南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

AlayaDB 太昊信息

AlayaDB: The Data Foundation for Efficient and Effective Long-context LLM Inference

Yangshen Deng*, *Zhengxin You**, *Long Xiang**, *Qilong Li*, *Peiqi Yuan*, *Zhaoyang Hong*, *Yitao Zheng*, *Wanting Li*, *Runzhong Li*, *Haotian Liu*, *Kyriakos Mouratidis*, *Man Lung Yiu*, *Huan Li*, *Qiaomu Shen*, *Rui Mao*, *Bo Tang*

research@alayadb.ai

AlayaDB Team


Yangshen Deng
In SIGMOD25


Zhengxin You
In SIGMOD25


Long Xiang
In SIGMOD25


Qilong Li


Peiqi Yuan


Zhaoyang Hong


Yitao Zheng
In SIGMOD25


Wanting Li


Haotian Liu
In SIGMOD25


Kyriakos Mouratidis
In SIGMOD25


Man Lung Yiu
In SIGMOD25


Huan Li
In SIGMOD25

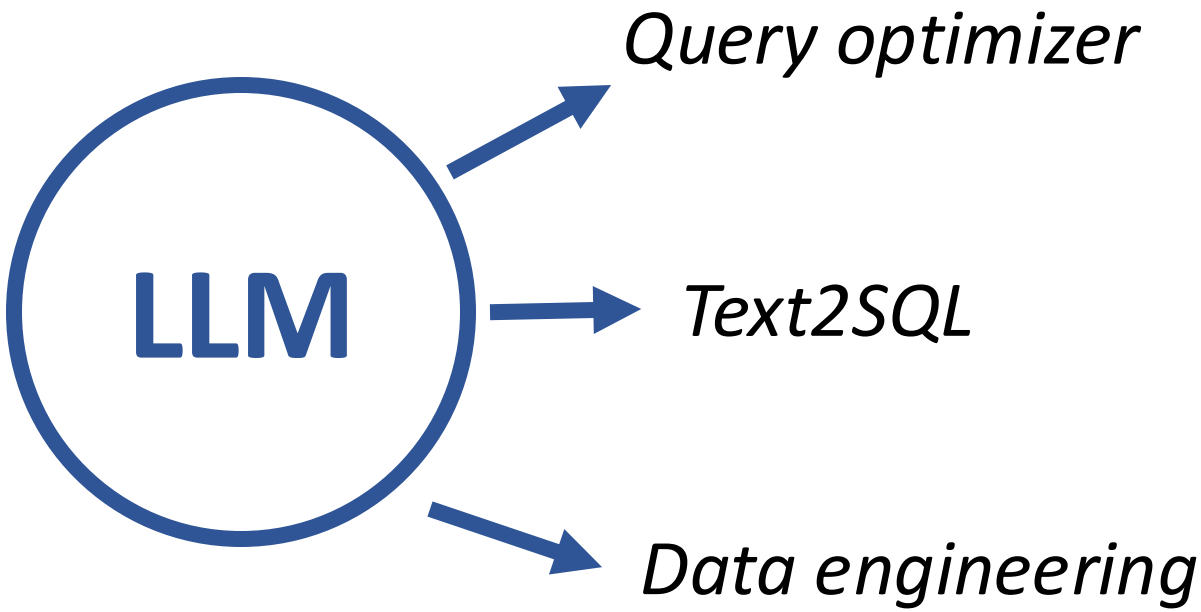

Qiaomu Shen


Rui Mao


Bo Tang
In SIGMOD25

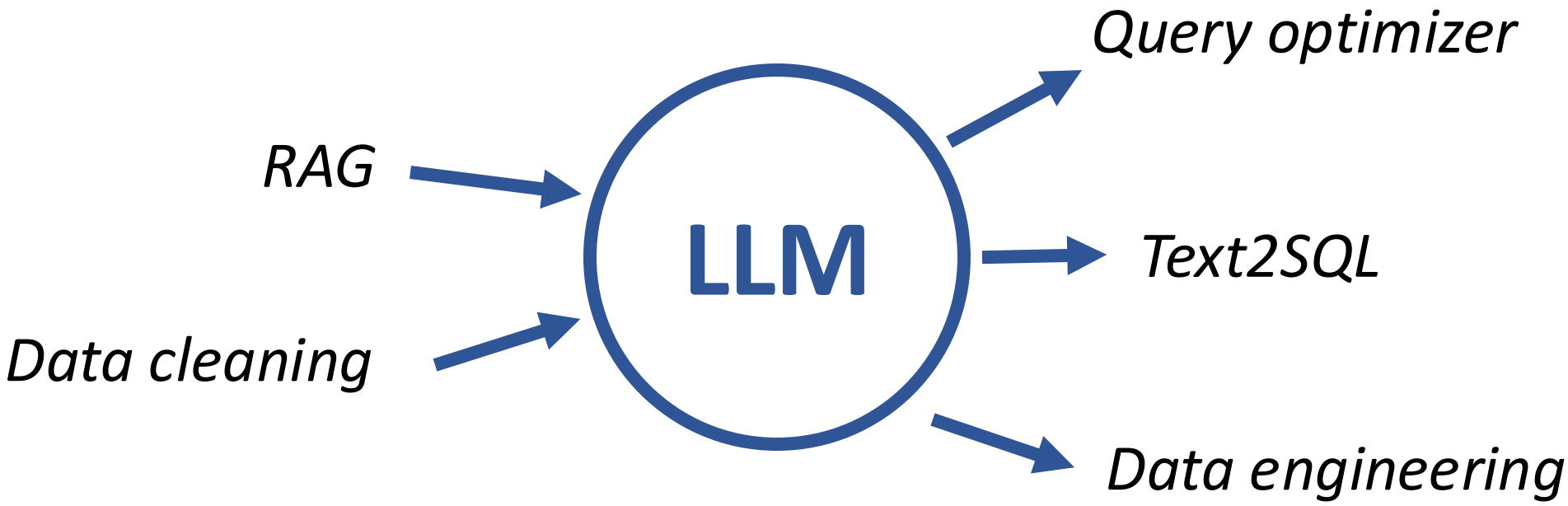
What are DB guys doing in LLM era?

- LLM for database



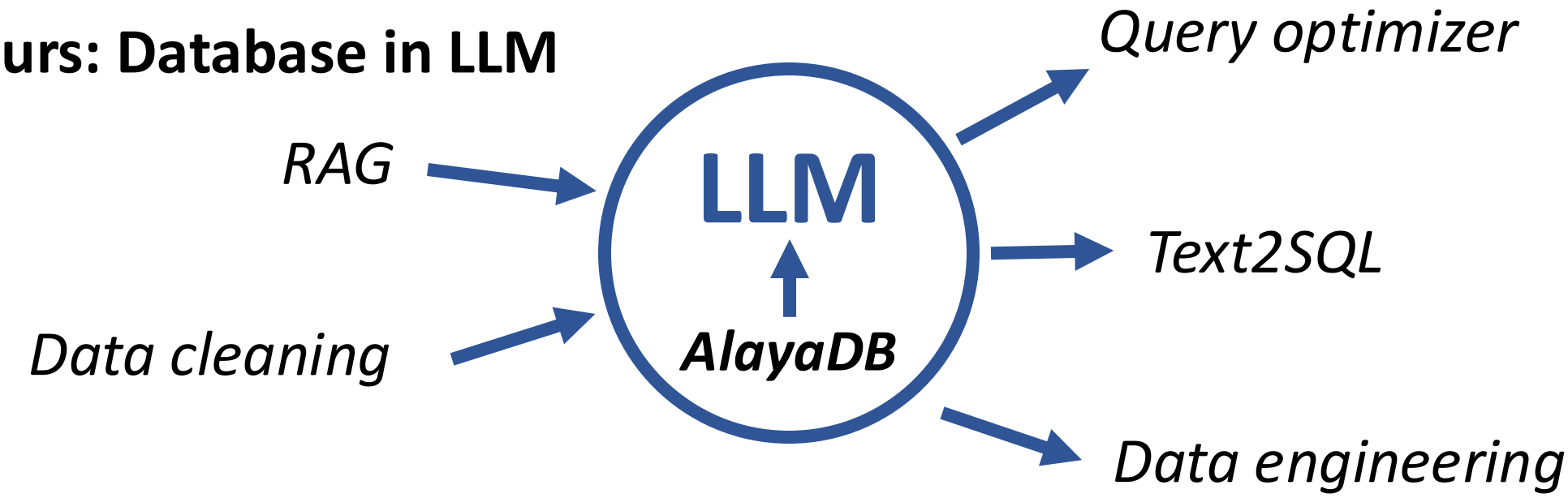
What are DB guys doing in LLM era?

- LLM for database
- Database for LLM



What are DB guys doing in LLM era?

- LLM for database
- Database for LLM
- **Ours: Database in LLM**



AlayaDB

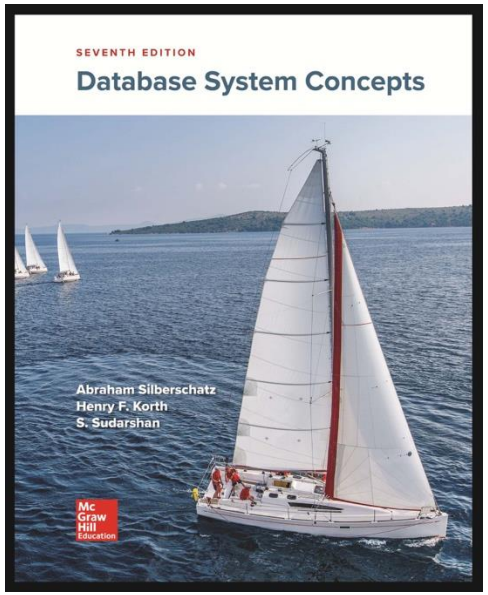
- A vector database in LLM
- KV cache management + attention computation
- Supports long context LLM inference with *low resource*, *low latency*, and *high quality*

Long context LLM inference

- Long document analysis
- Code analysis
- Chatbot with long chatting history
- …

Long context LLM inference

- Is expensive. Why?



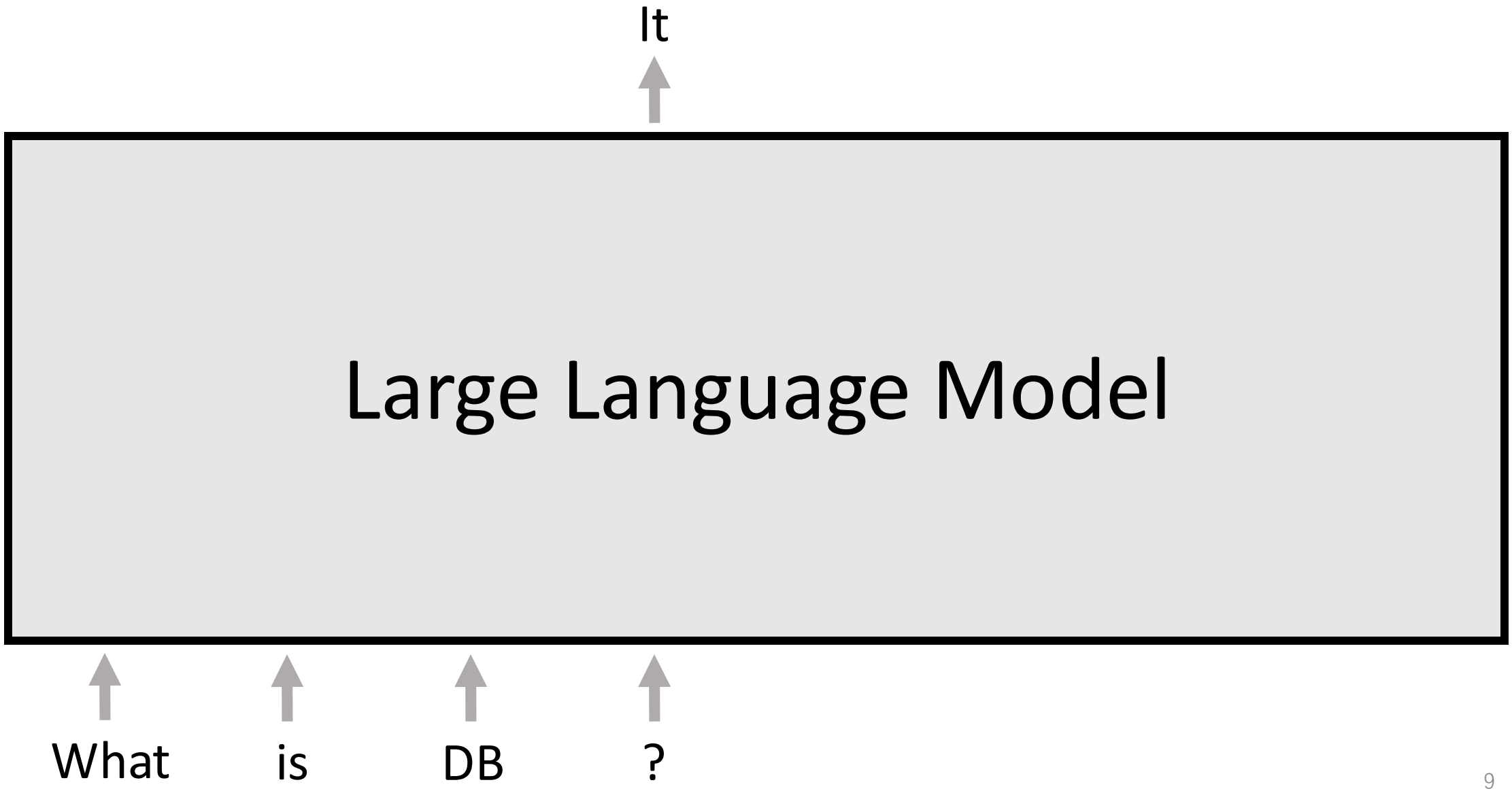
+ What is 2PL protocol?

→ LLM

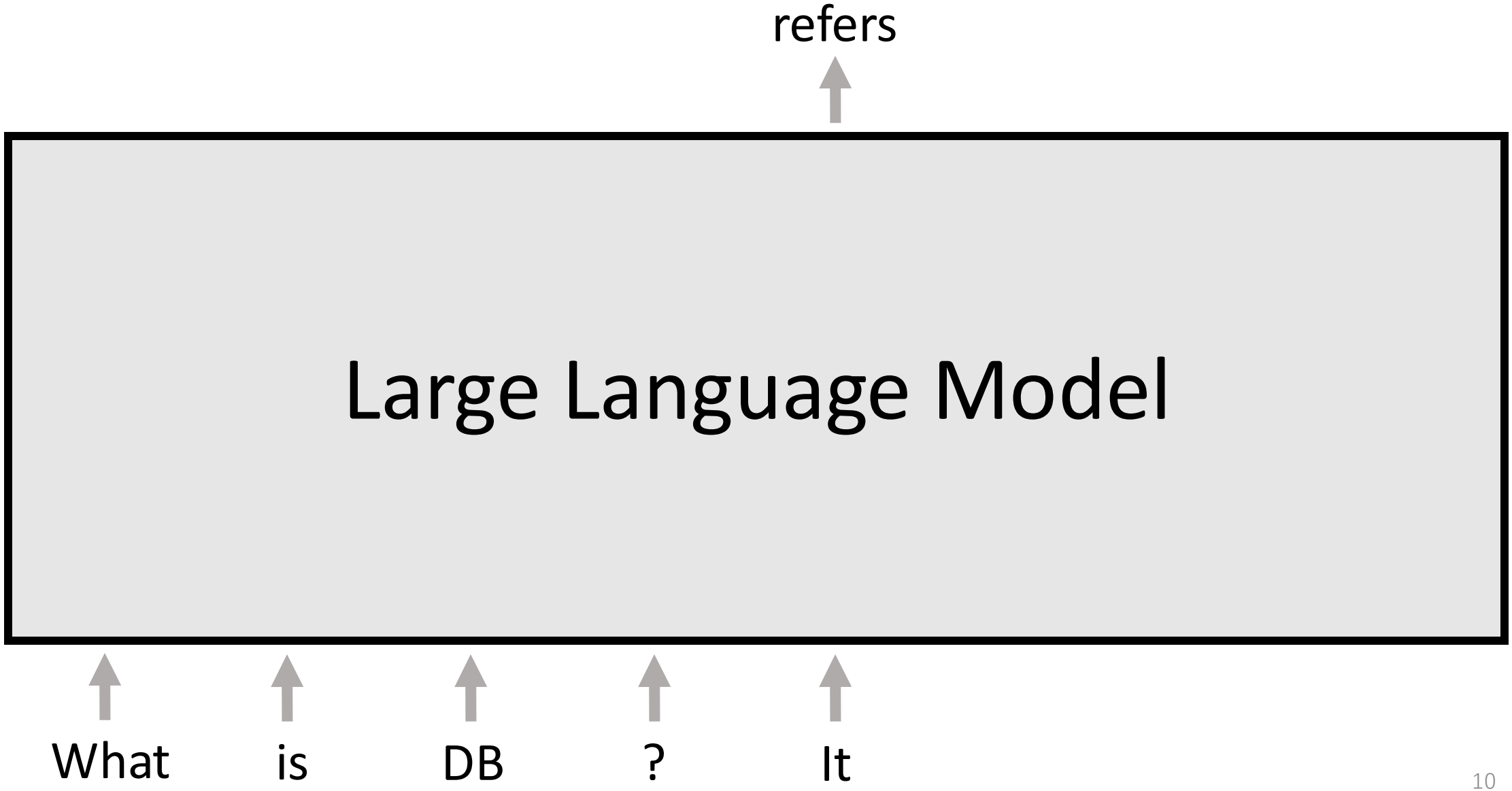
Model
Llama-3-8B

Cost
141.38GB GPU memory
2 x NVIDIA A800 (80GB)
6 minutes

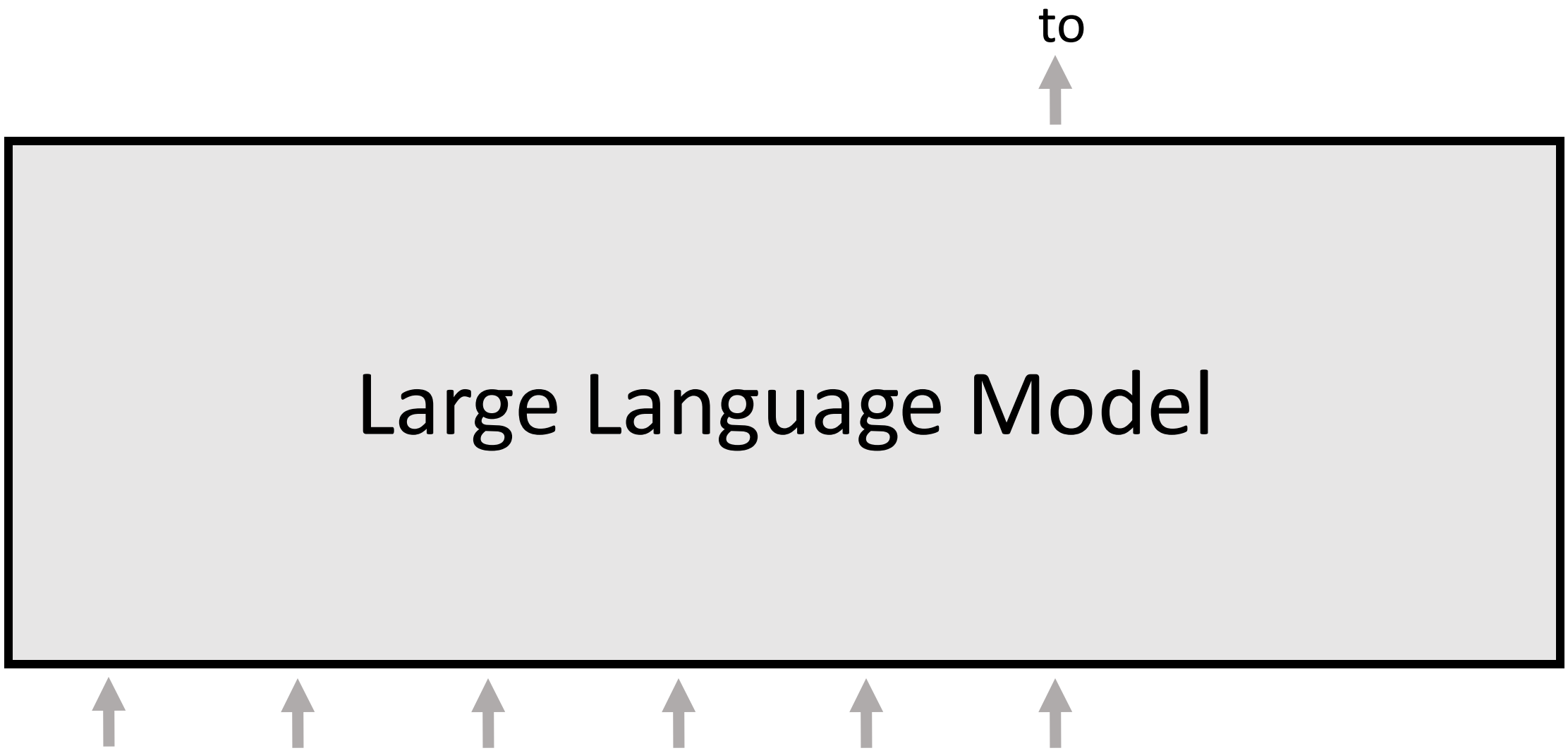
LLM inference basic



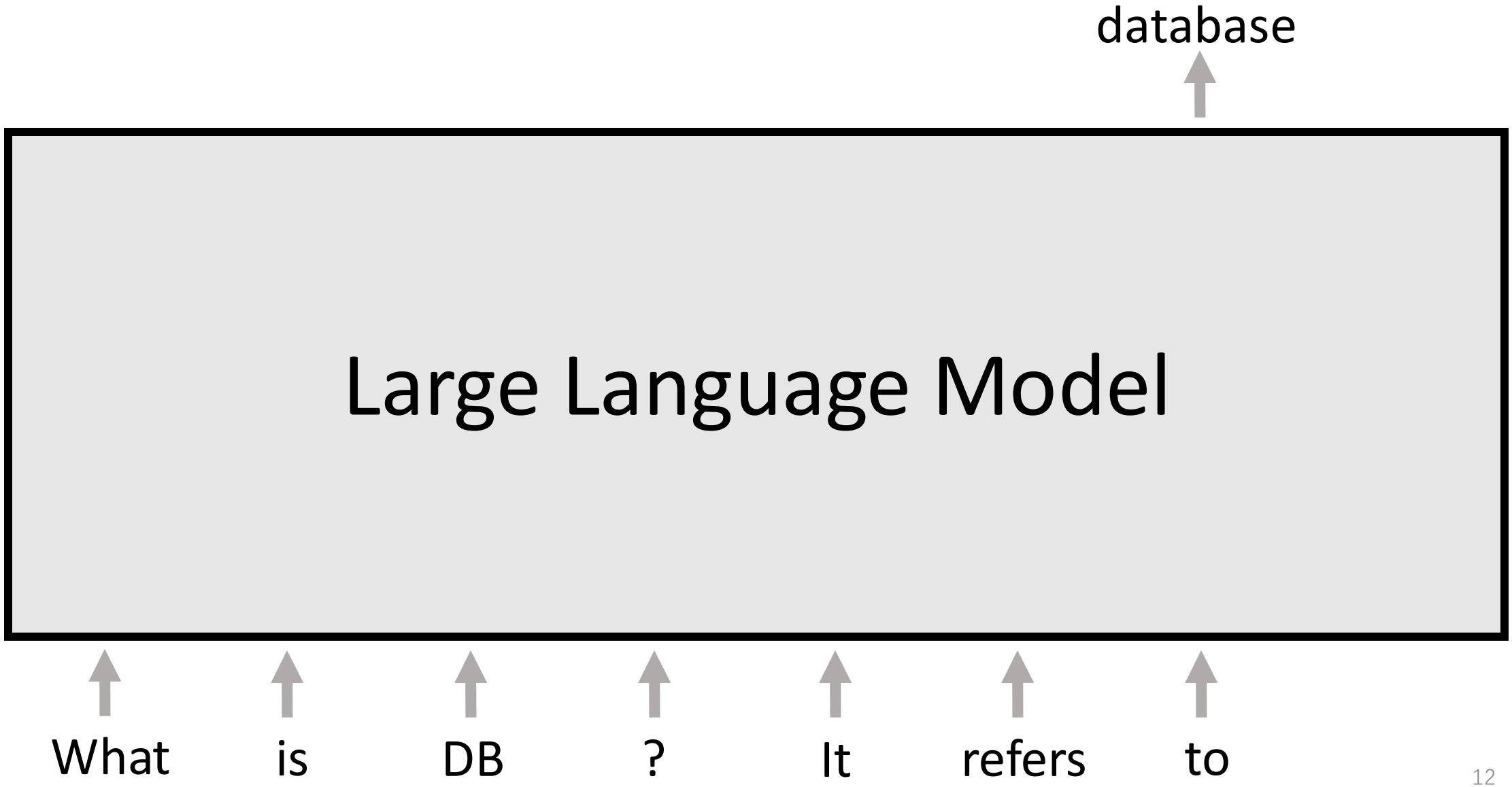
LLM inference basic



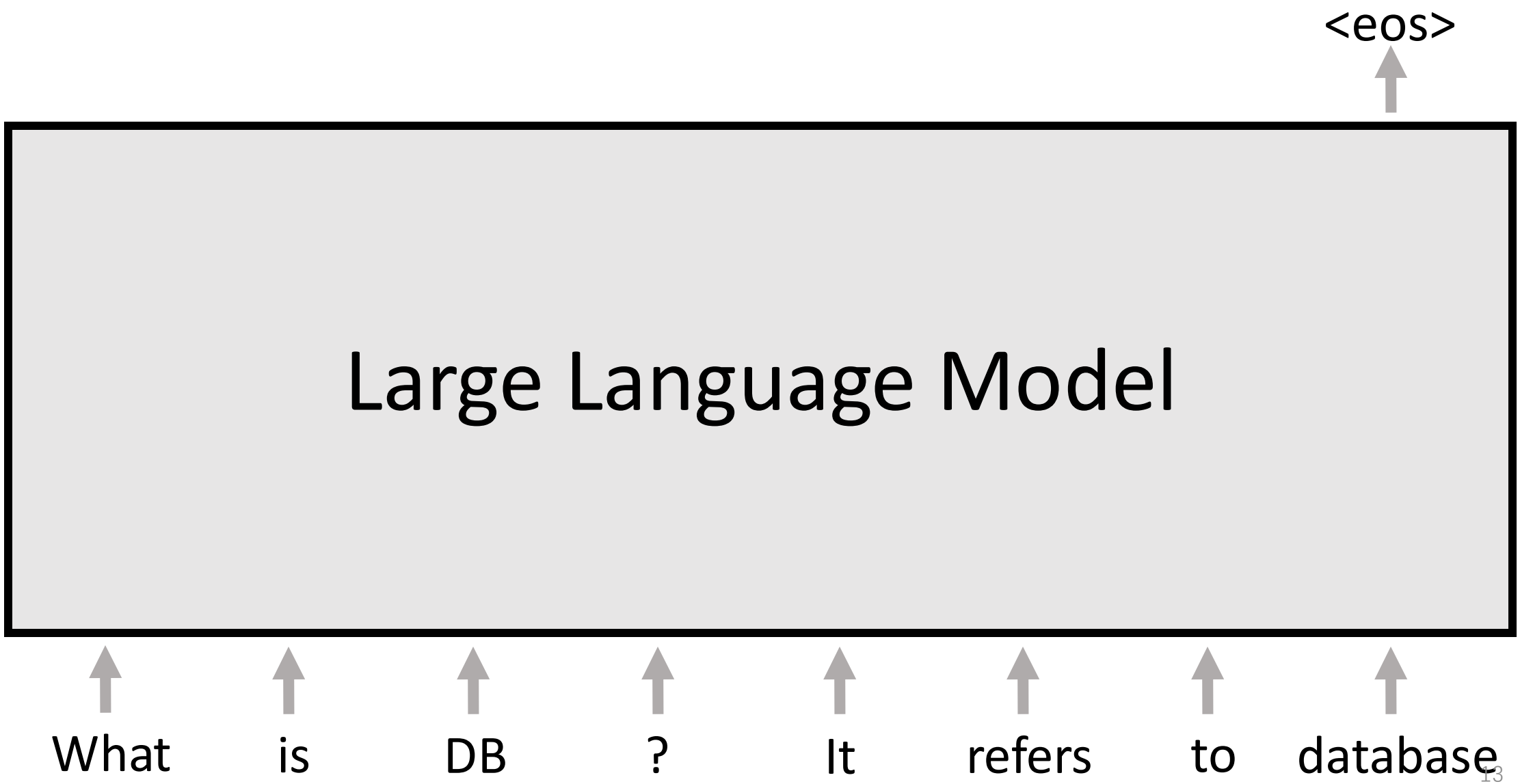
LLM inference basic



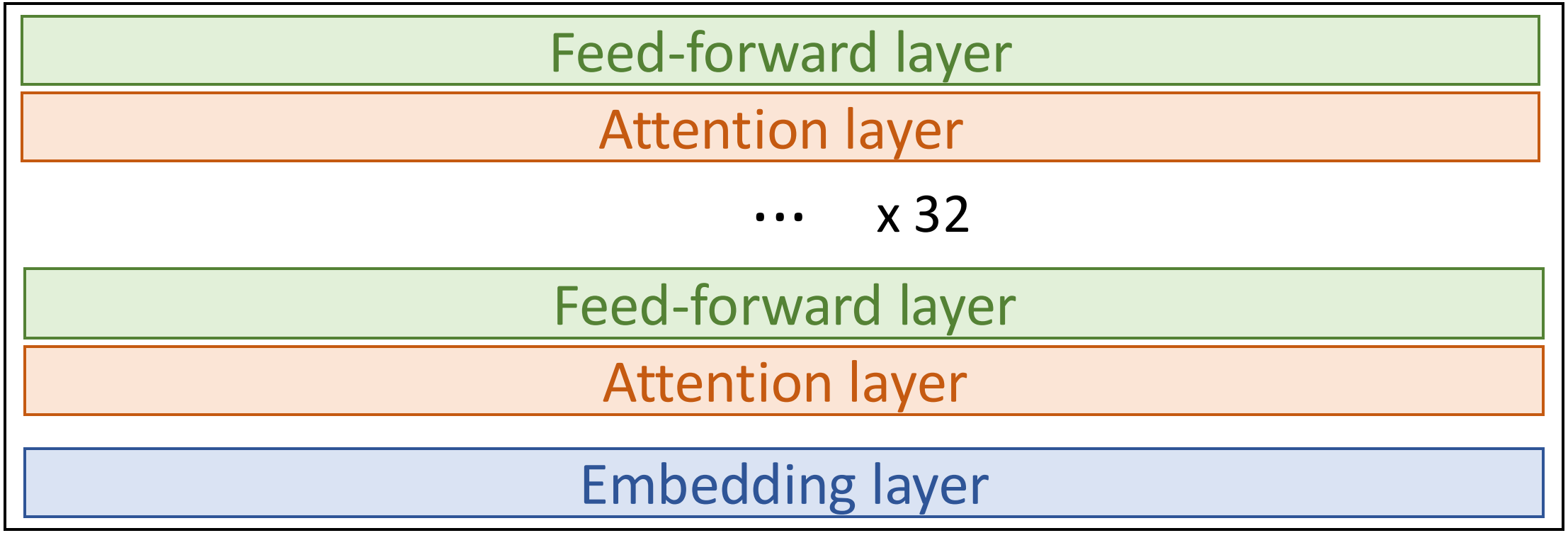
LLM inference basic



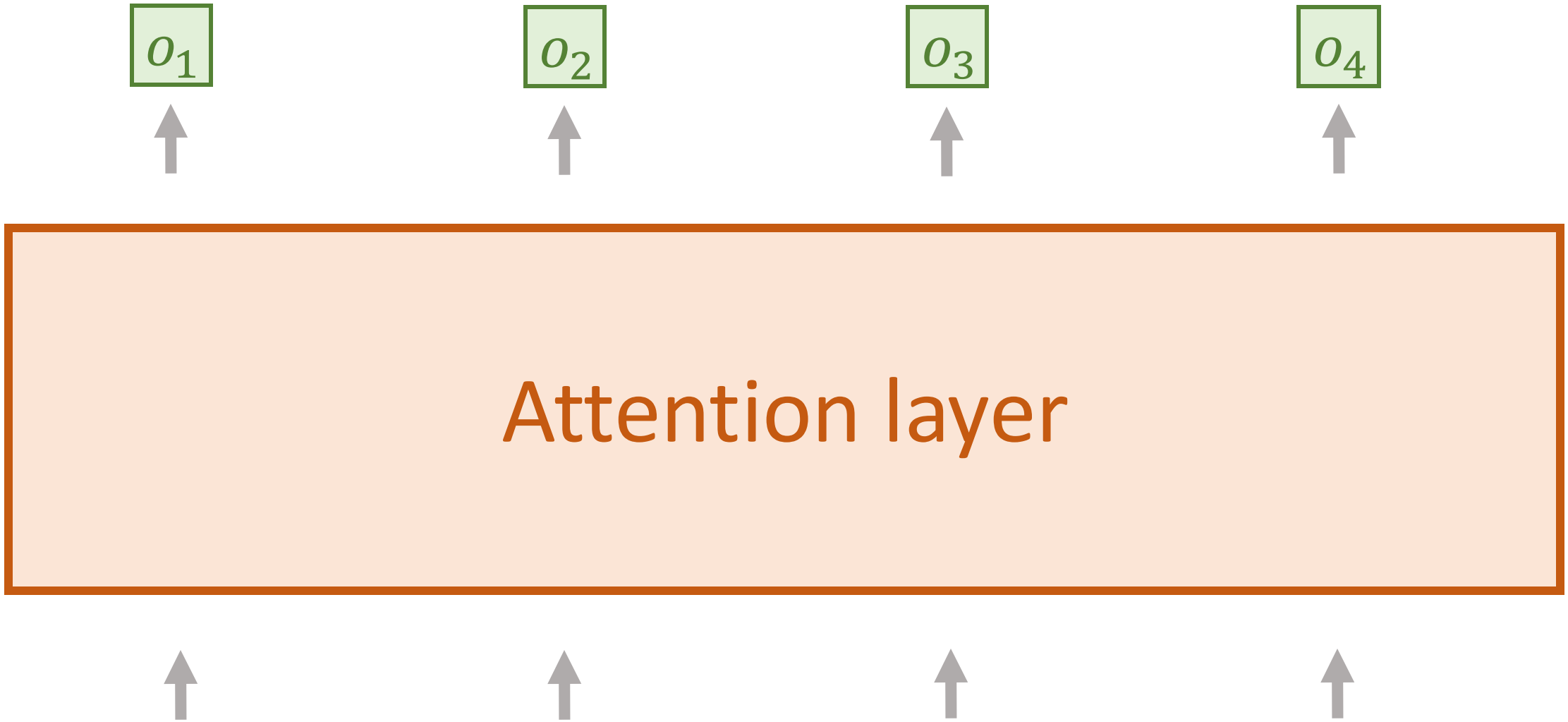
LLM inference basic



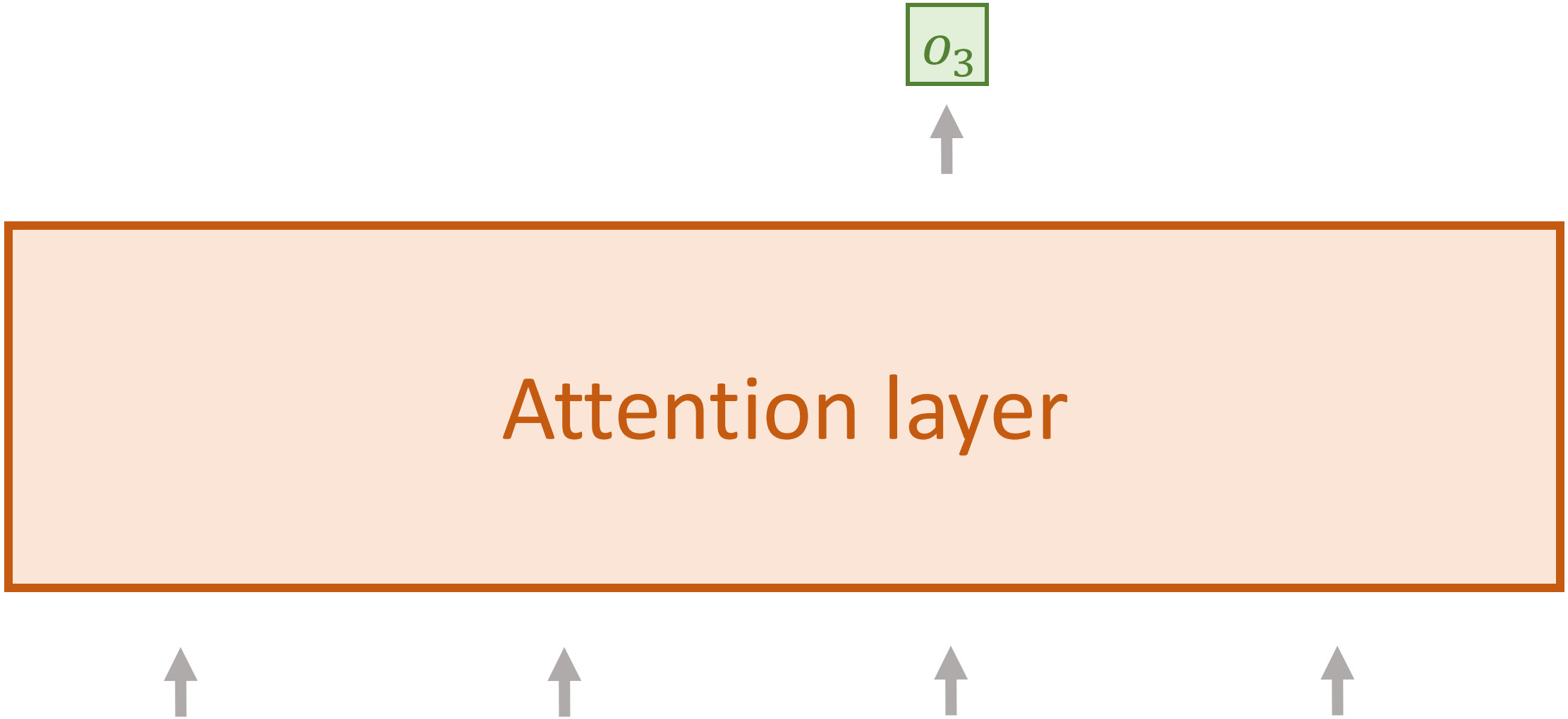
LLM inference basic



Attention



Attention

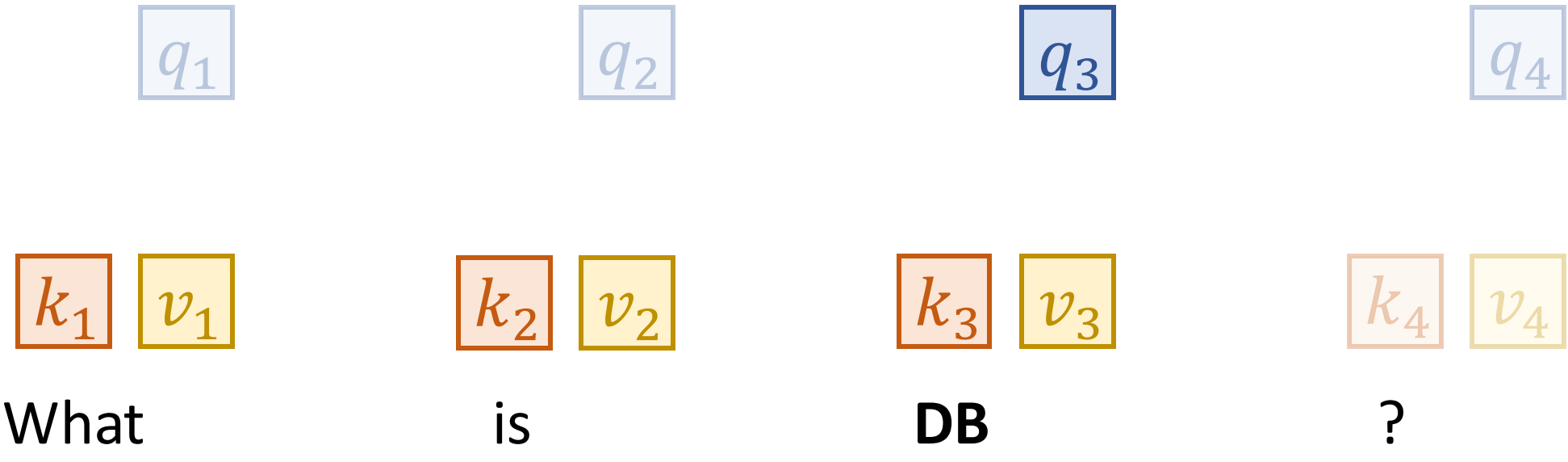


Attention

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^\top}{\sqrt{d_k}}\right) v_j$$

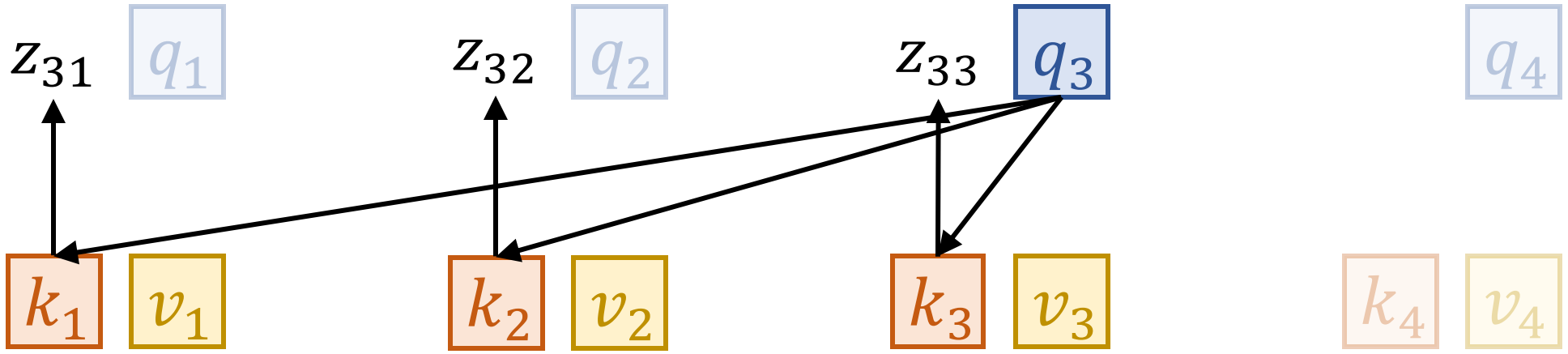
Attention

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^\top}{\sqrt{d_k}}\right) v_j$$



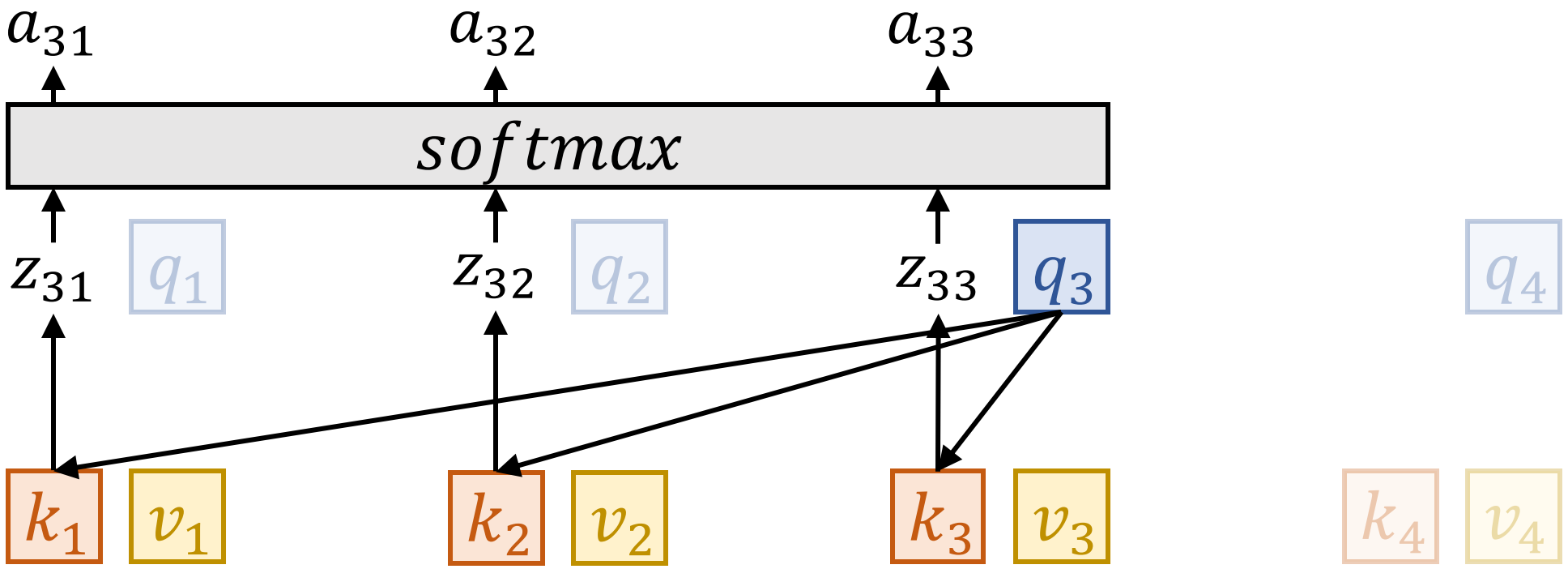
Attention

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^{\top}}{\sqrt{d_k}}\right) v_j$$



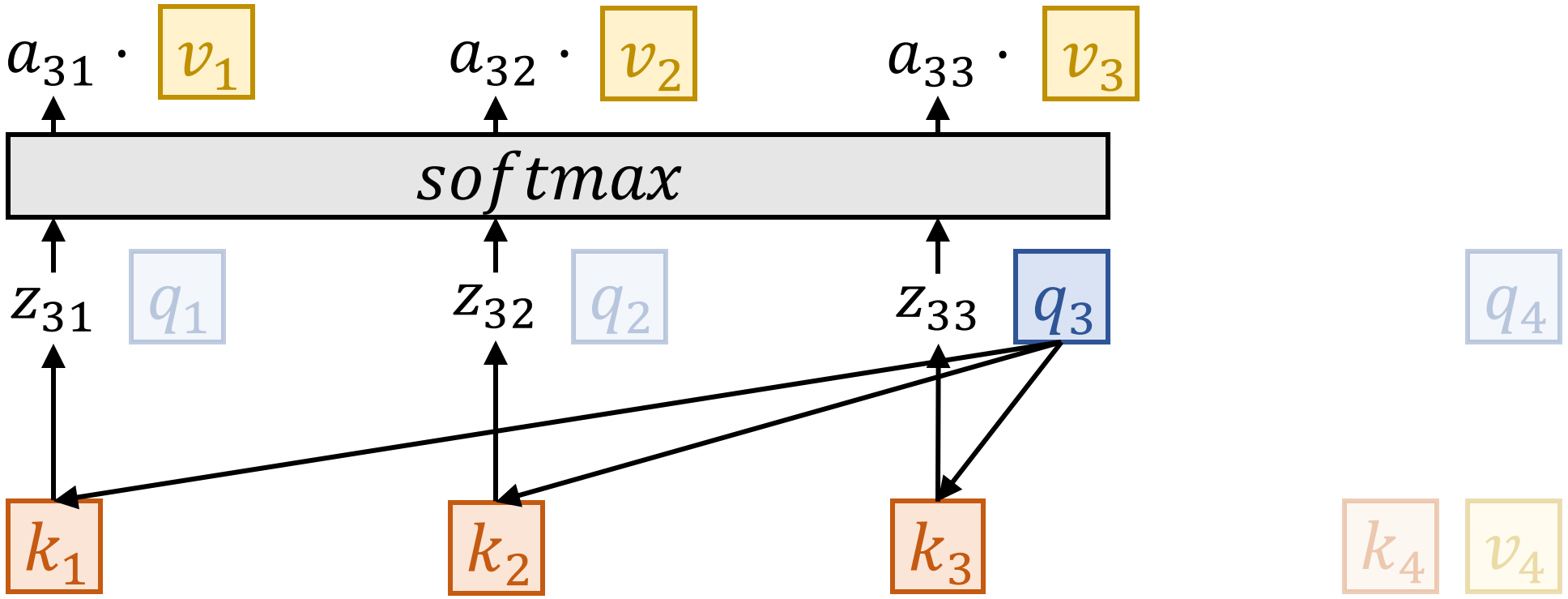
Attention

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^{\top}}{\sqrt{d_k}}\right) v_j$$



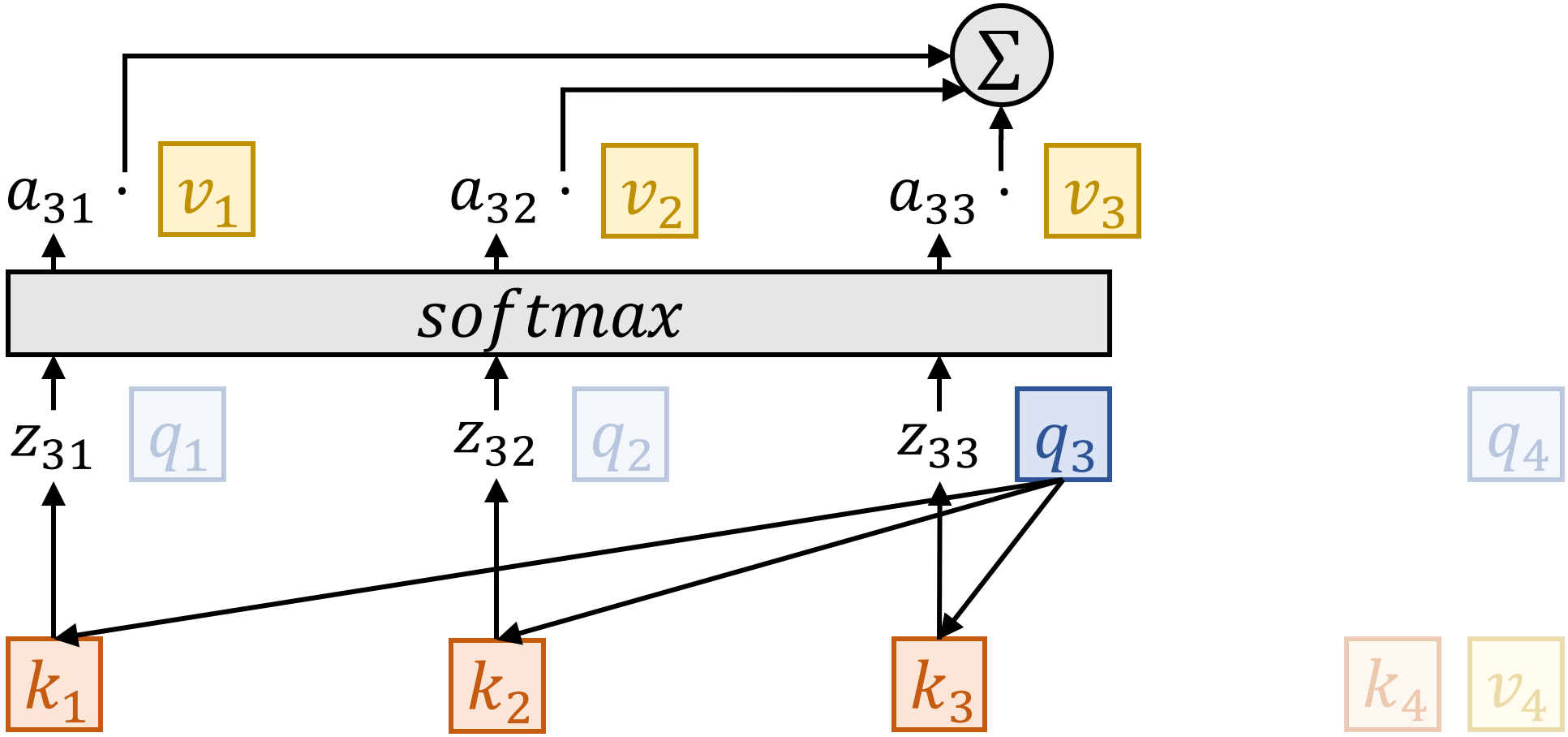
Attention

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^{\top}}{\sqrt{d_k}}\right) v_j$$



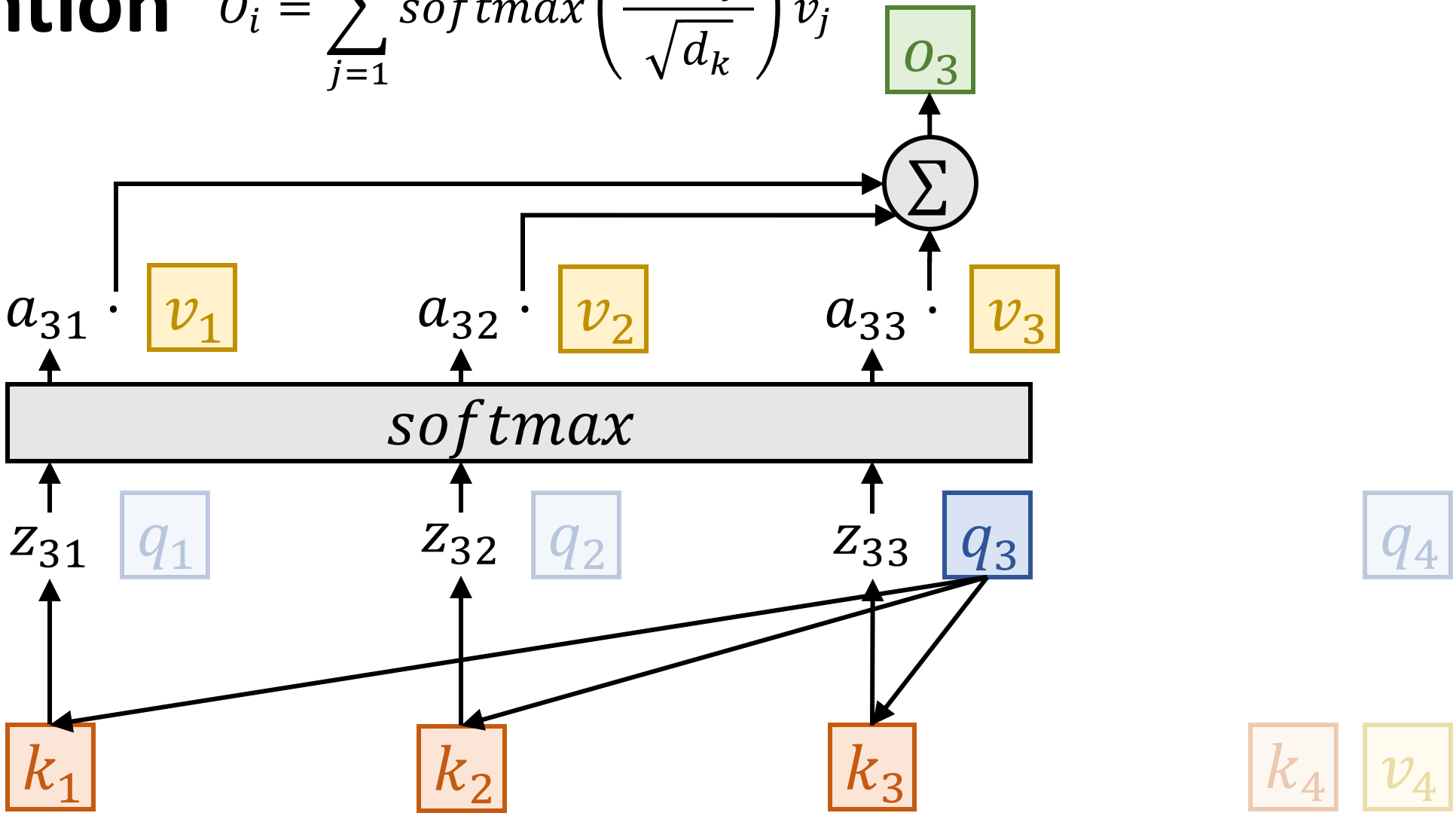
Attention

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^{\top}}{\sqrt{d_k}}\right) v_j$$

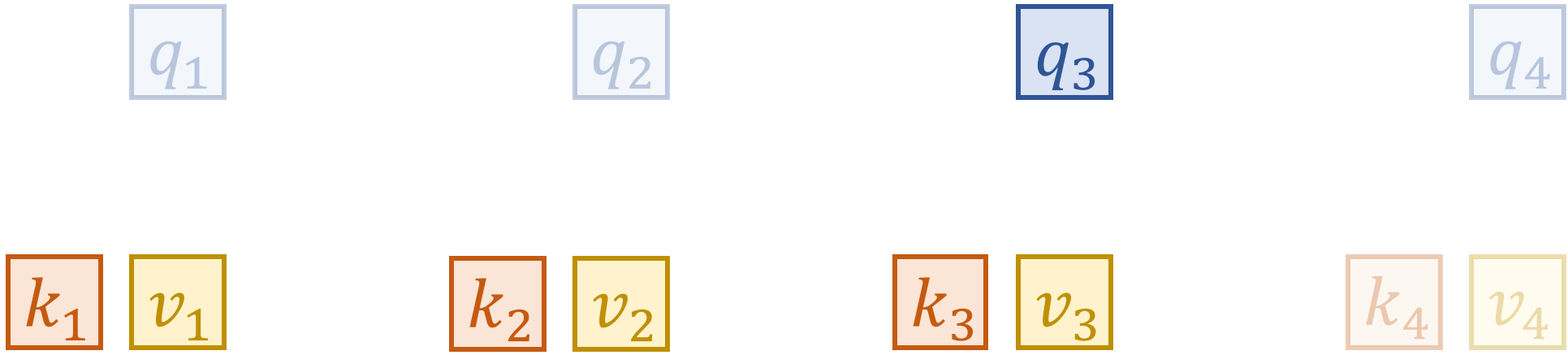


Attention

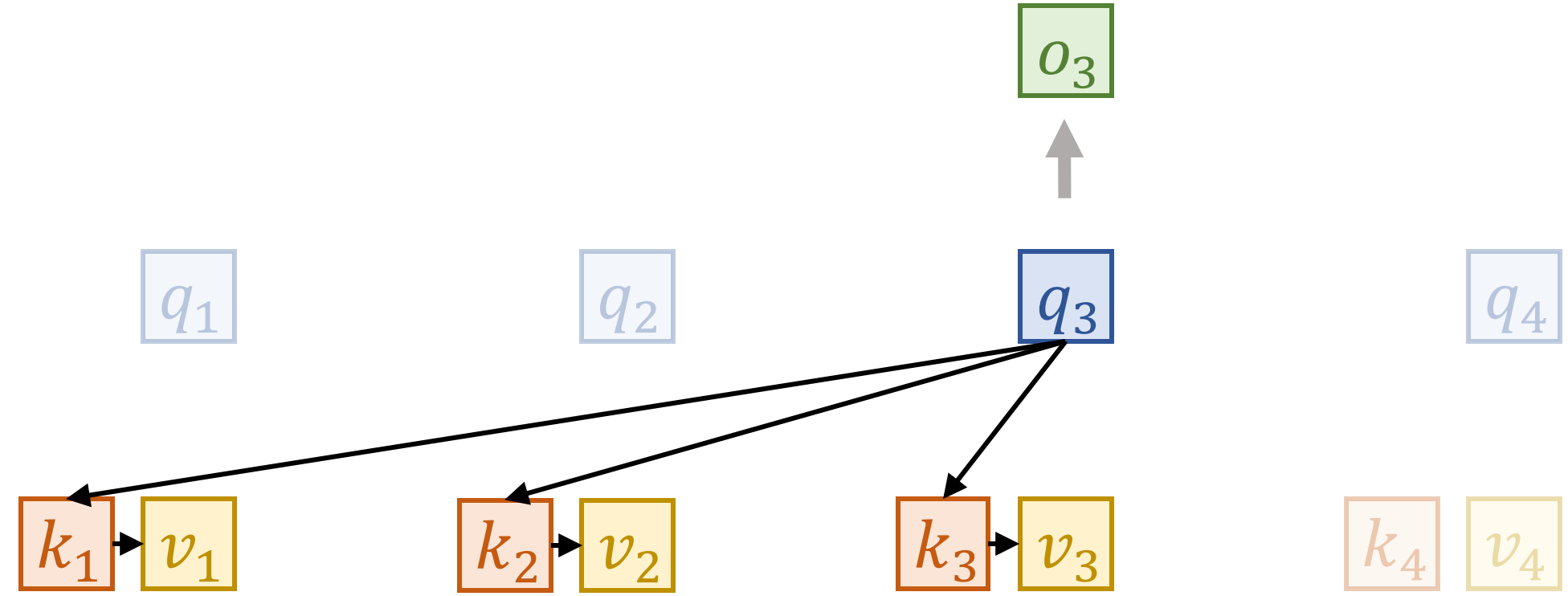
$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^{\top}}{\sqrt{d_k}}\right) v_j$$



Attention

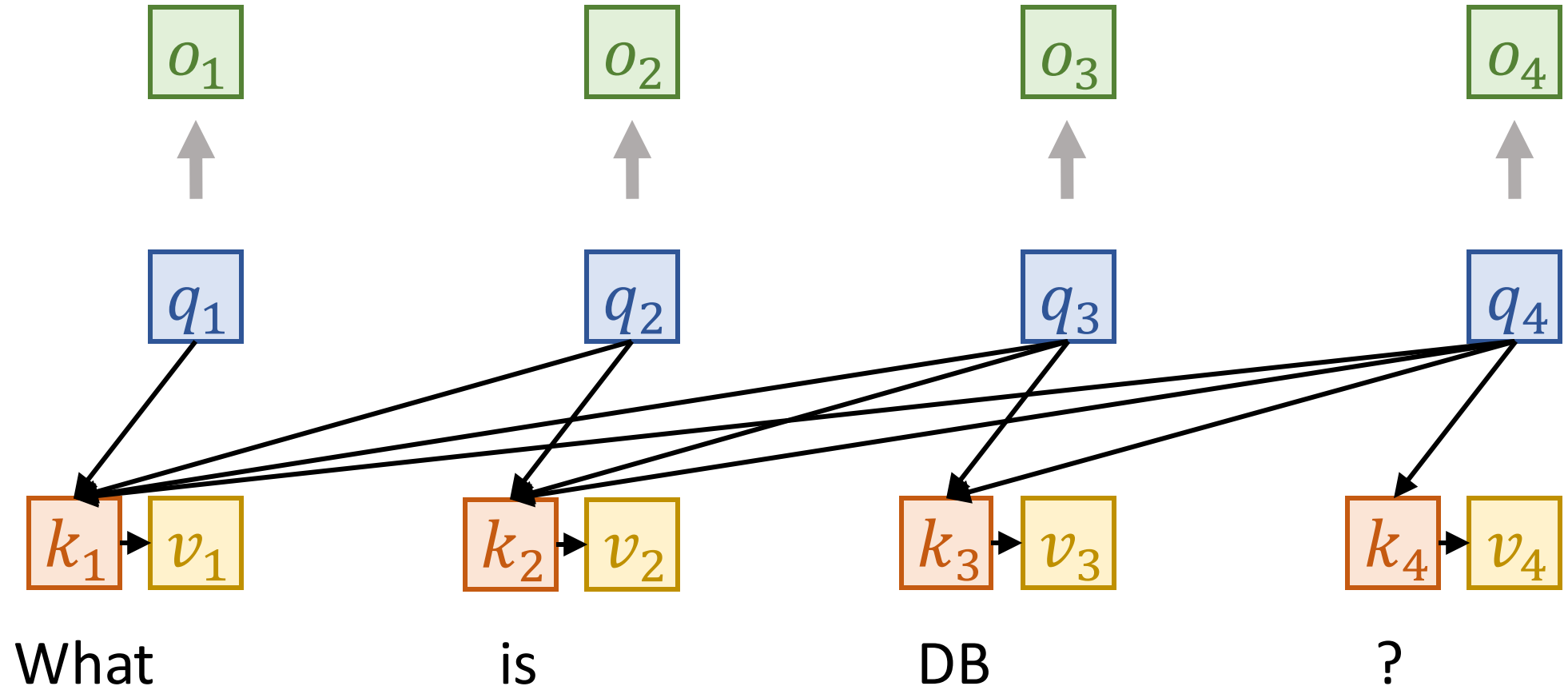


Attention



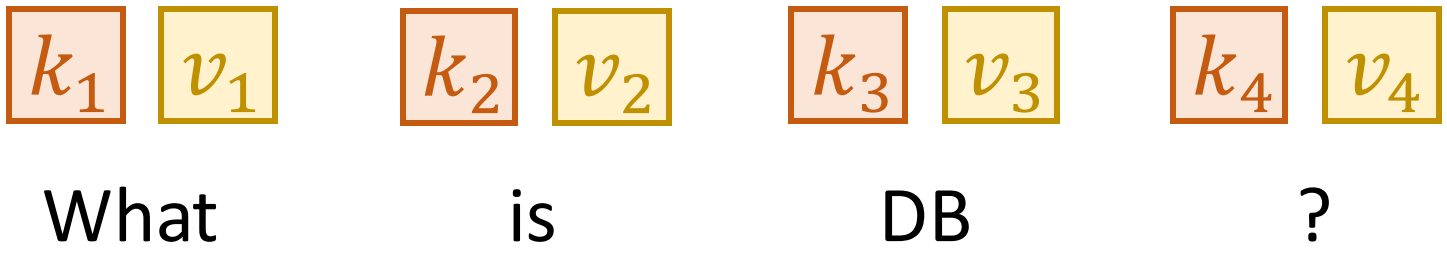
Attention

- Prefill, in parallel



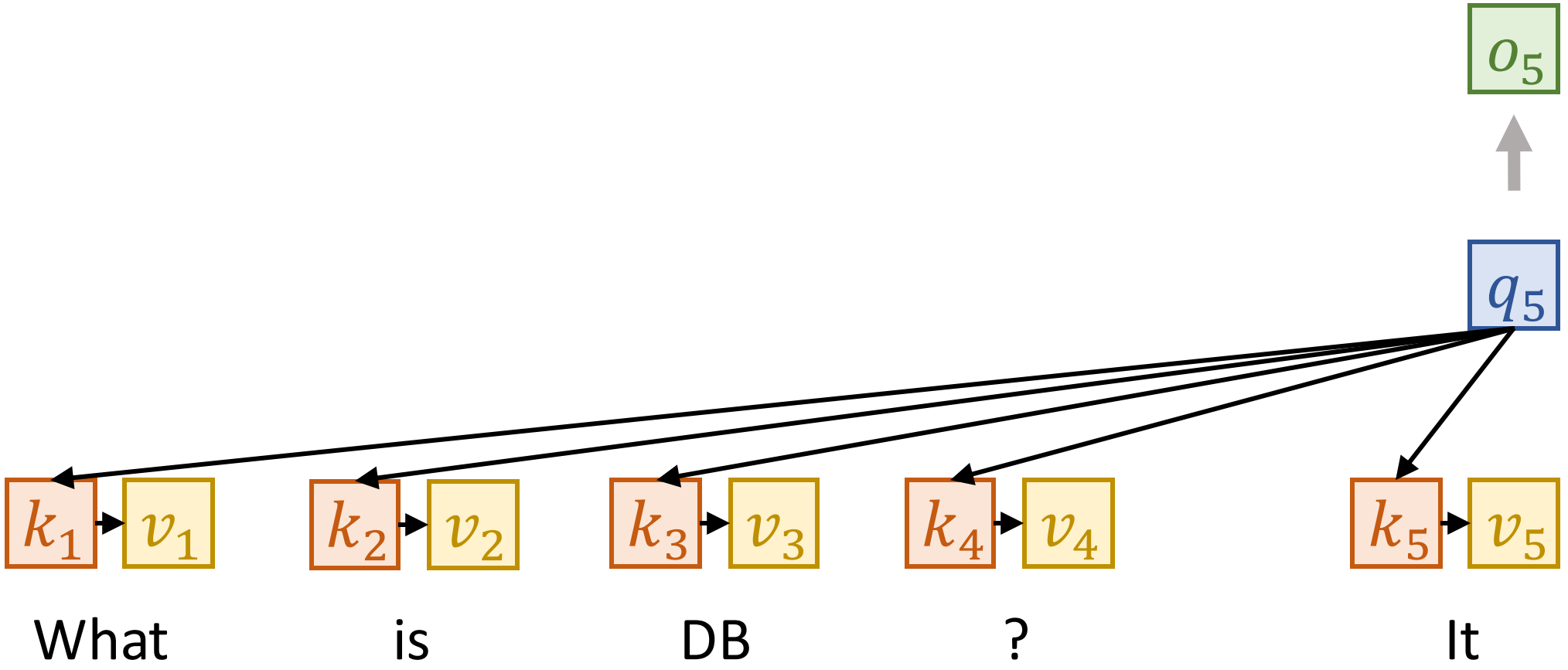
Attention

- Decode, on by one



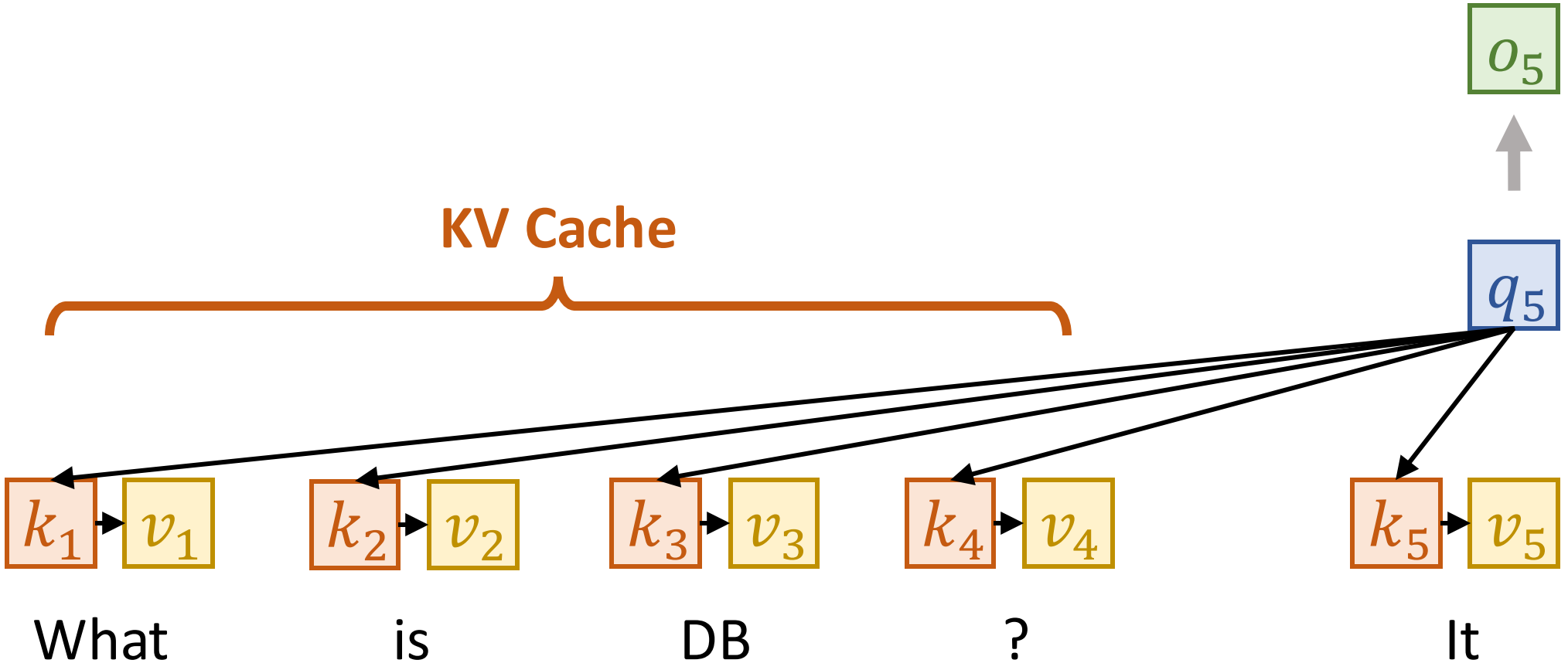
Attention

- Decode, on by one



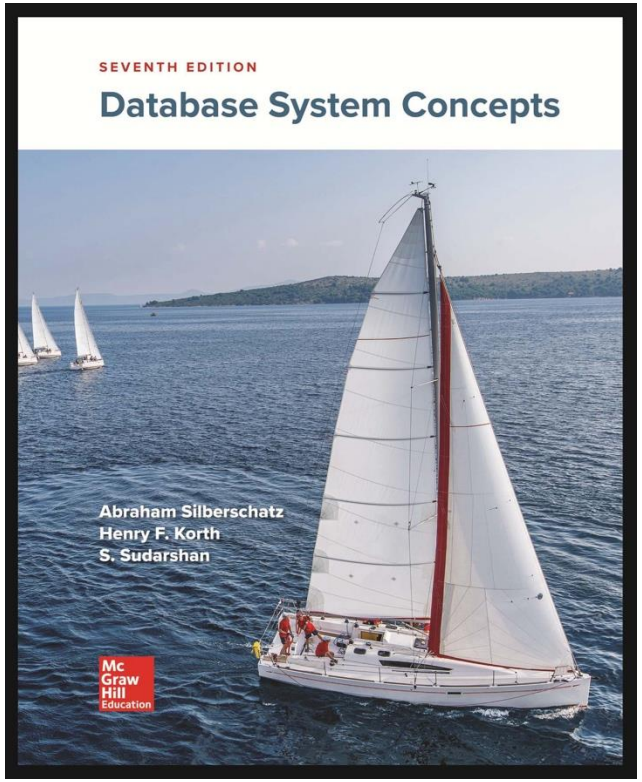
Attention

- Decode, on by one



Challenge of long context LLM inference

- **Large KV cache** – *141.38 GB*
- **Heavy attention computation** – *6 minute*

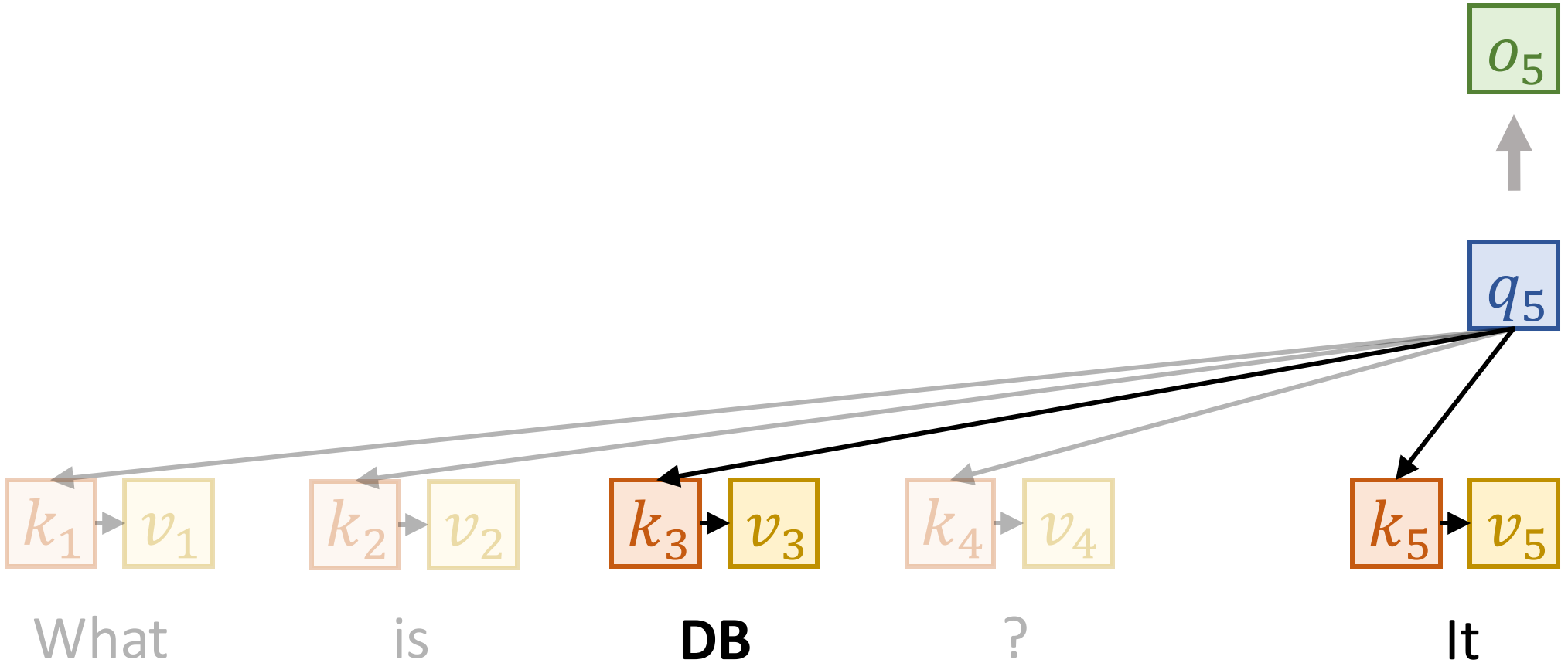


Opportunities

- **Large KV cache – *Offload & Reuse***
- **Heavy attention computation – *Sparse Attention***

Sparse attention

- A token only focuses on a specific part of context

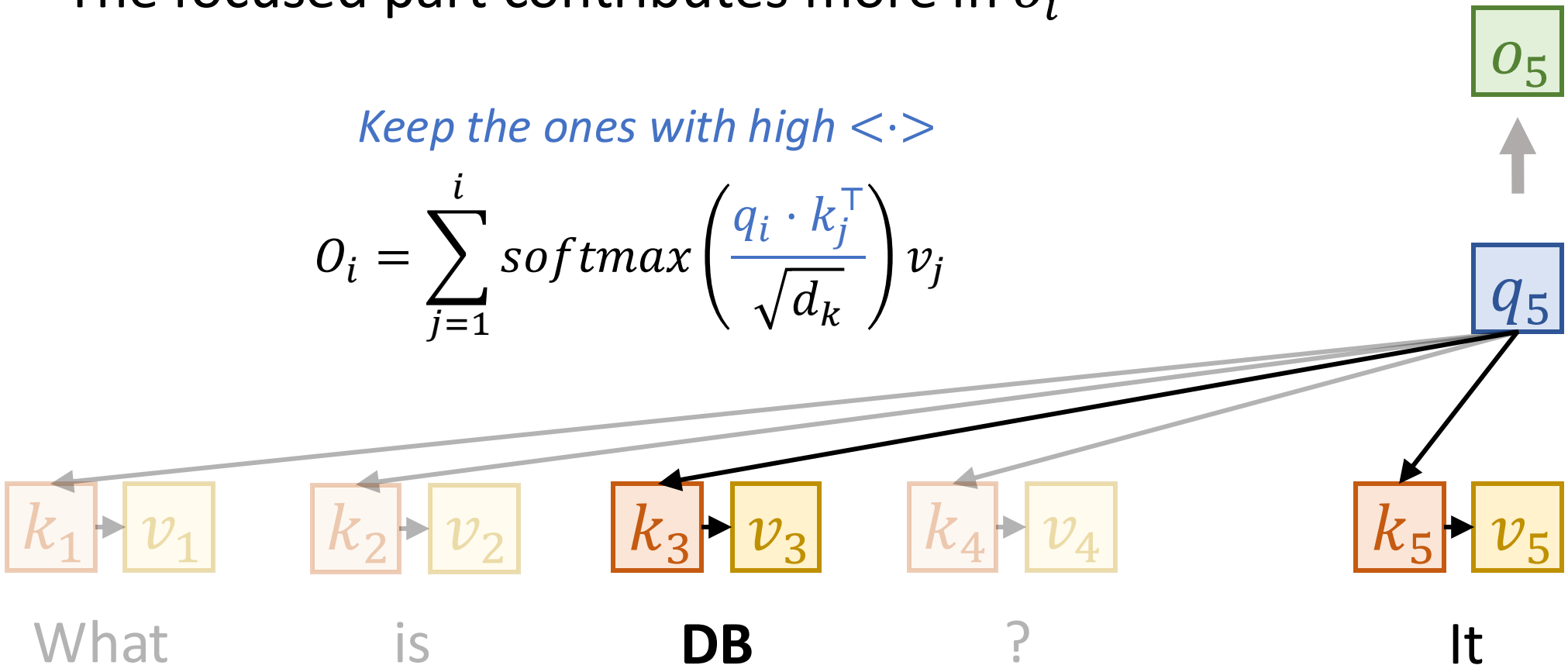


Sparse attention

- A token only focuses on a specific part of context
- The focused part contributes more in o_i

Keep the ones with high $<\cdot>$

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^\top}{\sqrt{d_k}}\right) v_j$$

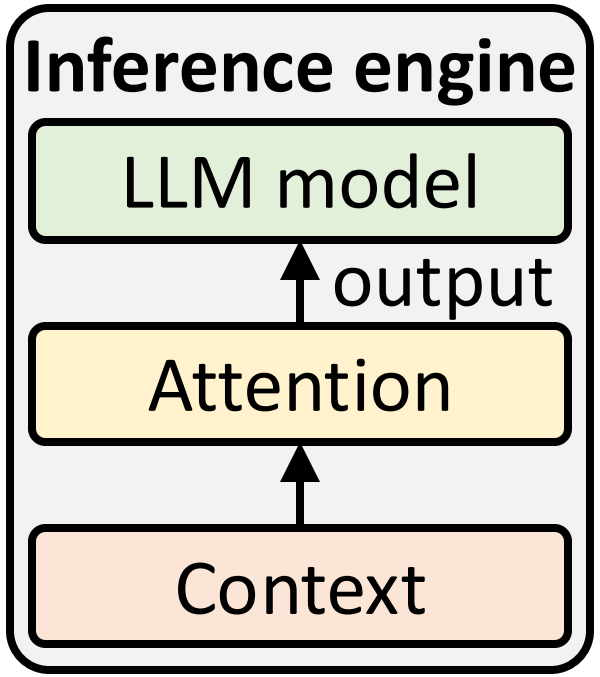


Opportunities

- **Large KV cache** – ***Offload & Reuse***
- **Heavy attention computation** – ***Sparse Attention***

Question: Are existing systems/algorithms good enough?

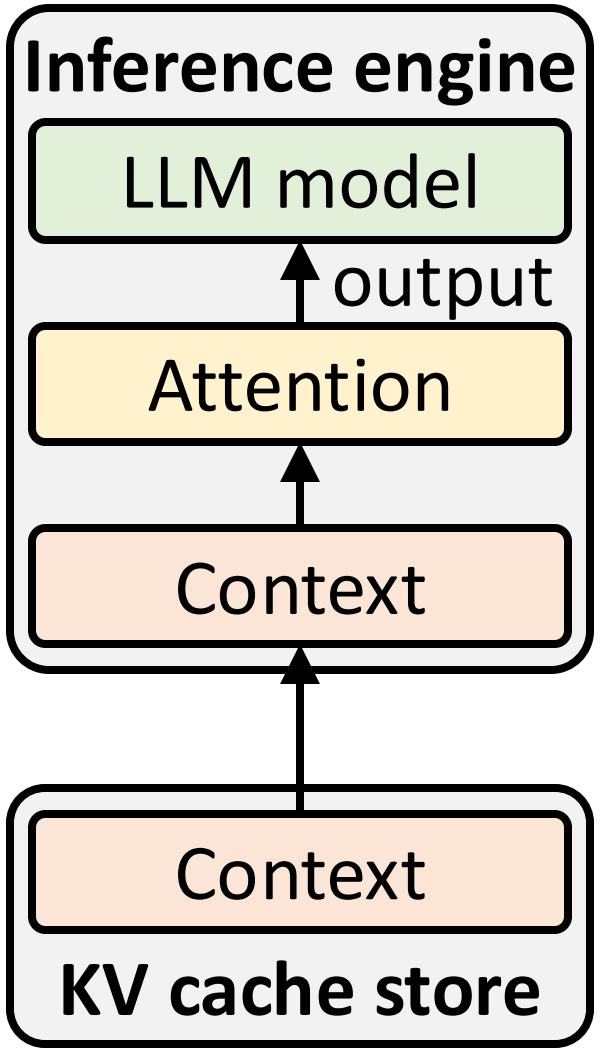
Coupled architecture



- Systems: *SGLang, vLLM*
- Manage & reuse KV cache in GPU
- Full attention

Latency	Quality	GPU memory	Usability
High	Good	Large	Good

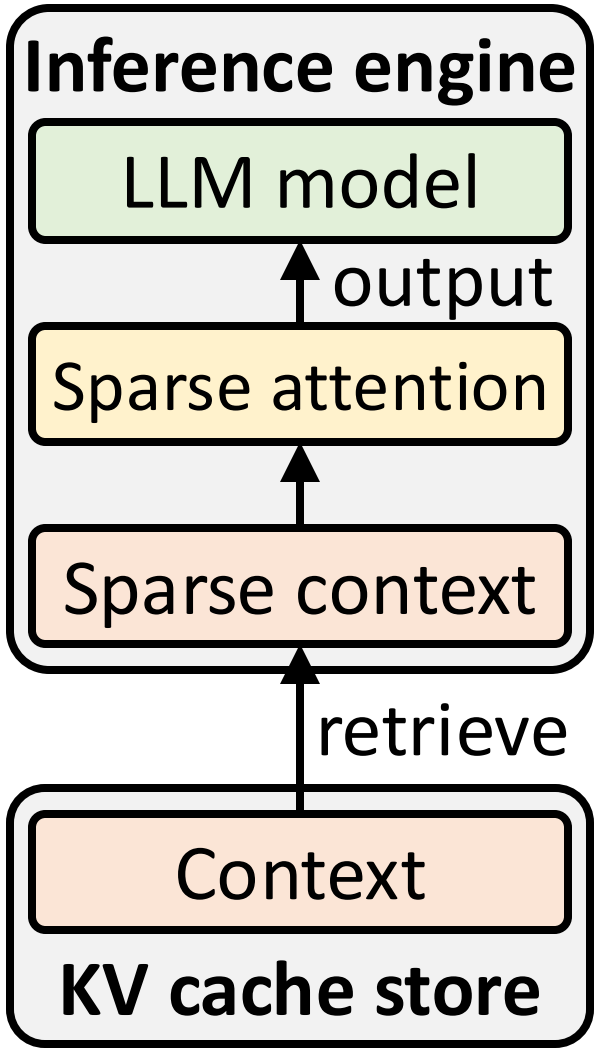
KV cache disaggregation



- Systems: *Mooncake, LMCache*
- Offload KV cache to a storage service
- Reuse KV cache by re-loading

Latency	Quality	GPU memory	Usability
Medium	High	Large	Medium

Retrieval-based sparse attention



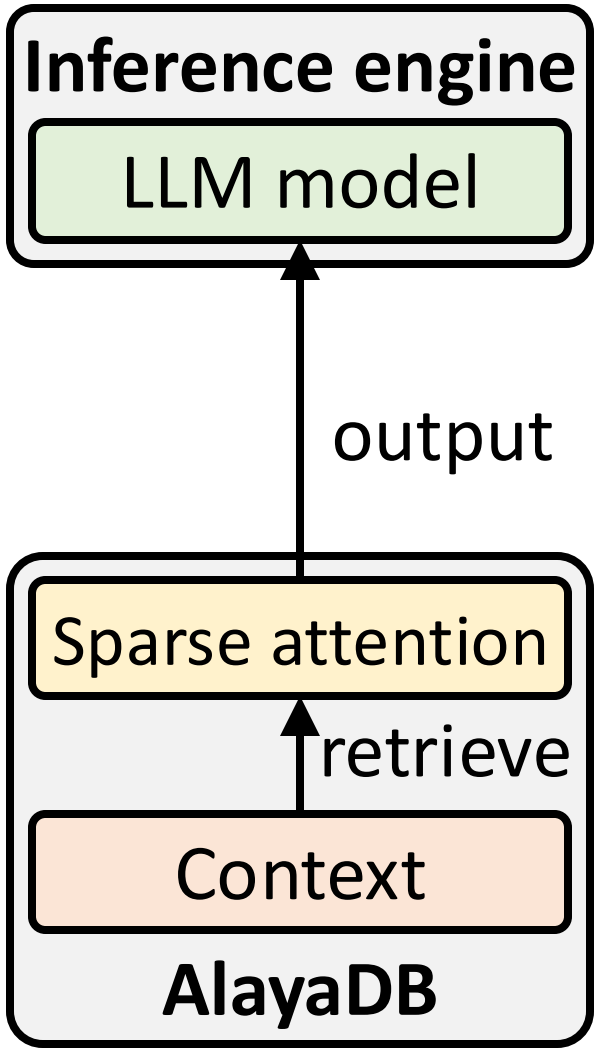
- Algorithms: *InfLLM*, *RetrievalAttention*
- Offload KV cache to a CPU
- Retrieve partial KV cache for attention

Latency	Quality	GPU memory	Usability
—	Medium	Small	Bad

How to meet all these goals?

	Latency	Quality	GPU memory	Usability
Coupled architecture	High	Good	Large	Good
KV cache disaggregation	Medium	High	Large	Medium
Retrieval-based sparse attn	—	Medium	Small	Bad
AlayaDB	**Low**	**Good**	**Small**	**Good**

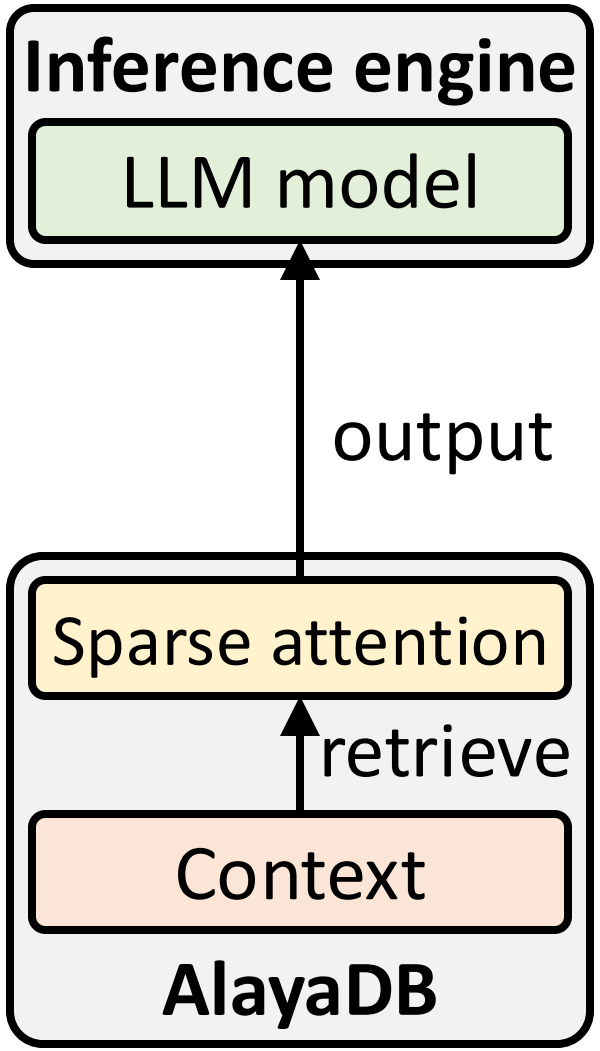
AlayaDB – New abstraction



- Decouple both *attention* & *KV cache*
- Encapsulate into a *vector database*
- Retrieve partial KV cache for attention

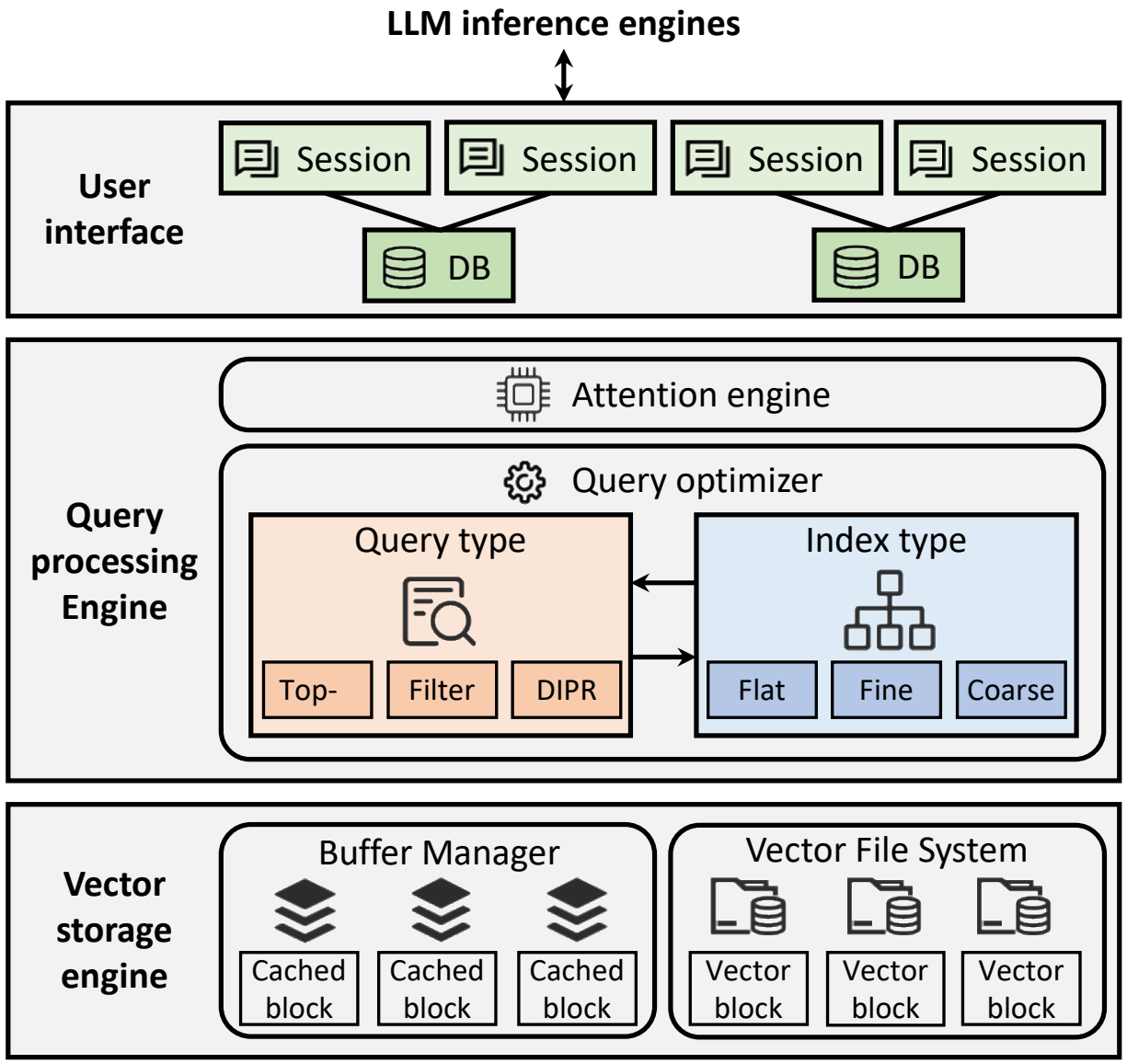
Latency	Quality	GPU memory	Usability
Low	Good	Small	Good

Seize the opportunities with DB techniques



- **Store & reuse context**
  - via *vector storage engine*
- **Sparse attention**
  - via *vector search engine*

What DB community is really good at!



- **Interface & API**
- Vector search query DIPR
- Query optimizer
- End-to-end optimizations

Simple and compatible interface

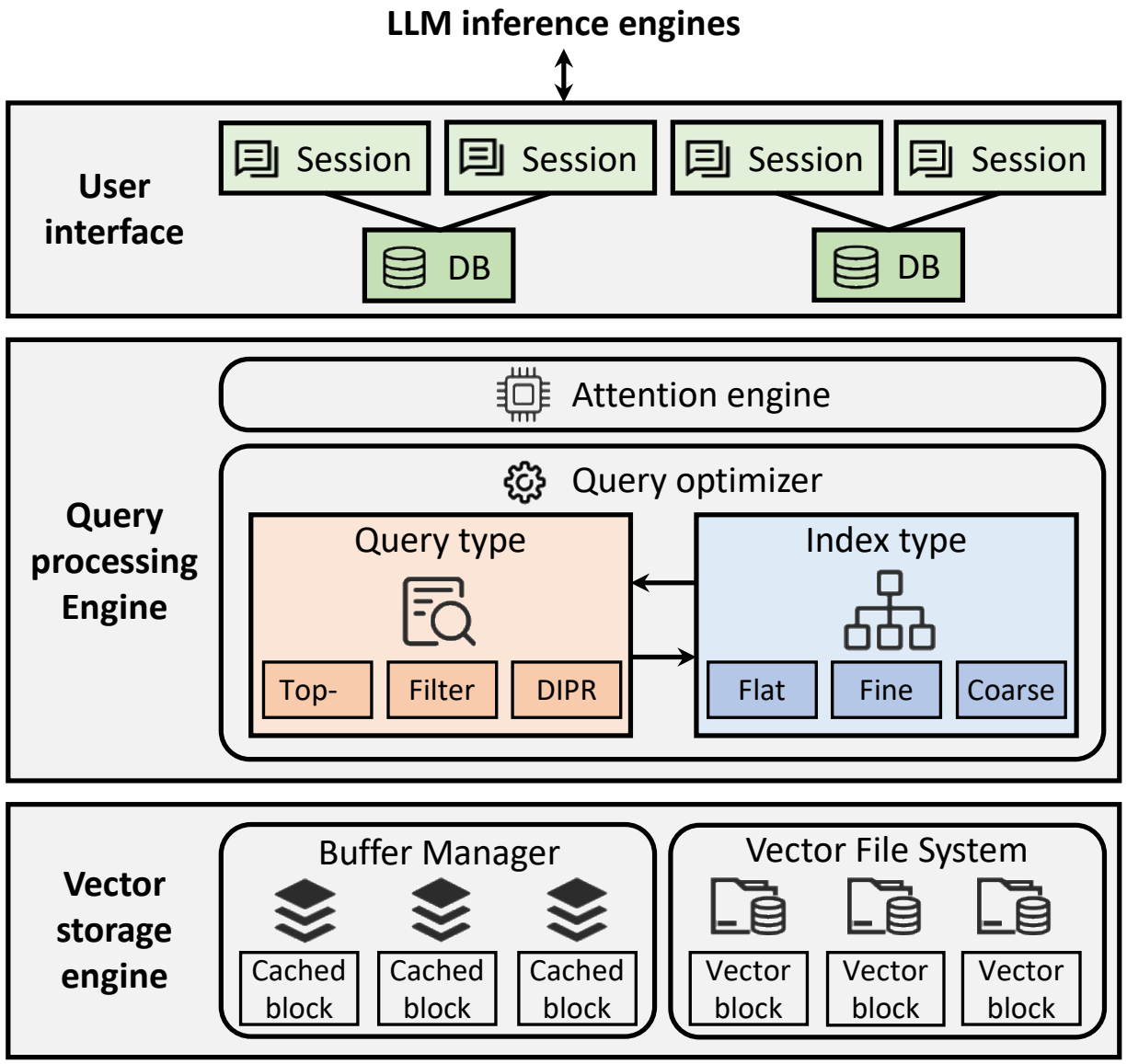
- **Session**: states of an on-going context
  - *Session* is compatible with DynamicCache() in **huggingface/transformers**
  - *Session.attention* is compatible with **flash-attention**

```
from transformers.cache_utils import DynamicCache
from flash_attn import flash_attn_func

def inference(model, prompts):
    past_key_values = DynamicCache()
    output = model(prompts, past_key_values)
    ...
class LlamaAttention:
def forward(self, ...):
    ...
    k, v = past_key_values.update(k, v, self.layer_idx)
    o = flash_attn_func(q, k, v)
    ...
```

```
from AlayaDB.LLM import DB

def inference(model, prompts):
    session, prompts = DB.CreateSession(prompts)
    past_key_values = session
    output = model(prompts, past_key_values)
    ...
class LlamaAttention:
def forward(self, ...):
    ...
    past_key_values.update(q, k, v, self.layer_idx)
    o = past_key_values.attention(q, self.layer_idx)
    ...
```



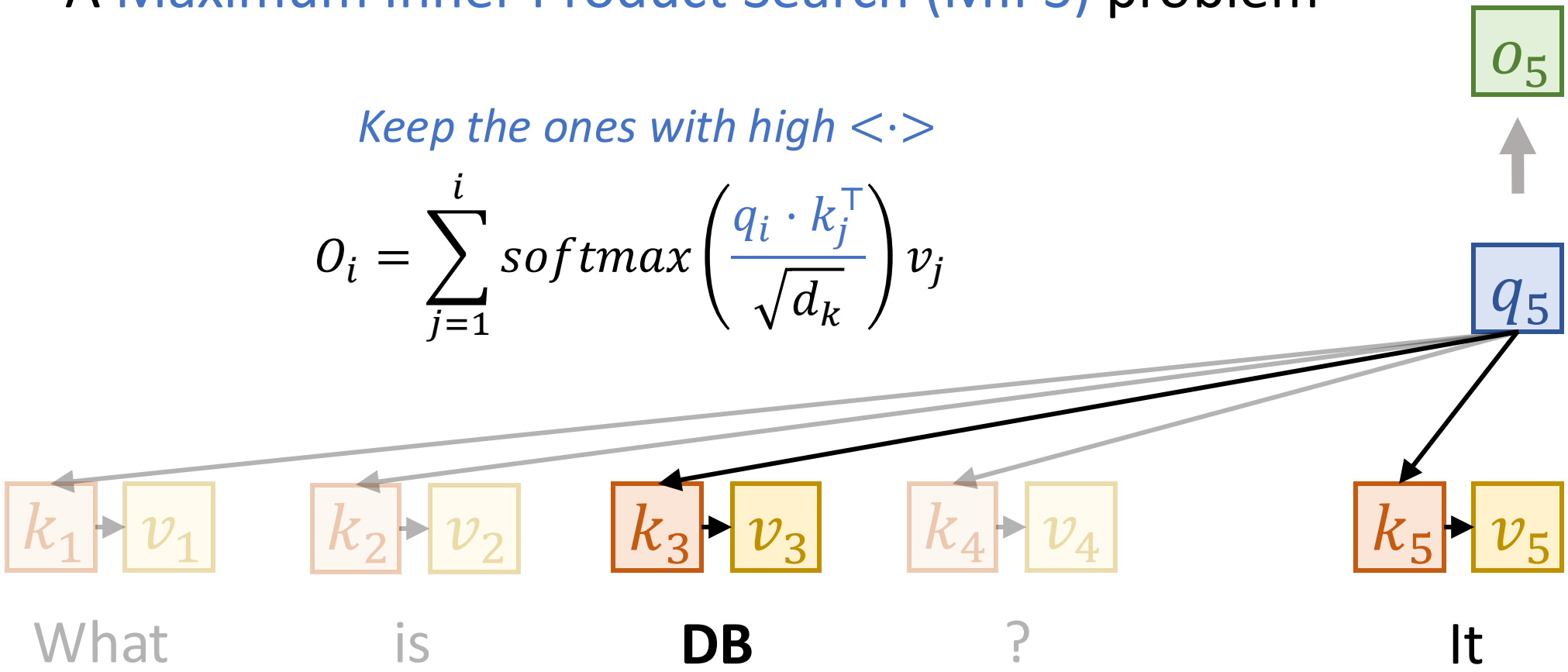
- Interface & API
- **Vector search query DIPR**
- Query optimizer
- End-to-end optimizations

From sparse attention to vector search

- How to locate the tokens with high attention score
- A Maximum Inner Product Search (MIPS) problem

Keep the ones with high $<\cdot>$

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^\top}{\sqrt{d_k}}\right) v_j$$



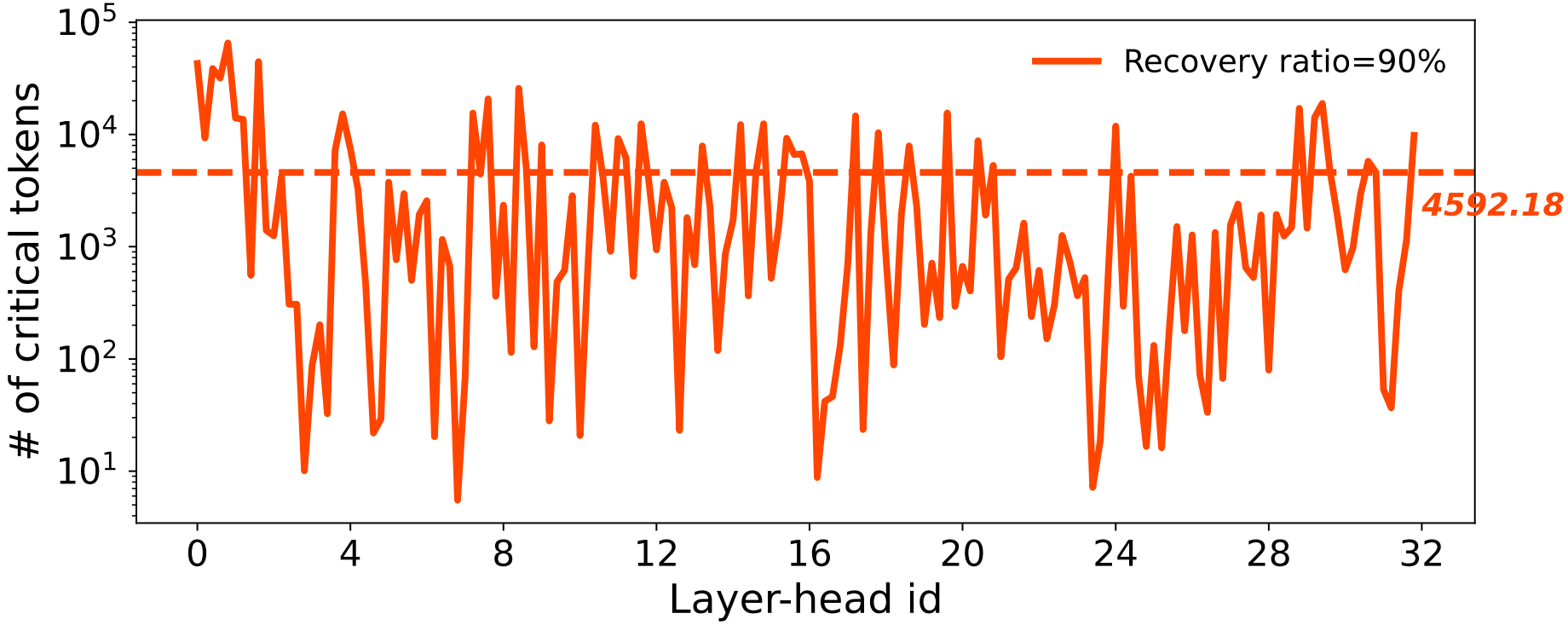
Limitation of Top-K

- Existing *sparse attention* & *vector search* works focus on **Top-K**
- However, it does not match the goal of attention approximation
  - Top-K focuses on the ***ranking***
- We want the selected attention weight a_{ij} close to the full attention
  - We want to focus on the ***score!***

$$O_i = \sum_{j=1}^{i} softmax\left(\frac{q_i \cdot k_j^\top}{\sqrt{d_k}}\right) v_j$$

of critical token is dynamic

- Different **heads** need different # critical tokens



of critical token is dynamic

- Different heads need different # critical tokens
- Different **tasks** need different # critical tokens

Task	k	propotion	Task	k	propotion
Qasper	350	9.67%	LCC	65	5.26%
Passage R.	250	2.69%	HotpotQA	200	2.19%
QMSum	150	1.41%	TriviaQA	20	0.24%

of critical token is dynamic

- Different heads need different # critical tokens
- Different tasks need different # critical tokens

- Can we design a new vector search target?
  - meets this dynamicity
  - can be searched efficiently

Dynamic Inner Product Range Query (DIPR)

AlayaDB 太昊信息

- **Intuition**
  - Use the largest attention weight as the pivot
  - Drop attention weights that are too smaller than the largest one

$$a_{ij} > \alpha \times \max_{s \in [1,n]} (a_{is})$$

Dynamic Inner Product Range Query (DIPR)

AlayaDB 太昊信息

- **Intuition**
  - Use the largest attention weight as the pivot
  - Drop attention weights that are too smaller than the largest one

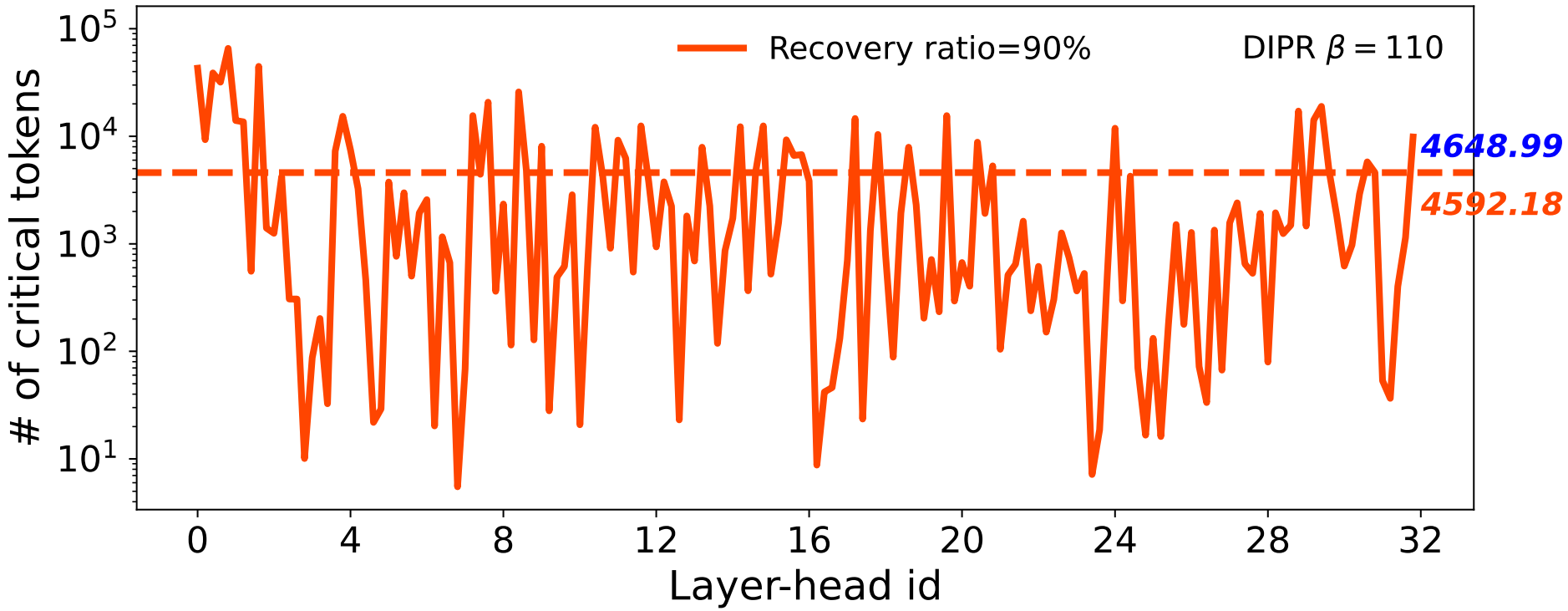
$$a_{ij} > \alpha \times \max_{s \in [1,n]} (a_{is})$$

- The formula can be transformed into a kind of range query on <·>

$$q_i \cdot k_j^\top > \max_{s \in [1,n]} (q_i \cdot k_s^\top) - \beta$$

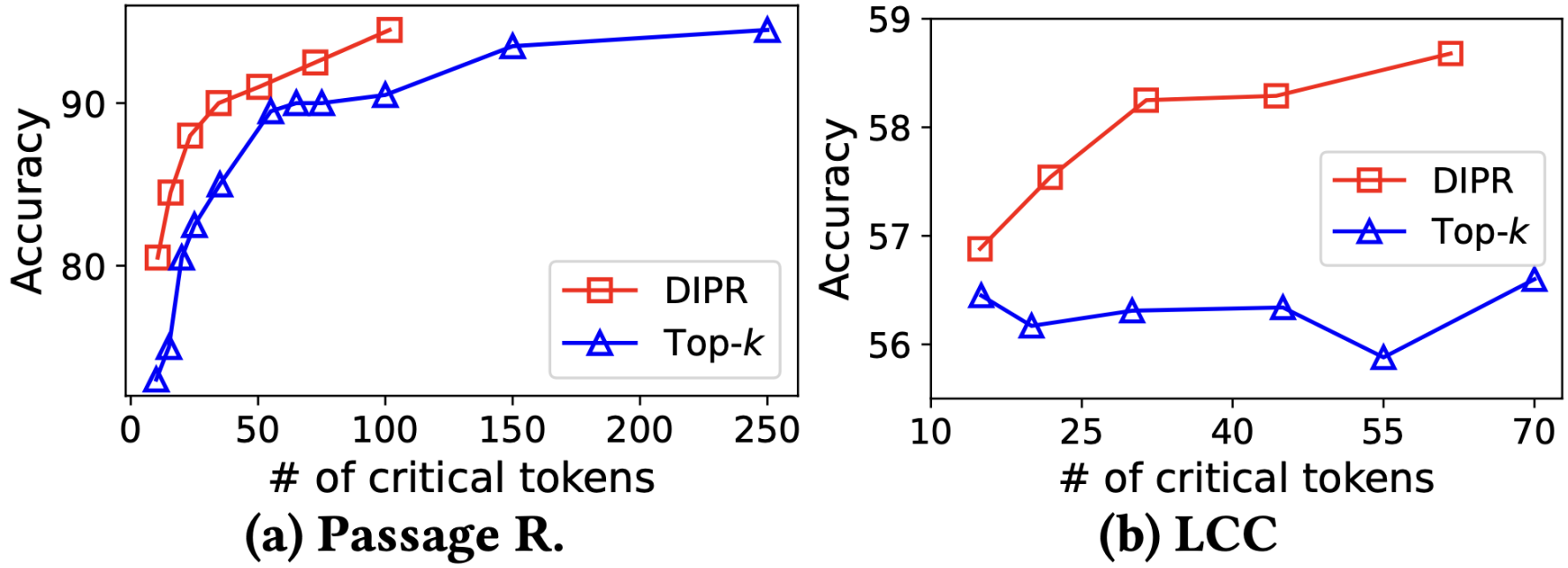
Dynamic Inner Product Range Query (DIPR)

- Can capture the dynamicity by giving a constant β



Dynamic Inner Product Range Query (DIPR)

- Can capture the dynamicity by giving a constant β



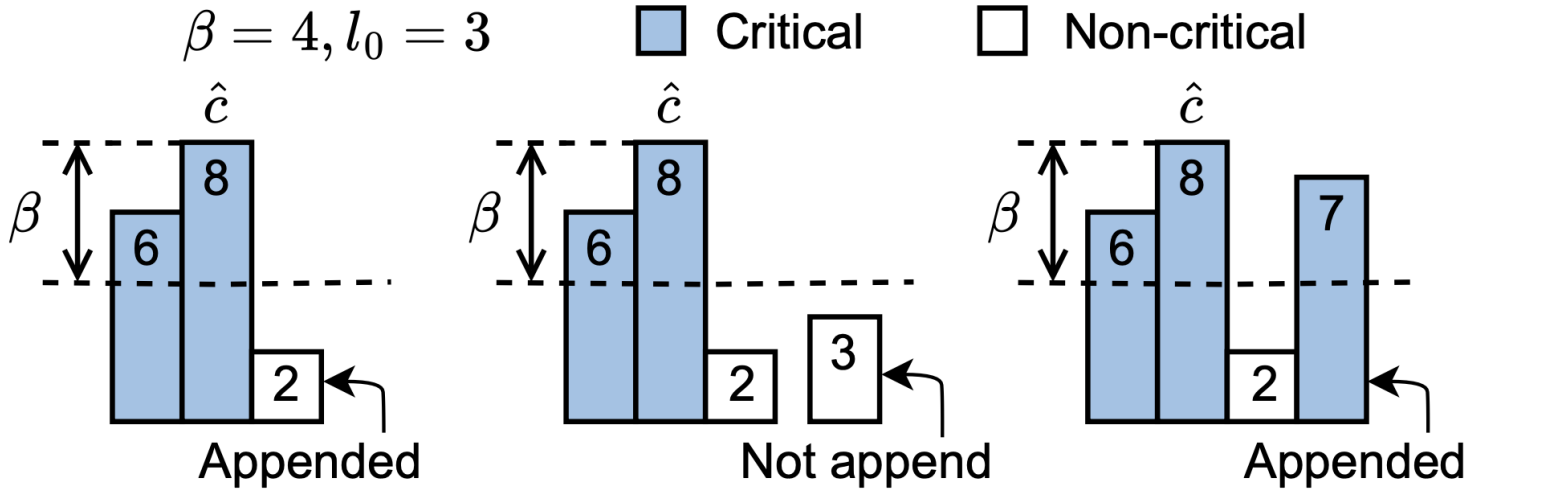
Challenges of DIPR search

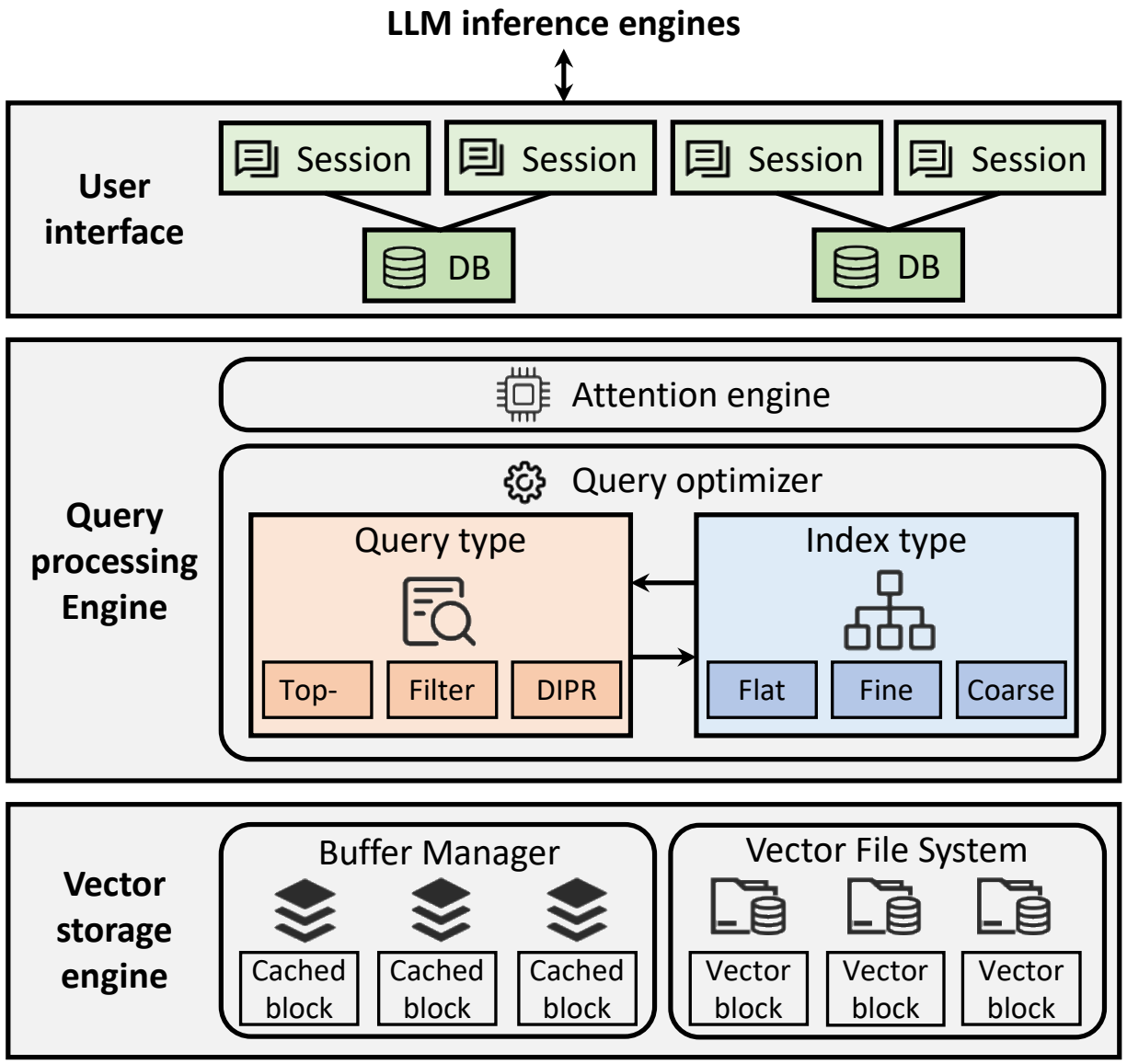
- The maximum value is unkown
  - Must not stop before the maximum value is founded
- The number of required points is unkown
  - Must converge and stop after required points are founded

$$q_i \cdot k_j^\top > \max_{s \in [1,n]} (q_i \cdot k_s^\top) - \beta$$

Efficient DIPR search on graph index

- Set a capacity threshold l_0
- Candidate list < l_0: explore every nodes *ensure max is founded*
- Candidate list > l_0: explore only critical nodes *ensure convergence*





- Interface & API
- Vector search query DIPR
- **Query optimizer**
- End-to-end optimizations

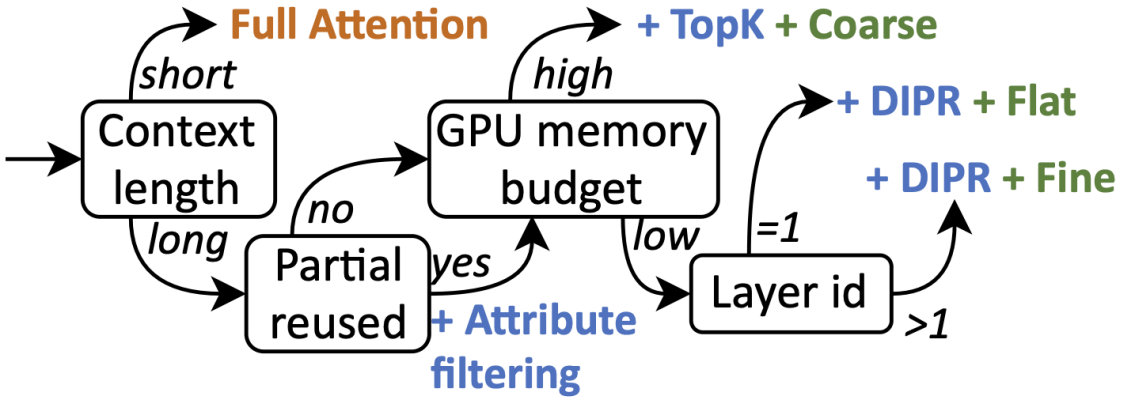
Unifying sparse attention algorithms

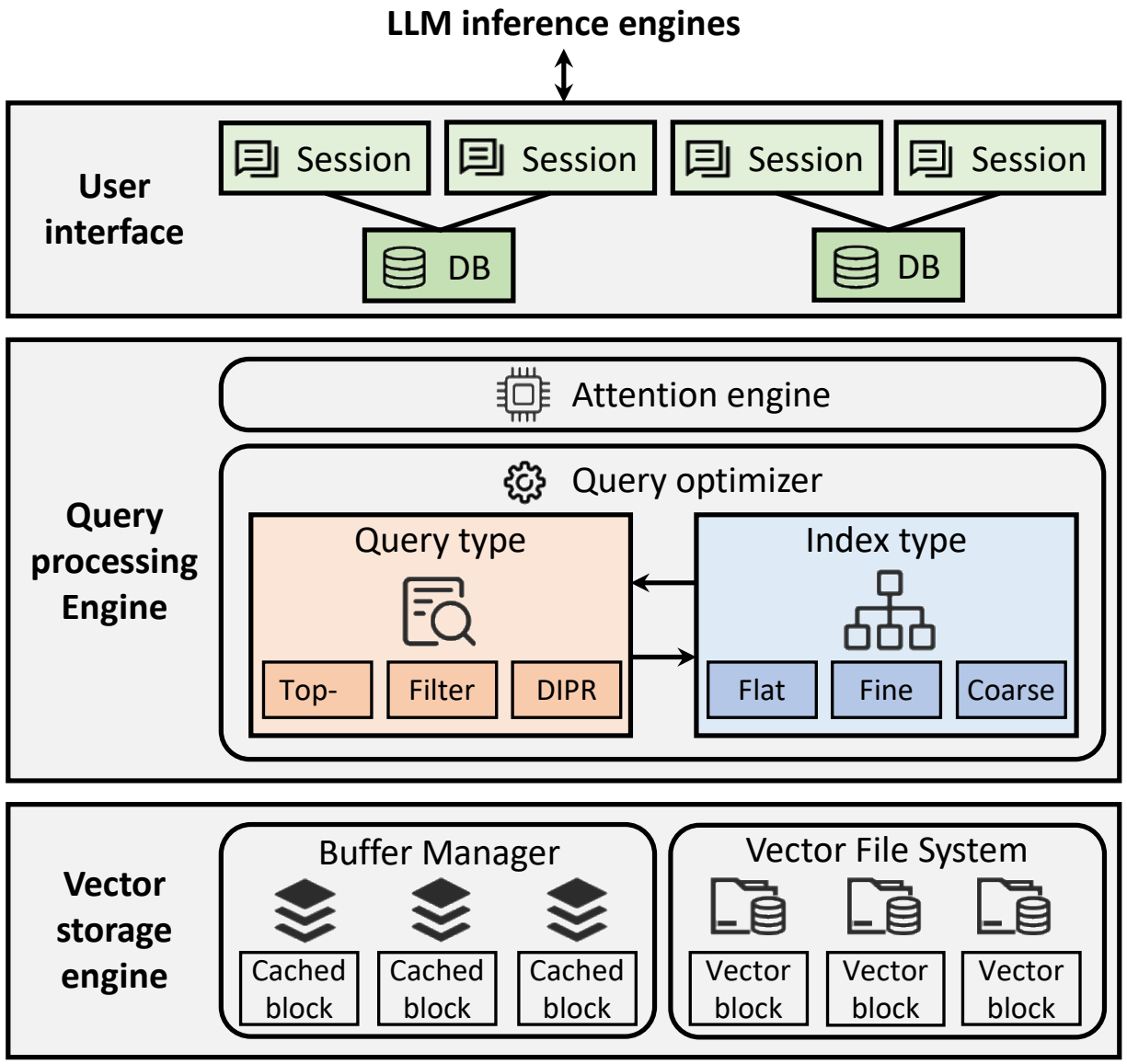
- **Coarse-grained index:** *InfLLM, Quest, Arkvale*
- **Fine-grained index:** *RetrievalAttention, MagicPiG*
- **Flat index:** *brute-force scan*

- AlayaDB provides a framework to integrate and optimize sparse attention algorithms

Rule-based query optimizer

Index type	Supported query type	GPU memory comsumption	Latency small k	Latency large k
Coarse	Top-k, Filter	Large	Low	Low
Fine	Top-k, Filter, DIPR	Small	Low	High
Flat	Top-k, Filter, DIPR	Small	Medium	Medium





- Interface & API
- Vector search query DIPR
- Query optimizer
- **End-to-end optimizations**

End-to-end optimizations

- Index construction acceleration
- Late materialization for index updating
- Data-centric attention engine
- Vector file system organized in graph
- Buffer manager
- …

Experiments

- NVIDIA L20 (48GB)
- 2 x XEON GOLD 6542Y CPU (48 x 2 threads, 512GB)
- SLO: *Time-Per-Output-Token* (*TPOT*) = 240ms

Can AlayaDB achieve ***low latency***, ***high quality***, and ***low resource consumption*** for long context LLM inference?

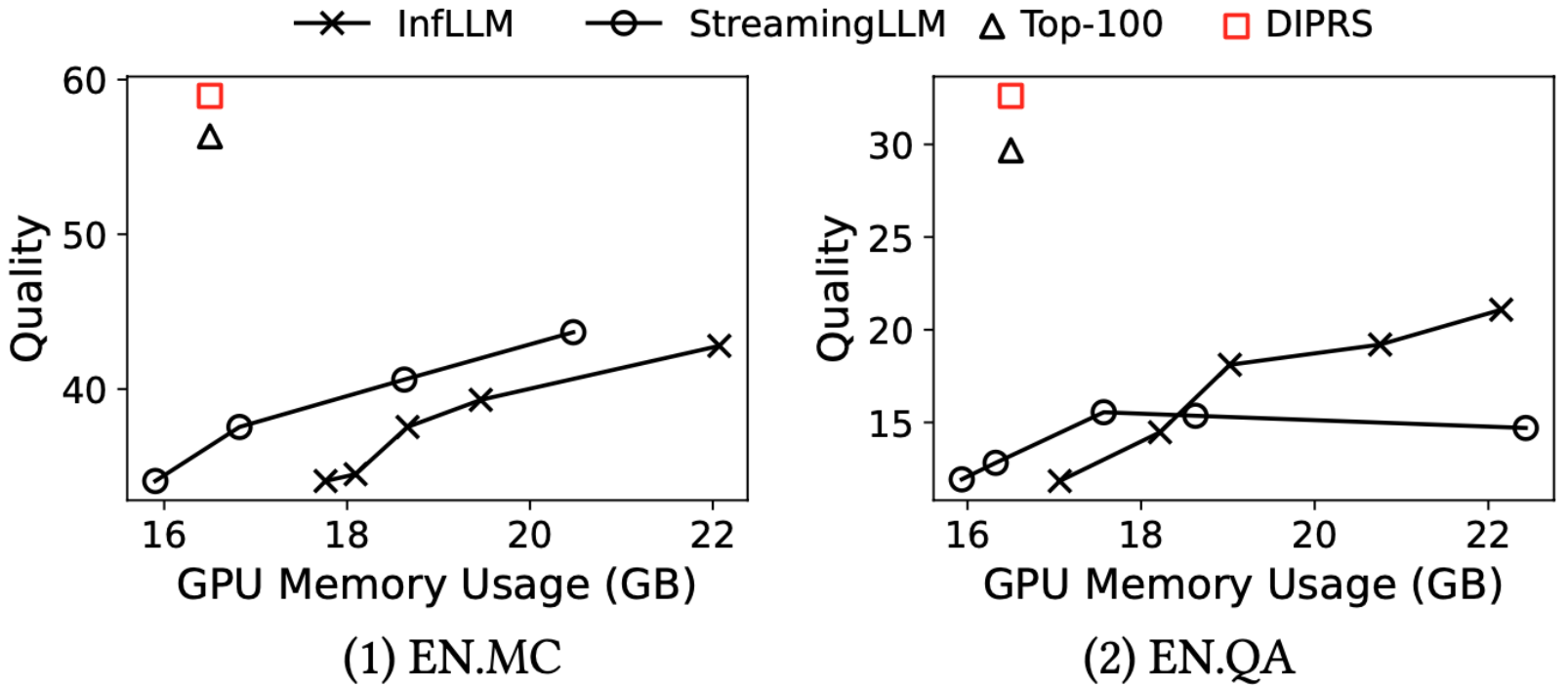
Generation quality

Table 5: Generation quality of different sparse attention algorithms in ∞-Bench. Each method used the number of [*initial*+*last*]+*retrieved* tokens for attention computation.

Methods	Setting	SLO	Retr.KV	Retr.P	Retr.N	Code.D	En.MC	En.QA	En.Sum	Math.F	Avg.
Full Attention	—	✗	15.8	100.0	100.0	27.4	55.9	31.0	15.1	19.1	45.6
InfLLM	[128+4K]+4K tokens	✓	**25.0**	100.0	100.0	28.2	39.7	18.7	15.3	23.4	43.8
StreamingLLM	[128]+8K tokens	✓	3.8	8.5	8.5	27.7	41.5	14.5	14.3	16.3	16.9
Top100	[128+512]+100 tokens	✓	6.6	100.0	100.0	30.0	56.3	29.7	15.2	24.6	45.3
Top2000	[128+512]+2K tokens	✗	14.6	100.0	100.0	29.7	58.1	31.2	16.0	24.3	46.7
DIPRS	[128+512] tokens, $\beta = 50$	✓	14.0	**100.0**	**100.0**	**30.7**	**58.1**	**32.1**	**16.4**	**24.9**	**47.0**

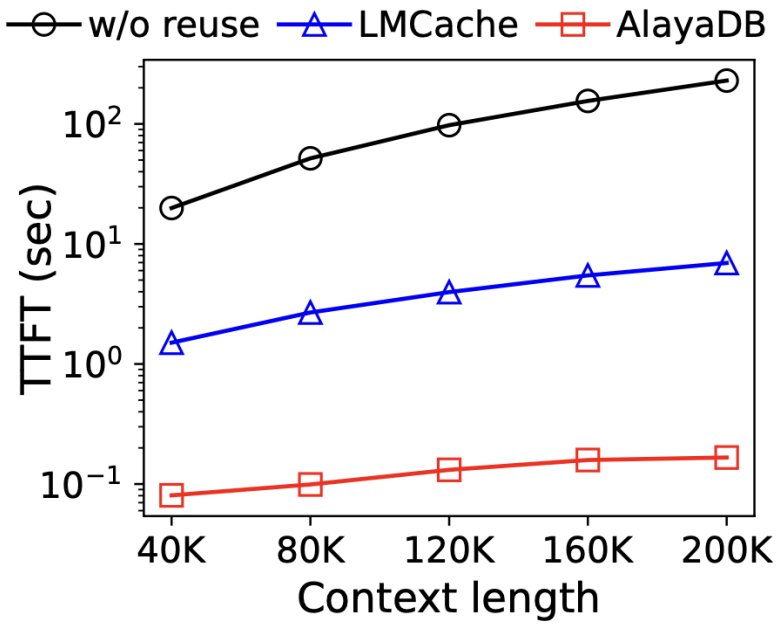
- Highest quality under the SLO

Resource consumption

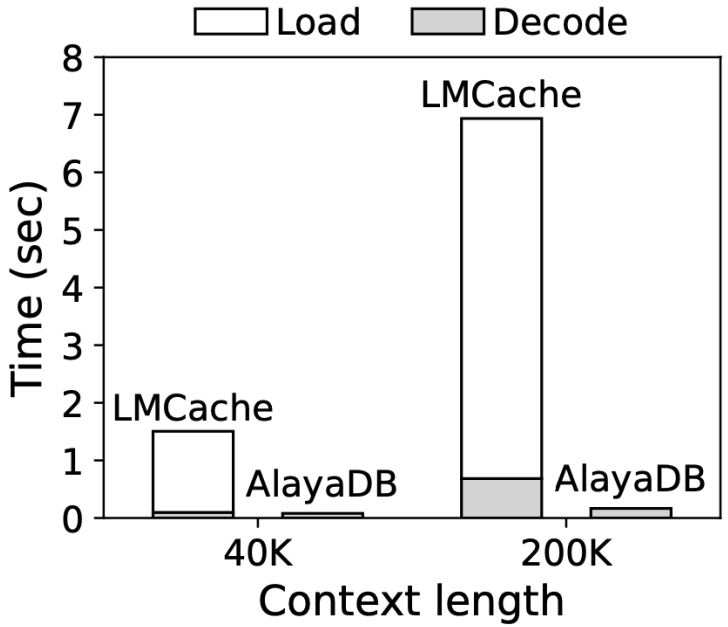


- Smallest GPU memory consumption under the SLO

Time-To-First-Token (TTFT)



(a) TTFT



(b) Latency breakdown

- Faster context reuse than KV cache disaggregation

Takeaways

- Using vector database for LLM inference is powerful.

- If need to disaggregate KV cache, should also disaggregate attention.
- Attention is a near-data computation (a new DB operator)

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

AlayaDB
太昊信息

Thanks

research@alayadb.ai