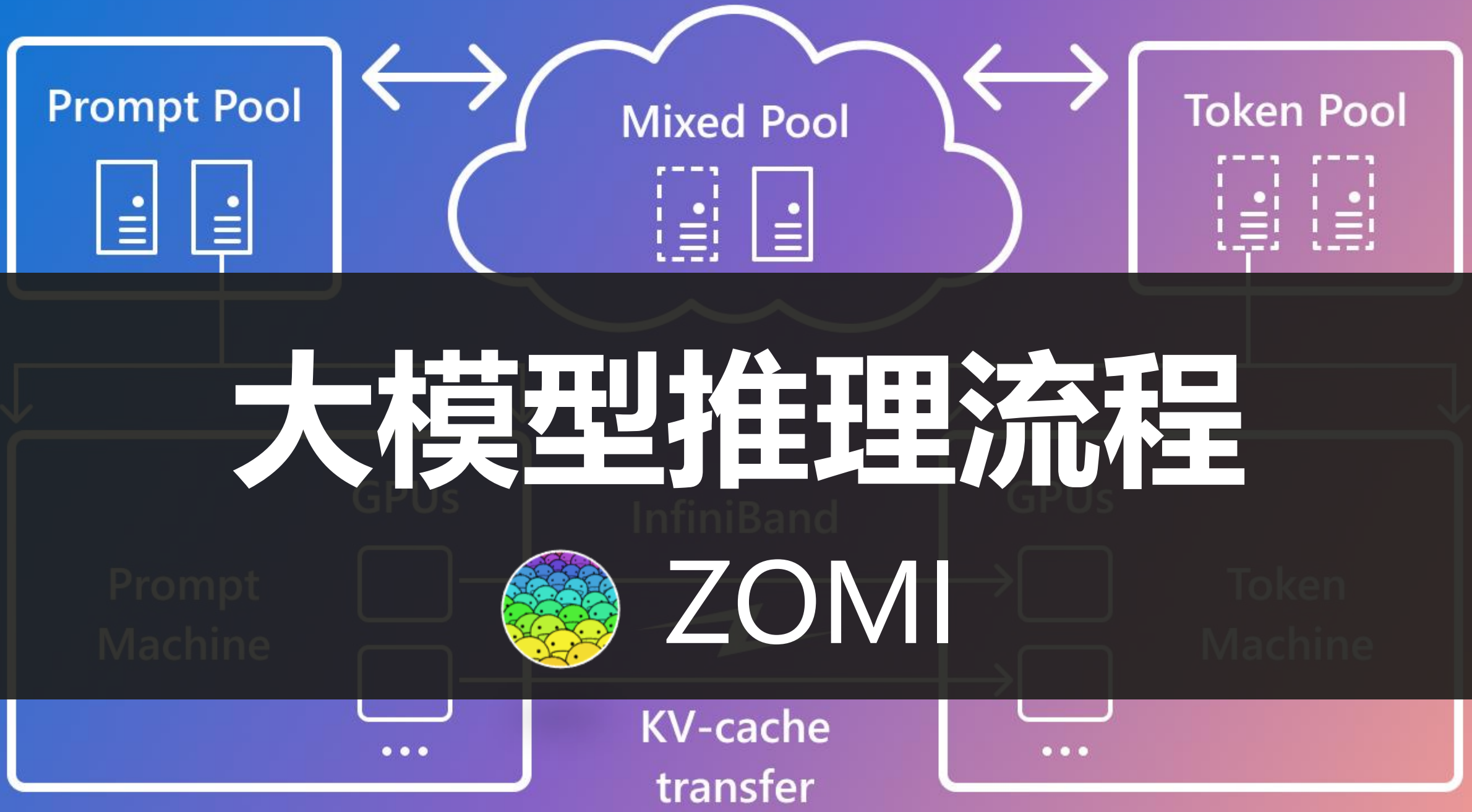
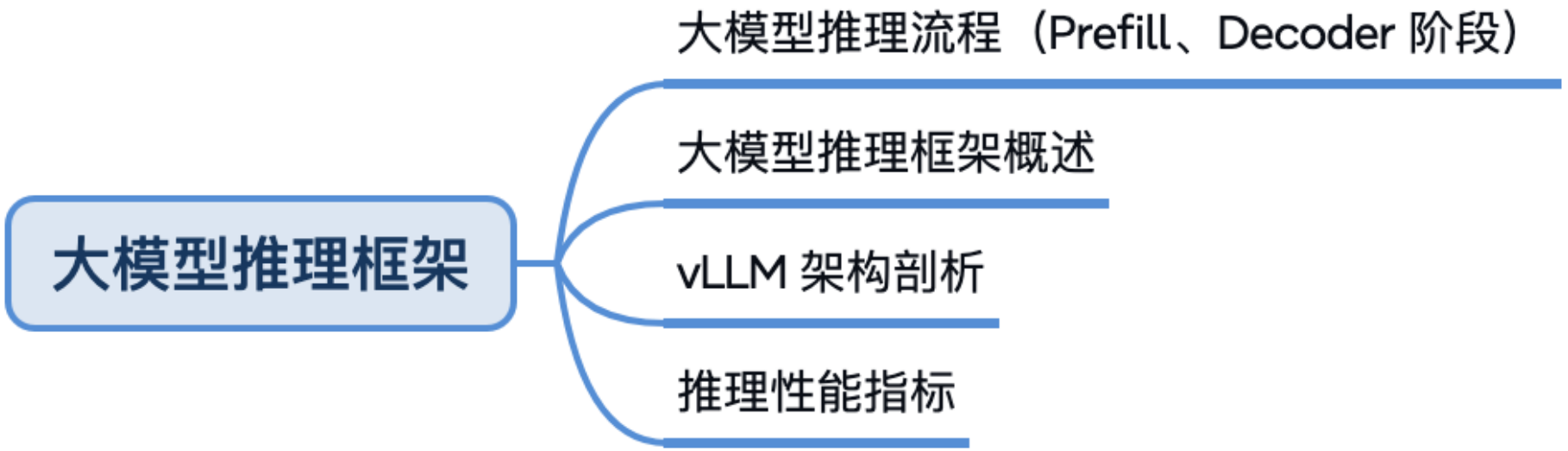


大模型推理流程

ZOMI

大模型推理


大模型推理框架
大模型推理流程（Prefill、Decoder 阶段）
大模型推理框架概述
vLLM 架构剖析
推理性能指标

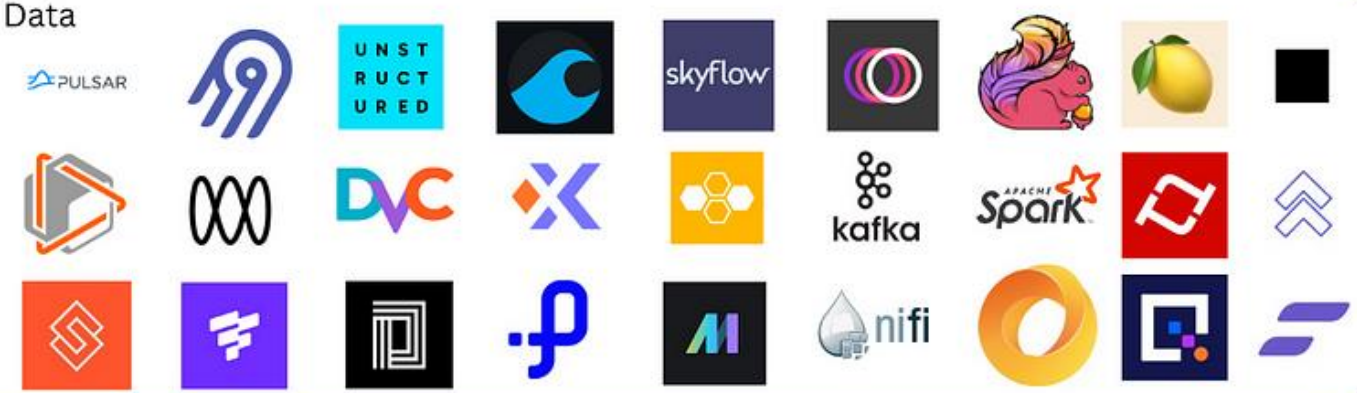
目录

01

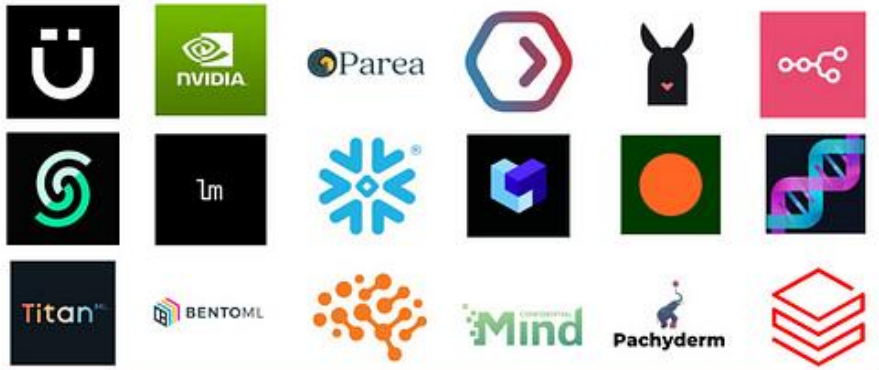
大模型推理需求增长


闭源
(语言模型)
文心一言
通义千问
腾讯混元
商汤日日新 sensenova
vivo BlueLM
360智脑
AISPEECH
字节豆包大模型
智谱·AI
MINIMAX
天工
DeepSeek
深言科技 DeepLang AI
澜舟科技 langboat
玉言
盘古大模型
百川智能 BAICHUAN AI
云从科技 CLOUDWALK 云从科技从容大模型
Moonshot AI
出门问问
西湖心辰
4Paradigm SageGPT
阶跃星辰
oppo AndesGPT
讯飞星火
MiLM-6B
云知声
山海
intellifusion
言犀
闭源
(多模态)
可灵AI
星火绘镜
PixVerse
文心一格
通义万相
Vimi
艺映AI
硅基智能
万兴播爆
HiDream.ai
即梦AI
meitu
Vidu
Vega AI
奇妙元
X-Avatar
有言
开源
Qwen2
deepseek coder
GLM-4
MiniCPM
Yi
Baichuan2
InternLM
OPEN SORA
混元-DiT
RWKV-LM
XVERSE 元象
部分领域
医疗
MedGPT
岐黄问道
华佗GPT
左医GPT
京医千询
.....
汽车
理想 MindGPT
DriveGPT
星睿AI
广汽AI大模型
NomiGPT
.....
教育
MathGPT
作业帮
子曰
EduChat
.....
金融
蚂蚁金融大模型
度小满
轩辕大模型
妙想大模型
.....
工业
AInno-15B
COSMO-GPT
SmartMore SMore LrMo
.....
文化/零售/交通
阅文集团
晓模型 XPT
乐言科技
妙笔大模型
PCI 佳都科技
.....

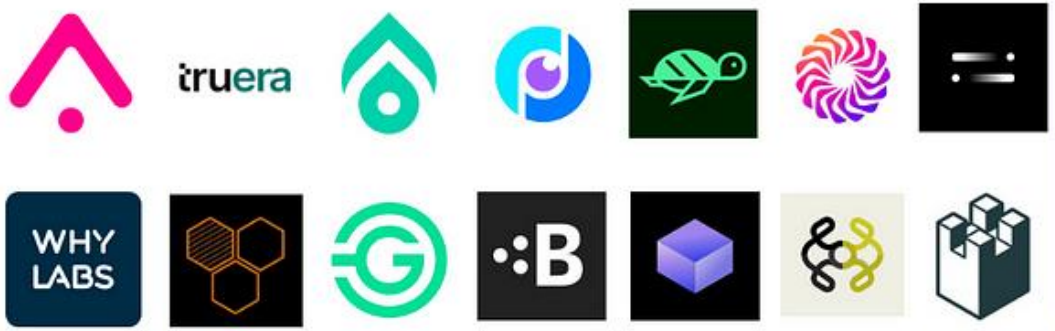
Data



Infrastructure



Quality Tuning



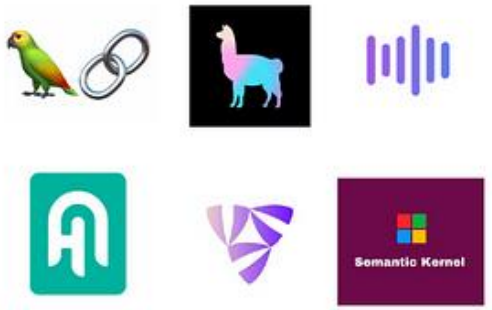
LLM Stack
Yujian Tang
May 2024

Vector Databases

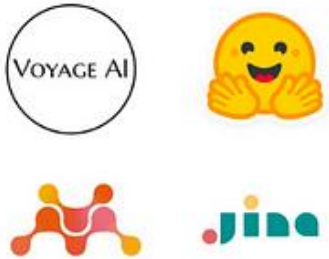
LLMs

LLM Providers

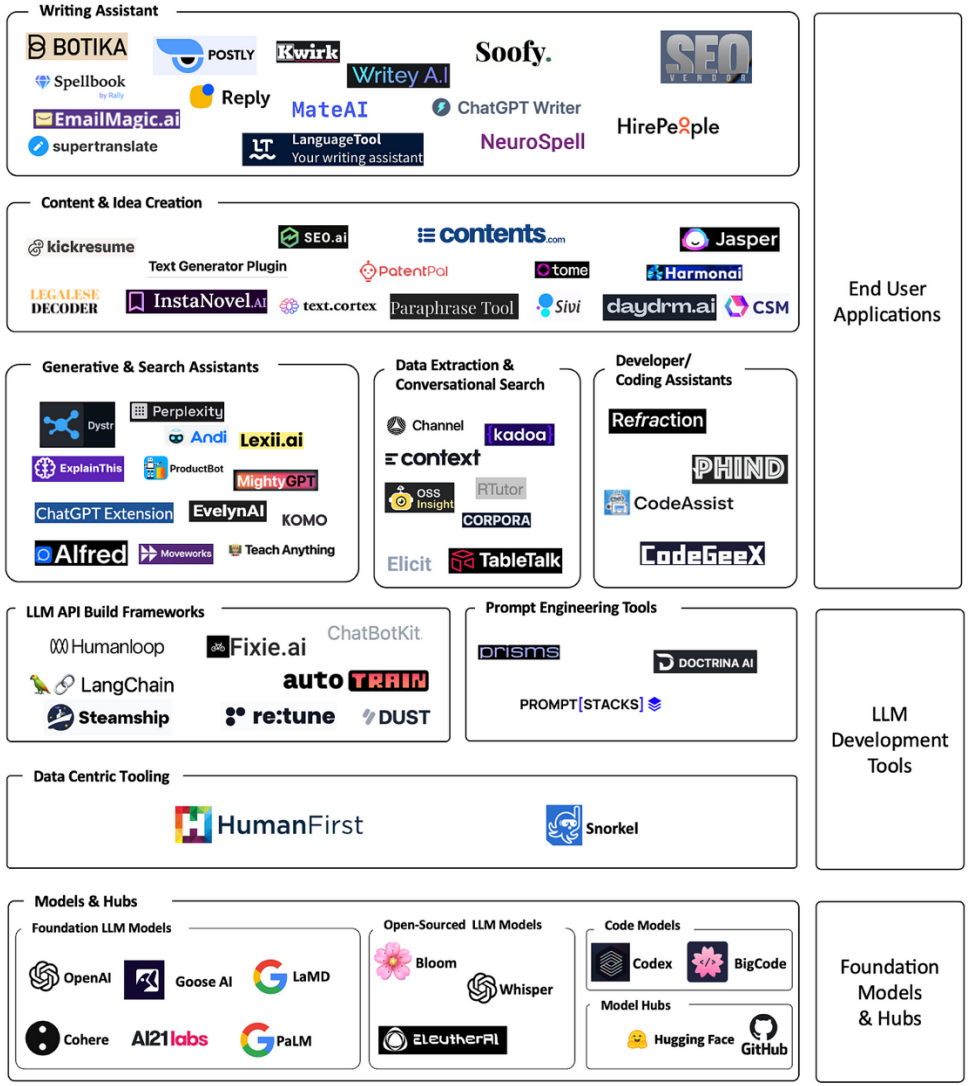
Orchestration

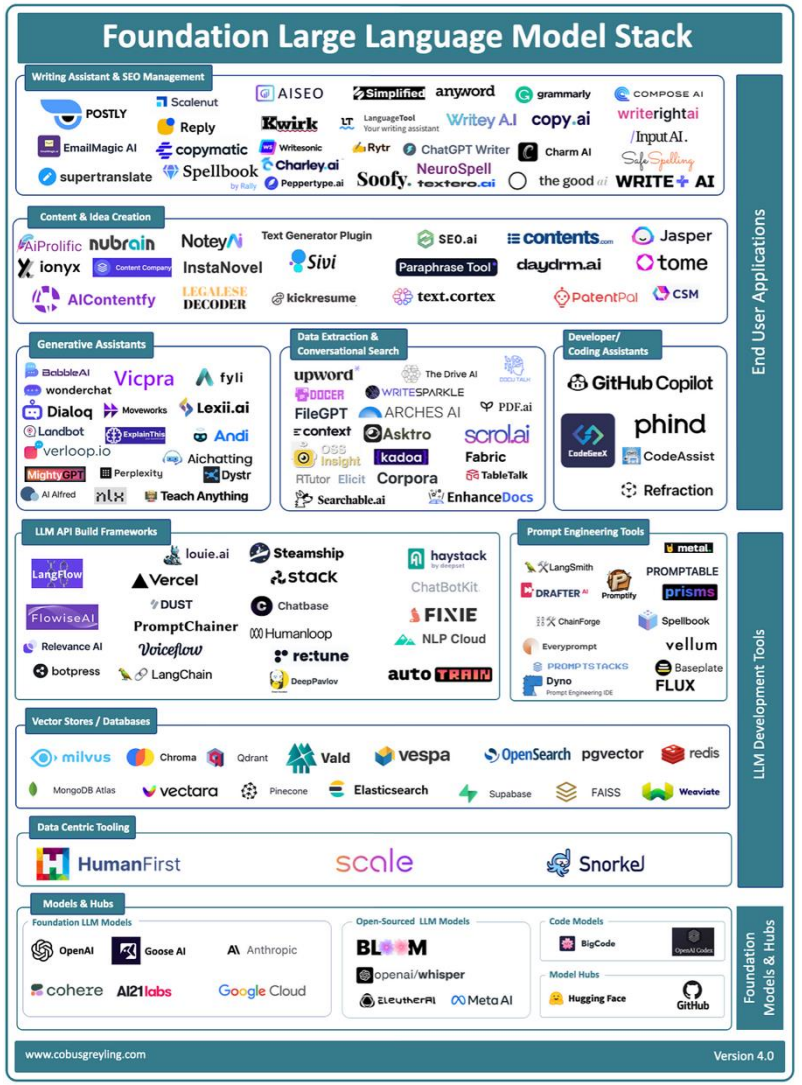


Embedding Models



Foundation Large Language Model Stack





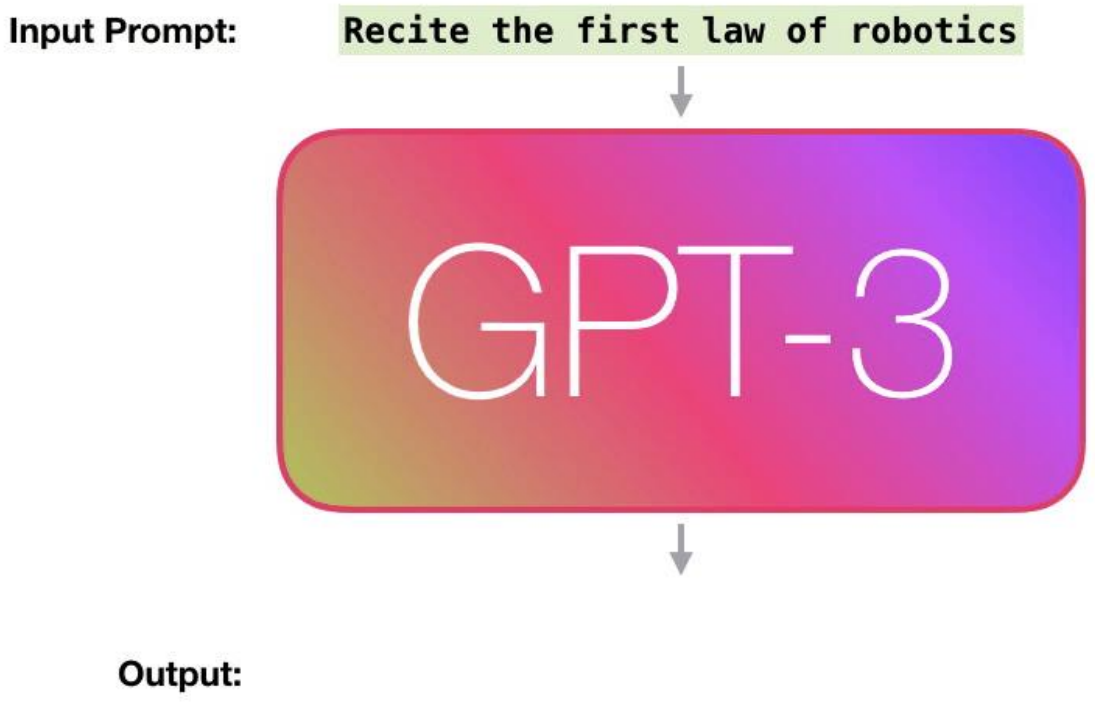
大模型推理需求

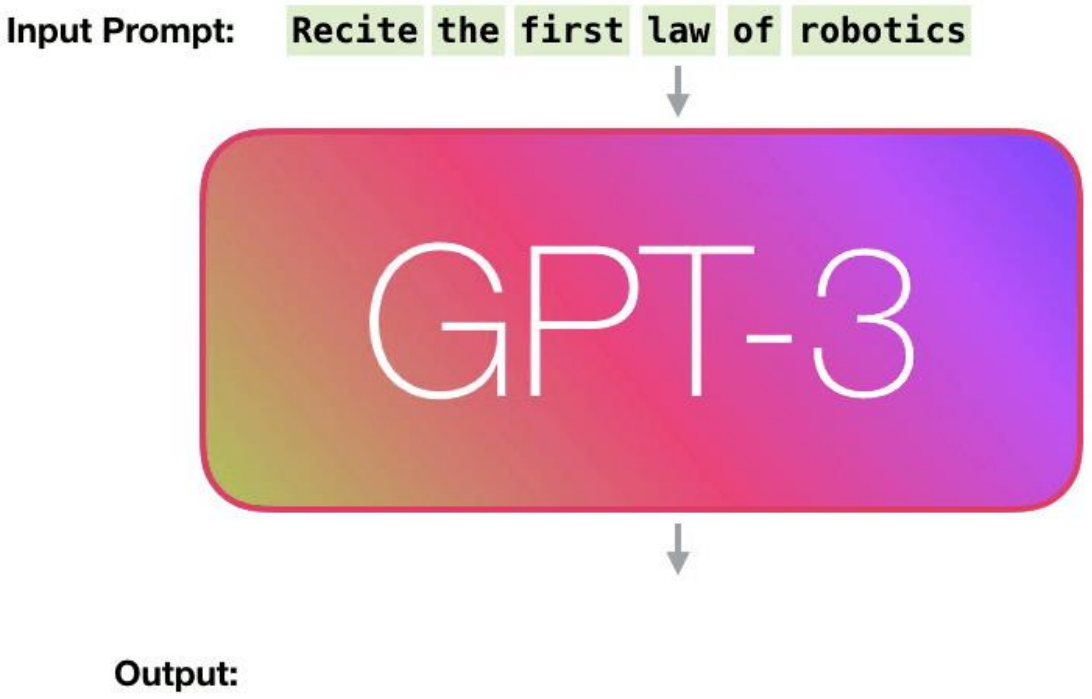
- 新的大模型不断发布
- 模型参数量不断加大
- 生成式 AI 应用爆发式增长

02

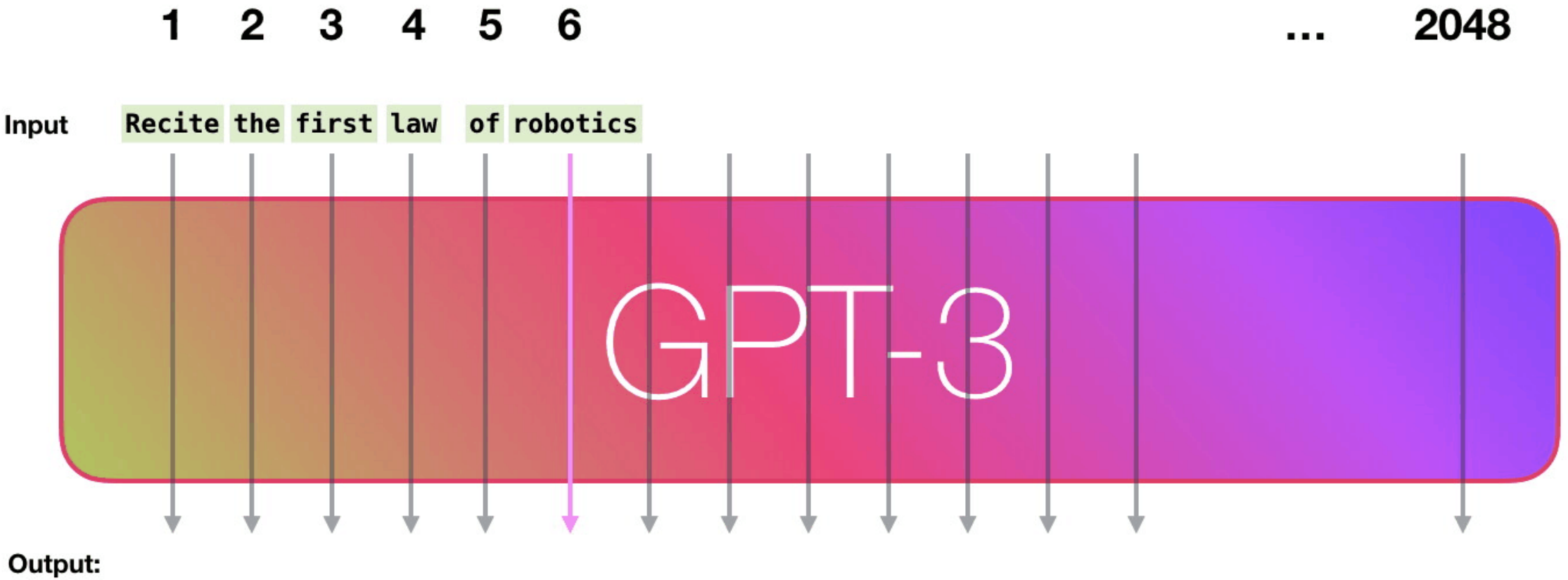
LLM 推理过程

LLM 推理过程

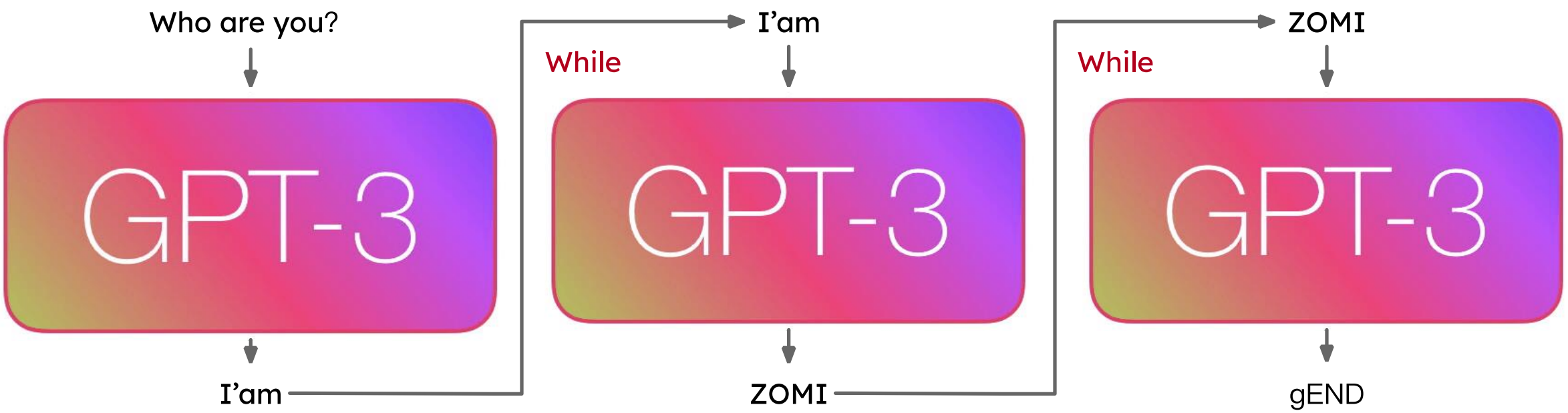

Input Prompt:
Recite the first law of robotics
GPT-3
Output:

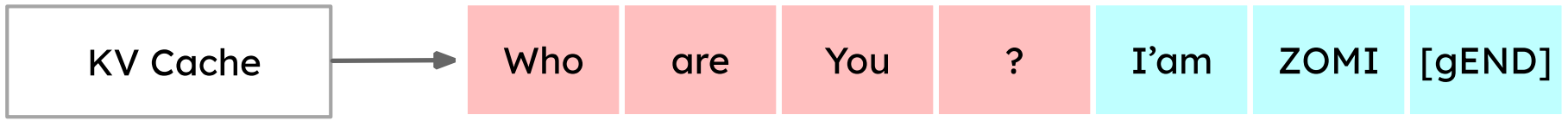

Input Prompt:
Recite the first law of robotics
GPT-3
Output:

LLM 推理过程

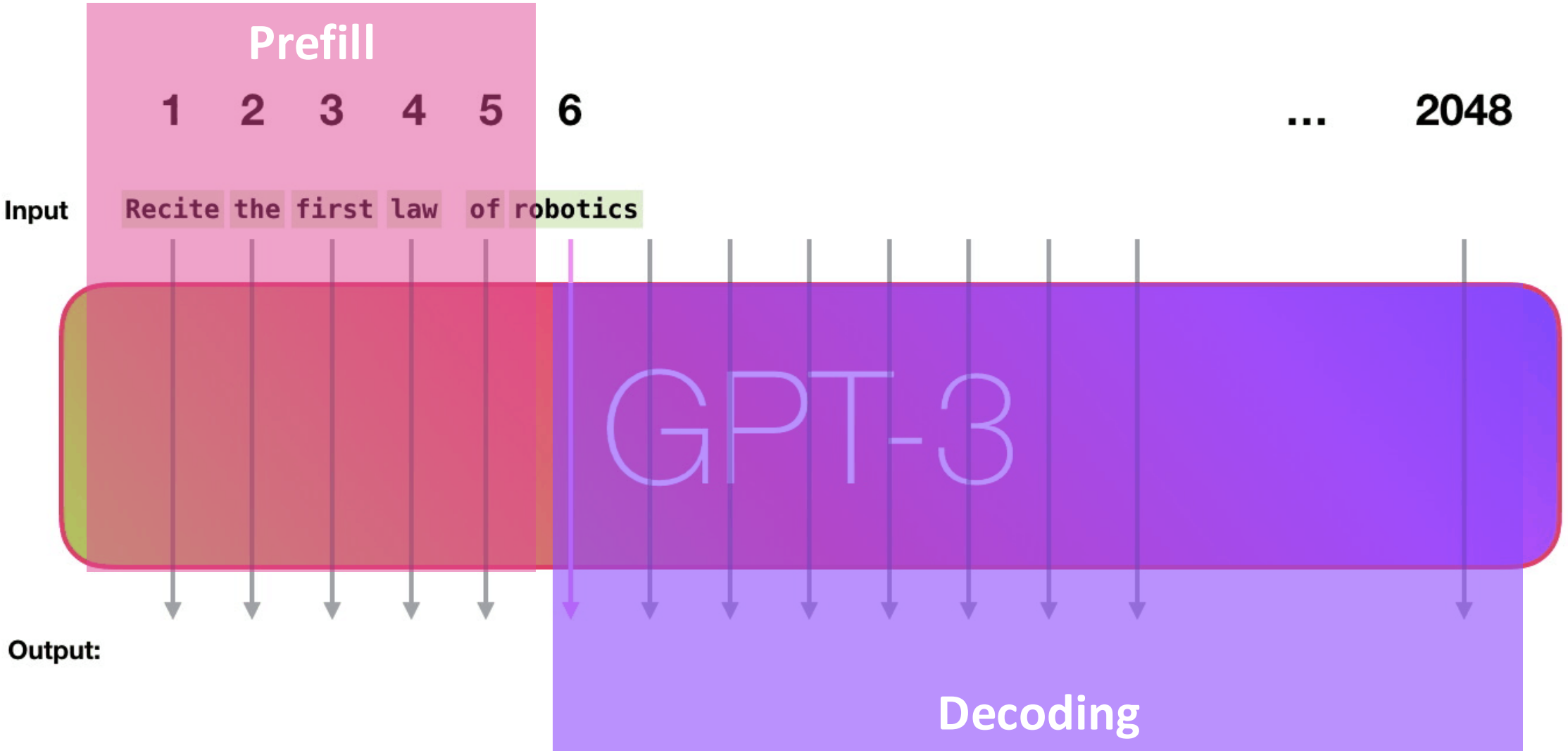

1 2 3 4 5 6 ... 2048
Input
Recite the first law of robotics
GPT-3
Output:

大模型推理自回归解码


Who are you?
GPT-3
I'am
While
I'am
GPT-3
ZOMI
While
ZOMI
GPT-3
gEND


KV Cache
Who
are
You
?
I'am
ZOMI
[gEND]

LLM 推理过程


Prefill
1 2 3 4 5 6 ... 2048
Input
Recite the first law of robotics
GPT-3
Output:
Decoding

LLM 推理过程

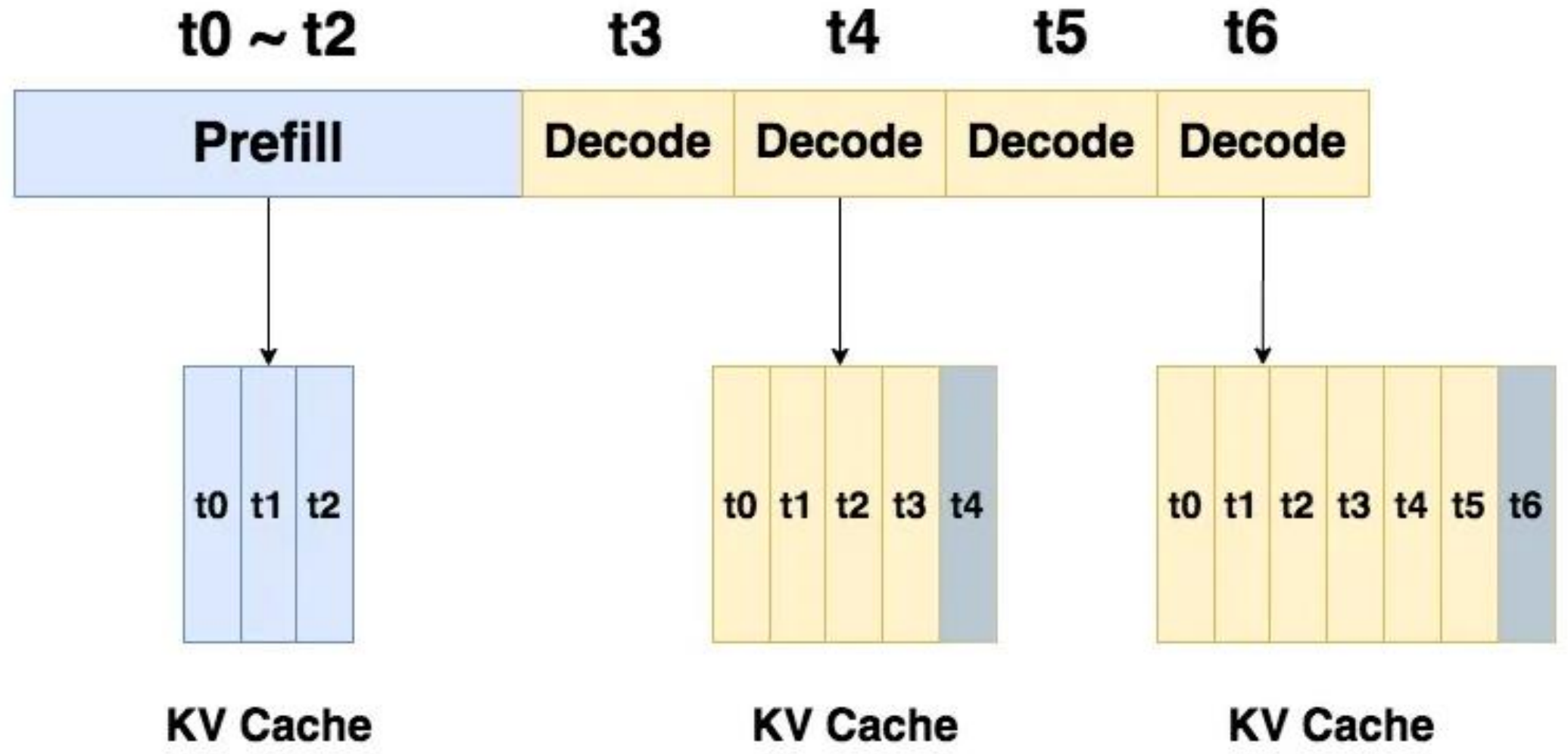
- **Prefill：**
  - 根据输入 Tokens 生成第一个输出 Token（A），通过一次 Forward 就可以完成
  - 在 Forward 中，输入 Tokens 间可以并行执行，因此执行效率很高

- **Decoding：**
  - 从生成第一个 Token 后，采用自回归一次生成一个 Token，直到生成 Stop Token 结束
  - 设输出共 N x Token，Decoding 阶段需要执行 N-1 次 Forward，只能串行执行，效率很低
  - 在生成过程中，需要关注 Token 越来越多，计算量也会适当增大

03

KV Cache

KV Cache


t0 ~ t2
t3
t4
t5
t6
Prefill
Decode
Decode
Decode
Decode
t0 t1 t2
t0 t1 t2 t3 t4
t0 t1 t2 t3 t4 t5 t6
KV Cache
KV Cache
KV Cache

KV Cache

- 把每个 token 在过 Transformer 时乘以 W_k, W_v 这俩参数矩阵的结果缓存下来，训练的时候不需要保存
- 推理解码生成时采用自回归 auto-regressive 方式，即每次生成一个 token，都要依赖之前 token 的结果
- 如果每生成一个 token 时候乘以 W_k, W_v 这俩参数矩阵要对所有 token 都算一遍，代价非常大，所以缓存起来就叫 KV Cache

Example

- 假如 prompt = The largest city of China is，输入是 6 个 tokens，返回是 Shang Hai 这两个 tokens。整个生成过程如下：

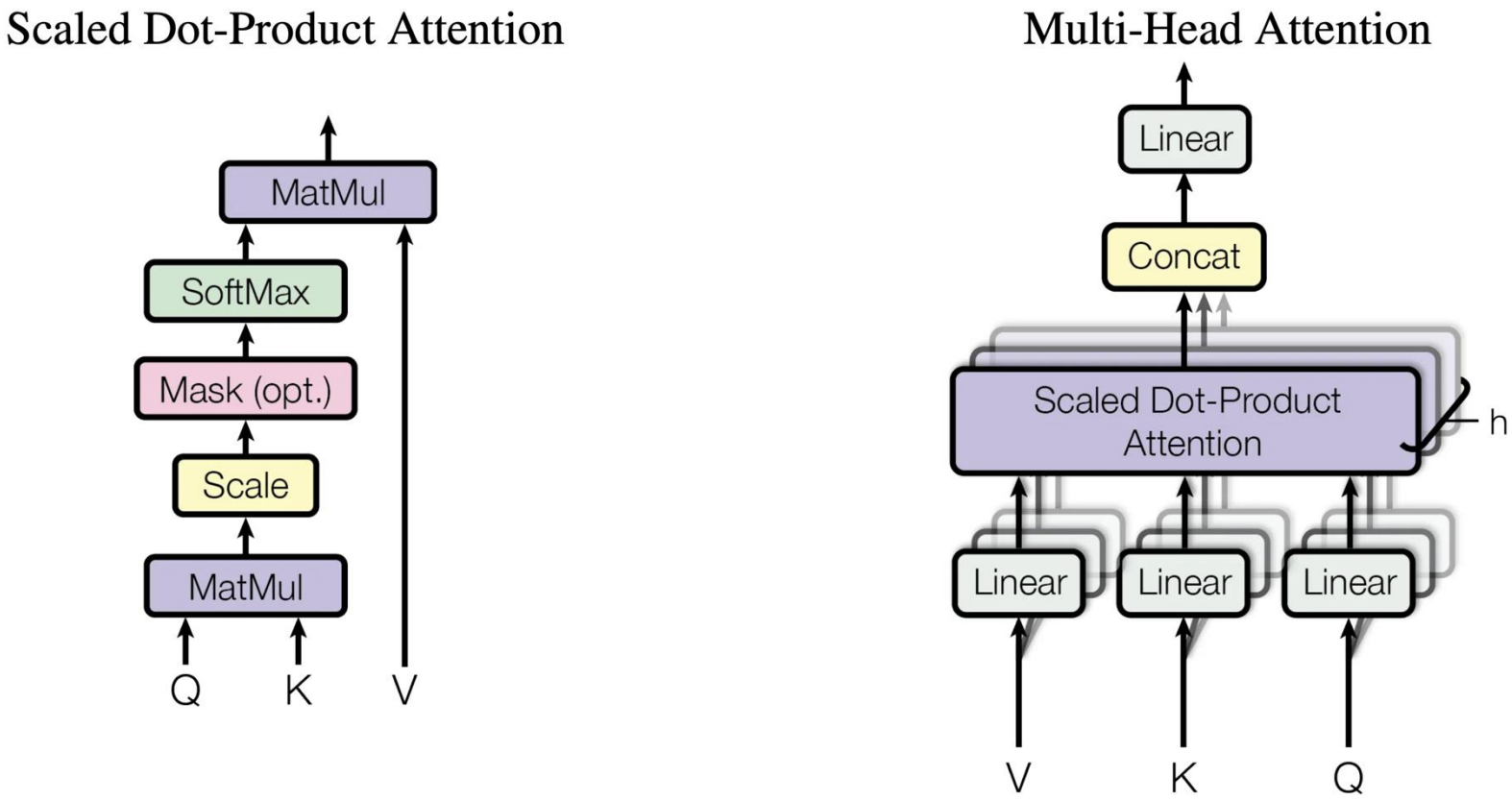
  1. 当生成 Shang 之前，KV Cache 把输入6个 tokens 都乘以 W_k，W_v 这两参数矩阵，也就是缓存了6个 K & V。这时候过 self-attention + 采样方案（greedy、beam、top-k、top-p 等），得到 Shang 这个token
  2. 当生成 Shang 之后，再次过 transformer 的 token 只有 Shang 这一个 token，而不是整个 The largest city of China is Shang 句子，这时再把 Shang 这一个token对应的 KV Cache 和之前 6 个 tokens 对应的 KV Cache 拼起来，成为了7个 KV Cache。 Shang 这一个 token 和前面 6 个 tokens 就可以最终生成“Hai”这个 token

思考

1. 为什么有 KV Cache，没有Q Cache?

Multi-Head Attention

- 在 LLM 推理中最关键的就是下图中的 Multi-Head Attention，其主要的计算集中在左图中灰色的 Linear（矩阵乘）和 Scaled Dot-Product Attention 中的 MatMul 矩阵乘法：



Multi-Head Attention

- 先来展开 scaled dot production 的计算：

$$Attention = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- 其中，每次新多一个 Q 中的 token 时，用新增加的 token 和所有 tokens 的 K、V 去乘，Q cache 就成多余的。
- 再拿刚才例子强调一下，当生成 Shang 之后，再次过 transformer 的 token 只有 Shang 这一个 token，而不是整个 The largest city of China is Shang 句子

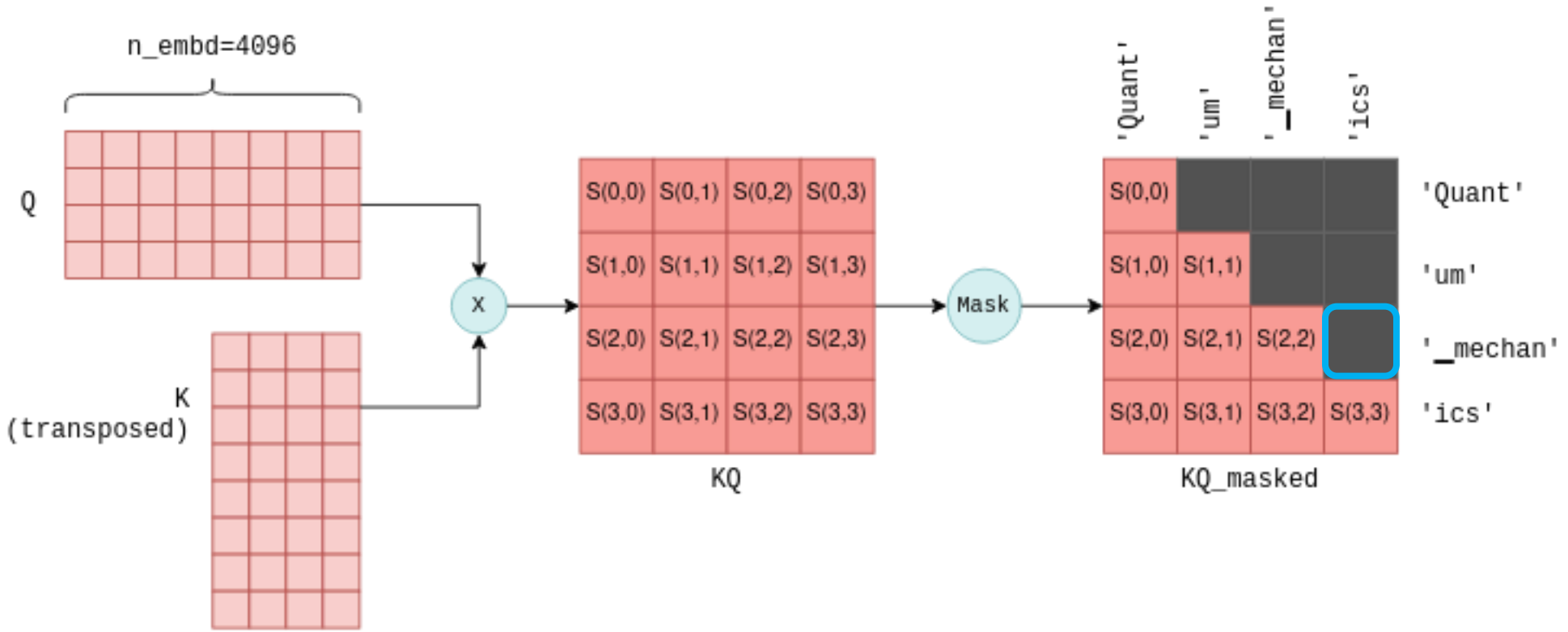
Question

1. 生成 Shang Token 时，感觉是 The largest city of China is 这 6 个 tokens 的 query 都用了，但是生成 Hai 这个 token 时，只依赖 Shang 这个 token 的 query 嘛?

- 这个问题其实是没有的，每个 token 的生成都只依赖前一个 Q 和之前所有的 K & V

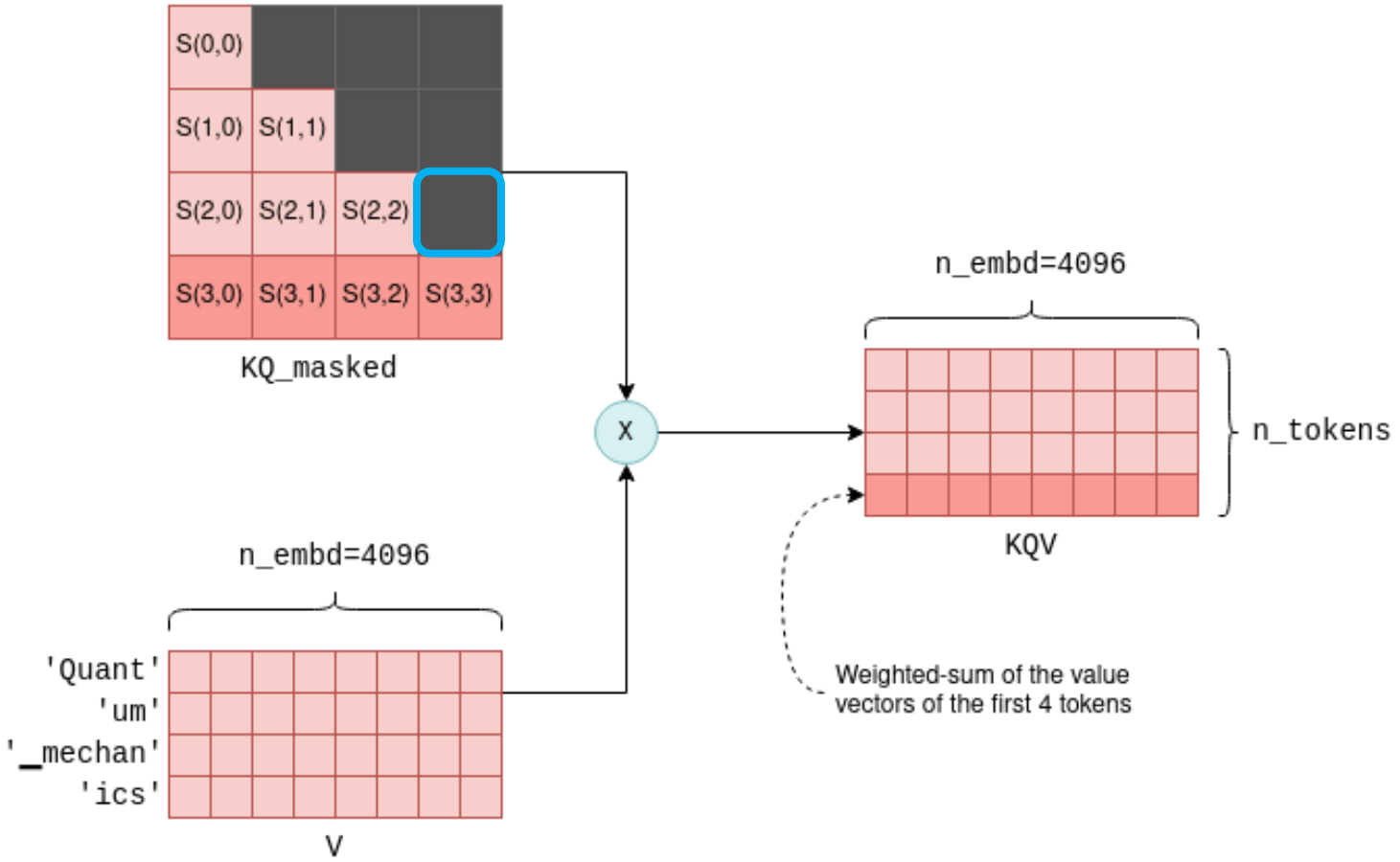
LLM Training

- 训练的时候 Quant, um, _mechan 下一个 token 在矩阵乘法时对应的是蓝框，被 mask 掉了



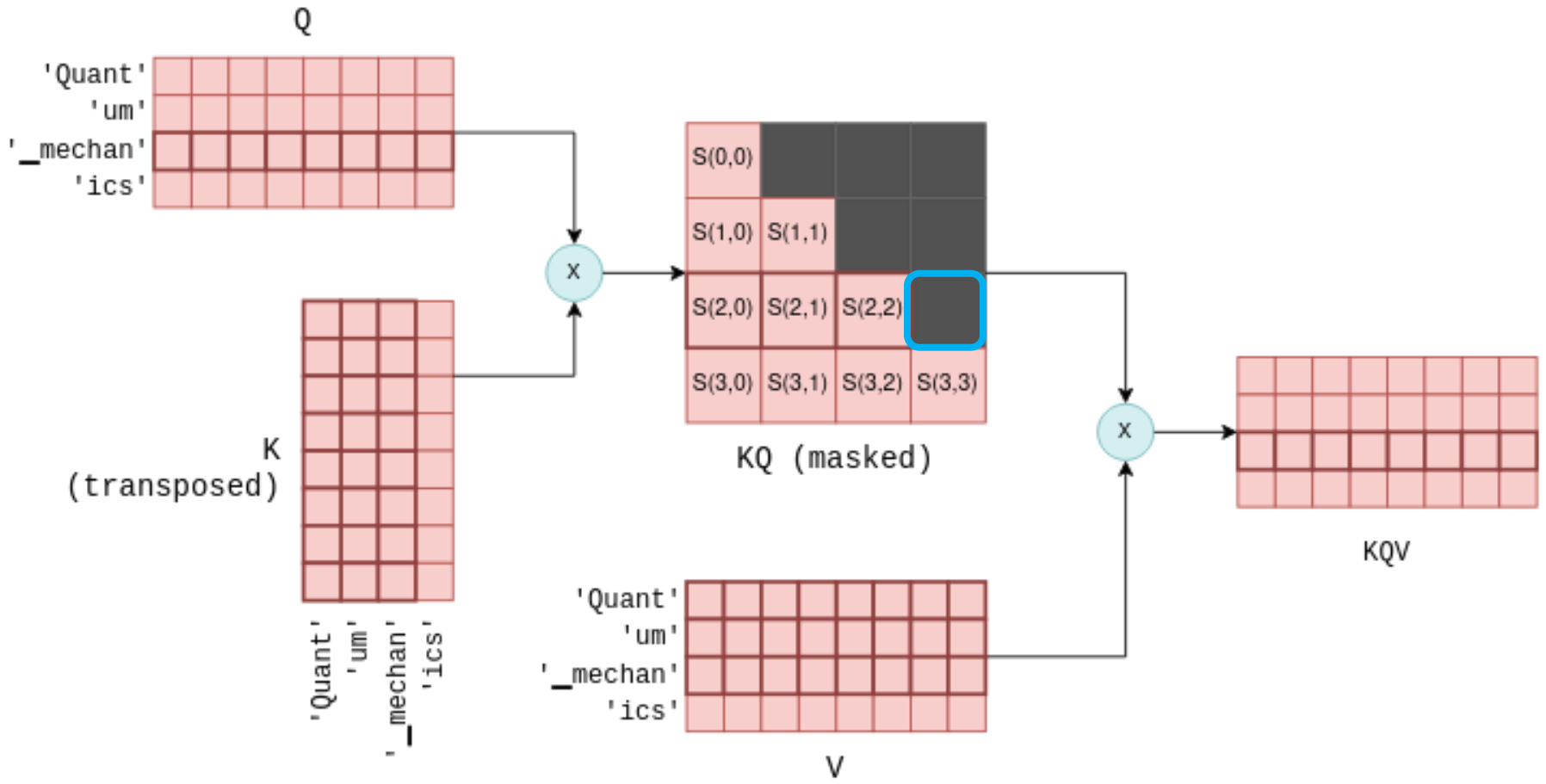
LLM Training

- 训练的时候 Quant, um, _mechan 下一个 token 在矩阵乘法时对应的是蓝框，被 mask 掉了



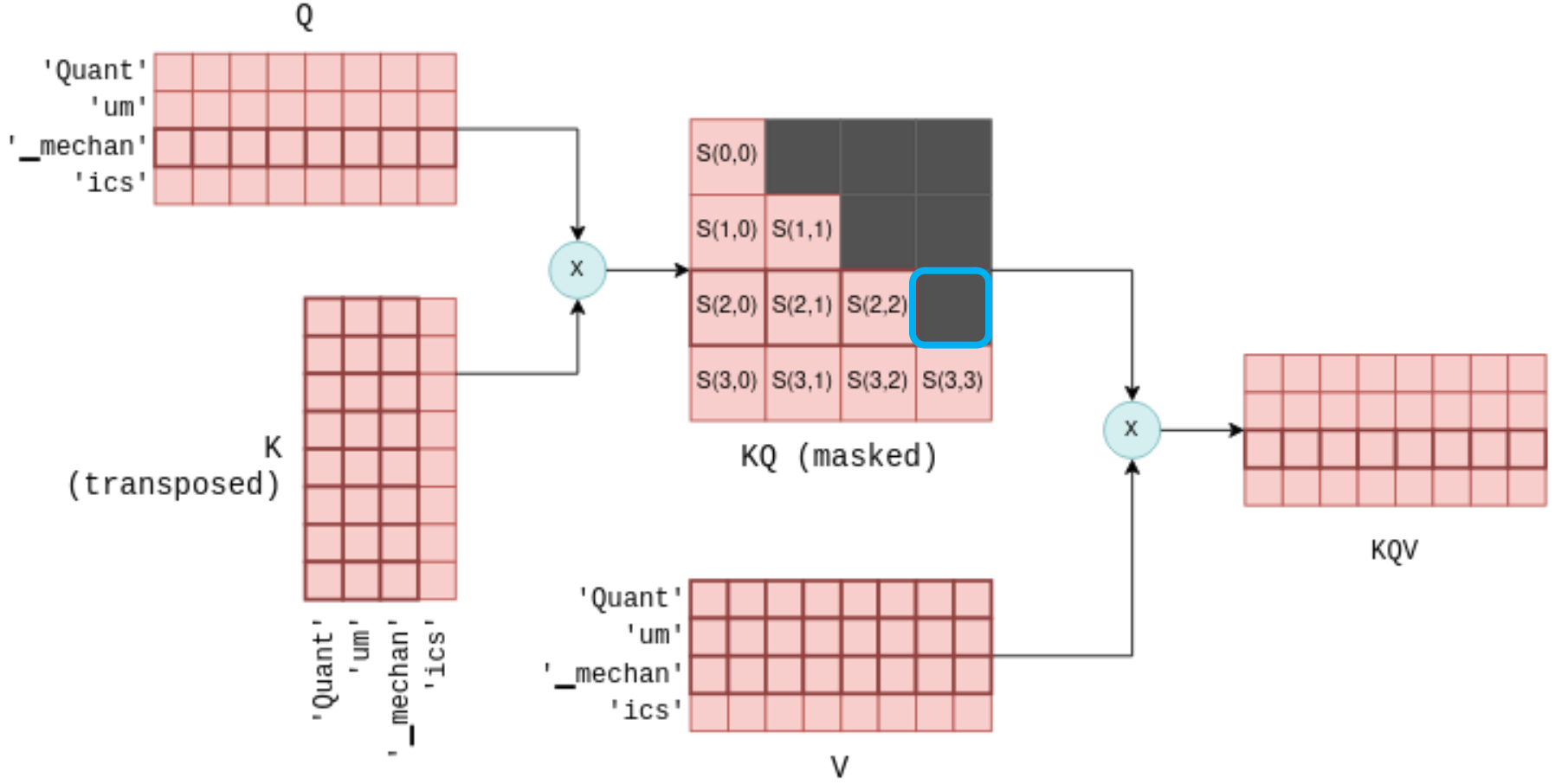
KV Cache

- 推理时在给定 Quant, um, _mechan，已有序列长 3，矩阵乘法由红框决定，和未来没读到的蓝框 token 没有任何关系。



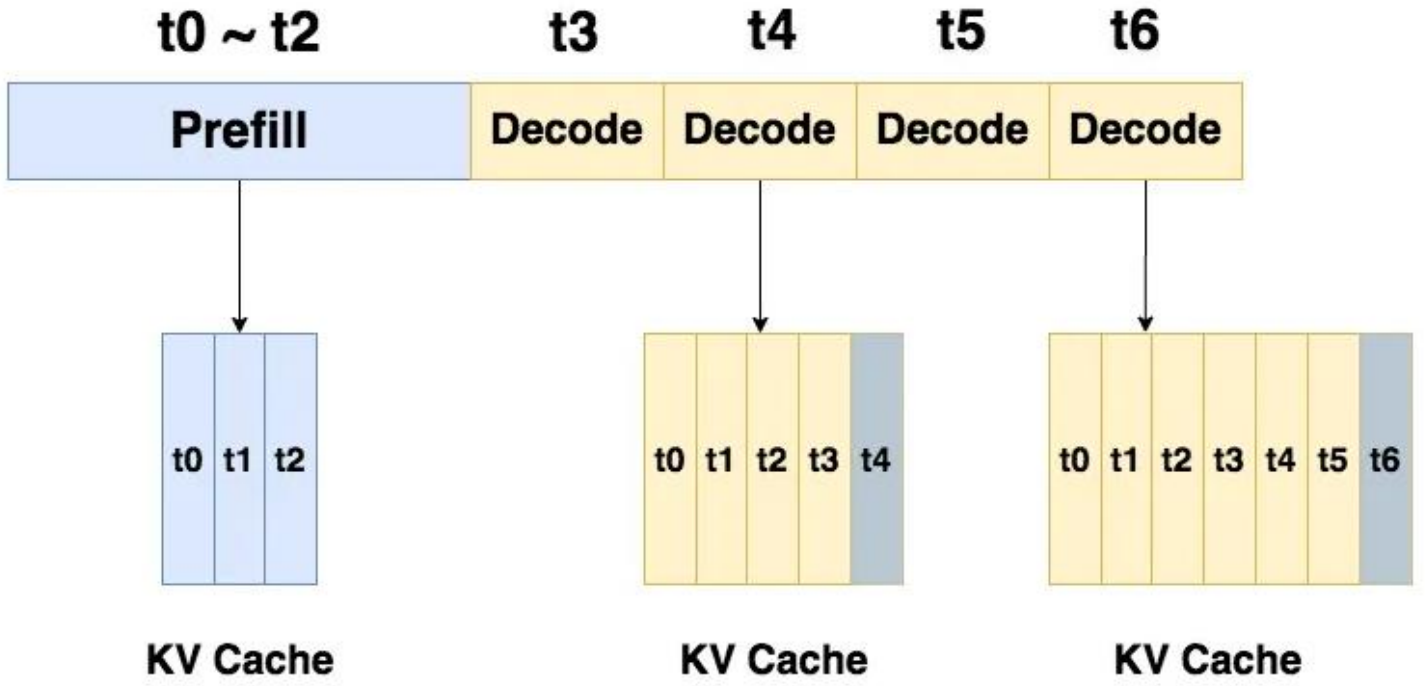
KV Cache

- 同时 _mechan 下一 token 只和 _mechan 的 Q 有关，和 Quant, um 的 Q 无关
- 所以每个 token 生成都依赖前一个 Q 和之前所有 K & V



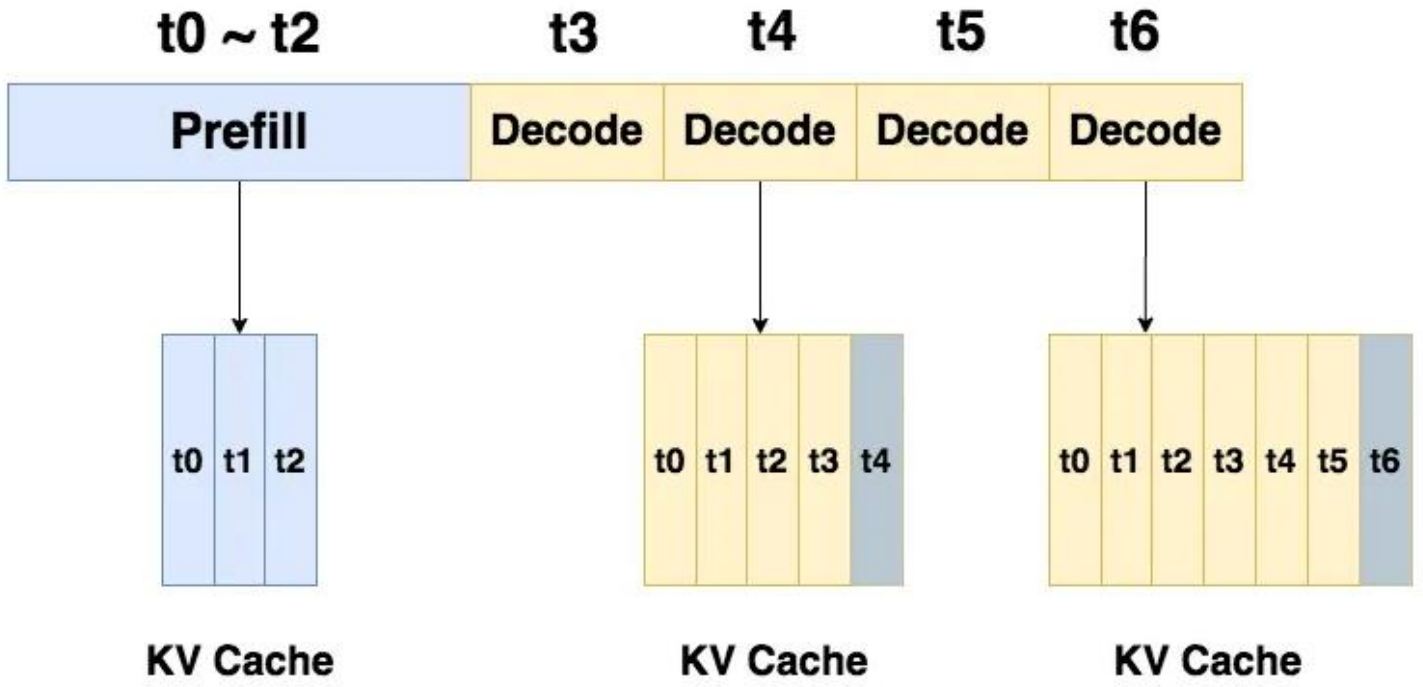
KV Cache：Prefill

- 把整段 prompt 输入模型执行 forward 计算。采用 KV cache 技术， Prefill 阶段中会把 prompt 过后得到的保存在 cache k 和 cache v 中。这样后面 token 计算 attention 时，就不需要对前面 token 重复计算，可以节省推理时间



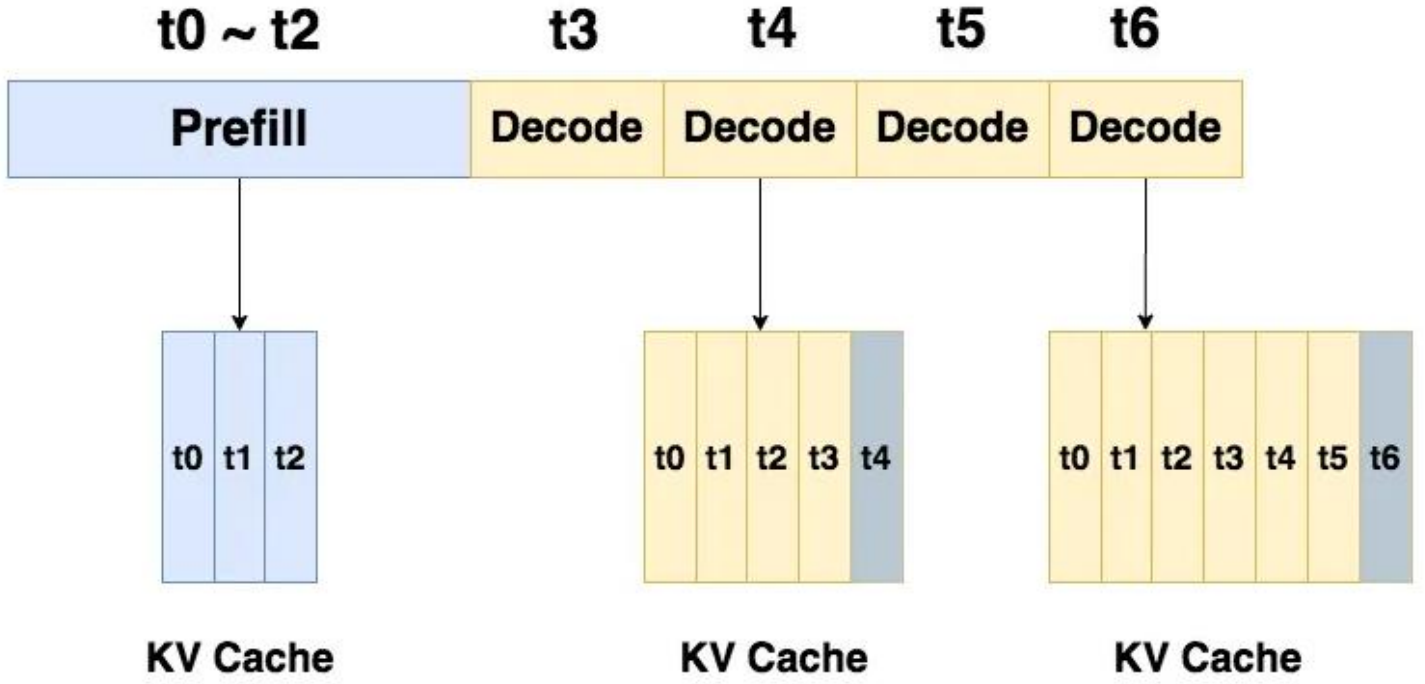
KV Cache：Prefill

- 假设 prompt 中含有 3 个 token，prefill 阶段结束后，这三个 token 相关的 KV 值都被装进了 cach e



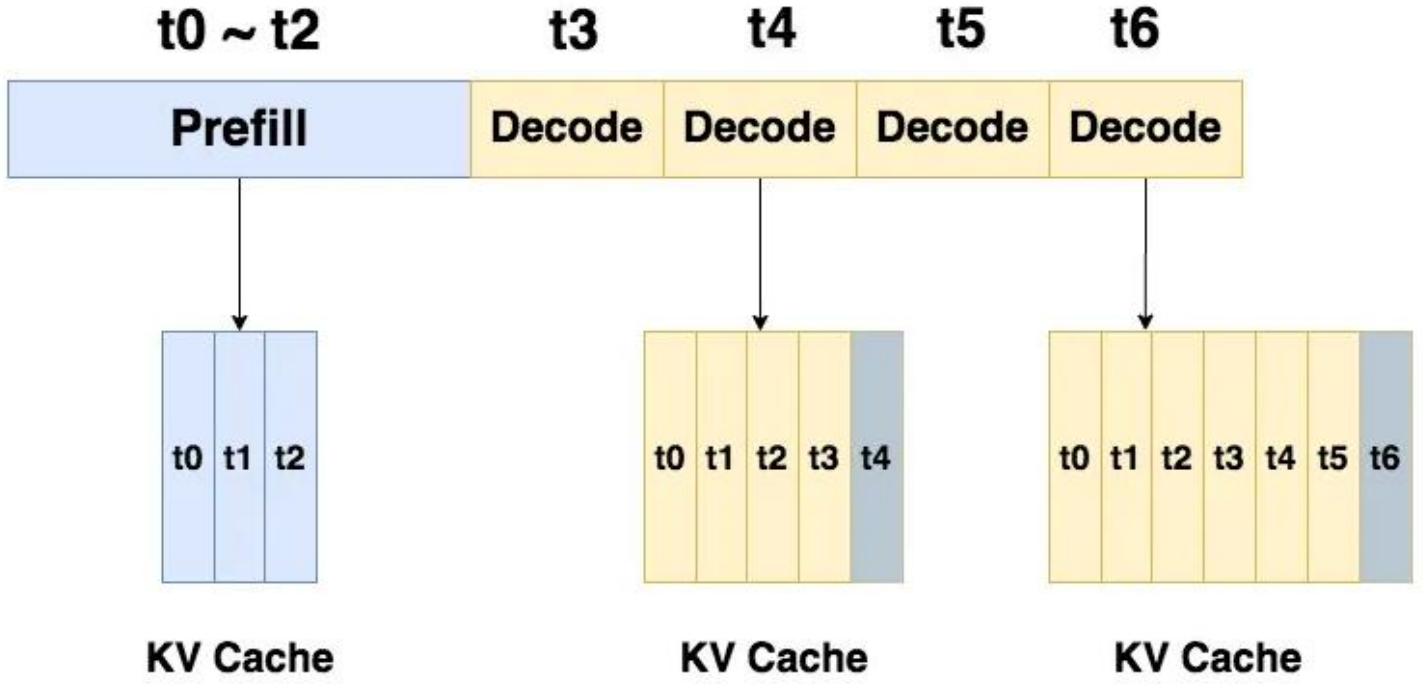
KV Cache：Decode

- Decode 阶段根据 prompt 的 prefill 结果，一个 token 一个 token 地生成 response
- 采用 KV cache 每一个 decode 把对应 response token KV 值存入 cache 中，能加速计算
- t4 与 cache 中 t0~t3 的 KV 值计算完 attention 后，就把自己的 KV 值也装进 cache 中



KV Cache：Decode

- 从上述过程中，可以发现 KV cache 推理时特点：
  1. 随着 prompt 数量变多和序列变长，KV cache 也变大，对 GPU 显存造成压力
  2. 由于输出的序列长度无法预先知道，所以很难提前为 KV cache 量身定制存储空间

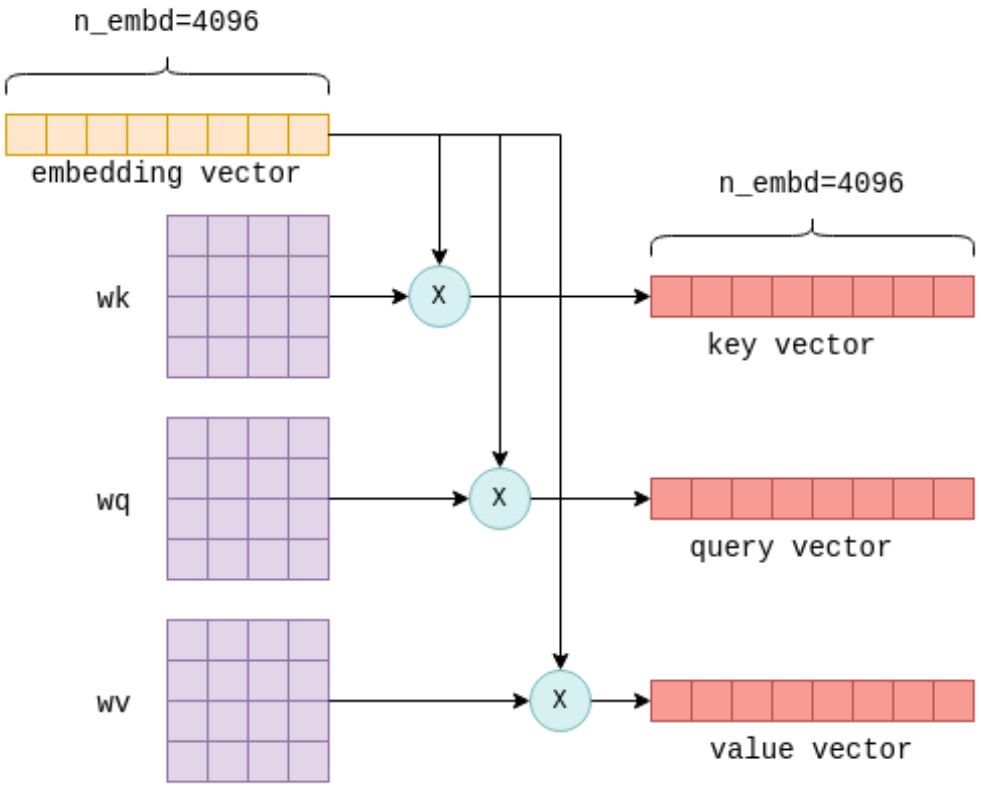


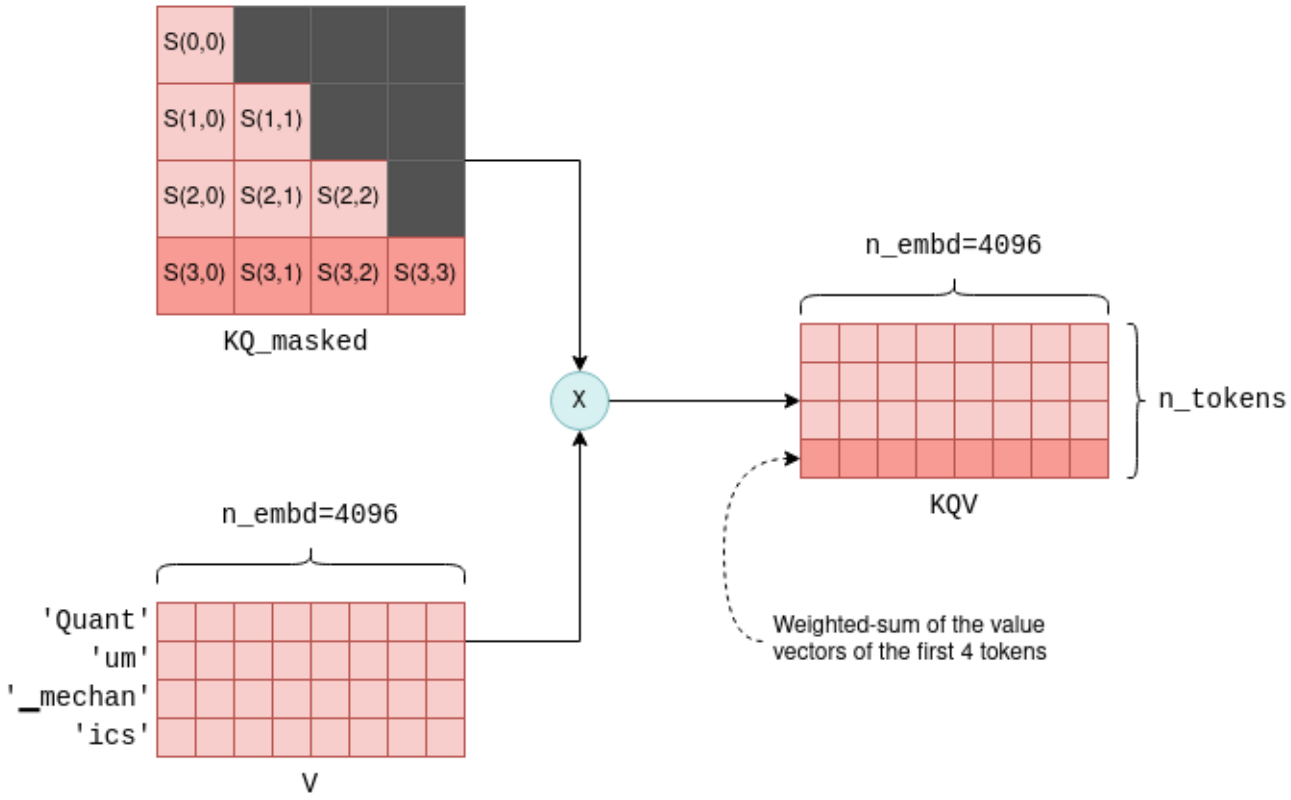
04

KV Cache 瓶颈分析

瓶颈分析

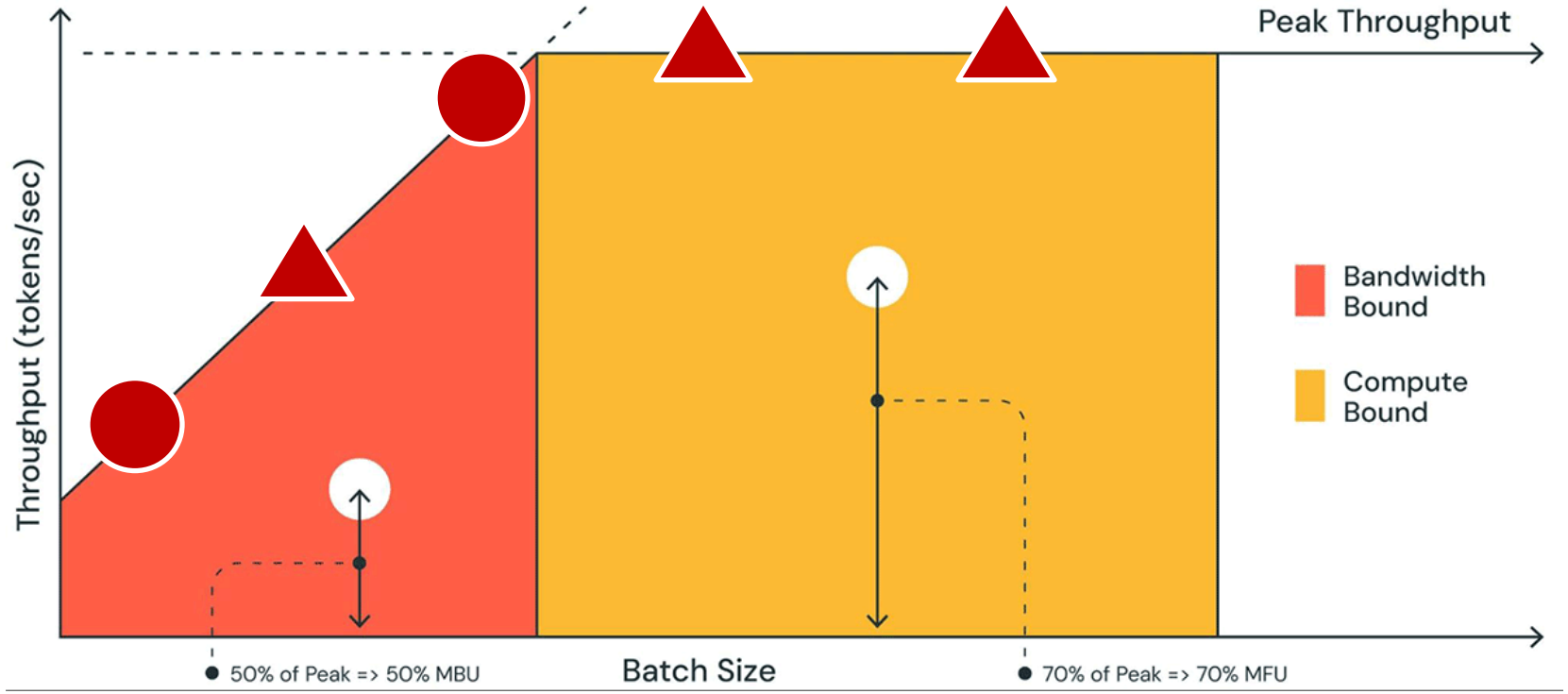
- 训练的过程，大量的矩阵乘，因此整体训练是计算密集型 Computer Bounding





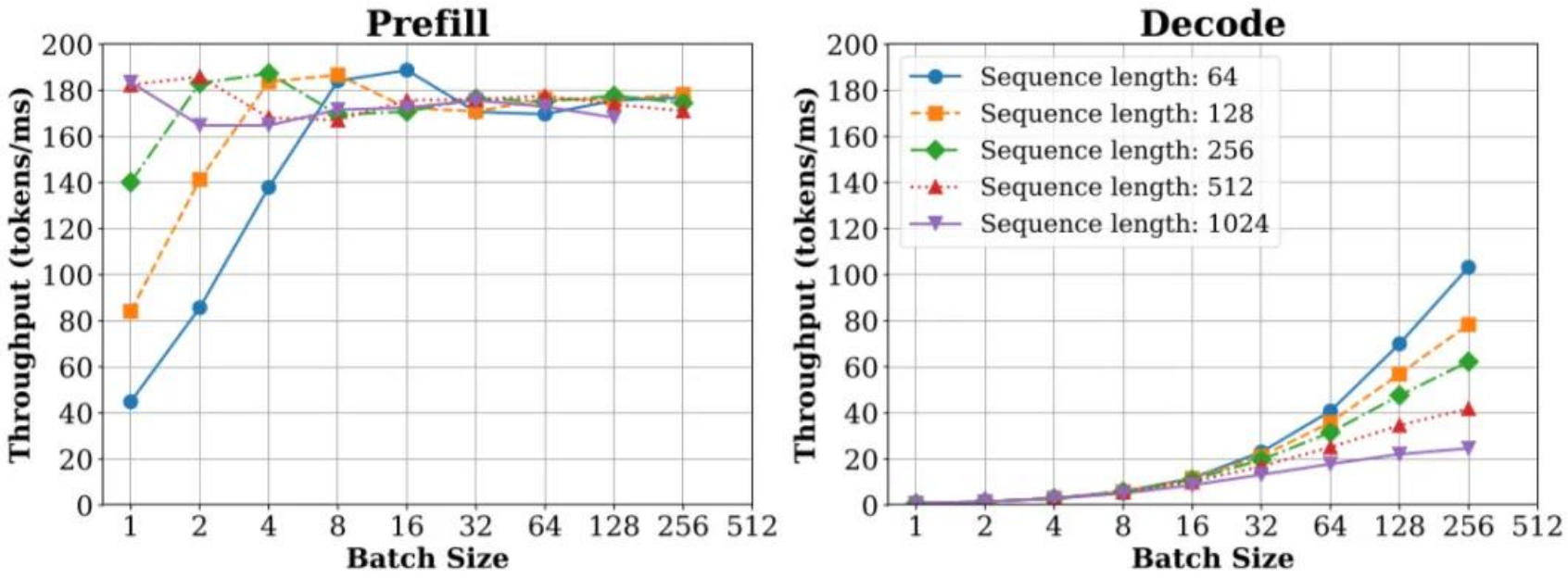
瓶颈分析

- LLM 推理 Prefill & Decoding 阶段 Roofline Model 近似如下，其中：
  1. **三角 Prefill**：设 Batch size=1，Sequence Length 越大计算强度越大，通常属于 Compute Bound
  2. **原型 Decoding**：Batch size 越大，计算强度越大，理论性能峰值越大，通常属于Memory Bound



瓶颈分析

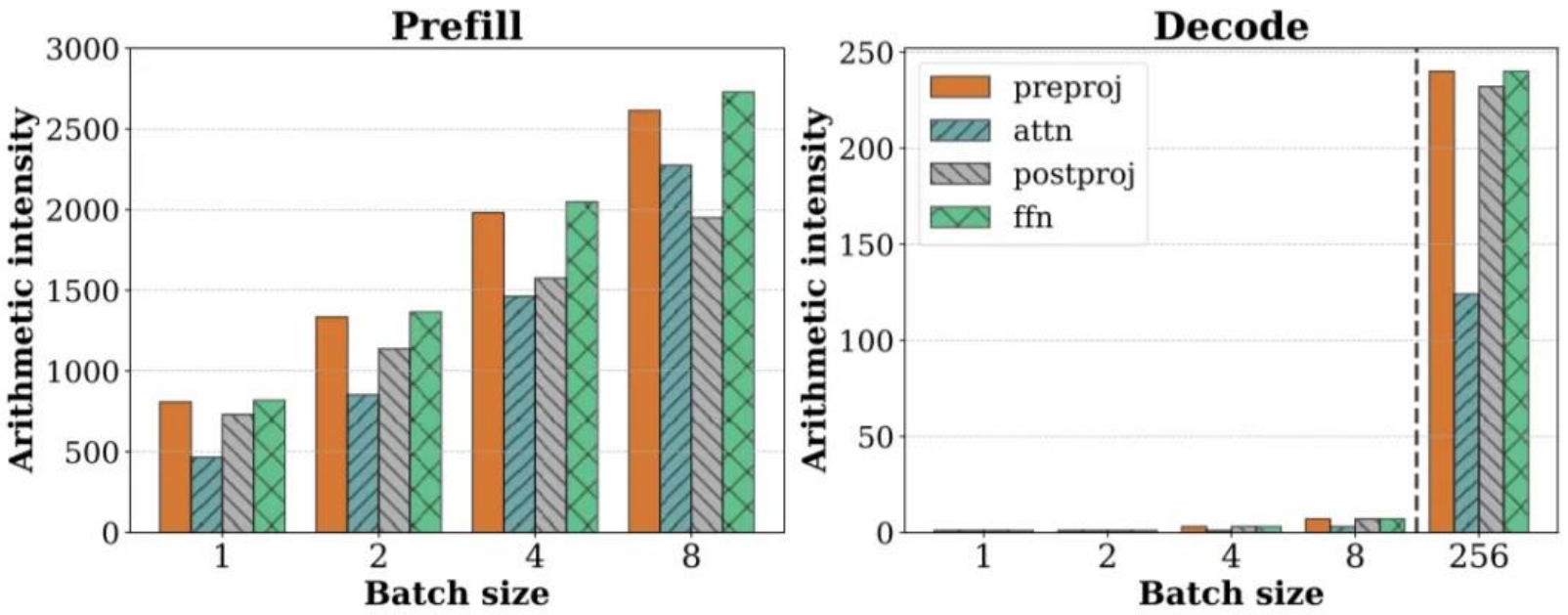
- Prefill 阶段在比较小 Batch Size 下就可以获得比较大的计算强度，相应的吞吐也很高
- Decoding 阶段需要比较大的 Batch Size 才能获得相对高的计算强度及吞吐



(a) Throughput of a single layer of LLaMA-13B on A6000 GPU.

瓶颈分析

- Prefill 阶段在比较小 Batch Size 下就可以获得比较大的计算强度，相应的吞吐也很高
- Decoding 阶段需要比较大的 Batch Size 才能获得相对高的计算强度及吞吐



(b) Arithmetic intensity with 1K sequence length (per-request).

Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

GitHub https://github.com/chenzomi12/AIFoundation

1. https://www.omrimallis.com/posts/understanding-how-llm-inference-works-with-llama-cpp/
2. https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices
3. SARATHI: Efficient LLM Inference by Piggybacking Decodes with Chunked Prefills