LLM Inference Optimization

Moonmoon Mohanty, Gautham Bolar, Prachi Rawat, Preetam Patil, Parimal Parag
UmaMaheswari Devi, Felix George, Pratibha Moogi
Technical University of Munich, Nov 26, 2025

Indian Institute of Science
IBM Research India

Acknowledgements

CENTRE FOR
NETWORKED INTELLIGENCE
Indian Institute of Science

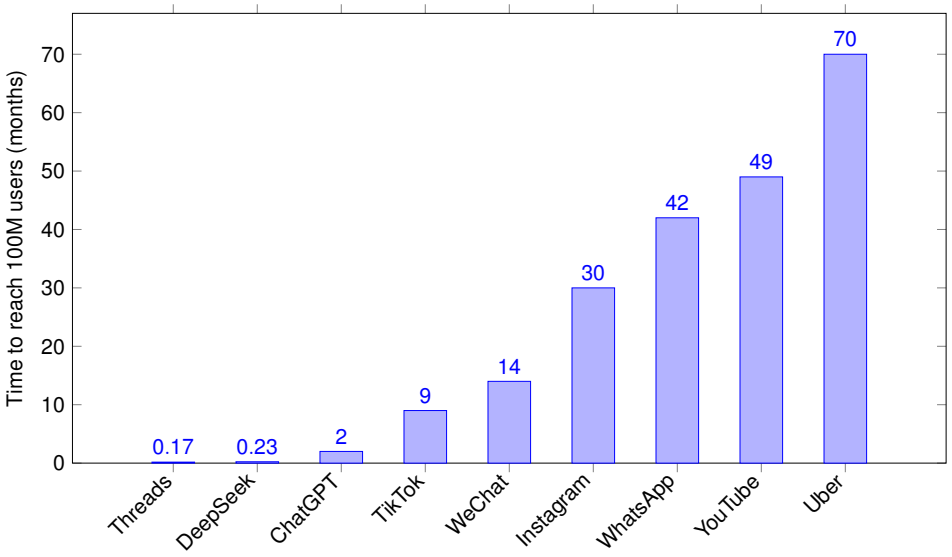
IBM

Qualcomm

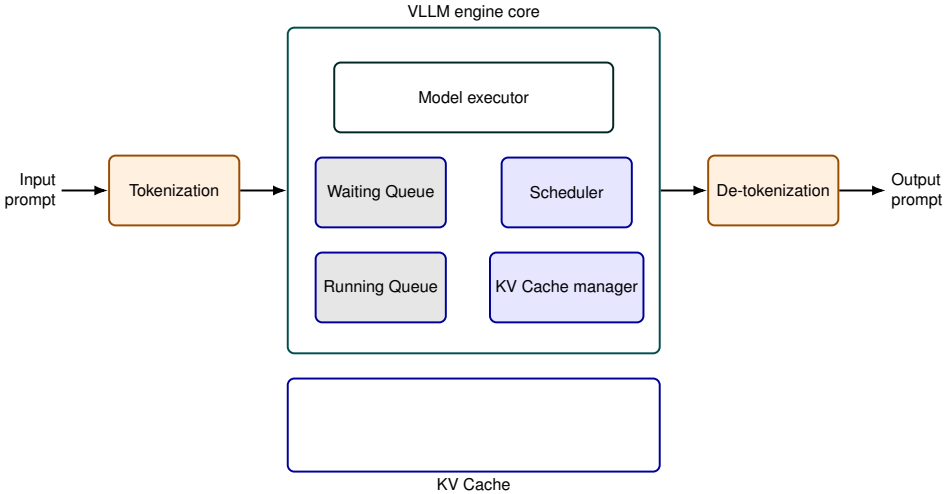
NRF National Research Foundation

intel

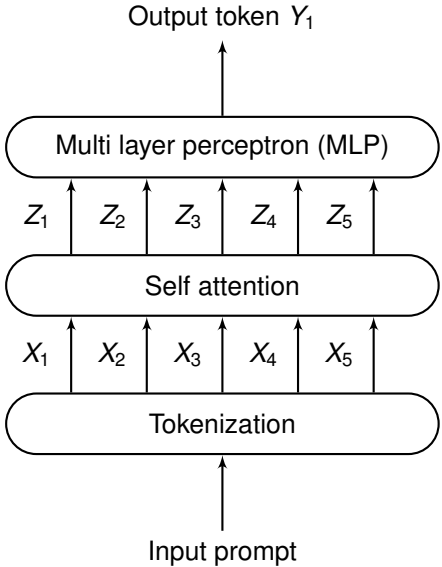
Fastest adopted apps

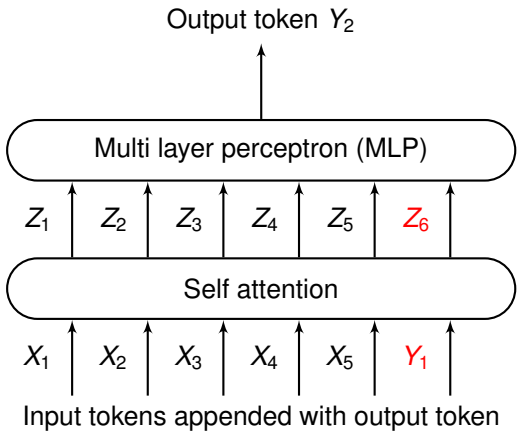


LLM inference server

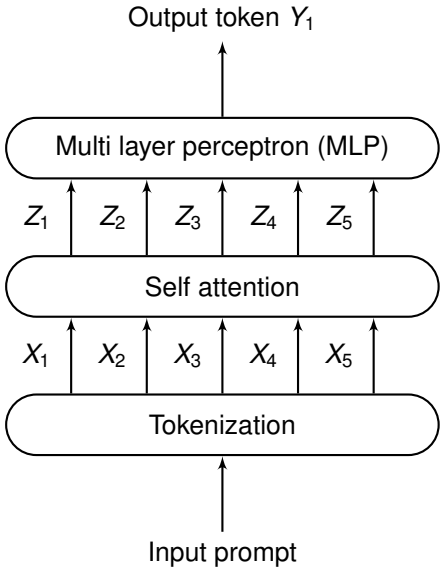


Processing pipeline





Processing pipeline



Attention computations

$$X \; W^Q = Q$$

$$X \; W^K = K$$

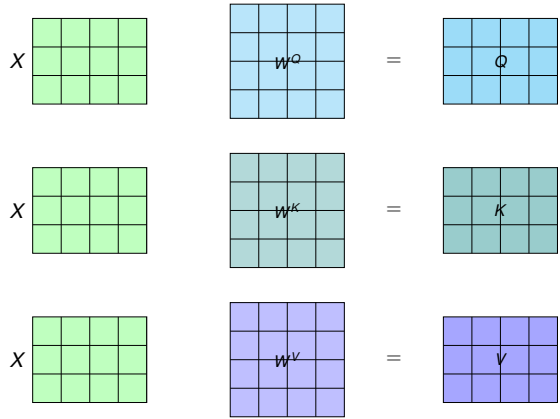
$$X \; W^V = V$$

- $Z = \texttt{softmax}(QK^\top)V$

MLP computations

- $Y = \texttt{MLP}(Z)$

Preprocessing prompts



- Prefill: Input tokens are *processed in parallel*
- Number of rows correspond to number of input tokens

Autoregressive processing

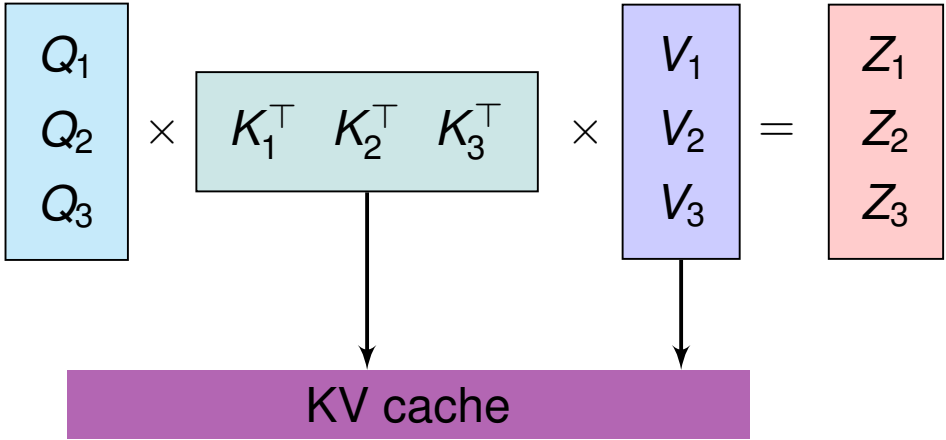
$$\begin{bmatrix} X \\ Y \end{bmatrix} \begin{bmatrix} W^Q \end{bmatrix} = \begin{bmatrix} Q \\ YW^Q \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \end{bmatrix} \begin{bmatrix} W^K \end{bmatrix} = \begin{bmatrix} K \\ YW^K \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \end{bmatrix} \begin{bmatrix} W^V \end{bmatrix} = \begin{bmatrix} V \\ YW^V \end{bmatrix}$$

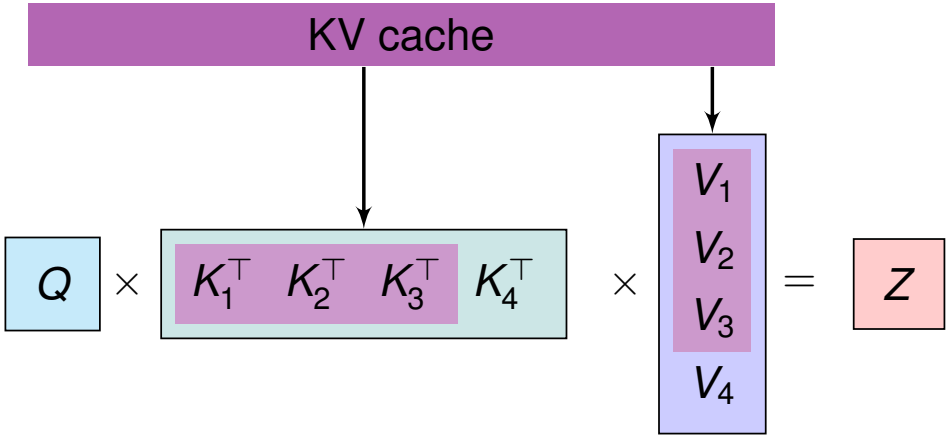
- Repeating computations from the previous round

Preprocessing with KV cache write



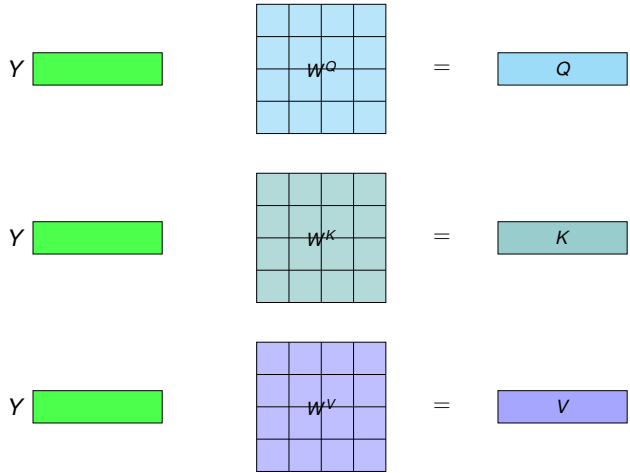
- (K, V) values stored in KV cache for subsequent decodes
- Output $Z \triangleq \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V$ has the same number of rows

Postprocessing with KV cache read and write



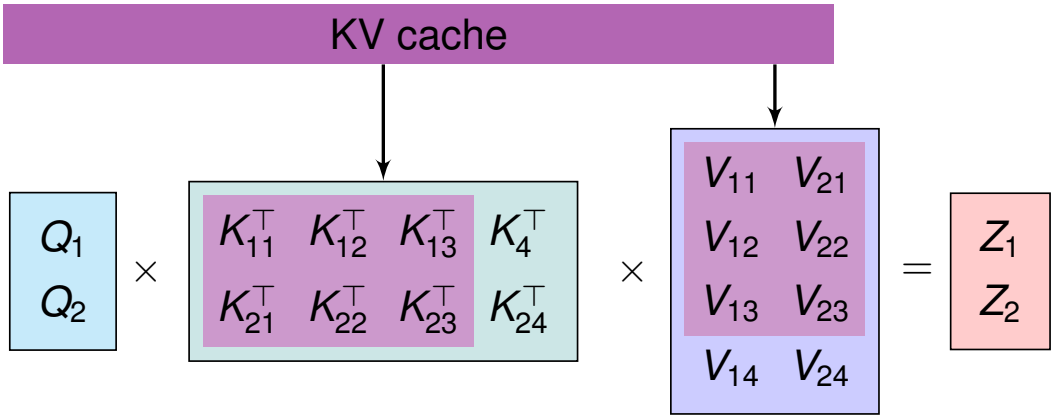
- Previously stored K, V values retrieved for each decode
- Output K, V values appended to previously stored K, V values in KV cache

Decode

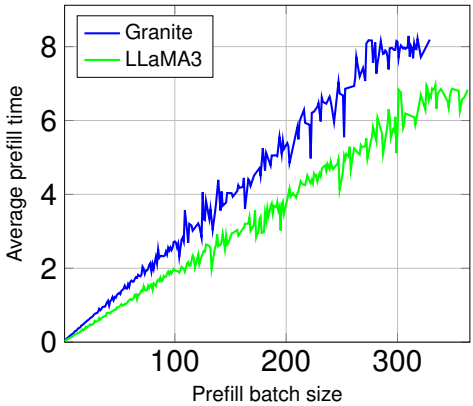


- Decode: Output tokens are *processed sequentially* from input tokens
- Underutilized compute capacity for each decode

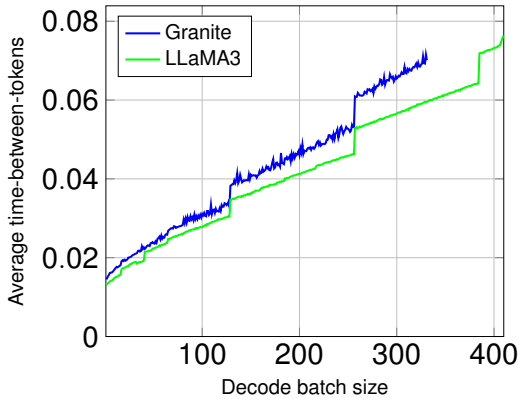
Continuous batching of decodes



Prefill and Decode times (ShareGPT dataset)



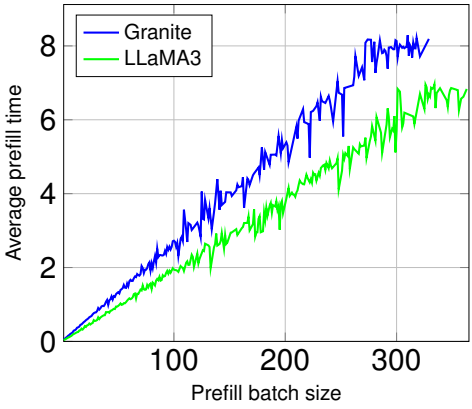
Prefill time

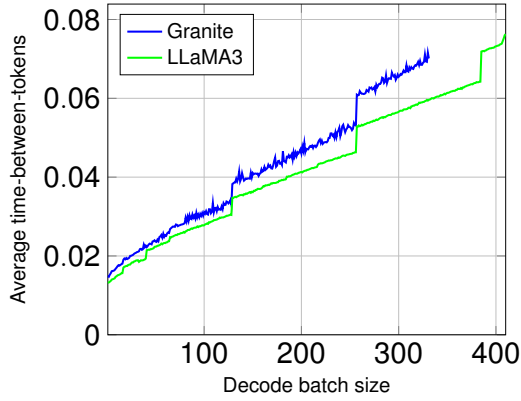


Decode time

Experiment setup: NVIDIA A100 (80GB) GPU, LLaMA and Granite 8GB models, vLLM V0

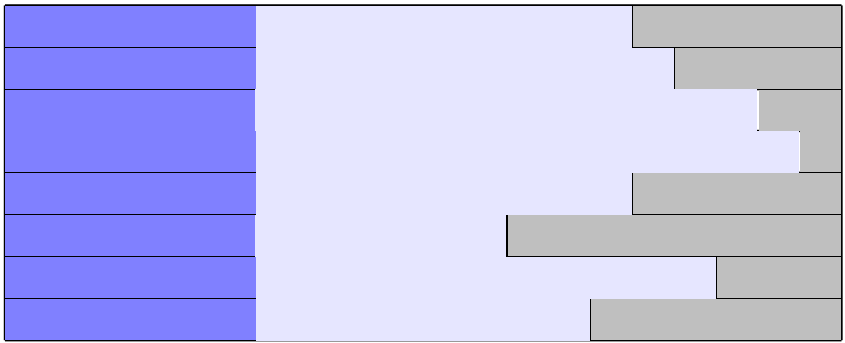
Prefill and Decode times





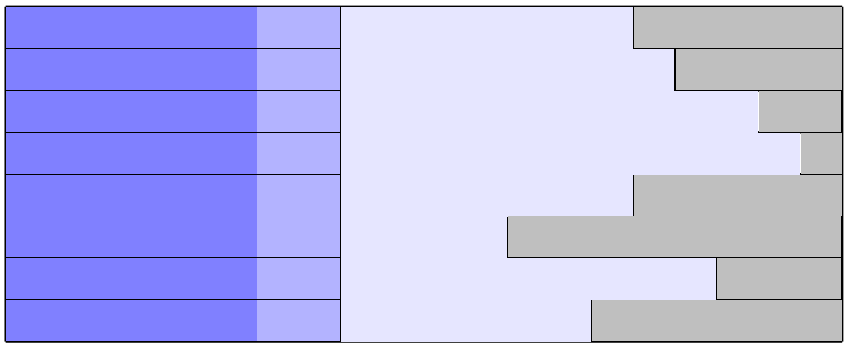
- Each prompt k has l_k input tokens
- Time to prefill K prompts is $c_p + t_p \sum_{k=1}^{K} l_k$
- Time to decode a batch of B tokens is $c_d + t_d B$

Prefilling prompts



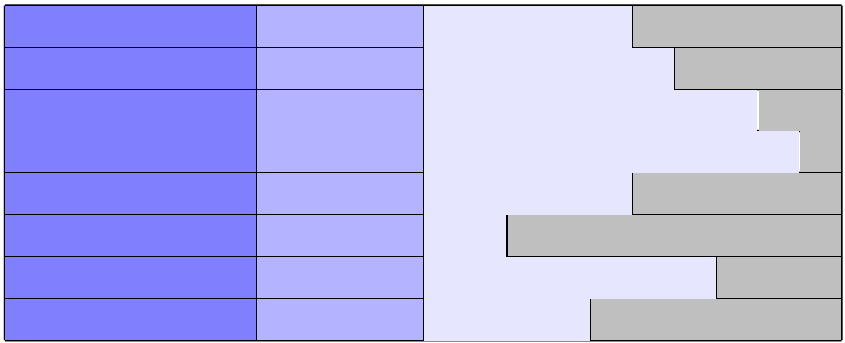
- Time to prefill B prompts is $c_p + t_p \sum_{k=1}^{B} l_k$

Decode for prefilled prompts



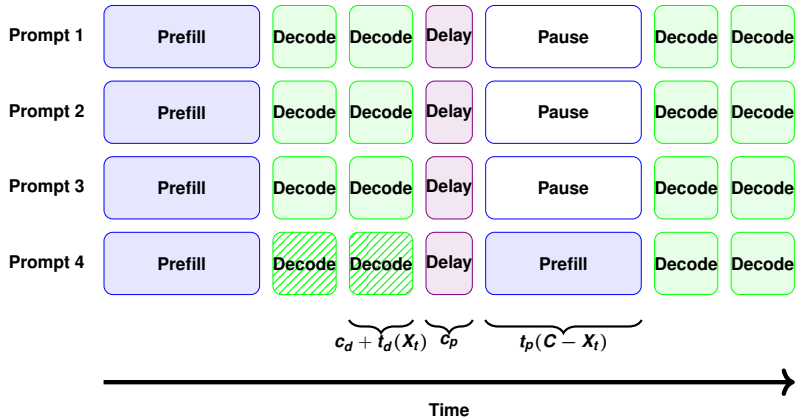
- Time to decode a batch of B tokens is $c_d + t_d B$

Decode for prefilled prompts



- Decode finishes for a prompt when the number of output tokens are generated
- This reduces the batch size

Alternating prefill and decode



Alternating prefill and decode

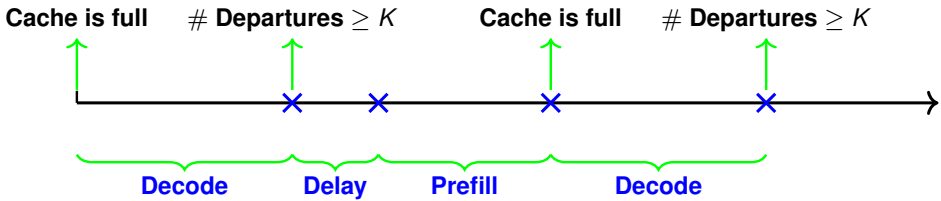
Default VLLM (v0) policy

- The inference engine alternates between prefill and decode phases
- Attempt a prefill whenever a prompt completes and there's a vacancy in the KV cache
- Switchover from decode to prefill incurs a fixed overhead

Observation

- Frequent prefills lead to larger switching delay and hence smaller throughput
- Infrequent prefills lead to smaller batch size and smaller throughput

Alternating prefill and decode: controlled transition



Key question

- What is the optimal departure threshold for average throughput maximization?

Analytical modeling and results

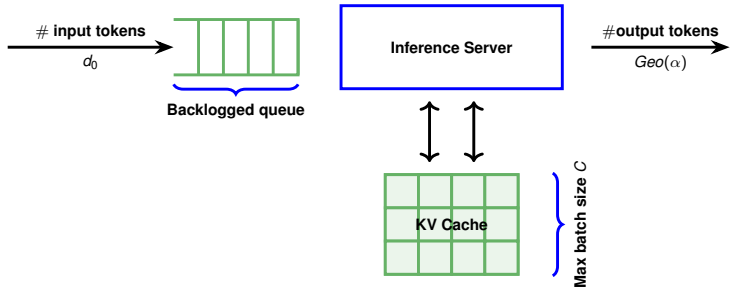
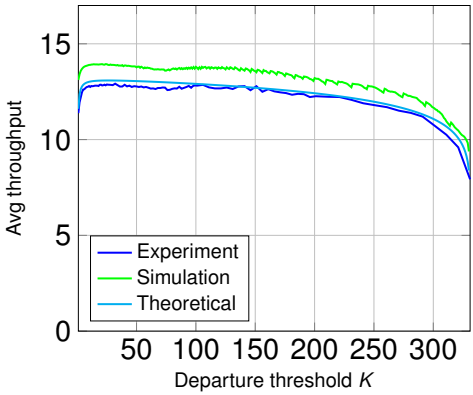


Figure: Timeline

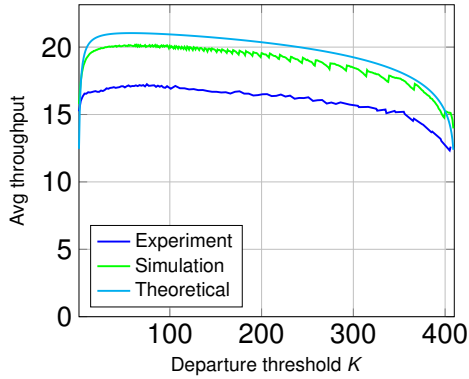
$$\rho(K)^{-1} \approx \frac{1}{K}\left(c_p + c_d \frac{\ln(1-\frac{K}{C})}{\ln(1-\alpha)}\right) + \frac{t_d}{\alpha} + \frac{t_p d_0}{N}$$

K: departure threshold
c_p: scheduling overhead
t_p: i/p processing time
c_d: o/p token compute
t_d: memory slowdown

Experimental validation



Granite



Llama

Figure: Performance metrics for different models with the ShareGPT dataset.

Conclusion

- Departure threshold-based scheduling algorithm.
- Analytical model for inference system, deduced closed form expression for throughput.
- Proved existence of optimal departure threshold that maximizes the system throughput.
- Characterization of LLM inference system to find the analytical model parameters.
- Experimental validation with vLLM inference server and NVIDIA A100 GPU.
- **Key outcome**: Proposed policy leads to 13% improvement in average throughput accompanied by 14% reduction in average prompt completion time.