Punica: Multi-Tenant LoRA Serving

MLSys'24

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, Arvind Krishnamurthy

University of Washington, Duke University

Author

Research Interests

- Distributed Systems
- Machine Learning Systems
- Operating Systems

University of Washington

[1] Nexus: a GPU cluster engine for accelerating DNN-based video analysis. SOSP’19

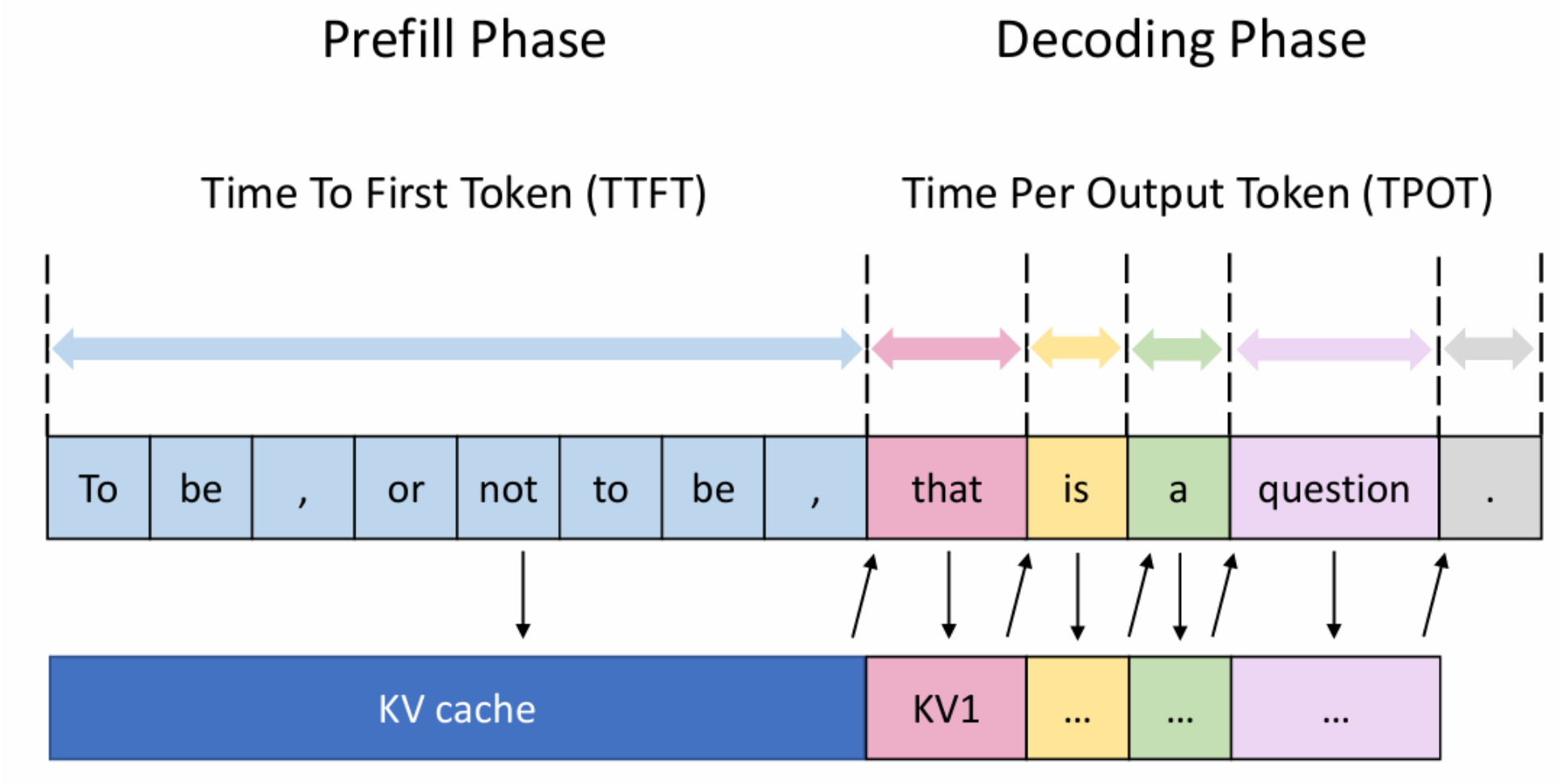
[2] Atom: Low-bit quantization for efficient and accurate llm serving. MLSys’24

Contents

- Background
- Related Work
- Design
- Evaluation
- Thinking

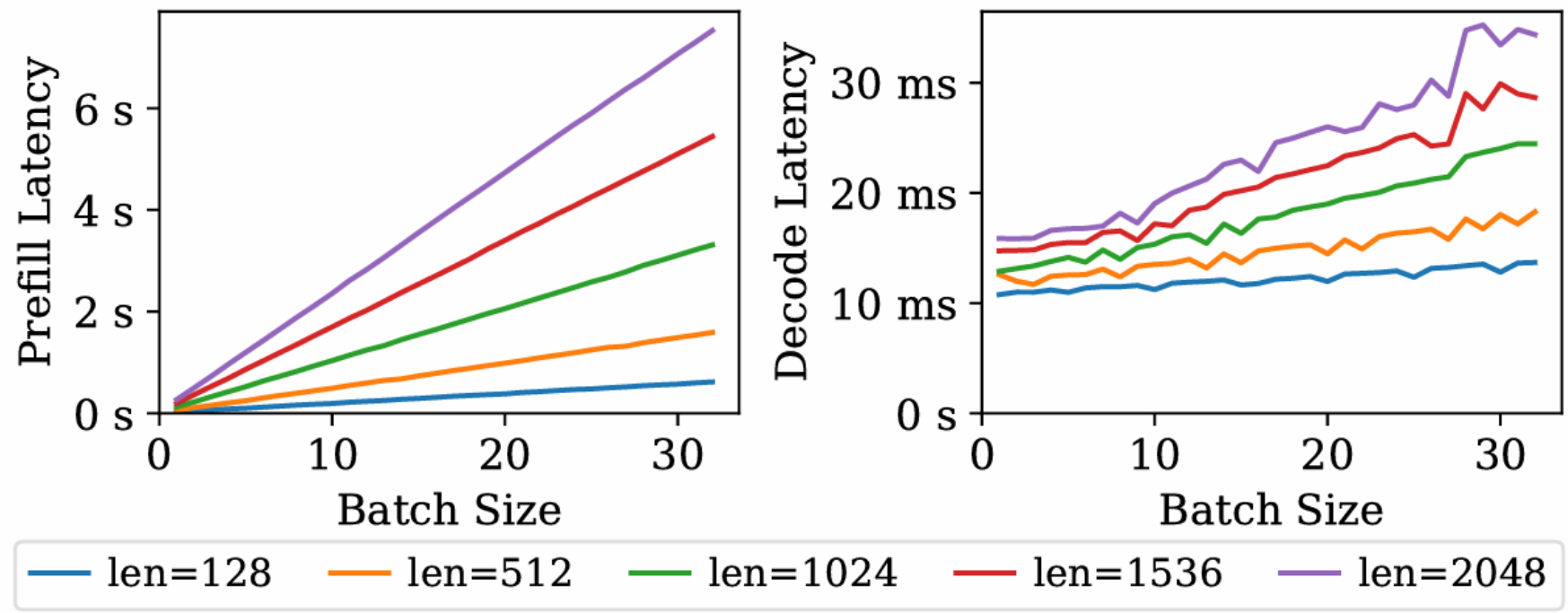
Background

- Transformer过程



Background

- Transformer过程

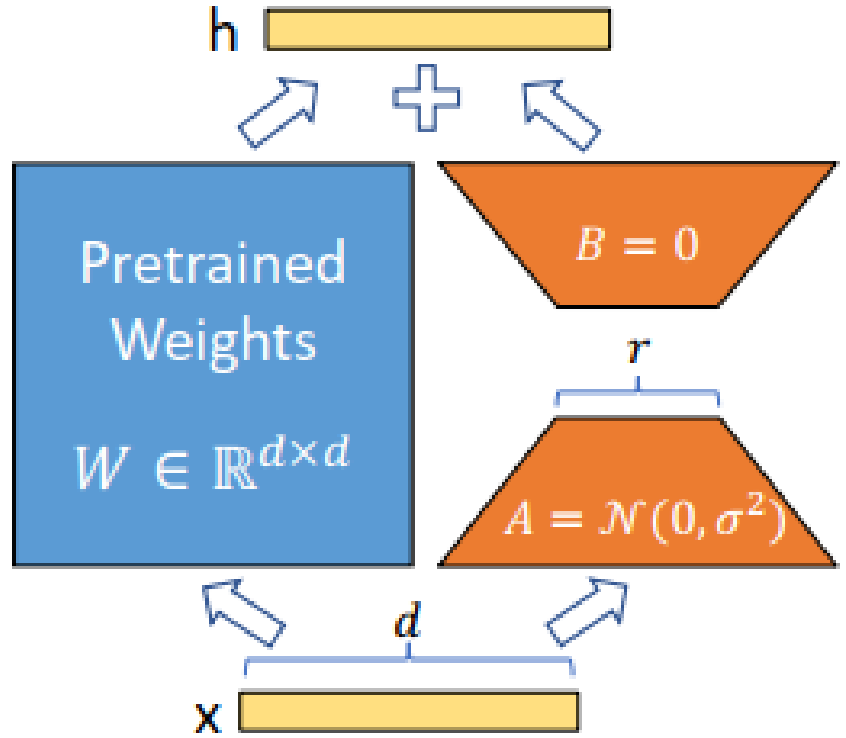


Prefill：GPU充分利用　　Decode：GPU利用率低

批处理提高GPU利用率

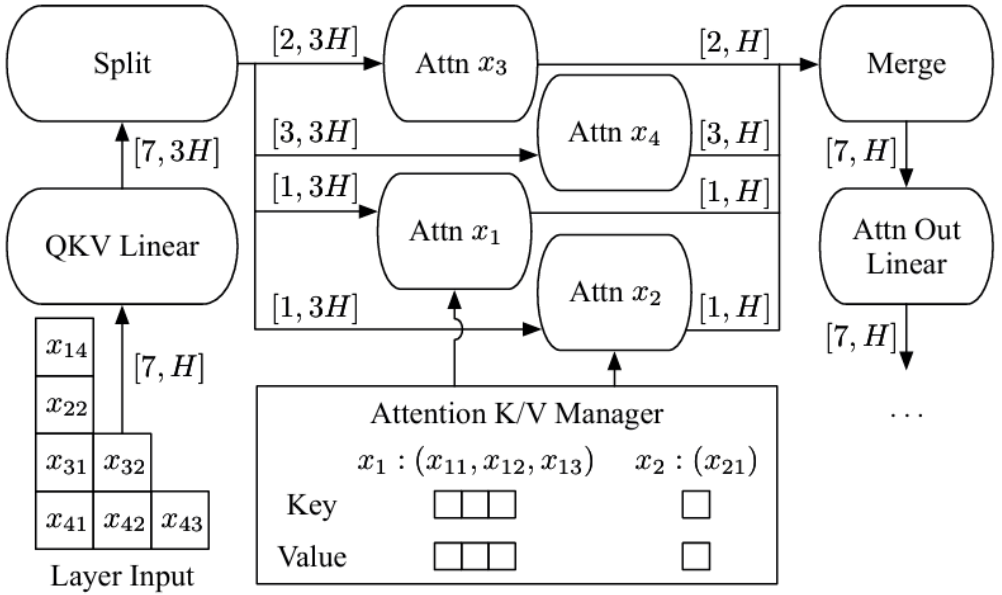
Background

- LoRA (Low-Rank Adaptation of LLMs)
  - 显著减少训练参数量 (e.g., r=16, d=4096)
    - 快速微调
    - 降低相同预训练模型不同LoRA模型存储和内存开销
      - $W' = W + AB$
      - $xW' = x(W + AB) = xW + xAB$



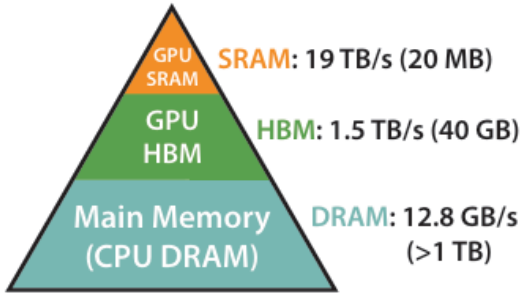
Related Work

- LLM推理优化
  - Orca: A Distributed Serving System for Transformer-Based Generative Models. OSDI'22 提出通过在自注意力操作处分割连接的批量输入来批量处理基于Transformer的文本生成

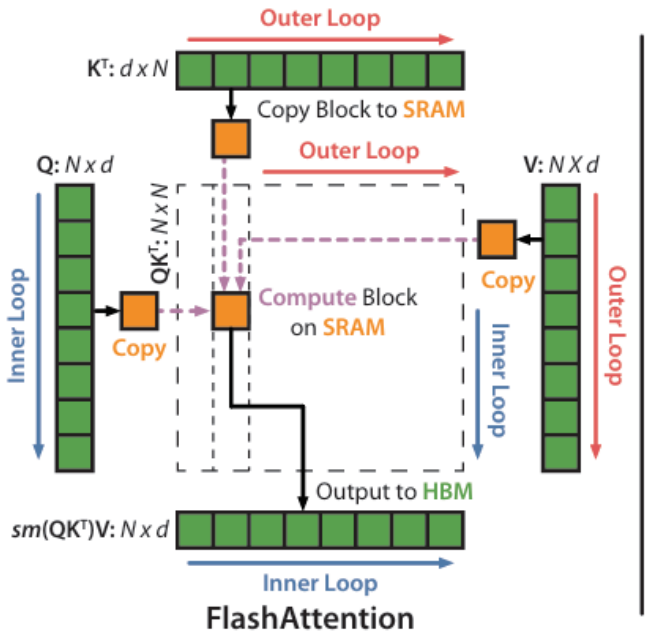


Related Work

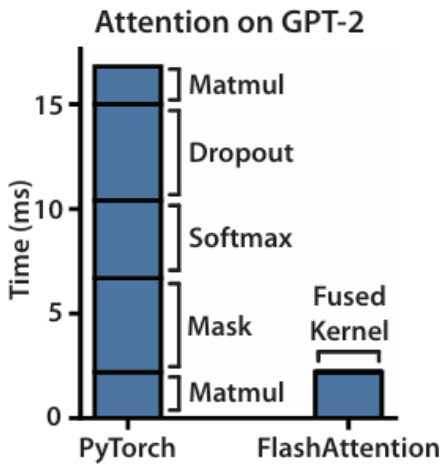
- LLM推理优化
  - FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. NeurIPS'22 提出通过块状计算来减少Attention的数据移动



Memory Hierarchy with Bandwidth & Memory Size

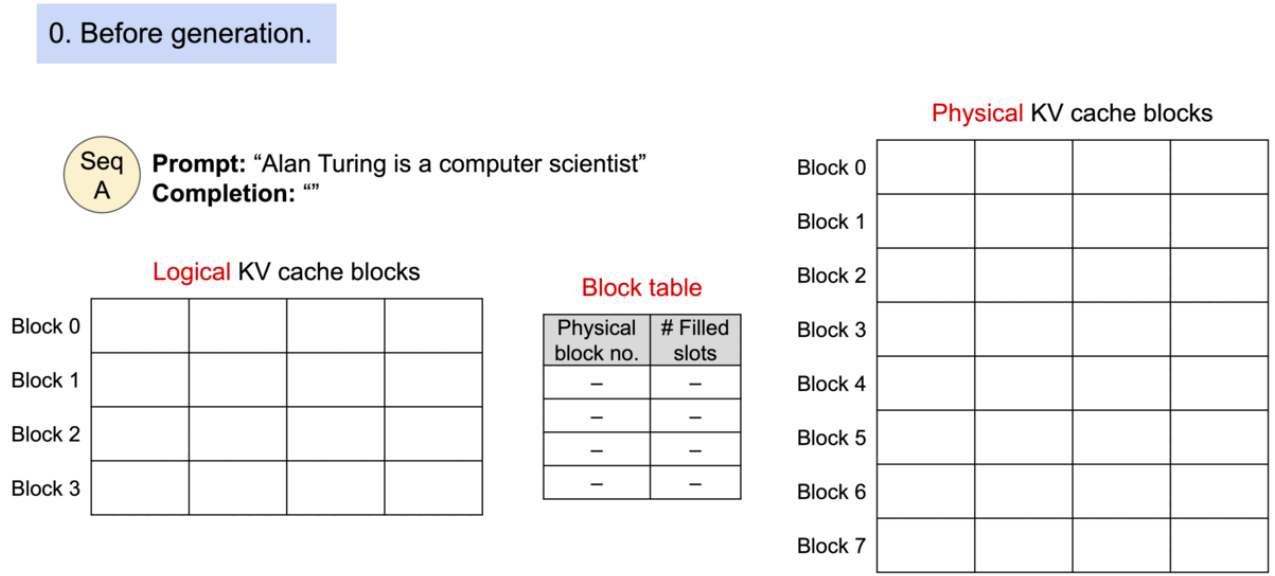


FlashAttention



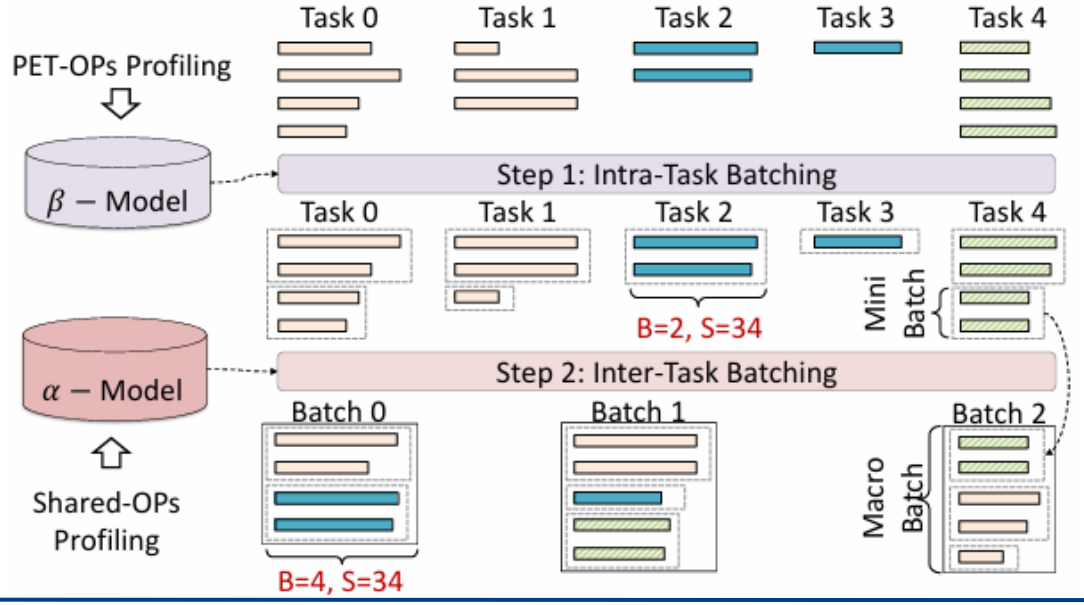
Related Work

- LLM推理优化
  - Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP'23 提出通过借鉴操作系统中的虚拟页面的概念来减少KvCache的内存碎片化



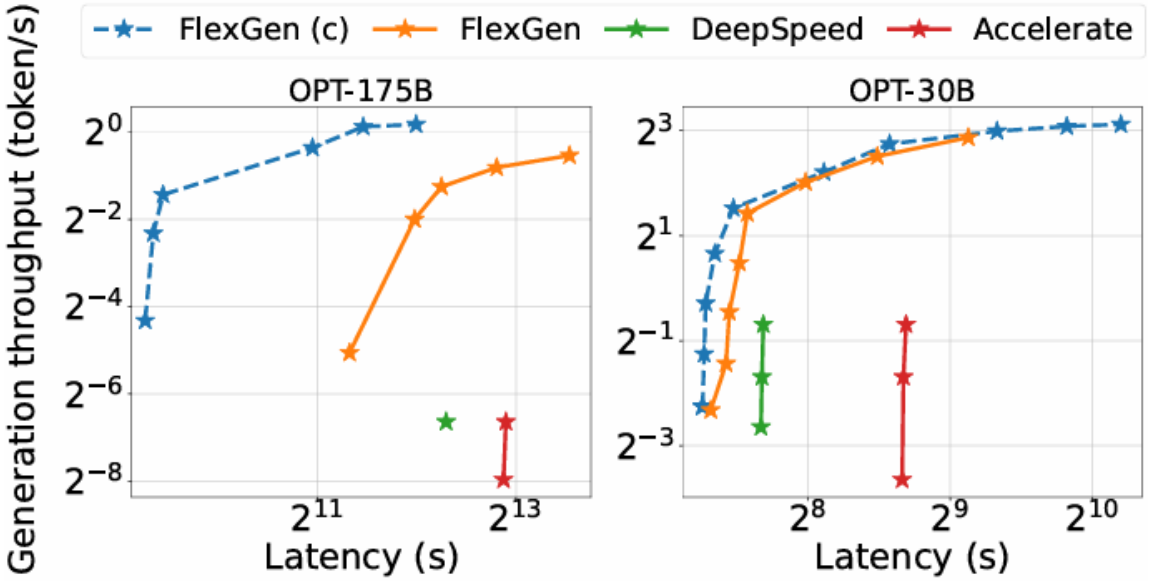
Related Work

- 多模型推理服务
  - PetS: A Unified Framework for Parameter-Efficient Transformers Serving. USENIX ATC'22 允许预训练模型在GPU内存中为不同的下游任务共享，但是它不能让多个不同的模型同时运行



Related Work

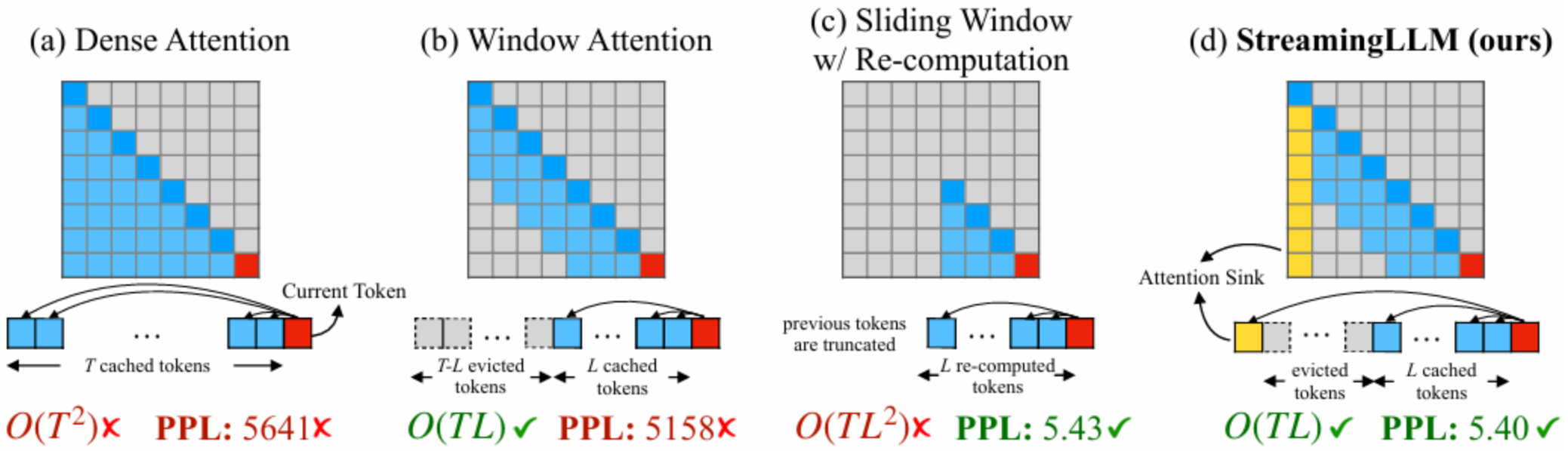
- 模型和KVCache压缩
  - Flexgen: High-throughput generative inference of large language models with a single GPU. ICML'23 模型和KVCache压缩节省了更多的内存消耗，使GPU能够处理更长序列的请求



Related Work

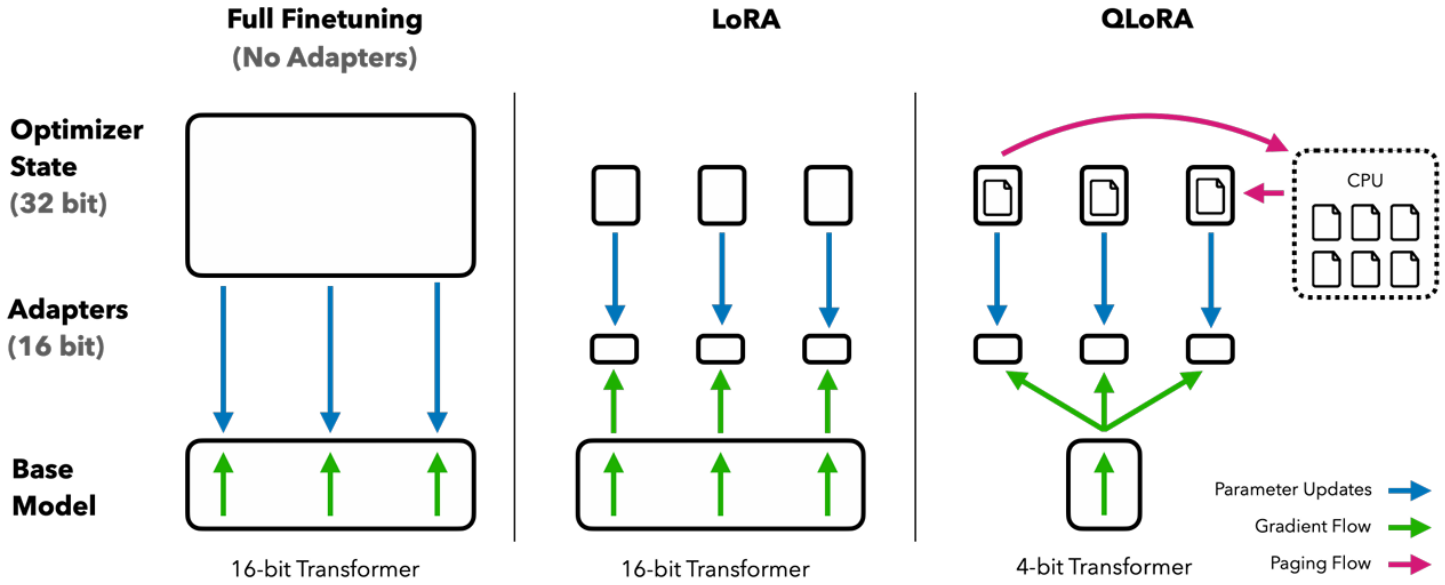
- 模型和KVCache压缩
  - Efficient Streaming Language Models with Attention Sinks. ICLR'24

    进一步减少了KVCache的内存IO，降低了推理延迟



Related Work

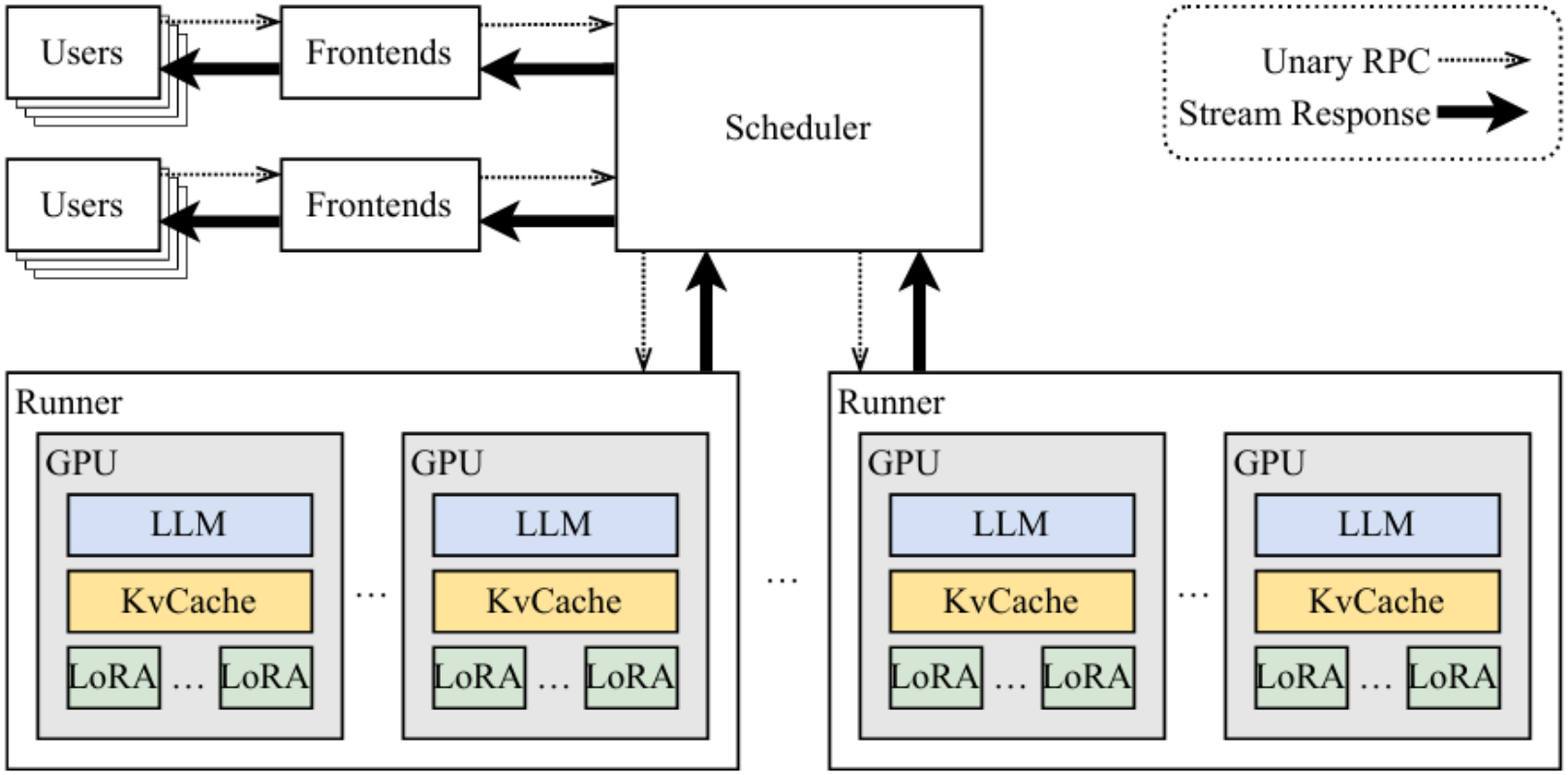
- 模型和KVCache压缩
  - QLoRA: Efficient Finetuning of Quantized LLMs. NeurIPS'23 提出通过高精度格式存储LoRA权重，同时保留原始权重的量化格式，以在微调过程中节省内存占用



Design

总体设计

- Frontends <===> Scheduler <===> Runner



Design

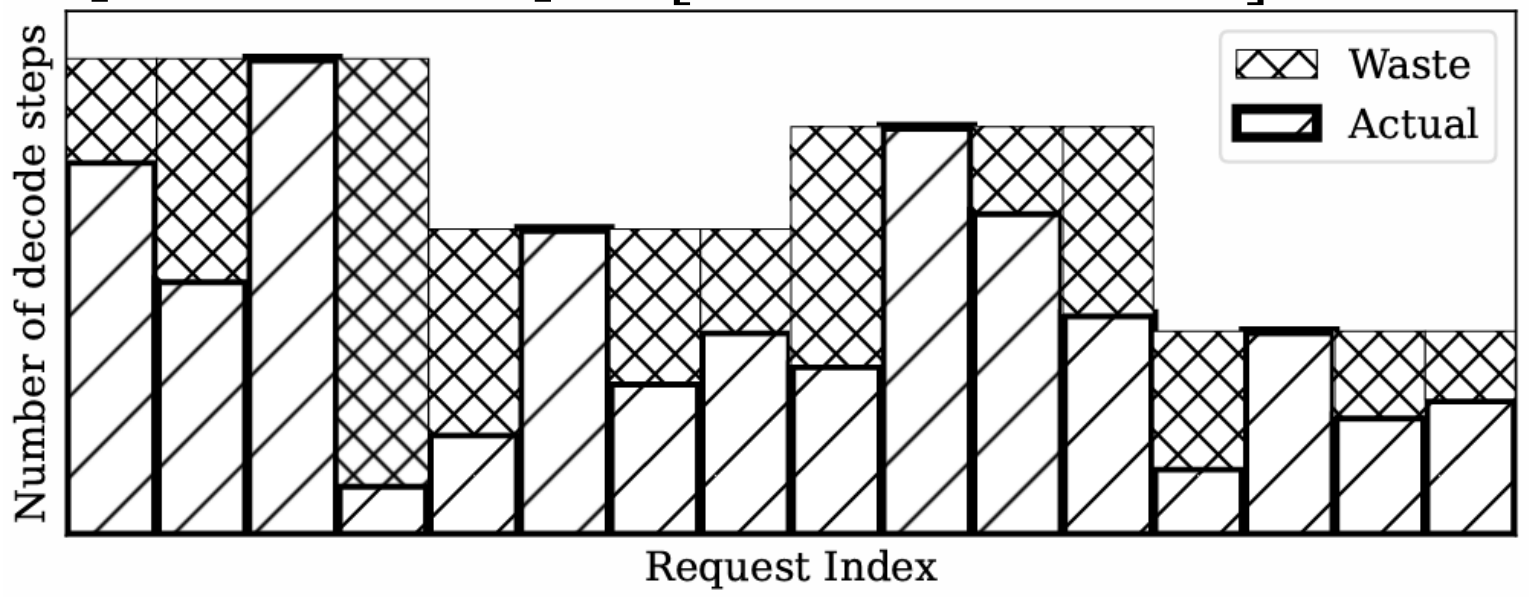
总体设计

- Q1: 如何在GPU上高效运行多个LoRA模型
  - 新的CUDA内核Segmented Gather Matrix-Vector Multiplication (SGMV) 实现对不同LoRA模型请求的批量处理
- Q2: 如何依靠任务调度实现工作负载集中
  - 对用户请求进行批处理，以提高GPU占用率
  - 实现按需LoRA模型加载，避免非活跃模型内存占用
  - 将用户请求调用到活跃的GPU上，以占用最少的GPU资源

Design

- Q1: 如何在GPU上高效运行多个LoRA模型
  - KVCache 张量嵌套
    - HuggingFace Transformers 库的 KVCache 张量嵌套：

$$[L, 2, B, N, S, D] \Rightarrow [L, 2, B, N, S+1, D]$$



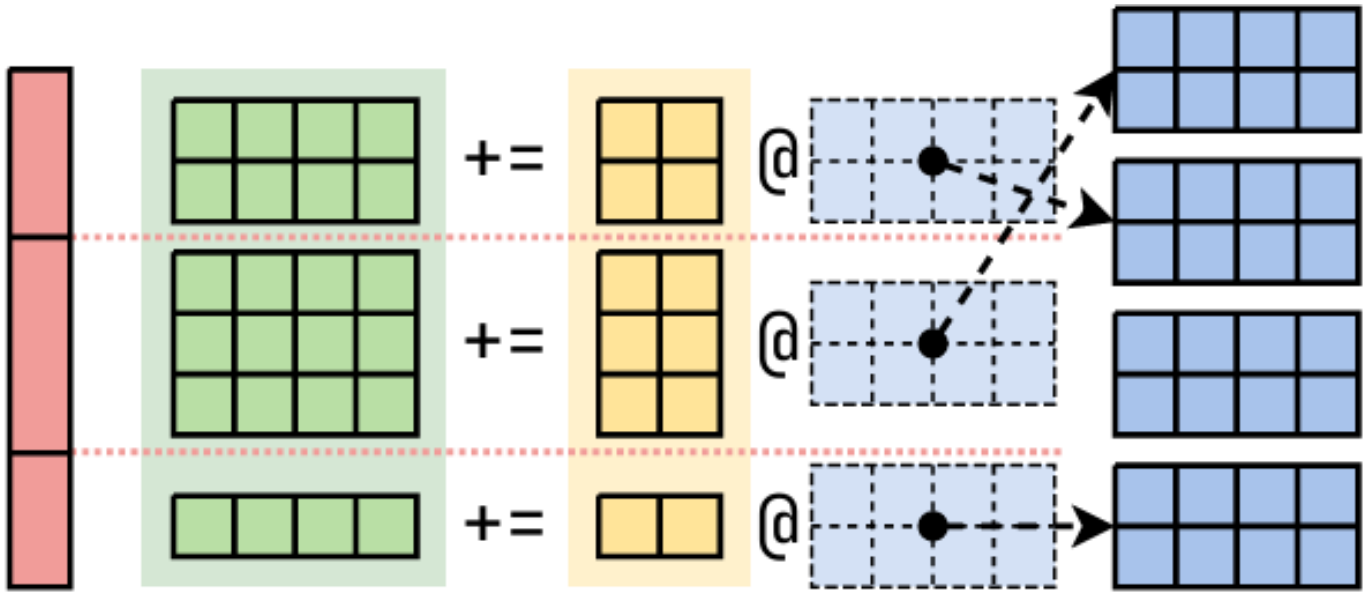
Design

- Q1: 如何在GPU上高效运行多个LoRA模型
  - KVCache 张量嵌套
    - Punica 的 KVCache 张量嵌套：

$$[\sum_i \left\lceil \frac{S_i}{P} \right\rceil, L, 2, N, P, D]$$

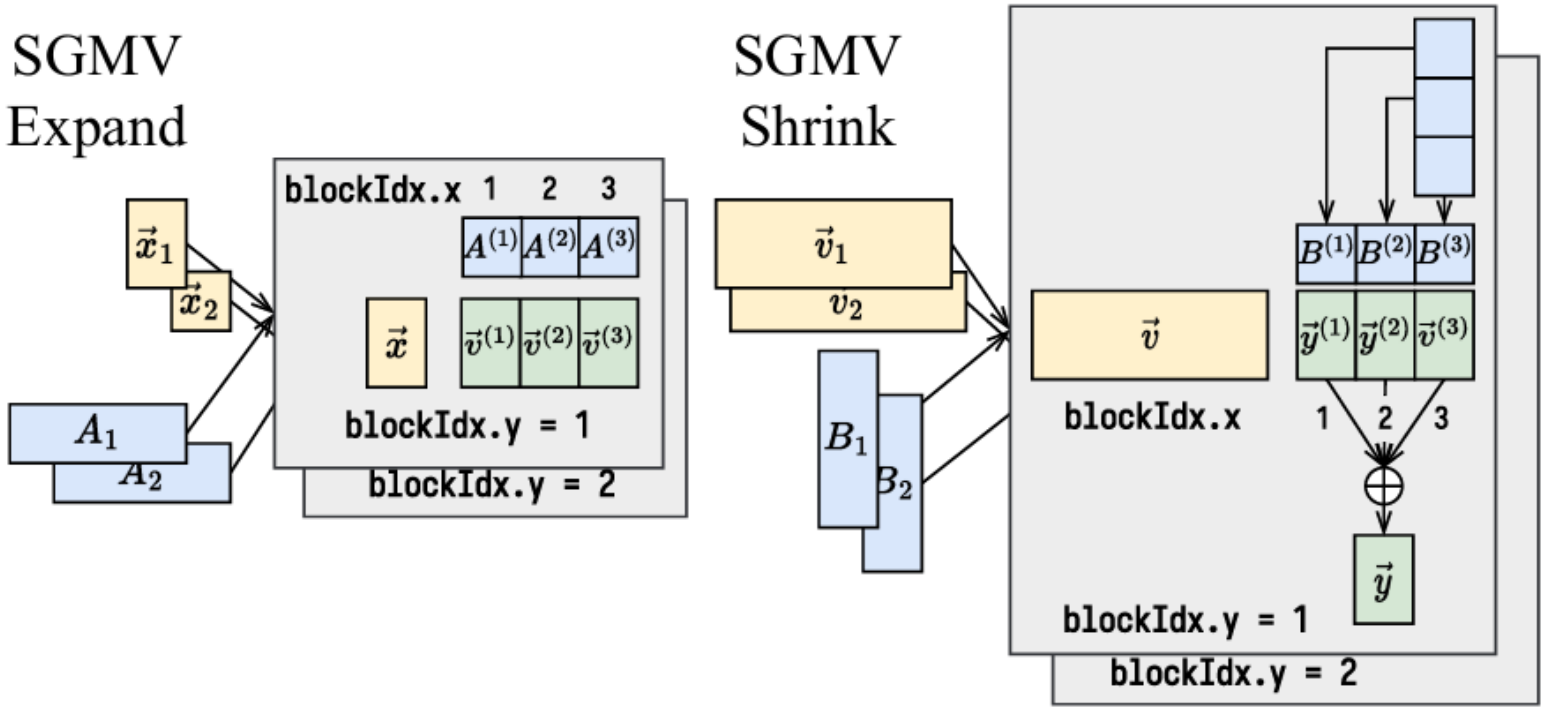
Design

- Q1: 如何在GPU上高效运行多个LoRA模型
  - Segmented Gather Matrix-Vector Multiplication (SGMV)
    - $xW' = x(W + AB) = xW + xAB$



Design

- Q1: 如何在GPU上高效运行多个LoRA模型
  - Segmented Gather Matrix-Vector Multiplication (SGMV)
    - $xW' = x(W + AB) = xW + xAB$



Design

- Q2: 如何依靠任务调度实现工作负载集中
  - 对用户请求进行批处理，以提高 Decode 环节 GPU 占用率
    - 以每个请求为单位进行调度
    - 每个批次中，Prefill数量限制为1
    - 当请求完成时，GPU将请求移除，并接受下一个请求
    - 最大批处理大小为32 (A100)

Design

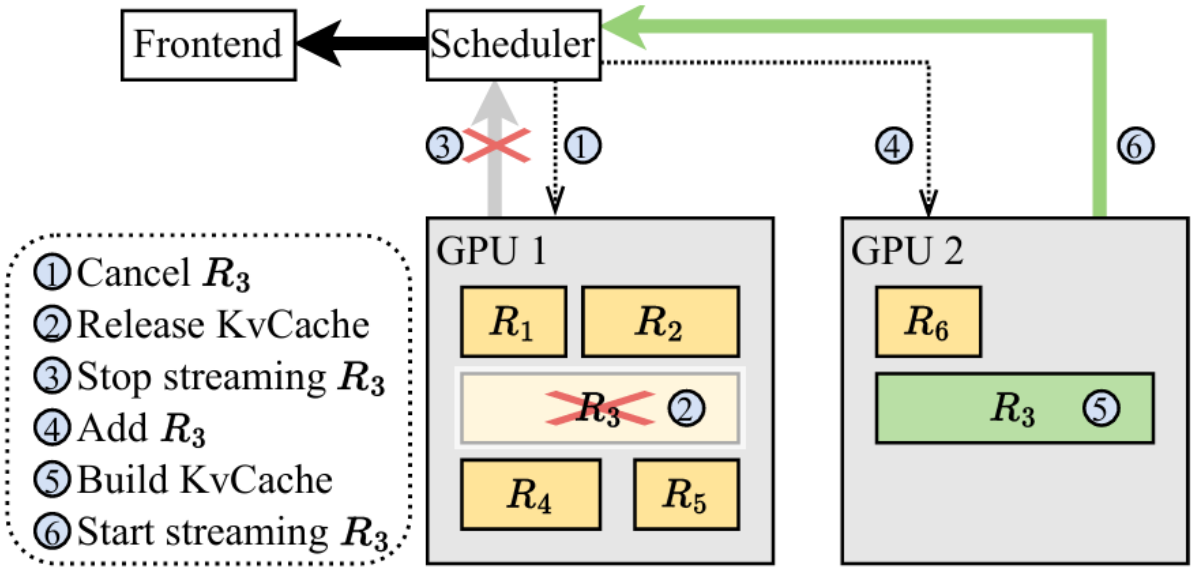
- Q2: 如何依靠任务调度实现工作负载集中
  - 实现LoRA模型按需加载，避免非活跃模型内存占用
    - 以PCIe GEN4 x16为例，加载一个模型时间只需要2ms
    - 当新的请求的LoRA权重尚未加载，可以在处理其他请求的同时异步内存复制来加载LoRA权重

Design

- 如何依靠任务调度实现工作负载集中
  - 将用户请求调度到活跃的GPU上，以占用最少的GPU资源
    - 调度器拥有全局信息，将新请求发往拥有最大批长度的GPU，且
      - 尚未到达最大批长度限制 (32)
      - 拥有**足够的内存**用于新请求的KVCache
    - 当存在多个GPU时，选取UUID最大的；当不存在时，排队，FCFS
    - 结果：忙的持续忙，闲的持续闲，负载轻的会逐渐空闲

Design

- 如何依靠任务调度实现工作负载集中
  - 请求迁移
    - token增加可能导致KVCache占据过多GPU内存
    - GPU内存不足时需要驱逐最新到达的请求，并添加到其他GPU中



Evaluation

实验设置

- TestBoard #1：一台配备 Nvidia A100 80G GPU的单卡服务器
  - GPU内存大，测试LoRA批处理效果
- TestBoard #2：两台服务器，每台8张Nvidia A100 40G GPU
  - 多机多卡，研究并行性和集群部署
- 使用 Llama-2:7B, 13B, 70B 参数模型，LoRA Rank为16

Evaluation

实验设置

LoRA模型请求分布

- Distinct: 每个请求对应单独一个LoRA模型
- Uniform: 所有LoRA模型使用频率相同
- Skewed: 使用频率遵循$Zipf-\alpha$分布，$\alpha = 1.5$
- Identical: 所有请求都是同一个LoRA模型

Evaluation

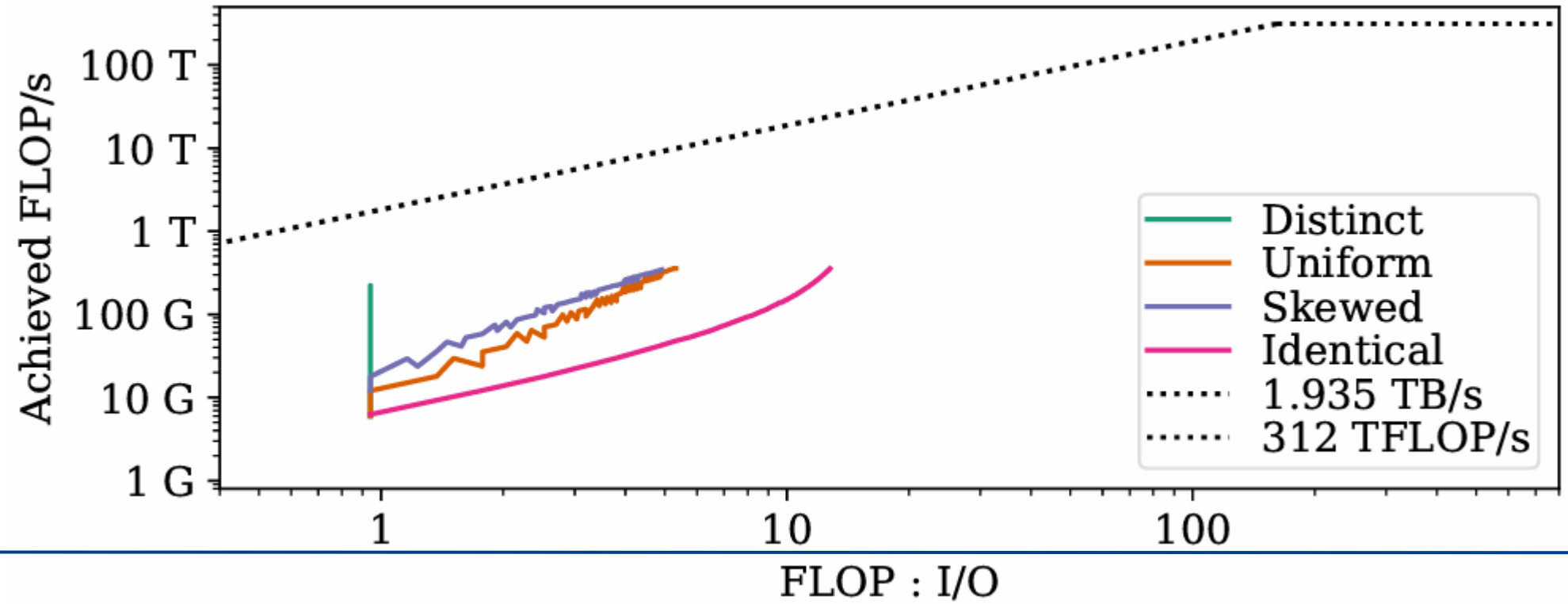
- 实验设置
  - Baselines
    - HuggingFace Transformers, EMNLP’20
    - DeepSpeed, SC’22
    - Faster Transformer (backbone-only)
    - vLLM (backbone-only), SOSP’23

Evaluation

- 模型请求分布 TestBoard #1

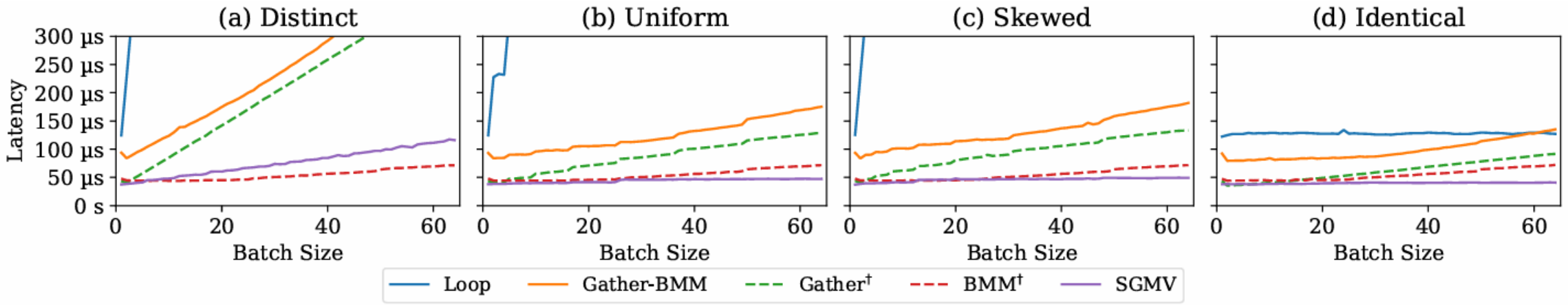
$$FLOP = s_n \times h_i \times h_o \times 2$$

$$I/O = [s_n \times (h_i + h_o) + n \times h_1 \times h_2] \times 2$$



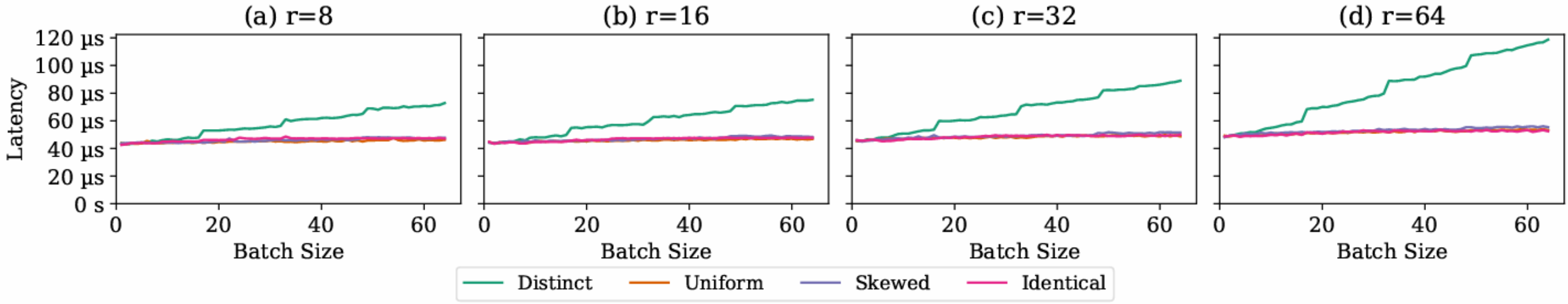
Evaluation

- SGMV 内核性能 TestBoard #1
  - Loop: for-loop: 循环，一个一个地处理这批数据
    - 效率极低，没有利用 GPU 大规模并行计算的能力
  - Gather-BMM: 内存复制收集LoRA权重，之后实现批处理矩阵乘法
    - 大量的内存读写操作，并且会出现重复读写



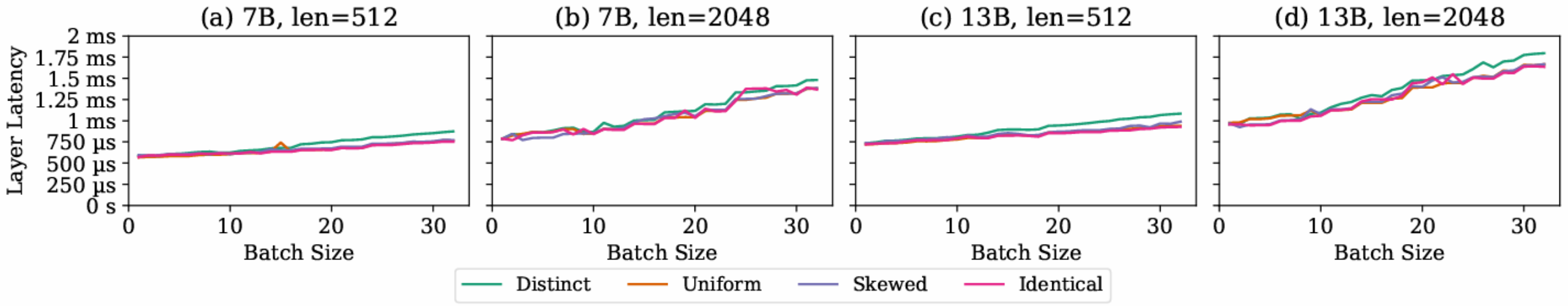
Evaluation

- 不同Rank的时延特征 TestBoard #1
  - LoRA Rank设定为 8，16，32，64
    - 存在权重共享时，时延几乎相同



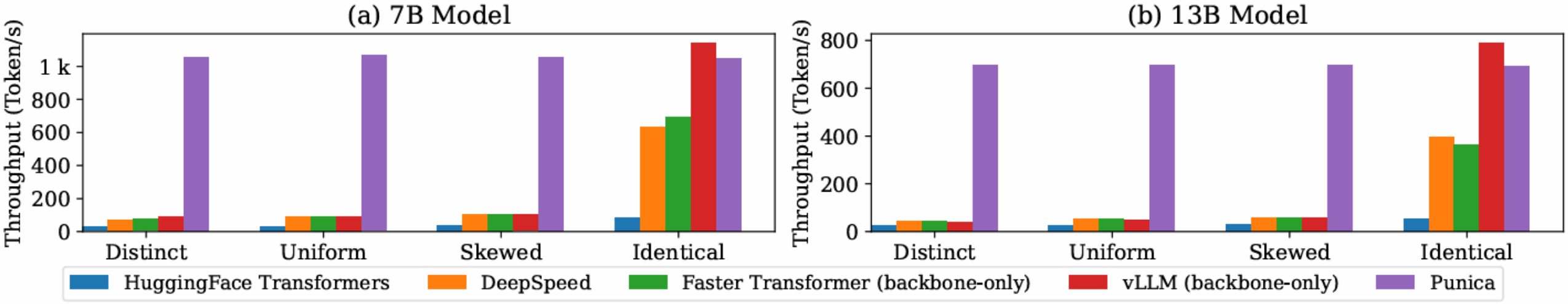
Evaluation

- 模型大小和序列长度影响 TestBoard #1
  - 模型参数量与序列长度分别为[7B, 13B]; [512, 2048]
    - 不同请求分布的延迟差异小
    - 序列长度短时，批处理效果更好，模型参数量大小影响小



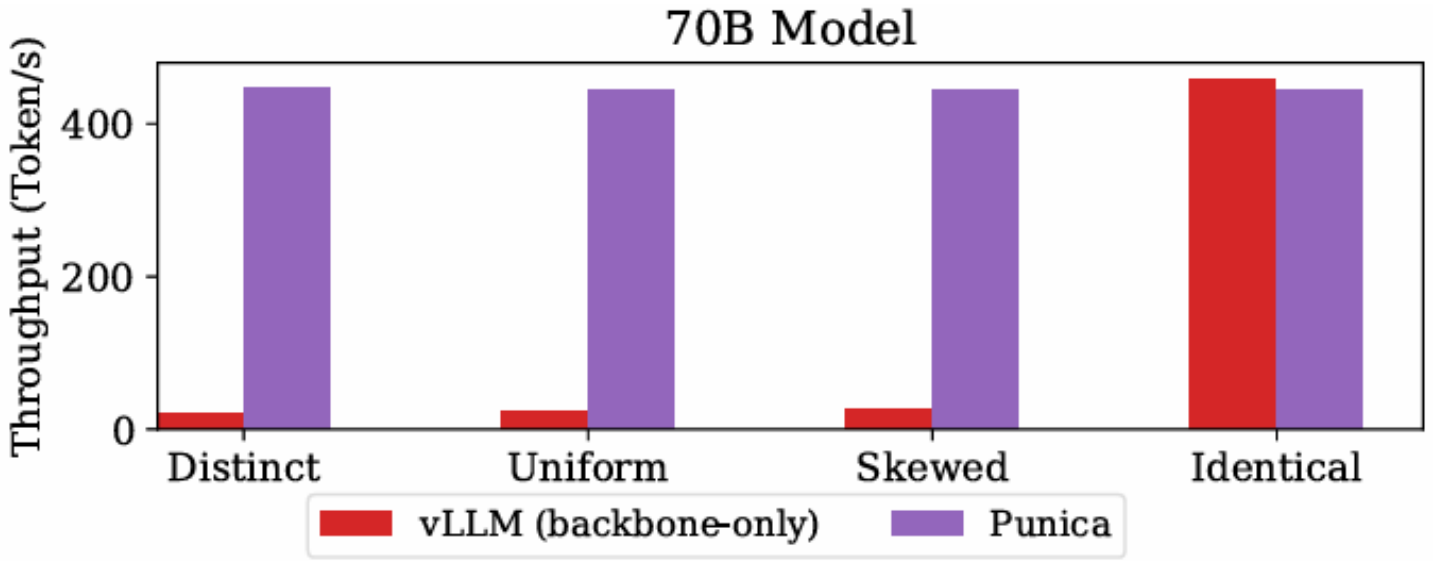
Evaluation

- Punica性能 TestBoard #1
  - 与Baselines对比，模型参数量为[7B, 13B]
  - 生成1000个请求，约101k个token
    - Punica在多LoRA中总是达到最佳性能



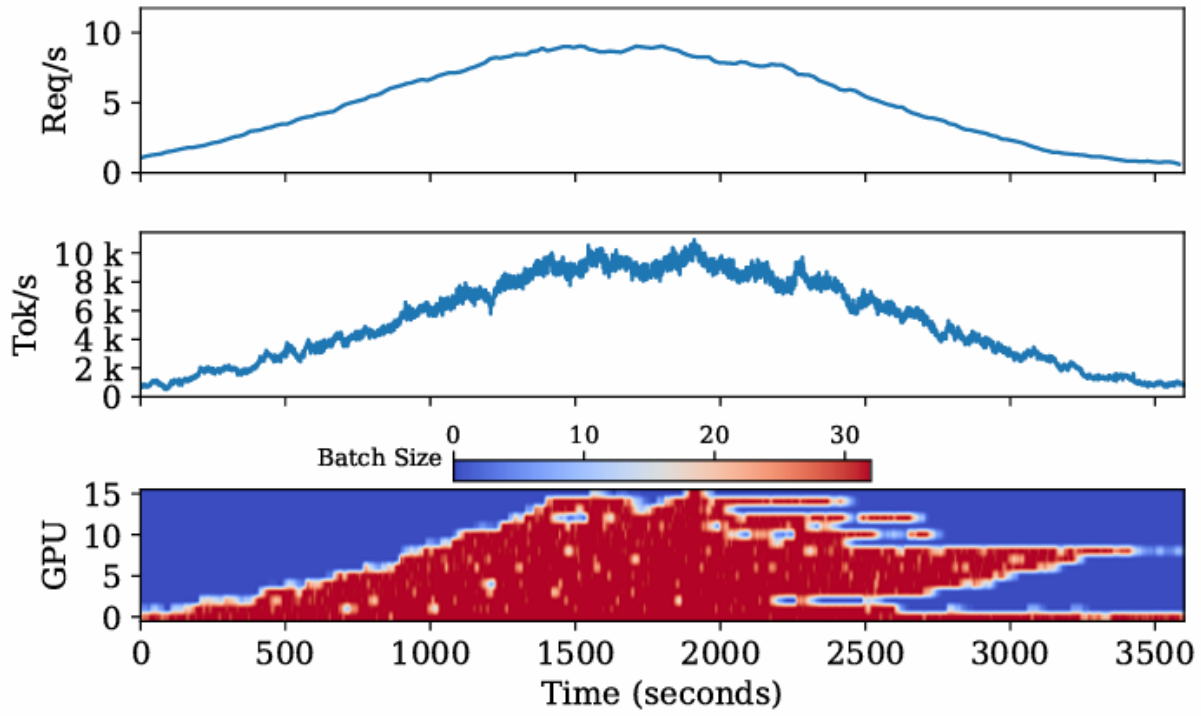
Evaluation

- Punica性能 TestBoard #2
  - 与vLLM对比，模型参数量为70B
    - Punica在多LoRA中总是达到最佳性能



Evaluation

- 工作负载变化 TestBoard #2
  - 模型参数量为7B
    - LoRA模型请求分布为Skewed



Conclusion

- Punica设计了一种新的CUDA内核 SGMV，对不同LoRA模型进行批处理
- 分离预训练模型与LoRA模型，减少内存占用，实现多LoRA模型载入
- 利用调度器整合GPU负载，提升单一GPU占用率，减少GPU占用数
- 实现了12倍不同LoRA模型服务吞吐量，每token仅增加2ms延迟(7%)

Thinking

- 能否泛化？
  - 个人用户 (设备内存小) 使用不同特化LoRA模型解决多维任务的能力提高了！不必考虑LoRA求和的问题了！
    - 比如我有十个实现特别任务的LoRA模型都基于同一个预训练模型，那么我只需要加载预训练模型和按需加载这十个LoRA模型即可
    - 单个LoRA模型相较于预训练模型内存消耗 < 1%，可以通过加载LoRA模型而不去考虑通过LoRA求和的方式将多个特化模型组合

Q & A