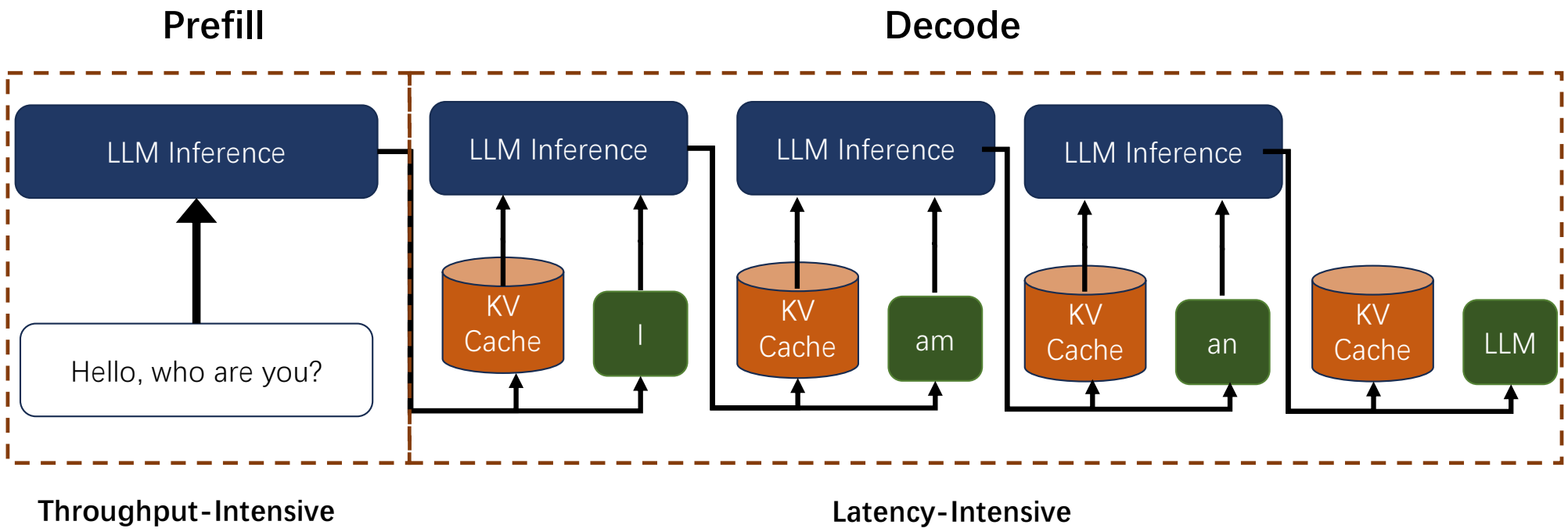
SolidAttention: Low-Latency SSD-based Serving on Memory-Constrained PCs

Xinrui Zheng, Dongliang Wei, Jianxiang Gao, Yixin Song, Zeyu Mi, Haibo Chen

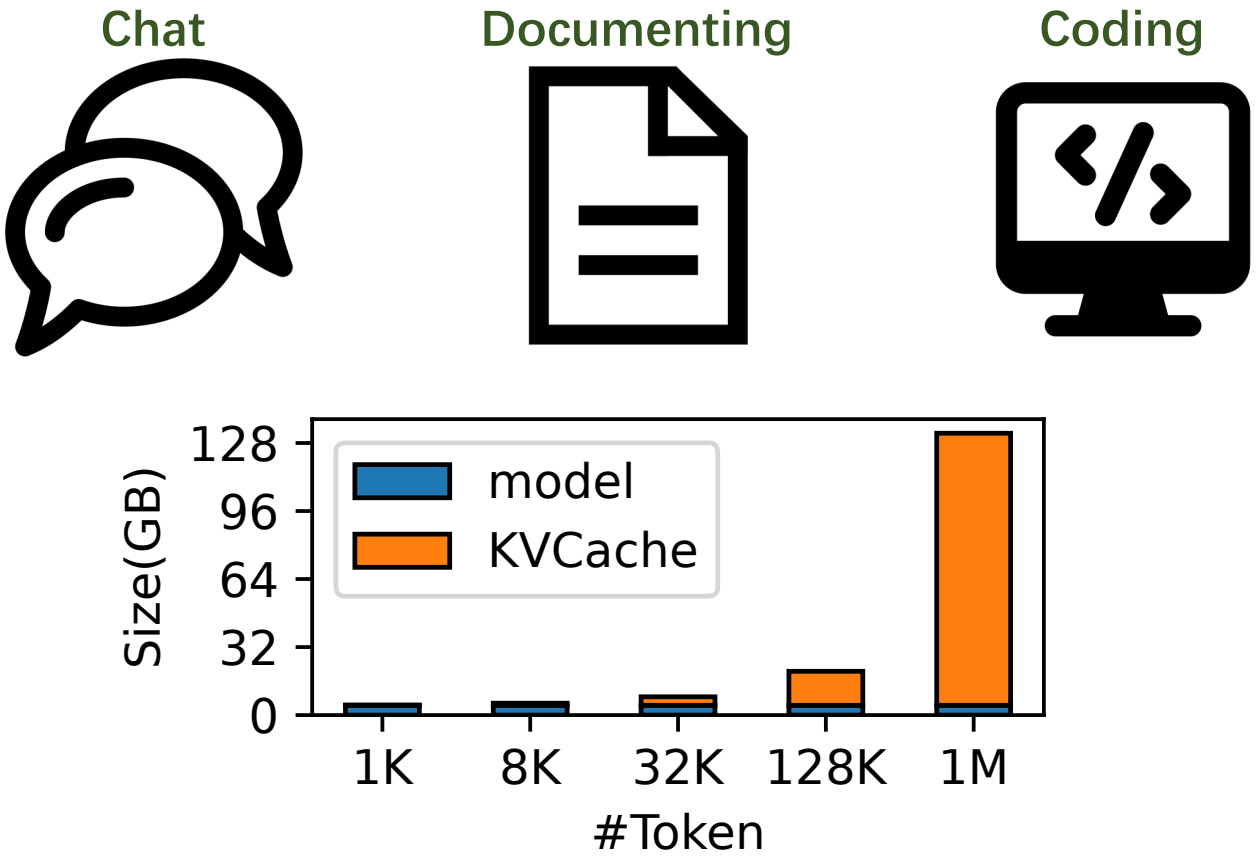
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

IPADS
INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

LLM Inference



LLM Inference – Memory Occupation



LLM Inference – Memory Occupation

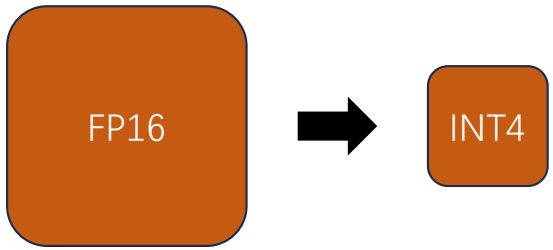
Most shipping PCs are equipped with

- 8-16GB DDR Memory
- integrated GPUs (iGPU) or entry-level discrete GPUs (dGPU)

Not Enough For Long-Context Inference !!!

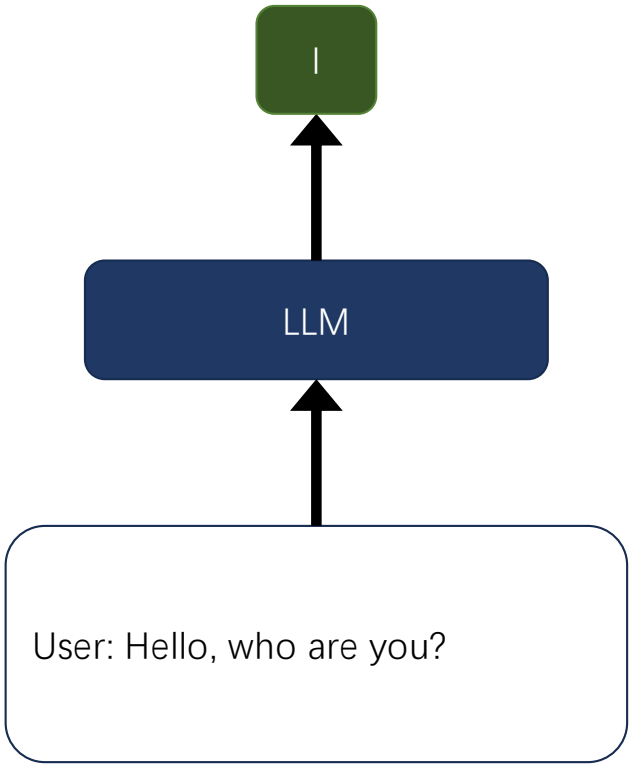
LLM Inference – Limitation on Memory-Constrained PCs

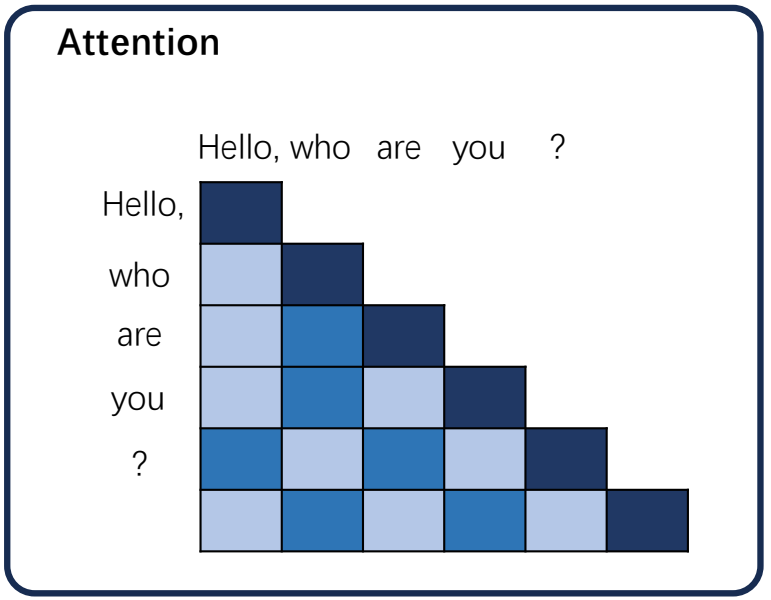
Quantize KV Cache



Model Accuracy Degradation

LLM Inference – Attention Sparsity

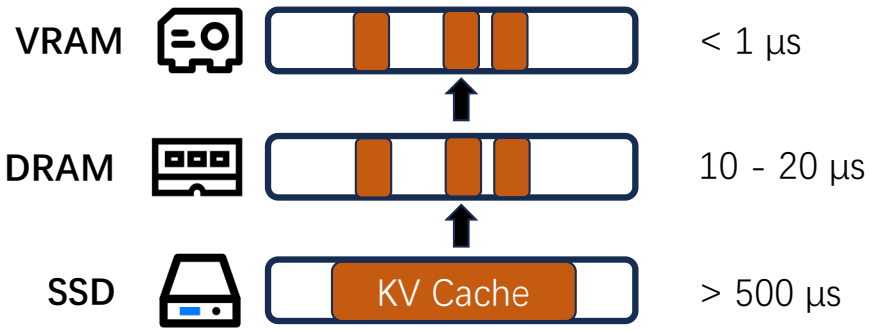




Different Tokens Focus on Different Context Information

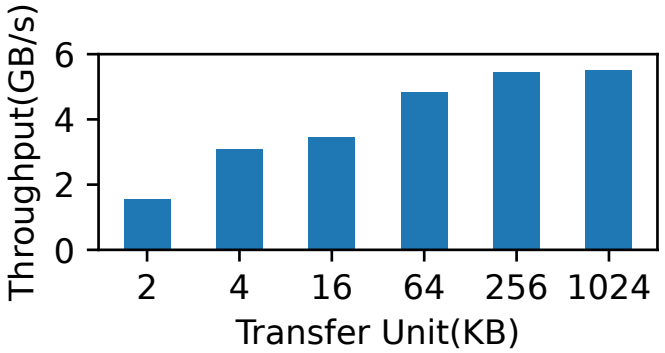
LLM Inference – Attention Sparsity

Offload KV Cache to SSD



Long Inference Latency

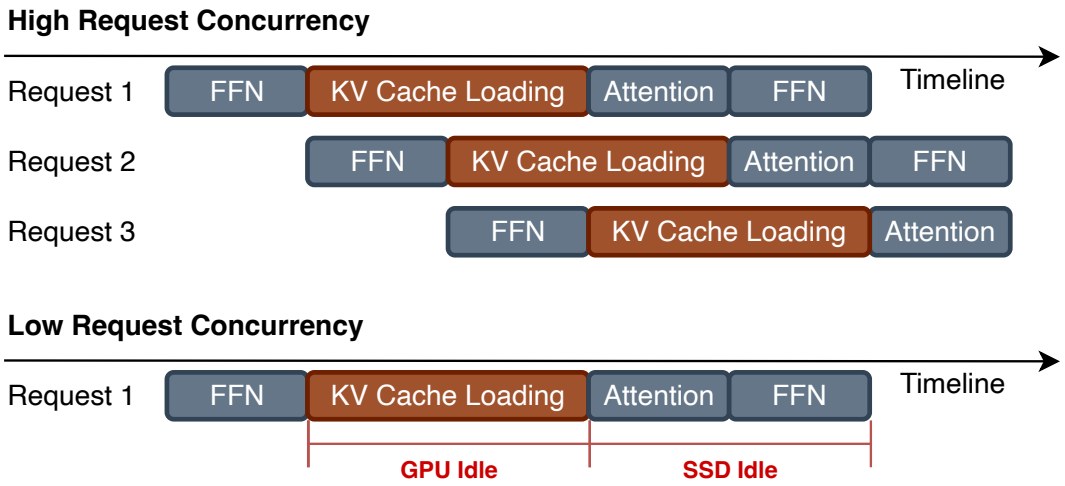
Tiny & Random Reads



Low SSD Throughput

LLM Inference – Overlap I/O & Computation

Pipeline Multiple Request's Inference



No Enough Request Concurrency

Co-Design Attention Sparsity & Storage Management

Consolidates multiple KV pairs into a block as the basic transfer unit.

This transforms irregular data access patterns into coarse-grained sequential ones.

Preselects and prefetches critical KV blocks

This provides sufficient time for computation-I/O overlap before attention computation.

Challenge – Accuracy Loss

Block-Wise Attention Sparsity: KV Cache is Selected in Block Granularity

Smaller blocks

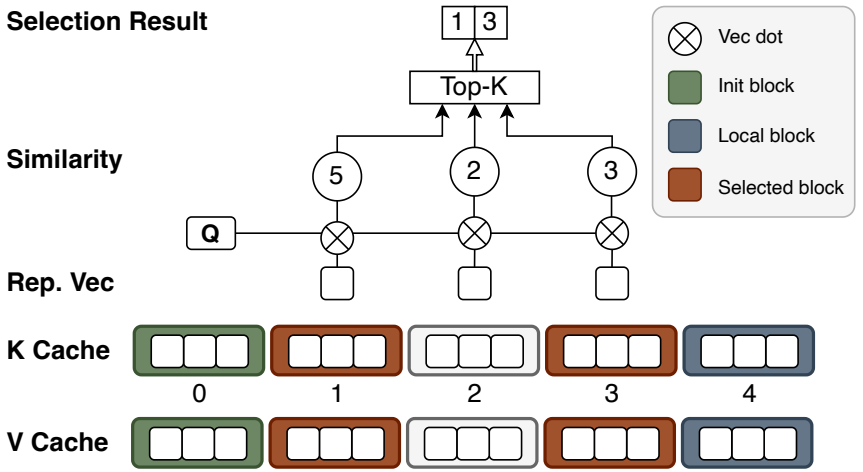
Preserve Accuracy

Fine-Granularity I/O Operations

Larger block

Coarse-Granularity I/O Operation

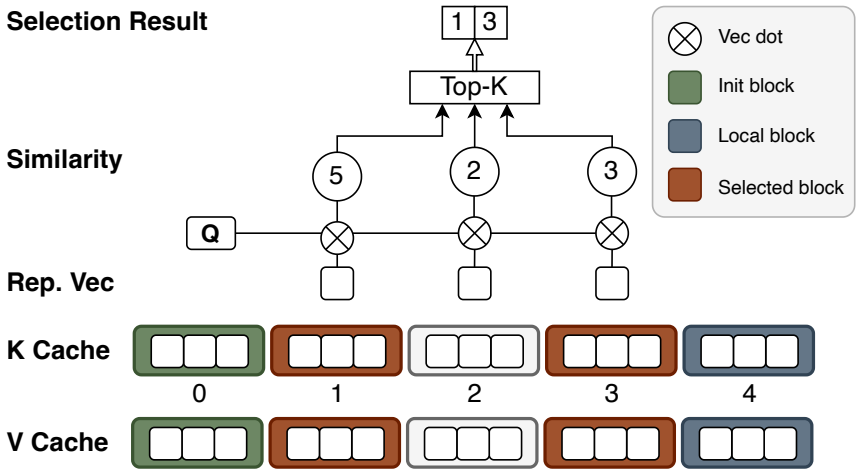
More Tokens Compressed to A Vector



Challenge – Prefetching Indeterminacy

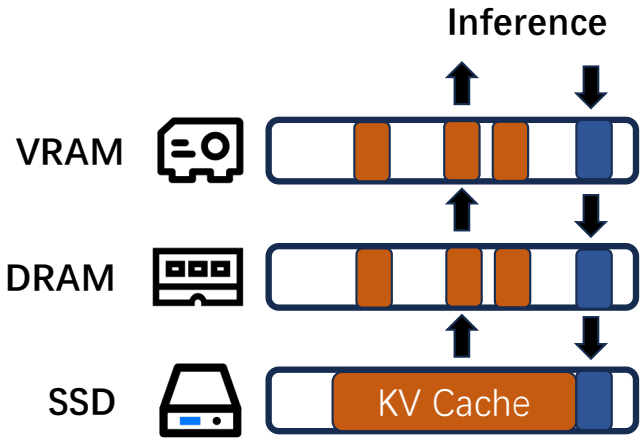
Token selection depends on dynamic computation during inference

The selection results remains unknown before attention computation



Challenge – Data Inconsistency

Computation-I/O overlap introduces concurrent accesses to the KV cache.



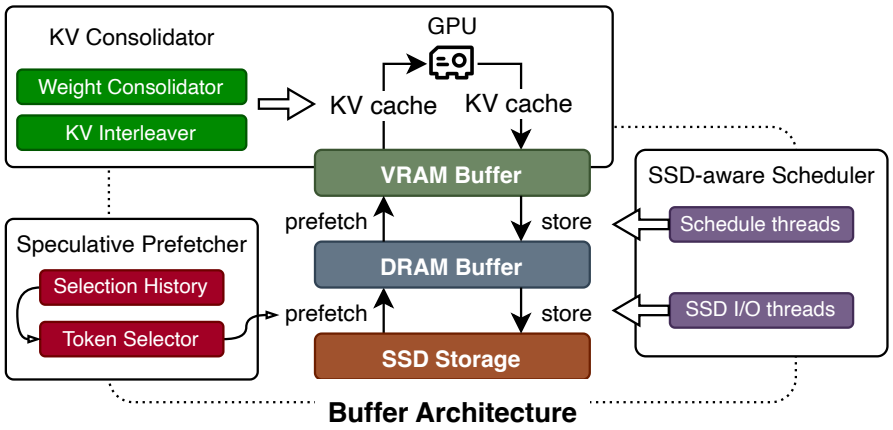
Add buffers

☹ More memory occupation

Lock or serialize the tasks

☹ Long idle time

Design – Architecture



Buffer Architecture

How to fully utilize the SSD bandwidth?

KV Consolidator: consolidate multiple KV cache entries to enlarge the SSD transfer unit.

How to reduce the KV cache loading latency?

Speculative Prefetcher: prefetch KV cache according to the temporal locality of importance.

How to efficiently overlap I/O and computation?

SSD-aware Scheduler: orchestrates fine-grained computation-I/O overlap according to their data dependencies.

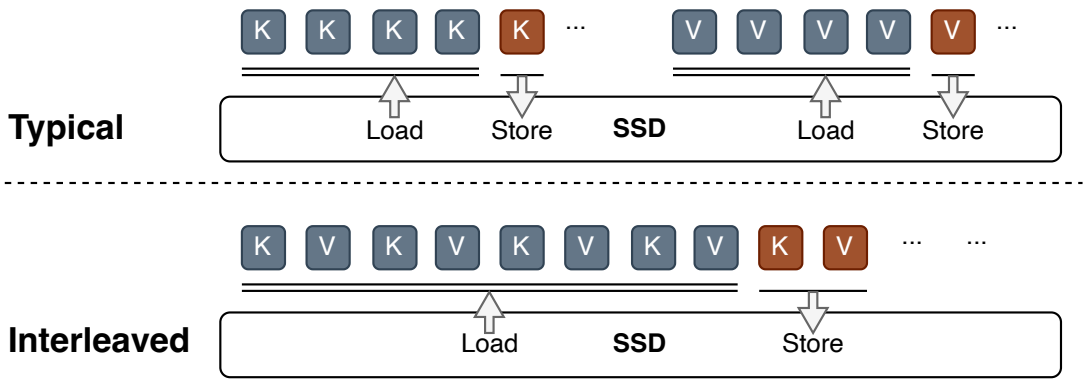
Design – KV Consolidator

Typically

K cache and V cache are stored and managed separately.

Interleave K & V cache in token granularity

The K and V cache are loaded and stored as an integration.



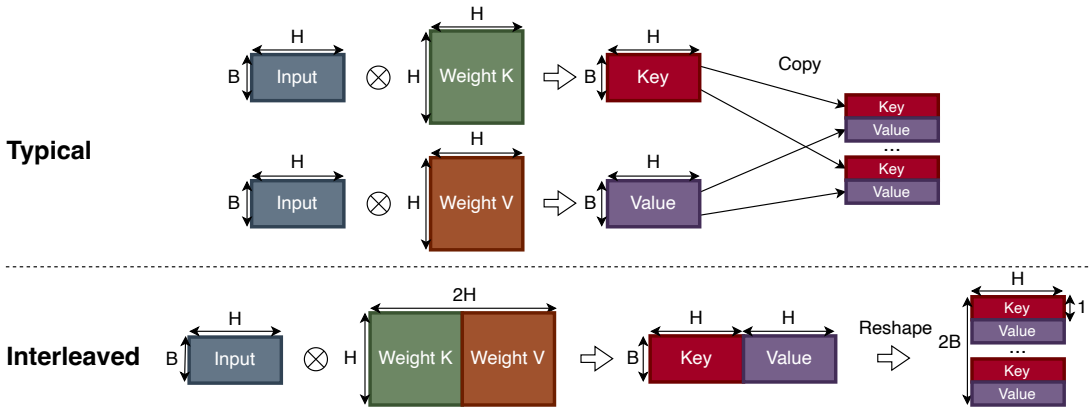
Design – KV Consolidator

Typically

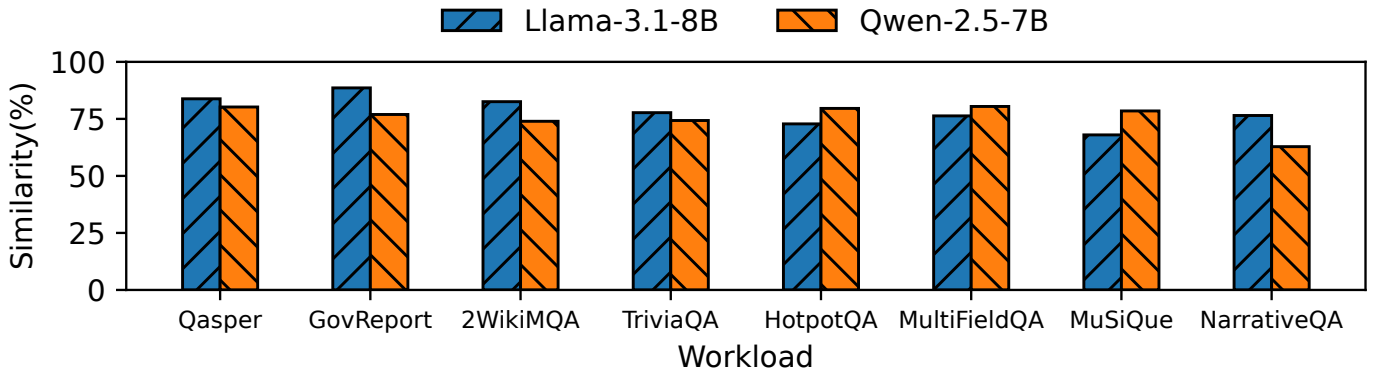
K and V are calculated through different projections.

Offline fusion of K&V projection matrices

K and V are calculated through one projection directly without costly reordering.



Design – Speculative Prefetcher



KV block Selection exhibits **temporal locality (~ 80%)**

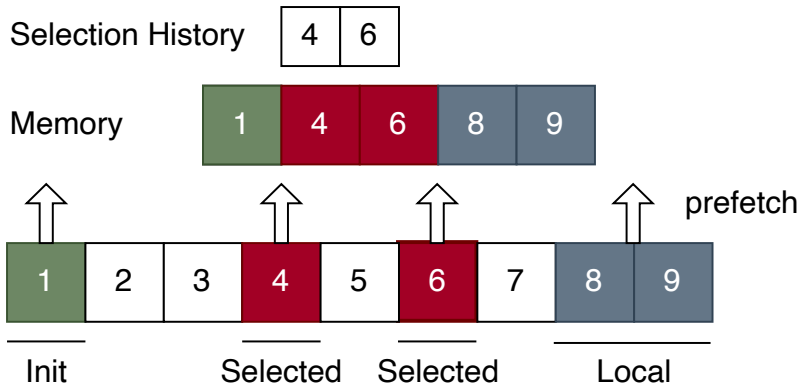
Design – Speculative Prefetcher

Prefetch init blocks and local blocks

These blocks are definitely involved in the sparse attention computation.

Prefetch selected KV blocks according to selection history

These blocks are selected dynamically during inference.

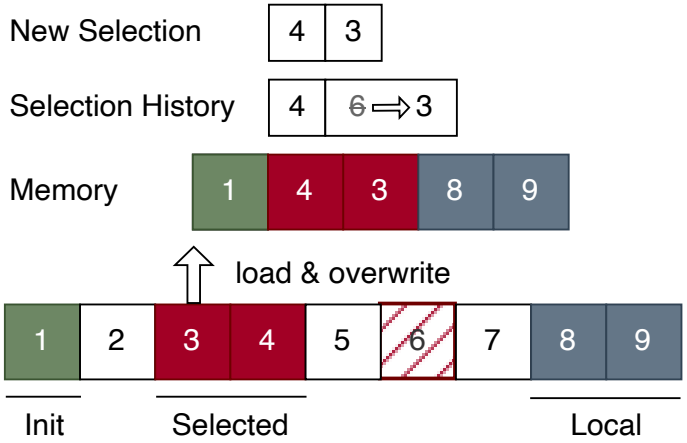


Design – Speculative Prefetcher

Keep accuracy without costly reordering

- Selected blocks are selected and loaded after their scores are calculated.

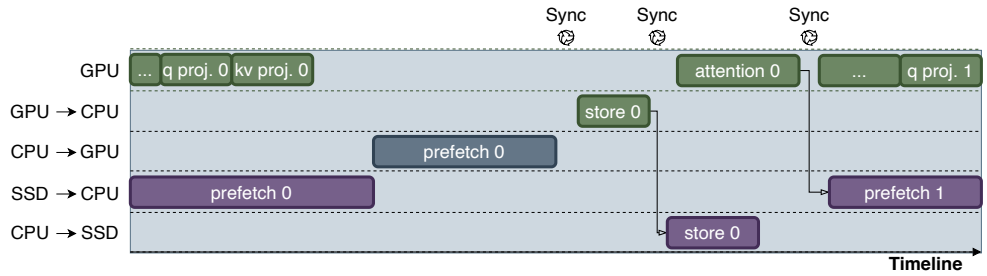
- Blocks prefetched by mistake are directly overwritten. **(Out-of-order KV blocks does not impact attention computation)**



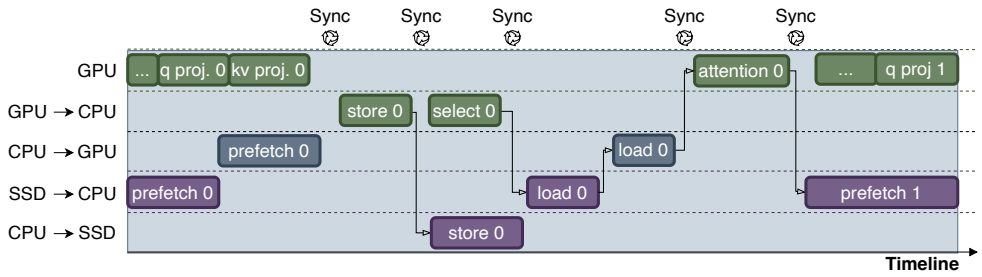
Design – SSD-aware Scheduler

Dynamic attention sparsity introduces

- **Frequent long-latency SSD accesses**
- **Massive synchronization points**



Typical implementation without attention sparsity



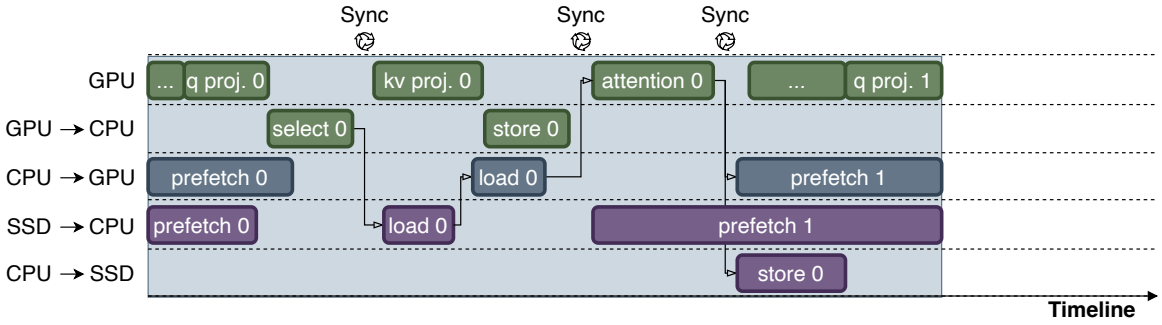
Naïve implementation with attention sparsity

Design – SSD-aware Scheduler

Schedule operations in fine granularity

Split inference into microtasks:

- Prefetch: prefetch KV blocks
- Load: load missed KV blocks
- Save: save newly generated blocks
- ...

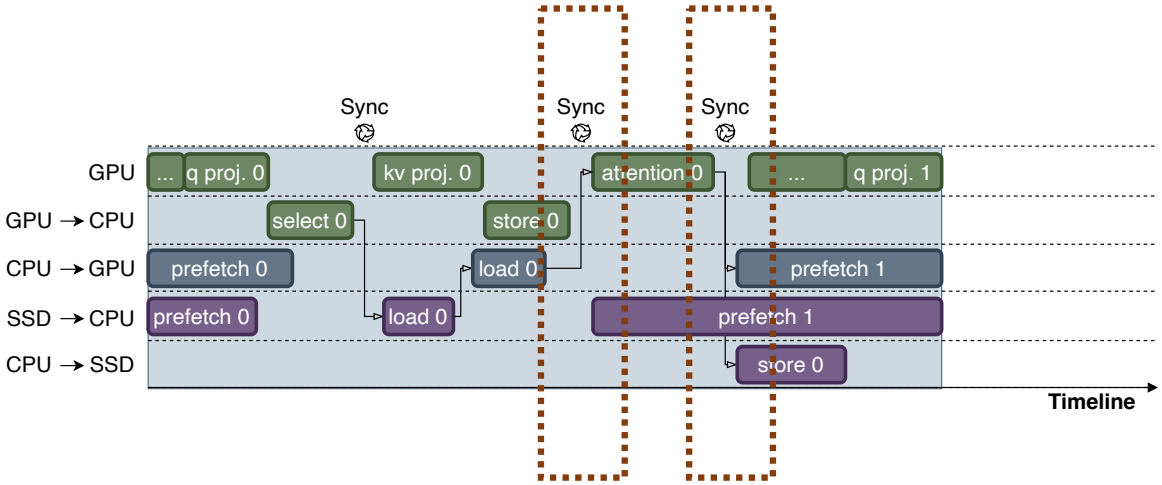


Schedule microtasks according to the data dependency

Design – SSD-aware Scheduler

Reuse Synchronization Points

Operations not in the critical path are grouped with critical tasks and share the same synchronization points



Evaluation

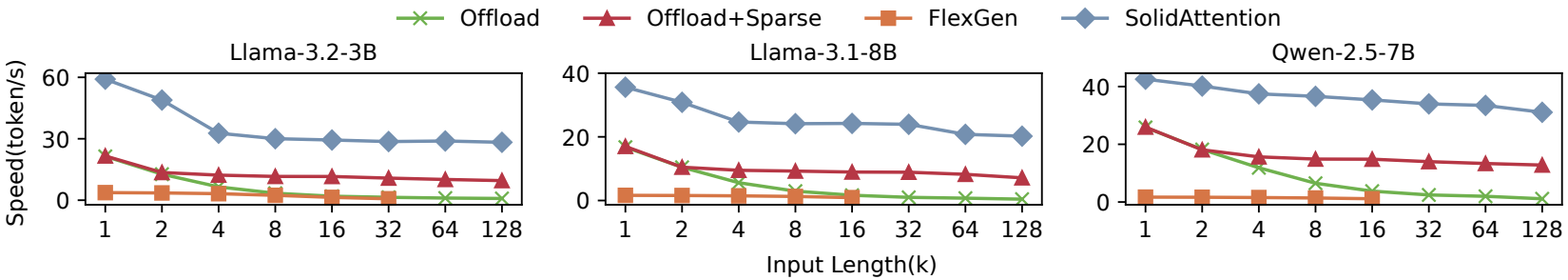
Evaluated on an AIPC prototype

CUDA Backend	SYCL Backend
 NVIDIA RTX 4070 Laptop GPU	 Intel Arc 140T iGPU
 8GB GDDR7 + 16GB DDR5	 16GB DDR5

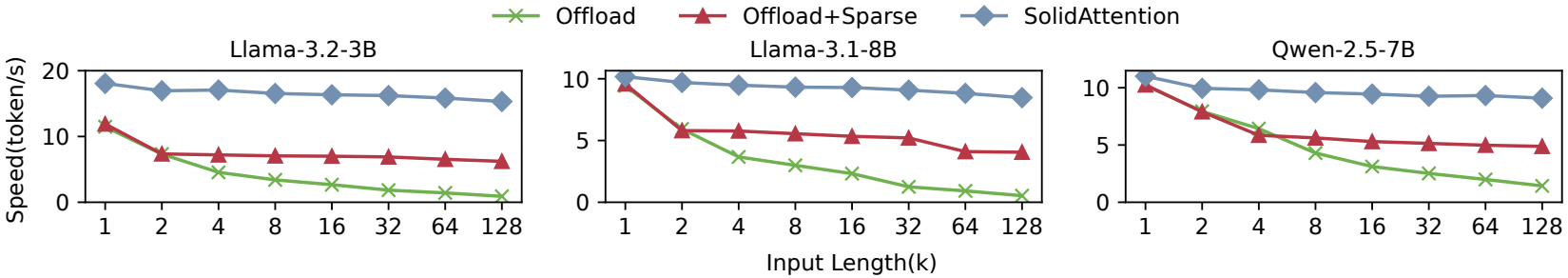
Compared Systems

- *llama.cpp* – *A famous LLM inference system which assume the whole cache in the memory.*
- *Sparse + Offload* – *A naïve implementation combining attention sparsity and KV Cache offloading.*
- *FlexGen* – *A system offloading KV cache using zig-zag scheduling.*

Evaluation: End-to-End Performance



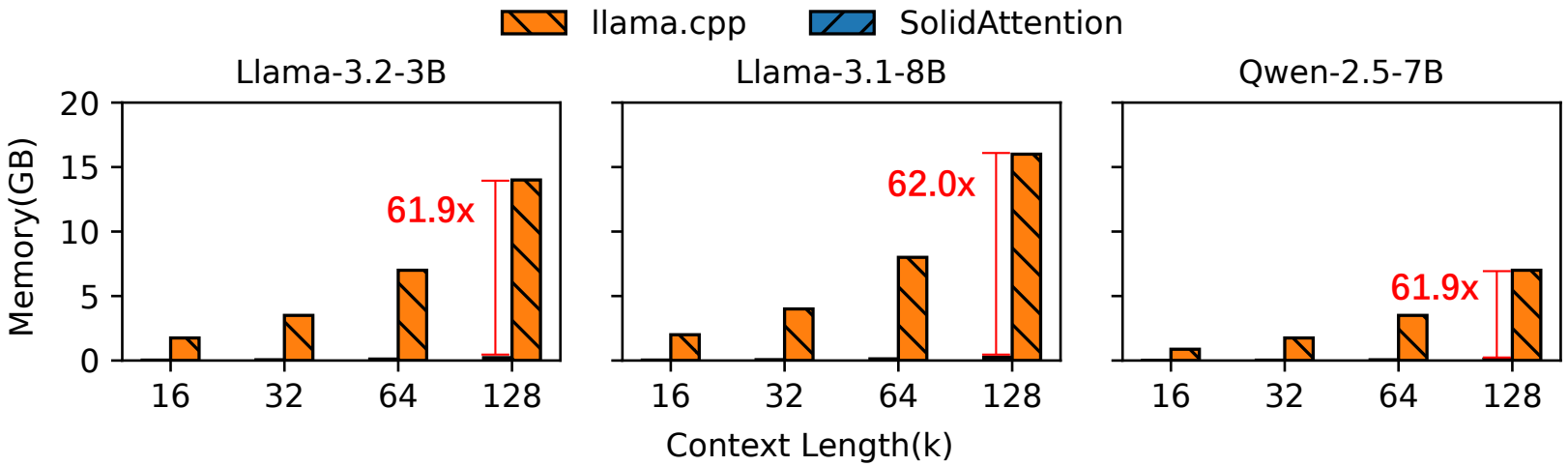
Performance on NVIDIA RTX 4070 Laptop GPU



Performance on Intel Arc 140T GPU

Compared to FlexGen and naïve implementation,
Achieve up to **3.1x speedup** for various LLMs

Evaluation： Memory Consumption



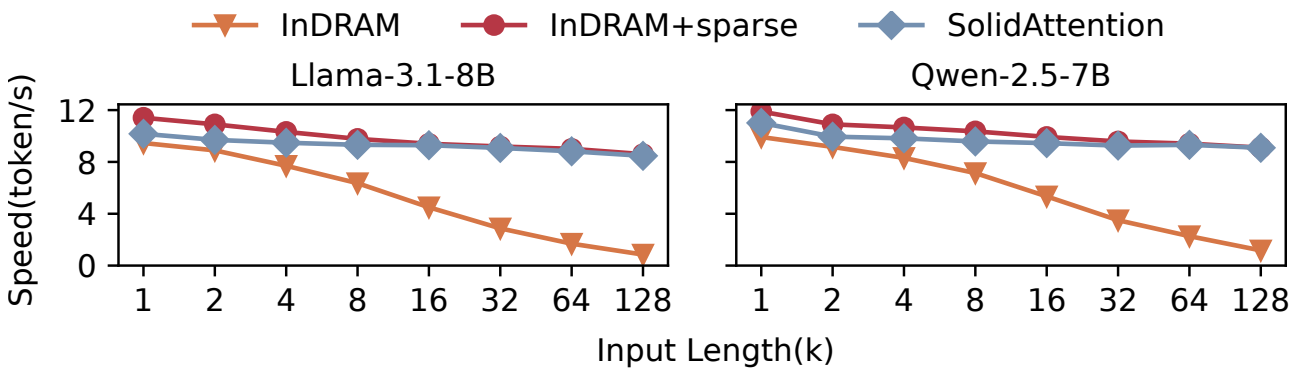
Compared to *llama.cpp*,

Saves about **98% KV cache memory occupation** for various LLMs

Evaluation: Compared To In-Memory Inference

InDRAM – inference with attention sparsity in-memory KV cache

InDRAM + sparse – inference with attention sparsity and in-memory KV cache



Compared to in-memory counterparts,

Experiences ≤ **11% throughput degradation** despite SSD offloading.

Evaluation： Accuracy

Model	Approach	Winogrande	Arc-Challenge	MMLU	GSM8K	LongBench	Average
Llama-3.2-3B	Origin	54.49	66.41	57.80	70.31	40.10	57.82
	Quant	51.93	65.76	55.48	64.06	36.29	54.70
	Ours	55.25	66.28	58.29	70.30	38.25	57.67
Llama-3.1-8B	Origin	56.59	78.31	65.91	81.25	46.75	65.76
	Quant	55.64	71.86	62.93	76.56	44.58	62.31
	Ours	57.46	80.00	66.16	80.69	45.35	65.93
Qwen-2.5-7B	Origin	67.96	87.12	73.30	82.81	45.77	71.39
	Quant	36.79	33.90	20.54	1.56	0.36	18.63
	Ours	67.88	87.46	73.46	81.25	43.75	70.76

Compared to KV cache quantization,

Keeps model accuracy even with long context length (LongBench)

Conclusion

- A SSD-based Serving on Memory-Constrained PCs.
- **Key Idea:** Co-design attention sparsity algorithm and SSD storage management.
- **Challenge:**
  - Constrained memory capacity
  - Limited request concurrency
- **Key techniques:**
  - Interleaving K and V cache.
  - Speculative prefetching KV cache.
  - Scheduling I/O and computation in fine granularity.

zxrblgs@sjtu.edu.cn