IEEE INFOCOM
IEEE International Conference on Computer Communications
20–23 May 2024 // Vancouver, Canada

IEEE ComSoc IEEE Communications Society

IEEE

Galaxy: A Resource-Efficient Collaborative Edge AI System for In-situ Transformer Inference

Shengyuan Ye[1], Jiangsu Du[1], Liekang Zeng[1,2], Wenzhong Ou[1], Xiaowen Chu[2], Yutong Lu[1], Xu Chen[1]

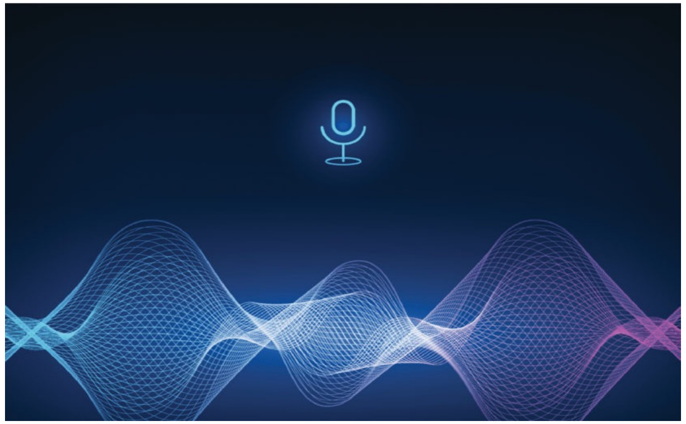
[1]Sun Yat-sen University, [2]HKUST(GZ)

中山大學 SUN YAT-SEN UNIVERSITY

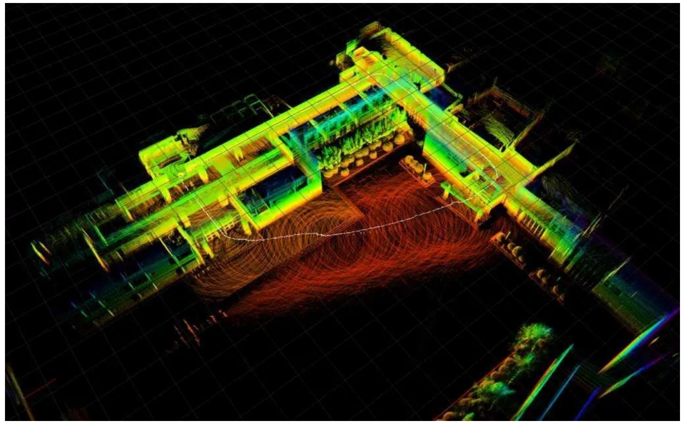
香港科技大学(广州) THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Intelligent edge applications

- **Transformer-based** models driven increasing intelligent applications.



Personal AI Assistants

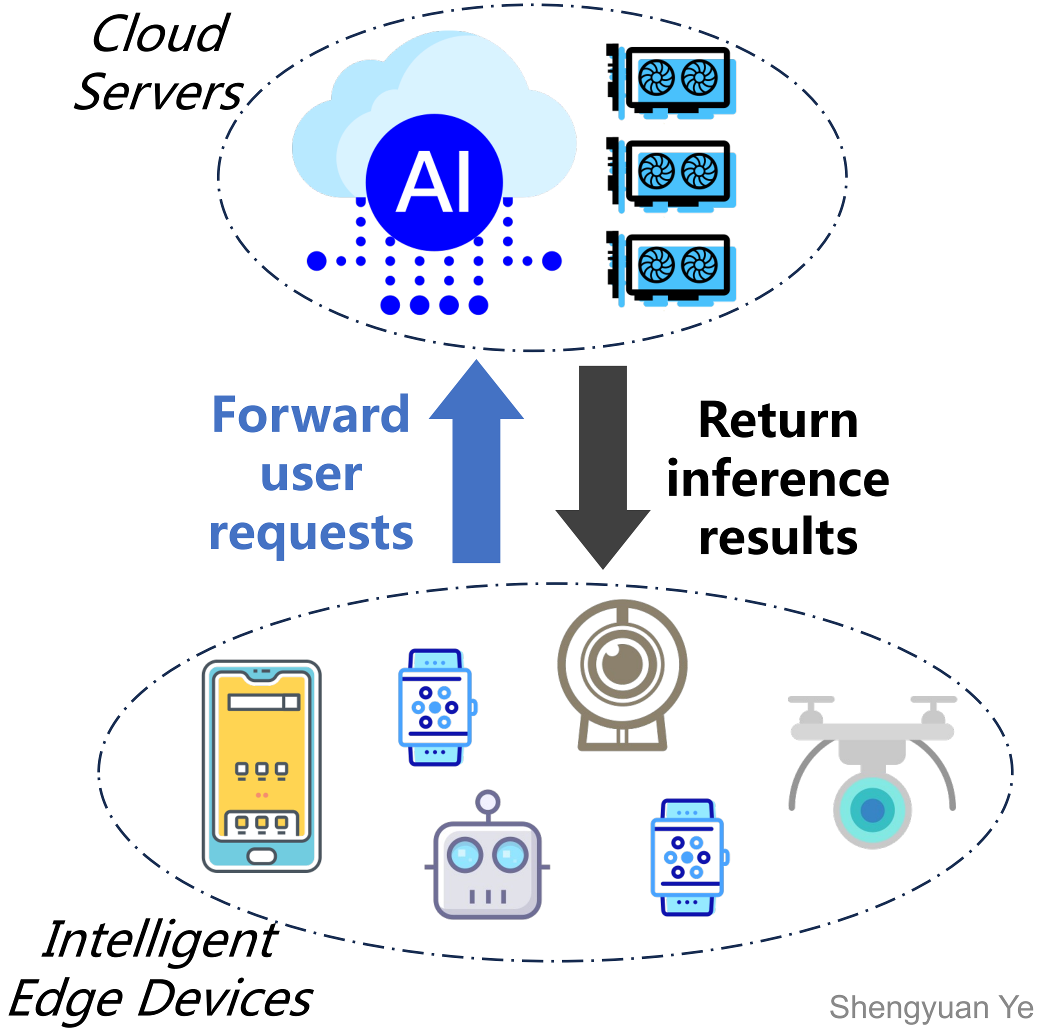


Smart Robotics/UAV

AR & VR APPs

Problems of cloud-assisted approaches

- Current Transformer-based applications heavily depend on **cloud services.**



Benefits of cloud deployment:

- ✔ **Powerful and scalable computing resources.**

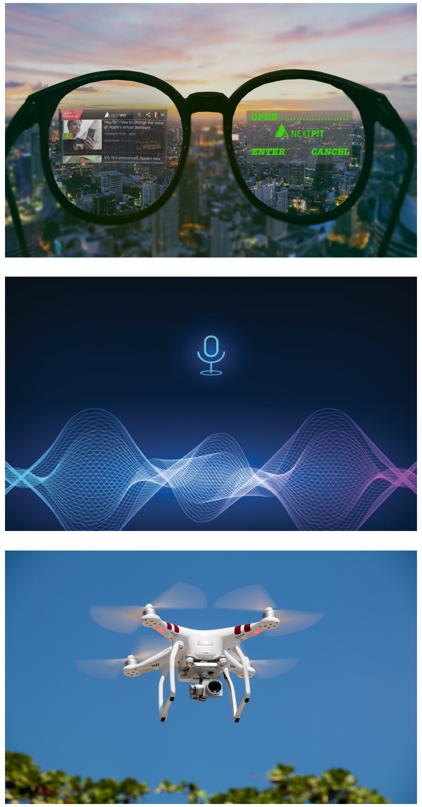
Raising three game-stopping problems:

- **Data privacy concerns.**
- **Unreliable WAN connections.**
- **Network and datacenter pressure.**

Problems of on-device deployment

- **On-device deployment** becomes a promising paradigm for intelligent edge APPs.

Transformer-based intelligent applications



Deploy

Intelligent Edge Devices

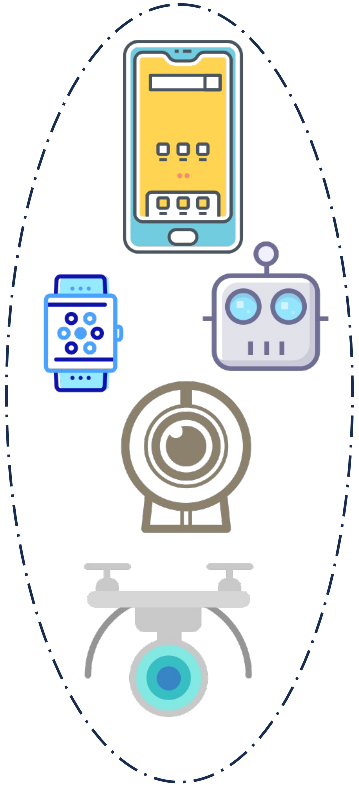


TABLE I

INFERENCE LATENCY AND MEM. FOOTPRINT OF TRANSFORMER MODELS

Model	DistilBert	Bert-L	GPT2-L	OPT-L	OPT-XL
Nano-M	0.37s	2.43s	OOM	OOM	OOM
Nvidia A100	5ms	20ms	29ms	27ms	38ms
Memory Footprint	130MB	680MB	1.6GB	2.6GB	5.4GB

121x performance gap.

Out of memory error.

Protect date privacy.

Without WAN transmission.

Limited and non-scalable on-board computing resources

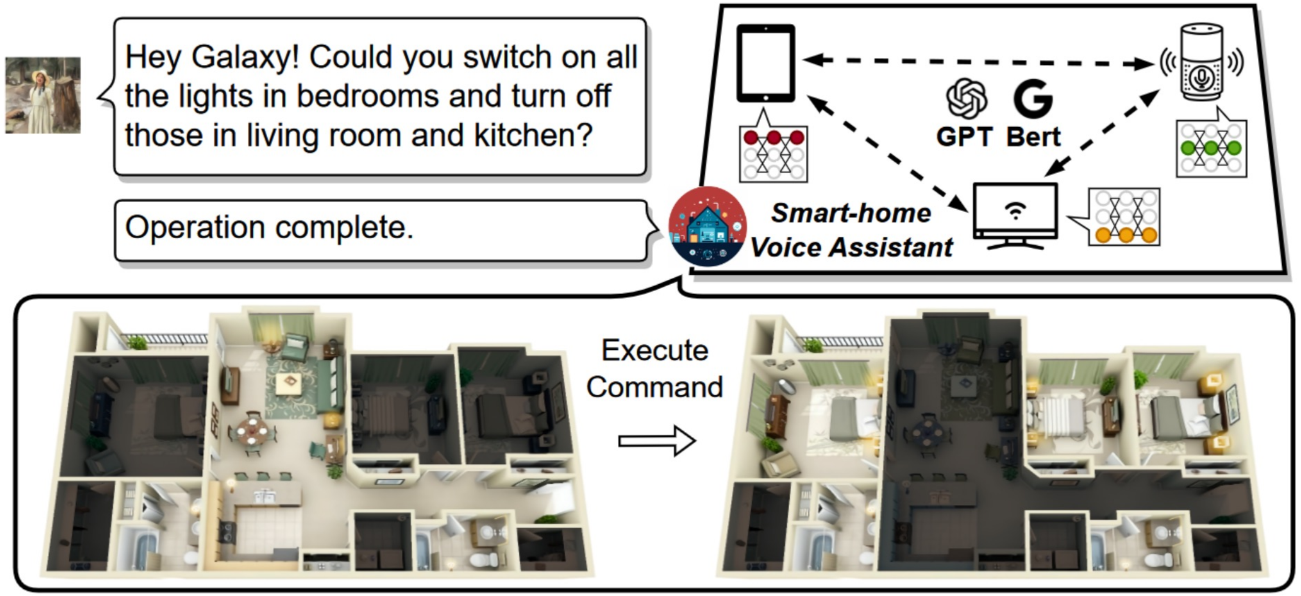
Opportunities

- Edge environment often comprise a rich set of trusted idle edge devices.

Opportunities

✓ Smart homes usually have multiple trusted devices, such as **mobile phones, laptops, and smart-home devices** owned by the same family.

✓ Utilize nearby edge devices as **resource augmentation** to render expedited Transformer inference at the edge.



An illustration of personal AI assistant in a smart home scenario empowered by collaborative edge devices in physical proximity.

Challenges

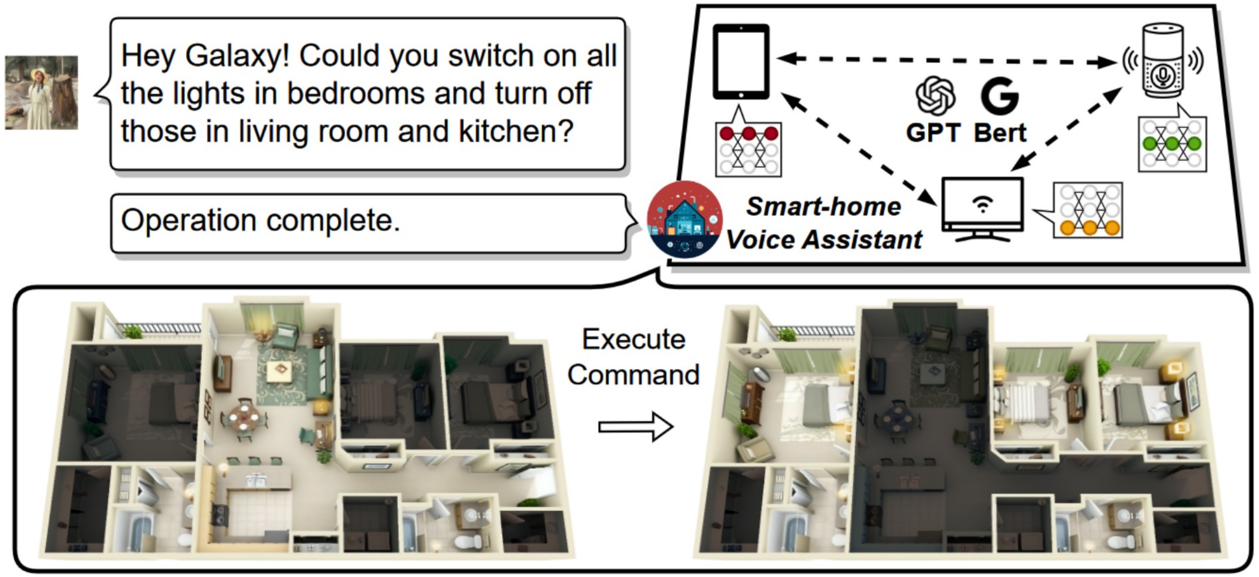
- Edge environment often comprise a rich set of trusted idle edge devices.

Challenges

1. How to distribute sparse Transformer inference, especially **single-sample request**, across multiple edge devices?

2. How to tailor workload partitioning to the resource budget of **heterogeneous** edge devices?

3. How to reduce collaborative inference latency in **bandwidth-limited** edge environments?



An illustration of personal AI assistant in a smart home scenario empowered by collaborative edge devices in physical proximity.

Challenges

- Edge environment often comprise a rich set of trusted idle edge devices.

Challenges

1. How to distribute sparse Transformer inference, especially **single-sample request**, across multiple edge devices.

2. How to tailor workload partitioning to the resource budget of **heterogeneous** edge devices.

3. How to reduce synchronization latency in **bandwidth-limited** edge environments.

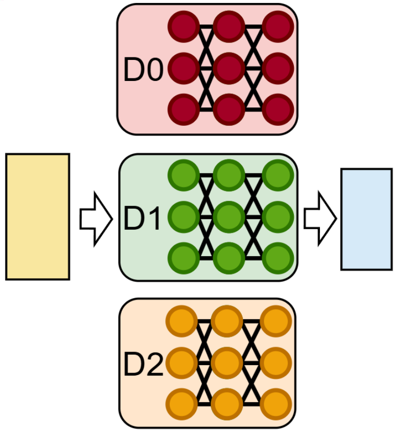
An illustration of personal AI assistant in a smart home scenario empowered by collaborative edge devices in physical proximity.

Hybrid Model Parallelism (HMP) in Galaxy

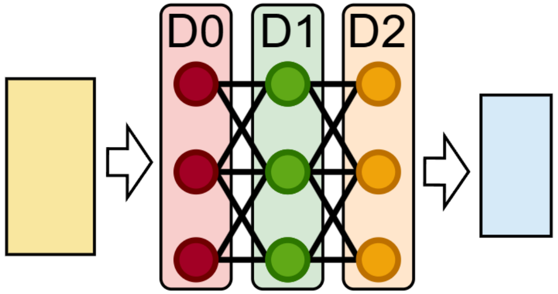
- **Solution to Challenge #1**: Choosing the most suitable parallelism strategy.

Inference in edge environments is often single-shot (e.g., a single voice command).

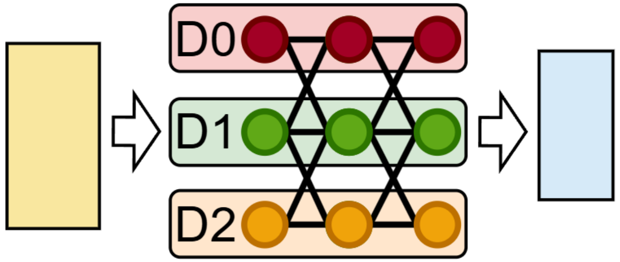
Data Parallelism



Pipeline Parallelism



Model Parallelism

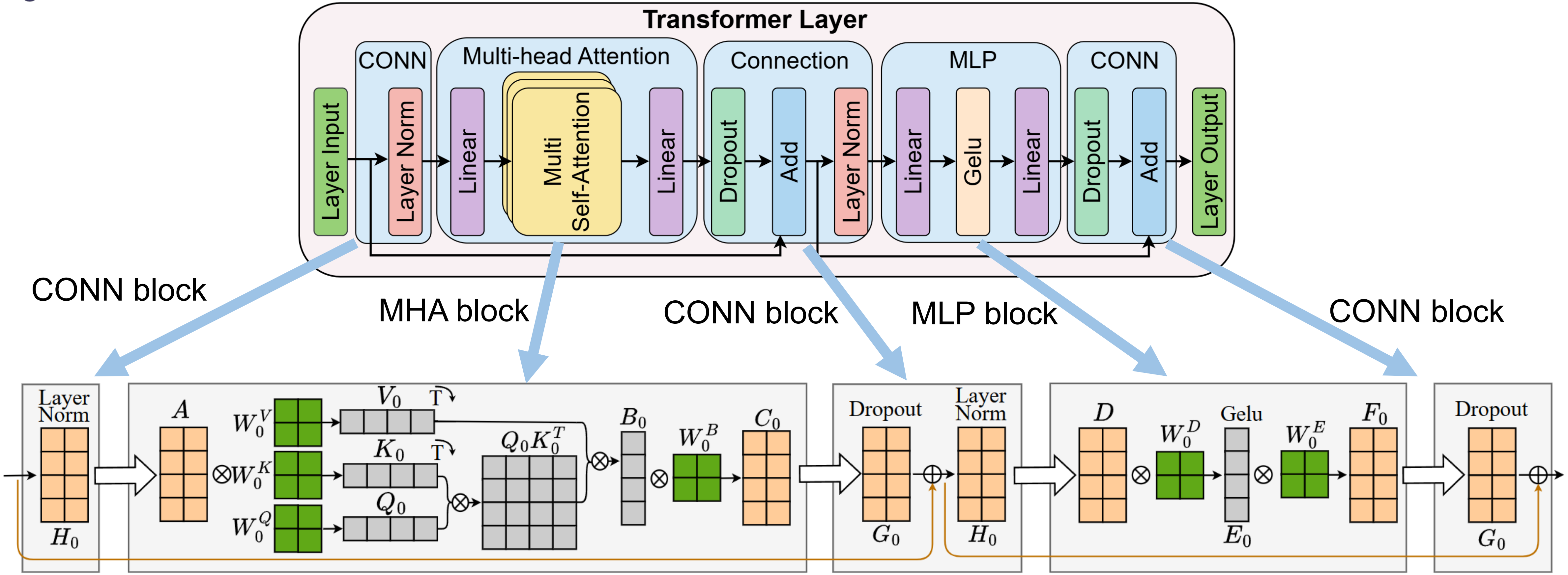


- Data parallelism and pipeline parallelism (batch-level parallelism) is unsuitable due to the inability to leverage multiple edge devices concurrently.
- Model parallelism (operator-level parallelism) is suitable as it facilitates the concurrent execution of single-shot inference.

Hybrid Model Parallelism (HMP) in Galaxy

- **Solution to Challenge #1**: Utilizing **Hybrid Model Parallelism** for orchestration.

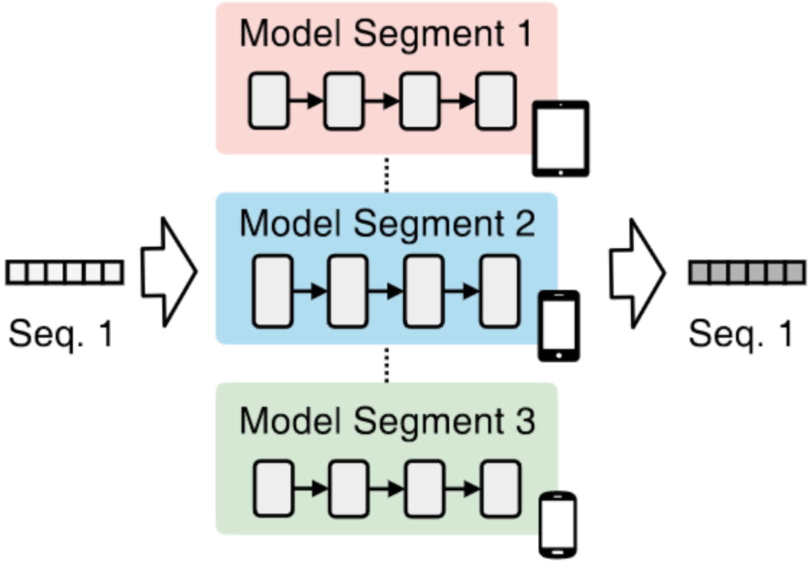
A Transformer layer can be divided into three blocks: MHA, MLP, and CONN.



Hybrid Model Parallelism (HMP) in Galaxy

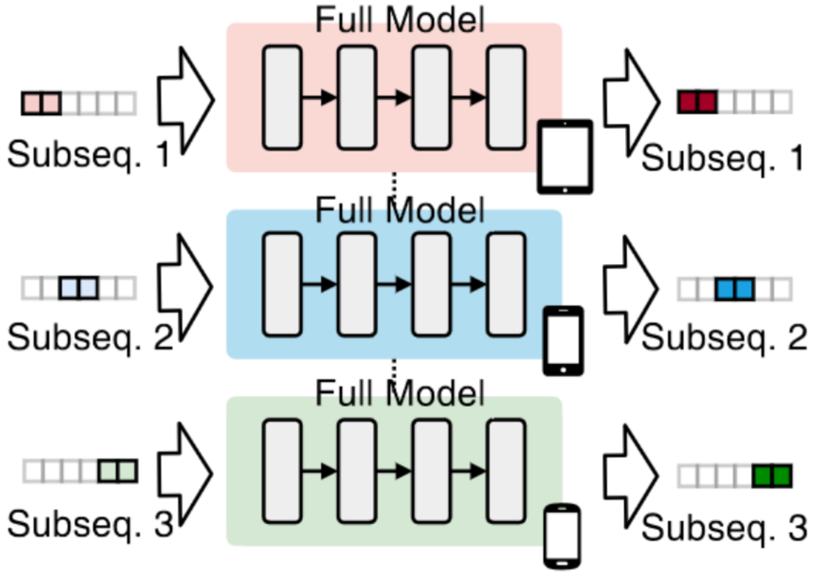
- **Solution to Challenge #1**: Utilizing Hybrid Model Parallelism for orchestration.

➢ **Tensor Model Parallelism**



Tensor model parallelism can only be applied to **MHA** and **MLP** blocks, requiring an tensor synchronization at the end of each block.

➢ **Sequence Model Parallelism**

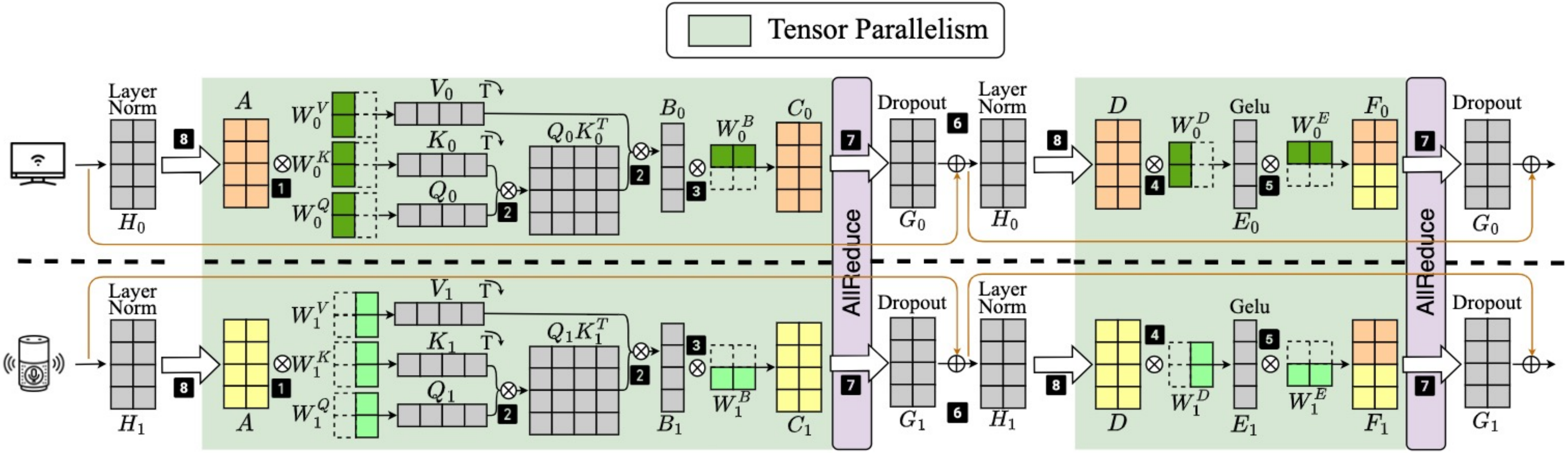


Sequence model parallelism can be applied to **Connection blocks**, which are element-wise operations and require no additional communication.

Hybrid Model Parallelism (HMP) in Galaxy

- **Solution to Challenge #1**: Utilizing hybrid model parallelism for orchestration.

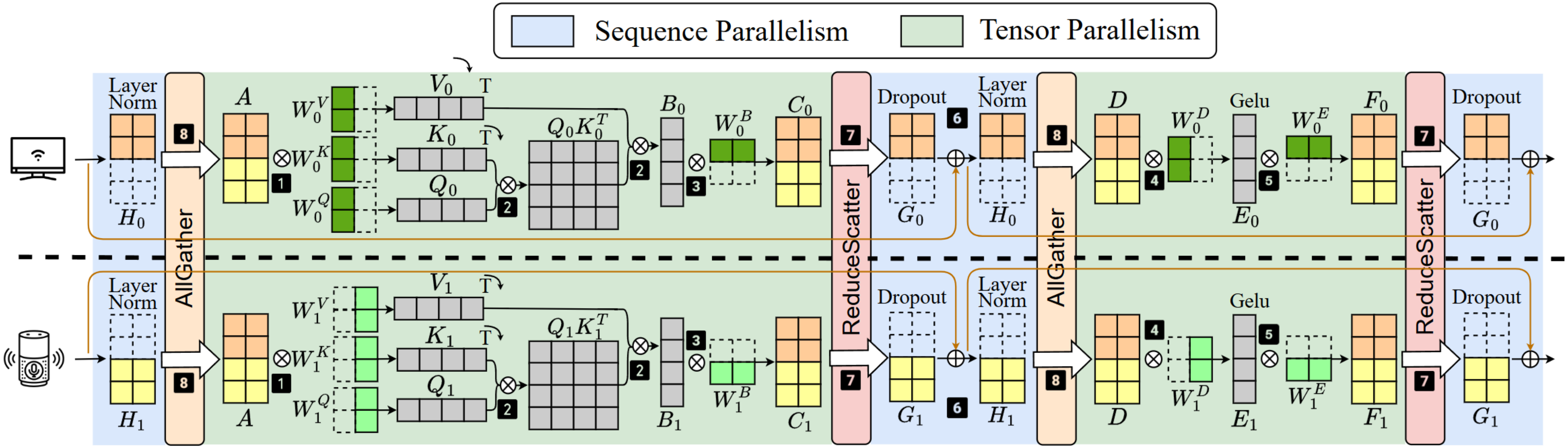
Using Tensor Model Parallelism for a Transformer layer.



An instance of tensor model parallelism across two edge devices

Hybrid Model Parallelism (HMP) in Galaxy

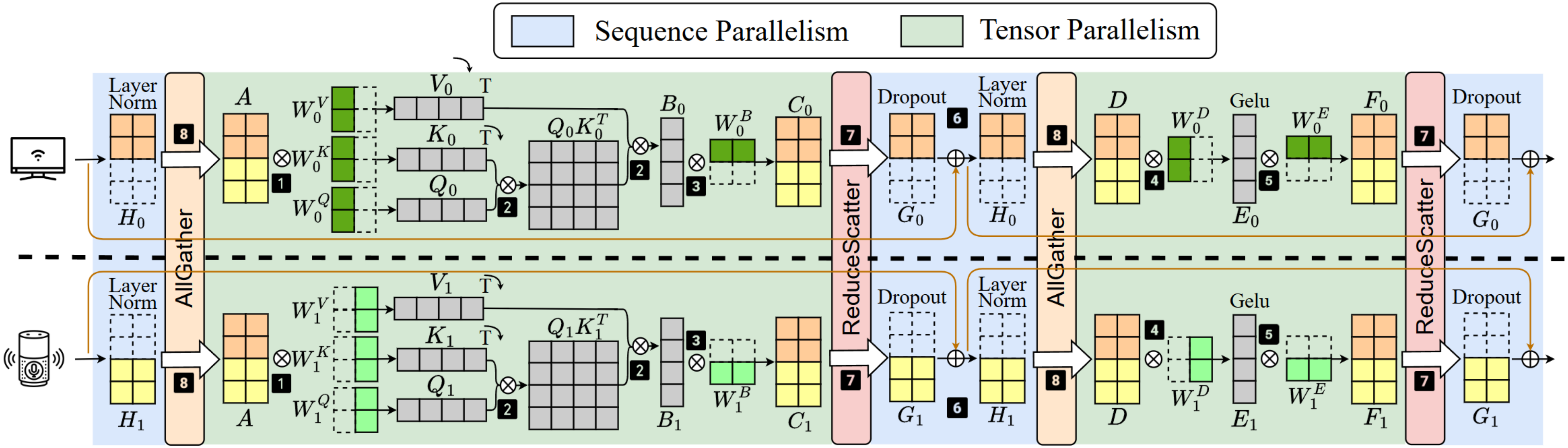
- **Solution to Challenge #1**: Utilizing hybrid model parallelism for orchestration.
  - Using <u>**Tensor Model Parallelism**</u> for MHA and MLP blocks, and <u>**Sequence Model Parallelism**</u> for connection blocks.



An instance of hybrid model parallelism across two edge devices

Hybrid Model Parallelism (HMP) in Galaxy

- **Solution to Challenge #1**: Utilizing hybrid model parallelism for orchestration.
  - **Opportunities:** Split the original heavy AllReduce into two smaller AllGather and ReduceScatter communications, which provides opportunities for overlapping communication and computation.



An instance of hybrid model parallelism across two edge devices

Challenges

- Edge environment often comprise a rich set of trusted idle edge devices.

Challenges

1. How to distribute sparse Transformer inference, especially **single-sample request**, across multiple edge devices.

2. How to tailor workload partitioning to the resource budget of **heterogeneous** edge devices.

3. How to reduce synchronization latency in **bandwidth-limited** edge environments.

An illustration of personal AI assistant in a smart home scenario empowered by collaborative edge devices in physical proximity.

Resource-Aware Workload Partitioning

- **Solution to Challenge #2**: Heterogeneity and Memory Aware Workload Planning.

The initiation of model parallel inference is bound by the completion time of the slowest device (straggler) !

MHA blocks partition results : $\mathcal{A} = \{a_0, a_1, ..., a_{\mathcal{D}-1}\}$

MLP blocks partition results : $\mathcal{B} = \{b_0, b_1, ..., b_{\mathcal{D}-1}\}$

CONN blocks partition results : $\mathcal{S} = \{s_0, s_1, ..., s_{\mathcal{D}-1}\}$

Determined the straggler

$$\mathcal{L}(\mathrm{MHA}, \mathcal{A}) = \max_{d \in \{0,1,\ldots,\mathcal{D}-1\}} L(\mathrm{MHA}, \mathcal{A}_d, d),$$
$$\mathcal{L}(\mathrm{MLP}, \mathcal{B}) = \max_{d \in \{0,1,\ldots,\mathcal{D}-1\}} L(\mathrm{MLP}, \mathcal{B}_d, d),$$
$$\mathcal{L}(\mathrm{CON}, \mathcal{S}) = \max_{d \in \{0,1,\ldots,\mathcal{D}-1\}} L(\mathrm{CON}, \mathcal{S}_d, d).$$

Optimization objective

$$\min_{\mathcal{A},\mathcal{B},\mathcal{S}} \Big(\mathcal{L}(\mathrm{MHA}, \mathcal{A}) + \mathcal{L}(\mathrm{MLP}, \mathcal{B}) + \mathcal{L}(\mathrm{CON}, \mathcal{S}) \Big),$$
$$\text{s.t.} \quad l \cdot (M_{att} \cdot \frac{a_d}{\sum \mathcal{A}} + M_{mlp} \cdot \frac{b_d}{\sum \mathcal{B}}) < \mathrm{Budget}_d,$$
$$\text{where } d \in \{0, 1, \cdots \mathcal{D} - 1\}.$$

Resource-Aware Workload Partitioning

- **Solution to Challenge #2**: Heterogeneity and Memory Aware Workload Planning.

We have designed a lightweight two-step greedy heuristic algorithm.

Algorithm 1: Heterogeneity and Memory Aware Workload Planning

Input: Profiling results of models and devices. $\mathcal{V}$: The list of computing capacity of devices.
Output: $\mathcal{A}, \mathcal{B}$: Partition configurations of MHA and MLP block.

Function `BalacedPartition`($T, \mathcal{V}$):
    Initialize partition configuration C;
    Workload $\leftarrow$ Total workload in block T;
    foreach $d \in \{0, 1, 2, ..., \mathcal{D}-1\}$ **do**
        $C_d \leftarrow (\mathcal{V}_d / \sum \mathcal{V}) \cdot$ Workload;
    Return C;
$\mathcal{A} \leftarrow$ `BalacedPartition`($MHA, \mathcal{V}$);
$\mathcal{B} \leftarrow$ `BalacedPartition`($MLP, \mathcal{V}$);
Function `MemoryAwareBalancing`($T, C, \mathcal{V}, \mathcal{L}$):
    $OOM\_Devices \leftarrow$ Out-of-memory devices under partition configuration C in $\mathcal{L}$;
    $Free\_Devices \leftarrow$ Devices retaining available memory under partition config. C in $\mathcal{L}$;
    if $OOM\_Devices = \emptyset$ **then**
        Return C;
    foreach $o \in OOM\_Devices$ **do**
        Waiting_Shift $\leftarrow$ Overflowing workload on device o;
        foreach $f \in Free\_Devices$ **do**
            Shift $(\mathcal{V}_f / \sum_{i \in Free\_Devices} \mathcal{V}_i) \cdot$ Waiting_Shift workload from o to f;
        Remove device o from $\mathcal{L}$;
    `MemoryAwareBalancing`($T, C, \mathcal{V}, \mathcal{L}$);
$\mathcal{L} \leftarrow [0, 1, .., \mathcal{D}-1]$; $\triangleright$ List of all devices
$\mathcal{B} \leftarrow$ `MemoryAwareBalancing`($MLP, \mathcal{B}, \mathcal{V}, \mathcal{L}$);
$\mathcal{A} \leftarrow$ `MemoryAwareBalancing`($MHA, \mathcal{A}, \mathcal{V}, \mathcal{L}$);
if *Out-of-memory devices still exist* **then**
    Exit with Fail;

Challenges

- Edge environment often comprise a rich set of trusted idle edge devices.

Challenges

1. How to distribute sparse Transformer inference, especially **single-sample request**, across multiple edge devices.

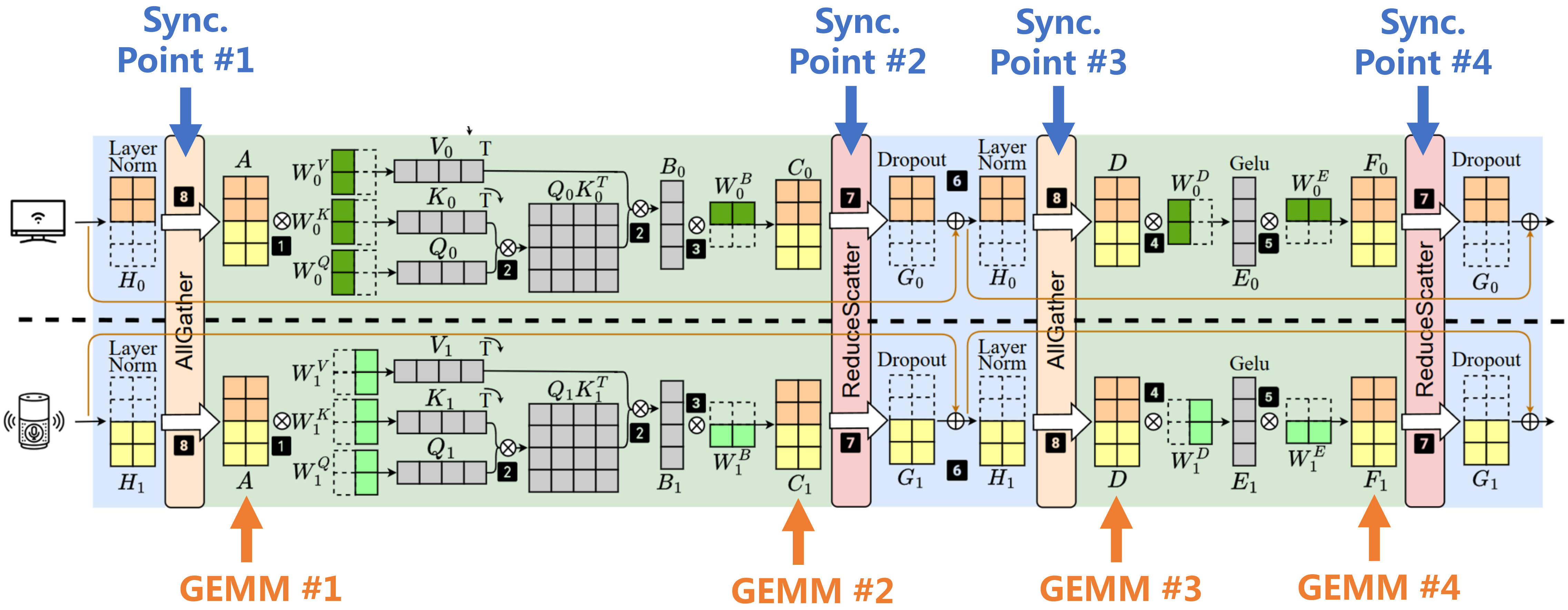
2. How to tailor workload partitioning to the resource budget of **heterogeneous** edge devices.

3. How to reduce synchronization latency under **bandwidth-limited** edge environments.

An illustration of personal AI assistant in a smart home scenario empowered by collaborative edge devices in physical proximity.

Tile-based Communication Optimization

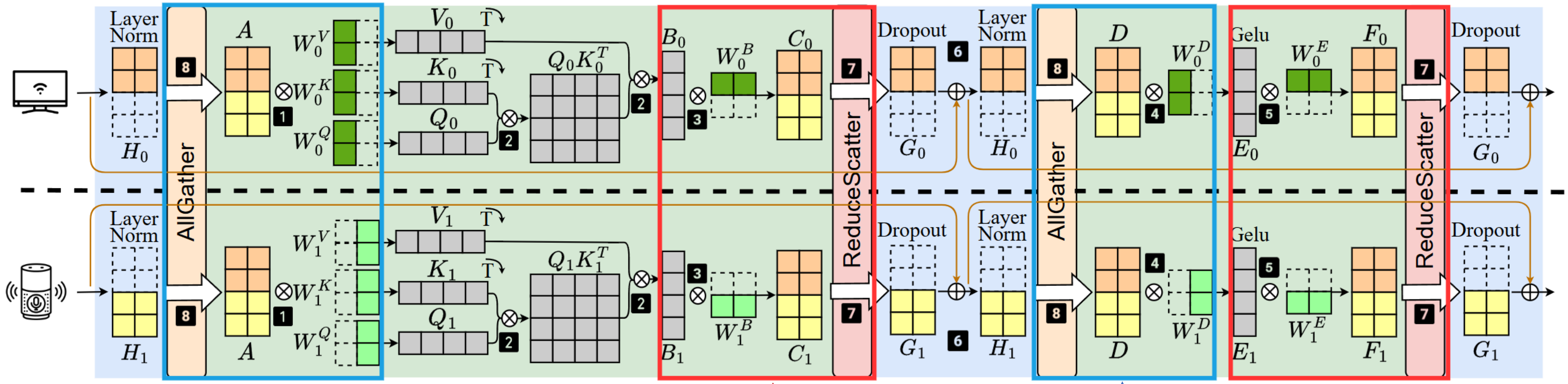
Each Transformer layer require **4 tensor synchronization points**.



GEMM: General Matrix Multiply

Tile-based Communication Optimization

Overlapping communication and computation is an effective optimization strategy.



AllGather-GEMM overlapping

ReduceSactter-GEMM overlapping

AllGather-GEMM overlapping

ReduceSactter-GEMM overlapping

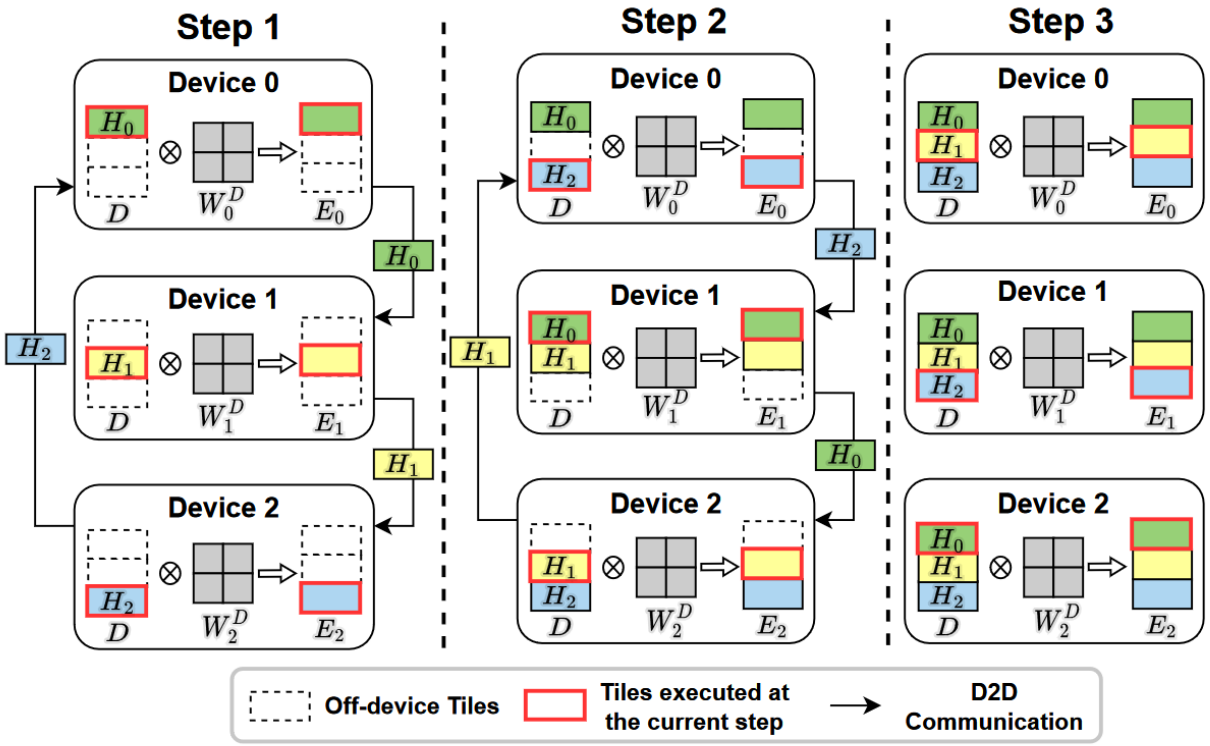
Tile-based Communication Optimization

Ring-AllGather Overlapping

1. Decouple the data dependency between synchronization and GEMM by **partitioning the matrix into submatrices**.

$$E_i = \begin{bmatrix} H_0 \cdot W_i^D \\ \hline H_1 \cdot W_i^D \\ \hline H_2 \cdot W_i^D \end{bmatrix} = \begin{bmatrix} H_0 \\ \hline H_1 \\ \hline H_2 \end{bmatrix} \cdot W_i^D = D \cdot W_i^D .$$

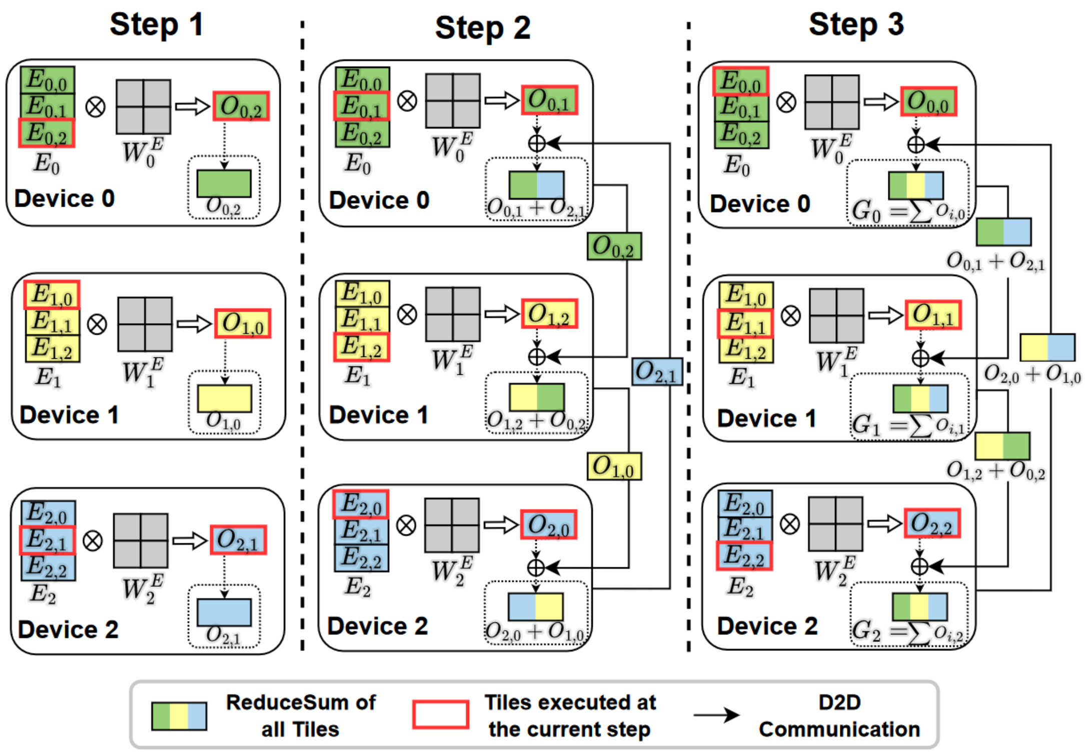
2. We start the GEMM computation for each submatrix immediately after its synchronization, **avoiding the need to wait for the entire matrix.**



An illustration of Ring-AllGather overlapping across three edge devices.

Tile-based Communication Optimization

Ring-ReduceScatter Overlapping



An illustration of Ring-ReduceScatter overlapping across three edge devices.

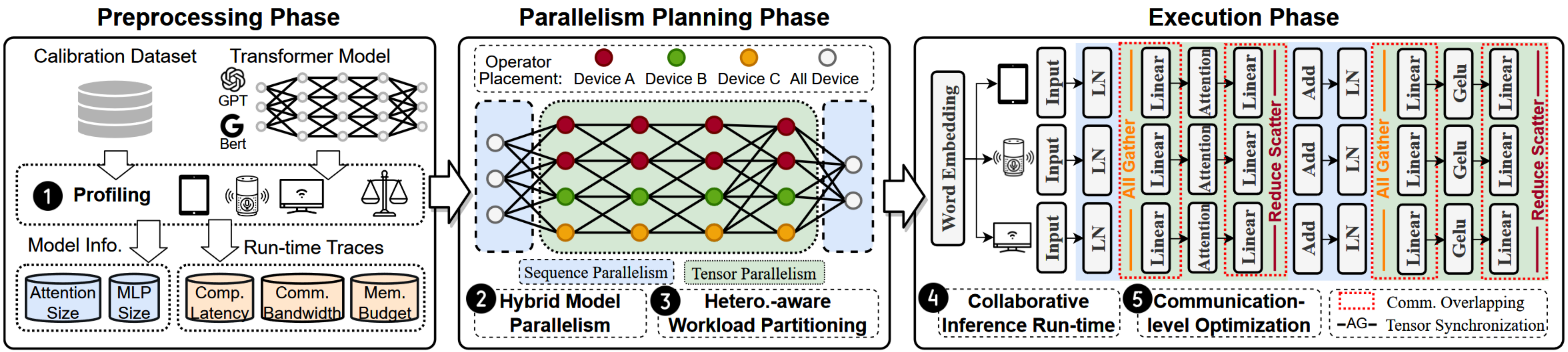
1. Decouple the data dependency between synchronization and GEMM by **partitioning the matrix into submatrices**.

$$\begin{bmatrix} O_{i,0} \\ \hline O_{i,1} \\ \hline O_{i,2} \end{bmatrix} = \begin{bmatrix} E_{i,0} \cdot W_i^E \\ \hline E_{i,1} \cdot W_i^E \\ \hline E_{i,2} \cdot W_i^E \end{bmatrix} = \begin{bmatrix} E_{i,0} \\ \hline E_{i,1} \\ \hline E_{i,2} \end{bmatrix} \cdot W_i^E = E_i \cdot W_i^E,$$

2. We start the GEMM computation for each submatrix immediately after its synchronization, **avoiding the need to wait for the entire matrix.**

Putting It All Together

- **Galaxy system workflow.**



1. Preprocessing phase:

- **Galaxy Profiler** records run-time traces needed for planning using calibration data on edge devices.

2. Parallelism Planning Phase:

- **Galaxy Planner** takes profiling results from Galaxy Profiler as input to generate a parallelism planning configuration.

3. Execution Phase:

- **Galaxy Runtime** applies the planning configuration to target models and edge devices for efficient edge collaborative inference.

Evaluation

- **Testbeds**
  - Off-the-shelf edge devices: NVIDIA Jetson Nano.
  - Simulate **three** heterogeneous computing devices by adjusting the SoC frequency.

Hardware	Specifications	
CPU	Quad Core ARM Cortex-A53 CPU	
GPU	128 Core Maxwell GPU	
CPU Frequency Mode	Nano-S	403MHz
	Nano-M	825MHz
	Nano-L	1.47GHz

Evaluation

- **Testbeds**
  - ➢ Using these 3 heterogeneous devices, we simulated **6 different edge clusters**, including both homogeneous and heterogeneous clusters.

ID	Homogeneous Edge Env.	ID	Heterogeneous Edge Env.
A	2 $\times$ Nano-M	D	Nano-L + Nano-M
B	3 $\times$ Nano-M	E	Nano-L + Nano-S
C	4 $\times$ Nano-M	F	Nano-L + Nano-M + Nano-S

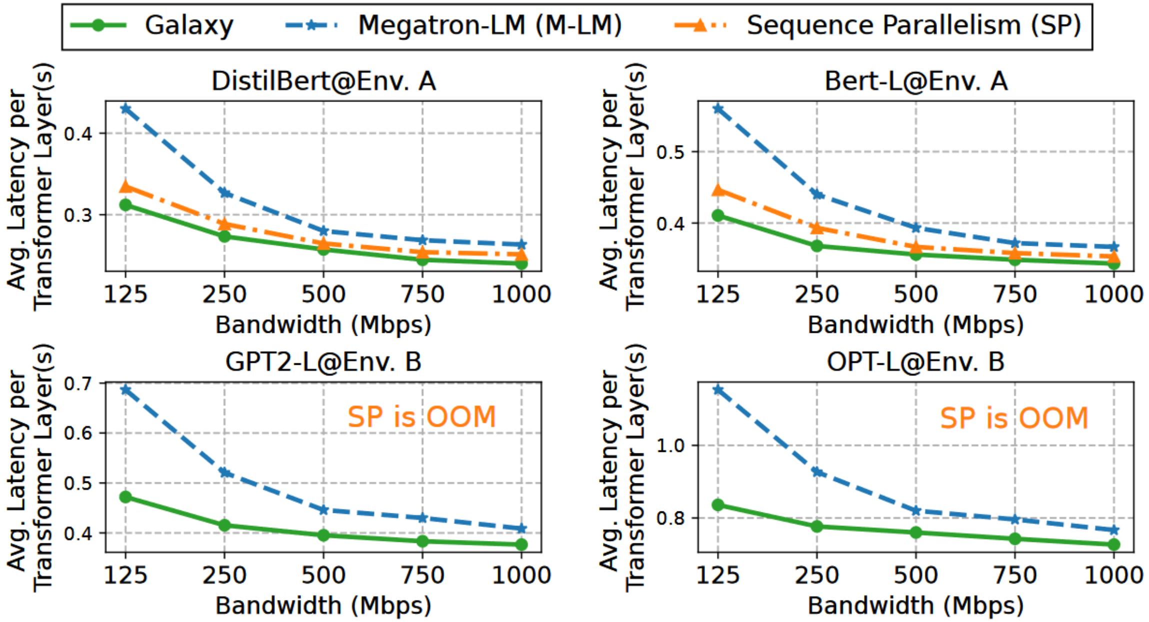
- **Models and datasets**
  - ➢ 5 prevalent Transformer-based models: DistilBert, Bert, GPT2-L, OPT-L, OPT-XL.
  - ➢ Evaluate with the input sequences from GLUE dataset.

Evaluation

- **Baselines**
  - Megatron-LM (M-LM) [24]: A state-of-the-art tensor model parallelism method.
  - Sequence Parallelism (SP) [25]: A state-of-the-art sequence model parallelism method

Maintained high performance **across various network environments**, with up to **46% latency reduction** compared to baseline methods!!!

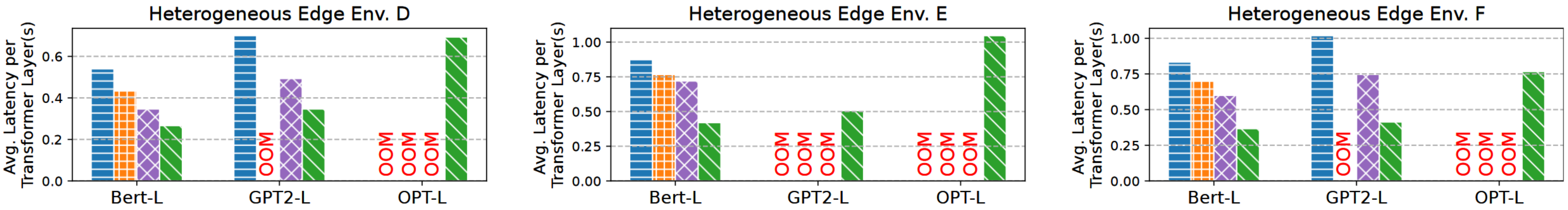
Model	Layers	Heads	Hidden Layer	Edge Env.	Speedup Over M-LM	Speedup Over SP
DistilBert [33]	6	12	768	A	1.37×	1.08×
Bert-L [1]	24	16	1024	A	1.36×	1.09×
				B	1.38×	1.11×
GPT2-L [12]	36	20	1280	A	1.31×	OOM
				B	1.46×	OOM
OPT-L [34]	24	16	2048	A	1.26×	OOM
				B	1.40×	OOM
				C	1.43×	OOM
OPT-XL [34]	32	32	2560	A	OOM	OOM
				B	OOM	OOM
				C	1.28×	OOM



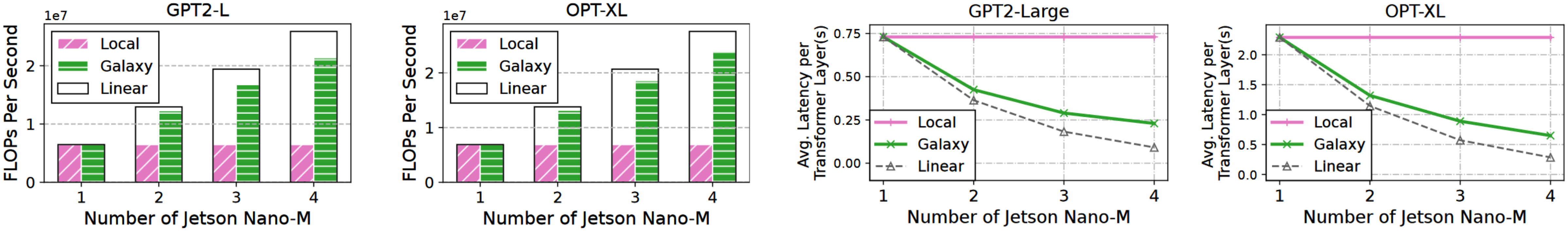
Evaluation

- System performance under **heterogeneous edge environments**.

Galaxy consistently and remarkly outperforms other parallelism methods in heterogeneous edge environments, reducing inference latency by **1.3x to 2.5x**.



Galaxy achieves **86% linear scaling** with parallel inference on 4 Nvidia Jetson Nano devices.



Takeaway

Eco: An <u>E</u>dge <u>**CO**</u>llaborative AI framework for serving miscellaneous AI model at the edge.

We aim to design affordable, accessible, and adaptive AI with your private group of mobile and edge devices.

https://collaborative-edge-ai.github.io/

Eco Project Page

Technical blog

Features

😊 Optimized Computation

- Language models
- Vision perceptrons
- Graph nets

⚒ Heterogeneity Awarenss

- Mobile phones
- Embedded devices
- Edge servers

🏄 Resilient Elasticity

- Device breakdown
- Load variation
- Bandwidth fluctution

IEEE ComSoc
IEEE Communications Society
IEEE
中山大學
SUN YAT-SEN UNIVERSITY
香港科技大学(广州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Thanks for listening

Shengyuan Ye[1], Jiangsu Du[1], Liekang Zeng[1,2], Wenzhong Ou[1], Xiaowen Chu[2], Yutong Lu[1], Xu Chen[1]

[1]Sun Yat-sen University, [2]HKUST(GZ)

Eco Project Page | Technical blog | Personal Website