计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200700059

面向 SW26010 处理器的三维 Stencil 自适应分块参数算法

朱 雨 庞建民 徐金龙 陶小涵 王 军
数学工程与先进计算国家重点实验室(信息工程大学) 郑州 450000
(i_zhuyu@126.com)

摘 要 Stencil 计算是科学应用中的一类重要计算,而分块是提升 Stencil 计算数据局部性的关键技术。针对现有三维 Stencil 优化在 SW26010 处理器上缺少时间分块以及分块参数需手工调优的问题,引入时间分块,提出了面向 SW26010 处理器的三维 Stencil 自适应分块参数算法。通过建立性能分析模型,结合硬件计算能力及存储容量等限制因素,文中系统地分析了分块参数对模型性能的影响,判断性能瓶颈,指导分块参数的优化方向。基于性能分析模型,自适应分块参数算法可给出预测性能最优时的分块参数,有利于三维 Stencil 在 SW26010 处理器上的快速优化部署。选取了三维 7 点和三维 27 点 Stencil 算例进行实验。与未使用时间分块的三维 Stencil 优化相比,以上两个算例在自适应选择的分块参数下可以达到 1.47 和 1.29 的加速比,且实际最优分块参数与理论最佳分块参数一致,这验证了所提性能分析模型及自适应分块参数算法的有效性。

关键词:三维 Stencil 计算;SW26010;分块大小;性能分析模型
中图法分类号 TP391

Adaptive Tiling Size Algorithm for 3D Stencil Computation on SW26010 Many-core Processor

ZHU Yu,PANG Jian-min,XU Jin-long,TAO Xiao-han and WANG Jun
State Key Laboratory of Mathematical Engineering and Advanced Computing,PLA Information Engineering University,Zhengzhou 450000,China

Abstract Stencil computation is an important part of scientific computing and large-scale applications. Tiling is a widely-used technique to explore the data locality of Stencil computation. In the existing methods of 3D Stencil optimization on SW26010,time tiling is rarely used and manual tuning is needed for tiling size. To solve this problem,this paper introduces time tiling method and proposes an adaptive tiling size algorithm for 3D Stencil computation on SW26010 many-core processor. By establishing a performance analysis model,we systematically analyze the influence of tiling size to the performance of 3D Stencil computation,identify the performance bottleneck and guide the optimization direction under the hardware resource constraints. Based on the performance analysis model,the adaptive tiling size algorithm provides the predicted optimal tiling size,which can be helpful to deploy 3D Stencil rapidly on SW26010 processor. 3D-7P Stencil and 3D-27P Stencil are selected for experiment. Compared with the result lacking of time tiling,the speedup rates of the above two examples with optimal tiling size given by our algorithm can reach 1.47 and 1.29,and the optimal tiling size in experiment is consistent with that given by our model,which verify the proposed performance analysis model and tiling size adaptive algorithm.

Keywords 3D Stencil computing,SW26010,Tiling size,Performance analysis model

1 引言

Stencil 计算是科学应用中的一类重要计算,被广泛应用于偏微分方程求解、高斯赛德尔方法、流体力学计算以及地球系统模拟等方面。Stencil 计算描述了一种在大量时间步迭代下遍历结构网格并按固定模式更新网格点的算法。结构网格中的每个点根据其邻居子集进行更新的固定模式称为 Stencil。根据数据维度和更新时所需邻居子集的不同,Stencil 计算分为不同的类型,例如 LBM(Lattice Boltzmann methods)核心为三维 7 点 Stencil(3D-7P),JM(Jacobi Method)核心为二维 5 点 Stencil(2D-5P)。

Stencil 计算的计算量和内存消耗随计算规模的增大呈线性增长,适合进行大规模并行。随着硬件的发展,Stencil 计算的应用平台从 CPU 逐渐转向多核和众核平台。Stencil 计算的计算强度低且访存不连续,其性能主要受内存访问延迟和数据低效重用的限制。分块是提升 Stencil 计算的并行性和数据局部性的关键技术。空间分块通过特定的遍历顺序充分利用缓存中的数据,时间分块通过对时间维度的划分增加了数据重用。目前,针对 Stencil 计算的分块方法已经有了很详细的研究。

到稿日期:2020-07-09 返修日期:2020-08-30 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:之江实验室重大科研基金资助项目(2018FD0ZX01)
This work was supported by the Major Scientific Project of Zhejiang Lab Advanced Industrial Network Security Platform(2018FD0ZX01).
通信作者:徐金龙(longkaizh@126.com)

我国的“神威·太湖之光”是国际上首台峰值计算性能超过 100PFlops 的超级计算机，所搭载的处理器是我国自主研发的 SW26010 众核处理器。为充分发挥 SW26010 处理器的性能，针对 Stencil 计算的优化研究十分重要。与二维 Stencil 相比，三维 Stencil 在 SW26010 处理器上的时间分块优化更具有挑战性。首先，三维 Stencil 的时间分块对存储容量要求更高，而 SW26010 处理器的运算核心存储容量有限；其次，时间分块会造成冗余计算，对性能的负面影响可能会掩盖其对性能的提升。“神威·太湖之光”上已有的三维 Stencil 优化工作大多只采用空间分块方法，同时利用 DMA 双缓冲、向量化和寄存器通信等技术进一步优化性能。在 SW26010 处理器上，时间和空间分块参数对三维 Stencil 计算的性能有较大影响。对于给定类型的三维 Stencil，在实际优化过程中往往需要针对分块参数进行手工调优。

针对以上问题，本文提出了面向 SW26010 处理器的三维 Stencil 自适应分块参数算法，在实际应用中可大大减少针对分块参数的手工调优工作，实现三维 Stencil 在 SW26010 处理器上的快速优化部署。本文的主要贡献如下：

(1)建立了面向 SW26010 众核处理器的三维 Stencil 性能分析模型，针对分块、DMA 双缓冲和向量化技术，结合存储容量等限制因素，给出了执行时间的计算函数，判断程序性能瓶颈，用于指导分块参数的优化。

(2)提出了自适应分块参数算法，在性能分析模型的指导下，对于任意类型的三维 Stencil，给出了模型预测最优性能下的分块参数。

(3)选取 3D-7P 和 3D-27P 算例进行实验，得到的最优分块参数与自适应分块参数算法给出的理论最优分块参数相符，性能分析模型的预测执行时间误差约为 7.2%。与未进行时间分块相比，两个算例在算法给出的分块参数下的性能加速比分别达到了 1.47 和 1.29。

本文第 2 节介绍了三维 Stencil 在 SW26010 处理器上的分块、DMA 双缓冲和向量化等优化技术；第 3 节针对分块方法建立了性能分析模型，分析了分块参数对程序性能的影响；第 4 节基于性能分析模型提出了自适应分块参数算法；第 5 节通过实验验证了所提算法和模型的有效性；最后总结全文。

2　背景及相关工作

2.1　SW26010 处理器

SW26010 处理器采用了异构众核架构，如图 1 所示。

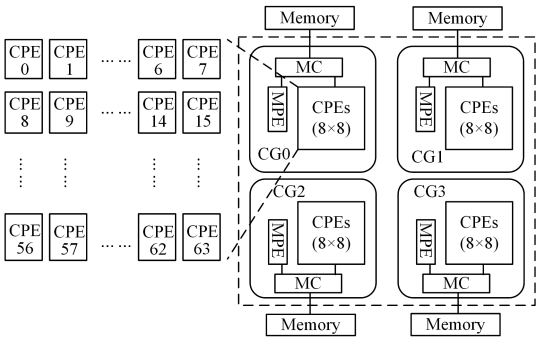


图 1　SW26010 处理器的架构

Fig. 1　SW26010 many-core processor architecture

处理器上集成了 4 个核组，单个核组包括 1 个管理核心(Management Processing Element，MPE)和 64 个运算核心(Computing Processing Elements，CPEs)，CPE 之间通过 8×8 Mesh 拓扑通信网络连接，可以进行寄存器通信。MPE 和 CPE 均支持 256 位的向量指令。MPE 支持中断和乱序执行等复杂逻辑处理，通常负责运算核心的管理和核组间的通信任务；CPE 只支持用户模式，主要负责处理逻辑简单的计算密集任务。

SW26010 处理器单个核组拥有大小为 8GB 的 DDR3 片下主存。MPE 具有两级缓存结构，一级缓存包括 32kB 指令缓存和 32kB 数据缓存，二级缓存由指令和数据共享，大小为 256kB。每个 CPE 具有 16kB 的一级指令缓存和 64kB 的软件管理的片上 SPM(Scratch Pad Memory)。相比传统的硬件管理的缓存，SPM 具有硬件结构简单、访存延迟低、带宽高等优势，但需要用户显式地通过 DMA (Direct Memory Access)方式完成数据的传输和布局。CPE 访问主存的方式包括全局离散访存和 DMA 批量访存，访问主存的延迟远大于访问片上 SPM 的延迟。

2.2　相关工作

分块技术是提升 Stencil 计算的数据局部性和并行性的关键技术[1-7]。针对 Stencil 的分块形状、大小以及自动代码生成和调优的研究工作有很多，大部分的分块方法可以扩展到高维数据空间。

在异构平台上，针对 Stencil 计算较为常用的分块技术是 overlapped tiling[8]。通过引入冗余的数据传输和计算，overlapped tiling 可消除数据块间的依赖，从而提升数据的局部性。冗余传输和计算的范围取决于重叠维度以及时间块的大小等因素。另外，冗余的数据传输和计算可能抵消掉分块带来的性能提升。

相比二维 Stencil，三维 Stencil 应用 overlapped tiling 方法时对计算核心的存储容量要求更高。由于异构平台片上存储容量的限制，现有的三维 Stencil 在异构平台上的并行优化通常在二维平面上实施 overlapped tiling，在另一维度上以流水方式串行计算。Nguyen 等[9]将该方法称为 2.5D 空间分块方法，结合 1D 时间分块，实现了三维 Stencil 在 GPU 上的 3.5D 分块。Rawat 等[10]通过有效管理 GPU 上的共享内存和寄存器等内存资源，结合二维分块和流水计算提升了 3D Stencil 计算在 GPU 上的性能。Szustak 等[11]提出了针对 Intel MIC 架构的 MPDATA Stencil 计算的(3+1)D 分解。Williams 等[12]在 CELL 处理器上实现了 Stencil 计算的分块并行，运算核心采用流水缓冲方式掩盖访存延迟，同时通过时间分块增加数据重用。Ao 等[13]在“神威·太湖之光”的千万核心上实现了大气模拟应用核心 3D-13P Stencil 计算的并行优化，采用了 2.5D 空间分块方法，由于 SPM 大小的限制，未采取时间分块方法。针对 overlapped tiling 产生的冗余计算，Meng 等[14]在 GPU 上建立了一个性能模型，以分析时间分块大小对性能的影响，为二维和三维 Stencil 自动选择最优的时间分块大小。

关于 Stencil 计算调优和代码自动生成的研究有很多[15-18]。Datta 等[16]提出了一种自动调优方法，针对不同的

架构选取适当的优化策略。Matsumura 等[17]提出了一个自动 Stencil 框架 AN5D,对给定的 Stencil 计算的 C 源代码进行转换和优化,生成相应的 CUDA 代码。Diamond 分块[19]是基于多面体模型的分块技术,为了方便自动调优和移植,分块大小一般作为参数,在编译运行过程中是固定的。

在 SW26010 处理器上,已有的三维 Stencil 优化工作[13,20-21]大多采用二维 overlapped tiling 和流水计算相结合的分块方法(2.5D 空间分块),通过手工调优确定分块大小。目前三维 Stencil 计算在 SW26010 众核处理器上的优化技术主要包括:通过 2.5D 空间分块有效利用 DMA 传输带宽,减少冗余数据传输;通过 DMA 双缓冲实现 DMA 时间和计算时间的重叠,隐藏访存延迟;通过向量化提升计算性能;通过寄存器通信减少冗余数据传输和计算等。由于运算核心存储空间有限,已有的工作较少使用时间分块方法,而且空间分块参数需要根据具体应用进行人工调优。时间分块和空间分块参数的选取对三维 Stencil 计算的性能有较大影响,对于给定类型的 Stencil 计算,应该选择合适的分块参数以适应 SW26010 处理器硬件架构,使其达到较优的运行性能。

3 SW26010 上的三维 Stencil 优化方法

本节介绍三维 Stencil 在 SW26010 处理器上的 2.5D 空间分块、1D 时间分块和 DMA 双缓冲技术等优化方法。

3.1 2.5D 空间分块

SW26010 处理器上各运算核心具有独立的存储空间,对三维 Stencil 数据进行划分并分配给各运算核心是实现并行的必要工作。SW26010 处理器运算核心的存储空间仅有 64 kB,往往无法存储需要计算的所有数据,通过对三维 Stencil 进行 2.5D 空间分块可充分利用加载到运算核心存储空间中的数据。

2.5D 空间分块方法如图 2 所示,设三维数据在内存中存储的优先级依次为 X,Y,Z,2.5D 分块在二维(XY)平面进行 overlapped tiling 分块并分配到各运算核心上,在另一维度(Z)以流水方式串行计算,每次更新一个平面。

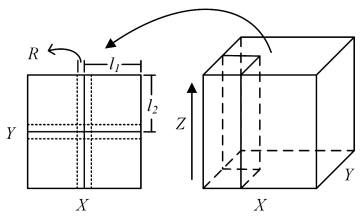


图 2 2.5D 分块示意图

Fig. 2 2.5D blocking

二维 overlapped tiling 分块首先将 XY 平面划分为多个 XY 子平面,设 X 维分块大小为 l_1,Y 维为 l_2。为保证 $l_1 \times l_2$ 大小的子平面的边界点可正确更新,传输数据需包含边界的邻居元素,设 Stencil 计算所需的邻居元素宽度为 R,每个子平面大小应为 $(l_1+2\times R)\times(l_2+2\times R)$,相邻分块具有重叠的部分。

为保证 Z 方向可计算,运算核心至少存储 Z 方向上连续 $2R+1$ 个子平面,另外还需 1 个写缓冲区,用于保存计算结果。在这种方式下,Z 方向每个平面只需被读入一次,相比 3D 分块可以避免 Z 方向上平面的重复读取,从而降低额外的带宽需求。其缺点是在 Z 方向上的计算只能串行进行。

3.2 DMA 双缓冲

针对 2.5D 分块 Z 方向的流水计算方式,根据运算核心的异步 DMA 机制,额外增加一个读缓冲区可使 Z 方向上第 $k+R+1$ 个子平面的读取和第 k 个平面的计算同时进行,增加一个写缓冲区可使第 $k-1$ 个子平面运算结果的写回和第 k 次计算同时进行。这种通过增加缓冲区来实现 DMA 和计算时间隐藏的方法称为 DMA 双缓冲机制。算法 1 给出了 SW26010 处理器运算核心上三维 Stencil 基于 2.5D 空间分块实现 DMA 双缓冲的过程。

算法 1 DMA 双缓冲算法

```
input:sub-block B
output:sub-block B' after one time step
1. N=2R+2
2. c=0
3. c'=1
4. for k=0 to 2R
5.   get(B(k),b(k))
6. endfor
7. get(B(2R+1),b(2R+1)) ‖ comp(R,output(c))
8. for k=2R+2 to Z+2R-1
9.  get(B(k),b(k%N));put(output(c'),B(k-R-2)) ‖ comp(k-
    R-1,output(c))
10.  c=c⊕1
11.  c'=c'⊕1
12. endfor
13.  comp(Z+R-1,output(c)) ‖ put(output(c'),B(Z+R-2))
14. put(output(c),B(Z+R-1))
```

算法 1 中,N 表示读缓冲区 b 存储的子平面个数,子数据块在 Z 方向上的长度范围为 $[0, Z+2R-1]$。$get(B(k), b(k))$ 函数表示通过 DMA 方式将子数据块 B 在 Z 方向上第 k 个子平面 $B(k)$ 读取至运算核心存储中的 $b(k)$ 缓冲区。$comp(k-R-1, output(c))$ 表示更新 Z 方向上第 $k-R-1$ 个子平面并将结果写至缓冲区 $output(c)$ 中。$put(output(c'), B(k-R-2))$ 函数表示将更新后的数据写回片下存储对应位置。变量 c 和 c' 用于切换写缓冲区。符号“‖”表示计算和 DMA 数据读写同时进行。

算法流程如下:

Step 1 初始读入 $2R+1$ 个子平面,在读入第 $2R+2$ 个子平面的同时计算第 $R+1$ 个平面。

Step 2 沿 Z 方向流水计算,读取第 k 个子平面和写回第 $k-R-2$ 个子平面数据的同时可计算第 $k-R-1$ 个子平面,实现 DMA 访存和计算时间的重叠。

Step 3 计算尾部数据并将结果写回。

以邻居宽度 $R=1$ 为例,图 3 给出了运算核心 SPM 的空间布局情况。箭头表示 Z 方向上第 k 次迭代的 DMA 读写和计算情况。在读取第 k 个子平面及写回第 $k-3$ 个子平面的同时,可计算第 $k-2$ 个子平面,实现 DMA 和计算时间的隐藏。

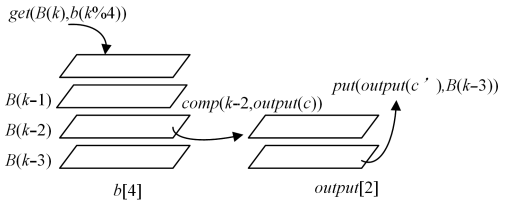


图 3　SPM 空间布局

Fig. 3　SPM allocation

DMA 和计算时间的隐藏程度取决于 DMA 带宽及运算核心的计算性能。图 4 给出了 DMA 时间大于计算时间的迭代时间示意图。此时迭代时间取决于 DMA 时间，Stencil 计算性能处于 DMA 瓶颈，只能通过优化 DMA 传输提升性能；反之程序性能处于计算瓶颈，需要提升计算性能。

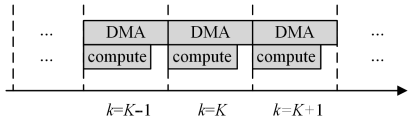


图 4　DMA 与计算时间重叠示意图

Fig. 4　Overlap of DMA and computation time

当 Stencil 计算性能处于 DMA 瓶颈时，运算核心的计算资源存在空闲状态。优化 DMA 瓶颈的方法包括减少冗余数据传输、提高 DMA 带宽等。Saidi 等[22]研究了二维数组在异构多核架构上的 DMA 传输性能模型。影响 DMA 传输的因素包括 DMA 初始化时间、DMA 跨步开销、DMA 带宽和传输子平面的大小等。该模型假设 DMA 传输带宽只与运算核心个数有关。在 SW26010 处理器上，DMA 传输带宽还与 DMA 连续传输粒度相关，粒度过小往往会导致 DMA 带宽减小[23]。

3.3　1D 时间分块

时间分块方法对一维时间轴进行划分，可以增加数据重用，减少运算核心的数据传输次数。设每个时间块内的时间步长为 j，单个时间块内需要进行一次 DMA 读写和 j 次时间迭代计算。每次计算时，子平面边界上宽度为 R 的数据由于缺少邻居元素未得到更新，在下一次计算时将被舍弃。为保证 j 次计算后 $l_1\times l_2$ 大小的子平面数据有效，初始读取的子平面大小应为 $(l_1+j\times 2\times R)\times(l_2+j\times 2\times R)$。运算核心存储空间上需要增加 $(j-1)\times(2R+1)$ 个子平面缓冲区以存储计算的中间结果，保证 Z 方向上的计算可流水进行。以 $R=1$ 为例，图 5 给出了时间分块参数 $j=2$ 时片上 SPM 空间布局情况，*middle* 数组用于存储中间计算结果，*comp2* 函数表示对子平面进行 2 次计算。

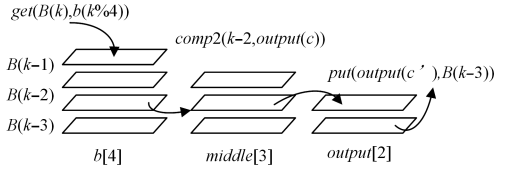


图 5　$j=2$ 时的 SPM 空间布局

Fig. 5　SPM allocation when $j=2$

4　性能分析模型

由于空间和时间分块参数对程序性能有较大影响，为分析程序性能和分块参数的关系，从理论上指导分块参数的优化，本节构建了一个面向 SW26010 处理器的三维 Stencil 性能分析模型，考虑到 SPM 空间大小的限制，给出了执行时间和硬件参数、Stencil 参数及分块参数的关系函数。

4.1　符号说明

表 1 列出了性能模型中的符号说明。其中硬件参数包括运算核心数目、运算核心的存储空间大小和 DMA 带宽。考虑到 SW26010 处理器上 DMA 连续传输粒度对 DMA 带宽的影响，用 $\alpha(l_1)$ 表示 DMA 连续传输粒度为 l_1 时单个运算核心的平均 DMA 带宽。三维 Stencil 的参数包括邻居元素宽度、输入数据规模、时间迭代次数、单个元素大小。

表 1　符号说明

Table 1　Symbols and their descriptions

参数符号	符号说明
p	运算核心数目
C	运算核心存储大小
$\alpha(l_1)$	运算核心平均 DMA 带宽
R	邻居元素宽度
X,Y,Z	输入数据规模
q	时间迭代次数
b	单个元素大小
ω	单个元素计算时间
ω_S	向量化后单个元素平均计算时间
l_1	X 方向分块大小
l_2	Y 方向分块大小
j	单个时间块内时间步长

ω 表示运算核心上单个元素计算的时间，其大小与硬件计算能力及 Stencil 类型有关。在 SW26010 处理器上，ω 的大小取决于 Stencil 计算的类型，如 3D-27P 的 ω 值大于 3D-7P 的 ω 值。考虑到向量化对计算性能的提升，用 ω_S 表示经过向量化之后单个元素的平均计算时间。

其他参数为分块参数，包括空间分块大小 l_1，l_2，以及时间分块大小 j，j 表示单个时间块内的时间步长。

选取 SW26010 处理器上的单个核组作为优化平台，将硬件参数代入性能模型中，得到 $p=64$，$C=64$ kB，DMA 带宽可采用 $p=64$ 时不同传输粒度下的 DMA 传输带宽实测值[23]。

4.2　SPM 空间限制

运算核心存储空间的大小限制了空间分块和时间分块的大小。如图 3 所示，不考虑时间分块的情况下，运算核心需要存储 $2R+2$ 个读缓冲区和 2 个写缓冲区。每个读缓冲区存储的元素个数 $S_r=(l_1+2\times R)\times(l_2+2\times R)$，每个写缓冲区存储元素个数 $S_w=l_1\times l_2$。

考虑 1D 时间分块，需要增加中间结果缓冲区以存储中间计算结果。每个时间块内时间步长为 j，时间块内每增加一个时间步长则需增加 $2R+1$ 个中间结果缓冲区。每次计算后舍弃边界上宽度为 R 的未更新数据，第 i 次计算的中间结果缓冲区大小为：

$$S_{m,i}=(l_1+2\times(j-i)\times R)\times(l_2+2\times(j-i)\times R) \tag{1}$$

读缓冲区大小为：

$$S_r=(l_1+2\times j\times R)\times(l_2+2\times j\times R) \tag{2}$$

写缓冲区大小为：

$$S_w=l_1\times l_2 \tag{3}$$

其中，l_1，l_2 和 j 的取值需要满足所有缓冲区大小不超过运算

核心存储空间大小，根据此条件可列出不等式：

$$(2R+2)\times S_r+(2R+1)\times\sum_{i=1}^{j-1}S_{m,i}+2\times S_w<\frac{C}{b} \tag{4}$$

其他限制条件如下：

$$l_1\geqslant(2R+1),l_2\geqslant(2R+1)$$
$$j\in N,l_1\in N,l_2\in N,\frac{X}{l_1}\in N,\frac{Y}{l_2}\in N \tag{5}$$

4.3 执行时间

SW26010 处理器上并行 Stencil 计算的执行时间包括 DMA 时间和计算时间。2.5D 空间分块和 DMA 双缓冲机制实现了 DMA 时间和计算时间的隐藏。考虑到理想情况下分配给各运算核心的任务是均匀的，则三维 Stencil 计算的总执行时间为：

$$T=MAX(T_{DMA},T_c) \tag{6}$$

其中，T_{DMA} 和 T_c 分别表示单个运算核心的 DMA 总时间和计算总时间。

经过时间分块后，时间块的个数为 q/j。经过 2.5D 空间分块之后，每个子数据块的规模为 $l_1\times l_2\times Z$。设 n 为每个运算核心需要计算的数据块个数，则有：

$$n=\frac{X}{l_1}\times\frac{Y}{l_2}\times\frac{1}{p} \tag{7}$$

根据算法 1，Z 方向上的流水串行计算过程共有 $Z-2$ 次迭代实现了 DMA 和计算时间的隐藏，忽略起始和末尾未重叠部分。用 t_c 表示每次迭代中计算部分的耗时，用 t_{DMA} 表示每次迭代中子平面的 DMA 读取和写回所需的时间。

t_{DMA} 的计算方法如下：

$$t_{DMA}=\frac{S_r\times b}{\alpha(l_1)}+\frac{S_w\times b}{\alpha(l_1)} \tag{8}$$

t_c 的计算方法如下：

$$t_c=\omega\times(S_w+\sum_{i=1}^{j-1}S_{m,i}) \tag{9}$$

由以上定义可得出单个运算核心在 q/j 个时间块内完成 n 个数据块的更新所需的 DMA 时间和计算时间：

$$T_{DMA}=t_{DMA}\times(Z-2)\times n\times\frac{q}{j} \tag{10}$$

$$T_c=t_c\times(Z-2)\times n\times\frac{q}{j} \tag{11}$$

将式(7)—式(9)代入式(10)和式(11)可得：

$$T_{DMA}=\left[\frac{S_r\times b}{\alpha(l_1)}+\frac{S_w\times b}{\alpha(l_1)}\right]\times(Z-2)\times\frac{X}{l_1}\times\frac{Y}{l_2}\times\frac{1}{p}\times\frac{q}{j} \tag{12}$$

$$T_c=\omega\times(S_w+\sum_{i=1}^{j-1}S_{m,i})\times(Z-2)\times\frac{X}{l_1}\times\frac{Y}{l_2}\times\frac{1}{p}\times\frac{q}{j} \tag{13}$$

令 ω 为自变量，当 j,l_1,l_2 取固定值时，T_{DMA} 为定值，T_c 是 ω 的一次函数。令 ω_j 为 $T_c=T_{DMA}$ 时 ω 的值。由式(6)可得到分段函数 T-ω 如下：

$$T=\begin{cases}T_{DMA} & \omega\leqslant\omega_j\\ T_c & \omega>\omega_j\end{cases} \tag{14}$$

当 j,l_1,l_2 取固定值时，T-ω 函数图像示例如图 6 所示。当 $\omega<\omega_j$ 时，Stencil 计算在处理器上的性能处于 DMA 瓶颈，可考虑通过增加时间分块参数来减少 DMA 时间；当 $\omega>\omega_j$ 时，Stencil 计算性能处于计算瓶颈，可通过向量化提升计算性能。

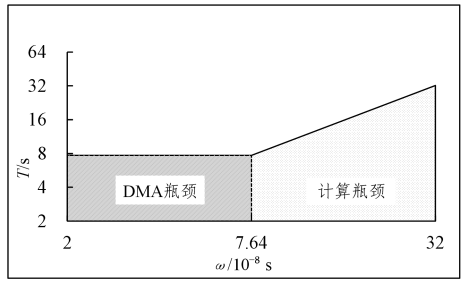


图 6 T-ω 函数

Fig. 6 T-ω function plots

4.4 示例分析

选取热传导计算中常用的 3D-7P Stencil 和 3D-27P Stencil 作为待优化算例。元素类型为双精度浮点，大小为 8B，数据规模为 512×512×512，时间迭代次数 $q=48$。表 2 列出了 3D-7P Stencil 和 3D-27P Stencil 在 SW26010 处理器运算核心上实测的 ω 和 ω_S 大小。

表 2 3D-7P 和 3D-27P 的 ω 和 ω_S

Table 2 ω and ω_S of 3D-7P and 3D-27P

Stencil 类型	$\omega(10^{-8}\ s)$	$\omega_S(10^{-8}\ s)$
3D-7P	3.88	1.08
3D-27P	14.31	5.03

根据硬件参数、Stencil 参数和分块参数可以得到 SW26010 处理器的单个核组上三维 Stencil 计算的执行时间函数。

以时间分块参数 $j=1$ 为例，选取空间分块参数 $l_1=128$，$l_2=8$。连续访存的 DMA 传输粒度为 128×8 Byte=1 024 Byte，根据 $p=64$ 时 DMA 带宽与 DMA 连续传输粒度的对应关系[23]，此时 64 个运算核心 DMA 总带宽约为 17GB/s，平均每个运算核心 DMA 带宽 $\alpha(l_1)=0.266$GB/s。将以上参数分别代入式(12)和式(13)计算 T_1-ω 函数，可得 $T_{DMA}=7.68$ s，$T_c=1.005\times10^8\omega$，$T=\mathrm{MAX}\{T_{DMA},T_C\}$。令 $T_{DMA}=T_c$，得到 $\omega_1=7.64\times10^{-8}$ s。

当 $j=2$ 时，令 $l_1=128,l_2=4$。此时 DMA 连续传输粒度仍为 1024B，DMA 带宽与 $j=1$ 时相同。代入参数计算 T_2-ω 函数，得到 $T_{DMA}=6.23$ s，$T_c=1.265\times10^8\omega$，$T=\mathrm{MAX}\{T_{DMA},T_c\}$，$\omega_2=4.925\times10^{-8}$ s。

时间分块参数 j 由 1 增加至 2，DMA 次数减半，但需要传输的冗余邻居元素增加，DMA 时间仅减少了约 1/5。在计算方面，j 值的增加导致冗余计算量增加，T_c-ω 函数的系数变大。

继续增加 j 的值可得到新的 T_j-ω 函数，但 DMA 时间不会一直保持下降趋势。首先，随着 j 值的增加，DMA 连续传输粒度减小，可能导致 DMA 带宽下降。其次，j 值增加后冗余的数据传输和计算的开销可能掩盖传输次数减少所带来的性能提升。

5 自适应分块参数算法

性能分析模型通过性能预测和瓶颈判断，为分块参数的优化方向提供了指导。基于性能分析模型，本节提出了自适应分块参数算法，给出了模型预测最优性能下的分块参数。

根据式(2)、式(3)、式(8)，空间分块参数 l_1 和 l_2 的大小会影响 DMA 带宽和数据传输量，最终影响 DMA 性能。在 SW26010 处理器上，DMA 带宽在总体趋势上与 DMA 连续传输粒度呈正相关[23]。从 DMA 带宽的角度考虑，在性能优化过程中，确定空间分块参数的方法是令 l_1 为满足式(4)、式(5)的最大整数，l_2 的值随之确定。

当三维 Stencil 计算的性能处于 DMA 瓶颈时，考虑通过增加时间分块参数 j 来优化 DMA 传输过程。j 值增加后，DMA 次数减少，但冗余的数据传输量和计算量随之增加，需要通过计算执行时间函数来判断性能是否有提升。根据式(4)、式(5)，当时间分块参数 j 改变时，空间分块参数 l_1 和 l_2 随之变化。

图 7 给出了 $j=1$ 和 $j=2$ 时 T-ω 函数变化的一个例子。若 Stencil 计算的 ω 值在区域①范围内，时间分块参数 j 从 1 增加到 2 后，程序执行时间减少，程序性能仍处于 DMA 瓶颈。若 ω 属于区域②，j 值增加后程序执行时间减少，性能瓶颈从 DMA 瓶颈转化为计算瓶颈。若 ω 属于区域③，j 值增加将导致程序执行时间增加。对于给定的三维 Stencil 算例，j 值增加对 DMA 瓶颈是否有优化效果需要根据具体的参数计算来判断。

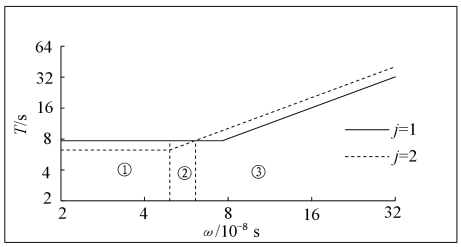


图 7　$j=1,2$ 时的 T-ω 函数

Fig. 7　T-ω function plots when $j=1,2$

综合以上分析过程，给出选取最优分块参数的思路：给定算例及 ω 值，根据性能分析模型可判断其性能瓶颈。若程序性能处于 DMA 瓶颈，可通过增加时间分块参数 j 来优化 DMA 传输，并根据模型预测性能是否有提升；若程序性能处于计算瓶颈，可通过向量化来优化计算性能。时间分块参数 j 改变时，空间分块参数 l_1 取满足式(4)、式(5)的最大整数，l_2 的值由 l_1 确定。当 Stencil 计算性能处于计算瓶颈(已向量化)，或增加 j 值无法再优化 DMA 传输时，程序性能达到预测最优。

算法 2 给出了自适应分块参数算法。输入为硬件参数、Stencil 计算参数及 ω 值，输出为预测最小执行时间及对应的分块参数。

算法 2　自适应最优分块参数算法

input：p,C,X,Y,Z,q,b,ω,ω_s

output：j,l_1,l_2,$T_j(\omega')$,ω'

1. j=1
2. $\omega'=\omega$
3. compute l_1, l_2, T_1-ω, ω_1
4. While 1 do
5.　if $\omega'<\omega_j$ then
6.　　j=j+1
7.　　compute l_1, l_2, T_j-ω, ω_j
8.　　if $T_j(\omega') < T_{j-1}(\omega')$ then
9.　　continue
10.　　else
11.　　j=j−1
12.　　break
13.　　endif
14.　else
15.　　if $\omega'>\omega_s$ then
16.　　$\omega'=\omega_s$
17.　　else
18.　　break
19.　　endif
20.　endif
21. endwhile

T_j-ω 表示根据当前 j 值得到的执行时间函数，$T_j(\omega')$ 表示 ω 取值为 ω' 时的预测执行时间，ω_j 表示 $T_c=T_{DMA}$ 时 ω 的取值。

算法流程如下：

Step 1(初始化)　令时间分块参数 $j=1$(即不进行时间分块)，计算性能分析模型函数 T_1-ω 和 ω_1。ω' 取未向量化时的 ω 值。

Step 2　通过比较 ω' 和 ω_j 的大小来判断程序当前的性能瓶颈。若程序性能处于 DMA 瓶颈，进入 Step 3，否则进入 Step 4。

Step 3　时间分块参数 j 加 1，重新计算 l_1，l_2，T_j-ω 和 ω_j，比较性能的变化。若性能有提升，回到 Step 2，否则 j 值减 1，当前 DMA 瓶颈无法得到优化，优化过程结束。

Step 4　若程序性能处于计算瓶颈且未向量化，对程序实施向量化，回到 Step 2；若程序已向量化，判断程序在当前优化方法下已达到最优性能，优化过程结束。

算法输出预测程序最小执行时间以及对应的时间和空间分块参数。另外，输出的 ω' 为 ω 或 ω_S，表示程序是否需要向量化。

下面分别对 4.4 节中的两个算例应用自适应分块参数算法。

对于 3D-7P Stencil，$j=1$ 时，首先计算 l_1，l_2，T_2-ω 和 ω_1，比较 3D-7P Stenicl 的 ω 与 ω_1 得到 $\omega<\omega_1$，表明 $j=1$ 时 3D-7P Stencil 性能处于 DMA 瓶颈。令 $j=2$，重新计算 l_1，l_2，T_2-ω 和 ω_2，将 ω 代入执行时间函数可得 $T_2(\omega< T_1(\omega)$，程序执行时间减少，继续向下优化，判断 $\omega<\omega_2$，表示 $j=2$ 时 3D-7P Stencil 性能仍处于 DMA 瓶颈。令 $j=3$，计算 $T_3(\omega)$ 得到 $T_3(\omega)> T_2(\omega)$，程序执行时间增加，说明 $j=3$ 时增加的冗余数据传输量和计算量掩盖了性能的提升。根据算法 2，恢复 j 的值为 2，结束优化过程。最终输出性能模型预测 3D-7P Stencil 在 SW26010 单核组上的最小执行时间 $T=6.23$ s，对应的分块参数为 $j=2$，$l_1=128$，$l_2=4$。此时 3D-7P Stencil 性能处于 DMA 瓶颈，向量化无收益。

3D-27P Stencil 优化过程类似于 3D-7P Stencil，$j=1$ 时判断 3D-27P Stencil 性能处于计算瓶颈，首先通过向量化提升计算性能，代入 ω_S，由 $\omega_S<\omega_1$ 判断向量化之后 3D-27P

Stencil 性能处于 DMA 瓶颈。令 $j=2$，计算 l_1, l_2, T_2-ω 和 ω_2。将其代入 ω_S 得到 $T_2(\omega_S)<T_1(\omega_S)$，程序执行时间减少。继续向下优化，由 $\omega_S>\omega_2$ 可知，$j=2$ 时 3D-27P Stencil 处于计算瓶颈，说明程序在向量化的情况下已充分发挥硬件的计算性能，结束优化过程。最终输出模型预测经向量化后的 3D-27P Stencil 在 SW26010 单核组上的最小执行时间 $T=5.74$ s。对应的分块参数为 $j=2, l_1=128, l_2=4$。此时 3D-27P Stencil 性能处于计算瓶颈。

图 8 给出了 $j=1$ 和 $j=2$ 时的 T-ω 函数图，直观反映了 3D-7P Stencil，3D-27P Stencil 以及经过向量化之后的 3D-27P Stencil 在 SW26010 处理器单核组上的预测执行时间。

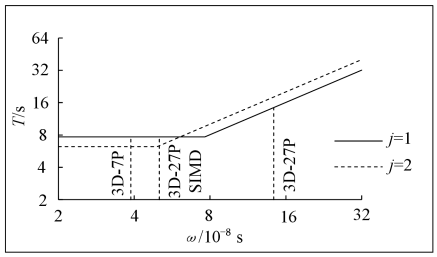


图 8 $j=1,2$ 时 3D-7P 和 3D-27P 的预测执行时间

Fig. 8 Performance of 3D-7P and 3D-27P predicted by model when $j=1$ and 2

6 实验

本节在 SW26010 处理器单核组上进行 3D-7P Stencil 和 3D-27P Stencil 的分块优化实验，首先根据第 5 节中两个算例的分块参数优化过程，比较实验结果和理论预测值，验证性能分析模型和自适应分块参数算法的合理性。然后测试了不同分块参数下的 DMA 时间和计算时间，通过比较理论和实际最优分块参数，验证了自适应分块参数算法的有效性。

6.1 3D-7P

初始分块参数 $j=1, l_1=128, l_2=8$，实际测得 $T_{DMA}=7.42$ s，$T_c=3.90$ s，$T=7.42$ s，程序性能处于 DMA 瓶颈。性能分析模型给出 $j=1$ 时 3D-7P Stencil 的预测执行时间为 7.68 s，预测性能瓶颈为 DMA 瓶颈。$j=1$ 时 3D-7P Stencil 的实际执行时间与性能优化模型中给出的预测执行时间非常接近。

令 $j=2, l_1=128, l_2=4$，测得 $T_{DMA}=5.04$ s，$T_c=4.42$ s，$T=5.04$ s，此时性能仍处于 DMA 瓶颈。性能分析模型给出 $j=2$ 时预测运行时间为 6.23 s，预测性能瓶颈为 DMA 瓶颈。预测 DMA 时间大于实际值，分析误差原因：当 j 增加时，需要传输的边界邻居层数增加，DMA 的连续传输粒度略有增加，带宽略有上升；l_2 减小使得跨步次数减少，相应的跨步开销减少。性能优化模型中为简化 DMA 性能计算忽略了边界邻居层数增加对带宽的影响以及跨步开销，因此预测 DMA 时间比实际运行时间要长一些。

为进一步优化 DMA 传输，令 $j=3$，测得 $T_{DMA}=4.93$ s，$T_c=7.92$ s，$T=7.92$ s，说明 $j=3$ 时冗余计算量过多导致计算时间急剧增加，程序性能下降，这与性能模型预测的结果相符。

6.2 3D-27P

对于 3D-27P Stencil，$j=1, l_1=128, l_2=8$ 时测得 $T_{DMA}=7.39$ s，$T_c=14.37$ s，$T=14.37$ s，此时性能处于计算瓶颈。经向量化后 $T_{DMA}=7.39$ s，$T_c=4.55$ s，此时性能处于 DMA 瓶颈，程序执行时间与模型给出的预测值接近。

令 $j=2$，测得向量化后 $T_{DMA}=4.98$ s，$T_c=5.74$ s，$T=5.74$ s。$j=2$ 时性能处于计算瓶颈，不需要优化 DMA 传输。3D-27P 在实验中的分块参数优化过程与算法给出的优化过程相同，预测执行时间也较为接近。实验结果表明，T_{DMA} 易受带宽波动影响，T_c 值较稳定且与预测值非常接近。

图 9 给出了实测数据和性能分析模型函数的对比。根据 3D-7P Stencil 和 3D-27P Stencil 实际执行时间和预测执行时间的差值，可计算出性能分析模型的平均预测误差约为 7.2%。

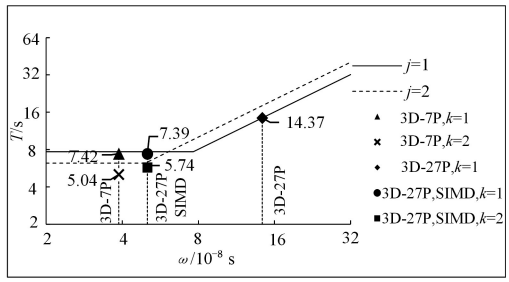


图 9 3D-7P 和 3D-27P 的实测性能

Fig. 9 Performance of 3D-7P and 3D-27P on SW26010

6.3 性能优化效果

图 10 给出了 3D-7P Stencil 和 3D-27P Stencil 在 SW26010 处理器单核组上的性能优化结果。横坐标 $j=1$，SIMD 和 $j=2$ 的数据缩小了纵坐标比例，实际值需要将对应纵坐标乘以 1/10。横坐标中 MPE 对应程序在 MPE 上的串行运行时间。经过 2.5D 空间分块并在运算核心上并行后，两个算例的性能加速比分别为 23.3 和 43.5。进一步向量化之后加速比分别为 23.3 和 82.6，其中 3D-7P Stencil 由于性能处于 DMA 瓶颈，向量化无加速效果。根据自适应分块参数算法，对两个算例应用时间分块，参数为 $j=2$。与串行结果相比，其加速比分别达到了 34.3 和 106.3；与已向量化但未进行时间分块的结果相比，自适应分块参数算法对两个算例的加速比分别为 1.47 和 1.29。

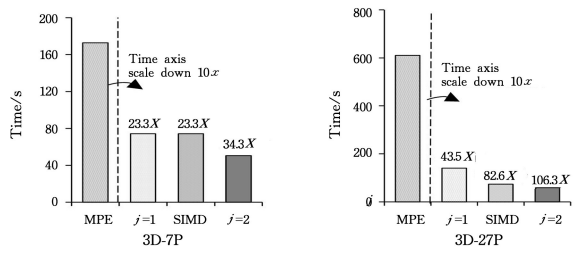
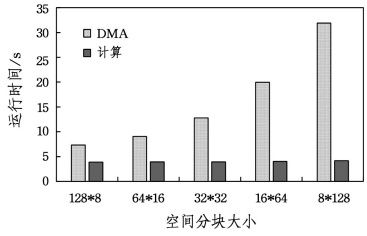


图 10 单核组的加速效果

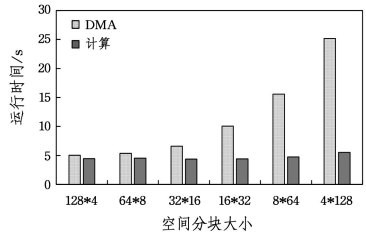
Fig. 10 Speedup on single core-group

为了验证自适应分块参数算法的优化效果，测试不同时间和空间分块参数下 3D-7P Stencil 和 3D-27P Stencil 的执行时间。时间分块参数 j 分别取 1 和 2，保持 $l_1 \times l_2$ 的大小不变，逐渐减小 l_1 的值，测出相应的 DMA 时间和计算时间。

由图 11(a)可直观看出，当 $j=1$ 时，3D-7P Stencil 的计算时间完全被 DMA 时间隐藏，随着 l_1 的减小，DMA 时间增加，程序性能下降且一直处于 DMA 瓶颈。图 11(b)给出了 $j=2$ 时 3D-7P Stencil 在不同空间分块下的性能。与 $j=1$ 时相似，程序一直处于 DMA 瓶颈，随着 l_1 的减小，DMA 时间不断增加。$j=2$ 时最优空间分块参数为 $l_1=128$，$l_2=4$，这与算法 2 给出的最优空间分块参数相符。



(a)$j=1$，3D-7P

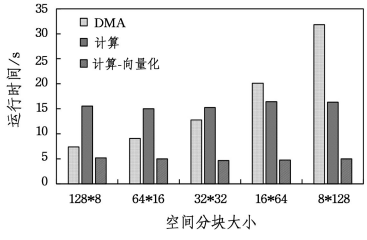


(b)$j=2$，3D-7P

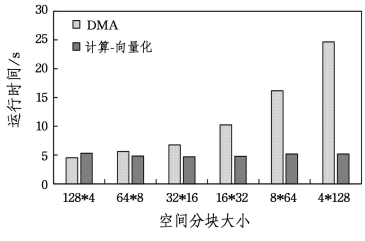
图 11　3D-7P 在不同分块参数下的性能

Fig. 11　Performance of different block sizes of 3D-7P

对于 3D-27P，如图 12(a)所示，当 $j=1$，$l_1=128$，$l_2=8$ 且未进行向量化时，程序性能处于计算瓶颈。随着 l_1 的减小，DMA 时间增加，当 $l_1=16$ 时，程序性能转化为 DMA 瓶颈。针对 3D-27P Stencil 的计算瓶颈，对其实施向量化之后，计算时间约为原计算时间的 1/3，程序性能处于 DMA 瓶颈。当空间分块参数取 $l_1=128$，$l_2=8$ 时程序性能最优。



(a)$j=1$，3D-27P



(b)$j=2$，3D-27P

图 12　3D-27P 在不同分块参数下的性能

Fig. 12　Performance of different block sizes of 3D-27P

在向量化的基础上，针对 $j=1$ 时 3D-27P 性能的 DMA 瓶颈，继续增大 j 值以优化 DMA 传输。$j=2$ 时测出 3D-27P Stencil 的 DMA 时间和计算时间如图 12(b)所示。当 $j=2$，$l_1=128$，$l_2=4$ 时，3D-27P Stencil 性能突破了 $j=1$ 时的 DMA 瓶颈，转化为计算瓶颈。当 l_1 逐渐减小时，程序性能从计算瓶颈转为 DMA 瓶颈且 DMA 时间不断增加。$j=2$ 时最优空间分块参数为 $l_1=128$，$l_2=4$，这与算法 2 给出的最优空间分块参数相符。

从不同分块参数的实验数据可以看出，影响 DMA 时间的主要因素是 DMA 连续传输粒度，即 l_1 减小会造成 DMA 时间增加。在 SW26010 处理器上，当 DMA 连续传输粒度较小时，DMA 带宽受传输粒度影响较大。另外，减小 l_2 的值可减少 DMA 跨步次数和 DMA 跨步开销。因此在自适应最优分块参数算法中，选择尽量大的 DMA 连续传输粒度是合理的。

本节的实验结果验证了自适应分块参数算法和性能分析模型的有效性，自适应分块参数算法依赖于性能分析模型的准确度，性能分析模型在对 DMA 性能进行建模时主要考虑 DMA 连续传输粒度对 DMA 带宽的影响，忽略了 DMA 跨步开销以及 DMA 对齐等因素，因此在 DMA 时间预测上略有误差。

结束语　对于给定类型和规模的三维 Stencil 算例，如何设置最优的时间和空间分块参数是三维 Stencil 在 SW26010 处理器上优化的重点。为减少实际优化中的手工调优工作，本文提出了面向 SW26010 处理器的三维 Stencil 自适应分块参数算法，并针对分块方法建立了性能分析模型。通过手工优化实验验证了所提算法和性能分析模型的有效性。未来可以将其应用到三维 Stencil 在 SW26010 处理器上的自动代码生成和调优工作中。

本文的自适应分块参数算法也具有一定的局限性。首先，性能分析模型中对 DMA 性能的建模忽略了部分细节，在 DMA 时间预测上有一些误差，在以后的工作中可以进一步完善对 DMA 性能的建模。其次，若三维 Stencil 的邻居元素在空间中不够集中，例如 3D-25P，分块所需缓冲区个数较多，分块参数优化的空间有限。另外，在并行层次方面，本文主要研究三维 Stencil 在 SW26010 处理器单核组上的线程级并行，在未来的工作中可以考虑进程级并行，并利用运算核心间的寄存器通信实现对三维 Stencil 的进一步优化和性能分析。

参 考 文 献

[1] RIVERA G，TSENG C W. Tiling optimizations for 3D scientific computations [C]// Proceedings of the 2000 ACM/IEEE Conference on Supercomputing(SC'00). Piscataway：IEEE，2000：32.

[2] YUAN L，ZHANG Y，GUO P，et al. Tessellating stencils [C]// Proceedings of the International Conference for High Performance Computing，Networking，Storage and Analysis. New York：ACM，2017：1-13.

[3] WIJESINGHE T，SENEVIRATHNE K，SIRIWARDHANA C，et al. Parameterized Diamond Tiling for Parallelizing stencil computations [C]// 2017 Moratuwa Engineering Research Conference (MERCon). Piscataway：IEEE，2017：99-104.

[4] GUO P，YUAN L，ZHANG Y，et al. Parallel Stencil Algorithm Based on Tessellating [J]. Journal of Frontiers of Computer Science and Technology，2019，13(2)：181-194.

[5] MURANUSHI T，MAKINO J. Optimal Temporal Blocking for

Stencil Computation [J]. Procedia Computer Science, 2015, 51: 1303-1312.

[6] WELLEIN G, HAGER G, ZEISER T, et al. Efficient temporal blocking for stencil computations by multicore-aware wavefront parallelization [C]// 2009 33rd Annual IEEE International Computer Software and Applications Conference. Piscataway: IEEE, 2009: 579-586.

[7] FRIGO M, STRUMPEN V. Cache oblivious stencil computations [C]// Proceedings of the 19th Annual International Conference on Supercomputing. New York: ACM, 2005: 361-366.

[8] KRISHNAMOORTHY S, BASKARAN M, BONDHUGULA U, et al. Effective automatic parallelization of stencil computations [C]// Programming Language Design and Implementation. New York: ACM, 2007: 235-244.

[9] NGUYEN A, SATISH N, CHHUGANI J, et al. 3. 5-D blocking optimization for stencil computations on modern CPUs and GPUs [C]// Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10). Piscataway: IEEE, 2010: 1-13.

[10] RAWAT P S, HONG C, RAVISHANKAR M, et al. Effective resource management for enhancing performance of 2D and 3D stencils on GPUs [C]// Proceedings of the 9th Annual Workshop on General Purpose Processing Using Graphics Processing Unit (GPGPU '16). New York: ACM, 2016: 92-102.

[11] SZUSTAK L, ROJEK K, OLAS T, et al. Adaptation of MPDATA Heterogeneous Stencil Computation to Intel Xeon Phi Coprocessor [J]. Scientific Programming, 2015(5): 1-14.

[12] WILLIAMS S, SHALF J, OLIKER L, et al. Scientific computing kernels on the cell processor [J]. International Journal of Parallel Programming, 2007, 35(3): 263-298.

[13] AO Y, YANG C, WANG X, et al. 26 pflops stencil computations for atmospheric modeling on sunway taihulight [C]// 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS). Piscataway: IEEE, 2017: 535-544.

[14] MENG J, SKADRON K. Performance modeling and automatic ghost zone optimization for iterative stencil loops on GPUs [C]// International Conference on Supercomputing. New York: ACM, 2009: 256-265.

[15] RAWAT P S, VAIDYA M, SUKUMARAN-RAJAM A, et al. Domain-Specific Optimization and Generation of High-Performance GPU Code for Stencil Computations [J]. Proceedings of the IEEE, 2018, 106(11): 1902-1920 .

[16] DATTA K, MURPHY M, VOLKOV V, et al. Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures [C]// Proceedings of the 2008 ACM / IEEE Conference on Supercomputing (SC'08). Piscataway: IEEE, 2008: 1-12.

[17] MATSUMURA K, ZOHOURI H R, WAHIB M, et al. AN5D: automated stencil framework for high-degree temporal blocking on GPUs [C]// Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization. New York: ACM, 2020 .

[18] UNAT D, CAI X, BADEN S B. Mint: realizing CUDA performance in 3D stencil methods with annotated C [C]// International Conference on Supercomputing. New York: ACM, 2011: 214-224.

[19] BONDHUGULA U, BANDISHTI V, PANANILATH I. Diamond Tiling: Tiling Techniques to Maximize Parallelism for Stencil Computations [J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(5): 1285-1298.

[20] WU Q, NI Y, HUANG X. Regional Ocean Model Parallel Optimization in "Sunway TaihuLight" [J]. Journal of Computer Research and Development, 2019, 56(7): 1556-1566.

[21] LV X J, LIU Z, CHU X S, et al. Extreme-scale Simulation Based LBM Computing Fluid Dynamics Simulations [J]. Computer Science, 2020, 47(4): 13-17.

[22] SAIDI S, TENDULKAR P, LEPLEY T, et al. Optimizing two-dimensional DMA transfers for scratchpad Based MPSoCs platforms [J]. Microprocessors Microsystems, 2013, 37(8): 848-857.

[23] XU Z, LIN J, MATSUOKA S. Benchmarking sw26010 many-core processor [C]// 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). Piscataway: IEEE, 2017: 743-752.

ZHU Yu, born in 1998, postgraduate. Her main research interests include high-performance computing and so on.

XU Jin-long, born in 1985, Ph.D, lecturer. His main research interests include high-performance computing and computer architecture.