PipeDecode: Efficient LLM Inference using Pipelines within Decoding

Yunkai Liang [student] Bin Gao* [student] Pengfei Zuo◇ Zhi Zhou Xu Chen
Sun Yat-sen University *National University of Singapore ◇Huawei Cloud

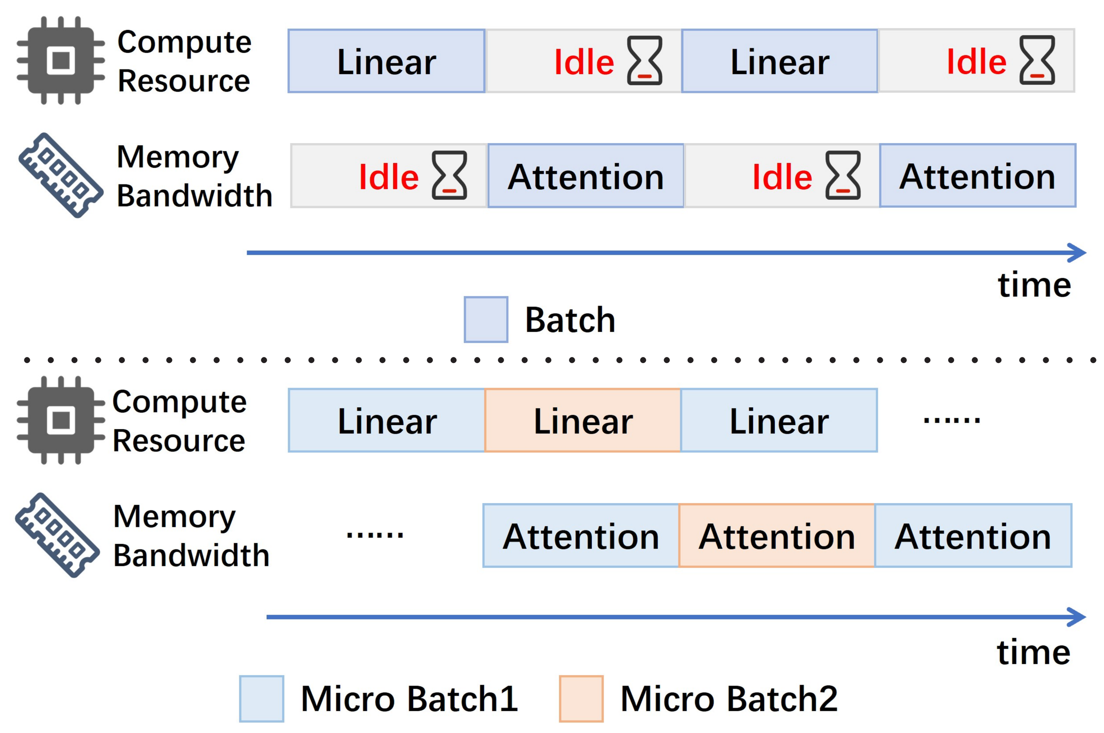
Problem and Contributions

Problem: LLM inference tasks involve multiple iterations of decoding phases, while the decoding phase often suffers from resource under-utilization.

Contributions:

- Reveal two factors that contribute to the low resource utilization in LLM inference from perspectives of heterogeneous compute-intensive and memory-intensive operators and imbalanced resource allocation.
- Propose an efficient LLM inference system, PipeDecode, that facilitates the concurrent execution of compute-intensive and memory-intensive operators through pipeline interleaving, thereby ensuring optimal resource utilization.
- Prototype PipeDecode and conduct a preliminary evaluation. The initial result shows that PipeDecode can reduce the decoding latency up to 31%.

Inference Sketch



Native Execution:

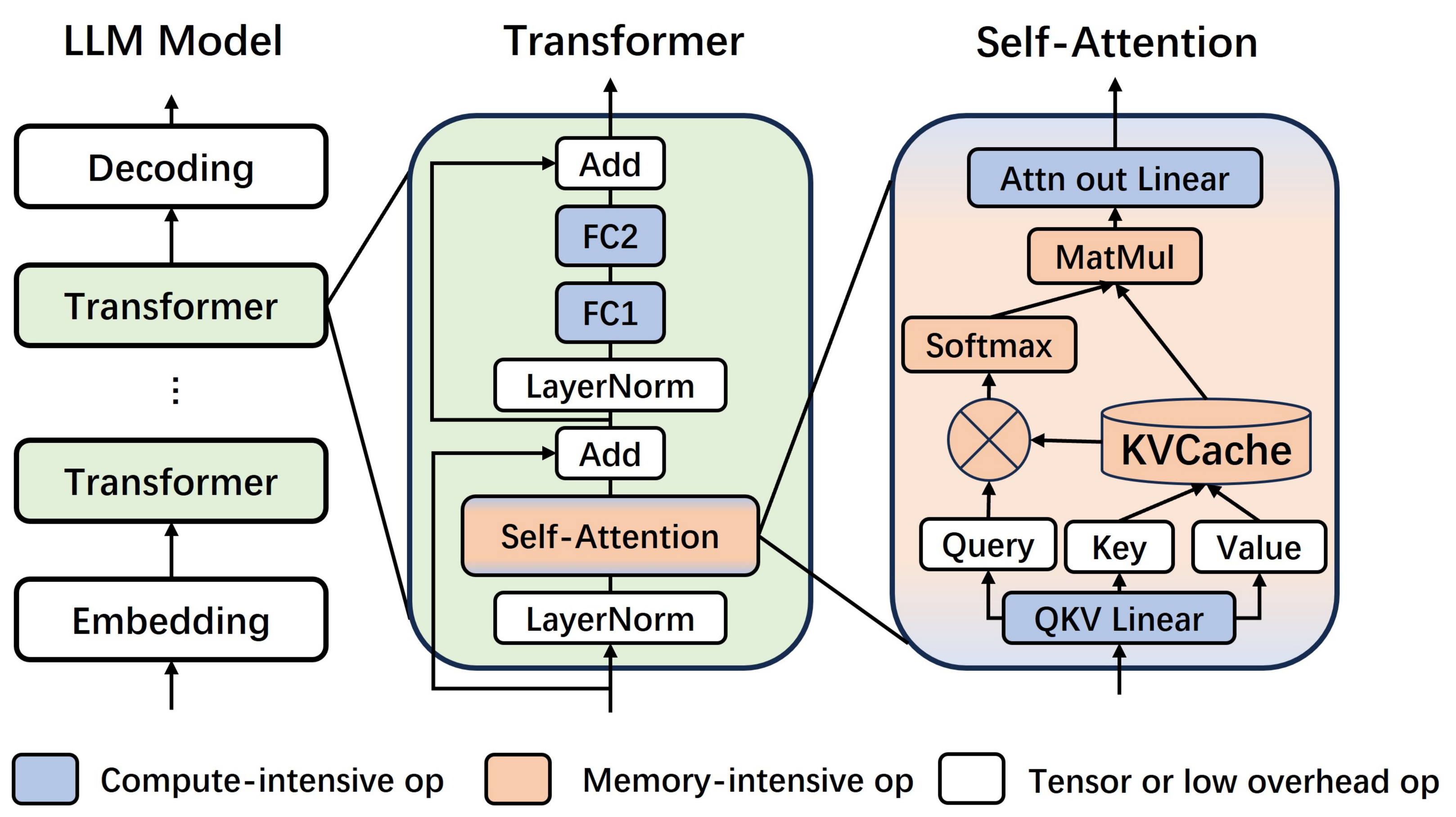
- Inference w/ bubbles
- Period of compute and memory resource idleness

Pipedecode:

- Inference w/o bubbles
- High compute and memory resource utilization

Observations

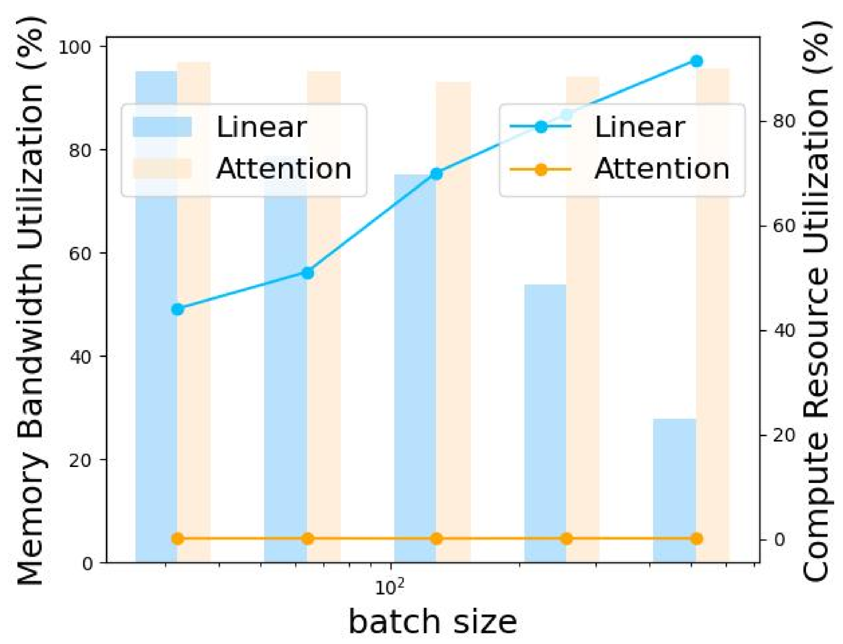
(a) Periods of compute and memory resource idleness within inference.



LLM Model Architecture

(b) Immutable and imbalanced resource allocation to distinct operators.

SM Percent	Bandwidth Utilization
7.81%	65 ~70%
21.88%	95 ~98%



Operator Profile

In the current inference system, there are **mismatched resource allocation**:

- The resource allocation becomes **immutable** once the model is installed.
- The same amount of computation resources is designated for distinct operators.
- The memory-intensive operator, the attention operator, can achieve most of the bandwidth utilization with significantly fewer SM resources.
- While the compute-intensive operator, the linear operators, require more compute resources, which is proportional to SM resources.

Challenges

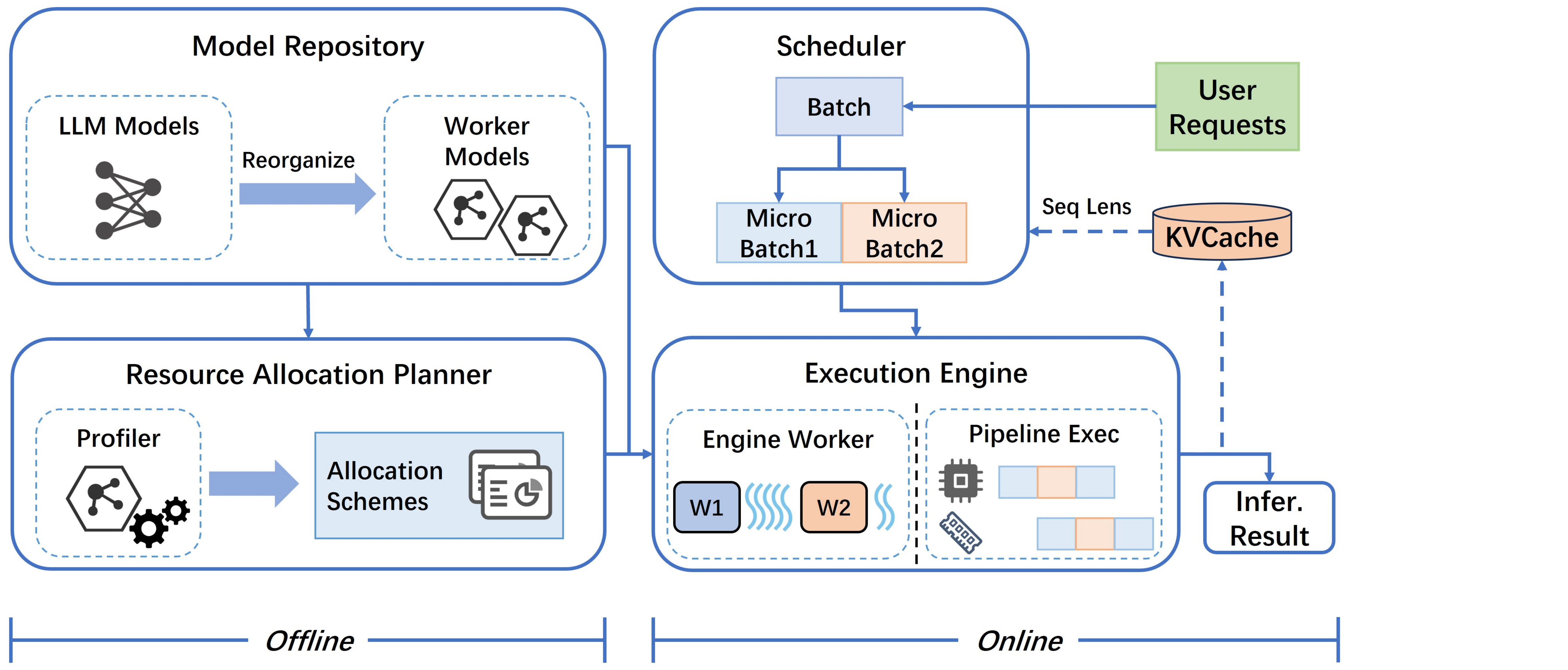
To achieve the perfect overlapping, implementing PipeDecode needs to tackle the following challenge:

- **Task scheduling**: Different context lengths[1] in inference tasks result in different execution times of the two different operators, which may cause bubbles in pipeline execution.
- **Dynamic resource allocation**: The execution times of the operators are sensitive to the resources allocated to them, managing resource allocation to operators is also vital in minimizing execution bubbles.

Methods

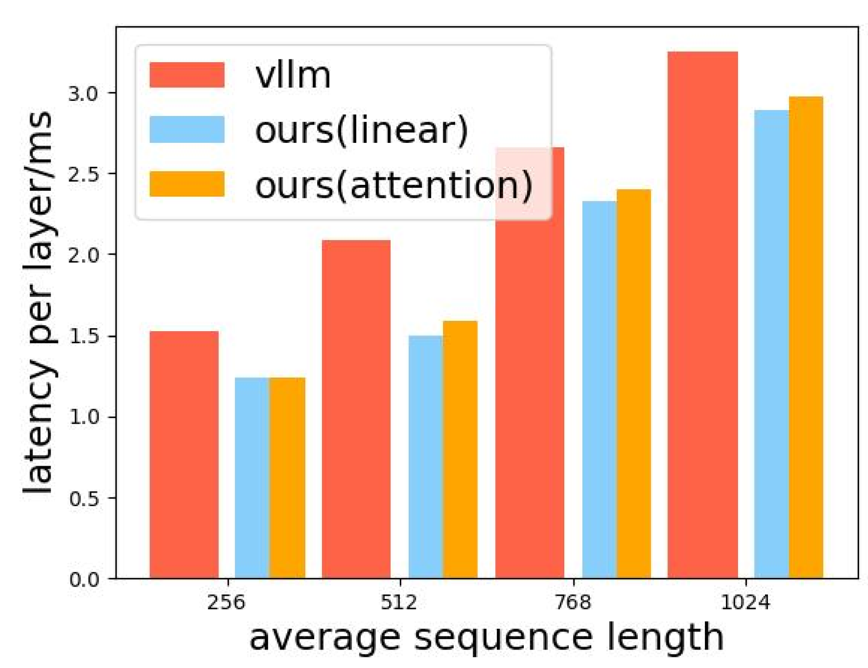
- To solve Challenge 1, PipeDecode selects suitable sequence length requests to combine into a micro-batch request. By rescheduling user requests, PipeDecode can balance the execution time and minimize possible bubbles between two micro-batches.
- To solve Challenge 2, PipeDecode assigns more SM resources for the linear operator and less for attention operator. Besides, PipeDecode dynamically adjust the SM allocation based on request lengths so as to minimize bubbles during inference.

System Overview



- Offline components: the Model Repository reorganizes the model for workers to execute; the Resource Allocation Planner profiles model execution and provides resource allocation schemes.
- Online components: the Scheduler dispatches requests to micro-batch based on their sequence lengths; the Execution Engine executes inference jobs on workers with distinct SM resources.

Evaluation



We prototyped PipeDecode on NVIDIA GTX 4090, and evaluated it with layers of Llama-7B model, a widely used open-source LLM.

Our evaluation shows that PipeDecode can reduce the decoding latency up to 31% across various sequence lengths compared to a state-of-the-art serving system vLLM[2].

References:

[1] Wang Y, *et al.* LLM Workload Study (2024)
[2] Kwon W, *et al.* Pagedattention. OSDI (2024)