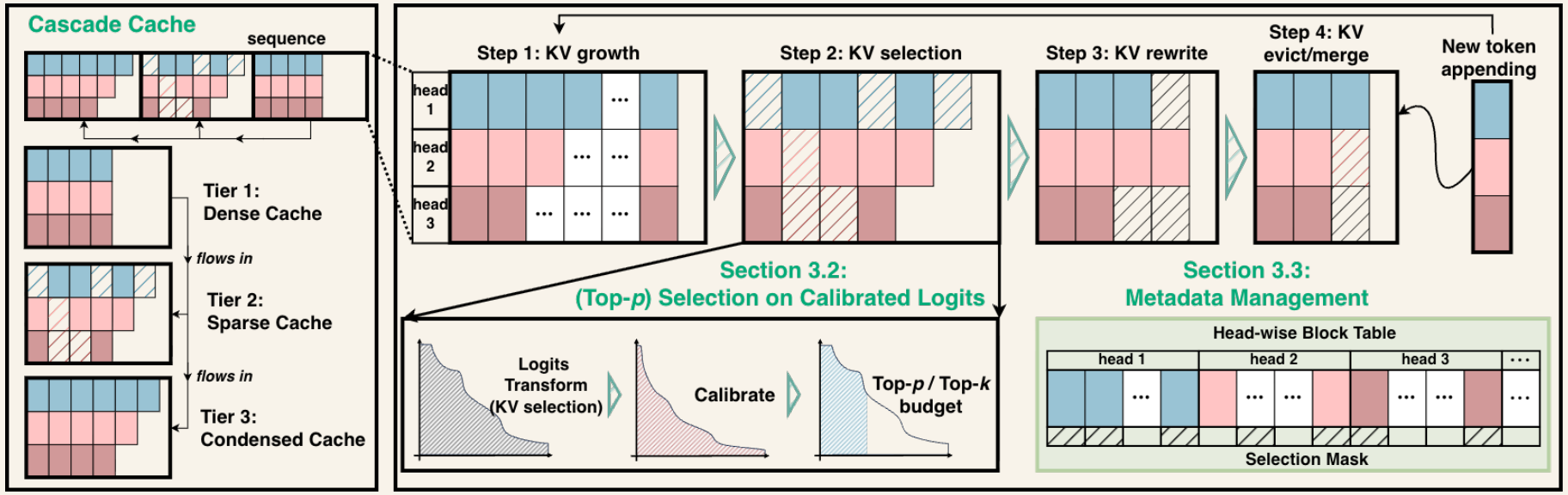
HARD-KV: Head-Adaptive Regularization for Decoding-time KV Compression

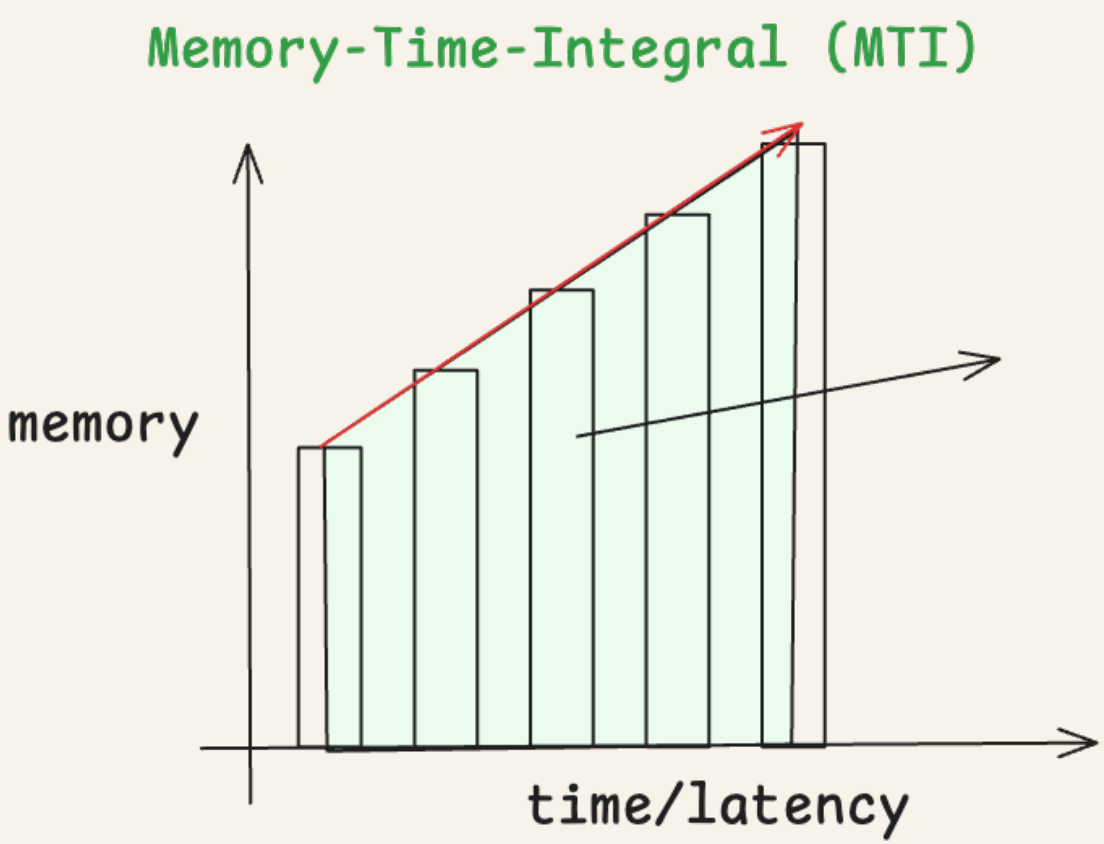
Yuxuan Yang[1], Feiyang Ren[1], Bowen Zeng[1], Dalin Zhang[2], JinPeng Chen[3], Gang Chen[1], Huan Li[1]

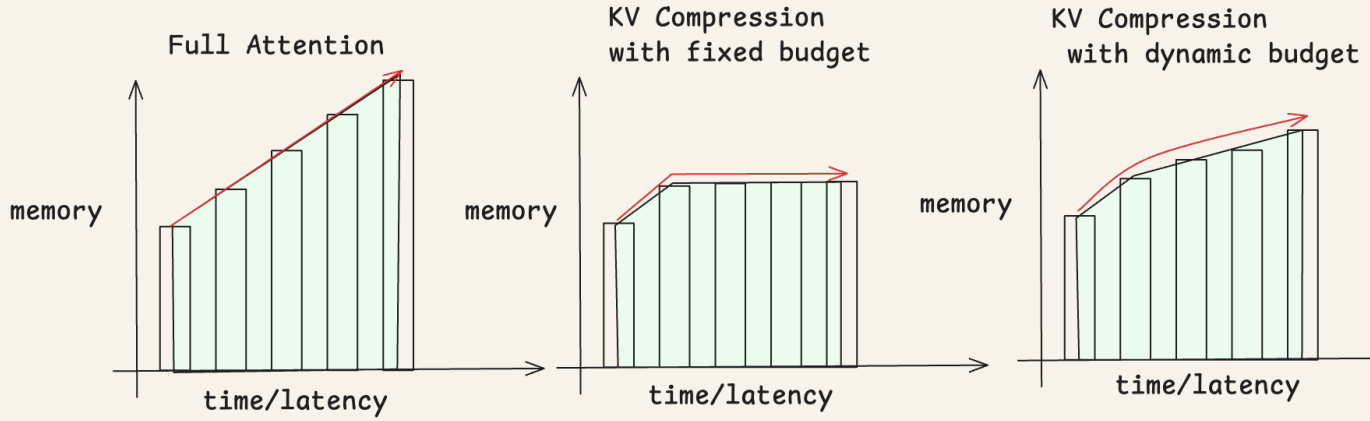


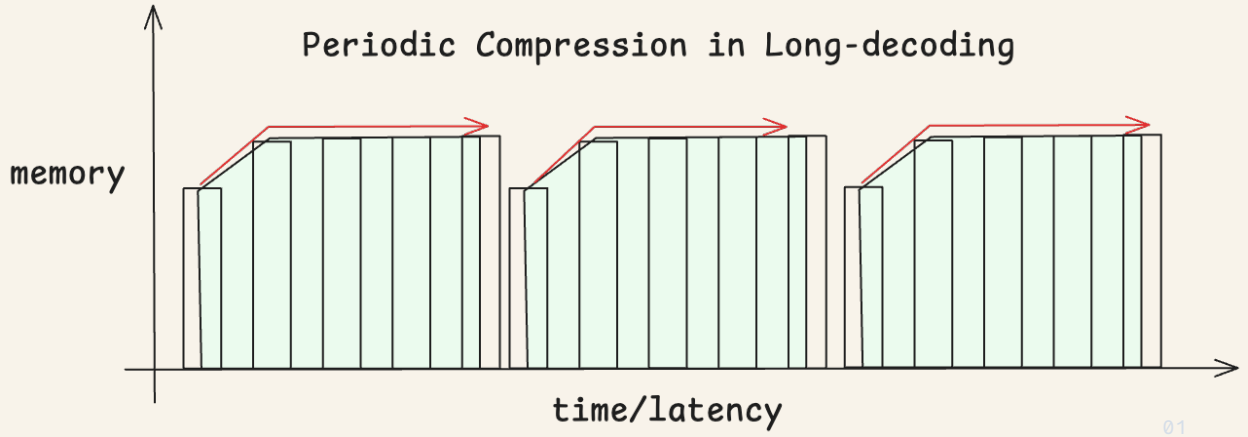
[1]The State Key Laboratory of Blockchain and Data Security, Zhejiang University
[2]Space Information Research Institute, Hangzhou Dianzi University
[3]School of Computer Science (National Pilot Software Engineering School), BUPT

Background: Decoding-time Compression

Memory-Time-Integral: the standing-point to think of latency-throughput trade-offs

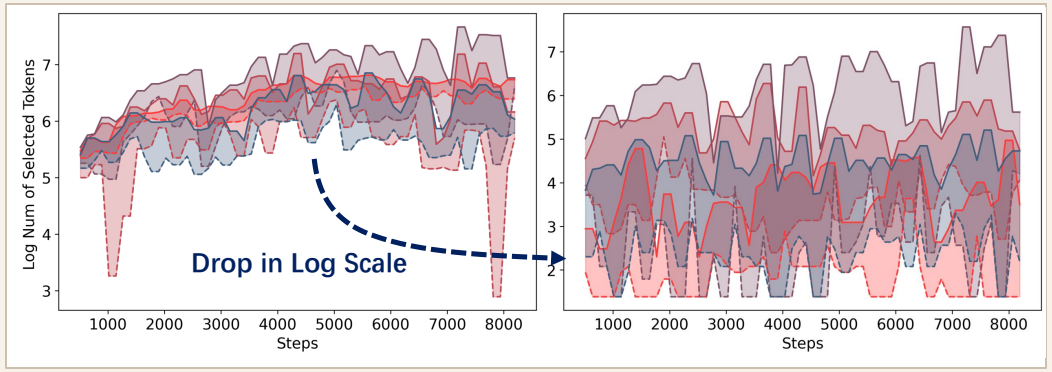


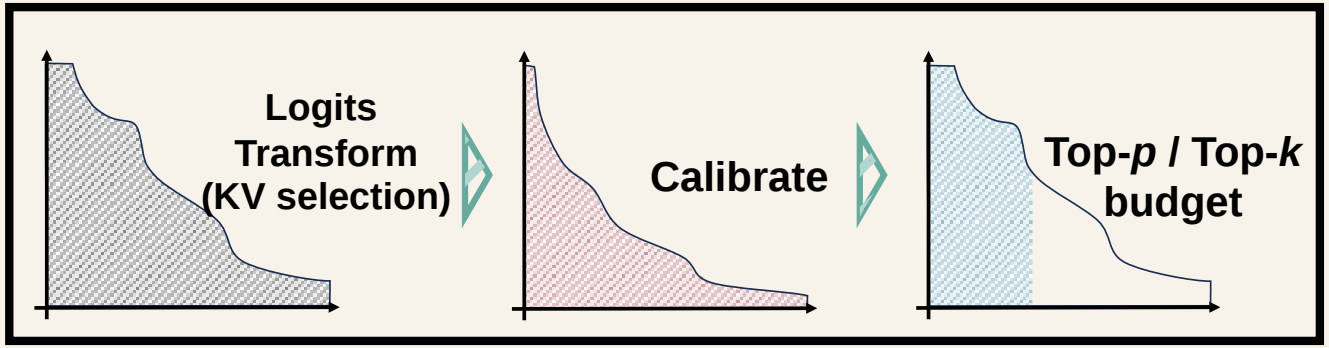




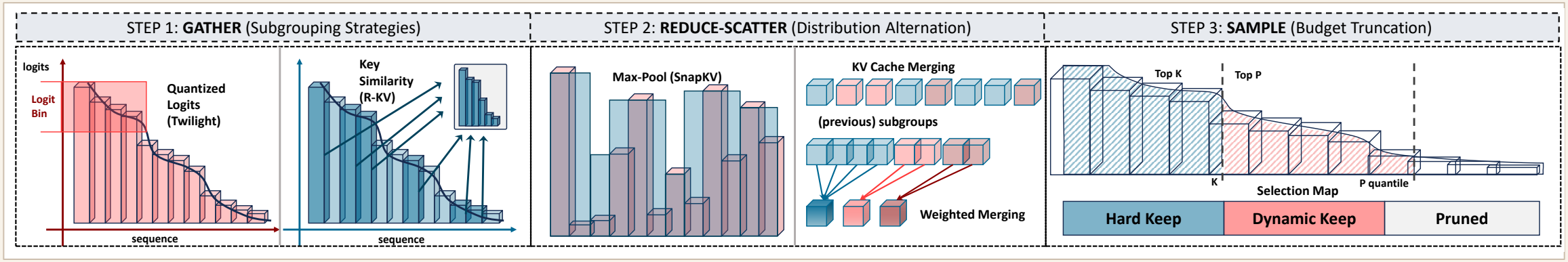
Natural Dynamic Budget: Top-P budget

Selection collapse





If a P mass corresponds to K selections in raw attention, we keep (roughly) same number of selections for different heuristics



Challenge: Static-Dynamic Mismatch

Algorithm side

Head-adaptive Top-p selection lets each head keep the tokens it needs.

Budgets fluctuate by context, layer, and head; this is exactly what protects long reasoning trajectories.

Top-p budget

probability mass budget

Head-Adaptive

fine-grained allocation

STATIC-DYNAMIC MISMATCH

System side

PagedAttention, continuous batching, and CUDA Graphs want stable block layouts.

Irregular per-head indices create metadata work, fragmented loading, and graph-breaking dynamic shapes.

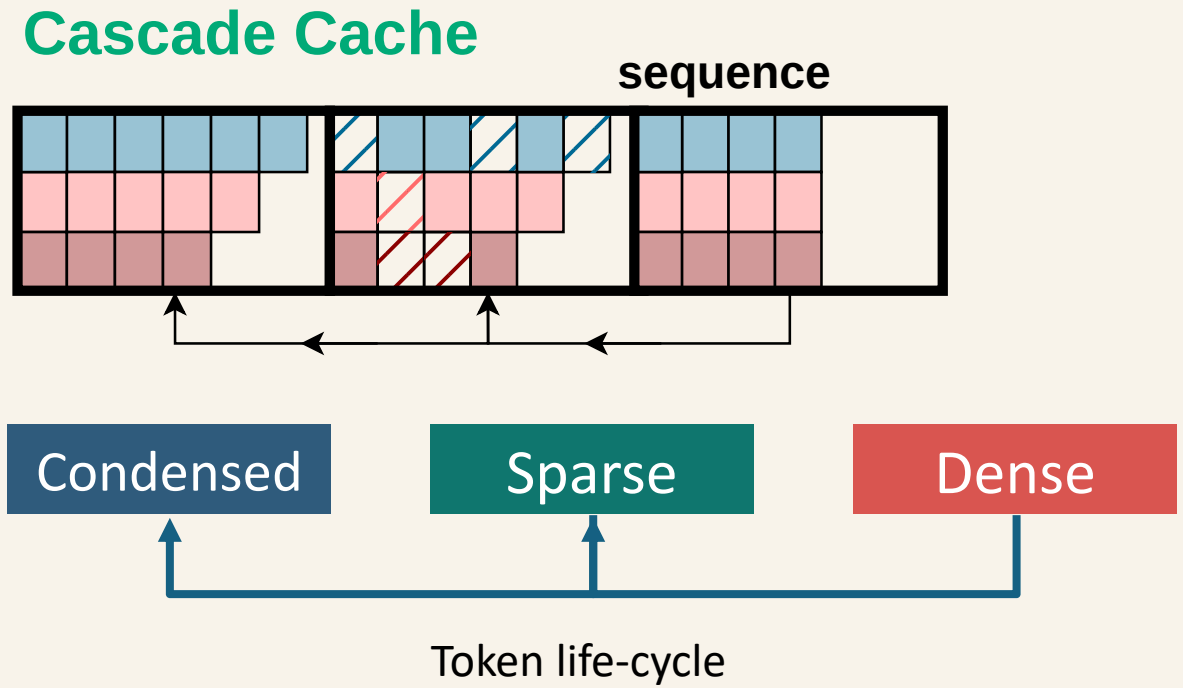
CUDA Graphs

static execution shape

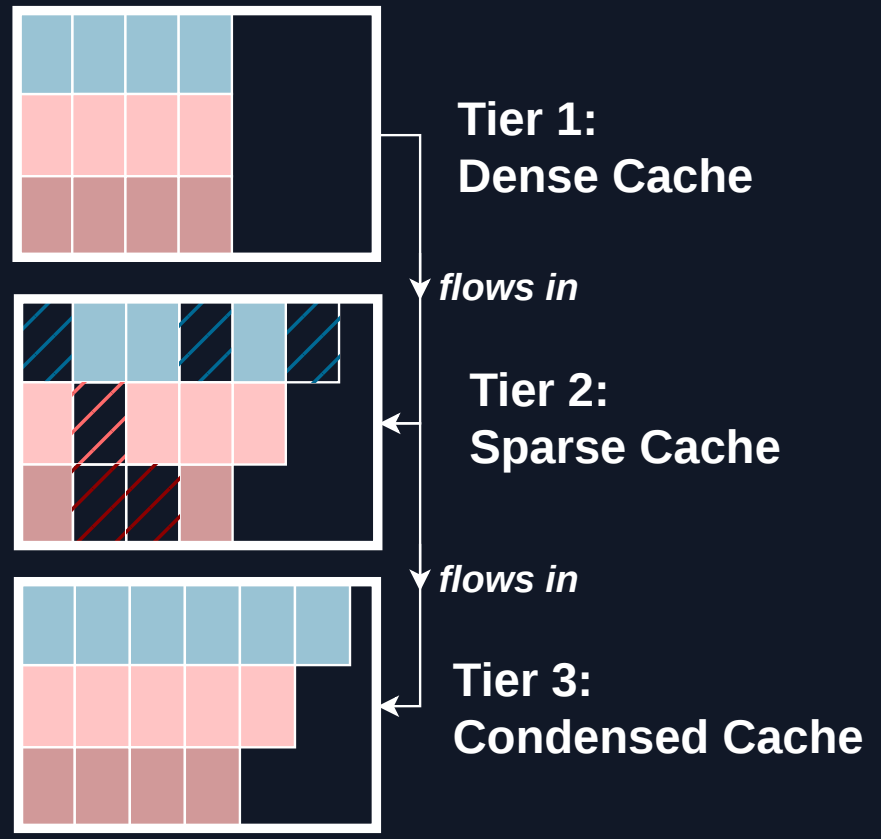
Regular Page Index

block table contract

Intuition: Sequential-hybrid of linear and full KV cache

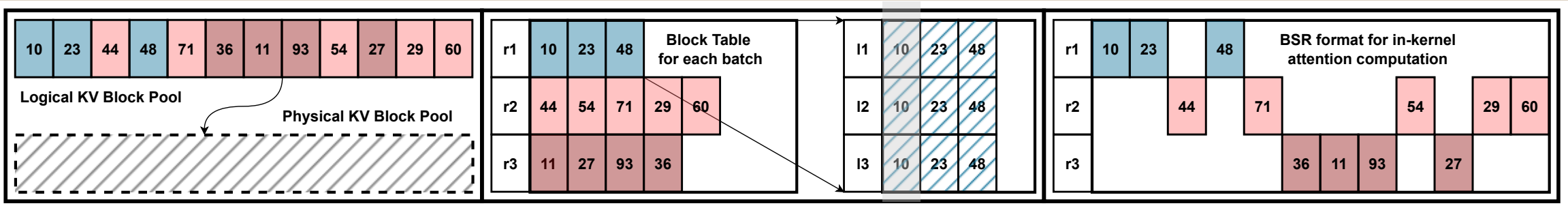


Cascade Cache: Three-tier Contract



Inference Engine Adaptations

Three metadata levels



Global: block allocation
allocated block ids:

[total_num_allocated_blocks]

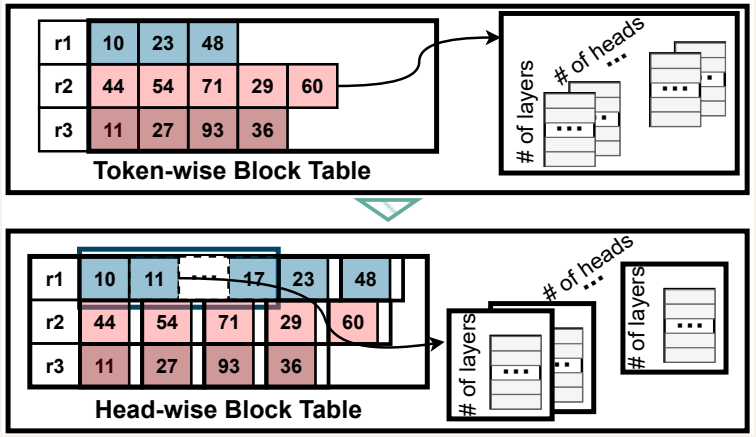
Batch: block table
request id maps to corresponding block id
[batch_size, max_num_blocks_per_req]

Important Note:
One single preparation of metadata will use same indices for every computation afterwards (for each layer/head).

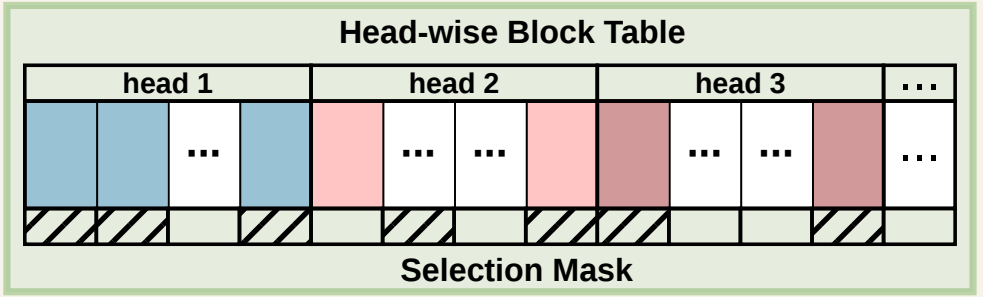
Kernel: BSR-based tiling

flashinfer will handle tiling information based on BSR format.

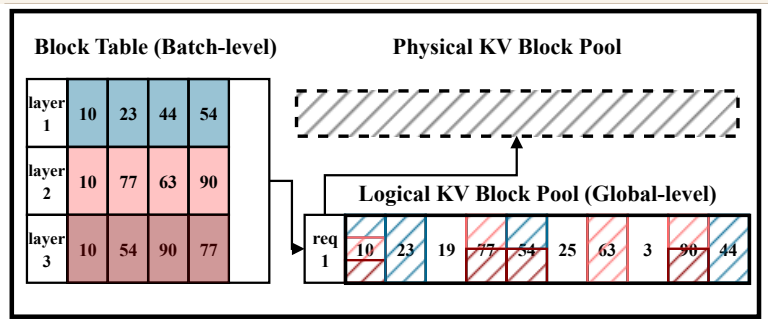
Marginal adjustments with head-wise block table

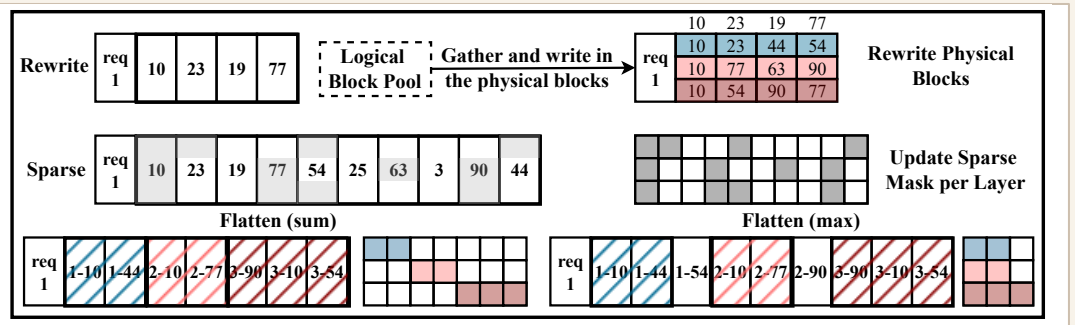


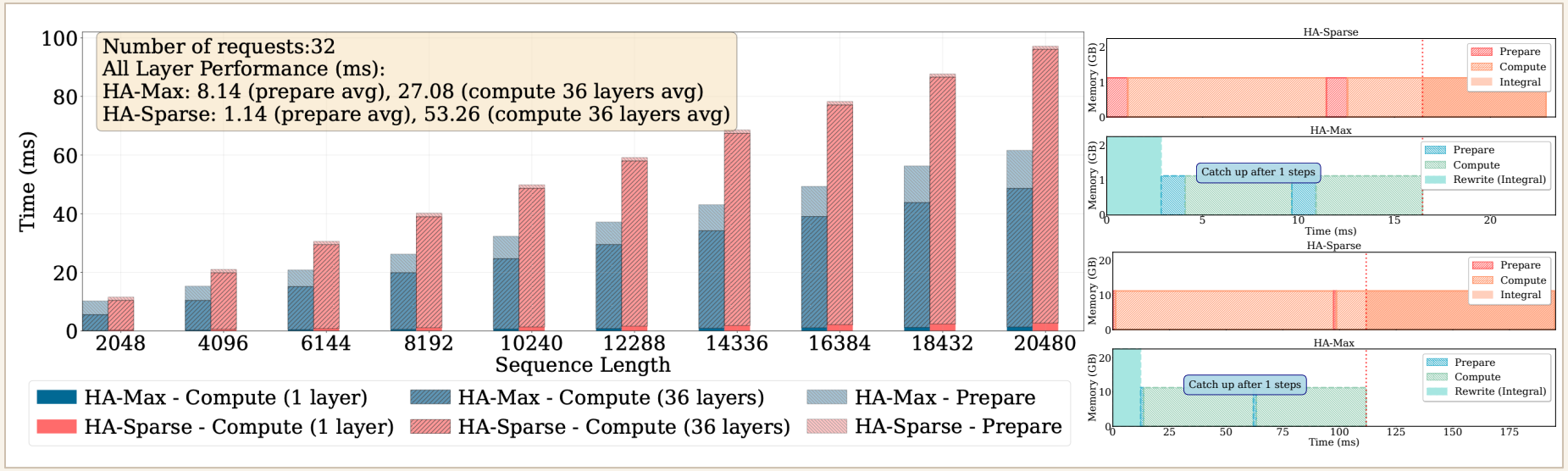
Supported by FlashInfer kernel library



Index Regularization: unifying indices across layers







Rewrite

contiguous cache

read selected blocks, write regularized physical layout

Sparse load

mask at kernel

regularize indices by carrying unused cache positions

Flatten

head/layer index

choose max or sum depending on cost trade-offlayout

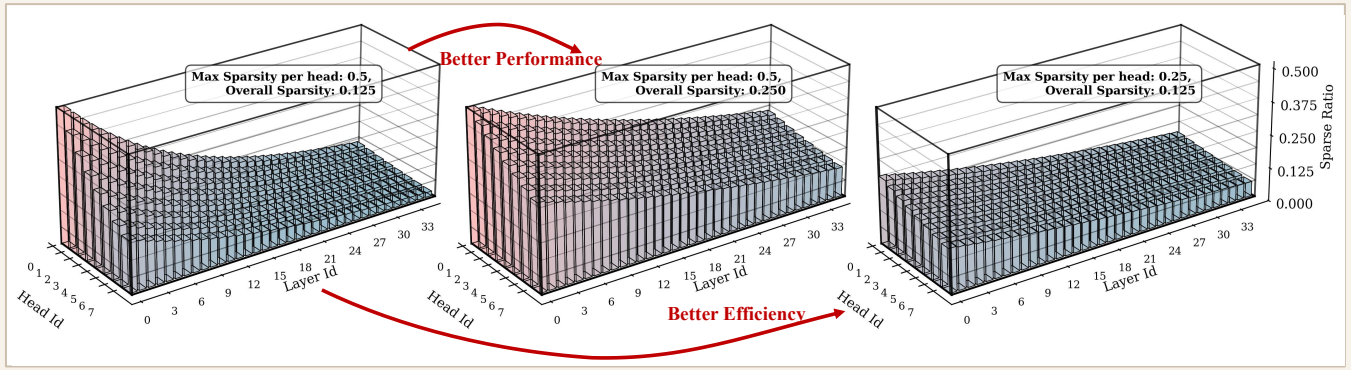
Rewrite: a naïve but efficient approach
Can catch-up counterparts in 1 step by MTI

Experiment Takeaway

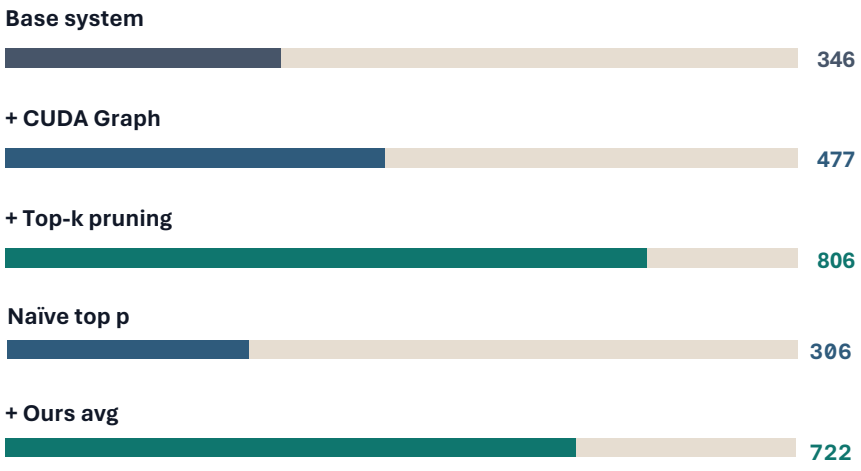
1.5x
performance gain

2x
throughput improvement

90%+
sparsity utilization



Throughput breakdowns under dynamic budget



Takeaways on tuning performances

Use k to cap physical cache growth;
use p to preserve probability mass. Hard-KV makes this two-knob trade-off executable.

Thank you