非结构网格下稀疏下三角方程求解器众核优化技术研究

倪 鸿 刘 鑫
(国家并行计算机工程技术研究中心 北京 100086)

摘 要 稀疏下三角方程求解器(SpTRSV)作为基础线性代数库中一个重要的算法,在大规模科学计算中有着广泛应用。在非结构网格中,由于非结构网格具有数据存储无序性、数据强相关性以及频繁地离散访存等特点,该算法在众核架构上难以实现有效的并行。文中基于国产异构众核处理器 SW26010 体系结构的特点,针对非结构网格计算,提出了一种基于流水线串行-局部并行思想的通用众核优化方法。该方法能够有效减少非结构网格计算中的随机访存,提高计算效率,并且具有很好的扩展性。基于该算法对多个实际应用算例进行众核优化,实验结果表明:该方法能够实现单核组 3 倍以上的加速,显著降低了运行时间。

关键词 稀疏下三角方程求解器,非结构网格,SW26010,异构众核优化,并行算法
中图法分类号 TP311 **文献标识码** A

Many-core Optimization for Sparse Triangular Solver Under Unstructured Grids

NI Hong LIU Xin
(National Research Centre of Parallel Computer Engineering and Technology, Beijing 100086, China)

Abstract Sparse Triangular Solver (SpTRSV), as an important algorithm in basic linear algebraic library, has been widely used in large-scale scientific computing. In unstructured-grids, because unstructured grid have the characte-ristics of data storage disorder, data depth correlation and frequent discrete-time memory access, this algorithm is difficult to achieve effective parallelism in the many-core architecture. In this paper, based on the architecture of the domestic heterogeneous multiprocessor SW26010 architecture, a general kernel optimization method based on pipelined serial and local parallel was proposed for unstructured grid computing. This method can effectively reduce random access in unstructured grid computing, improve the computing efficiency, and have the good scalability. Based on this algorithm, multiple kernel optimization is carried out for several practical applications. The experimental results show that the method can achieve more than 3 times acceleration of the single core group and significantly reduce the running time.

Keywords SpTRSV, Unstructured-grids, SW26010, Heterogeneous many-core Optimization, Parallel algorithm

1 概述

稀疏下三角方程求解器[1](Sparse Triangular Solver, SpTRSV)在大规模科学计算中有广泛的应用,特别是在稀疏线性方程组中,常被用作直接法进行 LU[2] 分解或者迭代法求解预条件子[3]。无论是直接法还是迭代法的稀疏方程组求解器,都广泛应用于数值模拟[4]、计算流体力学[5]和人工智能[6]等很多领域。

由于数据相关性和离散访存等固有特点,SpTRSV 具有局部性差、写冲突、离散访存、并发度低等问题,其众核优化一直以来都是一个比较棘手的问题。在非结构网格领域,由于非结构网格带来的数据存储无序性,以及更严重的计算依赖,其 SpTRSV 并行优化更为复杂困难。

目前,大多数研究工作主要集中在如何挖掘并发度及提高数据局部性上。在提高并行度问题上,主要是按照数据彼此之间是否有数据相关性分层[7-10],每一层可以并发操作,层与层之间串行执行,但若分层数过多,这种方法带来的同步开销往往非常高昂,经常占据求解的大部分时间,成为了 SpTRSV 并行问题最严重的性能瓶颈之一。

在提高数据局部性方面,也有很多人的工作,例如 Picciau 等[11]在 GPU 上将矩阵划分为若干子矩阵,意图通过优化全局调度提高共享缓存的使用效率。但是提高数据局部性不可避免要忽略 SpTRSV 中的层次信息,二者不可兼得。

SW26010 是国产超级计算机神威·太湖之光上的核心处理器。在 SW26010 平台上,由于处理器架构特点,SpTRSV 的并行优化更具有挑战性。敖玉龙[12]提出了分块着色并行方法,其主要思想是强行断开某些数据依赖关系,在提高并行度与降低收敛速度之间取得一个平衡,王欣亮[13]则设计了一种 swSpTRSV 算法,该算法适用于申威架构的稀疏层次块布局,利用生产-消费配对计算方法解决。但是这些方法比较适用于结构网格,对于非结构网格问题,由于数据之间的依赖关系很强,而且大量数据并不在矩阵对角块中,因此直

本文受"全球变化和应对"专项(2016YFA0602200)资助。

倪 鸿(1989-),男,硕士生,工程师,主要研究方向为并行算法与应用,E-mail:837896165@qqcom;刘 鑫(1979-),女,博士,副研究员,CCF 会员,主要研究方向为并行算法与应用。

接求解非结构网格矩阵往往会导致不收敛或者优化性能较差。

为了设计一套行之有效的通用方案,解决如何在国产异构众核处理器 SW26010 平台上实现非结构网格下的 SpTRSV 众核优化问题,本文提出了一种基于分块思想的并行算法。该算法完全摒弃了层次划分和同步思想,通过流水线作业避免了数据分层重排时间产生的额外开销,通过严格的指令序列实现局部并行,通过寄存器通信机制实现众核全局通信,极大地提高了数据复用性。

本文主要工作如下:

1)提出了一种新的 SpTRSV 并行优化策略,该策略通用性强,适用于非结构网格下的稀疏矩阵。

2)基于寄存器通信机制设计了一种众核通信框架,可以实现任意众核之间的高效通信。

3)测试了典型的 6 组不同规模的非结构网格 SpTRSV 问题,该方法能够得到平均 3 倍以上的加速。

本文第 2 节对国产异构众核处理器 SW26010 架构进行介绍;第 3 节对 SpTRSV 并行化进行分析;第 4 节针对 SW26010 平台提出新的 SpTRSV 优化策略;第 5 节基于 OpenFOAM 软件对实验结果进行分析和比对;最后总结全文。

2　概述

"SW26010"[14]异构众核处理器广泛应用在"神威・太湖之光"上,具体结构如图 2 所示。该处理器集成了 4 个运算核组,共 260 个运算核心,每个处理器配置 32GB 内存,平均分配给 4 个运算核组,每个核组包括 1 个运算控制核心(简称主核)和 1 个运算核心阵列(简称从核阵列),从核阵列按照 8×8 模式划分。

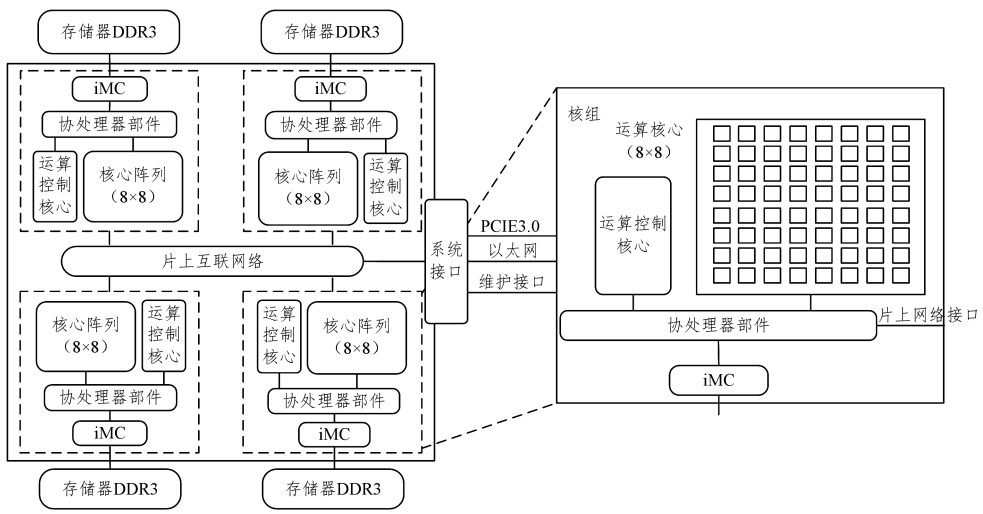

图 1　"SW26010"异构众核处理器架构图

每一个从核拥有 16 kB 一级指令缓存和 64 kB 局部存储空间(LDM)。从核可以直接通过 Gload/Gstore 指令对主存进行访问,也可以通过 DMA(Direct Memory Access)的方式进行批量数据传输。在从核阵列之间,有一个低延迟寄存器通信网络支持同行或者同列的从核相互直接通信。

3　SpTRSV 并行优化分析

在 LU 分解中,需要求解一个稀疏线性方程组 $\boldsymbol{L}x=b$,其中 $\boldsymbol{L}$ 是稀疏下三角矩阵,$\boldsymbol{b}$ 是右端向量,$\boldsymbol{x}$ 是解向量。图 3 列出了一个包含 5 个未知数的稀疏下三角方程,左侧为对应的矩阵形式,右侧为对应的方程形式和相应的解。

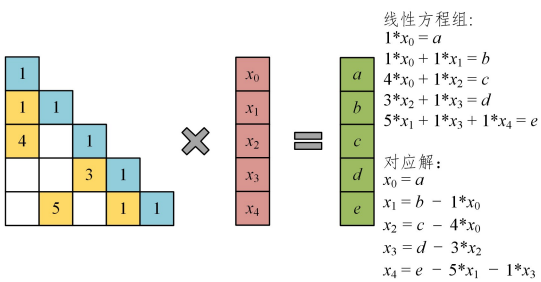

图 2 稀疏线性方程组求解示意图

由图 2 可知,解向量 $\boldsymbol{x}$ 中任意一个元素 x_i 的求解,都有可能依赖于前面的元素求解完毕,例如 x_4 的最后结果依赖于 x_1 与 x_3 的值。在稀疏矩阵下,任意一个 x_i 值在计算前,需要将第 i 行非零元素对应列所在的 $x_{j1},x_{j2},\cdots,x_{jn}$ 先计算完毕。

因此,SpTRSV 问题是一个非常串行的操作,即使是如此串行的问题,经过仔细分析也可以发现:x_1 与 x_2 没有数据依赖性,在计算出 x_0 之后,可以并发求解 x_1 与 x_2,以此类推计算完 x_1 与 x_2 后,再并行求解 x_3 和 x_4。这样实际上就是将 $\boldsymbol{x}$ 向量按照之间是否具有相关性进行分层处理,属于同一层的未知数可以并行求解,层与层之间串行执行,为了保证计算的时序正确性,计算完每一层之后需要同步。

分层并行算法的主要缺陷在于需要对解向量 $\boldsymbol{x}$ 进行重排处理,稀疏矩阵也要进行相应处理才能参与计算,得到结果 $\boldsymbol{x}'$ 之后还需要转换成原来的解向量 $\boldsymbol{x}$,相当于两次重排序,这部分时间开销不可避免。

另外,该算法需要每一层具有足够的并行度,同步开销才能足够小,如果几乎每一个 $\boldsymbol{x}$ 向量中的元素 x_i 都依赖于前一个元素 x_{i-1},在稀疏矩阵上等价于每一层几乎没有空三角(我们称之为强依赖关系,反之为弱依赖关系),每一层只有若干个少量元素,那么这种算法的性能将极差,很难有效并行。

另一方面,在数据复用性上,该算法也有缺陷。假设 x 向量中的元素 x_i 依赖于 x_j,而 i 与 j 之间的距离很大,在稀疏矩阵上等价于大量非零元素在非对角块上(即 $i-j>>1$,我们称之为深依赖关系,反之为浅依赖关系),这种情况下该算法无法预先缓存,只能够细粒度离散访存。

非结构网格问题由于数据的无序性造成了稀疏矩阵非零元素无规律存储。经过实际算例测试,非结构网格下的数据

往往都具有强依赖关系和深依赖关系，现有的申威平台上SpTRSV优化算法在面对这种问题时也无法实现有效加速。

总结下来，非结构网格下SpTRSV主要面临的挑战有：

1)离散访存问题：矩阵离散宽度(即稀疏矩阵中非零元素值距离该行对角线的最大区间跨度)很大，约为网格规模的8%～20%[15]左右。同时，矩阵规模与下三角非零元素比大约仅在1∶3左右，非结构网格的矩阵特点造成离散访存问题非常突出。

2)数据并发度低：数据由于具有强依赖关系，通过分层后其并发度仍然很低，无法有效提高计算效率。

3)LDM存储受限：由于数据具有深依赖关系，对数据复用性提出了很高要求。但从核LDM空间受限于64 kB大小，这意味着无法存储大量冗余数据。

4)从核通信受限：从核之间可以通过寄存器通信，但寄存器通信功能单一，目前仅支持同行或者同列从核之间直接数据传递，通信缓冲区有限，难以满足任意从核之间的大规模通信。

4 申威平台下SpTRSV优化策略

通过进一步分析发现，SpTRSV串行算法的主要时间开销集中在离散访存部分，而非计算部分，因此如果能够减少访存时间，提高数据复用性，那么也将有很好的加速效果。

基于上述分析，本文基于申威平台提出了一种新的算法思路，即流水线串行-局部并行。该算法摒弃传统的分层并行的思想，改成流水线并行，不过分追求各从核并行效率而是尽可能提高各从核之间的数据通信及相互协作性，减少访存时间开销，从而实现有效加速。

4.1 流水线作业

为尽可能减少对数据的预处理，我们对解向量 x 按块平均划分。通过这种方式，我们将 x 向量中数据之间的依赖性转换成 x 向量块与块之间的依赖性，也就是说所有的 x 向量块都依赖于前面若干块计算完毕后才能开始计算，而当该 x 向量块计算完毕后，其块内完成求解的元素又需要提供给后继若干个 x 向量块。进一步考察稀疏下三角方程组，我们把 x 向量块内元素求解完毕称为完成一次更新操作，事实上，每一个 x 向量块完成一次更新操作过程，仅需要3个步骤：接收数据、自我更新、发送数据，求解的分解过程如图3所示。

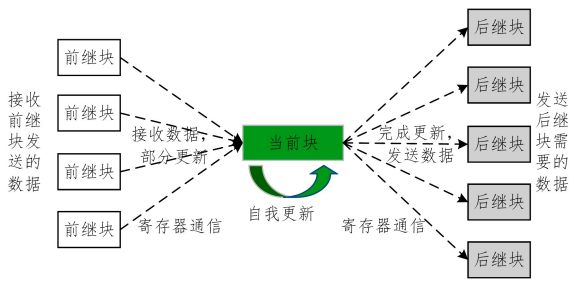

图3 流水线作业计算流程

Step1 其接收前面所依赖的 x 向量块内元素，每接收一个向量块数据，称为一次部分更新。

Step2 在接收完所有前继块内数据后，由于块内元素具有相互依赖性，对块内元素进行遍历计算，称为自我更新。

Step3 块内元素求解，所有值均已经是最新值。将已经求解完毕的 x 向量元素分别发送给相依赖的后继块，称之为完成更新。

重复Step1－Step3，每一个从核顺序计算一个 x 向量块，按块遍历整个 x 向量，即实现了整个流水线作业过程。与此同时，一旦 x 向量块收到前继块发送的数据，可以立即对块内部分元素进行更新，不同 x 向量块之间部分更新互不影响，因此真正的流水线串行部分其实是自我更新部分，而部分更新部分并行执行。其实现伪代码如算法1所示。

算法1 流水线作业-局部并行算法

```
1. for all pi,where 0≤i<p do
2.   block_num=pi;//初始化
3.   while(block_num<block_size)
4.     Download_block(block_num);//初始化
5.     Recv_data();//接收前继块数据
6.     Self_update();//自我更新
7.     Send_data();//发送给后继块
8.     block_num +=64;//处理下一个块
9.   endwhile
10. endfor
```

4.2 众核通信框架设计

实现优化算法的关键在于如何高效地在 $\boldsymbol{X}$ 向量块之间数据通信。在SW26010众核处理器中，从核之间的通信采用寄存器通信的方式，由于非结构网格稀疏矩阵的随机离散性，每个从核都有与任意一个其他从核通信的可能性，无法通过任何的任务分配方式规避。

一种方法是以某些核作为“路由”，帮助非同行同列从核进行数据传输，但是这将导致两个问题：1)随机通信引入了频繁的解包、判断等额外开销，从而极大削弱了寄存器通信本身高带宽低延迟的优势；2)产生死锁问题，对于大量的随机通信来说，从核很大概率会产生通信环，那么将产生通信死锁问题。这一问题在林恒[16]、王欣亮[13]等人在有关神威太湖之光上的研究工作中有所阐述。

从另一个角度思考通信问题，既然随机通信是导致从核产生通信死锁的原因，那么能否将随机通信转化成一种有序通信？基于这种有序通信，所有从核能够在固定时刻做固定的操作。通过这种思想，我们设想每一个从核都有一串指令流，所有从核完全根据解析自身的指令流进行有序操作，从而省去额外通信、同步等开销时间，同时也解决了通信死锁问题。基于此，通用众核通信模型的思路如下：

1)基于时间顺序，创建指令序列，保证操作指令按照时间顺序严格执行。

2)基于指令序列，创建消息序列，保证指令所对应的消息处理按照指令顺序严格执行。

4.2.1 创建指令序列和消息序列

由于流水线操作，在同一时刻，只有一个 x 向量块完成更新操作后，才能对下一个 x 向量块进行更新操作，因此在同一时刻，只有一个从核处于数据发送状态，其他从核要么处于数据接收状态，要么处于计算状态。因此，在这种情况下，不会产生死锁问题。因此，每一个从核在同一时刻，只会处于3种状态：发送数据状态、接收数据状态、计算状态。

对于发送核来说，其发送数据按照如下顺序进行优先级排序：

1)发送与该发送核处于同行的 x 向量块中的数据，按照块编号递增顺序发送。

2)发送与该发送核既不处于同行也不处于同列的 x 向量块中的数据，按照块编号递增顺序发送。

3)发送与该发送核处于同列的 x 向量块中的数据，按照

块编号递增顺序发送。

4)对该发送核自身需要处理的 x 向量块数据进行处理。

对于接收核来说,其接收数据按照如下顺序进行优先级排序:

1)接收自身需要处理的 x 向量块数据。

2)转发不属于自身处理的 x 向量块数据。

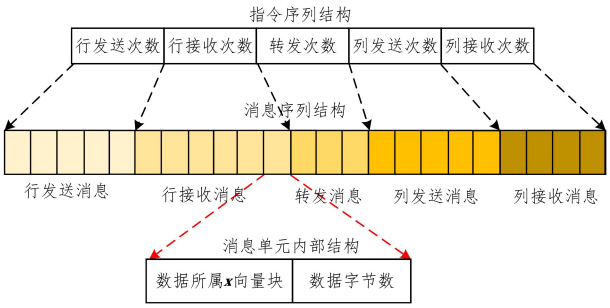

图 4　指令序列与消息序列的结构图

4.2.2　搭建众核通信框架

在同一时刻,只有一个从核处于发送状态,其他从核处于接收状态或者等待状态。所有从核根据自身读取的指令序列和消息序列,做出相应的操作。某一时刻,消息传递通信的过程如图 5 所示。

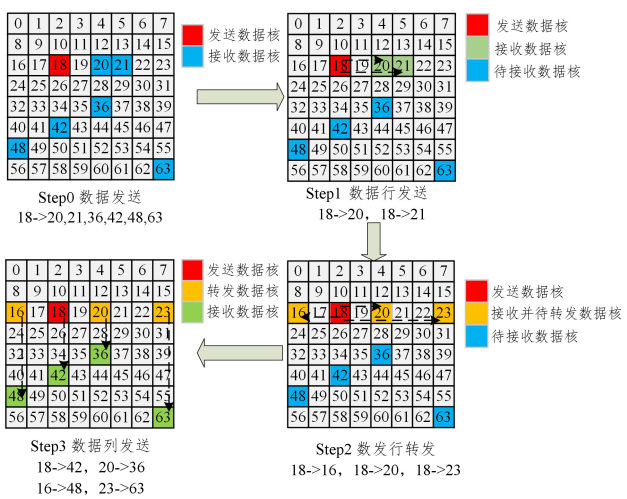

图 5　8 * 8 从核阵列消息传递示意图

其基本通信框架伪代码实现如算法 2 所示。

算法 2　众核通信框架算法

```
1. for all pi,where 0≤i<p do
2.    Read_instruct();//读取指令序列到 inst 数组
3.    Read_info();//读取消息序列到 info 数组
4.    for block ← 0 to block_size do
5.       flag=判断运行到哪一个块;
6.       if flag==发送核:
7.          size=inst[i].send_row_size;
8.          for(i=0;i<size;i++){
9.             putr(info[i].length,info[i].id);
10.         }
11.         size +=inst[i].send_col_size;
12.         for(;i<size;i++){
13.            putc(info[i].length,info[i].id);
14.         }
15.         size +=inst[i].self_handle_size;
16.         for(;i<size;i++){
17.            self_handle();
18.         }
19.      if flag==与发送核同一行:
20.         size=inst[i].recv_row_ size;
21.         for(i=0;i<size;i++){
22.            getr(info[i].length);
23.         }
24.      size +=inst[i].send_col_size;
25.      for(;i<size;i++){
26.         getr(info[i].length);
27.         putc(info[i].length,info[i].id);
28.      }
29.      else 其他:
30.      size=inst[i].recv_col_size;
31.      for(i=0;i<size;i++){
32.            getc(info[i].length);
33.         }
34.      ALLSYN;//硬件同步
35.   endfor
36. endfor
```

5　实验结果分析

在实验中,我们采用国产编译器 SW5CC 和 SW453++编译 C 和 C++程序。为验证该算法,我们选取 OpenFOAM 求解器公用基础件,对远景能源公司不同网格规模的真实算例核心段进行实验,该算例主要是通过对复杂地形的风场数值模拟,进行风电机组系统动力学模型和关键零件优化,具体算例数据如表 1 所列。

表 1　不同网格规模算例数据

矩阵规模 (*Cells*)	非零元素个数 (*Faces*)	稀疏度 (*Cells*:*Faces*)	单核运行时间/ms
124026	364825	1:2.94	9.586
187643	555240	1:2.96	15.490
374354	1112843	1:2.97	33.936
480368	1429188	1:2.97	48.183
497940	1479619	1:2.97	49.826
1498499	4475730	1:2.98	148.264

5.1　优化过程

5.1.1　避免全局同步

在每一个 x 向量块的计算中,可能存在靠后的向量块比前面的块先完成更新的情况,需要通过全局同步保证每一 x 向量块严格且有序执行,而这带来了同步开销。避免全局同步的一种方式是发送核在完成更新操作后,发送信号启动下一个发送核。

5.1.2　改进 DMA 计算-访存掩盖

在通信过程中,同一行从核承担了大量数据转发操作。因此在发送核更新并完成发送数据后,并不立即转入下一个循环 DMA 重新载入新数据,而是继续执行转发指令,等待与之同行的所有从核完成了更新操作后统一进行 DMA 访存载入数据。

5.1.3　数据压缩

通过压缩数据减少 DMA 时间。通过对索引表、矩阵非零元素位置信息等数据进行数据压缩,减少从核大量数据 DMA 拷入时间。

5.1.4　数学等价变换

数学形式等价变换的时间开销通过计算-访存掩盖,变换后的数学形式相对原形式减少了访存次数和计算次数,从而

减少了流水线串行部分的时间。

5.1.5 缓冲机制

在 LDM 中设立缓冲区，从核根据当前指令对收到的数据进行处理，或者立即处理，或者放入缓冲区，等到下一个周期再处理，从而解决数据复用性问题。

5.2 性能分析

考察不同网格规模下的加速优化效果，如图 6、图 7 所示。可以看出，随着网格规模增加，加速效果基本保持稳定，保持 3 倍以上加速。

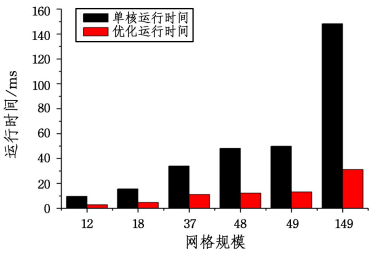

图 6 不同网格规模下优化性能比较

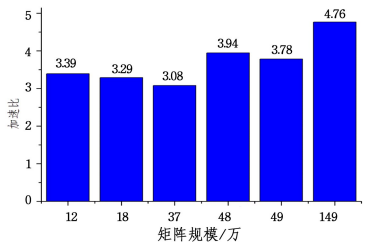

图 7 不同网格规模下的加速效果图

通过测试发现，在该算法中，其耗时最大部分在于算法 1 的自我更新部分(即 self_update 部分)，该部分时间占整个优化算法时间的 60%以上，约为单核运行时间的 1/5，而这部分只能串行完成。这充分说明了非结构网格的数据强相关性是限制该算法优化效果显著提高的根本原因。

5.3 可扩展性分析

由于各从核按块读取数据，可以通过调节块的大小来确定一次计算拷入从核的数据量。在上述实验中，LDM 空间平均利用率仅为 50%左右，因此有足够的空间建立 LDM 缓存区以应对离散数据随网格规模增加而增多的问题，理论计算表明：该算法可以支持 800 万以下网格众核优化，具有较好的可扩展性。

5.4 预处理时间代价

在优化算法过程中，需要对矩阵进行一次并行化预处理，这部分处理在整个计算过程中只需处理一次，在迭代过程中不改变数据存储格式，不涉及对数据的重排序。因此，相对于反复迭代过程来说，预处理产生的额外开销基本可以忽略不计。

5.5 负载平衡分析

通常，稀疏矩阵面临的一个问题是负载平衡问题，该算法很好地规避了负载平衡问题。在优化过程中，数据块(非矩阵块)平均划分给各个从核，从统计角度上看，各从核的计算任务量基本相当，从而解决了负载平衡问题。

结束语 本文针对非结构网格 SpTRSV 问题提出了一种新的优化算法。通过该方法，基于 SW26010 众核架构处理器平台，我们对不同规模的算例优化均得到单核组 3 倍以上加速。从算法原理上看，该优化方法适用于任意存储格式的稀疏矩阵，尤其适用于传统分层算法难以解决的低并发度、强相关性特征的矩阵。值得注意的是，该寄存器通信框架可以实现任意从核之间的通信，对通信频繁问题的解决有很好的参考意义，下一步研究方向将该算法应用于图搜索[16]等具有高度数据相关性等问题。

参 考 文 献

[1] ANDERSON E, SAAD Y. Solving sparse triangular linear systems on parallel computers[J]. International Journal of High Speed Computing, 1989, 1(1): 73-95.

[2] DUFF I S, ERISMAN A M, REID J K. Direct Methods for Sparse Matrices[J]. Matehematics of Computation, 1986, 52(185): 250.

[3] SAAD Y. Iterative methods for sparse linear systems[OL]. http://ieeexplore.ieee.org/ielx5/99/12132/001231631.pdf.

[4] REGULY I Z, MUDALIGE G R, BERTOLLI C, et al. Acceleration of a full-scale industrial cfd application with op2[J]. IEEE Transactionson Parallel and Distributed Systems, 2016, 27(5): 1265-1278.

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

[6] CHEN T, LI M, LI Y, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv preprint arXiv: 1512.01274, 2015.

[7] ANDERSON E, SAAD Y. Solving sparse triangular linear systems on parallel computers[J]. International Journal of High Speed Computing, 1989, 1(1): 73-95.

[8] KALIKAR S, NASRE R. Domlock: A new multi-granularity locking technique for hierarchies[C]// Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2016.

[9] SCHREIBER R, TANG W P. Vectorizing the conjugate gradient method[D]. Los Angeles: Stanford University, 1982.

[10] LIU W, LI A, HOGG J D, et al. Fast synchronization-free algorithms for parallel sparse triangular solves with multiple right-hand sides[J]. Concurrency and Computation: Practice and Experience, 2017, 29(21): e4244-n/a.

[11] PICCIAU A, INGGS G E, WICKERSON J, et al. Balancing locality and concurrency: Solving sparse triangular systems on gpus[C]// 2016 IEEE 23rd International Conference on High Performance Computing(HiPC). 2016: 183-192.

[12] 敖玉龙. 国产大型众核系统上稀疏矩阵和 Stencil 运算的性能优化关键技术研究[D]. 北京：中科院软件研究所，2017：27-47.

[13] 王欣亮. 关键稀疏数值计算核心在国产众核架构上的性能优化研究[D]. 北京：清华大学，2018：24-43.

[14] QI F B. Sunway TaihuLight super computer[J]. Communications of the CCF, 2017, 13(10): 16-22.

[15] ANDERSON W K, GROPP W D, KAUSHIK D K, et al. Achieving High Sustained Performance in an Unstructured Mesh CFD Application[C]// Conference on High Performance Computing (Super Computing). 1999.

[16] 林恒. 基于超大规模异构体系结构的图计算系统研究[D]. 北京：清华大学，2018：23-53.