Klotski: Efficient Mixture-of-Expert Inference via Expert-Aware Multi-Batch Pipeline

Zhiyuan Fang, Yuegui Huang, Zicong Hong, Yufeng Lyu, Wuhui Chen, Yue Yu, Fan Yu, Zibin Zheng

Presented by Jiawei Yi @ USTC

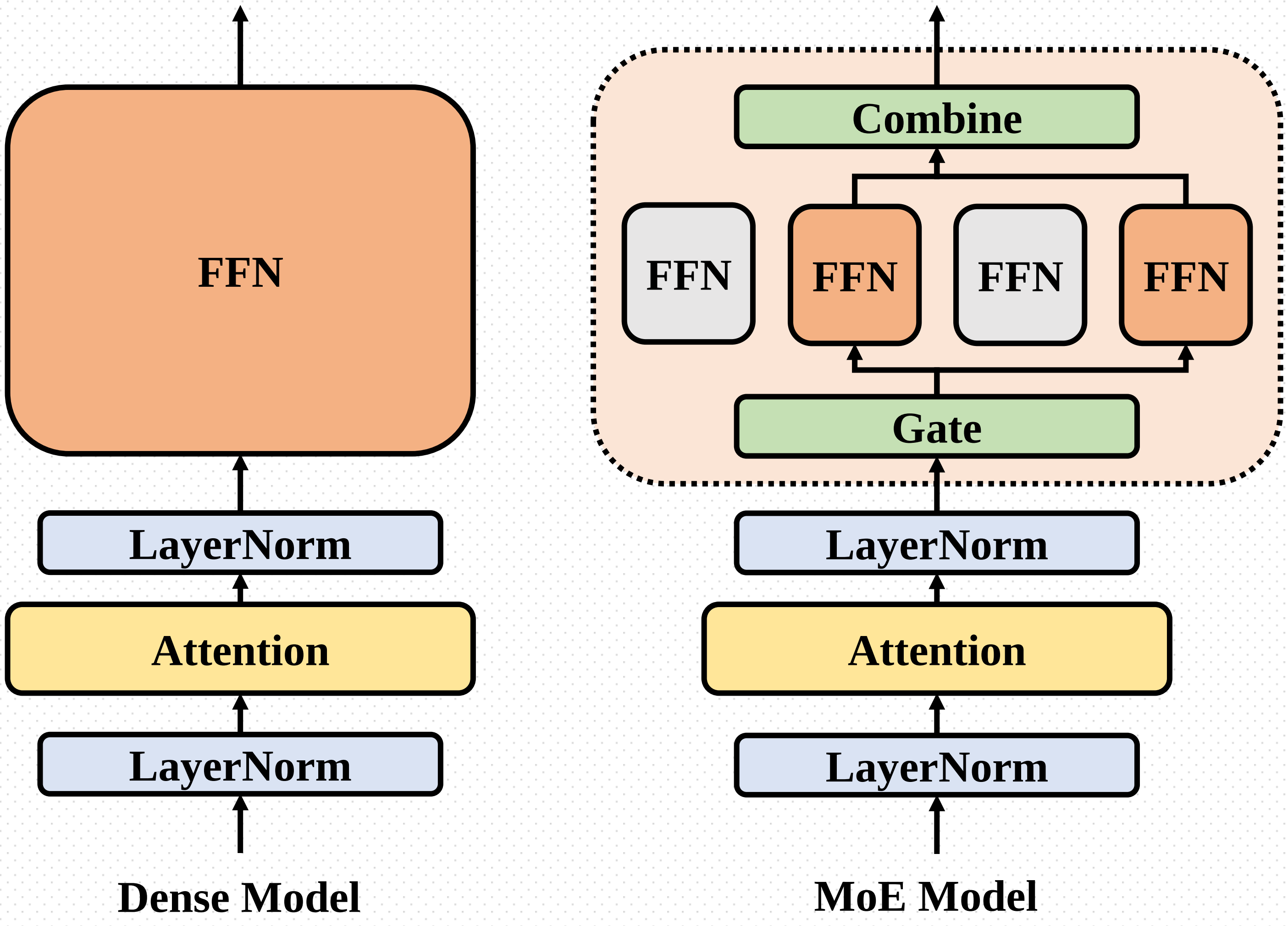
2025-04-08

Outline

- ❑Background
- ❑Motivation
- ❑Klotski
- ❑Evaluation

Background: MoE LLM

❑MoE - Mixture-of-Expert



Decompose the large FFN into smaller experts with selective activation

Scalability: Scale up model size without significantly increasing computational costs

Background: MoE LLM

❑The total size of MoE layers is too large!

DeepSeek-V3/R1		
Hidden Dim.	**Intermediate Dim.**	**Data Type**
7168	2048	Float8
# Experts	**# Activated Experts**	**# MoE Layers**
256	8	60

Parameter size:

- **A single expert:** $\mathbf{7168 * 2048 * 3 = 42\mathit{MB}}$
- **All the experts:** $\mathbf{42\mathit{MB} * 256 * 60 = 630\mathit{GB}}$

Background: MoE LLM

❑The total size of MoE layers is too large!

DeepSeek-V3/R1		
Hidden Dim.	**Intermediate Dim.**	**Data Type**
7168	2048	Float8
# Experts	**# Activated Experts**	**# MoE Layers**
256	8	60

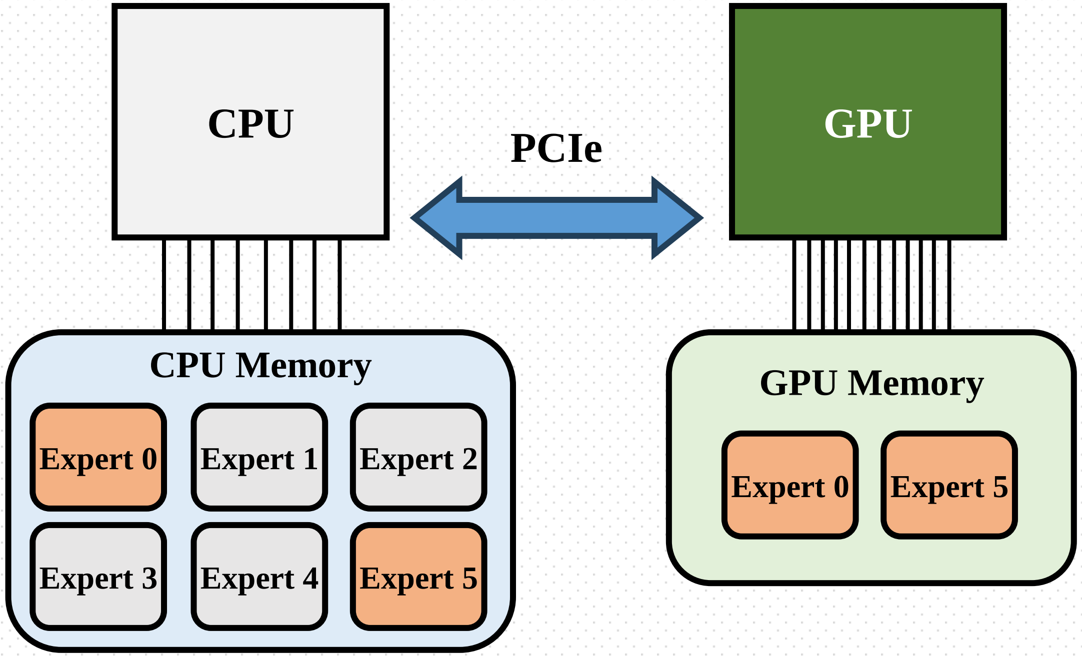
Parameter size:

- **A single expert:** $7168 * 2048 * 3 = 42MB$
- **All the experts:** $42MB * 256 * 60 = 630GB$
- **Activated experts (for one token):** $42MB * 8 * 60 = 19.7GB$

The size of activated model weights is much smaller than the total size

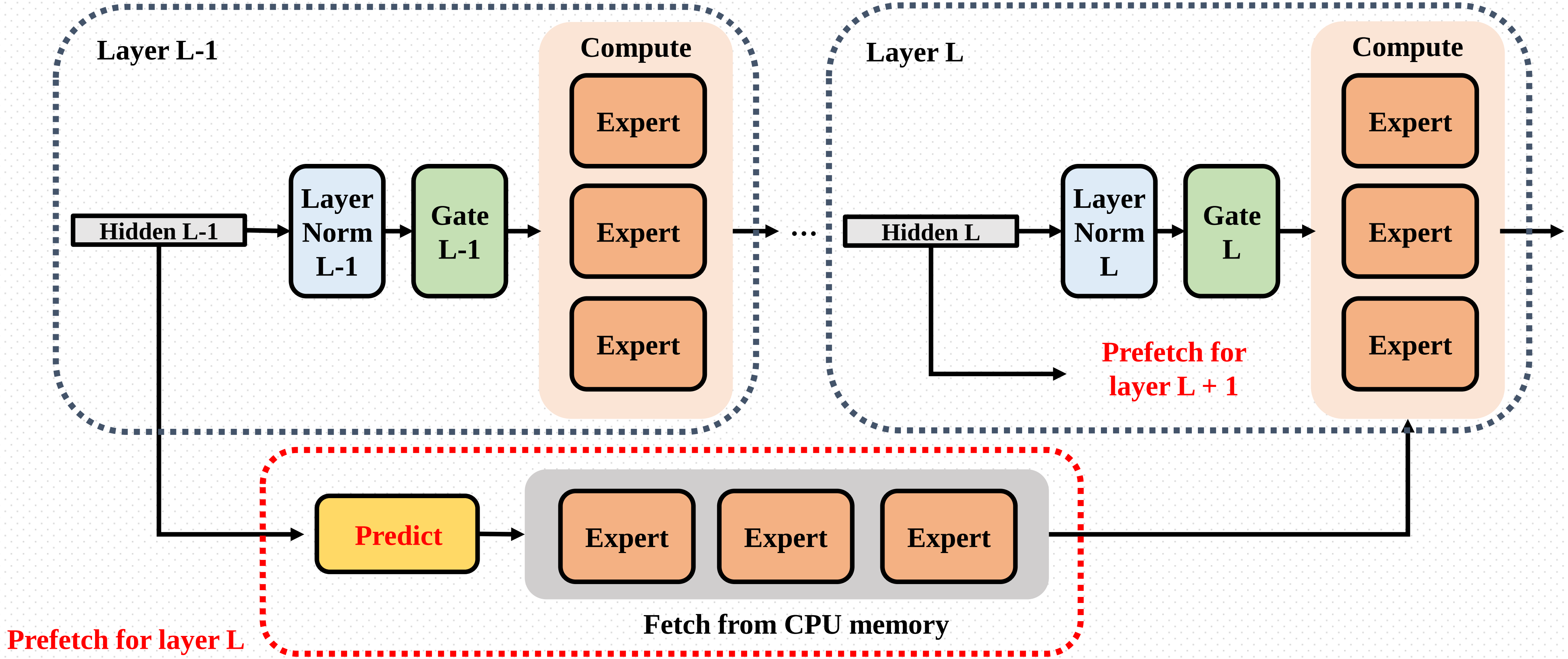
Background: Expert Offloading

- ❑Solution: Expert offloading
  - ❖CPU: hold all the experts
  - ❖GPU: only fetch activated experts
  - ❖Problem: PCIe transmission overhead



Background: Expert Offloading + Prefetching

❑Solution: Expert predicting & prefetching



Background: Expert Offloading + Prefetching

- **Targets of expert prefetching:**
  - **Completely hide the PCIe transmission overhead**
  - **Predict experts accurately**

Background: Expert Offloading + Prefetching

- **Targets of expert prefetching:**
  - **Completely hide the PCIe transmission overhead**
  - **Predict experts accurately**
- **Most existing works focus on the second target:**
  - **ProMoE: Trained predictor**
  - **fMoE: Predict with semantic hints**
  - **……**
  - **However, these works fail to hide transmission overhead due to limited PCIe bandwidth**

[1] ProMoE: Fast MoE-based LLM Serving using Proactive Caching
[2] fMoE: Fine-Grained Expert Offloading for Large Mixture-of-Experts Serving

Background: Expert Offloading + Prefetching

❑PCIe bandwidth cannot meet the overlapping requirements

❖DeepSeek-V2-Lite, SGLang, A40 GPU x1

Batch Size	Context Length	Decode Step Time (ms)	#Activated Experts (Worst)	#Prefetchable Experts
1	8K	14.56	156	22
2	8K	15.36	312	24
4	8K	15.64	624	24
1	16K	18.26	156	28
2	16K	19.12	312	29
1	32K	25.69	156	40

❖Are there any other solutions?

- **Expert Size ~0.016 GB**
- **PCIe Bandwidth ~ 25 GB/sec**

Outline

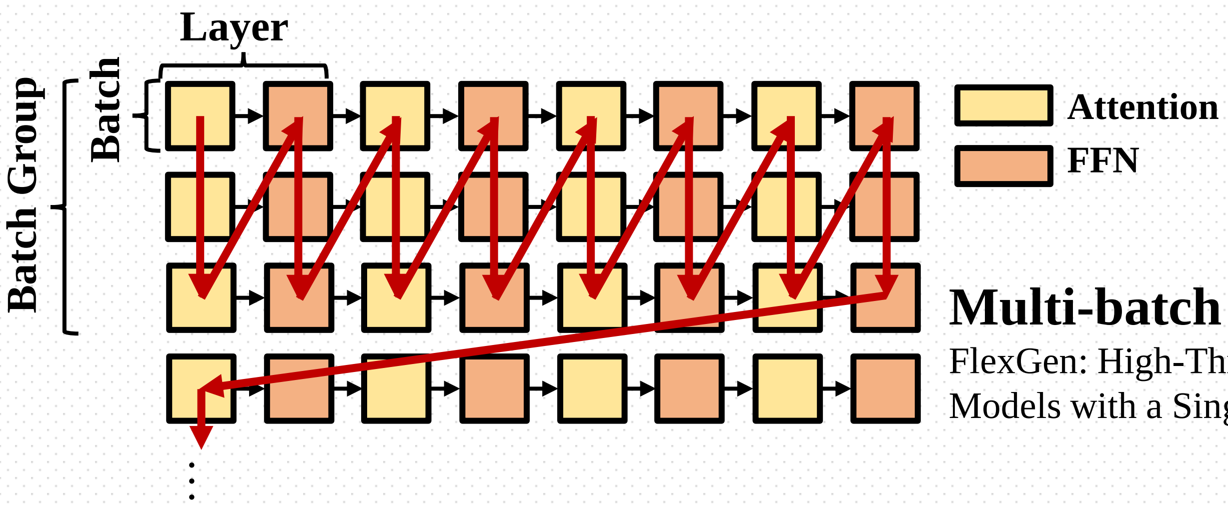
- ❑Background
- ❑**Motivation**
- ❑Klotski
- ❑Evaluation

Motivation

- ❑Target on offloading all the model weights and KVCache
- ❑Target on offline inference, focus on throughput
- ❑Explore the overlapping opportunities through multi-batch pipeline

Motivation

- For dense model offloading, previous work has proposed **multi-batch pipeline** to hide PCIe transmission overhead

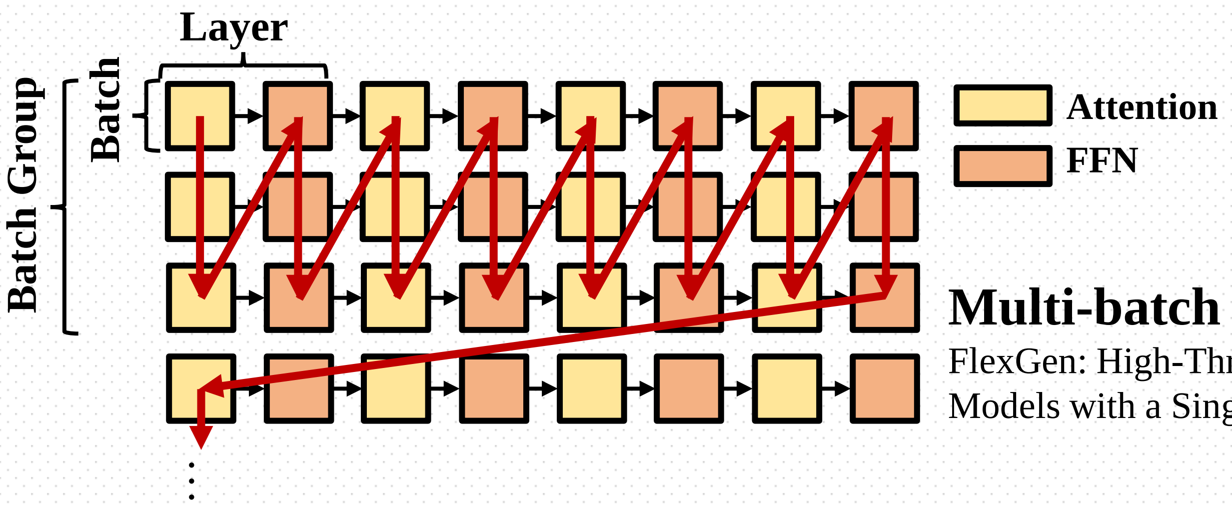


Multi-batch pipeline

FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU, ICML 2023

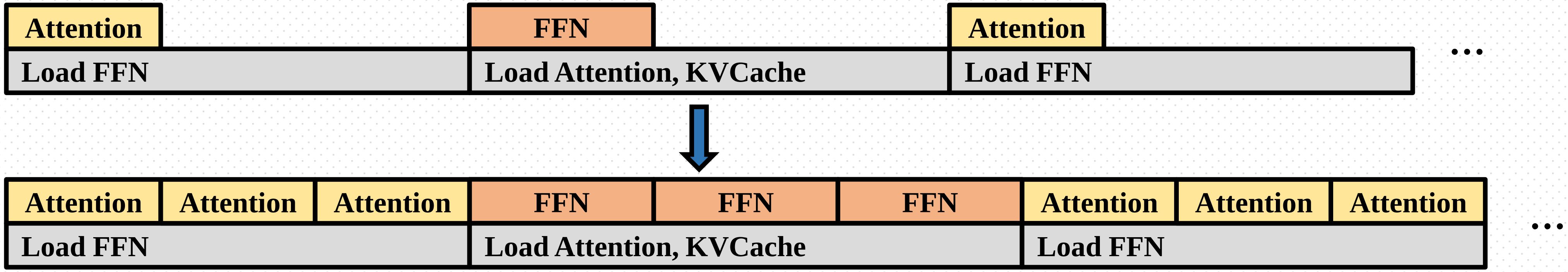
Motivation

❑For dense model offloading, previous work has proposed **multi-batch pipeline** to hide PCIe transmission overhead



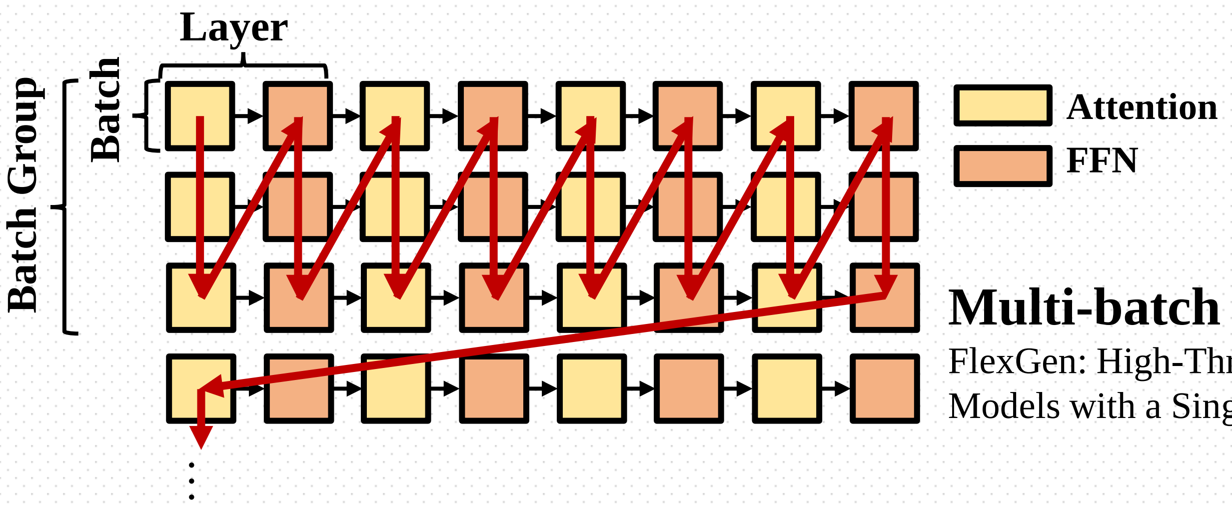
Multi-batch pipeline

FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU, ICML 2023



Motivation

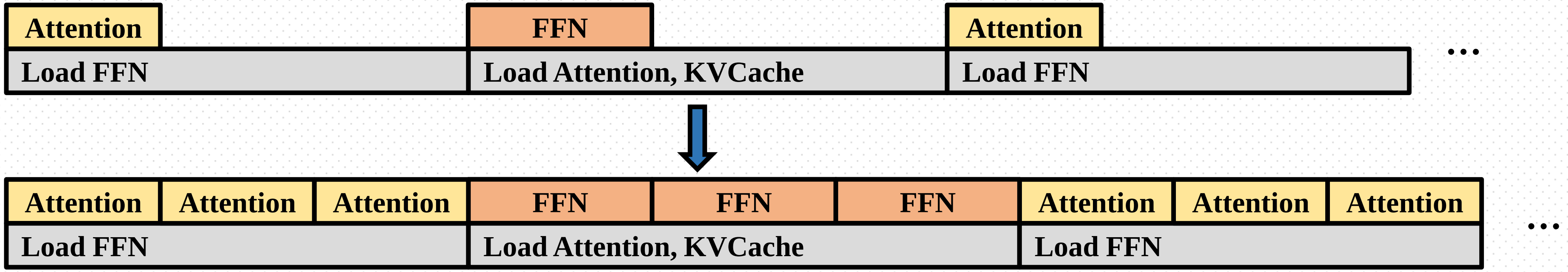
- ❑For dense model offloading, previous work has proposed **multi-batch pipeline** to hide PCIe transmission overhead



Amortize PCIe overhead
to improve overall throughput

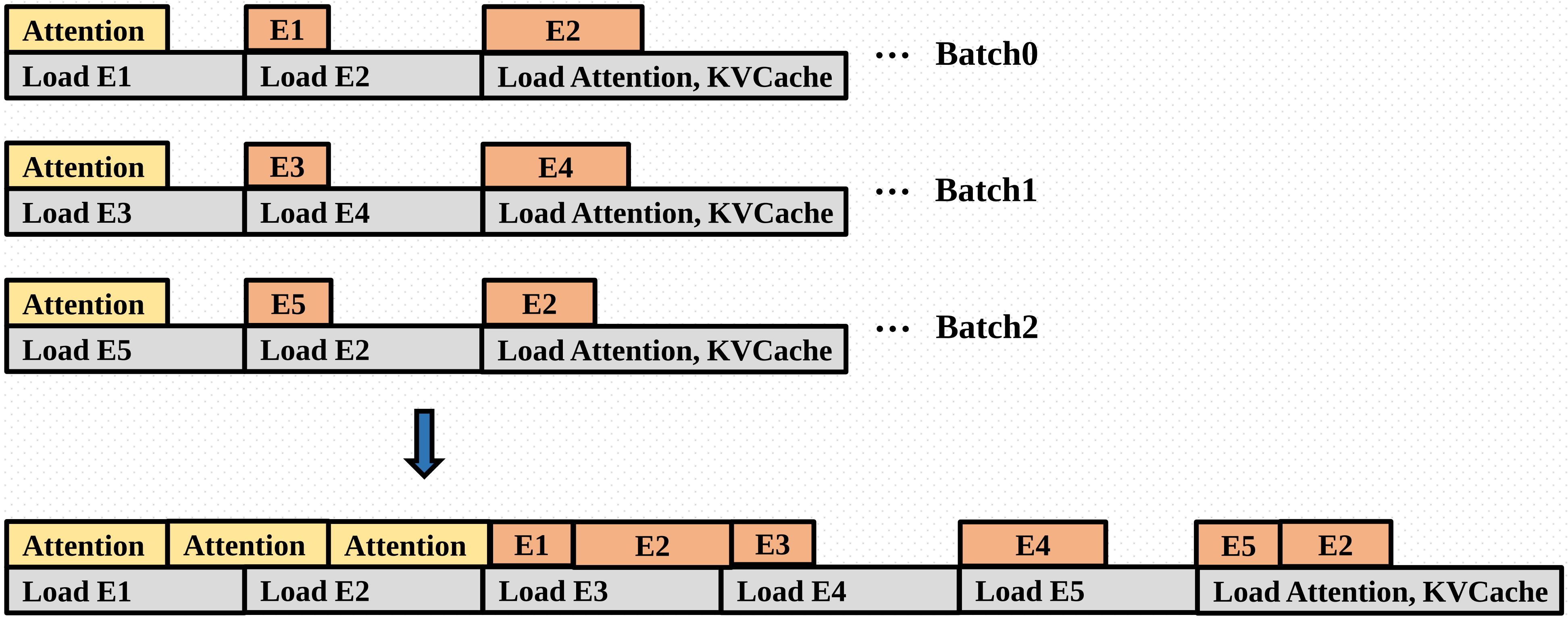
Multi-batch pipeline

FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU, ICML 2023



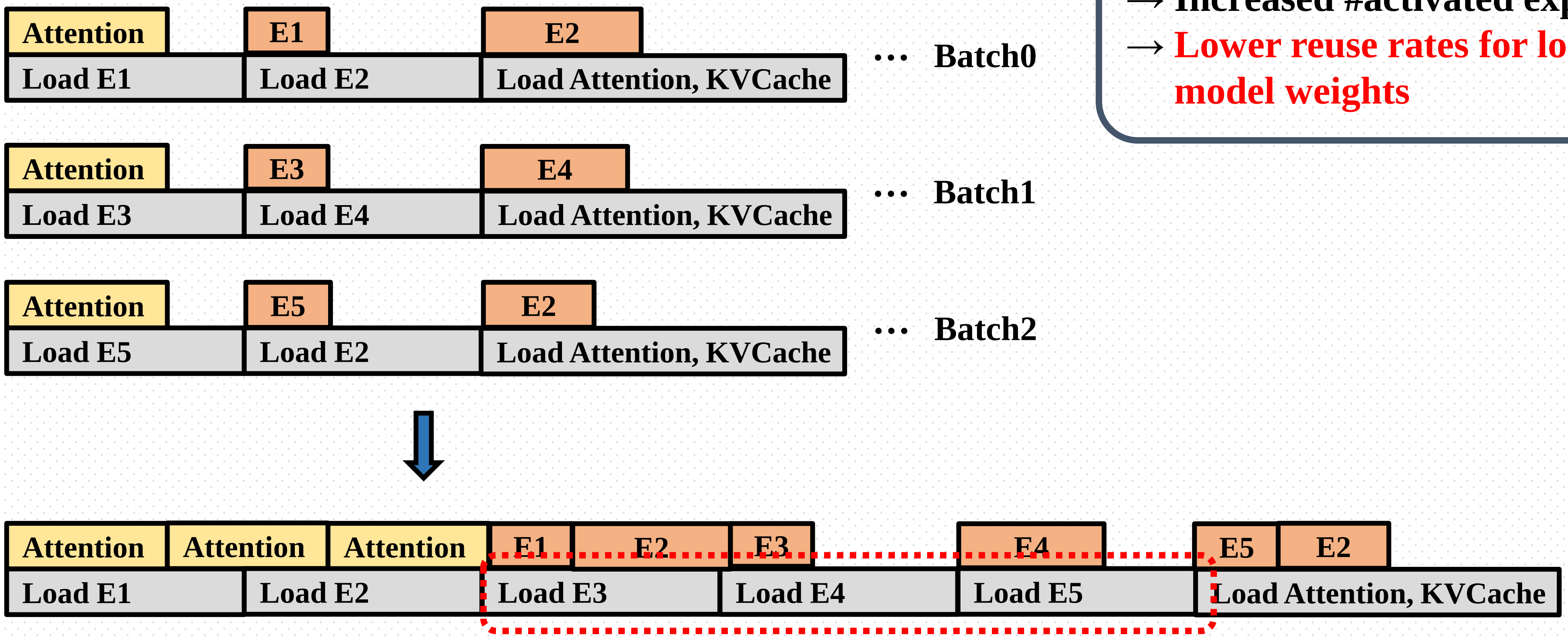
Motivation

❑What if MoE model?



Motivation

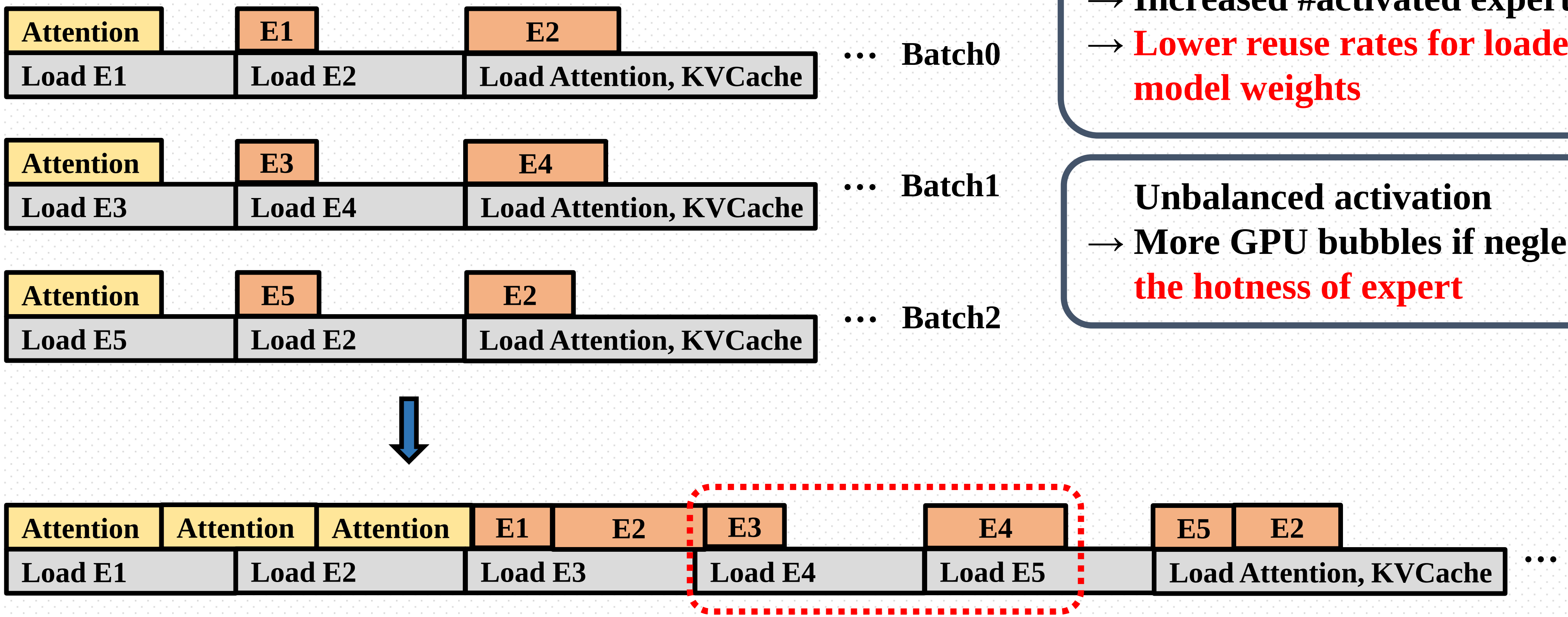
❑What if MoE model?



Multi-batch
→ Increased #activated experts
→ **Lower reuse rates for loaded model weights**

Motivation

❑What if MoE model?

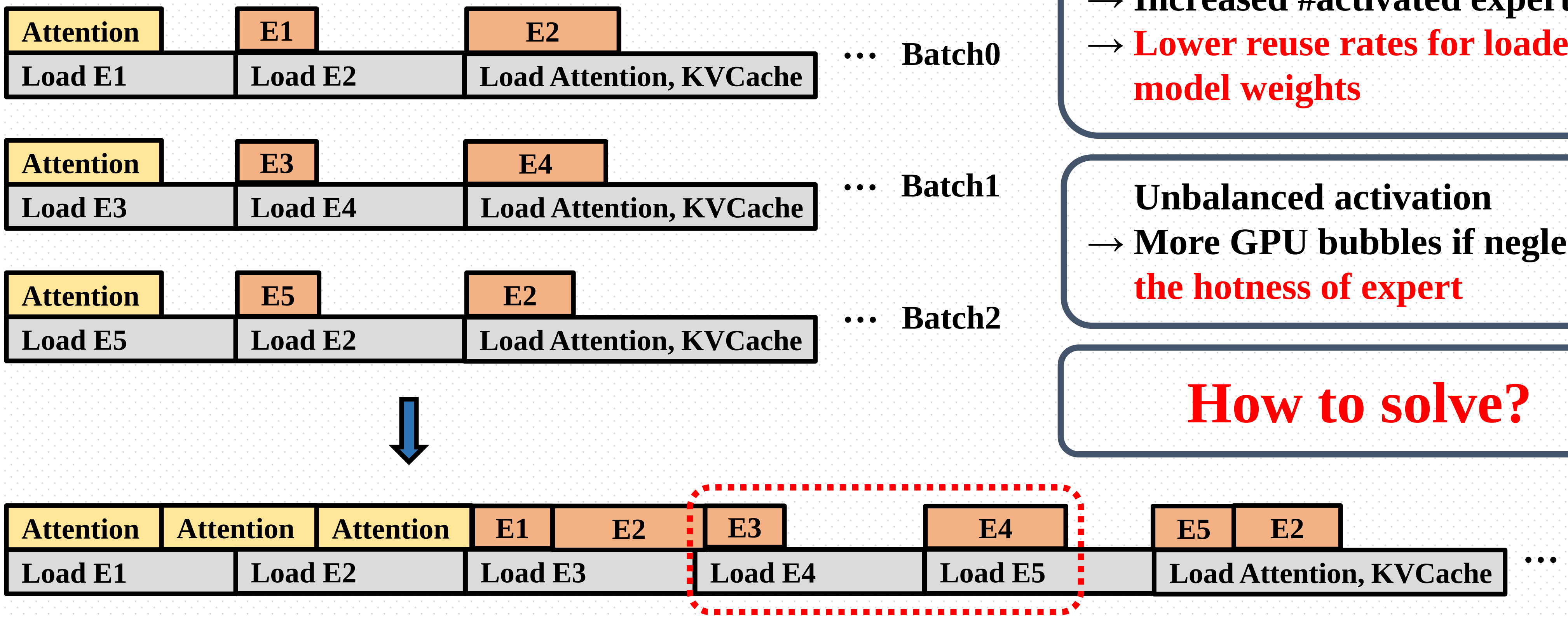


Multi-batch
→ Increased #activated experts
→ Lower reuse rates for loaded model weights

Unbalanced activation
→ More GPU bubbles if neglecting the hotness of expert

Motivation

❑What if MoE model?



Multi-batch
→ Increased #activated experts
→ Lower reuse rates for loaded model weights

Unbalanced activation
→ More GPU bubbles if neglecting the hotness of expert

How to solve?

Outline

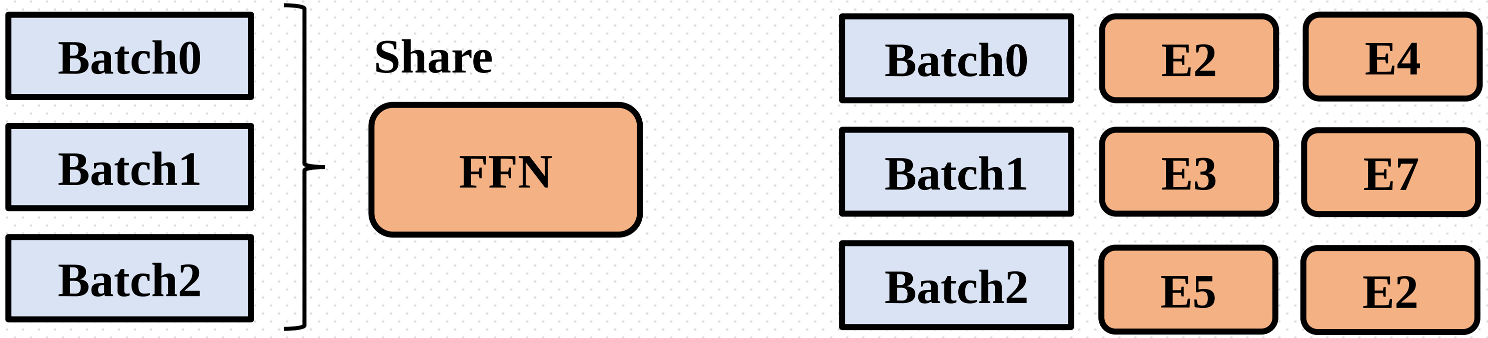
- ❑Background
- ❑Motivation
- ❑Klotski
- ❑Evaluation

Klotski

- Klotski: Expert-aware multi-batch pipeline
  - What is, and why expert-aware multi-batch pipeline?
  - How to implement expert-aware multi-batch pipeline?
  - Other technical points

Klotski: Expert-Aware Multi-Batch Pipeline

- **The key to amortize PCIe overhead: module weights sharing**
  - **Suppose: N batches, and m module weights are required**
  - **The amortized PCIe overhead is $\frac{m}{N} T_{PCIe}$**
  - **The smaller $\frac{m}{N}$, the easier to overlap / amortize**

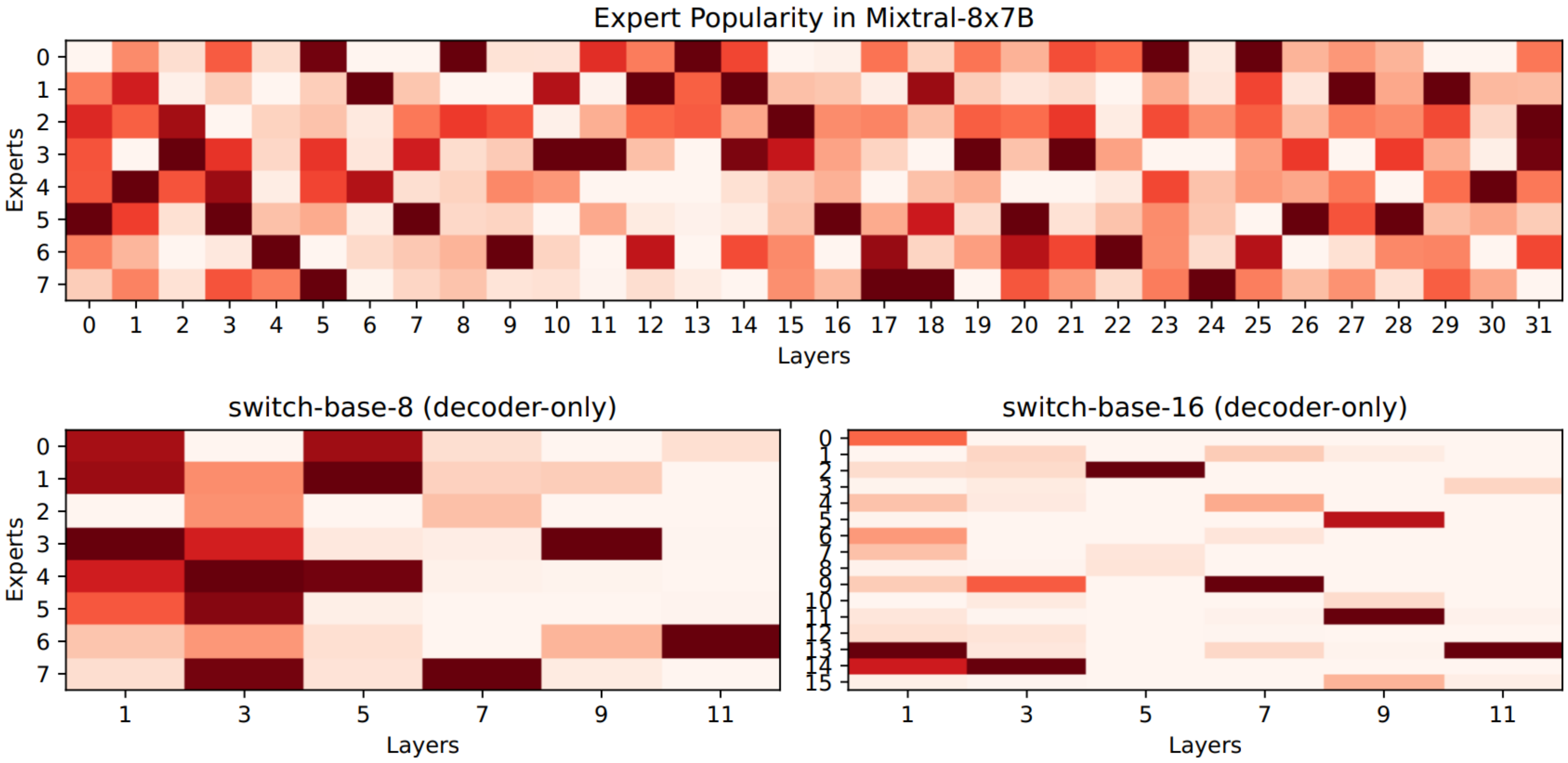


Dense Model: $m = 1$

MoE Model: $k \leq m \leq Nk$
k is #activated experts

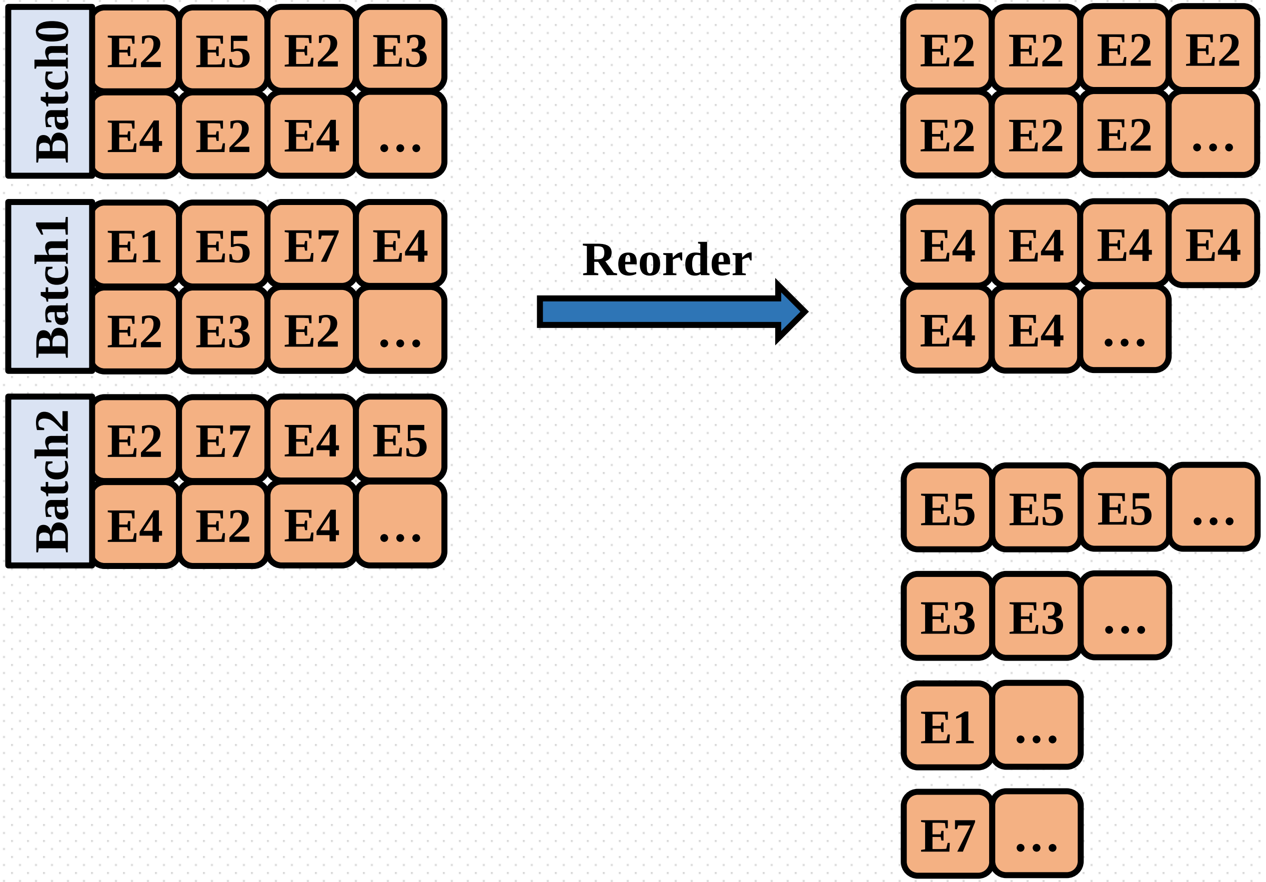
Klotski: Expert-Aware Multi-Batch Pipeline

❑Obversation: Expert activation are highly unbalanced



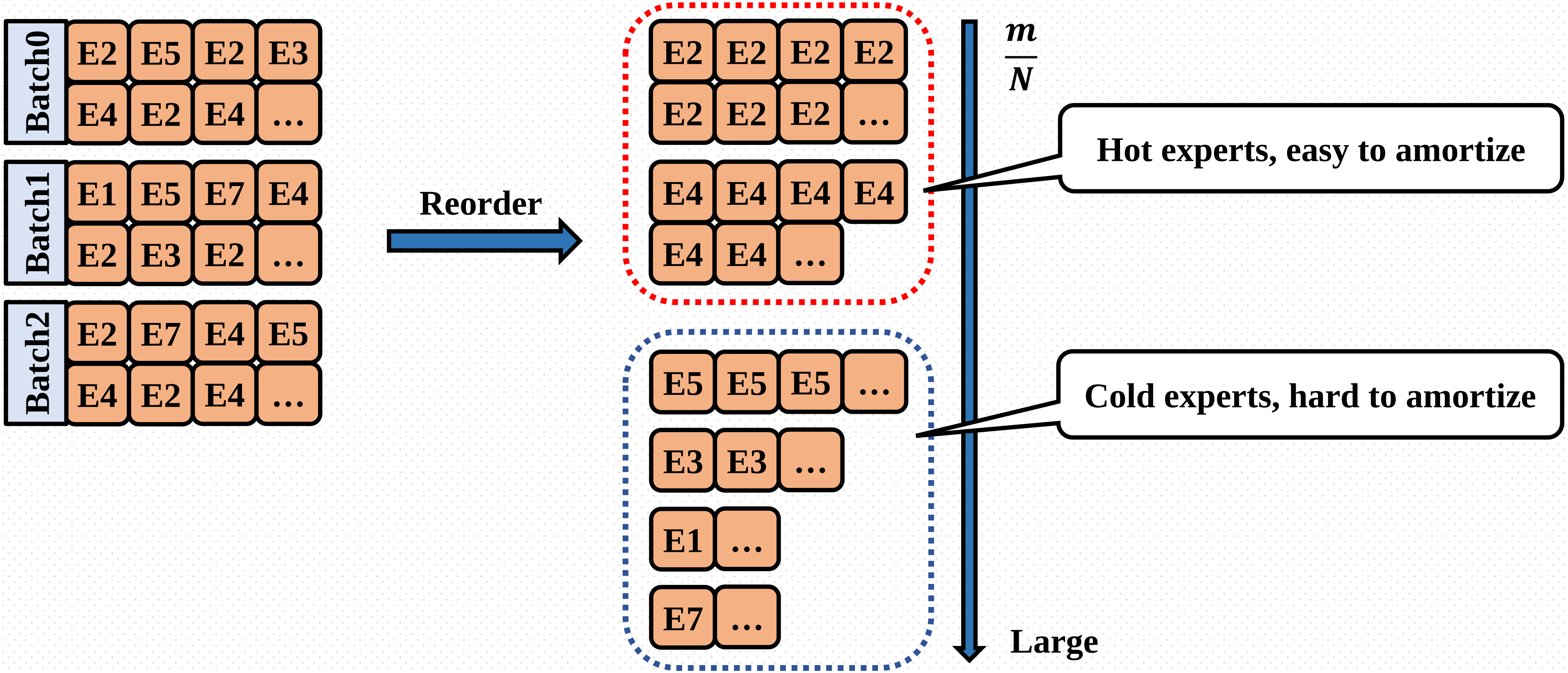
Klotski: Expert-Aware Multi-Batch Pipeline

- ❑Accordingly, batch(token)-centric computation can be reordered as expert-centric



Klotski: Expert-Aware Multi-Batch Pipeline

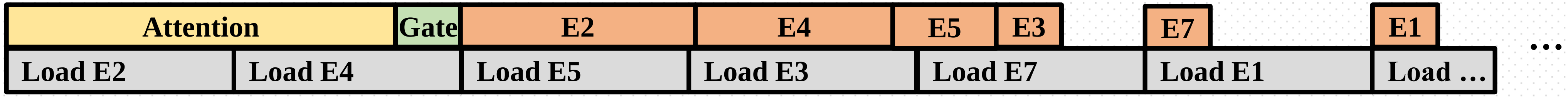
- ❑Accordingly, batch(token)-centric computation can be reordered as expert-centric



Klotski: Expert-Aware Multi-Batch Pipeline

❑Benefit:

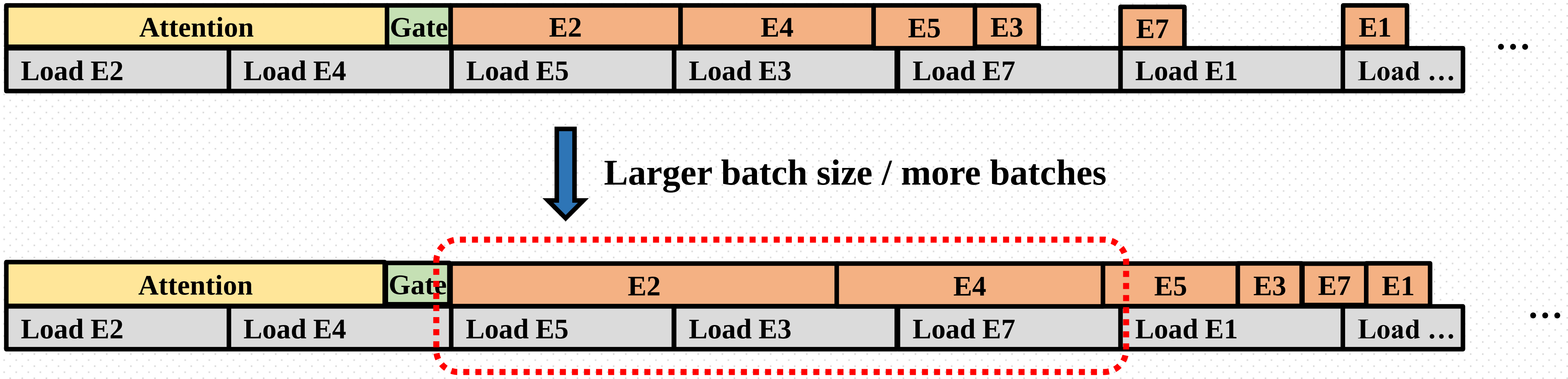
❖The computation of hot experts helps to hide the loading overhead of cold experts



Klotski: Expert-Aware Multi-Batch Pipeline

❑Benefit:

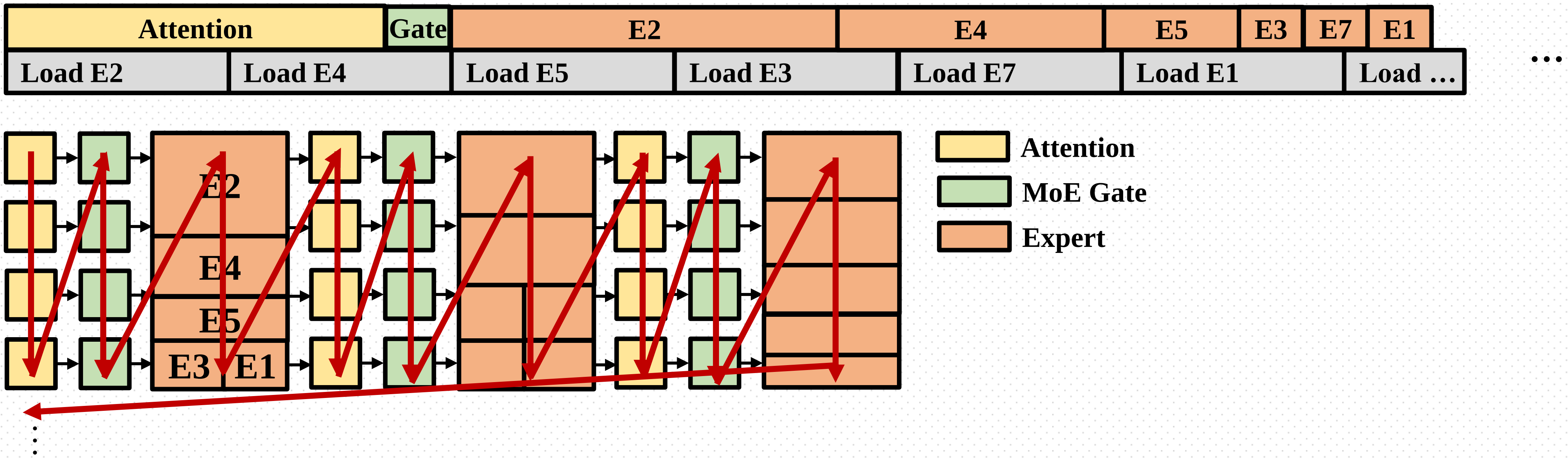
❖The computation of hot experts helps to hide the loading overhead of cold experts, especially for large batch sizes or #batches



Klotski: Expert-Aware Multi-Batch Pipeline

❑Summary - for MoE module:

- ❖Only prefetch hot experts before MoE gate
- ❖Reorder the computation to form expert-centric token batches, and prioritize the computation of hot experts
- ❖Prefetch cold experts during the computation of hot experts



Klotski

- Klotski: Expert-aware multi-batch pipeline
  - What is, and why expert-aware multi-batch pipeline?
  - How to implement expert-aware multi-batch pipeline?
  - Other technical points

Klotski: Technical Points

- To implement expert-aware multi-batch pipeline:
  - How to predict and prefetch hot experts?
  - How to set batch size and #batches?

Klotski: Hot Experts Prefetching

❑Expert correlation table

❖Key idea: the activation pattern of layer L-1 can reflect the activation tendency of layer L

For one token

L-1 activates:
- Expert 0
- Expert 1

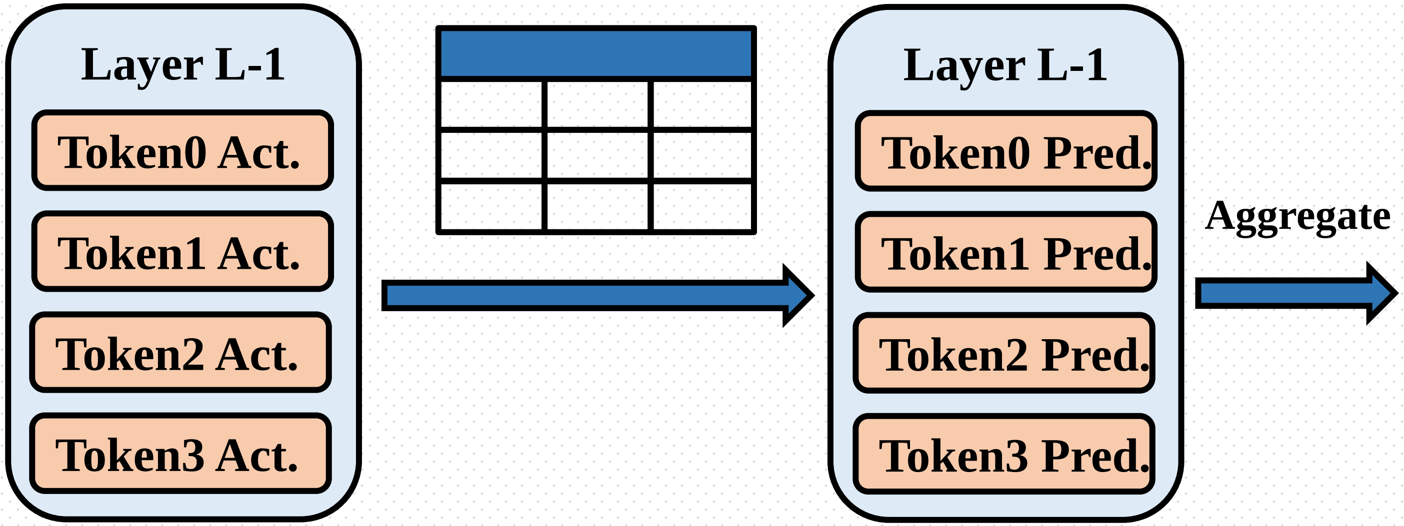
Layer L - 1	Layer L	
Expert	Expert	Activated Frequence
0	0	38
	1	27
	2	97
	3	15
1	0	66
	1	35
	2	41
	3	117
…	…	…

Predict L will activate:
- Expert 2
- Expert 3

Klotski: Hot Experts Prefetching

❑Expert correlation table

❖**Key idea: the activation pattern of layer L-1 can reflect the activation tendency of layer L**



Expert	Predicted Times
0	3
1	18
2	9
3	2

#Selected experts = *k*
(top-*k* setting of the model)

Klotski: Hot Experts Prefetching

❑Expert correlation table

- ❖Key idea: the activation pattern of layer L-1 can reflect the activation tendency of layer L
- ❖The correlation tables are produced during a pre-run, using random samples wikitext-2 dataset

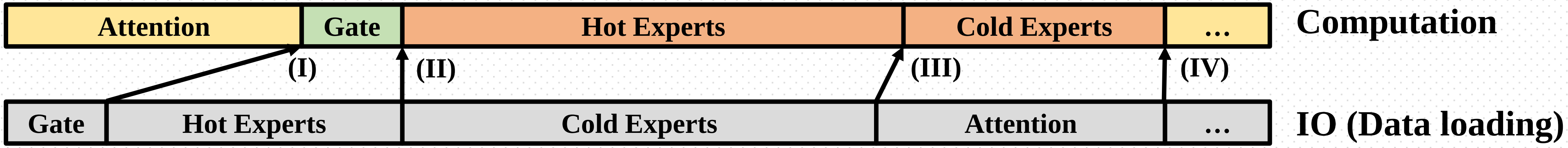
Klotski: Configuration Searching

- ❑The overlapping efficiency is significantly influenced by:
  - ❖Batch size
  - ❖#Batches (*n*)
- ❑Batch size is restricted to multiples of 4
  - ❖Only a few options are available, typically ranging from 4 to 64
- ❑For *n*, Klotski designs a cost model to search it

Klotski: Configuration Searching

❑Cost model

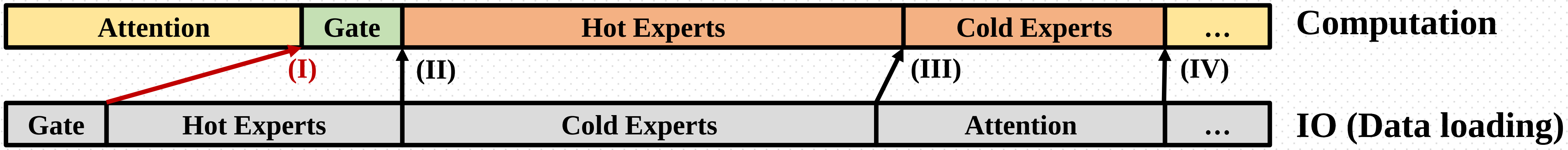
❖At 4 key moments of the pipeline, their required weights must have been completely loaded



Klotski: Configuration Searching

❑Cost model

❖At 4 key moments of the pipeline, their required weights must have been completely loaded



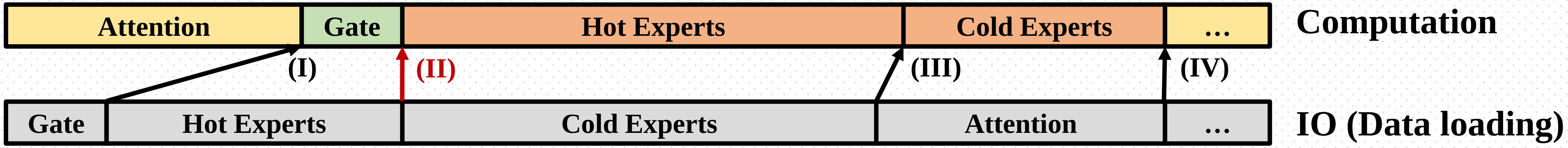
$$\boldsymbol{n * t_{c\_A} \geq t_{IO\_G}} \qquad \textbf{(I)}$$

Before the computation of gate begins, the IO of its weights should have finished

Klotski: Configuration Searching

❑Cost model

❖At 4 key moments of the pipeline, their required weights must have been completely loaded



$$n * t_{c\_A} \geq t_{IO\_G} \quad \text{(I)}$$

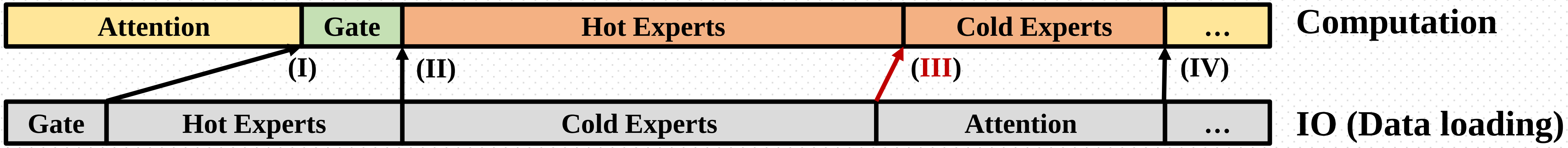
$$n * (t_{c\_A} + t_{c\_G}) \geq t_{IO\_G} + K * t_{IO\_E} \quad \text{(II)}$$

Before the computation of hot experts begins, the IO of all the K hot experts should have finished

Klotski: Configuration Searching

❑Cost model

❖**At 4 key moments of the pipeline, their required weights must have been completely loaded**



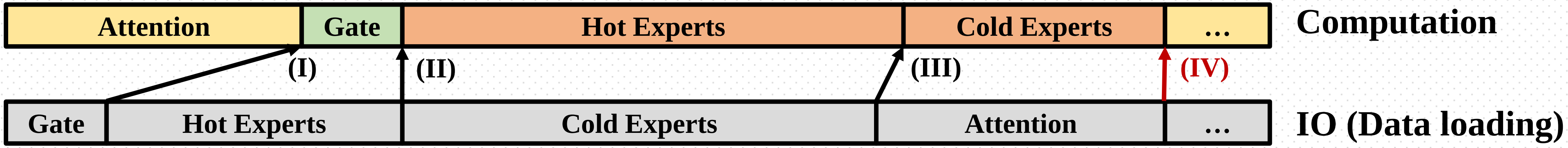
$$\begin{cases} n * t_{c\_A} \geq t_{IO\_G} & \text{(I)} \\ n * (t_{c\_A} + t_{c\_G}) \geq t_{IO\_G} + K * t_{IO\_E} & \text{(II)} \\ n * (t_{c\_A} + t_{c\_G}) + t_{c\_hot-E} \geq t_{IO\_G} + (K+1) * t_{IO\_E} & \text{(III)} \end{cases}$$

Before the computation of cold experts begins, the IO of at least 1 cold expert should have been finished

Klotski: Configuration Searching

❑Cost model

❖At 4 key moments of the pipeline, their required weights must have been completely loaded



Before the computation of the next layer begins, the IO of:

- all the cold experts (referred to as C)
- next layer's attention weights and the first batch's KVCache

should have finished

$n * t_{c\_A} \geq$ … (I)

$n * (t_{c\_A} +$ … (II)

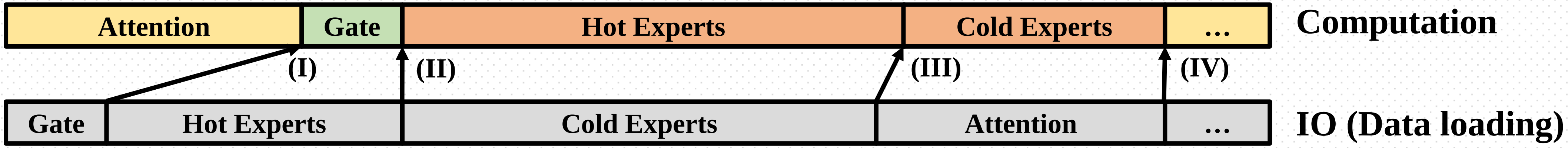
$n * (t_{c\_A} + t_{c\_G}) + t_c$ … $_{IO\_G} + (K + 1) * t_{IO\_E}$ (III)

$$n * (t_{c\_A} + t_{c\_G}) + t_{c\_hot-E} + \sum_{E_i \in C}^{C} t_{c\_E_i} \geq t_{IO\_G} + (K + len(C)) * t_{IO\_E} + t_{IO\_A}$$ (IV)

Klotski: Configuration Searching

❑Cost model

❖At 4 key moments of the pipeline, their required weights must have been completely loaded



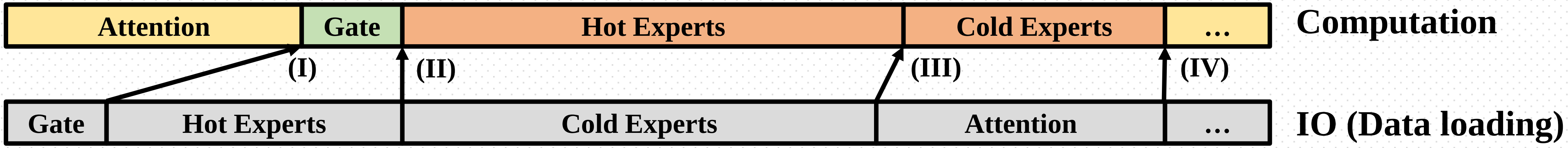
$$
\begin{cases}
n * t_{c\_A} \geq t_{IO\_G} & \text{(I)} \\
n * (t_{c\_A} + t_{c\_G}) \geq t_{IO\_G} + K * t_{IO\_E} & \text{(II)} \\
n * (t_{c\_A} + t_{c\_G}) + t_{c\_hot-E} \geq t_{IO\_G} + (K+1) * t_{IO\_E} & \text{(III)} \\
n * (t_{c\_A} + t_{c\_G}) + t_{c\_hot-E} + \sum_{E_i \in C}^{C} t_{c\_E_i} \geq t_{IO\_G} + (K + len(C)) * t_{IO\_E} + t_{IO\_A} & \text{(IV)}
\end{cases}
$$

How to get? Offline profiling

Klotski: Configuration Searching

❑Cost model

❖**At 4 key moments of the pipeline, their required weights must have been completely loaded**



$$\begin{cases} n * t_{c\_A} \geq t_{IO\_G} & \text{(I)} \\ n * \left(t_{c\_A} + t_{c\_G}\right) \geq t_{IO\_G} + K * t_{IO\_E} & \text{(II)} \\ n * \left(t_{c\_A} + t_{c\_G}\right) + t_{c\_hot-E} \geq t_{IO\_G} + (K+1) * t_{IO\_E} & \text{(III)} \\ n * \left(t_{c\_A} + t_{c\_G}\right) + t_{c\_hot-E} + \sum_{E_i \in C}^{C} t_{c\_E_i} \geq t_{IO\_G} + \left(K + len(C)\right) * t_{IO\_E} + t_{IO\_A} & \text{(IV)} \end{cases}$$

Search the smallest *n*!

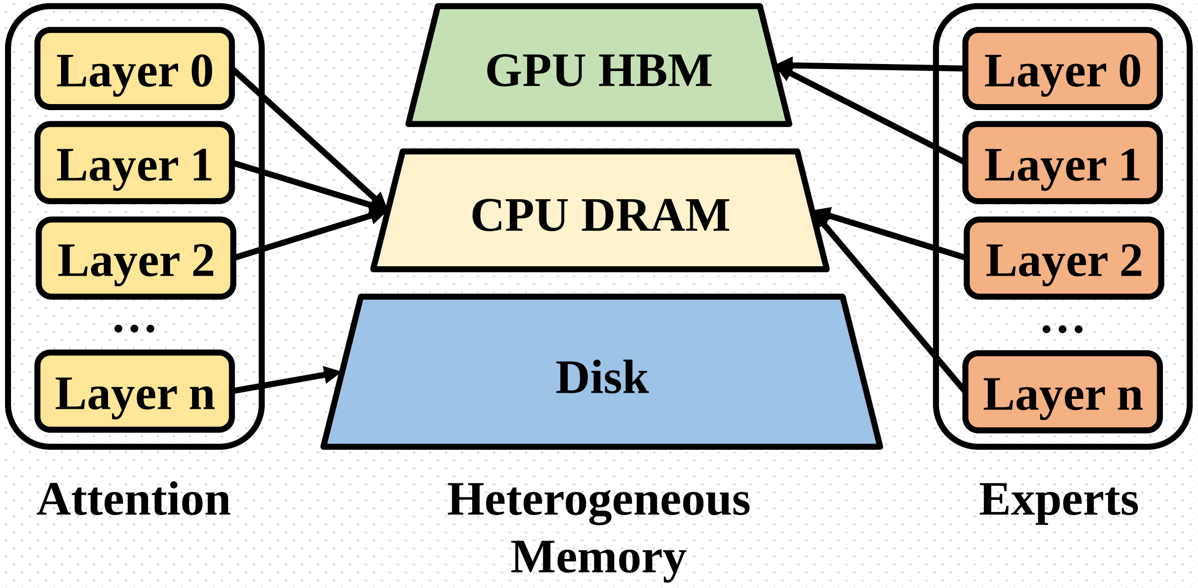
Klotski

- ❑ Klotski: Expert-aware multi-batch pipeline
  - ❖ What is, and why expert-aware multi-batch pipeline?
  - ❖ How to implement expert-aware multi-batch pipeline?
  - ❖ Other technical points

Klotski: Other Technical Points

❑Tensor Placement

- ❖GPU HBM + CPU DRAM + Disk
- ❖Different placement strategy for different types of tensors
  - ➢Priority of using high-end memory: expert weights > others
  - ➢Granularity: Layer



Klotski: Other Technical Points

- Klotski also supports compression as optional optimizations:
  - Quantization (4-bits) for model weights
  - Compression (StreamlingLLM) for KVCache

[1] Half-Quadratic Quantization of Large Machine Learning Models
[2] Efficient Streaming Language Models with Attention Sinks, ICLR 2024

Outline

- ❑Background
- ❑Motivation
- ❑Klotski
- ❑Evaluation

Evaluation: Setup

❑Hardware

Hardware	Environment 1		Environment 2	
	Model	Memory	Model	Memory
GPU	**NVIDIA RTX 3090 x1**	**24 GB**	**NVIDIA H800 x1**	**80 GB**
CPU	Intel Xeon Gold 5318Y	256 GB	Intel Xeon Platinum 8470	800 GB
Disk	SSD	2 TB	SSD	1 TB
PCIe	4.0 x 16 (~25 GB/sec)		5.0 x 16 (~47 GB/sec)	
Disk Read	1 GB/sec		/	

❑Models & Datasets

Model	Mixtral-8×7B	Mixtral-8×22B
#Params.	46.7 B	141 B
#Act. Params.	12.9 B	39.2 B
#Layers	32	56
#Experts	8	8
#Act. Experts	2	2

Dataset	wikitext-103
Input Len.	512
Output Len.	32

Evaluation: Setup

❑Baselines

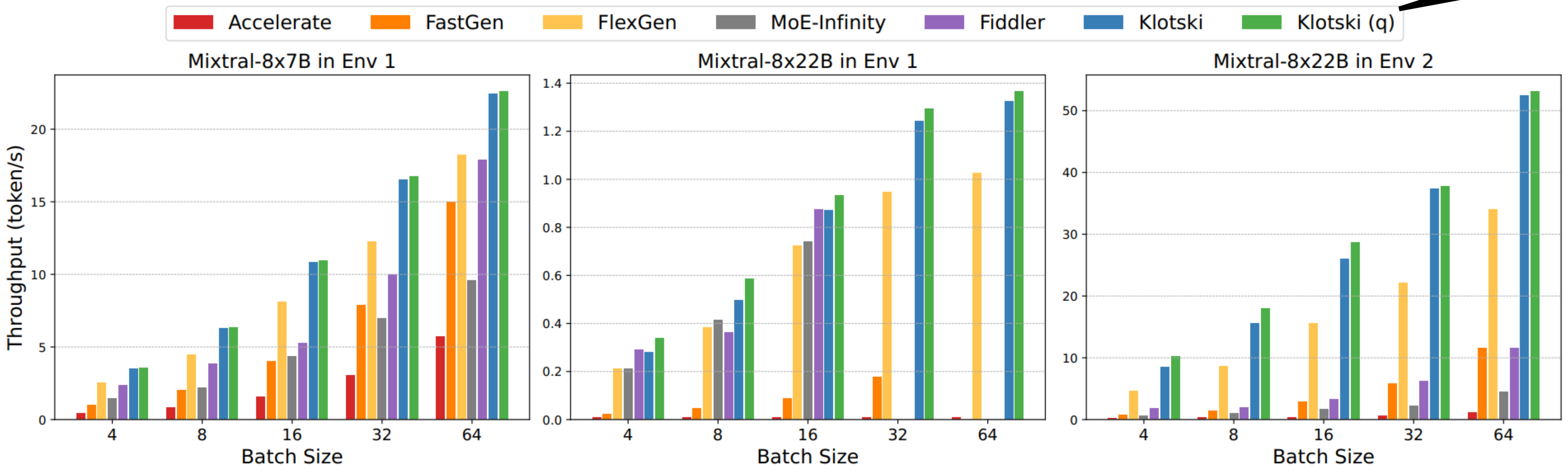
- ❖Single batch, no prefetching:
  - ➢Hugging Face Accelerate (Accelerate)
  - ➢DeepSpeed-FastGen (FastGen)
- ❖Multi-batch pipeline, no adaptation for MoE:
  - ➢FlexGen, ICML 2023
- ❖Single batch, compute MoE on CPU
  - ➢Fiddler, PML4LRS@ICLR 2024
- ❖Single batch, predict and prefetch experts
  - ➢MoE-Infinity, arXiv 24.01

Evaluation: Main Results

❑N2N Throughput

❖Throughput = #total_tokens / total_time_cost

4-bits quantization



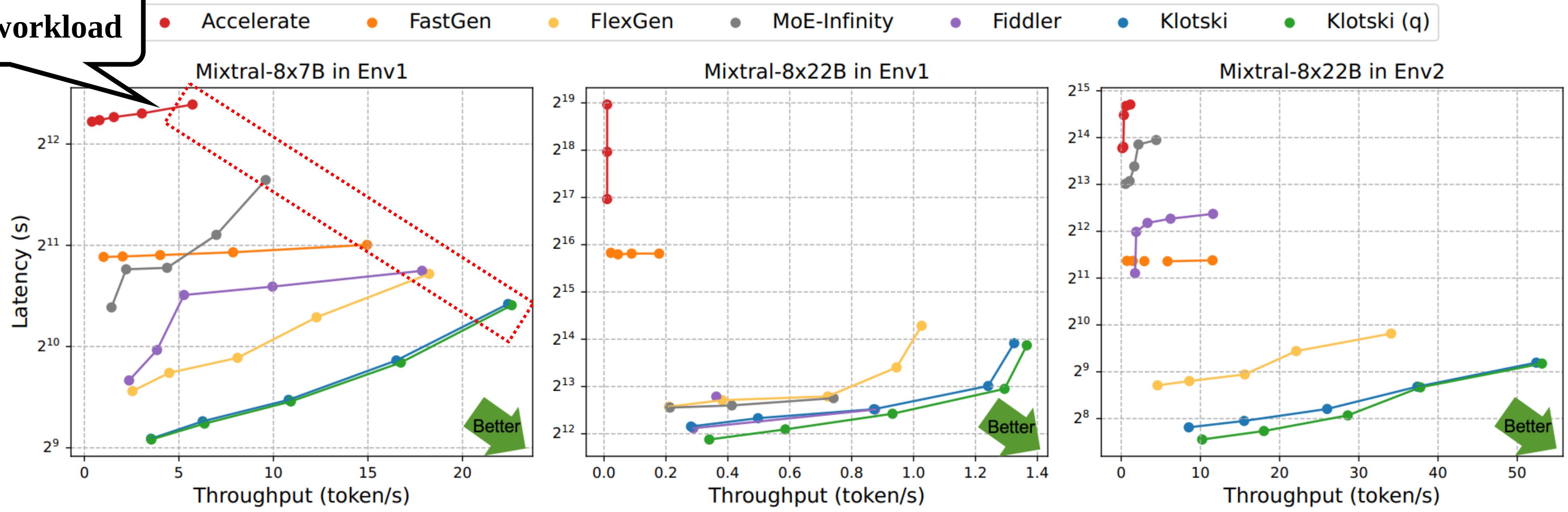
#Batches (n) = 15	#Batches (n) = 10	#Batches (n) = 15
Relatively constrained PCIe	Extremely constrained PCIe	Relatively constrained PCIe

Evaluation: Main Results

❑Latency(?)-throughput trade-off

❖Latency refers to the time cost of finishing a given dataset

The same workload



Evaluation: Ablation Study

❑Impacts of optimizations

Model	Environment 1		Environment 2
	Mixtral-8×7B	Mixtral-8×22B	Mixtral-8×22B
Simple Pipeline	5.721	0.01	1.149
+ Multi batches	18.24	0.97	34.07
+ Only prefetch hot experts	19.074	1.127	44.17
Klotski (+ adjust order)	22.414	1.325	52.85
Klotski (q)	22.604	1.366	53.125

Amortize IO overhead of non-MoE modules

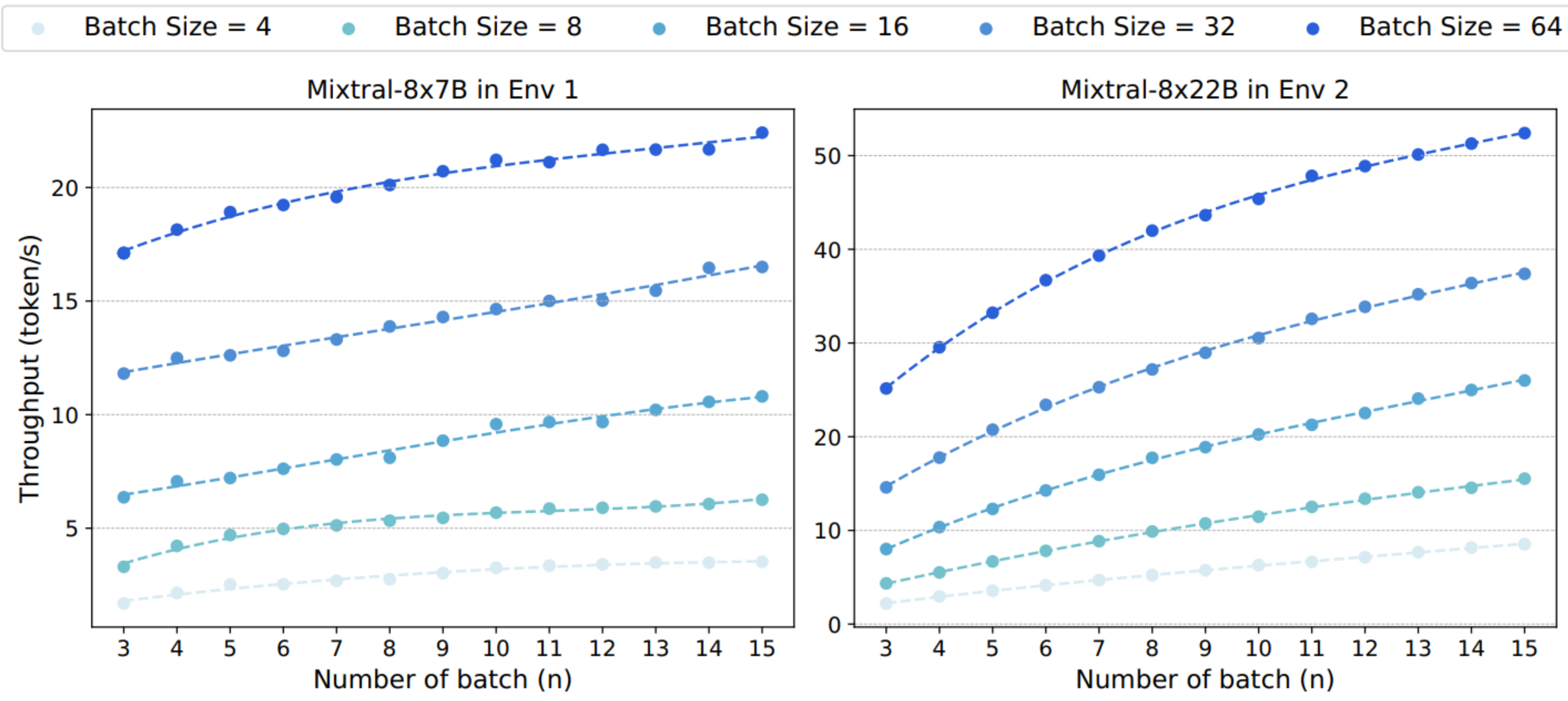
Avoid incorrect prefetching

Hide IO overhead of cold experts

Evaluation: Ablation Study

❑Impacts of batch size and *n*

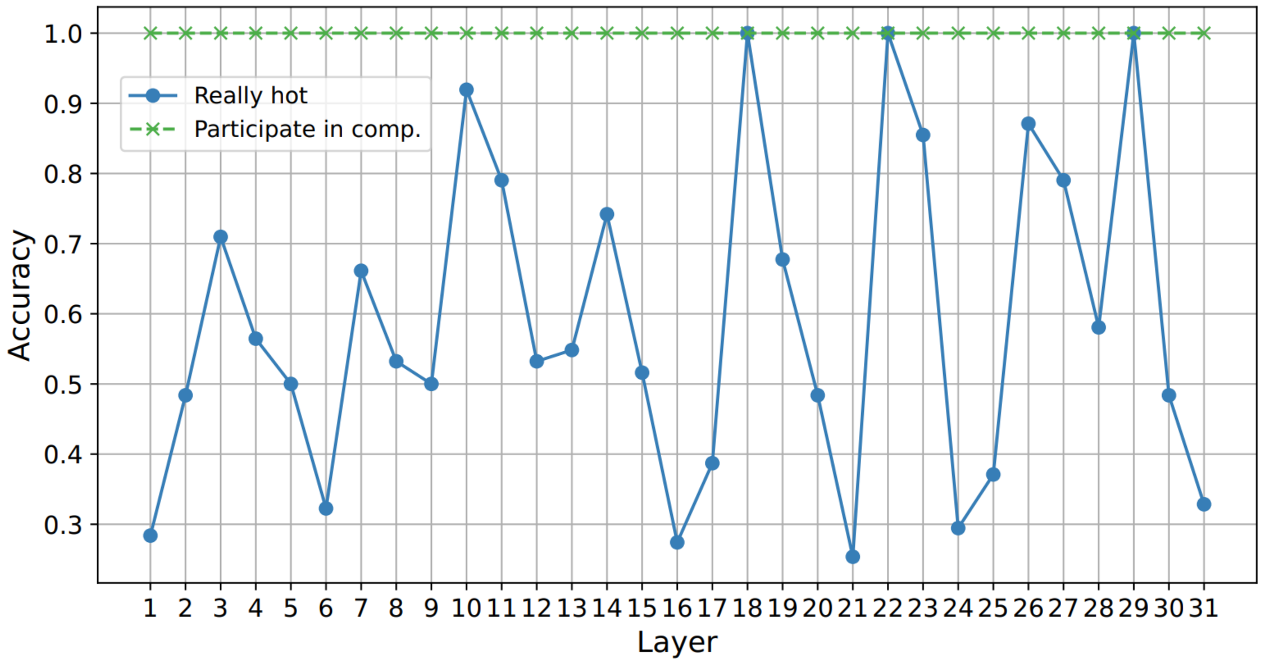
- ❖The larger, the higher throughput
- ❖Should not be too large to prevent KVCache being too heavy



Evaluation: Ablation Study

❑Accuracy of prefetching

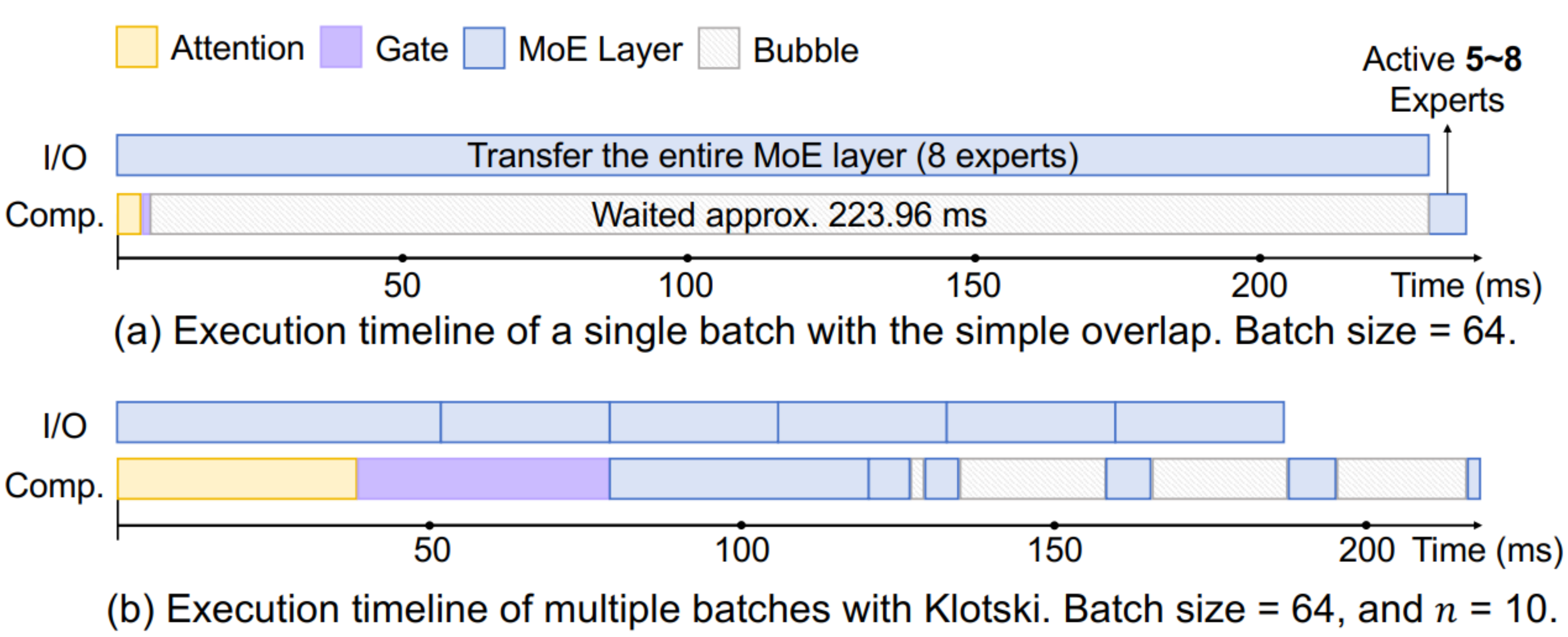
- ❖Green line: the recognized hot experts are always activated
- ❖Blue line: the accuracy of predicting hot experts, not very good



Evaluation: Ablation Study

❑GPU bubble reduction

- ❖Env1, Mixtral-8×7B
- ❖Klotski *vs* signle batch + simple prefetching



~2367 ms

~2100 ms reduction

~215 ms

Conclusion

- **Klotski:**
  - **Designed for MoE whole-model offloading in resource-constrained environments**
  - **Introduces expert-hotness-aware MoE pipeline to multi-batch pipeline, hiding PCIe transmission overhead and improving throughput**

Conclusion

❑Limitations:

- ❖Only recommended for offline large batch inference
  - ➢Multi-batch and large batch size significantly increase latency (TTFT, TPOT), making Klotski unsuitable for online serving deployment
  - ➢Personal deployment generally cannot provide sufficiently large or numerous batches

Conclusion

❑Limitations:

- ❖Only recommended for offline large batch inference
- ❖Only covers old MoE model like Mixtral 8x7B, Switch
  - ➢They are **large-expert, low-sparsity** models (Mixtral 8x7B, activates 2 of 8 experts), while current mainstream models are **small-expert, high-sparsity** (DeepSeek-V3, activates 8 from 256 experts)
  - ➢Klotski's design relies on unbalanced expert activation, while **current MoE models pursue expert load balancing**

Conclusion

❑Limitations:

- ❖Only recommended for offline large batch inference
- ❖Only covers old MoE model like Mixtral 8x7B, Switch
- ❖Constrained context length due to the heavy KVCache of multi-batch
  - ➢For example, 512+32 in the paper's evaluation section

The End

- Thank you!