为CCSM4和CESM1而研制的新一代可扩展的

用于地球系统建模的耦合器[[1]](#footnote-1)

Anthony P. Craig, Mariana Vertenstein

美国国家大气研究中心

Robert Jacob

阿岗国家实验室

**摘要**

通用气候系统模式（CCSM）已经开发了10多年，且被用于理解过去、现在和将来的气候。模式的最新版本，CCSM4和CESM1，在CPL7耦合器中包含了很多全新的耦合特性，它拥有更好的灵活性和更强的扩展性以解决在地球系统建模中遇到的挑战和问题。CPL7耦合构架在系统上层设计上采用了全新的方法。CCSM4现在拥有位于顶层的驱动器，驱动器通过特殊的接口来调用分量模式，驱动分量模式的初始化，运行和结束。顶层驱动器可使分量模式置于相对任意的硬件处理器布局中，并以串行、并发或者两者混合的形式运行。为了支持更高分辨率的设置，耦合器的内存使用和性能调试都得到了改进。在更大规模处理器使用情况，CCSM4有更好可扩展性，而且其有能力处理全球分辨率的为十分之一度的情况。

**关键词**： 耦合，气候，模型，CCSM，性能，扩展性，耦合器，CPL7

**引言**

**背景**

气候建模是一种复杂的多物理过程的应用，也是当今高性能计算的挑战之一。通用气候系统模式（The Community Climate System Model，CCSM）是一个先进的全球气候耦合模式，它包含了以下几个基本物理分量模式：大气模式，陆面模式，海洋模式以及海冰模式。通用地球系统模式（The Community Earth System Model，CESM）增加了新的陆地-冰分量模型，陆地和海洋地球生物化学功能模块，大气化学模式，以及可使大气模式扩展到更高高度的功能。CCSM和CESM的分量模式既可作为“数据模式”从文件中读取耦合数据，也可作为“活动”的动力分量模式直接预报耦合场。各种动力分量模式通常包括了流体力学求解器和详细的参数化过程，以便计算内部和外部的强迫项。这些强迫项来自各种各样现象，例如辐射穿透大气，水相变时的潜热释放，摩擦作用或者不可分辨尺度的湍流。

当CCSM/CESM各分量模式在时间上向前计算时，他们不断的利用耦合软件周期性的交换边界数据。这种耦合软件支持分量模式间数据的通信，不同分量格点数据插值，将几个分量模式的场合并到一个“目标”分量，以及生成各种如总体收支这样的诊断量。CCSM3 (Collins et al., 2006a使用 CPL6耦合器软件 (Craig et al., 2005)。下文中，我们将概述以成为CCSM4和CESM1一部分的新的CPL7耦合软件的上层设计和性能。在以下讨论中，“CCSM4”和“CPL7”将分别用于描述CCSM4和CESM1的性能。

耦合器是气候模式的关键成员。特别地，模式间的耦合通常涉及至少三个方面：耦合体系，通讯构件以及耦合方法。耦合构架通常跟系统的总体控制有关，包括制定分量模式的时间次序。通讯构件支持在不同分量间数据的传输，根据总体设计，通讯构件可以从上层启动器子程序中调用或者在模式分量中直接实现。最终，耦合方法提供了多种功能，例如在不同分量模式网格间映射（插值），计算物理属性如通量或者是计算诊断量。每个耦合气候模式的应用程序都需要解决这些方面的问题，所以他们一般综合使用各种通用工具（比如ESMF，MCT，OASIS，MPI或者NETCDF）以及自带的软件来实现。

耦合体系可以按照四种基本特性大概地来分类：数据是通过一个特定的耦合“枢纽”还是直接在分量间通信，数据的通信是通过上层驱动器调用还是直接在分量模式间调用，分量模式如何在硬件处理器上分配，系统是一个独立的还是多个可执行程序。CCSM一直以来使用的是“枢纽”式设计，将来也如此。为了将各种独立开发的模式耦合在一个单独的应用程序中，CCSM在过去15年中开发了一套耦合的构架，可以允许分量模式在尽可能少的修改下进行耦合。

当讨论耦合应用程序时，分清时间上的并发和处理器上的并发是很重要的。分量模式可以各自运行在专有的处理器上，可是如果它们运行时在时间上是依次运行的，那么一个并发的处理器布局实际上将是按串行运行。两个模式只有当它们运行在专有的处理器上，而且分量模式之间不存在限制并发的时间依赖性的时候，它们才能在处理器上同时运行。

单一应用程序设计并不自动地意味着模式只能在同样的处理器上顺序运行。通常，分量模式都是使用单个应用程序在相对任意的处理器进程组上分配。另一方面，多个可执行程序通常也意味着模式分量同时运行在独立的处理器集上，这是因为硬件系统一般不允许多个应用程序共享相同的处理器以及交错操作。

特定的耦合气候模式的实现基本上取决于耦合体系的选择，而这一选择则与分量模式的先后次序、耦合频率和延时、并发性、科学需求，作业启动，分量模式的集成度以及交互性有关。例如，the Parallel Climate Model (PCM) (Washington et al., 2000)引入了一种高层驱动器以便支持个分量模式在单一应用程序顺序执行。在这个例子中，耦合器就是一个驱动器，它通过子程序界面调用分量模式，执行顺序和耦合操作则按照特定的科学需要来制定。另一方面，OASIS耦合器(Valcke et al., 2006a)是一种更通用的耦合耦合器组件，它被设计为可以支持多个可执行分量模式的耦合，分量模式间的通讯通过调用一个称为PRISM的公用耦合接口来实现(Valcke, 2006b)。OASIS支持枢纽式耦合和分量间直接耦合。这两个耦合器的设计从根本上是不一样的，但是他们都满足了他们科研社团的科学需求。其他气候模式耦合器的例子还有FMS(http://www.gfdl.noaa.gov/fms)，FOAM的定制耦合器(Jacob et al., 2001)，PALM(http://www.cerfacs.fr/globc/PALM\_WEB）以及FLUME。其他的耦合器的开发库包括MCT(Larson et al., 2005)，ESMF(Hill et al., 2004)以及the Distributed Data Broker(Drummond et al., 2001)。

**动机**

在CCSM4之前，CCSM是以一种多个可执行程序系统来运行的，在这个系统上所有的模式都同时在各自的硬件处理器上运行而且每个分量模式都是一个独立的二进制程序。各分量模式独立启动，然后按照一定的时间间隔使用发送和接受的方法与耦合器通信。耦合器是一个独立的二进制程序，而且作为中央枢纽协调数据交换，管理延时和次序，执行耦合操作如映射（插值）和合并。实际上，耦合器的次序是很难搞清楚的，因为个分量模式中都内嵌了通讯操作。虽然在CCSM3中已经努力把并发度提到最大以增加吞吐量，多个可执行程序并发的设计在某种配置下会限制模式的吞吐量。另外，CCSM3的设计有时会让移植和调试工作富于挑战，因为在某些平台上缺少多个可执行程序作业启动的支持。所以，之前的CCSM并没有跟ESMF所提倡的标准协调一致。

最近在CCSM4中，分量模式的物理算法有了改进，包括了更新和改进的大气边界层方案，新的辐射和表面反照率算法，这些改进由于稳定性的原因需要分量模式间比过去耦合得更频繁。而随着分辨率的提高，分量模式的时间步长减小了，耦合的频率也增加了。由于最近需要更高频率的耦合，CCSM3处理并发可执行文件的限制显得越来越关键。

在过去，CCSM的大气分量CAM(Collins et al., 2006b)以及陆面分量CLM(Bonan et al., 2002; Oleson et al., 2010)，都是可在相同的网格上运行的独立耦合系统。他们都是1980年代NCAR的CCM和LSM模式的扩展。独立的CAM模式是一个大气、陆地-地表、数据-海洋（使用给定的海表温度），以及只含热力学海冰（使用给定的冰盖值）的耦合系统。独立的CLM模式包括了数据-大气以及预报的陆地-地表耦合分量。在过去的十年中，这些独立模式与CCSM一道发布和受到支持。为了代码的统一，减少维护，以及一些扩展的特性，如可以在单处理器上运行等，CCSM4的开发者们希望保持独立CAM和独立CLM系统的某些特性。

更新CCSM耦合框架有几个原因：为了更好的支持新科学，支持完全串行和单处理器积分，为了更容易移植和使用转为单独的可执行程序系统，增加分量在硬件处理器上布局的灵活性以提高针对更广泛问题的性能，支持通过ESMF超级构架耦合的能力，针对更高分辨率和更多处理器数目情况提高性能和内存的可扩展性。

**设计**

**概要**

在CCSM4中使用CPL7，系统的顶层框架和设计都使用了全新的方法。目前这个系统包括一个在所有处理器上运行的顶层驱动器，以及通过标准结构调用的可以在所有处理器上或者其任意子集上运行各个分量模式。驱动器还会直接的调用耦合器来映射（插值），重排，合并，计算通量以及产生诊断量。在CCSM4中，分量模式以及耦合器的方法能够在所有处理器的子集上运行。实际上，CPL6的顺序和枢纽属性已经被迁移至驱动器级别。但是，CPL6耦合器上的操作如映射和合并在CCSM4中仍如单独的耦合器分量那样完成。

CCSM4已经被转变成一种单一可执行程序的系统，这种系统继续提供处理器的并发布局并且支持各种分量模式串行运行或者以串行/并发的混合模式运行。这极大地扩展了分量模式布局的灵活性。这种更广泛的布局选择给了用户在优化负载均衡以及配置模拟时更多的灵活性。CPL7同样地经过重写后已经明显地减少内存使用并提高了内存可扩展性。之前的CCSM版本运行在1度到5度这个全球分辨率范围，所以内存使用还不形成一个瓶颈。如果给定问题的规模， CCSM4现在支持在拥有几万个处理器的超大规模并行机上运行十分之一度全球分辨率，这些并行机的每个处理器的内存都相对有限。

CCSM4支持数据模式和活动分量模式。在CCSM4中，如同CCSM3，大气，陆面，海冰模式都紧密地耦合起来以便更好的分辨日变化循环。重要的是，大气-海洋通量是在耦合器中以相同的频率计算出来的（而不是在海洋分量模式中计算的），所以大气、陆面和海冰模式之间可以通信。类似的，海洋表面反照率的日变化循环也是在耦合器中计算而且由大气模式使用。这种耦合通常是每小时一次的，虽然在高分辨率下耦合会更频繁。因为海洋模式不会计算大气/海洋通量，所以它通常每天耦合一到几次。这种宽松的海洋耦合频率说明海洋的状态和响应在系统中是滞后的。这样就允许海洋模式运行在跟其他分量模式相独立的一组处理器集合上。在CCSM4中有一个选项可用于运行紧密耦合且无滞后的海洋模式，可是这仅仅在当运行数据海洋分量模式时使用。

根据分辨率、硬件、运行时间长短和物理过程，CCSM4的运行通常需要几个小时到几个月的墙钟时间完成。模式模拟时间通常为几十年到几百年，而模式每天一般可以模拟从1年到50模式年。CCSM4能够精确的重启，所以通常模式都是一年一年或者几年几年的运行。

**串行和并行**

CCSM4使用Message Passing Interface(MPI)进行模式间以及模式内部的通讯。很多分量模式使用混合的并行模式，即在每个MPI任务内使用多个OpenMP线程。在CCSM4中，分量模式的处理器布局以及MPI通讯域是从一个简单的由用户指定的namelist中输入得到的。目前，在CCSM4中有几种基本的MPI进程组。它们分别跟大气，陆面，海洋，海冰，陆冰，耦合器以及全局进程组有关，而其他的进程组也可以被容易的添加。每个基本MPI进程组对应的是不同的处理器，而用户还可以相对任意地在硬件处理器上重叠MPI进程组。如果在一个处理器上有多于一个进程组，那么这些进程组里的分量模式将会依次运行。每个进程组在运行时使用三个变量描述：MPI任务数，每个MPI任务的OpenMP线程数，以及该进程组根进程的全局MPI进程编号。比如说，配置MPI进程数为8，每个MPI进程的线程数是4，而且根进程是MPI全局进程编号16，那么这样的一个处理器组群将包括32个硬件处理器，从全局MPI进程编号16开始，而且它将由8个MPI进程。CPL7驱动器会在一开始获取所有的MPI通讯域，接着把它们传递给各分量模式使用。该输入信息用于设置MPI进程组以及批处理和作业启动参数。

正如在引言中提到的，决定分量模式是否并发运行有两个方面。首先，工作的唯一区块是否运行在不同的进程组上。其次，该任务在驱动器中的顺序。CCSM4的驱动顺序已经尽可能的实现分量间的最大并发度。理想情况下，在单个耦合时间步中，所有模式的强迫项都先计算。因此模式能够并发的运行，接着驱动器的时间会向前一步。然而，像地表反照率和大气辐射的坐标以及边界层稳定度等科学上的需要使得对耦合延时加上了限制。图1说明了CCSM4支持的一个完全活动系统的配置所达到最大并发度。更多的并发在技术上是可行的，但是从科学上加了一定的限制条件，例如在大气模式和陆地海冰模式的耦合时，大气模式跟这两个地表模式是依次运行的。图1并没有完全给出对于任何配置都达到性能最优的处理器布局，可是它提供了对一个系统目前可以支持的实际并发度的上限。值得一提的是，无论处理器的布局如何， CCSM4的结果都是在数值舍入之内的。计算结果之所以不能达到字节上完全一样，是因为一些物理分量模式在处理器数发生变化时有舍入级别的变化。

|  |  |
| --- | --- |
|  | 图1.CCSM4中基于科学问题的并行能力 |

跟CCSM3相比，CCSM4在并发上会有一定的损失。在CCSM4中，模式运行通过驱动器调用完成，所以说从整个系统的角度来看，耦合就像“发送至分量模式，运行分量模式，从分量模式接收”一样。在CCSM3，耦合直接在每个分量模式中调用完成，而且发送和接收的调用是可以在工作中交错的，这样模式运行方法表现为两个阶段，一个位于耦合发送和接收之间，另一个位于接收和发送之间。虽然在CCSM3中允许更大程度的并发，但是它仅仅是为并发设计的。在CCSM4中，多运行阶段不是为了使简化模式以及给更好的互操作提供一个机会而实现的。所以，这样的决定是建立在清楚了解对某些特殊情况下并发有损失的情况下做出的。然而，我们觉得这种可以让模式于串行和并发混合的系统有更好的可扩展性，可克服因潜在的并发损失而导致的任何性能下降。

**分量模式接口**

CCSM4分量模式接口包括了各个分量模式间参数相协调的初始化，运行和结束的方法。虽然标准的CCSM4接口参数目前包括了Fortran和Model Coupling Toolkit(MCT) (Larson et al., 2005, Jacob et al., 2005)的数据类型，另一可用的版本是跟ESMF超级构架相兼容的。作为初始化的一部分，格点和分量模式划分的信息将从分量模式传递给驱动器。驱动器和耦合器从输入文件或者通过跟分量模式通信来获取实时的关于分辨率、配置以及处理器布局的所有信息。物理耦合场也会在初始化、运行和结束阶段通过接口传递。运行接口还包括了一个时钟数据的实例用于设定模式的当前时间和运行时间长度。

CCSM4的系统初始化比较直接。首先，各种MPI通信器在驱动器中被设置好。接着，分量模式的初始化方法在正确的进程组上被调用。一旦驱动器从分量模式获得格点和及其分解信息，各种重排函数和映射函数也被初始化，以便在处理器之间移动数据，格点及其划分也是驱动器层次需要的。耦合器的实现对于串行还是并发执行没有差别。总的来说，甚至对于那些具有相同格点和处理器布局的两个分量模式，他们的给点分解也会由于性能的原因有所不同。在那些分量模式之间的格点，格点分解以及处理器布局都一样的情况下，映射和重分配就会变为局地的数据拷贝。

耦合器，陆面，海冰，大气和海洋分量模式的CCSM4配置的驱动器运行流程见图2。这种操作顺序是CCSM4中编码实现，而且其首先满足科学上的限制，其次性能的优化。图2中的字母“C”显示了数据在耦合器与其他分量模式间通信时有一个耦合的操作。这一步引入了分量模式间的依赖性，其中，w表示一个分量模式在通信进行前通常要等待另一个分量模式。图2中的字母R表示这是分量模式独立完成的工作。每一步中，对大气，陆面和海冰模式的耦合都会发生，而对海洋的耦合的频率会少些。图2还说明了如果分量模式并发运行，那么海洋会首先开始运行，接着是陆面和海冰。海冰和陆地模式的数据传递给耦合器，这些数据是用来计算大气模式的耦合数据的，接着又会被耦合器发送给大气分量模式，此后大气模式才会运行。从设计上来说，海洋总是可以跟其他所有的分量模式一起积分运行的。图2提供了详细的实现方法，而这也限制了在图1中的总并发度。耦合器会一直在通信时间步之间计算，其中有些工作是跟其他分量模式重叠的。一般说来，耦合的计算和通信开销跟那些活动模式的运行时间相比是较少的。



图2：启动器层面的循环操作次序。方格代表运算，箭头代表资料耦合。虚线代表驱动器的时间循环

**额外软件和并行输入输出**

CPL6是使用Model Coupling Toolkit(MCT)库，CPL7也继续严重上依赖MCT。MCT支持很多关键的耦合需求，例如管理数据的并行和执行并行重映射。CPL7中的上层数据类型大部分都是未经修改的MCT数据类型，而MCT则用于在处理器之间数据，格点和格点划分的重排和映射。虽然CPL7的耦合构架依赖于MCT的内部框架，上层外部框架则被设计为和ESMF(Collins et al., 2005)兼容，所以CCSM4分量模式也是跟ESMF兼容的。映射的权重离线由SCRIP软件包生成好(Jones, 1998)。为了让内存使用最小化，映射权重进入CCSM4使用的是一种能在合理小的区块中读取和分配权重的方法。

CCSM4比CCSM的以往版本都更致力于提高分辨率。所有分量模式都致力于减小内存的使用而且提高内存使用可扩展性，其目标就是为了能够在几万个处理器上运行十分之一度的完全耦合系统，而其中的每个处理器可只有512M内存。这个目标使得那些每次可以在任何一个处理器上分配的非分布式数组的数目变得较少，所以在初始化时会导致显著的分量模式代码重写。另外，在高分辨率下传统串行I/O的内存量也是不能接受的，所以更高分辨率的串行I/O会成为一个显著的性能瓶颈。为了解决I/O的问题，一个并行的I/O（PIO）库(Dennis et al, 2011b)已经在CCSM共同体中开发从而解决模式中I/O性能和内存使用的问题，该库可以为NETCDF，pNETCDF([www.mcs.anl.gov/parallel-netcdf](http://www.mcs.anl.gov/parallel-netcdf))以及MPI-IO提供统一的接口。所有的模式分量目前都在大量使用PIO软件，而且PIO跟分量模式的代码重写配合起来可以让CCSM4运行在以前由于内存限制而不能达到的分辨率下。

**性能、扩展性和负载平衡**

为了可以在几万个处理器上有更好的可扩展性，所有分量模式的开发者已经开始通过改进算法，内部构架和格点划分的途径来提高性能。特别的，为了在分量模式中提高并行度以及改善可扩展性，共享内存块的分割，空间填充曲线(Dennis et al., 2011a)，以及所有三个空间维度分解都在某种程度实现。

CCSM4的负载平衡涉及在处理器上分配和布局分量模式以优化性能和最小化空闲时间。CCSM4的性能，负载平衡以及可扩展性是受问题的大小，复杂度以及多模型特征制约。在系统中，每个分量都有它自己的可扩展特征，由于内部负载不平衡，分解能力以及通信开销，一个分量模式可扩展性会发生变化。分量模式的性能同样会随着模式向前积分而发生改变。造成这种情况的原因有模式中物理过程的季节变化，在模式自我适应阶段性能的改变，以及调用某些随时间变化的物理过程，例如辐射，动力学，和I/O。有了这些限制，我们总可以为CCSM4建立一种配置，这种配置拥有在允许范围之内的空闲时间以及足够合理吞吐量。CCSM4已经显著的提高了可能的处理器布局的灵活性，而这已经使得该版本比CCSM的前一版本有了更好的负载平衡。

在实际中，CCSM4的负载平衡涉及到多方面的考虑，例如哪些分量模式将被运行，分量模式的分辨率是多少，各个分量模式的相对分辨率是多少；对于每个分量模式的开销、可扩展性以及处理器数量；一个分量模式中的内部负载不平衡。最好将重要的实时I/O关掉后再对系统进行负载平衡，因为这通常发生得并不频繁（在CCSM4中一个月通常只有一个时间步），因此可以当做另外的开销而且可能扭曲对总体模式负载平衡的解释。利用OpenMP的能力以及在一些或者所有分量模式上OpenMP线程的性能不但取决于硬件或者操作系统的支持，还取决于系统是否支持对不同分量模式在重叠的处理器上运行完全MPI和混合MPI/OpenMP。最后，处理器的布局，是否是串行还是并发，或者是两者的组合都可能发生变化。通常，会对于一个产品级任务（production job，长时间运行大规模输出的模拟），根据所希望的产品级配置（production configuration）完成一系列短的测试运行以建立一个合理的负载平衡设置。CCSM4提供了对性能和系统负载平衡的的后处理分析用来帮助用于改善处理器布局。

**结果**

本节将给出四种不同耦合器内核性能的可扩展性的结果。接着一些完整的模式结果会被用来说明硬件处理器的分量模式布局如何影响模式的总体性能。这四种将讨论到的核心计算是海洋格点上一个变量场的合并，大气海洋通量计算，海洋数据不同分解间的重新排以及大气格点到海洋格点的映射。这些核心代表了最常见的CCSM耦合器操作。结果将在三个不同的硬件平台bluefire、jaguarpf和intrepid以及在两种不同的模式分辨率“f19\_g16”，“f05\_t12”下给出。

Bluefire是在国家大气科学中心（NCAR）的IBM SP6，它拥有4096个4.7GHz的处理器，每个处理器4GB内存。处理器被分组为32向的节点，每个处理器的运算每个时钟周期达到4FLOPS（浮点运算）。其间的互联使用的是InfiniBand交换机，每个节点有八个4XDDR连接。Bluefire支持同时多线程（SMT），允许64个MPI任务被指定到每个节点上，所以在Bluefire上所有的计时结果使用的都是SMT。Jaguarpf是在橡树岭国家实验室（ORNL）国家计算科学中心（NCCS）的Cray XT5，它拥有18688个双16核AMD的2.6GHz处理器的节点，每个节点有16GB内存。可用的处理器的总数为224256，其间的互联使用SeaStar 2+路由器。Intrepid是在阿贡国家实验室(ANL)的IBM Bluegene/P，其拥有163840个850MHz的PowerPC450处理器。每个共享内存节点有4个处理器，每个机架有1024个节点，每个节点有2GB的内存。

中等分辨率格点“f19\_g16”包括了分辨率是大致2度的144×96大气/陆面格点以及一个分辨率大致1度的320×384海洋/海冰格点耦合。高分辨率格点“f05\_t12”拥有分辨率大致0.5度的576×384大气/陆面格点以及分辨率为大致为0.1度3600×2400海洋/海冰配置。所有的情况都是在共享内存并行机制（OPENMP）关闭的情况下运行的，每个处理器上只有一个MPI任务，但在Bluefire的IBM SP6上，按照上述使用的是SMT模式，而在Intrepid上的分辨率“f05\_t12”设置，由于内存限制，每个MPI任务只制定给一个4向的节点。

在所有机器上的测试都是在产品级环境下运行的。因此，我们在用时上会看到有一些差别。为了更好的理解这些差别，所有实例都会重新运行，而我们给出的一般都是最好的用时结果。所有实例都会使用“dead”的模式设置（这将用分析测试数据替换掉物理分量）运行20天，中间不涉及任何历史和重启I/O输出。所有的计时器都会跟MPI\_BARRIER调用分离开，而我们给出的是最慢任务所用的时间。在每个20天的积分中，每个内核都会被调用960次，而这些时间是所有调用和每个MPI任务的总和。图中给出的时间单位是模式每模拟天所用的秒数。所有的结果都是使用对数/对数的形式给出的，为了便于比较每个图使用的是同样的水平和垂直坐标刻度。另外，同样的线性标度参考线也在图中给出了，水平坐标轴给出的是测试实例耦合器所用的MPI任务数量。模式的性能是基于标准的“立即可用” 的CCSM编译器，作业和环境设置，没有经过任何额外的一个个例一个个例的微调。

图3给出了CCSM耦合器中海洋场合并核心的性能。该核心负责在海洋格点上合并不同分量模式的不同场。这种核心很容易并行（没有通信），只是主要受到内存访问的限制。在更高的“f05\_t12”分辨率下，所有平台的可扩展性能都会在超过1000个处理器的所有机器上得到几乎线性的扩展。相反的，所有平台的“f19\_g16”分辨率的可扩展性能在100个处理器一下的机器上就得到几乎线性的扩展。超过100个处理器之后，性能在Bluefire和intrepid上会下降，而jaguarpf的性能在超过1000个处理器后才会下降。



图3：CCSM耦合器中海洋场合并核心的性能。这里用了两种分辨率（‘f19\_g16’和‘f05\_t12’），三个硬件平台（bluefire, jaguar, intrepid）



图4：CCSM耦合器中大气/海洋通量计算的性能。这里用了两种分辨率（‘f19\_g16’和‘f05\_t12’），三个硬件平台（bluefire, jaguar, intrepid）

无畏

图4给出了大气/海洋通量计算的性能增长。这种计算容易并行而且是在海洋格点上计算的。跟海洋合并核心相比，数学运算主要包括加法，乘法，除法，求最小/最大，对数，指数和开方。跟图3对比，每次内存载入的FLOPS数是相当高的。甚至在中等分辨率下，所有机器的增长度至少在1000个处理器是几乎达到线性。这种核心的绝对性能在Bluefire和jaguarpf上几乎是一样的，可是Intrepid在这种高FLOP强度的计算下大概会比另两个机器慢超过10倍。



图5：CCSM耦合器中海洋到海冰重新排列的计算性能。这里用了两种分辨率（‘f19\_g16’和‘f05\_t12’），三个硬件平台（bluefire, jaguar, intrepid）



图6：CCSM耦合器中大气到海洋映射计算的性能。这里用了两种分辨率（‘f19\_g16’和‘f05\_t12’），三个硬件平台（bluefire, jaguar, intrepid）

图5描述了海洋格点上重排操作的可扩展性。在这个核心上没有FLOPS计算，只是内存访问和在同样一个格点上的海洋分区和海冰分区之间数据的MCT重排。这种重排在处理器之间几乎是全部数据的通信，所以其性能主要受到通信开销的制约。在所有的测试例子中，增长性同样在两种分辨率下都是次线性的。Jaguarpf中等分辨率“f19\_g16”的增长性在100个处理器时就会显著的下降，超过100个处理器后甚至会变慢（存在一个波动）。Bluefire在更高的“f05\_t12”分辨率下，扩展性好一些，可是它仍然在超过500个处理器后就变得平缓，Intrepid的在超过1000个处理器后也变平了，大约在500个处理器时Jaguarpf的性能会重新反弹。

图6给出了CCSM中大气格点到海洋格点映射的性能。核心是通信以及乘法-加法操作(Jacob et al. 2005)的混合，而且采用MCT映射方法。在这种情况下，映射权重的分布总是基于初始时的海洋分区。当该核心被调用时，用于映射的大气数据会重排到海洋分区，在每个处理器上，大气数据都会被要求映射到可获得局地的海洋格点。在许多例子中，大气数据被重排至多余一个海洋处理器。总的来说，这种核心的性能跟图5重排核心的性能是类似的，这也说明了重排操作占映射操作开销的。对于中等分辨率“f19\_g16”而言，所有机器的扩展度大约在100个处理时就开始变平或者转向。除了jaguarpf在500个处理器之后会迅速变平之外，“f05\_t12”分辨率在1000个处理器以下的可扩展性都很好。

对于一个CCSM耦合个例，图7给出了不同处理器布局可能造成性能改变的一些内部信息。从图7可以看到在Bluefire上以128个处理器运行一个“f19\_g16”中等分辨率预报实例的各个分量模式的总体性能。（a）是一个在处理器上分量模式完全并发的布局，（b）是一个在处理器上分量模式完全串行运行的布局，（c）除海洋模式外，系统的其他分量模式均串行运行，而海洋模式又跟其余的分量模式并发运行，（d）是一种更通用的串行/并发混合的布局。在所有的实例中，处理器的总数均为128，而表示为灰色方格的空闲时间都尽可能的少。对于（a）并发布局，（b）串行布局，（c）串行和海洋并发布局以及（d）混合串行/并发布局，四种配置的总运行时间分别是33.5，20.8，21.8和19.1秒每模式天。由于启动器中大气，陆地和海冰模式的次序有限制，（a）完全并发布局会存在大量的空闲时间，但除此之外，各种处理器布局的总体性能都是相似的。

图7中的并发布局不能跟CCSM3实时作比较，因为并发度是完全不一样的。正如我们在设计部分所讨论的，CCSM3耦合的实现采用两个运行阶段，这允许分量模式间作业有更多的重叠，因此，跟CCSM3中可行的情况相比，CCSM4的每个模式分量只有一个运行阶段，这样极大减少可能的并发度而且增加了CCSM4纯并发布局下的运行时间。然而CCSM4处理器布局灵活性的增加使其在几乎所有的配置下，跟CCSM3完全并行的实现相比，都有更好的性能。

总的来说，图7说明可以供CCSM4用户使用的一些可能处理器布局。对于一个给定的实例，最优的布局以及每个模式分量的处理器数量取决于每个分量模式的可扩展性能和分辨率。并发运行模式的有效程度，是在由并发造成的空闲时间以及分量模式随处理器数目增长的次线性增长之间的相互平衡。

**总结**

自从CCSM3发布以来，CCSM的耦合器实现已经进行了很大的修改。CCSM4现在包含了一个顶层驱动器，用来通过匹配的借口调用分量模式的初始化，运行以及结束方法。它还协调了时间的顺序以及分量MPI进程组的建立以便在处理器布局中提供更多的灵活性。很多改动都是用于提高耦合器内存和性能的增长性的，而且CCSM4耦合的额外开销已经比CCSM3减少了。绝大部分努力通过使用MCT库实现，而在CCSM4的分量模式也支持跟ESMF兼容的接口。

和CCSM3相比，CCSM4对于更多的处理器数目有更好的扩张性能，而且CCSM4能够比CCSM3处理更高的分辨率配置。CPL6的论文（Craig et al., 2005）给出CPL6的标度曲线，它采用的是在32个处理器上运行大致1度的海洋/海冰分辨率和T42大气/陆面分辨率的耦合实例。本文还给出了使用达10000个处理器的十分之一度全球分辨率的结果。特别的，CCSM4已经完成了在全球四分之一度分辨率的大气/陆面模式以及全球十分之一度的海洋/海冰模式的耦合运行。这是CCSM3/CPL6的耦合实现无法达到的。



图7：CCSM4中的分量模式在四种不同处理器设置下的时间安排示意图。这是对‘f19\_g16’分辨率的完全预测模拟所设的。在每种设置下，处理器的总数都是128。 灰色的格子代表机器闲置时间，数字代表每个子模式积分一天需要的机器秒数。总秒数是：(a)完全同步设置为33.5秒; (b)完全有序设置为21.8秒; (c) 有序设置但海洋同步设置时为21.8秒; (d) 有序与同步混合设置时为19.1秒。

我们还给出了单独耦合核心的性能。正如所预料的那样，容易并行的操作在高处理器数目下有很好的可扩展性。特别的，大气/海洋通量核在所有机器上都取得了几乎线性的可扩展性。强烈使用内存的合并核心也有很好的可扩展性，而通信强度大的重排和映射核心一般只在中等进程数量时有可扩展性。对于CCSM4的总体负载平衡和性能，选择耦合器处理数目和布局时有几个因素需要考虑。耦合器的开销与其他分量模式相比较小的，因此它经常会跟其他分量依次运行，特别是大气模式。这样，耦合器通常会根据其他分量模式的负载平衡以及优化被首先放在可用的处理器上。而耦合器一般在更多的处理器上也会运行得越快。然而基于jaguarpf通信核心的增长，也有可能随着处理器数目的增长耦合器会在某些机器上变慢，所以有个必要在长期运行前就评估好增加处理器数量对耦合器性能的改变。

在本研究中，关于不同硬件架构的相对性能我们可以得到一些通用的定论。以每个处理器为基础衡量，Bluefire在耦合器用时上一般都比Jaguarpf或者Intrepid要快。强烈使用内存的合并核心对于所有机器特别是在那些每个处理器都有很多格点的情况下，其用时都是类似的。而对于计算强度大的通量计算，Intrepid就比Bluefire和jaguarpf慢十倍。基于相对处理器的速度，这也是在意料之中的。所有机器上，通信强度大的核心的性能增长都会在高处理器数量下降，其中jaguarpf在更多处理数量下性能是最差的。目前我们还不明确这基于硬件的jaguarpf通信增长行为是跟本地系统设置有关，还是跟机器上给定时间内的各种负载有关。

最后，CCSM4中随着处理器布局而改变的灵活性也已经得到说明。与CCSM3相比，CCSM4模式支持串行、并发以及串并发混合的能力，对总体的负载平衡和性能优化有了更多的灵活性。

**致谢**

本工作是由美国能源部科学办公室（BER）以合同DE-FC02-97ER62402、DE-FC02-07ER64340以及DE-AC02-06CH11357支持的。额外的支持有国家科学基金AGS-0856145提供。作者感谢国家大气科学中心，橡树岭国家实验室，阿贡国家实验室提供对计算资源的访问和帮助。

1. 原文：Anthony P. Craig, Mariana Vertenstein, and Robert Jacob ,2011: A new flexible coupler for earth system modeling developed for CCSM4 and CESM1. *International Journal of High Performance Computing Applications February 2012 26: 31-42, first published on November 20, 2011 doi:10.1177/1094342011428141 。*

   谢歆译，何卷雄校。 [↑](#footnote-ref-1)