doi:10.11887/j.cn.202003001　　http://journal.nudt.edu.cn

GSVM:一种支持 Gather/Scatter 的向量存储器*

陈海燕,刘　胜,吴健虢
(国防科技大学 计算机学院, 湖南 长沙　410073)

摘　要:宽单指令多数据流(Single Instruction Multiple Data, SIMD)架构数字信号处理器一般都能高效支持地址连续或等距跨步等规则应用的向量访存,但对于科学与工程计算中广泛存在的不规则应用的数据访存则带宽利用率往往较低,从而大幅降低了其整体运算能效。为了提高不规则应用的向量访存性能,基于某 SIMD 数字信号处理器的体系结构,设计了一种支持 Gather/Scatter 访存的向量存储器 GSVM。通过设计与 SIMD 宽度相匹配的向量地址计算单元和合适深度的冲突缓冲器阵列,实现了 Gather/Scatter 指令向量地址计算、仲裁与缓存的全流水访存操作。实验结果表明,相比以前不支持 Gather/Scatter 访存的存储器,GSVM 在增加 22% 的硬件代价基础上,基于稀疏矩阵向量乘的测试程序集获得了 2 ~ 8 的性能加速比。

关键词:单指令多数据流;Gather/Scatter;向量随机访存;访存冲突
中图分类号:TN47　　**文献标志码**:A　　**文章编号**:1001 - 2486(2020)03 - 001 - 08

GSVM: a vector memory to support Gather/Scatter

CHEN Haiyan, LIU Sheng, WU Jianguo
(College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: The very wide SIMD (single instruction multiple data) digital signal processor often supports vector memory access mode of regular applications with contiguous or equal address strides, but its access bandwidth utilization is usually very low for the irregular application accesses that exist widely in scientific and engineering computations. It reduces the overall computing performance of digital signal processor. In order to improve the vector access performance for the irregular applications, the vector memory called GSVM to support the Gather/Scatter access was designed on the basis of the architecture of a SIMD digital signal processor. The vector address generation unit and the conflict buffer array matching the SIMD width were designed to realize the full pipeline operations of Gather/Scatter instruction. The experimental results show that compared with the vector memory without Gather/Scatter, the GSVM obtains 2 ~ 8 times speedup for sparse - matrix vector multiplication test programs with the 22% hardware overhead.

Keywords: single instruction multiple data; Gather/Scatter; vector random access; access conflicts

为充分开发微处理器的数据并行性、以较低的硬件代价和功耗实现密集型数据处理,获得较高的峰值运算性能,单指令多数据流(Single Instruction Multiple Data, SIMD)技术成为微处理器体系结构发展的一个重要扩展。对于地址连续或等距跨步等规则应用的数据访存模式,一般的 SIMD 结构处理器往往都能提供充足的访存带宽,得到较理想的峰值运算性能;但在处理不规则应用时,其访存带宽性能急剧下降,主要原因是不规则应用的访存模式复杂、无规律,数据的空间局部性差,目前的 SIMD 架构微处理器不能很好支持。如何高效支持不规则应用访存以提高通用性能成为 SIMD 微处理器设计中面临的一个重要问题。

科学计算和工程应用中存在大量的不规则数据访存,例如稀疏矩阵计算、图像边缘检测和语音识别等[1-2]。有学者总结出十余类基本的并行计算模式,其中一半以上都是不规则的[3],即这些并行计算模式中存在大量的不规则数据访存,数据访存模式随机无规律,时间和空间局部性差等[4]。随着 SIMD 技术的不断发展,片上集成的按 SIMD 方式操作的运算单元越来越多,需要越来越宽的 SIMD 访存带宽支持,而不规则访存潜在的访问冲突降低了处理器的实际数据带宽,增加了访存延时,降低了数据访存效率[5-6]。支持单指令多线程(Single Instruction Multiple Threads, SIMT)的向量访存仅支持 SIMD 结构中每个运算单元访问各自对应的数据地址空间,不能共享整个数据存储器,限制了其应用范围。通过支持

* **收稿日期**:2019 - 04 - 30
基金项目:国家自然科学基金青年科学基金资助项目(61602493)
作者简介:陈海燕(1967—),女,四川南充人,研究员,硕士,硕士生导师,E-mail:hychen608@163.com

SIMT方式的直接存储器访问(Direct Memory Access,DMA)控制器加载[7],能将外部存储器中不规则访存数据通过DMA方式重新组织并加载到向量存储器的连续地址空间中,但是这种方式需要程序员进行显式的访存数据组织,DMA设计需配置大量的数据缓冲器,可编程性差且DMA的硬件代价较大。

Gather/Scatter指令起源于早期的向量处理机,如Cray-2[8-9]、Fujitsu的VP-100/200[10]和NEC的SX[11],随着半导体工艺的发展开始被集成到了片上。近年来,Intel也为短向量SIMD结构增加了Gather和Scatter指令操作:AVX2设计了Gather指令,Knights Corner设计了有限的Gather/Scatter指令[12],AVX-512拥有完整的Gather/Scatter指令[13]。表1为Gather/Scatter的表示[14],其中Scatter指的是将向量寄存器R_{in}中的n个数据写入一组访存地址$R_{out}[\boldsymbol{L}[i]]$,即由整数向量$\boldsymbol{L}$索引的$n$个不同的访存地址。Gather指的是读取$n$个访存地址的数据,这些数据的地址由整数向量$\boldsymbol{L}$定义,并将其输出到向量寄存器$R_{out}$中。

表1　Gather/Scatter的表示

Tab.1　Expression of Gather/Scatter

	Gather	Scatter
输入	$R_{in}[1,2,\cdots,n]$, $\boldsymbol{L}[1,2,\cdots,n]$	$R_{in}[1,2,\cdots,n]$, $\boldsymbol{L}[1,2,\cdots,n]$
输出	$R_{out}[1,2,\cdots,n]$	$R_{out}[1,2,\cdots,n]$
功能	$R_{out}[i]=R_{in}[\boldsymbol{L}[i]]$, $i=1,2,\cdots,n$	$R_{out}[\boldsymbol{L}[i]]=R_{in}[i]$, $i=1,2,\cdots,n$

1　GSVM总体结构设计

1.1　宽SIMD架构DSP总体结构

V-DSP是一款自主研制的面向图像处理及稀疏矩阵运算等数据密集型应用的宽SIMD架构32位数字信号处理器(Digital Signal Processor,DSP),主频为1 GHz。为开发指令级和数据级并行性,V-DSP采用超长指令字和SIMD技术,支持标、向量并行处理,每拍最多可派发、执行11条标、向量指令。其总体结构如图1所示,主要包括取指单元、一级程序Cache(first Level Program Cache,L1P)、指令派发单元、标量处理部件(Scalar Processing Unit,SPU)、标量存储器(Scalar Memory,SM)、向量处理部件(Vector Processing Unit,VPU)、Gather/Scatter向量存储器(Gather/Scatter Vector Memory,GSVM)以及DMA等。其中,SPU完成指令流控制和标量数据处理,SM为其提供标量数据访存;VPU实现向量数据运算,其内部集成了16个按SIMD方式操作的向量运算单元(Vector Processing Element,VPE),每个VPE每拍能进行3次32位浮点乘加运算;GSVM是程序员可见的片上大容量向量存储器,为16个VPE提供两条并行向量访存指令操作支持,并通过DMA实现GSVM与DSP核外的数据传输。

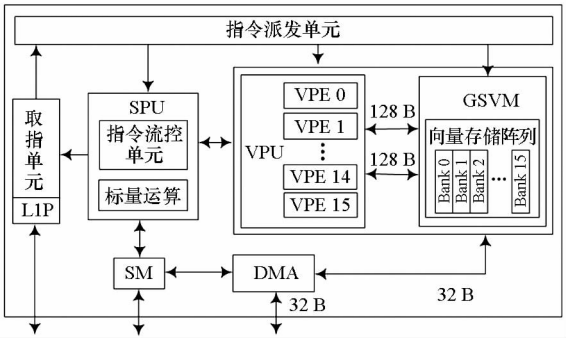

图1　V-DSP总体结构示意

Fig.1　Architecture of V-DSP

1.2　GSVM访存指令设计

GSVM支持半字(16 bit)、单字(32 bit)和双字(64 bit)粒度的常规向量访存和Gather/Scatter访存指令操作,可同时支持2条向量访存指令和DMA读、DMA写4请求并行访问,与VPU的最大数据带宽为2×128 B/Cycle,与DMA的数据传输带宽为32 B/Cycle。

常规向量访存操作只能对地址连续或等地址跨步的规则数据进行访问,只需提供一个访存起始地址即可得到各个VPE的访存地址,因而其访存指令只需指定一组基址寄存器和地址偏移寄存器就可实现线性寻址。而Gather/Scatter访存要支持不规则数据向量随机访存,即各个VPE的访存地址是随机、无规律的,因此GSVM设计了多组专用的向量地址寄存器文件,每组向量地址寄存器文件包括一个向量基址寄存器(Vector base Address Register,VAR)和一个向量地址偏移寄存器(Vector address Offset Register,VOR),单个VAR和VOR又分别由16个基址寄存器和16个地址偏移寄存器组成。Gather/Scatter访存指令译码后,GSVM会读取某组向量地址寄存器文件中16个基址寄存器和16个地址偏移寄存器,由地址计算单元根据向量地址寄存器的内容并行计算出16组访存地址,实现Gather/Scatter的向量访存。

GSVM 访存指令分为常规的向量 Load/Store 访存指令和 Gather/Scatter 访存指令两种类型,指令编码设计统一采用 V-DSP 的 32 位指令编码格式。

1.3　GSVM 总体结构

GSVM 的整体结构如图 2 所示,由向量访存指令译码、向量地址计算单元(Vector Address Generating Unit,VAGU)、向量存储体、向量存储控制器(Vector Memory Controller,VMC)、数据对齐与同步等功能模块构成,以实现向量访存流水线操作。其中,向量存储体采用多体地址交叉组织结构,为 VPE、DMA 提供并行访存带宽;向量访存指令译码模块对派发的向量访存指令进行译码,获得向量访存指令类型、寻址模式、访存粒度、地址寄存器文件、目标向量寄存器等访存信息;VAGU 根据寻址模式和地址寄存器文件计算 16 个 VPE 的访存地址并进行请求分流处理;VMC 实现向量访存和 DMA 读/写访存冲突仲裁和缓存,如果存在访存冲突,根据仲裁规则确定各 VPE 访问向量存储器的顺序。若访存指令为向量 Store 或 Scatter,则将向量数据按仲裁顺序写入对应的向量存储体地址中;若访存指令为向量 Load 或 Gather,则数据从向量存储体读出后还需经过数据对齐与同步后再返回给 VPU 的向量寄存器。若为 DMA 读访问,则通过向量转换缓冲接口(Vector Transformation Buffer interface,VTB)返回给 DMA。

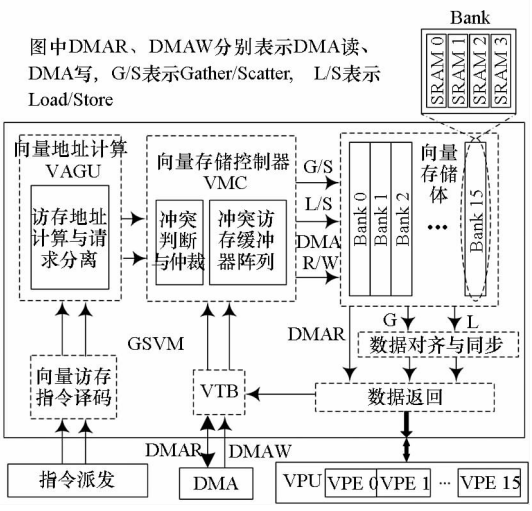

图 2　GSVM 的总体结构示意

Fig. 2　Overall structure of GSVM

1.4　向量存储体结构

向量存储体容量为 512 KB,为所有 VPE 所共享。为了满足 16 个 VPE 和 DMA 的并行向量访存需求,向量存储体由 16 个存储块(图 2 中 Bank 0 ~ Bank 15)按双字低位地址交叉组织,每个 Bank 容量为 32 KB。为减少片上存储器面积,同时提高并行访存带宽能力,单个 Bank 采用多个单端口静态随机存储器(Static Random Access Memory,SRAM)组成。

由于 Gather/Scatter 访存是无规律的,即任何一个 VPE 都可能访问任意地址,极端情况下同拍最多可能会有 16 个请求访问同一 Bank。设计 Bank 多体结构时,需要对并行访存带宽能力和硬件代价进行折中考虑。

假设每个 VPE 的访存地址为全随机均匀分布,即 16 个 VPE 随机访问任一 Bank 的概率都是相同的。根据搭建的 Gather/Scatter 访存分布模型和访存冲突模型对访存行为进行理论和量化分析[15],得到结果:16 个 VPE 随机访问 16 个 Bank,在所有可能的访存情况排列组合中,访问同一 Bank 的请求数小于或等于 4 的情况已经覆盖了 96% 以上的访存情况。因此,向量存储体的每个 Bank 采用 4 个单端口 SRAM(SRAM 0 ~ SRAM 3)按高、低位地址交叉方式组织,构成上下存储空间,每个 SRAM 容量为 8 KB,最多可完成 4 个并行访存请求。该存储体组织方式既支持常规向量访存方式下的 4 请求并行访问,减少了访存指令与 DMA 访问的冲突;又符合 Gather/Scatter 访存模式下多个 VPE 访问同一 Bank 时发生访存冲突概率的情况,提高了向量访存带宽效率。

2　GSVM 访存流水线设计

2.1　向量地址计算单元

在 Gather/Scatter 访存模式下,VPU 中的每个 VPE 的访存地址都是无规律的,因此需要提供和 VPE 个数相同的向量访存地址 VAddr(Addr 0 ~ Addr 15)。如图 3 所示,VAGU 首先实现向量访存地址计算,即根据指令译码的相关请求信号,读取某组 VAR 和 VOR 中 16 个基址寄存器(AR 0 ~ AR 15)和 16 个地址偏移寄存器(OR 0 ~ OR 15)

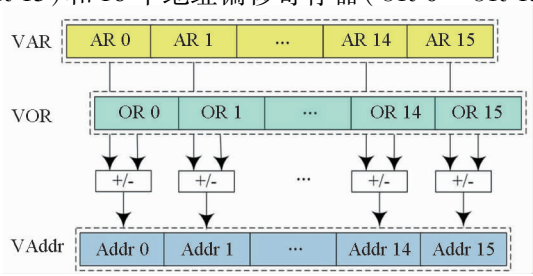

图 3　VAGU 结构示意

Fig. 3　Structure of VAGU

的数据,再根据指令访存粒度和寻址方式计算各VPE的访存地址,若指令需要更新基址,则使用计算出的访存地址更新基址寄存器。

由于Gather/Scatter访存是不规则的,各个VPE的访存地址随机分布,VAGU在计算各VPE的访存地址后,需要将每个VPE访存请求相关的控制信号、数据及地址信息组合成对应的访存请求数据包,通过访存地址译码出各个VPE访问的Bank编号,然后将各VPE访存请求的数据包按其访问的不同Bank进行并行分流处理,访问同一Bank的请求包按VPE编号从小到大的优先级顺序进入该Bank的请求打包分流缓冲器。如图4所示,VPE 0、VPE 1和VPE 5这3个访存请求都访问Bank 0的数据,VAGU分别将与这3个访存请求相关的控制信号和数据信息打包并分流到Bank 0中处理。极端情况下,16个VPE可能访问同一Bank,因此每个Bank对应的VPE请求打包分流缓冲器深度为16。

2.2　访存仲裁与冲突缓冲器阵列

各个Bank的访存请求打包分流后进入访存冲突仲裁与缓存流水站,如果有多个访存请求访问同一Bank的相同SRAM,则存在访存冲突。由于每个Bank由4个单端口SRAM体组成,最多可支持4个不冲突的并行访存请求进行读、写操作,为了减小冲突判断逻辑开销且不降低访问性能,每次最多对4个同一Bank的访存请求进行冲突判断,多余的请求则缓存至该流水线的站间寄存器中,当前面的请求处理完毕后再取出剩余的访存请求进行处理。当发生访存冲突时,该Bank的仲裁部件对访存请求进行仲裁,确定VPE访存请求的访问顺序。仲裁的优先级按照VPE编号由小到大的顺序递减;每拍访问同一Bank中的相同SRAM的请求只有1个仲裁成功,其余仲裁失败的VPE请求则按优先级顺序存入对应的冲突缓冲器;访问不同SRAM的请求则可以并行执行。

为了解决访存冲突问题,GSVM为16个Bank设计了一个冲突缓冲器阵列来缓存各Bank仲裁失败的访存请求,如图5所示,该冲突缓冲器阵列由与SIMD宽度匹配的(即16个)、深度为3的缓冲器(buffer 0 ~ buffer 15)组成。Bank和冲突缓冲器

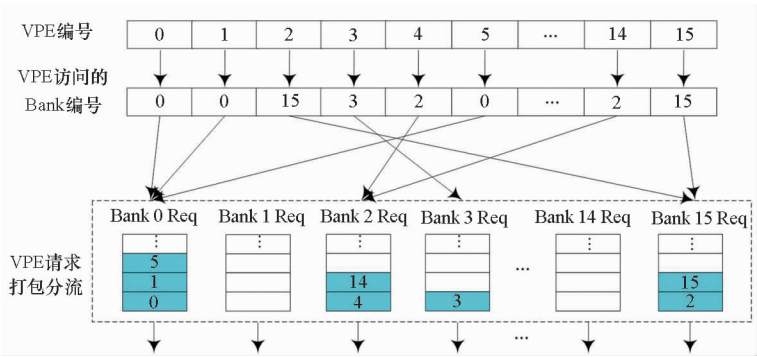

图4　VPE访存请求分流

Fig. 4　VPE access memory requests split

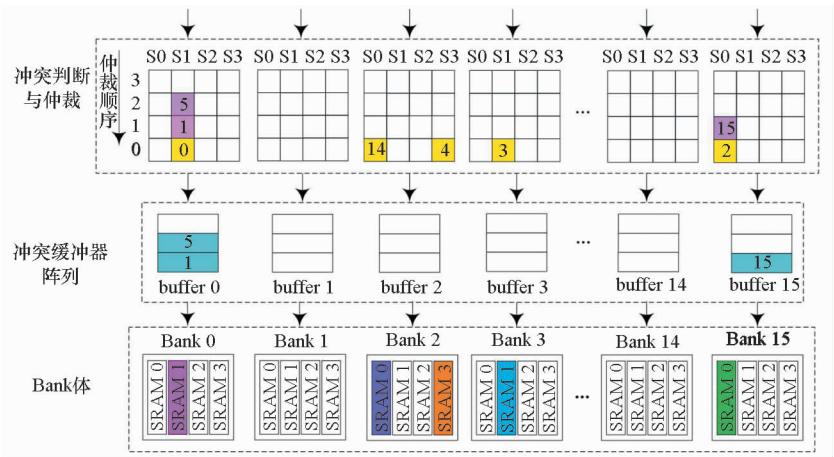

图5　冲突判断与缓冲器阵列结构

Fig. 5　Structure of conflict judgement and buffer array

一一对应,当拍访问同一 Bank 仲裁失败的请求按仲裁顺序都被缓存进对应的 buffer 中,仲裁成功的访存请求则直接访问该 Bank,下一拍再从该 buffer 中顺序取出一个访存请求继续访存操作。当同拍所有请求都处理完毕后才允许发送下一拍的请求。访存请求不存在冲突时,每拍最多可并行处理 4 个访问同一 Bank 的访存请求;存在冲突时,除了仲裁成功进入后面访存流水线的访存请求外,最多需要缓存 3 个仲裁失败的访存请求。因此,buffer 深度设计为 3 即可满足需求。缓冲器阵列的设计降低了冲突仲裁和访存控制器的复杂度,以较低的硬件开销实现了多请求随机访存。

2.3　数据对齐与同步

由于 V-DSP 采用 SIMD 结构,指令流水线采用锁步执行方式,故 Gather/Scatter 访存存在访存冲突时,不仅将造成其他指令流水线停顿,该拍访存指令操作也将延迟 1 拍或多拍完成,直到访存流水线按仲裁顺序完成当拍所有存在冲突的访存请求。对 Gather 指令而言,访存数据读出后还要进行数据整理对齐与同步,才能获得该读指令所需的全部向量数据。如图 6 所示,数据对齐与同步单元需要将当前读出的 16 个数据(D 0 ~ D 15)按照访存粒度截取有效部分,根据输出数据对应的 VPE 请求编号,将有效数据重新排列对齐,并完成同步后写回到对应的 VPE 中。

存在访存冲突时,同一条 Gather 指令不同时钟周期从向量存储器读出的数据需要进行节拍同步,优先读出的数据被缓存在该读出流水线的站间向量寄存器里,等待后面的读数据,每读出一个有效数据就将相应的标志位 Mask 置为 1。当向量寄存器对应的所有 Mask 位都置为 1 时,则表示该指令 16 个访存数据都全部读出,可以将向量寄存器的数据写回给 VPU,同时准备处理下一条指令。

3　面积与性能评估

3.1　逻辑综合结果分析

使用逻辑综合工具,时钟周期约束设置为 0.54 ns,基于某厂家 40 nm 工艺库,在相同的综合环境和约束下分别对支持 Gather/Scatter 和不支持 Gather/Scatter 的向量存储器进行逻辑综合[16]。综合结果表明,支持 Gather/Scatter 后,向量存储器面积增加了 22%。主要原因是:增加了向量地址寄存器文件,VAGU 由原来的 1 套地址计算逻辑增加为 16 套,并增加了 16 套 VPE 请求打包分流逻辑;每个 Bank 分别又增加了 1 套仲裁和 3 深度的冲突缓冲器以及大量的站间向量寄存器。

3.2　性能评估与分析

3.2.1　实验环境

稀疏矩阵向量乘(Sparse Matrix-Vector multiplication, SpMV)广泛应用于线性代数求解和科学计算领域,是科学计算中重要的计算核心[17-18]。由于 SpMV 计算中访存数据是不规则跨步的,常规的向量存储器访存带宽瓶颈导致 SpMV 的运算性能一般都比较低。

本文基于 V-DSP 的 RTL 级全芯片验证环境,通过 SpMV 部分测试集对 GSVM 的访存效率和性能进行评估,比较 Gather/Scatter 访存和常规向量 Load/Store 访存在 SpMV 运算中的效率。测试时采用 V-DSP 指令集编写系统级汇编激励,通过模拟工具 NC-Verilog 进行仿真。

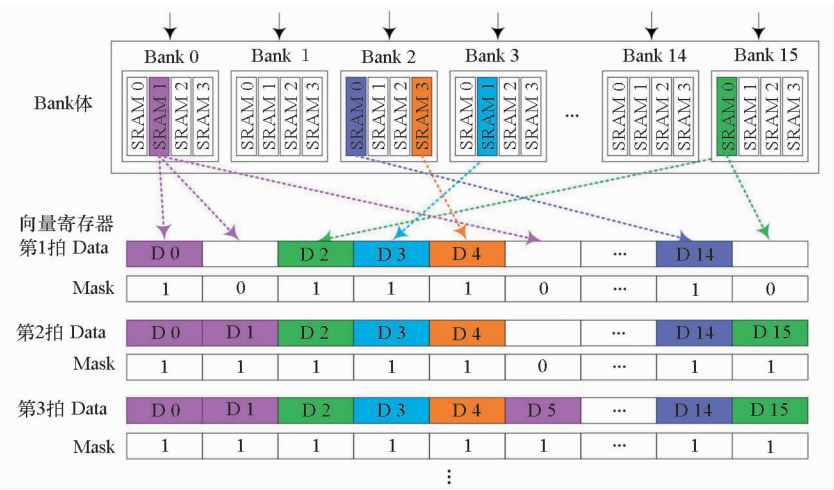

图 6　数据重整理对齐与同步结构

Fig. 6　Structure of data alignment and synchronization

实验采用的稀疏矩阵测试集来自佛罗里达大学稀疏矩阵集合(SuiteSparse Matrix Collection)[19],该测试集广泛用于线性代数领域中对稀疏矩阵算法进行开发和性能评估,涵盖了结构工程、计算流体力学,电路模拟等领域,本实验选取其中一部分作为测试集。测试前,首先使用MATLAB对测试集的矩阵数据进行压缩处理,然后分别使用Gather/Scatter和常规向量Load/Store两种不同的向量访存指令对不同的稀疏矩阵测试进行评估,对运行结果进行分析。

3.2.2　实验方案

为发挥SIMD宽向量处理单元的优势,采用基于状态压缩表(State transition Compress Table,SCT)格式的稀疏矩阵压缩算法,去除稀疏矩阵中多余的0元素以降低存储开销,并将压缩后的矩阵跨步合并为新的矩阵[20]。通过MATLAB对稀疏矩阵测试集的数据进行压缩,具体算法如下。

设VPU的SIMD宽度为L,每次取矩阵的L行进行压缩,去除矩阵每行多余的0元素。为了避免条件操作带来的性能开销,压缩后的矩阵长度(即矩阵的列数)与其中具有最多非0元素的矩阵行压缩后的长度一致,若某行非0元素较少,压缩后则需要在该行末尾补0,使得该行长度与对应的数据压缩矩阵长度相同。采用列索引矩阵记录压缩后矩阵中的非0元素在原矩阵中的列索引,行计数矩阵记录该矩阵的L行压缩后的长度。对于较大规模的稀疏矩阵,按SIMD宽度对矩阵按行分块进行压缩并补0,然后将压缩后得到的数据矩阵块、列索引矩阵块分别进行拼接,得到矩阵行数与SIMD宽度相同的压缩矩阵,并记录拼接的每个压缩数据矩阵块的长度。

如图7所示的一个8×8的稀疏矩阵,假设VPU的SIMD宽度$L=4$,则该矩阵可按SIMD宽度分为上下两个子块,每个子块压缩后的数据矩阵长度与其中具有最多非0元素的矩阵行压缩后的长度相同,即图7中两个数据矩阵子块对应的压缩后矩阵长度分别为原矩阵的第3行和第7行压缩后的元素个数,长度不够的行需在末尾补0;列索引矩阵记录压缩后数据矩阵中的非0元素在原矩阵中的列索引,在列索引矩阵中,数据矩阵补的0元素对应的列索引用x表示。通过列索引访问向量数据为不规则访存,为了避免压缩后数据矩阵补0带来的访存冲突,数据矩阵中补0的位置,即在列索引矩阵中索引值x的位置需填补与其他非0元素列索引值不同的值,从而减少访存冲突的发生。图8所示为压缩后的矩阵块拼接得到的数据矩阵、列索引矩阵和行计数矩阵。行计数矩阵记录压缩后的各个矩阵块的长度,1~4行压缩后矩阵块的长度为3,5~8行压缩后矩阵块的长度为2。压缩后的矩阵分别由一维数组存储,通过DMA传输将压缩后的矩阵搬运至向量存储器。实验中GSVM的SIMD宽度为16,每个VPE分别对压缩矩阵的一行进行乘加运算,各行之间的乘加运算是并行的,行计数值用来做判断条件,判断该行数据的运算是否结束。

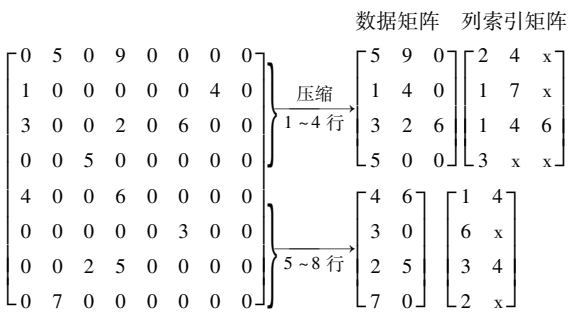

图7　稀疏矩阵按行压缩示意

Fig. 7　Sparse matrix compressed by rows

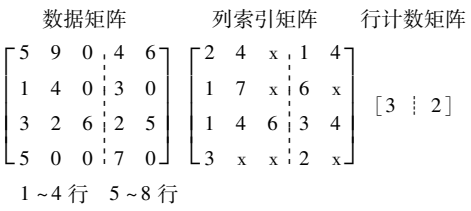

图8　稀疏矩阵行拼接示意

Fig. 8　Sparse matrix splicing by rows

3.2.3　实验结果分析

稀疏矩阵的非0元素分布是无规律的,矩阵的列索引也是无规律的,通过压缩矩阵元素的列索引访问数据属于不规则访存。Gather/Scatter访存指令根据矩阵列索引读取离散的向量数据,只需要一条Gather指令即可获取VPU所需数据,其向量访存带宽利用率高,因此程序执行时间较短。常规的向量访存指令不支持不规则数据访存,只能串行访问存储的数据,因此需要通过标量访存指令根据元素列索引读取数据,然后通过标向量转换寄存器将标量指令读取的数据转换为向量数据提供给VPU使用,因而常规向量访存指令在访问不规则数据时需要更多的时间。

图9所示为基于V-DSP的RTL级全芯片运行环境,使用Gather/Scatter访存指令运行部分稀疏矩阵测试程序相比常规向量Load/Store访存指

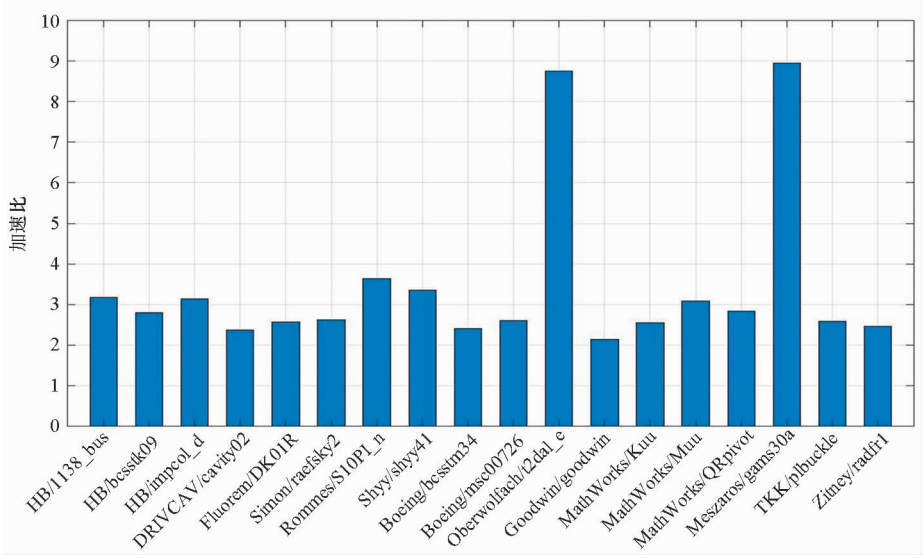

图 9　Gather/Scatter 访存加速比

Fig. 9　Access memory speedup of Gather/Scatter

令运行获得的加速比,可见这些测试程序的性能都至少提高了 1 倍以上。当无访存冲突发生时,GSVM 访存效率最高,如测试集 Meszaros/gams30a 和 Oberwolfach/t2dal_e,此时均获得了超过 8 的性能加速比[16]。实验结果表明,Gather/Scatter 访存能显著提升 SpMV 的运算效率,数据访存冲突越少则使用 Gather/Scatter 访存获得的性能越高。

4　结论

针对一般的宽 SIMD 架构 DSP 面向不规则数据访存应用的性能瓶颈,设计了支持 Gather/Scatter 的向量存储器 GSVM 整体架构,通过多存储体组织的方式来提供并行访存带宽、减少访存冲突,通过冲突缓冲器阵列来缓存仲裁失败的访问请求,有效解决不规则访存冲突的问题,实现了不规则向量访存全流水设计,提高了访存效率。通过实验对有无 Gather/Scatter 访存指令情况下稀疏矩阵乘的部分测试集进行性能测试,结果表明 GSVM 对不规则数据访存具有显著的加速作用,能有效提高 SIMD 架构 DSP 的性能。

参考文献(References)

[1] Hughes C J. Single-instruction multiple-data execution[M]// Synthesis Lectures on Computer Architecture. VT, US: Morgan & Claypool, 2015: 1 - 121.

[2] Kumar M, Serrano M, Moreira J, et al. Efficient implementation of scatter-gather operations for large scale graph analytics [C]//IEEE High Performance Extreme Computing Conference, 2016: 1 - 7.

[3] Asanovic K, Bodik R, Catanzaro B, et al. The landscape of parallel computing research: a view from Berkeley, UCB/EECS - 2006 - 183 [R]. Electrical Engineering and Computer Sciences, 2006.

[4] 穆帅. 针对不规则应用的图形处理器资源调度关键技术研究[D]. 北京: 清华大学, 2013.
MU Shuai. Research on resources schedule for irregular applications on graphics processing units [D]. Beijing: Tsinghua University, 2013. (in Chinese)

[5] Zhang E Z, Jiang Y L, Guo Z Y, et al. On-the-fly elimination of dynamic irregularities for GPU computing[C]// Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems, ACM, 2011: 369 - 380.

[6] Sung I J, Stratton J A, Hwu W M W. Data layout transformation exploiting memory-level parallelism in structured grid many-core applications[C]//Proceedings of 19th International Conference on Parallel Architectures and Compilation Techniques, IEEE, 2017: 4 - 24.

[7] 孔宪停, 陈海燕, 徐沛文. 一种支持 SIMT 方式 DMA 加载的向量存储器[C]//第十九届计算机工程与工艺年会, 2015.
KONG Xianting, CHEN Haiyan, XU Peiwen. A vector memory to support DMA load with SIMT method [C]// Nineteenth NCCET, 2015. (in Chinese)

[8] Cray Research, Inc.. CRAY-2 computer systems functional description Jun89 [M]. Various Electronics Service Manuals, 1989.

[9] Lewis J G, Simon H D. Theimpact of hardware gather/scatter on sparse Gaussian elimination [J]. SIAM Journal on Scientific and Statistical Computing, 2006, 9 (2): 304 - 311.

[10] Miura K, Uchida K. FACOM vector processor system: VP -

100/VP – 200 [C]//Proceedings of NATO Advanced Research Workshop on High-Speed Computing, 1984, 310(6991): 1344 – 1345.

[11] Watanabe T. Architecture and performance of NEC supercomputer SX system [J]. Parallel Computing, 1987, 5(1/2): 247 – 255.

[12] Intel® Corporation. Intel® Xeon Phi™ coprocessor instruction set architecture reference manual [M]. Intel® Corporation, 2012.

[13] Intel® Corporation. Intel® architecture instruction set extensions programming reference [M]. Intel® Corporation, 2014.

[14] He B S, Govindaraju N K, Luo Q, et al. Efficient gather and scatter operations on graphics processors[C]// Proceedings of the ACM/IEEE Conference on Supercomputing, 2007: 1 – 12.

[15] Tan H B, Chen H Y, Liu S, et al. Modeling and evaluation for gather/scatter operations in Vector-SIMD architectures[C]// Proceedings of IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2017.

[16] 吴健虢. 支持 Gather/Scatter 的向量存储器的设计与实现[D]. 长沙: 国防科技大学, 2018.
WU Jianguo. The design and implement of vector memory to support Gather/Scatter[D]. Changsha: National University of Defense Technology, 2018. (in Chinese)

[17] Goumas G, Kourtis K, Anastopoulos N, et al. Performance evaluation of the sparse matrix-vector multiplication on modern architectures[J]. Journal of Supercomputing, 2009, 50(1): 36 – 77.

[18] DuBois D, DuBois A, Connor C, et al. Sparse matrix-vector multiplication on a reconfigurable supercomputer [C]// Proceedings of 16th IEEE International Symposium on Field-Programmable Custom Computing Machines, 2008: 239 – 247.

[19] Davis T A, Hu Y F. The university of Florida sparse matrix collection [J]. ACM Transactions on Mathematical Software, 2011, 38(1): 1 – 25.

[20] 张凯. 向量 SIMD DSP 上高效矩阵运算技术研究[D]. 长沙: 国防科技大学, 2013.
ZHANG Kai. High efficient matrix operations on vector-SIMD DSPs [D]. Changsha: National University of Defense Technology, 2013. (in Chinese)