JANUS: Disaggregating Attention and Experts for Scalable MoE Inference

Zhexiang Zhang[1*], Ye Wang[1,2*], Yumiao Zhao[3], Jiayu Xiao[1], Qianjing Yang[1], Xiangyu Wang[2], Jingzhe Jiang[1], Qizhen Weng[2], Ruichuan Chen[4], Shaohuai Shi[5], Adel N. Toosi[6], Yin Chen[2], Minchen Yu[1†]

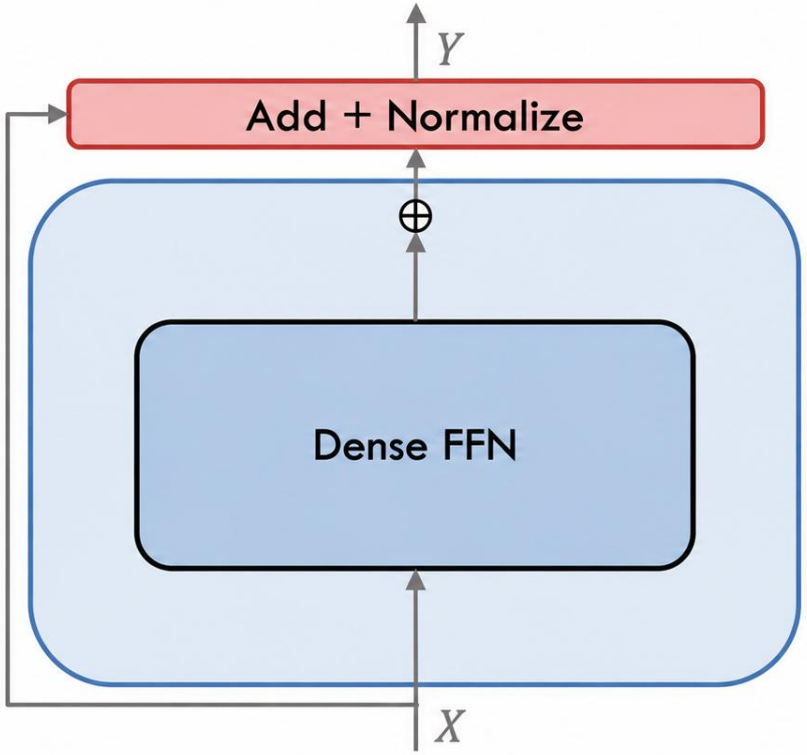
[1]*The Chinese University of Hong Kong, Shenzhen*
[2]*Institute of Artificial Intelligence (TeleAI), China Telecom* [3]*Shenzhen Loop Area Institute*
[4]*Nokia Bell Labs* [5]*Harbin Institute of Technology, Shenzhen* [6]*University of Melbourne*

Presenter: Chizheng Fang

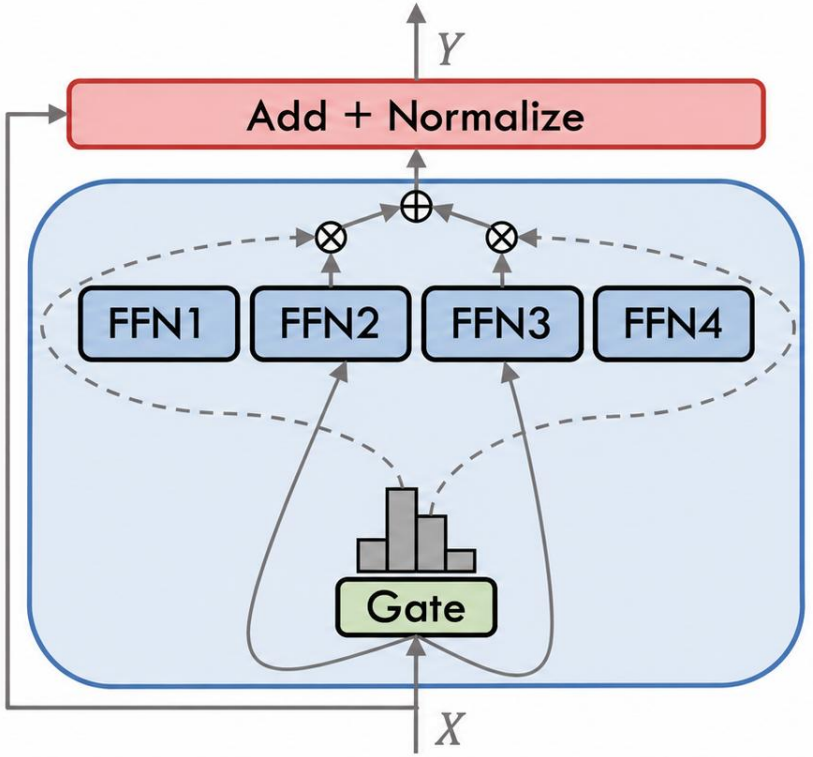
Outline

Background

- MoE models have a sparse expert-based architecture:
  - Activates only a subset of FFN modules (experts)



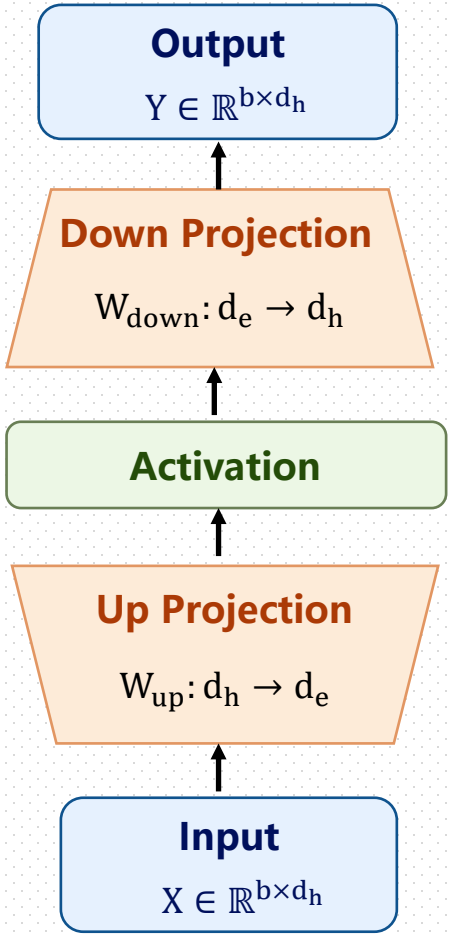
Dense model



MoE model

Observation I: MoE is memory bound

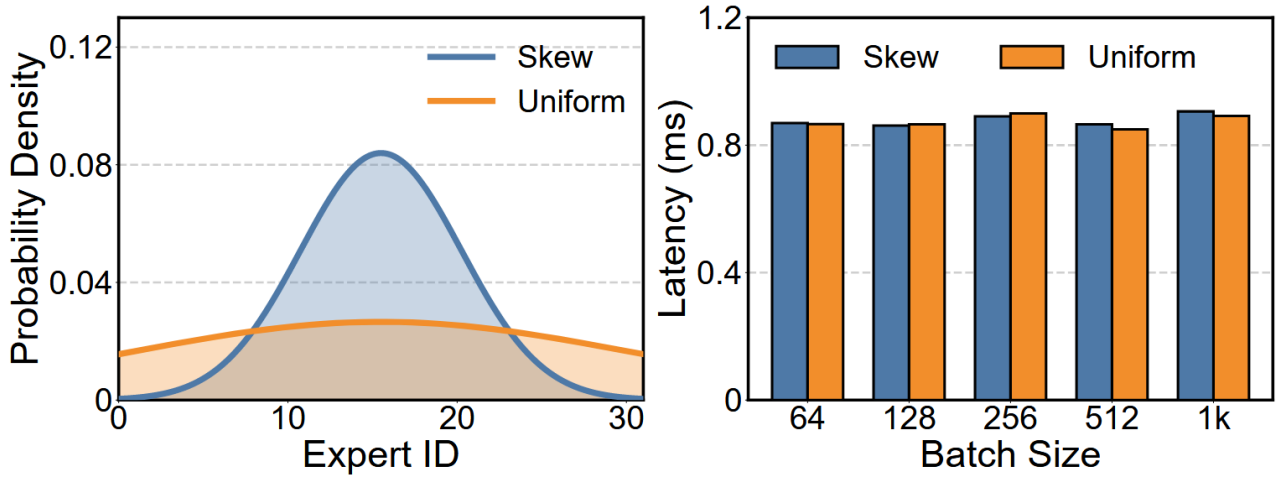
- Decode phase:
- For an expert, $I_e \approx \frac{2bd_hd_e}{2d_ed_h} = b$
  - b: number of routed tokens per expert
  - d_h: hidden dimension
  - d_e: expert intermediate dimension
- Compute bound: $I_e > \frac{\pi}{\beta}$
  - For n experts, top k, bsz=B
    - $b = \frac{Bk}{n} \Rightarrow B > \frac{n\pi}{k\beta}$
  - H100 & DS-V3: π=989 TFLOPS/s, β=3.35TB/s,
    - $B > 18000$



Typical online decode: B << 18000 ➔ Memory-Bound!

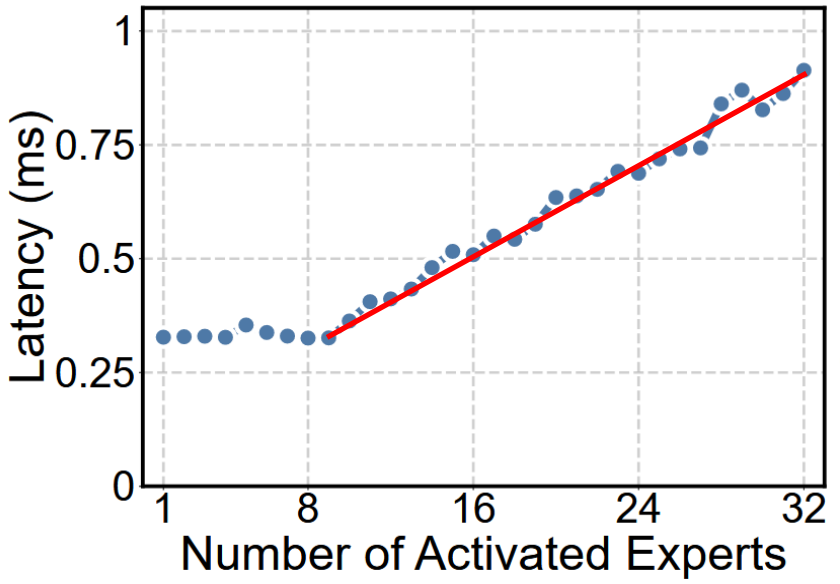
Observation I: MoE is memory bound

- Single MoE layer, 32 experts on one H100
  - Activate all experts following Gaussian-distributed pattern
  - Every expert is selected at least once
- MoE latency is **insensitive** to batch size and activation distribution.



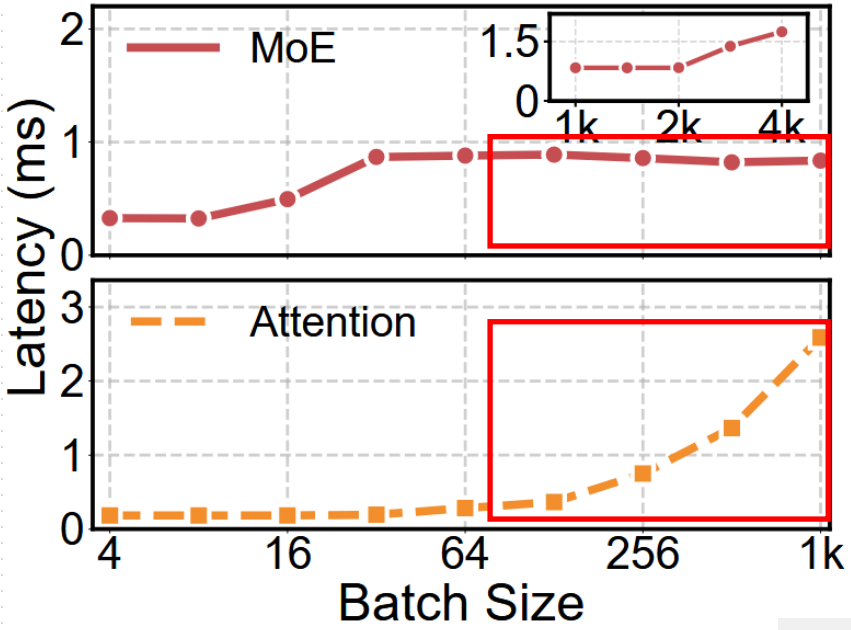
Observation II: Activated experts dictate latency

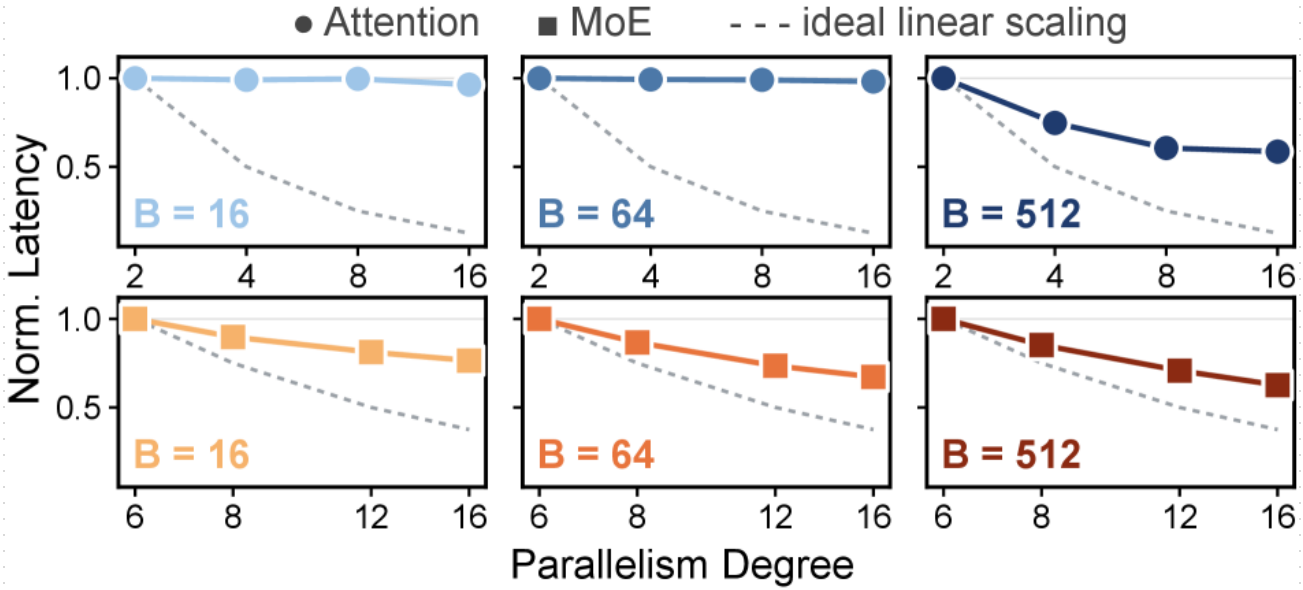
- MoE latency is **insensitive** to batch size and activation distribution.
- The **number of activated experts** dictates latency



Observation III: Divergent scaling patterns

- Attention and MoE favor different batch and parallelism regimes.
  - Attention becomes latency-sensitive at large decode batches
  - MoE benefits more consistently from higher expert parallelism





One scaling policy **cannot** fit both layers.

Limitations of Monolithic Methods

❑Resource Inefficiency

- ❖**Memory Bound MoE**: Hosting massive expert parameters(>90%) requires high degree of Expert Parallelism
- ❖**Homogeneous Configuration**: Identical parallelism strategies (e.g., DP, EP) across heterogeneous Attention and MoE layers
- ❖**Over-provisioning**: Scaling GPUs to satisfy MoE layers, wasting compute resources for Attention layers.

❑Inflexible Scaling

- ❖Scales only at the "model-instance" level
- ❖Incur high overhead, hard to elastically scale in dynamic workloads

Model	Expert Mem. (GB)	Total Mem. (GB)	Ratio(%)
Qwen3-235B	423.0	438	96.5
DS-V2	421.0	472	89.2
DS-V3/R1	1258.0	1342	93.7
Grok-1	586.0	628	91.7

Limitations of Monolithic Methods

❑Resource Inefficiency

- ❖**Memory Bound MoE**: Hosting massive expert parameters(>90%) requires high degree of Expert Parallelism
- ❖**Homogeneous Configuration**: Identical parallelism strategies (e.g., DP, EP) across heterogeneous Attention and MoE layers
- ❖**Over-provisioning**: Scaling GPUs to satisfy MoE layers, wasting compute resources

Attention and MoE should be **disaggregated**!

❑Inflexible Scaling

- ❖Scales only at the "model-instance" level
- ❖Incur high overhead, hard to elastically scale in dynamic workloads

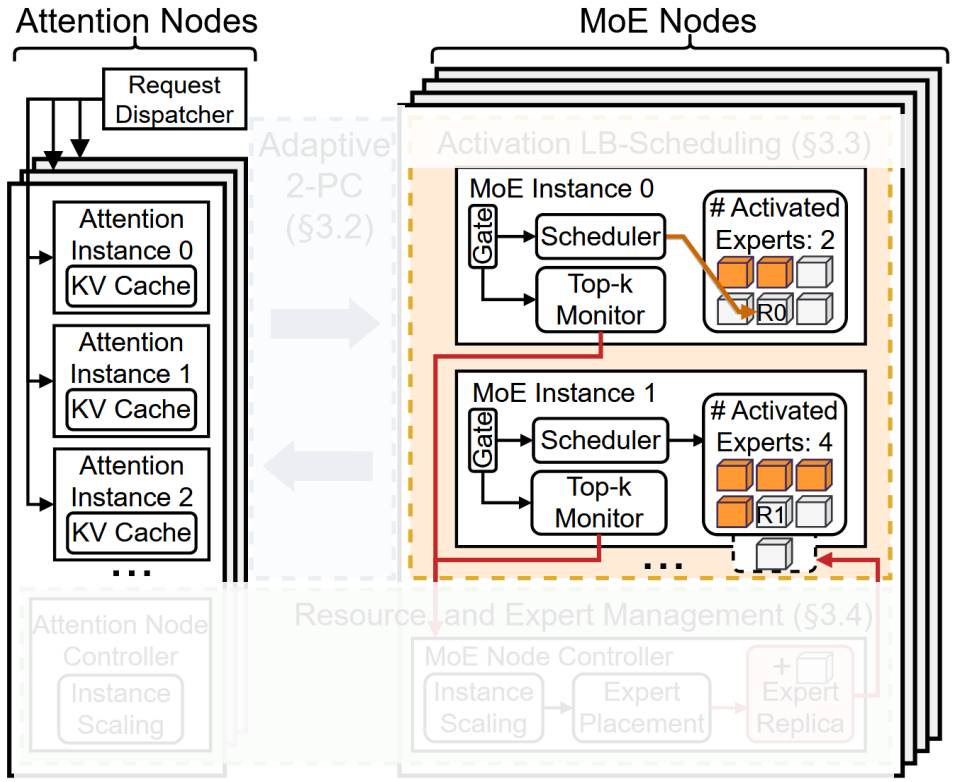
Model	Expert Mem. (GB)	Total Mem. (GB)	Ratio(%)
Qwen3-235B	423.0	438	96.5
DS-V2	421.0	472	89.2
DS-V3/R1	1258.0	1342	93.7
Grok-1	586.0	628	91.7

Outline

- Background
- **Design**
- Evaluation
- Discussion

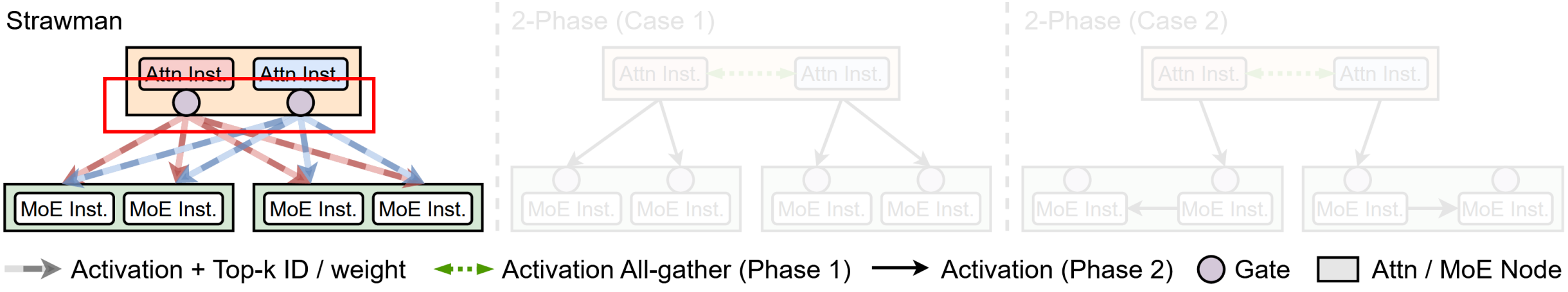
Design

- Key insights:
  - Attention and MoE should be disaggregated
  - Balancing activated experts across GPUs is critical
- Challenges:
  - **Communication Overhead**: Frequent m-to-n activation exchange at every layer
  - **Microsecond-scale Scheduling**: Expert activation must be balanced quickly (<100μs)
  - **High Resource Efficiency**: Resource scaling and activation-aware expert placement



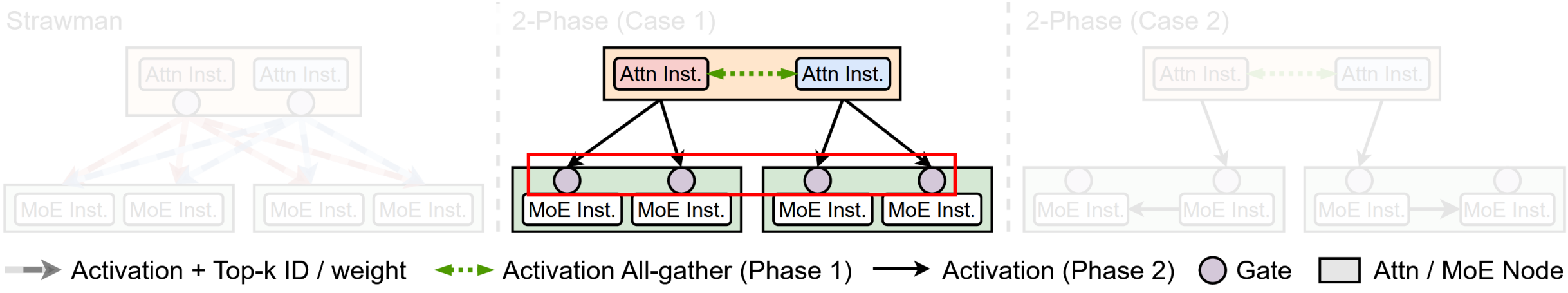
Design: Communication Overhead

- **Strawman Design:** Place the Gate on the Attention side, sending only routed tokens
  - Requires sending extra routing metadata(e.g., top-k expert ID per token)
  - Sending only routed activations requires memory re-layout, introduces extra CPU/GPU overhead
  - More fine-grained transfers



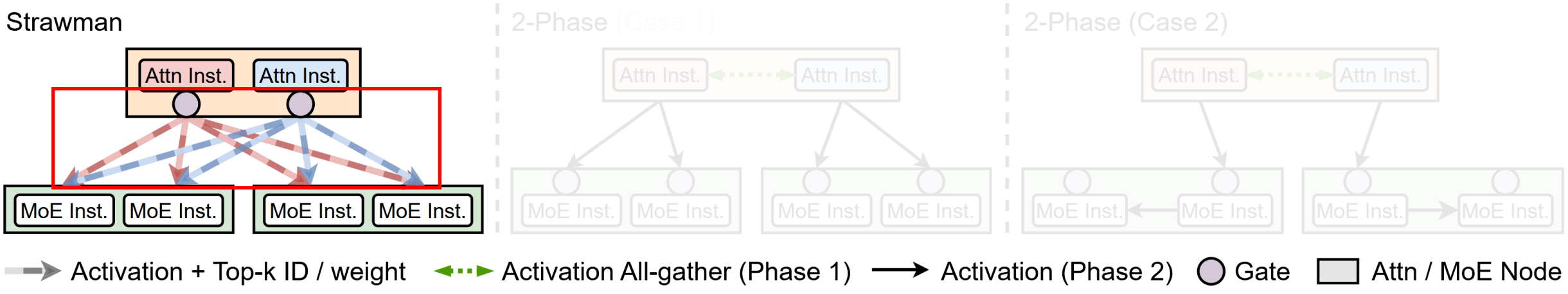
Design: Communication Overhead

- **JANUS Design**: Place the Gate on the MoE side, sending complete activations.
  - Introduce a larger communication volume
  - Simplified communication patterns
  - Reduce the overall number of transfers



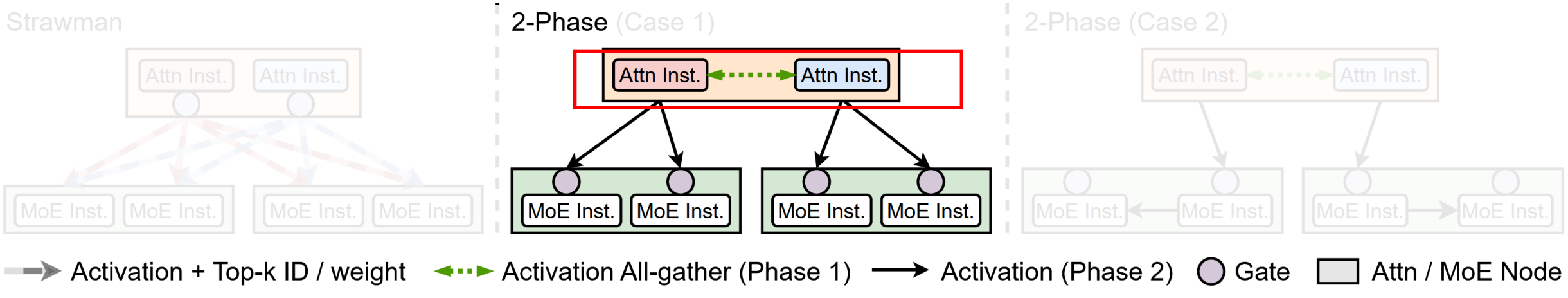
Design: Communication Overhead

❑**Strawman Design**: $O(m \times n)$ Point-to-Point Communication.



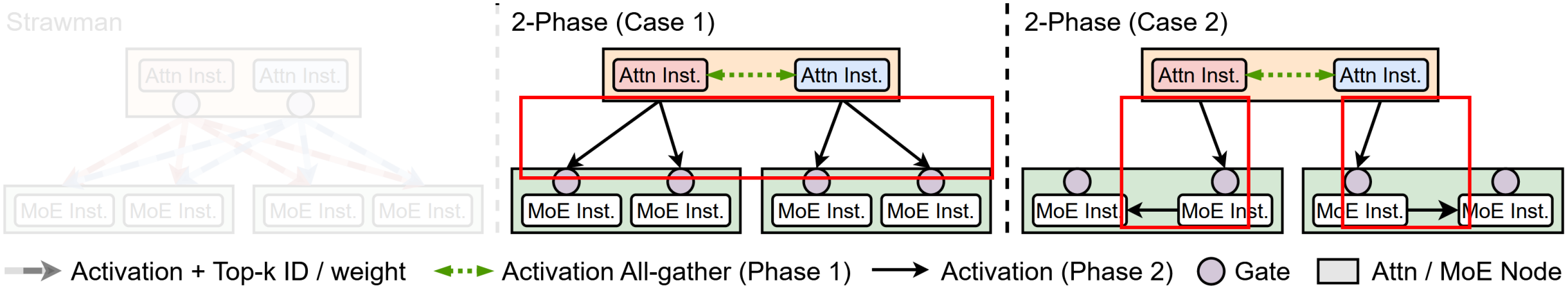
Design: Communication Overhead

- **Strawman Design**: $O(m \times n)$ Point-to-Point Communication.
- **JANUS Design**:
  - Phase 1(intra-node): Use NVLink(All-gather) to aggregate traffic.



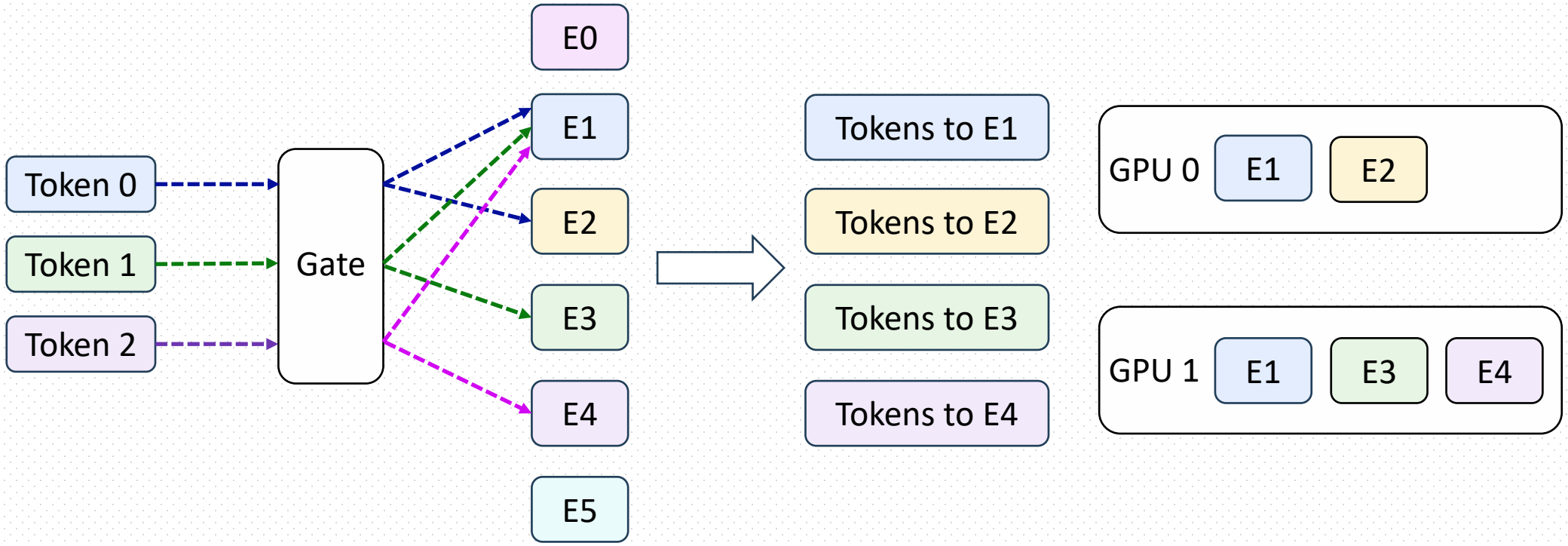
Design: Communication Overhead

- **Strawman Design**: $O(m \times n)$ Point-to-Point Communication.
- **JANUS Design**:
  - Phase 1(intra-node): Use NVLink(All-gather) to aggregate traffic.
  - Phase 2(inter-node):
    - Case 1(Small Destination Set): Directly transmit to target MoE nodes
    - Case 2(Large Volume/Destinations): Distributes to other local instances via NVLink



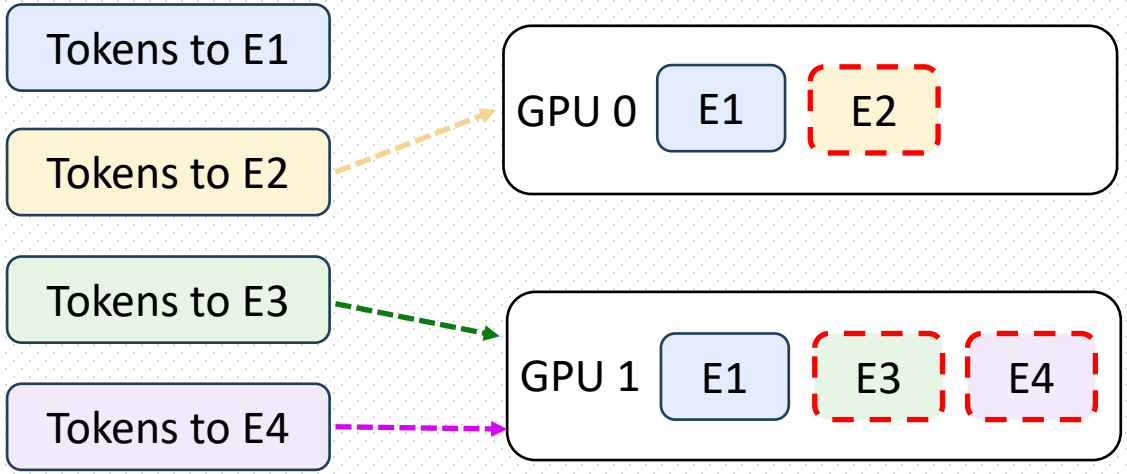
Design: Microsecond-scale Scheduling

- **Goal**: Assign physical expert replicas for each token to balance the number of activated experts across all GPUs.
- Only needs the expert activation set, not exact per-expert batch sizes



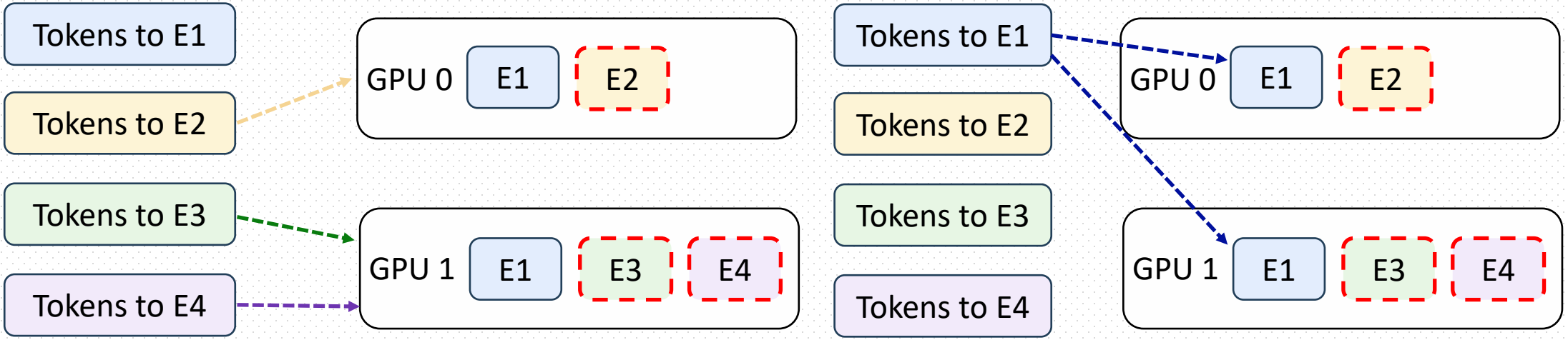
Design: Microsecond-scale Scheduling

- **Goal**: Assign physical expert replicas for each token to balance the number of activated experts across all GPUs.
- First, assign **single-replica** experts



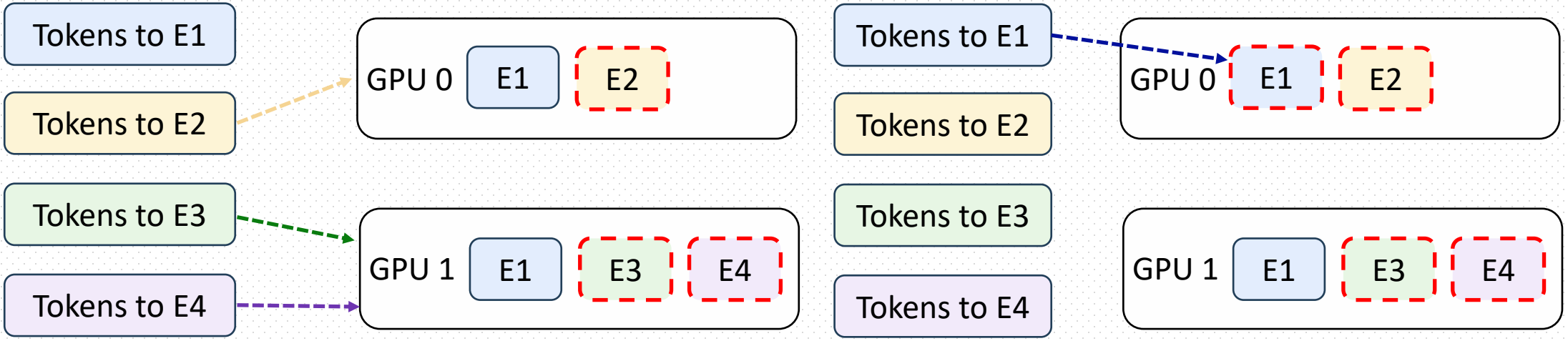
Design: Microsecond-scale Scheduling

- **Goal**: Assign physical expert replicas for each token to balance the number of activated experts across all GPUs.
- For multi-replica experts, **greedily** schedule them to the least-loaded instances



Design: Microsecond-scale Scheduling

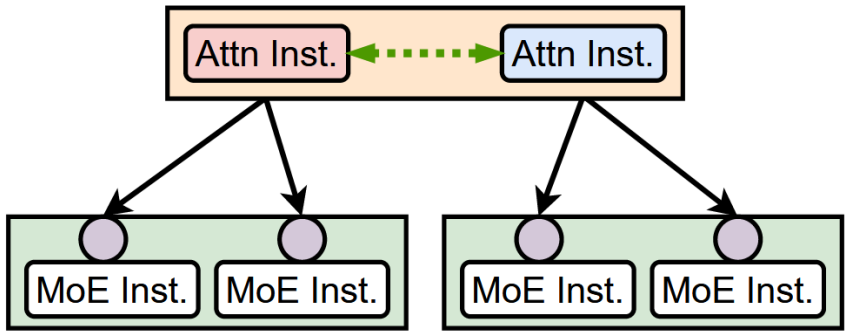
- **Goal**: Assign physical expert replicas for each token to balance the number of activated experts across all GPUs.
- For multi-replica experts, **greedily** schedule them to the least-loaded instances



Design: Microsecond-scale Scheduling

- **Goal**: Assign physical expert replicas for each token to balance the number of activated experts across all GPUs.

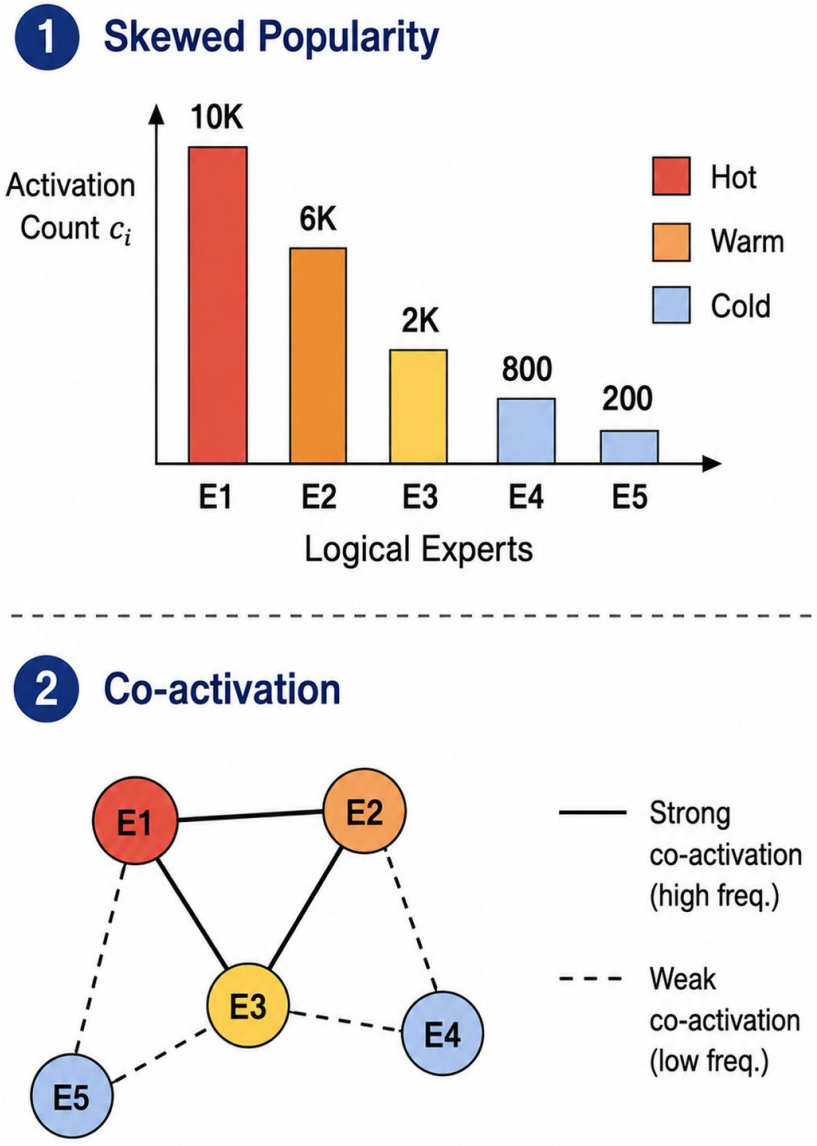
- **How to eliminate global synchronization overhead?**
  - Implementation with GPU Kernel
  - Zero Cross-GPU Coordination:
    - Each MoE instance receive same and complete activations
    - Redundant Computation with deterministic greedy algorithm



Design: Resource & Expert Management

- **Goal:** Optimize expert redundancy and placement over a longer time horizon

- **Observation**: Two patterns
  - **Skewed popularity**: Some experts requested more frequently than others
  - **Co-activation**: Certain pairs of experts activated together by the same tokens



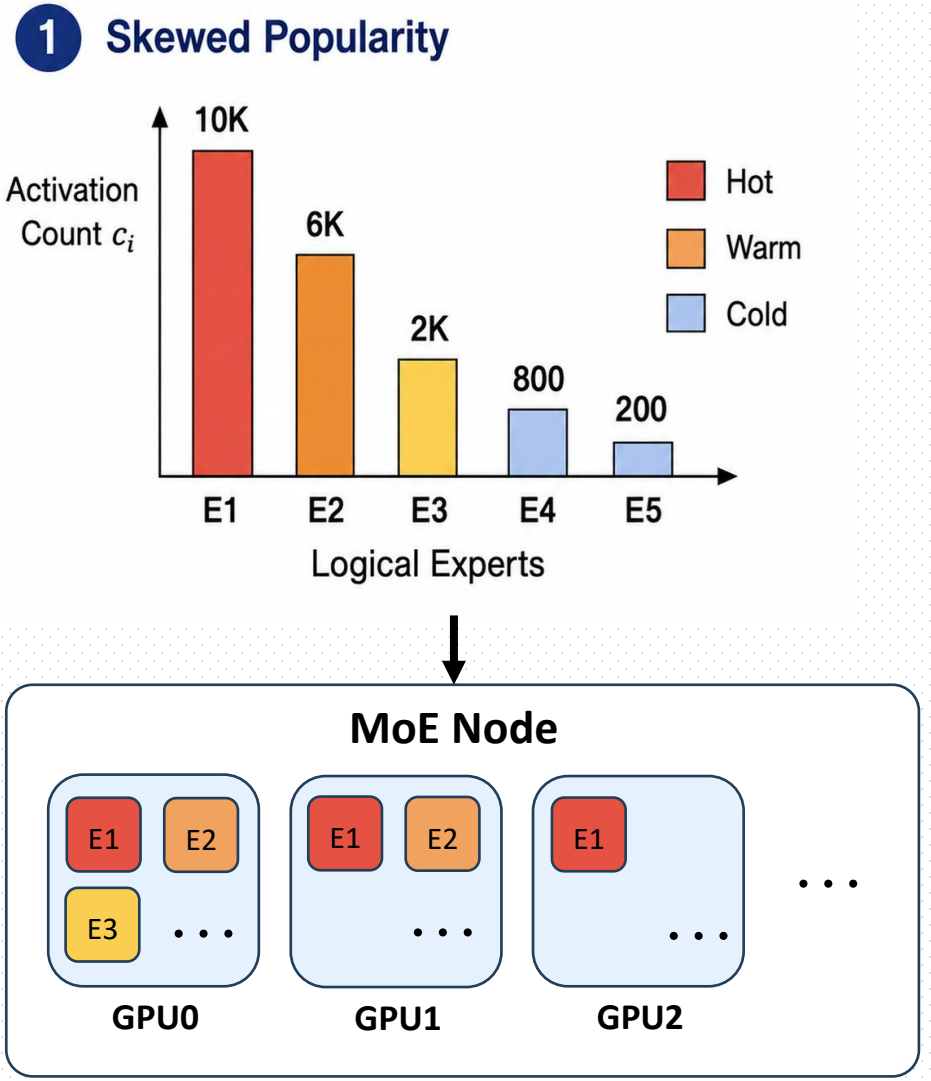
Design: Skewed popularity

- **Goal:** Hot experts get multiple replicas
- **Setup**: N instances, C expert slots/instance
- With sliding window, calculate expert load

$$l_i = \frac{c_i}{r_i}$$

  - c_i: number of activations of expert i
  - r_i: number of replicas of expert i

- Always allocate one additional replica to expert with **largest l_i**, till all $N \times C$ slots are consumed



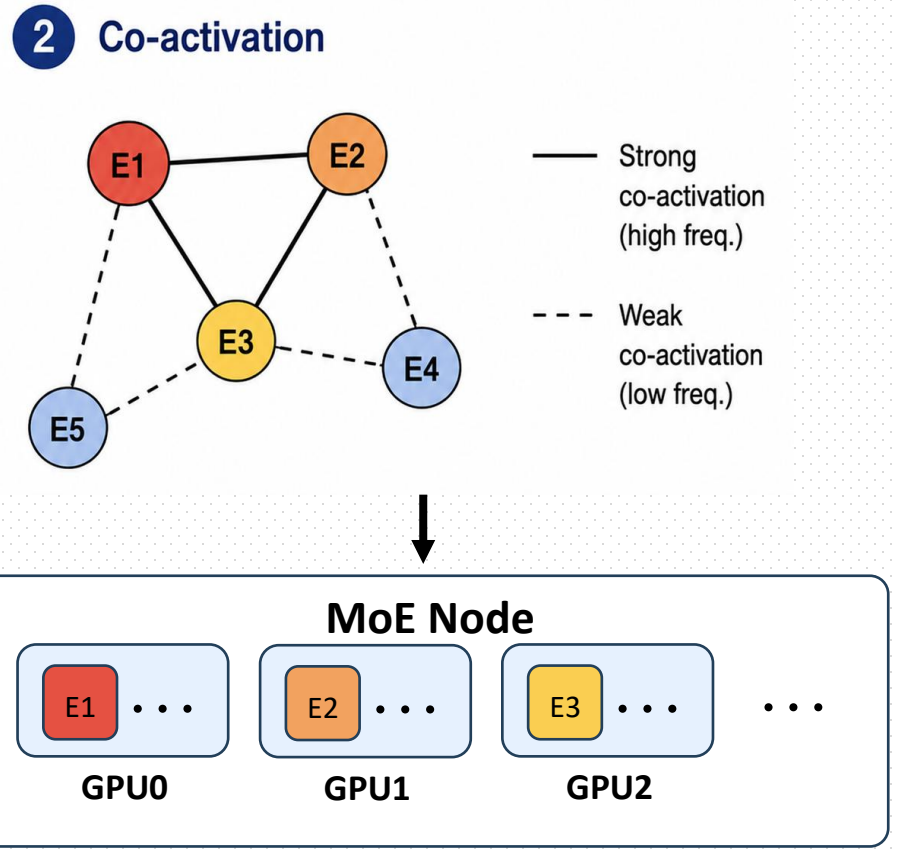
Design: Co-activation

- **Goal:** Placing co-activated experts on separated GPUs.
- With sliding window, for instance g, calculate load

$$I(g) = \sum_{i,j \in P(g)} a_{i,j}$$

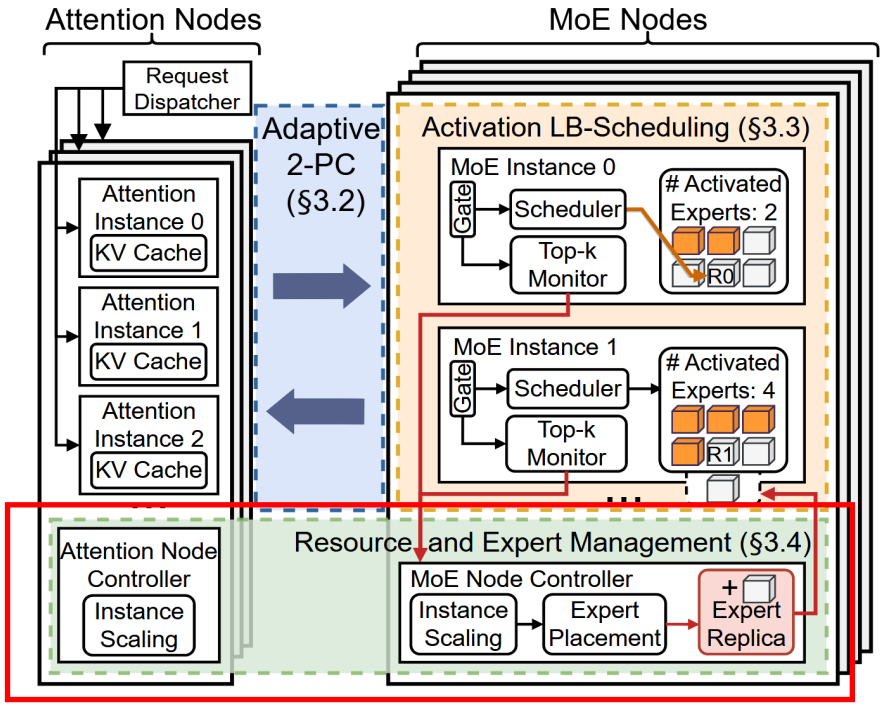
  - $P(g)$: set of replicas on instance g
  - $a_{i,j}$: co-activation frequency

- **Optimization :** $\min \max_{\{g \in \{1,\cdots,N\}} I(g)$
- NP-hard, use a heuristic solution
  - Choose the placement that incurs minimum additional co-activation cost



Design: Resource scaling

- Attention and MoE Controllers **independently** monitor their own queueing delays, latency, and utilization.
- Performance Model Prediction:
  - Adds instances **only** to the bottlenecked sub-cluster if SLO violations approach
  - Scales down safely when utilization drops.



Outline

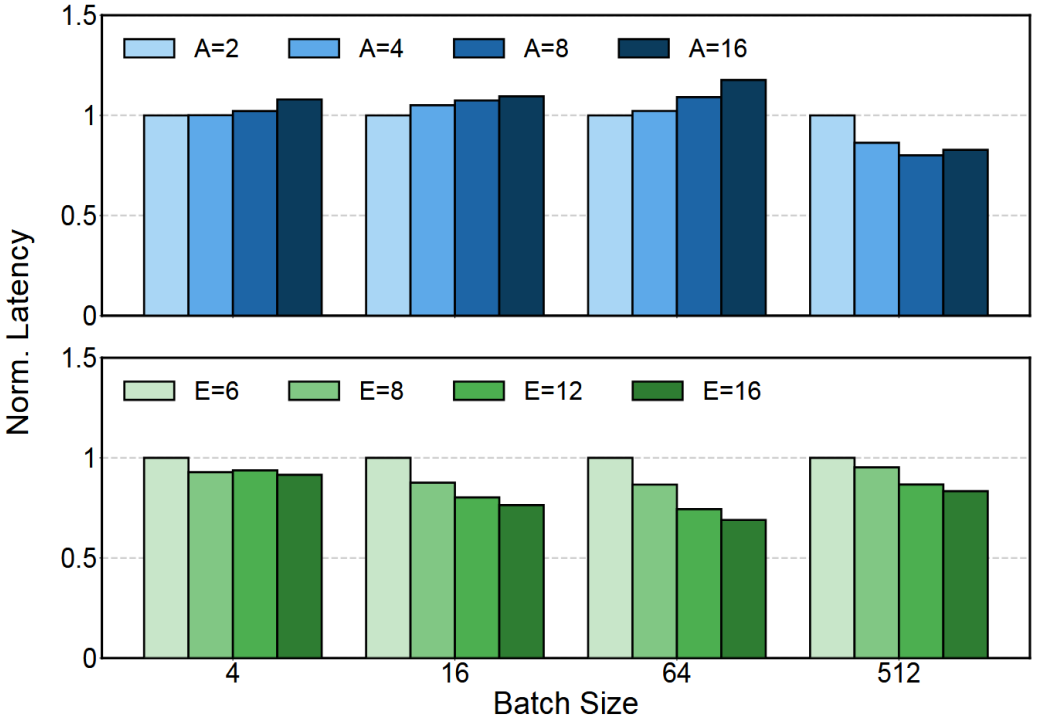
- Background
- Design
- **Evaluation**
- Discussion

Evaluation

- Testbed:
  - 4 nodes with 32 NVIDIA H100 GPUs, 900GB/s NVLink, 400Gbps InfiniBand
- Model
  - Deepseek-V2: 160+2 experts, topk = 6, expert size = 1536
  - Scaled-DS: DS-style variants, modified topk, expert number and expert size.
    - Scaled-DS-1: 160 experts, topk = 8, expert size = 1024
    - Scaled-DS-2: 200 experts, topk = 8, expert size = 1536
- Baseline
  - SGLang: SOTA Monolithic system
  - DisAgg: With random expert scheduling and coarse-grained resource management

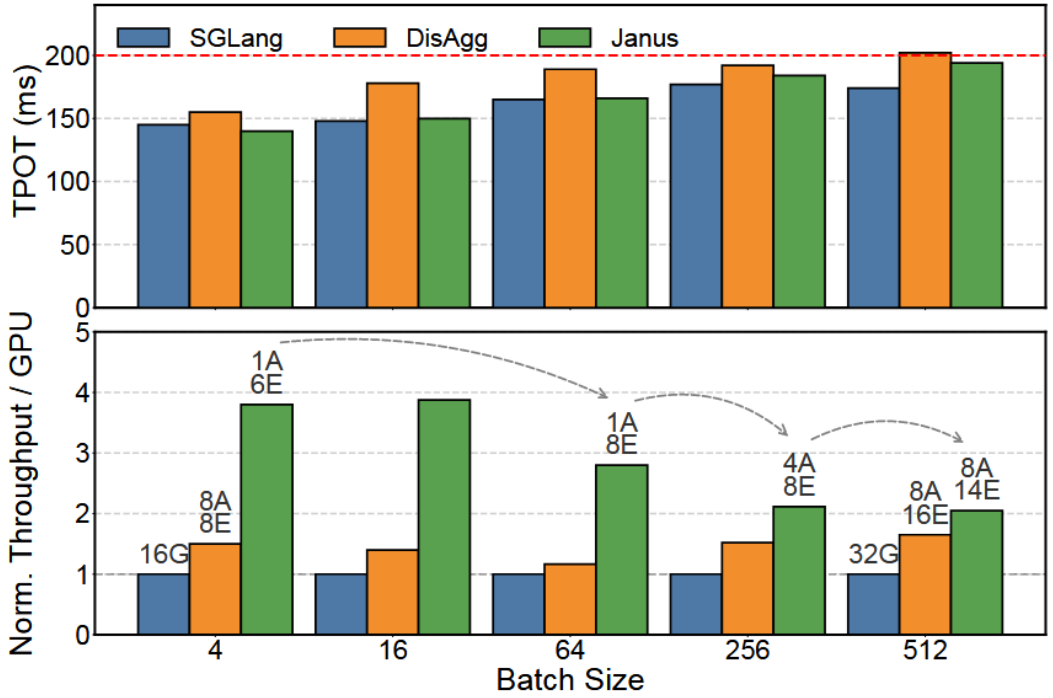
Eval I: Benefits of Disaggregated Architecture

- Attention: Under low/medium load, adding instances increases latency due to communication overhead.
- MoE: Adding instances consistently decreases latency by providing more aggregate memory bandwidth.
- Proof: No single static configuration can optimize both layers simultaneously



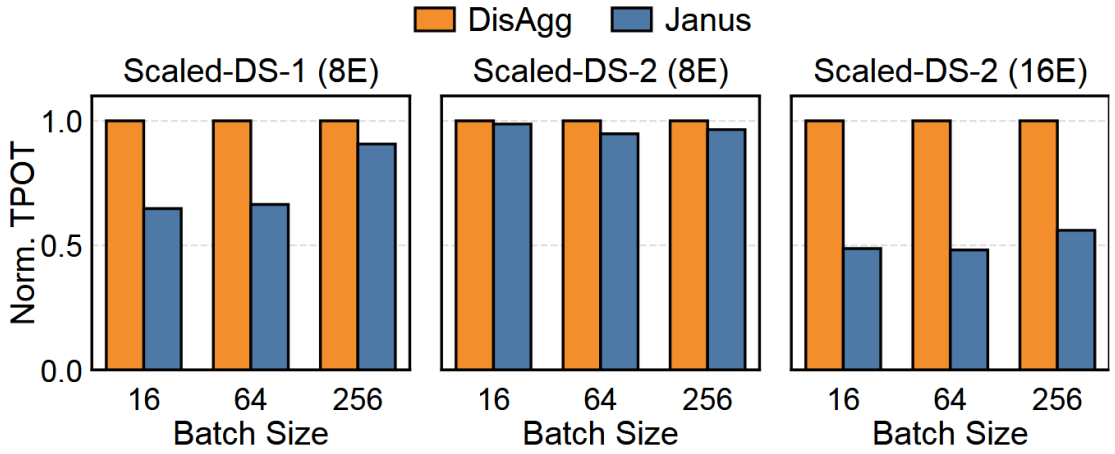
Eval 2: End-to-End Throughput & SLO Attainment

- Test under a 200ms TPOT SLO
- JANUS achieves up to **3.9x** higher per-GPU throughput than SGLang, and **2.8x** higher than DisAgg.



Eval 2: Stress Testing under Expanding Expert Scales

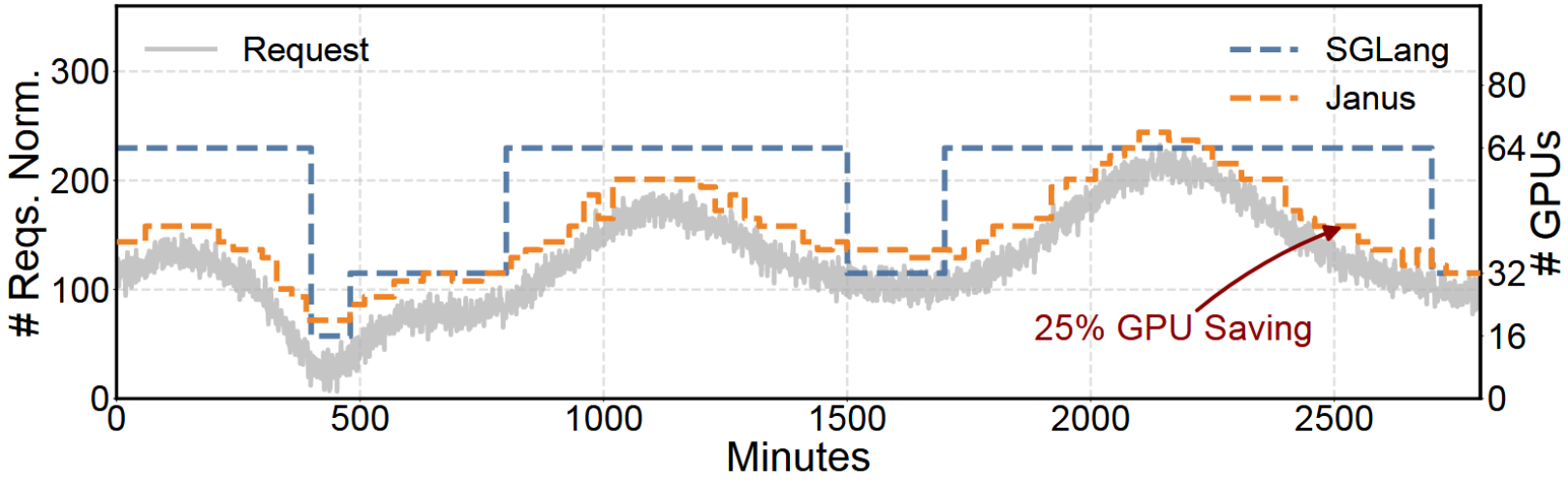
- Does JANUS remain effective in enlarged expert pool case?
- Scaled-DS-1: Memory is sufficient
- Scaled-DS-2:
  - 8 Instances: The scheduler loses flexibility, yielding only modest gains.
  - 16 Instances: Scaling out the cluster restores memory redundancy.



- Scaled-DS-1: 160 experts, topk = 8, expert size = 1024
- Scaled-DS-2: 200 experts, topk = 8, expert size = 1536

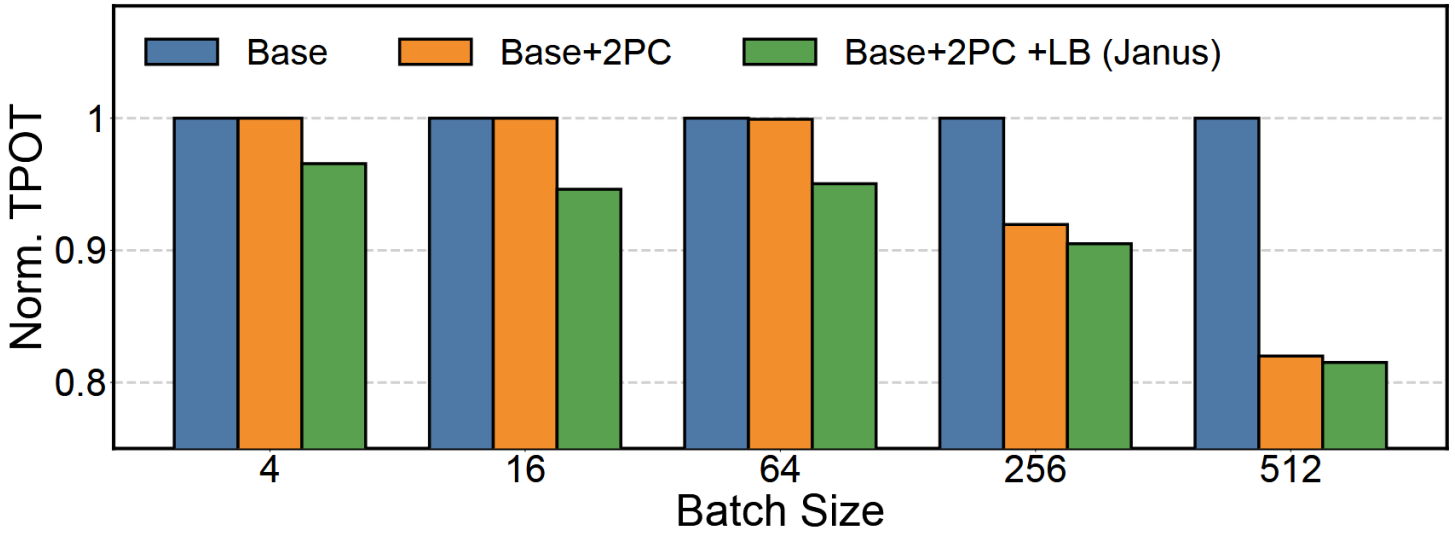
Eval 3: Real-World Dynamic Workloads

- 2-day simulation using real-world fluctuating traffic (BurstGPT)
- **SGLang**: Jumping abruptly between 16, 32, 64 GPUs
  - Massive over-provisioning
- **JANUS**: Smoothly tracks the diurnal traffic waves with fine-grained scaling.
  - Achieves a 25% reduction in overall GPU consumption



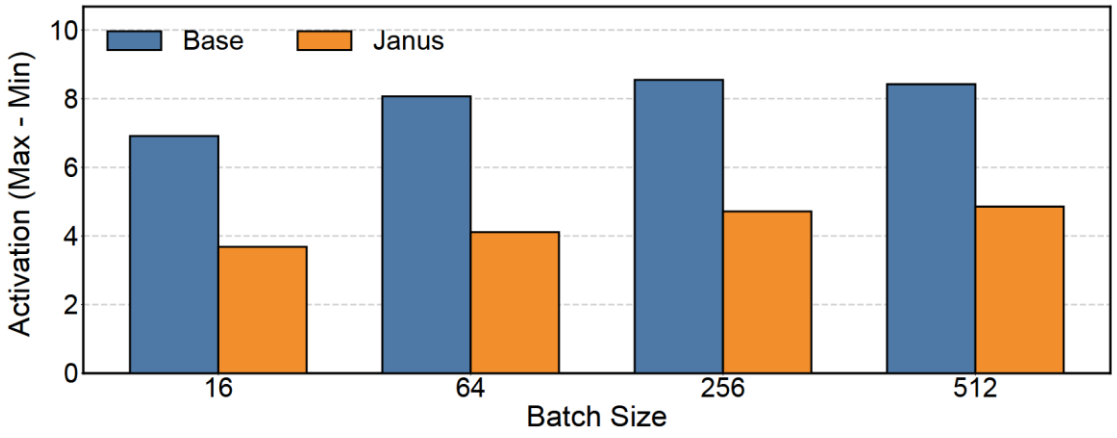
Eval: Ablation Study

- +Base: Disaggregation alone
- +2PC (Two-Phase Comm): At high loads, 2PC slashes TPOT latency by 18%, clearing the network bottleneck.
- +LB (Load Balancing & Placement): At low/medium loads, LB resolves expert imbalance, cutting latency by an additional 7%.

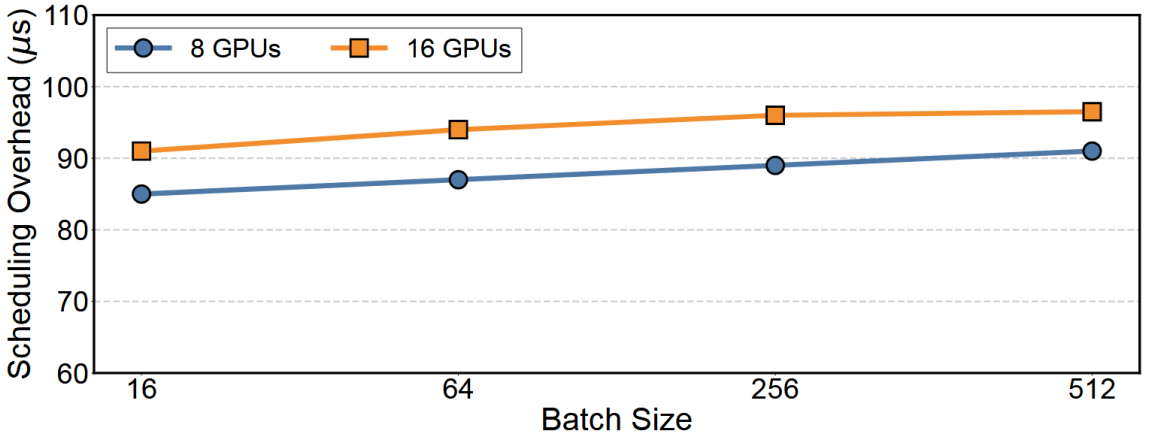


Eval: Scheduling Effectiveness & Overhead

- Reduce the Activation Imbalance from ~8 to ~4
  - (Difference in activated experts between the most and least loaded instances).
- Scheduling Latency: overhead remains **under 100 µs** across all batch sizes, whether on 8 GPUs or 16 GPUs.



Gain: Better Load Balancing



Cost: Minimal Overhead

Outline

- Background
- Design
- Evaluation
- **Discussion**

Discussion

- **Heterogeneous Hardware**: Naturally supports assigning compute-heavy Attention to compute GPUs, and MoE to memory-rich GPUs.
- **Compatibility**: Fully orthogonal to and compatible with other parallelism schemes like Tensor Parallelism.

JANUS: Disaggregating Attention and Experts for Scalable MoE Inference

Thanks for listening !

Presenter: Chizheng Fang