PAT: Accelerating LLM Decoding via Prefix-Aware Attention with Resource Efficient Multi-Tile Kernel

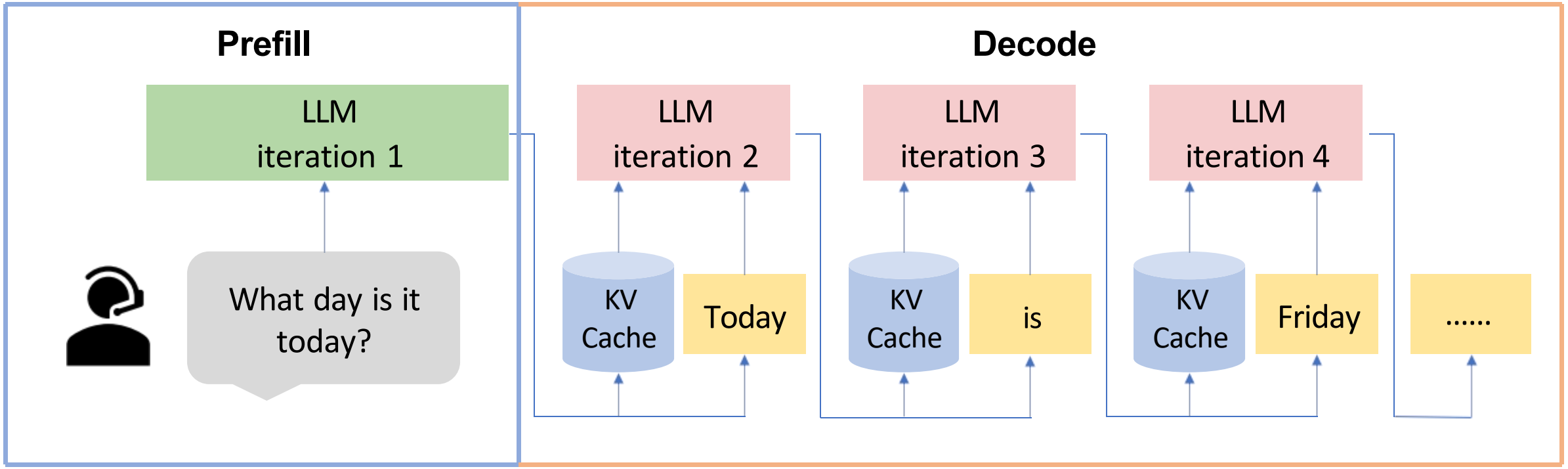
ASPLOS 2026

Jinjun Yi†, **Zhixin Zhao†**, Yitao Hu*, Ke Yan, Weiwei Sun,
Hao Wang, Laiping Zhao, Yuhao Zhang, Wenxin Li, Keqiu Li

https://arxiv.org/abs/2511.22333
https://github.com/flashserve/PAT

LLM Inference

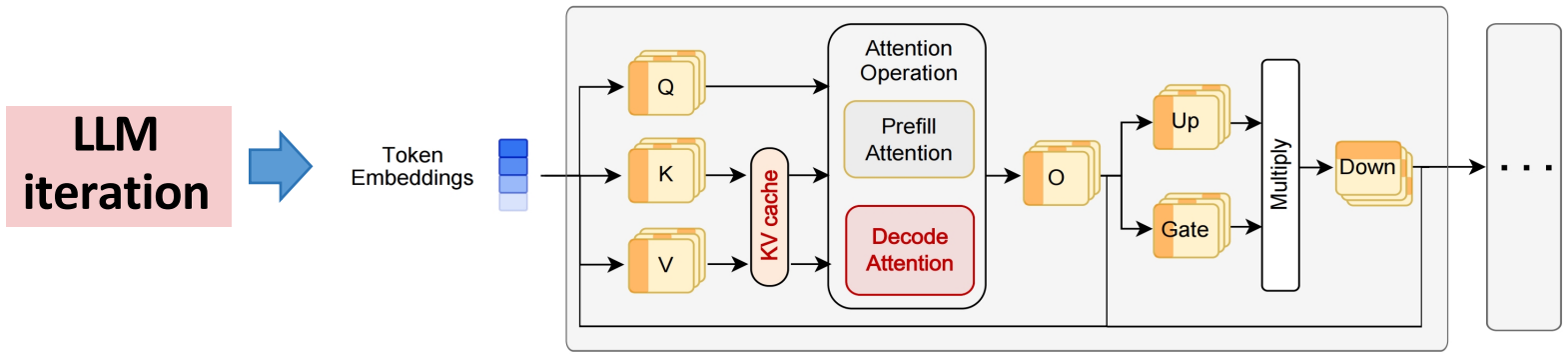
- **LLM inference** has two phases: **prefill + autoregressive decode**
  - **KV cache** reuses past KV, so each decode step only processes the new token

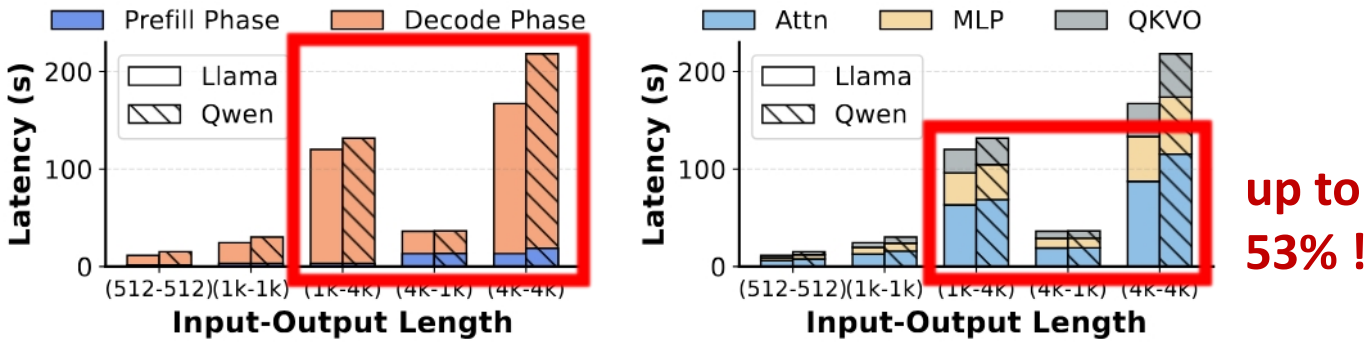


[1] https://www.usenix.org/conference/fast25/presentation/qin

Decoding Attention in LLM Inference

- **Long context and outputs shift latency toward decode attention**
  - **Decode attention contributes up to 53% of e2e latency** (A100, batch=64)

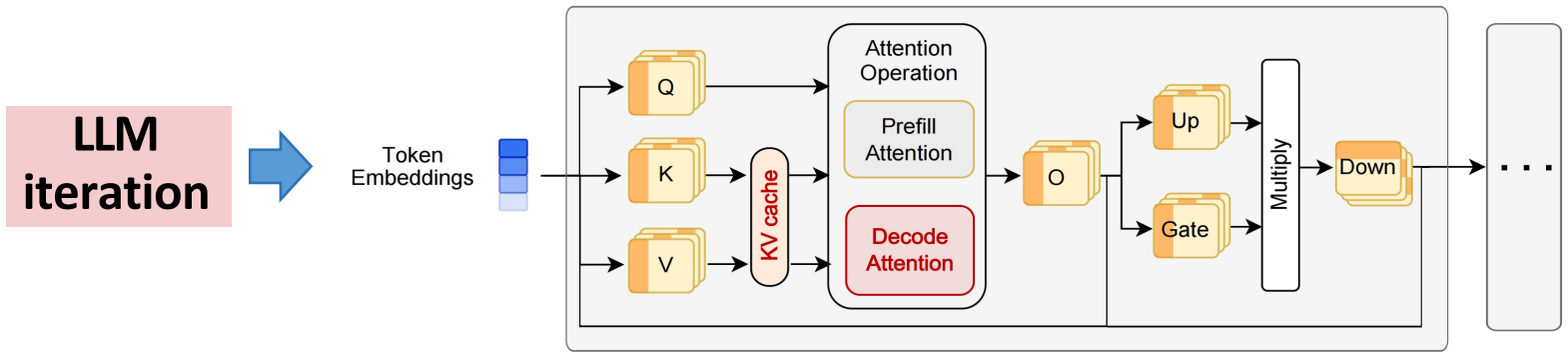


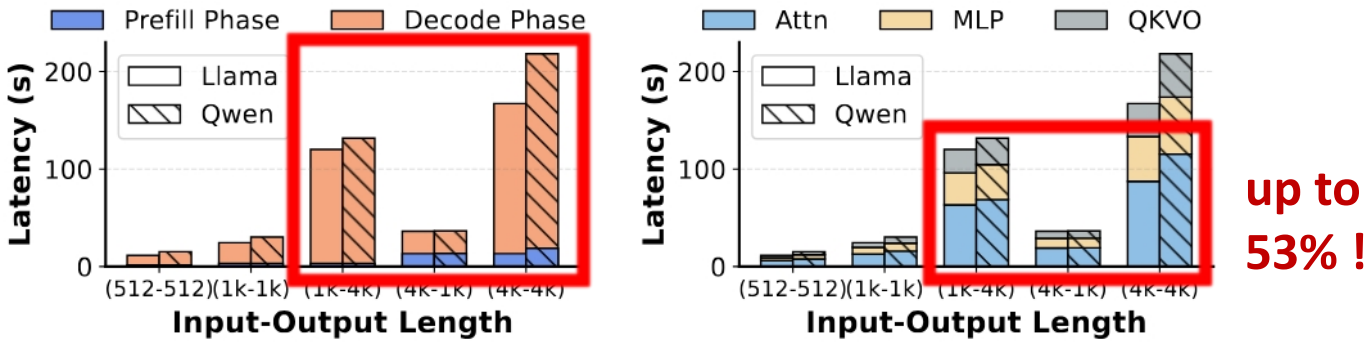


up to 53% !

Decoding Attention in LLM Inference

- **Long context and outputs shift latency toward decode attention**
  - **Decode attention contributes up to 53% of e2e latency** (A100, batch=64)



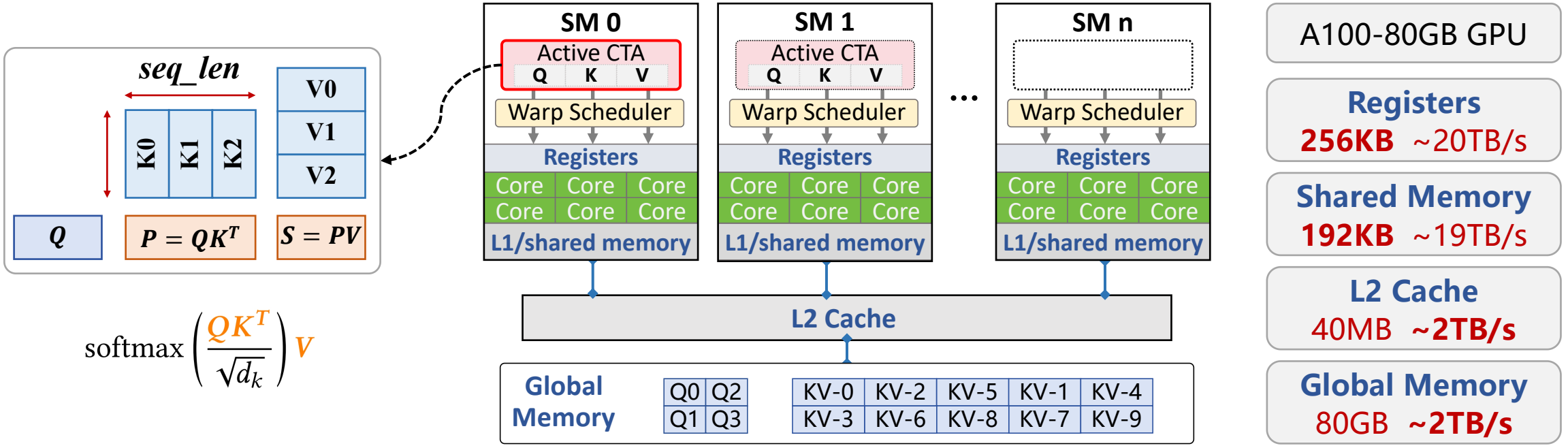


up to 53% !

How to speed up decoding attention?

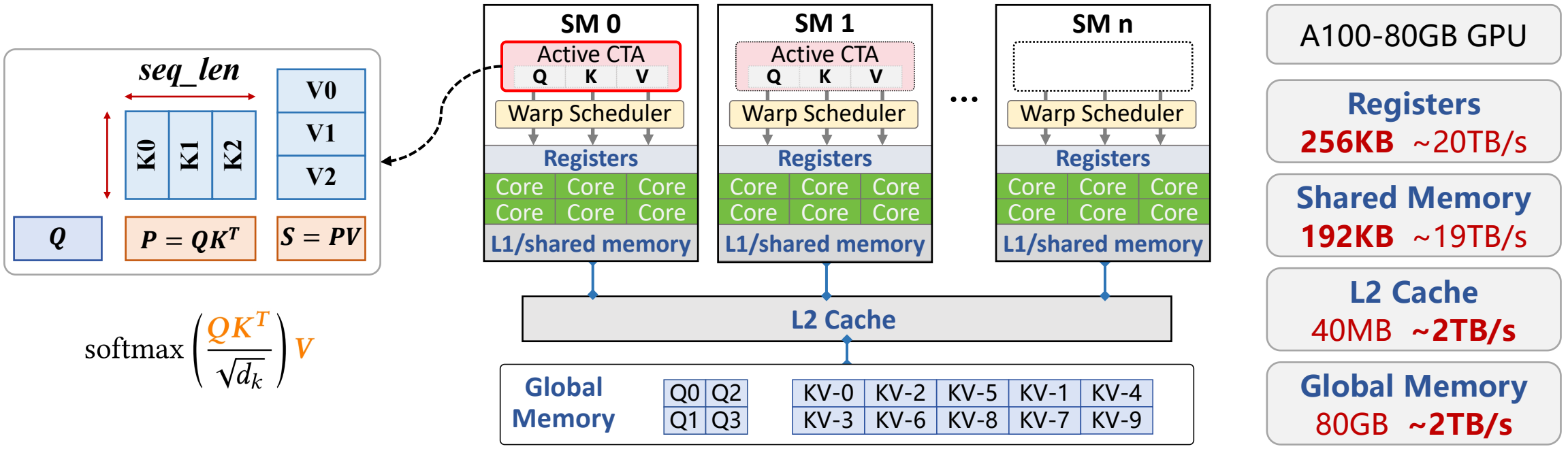
Decoding Attention in LLM Inference

- **Decode attention**: two GEMV → **heavy KV reads, light compute**



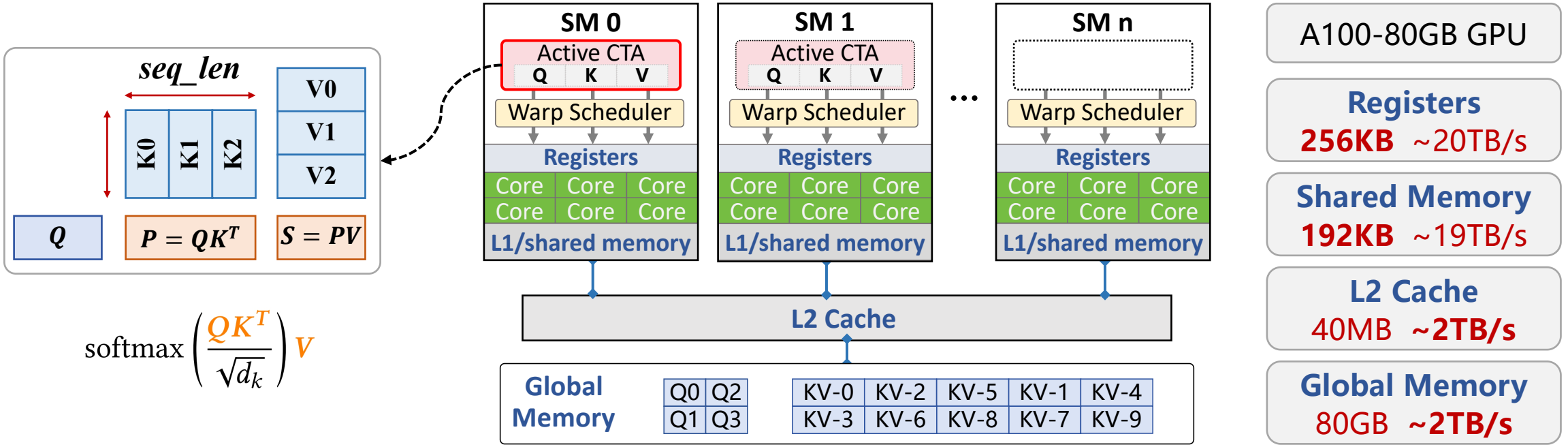
Decoding Attention in LLM Inference

- **Decode attention**: two GEMV → **heavy KV reads, light compute**
  - Goal 1: reduce slow global-memory accesses for KV reads



Decoding Attention in LLM Inference

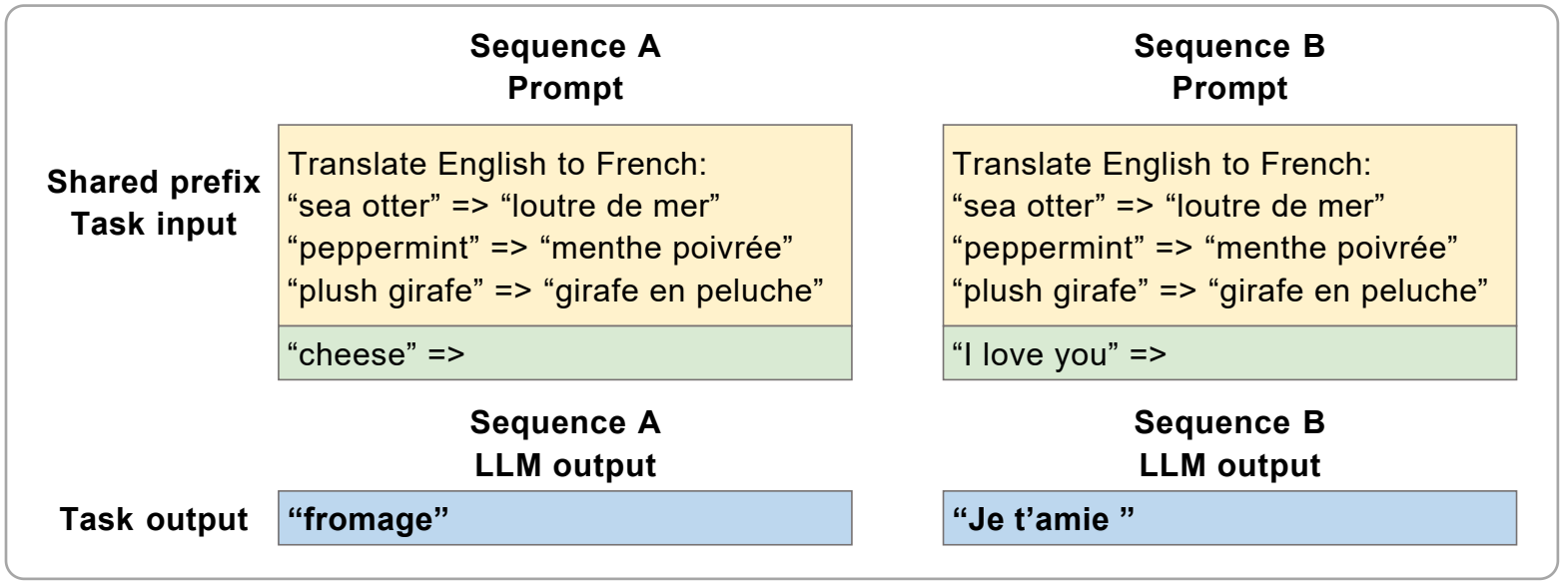
- **Decode attention**: two GEMV → **heavy KV reads, light compute**
  - Goal 1: reduce slow global-memory accesses for KV reads (memory-bound)
  - Goal 2: improve on-chip memory utilization (registers / shared memory)



In-batch Shared Prefixes

- **Observation**: **in-batch shared prefixes are common and hierarchical**
  - **Long system prompts**: thousands of tokens [1]
  - Prompt templates / few-shot bring **multi-level intra-batch prefixes**

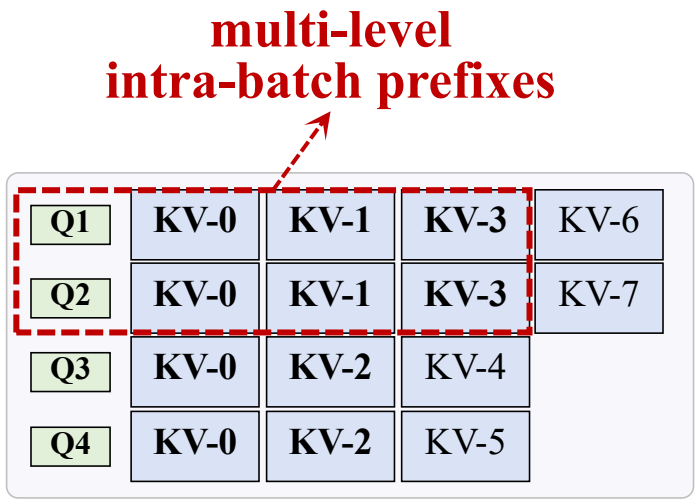
Q1	KV-0	KV-1	KV-3	KV-6
Q2	KV-0	KV-1	KV-3	KV-7
Q3	KV-0	KV-2	KV-4	
Q4	KV-0	KV-2	KV-5	

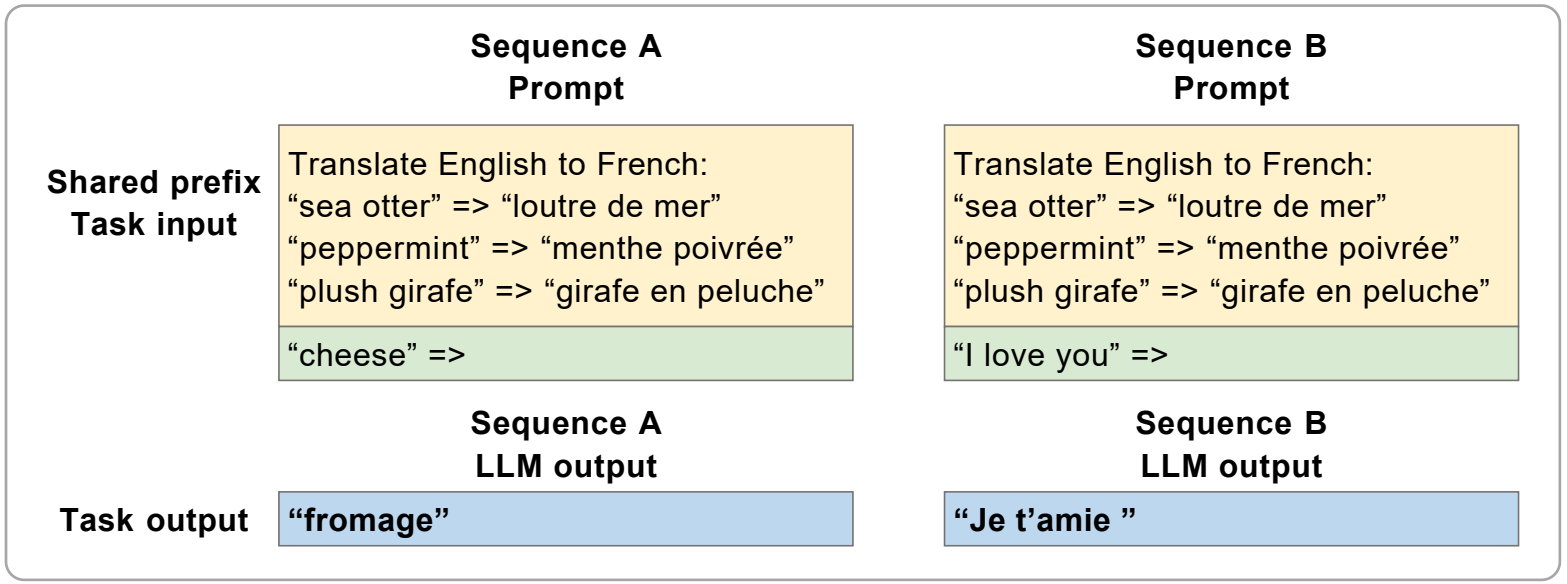


[1] https://github.com/x1xhlol/system-prompts-and-models-of-ai-tools
[2] https://arxiv.org/abs/2309.06180

In-batch Shared Prefixes

- **Observation**: **in-batch shared prefixes are common and hierarchical**
  - **Long system prompts**: thousands of tokens [1]
  - Prompt templates / few-shot bring **multi-level intra-batch prefixes**
    - Intra-batch prefix ratio in Mooncake trace: 2.8-83% (**36% on average**)

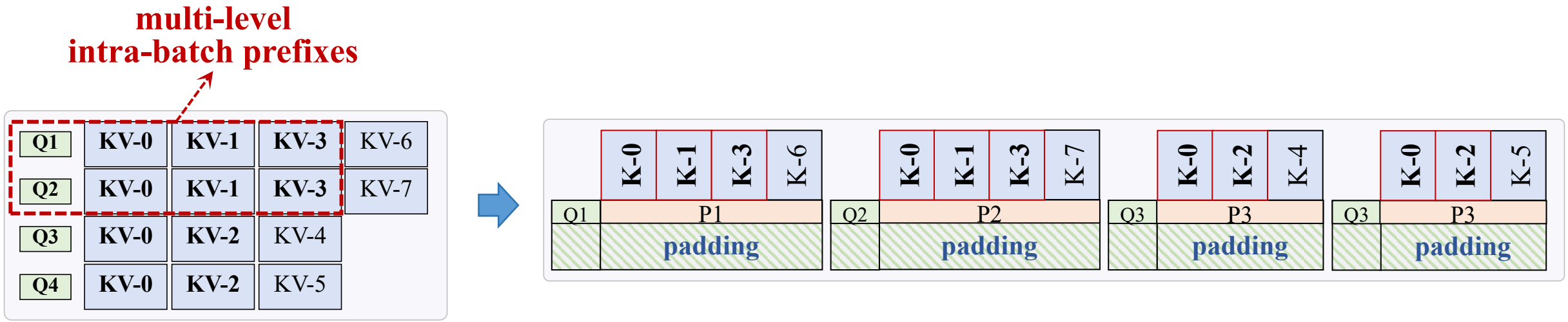




[1] https://github.com/x1xhlol/system-prompts-and-models-of-ai-tools
[2] https://arxiv.org/abs/2309.06180

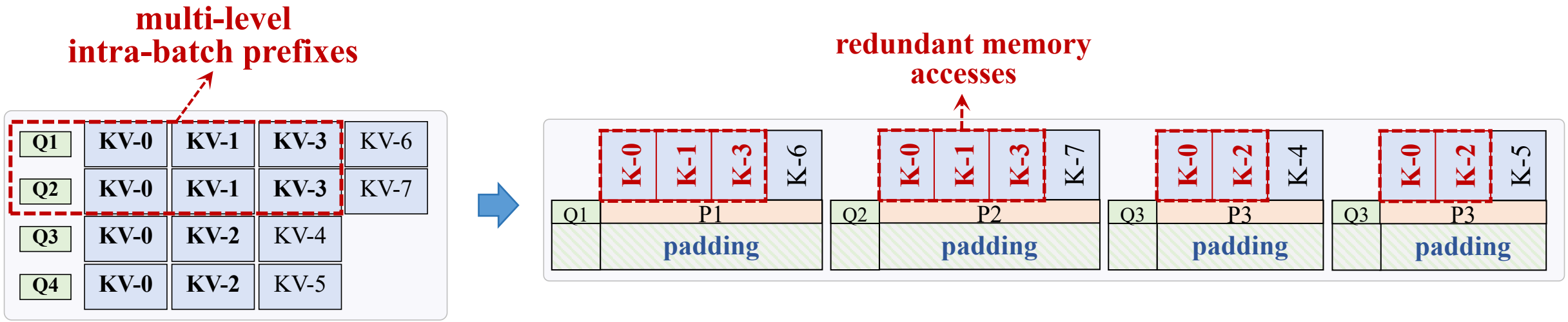
Inefficiency in Attention Execution

- **Existing kernels** underutilize prefix sharing and the memory hierarchy



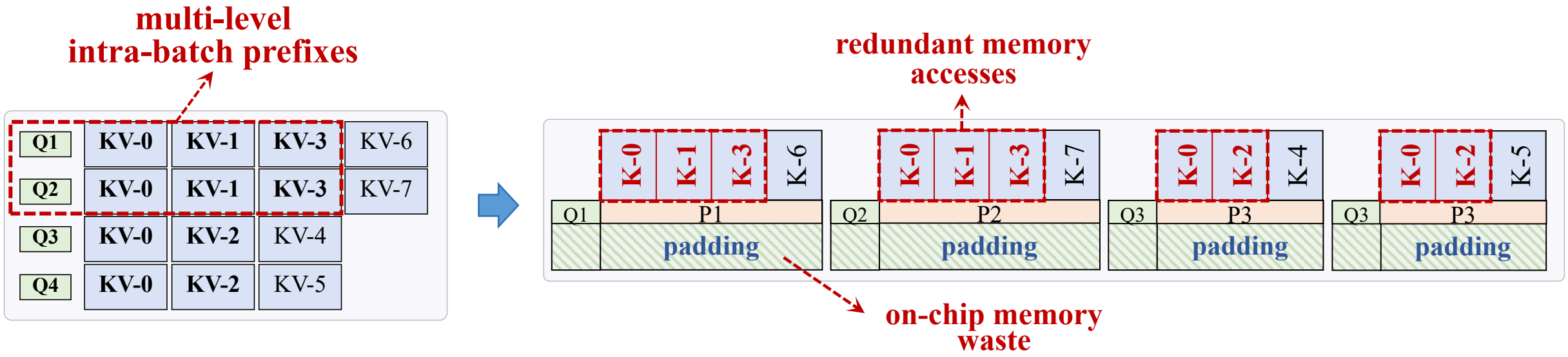
Inefficiency in Attention Execution

- **Existing kernels** underutilize prefix sharing and the memory hierarchy
  - **Shared-prefix KV blocks are reloaded repeatedly** across requests
    - Prefix reuse only reduces memory footprint, but not memory accesses.



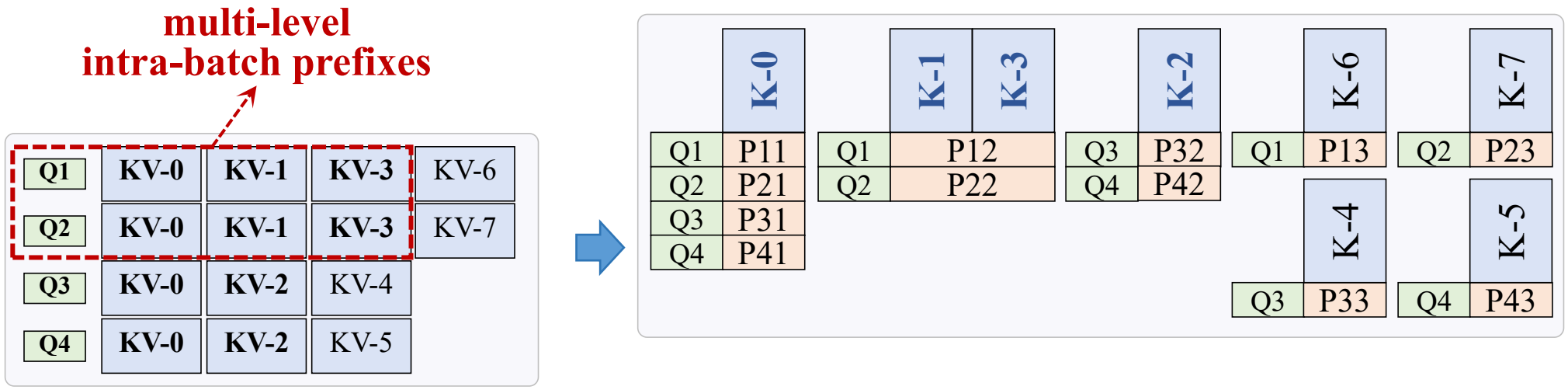
Inefficiency in Attention Execution

- **Existing kernels** underutilize prefix sharing and the memory hierarchy
  - **Shared-prefix KV blocks are reloaded repeatedly** across requests
    - Prefix reuse only reduces memory footprint, but not memory accesses.
  - **Per-request tiling introduces padding**, wasting on-chip memory



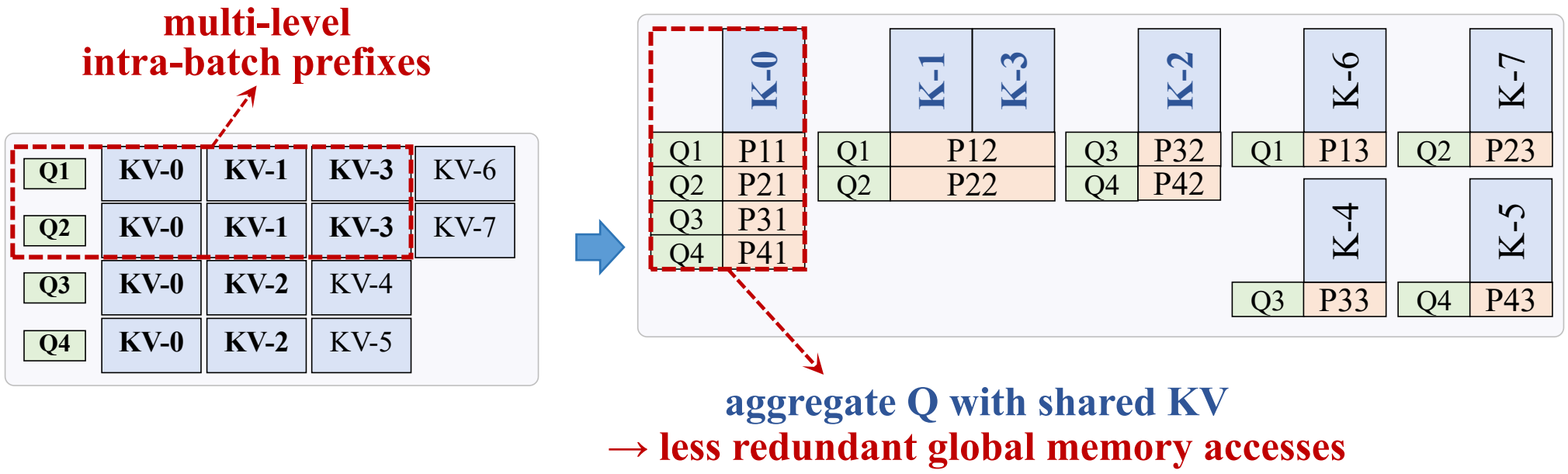
PAT: Prefix-Aware Attention

- **Insight: reuse shared prefixes to reduce redundant KV cache loads.**



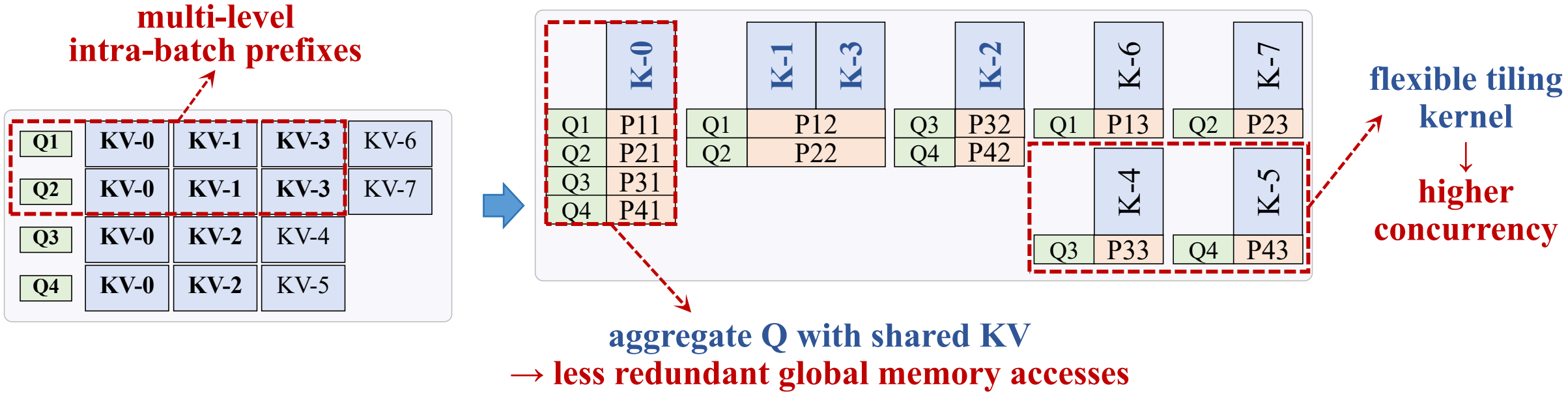
PAT: Prefix-Aware Attention

- **Insight: reuse shared prefixes to reduce redundant KV cache loads.**
  - **In-batch KV sharing**: aggregate queries with shared KV



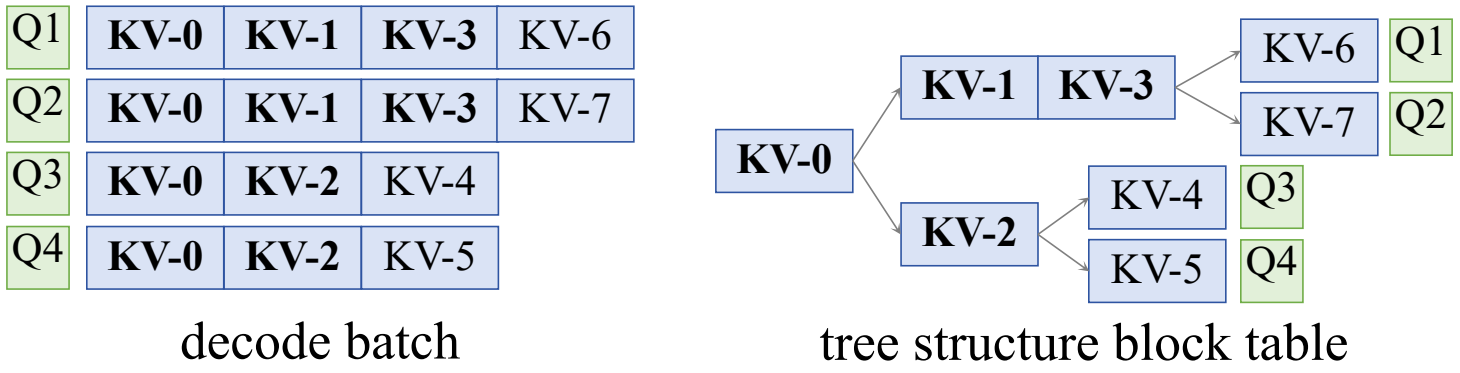
PAT: Prefix-Aware Attention

- **Insight: reuse shared prefixes to reduce redundant KV cache loads.**
  - **In-batch KV sharing**: aggregate queries with shared KV
  - **Resource-efficient kernel**: use multi-tile kernel for higher concurrency



Design-1: Prefix-Aware Pack Scheduler

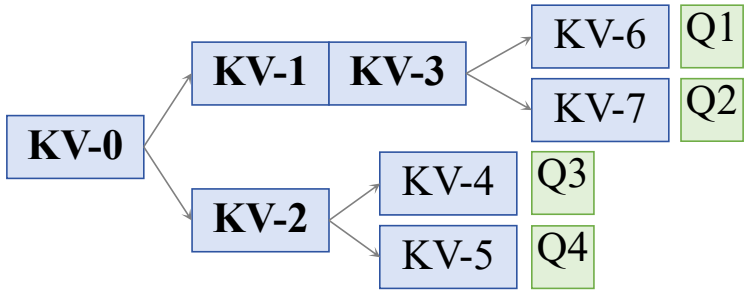
- **Challenge 1**: **dynamic batches + hierarchical prefixes**
  - exponential search space → exact solver is impractical online



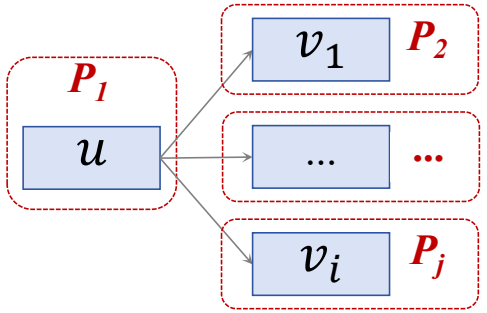
Design-1: Prefix-Aware Pack Scheduler

- **Challenge 1**: **dynamic batches + hierarchical prefixes**
  - exponential search space → exact solver is impractical online
- **Design 1**: **memory-centric, profit-guided tree heuristic**
  - **use profit model** to quickly pick low-memory-access split
  - overlap overhead with **lazy update + async scheduling**

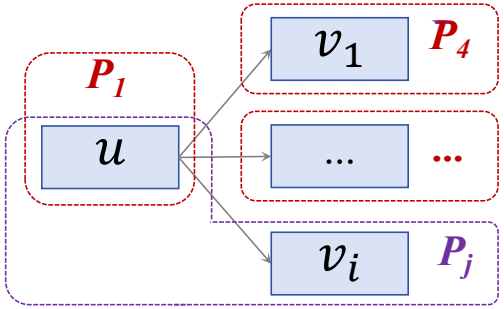
decode batch



tree structure block table



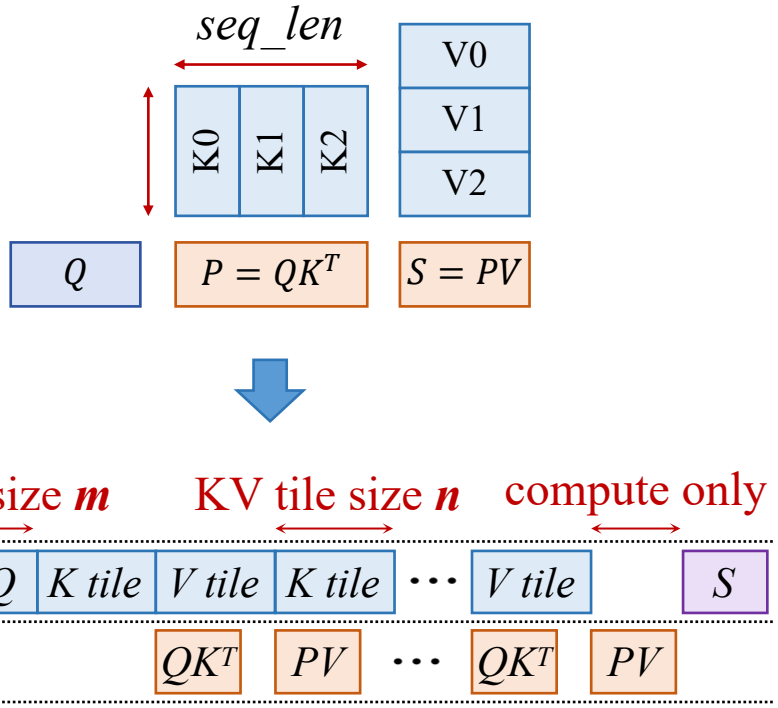
scheme 1: split

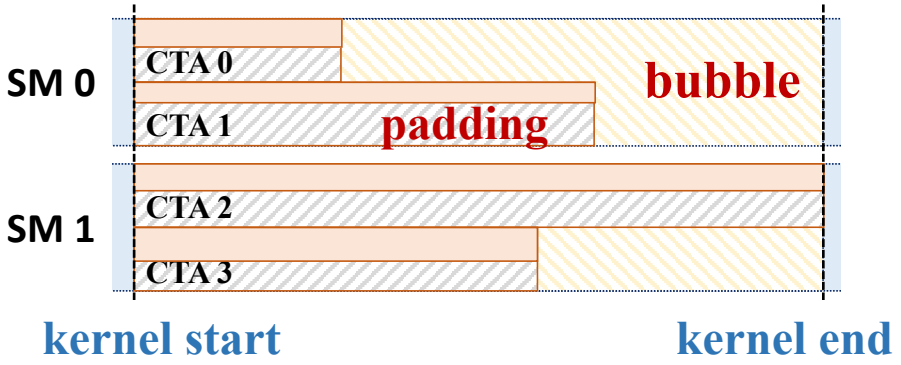


scheme 2: merge

Design-2: Multi-Tile Kernel

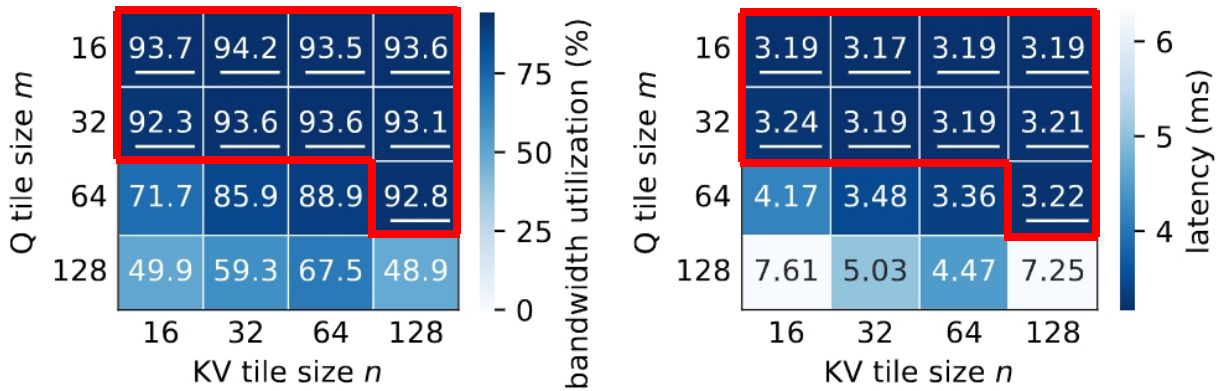
- **Challenge 2: packs have highly dynamic query counts and KV lengths**
  - single fixed-tile kernel suffers from padding and poor utilization





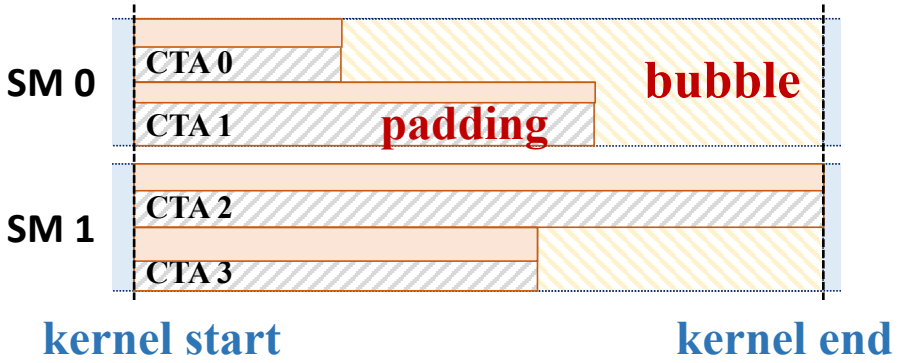
Design-2: Multi-Tile Kernel

- **Challenge 2:** packs have highly dynamic query counts and KV lengths
  - single fixed-tile kernel suffers from padding and poor utilization
- **Design 2:** multi-tile kernel suite + runtime tile selector
  - **Offline:** derive feasible and equivalent tile configurations



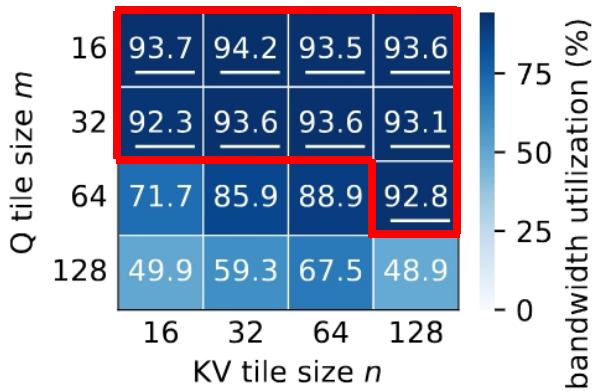
high bandwidth util.

low latency

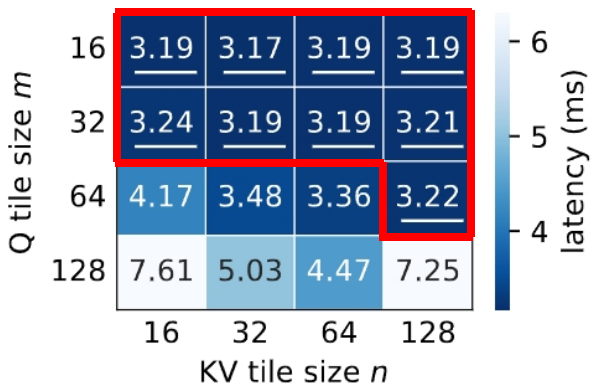


Design-2: Multi-Tile Kernel

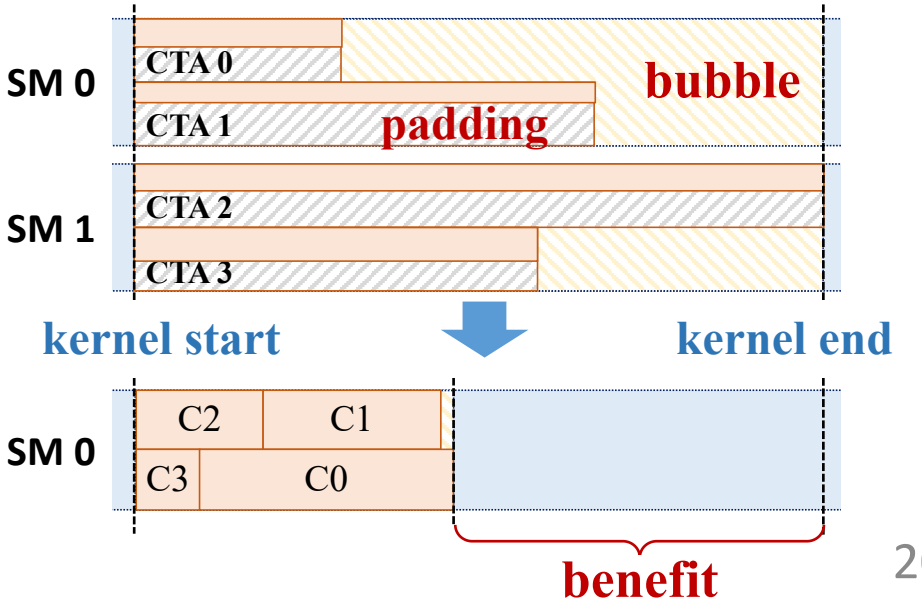
- **Challenge 2:** packs have highly dynamic query counts and KV lengths
  - single fixed-tile kernel suffers from padding and poor utilization
- **Design 2:** multi-tile kernel suite + runtime tile selector
  - **Offline:** derive feasible and equivalent tile configurations
  - **Online:** select tile sizes to reduce tail bubbles from long-KV CTAs



high bandwidth util.



low latency



Implementation & Evaluation

Tianjin Key Laboratory of Advanced Networking

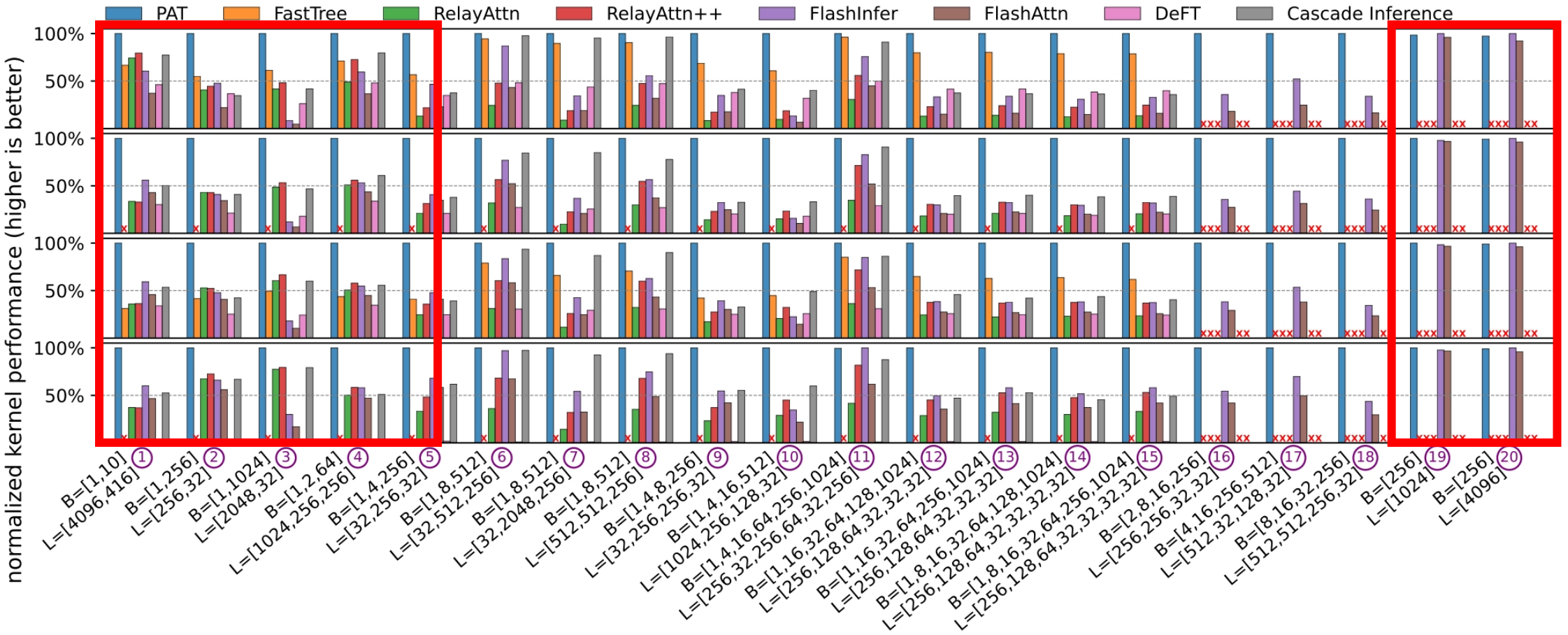
- **Implementation**
  - CUTLASS/CuTe kernel + a Python plugin for vLLM

```
VLLM_ATTENTION_BACKEND=PAT vllm serve <model>
```

- **Evaluation Setup**
  - **Hardware**: A100-80GB GPU & H100-80GB GPU
  - **Models**: Llama-3-8B, Qwen3-8B, Qwen3-30B-A3B, Qwen2.5-72B
  - **Baselines**: FlashAttention, FlashInfer, FastTree, RelayAttention ...
  - **Datasets**: Mooncake Trace (toolagent), BurstGPT (conversation)
- **Open-Sourced: https://github.com/flashserve/PAT**

Evaluation: Kernel Performance

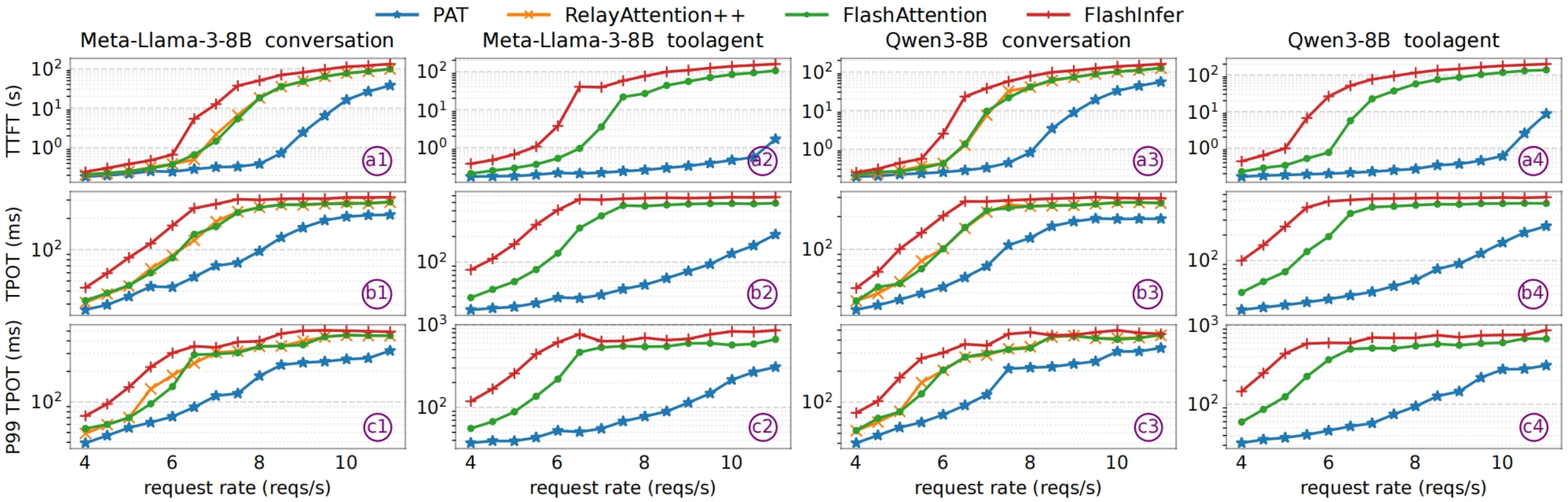
- **PAT reduces attention latency by 34%-77%** with shared prefixes
  - PAT achieves consistent or superior performance without shared prefixes



Evaluation: E2E Serving Performance

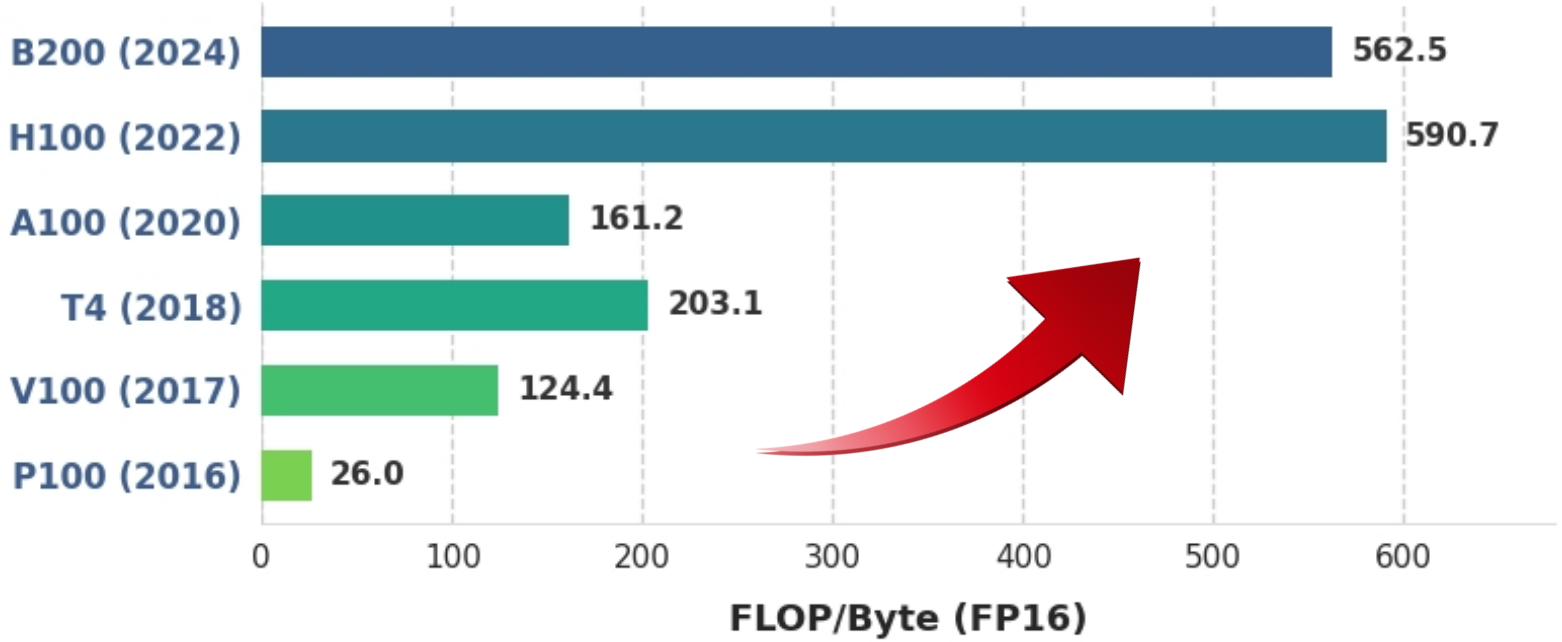
Tianjin Key Laboratory of Advanced Networking

- **PAT reduces mean TPOT by 17%-93%** under real-world workloads



Discussion

- **Discussion**
  - **Why it matters**: **PAT aligned with hardware trends**
    - V100 → B200: 124 → 563 FLOP/Byte
  - **When it helps more**: long system prompt / beam search / RL rollout
  - **When it helps less**: small batch / weak prefix sharing / KV quantization



Conclusion

- **Takeaway**: PAT reduces redundant KV loads via **prefix-aware attention**, and improves hardware utilization via **multi-tile kernel** for LLM decoding
- **Future Work**
  - **PAT for RLHF**: Extend PAT to RLHF rollouts with improved load balancing.
  - **Hardware Adaptation**: Adapt PAT to emerging architectures (Hopper, NPU).

https://github.com/flashserve/PAT

天津大学 赵志新

PAT: Accelerating LLM Decoding via Prefix-Aware Attention with Resource Efficient Multi-Tile Kernel

ASPLOS 2026

Jinjun Yi†, **Zhixin Zhao†**, Yitao Hu*, Ke Yan, Weiwei Sun,
Hao Wang, Laiping Zhao, Yuhao Zhang, Wenxin Li, Keqiu Li

https://arxiv.org/abs/2511.22333
https://github.com/flashserve/PAT