doi:10.3772/j.issn.1002-0470.2024.10.002

IR-GCN:二值图卷积神经网络推理加速器①

于启航②*** 文渊博* 杜子东③*

(*中国科学院计算技术研究所处理器芯片国家重点实验室 北京 100190)

(**中国科学院大学 北京 100049)

摘 要 针对图卷积神经网络(GCN)中数据规模庞大、不适合边缘端低功耗处理器高效推理计算的问题,本文提出一种将新型二值数据量化算法(IR-Net)应用于 GCN 模型推理计算的方法,并设计了对应的硬件加速器 IR-GCN。同时,针对计算过程中工作负载分布不均衡的问题,实现了一种负载均衡模块设计,显著提高了计算效率。实验结果表明,在较小精度损失范围内,IR-GCN 加速器可以同时降低计算延迟以及访存开销。与现有性能最优的研究相比,IR-GCN 加速器平均具有 2.4 倍的计算加速比、7.9 倍的功耗降低、13.7 倍的芯片面积减少以及 21.0 倍的访存量降低。

关键词 图卷积神经网络(GCN);二值神经网络(BNN);硬件加速器

图卷积神经网络[1](graph convolutional neural network,GCN)是传统卷积神经网络(convolutional neural metwork,CNN)算法在图神经网络[2](graph neural network, GNN)中的应用。GCN 及其变体在处理具有图结构的非欧几里得(non-Euclidean)数据方面取得了非常好的效果[3-6],被广泛应用于推荐算法[7]、知识图谱[8]、社交网络、分子生物学等领域。然而,这类计算通常包含大规模图节点特征和图邻接矩阵数据[9-11]。同时,这些图邻接矩阵通常非常稀疏[12-13],且数据分布不均匀,具有不规则性,不利于实时高效计算和访存,尤其限制了其在边缘端低功耗场景中的应用,如自动驾驶中的路径规划、电网管理中的故障分析[14]以及智能交通中流量控制等[9]。

针对上述问题和挑战,目前出现了很多针对 GCN 算法的硬件加速工作[12,15-16]。然而,这些工作仍局限于使用浮点数据类型,未能有效解决大规模输入的访存和计算开销问题。与之相对的是,二值神经网络[17-18](binary neural network, BNN)算法将特征和权重数据进行 1 比特量化,表示为 +1 和 -1(或 0),用逻辑异或(XNOR)代替浮点乘法,用比特"1"累加(pop-count)代替浮点加法,提高了访存和计算效率,具有硬件友好的特性[19]。基于此,双向图卷积神经网络(bi-directional graph convolutional neural networks, Bi-GCN)[20]算法提出将二值量化方法应用于 GCN 的节点特征表示,在精度损失允许范围内,实现了 GCN 算法在图形处理器(graphics processing unit,GPU)端的高效计算。但是,该工作也存在一定问题。首先,由于 Bi-GCN 使用传统的 XNOR-Net[18]二值量化方法,仍需要引入一定浮点操作,不利于硬件实现,在处理大规模数据时无法满足低延迟低功耗要求[10-11]。其次,由于邻接矩阵中非零元素服从幂律分布[15],会造成计算负载分配不均衡,该方法不能有效提高硬件利用率,会造成硬件实现的性能损失。因此,针对边缘端任务,仍需要设计专用的硬件加速器以满足低延迟和低功耗的要

① 国家重点研发计划(2022YFB4501601),国家自然科学基金(62222214,61925208,U19B2019,U22A2028)和中国科学院稳定支持基础研究领域青年国家团队计划(YSBR-029)资助项目。

② 男,1991 年生,博士生;研究方向:神经网络加速器;E-mail:yuqihang17b@ict.ac.cn。

③ 通信作者,E-mail:duzidong@ict.ac.cn。

(收稿日期:2023-06-15)

求。

根据上述讨论,本文使用一种新型二值量化方法 IR-Net(information retention Net)[21],在算法层面对 Bi-GCN 中的二值量化方法进行了优化,使计算过程具有硬件友好的特性。同时,在硬件实现层面又对 GCN 运算过程中负载分配方式进行了优化,进一步提高了计算效率,本文的主要贡献如下。

(1)将新型二值量化方法 IR-Net[21] 应用于 GCN 计算,并基于此设计了一种二值 GCN 推理计算加速器 IR-GCN(information retention GCN)。通过引入移位操作替代了现有 Bi-GCN 算法中的浮点计算,以牺牲一定精度为代价,用逻辑运算单元代替了浮点运算单元,提高了计算和访存效率。

(2)针对 GCN 算法中计算负载分配不均衡的问题,设计了一种负载均衡模块,可以有效实现计算负载的均衡分配,降低由于邻接矩阵非零元素分布不均匀造成的延迟开销,提高了硬件利用率和计算性能。

(3)将所提出的架构进行硬件实现,并在几种主流的数据集上进行实验。与现有的最优工作比较,IR-GCN 加速器平均具有 2.4 倍的计算加速比、7.9 倍的功耗降低、13.7 倍的芯片面积减少以及 21.0 倍的访存量降低,证明了本文方法的有效性。

1 移位二值量化 GCN 算法

本节首先介绍 GCN 算法的背景知识,并介绍了现有的二值量化 GCN 算法 Bi-GCN[20],在此基础上介绍了本文所提出的基于 IR-Net 的移位二值量化 GCN 算法,给出了相对于 Bi-GCN 算法的具体优化方式。

1.1 GCN 算法一般过程

对于一个图结构输入数据 $G = \{v, \varepsilon\}$,v 为图中节点集合,节点数量为 N,ε 为图中边集合,边的数量为 M。$j \in N(i)$ 表示局部子图的中心节点 i 及其邻节点 j, 邻节点构成的集合为 $N(i)$, 则图卷积计算过程可以用式 (1)表示。

$$\boldsymbol{x}_i^{(l)} = \gamma^{(l)}(\boldsymbol{x}_i^{(l-1)}, \bigoplus_{j \in N(i)} (\phi^{(l)}(\boldsymbol{x}_i^{(l-1)}, \boldsymbol{x}_j^{(l-1)}, \boldsymbol{e}_{j,i}))) \tag{1}$$

其中,$\boldsymbol{x}_i^{(l-1)} \in R^F$ 表示图卷积模型第 $(l-1)$ 层节点 i 特征向量;F 表示节点特征维度;$\boldsymbol{e}_{j,i} \in R^D$ 表示节点 j 到节点 i 的边特征向量,D 表示边特征维度;ϕ 表示节点特征提取函数;$\oplus$表示具有平移不变及可导性质的聚合函数,如 sum、max 或 mean 等;γ 表示可选的节点特征更新函数。其中,ϕ 和 γ 可选用可导函数,如 MLP(multi layer perceptrons)进行特征表示。为方便硬件设计分析,将式 (1)表示成矩阵形式,如式 (2) 所示。

$$\boldsymbol{X}^{l+1} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}) \tag{2}$$

式中,$\hat{\boldsymbol{A}}$ 为邻接矩阵 $\boldsymbol{A}$ 经过归一化的结果,例如原始 GCN[1] 中使用一种 $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ 的方法进行归一化,其中 $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$,$\boldsymbol{I}_N$ 为单位矩阵,$\tilde{D}_{ii} = \sum_j \tilde{\boldsymbol{A}}$;$\boldsymbol{X}^{(l)}$ 为第 l 层节点特征矩阵;$\boldsymbol{W}^{(l)}$ 为对应的权重参数矩阵;$\sigma(\cdot)$ 表示非线性激活函数。

根据式 (2)可以将 GCN 的计算表示为 2 个矩阵乘法的计算,分别为特征提取和特征聚合过程[22]。特征提取过程表示为 $\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}$,指将图上各节点进行特征表示,其结果作为特征聚合过程中的输入。特征聚合过程表示为 $\hat{\boldsymbol{A}}(\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)})$,指将特征提取过程中的结果 $\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}$ 通过邻接矩阵进行消息传递。特征聚合结果再经激活函数 $\sigma(\cdot)$ 得到下一层节点特征表示 $\boldsymbol{X}^{(l+1)}$。除了原始 GCN,不同的图卷积模型对应着不同的归一化邻接矩阵 $\hat{\boldsymbol{A}}$,由于 $\hat{\boldsymbol{A}}$ 可由邻接矩阵 $\boldsymbol{A}$ 计算,后文使用 $\boldsymbol{A}$ 代替 $\hat{\boldsymbol{A}}$。在 GCN 中,$\boldsymbol{A}$ 和 $\boldsymbol{W}^{(l)}$ 可以作为模型的离线参数,主要工作负载为 $\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}$ 和 $\boldsymbol{A}(\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)})$ 2 个过程中的矩阵运算以及非线性激活函数的实现。

1.2 二值量化 GCN 算法 Bi-GCN

为提高 GCN 模型的计算效率,Bi-GCN[20] 算法比较了特征提取阶段和特征聚合阶段的计算量,提出由于特征聚合阶段不包含可学习参数,计算负载和访存开销占比很小,所以计算 $\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}$ 是决定性能的主要因素[20]。因此,Bi-GCN 提出使用 XNOR-Net[18] 方法对 GCN 特征提取过程中的节点特征 $\boldsymbol{X}^{(l)}$ 和权重参数 $\boldsymbol{W}^{(l)}$ 进行二值量化,从而降低了计算和访存开销。表 1 比较了浮点乘累加(multiply accu-

mulate,MAC)运算器和二值 MAC 运算器的硬件开销,使用二值运算代替浮点运算可以降低 95% 的面积和 88% 的功耗。基于此,Bi-GCN 将实数特征和实数权重分别表示为一个二值矩阵与一个实值矩阵的 Hadamard 积,即 $\boldsymbol{X}^{(l)}$ 量化为 $\boldsymbol{\alpha}^{(l)}\mathrm{sign}(\boldsymbol{X}^{(l)})$,$\boldsymbol{W}^{(l)}$ 量化为 $\boldsymbol{\beta}^{(l)}\mathrm{sign}(\boldsymbol{W}^{(l)})$,从而可以用二值计算代替部分浮点运算,减少 $\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}$ 实数矩阵乘法规模。Bi-GCN 算法特征提取过程可以表示为

$$\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)} \approx \boldsymbol{\alpha}^{(l)}\boldsymbol{\beta}^{(l)}(\boldsymbol{B}_X^{(l)} \odot \boldsymbol{B}_W^{(l)}) \tag{3}$$

其中,$\boldsymbol{B}_X^{(l)} = \mathrm{sign}(\boldsymbol{X}^{(l)})$ 和 $\boldsymbol{B}_W^{(l)} = \mathrm{sign}(\boldsymbol{W}^{(l)})$ 为节点特征 $\boldsymbol{X}^{(l)}$ 和权值 $\boldsymbol{W}^{(l)}$ 的二值量化矩阵;$\boldsymbol{\alpha}_i^{(l)} = \frac{1}{N}\|\boldsymbol{X}_{i,:}^{(l)}\|_1$,$\boldsymbol{\beta}_j^{(l)} = \frac{1}{d_{\mathrm{out}}}\|\boldsymbol{W}_{:,j}^{(l)}\|_1$ 为对应的实数缩放系数向量,其中 N 和 d_{out} 表示节点数量和输出特征维度;$\boldsymbol{\alpha}^{(l)}\boldsymbol{\beta}^{(l)}$ 表示实数向量内积运算;⊙表示由 XNOR 和 pop-count 构成的二值内积操作[20]。而在特征聚合过程中,Bi-GCN 仍与传统图卷积算法保持一致。

表 1　浮点 MAC 运算与二值 MAC 运算面积与功耗比较

数据位宽/bit	运算器类型	面积/μm^2	功耗/μW
32	FP Mult&FP Add	12 779	3 730
1	XNOR &pop-count	706	439

虽然 Bi-GCN[20] 算法使用二值量化降低了 $\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}$ 的计算和访存开销,但为了保证精度,仍需要引入浮点操作计算缩放系数与二值计算结果的 Hadamard 积,对于大规模数据仍需要大量计算开销。因此本文在 Bi-GCN 算法基础上,使用基于 IR-Net[21] 的方法实现二值 GCN 量化,通过引入逻辑移位操作,避免了浮点运算,在保证精度的前提下进一步提高了 GCN 算法的推理计算效率。

1.3　基于 IR-Net 的二值量化 GCN 算法

与 Bi-GCN 算法类似,本文同样对 GCN 的特征提取过程 $\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}$ 进行优化,使用 IR-Net[21] 二值量化算法,引入比特移位操作替换了 Bi-GCN 算法中的浮点计算。为了降低二值量化的精度损失,IR-Net[21] 算法同时对正向传播和反向传播过程进行优化:在正向传播中,引入 Libra-PB (Libra parameter binarization) 平衡标准化量化方法,通过最大化权重参数信息熵,使权值参数在(-1,1)区间分布更均匀,降低了二值量化精度损失;在反向传播中,引入 EDE(error decay estimator)方法计算梯度信息,更精确地估计二值激活函数 sign 的梯度,以提高二值量化权重参数的训练精度。实验证明该方法在 CIFAR-10 和 ImageNet 数据集图像分类任务上都能取得良好的效果[21]。具体来说,不同于 Bi-GCN 中将节点特征 $\boldsymbol{X}^{(l)}$ 和权重参数 $\boldsymbol{W}^{(l)}$ 量化为浮点缩放系数和二值矩阵的 Hadamard 积,IR-Net 中将 $\boldsymbol{X}^{(l)}$ 直接量化为 $\mathrm{sign}(\boldsymbol{X}^{(l)})$,而将 $\boldsymbol{W}^{(l)}$ 量化为 $\mathrm{sign}(\boldsymbol{W}_{std}^{(l)}) \ll\gg \boldsymbol{S}^{(l)}$,其中 $\boldsymbol{S}^{(l)}$ 为根据权值 $\boldsymbol{W}^{(l)}$ 计算得到的位移参数,计算公式为

$$\boldsymbol{S}^{(l)} = round\left(\log_2\left(\frac{\|\boldsymbol{W}_{\mathrm{std}}^{(l)}\|_1}{n}\right)\right) \tag{4}$$

式中,$\boldsymbol{W}_{\mathrm{std}}^{(l)} = \frac{\boldsymbol{W}^{(l)} - \boldsymbol{W}_{\mathrm{mean}}^{(l)}}{\sigma(\boldsymbol{W}^{(l)} - \boldsymbol{W}_{\mathrm{mean}}^{(l)})}$,mean 和 σ 分别表示均值和标准差;n 为权重矩阵维度。操作符≪≫表示位移操作[21],$S^{(l)}$ 中元素 >0 表示左移≪,$S^{(l)}$ 中元素 <0 表示右移≫。根据 $\boldsymbol{X}^{(l)}$ 和 $\boldsymbol{W}^{(l)}$ 的二值量化结果则基于 IR-Net 量化的 GCN 特征提取过程可以表示为

$$\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)} \approx (\boldsymbol{B}_X^{(l)} \odot \boldsymbol{B}_W^{(l)}) \ll\gg S^{(l)} \tag{5}$$

式中,$\boldsymbol{B}_X^{(l)} = \mathrm{sign}(\boldsymbol{X}^{(l)})$,$\boldsymbol{B}_W^{(l)} = \mathrm{sign}(\boldsymbol{W}_{\mathrm{std}}^{(l)})$,均为二值矩阵;⊙仍表示二值内积操作;位移操作“≪≫”放在二值计算后再执行[21]。

将式(5)与式(3)比较,可以发现 IR-Net 中通过对二值计算结果 $\boldsymbol{B}_X^{(l)} \odot \boldsymbol{B}_W^{(l)}$ 进行移位操作,代替了 Bi-GCN[20] 中将二值计算结果 $\boldsymbol{B}_X^{(l)} \odot \boldsymbol{B}_W^{(l)}$ 与缩放系数 α 和 β 进行浮点 Hadamard 积计算,避免浮点运算器的使用,具有硬件友好特性。同时 IR-Net 具有良好的通用性,可以直接应用于 GCN 模型量化,因此本文使用 IR-Net[21] 算法实现了一种基于移位操作的二值量化 GCN 算法。该方法在保证计算精度的前提下,相较于使用浮点运算器,能够有效降低访存和计算开销,下一节将对基于该方法实现的硬件加速器进行介绍。

2　移位二值量化 GCN 加速器 IR-GCN

本节首先介绍 IR-GCN 总体架构,再分别对特

征提取单元(feature extraction unit,FEU)、特征聚合单元(feature aggregating unit,FAU)和负载平衡单元(load balance unit,LBU)的设计和主要功能进行介绍。

2.1 总体架构

图1为IR-GCN总体架构,由FEU、FAU和LBU这3部分构成。FEU主要包含二值移位阵列(binary shift array,BSA)、输入特征缓存(input buffer,IBUF)、权值数据缓存(weight buffer,WBUF)和偏移量缓存(shift buffer,SBUF)几个部分;FAU主要包含邻接聚合单元(adjacency aggregating unit,AAU)、邻接矩阵缓存(adjacency matrix buffer,ABUF)、阈值缓存(threshold buffer,TBUF)和输出缓存(output buffer,OBUF);LBU模块包含特征缓存(feature buffer,FBUF)以及负载均衡控制单元(load balance controller,LBC)。

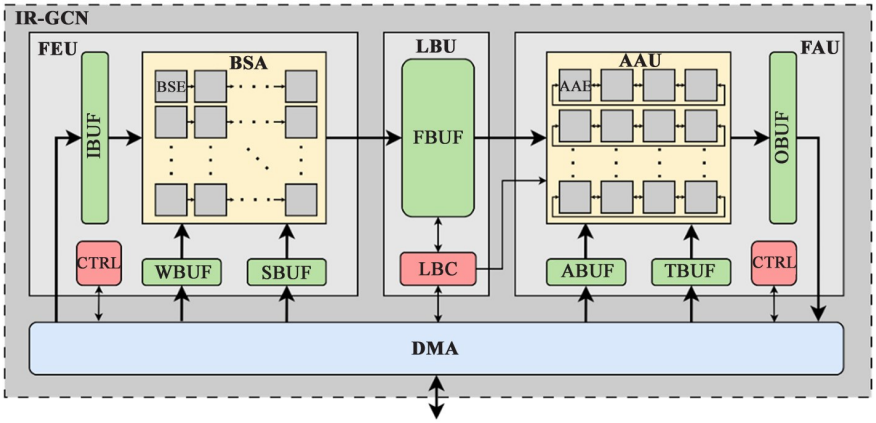

图1 IR-GCN总体架构

在IR-GCN中,FEU负责实现GCN中特征提取功能,输入为节点二值特征 $\boldsymbol{X}_b$、二值权重参数 $\boldsymbol{W}_b$ 以及偏移量 S,输出为整型二值特征提取结果 $\boldsymbol{X}_b\boldsymbol{W}_b$。FAU负责实现GCN中特征聚合及二值激活功能,输入为整型特征 $\boldsymbol{X}_b\boldsymbol{W}_b$、邻接矩阵 $\boldsymbol{A}$ 以及激活阈值 T,输出为二值激活 $\sigma_b(\boldsymbol{A}(\boldsymbol{X}_b\boldsymbol{W}_b))$。LBU负责将FEU写入FBUF的节点特征进行同步,再将同步后的数据发送到FAU,并根据FAU工作状态实现负载平衡功能。IR-GCN中除FBUF外,所有缓存通过直接内存访问(direct memory access,DMA)方式与外部存储进行数据交互。

2.2 特征提取单元 FEU

FEU特征提取功能主要通过BSA实现,如图2所示,其由32×8个比特移位单元(bit shift element,BSE)构成,使用脉动阵列方式排列,仅最左侧一列BSE与IBUF作数据交互,输入 $\boldsymbol{X}_b$ 从左至右在BSE间传递,其他BSE无需额外访问IBUF,减少了访存带宽。为了将二值特征提取计算通过逻辑运算单元实现,计算过程中使用文献[23]的方式对输入数据进行转换:原二值输入中的+1仍用+1表示,将-1

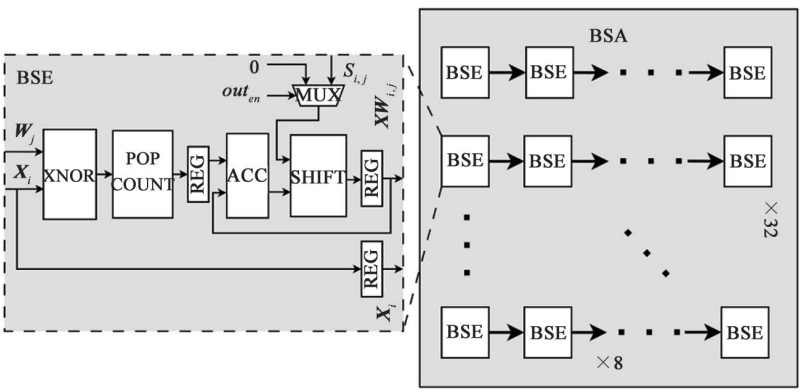

图2 BSA及基本单元BSE设计

用 0 表示,得到转换后的二值输入 $\boldsymbol{X}_b^{'}\boldsymbol{W}_b^{'}$。式(5)中的计算过程,经过数据转换后,可以用式(6)表示。

$$\boldsymbol{XW} = (2 \times popcount(xnor(\boldsymbol{X}_b^{'}\boldsymbol{W}_b^{'})) - n) \ll\gg S \tag{6}$$

式中,n 表示特征维度,作为模型参数在 FAU 中会融合为阈值参数,不需要引入额外的计算。通过上述数据转换,GCN 二值特征提取运算最终可以转化为逻辑 XNOR 和 pop-count 以及移位运算。为了简便,后文仍用 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 表示转换后的二值输入。

根据式(6),BSE 实现了二值特征提取计算功能,包含 XNOR 逻辑操作单元、pop-count 计算单元、ACC 累加和 SHIFT 移位单元。每个 BSE 输入位宽设置为 8 bit,表示将二值输入 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 分别按行和列进行 8 bit 分块,每个数据块包含 8 个 1 bit 数据元素。分块大小影响脉动阵列计算延迟、访存带宽以及硬件资源利用率,可以根据测试数据集使用的 GCN 模型规模设置。根据本文使用的 GCN 模型规模,综合考虑性能和计算效率,仅以 8 bit 作为例子进行讨论。每个周期内,各 BSE 从左侧相邻的 BSE(或 IBUF)读取输入特征 $\boldsymbol{X}$ 某行中 8 bit 数据块,且该数据块在该行 BSE 中由左至右依次传递,实现了对 $\boldsymbol{X}$ 按行复用。同一周期内,该 BSE 同步从 WBUF 中读取权值参数 $\boldsymbol{W}$ 某列中 8 bit 数据块,该数据块在该列 BSE 中共享,实现了对 $\boldsymbol{W}$ 按列复用。对输入 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 的复用降低了整个 BSA 的访存带宽,提高了访存效率。输入数据经过 XNOR 和 pop-count 计算单元后,进入累加单元 ACC 进行累加完成二值矩阵乘法 $\boldsymbol{XW}$ 计算,每个 BSE 的输出为 $\boldsymbol{XW}$ 矩阵中的一个元素。当输入特征全部经过 BSE 后,控制单元负责从 SBUF 读取各 BSE 对应的偏移量参数 $\boldsymbol{S}$,SHIFT 移位单元根据该参数进行移位操作,从而完成式(6)计算。最后,每行 BSE 将计算结果写入对应的 FBUF。

2.3 特征聚合单元 FAU

FAU 特征聚合单元实现邻接矩阵 $\boldsymbol{A}$ 与 FEU 计算结果 $\boldsymbol{XW}$ 的矩阵乘法操作,并计算二值特征激活,输出最终结果 $\sigma(\boldsymbol{A}(\boldsymbol{XW}))$。FAU 计算功能主要通过 AAU 实现,其设计如图 3 所示。AAU 包含 8×4 个邻接聚合单元(adjacency aggregating element,AAE),各行 AAE 使用环形方式进行连接,每个 AAE 包含整型介质访问控制(media access control,MAC)单元和阈值比较单元 COMP(comparator)。每个周期内,每列 AAE 读取邻接矩阵 $\boldsymbol{A}$ 某一行中 1 个非零元素(不同列 AAE 对应邻接矩阵 $\boldsymbol{A}$ 中不同行),并同步读取该非零元素对应的 $\boldsymbol{XW}$ 矩阵中一行元素(每个 AAE 读取 $\boldsymbol{XW}$ 矩阵该行中的一个元素)。$\boldsymbol{A}$ 中同一个非零元素在每一列 AAE 中共享,减少了 AAE 对 ABUF 的访存带宽。AAE 经过 MAC 运算和阈值比较运算,将 1 bit 二值激活结果写入 OBUF,每个 AAE 对应的激活阈值可通过离线计算得到。每列 AAE 的计算结果对应二值输出特征矩阵 $\boldsymbol{A}(\boldsymbol{XW})$ 的一行元素。每行 AAE 以环形方式进行连接的原因是,当存在需要负载均衡的情况,参与负载均衡的各列 AAE 完成计算后,需要将 MAC 部分和数据(含左侧 AAE 的部分和 sum_l 及右侧的 sum_r)写回同一列 AAE,最后由该列 AAE 进行激活输出。

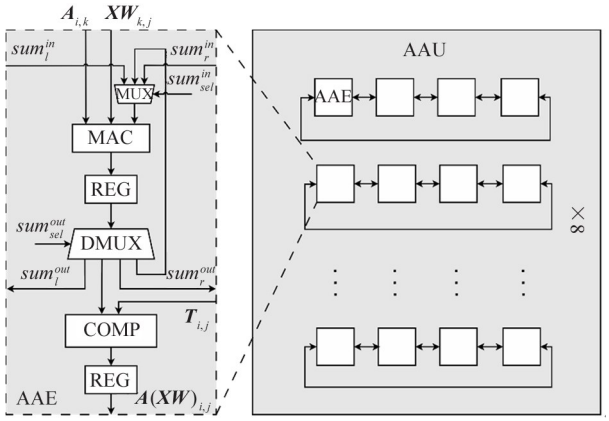

图 3 AAU 及基本单元 BSE 设计

2.4 负载均衡单元 LBU

LBU 设计如图 4 所示,其中 4 个 FBUF 队列用于存储 4 列 AAE 的输入,即 FEU 中输出的特征矩阵 $\boldsymbol{XW}$。LBC 模块控制 FBUF 完成数据同步,并控制 AAU 完成负载均衡。同步功能通过 LBC 中的 FBUF 工作状态寄存器完成,LBC 根据该寄存器状态控制 FBUF 数据读取,并将同步后的 $\boldsymbol{XW}$ 输出到 AAU 进行特征聚合操作。通过数据同步,将每一列 AAE 的输入数据保持同步,从而可以共享相同的邻接矩阵非零元素输入,降低了 AAE 对 ABUF 的访存

带宽,并简化了 AAU 的控制复杂度。

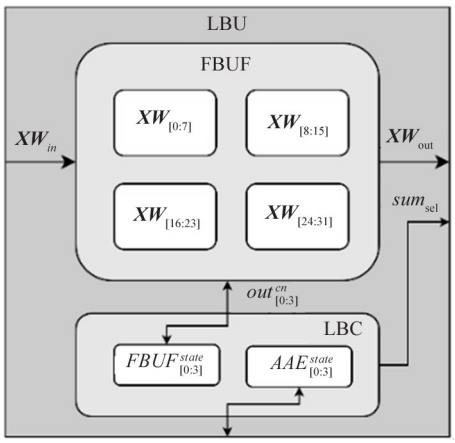

图 4　负载均衡单元 LBU 设计

引言中提到由于邻接矩阵 $\boldsymbol{A}$ 元素服从幂律分布,造成 $\boldsymbol{A}$ 中少数行的元素特别稠密而大部分行元素极为稀疏。图 5 表示了 3 个数据集中邻接矩阵各行非零元素的分布情况,图中大部分邻接矩阵行只有少量非零元素(2 ~ 20 个),而少数的行则出现了大量非零元素(大于 100 个)。由于 $\boldsymbol{A}$ 数据分布不均匀,计算时会出现某一列 AAE 尚未计算完成(工作负载为稠密行),而其他列 AAE 空闲的情况(工作负载为稀疏行),造成 AAE 工作负载不均衡。LBC 通过检测 AAE 的工作状态,若某列 AAE 出现空闲,则将工作中 AAE 尚未完成的计算负载平均分配给空闲的 AAE,达到平衡工作负载、提高 AAE 利用率的目的。LBC 负载均衡功能通过 AAE 状态寄存器完成,寄存器中相同的索引表示相同的工作负载($\boldsymbol{A}$ 中同一行),进而控制具有相同工作负载的各列 AAE 将中间结果进行聚合,完成多列 AAE 对 $\boldsymbol{A}$ 中同一行数据的计算。某列 AAE 完成特征聚合计算后,LBC 会清空对应的状态数据,并根据 FBUF 状态设置该列 AAE 新的状态(需要等待 FBUF 中数据同步)。

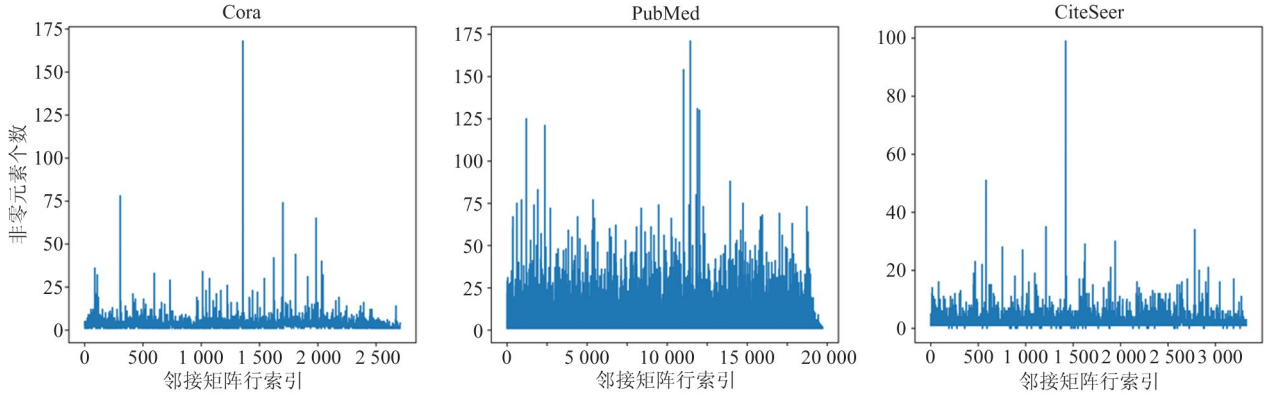

图 5　GCN 数据集邻接矩阵各行非零元素分布

3　IR-GCN 计算过程

本节首先对 IR-GCN 加速器 2 个主要的计算过程——特征提取和特征聚合进行介绍,包括数据在 FEU 和 FAU 上的扩展方式。然后介绍 LBU 单元数据同步和负载均衡的具体实现方式。IR-GCN 数据的映射方式如图 6 所示,虚线框标出了计算一行邻接矩阵 $\boldsymbol{A}$ 对应结果所需要的输入数据(简化了偏移量 $\boldsymbol{S}$ 和激活阈值 $\boldsymbol{T}$)及计算单元,即输入数据为 2 行 $\boldsymbol{X}$ 和整个 $\boldsymbol{W}$ 矩阵,计算单元为 2 行 BSE 和 1 列 AAE。$\boldsymbol{X}$ 和 $\boldsymbol{W}$ 中相同颜色的数据块表示 1 个 BSE 中同一周期的输入,每个 BSE 计算 1 行 $\boldsymbol{X}$ 和 1 列 $\boldsymbol{W}$ 的对应结果,每行 BSE 计算 1 行 $\boldsymbol{X}$ 和整个 $\boldsymbol{W}$ 矩阵的对应结果,每列 AAE 计算邻接矩阵一行中非零元素对应的特征聚合结果。

3.1　FEU 特征提取计算过程及扩展方式

FEU 输入为 N 个节点二值特征矩阵 $\boldsymbol{X}$,维度为 $[N,D_{\text{in}}]$(D_{in} 为输入节点特征维度),还包括 $\boldsymbol{X}$ 对应的权重矩阵 $\boldsymbol{W}$,维度为 $[D_{\text{in}},D_{\text{out}}]$($D_{\text{out}}$ 为输出节点特征维度),以及偏移量矩阵 $\boldsymbol{S}$,维度为 $[N,D_{\text{out}}]$。输出为特征提取计算结果 $\boldsymbol{XW}$,维度为 $[N,D_{\text{out}}]$,与偏移量矩阵 $\boldsymbol{S}$ 中的元素一一对应。根据 2.2 节的介绍,计算过程中,分别对 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 按行和按列进行 8 bit分块,各 BSE 对读取 $\boldsymbol{X}$ 的行数据块及 $\boldsymbol{W}$ 的列数

据块进行计算(对应图 6 中 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 相同颜色数据块),数据以流水方式经过各 BSE,并将结果累加保存在部分和寄存器内。计算完成后 BSE 将计算结果 $\boldsymbol{XW}$ 写入到 FBUF 队列,每个 FBUF 队列对应一列 AAE 的输入。为了提高输入数据的复用率,降低访存量,需要按邻接矩阵 $\boldsymbol{A}$ 非零元素索引读取对应的 $\boldsymbol{X}$ 和 $\boldsymbol{W}$,4.3.3 节实验分析了输入按序访问对访存量的影响。

FEU 运算单元支持输入数据在 3 个维度进行扩展。首先,在 D_{in} 维度方向,BSE 每个周期可以处理 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 矩阵 D_{in} 维度中 8 个数据元素(一个数据块),则完成 $\boldsymbol{XW}$ 中一个数据元素计算需要 BSE 执行 $D_{in}/8$ 个周期。因此,可以通过增加计算周期实现对 D_{in} 维度扩展。其次,在 D_{out} 维度方向,每列 BSE 计算 $\boldsymbol{XW}$ 矩阵中一列结果,且不同列 $\boldsymbol{XW}$ 计算不存在依赖,因此可以将 $\boldsymbol{XW}$ 在 D_{out} 维度(按列)划分成多个数据块逐块计算。最后,在 N 维度方向,N 大小仅影响某列 AAE 执行特征聚合计算的 MAC 次数,通过增加每行 BSE 计算的 $\boldsymbol{XW}$ 矩阵行数,并增加 AAE 计算时间即可支持对 N 的扩展。

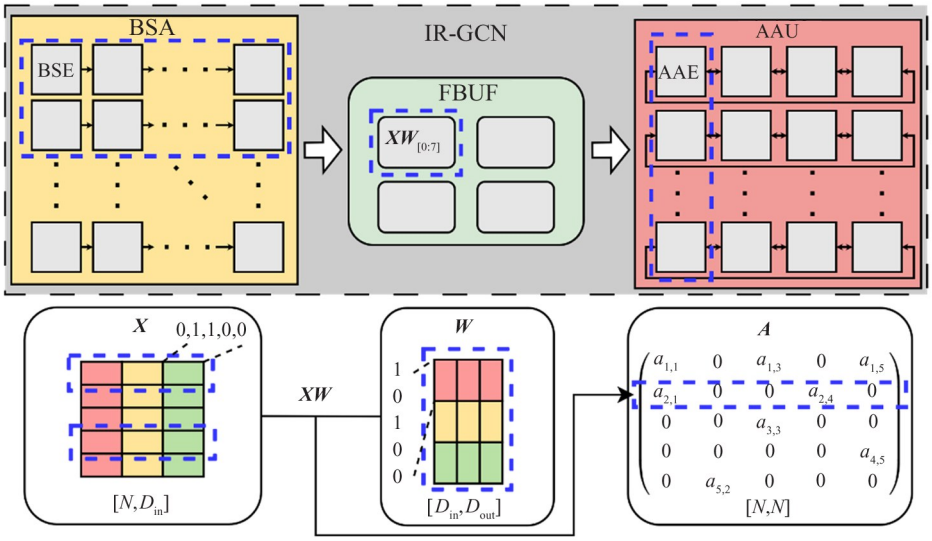

图 6　IR-GCN 加速器数据映射方式

3.2　FAU 特征聚合计算过程及扩展方式

FAU 输入为特征提取结果矩阵 $\boldsymbol{XW}$,维度为 $[N,D_{out}]$,还包括稀疏图邻接矩阵 $\boldsymbol{A}$,维度为 $[N,N]$,以及激活阈值矩阵 $\boldsymbol{T}$,维度为 $[N,D_{out}]$。输出为二值激活矩阵 $\boldsymbol{\sigma}(\boldsymbol{A}(\boldsymbol{XW}))$,维度为 $[N,D_{out}]$ 与 $\boldsymbol{T}$ 中元素一一对应。计算过程中,每个周期内各列 AAE 同步执行 $\boldsymbol{XW}$ 矩阵中一行数据元素与 $\boldsymbol{A}$ 中同一个非零元素的 MAC 运算,并将结果保存在部分和寄存器中。当完成 $\boldsymbol{A}$ 一行中所有非零元素的计算后,MAC 结果输入阈值比较单元 COMP 计算二值激活。数据以流水线方式经过各运算单元,每列 AAE 的计算是同步进行的。

FAU 支持输入在 N 和 D_{out} 2 个维度进行扩展。首先,在 D_{out} 方向,对 $\boldsymbol{A}(\boldsymbol{XW})$ 进行分块,每列 AAE 每次输入一个数据块(AAE 的列方向对应 $\boldsymbol{A}(\boldsymbol{XW})$ 的行方向),从而完成在 D_{out} 维度的扩展。另一方面,在 N 方向,对 $\boldsymbol{A}(\boldsymbol{XW})$ 进行分块,不同行的计算不存在数据依赖,$\boldsymbol{A}(\boldsymbol{XW})$ 中每一行数据都对应一列 AAE。扩展 N 仅需增加每列 AAE 输入的邻接矩阵行数,AAU 同时可以对 4 行 $\boldsymbol{A}$ 数据进行特征聚合操作。

3.3　LBU 数据同步和负载均衡过程

LBU 中 FBUF 的输入为特征提取过程结果 $\boldsymbol{XW}$,维度为 $[N,D_{out}]$,LBC 输入为 FBUF 同步状态标志及 4 列 AAE 对应的邻接矩阵 $\boldsymbol{A}$ 中 4 个行索引。LBC 控制 FBUF 完成 $\boldsymbol{XW}$ 数据同步,并控制 AAE 执行负载均衡操作。如图 7 所示,$\boldsymbol{XW}$ 同步过程与 FEU 计算过程以及 AAU 计算过程重叠,避免了额外的延迟开销。通过数据同步过程,保证了每列 AAE 的同步计算和访存,降低了访存带宽以及 AAE 控制

复杂度。

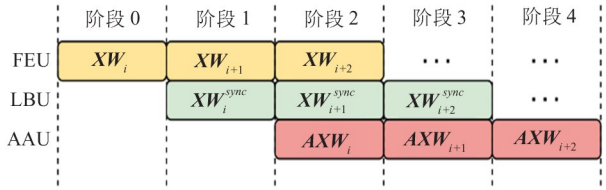

图 7　IR-GCN 执行流水线示意图

对于负载均衡过程,当某列 AAE 尚未完成邻接矩阵 $\boldsymbol{A}$ 某个稠密行的计算而 AAU 存在其他空闲的 AAE 队列时,LBC 会将工作中 AAE 计算的邻接矩阵行索引写到空闲的 AAE 状态寄存器。DMA 通过读取该状态寄存器,将稠密行剩余的工作负载发送到 IBUF 供 FEU 进行计算,同时也就对应空闲的各列 AAE,使得这些空闲的 AAE 可以参与计算。最后,当 AAE 计算完成后,LBC 根据状态寄存器控制具有相同索引的 AAE 队列将结果写回到同一列 AAE,从而对多列 AAE 结果进行聚合,共同完成 $\boldsymbol{A}$ 中同一行的计算。通过负载均衡,避免了出现某列 AAE 处理 $\boldsymbol{A}$ 中某个稠密行时,该列 AAE 长时间占用而其余 AAE 队列空闲的情况,提高了 FAU 在特征聚合过程中的计算性能。

表 2　实验使用的数据集参数

数据集	节点数	边数	标签类别	特征维度	稀疏度/%
Cora	2 708	10 556	7	1 433	0.14
CiteSeer	3 327	9 104	6	3 703	0.08
PubMed	19 717	88 648	3	500	0.02

4　实验结果及分析

本节主要介绍实验环境并对实验结果进行分析。4.1 节给出了实验环境配置及基准测试对象选取方式,4.2 节将 IR-GCN 算法在各数据集上的计算精度与其他二值 GCN 算法进行了比较;4.3 节给出了 IR-GCN 架构硬件实现的性能数据,并对实验结果进行了分析。

4.1　实验环境设置及基准测试对象介绍

4.1.1　硬件实验环境介绍

实验中使用 VerilogHDL 实现了 IR-GCN 架构的硬件设计,使用 Synopsys VCS 工具对计算过程进行功能仿真,得到推理计算延迟、硬件利用率及访存开销等性能数据。使用 SynopsysDesignCompiler 工具对设计进行逻辑综合,评估面积结果,并用 Synopsys PrimeTime PX 估计能耗结果。在 TSMC 45 nm 工艺节点下,加速器主频设置为 1 GHz。

4.1.2　算法基准测试对象及模型参数设置

实验中使用图卷积算法中 3 个最具代表性的引文网络数据集作为基准测试集,表 1 给出了数据集各项参数信息。为了评估计算精度,实验选取了全精度 GCN 以及现有的 2 种二值量化 GCN 模型 Bi-GCN[20] 和 BNN-GCN[24](相当于 IR-GCN 无移位操作)作为基准模型。为了保证计算精度,二值 GCN 模型需要更多的卷积层数和隐层特征维度[19]。因此实验中将全精度 GCN 模型层数设置为 2,隐层特征维度设置为 64;所有二值 GCN 模型层数设置为 3,隐层特征维度设置为 128。所有全精度模型和二值量化模型均使用相同的训练超参数。

4.1.3　硬件基准测试对象选取

为评估 IR-GCN 架构的性能和计算效率,实验中使用性能最优的图计算框架 PyTorch Geometric[22](PyG)为基准,将该框架在 Intel Xeon Gold 6142 CPU(PyG-CPU)和 NVIDIA V100 GPU(PyG-GPU)2 个平台上的性能数据以及 2 个现有性能最优的 GCN 硬件加速器 Hy-GCN[12] 和 GCoD[13] 作为基准硬件测试对象,比较了 IR-GCN 与测试对象的计算延迟以及访存开销等数据,分析了 IR-GCN 的计算和访存效率。

4.2　算法精度结果及分析

表 3 给出了 IR-GCN 模型以及另外 2 种二值量化模型和浮点模型的精度数据,相较于全精度 GCN 模型,IR-GCN 在 3 个数据集上平均仅有 2.20% 的精度损失。而相对于需要一定量浮点操作的 Bi-

GCN 二值量化算法,平均精度损失仅为 0.27%,并在 CiteSeer 数据集上取得了 0.30% 的精度提升。同时相对于未经移位操作的 IR-GCN 模型,IR-GCN 平均获得了 1.50% 的精度提升。因此,实验证明了 IR-GCN 算法在精度方面的有效性。另一方面,由于 IR-GCN 使用逻辑操作代替了全精度 GCN 和 Bi-GCN 中包含的浮点操作,可以极大地提高计算效率。与使用浮点运算单元完成一个乘累加操作的时间相比,相同时间内,使用逻辑运算单元可以执行 64 个二值运算操作[18],同时需要的访存量也可降低约 30 倍[20]。相对于二值量化模型取得的加速比和访存效率提升,IR-GCN 模型在精度方面的损失是可以接受的。

表 3 不同二值 GCN 模型在各数据集上的精度比较

模型	精度/%		
	Cora	CiteSeer	Pubmed
GCN	80.1 ±0.4	68.7 ±0.5	78.7 ±0.1
Bi-GCN	79.4 ±0.5	64.9 ±0.7	77.4 ±0.4
IR-GCN(无移位)	77.3 ±0.9	63.5 ±1.4	75.5 ±1.2
IR-GCN	79.1 ±0.8	65.2 ±0.9	76.6 ±0.8

4.3 硬件实现结果及分析

4.3.1 硬件功耗面积结果

表 4 给出了 IR-GCN 硬件实现具体的面积、功耗数据,主要包含各子模块中运算单元、片上 buffer 以及控制单元的面积和功耗开销。FEU、FAU 和 LBU 中 buffer 大小分别为 4 kB、1 kB 和 8 kB。其中,特征提取单元 FEU 占据了整个架构主要的面积(>84%)和功耗(>87%)开销,而 FEU 中的计算资源又为其中的主要部分。这是由于 GCN 中特征提取过程包含了主要的计算量[15,20],相较于 FAU 和 LBU 具有更大的计算负载。实现中,为保证 FEU 高度的并行性,使用了大量计算单元(32×8 个二值移位计算单元 BSE),同时需要使用大量片上 buffer 缓存计算过程中的输入特征、权值参数、偏移量等数据,造成了大量资源开销。但另一方面,由于 IR-GCN 使用二值量化方式,相对于浮点运算单元,仍降低了大量硬件资源开销,并获得了更高的计算加速比。在特征聚合过程中,由于计算仅针对邻接矩阵中的非零元素,相较于 FEU 中大量的资源开销,FAU 中仅需要 8×4 个 AAE 和少量的片上 buffer。其中计算单元 AAU 占据了主要开销,在保证计算精度的前提下,AAU 通过使用定点运算单元进一步提高了计算效率。在负载均衡单元 LBU 中,由于需要存储特征提取过程输出的中间结果,因此片上缓存 FBUF 占据了主要的硬件开销。在 IR-GCN 3 个主要模块中,与运算单元相比,控制单元均具有较小开销。这是由于 IR-GCN 根据邻接矩阵 $\boldsymbol{A}$ 非零元素的索引访问对应的输入特征和权值参数,对数据访问可以流水执行,并在计算过程中,对输入数据同步访问,简化了控制开销,更利于硬件实现。整个 IR-GCN 模块的面积约为 2.27 mm^2,功耗约为 1.1 W,相较于 12 nm 工艺实现的 Hy-GCN,根据文献[25]给出的参数缩放到同一工艺尺度下,面积降低约

表 4 IR-GCN 加速器各部分功耗面积结果

模块名称	子模块	面积/mm^2	面积/%	功耗/mW	功耗/%
FEU	BSA	1.76	77.5	719.5	66.2
	Buffer	0.13	5.7	221.9	20.4
	Control	0.02	0.9	8.7	0.8
FAU	AAU	0.12	5.3	70.9	6.5
	Buffer	0.02	0.9	33.1	3.0
	Control	0.01	0.4	7.9	0.7
LBU	Buffer	0.18	7.9	14.3	1.3
	LBC	0.03	1.3	10.6	1.0
IR-GCN 总计		2.27	100.0	1 086.9	100.0

13.7 倍,功耗降低约 7.9 倍,且具有更高的加速比,证明了 IR-GCN 硬件实现的高效性。

4.3.2 硬件加速比分析

图 8 给出了在各数据集上,各平台相对于 PyG-CPU 的硬件加速比。相较于通用计算平台 PyG-CPU 和 PyG-GPU,IR-GCN 平均取得了 2 420 倍和 337 倍的加速比。而相对于 GCN 专用加速器 Hy-GCN[12]和 GCoD[13],IR-GCN 平均仍可以取得 16 倍和 2 倍的加速器比。IR-GCN 的加速比主要来源于以下 3 个方面。首先,二值量化提高了运算单元计算效率,降低了硬件开销,相对于使用浮点运算单元,允许计算过程具有更高的并行度,同时降低了运算单元的计算延迟[18]。其次,IR-GCN 通过数据同步,将特征提取和特征聚合过程流水执行,将 2 个过程计算延迟进行重叠,降低了总的延迟开销,同时简化了数据调度方式。最后,IR-GCN 通过负载平衡功能提高了运算单元利用率,避免了邻接矩阵非零元素分布不均匀对特征聚合过程造成的影响,进一步提高了计算效率。

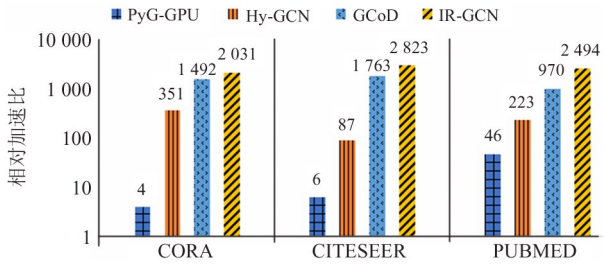

图 8 各平台在 3 个数据集上相对于 PyG-CPU 的推理计算加速比

图 9 给出了引入负载均衡单元 LBU 前后,IR-GCN 总的执行周期数和特征聚合单元 FAU 中计算单元利用率数据。实验结果表明,通过引入 LBU,在 3 个数据集中,IR-GCN 总的执行周期数平均可以降低为原来的 68%,FAU 计算单元利用率平均提升了约 20%,证明了 LBU 可以显著提高 IR-GCN 计算效率,进一步提高了总体硬件加速比。

4.3.3 访存带宽及访存量分析

为了评估 IR-GCN 的访存效率,实验中统计了 IR-GCN 硬件实现在各数据集上所需的访存带宽及访存量,并与Hy-GCN[12]和GCoD[13]中的结果进行了比较,访存带宽和访存量的实验结果分别如图 10 和图 11 所示。对于访存带宽,与 Hy-GCN 相比,IR-GCN 平均仅需要 41% 访存带宽,证明了 IR-GCN 通过对数据进行二值量化以及对输入数据 $\boldsymbol{X}$、$\boldsymbol{W}$ 及 $\boldsymbol{A}$ 的复用,可以有效降低访存带宽。而与 GCoD 相比 IR-GCN 虽然在小的数据集 Cora 和 CiteSeer 上具有更大的带宽开销,而在更大的数据集 PubMed 上仍可以降低访存带宽。这是由于 IR-GCN 具有更高的计算并行度,需要更大的访存带宽,但所需带宽不随数据集规模明显变化,二值量化和数据复用所带来的带宽收益在大的数据集上更为明显。对于访存量,与IR-GCN相比,Hy-GCN[12]和GCoD[13]平均所

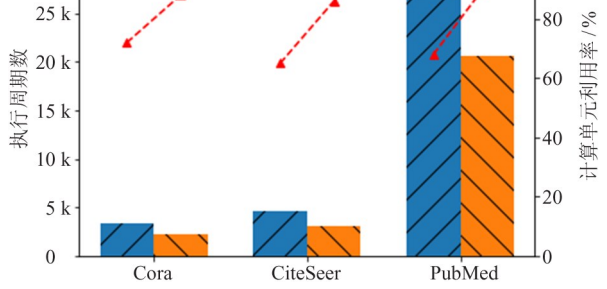

图 9 负载均衡单元对执行周期数和硬件利用率的影响

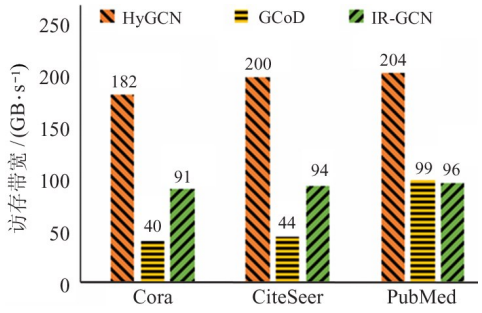

图 10 各平台在不同数据集上的访存带宽

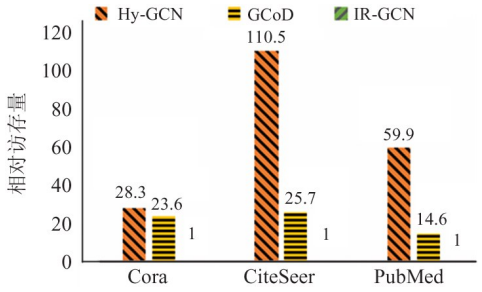

图 11 各平台在不同数据集上的相对访存量

需访存量为 66 倍和 21 倍。IR-GCN 通过二值量化理论上可将访存量降低约为原来的 1/30[18,20]，但为提升二值量化精度，需要对模型规模进行一定扩展，增加了访存量。同时，IR-GCN 根据邻接矩阵非零元素索引按序读取 $\boldsymbol{X}$ 和 $\boldsymbol{W}$，可以提高数据复用率，避免对数据进行重复访问，进一步降低了所需的访存量。

图 12 表示对输入数据 $\boldsymbol{X}$ 和 $\boldsymbol{W}$ 按邻接矩阵 $\boldsymbol{A}$ 非零元素索引顺序访问前后，对片外访存量的影响。实验结果表明，在 3 个数据集上对数据按序访问平均可以将访存量降低为原来的 42%，带来的访存量收益在大的数据集 PubMed 上尤为明显。通过对输入数据按序访问，避免了由于邻接矩阵 $\boldsymbol{A}$ 非零元素分布不均匀而导致的输入数据无法复用而造成额外的访存开销，提高了访存效率，保证了整个流水线高效并行执行。总体而言，相对于现有性能最优的工作，IR-GCN 架构在获得较大加速比的同时可以有效降低访存带宽和访存量。

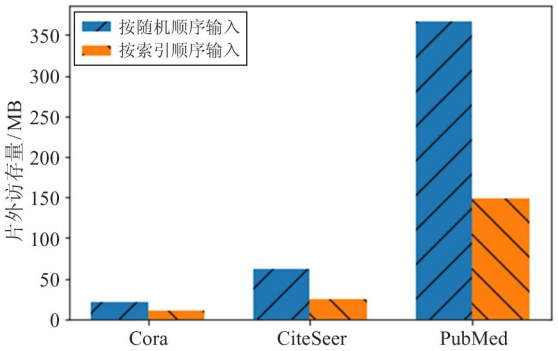

图 12　输入排序前后对片外访存量影响

5　结　论

本文针对图卷积神经网络（GCN）中数据规模庞大、不适合边缘端低功耗处理器高效计算的问题，提出了一种硬件友好且不明显损失精度的新型二值图卷积算法，并设计实现了对应的推理计算加速器 IR-GCN。该架构支持 GCN 模型以任意规模进行扩展，可以高效执行 GCN 推理计算。同时针对图卷积计算中工作负载分配不均衡的问题，设计了相应的负载均衡单元 LBU 以提高计算性能和资源利用率。在实验环节中对算法精度进行了评估，并对 IR-GCN 架构进行了硬件综合实现。与现有的 GCN 专用硬件加速器相比，IR-GCN 在保证计算精度的前提下同时获得了更高的加速比和更低的访存开销，与最优的专用加速器相比平均具有 2.4 倍的计算加速比、7.9 倍的功耗降低、13.7 倍的芯片面积减少以及 21.0 倍的访存量降低。

参考文献

[1] KIPF T N, WELLING M. Semi-supervised classification with graph convolutional networks[EB/OL]. (2017-02-22) [2023-02-15]. https://arxiv.org/pdf/1609.02907.

[2] SCARSELLI F, GORI M, TSOI A C, et al. The graph neural network model[J]. IEEE Transactions on Neural Networks, 2009,20(1):61-80.

[3] HAMILTON W L, YING R, LESKOVEC J. Inductive representation learning on large graphs-GraphSAGE[EB/OL]. (2018-09-10)[2023-02-16]. https://arxiv.org/pdf/1706.02216.

[4] XU K, HU W, LESKOVEC J, et al. How powerful are graph neural networks? -GIN[EB/OL]. (2019-02-22)[2023-02-16]. https://arxiv.org/pdf/1810.00826.pdf.

[5] LI G H, MüLLER M, THABET A, et al. DeepGCNs: can GCNs go as deep as CNNs? [C]//2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea: IEEE, 2019:9266-9275.

[6] CORSO G, CAVALLERI L, BEAINI D, et al. Principal neighbourhood aggregation for graph nets [EB/OL]. (2020-06-07) [2023-02-16]. https://arxiv.org/pdf/2004.05718v3.

[7] 罗可劲, 刘广聪, 杨文浩. 基于多任务学习的图神经网络推荐模型研究[J]. 计算机工程与科学, 2023,45(4):726-733.

[8] 孙水发, 李小龙, 李伟生, 等. 图神经网络应用于知识图谱推理的研究综述[J]. 计算机科学与探索, 2023,17(1):27-52.

[9] LIU X, YAN M, DENG L, et al. Survey on graph neural network acceleration: an algorithmic perspective [EB/OL]. (2022-04-22) [2023-02-16]. https://arxiv.org/pdf/2202.04822v2.

[10] ZHOU J, CUI G, HU S, et al. Graph neural networks: a review of methods and applications[EB/OL]. (2021-04-08) [2022-08-30]. https://arxiv.org/ftp/arxiv/papers/1812/1812.08434.

[11] WU Z, PAN S, CHEN F, et al. A comprehensive survey on graph neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021,32(1):4-24.

[12] YAN M, DENG L, HU X, et al. HyGCN: a GCN accelerator with hybrid architecture [C]//2020 IEEE Interna-

tional Symposium on High Performance Computer Architecture (HPCA). San Diego, USA: IEEE, 2020:15-29.
[13] YOU H, GENG T, ZHANG Y, et al. GCoD: graph convolutional network acceleration via dedicated algorithm and accelerator co-design[C]//2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA). Seoul, Korea: IEEE, 2022:460-474.
[14] 张晓华, 袁肖赟, 杨红英, 等. 基于GNN的电网动态特性评估及其知识图谱应用[J]. 华中科技大学学报(自然科学版), 2023,51(3):47-51, 59.
[15] GENG T, LI A, SHI R, et al. AWB-GCN: a graph convolutional network accelerator with runtime workload rebalancing[C]//2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Athens, Greece: IEEE, 2020:922-936.
[16] AUTEN A, TOMEI M, KUMAR R. Hardware acceleration of graph neural networks[C]//2020 57th ACM/IEEE Design Automation Conference (DAC). San Francisco, USA: IEEE, 2020:1-6.
[17] COURBARIAUX M, HUBARA I, SOUDRY D, et al. Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1 [EB/OL]. (2016-03-17) [2023-02-16]. https://arxiv.org/pdf/1602.02830.
[18] RASTEGARI M, ORDONEZ V, REDMON J, et al. XNOR-Net: imagenet classification using binary convolutional neural networks[EB/OL]. (2016-08-02)[2023-02-16]. https://arxiv.org/pdf/1603.05279.
[19] QIN H T, GONG R H, LIU X L, et al. Binary neural networks: a survey[EB/OL]. (2020-03-31)[2023-02-16]. https://arxiv.org/pdf/2004.03333.
[20] WANG J, WANG Y, YANG Z, et al. Bi-GCN: binary graph convolutional network[C]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Nashville, USA: IEEE, 2021:1561-1570.
[21] QIN H, GONG R, LIU X, et al. Forward and backward information retention for accurate binary neural networks (IR-Net)[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, USA: IEEE, 2020:2247-2256.
[22] FEY M, LENSSEN J E. Fast graph representation learning with pytorch geometric[EB/OL]. (2019-04-25)[2023-03-25]. https://arxiv.org/pdf/1903.02428.pdf.
[23] UMUROGLU Y, FRASER N J, GAMBARDELLA G, et al. FINN: a framework for fast, scalable binarized neural network inference[EB/OL]. (2016-12-01)[2023-03-25]. https://arxiv.org/pdf/1612.07119.
[24] HUBARA I, COURBARIAUX M, SOUDRY D, et al. Binarized neural networks[C]//Proceedings of the 30th International Conference on Neural Information Processing Systems. New York, USA: Curran Associates Inc, 2016: 4114-4122.
[25] VILLA O, JOHNSON D R, OCONNOR M, et al. Scaling the power wall: a path to exascale[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New Orleans, USA: IEEE, 2014:830-841.

IR-GCN: binary graph convolutional neural network inference accelerator

YU Qihang[*,**], WEN Yuanbo[*], DU Zidong[*]
([*]State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
([**]University of Chinese Academy of Sciences, Beijing 100049)

Abstract

This article proposes a method of applying a novel binary quantization algorithm IR-Net to the inference calculation of graph convolutional neural networks (GCN) and designing the corresponding hardware accelerator IR-GCN to address the problem of the large-scale data in GCN, which is not suitable for efficient inference calculation on low-power edging processors. Additionally, to address the issue of uneven workload distribution during the computation process, a load balancing module is implemented, which significantly improves the calculation efficiency. The experimental results show that the IR-GCN algorithm can reduce both calculation delay and memory access overhead simultaneously within a small range of precision loss. Compared with the existing state-of-art research, the IR-GCN accelerator has an average of 2.4 times calculation acceleration, 7.9 times power reduction, 13.7 times chip area reduction, and 21.0 times memory access reduction.

Key words: graph convolutional neural network (GCN), binary neural network (BNN), hardware accelerator