软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(12):3921−3932 [doi: 10.13328/j.cnki.jos.005309]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向国产申威 26010 众核处理器的 SpMV 实现与优化*

刘芳芳[1,2], 杨 超[1,3,5], 袁欣辉[4], 吴长茂[1], 敖玉龙[1,2,5]

[1](中国科学院 软件研究所 并行软件与计算科学实验室,北京 100190)

[2](中国科学院大学,北京 100049)

[3](计算机科学国家重点实验室(中国科学院 软件研究所),北京 100190)

[4](国家并行计算机工程技术研究中心,北京 100190)

[5](北京大学 数学科学学院,北京 100871)

通讯作者: 杨超, E-mail: chao_yang@pku.edu.cn

摘 要: 世界首台峰值性能超过 100P 的超级计算机——神威太湖之光已经研制完成,该超级计算机采用了国产申威异构众核处理器,该处理器不同于现有的纯 CPU,CPU-MIC,CPU-GPU 架构,采用了主-从核架构,单处理器峰值计算能力为 3TFlops/s,访存带宽为 130GB/s.稀疏矩阵向量乘 SpMV(sparse matrix-vector multiplication)是科学与工程计算中的一个非常重要的核心函数,众所周知,其是带宽受限型的,且存在间接访存操作.国产申威处理器给稀疏矩阵向量乘的高效实现带来了很大的挑战.针对申威处理器提出了一种 CSR 格式 SpMV 操作的通用异构众核并行算法,该算法从任务划分、LDM 空间划分方面进行精细设计,提出了一套动静态 buffer 的缓存机制以提升向量 x 的访存命中率,提出了一套动静态的任务调度方法以实现负载均衡.另外还分析了该算法中影响 SpMV 性能的几个关键因素,并开展了自适应优化,进一步提升了性能.采用 Matrix Market 矩阵集中具有代表性的 16 个稀疏矩阵进行了测试,相比主核版最高有 10 倍左右的加速,平均加速比为 6.51.通过采用主核版 CSR 格式 SpMV 的访存量进行分析,测试矩阵最高可达该处理器实测带宽的 86%,平均可达到 47%.

关键词: 稀疏矩阵向量乘;SpMV;申威 26010 处理器;异构众核并行;自适应优化

中图法分类号: TP303

中文引用格式: 刘芳芳,杨超,袁欣辉,吴长茂,敖玉龙.面向国产申威 26010 众核处理器的 SpMV 实现与优化.软件学报,2018,29(12):3921−3932. http://www.jos.org.cn/1000-9825/5309.htm

英文引用格式: Liu FF, Yang C, Yuan XH, Wu CM, Ao YL. General SpMV implementation in many-core domestic sunway 26010 processor. Ruan Jian Xue Bao/Journal of Software, 2018,29(12):3921−3932 (in Chinese). http://www.jos.org.cn/1000-9825/5309.htm

General SpMV Implementation in Many-Core Domestic Sunway 26010 Processor

LIU Fang-Fang[1,2], YANG Chao[1,3,5], YUAN Xin-Hui[4], WU Chang-Mao[1], AO Yu-Long[1,2,5]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

[3](State Key Laboratory of Computer Science (Institute of Software, The Chinese Academy of Sciences), Beijing 100190, China)

[4](National Research Center of Parallel Computer Engineering and Technology, Beijing 100190, China)

[5](School of Mathematical Sciences, Peking University, Beijing 100871, China)

Abstract: The fastest supercomputer in the world—Sunway TaihuLight with performance of more than 100P has been released. It makes use of heterogeneous many-core processors which is different from the existing pure CPU, CPU-MIC, CPU-GPU architecture. Each

* 基金项目: 国家重点研发计划(2016YFB0200603); 国家自然科学基金(91530323)
Foundation item: National Key R&D Program (2016YFB0200603); National Natural Science Foundation of China (91530323)
收稿时间: 2017-01-11; 采用时间: 2017-05-01

processor has 4 core groups (CGs), with each including one management processing element (MPE) and one computing processing element (CPE) cluster of 64 CPEs. The peak performance of single processor is 3TFlops/s, the memory bandwidth is 130GB/s. Sparse matrix-vector multiplication is a very important kernel in scientific and engineering computing, which is bandwidth limited and subject to indirect memory access. Implementing an efficient SpMV kernel is a big challenge in Sunway processor. This paper proposes a general SpMV heterogeneous manycore algorithm for the traditional sparse matrix storage format CSR, which divides the task and LDM space in detail, a cache mechanism of dynamic and static buffers to improve the hit rate of vector *x*, and a dynamic-static task scheduling method to achieve load balancing. In addition, several key factors affecting the performance of SpMV are analyzed, and adaptive optimization is carried out to further enhance the performance. Finally 16 matrix from matrix market collection are used to perform tests. The experimental results show that the algorithm achieves bandwidth of 86% and average bandwidth utilization of 47%. Compared with the implementation of the controller core, the speedup can be up to 10x, and average speedup is 6.51x.

Key words: sparse matrix-vector multiplication; SpMV; Sunway 26010 processor; heterogeneous many-core; adaptive optimization

稀疏矩阵向量乘(SpMV) $y=Ax$ 是科学与工程计算中一个非常重要的计算内核,其性能往往对应用整体性能有着很大影响.SpMV 是属于访存密集型的,算法中的浮点计算与存储访问的比率很低,且稀疏矩阵非零元素分布很不规则,使得向量 x 为间接访问且访问不规则,可重用性差,这些因素给 SpMV 的高效实现带来很大挑战.

目前,超级计算机的体系结构已经从多核向众核乃至异构众核发展,然而访存墙问题却越来越突出,带宽受限型操作的峰值性能也越来越低,并且实现难度逐步增大.由我国国家并行计算机工程技术研究中心研制的新一代申威异构众核处理器已经面世,其峰值性能为 3TFlops/s,聚合访存带宽为 130GB/s,相比计算能力,其访存能力偏弱,给稀疏矩阵向量乘的高效实现带来了巨大的挑战.本文针对该处理器特点,提出一种面向传统的稀疏矩阵存储格式 CSR 的通用 SpMV 异构众核并行算法,并从任务划分、LDM 空间划分、向量 x 访存优化、负载均衡、自适应优化等角度开展工作.

1 相关工作介绍

SpMV 的实现和优化,一直是高性能计算领域科研人员的研究重点.每当一款新的处理器问世,基于该处理器的 SpMV 实现及优化的工作就会持续出现.基于 CPU 的 SpMV 工作有很多,主要从存储格式[1,2]、分块算法[3−5]、值和索引压缩[6,7]、向量化[8,9]、自适应优化[10,11]等角度开展研究.

2008 年,GPGPU 出现,开启了 GPU 用于通用计算的热潮.随后,基于 GPU 的 SpMV 工作大量涌现,这些工作主要通过存储格式、重排、压缩、自适应调优等技术解决带宽利用率、负载均衡、并行度等问题,先后提出了 HYB[12],ELLPACK-R[13],sliced-ELLPACK[14],blocked ELLPACK[15],BRC[16],BCCOO[17]等新型存储格式;研究了稀疏矩阵的重排技术[18]及压缩格式[19],以减少访存开销;研究了 GPU 平台体系结构特征、稀疏矩阵存储格式、稀疏矩阵集之间的关系,并给出自动选择模型[20];另外还研究了自动调优技术[17,21,22],以根据稀疏矩阵的特征选择最优参数并获取较优的性能.

2011 年,Intel 公司的异构众核处理器 Xeon Phi 发布.随后,Liu 等人[23]提出了新的 ESB 格式,该格式可有效改善 Xeon Phi 上 SpMV 向量化性能,并能减少访存开销,另外还提出了混合的动态调度器以改善并行任务的负载均衡性;Tang 等人[24]通过新的存储格式 VHCC、二维不规则任务划分、自动调优技术等优化了一类 scale-free 稀疏矩阵 SpMV 的性能.

另外还有一类工作涉及到多个异构众核处理器,Kreutzer 等人[25]主要从改善向量化性能的角度提出新的存储格式 SELL-C-σ;Liu 等人[26]提出了 CSR5 存储格式用于改善不规则稀疏矩阵 SpMV 的性能,在多个异构众核处理器上实现并与现有最优工作进行了对比.

本文主要研究面向申威 26010 异构众核处理器的 SpMV 并行算法及实现和优化技术,以支撑该国产平台相关应用.

2 国产申威 26010 处理器介绍

国产申威 26010 处理器采用异构众核架构,由 4 个核组(core group,简称 CG)组成,其双精度计算能力

3TFlops/s,单处理器拥有 260 个核心,采用共享存储架构,聚合访存带宽 130GB/s.基于该处理器搭建了国产神威太湖之光超级计算机,已经部署于国家超算无锡中心,其峰值性能超过 100PFlops/s.

本文主要在其一个核组上开展工作,如图 1 所示.每个核组由控制核心(management processing element,简称 MPE,又称主核)、计算核心簇(computing processing elements clusters,简称 CPE cluster,又称从核)、协议处理部件(PPU)和存储控制器(memory controller,简称 MC)组成.

平均每个核组的访存带宽为 32.5GB/s,实测带宽为 27.5GB/s.

主核采用通用的 RISC 架构,向量化宽度 256 位,采用一级数据和指令 Cache 分离、二级指令数据共享的两级片上存储层次.从核核组采用拓扑为 8×8 mesh 互联,包含 64 个计算核心和 DMA(DMA controller)控制器.计算核心采用精简的 64 位 RISC 指令集,向量化宽度为 256 位,有 64KB 的 Scratch Pad Memory(又称 LDM),通过 DMA 可实现内存与 LDM 间的快速数据传输.应用程序由控制核心启动,借助高性能线程库 Athread 将计算任务异步加载到计算核心执行,双方通过同步接口协同.

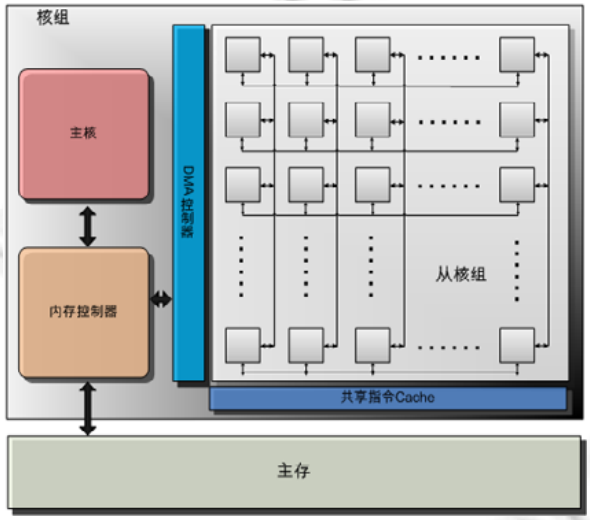


Fig.1 The architecture of SW26010 processor

图 1 国产申威 26010 处理器单核组架构图

3 CSR 格式简介

CSR 格式是目前稀疏矩阵使用最广泛的一种存储格式.设待存储的稀疏矩阵 A 是 $m\times n$ 维的,有 nz 个非零元,其通过 3 个一维数组来存储稀疏矩阵的信息,具体如下:

- $val[nz]$,记录每个非零元的值;
- $col[nz]$,记录每个非零元所在的列;
- $ptr[m+1]$,记录每行的第 1 个非零元在数组 $val[nz]$和 $col[nz]$中的索引,其中,$ptr[m]=nz$.

图 2 给出了一个示例.目前,大多数科学与工程计算应用的矩阵中均采用 CSR 格式进行存储,国际上 SpMV 算法的研究也大都以 CSR 格式为基准,如果采用其他存储格式,还需衡量该格式到 CSR 格式的转换开销,故本文直接研究基于 CSR 格式的 SpMV 算法.

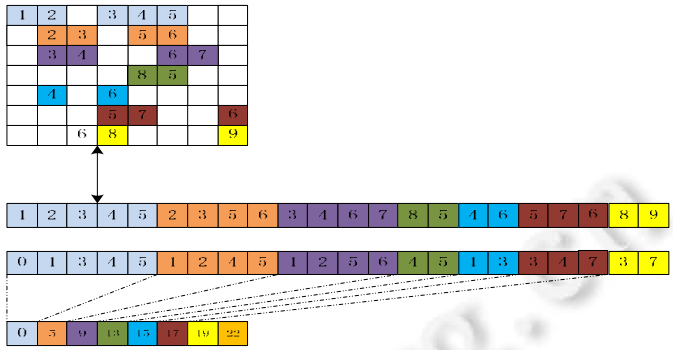


Fig.2 The CSR format

图 2 CSR 格式示意图

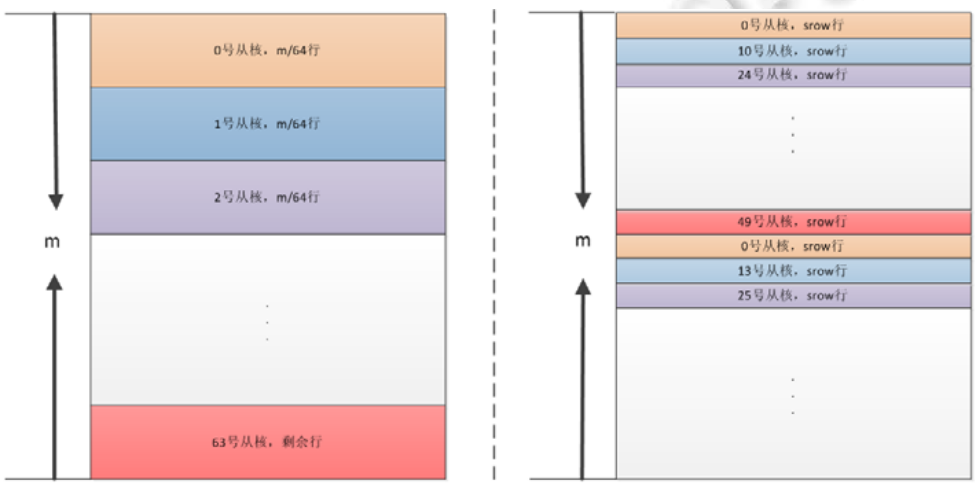
4 SpMV 异构众核并行算法

4.1 任务划分

申威众核处理器每个核组包括 1 个主核和 64 个从核,为了充分利用从核核组的计算资源,我们将计算任务尽可能的分给从核,主核主要负责前处理和控制.

对于稀疏矩阵而言,任务划分方法有两种:一维划分和二维划分.二维划分时,多个从核会同时更新 y 向量的一部分,需要加锁处理,从而导致额外的开销.对规则稀疏矩阵而言,每行的非零元个数较少,LDM 可以容纳至少一行计算所需的元素,所以我们采用一维的任务划分方法.如果矩阵一行的非零元太多,导致 LDM 空间不能一次容纳一行的元素进行计算,那么将采用主核进行计算.

一维划分方式又有两种(如图 3 所示,其中,*srow* 为当前申威处理器一个从核的 LDM 可以容纳的最多稀疏行大小).静态任务划分:将矩阵按行等分,每个从核计算 $m/64$ 行,从核的内部循环开始执行,每次只计算矩阵的 *srow* 行;动态任务划分:将矩阵 *srow* 行的计算视为一个子任务,形成任务池.每个从核一次只负责一个子任务,执行结束后,再取下一个子任务进行计算.



(a) 静态 (b) 动态

Fig.3 Task partition

图 3 任务划分示意图

具体计算方式见第 4.2 节.静态任务划分方式每个从核执行的矩阵行数基本相同;动态任务划分方式时,每

个从核执行的矩阵行数根据当前从核的执行情况动态调整,总矩阵行数可能大不相同.这两种方式分别适用于不同类型的稀疏矩阵,见第 5.2 节.

4.2 LDM空间划分

每个从核的 LDM 空间相当于一块高速缓存,从核访问 LDM 中的数据仅需要数拍即可完成,而从核直接访问主存则需要 200 多拍,所以 LDM 空间的使用对并行算法的设计至关重要.每个从核的 LDM 空间仅有 64KB,而 CSR 格式的 SpMV 计算需要 *val*,*col*,*ptr*,*x*,*y* 这 5 个数组的值才能完成.根据第 4.1 节中的任务划分方式,每个从核每次只计算 *srow* 行,那么 *y* 的空间只需 *srow* 大小,其余行计算时可以重复利用此块空间;*ptr* 数组类似,只需 *srow*+1 大小.由于 SpMV 计算中 *x* 的访存是不连续且不规则的,对整体性能影响很大.为此,我们为其预留较多的空间以增加命中率.

每个从核 64KB 空间分配如下:24KB 用于存储 *x*,*y*,*ptr* 和其他局部变量,40KB 用于存放 *val* 和 *col*.由于 *val* 为双精度数据类型,*col* 为整型数据类型,共占 12 字节,所以 40KB 空间最多只能存储 40×1024/12 个 *val* 和 *col* 元素,即 3 413 个.那么 *srow*=3413/max*nz*,其中,max*nz* 为该稀疏矩阵每行最大的非零元个数.若采用双缓冲优化,则该值减半,*val*,*col*,*ptr* 均设置两块 buffer,大小为原来的一半.

x 设置 2 块 buffer:一块静态 buffer,其大小为 x_{ssize};一块动态 buffer,其大小为 x_{dsize}.静态 buffer 加载一次后重复使用;动态 buffer 在静态 buffer 没有命中时使用,如没有命中,则从当前所需的 *x* 处加载 x_{dsize} 个数据到动态 buffer,后续计算时先查找静态 buffer,再查找动态 buffer,如果没有命中,继续加载 x_{dsize} 个数据到动态 buffer 中.具体流程如图 4 所示.

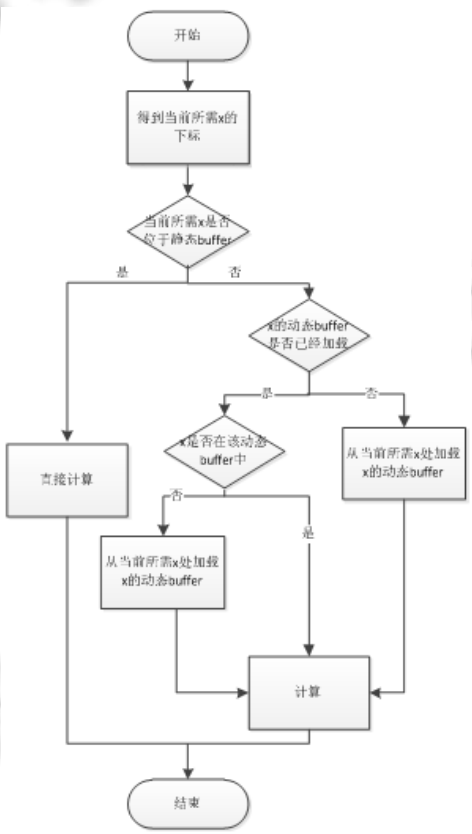


Fig.4 The flowchart of dynamic and static buffer loading of *x*

图 4 *x* 的动静态 buffer 加载示意图

对每个矩阵而言,x_{ssize}的最大值由 *srow* 确定,该缓冲区的大小直接影响了 SpMV 的最终性能.对于 x_{dsize}的选择,我们期望读取一次的开销与访问一次主存的开销相当.经测试,DMA 传递 32 个元素的开销与访问一次主存的开销相当,故 x_{dsize} 设置为 32.

5 实现及优化

5.1 *x*访存优化

稀疏矩阵向量乘中,*x* 是间接访存,访存行为很不规则,在申威众核处理器上,*x* 的访存是优化的重点,直接对其最终性能起到决定性的影响.*x* 的访存有几种方式.

(1) 所有的 *x* 直接从主存读取;

(2) 每个从核通过 DMA 预取部分 *x*,其余 *x* 通过访问主存得到,记为 *static-dma*;

(3) 动静态 buffer 方式,记为 *static-dynamic*.具体见第 4.2 节.

由于从核访问一次主存约需 200 多拍,方案 1 性能明显很差,所以实际中并未使用.方案 2 和方案 3 中静态 buffer 的大小见第 4.2 节.第 6.2.1 节给出了两种方案的性能对比结果.

另外,加载静态 buffer 的初始位置对 SpMV 的性能也有一些影响.初始位置有两种选择.

1) 从当前从核计算的行块的起始位置读取,记为 *start-x-row*;

2) 从当前从核计算行块所需的第一个 *x* 处读取,记为 *start-x-current*.

5.2 负载均衡

稀疏矩阵每行的非零元个数不尽相同,且分布不均.按照图 3(a)中的静态任务划分方法,对有些矩阵会导致从核间负载不均衡,这个负载不均衡来自两个方面.

- 每个从核计算的行块的总非零元个数可能差异较大;
- 每个从核计算的行块中 *x* 的访存行为可能差异较大.

为了解决负载不均衡的问题,本文还采用了动态任务划分的方式,如图 3(b)中所示.该方式中,从核间协同,通过采用我们自己用原子操作实现的锁来完成.

然而,由于目前的锁实现中需要访问主存,这个代价比较高,所以其性能较差,具体见第 6.2.2 节.

为此,本文对这种调度方式进一步进行了优化,只在第 1 次运行 SpMV 时采用动态调度,并记录每个计算核心所分配的任务,在以后的执行过程中,均按照这种方式来进行任务分配,我们将其称为动-静态任务调度.

5.3 自适应优化

由于实际应用中稀疏矩阵千差万别,非零元的分布方式各不相同.对每一个稀疏矩阵而言,任务分配方式、静态 buffer 大小、静态 buffer 加载的起始位置等均对其性能有着很大的影响,有必要针对该稀疏矩阵选择最优的参数组合.可选的参数如下:

- 调度方式,有两种选择:静态调度和动静态结合的调度方式;
- 静态 buffer 读取的起始位置;
- 静态 buffer 的大小,其最大值受 LDM 限制,每个矩阵均不同,初始值选为 128,每 128 递增.

为了减少搜索开销,本文对 Matrix Market 矩阵集中 57 个不同类型的稀疏矩阵选择不同参数的性能结果进行分析,发现任务调度方式、静态 buffer 读取的起始位置均与静态 buffer 的大小关系不大,据此,本文确定了如图 5 的搜索顺序.

该搜索过程需要大约 3~22 个 SpMV 的时间,但是对于实际应用来说,这个过程可以预先进行,以便于在以后的迭代过程中选用性能最高的 SpMV 实现.

Fig.5 The flowchart of search of optimal parameter

图 5 最优参数搜索顺序图

5.4 双缓冲优化

该处理器从核上支持 DMA 访存与计算重叠,为了验证其有效性,本文设置 LDM 上 *val*,*col*,*ptr* 的双 buffer. 图 6 中,上图展示了单 buffer 的计算和访存流程,下图展示了双 buffer 的计算和访存流程.但该异构众核 SpMV 算法中主要以 DMA 操作为主,计算所占的比重很小,该优化对整体的性能影响不大.

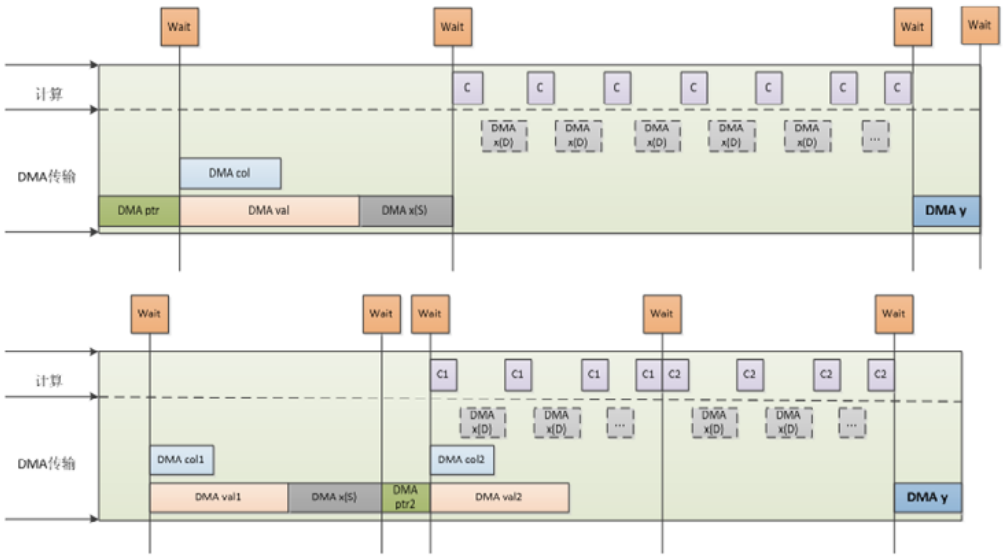


Fig.6 SpMV with CSR format in timeline

图 6 CSR 格式 SpMV 时序图

6 实验结果

6.1 实验平台

我们采用神威“太湖之光”的一个核组作为测试平台,借助高性能线程库 Athread 将计算任务异步加载到从核执行.测试矩阵选用了 Matrix Market 矩阵集中的矩阵进行测试,矩阵规模从数千到百万,矩阵非零元个数从数万到 1 00 多万.表 1 中给出了测试矩阵的基本信息.

Table 1 The information of test matrices

表 1 测试矩阵信息表

编号	矩阵名字	矩阵维数(m)	非零元数(*nz*)
1	bcsstk17	10 974	428 650
2	bcsstk28	4 410	219 024
3	raefsky2	3 242	294 276
4	Linverse	11 999	95 977
5	Cant	62 451	4 007 383
6	s3dkq4m2	90 449	4 820 891
7	fv2	9 801	87 025
8	nemeth01	9 506	725 054
9	LF10000	19 998	99 982
10	af_0_k101	503 625	17 550 675
11	cavity20	4 562	138 187
12	ecology1	1 000 000	4 996 000
13	epb3	84 617	463 625
14	qa8fk	66 127	1 660 579
15	Obstclae	40 000	197 608
16	af_shell3	504 855	17 588 875

注:下节给出的所有测试结果均为计算 50 次 SpMV 的时间.

6.2 测试结果

6.2.1 *x* 访存优化的对比结果

图 7 中比较了第 5.1 节中提到的方案 2 和方案 3 的性能,从图中可以看出,方案 3 明显优于方案 2,最高加速比可达 21 倍.这是因为方案 3 利用了稀疏矩阵的局部性,动态缓冲区的 *x* 数据得以重复利用.

图 8 中给出了部分矩阵选用两种加载静态 buffer 的起始位置的性能对比.矩阵 qa8fk,raefsky2,cavity20 选用方案 2 性能较好,而 cant 和 s3dkq4m2 是选用方案 1 性能较好,性能差最大的有 55%,最小的也有 11%.

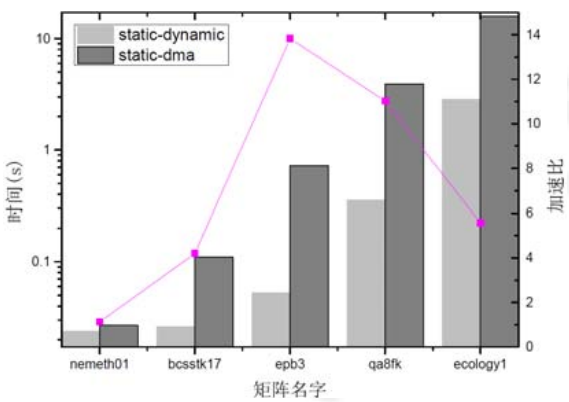


Fig.7 The optimized performance of *x* loading

图 7 *x* 访存优化效果对比图

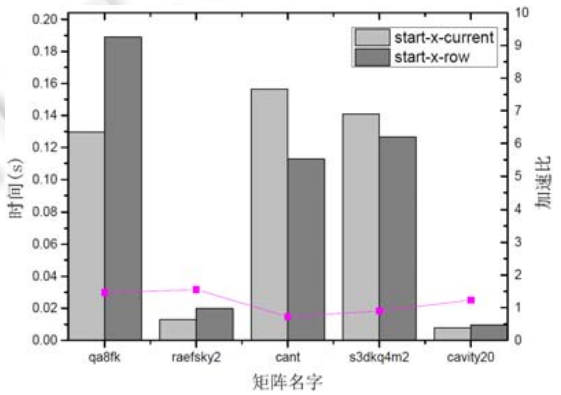


Fig.8 The impact of the start position of loading dynamic/static buffer on performance

图 8 加载动静态 buffer 的初始位置对性能的影响

6.2.2 任务调度

由于稀疏矩阵千差万别,不同的任务调度方式对其优化的效果也不尽相同,图 9 中给出了动-静态调度方式

性能较好的测试矩阵的结果,并将其与静态调度方式进行了对比.其中,两种调度方式均采用从当前所需的第 1 个 x 作为起点加载静态 buffer,并且选用了 x_{ssize} 可选范围内的最优性能.从图中可以看出,最大加速比可以达到 6 倍多,说明不同任务调度方式对某些矩阵的性能有着很大的影响.

第 6.2 节中提到:动态调度时,由于加锁引入了额外的开销.图 10 中比较了采用动态调度进行计算的时间(记为 *dynamic*)与利用动态调度的任务划分方式进行静态分配的计算时间(记为 *static*(*dynamic*)),其性能约有 10%~40%的差异.

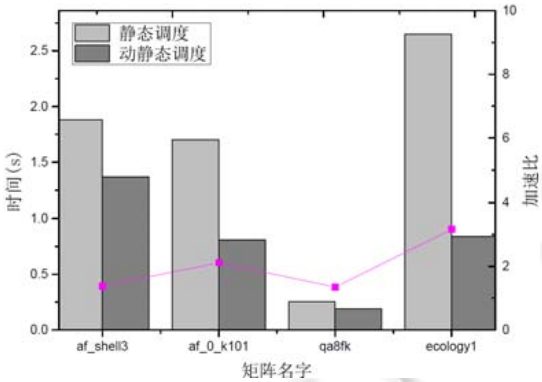


Fig.9 The impact of different scheduling method on performance

图 9 不同的任务调度方式的优化效果

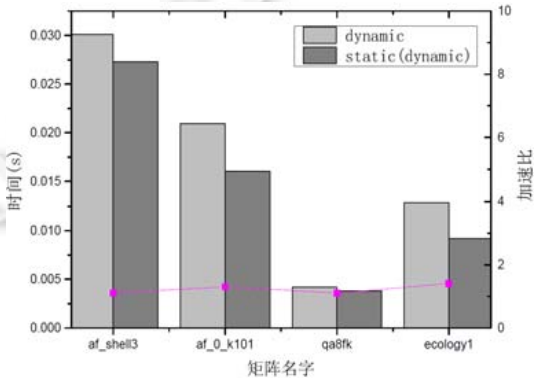


Fig.10 Comparison of the first two calculations using the static (dynamic) method

图 10 动静态调度方式前两次性能对比

6.2.3 自适应优化

图 11 给出了对第 5.3 节中提到的 3 个参数进行自适应优化的性能结果(不含调优时间),并与采用静态调度方式、x_{ssize}=1536、从当前所需的第 1 个 x 进行加载静态 buffer 的方法进行了对比.从图中可以看出:自适应优化取得了比较明显的加速效果,平均性能提升为 44%,最大的 ecology1 矩阵可达到 6 倍多,这主要是动静态任务调度带来的加速.

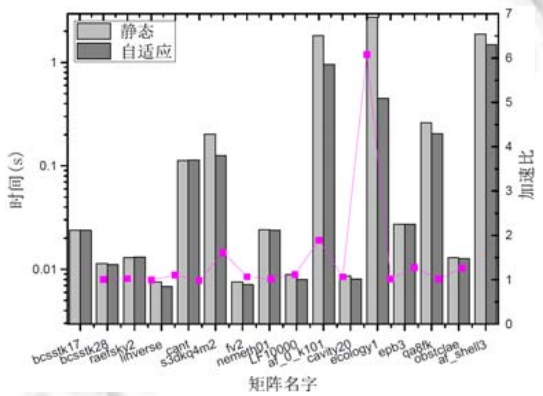


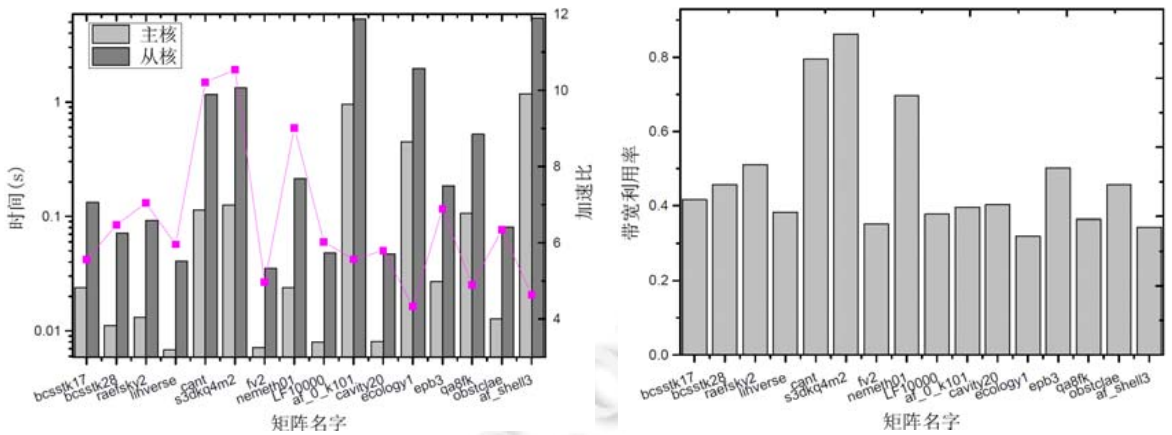
Fig.11 The performance of adaptive optimization

图 11 自适应优化效果图

6.2.4 性能结果

本文对选取的 16 个矩阵进行了测试,图 12(a)展示了分别在主核和从核运行的结果,主核版采用最原始的 CSR 格式 SpMV 实现.可以看出:测试的矩阵相对主核版均有不同程度的性能提升,最高可达 10 倍多,最低也有 4 倍多,平均加速比为 6.51 倍.另外,本文还测试了带宽利用率,总访存量采用公式 $nz\times12+(nrow+1)\times4+nrow\times8\times2$ 来计算.图 12(b)给出了测试矩阵的带宽利用率(总带宽按照实测带宽 27.5Gb/s 计算),最高可达 86.09%,最低可达

31.76%,平均带宽利用率为 47%.



(a) 性能结果 (b) 带宽利用率

Fig.12 The performance of bandwidth efficiency of test matrices

图 12 测试矩阵的性能及带宽利用率

6.3 测试结果分析

从测试结果来看,从核上 SpMV 的性能与其非零元的分布有很大关系.如果非零元分布的局部性特征比较明显,那么本算法中动静态 buffer 的命中率较高,从而整体性能较好.

目前,整体的带宽偏低,这是因为计算时采用了主核计算 CSR 格式 SpMV 的访存量,而实际在从核计算时,由于 x 的间接访问,必然会引入 x 的额外访存.未来将进一步改进 x 的访存策略,以提升整体性能.

对于一个特定的矩阵,可通过观察分析其非零元的分布规律,设计出特定的 x 的传输方案,这样能尽可能地减少 x 的冗余访存,进而提升带宽利用率和整体性能.

7 结论及下一步工作

SpMV 是众多科学与工程应用中经常调用的核心函数之一,其性能至关重要.而 CSR 格式是使用最广泛的一种稀疏矩阵存储格式.本文针对申威处理器提出了一种 CSR 存储格式 SpMV 操作的通用异构众核并行算法,该算法首先从任务划分、LDM 空间划分方面进行精细设计.为了提升向量 x 的访存命中率,本文提出了一套动静态 buffer 的缓存机制,并分析了加载静态 buffer 起始位置对性能的影响;对某些稀疏矩阵从核间负载不均衡的原因,提出了一套动静态的任务调度方法以实现负载均衡.另外,还分析了该算法中影响 SpMV 性能的几个关键因素,并开展了自适应优化,进一步提升了性能.

对于 CSR 格式的 SpMV,未来还需进一步考虑提升 x 访存命中率的方法,比如利用该处理器的寄存器通信.还可以考虑根据稀疏矩阵的 x 访存特征对其进行分类,对每一类的矩阵采用更加适合的访存方法,以提升自适应性.另外还需考虑新的整体访存量更少的存储格式,以提升整体性能.

References:

[1] Kourtis K, Karakasis V, Goumas G, Koziris N. CSX: An extended compression format for SpMV on shared memory systems. In: Proc. of the 16th ACM Symp. on Principles and Practice of Parallel Programming. San Antonio, 2011.

[2] Sun XZ, Zhang YQ, Wang T, Long GP, Zhang XY, Li Y. Crsd: Application specific auto-tuning of SpMV for diagonal sparse matrices. In: Proc. of the 17th Int'l Conf. on Parallel Processing—Vol. Part II (Euro-Par 2011). 2011. 316–327.

[3] Im EJ, Yelick K. Optimizing sparse matrix computations for register reuse in SPARSITY. In: Proc. of the Int'l Conf. on Computational Science. LNCS 2073, 2001. 127–136.

[4] Nishtala R, Vuduc R, Demmel, J, Yelick K. When cache blocking sparse matrix vector multiply works and why. In: Proc. of the Applicable Algebra in Engineering, Communication, and Computing. 2007.

[5] Mellor-Crummey J, Garvin J. Optimizing sparse matrix-vector product computations using unroll and JAM. Int'l Journal of High Performance Computing Applications, 2004,18:225−236.

[6] Willcock J, Lumsdaine A. Accelerating sparse matrix computations via data compression. In: Proc. of the 20th Annual Int'l Conf. on Supercomputing (ICS 2006). New York, 2006. 307−316.

[7] Kourtis K, Goumas G, Koziris N. Optimizing sparse matrix-vector multiplication using index and value compression. In: Proc. of the 5th Conf. on Computing Frontiers. Ischia, 2008.

[8] D'Azevedo E, Fahey M, Mills R. Vectorized sparse matrix multiply for compressed row storage format. In: Proc. of the Int'l Conf. on Computational Science. 2005.

[9] Williams S, Oliker L, Vuduc R, Shalf J, Yelick K, Demmel J. Optimization of sparse matrix-vector multiplication on emerging multicore platforms. In: Proc. of the 2007 ACM IEEE Conf. on Supercom-Putting. Reno, 2007.

[10] Vuduc R, Demmel J, Yelick K. OSKI: A library of automatically tuned sparse matrix kernels. In: Proc. of the SciDAC 2005, Journal of Physics: Conf. Series, 2005.

[11] Li JJ, Tan GM, Chen M, Sun NH. SMAT: An input adaptive auto-tuner for sparse matrix-vector multiplication. In: Proc. of the 34th ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI 2013). 2013. 117−226.

[12] Bell N, Garland M. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In: Proc. of the Conf. on High Performance Computing Networking, Storage and Analysis. ACM Press, 2009. 18.

[13] Vazquez F, Fernandez J, Garzon E. A new approach for sparse matrix vector product on NVIDIA GPUs. Concurrency and Computation: Practice and Experience, 2011,23(8):815−826.

[14] Monakov A, Lokhmotov A, Avetisyan A. Automatically tuning sparse matrix-vector multiplication for GPU architectures. In: Proc. of the Int'l Conf. on High-Performance Embedded Architectures and Compilers. Berlin, Heidelberg: Springer-Verlag, 2010. 111−125.

[15] Choi JW, Singh A, Vuduc RW. Model-Driven autotuning of sparse matrix-vector multiply on GPUs. ACM Sigplan Notices, 2010, 45(5):115−126.

[16] Ashari A, Sedaghati N, Eisenlohr J, *et al*. An efficient two-dimensional blocking strategy for sparse matrix-vector multiplication on GPUs. In: Proc. of the 28th ACM Int'l Conf. on Supercomputing. ACM Press, 2014. 273−282.

[17] Yan S, Li C, Zhang Y, *et al*. yaSpMV: Yet another SpMV framework on GPUs. ACM SIGPLAN Notices, 2014,49(8):107−118.

[18] Pichel JC, Rivera FF, Fernández M, *et al*. Optimization of sparse matrix-vector multiplication using reordering techniques on GPUs. Microprocessors and Microsystems, 2012,36(2):65−77.

[19] Tang WT, Tan WJ, Ray R, *et al*. Accelerating sparse matrix-vector multiplication on GPUs using bit-representation-optimized schemes. In: Proc. of the Int'l Conf. on High Performance Computing, Networking, Storage and Analysis. ACM Press, 2013. 26.

[20] Sedaghati N, Mu T, Pouchet LN, *et al*. Automatic selection of sparse matrix representation on GPUs. In: Proc. of the 29th ACM Int'l Conf. on Supercomputing. ACM Press, 2015. 99−108.

[21] Ashari A, Sedaghati N, Eisenlohr J, *et al*. Fast sparse matrix-vector multiplication on GPUs for graph applications. In: Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC 2014). IEEE, 2014. 781−792.

[22] Guo D, Gropp W. Adaptive thread distributions for SpMV on a GPU. In: Proc. of the Extreme Scaling Workshop. University of Illinois at Urbana-Champaign, 2012. 2.

[23] Liu X, Smelyanskiy M, Chow E, *et al*. Efficient sparse matrix-vector multiplication on x86-based many-core processors. In: Proc. of the ACM Int'l Conf. on Supercomputing. 2013. 273−282.

[24] Tang WT, Zhao R, Lu M, *et al*. Optimizing and auto-tuning scale-free sparse matrix-vector multiplication on Intel Xeon Phi. In: Proc. of the 2015 IEEE ACM Int'l Symp. on Code Generation and Optimization (CGO). 2015. 136−145.

[25] Kreutzer M, Hager G, Wellein G, *et al*. A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units. SIAM Journal on Scientific Computing, 2014,36(5):C401−C423.

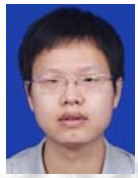
[26] Liu W, Vinter B. CSR5: An efficient storage format for cross-platform sparse matrix-vector multiplication. In: Proc. of the ACM Int'l Conf. on Supercomputing. 2015. 339−350.

刘芳芳(1982－),女,山西霍州人,高级工程师,CCF 专业会员,主要研究领域为高性能扩展数学库,稀疏迭代解法器,异构众核并行.

杨超(1979－),男,博士,研究员,博士生导师,主要研究领域为高性能计算,科学与工程计算.

袁欣辉(1989－),男,研究实习员,主要研究领域为国产高性能计算机的性能优化,MPI 库实现与优化.

吴长茂(1974－),男,博士,助理研究员,CCF 专业会员,主要研究领域为并行软件,并行算法,并行卫星成像仿真.



敖玉龙(1990－),男,博士,CCF 专业会员,主要研究领域为大规模并行计算和程序优化.