Niagara: Scheduling DNN Inference Services on Heterogeneous Edge Processors

Daliang Xu, Qing Li, Mengwei Xu, Kang Huang, Gang Huang, Shangguang Wang, Xin Jin, Yun Ma, Xuanzhe Liu

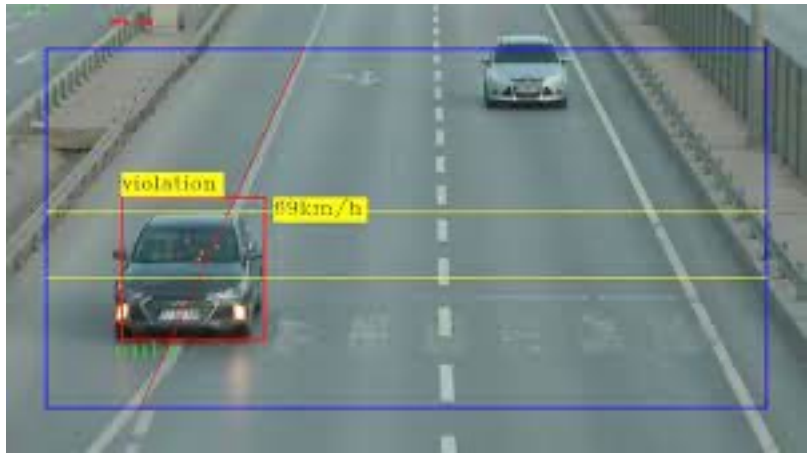
北京大学 PEKING UNIVERSITY

北京郵電大學 Beijing University of Posts and Telecommunications

Linggui Tech Company

ICSOC 2023

Intelligent edge applications



Vehicle detection

Immersive online shopping

AR emoji

AR & VR devices

Remote healthcare devices

Smart Home Devices

Problem

Helmet/gloves wearing detection

DNN services

- Person Detection
- Pose estimation
- Gloves Detection
- Helmet Detection

- low response latency
- protect privacy
- **Low throughput problem**

Opportunities

- An edge device contains multiple **heterogeneous** processors

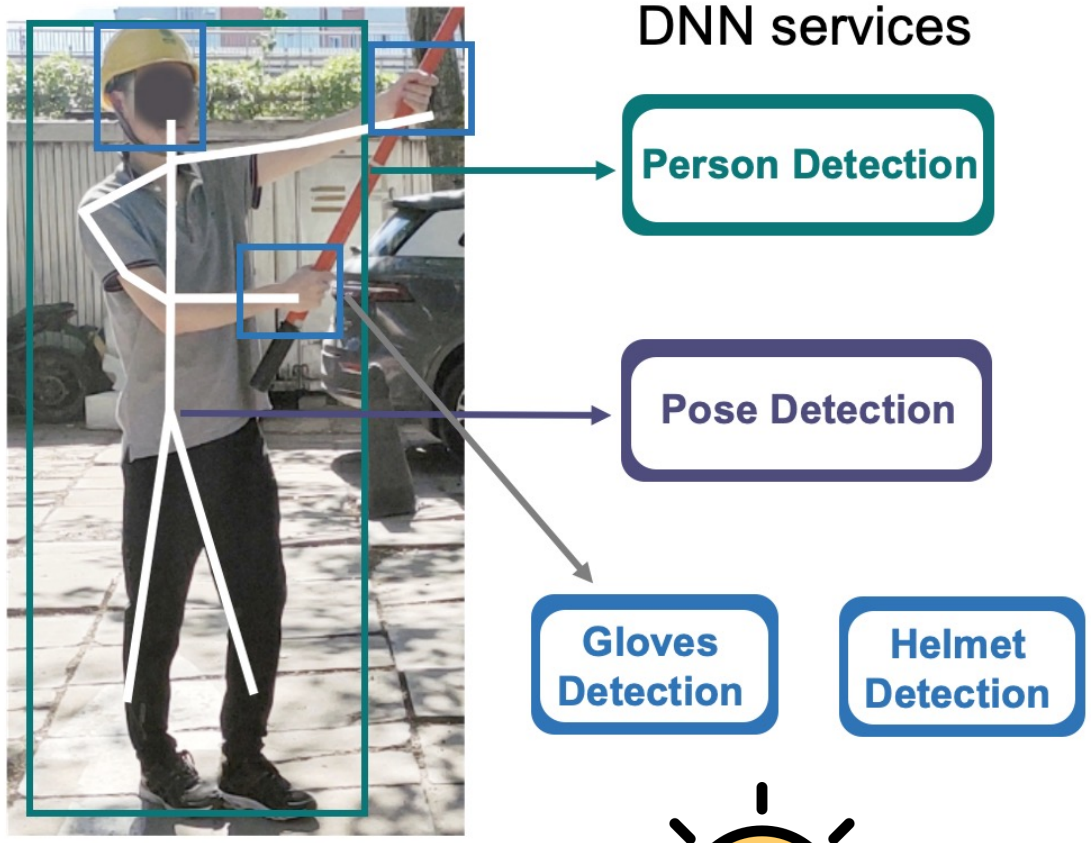
IP Cameras

Type	Processors
CPU	1- 3.0 GHz X2 + 3- 2.5 GHz Cortex A710 + 4- 1.8GHz Cortex-A510
GPU	Adreno 650
Accelerator	Hexagon 698 DSP

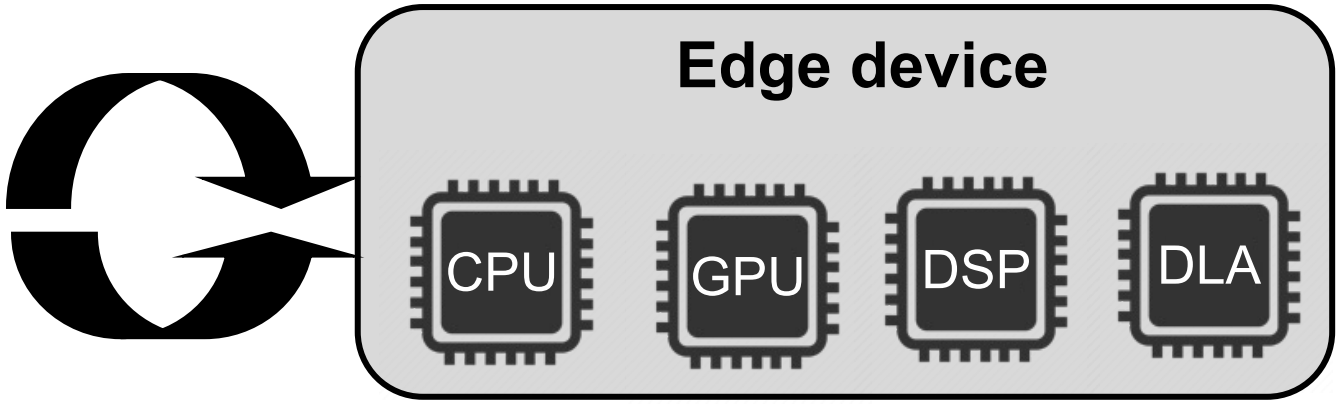
IoT devices

Type	Processors
CPU	12-core Arm Cortex-A78AE
GPU	Ampere 1024 CUDA core + 32 Tensor Cores
Accelerator	Deep learning accelerator (DLA)

Motivation



Efficient heterogeneous processors



Can we offload the DNN services to the heterogeneous processors?

Challenge#1: High complexity in scheduling design

- **DNN-inference-service-to-processor affinity and hardware support**

DNN Service	DNN Model	Latency			Utilization		
		CPU	GPU	DSP	CPU	GPU	DSP
Person detection	SSD-quant	112.1 ms	**79.9 ms**	103.1 ms	361%	56%	77%
Pose estimation	CenterNet	**22.9 ms**	31.7 ms	-	287%	30%	-
Helmet detection	SSD-Helmet-quant	25.6 ms	8.4 ms	**5.9 ms**	195%	58%	85%
Gloves detection	Pole-gloves	6.7 ms	**3.2 ms**	-	198%	34%	-
Text recognition	OCR-recognition	**30.8 ms**	38.1 ms	-	295%	35%	-

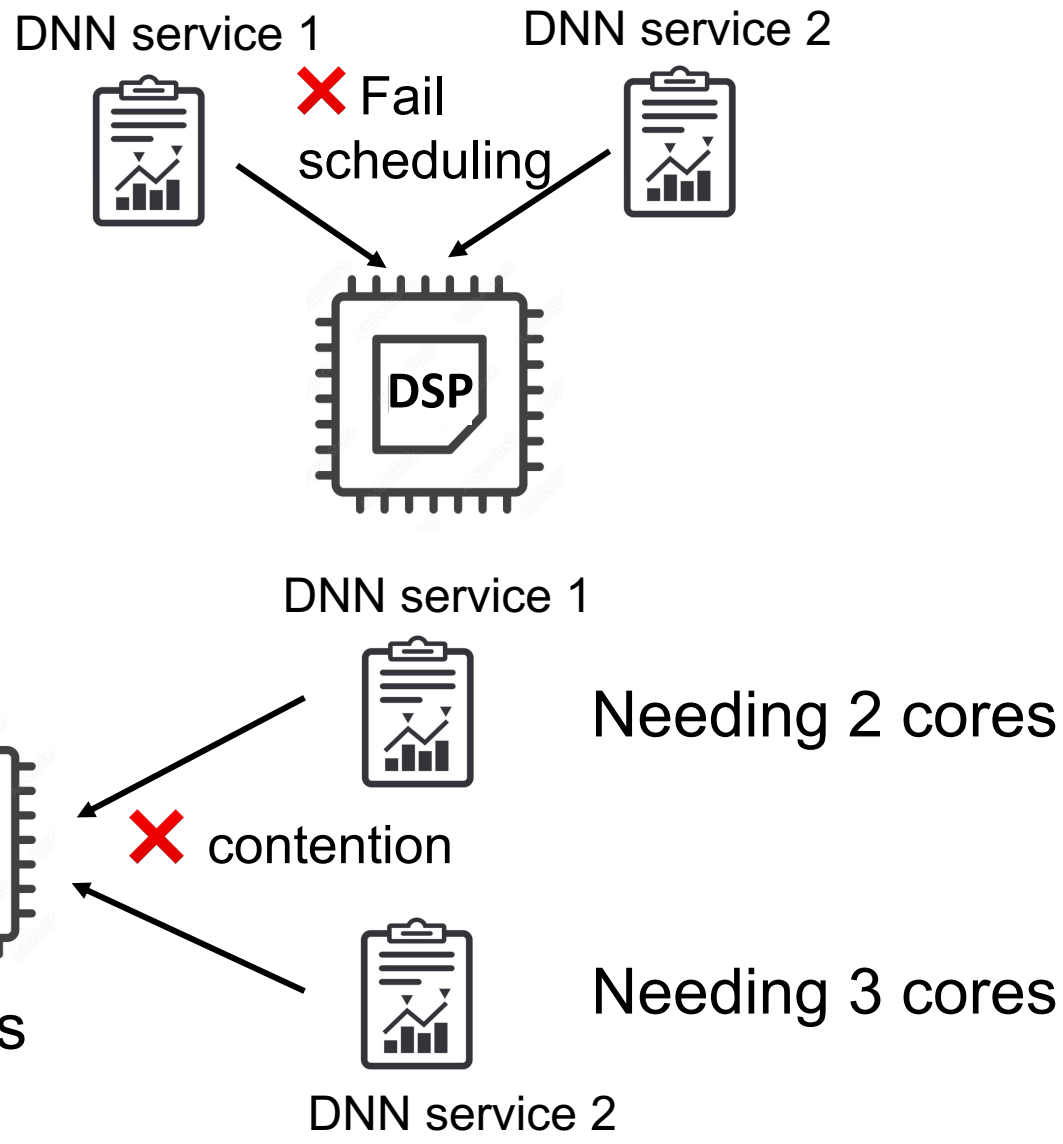
⚠ **No one-size-fits-all processor as cloud GPU.**

Challenge#1: High complexity in scheduling design

- **Parallel execution**

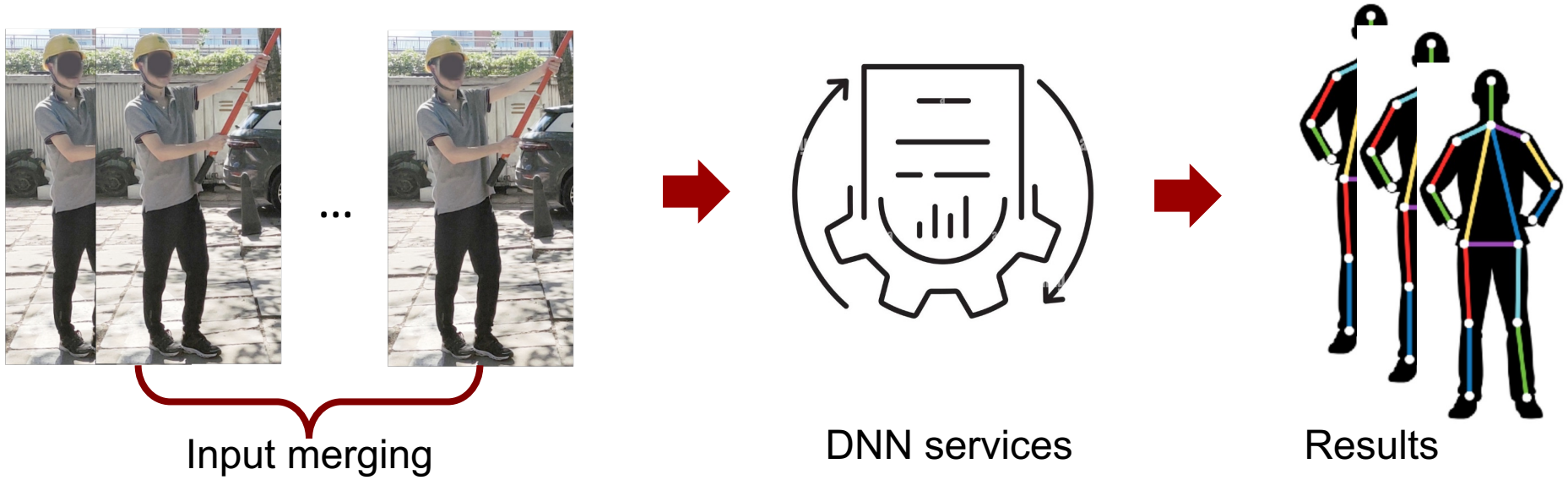
⚠ **Not all** processors support parallel execution

⚠ Optimal performance: **Not exceed** hardware capacity



Challenge#1: High complexity in scheduling design

- **Batch execution: higher throughput with longer latency**



Batch size	Latency (ms)	Throughput (DNN/s)
1	12	83.3
2	15	133.3
4	20	200.0
8	30	266.7

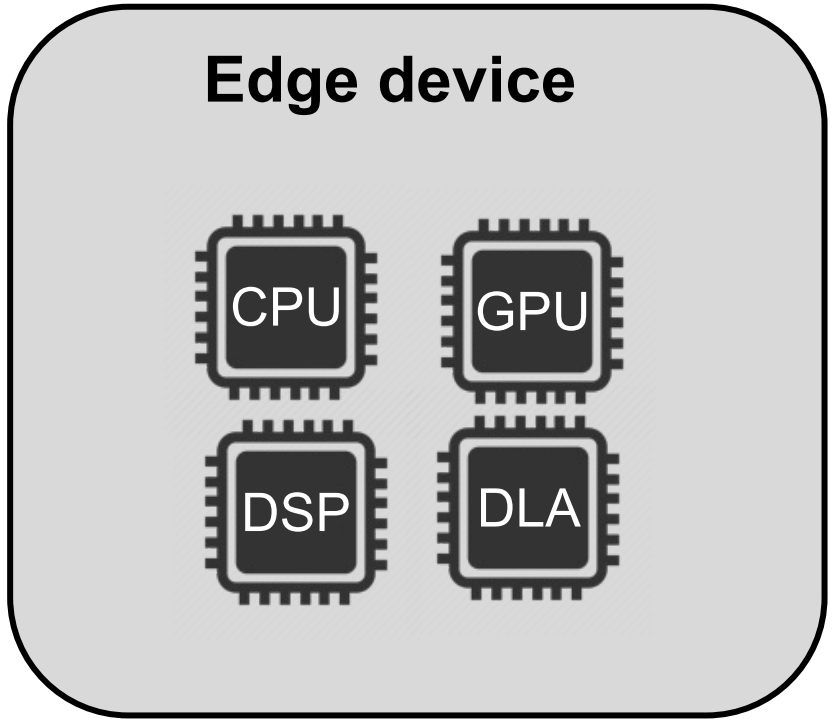
8x batch size → **2.5x latency delay** → **3.2x throughput enhancement**

Challenge#1: High complexity in scheduling design

- **Summary and implications**

Jointly considering

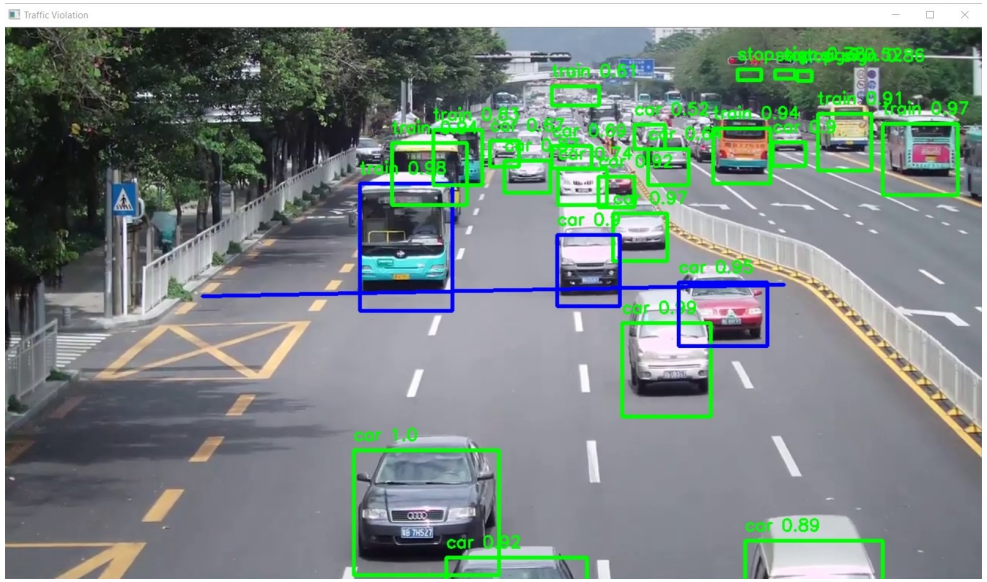
- DNN service-to-processor affinity
- Hardware support
- Parallel execution
- Batch execution

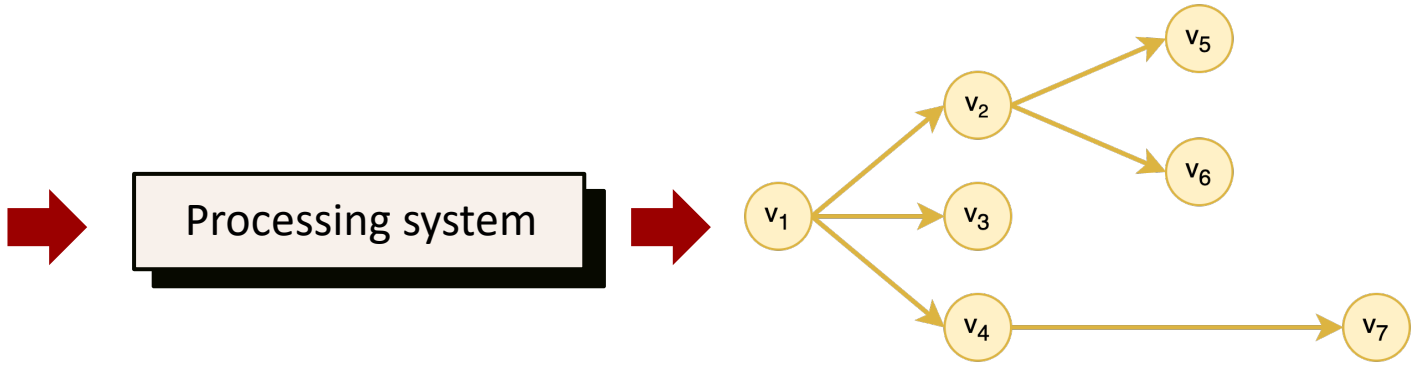


- **Existing work**
  - Online heuristic algorithm: Low performance ⚠
  - Offline algorithm: No dynamic support ⚠

Challenge#2: Unknown and mutative service workload

- **Unknown service workload limits scheduling space**





- **Existing work**
  - Lightweight methods: only one request service graph ⚠
  - Complex methods: high predicting overhead ⚠

Summary of challenges

DNN services

Person Detection

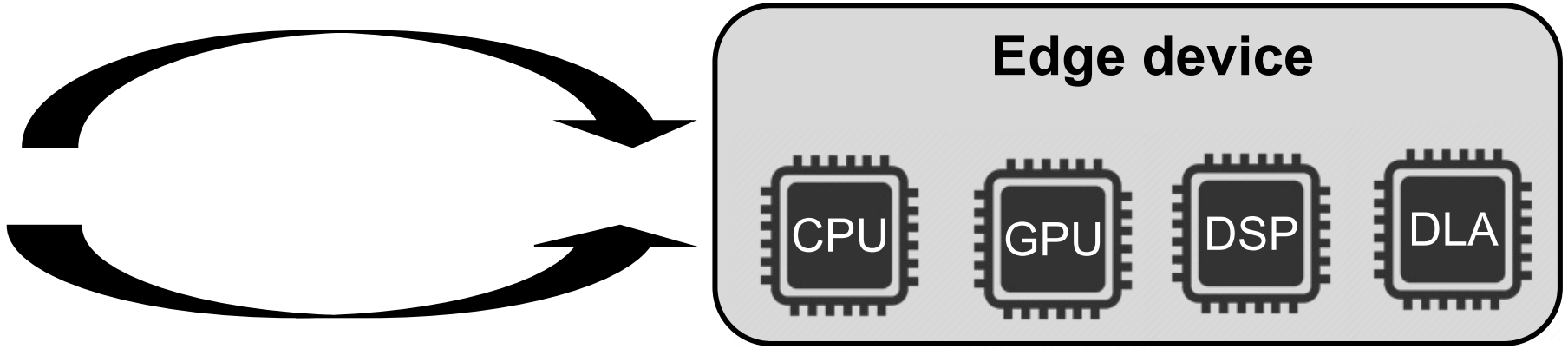
Pose Detection

Gloves Detection

Helmet Detection

#1: High complexity in scheduling design

Efficient heterogeneous processors



#2: Unknown and mutative service requests

Niagara

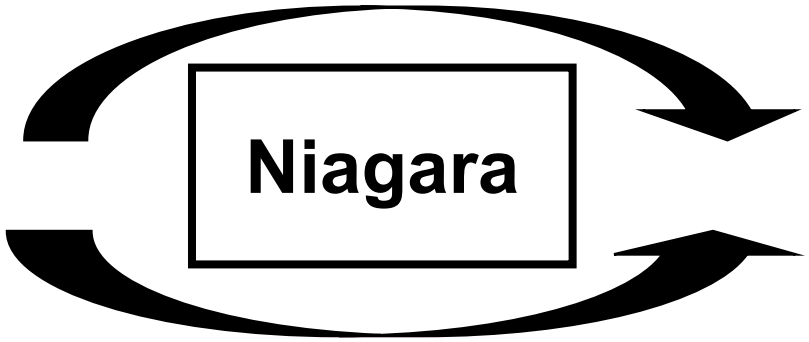
DNN services

Person Detection

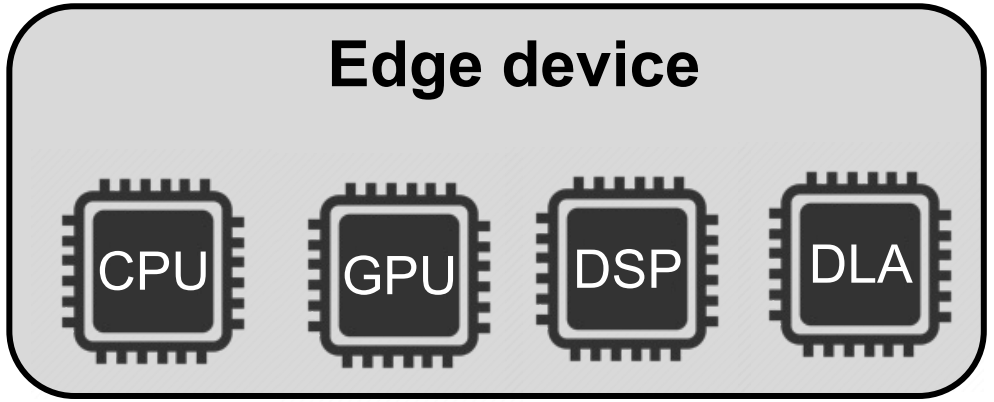
Pose Detection

Gloves Detection

Helmet Detection



Efficient heterogeneous processors



#1: offline optimization + online scheduling

#2: decouple the prediction of the service graph and coming requests

Niagara

DNN services

Person Detection

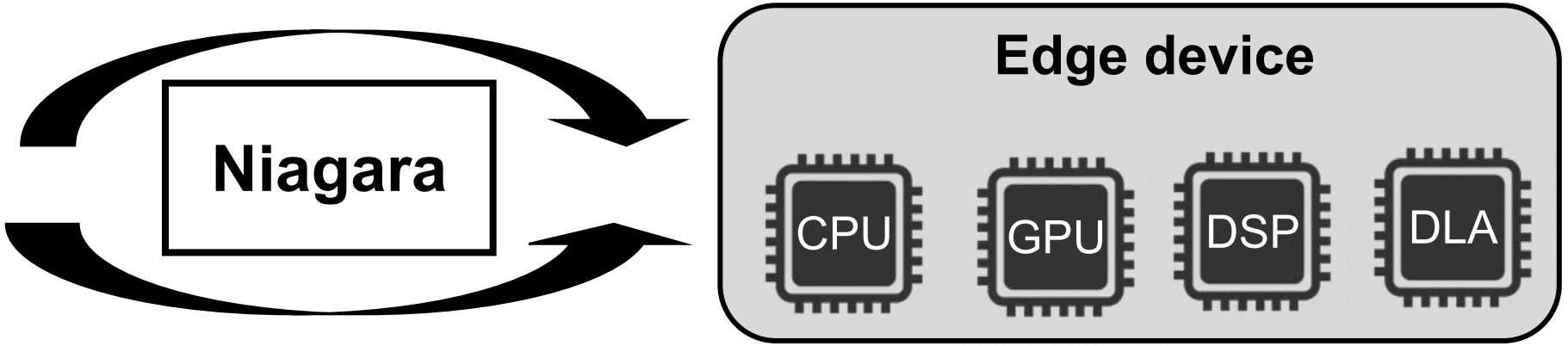
Pose Detection

Gloves Detection

Helmet Detection

#1: offline optimization + online scheduling

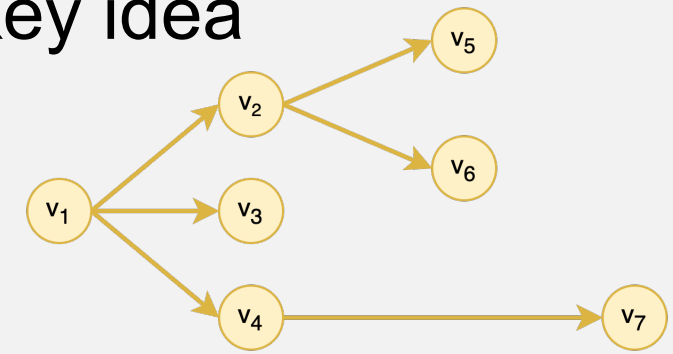
Efficient heterogeneous processors



#2: decouple the prediction of the service graph and coming requests

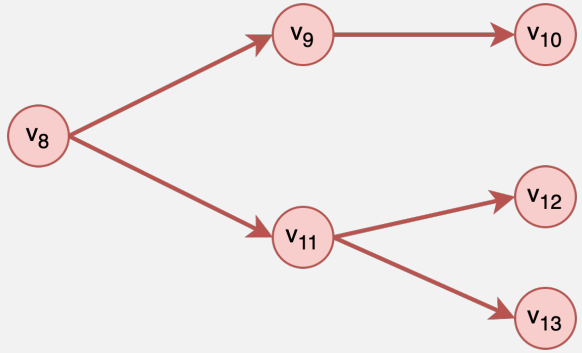
Offline optimization and online scheduling

Key idea



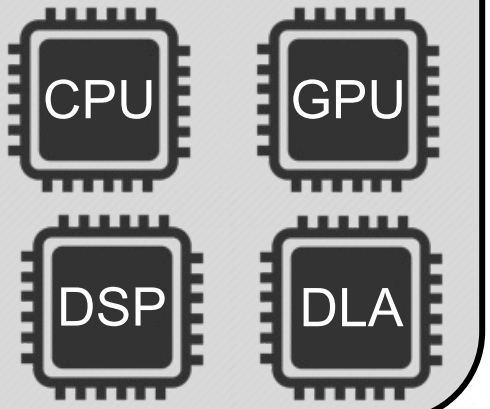
Online service graphs

Abstract



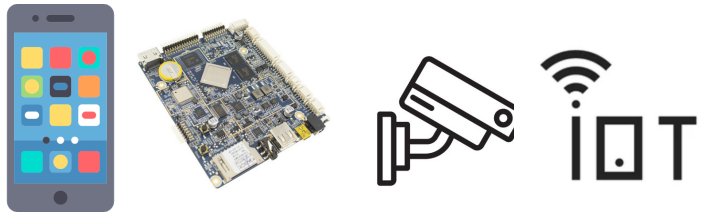
Offline service graph templates

Heterogenous processors



Template-based strategy matcher

Scheduling Problem



Deploy

Scheduling strategies

Offline optimization

- **Offline scheduling problem**
  - Decision variables

Variable	Description
Placement	Service-to-processor choice
Batch	Control batch execution
Parallel	Control parallel execution

  - Objective*: maximum throughput*
    - *max C s.t. Eq. (1)-(7)*

$$C = 1/\max\{t_i + \sum_{j=1}^{M} x_{i,j} T_{i,j}, \forall i \in N, \forall j \in M\}$$

$$\sum_{j \in \mathcal{R}_i} x_{i,j} = 1, \forall i \in N \quad (1)$$

$$t_i + \sum_{j=1}^{M} x_{i,j} T_{i,j} \leq t_k \quad (2)$$

$$\frac{t_k - t_i}{x_{i,j} x_{k,j} (1 - PL_{i,k,j})(1 - B_{i,k,j})} \geq T_{k,j} \quad (3)$$

$$x_{i,j} * x_{k,j} * PL_{i,k,j} * abs(t_k - t_i) \leq min(T_k, T_i) \quad (4)$$

$$\sum_{k=1}^{N} PL_{i,k,j} U_{k,j} + U_{i,j} \leq E_j \quad (5)$$

$$x_{i,j} * x_{k,j} * B_{i,k,j} * (t_k - t_i) \leq 0 \quad (6)$$

$$\forall v_i, v_k \in \mathcal{RQ}, t_k - t_i \leq Lat_{max}^{RQ} \quad (7)$$

Offline optimization

- **Offline scheduling problem**
  - Decision variables

Variable	Description

GUROBI solver: Less than 10% optimality loss

- *max C s.t. Eq. (1)-(7)*

$$\sum_{j \in \mathcal{R}_i} x_{i,j} = 1, \forall i \in N \quad (1)$$

$$t_i + \sum_{j=1}^{M} x_{i,j} T_{i,j} \leq t_i \quad (2)$$

$$\sum_{k=1} PL_{i,k,j} U_{k,j} + U_{i,j} \leq E_j \quad (5)$$

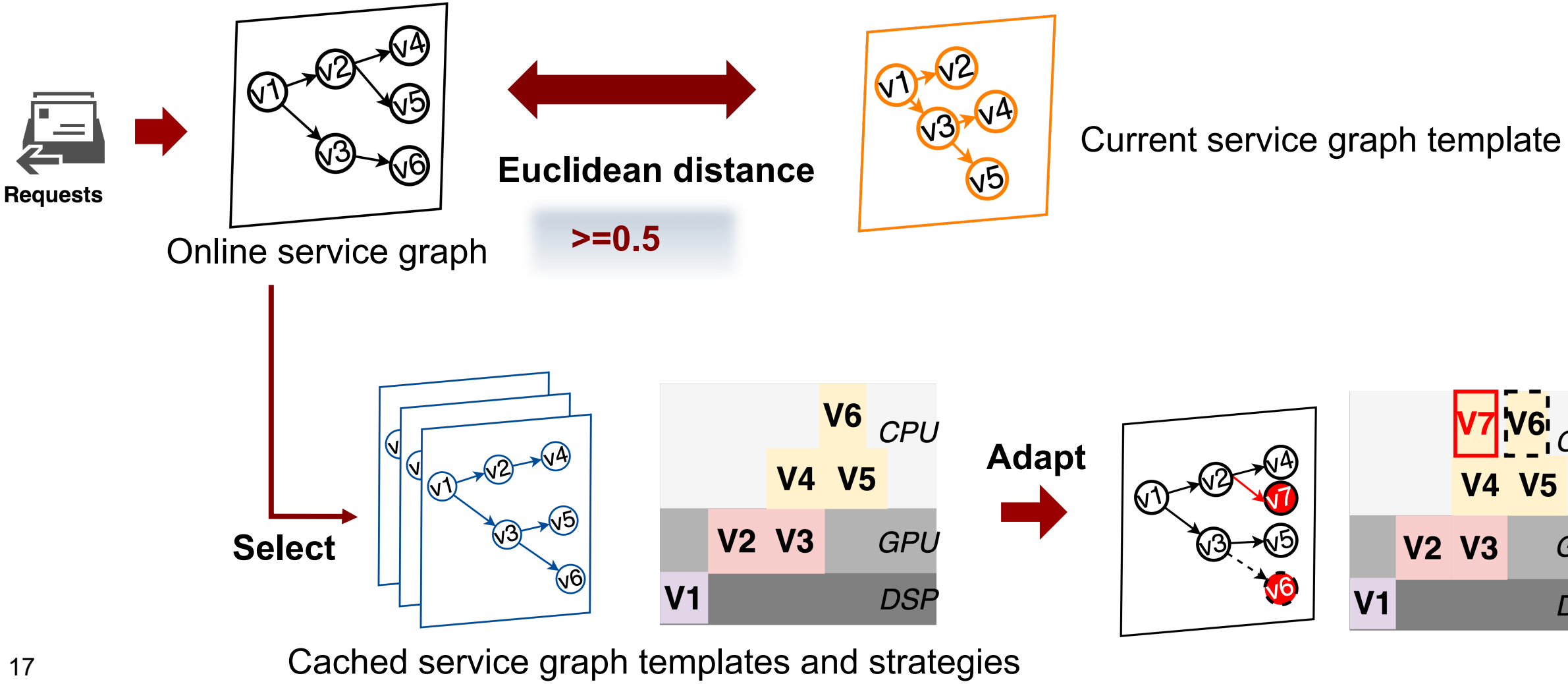
$$x_{i,j} * x_{k,j} * B_{i,k,j} * (t_k - t_i) \leq 0 \quad (6)$$

$$\forall v_i, v_k \in \mathcal{RQ}, t_k - t_i \leq Lat_{max}^{RQ} \quad (7)$$

$$C = 1/\max\{t_i + \sum_{j=1}^{M} x_{i,j} T_{i,j}, \forall i \in N, \forall j \in M\}$$

Online scheduling

- **Template-based strategy matcher**



Cached service graph templates and strategies

Niagara

DNN services

Person Detection

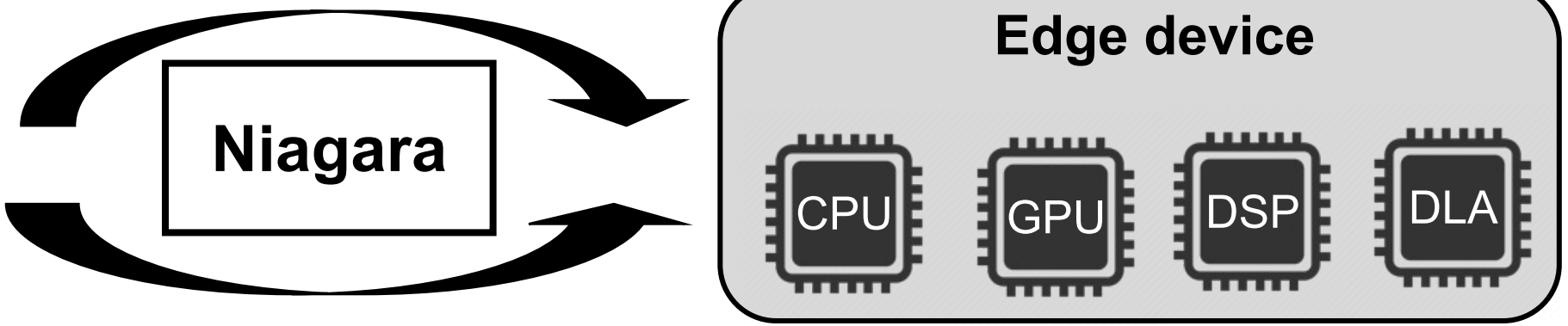
Pose Detection

Gloves Detection

Helmet Detection

#1: offline optimization + online scheduling

Efficient heterogeneous processors

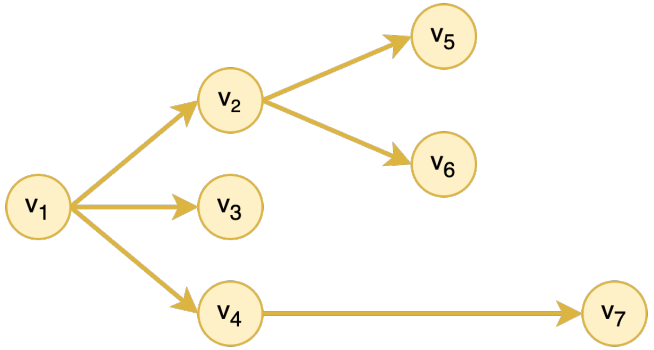


#2: decouple the prediction of the service graph and coming requests

Dynamic input predictor

- **The service graph tends to be more stable than the content.**

Different directions



Same service graph

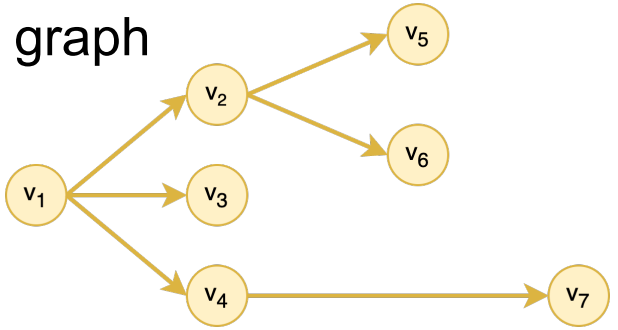
Different backgrounds

Dynamic input predictor

- **Decouple the prediction of the service graph and coming requests**

Predictor

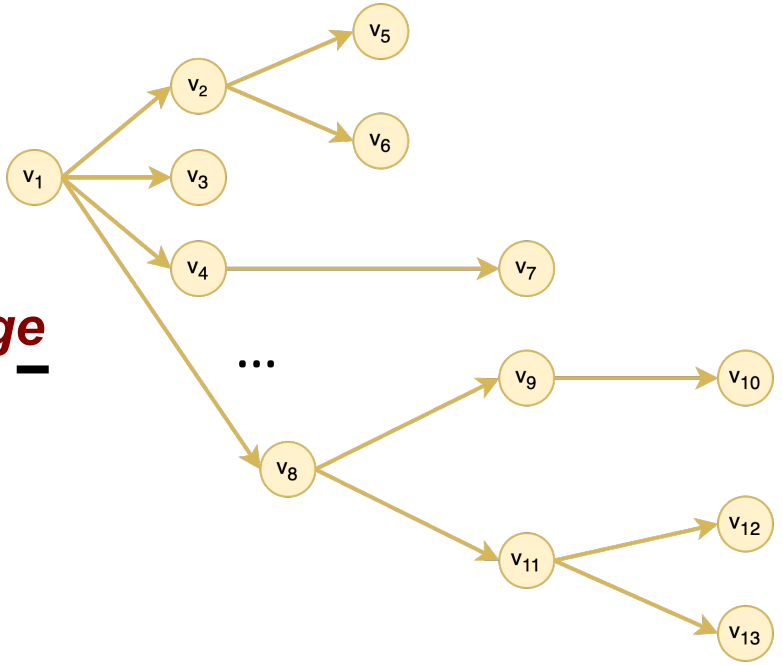
Service graph



Decouple

Merge

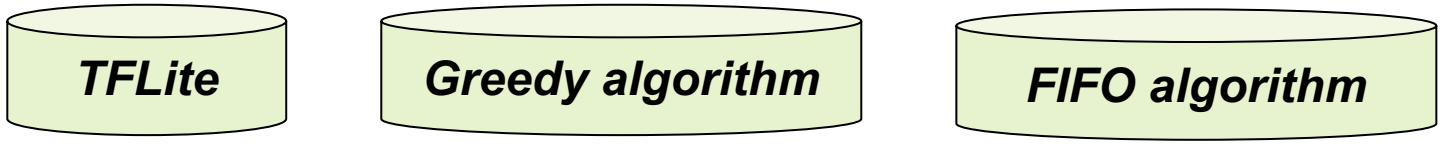
Number of the requests having the same service graph



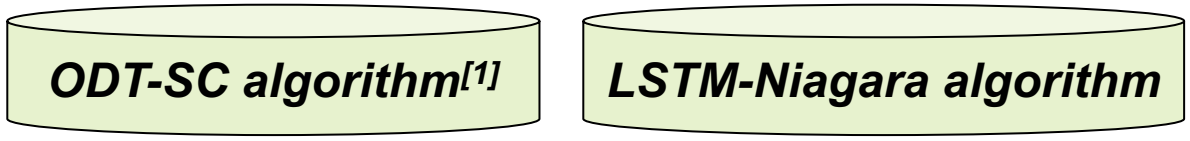
Multiple requests' service graph

Evaluation: methodology

- **Settings**
  - 8 service combinations + 11 DNN services + 3 real-world video stream
  - 3 commodity devices
  - Real-deployment
- **Baselines**
  - 3 industry baselines



  - 2 SOTA research baselines

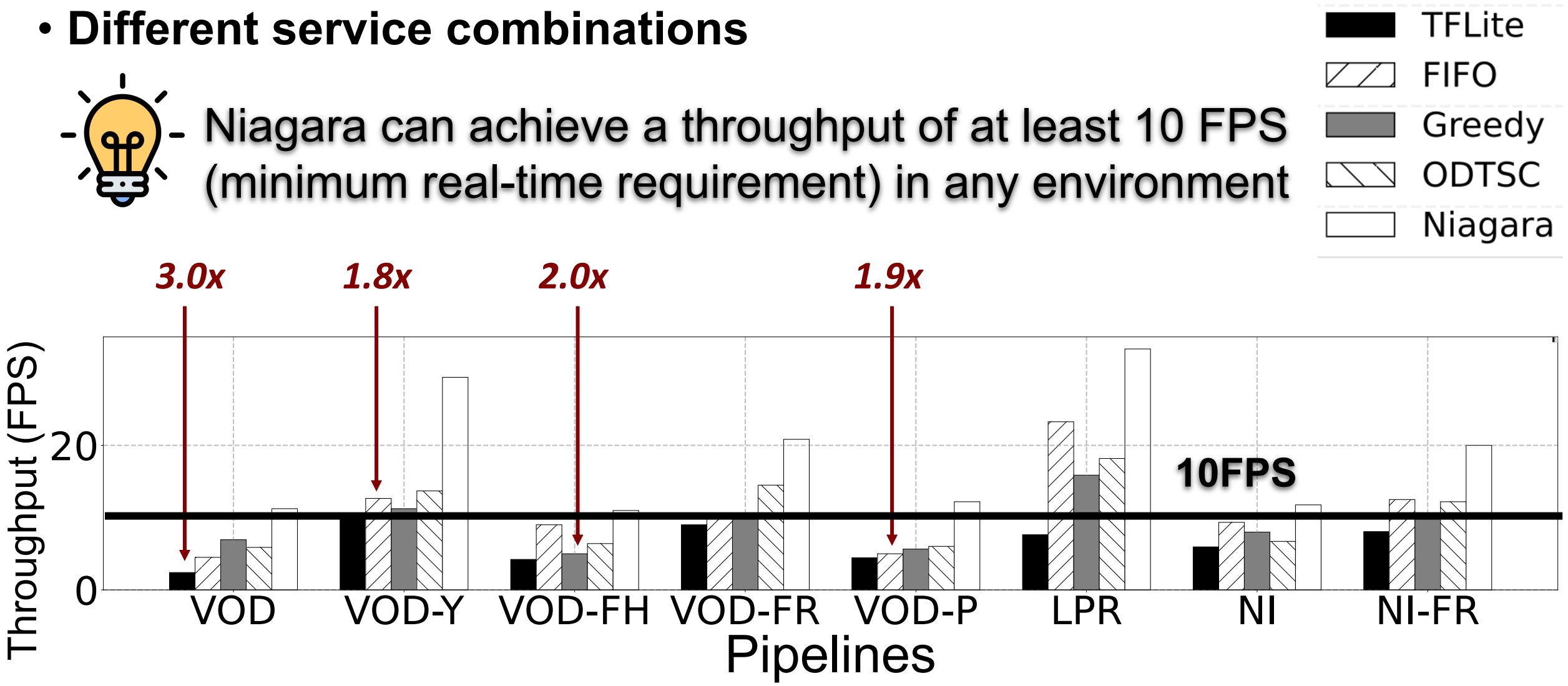


[1] Zhao G, Xu H, Zhao Y, et al. Offloading tasks with dependency and service caching in mobile edge computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(11): 2777-2792.

Evaluation: throughput

- **Different service combinations**

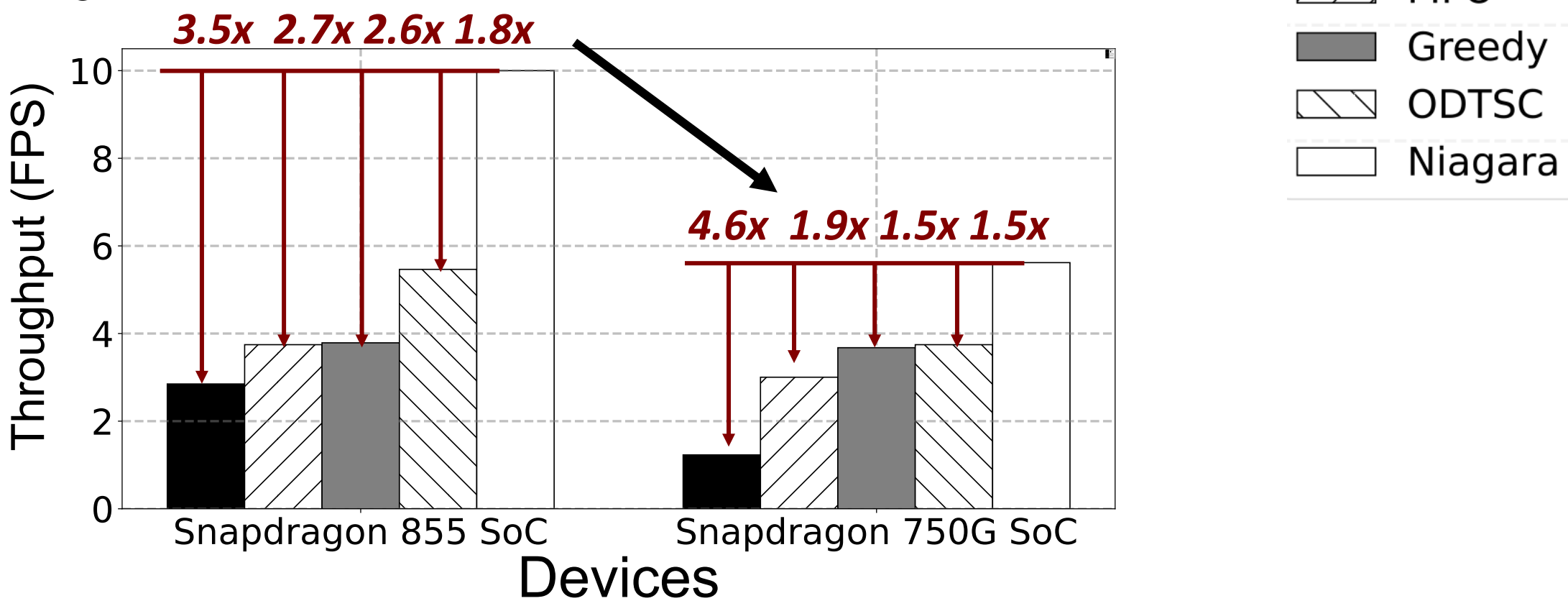
Niagara can achieve a throughput of at least 10 FPS (minimum real-time requirement) in any environment



Evaluation: throughput

- **Different devices**

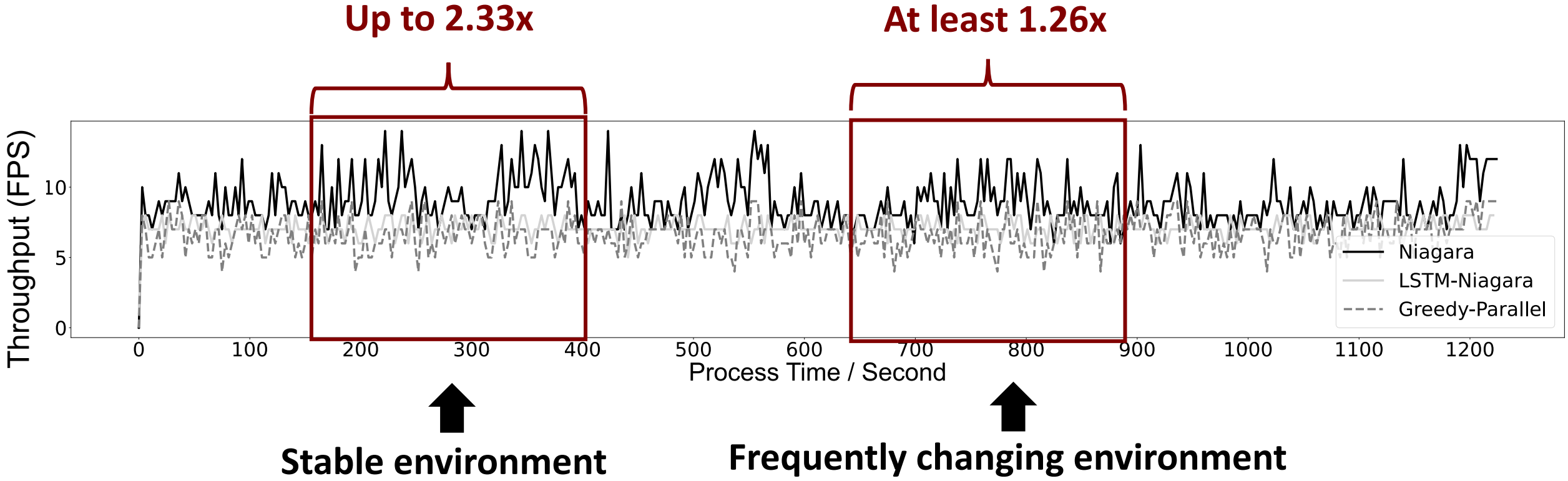
More powerful processors, more benefits



Evaluation: throughput

- **Real deployment**

Efficiently and stably improving throughput in real-world environments



Real deployment

Real execution

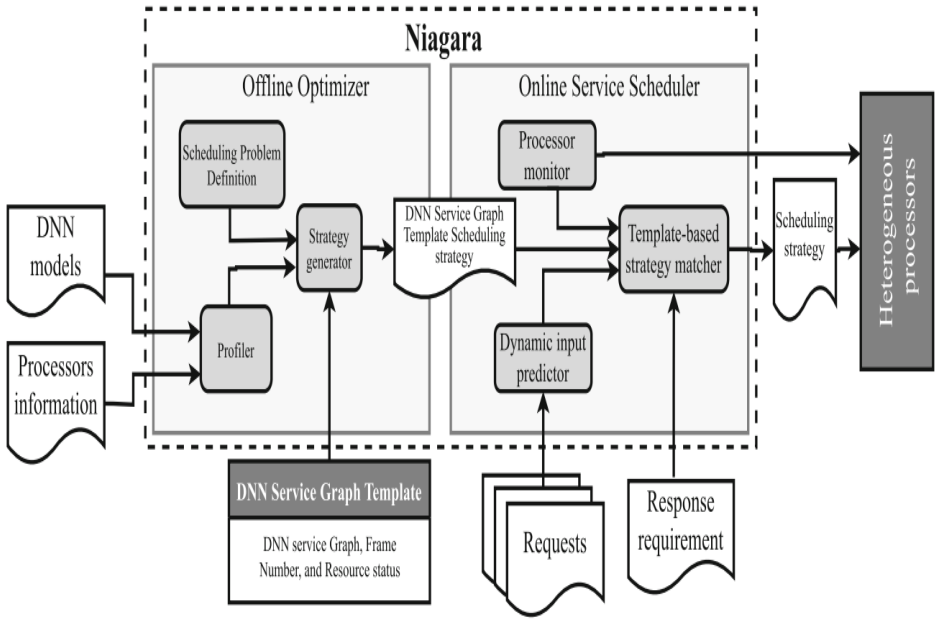
Custom-made
IP Cameras

State Grid
corporation of China

- **6-month** pilot run with **18,000** maintenance jobs
  - zero accidents
  - Save traditional human-based supervision

Thousands of substation in the future

Take-away



Niagara

- First scheduling engine for DNN services on edge heterogeneous processors
- Offline optimization + online scheduling
- Real deployment and saving traditional human-based supervision

xudaliang@pku.edu.cn

website https://daliangxu.github.io/

More about our affordable AI systems: http://www.liuxuanzhe.com