Tencent 腾讯

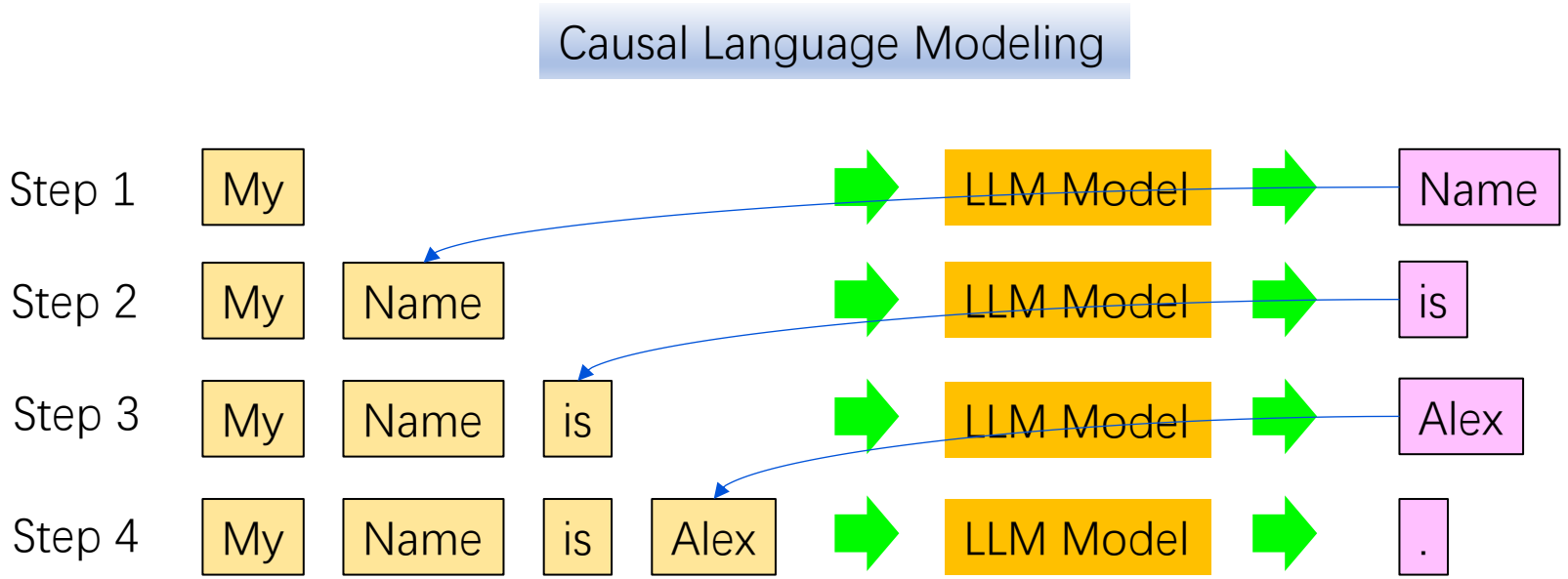
超越自回归限制：大模型倍速推理引擎TACO-LLM的设计与实践

腾讯云异构计算研发负责人　叶帆

目录

1 背景介绍

大模型推理瓶颈



Auto-regressive decoding process -> 性能被内存带宽限制

10B模型
20GB IO
20B FLOP

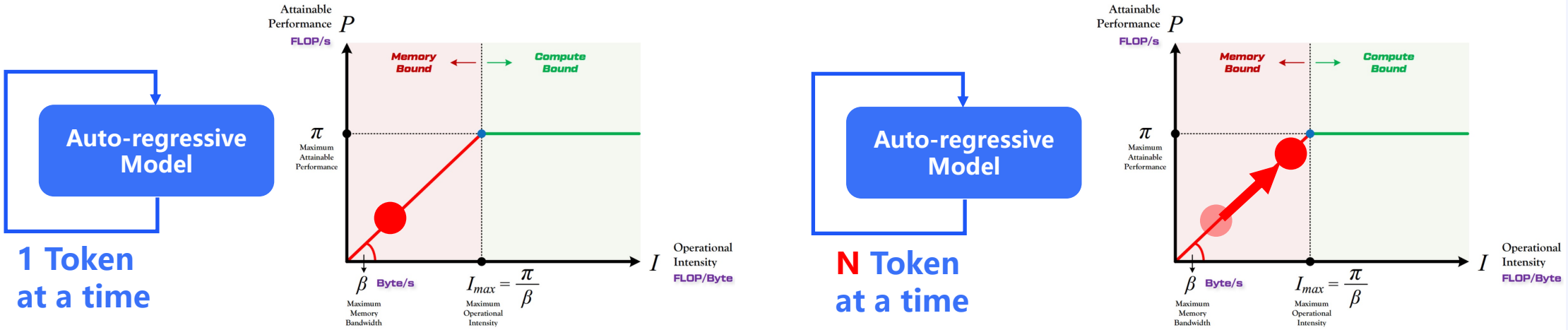
*A100
显存带宽：2039 GB/s
FP16算力：312 TFLOPS

IO耗时：10ms
计算耗时：0.07ms

* For technical discussion and reference only, performance will vary based on different hardware.

解决思路

Increase the arithmetic intensity, reduce # decoding steps



降低model size

- Quantization
- Sparsity
- Distilling
- Tensor-decomposition

减少解码步数

- Block decoding
- Speculative decoding
- Medusa
- LLMA

Non auto-regressive Approach

- NAT
- Retentive Networks
- Layerwise iterative
- Parallel decoding

方案对比

方案	Training-Free	精度无损	Full FLOPS leverage	局限性
Quantization	depends	✗	✗	
Distilling	✗	✗	✗	
Tensor-Decomposition	depends	✗	✗	
Block Decoding	✗	✓	✗	
Speculative Decoding	depends	✓	✗	
Medusa	✗	✓	✗	
LLMA	✓	✓	✗	冷启动
Non Autoregressive Approach	✗	✗	✓*部分	缺乏通用性

TACO-LLM方案设计

1 SOTA

- LLMA/Speculative Decoding
  - 优点：一次生成多个token，可以大大降低单请求的latency；
  - 不足：加速效果依赖token的命中率，存在冗余计算；对于draft model方案还有小模型耗时，小模型难获取等问题。
- PagedAttention
  - 优点：通过Page机制将不同序列的KV-cache统一管理，减少了显存占用，并支持continuous batching
  - 不足：现有kernel实现绑定了GPU的shared memory、registers等资源，对于长序列请求，耗时显著增加，甚至出现资源溢出。
- FlashAttention/FlashDecoding
  - 优点：将不同序列分块计算，将序列的长度和片上资源的使用量解绑，可以实现不同的序列长度，只需要消耗片上固定资源，从而支持更长的序列长度；FlashDecoding在序列维上进一步划分，增加了并行度，实现更低的latency；
  - 不足：不同序列的KV-cache没有做有效的融合；FlashDecoding对batch-size较小时，有比较好的效果，在batch-size比较大时，效果会减弱；另外全局的reduce过程，需要新的kernel来完成，这也增加了额外的开销。

2 Our Contribution

Lookahead Cache + TurboAttention

2 设计实现

优化方案

1 设计思路

- 面向通用性设计
- 提高性能的关键在于充分挖掘GPU冗余算力（bottleneck在访存上）
- 途径是前向计算产生多个token，在序列维增加并行度
- LLM模型由decoding process转变为validation process，validate提前获取的decoding candidate tokens
- 通过Lookahead Cache来获取decoding candidates

2 方案

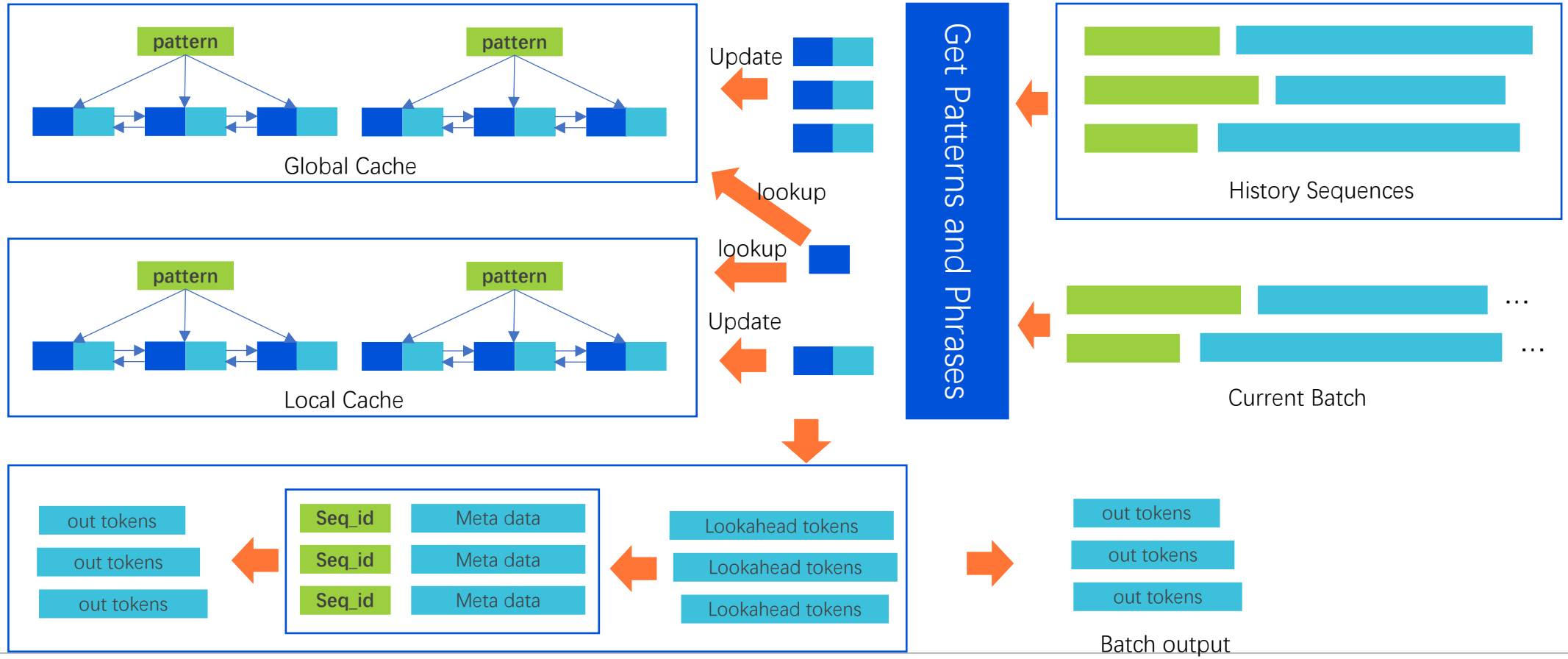
整体方案分两大部分：

- 基于批处理的Lookahead Cache：
  - 一次预测批量请求；
  - 根据batch-size和各个请求的命中率，对copy_len自适应惩罚；
  - 基于森林的多分支预测方法；
- TurboAttention：
  - 基于Paged Attention
  - 借鉴融合FlashAttention，节省显存同时解耦片上资源；
  - Lookahead将向量和矩阵运算，转化为矩阵和矩阵的运算，有效 leverage GPU tensor-core加速；
  - 在Head维使用Double Buffer，实现访存与计算overlap，同时将片上资源与head-size大小解耦。

优化方案

1 支持批处理的Lookahead Cache

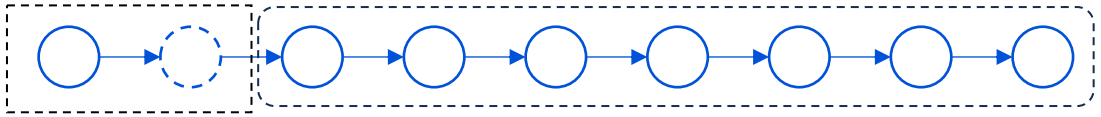
- 一个batch内，每一个序列维护自己的一个local cache，共享全局一个global cache；
- 命中tokens会根据当前序列的命中率和整体的并发度来调整lookahead的长度，不同序列可以支持不同的lookahead长度；



优化方案

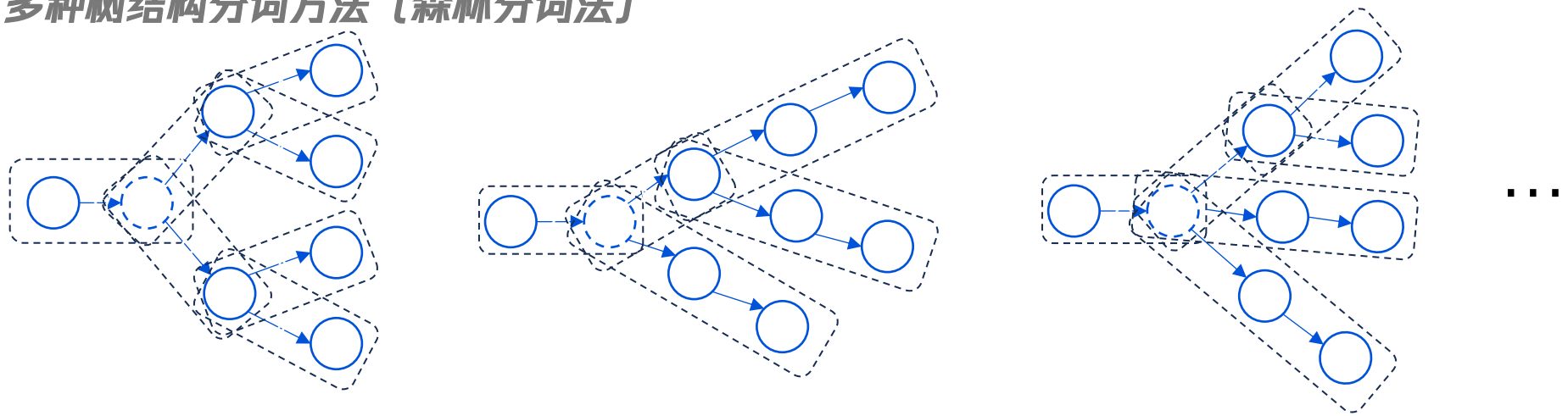
1 Lookahead cache分词方法

- **传统方法**



单分支结构，可以最大化lookahead命中长度，但是有一个大的问题，一旦有一个不命中，后面将都会被拒绝。

- **多种树结构分词方法〔森林分词法〕**



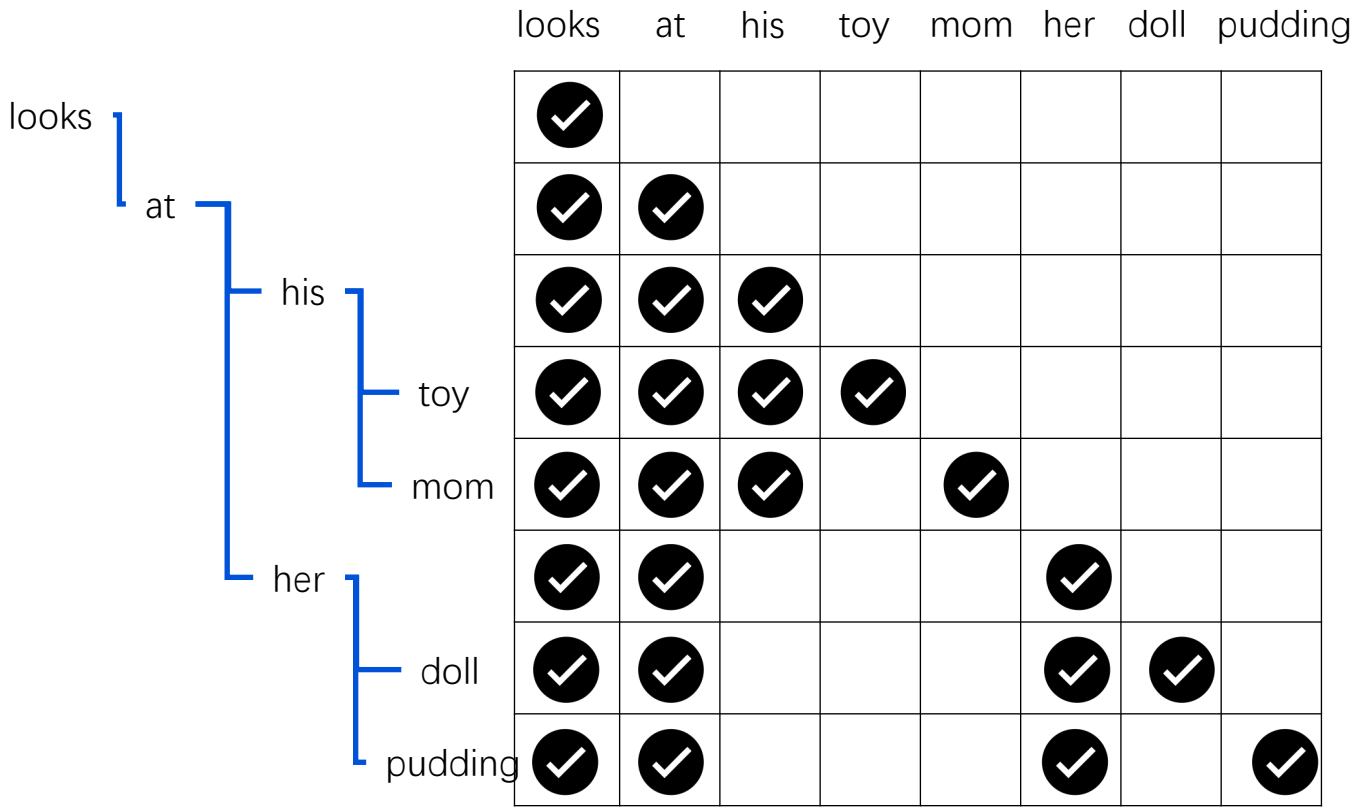
多分支结构（树）：

- 有多种结果候选，提高迭代命中率；
- 树结构多样可以结合词法，语法，将对应的pattern做动态的更新拟合；
- 每一个请求按照各种树结构的权重采样，最终通过命中情况，来调整各种树结构的概率。
- 可以通过简易地随机森林来自学习这个过程，甚至可以在线学习（在CPU上进行）；

优化方案

1 Tree Causal Attention Mask

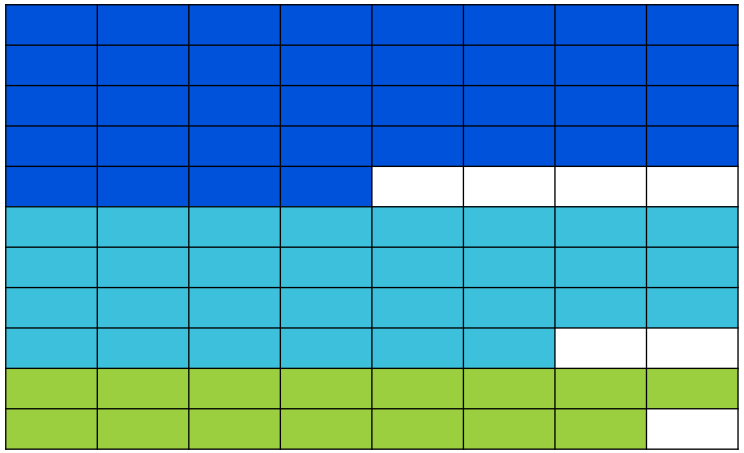
The baby **looks** …



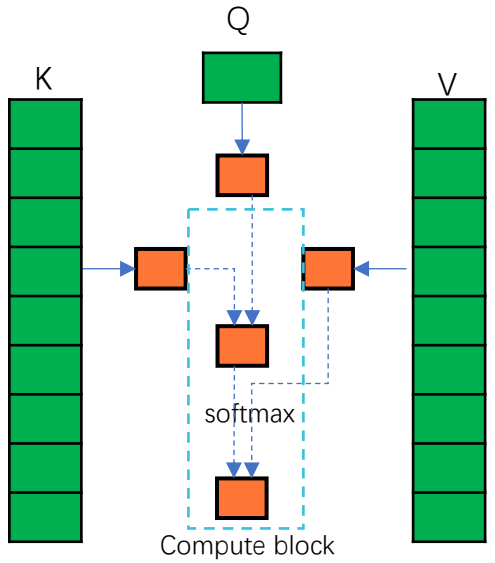
优化方案

2 Turbo Attention

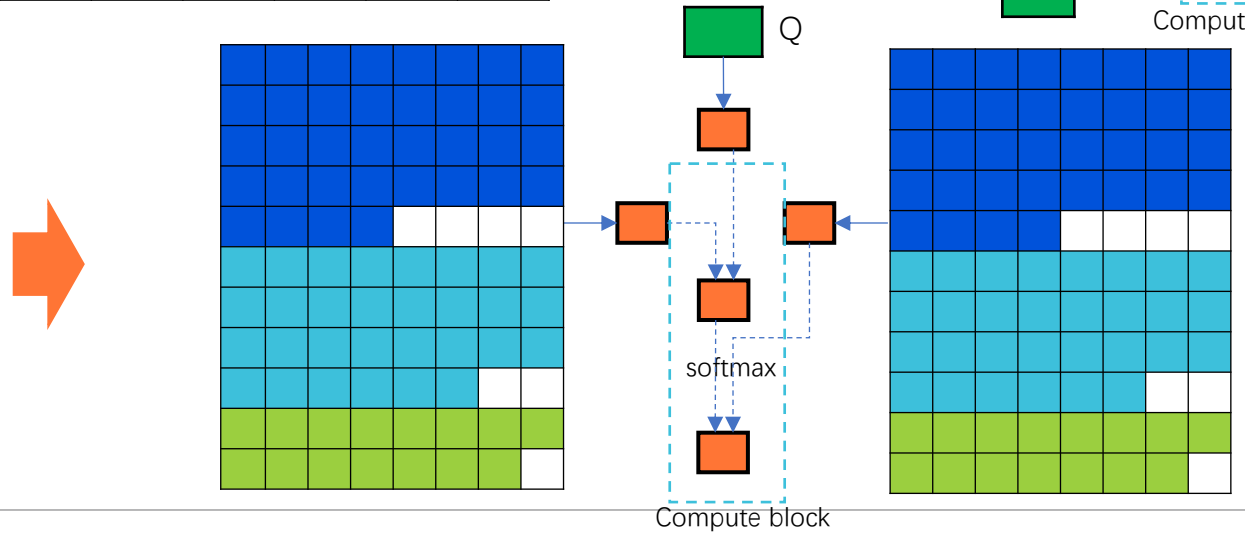
- 既要节省显存，又要节省片上资源（SRAM和register）；



PagedAttention通过一个大的block表，有效地将不同长度的序列batch化。相比之前简单padding的batch方式，这个方式可以显著节省显存，降低显存碎片。



在Decoder场景下，FlashAttention对一个固定的模型，计算时用到的SRAM和register数目与序列长度无关，这样不仅可以支持较长的序列，还可以在长序列时，保持较高的Occupancy。

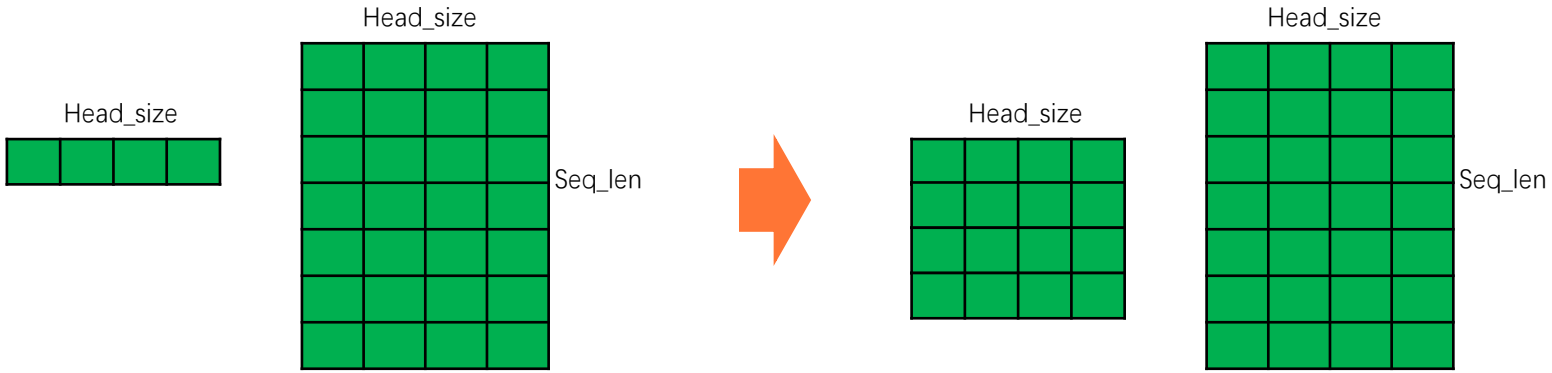


结合两者的长处得到TurboAttention

优化方案

2 Turbo Attention

- 原生支持Lookahead，将向量运算转化为矩阵运算，使$T_n \approx T_1$；



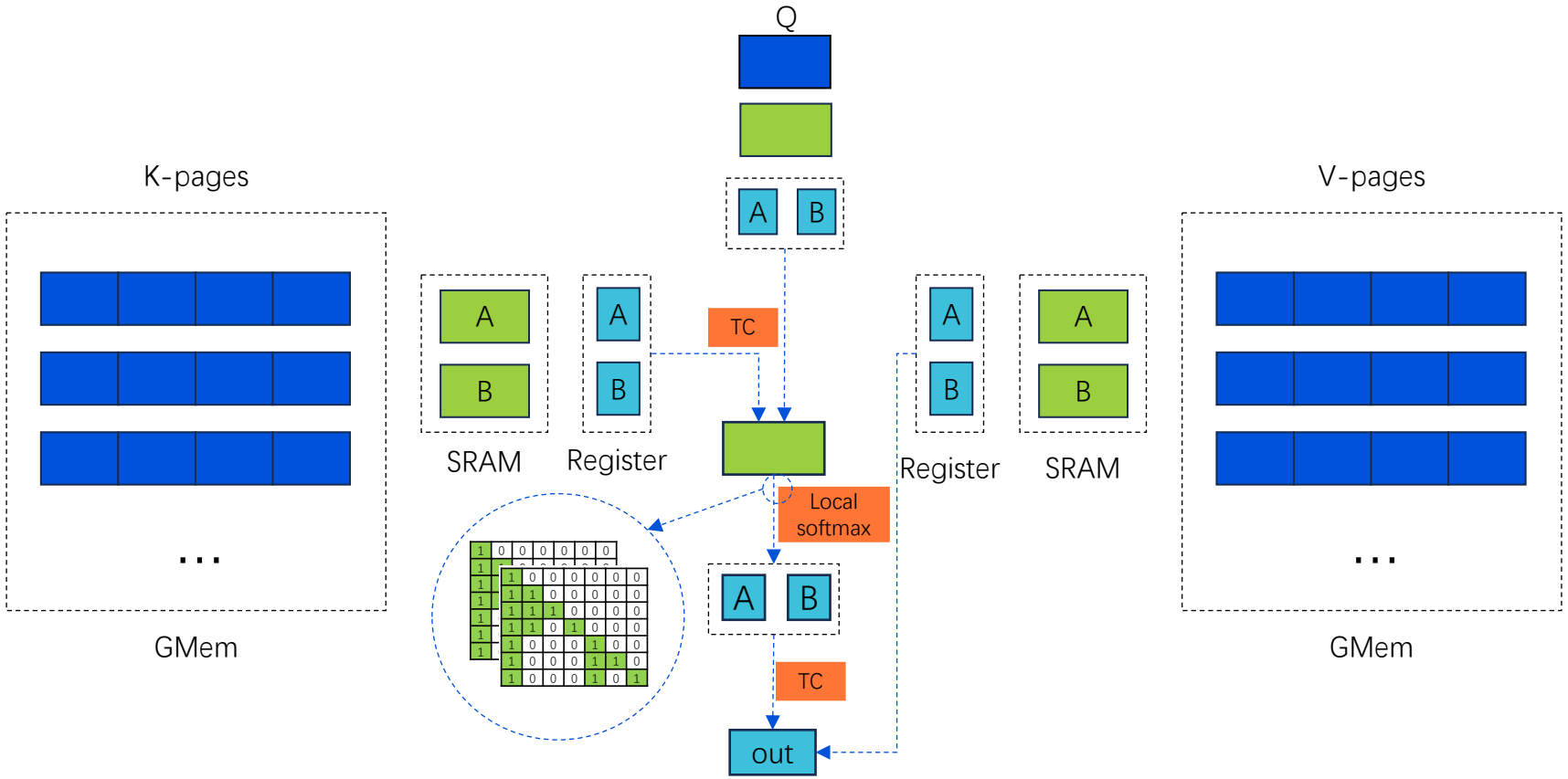
优点:

- 可以自然地利用加速单元例如Tensor-core;
- 可以减少kv cache的访存次数;
- 达到$T_n \approx T_1$的目的，使即使命中率即使为0，吞吐也不会明显下降。

优化方案

2 Turbo Attention

- Head维上使用Double Buffer，使访存和计算overlap，同时使SRAM和register的使用量与head_size的大小解耦；
- 对不同的序列长度和head-size，一个warp读取固定大小的数据且保证大小是一个tensorcore MMA指令要求的整数倍，这样通过ILP可以使每一个线程对Gmem的请求次数降低为1。



总结

1 提高Lookahead Cache命中率，减少无效计算

- 通过森林分词，有效提高decoding candidate tokens的命中长度，从而减少无效计算
- 通过Tree Attention，合并相同前序candidate tokens的计算，减少重复计算
- 自适应candidate tokens长度，各序列按自身情况灵活调整（命中率、并发度、自适应惩罚），避免过载GPU冗余算力

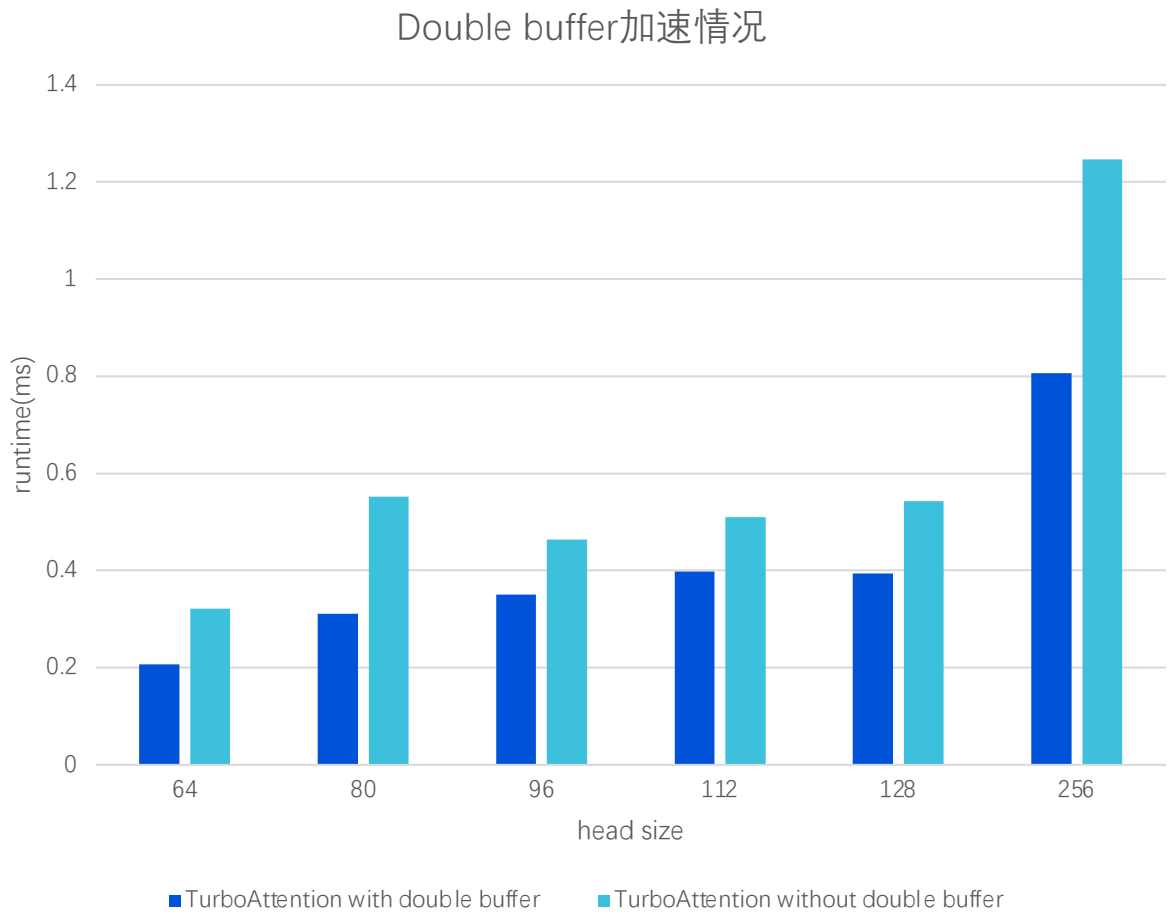
2 提高算力天花板

- 序列维增加的并行度可有效leverage tensorcore，提高理论算力上限，并使得性能下限和单token解码持平
- 将片上资源使用（SRAM、registers, etc.）和序列长度解耦，有效提升长序列性能
- 通过双buffer设计提高ILP，有效隐藏IO开销

3 性能分析

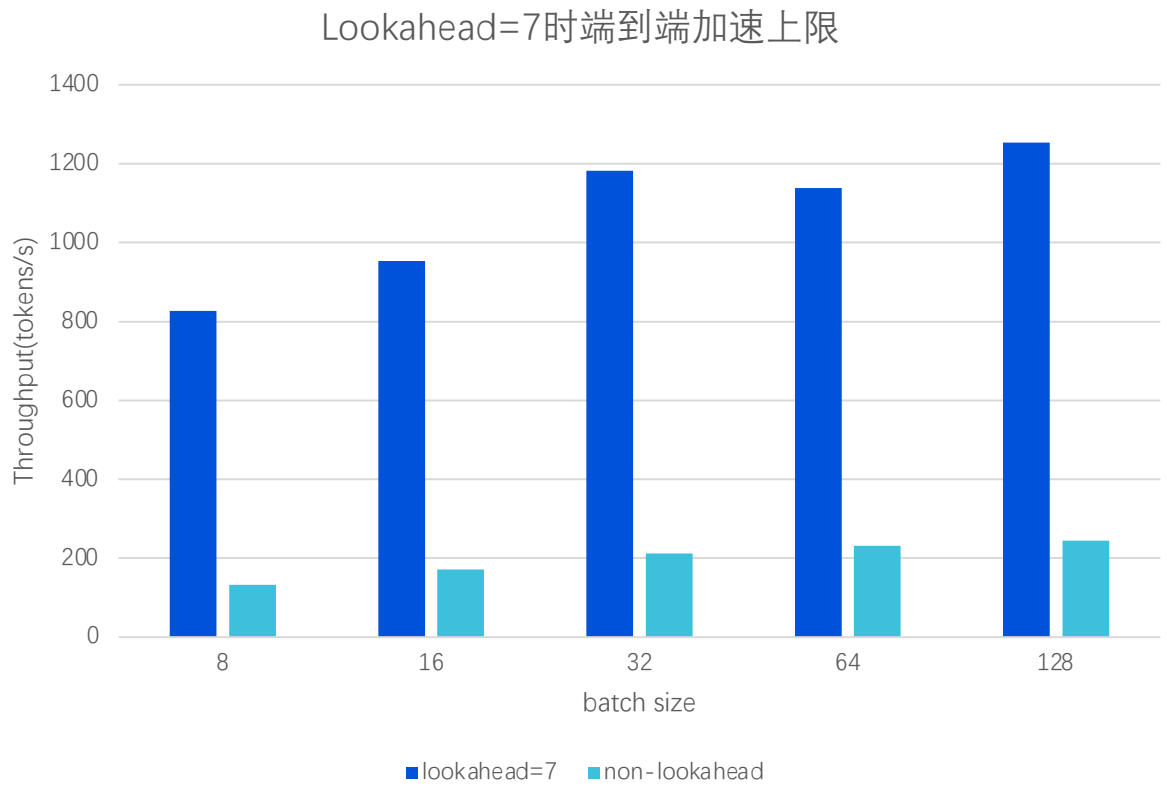
1 Double buffer加速效果

- 在各种head size下，使用head维double buffer技术，耗时明显减小，最大加速1.77x。



2 加速上限分析

- Lookahead=7
  - batch-size=8时，加速上限为6.2x
  - batch-size=128时，加速上限为5.12x
- 大多投机采样的加速方法，只在低并发（batch-size=1）时，才有加速效果。
- TACO-LLM的方案，一直到batch-size=128时，都保持较高的加速潜力。以batch-size=128的加速上限为5.12x为例，假设lookahead=7时，平均只命中了3.5个token，那么将有2.88x的加速。



4 *命中率分析

- 并非所有的迭代都会触发cache，所有迭代里只有60%的触发cache
- 预测tokens中，只有61%的命中率，全局命中长度最大为2.45，是最大预测长度的35%，远没有达到加速上限
- 当前工作采用森林分词改进线性预测的命中情况

Batch size	迭代命中率	全局命中长度	tokens命中率	加速倍数
128	0.5958	1.0796	0.5995	1.83x
64	0.6093	1.1304	0.6153	1.88x
32	0.5851	1.1294	0.5875	1.92x
16	0.5708	1.6543	0.5483	1.95x
8	0.556	2.4503	0.4495	2.48x
1	0.4929	1.9971	0.3828	2.46x

* 这里展示的是传统线性预测结构的命中率

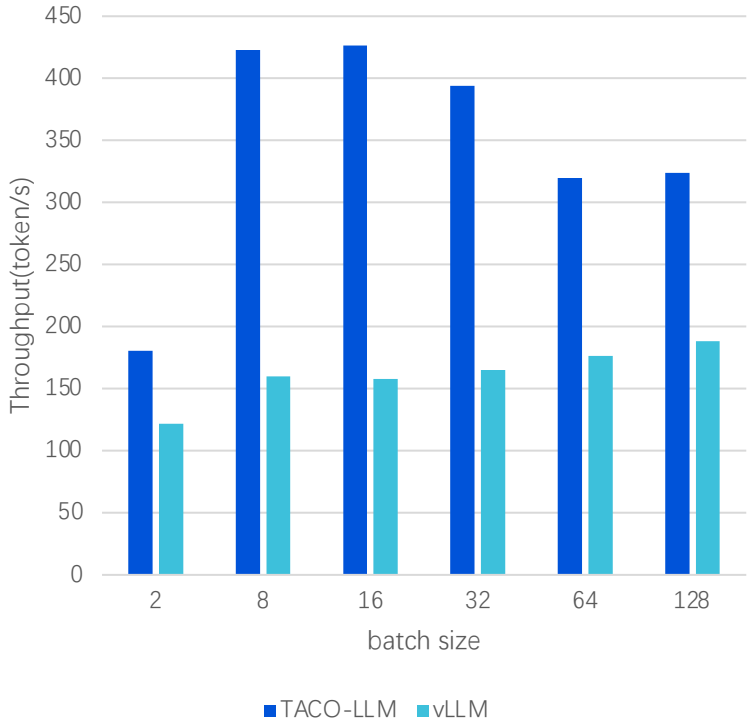
3 端到端性能

同batch size下，TACO-LLM

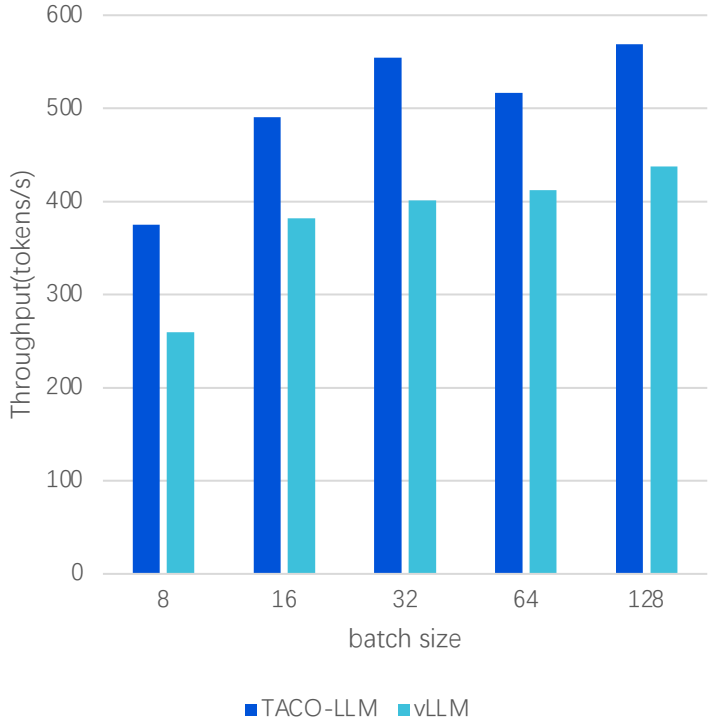
- 某LLM-6b-32k，相比vLLM最大提升**3.17x**
- 某LLM-13b，相比vLLM最大提升**1.44x**
- codellama-13b，相比vLLM最大提升**2.47x**

TACO-LLM全面优于vLLM

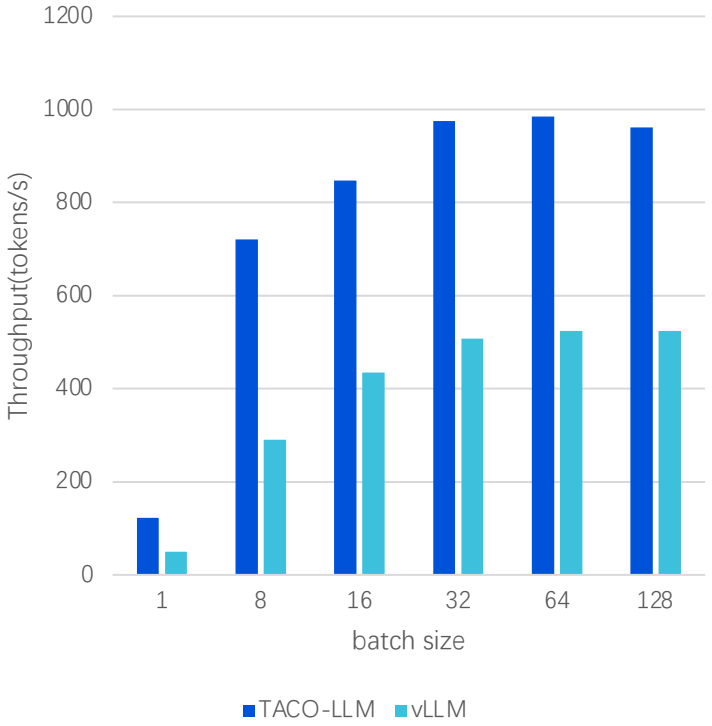
某LLM-6b-32K(*A100x1)



某LLM-13b(*A100x1)



*codellama-13b(*A100x1)



* For technical discussion and reference only, performance will vary based on different hardware.

5 实际业务效果

客户场景

- **模型**

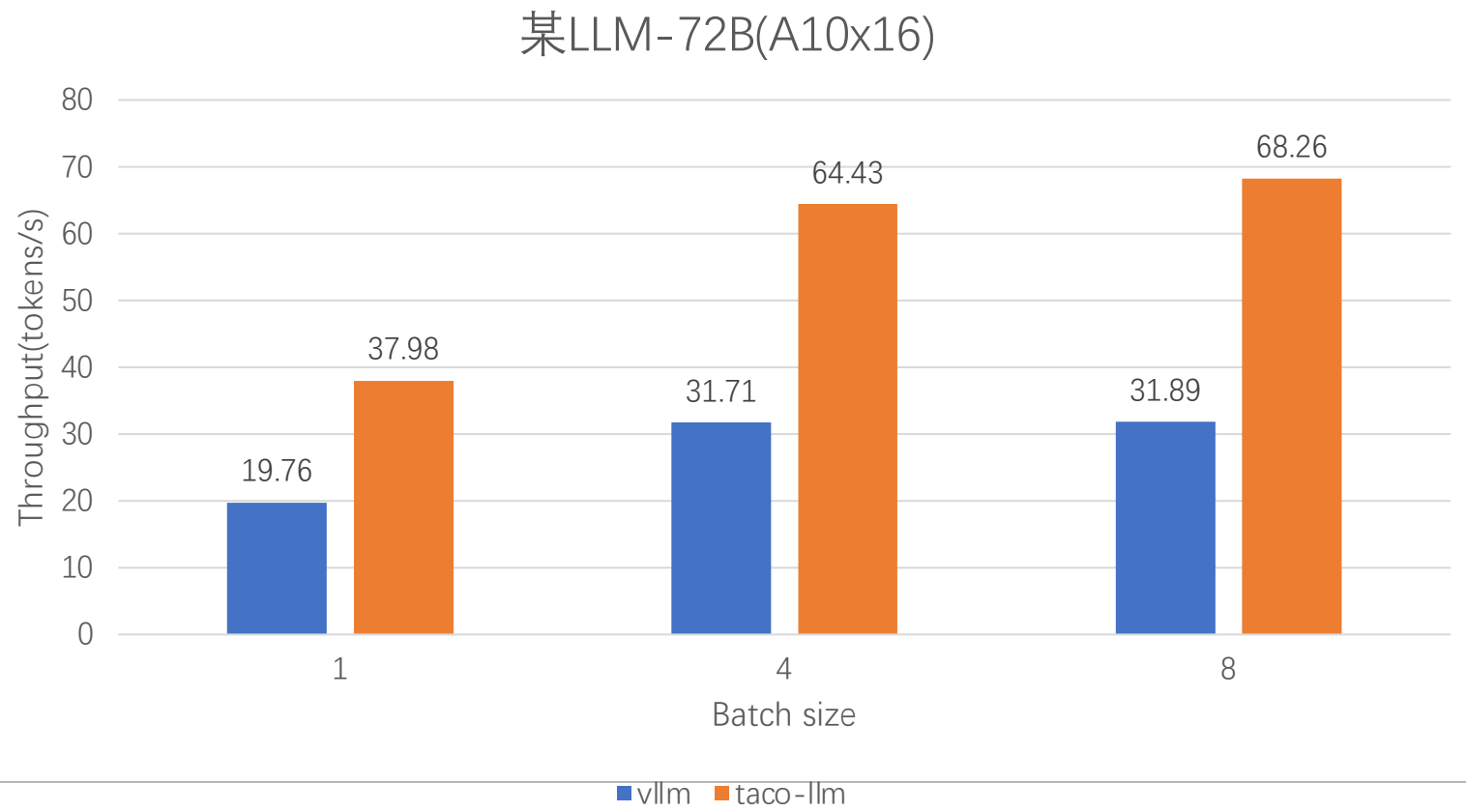
选用某LLM-72B模型，输入4K，输出1K，要求首字延迟2-3秒，全部请求5秒内完成。

- **业务**

1）人岗匹配业务。从一系列简历中，通过模型寻找合适的岗位候选人。
2）在线客户业务。当前有一些复杂的交互业务，机器人回答质量欠佳，引入该大模型优化业务体验。

测试效果

在batch size为**1/4/8**的情况下，taco-llm对比vllm吞吐加速分别达到了**1.92/2.03/2.14**倍。



4 未来展望

Future Work

- 进一步优化Lookahead Cache构建及维护方式
- 进一步提高cache命中率，在控制candidate tokens数量的前提下，进一步提高平均匹配长度。
- Join us: alexfye@tencent.com

Thanks

Tencent 腾讯