高性能计算多层次不连续非线性可扩展现象研究

张云泉[1] 袁 良[1] 陈一峯[2] 冯晓兵[1] 张 贺[3]

[1](中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)

[2](北京大学 北京 100871)

[3](中国科学院大气物理研究所 北京 100029)

摘 要 高性能计算是计算科学的具体实践,极大地促进了各领域的科学进展,也对国家的经济建设起到了无法替代的基础性作用.从几十年发展的时间尺度和十万至百万核量级并行规模尺度研究大规模并行软件的研制发展历史来看,发现大规模并行应用软件的开发中物理模型、并行算法、并行软件实现以及底层硬件多个层次中存在的可扩展性的两种有趣现象,即不连续性和非线性现象.本文总结分析这一普遍存在现象,系统梳理计算机软硬件发展,特别是高性能计算发展中的可扩展问题,为未来并行计算领域发展提供方法论层面的借鉴和指导.

关键词 高性能计算;超级计算;可扩展性; 多层次;不连续;非线性

中图法分类号 TP391.9 **DOI 号** 10.11897/SP.J.1016.2020.00973

Multi-Level Discontinuous and Nonlinear Scalability Phenomenon in High Performance Computing

ZHANG Yun-Quan[1] YUAN Liang[1] CHEN Yi-Feng[2] FENG Xiao-Bing[1] ZHANG He[3]

[1](*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

[2](*Peking University, Beijing* 100871)

[3](*Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing* 100029)

Abstract High Performance Computing (HPC) is the practice of computational science. It promotes the scienctific progress of many fields and serves as an irreplaceable foundation for national economic development. From the perspective of the large-scale parallel software development with a timescale of decades and the scalabilities of thousands and millions cores, we observe two interesting phenomena in multi-level of HPC including physical models, parallel algorithms, parallel software implementations and underlying hardware architectures: discountinuous and nonlinear. This paper presents a detailed study of this phenomenon especially the scalability development in HPC. This paper also provides a methodology guidance for future parallel computing progress.

Keywords High Performance Computing (HPC); supercomputing; scalability; multi-level; discontinuous; nonlinear

收稿日期:2019-01-17;在线出版日期:2019-11-01.本课题得到国家重点研发计划(2016YFB0200803)和国家自然科学基金(61432018,61521092,61272136,61402441,61502450)资助.**张云泉**,博士,研究员,博士生导师,主要研究领域为高性能计算、大数据处理.E-mail:zyq@ict.ac.cn.**袁 良**(通信作者),博士,助理研究员,主要研究领域为并行计算模型、并行算法.E-mail:yuanliang@ict.ac.cn.**陈一峯**,男,1973 年生,博士,研究员,博士生导师,主要研究领域为高性能计算、大数据处理.**冯晓兵**,男,1969 年生,博士,研究员,博士生导师,主要研究领域为高级编译技术与工具.**张 贺**,男,1981 年生,博士,副研究员,硕士生导师,主要研究领域为地球系统模式、气候模拟.

1 引 言

计算科学已成为继理论研究和实验分析后的第三种科学研究方法.在包括物理、化学、生物等几乎各个科学领域,对理论上难以精确求解的复杂问题,例如湍流、流体力学等,以及由于各种限制因素难以开展的实验测试,例如气候、天体物理等,大规模科学与工程计算都成为现阶段唯一的科学研究手段,极大地促进了各领域的科学进展,也对国家的经济建设起到了无法替代的基础性作用.大规模科学与工程计算领域的关键问题是性能,这一核心需求是计算机并行软硬件平台发展的主要动力之一.

从高性能计算机几十年的发展来看,底层硬件特别是计算单元的多样性日益丰富,从最早的向量处理机,到多核众核的出现,体系结构的进化是保持摩尔定律的一个重要方式.从全球超级计算机TOP500排行榜[1]的发展也可以清晰的看出这种趋势.同样,对并行软件而言,其发展路线也随着底层硬件的进化而完善.国际上代表超级计算应用最高水平并获得Gordon Bell奖[2-12]的应用软件,其并行度目前已经普遍达到几十万核的并行可扩展,部分应用软件甚至接近了数百万核的并行可扩展,持续运行性能超过了10 Petaflops.研究百万核量级的并行可扩展算法和软件,当前仍然是国际上,特别是国内的前沿热点挑战性问题.

可扩展性问题是高性能计算的核心问题,即如何在增加资源的情况下保证性能.可扩展性问题在图灵奖获得者Jim Gray所提出的计算机科学的未来十二个大挑战中排在第一位,是一个重要而长期的研究目标[13].近年来,随着多核技术的兴起以及各个应用领域内对更高计算能力的需求,超级计算机的可扩展性已经成为一个以待解决的研究问题[14-26].

本文对高性能计算的软硬件发展进行分析,总结了一种广泛存在于可扩展问题中称之为多层次不连续非线性的现象(Multi-level Discontinuous and Nonlinear Scalability,MDNS).我们从软件硬件两个角度,以及物理模型、计算模型、计算单元、网络通信等多个层次对高性能计算主要组成部分的发展进行了分析,描述了其中存在的不连续非线性现象.我们认为多层次不连续非线性可扩展现象是高性能计算发展的一种普遍现象.系统梳理这一现象对未来的研究工作具有重要意义.

本文认为应将软件硬件的协同设计方法进一步向上层应用扩展,提出了应用算法的协同设计概念,在未来解决多层次不连续非线性可扩展问题时应同时应用位于不同层次的两种协同设计方法,同时提高应用层次和硬件层次的可扩展性.我们以国产全球气候模拟并行应用程序的可扩展性研究为例,介绍了两层协同设计方法的有效性.

本文第2节详细介绍多层次不连续非线性可扩展现象;第3节介绍多层协同设计方法;第4节进行总结.

2 多层次不连续非线性可扩展现象

2.1 可扩展性

可扩展性并没有公认的标准定义,本文给出一个定性的概念:可扩展性是指并行软件和并行计算机体系结构在增大规模情况下的性能利用能力.并行算法和并行结构一软一硬是可扩展性的两个方面,也是对立统一的.并行软件或者并行算法的可扩展性高意味着在增加底层硬件时能够获得更好的性能,并行体系结构的可扩展性高意味着能够对并行软件和并行算法进行支撑,支持处理更大的问题规模.问题规模的扩大源自实际的需求,也依赖底层硬件规模发展的支持,反之,系统规模的发展也是源自上层应用的需求.

高性能计算中的可扩展性存在两种主要类型,即强可扩展性和弱可扩展性.前者指固定软件层面问题规模情况下,单纯增加硬件资源的性能加速比;后者指问题规模随着硬件资源增加而扩大时的性能加速比.事实上,两种扩展性的分析分别针对硬件和软件层面规模或者资源的增加,我们将在后文中详细分析.

2.2 多层次不连续非线性可扩展现象

如果从几十年发展的时间尺度和十万甚至百万核量级扩展的并行规模尺度来看大规模并行软件的研制发展历史,我们会发现大规模并行应用软件的开发中存在的物理模型、并行算法、并行软件实现三个层次可扩展性的两种有趣现象,即不连续性和非线性现象.

随着并行计算规模不断增加,在物理模型、并行算法设计和并行软件性能优化等多个层次中都出现了不同性质的不可连续扩展,即必须更换新物理模型,新并行算法设计或并行软件实现方法等现象,我们将这种现象称为可扩展性的不连续现象.其次,即

使在某种物理模型、并行算法设计和并行软件性能优化的组合可扩展的并行计算规模范围内，其计算性能并不能随着计算资源的增加而线性提高，我们将此现象称为可扩展性的非线性现象. 我们将以上两个现象统称为多层次不连续非线性可扩展现象. 非线性描述的是一种量变，在一定范围内某种模型、方法是适用的，但是随着规模的增加，其性能增加的幅度在降低. 不连续描述的是一种质变，量变达到某个阈值，不得不借助于新的具有创新性的模型或方法.

此外，在计算机硬件发展趋势中，也存在着非线性和不连续可扩展现象. 例如计算单元的发展经历的单核性能的提高、多核并行性的提升以及众核体系架构的出现，这是典型的不连续现象. 从并行计算机的发展历史来看，不同量级的计算核心组成的超算系统为上层的并行软件非线性现象的底层基础.

上述不连续非线性现象普遍存在于计算机软硬件系统中，但并非仅限于并行可扩展视角下的高性能计算领域. 例如，按照著名的摩尔定律[27]发展的处理器硬件会达到物理极限，这包括晶体管大小和漏电的限制，因此导致了多核众核等体系架构的蓬勃发展. 此外，不仅在计算机学科中，其他理论和工程技术学科中，也大量存在不连续现象和非线性现象. 例如，不同里程的导弹必须重新设计才能满足新的里程需求，又如，物理学中牛顿力学和量子力学分别适用于刻画不同的空间尺度的系统，再如，数值方法中格式的发展也受不同软件性能瓶颈的影响而变化.

本文的目的在于总结分析这一普遍存在现象，系统梳理计算机软硬件发展，特别是高性能计算发展中的可扩展问题，为未来并行计算领域发展提供方法论层面的借鉴和指导.

图 1 给出了本文提出的多层次不连续非线性可扩展现象概要. 如前所述，问题规模的可扩展性对应于弱可扩展，主要是在与底层硬件松耦合的物理模型和数值计算方法等上层问题层次进行研究，例如典型的非线性包括不同粒子数目或其他问题规模从 G 级到 T 级的发展，典型的不连续包括不同规模下采用的建模方法不同，例如稀疏问题、稠密问题、降维方法等. 系统规模可扩展性对应于强可扩展，主要是在和底层硬件紧耦合的并行算法和性能优化等方面开展研究，典型的非线性为不同量级处理器核心，典型的不连续包括不同体系架构的计算单元.

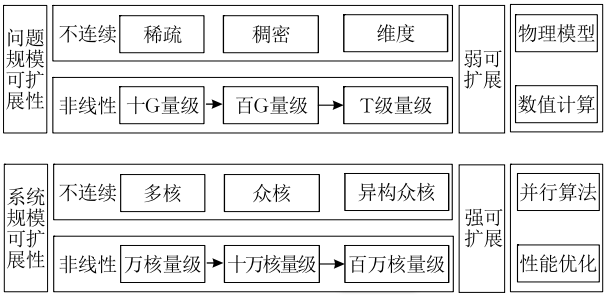

图 1　多层次不连续非线性可扩展现象

2.3　硬件层面的多层次不连续非线性

2.3.1　计算单元

传统上，由于摩尔定律和登纳德扩展定律(Dennard Scaling)[28]的有效性，每 18 个月，晶体管尺寸减小，在芯片面积相同情况下晶体管数目加倍，同时由于单个晶体管所需电压相应降低，总功耗保持不变. 晶体管尺寸减小同样带来了主频提升，并在更多可用晶体管的共同作用下，处理器性能每 18 个月加倍. 这就是所谓的给软件设计者的“免费午餐”，无需采用任何优化技术，只需等待一定时间，程序性能自然会提高.

但依赖单 CPU 主频提升性能的时代已经结束[29]，这是硬件上的不连续现象之一. 从 90 nm 开始，由于漏电流的问题，登纳德扩展定律已逐渐失效，即若想随着晶体管尺寸缩小继续提升主频，每个晶体管的电压不能相应降低，则总功耗会随着晶体管数目指数增长，这在当前功耗预算和散热能力的要求下都是不可接受的. 因此，借用在文献中经常出现的图 2 所示，处理器主频已经有十多年没有明显提升了. 若想利用芯片上越来越多的晶体管提升程序性能，只能另辟蹊径——采用并行的架构，这给硬软件设计都带来了划时代的变革，从此进入并行时代.

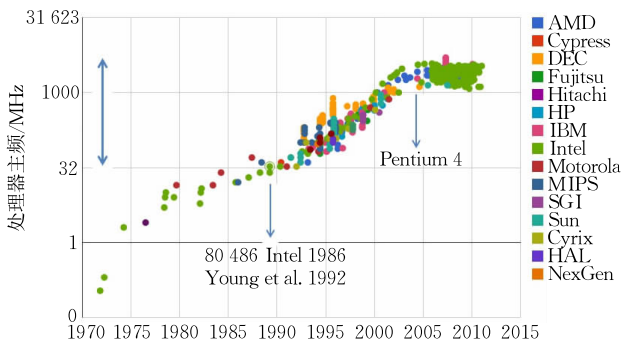

图 2　处理器主频历史变化图

另一方面，存储墙及功耗墙等问题也成为制约单核处理器性能提升的瓶颈. 这些因素导致芯片生

产商转而通过增加处理器核数使得处理器晶体管数量继续沿摩尔定律增加,多核/众核架构成为计算机体系结构发展的重要趋势.其中,以 GPU[30-33] 为代表的众核架构以其更强的计算能力和更高的访存带宽受到了越来越多应用开发人员的青睐.自 2007 年 NVIDIA 首次提出 GPGPU 概念以来,GPU 在通用计算领域得到了日益广泛地应用.主流 GPU 厂商也根据实际计算需求,对 GPU 架构进行了不断改进和完善[34].

与多核处理器相比,异构众核处理器提供了更高的计算密度,越来越多的应用程序被移植到众核处理器上,并取得了良好的加速比,使用众核处理器实现对应用程序的加速已成为提高程序性能的主要模式.到目前为止,NVIDIA 的 GPU 架构主要有 G80、G200、Fermi、Kepler、Maxwell 以及 Pascal.AMD 的 GPU 架构主要有 Cypress、Cayman 和 GCN 架构.这些架构在架构组织、性能、带宽和功耗方面的发展也呈现出非线性、不连续的特点.例如,一条指令从需要 4 个、2 个到 1 个时钟周期的性能提升,以及共享内存 16 个到 32 个 bank 的增加.其他包括完善的 cache 机制、Hyper-Q 和动态并行的支持等.

同样,在超级计算机的发展过程中也能清晰看到底层硬件的变化.2008 年由 LANL 和 IBM 共同研制的首台 P 级超级计算机系统 Roadrunner[35] 即采用异构加速架构,整个系统含有 3060 个计算结点,总共拥有 12240 个 IBM PowerX Cell 8i 协处理器和 12240 个 AMD Opteron 多核处理器,其 Linpack 实测性能首次突破了 1 Pflops.在 2018 年 6 月发布的世界 TOP500 高性能计算机排行榜中,排名前 10 的超算系统中有 9 台都采用异构众核体系架构.

美国、欧盟、日本和我国等超级计算大国都已经积极部署 E 级超级计算系统的研制,E 级超级计算系统将于 2020 年左右出现,系统将包含千万甚至亿量级的处理部件,其硬件平台的基础计算部件、系统规模和互联结构以及软件系统的计算模式和通信模式,都将远远超过现阶段 P 级系统的水平,在存储、计算、通信中都会呈现特别复杂的层次性和异构性.总之,处理器体系结构的发展存在明显的非线性和不连续性.

2.3.2 存储单元

计算机数据存储由最初的 SIMM[36] 一直发展到现在的主流内存架构 DDR,发展也遵循指数级递增,不过由于加速度相对计算单元的发展较慢,因此出现了内存墙等问题,特别是在多核/众核架构的情况下,内存带宽和延迟更成为程序的主要瓶颈所在.

同样,量变达到一定程度就会发生质变.目前由于物理方面的限制,继续沿原有路线方式发展已无法解决内存墙问题,各种新型的复杂的架构相继被提出.例如,最近几年出现的使用非易失性内存(Non-Volatile Memory,NVM)[37] 技术及其所带来的体系结构和编程思路上的转变,统称为内存计算方式.这一方式极大地提高了海量数据处理的性能.

3D XPoint[38] 是由 Intel 和 Miron 共同研究提出的一种三维立体存储架构,与已有内存设备基本原件不同,3D XPoint 采用新型可变电阻式记忆体,其价格将是 DRAM 的一半,并且支持位元可寻址,是内存发展历史上不连续现象的体现.

2.3.3 指令系统

指令系统的设计有两种截然不同的理念,即 RISC(Reduced Instruction Set Computers)和 CISC(Complex Instruction Set Computers),前者包括 Intel x86、PDP-11 和 VAX 等,后者包括 IBM Power PC、Sun Sparc、MIPS 和 Alpha 等.

最早的处理器仅有一个寄存器,因此需要在指令中配置丰富的内存寻址模式.随着硬件集成电路的发展,可利用片上资源成指数级增加,出现了特定寄存器、通用寄存器、内存-寄存器结构以及寄存器-寄存器结构,指令寻址方式变得十分丰富,指令集 ISA 也变得异常臃肿,例如 x86 指令集规模上千,长度从 1 到 15 字节,是与众寻址模式,这使得体系结构以及上层编译器的设计和优化变得极为复杂和困难.

19 世纪 80 年代由图灵奖得主 John Cocke、David Patterson 以及 John Hennessy 等科学家提出并设计实现了一种新型思路,即缩减指令集规模,固定指令长度,简化内存寻址模式,例如 MIPS 仅包含 200 余种指令,长度固定,为一种寄存器-寄存器体系结构,即所有数据需要加载到寄存器中,仅提供一种内存寻址方式,这极大地简化了硬件实现和编译器设计优化的难度,并且有效地降低了处理器能耗.

需要指出,这两种设计思路实际上在当前的处理器中也进行了某种程度的耦合,借鉴吸收了两种指令集的优势,例如 ARMv8 存在多种指令类型,Intel 处理器也参考利用了很多 RICS 技术.但是,两种指令集设计思路在指令集的关键特性,例如规模、寻址方式、指令长度等方面,都存在一定的断层,这

是指令集一种非线性发展的体现.

2.3.4　I/O

互联网络、高性能计算机的网络发展从其结构而言存在 Crossbar 和 Fat-tree 等结构，从其实现技术看存在 2000 年前后蓬勃发展的 Myrinet 和 SP Swith，以及当前主流的以太网和 Infiniband 技术，这些技术的发展均呈现出一定的不连续性. 例如，Crossbar 这种矩阵型的互联结构限制了其应用的规模，而 Fat-tree 结构在树的高层次结点间互联配置了更高的带宽，适用于更多节点数，已被广泛应用在高性能计算机系统中. 再如，2000 年前后研究提出的 InfiniBand 架构是一种支持多并发链接的转换线缆技术，相比传统以太网具有更优的带宽和更低的延迟，是高性能计算特别是 TOP500 排名前十系统上的标准网络配置.

I/O 存储系统的多样性要更加丰富，从最早到的磁带、软盘、光盘、硬盘到当前的 SSD 存储，其硬件介质的特性，例如容量和速度等，均在发展中遇到了技术天花板，因此不得不寻找新型材料或物理技术，这是一种典型的不连续性发展现象. 从计算机角度，存在的多种 I/O 存储系统的数据传输技术也是不连续性发展的实例，例如同步 I/O、异步 I/O 和 DMA(Direct Memory Access)技术，相比前者，后者允许处理器在 I/O 数据传输的同时执行更多操作，提高了处理器利用率.

2.4　软件层面的多层次不连续非线性

2.4.1　物理模型和数值方法

本节选取全球气候模拟和材料模拟为例介绍大规模并行应用在物理模型和数值方法层面发展的可扩展性问题.

大气环流模式(AGCM)[39]是研究气候变化及其成因的重要工具之一，是地球系统模式的重要组成部分，它的研制和发展是当前大气科学的研究热点之一. 以中国科学院大气物理研究所自主研发的大气环流模式 IAP AGCM，近 30 年的发展来看，可以清晰地看到随着底层硬件性能的不断提高，以及模拟实际问题的精度的细化，软件的物理数学模型经历了复杂的系列变化.

当前大气环流模式的发展主要集中在两个方向，一方面是物理过程越来越精细，一方面是分辨率越来越高，因为更高分辨率的气候系统模式能直接分辨更精细时空尺度的物理过程，从而有更高的模式性能. 如图 3 所示，精度和分辨率的提高可以对全球气候进行更为精细地模拟和分析，而精度和分辨率的提高导致了问题规模的扩大.

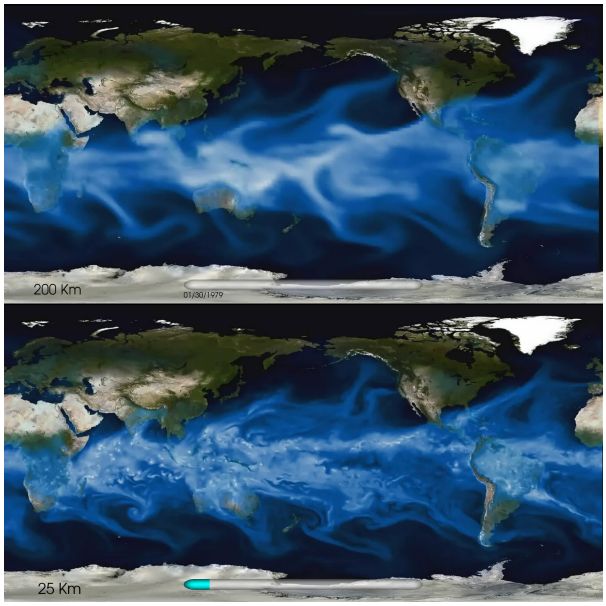

图 3　全球气候模拟不同分辨率模拟效果的对比图

依据计算内容和计算特点的不同，大气环流模式可以分为动力框架和物理过程两大部分. 动力框架主要是用来求解关于时间的偏微分方程组，离散方案的设计是关键，要同时保证计算的稳定性、守恒性、精确性和高效性. 动力框架的计算在空间上是三维的，每一点的计算都可能与周围的很多点有关系. 物理过程主要用来计算控制方程中的源汇项，以及一些诊断量(如降水、云量、辐射通量等). 物理过程的计算是在单柱(single column)上进行的，也就是说是一维的，仅有垂直方向的数据交换.

单核串行计算时动力框架与物理过程的计算时间相当，随着进程数的增加，动力框架所占比重逐渐增加. 以大气所自主研发的大气环流模式 IAP AGCM4.0(0.5×0.5)为例，在使用 1024 个进程时，动力框架部分的计算时间占了总计算时间的 85%，其中纯计算时间约 47%，通讯时间约 38%. 而国外的一些研究表明，当模式分辨率提高到 3 km 时，物理过程的计算时间仅占总计算时间的 1%，完全可以忽略不计[40]. 造成这种情况的主要原因是物理过程的计算是在一个个气柱上进行的，在水平方向彼此没有联系，因此其并行可扩展性远远高于三维间关系紧密的动力框架. 由此可见，增加大气模式并行可扩展性的关键是增加动力框架部分的可扩展性.

大气模式的非线性发展主要表现在硬件规模增大时性能无法大幅提高. 对于目前较成熟的中高分辨率的模式(50～100 km)而言，主流的并行规模在千核左右，核数再增加时，并行效率便迅速下降. 而

对于 NCAR 更高分辨率的模式(25 km),在使用格点的动力框架时,虽然使用 6 万核时仍保持一定的加速比,但并行效率已经非常低了[41]. 如图 4 所示,大气所的模式 FAMIL,虽然完成了 1 万核的测试试验,但在核数高于 6000 核后计算速度几乎不再提高. 初步分析表明,格点模式的并行可扩展性的瓶颈主要是通信. 以 IAP AGCM 为例,该模式动力框架采用的是精度较高的中央差方案,每个点的计算要与周围 12 个点发生联系,而在进行二维平滑时,则需要用到周围 24 个点的数据. 这种紧密的关系给数据剖分带来了困难,同时数据间的通信量也变得很庞大. 此外,高纬沿纬圈的 FFT 滤波也是并行可扩展性的一个瓶颈. 综上所述,要提高 IAP AGCM 的并行可扩展性,关键是改进模式的偏微分方程 Stencil 并行算法,FFT 并行算法,以及平滑模块的并行算法设计和优化实现.

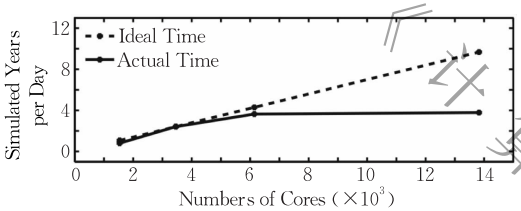

图 4 大气环流模式 FAMIL 在天河 1A 超级计算机上的计算速度测试[42]

大气模式的不连续性主要体现在模式物理参数化方案和数值计算方法的不连续性,即当模式分辨率提高到一定程度时,原有的方法不再适用,目前尚未看到在各种空间尺度(即分辨率)上均适用的方案或方法. 在物理参数化方案方面,当模式的分辨率较粗时(>100 km),模式网格无法分辨特征尺度在 1~10 km的积云,因此必须采用积云对流参数化的方法,用网格点上的大尺度变量近似地表示积云对流过程对大尺度环境场的影响. 而当模式分辨率提高到 5 km 甚至更高时,模式的网格点已经可以显式地分辨出积云,因此,粗分辨率模式中应用的积云对流参数化方案便不再需要了. 此外,物理参数化方案的许多参数也是和分辨率密切相关的,这些参数和分辨率的关系也是不连续的.

在数值计算方法方面,在模式发展的最初阶段,分辨率只有 500 km 左右,模式代码还是串行的,此时主流的离散方法是有限差分法,该方法设计简便且易于保持守恒. 当分辨率提高到 300 km 时,谱方法逐渐成为主流的离散方法,这是因为谱方法很好地解决了极点问题,可以使用较大的时间步长从而提高计算速度. 但谱方法也有其固有的缺陷,即并行可扩展性较低. 在计算规模较小时,这一缺陷尚无显著影响,但随着模式分辨率的提高和计算规模的增大,谱方法的这一不足愈发凸显,而谱元方法的优势则开始显现. 图 5 给出了 25 km 分辨率下谱框架(谱方法),格点框架(有限差分)和谱元框架的计算速度. 可以看出当计算进程数在千核以下时,谱框架计算速度优势明显,在 4000 核时,三种方法的计算速度相当,而当 10 000 核时,谱框架的计算速度已显著落后于格点框架和谱元框架.

由此可见,随着计算规模的扩大,物理模型的数学表达及离散化方案都会发生明显的变化. 同时,受到通讯、I/O 等瓶颈的制约,其计算速度并不能随着计算资源的增加而线性提高.

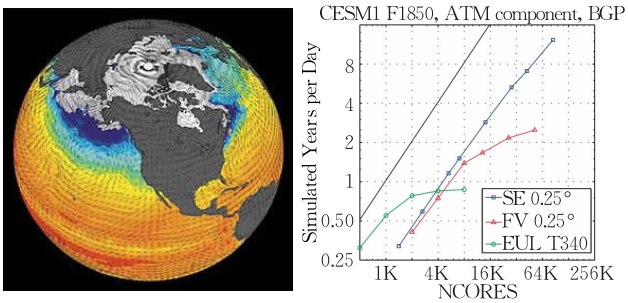

图 5 大气环流模式 CAM5 三种不同动力框架在 Blue Gene/P 超级计算机上的计算速度[43]

再以材料模拟为例进行介绍. 材料科学是基础科学应用的重要领域,高新技术材料的研究和发展水平是衡量一个国家科技实力的重要标志. 传统上材料科学的研究是从理论角度与物理实验相结合出发的,但是随着材料技术发展,这一研究手段的瓶颈逐步显现. 一方面是新型的复杂材料理论模型给数值求解带来的挑战,理论方法难以进行求解分析;另一方面,从实验角度开展新材料的研发周期太长,增加了大量的时间和人力成本. 此外,实验手段还受制于设备的分辨率等指标. 因此,以数值模拟方法为核心的材料模拟研究逐渐兴起,逐步成为材料模拟的重要研究手段.

我们以核材料辐照损伤微观模拟的两种方法:即分子动力学模拟算法[44](Molecular Dynamics,MD)和动力学蒙特卡洛算法[45](Kinetic Monte Carlo,KMC)为例开展介绍. 分子动力学模拟 MD 主要通过使用牛顿动力学方程求解方法实现材料中粒子运动轨迹的模拟,该方法的核心是一个确定性物理方程,该方程准确地记录和描述了分子和原子微观演化的轨迹,这种方法的计算量较大,计算复杂度较高,因此利用这种方法难以长时间地模拟材料

分子体系的演化.

动力学蒙特卡洛算法 KMC 通过使用随机过程理论实现对材料中微观尺度整个体系演化情况的模拟，与 MD 不同，KMC 将关注点从“粒子”扩大到“体系”，同时进一步将“原子运动轨迹”粗化为“体系组态跃迁”，这种方法复杂度较低，计算量适中，可以模拟大规模、长时间的核材料微观结构演化过程，但是这种随机过程的方法难以准确描绘每个原子的运动轨迹.

通过两个大规模科学和工程计算问题的发展可见，不同方法具有独特的优缺点，在应用环境变化的情况下，优势劣势往往能互相转变，需要考虑不同的方法，改进设计扬长避短.

2.4.2　计算模型

强可扩展性的理论分析模型是经典的 Amdahl 定律[15]，假定某一算法的可并行化比例为 e，则在 p 个处理器上的加速比为 $S=1/(1-e+e/p)$，当处理器数目为无穷时的最大加速比为 $1/(1-e)$. 因此，假设一个算法的可并行部分 $e=0.99$，则最大加速比为 100. 强可扩展性的分析结果指出并行应用的加速比受程序串行比例大小限制，一度使得并行计算前景不明，并行计算研究也陷入低潮. 1988 年，Gustafson[16] 根据在 1024 个处理器上获得千倍加速比的经验，提出了 Gustafson 定律，指出可扩展性应扩大并行处理器上处理的问题规模，即弱可扩展性概念. 之后，更多的学者推广了可扩展性的模型. 比如 Sun-Ni[17-18] 提出了内存受限的可扩展性模型，实际是对 Gustafson 定律的推广. 虽然 Gustafson 模型的假定仅与 Amdahl 在问题规模上不同，但是得到的最终结论却大相径庭，由此可见模型的发展对实际问题的指导也具有一定的不连续性.

这种不连续性也体现在 Amdahl 在多核架构上的推广. 体系结构权威 Mark Hill 教授将 Amdahl 定律推广应用到多核处理器的性能模型上[19]，得出异构多核比同构多核性能好，不能一味追求核的数量，数量少而单核性能高可能是全局最优的结论. 姚二林等人[20-21] 进一步发展了 Mark Hill 教授的结论，指出虽然异构多核比同构多核更好，但是异构多核相对于同构多核的加速比仍然受限于问题的串行部分比例；只要处理器内总的资源在一直增加，则对于任何实际的单核性能模型，全局最优点总是在核的数量和单核的性能间取折中[22].

并行计算模型可分为面向共享存储、面向分布式存储和面向存储层次三类，这三类模型代表着高性能计算不同方面量化模型发展的不连续性. 而每个类型内模型的发展又是非线性的. 我们以面向分布式内存系统，即处理器间通过网络消息传递进行通信为例介绍，此类模型更真实地反应了当前并行计算机系统的通信瓶颈. 经典模型包括 BSP 模型[46-47] 和 LogP 模型[48]，然而这些模型也十分简洁，并且包含一些不合理假设，忽略了一些重要的实际网络通信中的硬件参数. 例如 LogP 模型中没有考虑长消息带宽，为此 LogGP 模型[49] 增加了一个参数 G 用以描述长消息带宽. LoPC 模型[50] 和 LoGPC 模型[51] 考虑了网络通信竞争对系统通信性能的影响，LogGPS 模型[52] 增加了参数 S 表示不同通信协议的消息大小阈值，对不同的协议建立不同的通信开销. LogGPO 模型[53] 考虑了计算和通信的重叠对程序性能的影响. Memory LogP[54] 考虑了消息的数据大小和分布，区分了传递连续数据和非连续数据的开销. 这些模型虽然都基于经典的 LogP 模型，但是其发展体现了对不同实际特性的考虑，可看做一种非线性发展.

2.4.3　并行算法

在中间的并行算法设计层次，计算机专家通常抽取大规模应用中的关键算法进行研究，并分类为几种常见模式(Dwarfs)[29]，相关算法在可扩展性方面也存在不连续非线性可扩展的现象. 同一算法的各种不同并行编程模型和优化方法的软件实现适用的计算平台也不同. 下面总结几种典型算法的发展，可充分体现同样算法的软件实现和优化方法必须随着底层体系结构的演变而演变，体现了可扩展的不连续和非线性现象.

以线性代数的稠密矩阵计算的数学库实现的发展为例. 20 世纪 70 年代大量使用的向量处理器，需要将矩阵存储和操作划分为向量模式，并且大多数稠密矩阵算法的向量化设计也较为简单，基于这一特点设计实现了 LINPACK[55] 和 EISPACK[56]，分别包含支持几种矩阵类型的线性方程求解和最小二乘求解，以及特征值计算的 Fortran 程序，其大部分函数都通过调用一级 BLAS[57] 实现，可提高程序性能移植性.

20 世纪 80 年代的硬件更注重利用多层次高速缓存来缓解硬件计算访存性能的差距，因此调用数据局部性较差的一级 BLAS 实现算法的性能较低. 设计实现的 LAPACK[58] 利用三级 BLAS 实现，这种利用分块的矩阵算法能有效提高数据时空局部性.

20 世纪 90 年代适用于分布式存储系统的 ScaLAPACK[59]利用并行 BLAS 和负责线性代数通信的基本子程序集 BLACS,PBLAS 利用二维循环分块布局,实现了数据局部性和负载平衡的统一.此外还研究设计了两种类似的库 PLAPACK[60]以及其 OOC 版本 POOCLAPACK[61].这些库充分降低了通信开销,提高了并行效率.

2000 年以后蓬勃发展的多核众核系统上,可以借鉴利用底层多线程 BLAS 库来继续支持上层线性代数算法,然而这种方式会导致过多的细粒度同步,严重影响了程序性能,因此需要全新的算法和数据布局.适用于多核的 PLASMA[62]和众核上的 MAGMA[63]利用基于有向无环图的乱序异步执行技术、细粒度任务调度和 BDL(Block Data Layout)数据存储方式进行优化,并且这种思路可以推广到分布式异构环境下,基于运行时系统 PaRSEC 的 DPLASMA 实现了细粒度的任务图并行.

近几年来,在大规模并行计算机系统上,特别是随着面向 E 级计算能力的硬件发展,算法的通信性能成为程序的主要瓶颈之一,在这方面,提出了一类半整数维度划分策略[64]和一类通信避免策略[65-70],两种方法都能应用到大量矩阵类算法中.利用半整数维度划分方法,可以在并行性、局部性、通信量等方面进行平衡,例如 3D 划分的矩阵乘法需要较大的冗余输入数据拷贝,但是降低了通信量同时也提高了并行性,而 2D 算法例如 Cannon 算法的特征正好相反.在这两种算法之间存在 2.5D 划分[71],通过配置冗余存储参数 c,即数据冗余数目,将带宽需求和延迟分别降低 $c^{1/2}$ 和 $c^{3/2}$,可以在这些性能指标间平衡调整,在不同系统上通过自适应参数优化达到最优性能.

如上所述,沿着稠密矩阵数学库核心算法的发展来看,不同的并行算法适用于不同底层硬件,当硬件的体系架构,特别是关键计算单元的性能构件变化之后,就要以新的思路来设计并行算法,这是可扩展的不连续性现象的主要体现.

再以大规模 FFT 为例.针对集群体系结构特点、集群规模、计算规模,会采用不同的优化方法和算法实现.现有传统集群上的商业版 FFT 数学库主要有 FFTW 和 Intel MKL.FFTW 是使用最为广泛的 FFT 库,其基于 MPI 的 3 维 FFT 并没有考虑计算和通信的重叠优化问题.P3DFFT[72]及其他许多研究[73-80]通过对数据进行节点间高维分布以提高 3 维 FFT 的延展性,但没有考虑计算和通信的重叠优化问题.2DECOMP&FFT[81]通过优化多组输入数组间的计算通信重叠问题实现对批量数据 3 维 FFT 计算的性能优化.我们在异构众核计算系统上优化 FFT 算法,基于并行系统多个层次的不同通信性能,研究新的 FFT 算法和优化方法,图 6 显示了 PKUFFT 在节点数为 2048 以上时,性能较 MKL 数学库更高,而 MKL 性能可扩展性峰值出现在 4096 节点数,大于该节点数后,性能会几乎保持不变,无法继续扩展.而陈一峯等人[82]提出的新 FFT 算法依然能几乎线性扩展到 8192 核,这展示了可扩展性不连续现象.

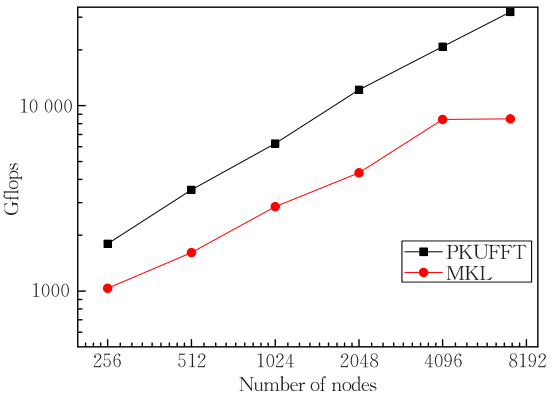

图 6 不同 FFT 算法可扩展性比较[82]

最后再以稀疏问题为例分析说明可扩展的非线性现象.稀疏问题的关键计算核心是稀疏矩阵向量乘法 SpMV[83].图 7 显示了稀疏矩阵向量乘的例子,在图中的三个实现对称矩阵 SpMV 的程序,分别在延迟隐藏、对角线计算顺序、目标向量传送方式上有所不同,从图中结果可以看到,底层硬件规模越大时,各种方法的可扩展性愈加呈现非线性,而且各种方法间出现交替上升趋势,具有不同的适用区间.因此,不同的算法对不同的底层硬件平台不具有性能可移植性,并且随着硬件规模的增加,性能不能线性增长.

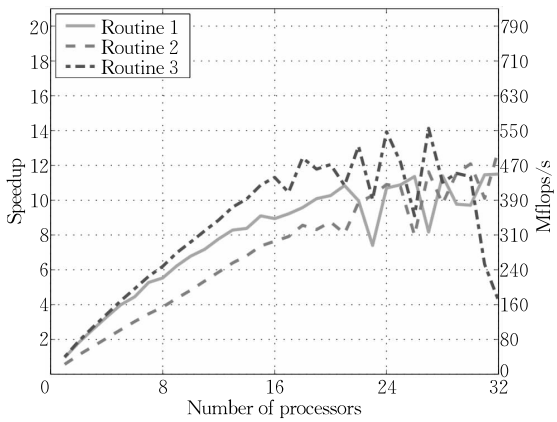

图 7 不同 SpMV 算法比较[83]

2.4.4　优化方法

优化方法与并行算法的主要区别在于其开展研究的环境不同，前者与底层硬件紧耦合，通常一种优秀的优化方法或者思想适用于不同的问题和算法，而后者与所求解的问题紧耦合.从此角度看，广泛使用的分块这一编译技术应视为一种优化方法.分块方法的变化也能体现对底层硬件存储层次发展的不断适应.本节还将以调度优化方法的发展为例介绍发展的不连续和非线性现象.

最早的分块方法应用于矩阵类型计算[84]，通过手工或编译器等方法自动对循环进行划分，使得内层循环处理某个循环迭代空间中的子块，以提高数据的时间和空间局部性.分块方法是最为常用和有效的性能优化方法，几乎所有算法都通过分块来提升实现性能，存在大量后续研究工作[85-88].分块方法根据高速缓存的大小及其他例如缓存行大小、相联度等特征进行参数调优，通过设置不同的分块大小、预取距离等参数进行细粒度调优.这种变化体现了分块方法在优化方法层面的不连续性现象.

随着硬件的发展，单一层次的高速缓存无法平衡缓存大小和存取速度之间的矛盾，因此当前多核和众核处理器均出现了多层次高速缓存，处理器核一般独占一至两层私有高速缓存，而多个片上内核共享最外层高速缓存.这种硬件的发展给分块方法的调优和设计带来了一定的挑战.1999 年 MIT 的科学家设计提出了缓存无关分块方法[89]，并将以前的分块方法称为缓存相关方法.缓存无关方法可以忽略高速缓存具体硬件特征，自动适应不同存储层次数量、缓存大小.在缓存无关方法中，理想的缓存模型被抽象为只包含两层存储架构：一个是一定大小的理想高速缓存，另一个是无限大系统内存.通过分而治之的方法，缓存无关方法可以适应实际高速缓存大小以及存储层次等硬件参数，从而总能获得最优访存复杂度.与缓存相关方法类似，针对大量已有问题和算法也设计提出了相应的缓存无关方法实现[90-93].以缓存相关方法和缓存无关方法的发展而言，这是典型的优化方法的非线性现象.

调度是重要的运行时优化方法之一，我们以调度方法的发展为例进行分析.任务调度模式主要分为主从模式和对等模式两种.主从模式是一种经典的调度方法，主控端生成任务并将其放入集中任务队列中，然后将任务发送给空闲的从属端.这种模式的优点是实现简单，通信开销小，且容易实现负载均衡；缺点是主控端维护集中任务队列容易形成性能瓶颈，尤其是当任务粒度特别小时，主控端生成任务速率小于从属端任务执行速率，从而导致从属端饥饿.因此主从模式不适用于大规模并行系统的调度.在对等模式下，各个线程处于平等地位.为了实现负载均衡，空闲线程从某个繁忙线程窃取任务.这种模式采用分布式任务队列，缓解了主从模式下管理集中任务队列而引发的竞争，因此在大规模并行时可以获得较好的可扩展性.对等模式的缺点是任务窃取通常会对数据局部性造成影响.主从模式到对等模式的发展是一种典型的随着硬件系统规模增大而导致的调度系统设计的不连续现象.

在对等模式的发展中，也存在着非线性发展现象.例如在对等模型调度中，工作优先和求助优先是两种常用的任务生成策略.工作优先是指工作线程立即执行生成的任务，而将剩余任务留给其他工作线程窃取[94-95].此生成策略的优势是数据局部性较好，但是由于函数多层嵌套容易导致线程栈空间溢出.此外，工作优先生成任务速度慢，当空闲线程较多时容易导致线程饥饿.求助优先是指工作线程继续执行剩余任务，而将新生成任务留给其他工作线程窃取.此任务生成策略可以快速生成任务以供其他线程窃取，适用于空闲线程较多的情况，但是这种策略任务生成开销大且会破坏数据的局部性.自适应任务生成策略会根据栈阈值自动切换到工作优先或求助优先生成策略，弥补单一生成策略的不足.

2.4.5　编程模型

编程模型是提供给程序员编写算法实现的软件平台，它与底层硬件和上层问题紧密相关，不同编程模型适用于不同平台和问题，这种适用性是不连续性的根本原因，如 MPI[96] 是适用于大规模分布式存储系统的并行通信库，OpenMP[97] 是共享存储以及多核处理器的常用并行方法，OpenCL[98] 是众核平台的统一编程标准.不同的语言表面上只支持特定并行架构，其本质核心在于硬件可扩展性的不连续性发展这一现象.

对于非线性而言，即使是同一语言，也会随着硬件平台的进化而不断演变，以 MPI 和 OpenMP 的不同版本演化为例，2003 年底 MPI-2[99] 发布，增加了进程拓扑映射、更好的单边通信接口，后续版本陆续增加共享内存支持、混合编程、甚至是 GPU 支持.MPI 标准 3.0 提供了通信重叠的更多机会，支持异步和近邻的集合通信，更好的单边通信接口，以改善应用的扩展性.OpenMP 起初主要是循环级并行，3.0 规范中增加任务并行，4.0 中增加处理器绑

定、任务组、众核加速器和 SIMD 等支持. 这都说明了底层实现优化层次中的方法和模型也只能在一定范围内适用.

针对同一硬件平台而设计的不同编程模型也可看为一种非线性发展,例如针对目前广泛使用的大规模异构众核平台,没有统一的并行编程解决方案. 目前的趋势有几个特点:(1)在大规模异构众核计算上,学术界普遍认为异步的任务图并行是隐藏通信和同步开销,获得应用扩展性的一个重要手段;(2)对于异构众核处理器所带来的结构复杂性,产业界还是采用 MPI 与 OpenCL 等异构众核平台编程接口的混合编程,学术界在统一并行编程方面的研究则比较发散,这种发散恰恰体现着一种非线性现象.

具体而言,在异构众核并行编程的研究方面:单个结点内的异构众核编程主要有 OpenCL、OpenAcc[100] 和 OpenMP 语言. OpenCL 是异构众核平台上并行编程的一个开放的行业标准,基于 C++语言. 它采用 kernel 函数机制描述"异构众核特征",用 SPMD 风格来表达并行性,通过命令队列(command queue)支持任务并行;允许程序员明确地指定数据在异构众核系统中的存储位置;提供组内同步和任务依赖的表达. OpenCL 是一个比较低级的编程接口,给用户带来很大的编程负担. 为了简化异构众核平台的编程,多家企业联合提出了 OpenACC 编程标准. 它是 C、C++、Fortran 的语言制导扩展,为程序员提供多组制导命令,进行任务划分、数据分布和通信以及相关的优化表达. OpenACC 需要编译器的支持,目前编译器发展还比较滞后. 新发布的 OpenMP4.0 已经新增了一组面向加速器设备的语言制导,可以描述一段代码中的数据和计算如何映射到一个加速设备上. MPI 的最新标准 3.0 提供了对多线程混合编程和共享内存的更多支持. 一些开源的 MPI 实现(比如 MVAPICH2)已经能利用 GPUDirect 的底层支持来优化 GPU 所带来的通信.

编程模型另一个层面的发展即根据不同问题设计领域编程语言. 例如 UIUC 的 PPL 实验室 20 世纪 90 年代就研发了 Charm++[101],支持计算资源的虚拟化和细粒度的任务图并行,可实现全系统的大规模细粒度任务图并行. Charm++属于一个比较低级的编程接口,不易使用,所以该课题组开发了一些面向领域的应用编程框架. Barnes-Hut(层次 N 体模拟)的 Charm++程序优化后通过 CPU 和 GPU 的有效协作最高可以扩展到 1024 个 GPU 的集群. 编程模型的这种类型的发展可以看为是依赖问题的一种非线性可扩展现象.

3 应对方法

前文从超算系统的发展和大规模应用的优化角度出发,从上层物理建模、中层并行算法设计、低层软件实现与优化方法以及硬件的基本构件等多个层次分析了可扩展性的不连续非线性现象. 本节我们提出一个各个层次之间相互配合互相融合的大规模可扩展并行应用软件研究的新的系统研究方法和研究思路,可为后续更多大规模并行应用的研制与优化提供理论和方法指导.

3.1 两层协同设计

描述和克服异构众核大规模并行算法的可扩展性的不连续非线性现象,进而突破算法的有限区间可扩展性,在多个层次建立正确的理论解释,提出有效的解决方案,是缓解或解决可扩展性的不连续非线性现象最关键问题及难点. 对此,我们认为要采取分层次研究方法,如图 8 所示,即分别从物理模型、算法模型和性能模型这三个层次研究可扩展性问题,从研究初始就面向可扩展性问题. 其中物理层次主要从应用角度考虑高可扩展物理模型和数值算法,算法层次主要研究应用包含的核心函数的大规模高性能算法,而性能层次主要考虑硬件平台的高可用性能模型. 这一思路考虑了从物理模型到最后并行程序的完整高性能计算研究链,然后利用两层协同设计方法,即对于物理建模层次和算法层次的可扩展性问题,采用应用-算法协同设计,对于软件实现和硬件构件的可扩展性问题,采用算法-体系结构协同设计,这种思路系统研究并分层次考虑可扩展性. 以往大多数研究更多只关注算法层次问题,例如算法可扩展性或优化方法可扩展性,虽然研究对核心函数取得一系列研究成果,提出了大量优秀算法和关键优化技术,并取得良好的可扩展性,但整个应用程序自身并不能完全依赖其核心算法的性能和可扩展性,因此在实际应用常常不能达到层次间的完美匹配,且单独考虑某一层次可扩展无法满足整个应用的需求.

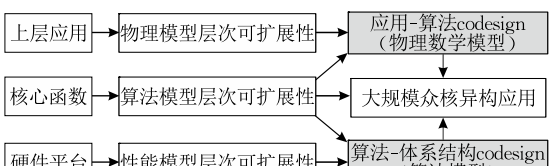

图 8 两层 codesign 研究方法

3.2　大规模并行程序优化

现有面向百万量级并行程序大多关注算法和实现两个层次的优化上，在设计和优化伊始就协同融合物理模型层次角度的研究并不多见，但这又是限制程序优化的关键瓶颈所在. 我们以全球气候模拟项目为例开展了百万核量级处理器上大气模式的可扩展性优化，着重从物理模型研究其适应计算平台并行度的方法. 进而采用两层协同设计方法研究解决方法，即对可扩展性细化得出的物理层次可扩展性问题采用应用-算法协同设计，对算法层次和优化层次可扩展性问题采用算法-体系结构协同设计，提出三层协同的大规模异构众核算法、并行实现优化方法.

具体而言，针对气候模拟动力方程组中核心方程的一些列核心参数，包括水平扩散系数，高纬灵活性跳点权重系数，时间分解算法比例系数，地表大气耗散系数等进行改变和调整，提高稳定性和计算效率. 最为重要的是从物理模型角度确定了大气模式可扩展性受限的主要原因在于纬度方向采用了并行效率较低的 FFT 滤波方式，限制了纬度方向的并行性，因此国内外已有的大气模式所采用的并行方案基本都是建立在二维划分上的，不能有效扩展到大规模系统.

我们提出并实现了面向高分辨率大气模式模拟的三维剖分并行算法. 通过对纬向进行剖分提升大气模式的并行度，减小了各进程负责的数据大小，降低了计算量与通信量. 针对已有的通信复杂度较高，限制纬度并行的 FFT 滤波方式，我们从物理模型角度设计了一系列优化滤波方式，最终设计实现了一种高纬度自适应滤波方法，完美支持了三维剖分的底层算法设计，从理论上支持百万核量级并行性. 这证明了我们提出的物理模型与算法模型相融合的协同研究方法这一思路的有效性.

在大规模并行软件实现一级，我们从算法与硬件协同设计角度，确定了不同硬件平台的具体实现和优化方法，以天河三号原型机为例，我们基于设计的三维剖分算法，进一步结合原型机的“MATRIX 2000＋”国产处理器的计算特性，对具体计算过程进行了多线程实现，采取了一系列优化方法，例如通过节点内共享内存多线程优化方法对大气模式最为重要的核心三层计算循环进行加速，共享了节点内计算数据的存储空间，提高了循环操作的执行效率. 这种算法体系结构协同设计思路支持了并行软件的高效实现.

我们对 0.5°分辨率的二维与三维剖分大气模式以及利用有限体积法的 NCAR CESM-1.0.3 进行了可扩展性和性能加速比对比测试. 测试结果如图 9 所示，0.5°的全球大气模式网格数为 720×361×30，131 072 核中共有 16 384 进程，每个进程包含 8 个线程并行处理核心循环，16 384 进程的网格配置为 32×32×16，因此每个进程处理的格点数近似为 22×11×2，若不进行三维划分，则根据 yz 方向格点粒度要求，最大进程并行度为 60×16＝960，理想情况下只能最大支持 960 个进程，加上处理核心循环的线程级并行，最大并行度不超过万核量级. NCAR 使用不同的数值方法，性能较高，但在物理层次，而 IAP 增加了一系列新方案，例如可允许替代、高纬灵活性跳点、时间分解算法及半拉格朗日水汽输送方案等，在模式检验的某些模拟能力上，例如对海平面气压场、纬向风场及温度场方面要优于 NCAR.

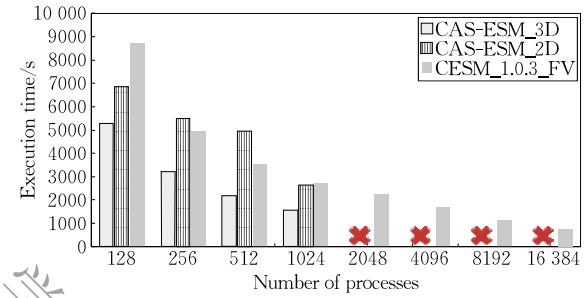

图 9　AGCM 大气动力框架可扩展性对比[102]

可以看到，三维剖分后的滤波算法的开销要高于二维剖分的 FFT 滤波，但是三维剖分的通信时间大大降低，从物理模型角度提高了可扩展性，并且三维自适应滤波方式也具有良好的可扩展性，支持大规模并行. 实验结果表明优化后的三维剖分大气模式性能可扩展至 13 万核. 且相比 8192 核，并行效率达到了 44%. 这一性能结果已达到国际领先水平，据我们所知，国际上尚未有针对大气模式百万核量级的研究成果，我们正在采用本文所提出的研究思路继续优化可扩展性和并行效率.

再以材料模拟为例. 在上一节中介绍了材料模拟两种主要方法的优势和缺点，可以看到，从物理模型看，两种方法有其各自适用范围. MD 的时间尺度基本限制在纳秒量级，只能进行短时间的模型，而 KMC 实现了整个模拟体系势能的随机跃迁过程，其时间步长取值达到了微秒甚至秒级，我们设计实现了一种 MD-KMC 两种物理模型层次结合的方法，并针对新型方法开展了一系列通信优化以及计算优化工作[102-103]，最终实现了百亿规模原子的 α-Fe 材料受辐照 19.2 天的损伤演化过程并行模拟，

如图 10 所示，在 624 万核(十万主核)神威太湖之光众核架构上获得了 75%的并行效率，执行时间为 8.6 h.

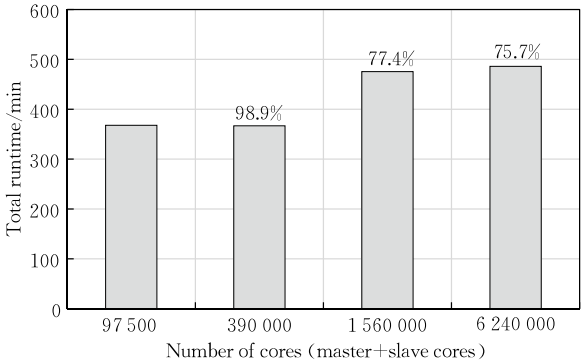

图 10 MD-KMC 结合的模拟方法在神威太湖之光系统上弱可扩展性的运行效率[103]

3.3 高效能并行编程模型设计

并行编程模型的设计不仅需要考虑上层并行应用或者算法的计算访存特征，还需要适应不断发展的底层并行计算机体系结构的变化. 首先，在算法实现层面，需要考虑如何将一个计算任务分配给多个线程或是进程，并通过线程或进程间的合作共同完成计算任务. 这其中，计算任务的划分，就必然会涉及数据的划分；而线程或进程间的合作，则大多数表现为线程或进程间的数据通信和同步. 简要而言，一个传统而典型的并行程序的设计通常包括：划分数据、划分计算任务、设计进程或线程间进行的数据通信、控制进程或线程间同步操作.

其次，在体系结构发展方面，集群是目前主流的超级计算机体系结构，Top500 超级计算机排名中大部分机器的体系结构为集群. 当前集群最为重要的异构性主要体现在两个方面：一是计算单元的异构，例如计算单元可能有多核处理器，以及众核处理器等；二是存储器的异构，例如主存与流多处理器片上存储器访问带宽相差数十倍. 大规模异构集群的并行程序设计更具挑战性. 程序员需要掌握各种计算单元的并行编程方法，依据更多类型存储器进行数据划分，通信模式也更为复杂等等.

针对这些问题和挑战，我们结合应用算法特点和体系结构发展特征，提出了 PARRAY[104]，一个面向并行异构集群程序开发的并行接口，对异构并行计算中的多种存储、控制流转等提供了统一的表达方式，在一定程度上很好地缓解了上述问题. PARRAY 采用统一的表达方式“嵌套维度”表示数据和线程，以及数据与线程的混合体. 通过这样的表达方式描述数据的元素排列、数据的划分模式、线程的组织结构等. PARRAY 可以指定数据所在的存储器类型，例如可以位于主存中、GPU 加速卡的显存、或是 GPU 加速卡的共享存储等等. 此外，线程间数据通信为显式，便于掌控性能，但同样通过隐藏通信细节简化编程. 相较于其他语言，PARRAY 程序代码的简短使得程序易于分析. 同时 PARRAY 充分暴露了性能相关的操作，且由类型集中反映计算中的关键问题，提供了非常好的研究环境. 事实上，根据现有工作，可以发现 PARRAY 在开发并行程序时具有表达方式简单灵活、程序简短、可以充分控制性能等优点.

4 结 论

本文发现并总结了高性能计算几十年发展历史中在多个层次上出现的两种可扩展性现象，即不连续现象和非线性现象. 为了解决这两种现象所带来的挑战，我们提出了在多个层次上研究可扩展性的必要性，特别强调了物理模型和数值算法层次上可扩展性研究的重要性，进而丰富了已有的算法-体系结构这一协同设计方法，在其上一层提出了应用-算法协同设计思路，形成了两层协同设计研究手段. 我们以全球气候模拟中的大气模式、材料模拟两个应用以及并行编程模型的设计研究为例进行了介绍，证明了分层考虑可扩展性和应用两层协同设计方法优化实现并行应用的必要性和有效性.

参 考 文 献

[1] Kramer W. Top500 versus sustained performance the top problems with the TOP500 list and what to do about them//Proceedings of the 21st 2012 International Conference on Parallel Architectures and Compilation Techniques (PACT). LA, Los Angeles, USA, 2012: 223-230

[2] Joubert W, Weighill D, Kainer D, et al. Attacking the opioid epidemic: Determining the epistatic and pleiotropic genetic architectures for chronic pain and opioid addiction//Proceedings of the ACM/IEEE Conference on Supercomputing. Seattle, USA, 2018: 1-14

[3] Yang Chao, Xue Wei, Fu Hao-Huan, et al. 10m-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'16). Denver, USA, 2016: 1-12

[4] Fu Hao-Huan, He Cong-Hui, Chen Bing-Wei, et al. 18.9p opss nonlinear earthquake simulation on sunway taihulight: Enabling depiction of 18-hz and 8-meter scenarios//Proceed-

ings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'17). Salt Lake City, USA, 2017: 1-12

[5] Kurth T, Treichler S, Romero J, et al. Exascale deep learning for climate analytics//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'18). Denver, USA, 2018: 1-12

[6] Rudi J, Malossi A C I, Isaac T, et al. An extreme-scale implicit solver for complex pdes: Highly heterogeneous flow in earth's mantle//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'15). Salt lake City, USA, 2015: 1-12

[7] Shaw D E, Grossman J P, Bank J A, et al. Anton 2: Raising the bar for performance and programmability in a specialpurpose molecular dynamics supercomputer//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'14). Portland, USA, 2014: 41-53

[8] Rossinelli D, Hejazialhosseini B, Hadjidoukas P, et al. 11 pop/s simulations of cloud cavitation collapse//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'13). Denver, USA, 2013: 1-13

[9] Ishiyama T, NitadoriK, Makino J. 4.45 p ops astrophysical n-body simulation on k computer—The gravitational trillion-body problem//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12). Salt lake City, USA, 2012: 1-10

[10] Hasegawa Y, Iwata J-I, Tsuji M, et al. First-principles calculations of electron states of a silicon nanowire with 100,000 atoms on the k computer//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'11). Denver, USA, 2011: 1-11

[11] Rahimian A, Lashuk I, Veerapaneni S, et al. Petascale direct numerical simulation of blood on 200k cores and heterogeneous architectures//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'10). New Orleans, USA, 2010: 1-11

[12] Shaw D E, Dror R O, Salmon J K, et al. Millisecond-scale molecular dynamics simulations on anton//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'09). Denver, USA, 2009: 1-11

[13] Gray J. What next?: A dozen information-technology research goals. Journal of ACM, 2003, 50(1): 41-57

[14] Sun Ning-Hui, et al. Scalable applications and systems. Information Technology Letter, 2009, 7(1): 11-19 (in Chinese)
(孙凝辉等. 可扩展应用与可扩展系统. 信息技术快报, 2009, 7(1): 11-19)

[15] Amdahl G M. Validity of the single processor approach to achieving large scale computingcapabilities//Proceedings of the Spring Joint Computer Conference (AFIPS'67). Atlantic City, USA, 1967: 483-485

[16] Gustafson J L. Reevaluating amdahl's law. Communication of ACM, 1988, 31(5): 532-533

[17] Sun Xian-He, Ni Lionel M. Another view on parallel speedup//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (Supercomputing'90). Los Alamitos, USA, 1990: 324-333

[18] Sun X H, Ni L M. Scalable problems and memory-bounded speedup. Journal of Parallel and Distributed Computing, 1993, 19(1): 27-37

[19] Hill M D, Marty M R. Amdahl's law in the multicore era. Computer, 2008, 41(7): 33-38

[20] Yao Erlin, Bao Yungang, Tan Guangming, Chen Mingyu. Extending amdahl's law in themulticore era. SIGMETRICS Performance Evaluation Review, 2009, 37(2): 24-26

[21] Yao Erlin, Bao Yungang, Chen Mingyu. What hill-marty model learn from and break through amdahl's law? Information Processing Letter, 2011, 111(23/24): 1092-1095

[22] Chi Li-Hua, Liu Jie, Hu Qing-Feng. Evaluation and test for scalability of numerical parallel computation. Journal of Computer Research and Development, 2005, 42(6): 1073-1078 (in Chinese)
(迟利华,胡庆丰,刘杰. 数值并行计算可扩展性评价与测试. 计算机研究与发展, 2005, 42(6): 1073-1078)

[23] Wang Yu-Li, Yang Xiao-Dong. A more effective scalability model for parallel system. Chinese Journal of Computers, 2001, 24(1): 84-90(in Chinese)
(王与力,杨晓东. 一种更有效的并行系统可扩展性模型. 计算机学报, 2001, 24(1): 84-90)

[24] Wang Zhi-Yuan, Yang Xue-Jun. Metrics of measuring parallel computing systems. Computer Engineering and Science, 2010, 32(10): 44-48(in Chinese)
(王之元,杨学军. 并行计算系统度量指标综述. 计算机工程与科学, 2010, 32(10): 44-48)

[25] Chen Jun, Li Xiao-Mei. Near-optimal scalability, a pratical scalability metric. Chinese Journal of Computers, 2001, 24(2): 179-182(in Chinese)
(陈军,李晓梅. 近优可扩展性:一种实用的可扩展性度量. 计算机学报, 2001, 24(2): 179-182)

[26] Zhang Li-Lun, Wu Jian-Ping, Song Jun-Qiang. Load-balancing evaluation model based on pre-defined variability. Journal of Computer Applicaiotns. 2009, 29(10): 2849-2851(in Chinese)
(张理论,吴建平,宋君强. 基于前验负载差异的负载平衡性能模型. 计算机应用, 2009, 29(10): 2849-2851)

[27] Moore G E. Cramming more components onto integrated circuits. Electronics, 1965, 38(8): 114

[28] Dennard R H, Gaensslen F, Yu H-N, et al. Design of ion-implanted MOSFET's with very small physical dimensions. IEEE Journal of Solid State Circuits, 1974, 9(5): 78-89

[29] Asanovic K, Bodik R, Demmel J, et al. A view of the parallel computing landscape. Communication of ACM, 2009, 52(10): 56-67

[30] Nvidia corporation, NVIDIA's Next Generation CUDA Compute Architecture: Fermi. Technical Document, 2009

[31] Nvidia corporation, NVIDIA Kepler GK110 Architecture. Whitepaper, 2014

[32] Nvidia corporation, NVIDIA GeForce GTX 750 Ti: Featuring First-Generation Maxwell GPU Technology, Designed for Extreme Performance per Watt. Technical Document, 2019

[33] George Kyriazis. Heterogeneous System Architecture: A Technical Review. Santa Clara, USA: Amd Corporation, Technical Report: Hsa14, 2014

[34] Vuduc R W, Czechowski K. What gpu computing means for high-end systems. IEEE Micro. 2011, 31(4): 74-78

[35] Barker K J, Kei D, Adolfy H, et al. Entering the petaflop era: The architecture and performance of Roadrunner//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'08). Denver, USA, 2008: 1-13

[36] Clayton J E. Low-cost, high-density memory packaging: A 64K X 9 DRAM SIP module. The International Journal for Hybrid Microelectronics, 1983, 22(4): 56-67

[37] Xilinx. Memory Interfaces Data Capture Using Direct Clocking Technique. San Jose, USA: Xilinx Technical Report: UG086, 2010

[38] Mittal S, Vetter J S. A survey of software techniques for using non-volatile memories for storage and main memory systems. IEEE Transactions on Parallel and Distributed Systems, 2015, 27(5): 1537-1550

[39] AnandTech. Intel Launches Optane Memory M.2 Cache SSDs For Consumer Market, Technical Document, 2017

[40] Zhang He. Development of IAP atmospheric general circulation model version 4.0 and its climate simulations. The Institute of Atmospheric Physics, Chinese Academy of Sciences, 2009(in Chinese)
(张贺. 大气环流模式 IAP AGCM4.0 的设计及其数值模拟. 中国科学院大气物理研究所, 2009)

[41] Wehner M, Oliker L, Shalf J. Towards ultra-high resolution models of climate and weather. International Journal of High Performance Computing Applications, 2008, 22(2): 149-165

[42] Neale R, Hannay C E, Zhang M, Taylor M A. The community atmosphere model (cam) in the cesm//Proceedings of the World Climate Research Program Open Science Conference. Denver, USA, 2011: 1156-1167

[43] Zhou L, Bao Q, Yu H. High-Resolusion FAMIL. Zhou T et al. (Eds.). Flexible Global Ocean-Atmosphere-Land System Model. Springer, London, 2014: 75-86

[44] Hoover W G. Modecular Dynamics. Lecture Notes in Physics (volume 258). Berlin: Springer, 1996

[45] Hastings W K. Monte carlo sampling methods using Markov chains and their applications. Biometrika, 1970, 57(1): 97-109

[46] Valiant L G. A bridging model for parallel computation. Communication of ACM, 1990, 33(8): 103-111

[47] Valiant L G. A bridging model for multi-core computing. Journal of Computer System and Science, 2011, 77(1): 154-166

[48] Culler D, Karp R, Patterson D, et al. Logp: Towards a realistic model of parallel computation//Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Denver, USA, 1993: 1-12

[49] Alexandrov A, Ionescu M F, Schauser K E, Scheiman C. Loggp: Incorporating long messages into the logp model one step closer towards a realistic model for parallel computation//Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures. Washington, USA, 1995: 95-105

[50] Frank M I, Agarwal A, Vernon M K. Lopc: Modeling contention in parallel algorithms//Proceedings of the 6th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York, USA, 1997: 276-287

[51] Moritz C A, Frank M I. Logpc: Modeling network contention in message-passing programs. IEEE Transactions on Parallel and Distributed Systems, 2001, 12(4): 404-415

[52] Ino F, Fujimoto N, Hagihara K. Loggps: A parallel computational model for synchronization analysis//Proceedings of the 8th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming. New York, USA, 2001: 133-142

[53] Chen Wen-Guang, Zhai Ji-Dong, Zhang Jin, Zheng Wei-Min. Loggpo: An accurate communication model for performance prediction of mpi programs. Science in China Series F: Information Sciences, 2009, 52(6): 1785-1791

[54] Cameron K W, Sun Xian-He. Quantifying locality effect in data access delay: Memory logp//Proceedings of Parallel and Distributed Processing Symposium. Paris, France, 2003: 1-11

[55] Dongarra J J, Luszczek P, Petitet A. The LINPACK Benchmark: Past, present and future. Concurrency and Computation: Practice and Experience, 2003, 25(9): 803-820

[56] Garbow BS. EISPACK—A package of matrix eigensystem routines. Computer Physics Communications, 1974, 7(4): 179-184

[57] Lawson C L, Hanson R J, Kincaid D, Krogh F T. Basic linear algebra subprograms for FORTRAN usage. ACM Transactions on Mathematics Software, 1979, 5(3): 308-323

[58] Anderson E, Bai Z, Bischof C, et al. LAPACK Users' Guide. Philadelphia: Society for Industrial and Applied Mathematics, 1999

[59] Dongarra J, Walker D. The Design of Linear Algebra Libraries for High Performance Computers. Philadelphia: Society for Industrial and Applied Mathematics, 1999

[60] Robert A van de Geijn with contributions by Philip Alpatov Greg Baker Carter Edwards John Gunnels. Using PLAPACK: Parallel Linear Algebra Package. Philadelphia: Society for Industrial and Applied Mathematics, 1999

[61] Reiley W C, Van De Geijn R A. Pooclapack: Parallel out-of-core linear algebrapackage. Texas: The University of Texas. Technical Report:tr-99-12-3345, 1999

[62] Agullo E, Demmel J, Dongarra J, et al. Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects. Journal of Physics, 2009, 35(7): 114-124

[63] Tomov S, Dongarra J, Baboulin M. Towards Dense Linear Algebra for Hybrid GPU Accelerated Manycore Systems. Tennessee: University of Tennessee Computer Science. Technical Report: TR-08-1253, 2008

[64] Nguyen A, Satish N, Chhugani J, et al. 3.5-D blocking optimization for Stencil computations on modern cpus and gpus//Proceedings of the ACM/IEEE Conference on Supercomputing. Seattle, USA, 2010:1-10

[65] Demmel J. Avoiding communication in numerical linear algebra//Proceedings of the 21st Algorithm Engineering and Experiments. San Diego, USA, 2011: 59-68

[66] Ballard G, Demmel J, Holtz O, Schwartz O. Communication-optimal paralleland sequential cholesky decomposition. SIAM Journal on Scientific Computing, 2010, 32(6): 3495-3523

[67] Ballard G, Demmel J, Holtz O, Schwartz O. Communication-optimal paralleland sequential cholesky decomposition: Extended abstract//Proceedings of the Annual ACM Symposium on Parallelism in Algorithms and Architectures. New York, USA, 2009: 245-252

[68] Ballard G, Demmel J, Holtz O, Schwartz O. Minimizing communication innumerical linear algebra. SIAM Journal on Matrix Analysis Applications, 2011, 32(3): 866-901

[69] Demmel J, Grigori L, Hoemmen M, Langou J. Communication-optimal paralleland sequential qr and lu factorizations. SIAM Journal of Scientific Computing, 2012, 34(1): 478-489

[70] Jeannot E, Namyst R, Roman J. A three-dimensional approach to parallel matrix multiplication//Proceedings of the International Euro-Par Conference. Rhodes Island, Greece, 2011: 65-77

[71] Edgar S, James D. Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms//Proceedings of the International Euro-Par Conference. Bordeaux, France, 2011:158-170

[72] Pekurovsky D. P3DFFT: A framework for parallel computations of Fourier transforms in three dimensions. SIAM Journal on Scientific Computing, 2012, 34(4): 192-209

[73] Frigo M, Johnson S G. The design and implementation of fftw3. Proceedings of the IEEE, 2005, 93(2): 216-231

[74] Püschel M, Moura J M F, Johnson J, et al. SPIRAL: Code generation for DSP transforms. Proceedings of the IEEE, 2005, 93(2): 232-275

[75] Nukada A, Matsuoka S. Auto-tuning 3-d fft library for cuda gpus//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. New York, USA, 2009: 1-10

[76] Liang Gu, Jakob S, Li Xiaoming. Using gpus to compute large out-of-card ffts//Proceedings of the International Conference on Supercomputing. Austin, USA, 2011: 255-264

[77] Li Yan, Zhang Yun-Quan, Liu Yi-Qun, et al. Mpfft: An auto-tuningfft library for opencl gpus. Journal of Computer Science and Technology, 2013, 28(1): 90-105

[78] Mundo C del, Feng W. Towards a performance-portable fft library for heterogeneous computing//Proceedings of the ACM International Conference on Computing Frontiers. Atlanta, USA, 2014: 1-10

[79] Czechowski K, McClanahan C, Battaglino C, et al. Prospects for scalable 3D FFTs on heterogeneous exascale systems//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. New York, USA, 2011: 1-10

[80] Czechowski K, Battaglino C, McClanahan C, et al. On the communication complexity of 3d ffts and its implications for exascale//Proceedings of the 26th ACM International Conference on Supercomputing. Vienna, Austria, 2012: 205-214

[81] Li N, Laizet S. 2DECOMP&FFT—A highly scalable 2D decomposition library and FFT interface//Proceedings of the Cray User Group 2010 conference. Edinburgh, UK, 2010: 1-10

[82] Chen Yi-Feng, Cui Xiang, Mei Hong. Large-scale FFT on GPU clusters//Proceedings of the International Conference on Supercomputing 2010. Tokyo, Japan, 2010: 315-324

[83] Sun Xiang-Zheng, Zhang Yun-Quan, Wang Ting, et a. Optimizing spmv for diagonal sparse matrices on gpu//Proceedings of the International Conference on Parallel Processing. San Diego, USA, 2011: 492-501

[84] McKellar C, man Jr E G Co. Organizing matrices and matrix operations for paged memory systems. Communication of the ACM, 1969, 12(3): 153-165

[85] Wolf M E, Lam M S. A data locality optimizing algorithm//Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation. San Jose, USA, 1991: 35-46

[86] Song Yonghong, Li Zhiyuan. New tiling techniques to improve cache temporal locality//Proceedings of the ACM SIGPLAN 1999 Conference on Programming Language Design and Implementation. Santa Barbara, USA, 1999: 215-228

[87] Bondhugula U, Hartono A, Ramanujam J, Sadayappan P. A practical automatic polyhedral parallelizer and locality optimizer//Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation. Snowbird, USA, 2008: 101-113

[88] Rivera G, Tseng C-W. Tiling optimizations for 3d scientific computations//Proceedings of the 2000 ACM/IEEE Conference on Supercomputing; International Conference on High Performance Computing, Networking, Storage and Analysis. Washington, USA, 2000: 88-97

[89] Frigo M, Leiserson C E, Prokop H, Ramachandran S. Cache-oblivious algorithms//Proceedings of the 40th Annual Symposium on Foundations of Computer Science. Washington, USA, 1999: 285-296

[90] Strzodka R, Shaheen M, Pajak D, Seidel H-P. Cache oblivious parallelograms in iterative stencil computations//Proceedings of the 24th ACM International Conference on Supercomputing. Tokyo, Japan, 2010: 49-59

[91] Frigo M, Strumpen V. The cache complexity of multithreaded cache oblivious algorithms//Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures. New Orleans, USA, 2006: 271-280

[92] Tang Yuan, You Ronghui, Kan Haibin, et al. Cache-oblivious wavefront: Improving parallelism of recursive dynamic programming algorithms without losing cache-efficiency//Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Yorktown, USA, 2015: 205-214

[93] Yuan Liang, Zhang Yunquan, Guo Peng, Huang Shan. Tessellating stencils//Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis SC'17. Denver, USA, 2017: 1-13

[94] Blumofe R D, Leiserson C E. Scheduling multithreaded computations by work stealing. Journal of the ACM, 1999, 46(5): 88-98

[95] Agrawal K, He Y, Leiserson C E. Adaptive work stealing with parallelism feedback//Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Boston, USA, 2007: 112-120

[96] Message passing interface forum. Mpi: A message-passing interface standard. version 2.1. Stuttgart, Germany: High Performance Computing Center, 2008

[97] Chapman B, Jost G, van der Pas R. Using OpenMP. Portable Shared Memory Parallel Programming. Boston: The MIT Press, 2007

[98] Lee H. The OpenCL Specification. Beaverton, USA: Khronos Group. Technical Report: Openl-21, 2018

[99] Gropp W, Lusk E, Thakur R. Using MPI-2 Advanced Features of the Message-Passing Interface. Boston: The MIT Press, 1999

[100] Chandrasekaran S, Juckeland G. OpenACC for Programmers: Concepts and Strategies. London, United Kingdom: Pearson, 2007

[101] Parallel Programming Laboratory. The Charm++ Parallel Programming System. Champaign, USA: University of Illinois at Urbana-Champaign, Technical Report: Charm690, 2018

[102] Wu Baodong, Li Shigang, Zhang Yunquan, Nie Ningming. Hybrid optimization strategy for the communication of large-scale kinetic monte carlo simulation. Computer Physics Communications, 2017, 211(8): 113-123

[103] Li Shigang, Wu Baodong, Zhang Yunquan, et al. Massively scaling the metal microscopic damage simulation on sunway taihulight supercomputer//Proceedings of the International Conference on Parallel Processing. Oregon, USA, 2018: 1-11

[104] Chen Yifeng, Cui Xiang, Mei Hong. PARRAY: A unifying array representation for heterogeneous parallelism//Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Pittsburg, USA, 2012: 171-180

ZHANG Yun-Quan, Ph. D., professor, Ph. D. supervisor. His research interests include high performance computing and big data processing.

YUAN Liang, Ph. D., assistant professor. His research interests include parallel computational model and parallel algorithm.

CHEN Yi-Feng, Ph. D., professor, Ph. D. supervisor. His research interests include high performance computing and big data processing.

FENG Xiao-Bing, Ph. D., professor, Ph. D. supervisor. His research interests include advance compiling technology and related tools.

ZHANG He, Ph. D., associate professor, master supervisor. His research interests include earth system model and climate simulations.

Background

High Performance Computing (HPC) is the practice of computational science. It promotes the scientific progress of many fields and serves as an irreplaceable foundation for national economic development. Scalability is regarded as one of the most challenging problem of the computer science in the 21th century. Especially with more and more obvious development trend of future hundreds of Petaflops and Exascale supercomputing systems architecture my adopt heterogeneous

manycore architecture, more and more parallel applications face with the performance continuous linear scalable and portability challenge.

This project arose from our obersevations of two interesting phenomena in multi-level of HPC including physical models, parallel algorithms, parallel software implementations and underlying hardware architectures: discountinuous and nonlinear. We identify this phenomenon from the perspective of the large-scale parallel software development with a timescale of decades and the scalabilities of thousands and millions cores. We call it MDNS, A Multi-level Discontinuous and Nonlinear Scalability.

This paper presents a detailed study of this phenomenon especially the scalability development in HPC. In particular, we identify the scalability challenges from both the problem level and the hardware level. To improve these challenges, we propose a two-level codesign methodology. It includes the architecture-algorithm codesign and model-algorithm codesign. In the past the former codesign has been well studied. We think the last codesign is of the same significance that should be further deeply studied. We provide an example of our Global Climate Simulation optimization to the effectiveness of the methodology of the two-level codesign.

This research is mainly supported by a Key Program of NSFC (No. 61432018) "Scalable Parallel Algorithms and Optimization methods for Global Climate Simulation and Direct Turbulence Simulation on million cores".