计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202220562
60(10): 2405−2417, 2023

# 面向新一代国产异构众核处理器的数据流计算系统

肖 谦[1] 赵美佳[2] 李名凡[1] 沈 莉[1] 陈俊仕[1] 周文浩[2] 王 飞[3] 安 虹[1]
[1](中国科学技术大学计算机科学与技术学院 合肥 230026)
[2](国家超级计算无锡中心 江苏无锡 214100)
[3](清华大学计算机科学与技术系 北京 100084)
(xqbeida@mail.ustc.edu.cn)

# A Dataflow Computing System for New Generation of Domestic Heterogeneous Many-Core Processors

Xiao Qian[1], Zhao Meijia[2], Li Mingfan[1], Shen Li[1], Chen Junshi[1], Zhou Wenhao[2], Wang Fei[3], and An Hong[1]
[1] (*Institute of Computer Science and Technology*, *University of Science and Technology of China*, *Hefei* 230026)
[2] (*National Suppercomputer Center in Wuxi*, *Wuxi*, *Jiangsu* 214100)
[3] (*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084)

**Abstract** Today, scientific research has moved from the era of computational science to the era of data science. Discovering laws from massive data and breaking through bottlenecks in scientific development are the main goals of the data science paradigm. At the same time, high performance computers are also paying more and more attention on intelligent computing power. Integrating AI algorithms on the basis of traditional high performance computing methods (HPC+AI) is more conducive to solving practical science problems in the era of data science, and can give full play to the intelligent computing power of high performance computers. However, on domestic HPC systems, especially on HPC systems constructed by the new generation of domestic heterogeneous many-core processors, there are many challenges to support HPC+AI programs. In this paper, we propose a data flow computing system for domestic heterogeneous many-core processors, which is called swFLOWpro. The system supports the use of TensorFlow interface to build data flow programs, and realizes many-core parallel acceleration transparent to users, and implements two-level parallel strategy based on the whole processor perspective. Testing on sw26010pro processor, swFLOWpro can get up to 545 times single core group (CG) many-core speedup ratio for typical OP, 346 times for typical deep learning models. Compared with the single CG of sw26010pro, we execute ResNet50 model on all the 6 CGs for one whole processor, and the speedup ration is up to 4.96 times, whose parallel efficiency is 82.6%. Experiments show that swFLOWpro can support the efficient execution of data flow programs represented by deep learning on domestic heterogeneous many-core processors.

**Key words** dataflow; deep learning; heterogeneous many-core; swFLOWpro system; high performance computing

**摘 要** 如今,科学研究已从计算科学时代进入数据科学时代.从海量数据中发现规律和突破科学发展瓶颈是数据科学范式的主要目标.与此同时,高性能计算机(HPC)也越来越重视智能算力,在传统高性能计算方法的基础上融合人工智能算法(HPC+AI),更有利于在数据科学时代解决实际问题,并能充分发挥高性能计算机的智能算力.不过,在国产HPC系统——特别是面向由新一代国产异构众核处理器sw26010pro构建的HPC系统——上支撑HPC+AI领域应用,则面临着诸多挑战.提出了一种面向国产异构众核处理器的数据流计算系统swFLOWpro,支持使用TensorFlow接口构建数据流程序,实现对用户透明

**收稿日期**: 2022-06-16; **修回日期**: 2023-01-16
**通信作者**: 安虹(han@ust.edu.cn)

的众核加速,并实现了面向全处理器视角的两级并行策略.经测试,系统针对典型核心计算,单核组众核加速比最高可达 545 倍、典型模型众核加速比最高可达 346 倍,全片 6 核组并行执行 ResNet50 模型训练,对比单核组加速比达到 4.96 倍,并行效率 82.6%.实验表明,swFLOWpro 能够支持以深度学习为代表的数据流程序在国产异构众核处理器上的高效运行.

**关键词** 数据流;深度学习;异构众核;swFLOWpro 系统;高性能计算

**中图法分类号** TP319

一直以来,高性能计算机(high performance computer, HPC)都是解决科学研究各领域实际问题的重要工具,依靠 HPC 的强大计算能力,能使很多求解空间极大的科研问题在现实可见的时间内完成解算.

最近 20 年,科学研究已经从计算科学时代进入数据科学范式时代,科学家需要从海量的数据中去探索科学规律和突破科学发展瓶颈,这就意味着传统的用高密度计算去模拟复杂现象进行科学研究的方法需要创新发展,用高性能计算与人工智能相融合的新方法(HPC+AI)去解决实际问题,正逐渐成为一种行之有效的科研方法,例如 2020 年的戈登贝尔高性能计算应用奖就颁发给基于深度学习实现 1 亿原子分子动力学的应用[1].

人工智能应用的开发和运行,往往依赖于人工智能编程框架,如 TensorFlow[2], Pytorch[3] 等,这些框架在本质上均是数据流计算系统,它们将神经网络模型组织成数据流图,并利用图节点融合、图剪枝、常量传播等技术进行图优化,然后再通过运行系统将图节点调度到实际的计算资源上执行.换言之,数据流计算系统是支撑人工智能应用的重要基础软件,但是,要在国产高性能计算机上支持高效的数据流系统,则面临着严峻的挑战.

从底层硬件的角度来说,国产异构众核处理器具有独特的复杂结构,新一代国产异构处理器 sw26010pro[4-5] 具有多级计算资源、多层次存储和多级互联网络结构,在体系架构上与传统的多核 CPU、众核 GPU 以及专用的人工智能处理器相比有着本质的区别.

要在 sw26010pro 上高效执行数据流系统,需重点解决 2 个问题:

1)如何充分利用 sw26010pro 的众核计算资源.计算核心阵列是国产异构众核处理器的性能来源,具有众多的精简核心和强大算力,但也存在着访存效率低、片上缓存小和管理复杂等实际问题.为实现数据流图的高效执行,就需要实现自适应的众核阵列加速方法,能够自动加速数据流图中的关键节点,充分利用众核计算资源.

2)如何设计高效的两级并行策略,充分利用国产异构众核处理器的全片计算资源.sw26010pro 采用全片多核组集成的体系结构,每个核组都是同构的众核阵列,多核组之间可以共享全局存储,数据流图中的节点执行以单核组为基本单元.要充分结合硬件结构特性,实现两级并行策略,通过相应的图优化和调度方法,确保多核组能够并行执行数据流图,提升系统性能.

对于上层用户而言,传统高性能计算机的软件环境也很难满足 HPC+AI 领域应用的动态化和智能化需求,并且,用户更希望将重心放在上层算法设计上,而非底层体系结构相关的优化上.

为此,本文提出了一种面向国产异构众核处理器的数据流计算系统 swFLOWpro,支持使用 TensorFlow 接口构建数据流计算和深度学习典型模型,并实现了对用户透明的众核并行加速,可以支持数据流程序的高效开发,在运行时充分利用国产异构众核处理器的硬件能力.

本文的主要贡献有 4 点:

1)在国产异构众核处理器上构建了功能完备的数据流计算系统 swFLOWpro,能够支持以深度学习为代表的数据流应用的开发和运行;

2)设计并实现了一种专门针对国产异构众核处理器的核心计算加速引擎 swHMAE,将之与数据流计算系统松耦合,实现自动化的众核并行加速及算子分析、调试功能;

3)针对 sw26010pro 的多核组共享内存结构,设计了一种面向异构融合体系结构的两级并行策略,结合图分裂技术,能充分利用全片核组的计算资源;

4)基于 swFLOWpro 进行 Alexnet, ResNet, VGG, Inception 等典型 CNN 神经网络模型训练测试,实验结果表明本文设计的数据流计算系统能够获得很好的异构众核并行加速效果.

## 1 相关工作

### 1.1 数据流计算

传统的冯•诺依曼计算机以控制流为执行模型,

而数据流计算则采用了不同的思路，将程序组织成有向图，每个图节点表示一个算子，边则代表节点之间的依赖关系，数据在边上流动，当一个节点的所有输入数据均已就绪时，该节点就会被启动. 数据流计算由程序本身的数据依赖关系来激活计算，更有利于充分发挥其天然的可并行性.

20 世纪 90 年代，麻省理工大学提出一种基于数据流思想的处理器设计方案，该方案没有共享存储和寄存器的设计，数据直接在计算部件之间流动. 当一条指令所有操作数均已就绪时，即可以进入执行状态. 这种体系架构能够充分挖掘程序的指令级可并行性，但也存在着运行开销大、并行粒度过小等实际问题，与传统计算机系统的天渊之别也限制了其进一步发展.

相关研究[6]还提出了一种硬件集成数据流芯片和冯•诺依曼架构芯片的体系结构设计，程序在编译系统的支持下，可以在运行过程中动态调度到不同的芯片架构上去. 这种处理器架构设计比较新颖，但对编译和硬件实现的环境要求比较高.

纯硬件的数据流计算系统面临着诸多问题，最本质的问题在于其与传统计算机软硬件生态无法兼容，发展严重受限. 于是，数据流计算机逐渐向与冯•诺依曼架构融合的方向发展，出现了“类数据流”计算机，这一类计算机融合了控制流和数据流的思想，将程序组织成一系列的宏指令或者代码块，每个宏指令或代码块内部采用数据流执行模式，而在宏指令和代码块之间依然采用传统的控制流思想进行组织管理，该类计算机包括 TRIPS[7], T3, EVX 等. 类数据流计算机将数据流的思想用于最底层的指令层面，在程序层面则保持着和传统架构相同的程序逻辑，例如 EDGE[8]架构执行模型，就是将程序编译成由超块组成的控制流图，将超块内部的代码编译成数据流指令，数据直接在计算部件之间流动而不通过寄存器，但 EDGE 架构必须运行在专门的类数据流处理器架构上，通用性较差. 目前，最常见的数据流系统是在传统冯•诺依曼架构上实现的软件数据流系统.

Codelet[9-10]执行模型由特拉华大学提出，它是一种针对 E 级计算机的需求而进行设计的细粒度并行、事件驱动的程序执行模型. Codelet 模型从数据流执行模型中得到启发，结合传统的冯•诺依曼体系架构，形成了一种在通用计算机上运行的数据流程序执行系统.

TensorFlow 是一款具有数据流思想的软件计算系统，该系统运行于通用处理器架构上，并对众核 GPU 和人工智能专用芯片 TPU 有后端支持.TensorFlow 是人工智能领域非常热门的编程框架，它将人工智能的算法模型组织成数据流图，并通过运行支持数据流图的高效映射和资源分配.TensorFlow 给用户提供了丰富的 API 接口来构建数据流计算，对深度学习的支持也比较完善，不过缺乏对于国产异构众核架构的后端支持.

### 1.2 在国产异构众核处理器上的数据流计算系统

在国产异构众核处理器上，关于数据流计算系统的研究也一直在进行中.

SunwayFlow[11]是基于神威太湖之光高性能计算机系统开发的数据流计算系统，该系统将 Codelet 执行模型移植到国产处理器上，并使用高性能共轭梯度基准测试（HPCG）作为测试数据，获得 10.32 倍的加速效果. 但 SunwayFlow 支持的 Codelet 模型适用范围有限，特别是对深度学习的支持严重不足.

swCaffe[12-13]是面向国产异构众核处理器的深度学习编程框架，它在底层通过 swDNN 库支持众核加速，针对 VGG-16 有 4 倍的加速效果. 但是 Caffe 框架[14]的编程接口已逐渐被淘汰，而 swCaffe 要实现众核加速，对模型的参数也有严格的限制，已无法适应数据流计算和深度学习应用的实际需求.

swFLOW[15]是 2021 年推出的针对国产异构众核处理器的数据流计算系统，该系统重构了 TensorFlow 框架，支持在 sw26010 处理器上执行数据流计算，针对典型神经网络模型有 10.42 倍的加速. 不过，swFLOW 在功能上只支持 TensorFlow 的 C++接口，在优化设计上重点考虑面向大规模计算资源的分布式训练，缺乏针对单进程的深度优化，也没有针对全片多核组运行模式的优化支持，实际使用效果有待增强.

针对上述多款数据流计算系统软件的缺陷，本文设计并实现了面向国产异构众核处理器 sw26010pro 的新一代数据流计算系统 swFLOWpro.该系统在编程接口支持上复用了 TensorFlow 的前端模块，可以完全兼容 TensorFlow 的 Python 和 C++编程接口，提升系统易用性，在后端则通过独立的核心计算加速引擎模块来提供针对数据流图节点的执行加速，除此之外，还针对 sw26010pro 的多核组共享内存设计，开发了一种面向异构融合的两级并行方法，从而提升全片计算资源的应用效率.

### 1.3 面向数据流计算的相关研究

Megatron-LM[16]主要讨论如何在大规模 GPU 集群上通过 tensor/pipline/data 等多种并行模式高效实现大模型的训练，通过并行模式的混合能够提升 10% 的数据吞吐量，在 3 072 个 GPU 上训练 1 万亿参

数模型，单 GPU 峰值效率达到 52%.

Gspmd[17] 提出一种基于编译器的自动化机器学习并行系统，可以在单节点代码上通过添加编译指示实现自动化并行代码生成，在 2 048 块 TPUv3 上达到了 50%~62% 的计算利用率.

DAPPL[18] 面向大模型提出了结合数据并行和流水线并行方法的并行训练框架，主要解决的问题是针对模型结构和硬件配置决策最优并行策略，如何调度数据流计算的不同流水线阶段.

Alpa[19] 针对分布式深度学习训练提出了算子内和算子间并行策略，通过系统化的方式将分布式并行策略的优化空间结构化，并在这个优化空间中寻找最优策略并实现自动化.Alpa 以计算图为输入，输出并行方案，主要考虑如何划分子图和计算任务调度.

目前，面向数据流计算的相关研究大多是针对大模型和大规模并行系统，专注于如何切割数据流图并将其调度到各计算节点上，本文则主要针对 sw26010pro 的异构众核结构和普通深度学习模型，专注于单处理器内部的计算流程，通过算子内和算子间的两级并行策略，高效利用单处理器计算能力. 在后续工作中，swFLOWpro 会在寻找最优并行策略以及调度模型的优化上加强研究.

## 2 swFLOWpro: 新一代数据流计算系统

本节主要介绍国产异构众核处理器 sw26010pro 的结构特点，以及 swFLOWpro 的整体架构和工作流程.

### 2.1 sw26010pro 架构

sw26010pro 是一款国产异构众核处理器，它包含 6 个核组（core group , CG），核组之间通过片上环网互连，每个核组包含 2 种异构核心，一种是管理核心（management processing element, MPE），另一种是计算核心（computing processing elements, CPE），1 个 MPE 和 1 个 8×8 的 CPE 组成 1 个异构计算阵列. 一般而言，MPE 主要负责计算任务、全局内存和运算核心的管理；CPE 负责计算任务的执行，每个 CPE 通过一个软件管理的片上便签存储器（LDM）来提升访存效率. sw26010pro 结构如图 1 所示.

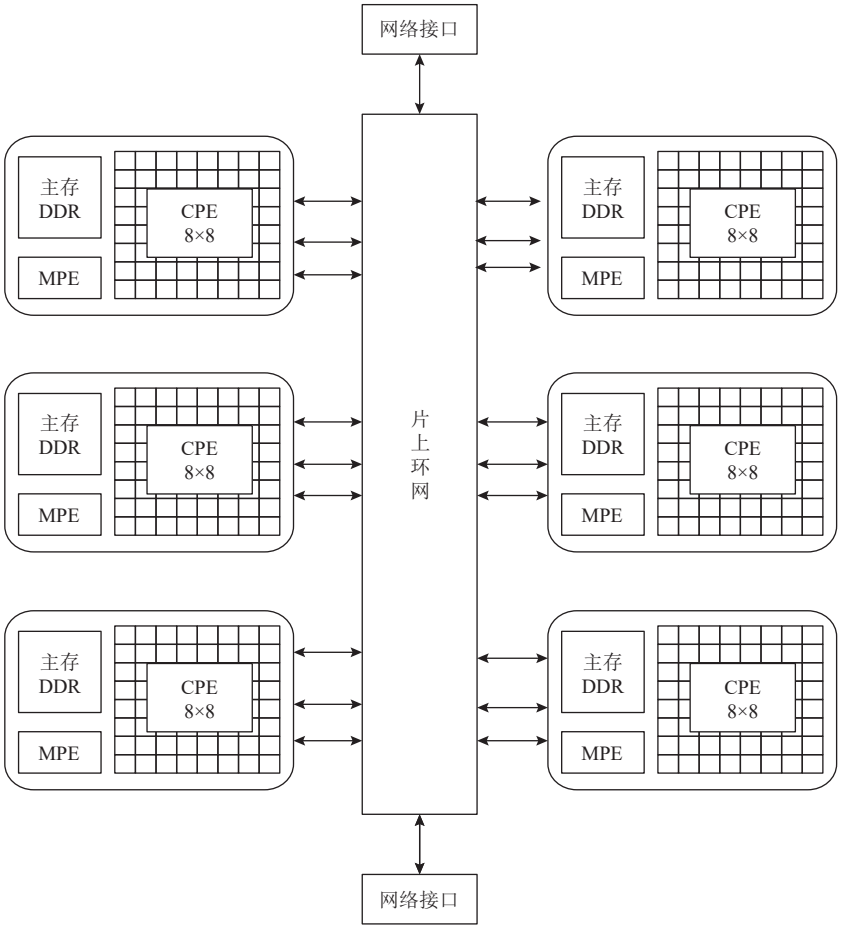


Fig. 1 The structure of sw26010pro

图 1 sw26010pro 结构

sw26010pro 采用SW64 自主指令集设计.其中, MPE 具有 32 KB L1 指令缓存、32 KB L1 数据高速缓存和 512 KB L2 高速缓存; CPE 支持 512 b 的 SIMD 运算, 支持双精度、单精度和半精度浮点及整数等多种数据类型的向量运算, 每个 CPE 具有独立的指令缓存和片上 LDM 存储, 其中 LDM 可以配置为 L1 数据缓存, 也可以配置为用户管理的局存空间, 支持通过 DMA 方式实现 LDM 和全局内存之间的数据传输; 支持通过 RMA 方式实现不同 CPE 之间的 LDM 存储传输.

sw26010pro 全处理器包含 6 个同构的核组, 6 核组之间可以共享全局内存. 通常情况下, 1 个进程运行在 1 个核组上, 多个核组之间通过 MPI 消息进行通信, 但这样会导致单进程可用的内存空间和计算能力都较小, 频繁的 MPI 通信也会造成性能损失. 事实上, 通过多核组的共享全局内存, 可以结合多线程管理和核组资源分配, 实现全片视角的统一编程, 这样能大幅度提升单进程的可用内存空间和计算能力, 减少进程间通信造成的性能损失.

与常规的处理器设计不同, sw26010pro 将更多的硬件逻辑用于计算, 从而最大程度地提升计算密度, 精简的 CPE 核心设计导致了其计算能力很强, 但访存能力较弱. sw26010pro 提供了用户可以显式管理的 LDM 存储来弥补访存与计算能力不匹配的问题, 从而支持用户充分挖掘异构众核的计算能力. 不过, 这种设计模式就意味着程序的高效运行需要更加复杂的优化策略和更加全面的算法改造.

对于数据流计算和深度学习领域的编程用户来说, 他们更关注的是数据流图的结构、模型的构造以及训练模型的超参数调整等上层算法设计, 而非底层硬件细节和体系结构相关优化技术.

为此, swFLOWpro 的主要设计目标就是构建国产异构众核处理器与用户之间的桥梁, 提供可移植性强、功能丰富的编程接口, 并将底层硬件细节对用户透明, 实现自动化的众核并行加速.

### 2.2 swFLOWpro 结构设计

swFLOWpro 数据流计算系统的整体架构图如图 2 所示.swFLOWpro 系统可以划分为 2 个子模块: 前端模块和后端模块. 中间层由 C-API 桥接.

前端模块是一个支持多语言的编程环境, 它提供基于数据流图的编程模型, 方便用户使用 TensorFlow 的 Python 和 C++编程接口构造各种复杂的计算图, 从而实现各种形态的模型搭建.

C-API 是桥接前端模块和后端模块的中间层次, 主要是通过 SWIG(simplified wrapper and interface generator)机制支持前端多语言编程环境与 C++实现的后端模块之间的通道.

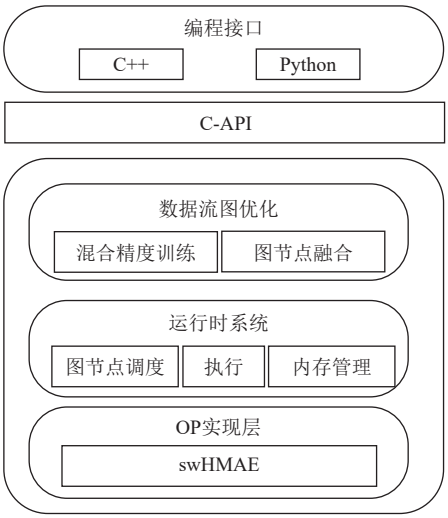


Fig. 2 The overall architecture of swFLOWpro

图 2 swFLOWpro 整体架构

为保证系统的易用性和提升深度学习程序的可移植性, swFLOWpro 框架的前端模块和 C-API 复用了 TensorFlow 框架的相应模块, 主要是为了保持对 TensorFlow 编程的兼容性.

后端模块则是体系结构相关的运行模块, 也是 swFLOWpro 针对 sw26010pro 架构特点重点开发的模块.swFLOWpro 的后端模块主要包括数据流图优化、运行时系统、算子(OP)实现层等子模块. 其中, 数据流图优化模块支持面向异构众核处理器的混合精度训练优化和节点融合优化, 混合精度训练优化在数据流图中插入数据类型转化节点, 将单精度运算转换为效率更高、精度更低的半精度运算, 而在更新参数节点等对精度要求更高的节点之前, 再将半精度转化为单精度, 从而支持混合精度训练; 图节点融合优化则将多个图节点融合, 形成更大的计算单元, 减少内存管理开销, 提升运行效率. 运行时系统主要负责计算图节点的管理、调度、执行以及内存分配, 根据图中依赖关系依次执行各个节点. OP 实现层则是针对 sw26010pro 的存储层次和结构特点, 将 OP 的定义和执行解耦, 通过独立的异构众核加速引擎(swHMAE)实现对关键性能 OP 的众核加速, 这一部分将在 2.3 节中详细介绍.

### 2.3 异构众核加速引擎(swHMAE)

swHMAE 是一个独立于 swFLOWpro 系统之外的独立模块, 其设计目的是为数据流计算系统提供一个松耦合的、体系结构相关深度优化的核心计算加速框架. 框架整体结构如图 3 所示.

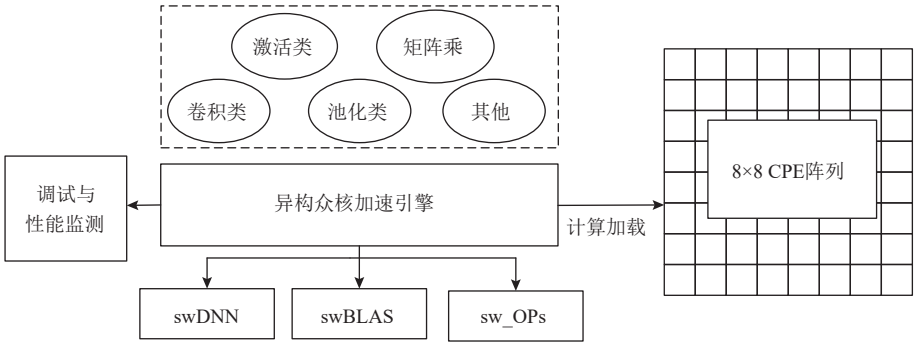


Fig. 3 The structure of swHMAE

图 3 swHMAE 结构

swHMAE 提供了一系列性能关键计算的调用接口，这些接口在 swFLOWpro 的算子实现层进行调用，而其真正实现则集成于一个独立的动态库中.

swHMAE 提供的这些接口是完全虚拟化的，仅用来描述要完成哪种运算和需要哪些参数，swHMAE 可以向上支持不同的人工智能编程框架或数据流系统的图节点实现模块，向下则可以调用多种众核加速算法库，也可以集成用户自定义的众核算法，具有很好的可扩展性.

在 swHMAE 中，针对不同的计算类型，主要完成 2 方面的工作：1)收集核心计算的参数. 2)根据参数类型、参数特性及输入规模，判断是否适合使用众核加速，如不适合，则该 API 返回失败，swFLOWpro 将调用默认的实现算法；否则，swHMAE 将根据不同的参数类型和规模自适应地选择最优的异构众核加速算法.

swHMAE 支持的核心计算类型涵盖了数据流计算常见的计算类型，核心计算类型既有深度学习领域的常见计算，例如卷积、矩阵乘、激活、归一化等，这类计算的众核加速主要是通过 swDNN, swBLAS, Sw_OPs 等第三方库来支持，又有一些更通用的数据流计算节点类型，如批量数据的基础运算、数据的 Padding, tile, slice 等访存操作，以及其他一些定制的计算类型.

swHMAE 的工作原理算法如算法 1 所示：

**算法 1.** swHMAE 工作原理算法.

输入：计算类型 *OP-type*，张量 $\boldsymbol{t}_1,\boldsymbol{t}_2,\cdots$，数据类型 *data-type*，常量参数 *params*；

输出：计算结果张量 ***t-results***.

① if *notSuitforMC*(*OP-type,params,*$\boldsymbol{t}_1,\boldsymbol{t}_2,\cdots$)

② return false; /* 如果该 OP 不适合众核加速返回 false，执行 swFLOWpro 的默认计算模式 */

③ end if

④ *timing_or_debug_this_op_start*();

⑤ if *OP-type* ∈ {*SW_Conv,SW_Activate, SW_Pooling*}

⑥ ***t-results*** = *swDNN*(*OP-type,params,*$\boldsymbol{t}_1,\boldsymbol{t}_2,\cdots$);

⑦ else if *OP-type* ∈ {*SW_Matmul*}

⑧ ***t-results*** = *swBLAS*(*OP-type,params,* $\boldsymbol{t}_1,\boldsymbol{t}_2,\cdots$);

⑨ else

***t-results***=*MC_accelete_op*(*OP-type,params,* $\boldsymbol{t}_1,\boldsymbol{t}_2,\cdots$);

⑩ end if

⑪ *timing_or_debug_this_op_end*();

⑫ return true.

swHMAE 是面向国产异构众核处理器的数据流计算后端，作为一个独立模块，它将关键计算的众核加速与数据流系统的整体框架解耦，既能够高效利用 swDNN, swBLAS, sw_OPs 等众核计算库，由于本身也集成了一系列众核优化算子，也能够对更多的核心计算进行众核加速.

swHMAE 针对非计算密集类运算实现了众核加速算法，其主要思想是通过数据分割将运算任务分配到各 CPE 上执行，通过 DMA 数据传输机制将具有局部性的数据显式地搬运到 CPE 的片上内存 LDM 中，并通过 2 个数据传输缓冲的动态切换，实现数据传输与数据计算的并行操作，其算法思想如图 4 所示.

除此之外，swHMAE 还可以通过多种方式对关键计算进行调试、错误定位和性能分析，进一步提升易用性.

swHMAE 的松耦合和模块化设计使得用户可以更加方便地集成新的众核计算到 swFLOWpro 系统中去. 事实上，swHMAE 还可以支持其他的数据流计算系统，其仅需要在原始系统中做极少量的修改.

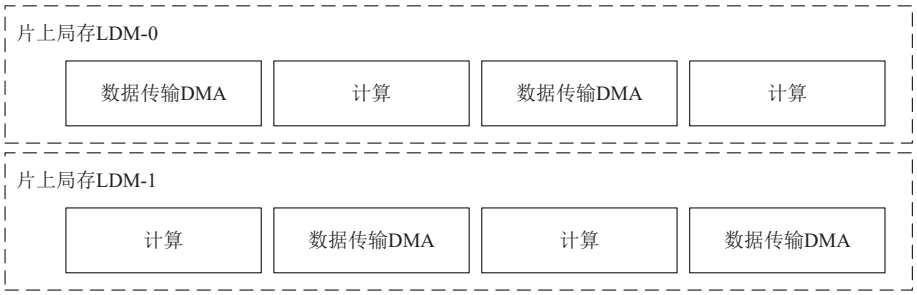


Fig. 4 Double buffer algorithm idea for CPE

图 4 CPE 双缓冲算法思想

## 3 面向异构融合的两级并行策略

在异构融合的众核处理器上执行数据流图的基本流程为：MPE 负责数据流图的生成、优化和调度管理；在执行过程中，将已满足执行需求的图节点分配到众核阵列上执行.

有 2 种任务分配方法可以考虑：1)将每个节点调度到 1 个 CPE 上，CPE 阵列协同完成整个数据流图的执行过程；2)将 CPE 阵列视为一个整体部件，所有计算核心共同完成数据流图中的一个节点.

第 1 种任务分配方法与异构融合众核架构的适应性并不好，其主要原因有 3 点：1)单 CPE 的访存能力有限，其 LDM 的容量大小也很难承载一个完整的图节点计算逻辑，比如卷积、矩阵乘等常用算子，在单 CPE 上执行效率较差；2)数据流图的可并行性有限，考虑某些具有强相关性的数据流图，每个节点都依赖于上一个节点的计算结果，则程序在这种模式下执行的效率就会很差，因为大部分时间内 CPE 可能因为依赖另一个 CPE 的计算结果而处于等待状态；3)负载均衡问题，由于每个数据流图节点运算量相差较大，保证各计算核心的负载均衡也是个难以解决的问题.

本文主要采用第 2 种任务分配方法，也就是将 CPE 阵列视为整体部件，所有 CPE 协同完成一个图节点的执行过程，这样每个 CPE 的计算任务量都在可以接受的范围之内，而在每个图节点内部，主要通过数据分割的方式将输入数据映射到各个 CPE 上，这样能保证 LDM 空间够用和保证各计算节点的负载均衡性. 并且，由于并行发生在图节点内部，整体效率不会受限于数据流图本身的可并行性.

图 5 是在 sw26010pro 的单核组上运行一个数据流图的示例.

输入数据的后继图节点是 Reshape，该节点是为了改变输入形状，属于功能类算子，所以将其调度到 MPE 上执行即可；其后的 Matmul, Biasadd, Softmax 都是计算密集的图节点，需要调度到 CPE 阵列上进行众核并行计算，例如，CPE 在执行 Matmul 图节点的时候，首先将矩阵进行分块，每个 CPE 执行子矩阵乘法运算，再通过 CPE 阵列内部的 RMA 操作进行全局通信，获得原始矩阵的乘法运算结果.

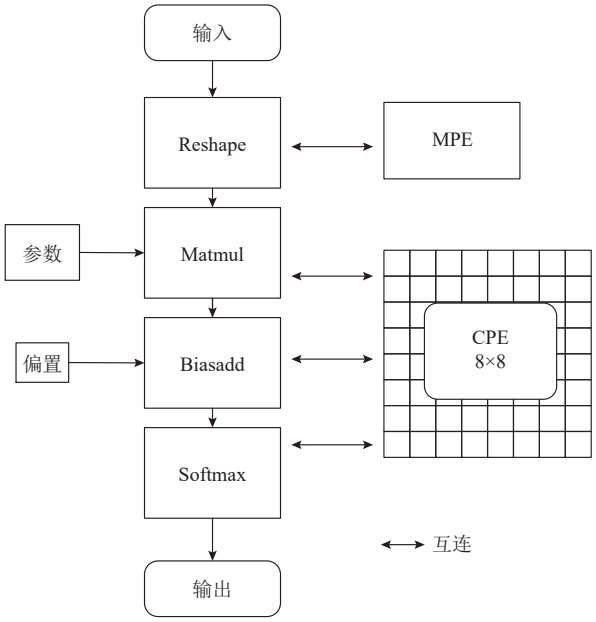


Fig. 5 Dataflow scheduling example for single CG

图 5 面向单核组的数据流调度示例

sw26010pro 异构众核芯片采用多核组设计，处理器内部包含 6 个同构的核组，每个核组都有一个 MPE 和一个 8×8 的 CPE 阵列. 因此，要在 6 核组结构上实现更高层次的并行.

在单核组内部，我们将 1 个图节点分配到 1 个 MPE 或者 1 个 CPE 阵列上执行，实现了低层次的图节点内并行；在基于全片视角的多核组上，利用 6 个等价队列分别维护由上层图计算过程产生的计算任务；在运行核组选择过程中采用 Round-Robin 的轮询调

度策略;在计算任务选择中采用先入先出(FIFO)方法,进而支持高层次的图节点间并行.这就是本节提出的两级并行策略,该策略能够充分适应 sw26010pro 的异构融合架构.

值得注意的是,图节点间的并行要求图节点之间没有数据依赖关系,但实际上一般单输入的数据流计算图可并行性并不高,如果将不同的图节点调度到不同核组上,由于图节点之间的数据依赖关系,会导致部分核组处于空闲状态,需要等待其他核组的计算结果才能开始计算.

为此,本文设计了一种图分裂优化方法,首先将输入数据进行平均分割,分割之后的每个输入都进行相同的数据流图执行流程,在输出结果时再进行归并,从而生成并行性更好的数据流计算图.

以图 5 的数据流图为例,将 *split* 值设置为 2,经过图分裂之后的数据流图如图 6 所示.

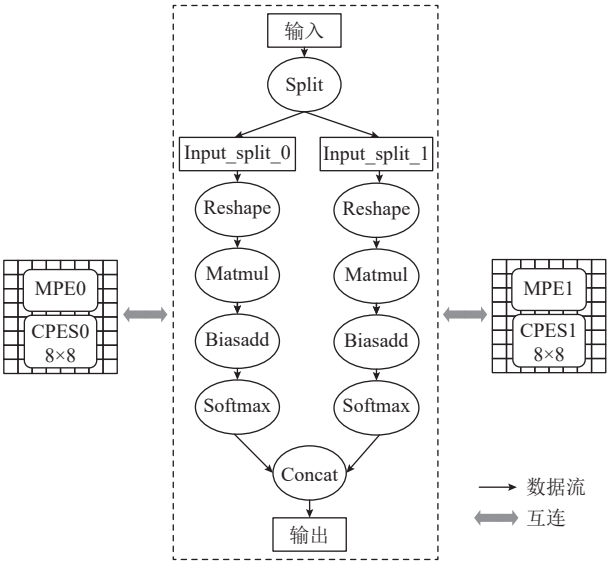


Fig. 6 Dataflow scheduling example after graph splitting

图 6 经过图分裂之后的数据流调度示例

经过图分裂之后,数据并行输入到不同的数据流子图中,每个子图都是原数据流图的一个复制,各个子图之间没有强相关性,从而具有很好的可并行性,可以映射到不同的处理器分区上执行.

图分裂是一种与体系结构无关的图变换技术,分裂值 *split* 可以调整,以适应不同的硬件体系结构.如果众核处理器集成更多的核组数,只需要提升分裂值,无需改变整体算法就能充分利用硬件计算资源.

在具体实现上,本文采用多线程机制来管理图节点的调度,根据核组数来确定线程个数.在 sw26010pro 上会启动 6 个线程来执行数据流图,每个线程绑定在 1 个核组上运行,这样能保证各线程不存在资源冲突问题.

调度器将所有图节点组织成任务池,并记录每个节点的前继节点.在执行过程中,一个图节点可能处于不可用、可用、执行中、完成中这 4 种状态的一种.每种状态对应一个任务池.

初始情况下,将没有前继节点的图节点状态设置为“待命”,其余节点状态均设置为“不可用”.线程函数从任务池里通过抢占方式获取一个图节点任务,如果该图节点的已处于可用状态(所有前继节点均已完成),则执行该节点,并将该节点状态设置为“执行中”,完成后则设置状态为“完成”.值得注意的是,线程选择下一个执行节点时,优先从该节点的后继节点中选取,如果后继节点不可用,则从该节点前继节点的其他后继节点中选择.这种搜索方法可以使得单个相对独立的数据流子图在一个线程内部完成.

图节点状态变换关系如图 7 所示.

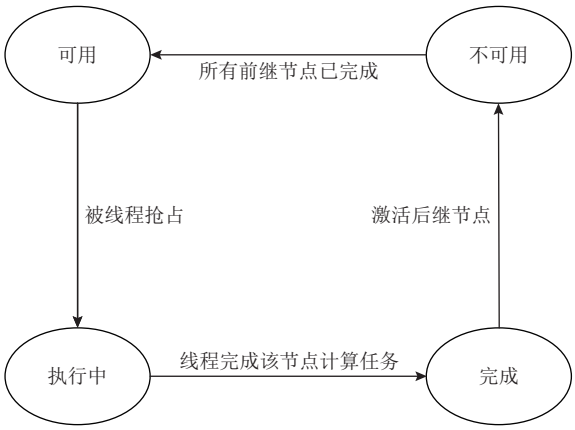


Fig. 7 Graph node state transformation diagram

图 7 图节点状态变换图

## 4 实 验

本文选择 6 种典型神经网络模型作为数据流计算的输入,通过 TensorFlow 编程接口编写数据流程序,实现这些模型的训练过程,这些模型及其变种也是 HPC+AI 领域应用经常使用的模型.具体模型信息如表 1 所示.

测试硬件平台为 sw26010pro 处理器,其包含 6 个核组,6 个 MPE 和 384 个 CPE,全片主存空间大小为 96 GB,每个 CPE 的片上高速缓存 LDM 大小为 256 KB.

软件环境为 swFLOWpro 数据流计算系统、swHMAE

**Table 1 Six Typical Neural Network Models**

**表 1 6 种典型神经网络模型**

| 典型模型 | 输入数据 | 参数量 | 计算量/GFLOP |
|---|---|---|---|
| Alexnet[20] | 227×227×3 | $61.1\times10^6$ | 0.77 |
| VGG16[21] | 224×224×3 | $138.36\times10^6$ | 15.61 |
| ResNet50[22] | 224×224×3 | $25.56\times10^6$ | 4.14 |
| ResNet101 | 224×224×3 | $44.55\times10^6$ | 7.87 |
| Inception3[23] | 299×299×3 | $21.16\times10^6$ | 5.75 |
| Inception4[24] | 299×299×3 | $41.22\times10^6$ | 10.48 |

**Table 2 Typical Core Computing**

**表 2 典型核心计算**

| 核心计算类型 | 分类 |
|---|---|
| Conv2D | SW_Conv |
| Conv2DBackpropFilter | SW_Conv |
| Conv2DBackpropInput | SW_Conv |
| Matmul | SW_Matmul |
| Relu | SW_Activate |
| Poolmax | SW_Pooling |
| ApplyGradientDescent | SW_OPs |

核心计算加速引擎, 以及 swPython 编程环境.

本文选择众核加速比 *ManyAccRatio* 作为主要的性能评价指标, 其定义为:

$$ManyAccRatio = \frac{MPE\_time}{CPE\_time}\times 100\%,$$

其中 *MPE_time* 表示在 MPE 主核上的运行时间, *CPE_time* 表示在单核组阵列上的运行时间. 由于 sw26010-pro 结构的特殊性, 其与 GPU, TPU 等人工智能专用芯片的性能对比意义不大, 通过众核加速比可以体现 SwFlowpro 在 sw26010pro 独特的异构融合结构上的适配性和优化效果.

### 4.1 swHMAE 针对典型模型核心计算的众核加速效果

本文使用 swFLOWpro 构建了 6 种典型模型, 并统计了模型中所有核心计算(数据流图节点)类型, 选择 7 种典型核心计算类型, 通过 swHMAE 引擎进行众核加速. 具体统计信息如表 2 所示.其中 Conv2D, Conv2DBackpropFilter, Conv2DBackpropInput 都是卷积类计算, Matmul 是矩阵乘计算, Relu是激活类计算, Poolmax 是池化类计算, ApplyGradientDescent 是训练更新参数计算.

在表 1 的 6 种典型模型中, 统计了典型核心计算在 sw26010pro 的单核组 CPE 上的运行时间, 通过对比 swFLOWpro 未经众核优化的 MPE 运行时间, 并获得众核加速比.详细测试数据如表 3 所示.通过计算, 获得的各类型的典型核心计算众核加速比如图 8 所示.

卷积类运算是实验的 6 种典型模型的关键, 也是 swHMAE 实现众核加速的重点运算.swHMAE 会根据输入规模和相关参数, 自适应选择 swDNN 库中最优的算法实现. 由图 8 实验结果可以看出, Conv2D 的众核加速比达 250~545, Conv2DBackpropFilter 的众核加速比达 107~583, Conv2DBackpropInput 的众核加速比达 90~310, 加速效果良好.

其他核心计算类型的众核加速比测试数据如图 9 所示.

针对矩阵乘类核心计算, swHMAE 从 swBLAS 库中自适应选择众核算法. 测试表明, 矩阵乘核心计

**Table 3 Test Time of Typical Core Computing in Typical Models**

**表 3 典型模型中的典型核心计算测试时间** ms

| 模型 | Conv2D | Conv2DBackpropFilter | Conv2DBackpropInput | Matmul | Relu | Poolmax | ApplyGradientDescent |
|---|---|---|---|---|---|---|---|
| Alexnet-MPE | 153 470 | 141 700 | 51 100 | 20 260 | 352 | 340 | 528 |
| Alexnet-CPE | 611 | 1 320 | 238 | 548 | 11 | 23 | 21 |
| VGG16-MPE | 1 629 210 | 1 114 340 | 718 380 | 37 430 | 5 260 | 2 120 | 1 080 |
| VGG16-CPE | 3 140 | 1910 | 2 310 | 1 120 | 119 | 172 | 50 |
| ResNet50-MPE | 250 690 | 285 090 | 152 940 | 181 | 2 170 | 373 | 250 |
| ResNet50-CPE | 652 | 1 690 | 1 690 | 9 | 56 | 24 | 14 |
| ResNet101-MPE | 578 400 | 552 900 | 303 010 | 182 | 3 410 | 372 | 449 |
| ResNet101-CPE | 1 060 | 2 200 | 2 220 | 9 | 90 | 24 | 25 |
| Inception3-MPE | 489 790 | 443 940 | 216 850 | 86 | 2 140 | 3 730 | 215 |
| Inception3-CPE | 1 160 | 1 480 | 1 540 | 3 | 61 | 278 | 16 |
| Inception4-MPE | 1 019 110 | 1 010 510 | 472 480 | 21 | 3 610 | 6 370 | 413 |
| Inception4-CPE | 2 670 | 2 960 | 3 660 | 0.8 | 101 | 503 | 26 |

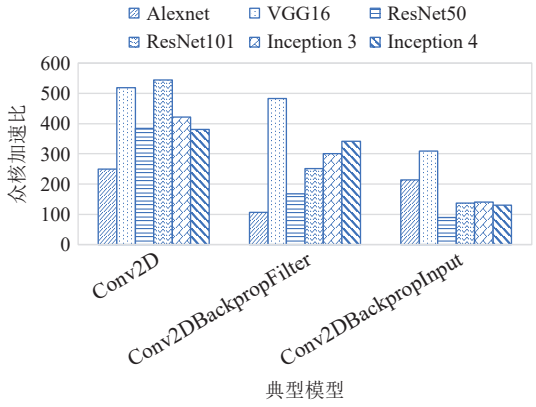


Fig. 8 The many-core acceleration ratios of convolutional core computing for different typical models

图 8 不同典型模型的卷积类核心计算众核加速比

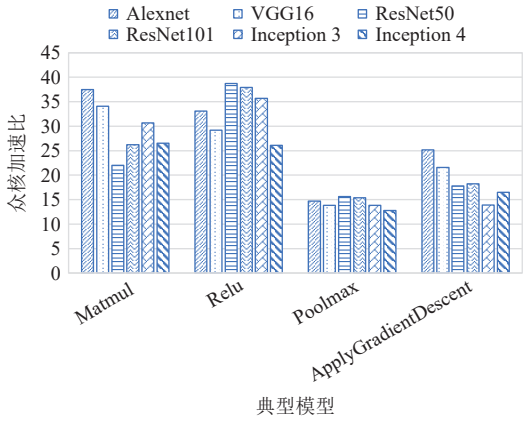


Fig. 9 The many-core acceleration ratios of other core computing for different typical models

图 9 不同典型模型的其他核心计算众核加速比

算的众核加速比仅有 26.1~38.7, 由表 3 可以看出, 本文选择的典型模型都是卷积类神经网络, 矩阵乘的计算量很小, 不能充分发挥 CPE 从核阵列的全部计算能力. 除此之外, swBLAS 库中矩阵是按列优先模式存储, 在接入模型时还需要先进行矩阵转置. 所以, 矩阵乘的实际众核加速比效果远低于卷积类算子, 在后续工作中可以针对矩阵转置进行优化.

针对 Relu 激活类运算, swHMAE 通过 swDNN 库进行加速, 众核加速比达到 26.1~38.7.

除了计算密集类运算之外, 模型中也会用到一些其他算子, 这类算子虽然计算量小, 但如果不进行众核优化, 则会成为性能瓶颈. 如本文实验选择的更新参数操作(ApplyGradientDescent), 是模型训练中常见的算子类型, 但缺乏专属的算法库支持. 本文选择在 swHMAE 中直接集成其众核优化算法, 实验表明众核加速比达 13.9~25.2.

测试结果表明, 在 sw26010pro 上, 卷积类运算的众核加速比要远高于其他运算类型, 这主要是因为国产异构众核的架构设计对于卷积这类计算密集类运算的适应性更好.

### 4.2 swFLOWpro+swHMAE 针对典型模型训练的众核加速效果

本文使用 swFLOWpro+swHMAE 运行 6 种典型模型的训练过程, 单步训练 *batch* 大小统一设置为 32.

实验分别测试在 sw26010pro 的单 MPE 和单 CPE 阵列上的单步训练时间, 并计算众核加速比. 测试数据如表 4 所示.

**Table 4 Single Step Training Test Data of Typical Models**

**表 4 典型模型的单步训练测试数据**

| 典型模型 | MPE 运行时间/s | CPE 运行时间/s | 众核加速比 |
|---|---|---|---|
| Alexnet | 379.8 | 3.1 | 123 |
| VGG16 | 3 525.5 | 10.2 | 346 |
| ResNet50 | 973.6 | 8.2 | 119 |
| ResNet101 | 1 876.6 | 12.1 | 155 |
| Inception3 | 1 373.1 | 11.9 | 115 |
| Inception4 | 2 996.9 | 20.1 | 149 |

由图 10 可见, VGG16 模型的众核加速比最高, 达到 346, 其余的模型加速比相差不大, 在 115~155 之间.

模型的性能与模型中各类型核心计算的性能紧密相关, 由 4.1 节测试结果可知, 在 sw26010pro 上, 卷积类运算的众核加速比要远高于其他运算, 所以卷积类运算占比较高的模型, 在 sw26010pro 上的整体加速比也更高.

本文统计了在 6 种典型模型中, 卷积类和非卷积类核心计算的运行时间占比, 如表 5 所示. 这 6 种典型模型都属于卷积神经网络, 它们的卷积类运算占比为 82.5%~97.4%.

表 5 中, VGG16 的卷积类运算占比最高, 达到了

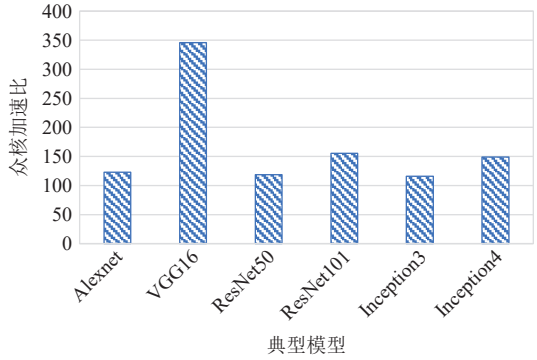


Fig. 10 Muti-core acceleration ratios of typical models

图 10 典型模型的众核加速比

**Table 5 Core Computing Proportion of Convolutional and Non-Convolutional of Typical Models**

**表 5 典型模型的卷积类和非卷积类核心运算占比** %

| 典型模型 | Conv2D | Conv2DBackpropFilter | Conv2DBackpropInput | 非卷积类运算 |
|---|---|---|---|---|
| Alexnet | 11.5 | 27.2 | 54.8 | 6.5 |
| VGG16 | 11.1 | 37.2 | 49.1 | 2.6 |
| ResNet50 | 9.2 | 25.3 | 48 | 17.5 |
| ResNet101 | 10.1 | 23.7 | 51.3 | 14.9 |
| Inception3 | 11.8 | 31.6 | 40.5 | 16.1 |
| Inception4 | 11.6 | 29.7 | 45.2 | 13.5 |

97.4%(11.1%+37.2%+49.1%)，所以这个模型的众核加速比也最高，Alexnet 的卷积类运算虽然占比高达 93.5%(11.5%+27.2%+54.8%)，但由于其卷积类运算的计算量较小，不能充分发挥 sw26010pro 的计算能力，所以整体众核加速比只有 123.

实验表明，针对典型模型的训练过程，swFLOWpro+swHMAE 比原始运行模型，特别是卷积类计算占比较高的模型(如实验中的 VGG16)有显著的众核加速效果.

### 4.3 两级并行优化效果

我们将 sw26010pro 的单处理器(包含 6 个核组)作为一个执行单元，测试 6 种典型模型经过面向全片的两级并行优化之后的加速效果.

首先，测试不使用图分裂技术的 6 个核组并行加速效果，在这种模式下，6 个核组的利用效率受限于不同模型构建出的数据流图本身的可并行性，测试数据如图 11 所示.

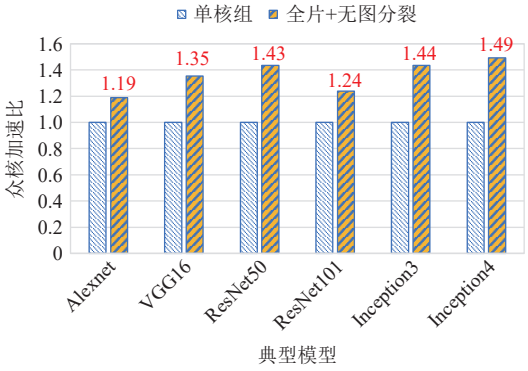


Fig. 11 Full chip acceleration ratios of typical models without graph splitting technology

图 11 不使用图分裂技术的典型模型全片加速比

加速比最高的是 Inception4 模型，达 1.49；最低的是 Alexnet 模型，达 1.19.这是因为，Inception 模型本身的数据流图具有不错的可并行性，而模型结构简单的 Alexnet 模型可并行性并不好.

整体而言，在不使用图分裂的情况下，6 核组的加速比较低，这是因为计算图的核心计算节点之间存在依赖关系，导致高层次的节点间并行不能同时进行计算，限制了并行效果，这也是本文提出图分裂技术的主要原因.

然后，使用图分裂技术进行优化，将 *split* 值分别设为 2, 4, 6, 并测试典型模型在全片 6 核组上运行对比单核组(*split* = 1)运行的加速比，测试数据如图 12 所示.

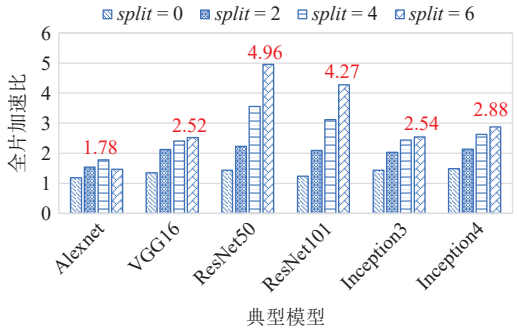


Fig. 12 Full chip acceleration ratios of typical models with graph splitting

图 12 典型模型使用图分裂技术后的全片加速比

图 12 中加速效果最好的是 ResNet50(*split* = 6)，加速比达 4.96，并行效率达到了 82.6%. 通过使用图分裂技术，选择合适的参数 *split*，典型模型全片加速比能达到 1.78~4.96.

图分裂技术结合面向异构融合的两级并行策略，在 sw26010pro 的多核组异构众核结构上取得了很好的并行效果，测试表明，图分裂技术针对典型模型的性能提升效果最高达到 246%(ResNet50)，最低也能达到 50%(AlexNet).

值得一提的是，从实验数据中也可以看出 2 个问题：1)sw26010pro 的众核结构对模型和核心计算的计算量要求较高，一些轻量级的模型无法充分利用众核资源，所以 Alexnet 的单核组和 6 核组加速比都不理想. 2)图分裂结束也会带来图节点数量的大幅度增长，会增大内存需求，对于 Inception 这种本身就

具有一定并行性的计算图,会出现图节点膨胀的现象,进而增大节点调度和分配的开销,所以其6核组并行加速比只有2.54~2.88,这也是图分裂技术目前存在的缺陷.

## 5 结 论

本文提出了一种面向新一代国产异构众核处理器的数据流计算系统swFLOWpro,该系统通过核心计算加速引擎swHMAE支持在国产异构众核处理器上的并行加速,并提出面向异构融合的两级并行策略,支持面向国产异构众核处理器全芯片视角的调度和并行方法.实验表明,swHMAE针对卷积类核心计算,众核加速比达90~545,针对其他核心计算,众核加速比达13.9~38.7;swFLOWpro+swHMAE支持典型模型在sw26010pro上的高效执行,VGG16模型众核加速比可达346;通过面向异构融合的数据流调度策略,全片ResNet50加速比达4.96倍,6核组并行效率达到82.6%.

未来的工作主要包括3个方面:1)继续拓展swHMAE支持的核心计算类型;2)优化面向全片多核组的两级并行策略,优化图分裂算法,探索更高效的数据流调度算法,提升图节点间并行效率;3)完善系统,支持更多种类的神经网络模型高效运行,并引入新的优化算法.

**作者贡献声明:** 肖谦提出了技术方案,实现系统和撰写论文;赵美佳和李名凡负责核心计算众核优化实现和论文完善;沈莉和陈俊仕负责数据流调度算法实现和优化;周文浩和王飞负责部分实验代码编写;安虹提出指导意见并修改论文.

## 参 考 文 献

[1] Jia Weile, Wang Han, Chen Mohan, et al. Pushing the limit of molecular dynamic with abinitio accuracy to 100 million atoms with machine learning[C] //Proc of the 33rd Int Conf for High Performance Computing, Networking, Storge and Analysis(SC'20) . Piscataway, NJ: IEEE, 2020: 1−14

[2] Abadi M, Barham P, Chen Jianmin , et al. TensorFlow: A system for large-scale machine learning[C] //Proc of the 12th USENIX Symp on Operating Ssytems Design and Implementation(OSDI'2016). Berkeley, CA: USENIX Association, 2016: 265−283

[3] Paszke A, Gross S, Massa F, et al. Pytorch: An imperative style high-performance deep learning libray[J]. Advances in Neural Information Processing Systems, 2019, 32: 8026−8037

[4] Liu Yong, Liu Xin, Li Fang, et al. Closing the "quantum supremacy" gap: Achieving real-time simulation of a random quantum circuit using a new Sunway supercomputer [C/OL] //Proc of the 34th Int Conf for High Performance Computing, Networking, Storge and Analysis(SC'21) . Piscataway, NJ: IEEE, 2021[2023-01-11].https://arvix.org/abs/2110.14502

[5] Hu Xiangdong, Ke Ximing, Yin Fei, et al. Shenwei-26010: A high-performance many-core prcocessor[J]. Journal of Computer Research and Development, 2021, 58(6): 1155−1165 (in Chinese)
(胡向东, 柯希明, 尹飞, 等. 高性能众核处理器申威26010[J]. 计算机研究与发展, 2021, 58(6): 1155−1165)

[6] Tony N, Vinay G, Karthikeyan S, et al. Exploring the potential of heterogeneous von neumann/dataflow execution models[C] //Proc of the 42nd Annual Int Symp on Computer Architecture(ISCA'15). Piscataway, NJ: IEEE, 2015: 298−310

[7] Sankaralingam K, Nagarajan R, Liu Haiming, et al. Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture[C] //Proc of the 30th Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE , 2003: 422−433

[8] Burger D, Keckler S W, Mckinley K S, et al. Scaling to the end of silicon with EDGE architectures[J]. Computer, 2004, 37(7): 44−55

[9] Zuckerman S, Suetterlein J, Knauerhase R, et al. Using a Codelet program execution model for exascale machines [C] //Proc of the 1st Int Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era. New York: ACM, 2011: 64−69

[10] Lauderdale C, Khan R. Towards a Codelet-based runtime for exascale computing[C]//Proce of the 2nd Int Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era. New York: ACM, 2012: 21−26

[11] Su Zhichao, Chen Junshi, Lin Han, et al. A dataflow-based runtime support on a 100P actual system [C] //Proc of the 15th Int Symp on Parallel and Distributed Processing with Applications. Piscataway, NJ: IEEE, 2017: 599−606

[12] Zhao Wenlai, Fu Haohuan, Fang Jiarui, et al. Optimizing convolutional neural networks on the Sunway Taihulight supercomputer[J]. ACM Transactions on Architecture and Code Optimization, 2018, 15(1): 13−25

[13] Li Liandeng, Fang Jiarui, Fu Haohuan, et al. swCaffe: A parallel framework for accelerating deep learning applications on Sunway Taihulight[C] //Proc of the 20th IEEE Int Conf on Cluster Computing (CLUSTER). Piscataway, NJ: IEEE, 2018: 413−422

[14] Jia Yangqing, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[C]// Proc of the 22nd ACM Int Conf on Multimedia. New York: ACM, 2014: 675−678

[15] Li Mingfan, Lin Han, Chen Junshi, et al. swFLOW: A large-scale distributed framework for deep learning on Sunway Taihulight supercomputer[J]. Information Sciences, 2021, 570(9): 831−847

[16] Deepak N, Mohammad S, Jared C, et al. Efficient large-scale language model training on GPU clusters using megatron-LM [C] //Proc of the 34th Int Conf for High Performance Computing, Networking, Storage

and Analysis. (SC'21). Piscataway, NJ: IEEE, 2021: 401−412

[17] Xu Yuanzhong, HyoukJoong L, Chen Dehao, et al. Gspmd: General and scalable parallelization for ml computation graphs [J]. arXiv preprint, arXiv: 2105.04663, 2021

[18] Fan Shiqing, Yi Rong, Meng Chen, et al. DAPPLE: A pipelined data parallel approach for training large models [C] //Proc of the 26th ACM SIGPLAN Symp on Principles and Practice of Parallel Programming. New York: ACM, 2021: 431–445

[19] Zheng Lianmin, Li Zhuohan, Zhang Hao, et al. Alpa: Automating inter- and intra-operator parallelism for distributed deep learning[C] //Proc of the 16th USENIX Symp on Operating Systems Design and Implementation. New York: ACM, 2022: 559–578

[20] Krizhevsky A, Sutskever I, Hinton G, et al. ImageNet classification with deep convolutional neural networks[C] //Proc of the 26th Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2012: 1097−1105

[21] Alexis C, Holger S, LeCun Y, et al. Very deep convolutional networks for large-scale image recognition[C]//Proc of the 15th European Chapter of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2017: 1107−1116

[22] He Kaiming, Zhang Xiangyu, Ren Shaoqing, et al. Deep residual learning for image recognition[C] //Proc of the 33rd IEEE Conf on Computer Vision and Pattern Recognition(CVPR). Piscataway, NJ: IEEE, 2016: 770−778

[23] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception architecture for computer vision[C] //Proc of the 33rd IEEE Conf on Computer Vision and Pattern Recognition(CVPR). Piscataway, NJ: IEEE, 2016: 551−561

[24] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[J]. arXiv preprint, arXiv: 1602.07261, 2016

**Xiao Qian**, born in 1988. PhD candidate. His main research interests include compiler optimization, data flow computing, and AI framework.

**肖　谦**, 1988 年生.博士研究生.主要研究方向为编译优化、数据流计算和 AI 框架.

**Zhao Meijia**, born in 1992. Master. Her main research interest includes AI framework.

**赵美佳**, 1992 年生. 硕士. 主要研究方向为 AI 框架.

**Li Mingfan**, born in 1995. PhD candidate. His research interests include dataflow system, parallel and distributed computing in heterogeneous environments.

**李名凡**, 1995 年生. 博士研究生.主要研究方向为数据流系统、异构环境下的分布式并行计算.

**Shen Li**, born in 1981. PhD candidate. Her main research interests include compiler and AI basic software.

**沈　莉**, 1981 年生.博士研究生.主要研究方向为编译器和人工智能基础软件.

**Chen Junshi**, born in 1990. PhD. His main research interests include high performance computing, general processor architecture, and optimization of scientific applications on large scale systems.

**陈俊仕**, 1990 年生. 博士. 主要研究方向为高性能计算、通用处理器架构和大规模系统上的科学应用优化.

**Zhou Wenhao**, born in 1992. Master. His main research interests include compiler optimization and many-core program environment.

**周文浩**, 1992 年生. 硕士. 主要研究方向为编译优化和众核编程环境.

**Wang Fei**, born in 1981. PhD candidate. His main research interests include compiler optimization and many-core program environment.

**王　飞**, 1981 年生. 博士研究生. 主要研究方向为编译优化和众核编程环境.

**An Hong**, born in 1963. PhD, professor, PhD supervisor. Her main research interests include parallel computing system and many-core chip architecture.

**安　虹**, 1963 年生. 博士, 教授, 博士生导师. 主要研究方向为并行计算系统和众核芯片架构.