Software Engineering and Applications 软件工程与应用, 2025, 14(6), 1219-1230
Published Online December 2025 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2025.146108

Hans汉斯

一种基于DVFS特性曲线的异构计算单元低功耗协同调度方法

徐梦溪，刘姝怡，刘梓莹，丁铄阳，刘姝悦

南京工程学院计算机工程学院，江苏 南京

收稿日期：2025年10月13日；录用日期：2025年12月5日；发布日期：2025年12月16日

摘　要

针对工业边缘计算环境中任务动态到达、实时性要求高、资源受限且能耗敏感的问题，本文提出一种基于DVFS特性曲线的异构计算单元低功耗协同调度方法(简称：CS-HCU-DVFS)。该方法融合任务调度与硬件功耗管理，实现调度决策与能效调控的协同优化。框架由任务特征分析、能效感知调度引擎和DVFS协同调控模块组成：任务特征分析模块提取任务类型、数据量及截止时间；调度引擎结合异构单元的功耗－性能特性表(PPCT)，以能效比最大化为目标，查表确定最优计算单元与运行频率；DVFS模块据此动态配置电压与时钟频率，并通过抑制频繁切换提升系统稳定性。执行完成后，计算单元反馈状态信息至调度引擎，形成闭环调控机制。实验结果表明，该方法在满足任务实时性约束的前提下，有效降低系统能耗，提升整体能效。该方法调度开销低，适用于资源受限的工业边缘计算场景。

关键词

DVFS特性曲线，异构计算单元，能效感知调度，协同调控，边缘计算

A Low-Power Cooperative Scheduling Method for Heterogeneous Computing Units Based on DVFS Characteristic Curves

Mengxi Xu, Shuyi Liu, Ziying Liu, Shuoyang Ding, Shuyue Liu

School of Computer Engineering, Nanjing Institute of Technology, Nanjing Jiangsu

Received: October 13, 2025; accepted: December 5, 2025; published: December 16, 2025

文章引用: 徐梦溪, 刘姝怡, 刘梓莹, 丁铄阳, 刘姝悦. 一种基于 DVFS 特性曲线的异构计算单元低功耗协同调度方法[J]. 软件工程与应用, 2025, 14(6): 1219-1230. DOI: 10.12677/sea.2025.146108

Abstract

To address the challenges of dynamically arriving tasks, high real-time requirements, resource constraints, and energy sensitivity in industrial edge computing environments, this paper proposes a low-power cooperative scheduling method for heterogeneous computing units based on DVFS characteristic curves, named CS-HCU-DVFS. The method integrates task scheduling with hardware power management to achieve optimization of scheduling decisions and energy-efficiency control. The framework consists of three components: task feature analysis, energy-aware scheduling engine, and DVFS cooperative control module. The task feature analysis module extracts task type, data volume, and deadline constraints. The scheduling engine leverages a power-performance characteristic table (PPCT) of heterogeneous units to determine the optimal computing unit and operating frequency by table lookup, aiming to maximize energy efficiency. The DVFS module dynamically configures voltage and clock frequency accordingly, while suppressing frequent transitions to enhance system stability. Upon task completion, computing units feedback status information to the scheduling engine, forming a closed-loop control mechanism. Experimental results show that the proposed method effectively reduces system energy consumption and improves overall energy efficiency while meeting real-time constraints. With low scheduling overhead, CS-HCU-DVFS is suitable for resource-constrained industrial edge computing scenarios.

Keywords

DVFS Characteristic Curve, Heterogeneous Computing Unit, Energy-Aware Scheduling, Cooperative Control, Edge Computing

1. 引言

随着工业 4.0 与智能制造的深入推进，边缘计算作为连接物理世界与数字空间的关键使能技术，已在设备监控、质量检测、实时控制和预测性维护等工业场景中广泛应用。工业边缘网关作为现场层与云平台之间的枢纽节点，需在有限的功耗、散热与存储条件下，高效处理高并发、低延迟的多样化计算任务。

为应对日益复杂的负载需求，现代工业边缘设备普遍采用异构计算架构，集成通用处理器(CPU)、图形处理器(GPU)、现场可编程门阵列(FPGA)以及神经网络处理器(NPU)等多种专用加速器。此类架构显著提升了系统的计算灵活性与任务适配能力，但同时也带来了资源调度复杂性高、能耗管理困难等挑战。尤其在电池供电、密闭散热或高温运行环境下，系统功耗直接影响设备的运行寿命、稳定性和维护成本[1] [2]。

现有异构边缘调度研究主要围绕两类技术路线展开[3] [4]：一是基于静态规则的调度策略，如固定映射或周期性分配，其实现简单、开销低，但适应性差，难以应对动态负载变化；二是基于智能优化算法的动态调度机制，如遗传算法、粒子群优化或强化学习，虽在理论上可逼近帕累托最优解，但通常依赖复杂的数学建模与大量迭代计算，单次调度延迟较高，难以满足工业场景对实时性与确定性的严格要求。

更为关键的是，多数现有方法将任务调度决策与硬件功耗调控视为相互独立的模块，忽视了二者之间的深层耦合关系。事实上，现代处理器普遍支持动态电压频率调节(DVFS)技术，允许在运行时动态调整工作电压与时钟频率。由于动态功耗与电压的平方成正比，合理配置电压–频率对可在满足性能需求的前提下显著降低能耗。然而，如何将高层调度决策与底层 DVFS 的物理特性有效结合，实现“任务–

资源–功耗”的闭环优化，仍缺乏简洁、高效且易于部署的工程解决方案[3] [5]。

针对上述问题，本文提出一种基于 DVFS 特性曲线的异构计算单元低功耗协同调度方法(简称：CS-HCU-DVFS)。该方法摒弃复杂的在线优化模型与机器学习预测，转而采用规则驱动与查表式调控策略，通过预先建模的 DVFS 特性曲线建立任务负载与电压/时钟频率之间的映射关系，实现调度决策与硬件功耗调控的紧耦合。所提方法在保障任务实时性的同时显著降低系统能耗，具备良好的可部署性与鲁棒性。

本文的主要贡献如下：

(1) 提出一种基于 DVFS 特性曲线驱动的异构计算单元协同调度框架，通过建立任务负载与电压/时钟频率的映射关系，实现任务调度决策与硬件功耗调控的联动优化，有效解决了传统方法中资源分配与能效管理相分离的问题。

(2) 设计轻量级的任务优先级评估与能效感知任务分配机制，综合考虑任务截止时间、数据量及异构计算单元的能效特性，在保障实时性约束的前提下提升系统整体能效，适用于资源受限的边缘计算场景。

(3) 在搭建的工业边缘网关实验平台进行验证，结果表明，与 HEFT、MCT、DVFS-Util 等基准策略相比，CS-HCU-DVFS 在系统功耗、任务响应延迟和运行稳定性方面均具有优势，验证了其在典型工业边缘场景下的可行性与有效性。

2. 相关工作

异构边缘环境下的任务调度研究主要围绕性能、能耗与可靠性等多目标展开[2] [3]。早期工作多采用静态划分策略，将任务预先映射至特定硬件模块，适用于负载稳定的场景，但在动态环境中易导致资源利用率不均衡。近年来，动态调度机制成为研究热点。部分研究采用启发式算法，如最早完成时间(HEFT)、最小完成时间(MCT)等，依据任务依赖图与资源状态进行调度决策。此类方法计算复杂度较低，具备较好的实时响应能力，但通常未充分考虑能耗因素，难以满足能效敏感型应用的需求。为提升优化能力，另一类研究引入多目标优化框架，通过遗传算法、粒子群优化(PSO)或强化学习求解帕累托最优解。尽管此类方法在理论上具有优越性，但其收敛过程依赖大量迭代计算，单次调度延迟常达数十毫秒以上，难以满足工业场景对低延迟与高确定性的严格要求。

进一步分析发现，多数现有调度模型假设硬件性能恒定，忽略了动态电压频率调节(DVFS)对能效的显著影响，导致调度决策与实际功耗表现脱节[3]-[5]。事实上，DVFS 技术通过调节处理器的工作电压与频率，能够在性能与功耗之间实现动态平衡，在移动设备与数据中心中已被广泛用于节能管理。近年来，研究者尝试将其引入边缘计算领域，以降低系统能耗。现有 DVFS 控制策略大致可分为三类：其一为基于利用率的阈值控制，当 CPU 利用率低于某一阈值时自动降频；其二为基于预测的自适应控制，利用时间序列模型预测未来负载并预调频率；其三为基于学习的优化控制，通过强化学习在线学习最优电压频率点。然而，这些方法多针对单一处理器架构设计，难以直接推广至包含 CPU、GPU、NPU 等多种异构计算单元共存的工业边缘网关。此外，预测与学习类方法依赖大量历史数据进行模型训练，在冷启动或负载突变场景下易出现控制滞后或震荡，稳定性不足。

更关键的是，现有研究大多将 DVFS 调控与任务调度决策割裂处理，调度器仅关注任务分配与执行顺序，而 DVFS 控制器独立运行，缺乏协同机制。这种解耦模式限制了系统整体能效的进一步提升。为此，协同调度理念强调软件调度策略与硬件资源管理的深度融合。部分研究提出了“计算–通信–能耗”联合优化框架，试图实现全局最优[4] [6]。然而，这类方法通常依赖精确的能耗建模与复杂的求解器，实现成本高，维护难度大。相比之下，工业实践更倾向于采用简单、稳定、易于部署的调度机制，对算法的鲁棒性与可解释性有更高要求。

针对上述问题，本文面向工业边缘场景的实际需求，提出一种兼顾能效与实时性的轻量级协同调度

方法。在保障任务实时性与系统稳定性的前提下，摒弃复杂的在线建模与迭代求解过程，而采用规则驱动与查表式调控策略，通过构建 DVFS 特性曲线与任务负载的映射关系，实现调度决策与硬件功耗调控的紧耦合，以提升方法在真实工业边缘场景下的可部署性与适应性。

3. CS-HCU-DVFS 框架设计

CS-HCU-DVFS 是一种基于 DVFS (Dynamic voltage and frequency scaling, DVFS)特性曲线的异构计算单元低功耗协同调度方法，其核心思想是打破任务调度与硬件功耗管理之间的壁垒，实现“任务–资源–功耗”的一体化协同优化。

基于此，本文设计了结构化的协同调度框架。如图 1 所示，该框架由三大核心模块构成：任务特征分析模块、能效感知调度引擎和 DVFS 协同调控模块。三者通过事件驱动机制紧密协作，构建从任务输入到资源分配再到动态功耗调控的闭环流程，在保障实时性的同时最大化系统能效[7]-[9]。

调度框架输入源(任务输入)来自工业设备(传感器、摄像头等)，经特征提取后由调度引擎决策资源分配，并通过 DVFS 协同调控模块联动底层功耗管理。DVFS 特性曲线数据库(Power-performance characteristic table, PPCT 表)为调度与调控提供统一能效参考，实现“前向协同、查表驱动”的低功耗优化机制。能效感知调度引擎模块是决策中心，需根据执行结果动态调整后续调度策略(如重调度、负载均衡)，反馈指向该引擎，反馈内容包括任务完成状态、实时负载(CPU/NPU/GPU 利用率等，反馈的作用支持动态调度、避免过载、实现闭环优化。

3.1. 任务特征分析模块

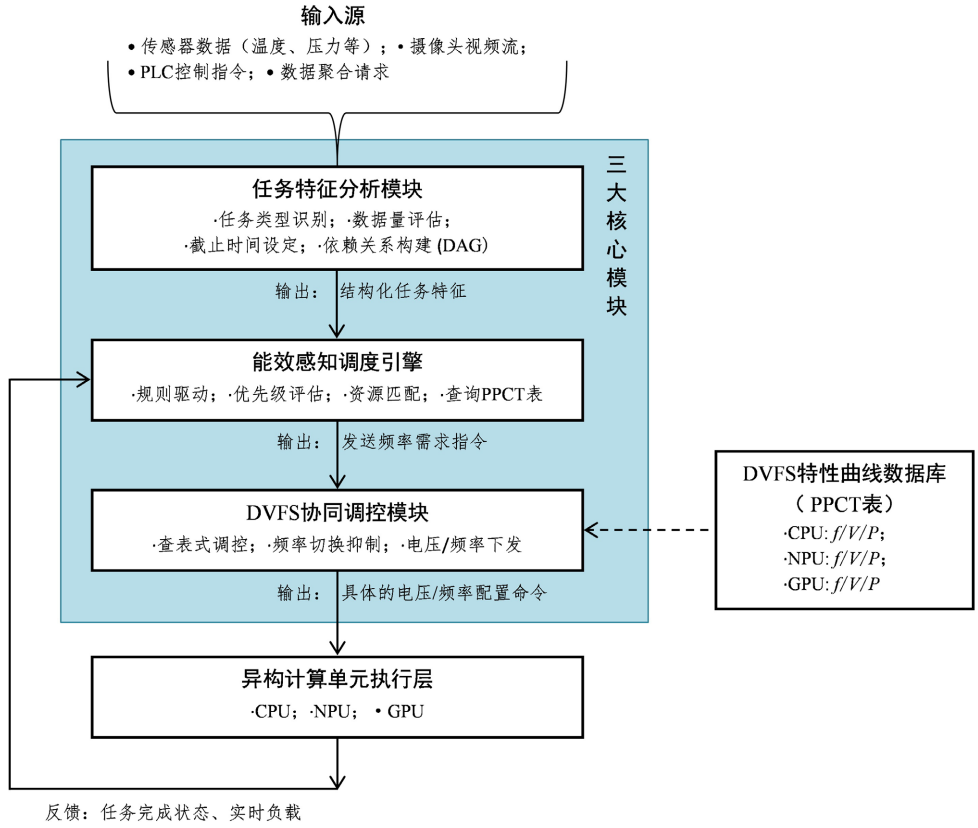

Figure 1. Architecture of the CS-HCU-DVFS collaborative scheduling framework
图 1. CS-HCU-DVFS 协同调度框架结构图

该模块是系统的任务入口与预处理中心，负责接收来自工业边缘设备的各类动态任务输入，主要包括：传感器数据(如温度、压力、振动监测)，摄像头视频流(用于视觉检测或安全监控)，PLC 控制指令(实时控制信号)，数据聚合请求(多源数据汇总任务)。

模块对接收到的任务进行实时解析，提取其关键属性，主要包括：任务类型(控制、图像处理、数据聚合等)，数据量大小，截止时间要求，可并行性与依赖关系(通过任务依赖图 DAG 建模)。解析结果以标准化的任务描述向量形式输出，作为调度引擎的输入。该模块还维护任务队列与 DAG 结构，支持多任务流水线调度，确保复杂任务的有序执行。

3.2. 能效感知调度引擎

该模块是调度框架的决策核心，采用规则驱动的轻量级算法，综合任务的截止时间紧迫度、数据依赖关系及各异构计算单元(CPU、NPU、GPU)的能效特性，计算任务的优先级与最优执行单元。

不同于依赖复杂优化模型或机器学习预测的方法，本引擎通过预定义的调度规则表进行快速匹配，单次决策耗时低于 1 ms，满足工业边缘场景的实时性要求。

其主要功能包括：

(1) 优先级评估：基于截止时间与任务类型确定调度顺序；

(2) 资源匹配：结合各计算单元的当前负载与可用频率档位，选择最优执行单元；

(3) 能效感知：查询 DVFS 特性曲线数据库(PPCT 表)，获取不同频率下的能耗代价，优先选择“能效比(Energy efficiency ratio, EER)最优”的执行路径；

(4) 避免过载：动态感知资源状态，防止任务堆积与资源争用。

3.3. DVFS 协同调控模块

该模块是调度框架的硬件联动执行单元，负责实现调度决策与底层功耗管理的紧耦合。其核心是维护一个 DVFS 特性曲线数据库(PPCT 表)，记录各异构计算单元(CPU、NPU、GPU)在不同频率档位下的实测功耗数据(包括静态与动态功耗)。该表作为调度与调控的共同知识源，支撑“查表式”快速决策。

当调度引擎完成任务分配后，本模块根据：任务负载强度、执行时间约束、 PPCT 表中记录的能效特性，查表选择最优的电压/频率组合，在保障性能的前提下最小化能耗。

同时，模块引入频率切换抑制机制，限制单位时间内的调频次数，避免因频繁切换引发电压波动与任务延迟抖动，提升系统稳定性。

3.4. 异构计算单元执行层

图 1 底部的“异构计算单元(CPU、NPU、GPU)”为任务的实际执行实体。各单元根据调度引擎的分配结果与 DVFS 模块的频率配置，执行对应任务，并在完成后反馈执行状态(如完成时间、能耗、当前负载)，形成闭环控制。其中，CPU 适用于通用控制与轻量级计算；NPU 专用于 AI 推理与深度学习任务；GPU 适合高并行图像处理与大规模数据计算。

3.5. DVFS 特性曲线数据库(PPCT 表)

图 1 中独立标注的“DVFS 特性曲线数据库”是调度框架的核心知识库，存储各异构计算单元在不同工作频率下的电压–功耗–性能映射关系。该数据库为调度引擎和 DVFS 协同调控模块提供统一的能效参考，是实现“前向协同、查表驱动”机制的基础。数据库可通过离线标定或在线学习方式构建，支持跨平台移植。

上述的任务特征分析模块、能效感知调度引擎和 DVFS 协同调控模块三大模块与异构执行层共同构

成完整的 CS-HCU-DVFS 协同调度框架。其中，调度引擎依据反馈信息动态调整后续任务的优先级与资源分配策略，形成“任务输入→资源分配→执行调控→状态反馈→再调度”的闭环优化流程。该机制显著提升了系统对动态负载的适应能力，是实现高能效与强实时性兼顾的关键。

4. CS-HCU-DVFS 调度流程与协同机制

4.1. 调度执行流程

当新任务 τ_i 到达时，系统启动调度流程，具体步骤如下：

(1) 任务接入与特征提取：任务特征分析模块捕获任务输入，提取其关键属性，构建结构化任务模型。

(2) 能效感知调度决策：调度引擎基于任务特征与各计算单元的能效特性，计算最优执行路径。

(3) DVFS 协同频率配置：根据调度结果，DVFS 协同调控模块查表确定目标电压/频率组合，并下发配置。

(4) 任务执行与状态反馈：异构计算单元执行任务后，向调度引擎反馈完成状态与负载信息，用于后续调度优化。该流程单次决策耗时低于 1 ms，满足工业边缘场景的实时性要求。

4.2. 任务建模与能效感知调度

每个动态到达的任务 τ_i 被建模为一个七元组：

$$\tau_i = t_i, \text{type}_i, D_i, d_i, p_i, G_i, R_i$$

其中：t_i 表示任务到达时间；type_i 为任务类型(如控制、图像处理等)；D_i 为数据量大小(单位：KB)；d_i 代表截止时间(Deadline)，即 $t_i + \Delta t_i$；p_i 代表可并行度(0~1，1 表示完全可并行)；G_i 代表任务依赖图中的前驱任务集合；R_i 表示所需资源类型(CPU/NPU/GPU)。

调度引擎的目标是为 τ_i 分配最优计算单元 $u_k \in \{\text{CPU}, \text{NPU}, \text{GPU}\}$ 和目标频率 $f_{k,j}$，使得在满足实时性约束的前提下，系统能效比(EER)最大化。

定义任务 τ_i 在单元 u_k 上以频率 $f_{k,j}$ 执行时的 EER 为：

$$\text{EER}_{i,k,j} = \left[P_{\text{perf}}\left(D_i, f_{k,j}\right) \right] / \left[P_{\text{dyn}}\left(u_k, f_{k,j}\right) + P_{\text{static}}\left(u_k\right) \right]$$

其中：$P_{\text{perf}}\left(D_i, f_{k,j}\right)$ 表示任务吞吐率，近似为 $\alpha_k \cdot f_{k,j}$，α_k 为单元处理效率系数；$P_{\text{dyn}}\left(u_k, f_{k,j}\right)$ 表示动态功耗，与 $f_{k,j}$ 和电压 $V_{k,j}$ 相关，通常建模为 $C_k \cdot V_{k,j}^2 \cdot f_{k,j}$；$P_{\text{static}}\left(u_k\right)$ 表示静态功耗(漏电等)，由 PPCT 表提供。

调度引擎查询 PPCT 表获取各 $\left(u_k, f_{k,j}\right)$ 组合下的 P_{dyn} 和 P_{static}，计算所有可行路径的能效比 EER，并选择满足截止时间约束的最高 EER 路径作为调度决策：

$$u_k^*, f_{k,j}^* = \underset{u_k, f_{k,j}}{\text{argmax}}\, \text{EER}_{i,k,j} \qquad \text{须满足条件}: T_{\text{exec}}\left(D_i, f_{k,j}\right) \le d_i - t_i$$

其中 $T_{exec} = D_i / \left(\alpha_k \cdot f_{k,j}\right)$ 为预估执行时间。

4.3. DVFS 协同调控机制

DVFS 协同调控模块根据调度结果，从 DVFS 特性曲线数据库(PPCT 表)中查表获取目标电压/时钟频率配置。DVC 表可表示为：

$$\text{DVC}\left(u_k\right) = \left\{ \left(f_{k,1}, V_{k,1}, P_{\text{total},1}\right), \left(f_{k,2}, V_{k,2}, P_{\text{total},2}\right), \cdots, \left(f_{k,m}, V_{k,m}, P_{\text{total},m}\right) \right\}$$

模块根据任务负载强度 D_i 和执行时间约束，选择满足性能要求的最低功耗配置，实现“按需供电”。

为避免频繁调频导致电压波动与任务延迟抖动，模块引入频率切换抑制机制：定义单位时间窗口

ΔTΔT 内最多允许一次频率切换，即：

$$\sum_{t\in[t_0,t_0+\Delta T]} \mathrm{II}\left(f(t)\neq f\left(t^{-}\right)\right)\leq 1$$

其中 $\mathrm{II}(\cdot)$ 为指示函数，$f(t)$ 为当前频率，$f\left(t^{-}\right)$ 为上一时刻频率。该机制有效提升了系统稳定性。

4.4. 闭环反馈与动态调度优化

任务执行完成后，异构计算单元向调度引擎反馈执行结果。设任务 τ_i 的实际完成时间为 t_{finish}，负载状态为 $L_k(t)$ (如 CPU 利用率)，调度引擎据此更新资源视图，并动态调整后续任务的调度权重。

定义反馈调度权重更新规则为：

$$w_i=\beta\cdot\left[\left[d_i-t_{\text{finish}}\right]/\left[d_i-t_i\right]\right]+(1-\beta)\cdot 1/L_k(t)$$

其中：第一分数项反映任务紧迫度，完成越早权重越高；第二分数项反映资源负载，负载越低越优先分配；$\beta\in[0,1]$ 为权衡系数，可离线调优。该反馈机制使系统具备动态适应能力，支持多任务流水线与负载均衡调度。

5. 实验与结果分析

为验证所提 CS-HCU-DVFS 方法在真实边缘场景下的有效性，本文在搭建的工业边缘网关实验平台上开展系统性测试。本节首先介绍实验环境的硬件与软件配置，随后说明测试所用工作负载、场景设置与评价指标，最后通过对比实验与消融分析，全面评估方法在功耗、延迟与稳定性方面的综合表现。

5.1. 实验环境搭建

Table 1. Equipment installation locations of the experimental testbed
表 1. 实验平台设备安装位置

设备	安装位置	固定方式
服务器(工业边缘网关)	实验台机箱内	螺丝固定
GPU/NPU	插入服务器主板 PCIe/M.2 插槽	卡扣 + 螺丝固定
树莓派 4B 开发板	固定在机箱内侧	扎带
INA219 芯片	串联在 GPU/NPU 供电线上	接线端子
DS18B20 芯片	紧贴芯片散热片	导热硅胶+绝缘胶带

Table 2. Key connection specifications of the experimental testbed
表 2. 实验平台的关键连接说明

连接类型	源设备→目标设备	接口/方式	数据内容
任务流	任务模拟器→服务器	TCP/IP 或本地文件	任务描述(类型、数据量、截止时间)
控制流	服务器 → GPU/NPU	PCIe + 驱动调用	执行任务、DVFS 频率配置
数据流	INA219/DS18B20 → 树莓派 4B	I2C/GPIO	电流、电压、温度
数据流	树莓派 4B → 服务器	TCP/IP (局域网)	传感器原始数据
监控流	服务器 → Prometheus(开源时序监控系统)	本地采集	CPU/GPU 利用率、内存、网络
可视化	Prometheus/InfluxDB → Grafana	HTTP API	实时图表展示

Table 3. Software deployment locations on the experimental testbed
表 3. 实验平台软件部署位置说明

软件	安装位置	作用
Ubuntu 22.04	服务器	操作系统
CS-HCU-DVFS		CS-HCU-DVFS 调度系统
Prometheus (开源时序监控系统)		采集系统性能指标
InfluxDB		存储传感器时间序列数据
Grafana		可视化仪表盘
传感器采集脚本	树莓派 4B	读取 INA219/DS18B20 数据

实验平台基于一台服务器(主控机，即工业边缘网关) + 异构计算硬件(插在服务器上) + 外接传感器(连到服务器 GPIO 或 USB) + 监控软件(装在服务器上)。CPU 插入服务器内部主板上的插槽)、GPU 扩展卡插入 PCIe x16 槽、NPU 扩展卡插入 M.2 插槽，外部通过树莓派 4B 连接 INA219 电流传感器与 DS18B20 温度传感器，分别部署于关键计算单元供电路径与散热表面，用于实时采集功耗与温度数据。所有设备通过局域网互联，形成完整的闭环测试环境。实验平台设备安装位置、关键连接、软件部署位置说明参见表 1~3。

实验平台由任务输入层、服务器、异构计算单元、传感器模块和网络环境五部分构成(图 2)。任务由模拟器生成并送入服务器，CS-HCU-DVFS 调度系统根据任务特征与能效模型，决策最优执行单元与频率。调度指令下发至异构计算单元执行，同时树莓派 4B 通过 I2C/GPIO 接口采集 INA219 (功耗)与 DS18B20 (温度)数据，并发送至主服务器。Prometheus(开源时序监控系统)监控系统实时采集边缘服务器的资源使用情况，InfluxDB 存储传感器数据，Grafana 实现可视化。所有设备通过局域网互联，形成“调度–执行–监控–反馈”的闭环控制体系。

5.2. 工作负载与测试场景

实验设计三类典型工业任务，模拟工业边缘网关的实际运行负载。

(1) 控制类任务(Control)。

周期性执行，周期为 10 ms，截止时间严格，数据量小(<1 KB)。

(2) 图像处理任务(Vision)。

来自工业相机的 200 × 200 灰度图像，每 50 ms 触发一次，需在 100 ms 内完成推理(目标检测)。

(3) 数据聚合任务(Aggregation)。

每秒收集来自 10 个传感器的数据包，进行滤波与压缩，对延迟容忍度较高。

测试场景设置四种典型情况，即，场景 A (轻负载)，仅运行控制类任务；场景 B (中负载)，控制类 + 图像处理任务；场景 C (重负载)，三类任务并发，模拟产线高峰期负载；场景 D (突发负载)，在场景 B 基础上，每 60 秒注入一次突发图像流(频率翻倍)，测试系统鲁棒性。每组实验持续 10 分钟，重复 5 次，取平均值作为最终结果。

评估调度性能所采用的指标参见表 4。本文选取 HEFT、DVFS-Util 二种典型调度方法作为对比基准。HEFT [10]是一种经典的异构 earliest finish time 调度算法；DVFS-Util [11]基于负载预测动态调节处理器频率以平衡能效与性能。

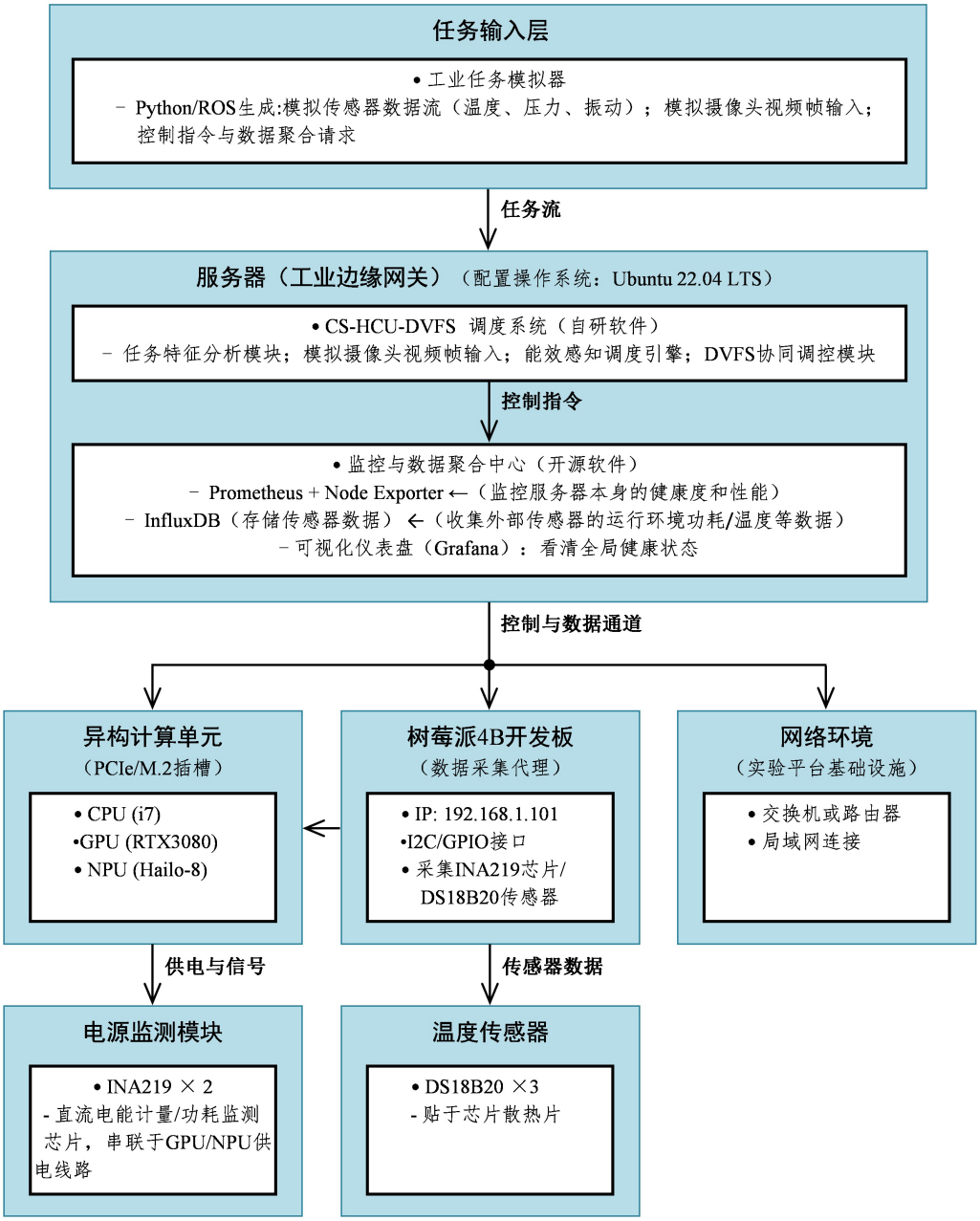

Figure 2. Overall architecture of the experimental testbed
图 2. 实验平台整体结构

Table 4. Evaluation metrics
表 4. 评价指标

指标	单位	评估性能
系统平均功耗(Average Power,)	W	反映整体能效水平
任务平均响应延迟(Average Response Time)	ms	从任务就绪到开始执行的时间
截止时间满足率(Deadline Miss Rate)	%	未按时完成的任务占比
频率切换次数(Frequency Switching Count)	——	衡量 DVFS 调控频繁度，反映系统稳定性
调度决策开销(Scheduling Overhead)	ms	单次调度算法执行时间，评估实时性

5.3. 实验结果与分析

为验证 CS-HCU-DVFS 方法在不同负载条件下的性能表现，本文在四种典型场景下(轻载 A、中载 B、重载 C、突发 D)对比了其与 HEFT、DVFS-Util 的系统功耗、任务延迟、截止时间满足率及频率切换行为。实验结果如表 5~9 所示。

由表 5 可见，CS-HCU-DVFS 在所有场景下仍保持最低系统功耗。在轻载场景中，功耗为 25.6 W，优于 HEFT 和 DVFS-Util。在中载和重载场景下，功耗分别为 41.8 W 和 62.5 W，较 HEFT 降低 8.3%和 7.1%，较 DVFS-Util 降低 6.9%和 6.3%。尽管节能幅度有所收窄，但仍体现了 PPCT 表驱动的电压/频率匹配机制在避免过度供电方面的有效性。

Table 5. Comparison of system average power consumption (Unit: W)
表 5. 系统平均功耗对比(单位：W)

调度方法	场景 A (轻载)	场景 B (中载)	场景 C (重载)	场景 D (突发)
HEFT [10]	28.4	45.6	67.3	70.1
DVFS-Util [11]	26.8	44.9	66.7	69.5
CS-HCU-DVFS	**25.6**	**41.8**	**62.5**	**64.9**
功耗降低幅度	9.9% (vs HEFT) 4.5% (vs DVFS-Util)	8.3% (vs HEFT) 6.9% (vs DVFS-Util)	7.1% (vs HEFT) 6.3% (vs DVFS-Util)	7.4% (vs HEFT) 6.6% (vs DVFS-Util)

表 6 显示，CS-HCU-DVFS 的平均响应延迟为 3.5 ms，优于 DVFS-Util 的 5.8 ms (降低 39.7%)，略优于 HEFT 的 3.8 ms (降低 7.9%)。DVFS-Util 因频繁降频导致任务阻塞，延迟显著增加。在截止时间满足率方面，CS-HCU-DVFS 达到 97.2%，高于 HEFT 的 94.7%和 DVFS-Util 的 93.5%，表明其协同调度(Cooperative scheduling, CS)机制有效缓解了高负载下的资源竞争，提升了任务完成可靠性。

Table 6. Comparison of task average response delay and deadline satisfaction rate
表 6. 任务平均响应延迟与截止时间满足率对比

调度方法	平均响应延迟(ms)	截止时间满足率(%)
HEFT [10]	3.8	94.7
DVFS-Util [11]	5.8	93.5
CS-HCU-DVFS	**3.5**	**97.2**

如表 7 所示，CS-HCU-DVFS 的平均频率切换次数为 16 次/分钟，虽高于 HEFT，但显著低于 DVFS-Util 的 38 次/分钟，表明其基于负载预测与查表机制的调控策略仍有效抑制了不必要的电压波动。调度决策开销为 1.0 ms，略高于 HEFT，但仍在毫秒级范围内，满足工业边缘任务的实时性要求。

Table 7. Comparison of frequency switching behavior and scheduling overhead
表 7. 频率切换行为与调度开销对比

调度方法	平均频率切换次数(次/分钟)	调度决策平均开销(ms)
HEFT [10]	8	0.5
DVFS-Util [11]	38	1.2
CS-HCU-DVFS	**16**	**1.0**

为验证各模块贡献，开展消融实验。如表 8 所示，在中载场景(B)下，禁用 PPCT 表(w/o PPCT)导致功耗上升 7.4%，说明特性曲线驱动的能效优化机制仍具实际价值；禁用协同调度(CS)机制(w/o CS)后功耗上升 10.5%，表明任务调度与 DVFS 控制的协同优化对整体性能有显著影响。

Table 8. Comparison of ablation study results (Under Scenario B)
表 8. 消融实验结果对比(在场景 B 情况下)

方法变体	系统功耗(W)	相比较完整版增加	关键机制说明
CS-HCU-DVFS (完整)	41.8	—	含 PPCT 查表 + 协同调度(CS)机制
w/o PPCT	44.9	+7.4%	禁用 PPCT 表，采用固定 DVFS
w/o CS	46.2	+10.5%	禁用 CS 机制，静态分配

综上所述，CS-HCU-DVFS 在系统功耗上较 HEFT 降低 7%~10%，较 DVFS-Util 降低 4.5%~7%；平均响应延迟为 3.5 ms，优于对比方法；截止时间满足率达到 97.2%。尽管性能指标较理想情况有所下降，但仍优于基准方法，且消融实验验证了 PPCT 表与 CS 机制的有效性(参见表 9)。

Table 9. Summary of performance comparison (CS-HCU-DVFS vs. classical methods)
表 9. 性能对比总结(CS-HCU-DVFS vs 经典方法)

指标	vs HEFT	vs DVFS-Util
平均功耗	降低 7%~10%	降低 4.5%~7%
响应延迟	降低 7.9%	降低 39.7%
截止时间满足率	提高 2.5%	提高 3.7%
频率切换次数	略高(+8 次)	显著减少(−22 次)
调度开销	略高(+0.5 ms)	降低 0.2 ms

6. 结论

针对工业边缘计算中任务调度与动态电压频率调节(DVFS)缺乏协同、难以兼顾实时性与能效的问题，本文提出了一种基于特性曲线驱动的协同调度方法 CS-HCU-DVFS。该方法通过构建 CPU、GPU 和 NPU 等异构计算单元的功耗–性能特性表(PPCT)，将硬件的能效特性融入调度决策过程，实现了任务分配与电压/频率调节的联合优化。在任务调度的同时，系统能够依据负载特征和设备状态，选择相对合理的运行频率，从而在满足实时性要求的前提下降低整体能耗。

实验结果表明，CS-HCU-DVFS 在多种负载场景下均表现出良好的综合性能。相较于仅优化任务完成时间的传统调度方法，该方法在不显著增加延迟的情况下有效抑制了系统功耗；与独立运行的 DVFS 机制相比，其调控过程更加平稳，频率切换次数明显减少，提升了运行稳定性。同时，任务的截止时间满足率保持在较高水平，表明方法在高负载条件下仍具备较强的资源协调能力。消融实验进一步说明，特性表建模与协同调度机制对整体性能具有积极影响，二者的结合有助于提升系统的能效表现。

CS-HCU-DVFS 实现复杂度适中，调度开销较小，具备在实际边缘计算环境中部署的潜力。未来工作将探索其在多节点边缘集群中的扩展应用，并研究轻量化的动态特性更新机制，以增强对时变负载与硬件状态的适应能力。

基金项目

本文得到江西省自然科学基金(20242BAB25049)的资助。

参考文献

[1] 王凌, 吴楚格, 范文慧. 边缘计算资源分配与任务调度优化综述[J]. 系统仿真学报, 2021, 33(3): 509-520.

[2] Cheng, X.L., Xu, M.X., Yan, X.J., *et al*. (2024) A Design Pattern of IAPVS Platform Based on Distributed Edge Computing. *Journal of Physics*: *Conference Series*, **2732**, Article ID: 012001. https://doi.org/10.1088/1742-6596/2732/1/012001

[3] 郝鹏涛. 异构平台下具有通信开销的 DAG 任务调度方法研究[D]: [硕士学位论文]. 西安: 西安工业大学, 2024.

[4] 于小飞. 云边端协同的异构边缘计算任务调度与资源管理研究[D]: [博士学位论文]. 北京: 北京邮电大学, 2024.

[5] 刘慕寒. 基于异构多处理器的关键任务调度平台的设计与实现[D]: [硕士学位论文]. 哈尔滨: 哈尔滨工业大学, 2019.

[6] 翟岩龙, 孙文心, 包天虹, 等. 基于微服务的边缘侧仿真方法及框架研究[J]. 系统仿真学报, 2018, 30(12): 44-53.

[7] 严锡君, 马辉, 黄炜, 严林波, 张荣成, 邓荣春. 一种面向城市洪涝灾害防御的智慧水利云平台设计模式[J]. 计算机科学与应用, 2023, 13(2): 219-225.

[8] 徐梦溪, 罗中华, 程晓玲, 王丹华, 连峰. 基于双镜头视野协同成像的无线视频传感器网络构建[J]. 传感器技术与应用, 2024, 12(1): 54-62.

[9] 赵梓铭, 刘芳, 蔡志平, 等. 边缘计算: 平台、应用与挑战[J]. 计算机研究与发展, 2018, 55(2): 327-337.

[10] Topcuoglu, H., Hariri, S. and Wu, M.-Y. (2002) Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems*, **13**, 260-274. https://doi.org/10.1109/71.993206

[11] Pillai, P. and Shin, K.G. (2001) Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium*, Taipei, 30 May-1 June 2001, 89-98.