

# CS 744: NANOFLOW, VLLM

Shivaram Venkataraman

Spring 2025

# ADMINISTRIVIA

Assignment grading

Project Preferences

# SELF ATTENTION

Input vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t$

Output vectors  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_t$

*Attention: weighted average over all the input vectors*

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{x}_j .$$

$$w'_{ij} = \mathbf{x}_i^T \mathbf{x}_j .$$

$$w_{ij} = \frac{\exp w'_{ij}}{\sum_j \exp w'_{ij}} .$$

# QUERIES, KEYS, VALUES

Every input vector  $\mathbf{x}_i$  is used in three different ways

- Query: Compared to every other vector for its own output  $\mathbf{y}_i$
- Key: Compared to every other vector for the output vector  $\mathbf{y}_j$
- Value: Part of the weighted sum to compute each output vector

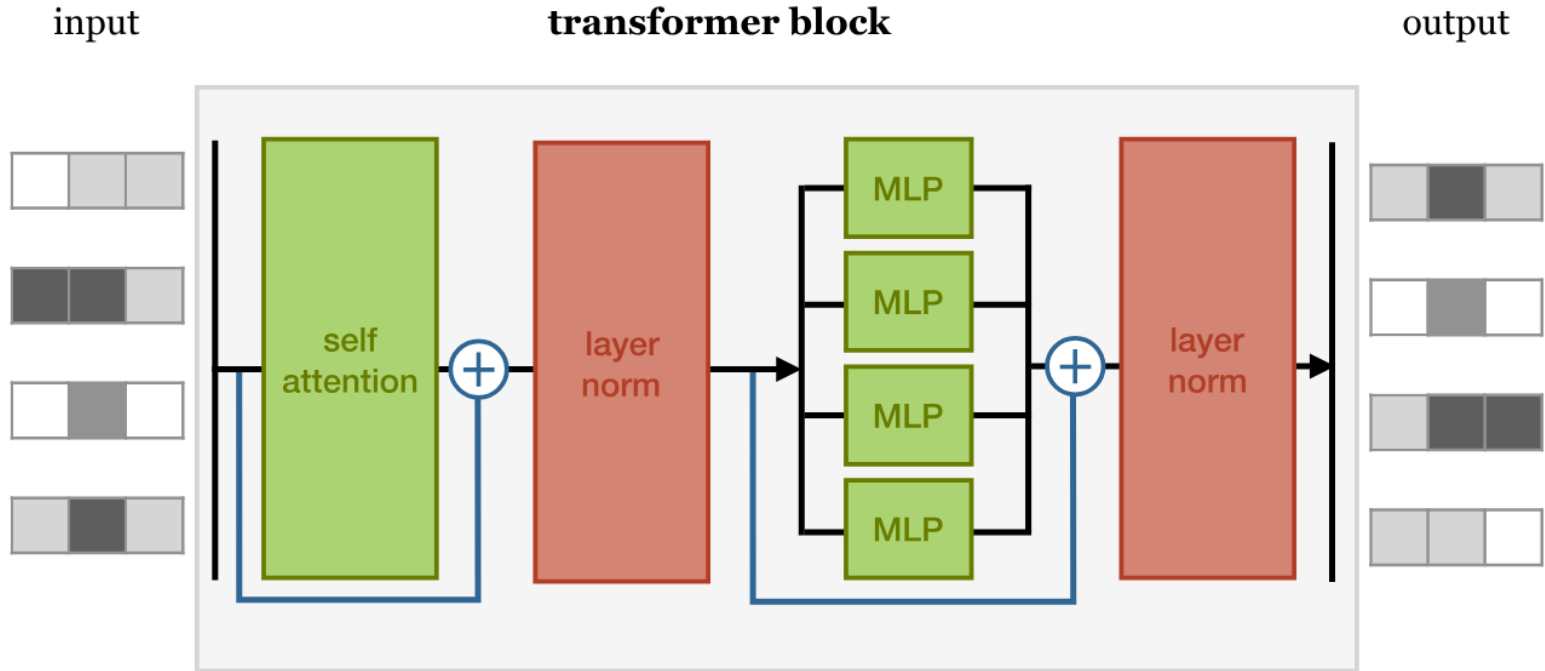
$$\mathbf{q}_i = \mathbf{W}_q \mathbf{x}_i \quad \mathbf{k}_i = \mathbf{W}_k \mathbf{x}_i \quad \mathbf{v}_i = \mathbf{W}_v \mathbf{x}_i$$

$$w'_{ij} = \mathbf{q}_i^\top \mathbf{k}_j$$

$$w_{ij} = \text{softmax}(w'_{ij})$$

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{v}_j .$$

# TRANSFORMER BLOCK



# KV CACHE, PREFILL VS DECODE

$$q_i = W_q x_i \quad k_i = W_k x_i \quad v_i = W_v x_i$$

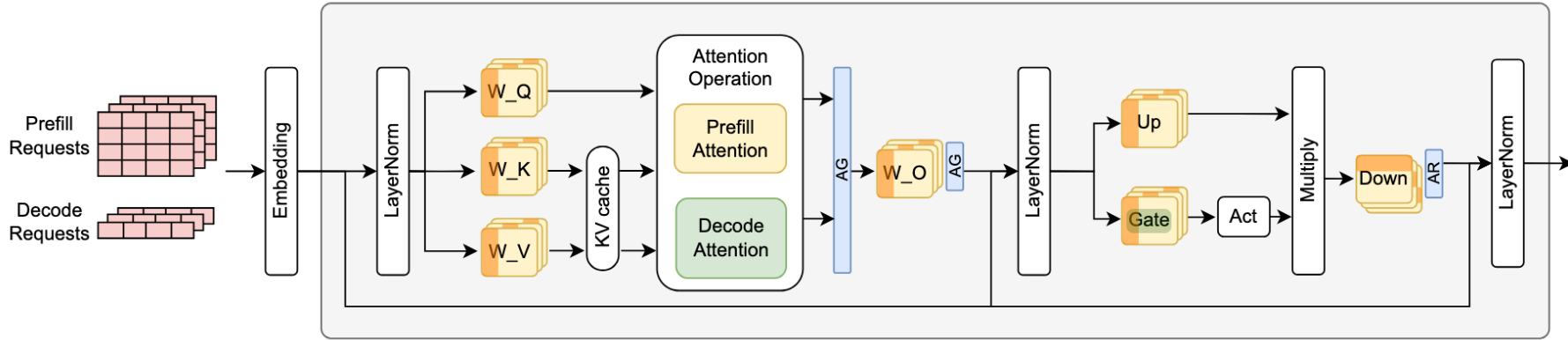
Cache Key and Value vectors for the **context length**

Minimize re-generation of key and value vectors for prior tokens.

Pre-fill: Digests all the input tokens (prompt). Fill KV Cache

Decode: Generate token-by-token till termination

# PUTTING IT ALL TOGETHER



# HOW TO ENSURE HIGH THROUGHPUT?

Approach: Try to derive hardware bounds.

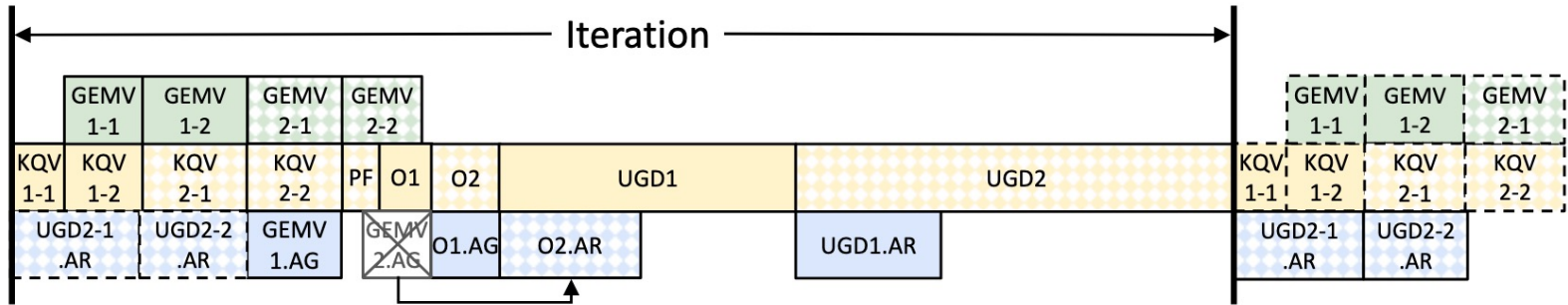
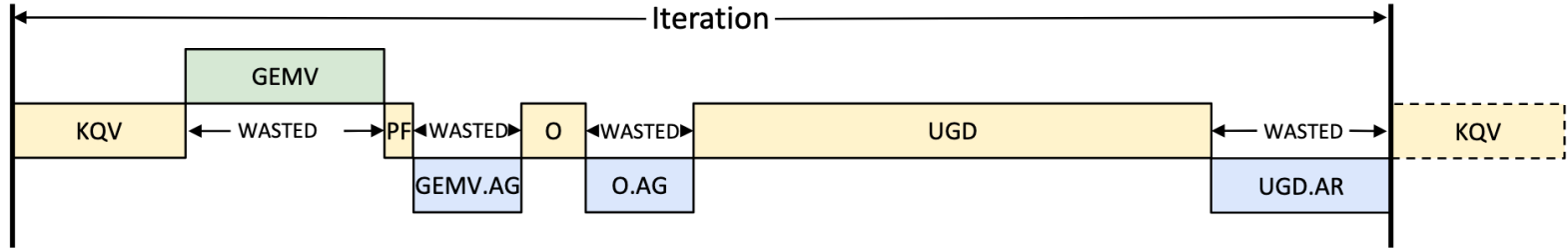
Investigate which hardware resource is the bottleneck

GPU Model	Release Year	NetBW (GB/s)	Compute (FP16 GFLOP/s)	MemBW (GB/s)	Ratio (FLOP/B)
V100	2017	300	125,000	900	139
A100 - 40G	2020	600	312,000	1555	200
A100 - 80G	2021	600	312,000	2000	156
H100	2023	600	989,000	3352	295
H200	2024	900	989,000	4800	206
B100	2024	1800	1,800,000	8000	225
B200	2024	1800	2,250,000	8000	281
MI250	2021	800	362,000	3352	107
MI300	2023	1024	1,307,000	5300	246

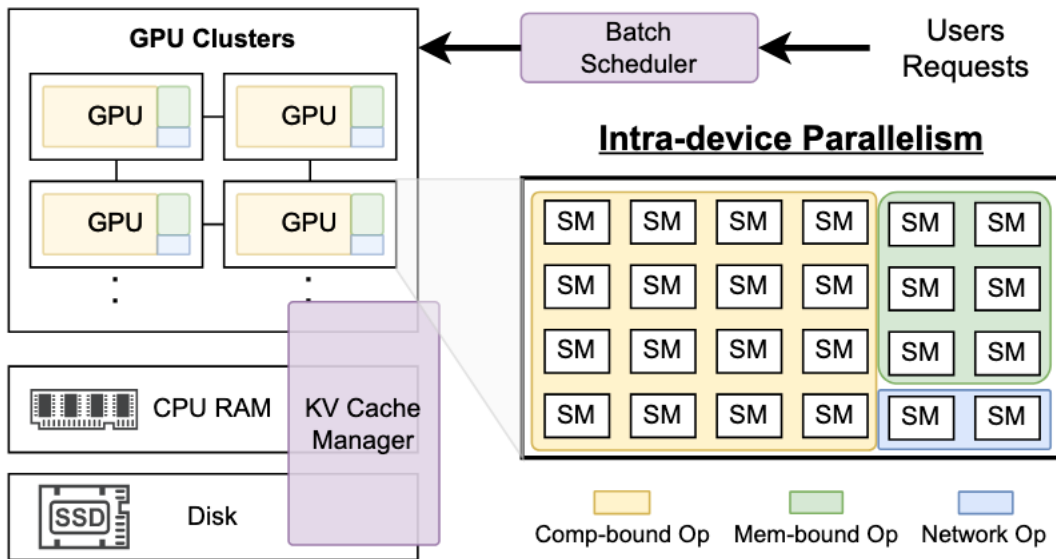
# PROFILING RESULTS

Operation	Compute (GFLOP)	Memory Movement (GB)	Network Usage (GB)	$T_{compute}$ (ms)	$T_{mem}$ (ms)	$T_{net}$ (ms)	Measured Time (ms)
GEMM-KQV	27487.8	19.5	0	<b>11.01</b>	1.22	0	<b>16.08</b>
GEMM-O	21990.2	16.1	0	<b>8.81</b>	1.01	0	<b>16.01</b>
GEMM-UG	153931.6	96.6	0	<b>61.67</b>	6.04	0	<b>69.92</b>
GEMM-D	76965.8	49.7	0	<b>30.84</b>	3.11	0	<b>34.96</b>
Decode Attention	3665.9	462.2	0	1.47	<b>28.89</b>	0	<b>35.60</b>
Prefill Attention	916.3	2.1	0	<b>0.37</b>	0.13	0	<b>4.56</b>
Communication	18.8	75.2	75.2	0.01	4.70	<b>31.33</b>	<b>47.92</b>
Total				<b>114.17</b>	45.09	31.33	

# HOW TO IMPROVE UTILIZATION? NANO BATCHES



# DESIGN



Pick batch sizes which maximize throughput (2048)

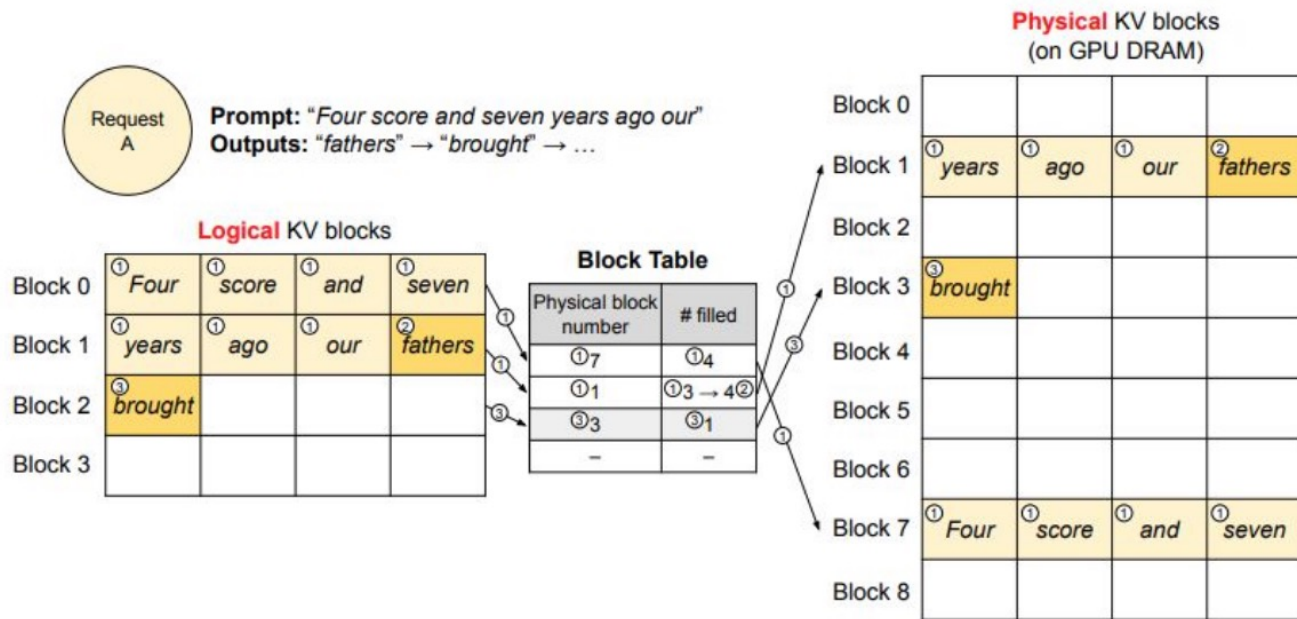
Minimize number of nanobatches, avoiding pipeline bubbles

Intra-device: Search for number of SMs to allocate for each nanobatch

# DESIGN: KV CACHE MANAGEMENT

Offload unused KV caches to SSDs

Load KV cache back for a multi-round use case



# SUMMARY

LLM Inference: Mix of compute, memory bound ops

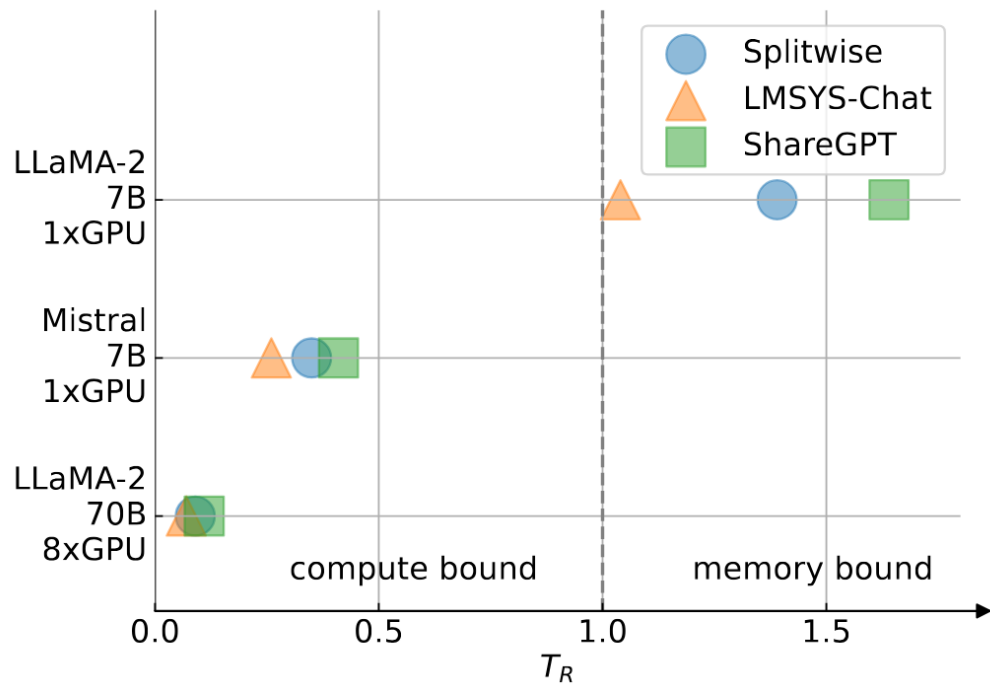
Key idea: Minimize idle resources through pipelining

Careful memory management, scheduling with nanobatches



# DISCUSSION

<https://forms.gle/2fnsyMI2xEWtj7TA7>



What are some similarities of nanoflow with prior papers we have read in the class so far? What are some differences?

# NEXT STEPS

Next class: Scheduling