-265

[7] Kong Da-li, Zhang Ke-ke, Gerald S, et al. A three-dimensional numerical solution for the shape of a rotationally distorted polytrope of index unity[J]. The Astrophysical Journal, 2013, 763(2): 116-126

[8] Kong Da-li. Analytical and Numerical Studies of Several Fluid Mechanical Problems[D]. University of Exeter, 2012

[9] 王恩东，张清，沈铂，等. MIC 高性能计算编程指南[M]. 北京：中国水利水电出版社，2012
Wang En-dong, Zhang Qing, Shen Bo, et al. High-Performance Computing on the Intel Xeon Phi-How to Fully Exploit MIC Architectures[M]. Beijing: China Water and Power Press, 2012

[10] Kong Da-li, Zhang Ke-ke, Gerald S. Shapes of two-layer models of rotating planets[J]. Journal of Geophysical Research, 2010, 115(E12003): 1-11