基于软件定义的并行计算架构设计与实现

聂飞，江波

（中国电子科技集团公司第三十二研究所，上海 200233）

摘　要： 近年来，随着人工智能（AI）技术的快速发展，神经网络以较强的建模性能和优秀的多场景适应能力被广泛应用于各人工智能领域。然而，随着逻辑和算法复杂度的增加，对平台的算力需求日益增加，这一问题在资源受限的嵌入式异构平台尤为突出，已成为研究热点。深入分析国内外并行计算引擎和软件定义架构的发展规律，探索软件定义架构的发展特点，提出一种基于软件定义的并行计算架构（SDPCA）方案。该方案采用“数据-控制-能力”模块化设计结构，并综合运用软件定义资源、动态调度策略、虚拟化内存、算子优化等关键技术，有效提升 SDPCA 运行的计算效率，并具备轻量化和动态重构特征。在嵌入式异构平台典型硬件条件下，使用神经网络卷积计算和模型推理对该计算架构进行测试，实验结果表明，SDPCA 功能稳定、运行准确，卷积计算性能较 OpenCL 提升 4 倍以上，在 4 种常见神经网络模型的图像推理测试中，SDPCA 识别效率较 OpenCL 提升 2.6 倍以上，在不同模型计算时抖动更小。在识别准确率方面，SDPCA 对 4 种模型的识别准确率均在 91.7%以上，推理平均准确度较 OpenCL 提升 3.9 百分点，功能和性能指标符合设计预期，具有一定的参考价值。

关键词： 神经网络；嵌入式异构平台；软件定义架构；分布式计算；算子优化

中图分类号： TP391　　**文献标志码：** A　　**DOI：** 10.19678/j.issn.1000-3428.0070349

Design and Implementation of Parallel Computing Architecture Based on Software Definition

NIE Fei, JIANG Bo

(The 32nd Research Institute of China Electronics Technology Group Corporation, Shanghai 200233)

【Abstract】 In recent years, with the rapid development of Artificial Intelligence (AI), neural networks have been widely used in various artificial intelligence fields. However, the demand for computing power is increasing with the complexity of logic and algorithms. This issue is particularly prominent in resource-constrained embedded heterogeneous platforms. Therefore, it has become a research hotspot. This paper analyzes the development of parallel computing engines and software-defined architecture globally, explores the characteristics of software-defined architecture, and proposes a design scheme for a Parallel Computing Architecture based on Software Definition (SDPCA). This computing architecture adopts a modular structure of "data-control-capability" and comprehensively utilizes key technologies of parallel computing architecture such as software-defined resources, dynamic scheduling strategies, virtualized memory, and operator optimization. These technologies effectively improve the runtime computational efficiency of SDPCA, providing a lightweight framework with dynamic reconstruction features. Under the typical hardware conditions of an embedded heterogeneous platform, SDPCA is tested using neural network convolutional layer computation and model inference. The experimental results show that the SDPCA function is stable and runs accurately, and that the computational performance of the convolutional layer is more than four times higher than that of OpenCL. The image inference tests of the four common neural network models reveal that the recognition efficiency of SDPCA increases by more than 2.6 times compared to OpenCL and it is more stable in different model calculations. Additionally, SDPCA achieves a recognition accuracy of over 91.7% for all four models, with an average inference accuracy improvement of 3.9 percentage points compared with OpenCL. The functional and performance indicators meet the design expectations and have certain reference values.

【Key words】 neural network; embedded heterogeneous platform; software defined architecture; distributed computing; operator optimization

基金项目： 国家部委基金。

作者简介： 聂飞，男，高级工程师、博士，主研方向为操作系统、计算中间件技术；江波，研究员、博士。

收稿日期： 2024-09-10　　**修回日期：** 2024-12-02　　**E-mail：** niefnet@163.com

0 引言

过去几年间，随着人工智能(AI)技术的不断创新演进，中国已成为全球 AI 技术创新和场景应用方面领先的国家，相关技术已广泛应用于自动驾驶、智能医疗、人像识别等领域[1]。人工智能的推广和应用使得算力需求呈爆发式增长，AI 计算需求以每 4 个月翻一倍的速度增加，这比摩尔定律更高效[2]。在嵌入式领域，作为算力基础设施的嵌入式处理器已很难凭借一己之力完成人工智能算力需求，基于中央处理器(CPU)＋现场可编程门阵列(FPGA)＋可扩展处理器单元(XPU)的异构并行计算系统有望成为解决这一问题的关键。本文收集了近年来并行计算和软件定义架构相关的中英文实证文献，以软件定义思想为主线，提出一种基于软件定义的并行计算架构(SDPCA)，并搭建实验环境验证该方案的合理性。

1 背景概述

1.1 软件定义技术

“软件定义”最早于 1992 年 5 月由 MITOLA 博士[3]在美国电信会议上提出“软件无线电”的概念演进而来。1999 年联合战术无线电系统(JTRS)联合计划办公室推出首个软件通信体系(SCA)结构标准版本，通过对硬件设备进行抽象化描述，向上提供统一的互联互通操作。

在软件定义技术发展的过程中，相关软件定义架构不断完善，软件定义架构可以从架构的层面更好地解决软硬件解耦、系统灵活性和资源管理等问题。近年来，软件定义架构逐步向更直观的模块化、组件化方向发展。2018 年，美国国防部联合战术网络中心(JTNC)正式提出模块化体系架构(MRA)的概念，将包括软硬件在内的模块化思想带到软件定义架构的设计规范中，并逐步成为发展“云-边-端”结构的关键技术点[4]。结合软件定义技术在通信系统的发展趋势，本文合理推测软件定义架构与计算相结合具备较大的可行性，未来计算资源的管理与部署很大可能依托软件定义来实施。

1.2 并行计算架构

并行计算架构的发展与当今人工智能日益增长的网络规模息息相关，传统的单 CPU 计算已成为技术迭代的瓶颈。近年来，基于并行处理的计算架构具有明显优势，其中以统一计算设备架构(CUDA)和开放计算语言 OpenCL 最为典型。

CUDA 由 NVIDIA 推出，该架构通过抽象计算资源和存储资源向开发者提供并行编程模型，在计算架构模型上 CUDA 将资源划分为网格(Grid)、块(Block)和线程(Thread)3 个层次，其计算模型架构如图 1 所示。CUDA 计算架构 CPU 主机端由一系列计算任务组成，这些任务通过编程框架接口将计算交给图形处理器(GPU)设备端，GPU 设备端的网格由多个块组成，块又分成若干个并行执行的线程，这些块和线程资源共同构成了一个 CUDA 程序执行的整体[5]。

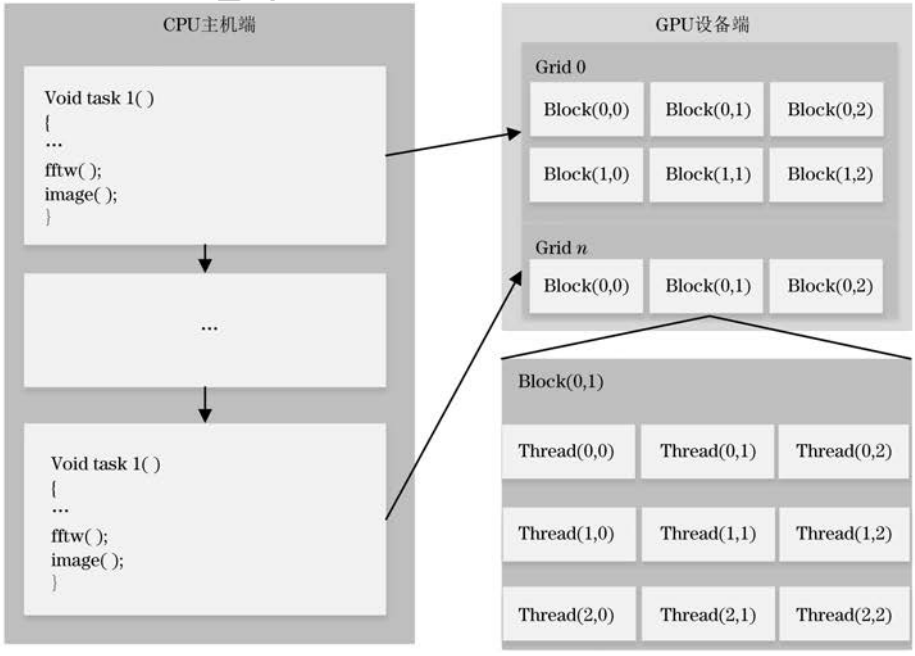

图 1 CUDA 计算模型架构

Fig. 1 Architecture of CUDA computing model

OpenCL 是针对各种异构硬件加速器的跨平台并行编程标准，其定义了平台模型、执行模型、存储模型和编程模型 4 个抽象模型。平台模型包括 1 个主机端和与之连接的数个 OpenCL 设备，又称“硬件加速器”，设备被划分为多个计算单元，计算单元又被进一步划分为多个处理单元。OpenCL 计算平台模型如图 2 所示[6-7]。OpenCL 执行模型将程序分为执行在主机端的程序和设备端内核程序。存储模型涵盖了程序运行时平台所利用的存储架构、内容和行为。编程模型具有显式的逻辑控制，可以选择执行的程序、设备和数据存储区域[8]。

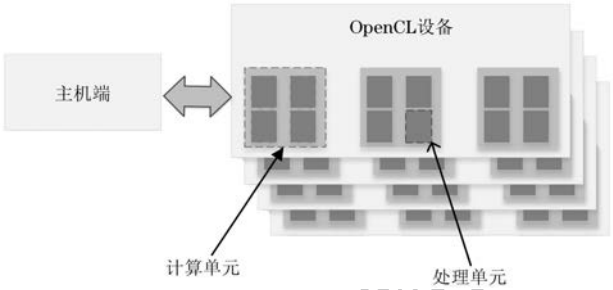

图 2 OpenCL 计算平台模型

Fig. 2 OpenCL computing platform model

OpenCL 与 CUDA 两种计算架构各有所长，跨平台性和通用性是 OpenCL 的优势，但 CUDA 占有 GPU 主流市场。目前，新一代处理器架构以及基于单指令多数据流(SIMD)技术的计算架构出现，未来计算架构的发展存在太多不确定性。随着技术的演进，CUDA 和 OpenCL 都不是计算架构的终点[9]。

2 SDPCA 设计与实现

2.1 软件定义架构总体设计

SDPCA 是在 SCA 和软件定义网络(SDN)思想的基础上，融合模块化、组件化和资源虚拟化的思想，将计算从架构上分为操作系统域、系统服务域和统一接口域，在架构核心的系统服务域基于模块化思想构建了数据、控制和能力 3 个平面。基于平面特征对计算资源进行软件化定义，架构维护了一个可动态重构的虚拟计算资源池，不同应用组件能够在相同标准的架构之间实现移植与重用，其系统架构如图 3 所示。

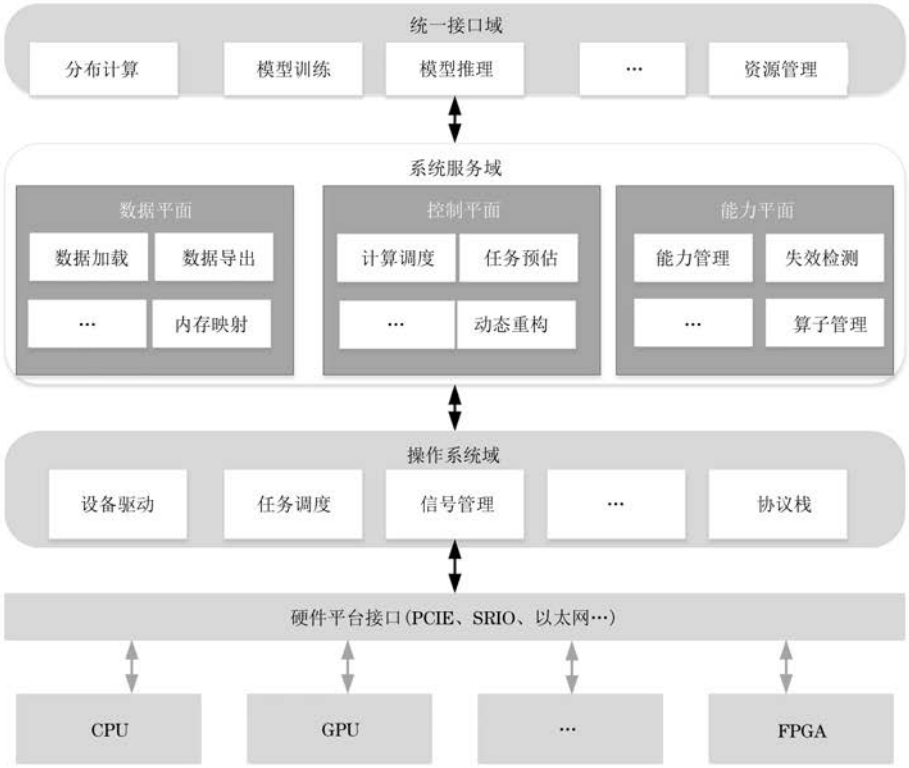

图 3 SDPCA 软件定义架构总体设计

Fig. 3 Overall design of software-defined architecture for SDPCA

SDPCA 架构的底层由一系列 CPU、GPU、信号处理器(DSP)、FPGA、神经网络处理器(NPU)等组成，这些处理器是构成高性能计算的基本物理单元。操作系统域主要是整个架构的基础软件支撑，也是底层硬件加速器软件化的转接口，通过设备驱动将硬件能力注册给上层系统服务域，该域主要包含设备驱动、任务调度、信号管理、内存管理以及各类协议栈。系统服务域是 SDPCA 的核心，划分为数据平面、控制平面、能力平面。数据平面负责所有数据的加载、转发和映射等，是数

据快速交换的主体。控制平面负责计算任务预处理、负载分隔、调度分配、动态重构等功能，并支持扩展定制。能力平面主要负责系统内计算节点能力注册、能力重构、算子管理等功能。统一接口域向上层应用提供了跨平台编程接口，兼容支持 NCNN、MNN 等框架，实现 SDPCA 的跨平台兼容性。

2.2 控制平面设计

逻辑控制是构建整个系统软件定义计算能力的大脑，一个优秀的计算模型应该具有良好的可扩展性、可靠性和灵活性，可以面向不同场景的应用，具备软件定义的能力[10-12]。传统并行计算系统中将平台模型划分为主机端和设备端。主机端由 CPU 承担，通过给设备端发送命令并接收设备处理后的反馈信息，负责管理任务队列并安排新任务进入队列等[13-14]。设备端由一系列 GPU、NPU、FPGA 等硬件加速器组成，是高性能计算的实际执行体，完成计算并将结果反馈给主机端系统。

并行计算系统中影响其整体计算能力的主要因素有主机调度能力、设备计算能力、设备数量和高速总线数据通信能力等，其中主机调度能力的影响因素包含任务数量、调度策略、数据块大小和数据的规律性等，理论上在各影响因素均不构成瓶颈的前提下，系统整体计算能力与影响因素成正相关性。为了明确并行平台模型与系统计算能力的约束关系，本文使用单核 ARM 处理器(主频 1.5 GHz)连接不同数量的 FPGA 设备仿真系统整体计算能力。图 4 为仿真系统计算能力数据图。

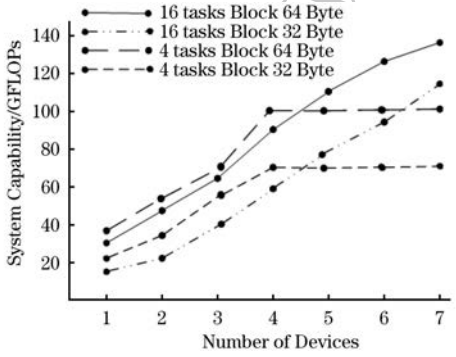

图 4 并行计算能力曲线

Fig. 4 Parallel computing capability curve

仿真结果表明，并行计算系统的能力与多个相关因素关联，其中增加计算设备数量可以明显提升系统的计算能力。当设备数量一定时，增加主机并行任务数量会给任务调度带来额外的开销，反而降低系统的计算能力。因此，并行计算系统的最大计算能力值应由一组相关因子的集合来表征，记第 i 个设备计算力为 $D[i]_{cap}$，调度策略能力 S_{cap}、内存交换能力 M_{cap}，则系统计算能力 C_{cap} 表达式如式(1)所示：

$$C_{cap}=\min\left\{S_{cap},M_{cap},\cdots,\sum_{i=0}^{n}D[i]_{cap}\right\} \tag{1}$$

结合并行计算以上特征，为最大化系统的计算能力，本文基于软件定义思想构建 SDPCA 框架的控制平面，实现主机端与设备端根据计算需求动态协同的计算功能，降低短板因子对系统计算能力的影响。在控制平面中设计任务预估模块、能力重构模块和能力数据库，任务预估模块由计算特征获取、加速器特征提取和任务评估 3 个部分组成。图 5 为任务预估模块和能力重构模块关系。

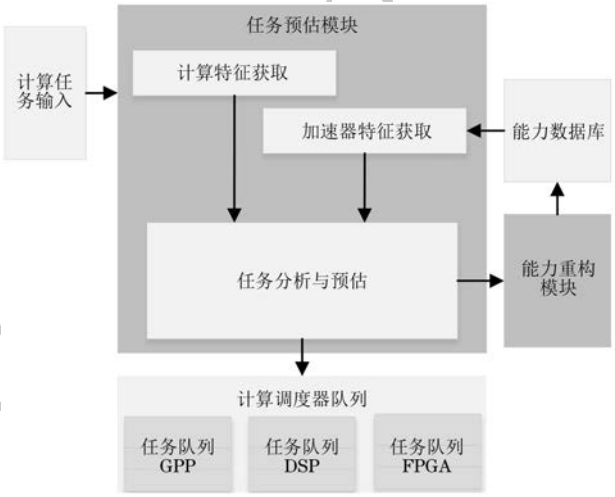

图 5 任务预估模块与能力重构模块关系

Fig. 5 Relationship between task estimation and capability reconstruction modules

计算特征主要通过输入计算任务的静态代码来获取，由前端编译器对静态代码分析来确定。例如一个神经网络计算任务，在任务加载时可以快速完成计算任务中卷积、池化、归一化、采样等大类别划分，并将大类别中的输入数据块属性和相关算子的调用次数等信息作为计算任务特征数据集，由任务分析与预估模块将特征化后的计算任务切割为加速器设备子任务，通过查找能力数据库中设备特征获取当前可用加速器能力，并与本次计算特征进行匹配，评估该计算任务在对应设备上的运行时间。当能力数据库中加速设备能力不满足任务要求的算子功能或执行时间要求时，则能力重构模块对加速设备进行能力重构，并将能力描述信息更新到数据库。能力重构子模块则根据计算任务特征使用主机端与设备端交互接口将算子动态部署到对应的加速器设备。主机端与设备端交互接口定义如表 1 所示。

表 1 SDPCA 接口功能定义

Table 1 SDPCA interface functions definition

序号	接口名称	接口用途
1	SDPCA_GetPlatformInfo	获取平台相关信息
2	SDPCA_GetDeviceCap	获取设备能力信息
3	SDPCA_RegDeviceCap	系统注册能力
4	SDPCA_RebuildDeviceCap	重构设备能力
5	SDPCA_CreateCmdQue	创建计算命令队列
6	SDPCA_CreateContext	创建计算上下文
7	SDPCA_CreateKernel	将核算子加载到系统
8	SDPCA_SetKernelArg	设置算子参数
9	SDPCA_TaskEmqueue	设置任务入队

在计算调度方面，技术基本走向静态调度和动态调度 2 个方向[15]。文献[16]在有向无环图的基础上通过对应用程序进行纵向切割，提供异步执行、空间共享和传输计算重叠，而无需事先了解有关程序依赖结构的任何信息进行静态调度，但依赖关系复杂，背离了并行计算的特征。文献[17]提出一种并行计算中基于长延迟操作的 Warp 调度方案，它是一种横向切割的方案，在进行调度时 Warp 根据执行指令的特征被划分到不同的区域中，但该方案中 GPU 内存层级的不断细分会导致并行计算结构逐渐复杂。文献[18]通过建立计算节点的利用率模型，分析作业执行能耗和贪心算法来判断最佳作业调度方法。文献[19]采用计算和数据迁移进行统一管理概念，使用专用 TensorCore 硬件将数据单独映射到线程。上述方法大多数对内核执行静态分析以确定调度策略，使用上较为刻板。动态调度可以解决系统运行时的调度需求，一方面在静态调度出现无法准确分析的情况时，动态调度可以进行实时处理，另一方面可以减少调度所带来的开销[20]。文献[21]提出了 GPU Navigator 技术，一种基于查找表的动态多内核调度程序，可通过在线分析最大限度地提高整体性能。

本文引入工作负载切割、模拟采样运行以及缓存一致性处理等模块。通过精确时序模型对硬件加速器中线程的执行进行动态实时分析和调度，以满足计算时间期限的要求。当硬件加速器进行并行加速时，在运行时状态根据硬件加速器中各线程工作负载和内置调度检测点动态分析各加速器负载状态，进而动态地将工作分配到相对空闲线程中执行。图 6 给出了计算调度器融合静态调度和动态调度执行示意图。在一个包含傅里叶变换、图像处理和滤波处理的程序中，通过静态调度将任务数据部署到不同的虚拟内存空间，动态调度则在程序执行过程中插入调度检测点，实时控制计算任务在加速器中线程执行流程。

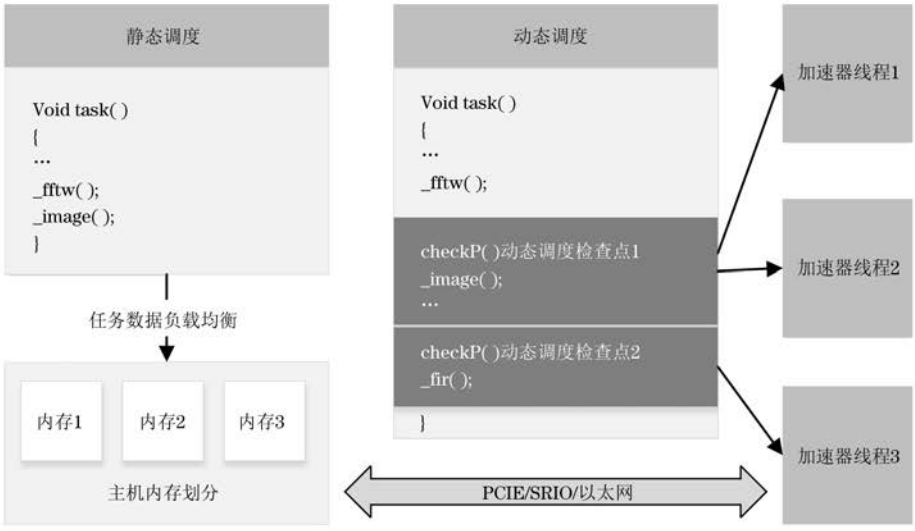

图 6 计算调度器执行示意图

Fig. 6 Schematic diagram of computing scheduler execution

2.3 数据平面设计

数据平面主要解决主机与加速设备之间的数据传输问题，目前的计算框架广泛存在数据冗余复制的开销，主要体现在数据传输和框架数据格式的转换过程[22]。本文设计一个极简协议分布式软总线，该软总线通过虚拟内存管理单元在主机端虚拟一系列的计算内存块，这些内存块驻留在主机端物理内存中，由虚拟内存管理单元负责主机端和设备端之间的统一编址，摒弃了复杂的表示层、会话层、传输层等协议交互流程，使得性能

无限逼近硬总线能力。在物理层直接调用RDMA、RapidIO等高速接口进行内存数据传输，在主机端和设备端之间形成一个共享内存空间，在整个系统运行时始终保持主机内存、加速设备内存和高速缓存数据的一致性。图7描述了数据平台内存管理模型，为了与其它计算框架保持兼容，在主机端内存模型上沿用OpenCL内存变量概念，共有3个内存类型：数组缓存(buffer)、图像对象(image)、管道对象(pipe)，其中数组缓存与C语言中用malloc申请空间相似，数据在地址空间上是连续的[23-24]。图像对象与普通的数组不同，图像数组中相邻的数据在内存中不一定连续存储，其设计目的是充分发挥硬件对空间局部性的优势，并利用设备硬件加速的能力，实现对图像数据的高效处理。管道对象是一种数据元素队列，遵循先进先出(FIFO)的原则，以支持特定类型的并行计算任务[25]。设备端定义了全局内存、常量内存、本地内存和私有内存。全局内存对于执行内核中的每个工作项都是可见的，是进行主机端和设备端传输时使用的内存。常量内存为所有的工作项访问，用于传入配置参数，本地内存用于同一工作组内工作项之间的共享，通常使用设备端的片内内存，私有内存则只在工作项内访问。

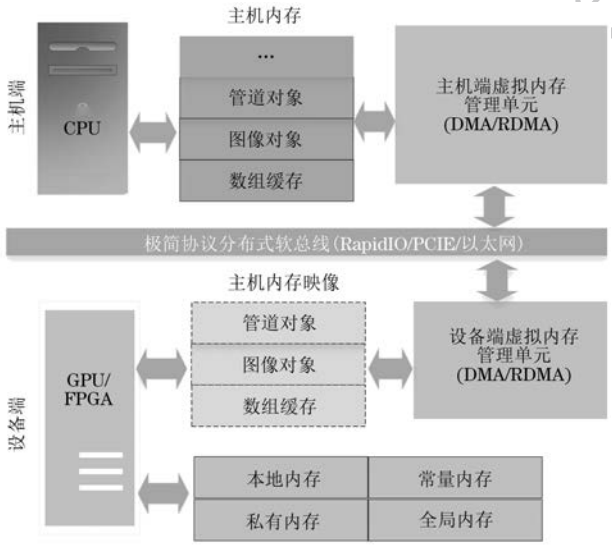

图7 数据平台内存管理模型

Fig.7 Memory management model of data platform

2.4 能力平面设计

能力平面主要实现设备能力和算子能力的动态构建，从系统层面抽象描述资源计算能力，设备能力是对硬件资源的一种软件化表达，通过框架提供的接口向系统注册。本文沿用欧洲安全软件无线电(ESSOR)规范将设备划分为高速通用处理器(GPP)、DSP和FPGA[26]。本文定义系统中GPP资源由N个特定拓扑互联的GPP设备组成，建模长度为N的GPP设备链表，每个GPP设备都动态维护了M种匹配特性和A种分配特性，分别用P_M和P_A表达，系统中GPP设备建模如式(2)所示：

$$\mathrm{GPP}=\begin{Bmatrix}R_M(\mathrm{GPP}_1),R_M(\mathrm{GPP}_2),\cdots,R_M(\mathrm{GPP}_N)\\R_A(\mathrm{GPP}_1),R_A(\mathrm{GPP}_2),\cdots,R_A(\mathrm{GPP}_N)\end{Bmatrix} \tag{2}$$

式中：$R_M(\mathrm{GPP}_1)$代表GPP_1设备的匹配能力，匹配能力是指计算环境中会对计算过程产生约束，但又无法定量计算的设备属性特征，主要包括处理器型号、操作系统、系统时钟等；$R_A(\mathrm{GPP}_1)$代表GPP_1设备的分配能力，是设备中对计算过程产生定量影响的资源能力属性，主要包括核心数量、主频、内存大小、算子能力描述表等。GPP_i设备能力的理论值由$R_A(\mathrm{GPP}_i)$决定，当一个新的计算设备GPP_i注册到系统后，能力平面自动完成该设备软件化描述，为其构建典型匹配特性和分配特性，并全生命周期动态维护该设备的$R_A(\mathrm{GPP}_i)$，系统中计算设备的P_M和R_A描述如图8所示。

算子是构建能力平面的基础，在神经网络计算中表示为核函数，由异构编程环境中特定的程序语言实现[27]。当核函数在异构设备上运行时，通过对异构设备的资源进行分块处理，使计算能够分配在多个计算单元中并行执行。HACHEM等[28]提出在FPGA加速器平台上通过调整内存布局、循环展开等方法对SHA-256安全哈希算法进行加速，性能得到4.3倍的提升。同时他们实现的一个SHA-3协处理器，通过核函数将SHA-3算法的速度提升了310倍[29]。文献[30]研究了2D卷积核的不同参数在Intel FPGA、CPU和GPU平台上的能耗、性能等，实验发现FPGA平台在9×9次参数输入下获得较CPU和GPU平台更优的性能，但在其他参数输入下，GPU则具有更高的性能。

本文研究了相关核函数转换的基本规则，在SDPCA框架下对相关核函数进行移植，并基于SIMD技术进行性能优化。SIMD可以实现单指令多组数据处理，由于多个并行处理器单元共享指令、译码逻辑，使得SIMD结构可以获得较高的性能功耗比。本文最终完成了近100个核函数，包含字符串分割、字符串连接、傅里叶变换、矩阵求逆等。表2为经SIMD优化前后典型核函数性能实测数据，多数算子经SIMD优化后计算性能提升了2倍以上，部分算子性能提升了17倍。

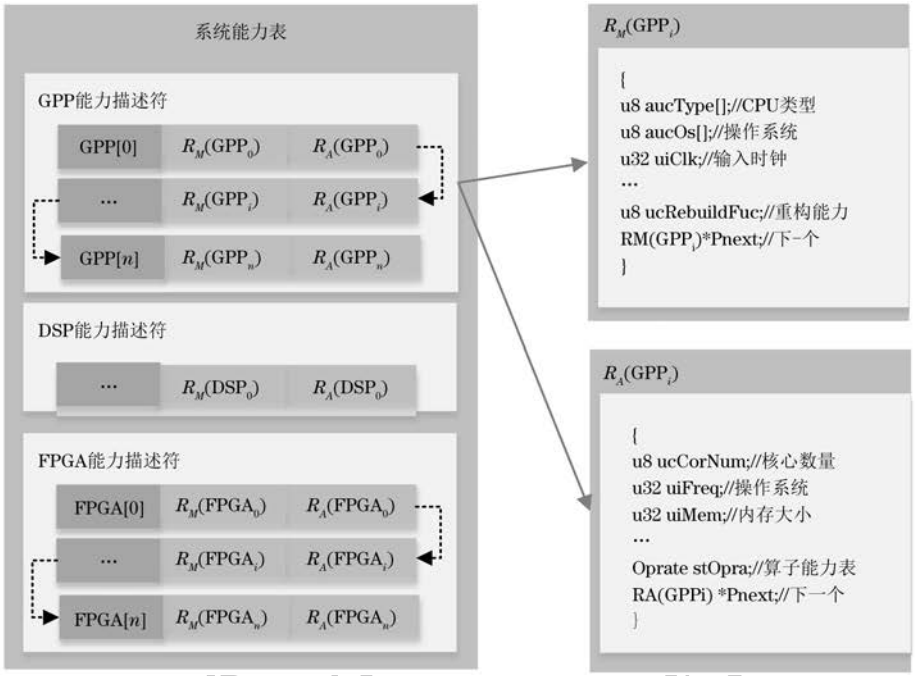

图 8 系统能力描述

Fig. 8 System capability description

表 2 典型算子经 SIMD 优化前后性能

Table 2 Performance of typical operators before and after SIMD optimizations

单位：μs

算子接口	优化前	优化后
FFT(128 点)	16.52	3.39
IFFT(128 点)	17.62	3.66
矩阵求逆(5×5)	263.73	15.39
快速卷积(128 点)	578.91	143.83
直方图(4 096 点)	535.05	202.17

3 实验结果与分析

由于 CUDA 具有闭源特性，不支持国产硬件加速器，因此本文实验选用开源 OpenCL 基于同一国产硬件加速器平台与 SDPCA 进行对比验证，通过神经网络计算和推理实验评估 SDPCA 在计算功能、性能方面的效果。

3.1 实验环境

实验采用 2 块复旦微 FMQL100 平台作为硬件加速器，1 块飞腾 D2000 平台作为 CPU 控制主机，D2000 包含 8 个 FTC663 ARM 核，主频 2.3 GHz。实验中分别加载 OpenCL 和 SDPCA 计算框架，FMQL100 模块与 D2000 模块通过基于 RapidIO 协议的软总线进行互联。测试软硬件环境配置如表 3 所示。

表 3 测试软硬件环境

Table 3 Test software and hardware environment

系统架构	主机配置	操作系统	硬件加速器配置
OpenCL	CPU 飞腾 D2000@ 2.3 GHz DDR4 4 GB	ReWorks 6.1.1 (64 位)	2×FMQL100，通用算子集
SDPCA			2×FMQL100，SDPCA 优化算子集

3.2 实验结果

3.2.1 卷积神经网络计算性能测试

本文测试卷积神经网络的计算性能使用 3 个典型尺寸的卷积计算层分别测定其在 OpenCL 和 SDPCA 架构下的性能，实验中卷积尺寸层可自定义，测试数据如表 4 所示。

实验结果表明，在 1-16-16-16 卷积层尺寸下使用本文设计的 SDPCA 框架较 OpenCL 计算效率提升 3.1 倍，在 64 层和 96 层运算上，SDPCA 较 OpenCL 的计算效率分别提高 4.25 倍和 4.88 倍。

表 4 卷积计算时间对比

Table 4 Comparison of convolutional computing time

单位：ms

计算框架	(1-16-16-16) 卷积	(1-64-64-64) 卷积	(1-96-96-96) 卷积
OpenCL	6.95	53.78	64.33
SDPCA	2.25	12.63	13.16

3.2.2 神经网络推理测试

在神经网络推理实验中，使用 4 种常用神经网络基于 NCNN 推理框架分别集成 OpenCL 和

SDPCA 测试图片的识别耗时和准确率，图片源随机选用 ImageNet 图片集中大小为 224×224×3 的图片，共计 1 000 张，基于推理框架循环识别图片中的所有目标，将记录结果并直观显示对应图片中目标的推理结果。图 9 为实验中随机一张图片在 SDPCA 下的可视化推理结果（彩色效果见《计算机工程》官网 HTML 版），从图 9 可以看出，在 SDPCA 下神经网络推理框架可以准确识别出对应目标，并给出识别目标可能最高百分比。实验结果表明，SDPCA 能够正确支撑神经网络目标推理应用，功能设计正确。

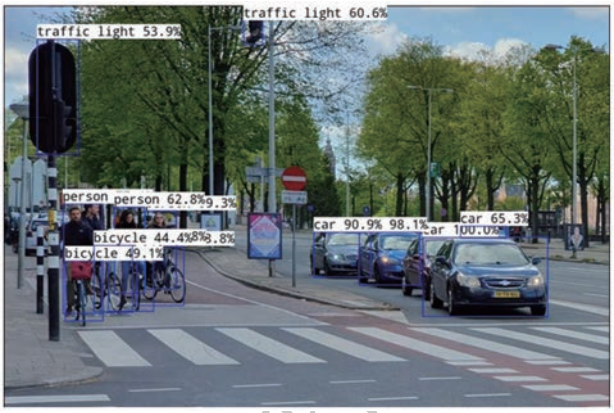

图 9 SDPCA 图像推理测试

Fig. 9 SDPCA image inference test

为验证 SDPCA 在神经网络模型推理应用场景下的性能，本文实验基于上述图片源，分别在 SDPCA 和 OpenCL 架构下进行推理测试，推理过程中通过操作系统程序自动统计图片中目标识别耗时和推理结论，实验数据统计如表 5 所示。

表 5 神经网络模型性能测试

Table 5 Performance test of neural network models

神经网络模型	OpenCL		SDPCA	
	耗时/ms	准确率/%	耗时/ms	准确率/%
ShuffleNet	1 862	90.3	512	95.1
blazeface	1 554	86.7	484	91.7
YOLO-FastestV2	1 612	89.9	577	92.5
FastestDet	1 740	90.1	551	93.2

在神经网络推理实验中，SDPCA 计算性能全面优于 OpenCL 架构，目标识别效率较 OpenCL 提升了 2.6 倍以上，在不同模型下推理计算用时更稳定，能够更好地适应不同类型的训练模型。在识别准确率方面，SDPCA 对 4 种模型的识别准确率均在 91.7%以上，高于 OpenCL 架构，且 4 种模型推理平均准确度较 OpenCL 提升了 3.9 百分点，这主要获益于 SDPCA 架构基于“数据-控制-能力”的轻量模块化设计、核函数的优化、高速数据交换和软件定义动态重构算子能力，使得架构可以主动识别计算流程的最佳并发执行方法，并更新算子匹配能力，从而提升神经网络模型的计算效率、准确度和结果稳定性。

4 结束语

本文提出一种基于软件定义的并行计算架构，该计算架构创新性采用“数据-控制-能力”模块化设计，使得计算资源具备依据计算需求动态协同自组织重构能力，同时优化了相关算子函数，可用于资源受限的嵌入式平台。实验结果表明，SDPCA 具有较优的卷积计算性能和识别准确率。虽然这些研究已取得了些许成果，但调度算法粒度比较粗，数据平台软总线功能不完善，对 GPU 的支持方面，后续将开展 SDPCA 框架主机端相关接口适配，移植或重构设备端内核函数，使 GPU 设备的工作组和工作项尽可能向量化和并行化，从而提升 GPU 并行计算能力。下一步研究方向主要集中在数据平台软总线功能完善、调度算法细化和国产 GPU 支持方面，进一步提升计算框架的完备性。

参考文献

[1] 姜李丹，薛澜. 我国新一代人工智能治理的时代挑战与范式变革[J]. 公共管理学报，2022，19(2)：1-11，164.
JIANG L D，XUE L. The current challenges and paradigm transformation of new-generation AI governance in China [J]. Journal of Public Management，2022，19(2)：1-11，164. (in Chinese)

[2] KONSTANTIN F P，SANDERS J，RAHMAN R，et al. Trends in AI supercomputers [EB/OL]. [2024-08-10]. https://arxiv.org/pdf/2504.16026.

[3] MITOLA J I. Software radios：survey，critical evaluation and future directions[C]//Proceedings of NTC-92：National Telesystems Conference. Washington D. C.，USA：IEEE Press，1992：13-23.

[4] 庄佩文. 基于软件定义架构的计算资源管理与部署研究[D]. 长沙：国防科技大学，2021.
ZHUANG P W. Research on management and deployment of computing resource based on software defined architecture [D]. Changsha：National University of Defense Technology，2021. (in Chinese)

[5] HWU W M，RODRIGUES C，RYOO S，et al. Compute unified device architecture application suitability [J]. Computing in Science & Engineering，2009，11(3)：16-26.

[6] 孙祥杰，朱亮，余同欢. 基于 OpenCL 的 SAR 影像快速浏览方法研究[J]. 电子质量，2023(3)：24-30.
SUN X J，ZHU L，YU T H，Research on the fast browsing method of the SAR images based on OpenCL[J]. Electronics Quality，2023(3)：24-30. (in Chinese)

[7] TANG T，LU K，PENG L，et al. SNCL：a supernode OpenCL implementation for hybrid computing arrays [J]. The Journal of Supercomputing，2024，80：9471-9493.

[8] 林晓泽. 基于异构计算的高效并行静态学习研究[D]. 汕头：汕头大学，2022.
LIN X Z. Research on high-performance parallel static learning based on heterogeneous computing [D]. Shantou：

Shan Tou University, 2022. (in Chinese)
[9] 王琳博,白林亭,文鹏程.国产深度学习推理框架嵌入式适用性研究[J].航空计算技术,2023,53(3):121-125.
WANG L B, BAI L T, WEN P C. Research on domestic deep learning inference engine's embedded applicability [J]. Aeronautical Computing Technique, 2023, 53(3): 121-125. (in Chinese)
[10] ZENG J, KOU M Y, YAO H L. KunlunTVM: a compilation framework for Kunlun chip supporting both training and inference[C]//Proceedings of the Great Lakes Symposium on VLSI 2022. New York, USA: [s.n], 2022: 299-304.
[11] 张登科. Spark 分布式计算平台性能优化研究[D].西安:西安电子科技大学, 2022.
ZHANG D K. Research on performance optimization of Spark distributed computing platform [D]. Xi'an: xidian university, 2022. (in Chinese)
[12] 史健男. 基于异构平台的实时操作系统推理框架算子加速器的实现[D].成都:电子科技大学,2024.
SHI J N. Implementation of operator accelerator for real-time operating system inference framework on heterogeneous platform [D]. Chengdu: University of Electronic Science and Technology of China, 2024. (in Chinese)
[13] JI F, LIN H, MA X. RSVM: a region-based software virtual memory for GPU[C]//Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques. Washington D. C., USA: IEEE Press, 2013: 269-278.
[14] RAVI P K. FPGA acceleration of CNNs using OpenCL[D]. Tempe: Arizona State University, 2020.
[15] 袁野.基于 OpenCL 的多 GPU 调度系统的设计与实现[D].杭州:浙江大学,2022.
YUAN Y. Design and implementation of multi-GPU scheduling system on OpenCL [D]. Hangzhou: Zhejiang University, 2022. (in Chinese).
[16] PARRAVICINI A, DELAMARE A, ARNABOLDI M, et al. DAG-based scheduling with resource sharing for multi-task applications in a polyglot GPU runtime[C]//Proceedings of 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS). Washington D. C., USA: IEEE Press, 2021:111-120.
[17] DO C T, CHOI H J, CHUNG S W, et al. A novel warp scheduling scheme considering long-latency operations for high-performance GPUs [J]. The Journal of Supercomputing, 2020, 76(4):3043-3062.
[18] TANG X Y, FU Z J. CPU-GPU utilization aware energy-efficient scheduling algorithm on heterogeneous computing systems[J]. IEEE Access, 2020, 8:58948-58958.
[19] HAGEDORN B, ELLIOTT A S, BARTHELS H, et al. Fireiron: a data-movement-aware scheduling language for GPUs[C]//Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques. New York, USA: ACM Press, 2020:71-82.
[20] 顾经纬,宁成明,郑启龙.基于 HXDSP 的 OpenCL 运行时任务调度[J].计算机系统应用,2022,31(11):130-138.
GU J W, NING C M, ZHENG Q L. Task scheduling of OpenCL during operation based on HXDSP [J]. Computer Systems & Applications, 2022, 31 (11): 130-138. (in Chinese)
[21] KIM J, KIM J, PARK Y. Navigator: dynamic multikernel scheduling to improve GPU performance[C]//Proceedings of the 57th ACM/IEEE Design Automation Conference. New York, USA: ACM Press, 2020:1-6.
[22] 雷斗威, 何德彪, 罗敏, 等. 基于 AVX512 的格密码高速并行实现[J]. 计算机工程, 2024, 50(2): 15-24.
LEI D W, HE D B, LUO M, et al. High-speed parallel implementation of lattice-based cryptography based on AVX512[J]. Computer Engineering, 2024, 50(2): 15-24. (in Chinese)
[23] SEUNG-HUN C. Optimization of compiler-generated OpenCL CNN kernels and runtime for FPGAs [D]. Toronto: University of Toronto, 2021.
[24] 张望.基于 FT-M7002 的 OpenCL 移植与实现[D].西安:西安电子科技大学, 2021.
ZHANG W. Porting and implementation of OpenCL based on FT-M7002 [D]. Xi'an: xidian university, 2021. (in Chinese)
[25] DANIELE A, FRANCESCO B, CRISTIANA B, et al. A runtime resource management policy for OpenCL workloads on heterogeneous multicores design [C]// Proceedings of Conference on Automation and Test in Europe. Berlin, Germany: Springer, 2019:1385-1390.
[26] HOFFMANN L. OCCAR seeks NATO standardization of ESSOR waveforms [J]. European Security & Defence, 2022(3): 38.
[27] 陈锐, 孙羽菲, 程大果, 等. TensorFlow 框架中 OpenCL 核函数的实现与优化[J]. 计算机学报, 2022, 45(11): 2457-2474.
CHEN R, SUN Y F, CHEN D G, et al. Implementation and optimization of OpenCL kernels in TensorFlow[J]. Chinese Journal of Computers, 2022, 45 (11): 2457-2474. (in Chinese)
[28] HACHEM B, YVES B, YVON S. Acceleration of the secure Hash algorithm-256(SHA-256) on an FPGA-CPU cluster using OpenCL[C]//Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS). Washington D. C., USA: IEEE Press, 2021:920-940.
[29] HACHEM B, YVES B, YVON S. An efficient OpenCL-based implementation of a SHA-3 co-processor on an FPGA-centric platform [J]. IEEE Transactions on Circuits and Systems, 2023, 70(3):1144-1148.
[30] JIN Z M, FINKEL H. Exploration of OpenCL 2D convolution kernels on Intel FPGA, CPU, and GPU platforms [C] // Proceedings of 2019 IEEE International Conference on Big Data. Washington D. C., USA: IEEE Press, 2019:4460-4465.

文字编辑 薛晋栋
栏目编辑 宋 圆