IEEE INFOCOM
IEEE International Conference on Computer Communications
19–22 May 2025 // London, United Kingdom

# Jupiter: Fast and Resource-Efficient Collaborative Inference of Generative LLMs on Edge Devices

**Shengyuan Ye**[1], Bei Ouyang[1], Liekang Zeng[2], Tianyi Qian[1], Xiaowen Chu[3], Jian Tang[4], Xu Chen[1]

[1] Sun Yat-sen University
[2] The Chinese University of Hong Kong
[3] The Hong Kong University of Science and Technology (GZ)
[4] Midea Group

# Intelligent edge applications

- Large language models driven increasing intelligent applications.

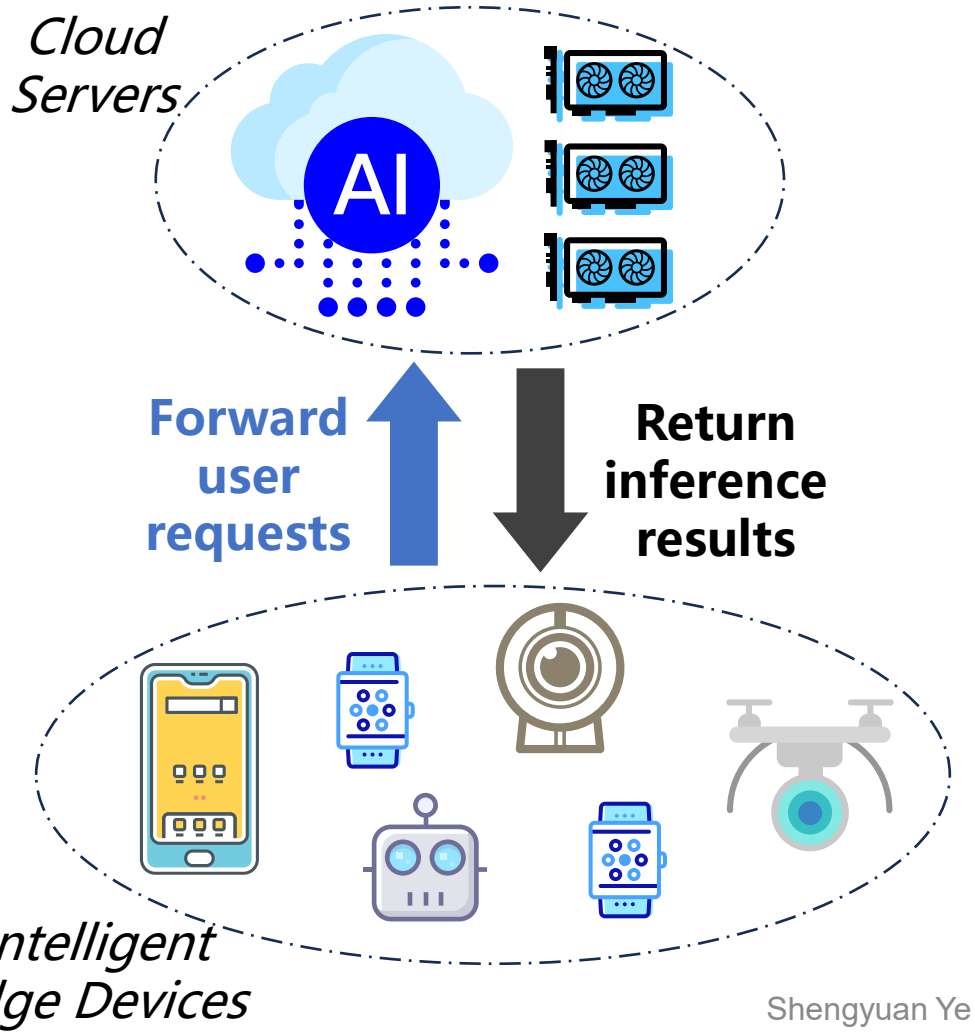

Personal AI Assistants

Smart Robotics/UAV


AR & VR APPs

# Problems of cloud-assisted approaches

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




**Benefits of cloud deployment:**

- **Powerful and scalable computing resources.**

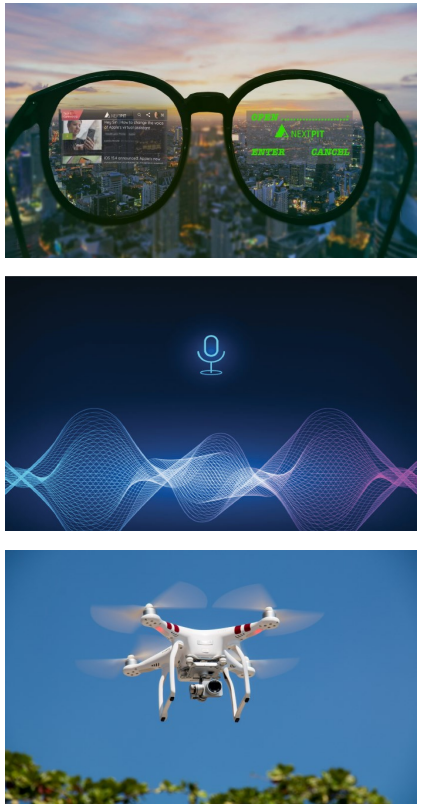
**Raising three game-stopping problems:**

- **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*

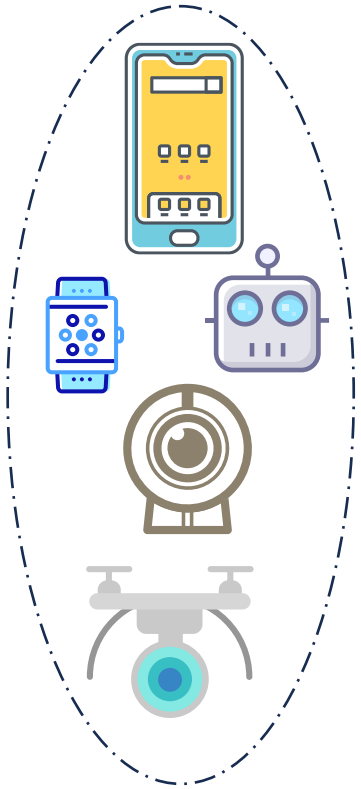


TABLE I
INFERENCE LATENCY AND MEM. FOOTPRINT OF TRANSFORMER MODELS

| 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.

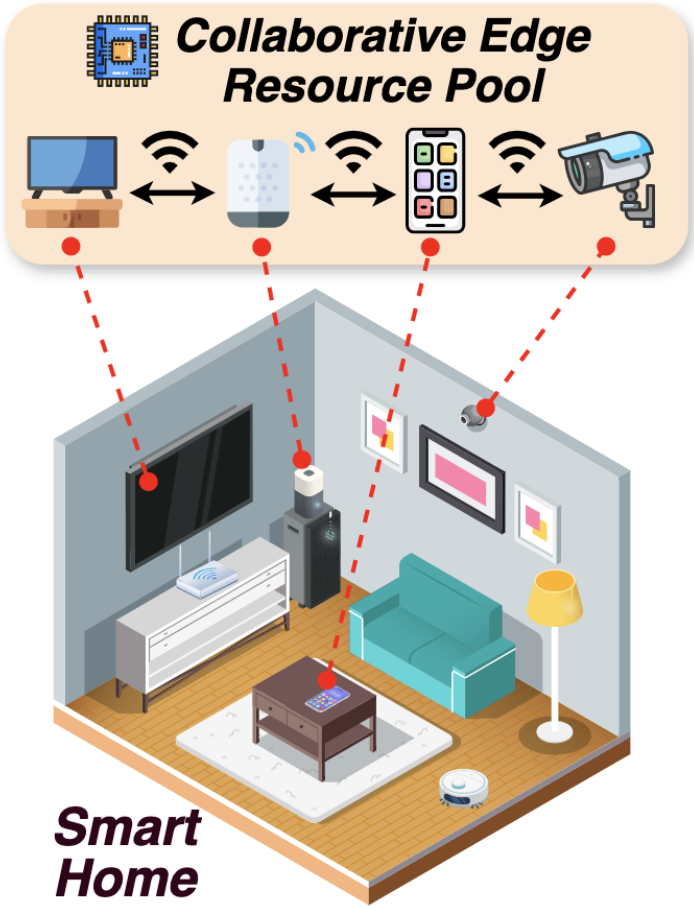
- **Protect date privacy.**
- **Without WAN transmission.**
- **Limited and non-scalable on-board computing resources**

# 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 LLMs inference at the edge.



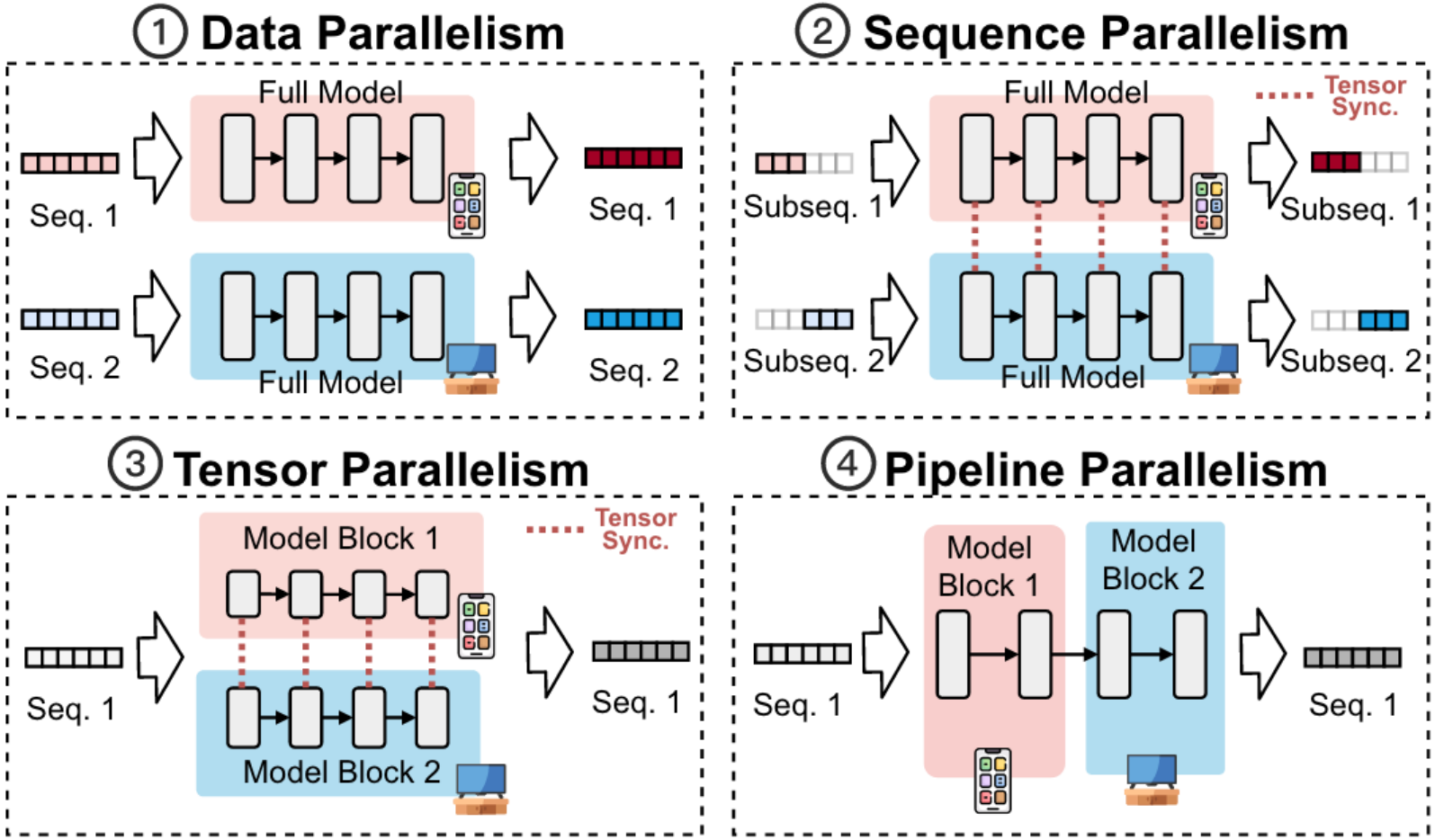

*Smart Phones/Watches*

*Personal Laptops/Server*

*Edge Computing Platforms*

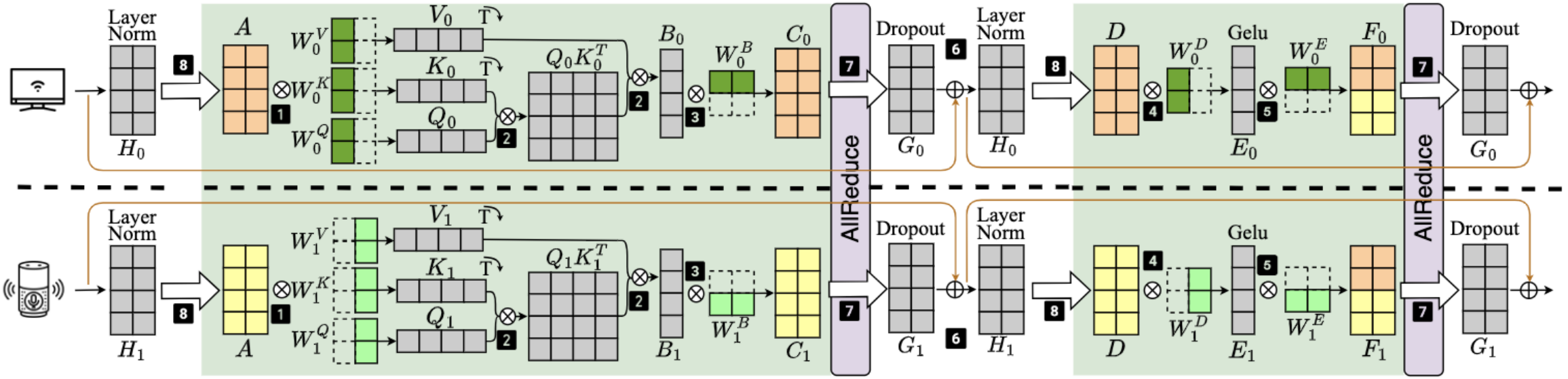
# Distributed Inference Methods for LLMs

- DP, SP, TP, and PP represent the four most prominent techniques for parallel inference in large language models (LLMs).



# Issues in Existing Collaborative Edge System

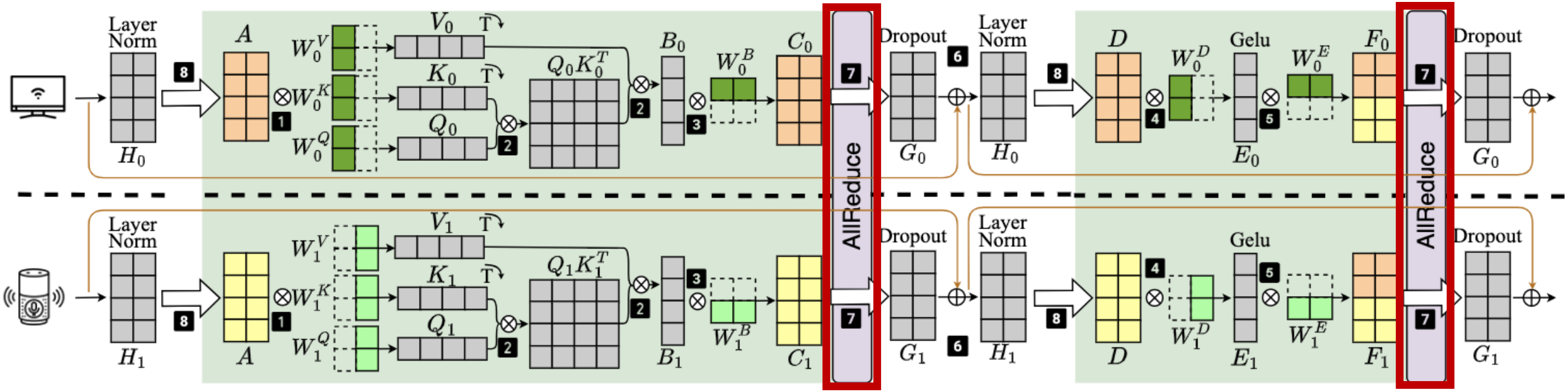
- The primary goal of edge inference systems is to reduce the latency of **single-sequence requests** from users. Common DP or PP fail to meet the requirements.
- Existing collaborative edge inference systems predominantly use **TP or SP** as the major parallel architecture, as exemplified by Galaxy [1] and DeTransformer (DT) [2].




[1] Ye et al., "Galaxy: A Resource-Efficient Collaborative Edge AI System for In-situ Transformer Inference". IEEE International Conference on Computer Communications (INFOCOM), 2024.
[2] Wei et al., "Communication-Efficient Model Parallelism for Distributed In-situ Transformer Inference". Design, Automation and Test in Europe Conference (DATE), 2024.

# Issues in Existing Collaborative Edge System

- TP/SP-based architectures require **multiple collective communications** within each decoder layer, leading to significant communication overhead.
- In each decoder layer, **TP** requires **two AllReduce** operations to synchronize tensors, while **SP** requires **two AllGather** operations to collect keys and values.




An instance of tensor model parallelism across two edge devices

# Issues in Existing Collaborative Edge System

- TP/SP-based architectures require **multiple collective communications** within each decoder layer, leading to significant communication overhead.

TABLE II

COMM.-TO-COMP. RATIO OF VARIOUS PARALLELISM METHODS.

| Model Name | Network Bandwidth | Communication-to-Computation Ratio | | | | |
|---|---|---|---|---|---|---|
| | | SP | TP | DT [7] | Galaxy [6] | Jupiter |
| Llama2-7B | 100Mbps | 8.16 | 6.96 | 3.48 | 5.19 | **0.08** |
| | 1Gbps | 0.92 | 0.88 | 0.45 | 0.69 | **0.01** |
| Llama2-13B | 100Mbps | 5.71 | 6.06 | 3.03 | 4.63 | **0.05** |
| | 1Gbps | 0.73 | 0.81 | 0.38 | 0.56 | **0.01** |

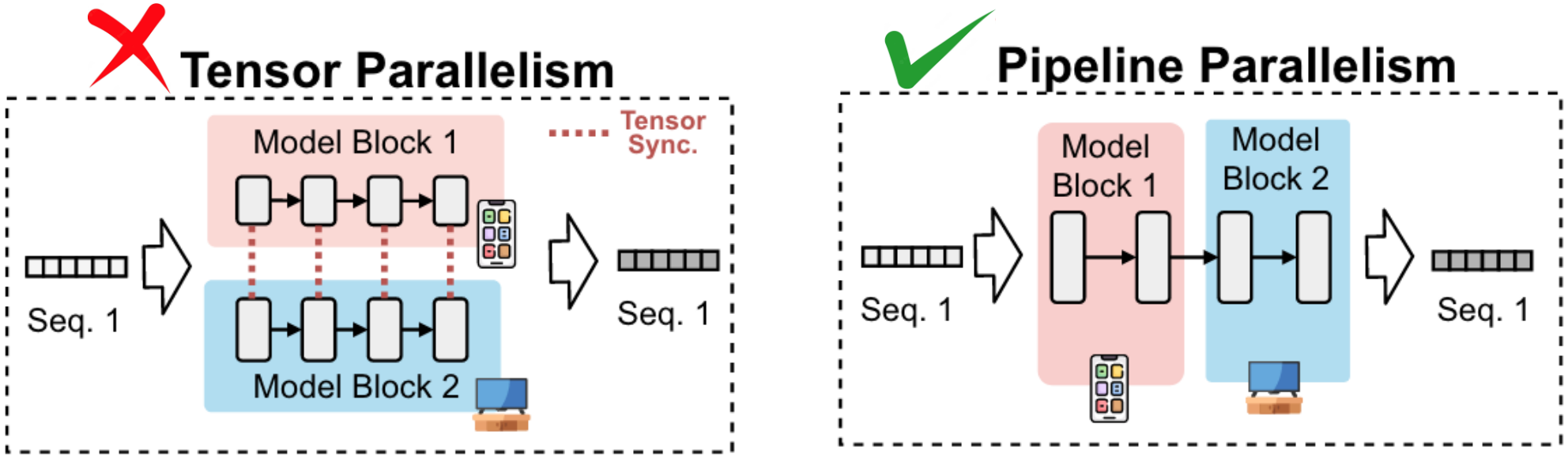
Up to 92.6x

**Existing TP/SP-based systems exhibit a high communication-to-computation latency ratios making communication a critical bottleneck.**

# Architecture Choices of Jupiter

- Jupiter employing a **pipeline architecture** to orchestrate multiple edge devices.

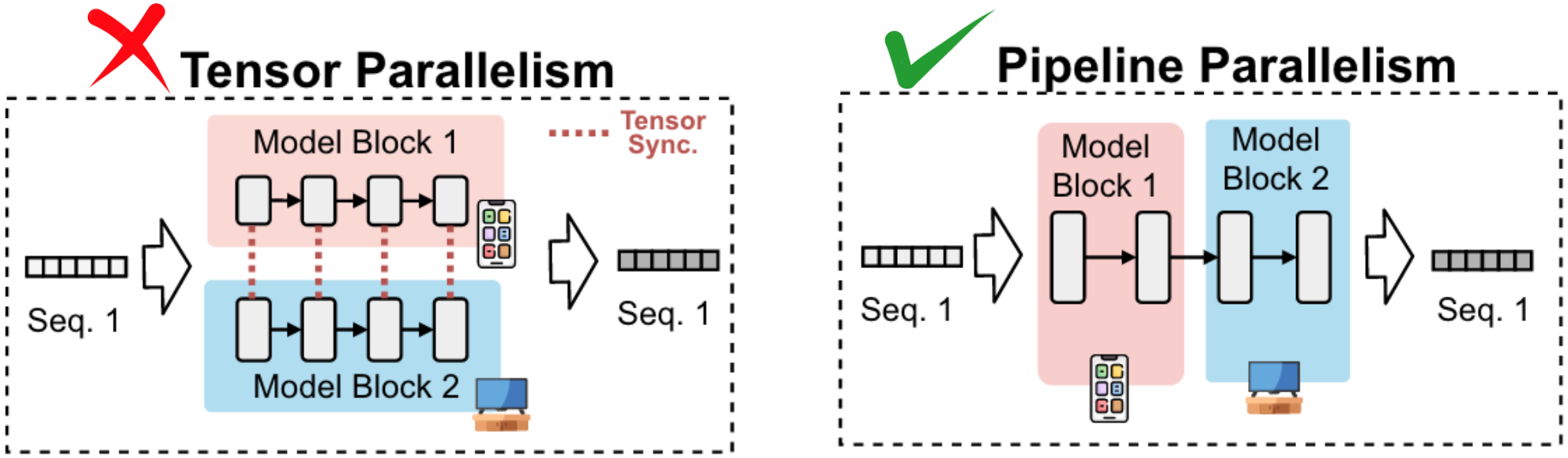
**Observation:** Pipeline architecture has a ***lower communication-to-computation ratio*** during collaborative LLM inference, making it more suitable for low-bandwidth edge environments.



# Architecture Choices of Jupiter

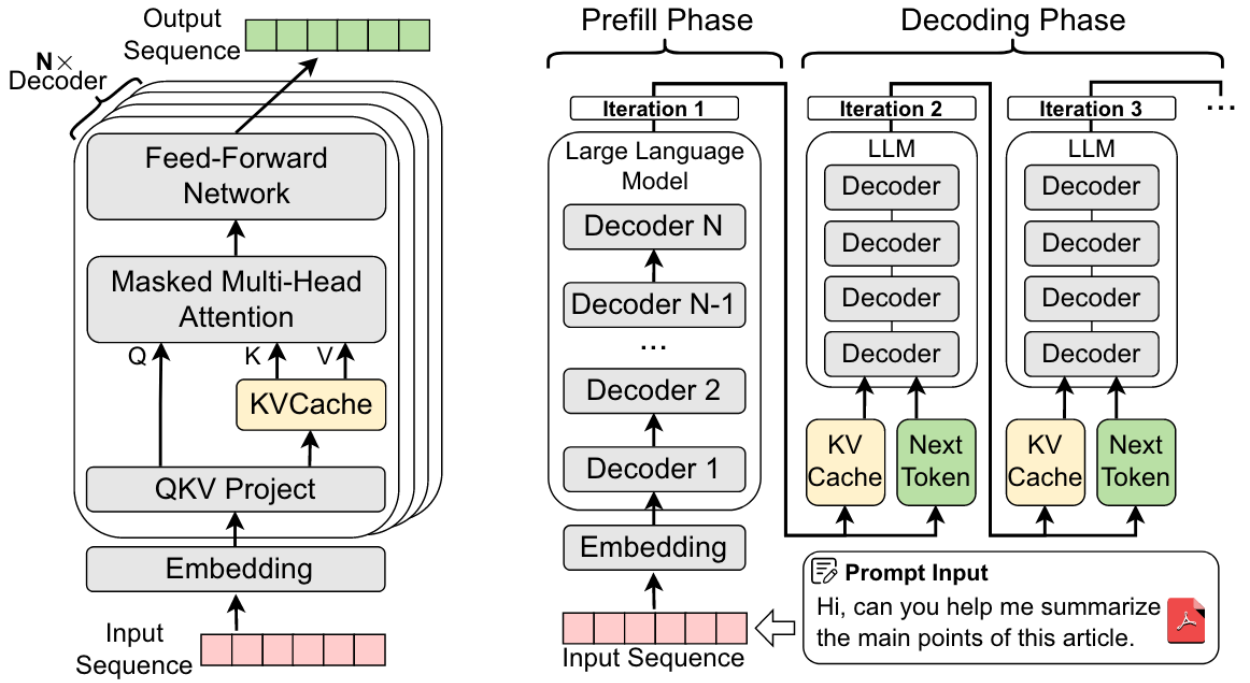
- Jupiter employing a **pipeline architecture** to orchestrate multiple edge devices.

**Challenges:** Pipeline architecture struggles to accelerate inference for single-sequence requests ***due to the lack of batch dimension***.



# Issues in Existing Collaborative Edge System

- Existing research works are mostly limited to discriminative tasks and **focus solely on the prefill phase**.
- The main strength of modern LLMs lies in generative decoding tasks, but most existing systems are **not designed or optimized for the decode phase**.



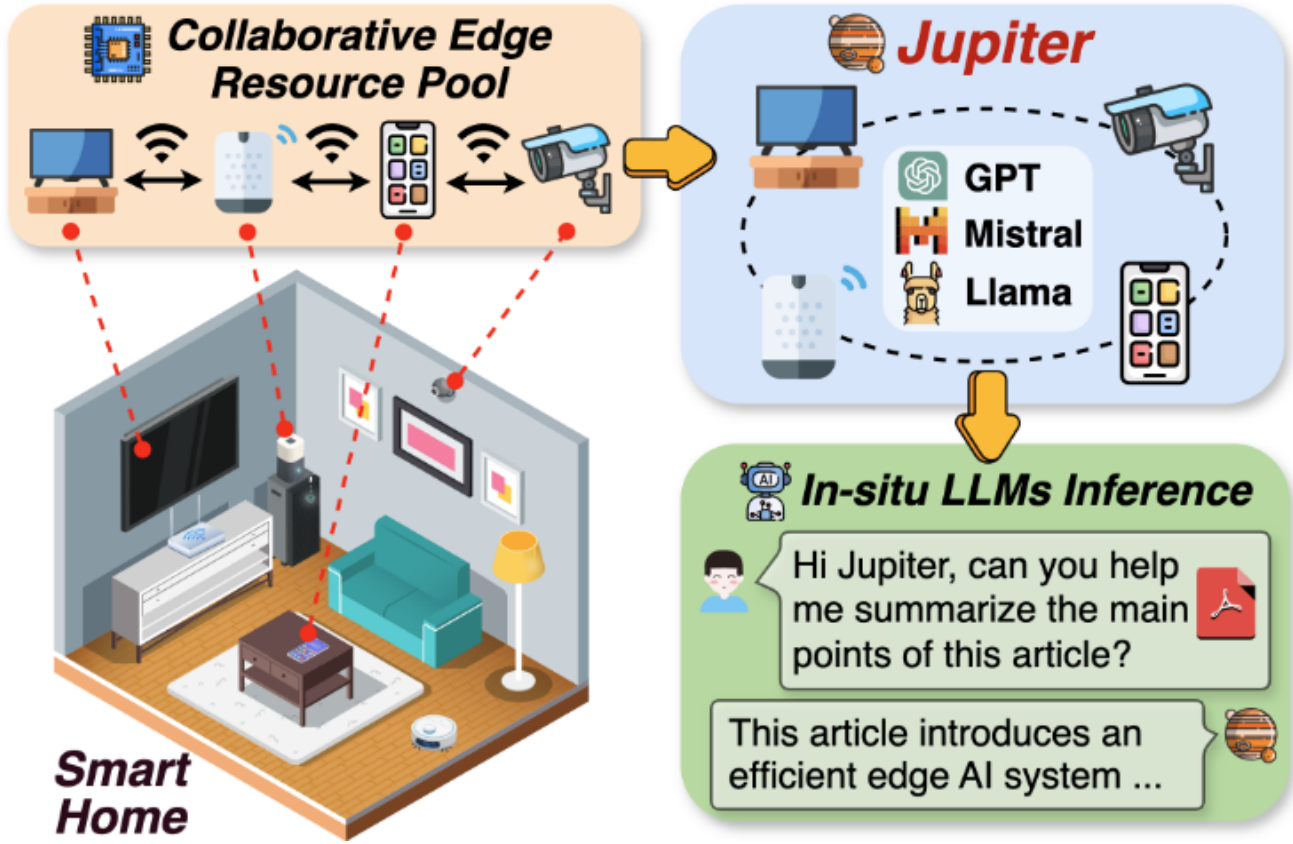
# Challenges

- Efficiently leveraging the computational resources of multiple edge devices under a **pipeline architecture** is a non-trivial problem.

## Challenges

**1.** How to accelerate the **prefill phase** with pipelined architecture for single sequence request.

**2.** How to accelerate the **autoregressive decoding phase** with pipelined architecture.



# Challenges

- Efficiently leveraging the computational resources of multiple edge devices under a **pipeline architecture** is a non-trivial problem.

## Challenges

**1.** How to accelerate the **prefill phase** with pipelined architecture for single sequence request.

**2.** How to accelerate the **autoregressive decoding phase** with pipelined architecture.

# Collaborative Inference for Prefilling Phase

- **Solution to Challenge #1**: **Utilizing Intra-sequence pipeline parallelism.**

1. Partition the LLM into multiple stages, each handled by a separate edge device.
2. Partition the input sequence into **multiple sub-sequences** to increase parallelism.
3. When computing the sub-sequence $s_i$, the **cached key and value** of $s_1, \ldots, s_{i-1}$ are utilized to ensure accurate self-attention results.

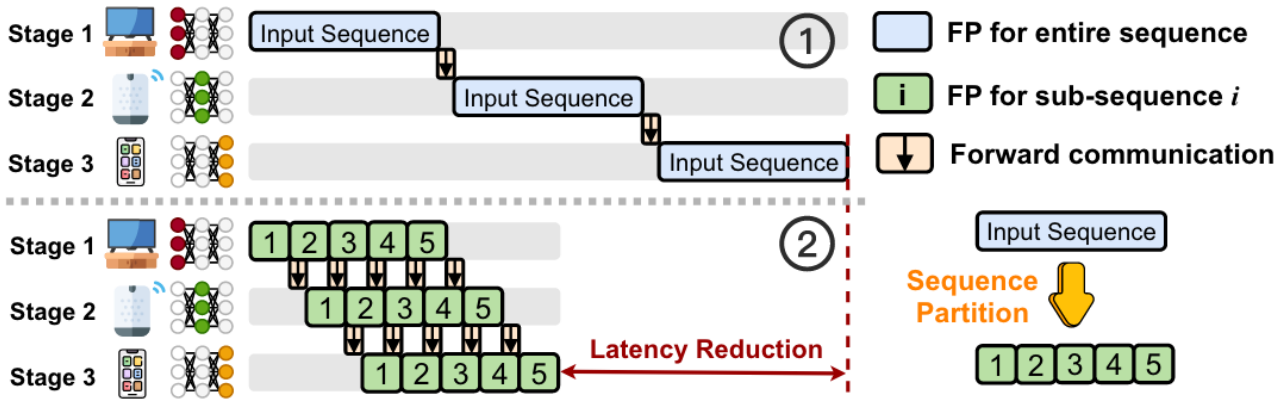
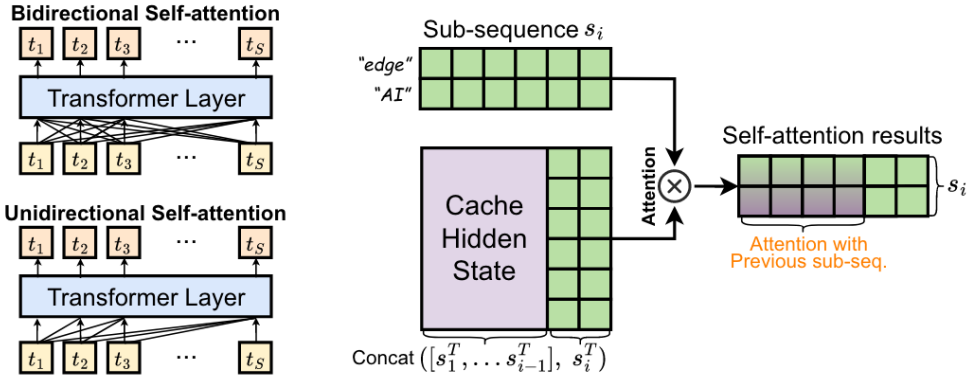



Fig. 5. An illustration of pipelined inference with three edge devices.



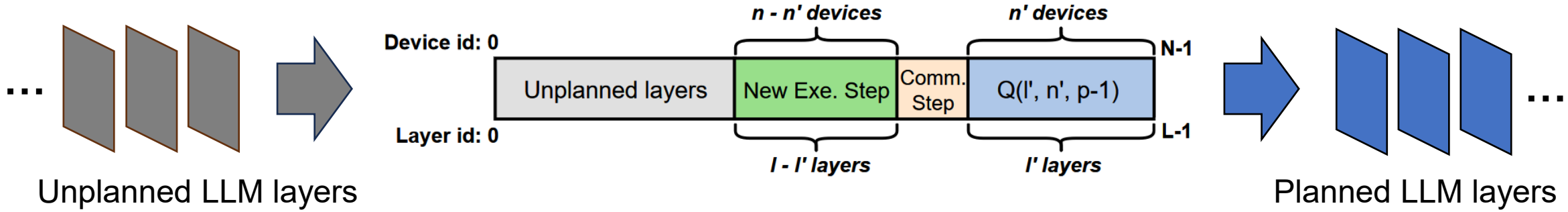

(a) A comparison of two mainstream self-attention.

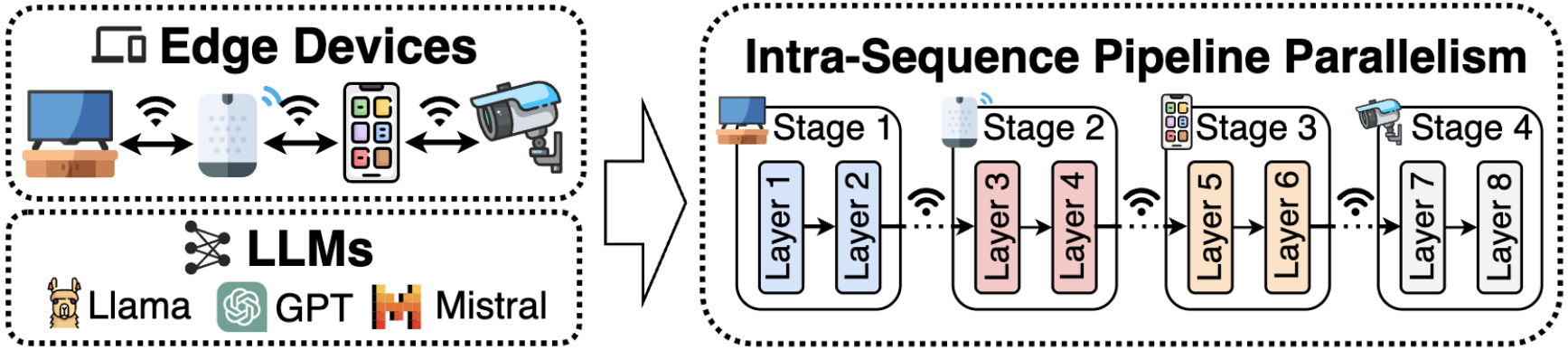
(b) The self-attention for sub-sequence $s_i$ utilizes the cached hidden state of previous sub-sequences.

Fig. 6. An illustration of opportunities of intra-sequence parallel inference.

# Optimal Model and Sequence Partition

- **Selecting Optimal LLMs Partition**
  - Use dynamic programming algorithm to partition the target LLM into multiple stages with **equal execution time**.



# Optimal Model and Sequence Partition

- **Selecting Optimal Sequence Partition**

**Sequence Partition is not trivial for the following reasons:**

1. Increasing the number of sub-sequences can efficiently boost parallelism. However, this results in shorter sub-sequences, which may **underutilize the mobile accelerators**.
2. Partitioning the sequence into sub-sequences of equal length for pipelining is not optimal. **Later sub-sequences carry a heavier computational load** than earlier ones.
3. The input sequence **lengths vary across different requests**, necessitating the determination of the optimal partitioning strategy for any possible length prior to online service deployment

# Optimal Model and Sequence Partition

- **Selecting Optimal Sequence Partition**
  - We carefully profile and model the inference latency and design a dynamic programming algorithm to determine the optimal partitioning for requests with varying sequence lengths.

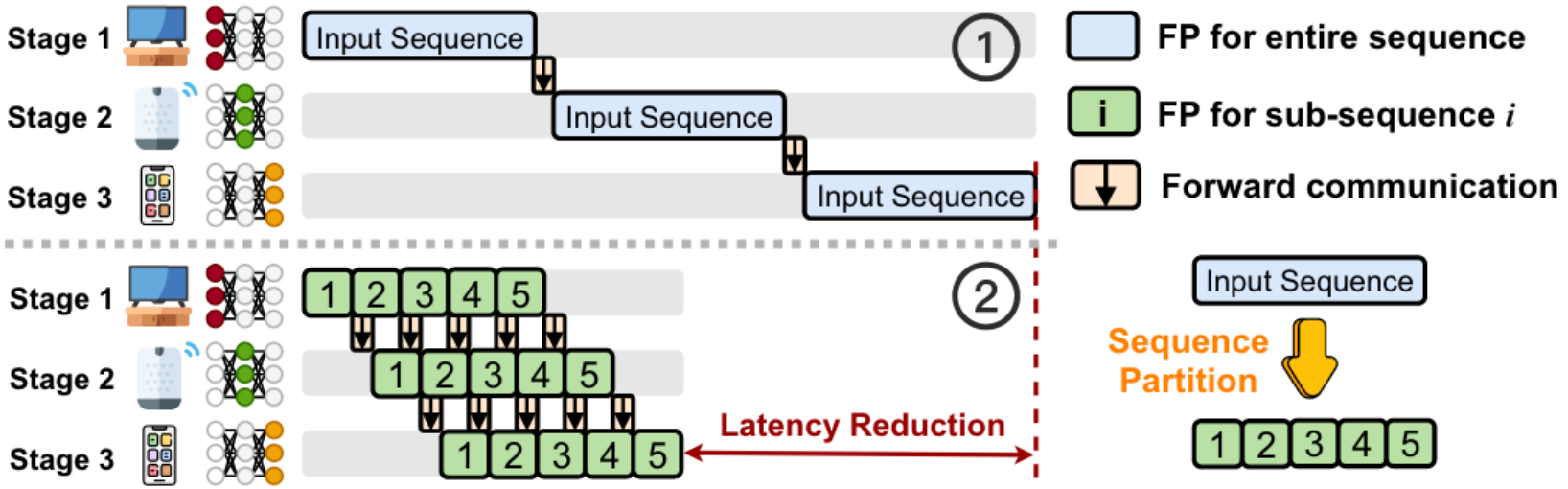



Fig. 5. An illustration of pipelined inference with three edge devices.

# Collaborative Inference for Decoding Phase

- **Solution to Challenge #2**: Integrate speculative decoding with pipeline architecture

**Observation:** Compared to speculative decoding with small-large model collaboration, self-speculative decoding is better suited for distributed pipelined deployment.

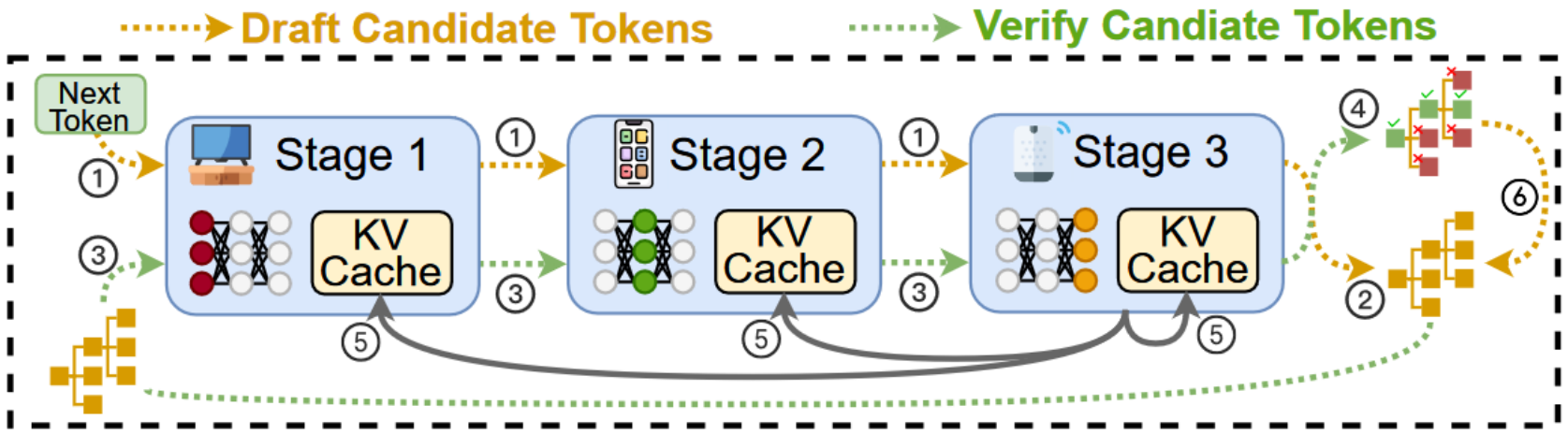



Fig. 8. A workflow of our collaborative inference with speculative decoding.

# Collaborative Inference for Decoding Phase

- **Solution to Challenge #2**: Outline-Based Pipeline Parallel Decoding

**Observation:**

- LLMs tend to generate structured, *point-by-point responses*.
- The generation of each individual point is often ***independent*** *and lacks contextual coherence*.

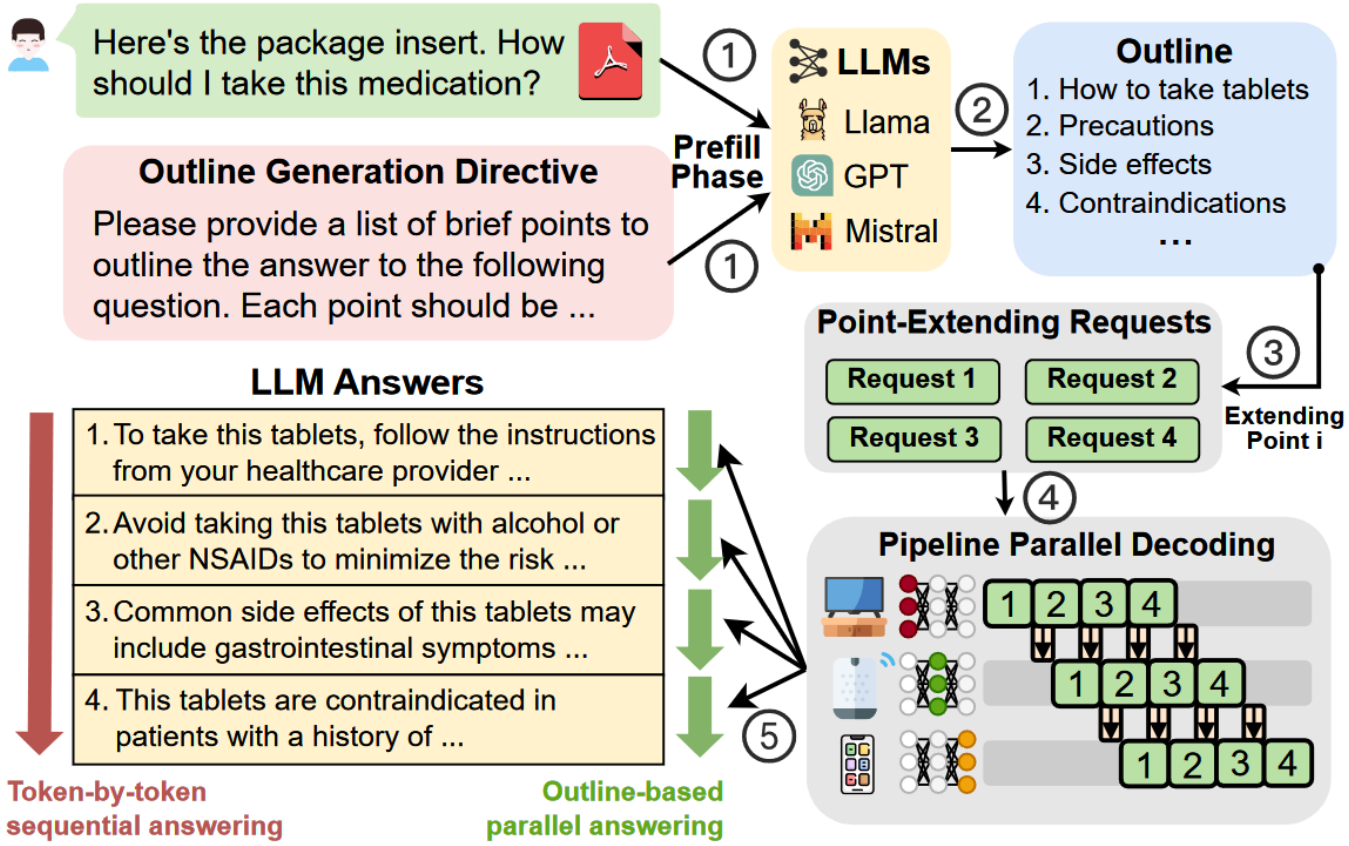
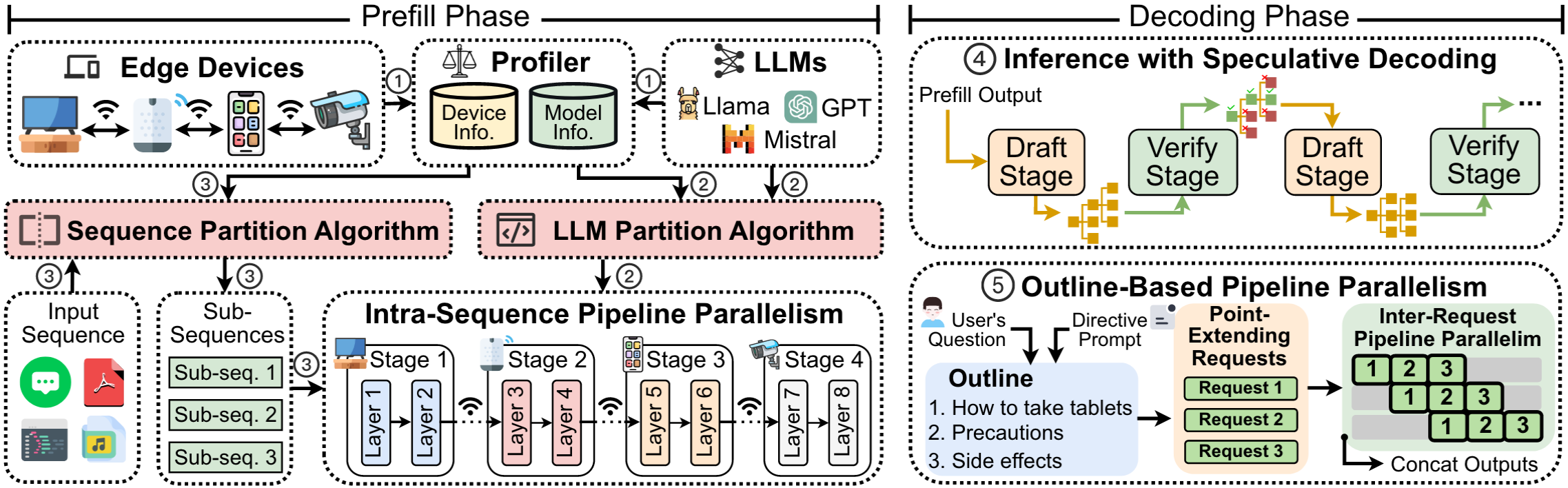



Fig. 9. An illustration of our outline-based pipeline parallel decoding.

# Putting It All Together

- **Jupiter system workflow.**

**Jupiter** uses a pipelined architecture to significantly reduce communication overhead. It optimizes not only the prefill phase, but also the decoding phase.



# Evaluation

- **Testbeds**
  - Using these **3 heterogeneous devices**, we simulated **2 different edge clusters**, including both homogeneous and heterogeneous clusters.

TABLE III
SPECIFICATIONS OF EDGE DEVICES IN EXPERIMENTS.

| Edge Device | GPU Processor | Memory | Power |
|---|---|---|---|
| Jetson Xavier NX [16] | 384-core NVIDIA Volta | 8GB | 20W |
| Jetson TX2 [30] | 256-core NVIDIA Pascal | 8GB | 20W |
| Jetson Nano [31] | 128-core NVIDIA Maxwell | 8GB | 10W |

**Env. A (Homo.):** 4 x NX

**Env. B (Hetero.):** 1 x NX + 2 x TX2 + 1 x Nano

- **Models and datasets**
  - 2 LLMs from the **Llama2 series**, specifically Llama2-7B and Llama2-13B (both with INT4 quantization)
  - Evaluate with **3 recent assistant-style datasets**: LiMA, Vicuna-80, and WizardLM.

# Evaluation

- **Baselines**
  - **Megatron-LM (M-LM):** A state-of-the-art tensor model parallelism method.
  - **Sequence Parallelism (SP):** A state-of-the-art sequence model parallelism method
  - **DeTransformer (DT):** TP-based collaborative edge system that optimizes communications by reducing the frequency of tensor synchronization.
  - **Galaxy:** TP-based collaborative edge system that optimizes communications by fine-grained overlapping of comm. and comp.
  - **EdgeShard (ES):** PP-based collaborative edge inference system that employs pipelined architecture to orchestrate edge devices.

# Evaluation

Jupiter achieves up to **26.1x end-to-end generation latency reduction** compared to baselined system!

Average input sequence length: **260 tokens**

Maximum generation length: **64 tokens**

| Edge Environment | Network Bandwidth | Llama2-7B | | | | | | Llama2-13B | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SP | M-LM | DT | Galaxy | EdgeShard | Jupiter | SP | M-LM | DT | Galaxy | EdgeShard | Jupiter |
| Homo. Env. A | 100Mbps | 53.5 | 431.2 | 228.5 | 427.6 | 42.2 | **16.5** | OOM | 503.4 | 270.1 | 496.5 | 66.2 | **26.3** |
| | 500Mbps | 37.4 | 106.9 | 66.4 | 103.9 | 39.0 | **15.2** | OOM | 130.1 | 83.4 | 125.0 | 63.4 | **25.2** |
| | 1Gbps | 35.4 | 66.4 | 46.1 | 65.0 | 38.6 | **14.9** | OOM | 83.4 | 60.1 | 81.3 | 63.1 | **24.9** |
| Hetero. Env. B | 100Mbps | 63.1 | 491.2 | 288.6 | 458.3 | 59.3 | **22.4** | OOM | 624.5 | 391.2 | 566.4 | 102.4 | **38.8** |
| | 500Mbps | 47.0 | 167.0 | 126.4 | 142.9 | 56.1 | **21.4** | OOM | 251.2 | 204.5 | 208.0 | 99.7 | **37.3** |
| | 1Gbps | 44.8 | 126.4 | 106.2 | 104.9 | 55.7 | **20.9** | OOM | 204.5 | 181.2 | 165.7 | 98.3 | **36.8** |

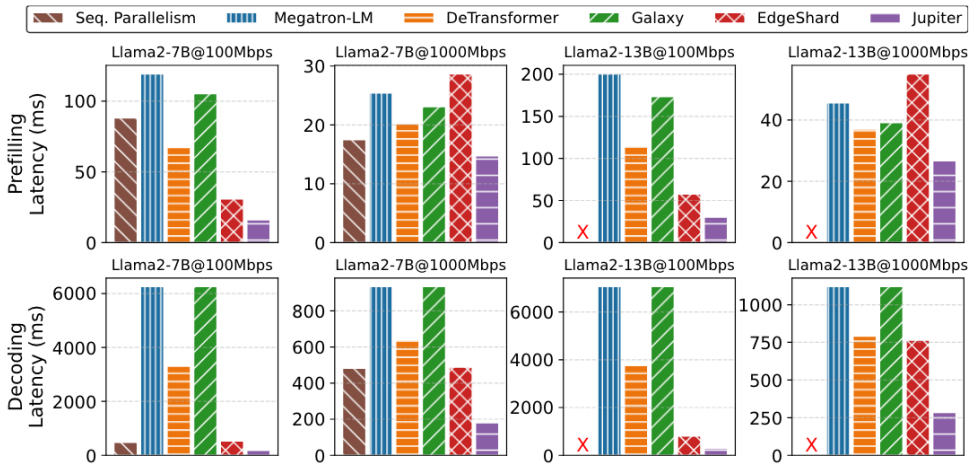



Fig. 10. Evaluate in Homogeneous Environment A. The average per-token processing/generation latency in prefill/decoding phase. × indicates OOM.

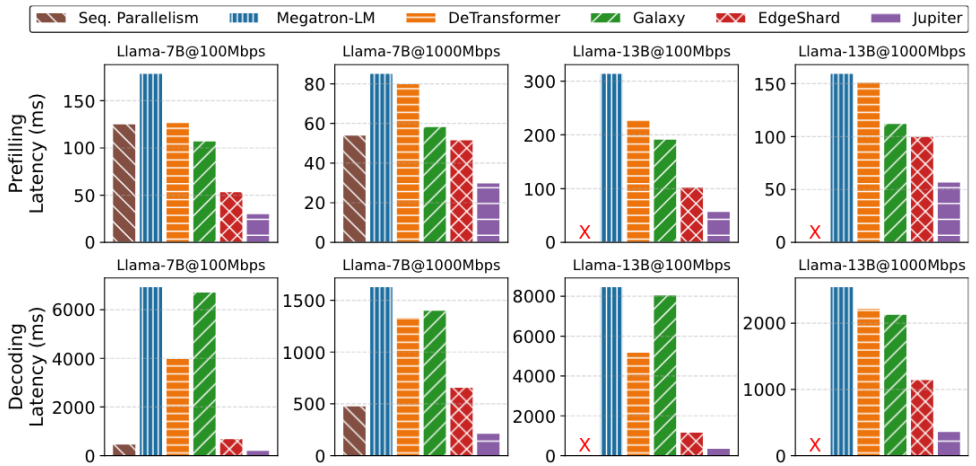



Fig. 11. Evaluate in Heterogeneous Environment B. The average per-token processing/generation latency in prefill/decoding phase. × indicates OOM.

# Evaluation

The high comm.-to-comp. ratio of baselines hinders resource-efficient scaling in bandwidth-limited edge environments. Jupiter outperforms baseline methods in **scalability** across varying bandwidths.

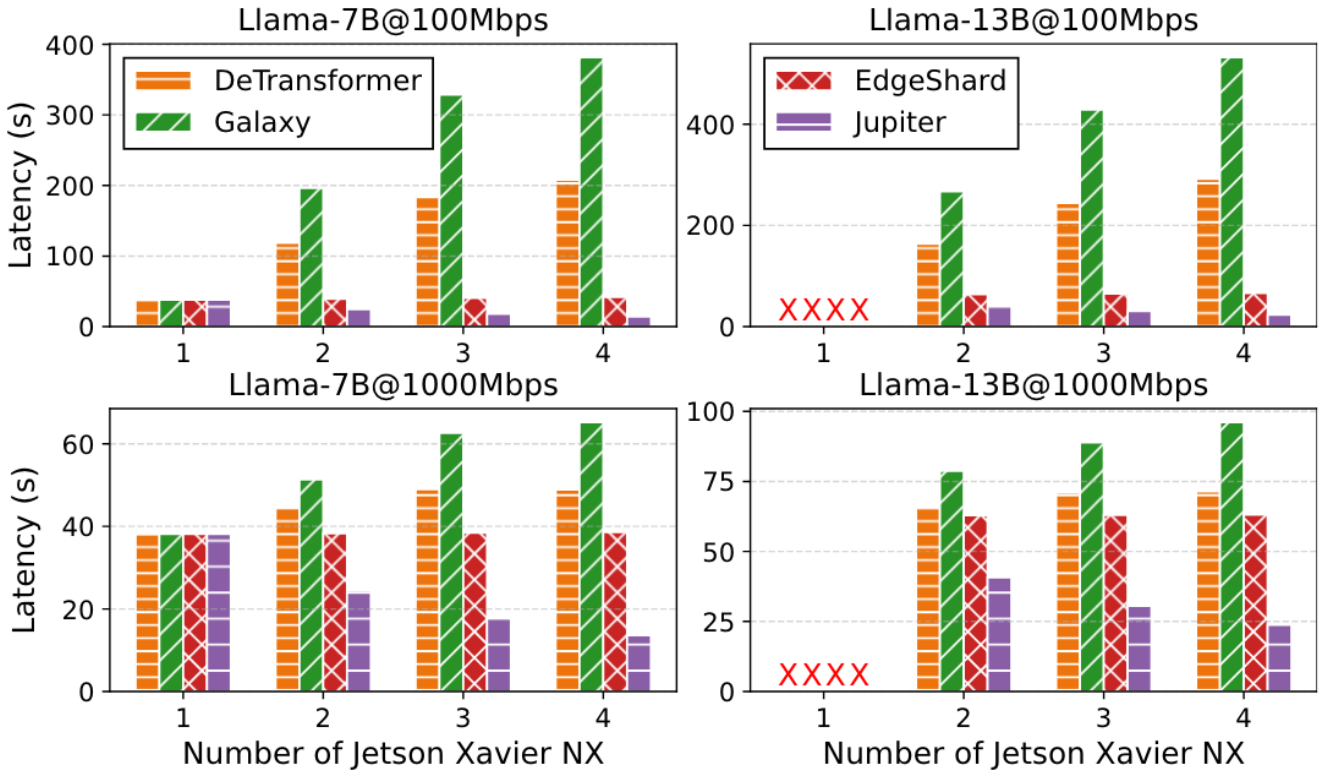



Fig. 12. End-to-end inference latency with a varing number of Jetson Xavier NX under 100Mbps and 1Gbps network bandwidths. × indicates OOM.

# Evaluation

Jupiter achieves up to **3.9x speedup** over naive sequential generation while maintaining comparable generation quality.

TABLE V

SPEEDUP OVER NAIVE SEQUENTIAL GENERATION. SD: SPECULATIVE DECODING. OP: OUTLINE-BASED PARALLEL DECODING.

| Model | Speedup Over Naive | | | |
|---|---|---|---|---|
| | Naive | Jupiter w/o OP | Jupiter w/o SD | **Jupiter** |
| Llama2-7B | 1.0× | 1.8× | 2.3× | 3.6× |
| Llama2-13B | 1.0× | 2.0× | 2.4× | 3.9× |

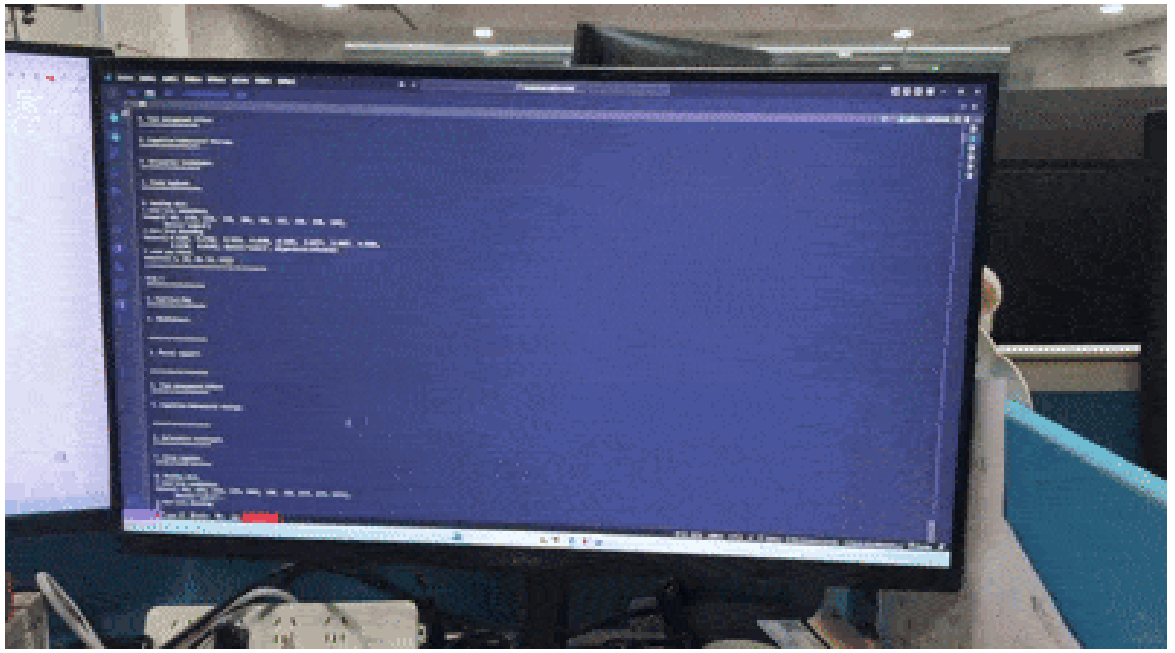
TABLE VI

OVERALL ANSWERS QUALITY OF NAIVE AND JUPITER'S METHOD.

| Method | Vicuna-80 | | WizardLM | | LiMA | |
|---|---|---|---|---|---|---|
| | Llama2-7B | Llama2-13B | Llama2-7B | Llama2-13B | Llama2-7B | Llama2-13B |
| Naive | 7.05 | 7.19 | 6.26 | 6.85 | 6.72 | 6.77 |
| Jupiter | 6.59 | 6.85 | 6.21 | 6.70 | 6.20 | 6.25 |

# Jupiter Demo

Run collaborative inference of the Llama2-7B model using four NVIDIA Jetson edge devices.



IEEE INFOCOM
IEEE International Conference on Computer Communications
19–22 May 2025 // London, United Kingdom

# Thanks for listening

**Shengyuan Ye**[1], Bei Ouyang[1], Liekang Zeng[2], Tianyi Qian[1], Xiaowen Chu[3], Jian Tang[4], Xu Chen[1]

[1] Sun Yat-sen University
[2] The Chinese University of Hong Kong
[3] The Hong Kong University of Science and Technology (GZ)
[4] Midea Group