HAMS: Hardware-Aware Model Scheduling on Heterogeneous Platforms

Haofeng Kou - Baidu Research Institute *
Yongtao Yao - Wayne State University *
Sidi Lu - Wayne State University
Yueqiang Cheng - Baidu Research Institute
Weijia Shang - Santa Clara University
Weisong Shi - Wayne State University

Problems

How to concurrently & efficiently deploy and execute the collaborative models on heterogeneous devices with different deployment constraints?

- The real-world applications usually require collaboration of multiple DNN models on edge computing platforms to finish complicated tasks with outstanding performance
- Explosive growth in model size, computational requirements, increasing number of involved models and devices

Previous Work

One-to-One: One DNN architecture to one hardware platform

- Design a network architecture that is both accurate and efficient on a given edge device
- Train a separate model for each device of interest and each latency budget of interest

- Too resource demanding for the case-by-case deployment environment
- Not practical enough when the real-world application requires the involvement of multi-models and diverse devices at the same time

Our Research - Innovation

Many-to-Many: providing actionable insights on scheduling the efficient deployment of a group of collaborative DNN models among heterogeneous hardware devices and assessment of our proposed partition and scheduling algorithm

- The multiple models scheduling problem for the edge computing tasks in the heterogeneous environment has **not been deeply studied** yet.
- Our proposed framework is the pioneer that points out the importance of this **new research direction** with useful insights for related research.

Our Research - Algorithm

Many-to-Many: providing actionable insights on scheduling the efficient deployment of a group of collaborative DNN models among heterogeneous hardware devices and assessment of our proposed partition and scheduling algorithm

- We have demonstrated the applicability of the proposed scheduling algorithms **MFS** and **HFS**, in three typical application scenarios of the computer vision field, with the ability of hardware adaptive self-learning to automatically schedule the deployment and execution of multiple models on heterogeneous edge services

Our Research - Result

Many-to-Many: providing actionable insights on scheduling the efficient deployment of a group of collaborative DNNs among heterogeneous hardware devices and assessment of our proposed partition and scheduling algorithm

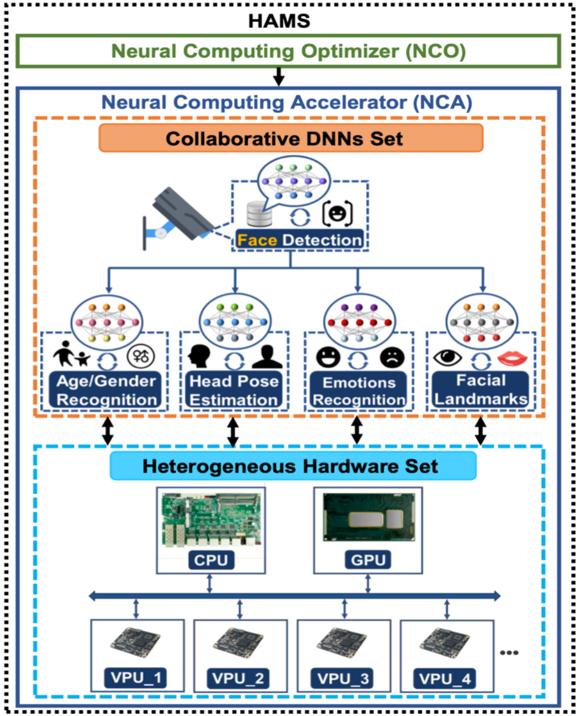
- Our analysis reveals that HAMS can balance computation resource utilization and reduce the inference time of the whole group of models up to **28.77%**.

NCO & NCA

HAMS contains two core components:

NCO - Neural Computing Optimizer responsible for training, optimizing, and transforming DNN models into a hardware-specific format so that the model can fit a given hardware platform well

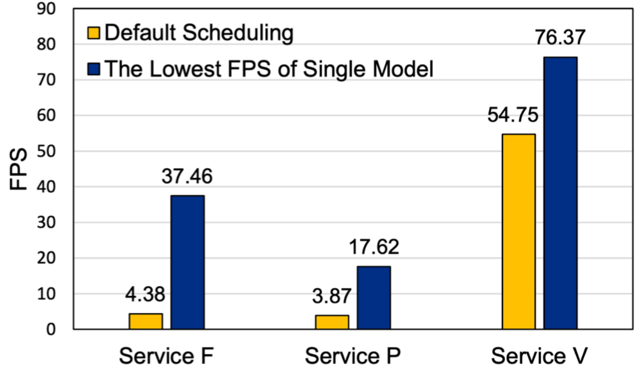
NCA - Neural Computing Accelerator integrate of HAMS that contains our proposed design



FPS Matrix

Matrix Generation:

- Calculate FPS of each model running independently on each device
- Overall inference speed dependent on where the slowest speed is



	Face Detection	Age/Gender Recognition	Emotion Recognition	Facial Landmarks	Head Estimation
CPU	8.69	241.51	212.06	265.37	241.46
GPU	37.46	370.82	514.46	151.77	389.72
VPU	12.93	399.99	345.88	185.42	317.52

(a)

	Person Detection	Person Attributes Recognition	Person Reidentification
CPU	3.30	132.35	116.68
GPU	17.62	198.82	113.30
VPU	3.73	168.40	111.76

(b)

	Vehicle License Plate Detection	Vehicle Attributes Recognition	License Plate Recognition
CPU	65.03	217.84	76.37
GPU	131.31	311.73	199.57
VPU	37.68	406.85	164.93

(c)

MFS

Target at finding an appropriate model for edge devices

- ModelAllocations
- QueryWorstCaseModel
- QueryModel

Algorithm 2 Hardware-first scheduling

```
input: array: FPS matrix, models: list, devices: list
function DeviceAllocations
    for i = 0 → models.length − 1 do
        dId ← QueryWorstCaseDevice
        value, mId ← QueryModel(dId)
        if i == models.length − 1 then
            dId ← QueryDevice(mId)
        models[mId].deviceName ←
                devices[dId].name
        models[mId].status ← true
        if devices[dId].name ==”VPU” then
            continue
        devices[dId].status ← true
    return models

function QueryWorstCaseDevice
    min ← array[0, 0]
    for i = 0 → array.cols − 1 do
        if models[i].status then
            continue
        for j = 0 → array.rows − 1 do
            if devices[j].status then
                continue
            if min > array[j, i] then
                dId ← j
                min ← array[j, i]
    return dId

function QueryModel(dId)
    max ← array[dId, 0]
    for i = 0 → array.rows − 1 do
        if devices[i].status then
            continue
        if max < array[dId, i] then
            mId ← i
            max ← array[dId, i]
    return max, mId
```

HFS

Aim to find a suitable edge device for specific models

- DeviceAllocations
- QueryWorstCaseDevice
- QueryDevice

Algorithm 1 Model-first scheduling

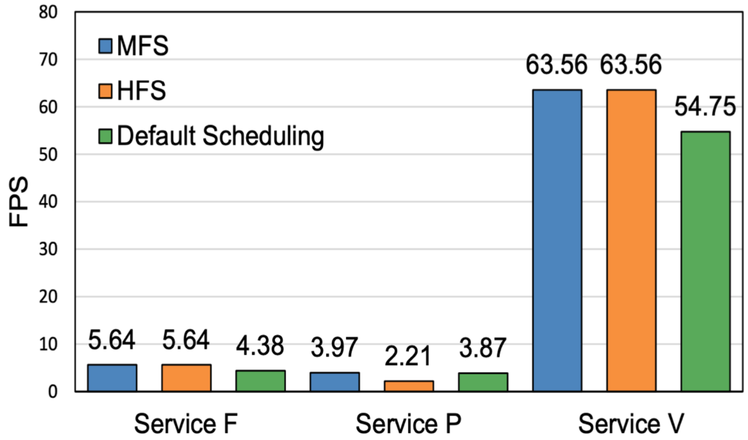
input: *array*: FPS matrix, *models*: list, *devices*: list

```
function ModelAllocations
    for i = 0 → models.length − 1 do
        mId ← QueryWorstCaseModel
        value, dId ← QueryDevice(mId)
        models[mId].deviceName ←
                devices[dId].name
        models[mId].status ← true
        if devices[dId].name ==”VPU” then
            continue
        devices[dId].status ← true
    return models

function QueryWorstCaseModel
    min ← array[0, 0]
    for i = 0 → array.cols − 1 do
        if models[i].status then
            continue
        for j = 0 → array.rows − 1 do
            if devices[j].status then
                continue
            if min > array[j, i] then
                mId ← i
                min ← array[j, i]
    return mId

function QueryDevice(mId)
    max ← array[0, mId]
    for i = 0 → array.rows − 1 do
        if models[i].status then
            continue
        if max < array[i, mId] then
            dId ← i
            max ← array[i, mId]
    return max, dId
```

Single Service

The individual service models assigned to their most suitable edge devices
Overall FPS for each service will be calculated saperately

	DNN Model	MFS	HFS
Service F	Face Detection	GPU	GPU
	Age/Gender Recognition	VPU	VPU
	Emotions Recognition	VPU	VPU
	Facial Landmarks	CPU	CPU
	Head Pose Estimation	VPU	VPU
Service P	Person Detection	GPU	GPU
	Person Attributes Recognition	VPU	CPU
	Person Reidentification	CPU	VPU
Service V	Vehicle License Plate Detection	GPU	VPU
	Vehicle Attributes Recognition	VPU	CPU
	License Plate Recognition	CPU	GPU

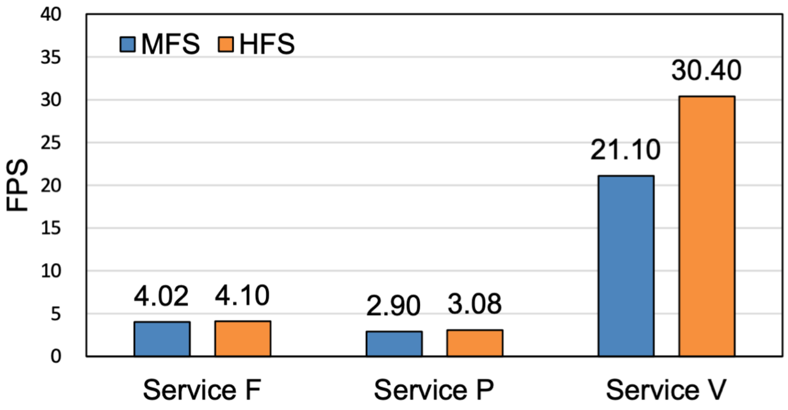


- Service F: MFS & HFS leads to the same FPS(5.64), 28.77% higher than default FPS (4.38)
- Service P and Service V: HAMS improve FPS by 2.58%

Multiple Service

Three sets of 11 models assigned to their most suitable edge devices
VPUs can be expanded - one model to one edge device
Overall FPS for all services & models are calculated together

ModelName	MFS	HFS
Face Detection	VPU	VPU
Age/Gender Recognition	VPU	VPU
Emotions Recognition	VPU	VPU
Facial Landmarks	VPU	CPU
Head Pose Estimation	VPU	VPU
Person Detection	GPU	GPU
Person Attributes Recognition	VPU	VPU
Person Reidentification	VPU	VPU
Vehicle License Plate Detection Detection	CPU	VPU
Vehicle Attributes Recognition	VPU	VPU
License Plate Recognition	VPU	VPU



- Service F/P/V shows better FPS than default FPS scheduling

Open Discussion

- Task-Level Scheduling on Heterogeneous Platforms
  - StarPU on HPC
  - ESTS on HCS
  - OmpSs
  - AlEbrahim

- *Neural Architecture Search*
  - MnasNet
  - DARTS - Differentiable ARchiTecture Search
  - FBNets - Facebook-Berkeley-Nets
  - Once-for-All

- *Gap between Previous Work*
  - Compared with Task-Level Scheduling
  - Compared with Neural Architecture Search

Summary

- Prove the importance of model scheduling for multiple DNNs and heterogeneous edge devices with diverse computation resources

- Key concept is *Worst-Case-First* for hardware-aware models scheduling

- Introduce and discuss two scheduling algorithms and get the evaluation results of three DNN groups on CPU, GPU and multiple VPUs

- The evaluation results demonstrate the effectiveness of HAMS on accelerating the co-inference of multi-models on the heterogeneous edge devices by up to 28.77%

Acknowledge & QA

- Thanks for the collaboration from WSU, SCU and BRI !
- Thanks SEC20 offering the chance !

- We can be reached at: BRI & WSU & SCU
    - kouhaofeng@baidu.com
    - yongtaoyao@wayne.edu