基于申威众核处理器的海冰模式并行加速方法

李镔洋 李 波 钱德沛
(北京航空航天大学计算机学院 北京 100191)

摘 要 海冰模式是地球模式的重要组成部分,其使用不同的网格和时间梯度来模拟海冰区域随时间的变化.海冰模式具有计算密集的特性,随着海冰模式计算精度的提升,传统的硬件已难以满足其计算需求.申威太湖之光超级计算机是第一台峰值性能超过 100 Pflops 的超级计算机,其为高精度的海冰模式过程模拟提供了新的硬件平台,但在该平台上实现算法高效并行化仍面临着诸多问题.一些应用程序已经在众核平台上实现移植和并行化,但是相比其他领域,气候软件在众核平台移植和并行化的过程相对缓慢.有关气候模式在众核平台的并行化研究大多基于 GPU 实现.早期的研究多基于单个气候运算过程,该过程通常为计算密集型程序,通信过程相对较少,基于 GPU 的实现可以取得较好的并行效果.与单一的运算过程不同,海冰模式程序需要与多个气候模式进行交互,如何减少通信过程开销以及如何充分利用申威处理器所提供的并行性能是我们遇到的主要问题.为解决这一问题,该文基于申威众核处理器,设计了一种针对海冰模式算法移植和并行化的方法.每个申威众核处理器包含有 4 个核组,每个核组包含有一个管理核心和 64 个计算核心.为充分发掘申威众核处理器的并行特性,该方法分别对海冰模式数据分割方式,数据传输过程以及计算方式进行了改进和优化.该文利用该方法对海冰模式的两个算法进行了移植和并行化,并使用 CICE 测试数据集和 COREv2 数据集对该方法的性能进行测试.实验表明,并行优化后的两个算法相较其只在管理核心上运行分别可获得 11.6 倍和 9.8 倍的性能提升,且与基本并行化方法相比,该方法最高可获得 40%的性能提升.

关键词 申威众核处理器;海冰模式;数据传输;数据分割;计算方式
中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2018.02221

Accelerating a Sea Ice Model on Sunway Many-Core Processor

LI Bin-Yang LI Bo QIAN De-Pei
(*School of Computer Science and Engineering, Beihang University, Beijing* 100191)

Abstract Sea ice model is an important part of the earth system model which uses a finite difference grid and time stepping to simulate the sea ice vicissitudes. Sea ice model simulations can take from hours to days to complete due to the compute-intensive nature of the model. As a result, the size and resolution of simulations are constrained by the performance limitations of modern computing hardware. Sunway TaihuLight supercomputer is the world's first system with a peak performance greater than 100 Pflops. It brings a new opportunity for high resolution earth system model simulations but exploiting the parallelism in this architecture is not trivial. Some applications have been ported and parallelized on many core architecture platforms. But when compared with other HPC application domains, the porting of the climate models onto many-core architectures has been relatively slow. Most of efforts are based on GPU. And most early-stage efforts focused on the physics modules. Since the modules are usually compute-intensive and do not involve

收稿日期:2016-11-29;在线出版日期:2017-10-19.本课题得到国家自然科学基金项目(61133004,61502019)、重点研发项目(2016YFB0200100)资助.**李镔洋**,男,1992 年生,硕士,主要研究方向为高性能计算、分布式系统.E-mail:bin.yang.li@outlook.com.
李 波,男,1973 年生,博士,副教授,主要研究方向为高性能计算、高性能服务.**钱德沛**,男,1952 年生,教授,博士生导师,主要研究领域为高性能计算体系结构等.

communications, GPU-based acceleration can generally achieve a speedup of one order of magnitude. Different from the projects mentioned above, our work focuses on CICE 5.1 and uses Sunway many-core processor as our hardware platform. Since CICE sea ice model has many relationships with other climate components, optimizing communications between Sunway many-core processors becomes an important issue we need to address. Besides, tuning the original CICE algorithms to fit the new calculation elements is another problem we need to resolve. To address these issues, in this work we implement a new parallelization of sea ice model algorithm on Sunway many-core processor. Sunway many-core processor architecture is much different from the architecture of CPU which we use now. Each Sunway many-core process contains 4 CGs (Core Groups), each CG contains 1 MPE (Management Process Element) and 64 CPEs (Computing Process Element). To exploit the massive parallelism offered by Sunway many-core processor, we propose a parallel algorithm for sea ice model which contains the ways for data transmission, data dividing as well as the calculation method. The main purpose of our work is reducing the data transfer time and exploiting the parallelism offered by the platform. The methods we used to achieve this goal is easy-understanding and we believe simply methods are more efficient in this scenario. Some tricks are also used to achieve good performance. We benchmark with CICE test dataset as well as real world data collected from COREv2 dataset to test our implementation. Two sea ice model algorithms have been ported on Sunway many-core processor and our implementation yields speedups of up to 11.6x and 9.8x over the parallel implementations running on MPEs, while demonstrating comparable performance to the basic parallel implementation on Sunway many-core processor which improve the performance up to 40%.

Keywords Sunway many-core processor; sea ice model; data transmission; data dividing; calculation method

1 引 言

海冰模式是地球模式的一个重要组成部分.其主要包含热力学模型、海冰动力学模型、海冰迁移模型等.海冰模式是典型的高性能计算应用,与其他地球模式类似,其通过耦合器或数据文件获取所需数据,后利用计算来模拟海冰的物理和化学过程,预测海冰的状态与特征.

随着地球模式计算精度的提高和预测时间的增长,地球模式面临着如何高效并行化的问题.目前主流的地球模式软件 Community Earth System Model(CESM)①以及海冰模式软件 The Los Alamos Sea Ice Model(CICE)②普遍采用 MPI+OpenMP 混合编程的方式实现计算并行化.但随着通用体系结构发展受限,频率墙、功耗墙、存储墙的产生[1],传统结构已无法满足地球模式大规模并行化计算的需求.现代超级计算机已普遍引入 GPU、众核等新型计算单元,基于新型体系结构的并行化方法已成为相关研究的热点.

目前地球模式并行化方法改进主要集中在基于传统体系结构的并行化方法优化以及基于异构和新型体系结构的并行化方法研究上.对于传统体系结构,Huang 等人提出了 CFIO(Climate Fast Input/Output)[2]方法,该方法通过分配专门的 I/O 处理节点来使计算与 I/O 操作并行执行.对于异构体系结构,大量的研究集中在基于 GPU 的并行加速方法上.其中 Xu 等人提出的 POM(Princeton Ocean Model)[3]方法通过分离中心计算区域和边界计算区域(Halo area),使大量计算任务持续运行在 GPU 上,提升了整体运行效率.同时关于 stencil 计算在 GPU 上的并行加上方法已有较多的研究成果提出.本文所研究的海冰模式算法为 stencil 计算的一种具体实现,对于这种计算模式,重叠平铺(Overlapped tiling)[4]是一种较为常见的并行优化方法.该方法

① Cesm user's guide (cesm1.2 release series user's guide). http://www.cesm.ucar.edu/models/cesm1.2/cesm/doc/usersguide/book1.html

② CICE: the Los Alamos Sea Ice Model Documentation and Software User's Manual Version 5.1. oceans11.lanl.gov/trac/CICE/raw-attachment/wiki/WikiStart/cicedoc.pdf

通过给不同计算核心传入冗余的数据来减少计算过程中出现的同步开销，同时加大数据的局部性. Holewinski 等人[5]将该方法运用到 GPU 中并针对 stencil 计算基于 GPU 提出了一种高性能代码生成方式. Nguyen 等人对 stencil 计算在 CPU 和 GPU 上的优化和调优手段做了一定分析，并提出了一种 3.5-D 的优化措施[6]. 该措施在数据分割上对 3D 数据使用 2.5D 的数据分割方式，即一次计算只传入 X,Y 方向维度和必要的 Z 方向维度的数据，将整个数据分为多层，再进行分别计算从而增大计算核心与内存直接有效数据传输量. 本文所提出的数据分割方法在一定程度上借鉴了该方法的实现，不过与该方法不同的是，本文所研究的算法在 Z 方向维度(冰层厚度类别)上并不存在依赖关系，但在 X,Y 维度上存在额外维度(示踪器维度)的计算. 此外，Fu 等人基于申威众核处理器对 Community Atmospheric Model(CAM)①程序并行化方法进行了研究[7]. 其通过采用预调节器来改进数据分割方式以及对计算方程进行调整来发掘程序内在的并行特性.

上述这些工作主要或是基于不同于申威众核的体系结构，或是主要关注于在整体层面上对地球模式并行化方法进行研究，在更细粒度的算法优化层面上并没有过多涉及. 而基于新型体系结构计算单元的代码优化往往需要结合其体系结构特征才可充分发挥该计算单元性能. 对此本文基于申威众核处理器对海冰模式算法并行优化方法进行了研究. 申威众核处理器为神威·太湖之光超级计算机所采用的处理器单元，其每个处理器包含有 4 个 CGs(Core-Groups)，每个 CGs 由 1 个 MPE(Management Processing Element)和 64 个 CPE(Computing Processing Element)[7]组成. 每个申威众核处理器包含有 260 个核心，这为其提供了强大的计算性能.

申威众核体系结构具有与其他体系结构不同的特点，现有算法无法直接在该体系结构下高效运行. 首先，相较传统体系结构而言，申威众核体系结构的每个 CPE 拥有 64 KB 的局部存储空间，且每个核心的局部存储空间完全交由用户管理. 充分并合理使用有限的局部存储空间往往成为提升程序运行效率的关键. 其次，申威众核处理器的 CPE 所具有的逻辑运算处理能力弱于通用 CPU，深层次的计算方法优化需要完成. 最后，海冰模式涉及大量的矩阵运算，为典型的内存和计算密集型程序，简单的并行化不可避免的会造成大量的数据传输操作，细粒度数据传输的优化措施亟需提出.

本文针对以上问题，提出了一种基于申威众核处理器的海冰模式并行化方法. 该方法结合申威众核处理器在数据传输，计算上的特点和海冰模式算法的特征对海冰模式算法进行了优化. 该方法通过改进数据分割方式，使用数据传输量感知的数据传输方法和对计算方法的优化使原有算法的运行效率得到了提升. 实验结果表明，移植和优化后的两个算法，相较其只在 MPE 上并行运行，性能分别提升了 11.6 倍和 6.5 倍. 且该方法对比普通优化方法，算法执行效率最多可提升 40%.

本文的主要贡献如下：

(1) 提出一种基于申威众核处理器的海冰模式并行化方法. 该方法对于海冰模式算法的数据分割、传输和计算均进行了优化.

(2) 在数据传输方面，提出一种传输量感知的传输方式. 该方式通过对数据传输过程的优化，减少了 CPE 与主存之间的数据传输次数. 本文 3.2 节对该方法进行了详细介绍，并与 4.5 节结合实验结果对其适用范围进行了分析.

(3) 在申威众核处理器上用上述方法对海数据冰模式的两个算法进行移植和并行化，并用真实数据对所提出的方法进行了验证. 实验结果表明，与在申威众核处理器上的基本并行化实现相比，使用该方法在大多数情况下可显著缩短程序执行时间.

本文第 2 节介绍申威众核处理器的体系结构与特点以及海冰模式的相应算法；第 3 节详细介绍本文提出的设计、方法中各部分的实现以及执行过程；第 4 节展示和分析实验结果；第 5 节总结全文.

2　背景知识

2.1　申威众核体系结构

神威·太湖之光超级计算机采用不同于通用计算机的众核体系结构. 其每个众核处理器包含 4 个核组(CGs)，每个核组包括 1 个 MPE，1 个 CPE 集群以及一个 MC(Memory Controller). 所有核组通过片上网络相连. 每个核组都拥有自己的存储空间并通过 MC 与 MPE 和 CPE 集群相连. 处理器通过

① National Center for Atmospheric Research, Boulder, CO, 30pp. Available at: http://www.cesm.ucar.edu/models/ccsm4.0/cam/docs/users_guide/ug.pdf, 2010

系统接口(System Interface,SI)来连接其他外部硬件[7].图 1 展示了申威众核处理器的基本结构.

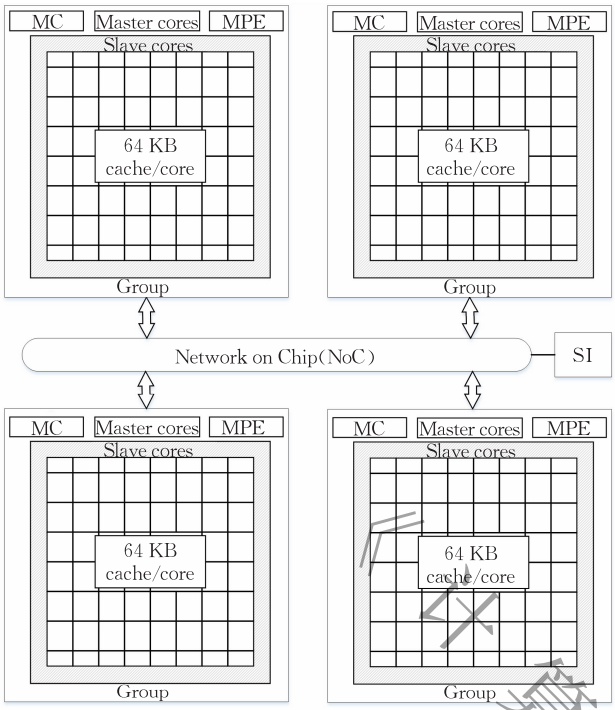


图 1 申威众核处理器的基本结构

其中,MPE 为申威众核处理器每个核组的管理核心.该核心采用 64-bit RISC 体系结构并支持指令中断、内存管理、超标量指令以及指令乱序执行.其在大规模计算过程中通常用于完成管理计算核心以及与其他核组进行通信的任务.

CPE 作为申威众核处理器中的计算核心,相较管理核心而言只支持有限的处理指令.CPE 只能运行于用户模式并且不支持中断函数,使得其比较适用于逻辑简单而计算相对密集的过程.

在本文的并行加速方法中,为加速整个计算过程和简化 CPE 中执行任务的计算步骤,MPE 除完成控制计算核心完成计算以及与其他核组通信外还需分担少量的计算任务.3.4 节对本文并行加上方法的计算过程进行了详细介绍.

由于申威众核处理器中每个 CPE 只有 64 KB 的局部存储空间,这使得数据传输往往成为程序运行的瓶颈,本文分别对使用不同核心数和不同数据传输粒度时的传输带宽做了测量,作为之后数据传输优化的依据,测量结果如图 2 和图 3 所示.

图 2 和图 3 并没有展示传输数据量大于 4 KB 时的情况,这是由于当一次传输数据量超过 4 KB 时,传输带宽趋于稳定且大致与 4 KB 时相同.详细数据见附录表 9.

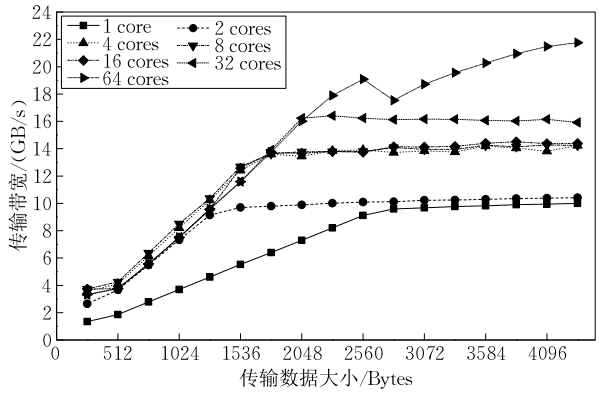


图 2 主存到 CPE 的数据传输速率

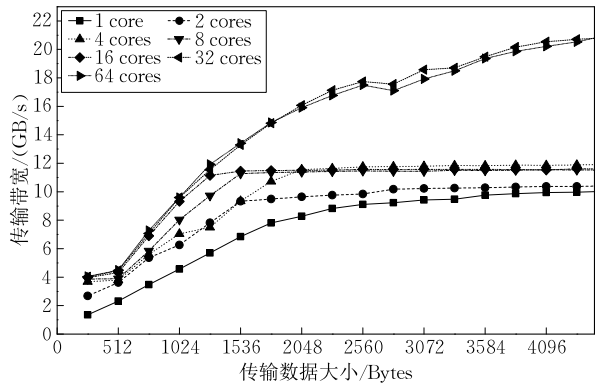


图 3 CPE 到主存的数据传输速率

通过实验可以观察到两个现象:(1)当数据传输量在 2 KB 及以上时主存与 CPE 之间的带宽可以得到充分利用,在此之前,传输带宽随着数据量的增加基本呈线性增长;(2)传输带宽随参与传输的计算核心增加而上升,虽然此时每个 CPE 的传输带宽有所降低,但核组的整体带宽得到了提升.本文 3.2 节中的传输优化方法即是基于这两个事实.

2.2 海冰模式算法介绍

本文选用 CICE v5.1 作为要移植的海冰模式.CICE 海冰模式根据海冰的厚度,将整个海冰分为 N_C 层,如用 a_{in} 表示第 n 层占整个海冰面积的比例,则有 $\sum_{n=0}^{N_C} a_{in}=1$,其中当 $n=0$ 时 a_{i0} 代表开放水面所占的比例.对于每个冰层而言,都有一定数量的示踪器(tracer)用于记录该冰层的特定状态.在本文的实验中,$N_C=5$, tracer 数量为 24.

本文对 CICE 中的两个算法进行移植,分别为 Reconstruct 算法和 Integrating Field 算法.这两个算法主要对以矩阵形式存放的数据进行计算.其基本操作均为读入数据矩阵进行计算,输出结果矩阵.两个算法的计算过程均需要用到邻居网格的数值,

且参与计算的矩阵之间具有一定的数据依赖关系.下面分别对这两个算法进行简要介绍.

$$\frac{\partial g}{\partial t}=-\nabla\cdot(g\boldsymbol{u})-\frac{\partial}{\partial h}(fg)+\varphi \tag{1}$$

CICE 海冰模式被用来描述和预测不同厚度冰层在时间和空间上的变化,其基本方程为式(1)所示,其中 u 是海冰的水平速度,$\nabla=\left(\frac{\partial}{\partial x},\frac{\partial}{\partial y}\right)$,$f$ 是热力学海冰生长速率,φ 是冰脊重分布方程.g 是海冰厚度分布方程.在这里我们定义 $g(x,h,t)dh$ 表示在给定时间和区域下,冰层厚度在$(h,h+dh)$区间的海冰覆盖率.

而海冰模式水平方向移动的计算主要是对方程(2)进行求解.

$$\frac{\partial a_{in}}{\partial t}+\nabla\cdot(a_{in}\boldsymbol{u})=0 \tag{2}$$

方程(2)描述了海冰在水平移动过程中,海冰面积的守恒.方程(2)可由式(1)推导得出,对式(1)中的 g 进行离散化并忽略式(1)右侧的第二项与第三项(这两项与冰层水平运动无关)便可推出方程(2).而对于特定厚度冰层示踪器的相关数值计算可通过构造与方程(2)结构相近的特定方程来求解.

Reconstruct 算法为冰层水平移动计算的一部分,亦是对方程(2)求解的第一步.

在 CICE 海冰模式中,已把整个地球通过一定的规则分割成若干个的 cell,在进行冰层水平移动计算时,首先需要将当前 cell 的状态表示成与 x,y 相关的线性函数,这一计算由 Reconstruct 算法完成.在该算法中,每个 cell 的中心被看作坐标原点且拥有相应的数值,该算法需要计算的数值为冰层坐标原点处的数值以及根据 van Leer 限制[8]得出的 x,y 方向的梯度限制.式(3)为该算法的主要方程.

$$a_c=\bar{a}-a_x\bar{x}-a_y\bar{y} \tag{3}$$

式(3)中 a_c,a_x 和 a_y 为该算法的输出,分别表示 cell 中心处的数值,x 方向的梯度限制和 y 方向的梯度限制.该式中其余变量为本算法的输入.其中 $\bar{x}=\int_A x\mathrm{d}A/A$,$\bar{y}=\int_A y\mathrm{d}A/A$,$A$ 表示这一 cell 的面积,$\bar{x}$ 和 $\bar{y}$ 的数值由 CICE 水平移动计算的另一算法给出.为完成 Reconstruct 算法的计算过程,算法还需要 x 和 y 的高阶平均值作为输入,例如:$\overline{x^2}$,$\overline{y^2}$,$\overline{xy}$.图 4 显示了 Reconstruct 算法的输入输出变量与基本执行流程.

从图 4 中可以看出 Reconstruct 算法在运算过

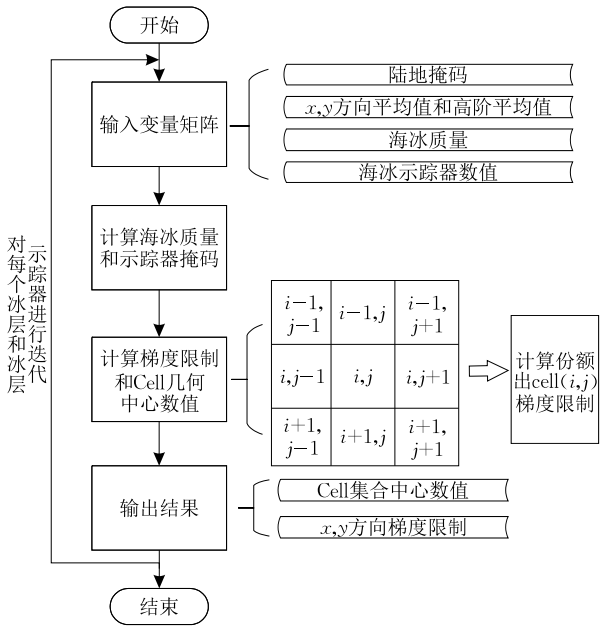


图 4　Reconstruct 算法执行流程

程中需要用到 8 个邻居 cell 的数值来计算梯度限制,且对于每一冰层和示踪器都需要进行输入和输出操作,其中输入变量包括海冰的质量、陆地掩码、海冰示踪器的质量,$\bar{x}$,$\bar{y}$的值以及其高阶平均值.而对于每个 cell 均会产生 3 个输出分别为 cell 中心处的数值以及 x,y 方向的梯度限制.是典型的内存计算密集型算法.

Integrating Field 算法为冰层水平移动过程计算中的另一算法,其主要计算冰层水平移动后改变的通量面积(flux area).该算法通过计算由每个 cell 和其相邻的 cell 构成的所有三角形通量区域的面积,来达成积分计算的效果.该算法的主要流程如图 5 所示.

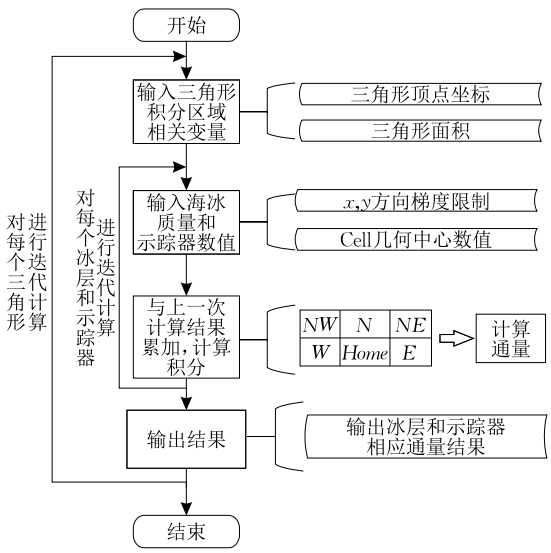


图 5　Integrating Field 算法执行流程

与 Reconstruct 算法类似，该算法需要多种数据来完成整个计算.且由图 5 可以看出，对于每一次通量面积的计算，都会用到相邻的 5 个 cell 的相关数值.与 Reconstruct 算法不同的是，该算法涉及到双重循环，并增加了一次判断是否需要输入的操作，且内层循环仍有数据输入，这使得该算法输入数据量随着计算网格的不同有着很大变化，计算任务量与计算所时长也随着数据输入量的不同呈现出较大差异.且该算法需要输入的矩阵数量也相较 Reconstruct 算法多出约 18%，是一种计算数据较为不规整，运算情形较为复杂的算法.其中该算法的主要输入数据包括要计算三角形的坐标；相关三角形的面积；由 Reconstruct 算法计算得出的 cell 中心数值和 x，y 方向的梯度限制以及该 cell 对应示踪器的相应数值.而该算法的输出为通过该 cell 边的通量数值以及对应示踪器的变化通量.

通过对以上两种算法的分析可知，这两种算法具备以下两个特点：(1) 需要大量的数据输入作为算法执行前准备；(2) 每一网格的数值计算过程依赖于相邻网格的数值.由于申威众核处理器一次可计算的数据大小和数据传输量受限于 CPE 的局部存储大小和主存与 CPE 之间的带宽，为使具备上述特点的算法在申威众核处理器上高效运行，细粒度的并行化方法需要被提出和采用.

3 海冰模式算法并行化方法

本节分别对海冰模式并行化方法所涉及的数据分割方式，传输量感知的数据传输方法以及计算优化方法进行了详细的介绍，并对使用这些方法的原理进行了分析.

3.1 基于申威众核平台海冰模式算法基本并行方法

现有的海冰模式算法基于 MPI 和 OpenMP 并行框架实现.其中不同 CPU 之间使用 MPI 机制实现相互间消息通信，而在同一 CPU 内利用 OpenMP 实现线程级加速.由于基本的海冰模式算法并没有对针对申威众核体系结构进行实现，为了实验对比，我们在进行具体算法优化前首先将原算法通过交叉编译的方式移植于申威众核平台，并利用计算核心对移植后的算法进行基本的并行化处理.

对于要移植和并行化的 Reconstruct 算法和 Integrating Field 算法，我们采用的基本并行化方案为：利用控制核心完成对计算核心任务的启动和收集控制，运用 MPI 完成不同核组间的消息通信，并完成一定量的计算任务；计算核心完成原程序计算密集部分计算并将计算结果传回主存.

基本并行方法使用块分割的方式将每一核组的网格数据分配到不同计算核心中并使用原算法的计算流程完成整个任务计算.该并行方法作为对比方法来验证本文 3.2 节所提出的数据分割方法的有效性.为对比本文所提出的数据传输方法和数据计算方法相较于原有算法性能的改进，3.2 节之后我们将基本并行方法的数据分割方式改为与改进后方法相同的数据分割方式.且实验部分也使用修改后的基本并行方法与改进后算法进行对比.

3.2 数据网格分割方法

本文 2.2 节中提到，在 CICE 海冰模式中已经把地球分割成若干个 cell，且所有 cell 拼接成一个规整的矩形，这为以数据并行的方式实现并行化提供了可能.

海冰模式中常见的网格尺度有 gx3，100×116 大小的网格以及 gx1，320×384 大小的网格.随着对测量精度要求的提高，现有的网格精度已经提升到 0.1°，即 3200×2400 大小的网格.但由于在高精度计算时往往会对网格数据进行分割，分配在每个核组上的数据规模并不会很大.在使用 200 个申威众核处理器对 0.1° 精度的数据进行计算时，分配到每个核组的数据大小已小于 100×116 的规模.故对于申威处理器的一个核组来说，计算 100×116 大小网格已可满足高精度计算的需求.

这里以 gx3(100×116) 网格为例，为将其分割为 64 个任务交由计算核心处理，最原始的分割做法如图 6 所示.

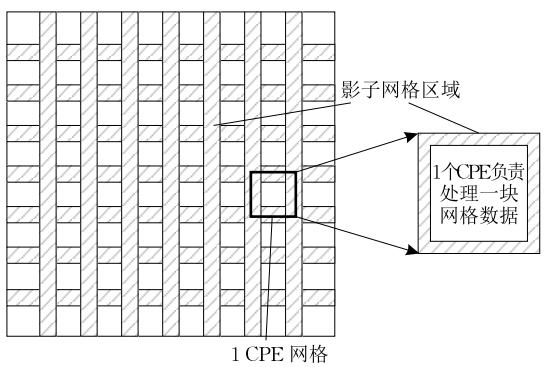


图 6 传统数据分割方式

这里按照行列将数据分解为 8×8 份，每个 CPE 选择一份来计算，对于 Reconstruct 算法以及 Integrating Field 算法来说，由于在计算过程中会使

用到邻居网格的数据，在分割过程中需要将邻居网格包含.

这种分割方式可以平均地把网格分配到多个CPE，但每个CPE所分到的数据在物理存储上并不连续，这就使得一次计算所需的数据需要连续多次传输才能完成. 以100×116的网格为例，每个CPE所分到的最大数据大小为13×15的网格，若数据按双精度浮点数存储，此时每次的数据传输量只有112 B. 根据本文2.1节所描述的申威处理器的数据传输性能来看，该情况下主存与CPE之间的带宽利用率极低(目前申威众核DMA数据传输支持跨步传输的数据传输方式，但该方法的数据传输速率与异步的点对点数据传输速率基本相当. 在每一步长数据传输量较小时，数据传输延迟依旧是整个算法执行流程的瓶颈).

为解决该问题，我们对海冰模式的数据特点进行了分析. 通过分析可以发现，CICE海冰模式将整个海冰分为N_C层，每一层的计算相互独立. 基于这一现象将数据分割方式进行更改，更改方式如图7所示.

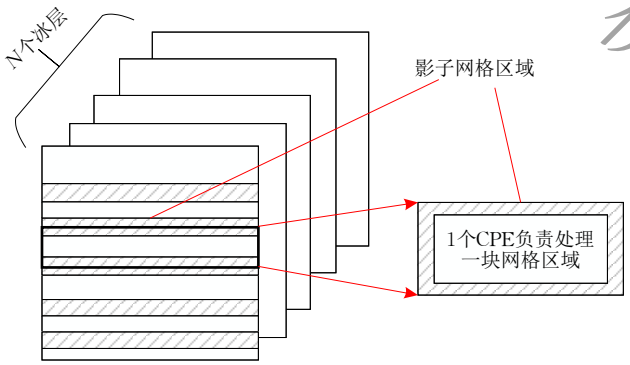


图7　改进后数据分割方式

此处采用若干个CPE处理一层的方法，加大了每个CPE一次处理的数据量. 同时对于特定的一层，使用按行分割的方式对网格进行划分. 通过这一改进使得每个CPE处理的数据在主存中实现地址连续，一次传输的数据量不再由数据分割方式来限制，转而受限于每个CPE中局存空间的大小. 同样以100×116的网格数据为例，当$N_C=5$时，改进后的算法每个核心分得的网格大小大致为100×12，且整个网格数据在内存地址空间中连续. 在计算核心局部存储空间足够大的情况下输入数据可以通过一次读取完成. 我们对Reconstruct算法在gx3网格尺度下分别使用传统方法和改进后方法进行了实现，并比较了这两种实现在不同大小网格下的运行效果. 本实验中$N_C=5$，表1为实验结果，由表可见改进后的方法较传统分割方法比，性能提升了3.5～5倍.

表1　采用不同分割方式算法运行时间比较

网格大小	传统计算方法/μs	改进后方法/μs
100×116	49446	13863
50×116	42477	9714
50×58	21520	5029
25×58	19661	3886

相较基本的块分割数据分割方式，本文基于申威众核体系结构，提出了一种针对高维度stencil计算的数据分割方法. 该方法主要关注点在于提升数据读取的连续性. 这一方面是由于申威众核体系结构中计算核心与管理核心之间的关系更接近于与异构体系结构中GPU和CPU之间的关系. 控制核心和计算核心不共享存储空间使得数据凝聚读取成为提升性能的关键. 另一方面是由于申威众核处理器计算核心众多且每个计算核心局部存储空间有限，加之主存和计算核心局之间有限的带宽，使得对于该处理器结构来说，数据传输速率成为制约程序性能提升的瓶颈. 本文的数据分割方式相较块分割的数据分割方法可以更好的适应申威众核处理器体系结构特点. 文中之后的优化均在此数据分割方法的基础上进行.

3.3　传输量感知的数据传输方法

CPE与主存之间的数据传输往往占程序运行时间的绝大部分，减少数据传输时间可以极大的提升程序运行速度. 本文根据申威处理器数据传输的性能特点与CICE海冰模式的数据特征，提出了一种传输量感知的数据传输方式.

3.3.1　数据传输方法概述

在3.1节的基础上，数据传输量的限制已经从网格划分变为CPE局存空间大小. 为了充分利用局存空间，我们分析了CICE程序. 前文提到，CICE将整个海冰分为N_C层，并有$\sum_{n=0}^{N_C} a_{in}=1$，其中$a_{in}$表示第$n$层占整个海冰面积的比例. 由于每一层由同样规格的地球网格构成，除去陆地所占的面积以及不属于该层的海冰覆盖，每一层对应矩阵中，参与计算的网格数量并不多. 表2显示了1997年gx3网格尺度数据不同冰层有效数据的分布情况，由表2可见对于每一冰层来说，其有效数据量并不超过14%.

表 2 1997 年 gx3 网格尺度数据不同冰层有效数据量

冰层编号	有效数据比例
0	0.089
1	0.136
2	0.137
3	0.136
4	0.135
5	0.131

基于每一冰层有效数据量有限的事实，本文采用 CSR① 方法对已传入 CPE 中的数据进行压缩，并在计算结束后再将计算结果解压，传回主存. 通过该方法可以进一步释放 CPE 中的可用空间，同时使 CPE 一次处理更多的数据. 该方法本质上是通过增大每次数据传输量，减少总数据传输次数来缩短数据传输所消耗的时间. 但由于网格中的数据分布是不规律的，程序实际运行时并不知道这一次可以传入的数据大小. 为解决这一问题，本文提出了传输量感知的数据传输方法. 该方法使用类似分支预测的方式[9]，根据过往传输数据中有效数据量的大小，预测出下一次的数据传输量，从而为程序执行预先做好初始化工作. 算法 1 展示了该方法的执行过程.

算法 1. Data transmission.

输入：CPE 局存大小(*local_size*)，传输缓冲区大小(*trans_size*)，初始传输数据大小(*init*)，数据增加或缩小步长(*pace*)，需要传输的数据量(*left_blocks*)，传输缓冲区(*trans_area*)

变量：*limit*：压缩数据大小限制

next：下次输入的数据大小

csr：使用 csr 方法压缩数据后的数据状态

pre：当前的预测状态

max_size：一次最大传输数据量

real：实际可输入的数据大小

函数定义：*get_max_size*：获取一次可以传输的最大数据量

get_limit：得到对于每个变量而言，当前局存最多可以容纳的数据量

init_predictor：初始化预测器结构

init_csr：对传入的数据进行压缩

change_state：更改预测器状态

shrink：对已经压缩的数据根据可以存放数据的大小再进行调整

1. *max_size*=*get_max_size*(*trans_size*, *pace*)
2. *limit*=*get_limit*(*trans_size*, *local_size*)
3. *init_predictor*(*pre*, *pace*, *init*, *limit*, *max_size*)
4. When *left_block* >0
5. *next*=*next_trans*(*pre*)
6. *trans_data*(*trans_area*, *next*)
7. *init_csr*(*csr*, *trans_area*, *next*)
8. *real*=*change_state*(*pre*, *csr*—>*val_length*)
9. IF *real* <*next*
10. *shrink*(*csr*, *real*)
11. END IF
12. *left_blocks*=*left_blocks*−*real*
13. END WHEN

如算法1所示，程序在执行过程中首先会计算 CPE 中可以存放的最大数据大小，而后使用 *init_predict* 初始化预测器结构. 之后，程序根据从 *next_trans* 函数中获得的预测数据输入大小传输数据. 下一次的预测值会依据本次实际有效数据的多少进行调整，在算法 1 中即调用 *change_state* 函数，该函数返回本次传输最大可保存的数据量大小并对预测器结构进行更新. 若本次传输数据量过大，还需要调整已经生成的压缩数据，即调用 *shrink* 函数，使其达到合适大小.

3.3.2 预测器结构与预测过程

本节对传输感知算法的数据预测部分进行详细介绍. 该算法和分支预测算法类似，分支预测算法有跳转和不跳转两个预测状态，可以用 1 bit 来表示. 该算法拥有传输量增加，传输量不变和传输量减少 3 种状态，这里我们使用 2 bit 来表示. 算法中采用 1 个全局历史状态寄存器(Global Phase History Register，GPHR)和一张模式历史表(Pattern History Table，PHT)来产生下一次数据传输预测结果. 图 8 展示了预测器的基本结构.

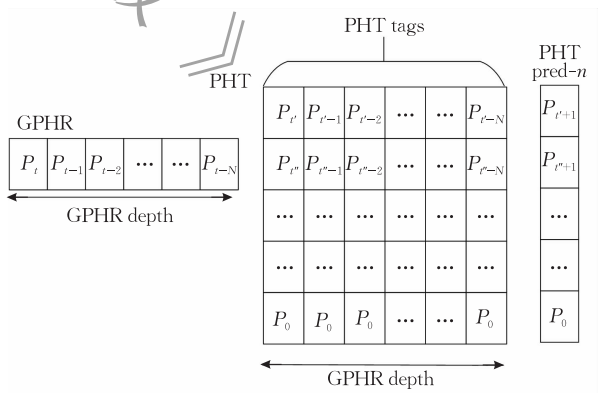


图 8 预测器的基本结构

图 8 中 GPHR depth 表示全局历史寄存器保存的历史状态记录数量，PHT tags 相当于 PHT 的一个索引，每一个索引都关联着一个在该状态下应该进行的操作. 当有新的状态到来使，GPHR 中便会

① A basic tool kit for sparse matrix computations, 1990. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.3853

记录最新的状态并删除最旧的状态，进而生成新的索引，通过索引可以在 PHT 中找到对应的预测结果.

在实现中我们用 GPHR 保存最近两次的预测. 由于每次预测有 3 种情形，所以使用一张有 9 个表项的 PHT 来记录每种状态下应进行的操作. 对于每种情形下应进行的操作，我们提供一种类似于饱和计数器的方式来给出结果. 该饱和计数器拥有 6 种状态，图 9 为 PHT 中一个表项的预测数值的状态转移过程. 其中每种状态都有该状态编号和应给出的预测状态结果，状态转移条件由实际传输数据量与预测传输数据量的比较得出，所有表项的初始状态为状态 2.

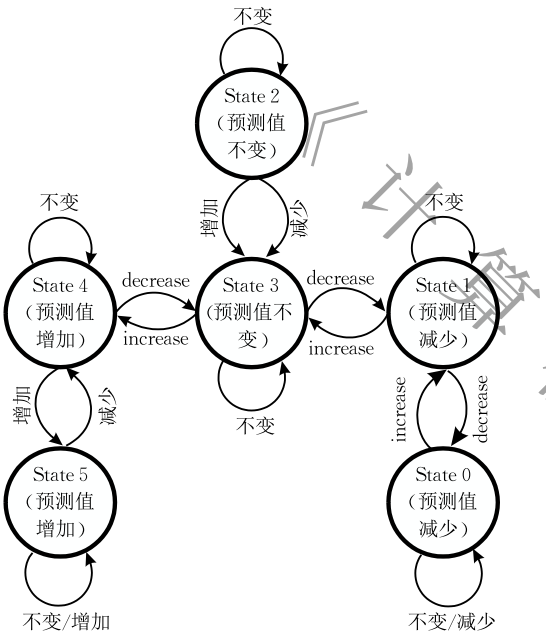


图 9　状态转移图

对于每次预测来说，程序可以根据 GPHR 查找到对应表项的预测值，并根据该表项对应饱和计数器的状态，对预测值进行修改并将修改后的预测值返回给用户. 后根据实际传输有效数据的多少，对该表项饱和计数器的状态进行更新，从而保证预测的准确性.

3.4　海冰模式算法计算优化

本节对海冰模式算法在申威众核处理器上的计算优化方式进行讨论. 本文研究的海冰模式算法在计算结构上大体可分为对开放水面进行计算和对冰层计算两个部分. 其中对开放水面计算部分计算量较小，故在并行方法实现时我们将该部分放入控制核心上执行而冰层计算部分放入计算核心执行.

由于申威众核处理器体系结构的特殊性，具体算法在计算过程实现上需要遵循减少额外存储空间开辟和避免逻辑判断语句两个准则. 本节针对海冰模式算法依照这两个准则介绍该算法在申威众核体系结构下的具体计算优化方法和实现方式.

3.4.1　局部存储空间优化

在第 2 节和第 3 节中已经分析并指出，众核计算核心局部存储空间的大小限制着海冰模式算法运行效率的提升，如何合理运用有限的局部存储空间成为算法效率提升的关键. 虽然不同算法所使用的计算变量是不同的，但存储空间优化方法基本相似. 下面对 Reconstruct 算法和 Integrating Field 算法中主要使用的存储空间优化方法进行讨论.

(1) 重用变量空间

海冰模式算法在计算过程中通常需要众多变量参与计算，但根据数据局部性的原理可以得知，同一时刻使用到的变量比较有限. 所以在具体实现时我们可以根据变量的使用时间，重用已经不再使用的变量空间.

对于 Reconstruct 算法来说，计算过程分为两部分：冰层数值计算和冰层示踪器计算. 这两部分计算完全独立，冰层示踪器计算会用到冰层计算的计算结果以及相同输入变量. 故在传输框架初始化时，计算必要的存储空间即可，并不需要计算所有变量的存储空间之和. 图 10 对 Reconstruct 算法的变量使用情况进行了分析.

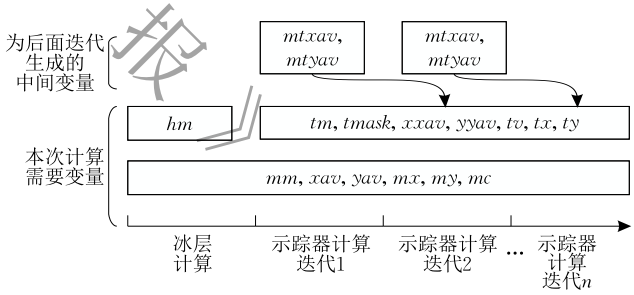


图 10　Reconstruct 算法变量使用情况

如图 10 可知 Reconstruct 方法中示踪器计算需要的计算空间大于冰层计算，故算法 1 中 *get_limit* 函数使用示踪器所需的变量空间来进行计算，同时冰层计算可以复用示踪器变量申请的变量空间.

Integrating Field 算法结构上与 Reconstruct 方法类似，同样是分为冰层计算和示踪器计算两个部分，只是计算过程较为复杂，完成计算参与的变量多于 Reconstruct 算法. 为充分利用有限的存储空间，亦需要对参与不同部分计算的计算变量进行分析，定位可重复利用的空间. 该算法的变量分析如图 11 所示. 在具体实现中，冰层计算中的 $mflx$，xp，yp，

mc，mx，my 的变量空间在示踪器计算中由 $mtflx$，$mtsum$，$mtxsum$，$mtysum$ 等变量重复使用.

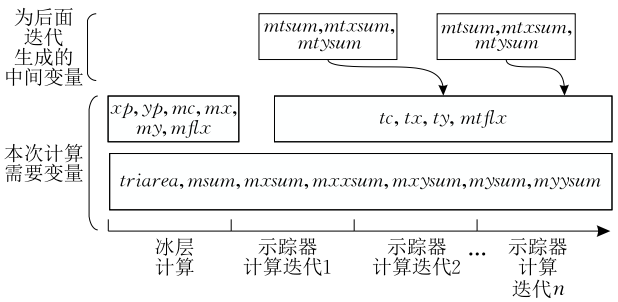


图 11 Integrating Field 算法变量使用情况

(2) 使用辅助存储

虽然数据传输往往是整个算法的运行瓶颈所在，但适当的数据传输可以简化算法复杂度，节约局部存储空间，同时增大每次计算的计算量.

在 Reconstruct 算法中，对示踪器进行迭代时，后面示踪器计算可能会用到前面某一次示踪器计算产生的中间结果. 图 10 中的变量 $mtxav$ 和 $mtyav$ 即为这种情况，对于这种情况可采用将中间数据传回主存，使用时再传入计算核心的方式，避免存储多次中间结果，进一步增大计算核心一次可计算的数据量.

同样在 Integrating Field 算法中，示踪器计算也会用到前面冰层计算得出的中间数值. 在这里同样采用将中间数据传回主存需要时再传入的方式. 值得注意的是，由于该数据为中间数据，为了减少数据传输造成的开销，可以将这些中间数据合并传输. 由于本文使用了数据压缩的方法来减少局部空间的使用，在数据传输时可以直接传输压缩后的数据，进一步减少数据传输量和减少额外的数据传输所带来的开销.

3.4.2 计算过程优化

实验发现，在申威众核处理器中，CPE 处理逻辑计算的能力远低于处理算数运算的能力. 针对这一问题，本文对 Reconstruct 和 Integrating Field 两个算法的计算过程进行了优化.

在第 2 节对原算法分析时可以发现，两个算法在计算时均要用到邻居位置的数值，但在采用传输感知的传输方法后，由于数据压缩的缘故，邻居位置的数值位置并不能直接通过给定的数组下标直接获得，使得每次查找都需要极大的开销. 这里我们通过重用传输缓冲区，首先将计算部分与其邻居位置的数据还原为可通过给定数值下标直接获得的数据形式，从而得以在 $O(1)$ 的时间内得到需要计算的数值，之后再对计算区域的数据进行计算. 算法 2 对这一方法进行了详细的描述.

算法 2. Calculation method.

输入：传输缓冲区($trans\_area$)，压缩数据区域(cal_area)，压缩后的数据状态(csr)，存放结果的空间(ans_area)，需要数据大小(cal_size)，传输缓冲区数据大小($trans\_area\_size$)

输出：压缩后的计算结果

变量：$temp\_area$ 临时存放结果的空间

```
1.  IF trans_area_size >= cal_size
2.      retrains_with_val(csr, cal_area, trans_are)
3.      calculate_ans(temp_area, trans_area)
4.      construct_same(csr, ans_area, temp_area)
5.  ELSE
6.      times=(cal_size - trans_area_size + 1)/
                trans_area_size + 1
7.      split cal_area and ans_area into times pieces
8.      FOR i in 1, times
9.          retrains_with_val(csr, cal_area[i], trans_are)
10.         calculate_ans(temp_area, trans_area)
11.         construct_same(csr, ans_area[i], temp_area)
12.         i++
13.     END FOR
14. END IF
15. Return ans_area
```

通过该方法，算法在实际计算过程中可以使用一个简单循环完成，循环主体即为计算过程，并不需要添加额外的逻辑判断操作.

在实际实现时我们同样对通过判断访问邻居节点的方法进行了实现. 在运行时我们发现通过判断访问邻居节点的计算方法运行效率与基本并行优化方法效率相当. 循环迭代过程中的条件判断语句极大的制约了程序执行效率的提升.

4 实验结果与分析

本节对 Reconstruct 和 Integrating Field 两个算法分别采用基本并行化方法(3.1 节所述方法)和本文提出的并行化方法实现的实验结果进行了比较. 并与在 Intel Xeon 上并行执行的结果进行了对比. 并在本节最后对实验结果进行了详细分析.

4.1 实验准备

本次实验使用申威众核处理器和 Intel Xeon E5620(2.40GHz)处理器，具体的软硬件环境如表 3 所示.

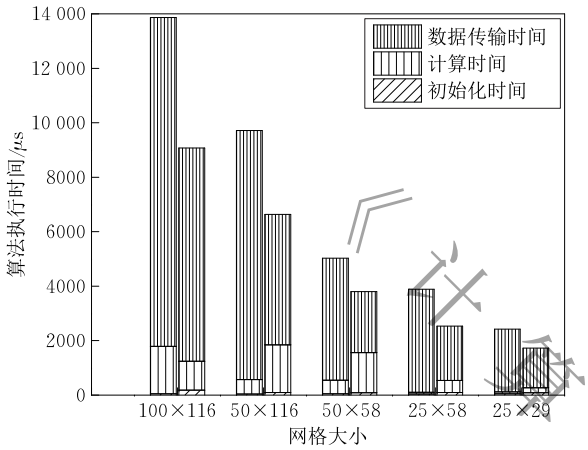
表 3　软硬件环境清单

CPU	内存	编译器	Cache 大小
Sunway Taihu Light 1.45 GHz	32 GB for 4 CG	mpif90 sw5cc	256 KB per MPE
Intel(R) Xeon(R) 2.4 GHz	12 GB	gfortran	12 MB per core

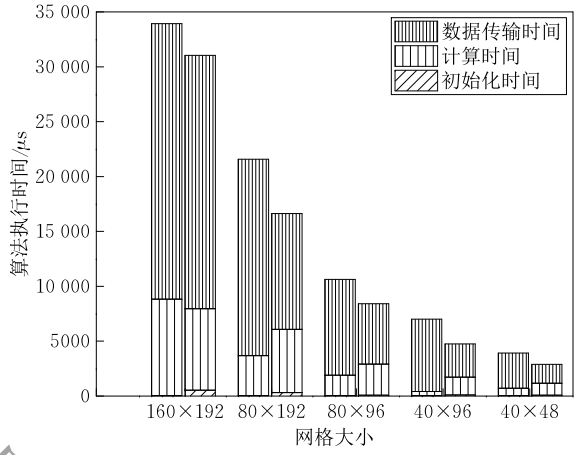
本次实验使用的数据分为 gx3 精度和 gx1 精度两种格式. 其中 gx3 精度的数据采用 CICE 提供的测试数据集，gx1 精度的数据使用 COREv2 (version 2 forcing for Coordinated Ocean-ice Reference Experiments)[10] 中所提供的数据. 由于 COREv2 中的原始数据为 T62 格式，与 CICE 中所需的数据格式不符，在实验前需使用 SCRIP(Spherical Coordinate Remapping and Interpolation Package)① 和 CDO (Climate Data Operators)②对原始数据使用双线性插值的方法进行预处理，转换成所需格式.

4.2　不同网格维度下算法改进前后实验对比分析

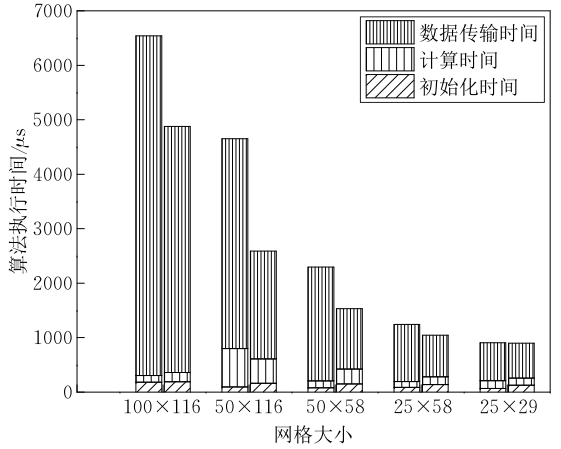
本节对比分析了不同网格尺度下基本并行方法与改进后算法的在算法初始化部分、数据传输部分和计算部分所消耗时间的差异. 实验结果如图 12 所示.



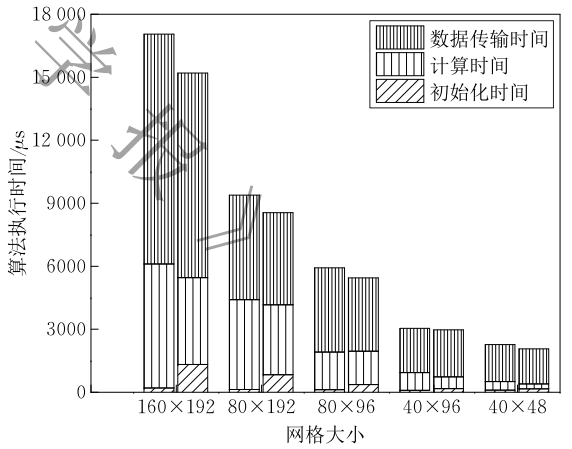
(a) Reconstruct算法gx3精度下不同实现方法对比



(b) Reconstruct算法gx1精度下不同实现方法对比(改进后方法在后)



(c) Integrating Field算法gx3精度下不同实现方法



(d) Integrating Field算法gx1精度下不同实现方法对比(改进后方法在后)

图 12　不同并行化实现在各网格尺度下运行时间对比

图 12 中对比了 gx3 精度下和 gx1 精度下，每个核组使用不同的算法实现方法在不同大小的网格下所运行的时间. 其中基本并行方法在前，本文提出的方法在后，对于每个算法的运行时间我们都将其分为初始化、数据传输和计算三个部分，并在图中加以展示. 从图中可以看出改进后的算法在多个网格维度下的数据传输时间均有较大幅度的降低，这与预期相符，其中最大数据传输时间下降比例达到 50% 以上. 计算部分的时间在一些维度下有所下降，在另一些维度下有所增长，而算法初始化的时间相比基本并行算法均有一定程度的增加. 初始化时间增加是由于采用传输感知的传输方式引入了一些逻辑计算和初始化准备的过程. 而计算时间的增长是由于在计算执行前，为适应传输方式的改变，引入了数据

① A spherical coordinate remapping and interpolation package. http://oceans11.lanl.gov/trac/SCRIP

② CDO user's guide. https://code.mpimet.mpg.de/projects/cdo/embedded/cdo.pdf

准备阶段(如 3.4 节所述).对于不同的算法,计算部分的时间呈现出不同的变化趋势.在一些情况下,如所需要计算的数据只占数据矩阵的很小一部分时,由于该算法可以利用到数据压缩后的结果,从而可以避免不必要的计算进而缩短计算所需时间,但对于数据量比较多或数据矩阵较小的情况,计算部分所需时间往往会比原有时间略微增加.但这些增加的时间相较数据传输缩短的时间而言是可以接受的.

在图 12 中可以看出改进后的算法在大多数网格尺度下相较原算法运行时间都有显著的降低.但对于 Reconstruct 算法和 Integrating Field 算法在 160×192 网格尺度下以及对于 Integrating Field 算法在 80×96,40×96 和 40×48 网格尺度下,传输和运行时间与原算法基本相当.表 3 列出了在这些网格尺度下使用基本并行算法和改进后算法每个核心涉及的数据传输次数.这里我们将完成从主存到 CPE 一次计算所需的所有数据传输称为一次传输.

表 4 不同实现方式的数据传输次数比较

网格大小	基本并行方法	改进后并行方法
192×168[1,2]	16	16
80×96[2]	4	3
40×92[2]	2	2
40×48[2]	1	1

注:[1]代表 Reconstruct 算法;[2]代表 Integrating Field 算法.

从表 4 中可以看出在这些网格大小下,改进后的传输次数与基本并行算法传输次数基本相当,造成这一现象的原因将在 4.5 节中进行分析.由于这一现象的存在,在这些尺度下,传输效率的提升受到了限制,但较基本并行方法相比,改进后方法的运行效率仍略有提高.

4.3 传输与计算过程优化效果

对数据传输和计算过程的优化是本文算法优化的主要内容,这里主要就数据传输和计算过程优化方法对算法整个性能提升的进行讨论.

4.3.1 有效数据量感知的数据传输方法对性能提升的贡献

有效数据量感知的数据传输方法本质上是通过增大每次的数据传输量,减少数据传输次数来减少数据传输开销.下面对有效数据量感知的数据传输方法中预测器的准确度以及其对数据传输性能提升效果进行分析.

由于数据传输方法中的预测器作用到每个计算核心,这里选取 0 号计算核心进行分析.我们使用 1997 年 gx3 100×116 的数据集作为输入.由于在计算时需要邻居网格参与,所以实际计算的网格大小为 102×118.表 5 给出了对于 Reconstruct 算法,当 0 号核心一次可以计算的有效数据量从半行增大到两行时,基本并行方法每次可读取的数据量,改进后算法每次可读取的数据量以及每次预测结果的正确性.

表 5 不同数据传输方法每次读取数据量

(a) 一次可计算 51 个有效数据

数据传输次数	基本并行方法数据传输量	改进后方法数据传输量[1]	预测结果	是否正确
1	51	102	不变	错误
2	51	102	不变	错误
3	51	102	不变	正确
4	51	102	不变	正确
5	51	102	不变	正确
6	51	102	不变	正确
7	51	51	不变	错误
8	51	51	—[2]	—
9	51	51	减少	正确
10	51	51	—	—
11	51	51	不变	正确
12	51	51	—	—
13	51			
14	51			
15	51			
16	51			
17	51			
18	51			

注:[1]预测器以每行数据量单位进行增加或减少,故初始化为 1 行数据量;[2]对于不足一行的数据读取,不参与数据预测器状态转移.

(b) 一次可计算 102 个有效数据

数据传输次数	基本并行方法数据传输量	改进后方法数据传输量	预测结果	是否正确
1	102	102	不变	错误
2	102	102	不变	错误
3	102	204	增加	正确
4	102	204	不变	正确
5	102	102	不变	错误
6	102	102	不变	正确
7	102	102	不变	正确
8	102	102	不变	正确
9	102			

(c) 一次可计算 204 个有效数据

数据传输次数	基本并行方法数据传输量	改进后方法数据传输量	预测结果	是否正确
1	204	204	不变	错误
2	204	204	不变	错误
3	204	306	增加	正确
4	204	204	不变	错误
5	102			

从表 5 中可以看出本文所提出的预测方法可以有效地缩短数据传输次数.在计算核心一次可计算的数据量为 51,102 和 204 时,本文所提出的数据传输量预测方法准确率分别为 66.6%,62.5%,25%.同时该方法可以将数据传输次数分别缩短了 38%,11%和 20%.

图 13 显示了 Reconstruct 算法中，对于 gx3 100×116 网格尺度数据，在申威众核处理器上运行时不同计算核心在基本并行方法下所需的数据传输次数和改进后数据传输方法下所需的数据传输次数. 从图 11 中可以看出，使用改进后的数据传输方法可以将数据传输次数最多减少 25%. 在数据传输上加快了整个算法的执行流程.

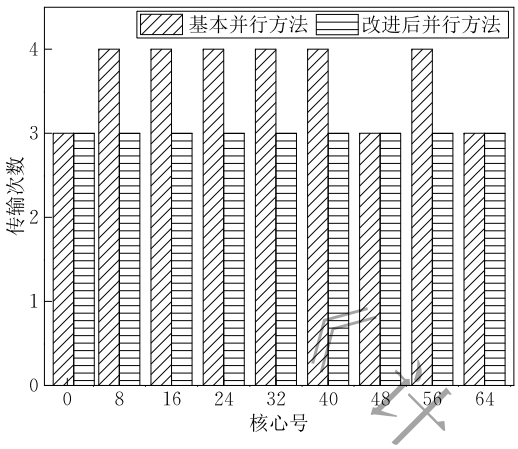


图 13　不同实现方式的数据传输次数比较

4.3.2　计算优化方法对算法性能提升的贡献

本文提出的计算优化方法主要为平衡数据传输方法带来的额外计算开销. 基本并行方法算法的数据计算过程一般为简单的矩阵计算操作而由于数据传输方式和计算核心数据组织结构的改变，使得改进后方法的数据计算过程增加了许多额外的逻辑判断操作. 本文的方法通过对计算过程的优化使得优化后的计算过程执行时间与基本并行方法的计算执行时间基本相当，且由于计算时间并不是算法执行流程的主要耗时部分，故相应的开销并不影响算法整体性能的提升. 从图 10 中可以看出，改进后的算法计算过程所需时间相较基本并行方法虽然有所提高但对算法执行效率并没有显著影响.

4.4　通用 CPU 与众核处理器运行时间对比分析

本节首先对算法的并行效率进行了衡量. 这里我们使用 gx3 精度的数据对海冰模式算法进行了 72 小时迭代，表 6 显示了使用控制核心(MPE)执行计算与使用控制核心和计算核心协同计算(CPE+MPE)执行计算时算法的运行效率.

表 6　算法在 Intel CPU 和申威众核上执行时间比较

	1CG	2CG	4CG	8CG
Reconstruct 算法－MPE(s)执行	8.18	4.14	2.05	1.15
Integrating Field 算法-MPE(s)执行	6.81	3.65	1.84	1.04
Reconstruct 算法-MPE+CPE(s)执行	0.70	0.42	0.25	0.18
Integrating Field 算法-MPE+CPE(s)执行	0.69	0.41	0.29	0.16

由表 6 可以看出使用 MPE+CPE 协同执行的方式与只在 MPE 上执行相比，加速比最高可达 11.6 倍. 由于一个 MPE 的计算峰值为 23.2 Gflops，CPE 计算峰值为 1.6 Gflops[11]，故理论上来说一个核组的计算能力相当于 33 个 MPEs. 考虑到数据传输带宽的限制，本文所提出的并行策略达到了较好的并行效果，取得的最大并行度已可以达到 36%.

表 7 给出了本文所采用的海冰模式算法在 Intel Xeon 处理器上使用多核心并行执行的运行时间. 该结果显示与申威众核并行化结果相比，在采用相同核心(核组)数执行海冰模式算法时，申威众核处理器可取得 3 倍左右的加速比，而申威众核处理器取得这一结果所需的能耗远小于 Intel CPU.

表 7　算法在 Intel CPU 多核并行执行时间

	1 core	2 cores	4 cores	8 cores
Reconstruct 算法/s	3.29	1.81	0.98	0.68
Integrating Field 算法/s	1.90	1.05	0.59	0.42

同时表 8 对比了 Intel CPU 和申威众核处理器在并行度增加时算法的可扩展性. 从表 8 中可以看出在并行度增加时算法在 Intel 处理器取得的可扩展性优于本文基于申威众核处理器提出的并行方法. 这主要是由于每个核组执行的任务分到 64 个计算核心上执行，当计算资源增加时每个计算核心分得的任务计算量有时差异较大，从而造成该算法可扩展性略差于算法在 Intel 处理器上的执行结果.

表 8　算法在 Intel CPU 和申威众核处理器可扩展性对比

	1 core	2 cores	4 cores	8 cores
Reconstruct 算法(Intel)	1	1.81	3.35	4.83
Integrating Field 算法(Intel)	1	1.80	3.22	4.52
Reconstruct 算法(SW)	1	1.66	2.80	3.88
Integrating Field 算法(SW)	1	1.68	2.37	4.31

4.5　并行方法适用性分析

根据 4.2 节和 4.3 节的结果，可以得出该并行方法对于内存计算密集型的应用具有显著的加速作用，其对于传输过程的加速效果主要体现在两个方面：(1) 通过传输感知的传输方式可以加大一次传输的数据量，进而减少传输次数，增大每次传输的带宽；(2) 通过对数据的压缩，在中间计算结果需要存储时可以减少传回主存的数据量，从而减少数据传输时间. 但在实验过程中发现该方法存在以下三点限制：(1) 该方法受限于传输缓冲区的大小. 在实验中对于 160×192 的网格尺度，传输缓冲区一次最多只能存放 1 行数据，这限制了最多传输的数据量，使得改进后方法与原方法数据传输次数相当；(2) 该

方法受限于计算时分配的网格大小.当分配网格过小时,预测传输量还没有增长,无法体现该算法优势.如表 4 中的 40×92 和 40×48 的网格情况;(3)该算法对于输入变量众多且每行数据量较大的情形提升效果并不明显,如表 4 中 Integrating Field 算法在 80×96 网格尺度下的情形.这是由于众多的输入变量使得分配给每个变量的空间大小有限,缩小了每个变量数据传输量的提升空间从而影响了每次数据传输量的提升.但总体而言,该方法对于计算和内存密集型的应用展示出良好的性能提升效果与潜力.

5 总结和未来工作

地球模式相关模式的并行化一直是科学计算与高性能领域的热点.申威众核处理器的发布,为地球模式的并行执行提供了新的平台.本文选取地球模式中海冰模式中的 Reconstruct 和 Integrating Field 两个算法,对海冰模式算法在申威众核体系结构下实现细粒度并行化的方法进行了研究.本文提出了一种基于数据分割,数据传输与计算优化的并行化方法.该方法与基本并行方法相比,程序的运行时间最高缩短了 40%,与原始程序在通用众核管理核心上执行相比加速比最高可为 11.6 倍和 9.8 倍.

本文主要研究集中在细粒度层面上的海冰模式算法并行化方法,对于多核情形下的情况并没有做出深入研究.从目前对海冰模式算法进行数据分割后的不同数据在不同核心上的执行时间来看,不同核心间算法执行时间有着比较大的差异,并随着计算量的增加,差异越发明显.今后将改进核心间任务分配的策略并考虑将算法扩展到多核的情形,进而对多核情形下算法的扩展性和并行度进行进一步研究.同时将结合大规模地球模式计算的特点,探究本文提出的方法对其他地球模式算法的适用性.

参 考 文 献

[1] Manferdelli J L, Govindaraju N K, Crall C. Challenges and opportunities in many-core computing. Proceedings of the IEEE, 2008, 96(5): 808-815

[2] Huang X M, Wang W C, Fu H H, et al. A fast input/output library for high-resolution climate models. Geoscientific Model Development, 2014, 7(1): 93-103

[3] Xu S, Huang X, Oey L Y, et al. POM. gpu-v1. 0: A GPU-based princeton ocean model. Geoscientific Model Development, 2015, 8(9): 2815-2827

[4] Krishnamoorthy S, et al. Effective automatic parallelization of stencil computations. ACM SIGPLAN Notices, 2007, 42(6): 235-244

[5] Holewinski J, Pouchet L-N, Sadayappan P. High-performance code generation for stencil computations on GPU architectures //Proceedings of the 26th ACM International Conference on Supercomputing. Salt Lake City, USA, 2012

[6] Nguyen A, Satish N, Chhugani J, et al. 3.5-D blocking optimization for stencil computations on modern CPUs and GPUs//Proceedings of the 2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC). New Orleans, USA, 2010: 1-13

[7] Fu Haohuan, et al. The Sunway TaihuLight supercomputer: system and applications. Science China Information Sciences, 2016, 59(7): 1-16

[8] van Leer B. Towards the ultimate conservative difference scheme. Journal of Computational Physics, 1997, 135(2): 229-248

[9] Yeh T Y, Patt Y N. Alternative implementations of two-level adaptive branch prediction. ACM SIGARCH Computer Architecture News, 1992, 20(2): 124-134

[10] Griffies S M, Biastoch A, Böning C, et al. Coordinated ocean-ice reference experiments (COREs). Ocean Modelling, 2009, 26(1): 1-46

[11] Fu H, Liao J, Xue W, et al. Refactoring and optimizing the community atmosphere model (CAM) on the Sunway TaihuLight supercomputer//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Salt Lake City, USA, 2016: 83

附录 1.

表 9 申威众核处理器主存和计算核心之间数据传输带宽(64 核心同时传输)

数据传输量/B	输入带宽/(GB/s)	输出带宽/(GB/s)
256	3.326591441	4.047927461
512	3.776891467	4.488652686
768	5.585011319	7.271951598
1024	7.509311546	9.593246355
1280	9.587065898	11.927480920
1536	11.589813330	13.406263410
1792	13.747486170	14.860733700
2048	16.011271940	15.911405300

(续 表)

数据传输量/B	输入带宽/(GB/s)	输出带宽/(GB/s)
2304	17.884395527	16.759027553
2560	19.089798441	17.497200445
2816	17.538265331	17.105394111
3072	18.720047992	17.918577998
3328	19.563228335	18.496175556
3584	20.264011112	19.351556997
3840	20.945040221	19.867339115
4096	21.470285113	20.211819187
4352	21.763621447	20.525848408
4608	22.234959529	21.009188802

LI Bin-Yang, born in 1992, M. S. His research interests focus on high performance computing and distributed system, etc.

LI Bo, born in 1973, Ph.D., associate professor. His research interests include high performance computing and high performance server.

QIAN De-Pei, born in 1952, professor, Ph. D. supervisor. His research interests include high performance computing architecture, etc.

Background

High-resolution climate modelling remains a significant scientific and engineering challenge because of the enormous computing, communication, and storage requirements. In recent years, many heterogeneous architectures have been proposed and have achieved much higher peak performance than ever before. Porting and parallelizing climate model on the new hardware platforms is one way to accelerate high resolution climate modeling process. Parallel programming methods, such as MPI and OpenMP, have been widely used to support the parallelization of climate models. However, supercomputers are becoming increasingly heterogeneous and diverse, involving devices such as the GPU and the Intel Many Integrated Core (Intel MIC), new approaches are required to effectively utilize the new hardware.

When compared with other HPC application domains, the porting of the climate models onto many-core architectures has been relatively slow. Many researches are based on GPU. The typical examples include GPU-based accelerations for the ROMS (Regional Ocean Modeling System), and the POM. gpu-v1. 0 which redesigned the parallel version of the Princeton Ocean Model to fit the GPU.

In recent years, some climate models are ported to the Sunway processor. Haohuan Fu has ported and refactored the CAM on the Sunway TaihuLight supercomputer and other efforts are in progress now.

Different from the projects mentioned above, our work focuses on the sea ice model. Moreover, we concentrate our efforts on propose a framework which need to dig into the algorithms as well as the new architecture.

This project is sponsored by the National Natural Science Foundation of China (No. 61133004, No. 61502019) and the National Key R&D Plan (2016YFB0200100). This project is a subproject of National Supercomputing Center of Sun Yat-sen University Specific Open Program.