dLoRA: Dynamically Orchestrating Requests and Adapters for LoRA LLM Serving

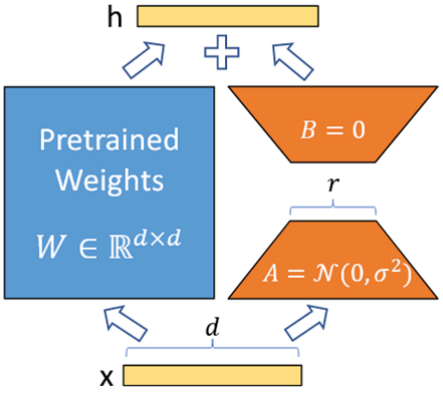
osdi24-wu-bingyang.pdf (usenix.org)

论文作者

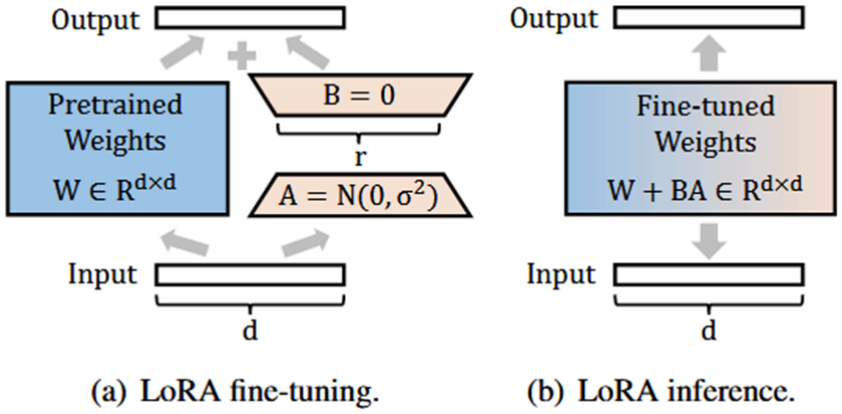
- Bingyang Wu, Ruidong Zhu, and Zili Zhang, School of Computer Science, Peking University;
- Peng Sun, Shanghai AI Lab;
- Xuanzhe Liu and Xin Jin, School of Computer Science, Peking University

LoRA

- 冻结预训练的模型权重，只对低秩矩阵A和B（适配器adapter）进行训练
- 有效降低内存占用，减少计算量，同时不降低模型精度
- 与使用全参数微调进行微调的 GPT-3 175B 相比，LoRA 可将可训练参数的数量减少 **10,000 倍**，GPU 内存需求减少 **3 倍。**



在进行推理时，将适配器权重BA和预训练的基础权重相加进行合并推理，这样做的好处是在推理时不产生额外开销。



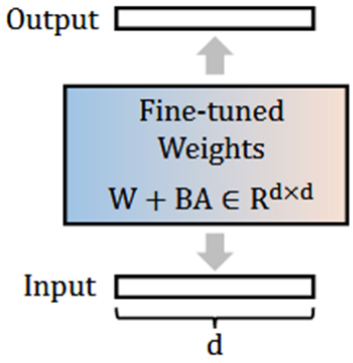
研究背景

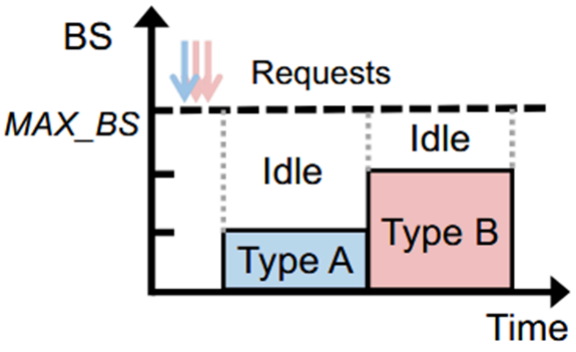
- 新的应用场景：base model+多lora adapter的LLM推理服务。

- 研究现状：
  - LLM serving systems (Orca [11] 、vLLM [12])：只为单个LLM loRA推理提供服务。
  - Traditional DNN serving systems (SHEPHERD [13]、 AlpaServe [14]、PetS [24]) : 不支持共享base model，不针对自回归 LLM，且无法提供LoRA。
- 当前研究主要存在问题：无法有效地为集群范围内的多个 LoRA LLM 提供服务。

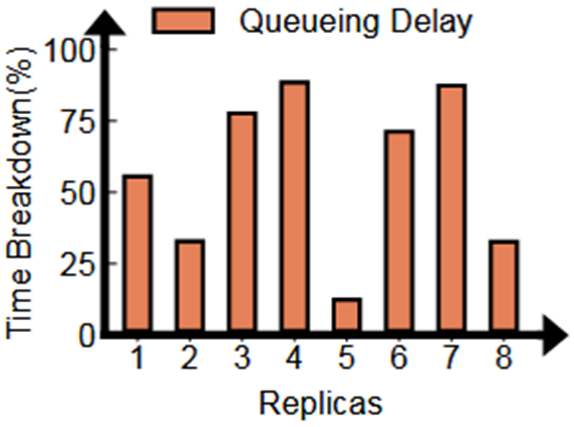
挑战

- 副本中GPU利用率不足
  - 采用合并推理：当多种不同请求到达时，仅支持具有相同 LoRA 适配器（即相同类型的请求）的批处理，其他类型的请求等待当前批次完成。
  - 示例中GPU资源利用率仅有50%，Type B请求的总延迟增加一倍。



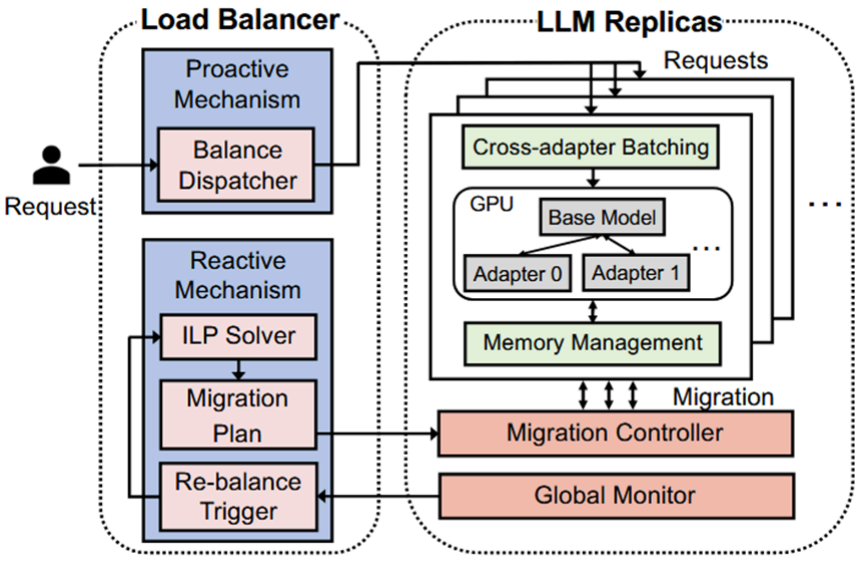


- 副本间负载不平衡
  - 可能存在某些适配器类型的请求数量突增，导致只有少数副本被利用，而其他副本处于空闲状态的不平衡情况。
  - LLM 任务具有可变的输入和输出长度，请求的可变输入和输出加剧了负载间不平衡。



架构设计

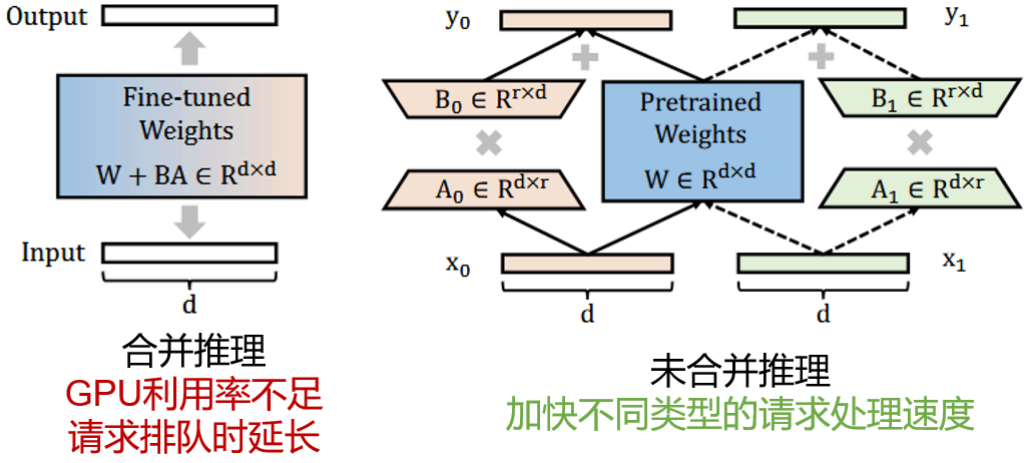
本文旨在解决当前研究现状中副本中GPU利用率不足、副本间负载不平衡两个问题，提出了dLoRA推理服务系统，可为集群中的多个 LoRA 模型提供服务。



- Intra-replica： Dynamic batching+ Memory management 解决了副本中GPU利用率不足的问题。
- Inter-replica： Proactive mechanism+ Reactive mechanism 解决了副本间负载不平衡的问题。

Intra-replica：Dynamic batching

引入未合并推理：将预训练的 LLM 权重和每个 LoRA 适配器权重的计算分开。在推理过程中，不同类型的请求共享预训练LLM 权重的计算，并行处理LoRA 适配器权重的计算。

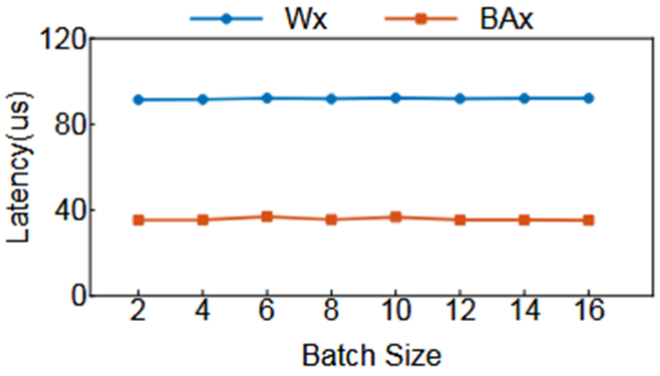


challenge1

然而引入额外的计算开销：

- 两个矩阵乘法（即适配器推理的单独计算）
- 一个矩阵加法

LoRA 适配器推理计算 BAx 的执行时间为预训练LLM权重计算Wx的38.9%，说明未合并的推理可能并不总是产生性能优势，当仅处理少数类型的请求时，性能可能会降低。（这是由于当请求的类型较少时，用合并推理能够实现充分的批处理，充分利用GPU资源，请求的排队时延也不会延长，这时候未合并推理用额外的计算开销去换排队时延是不值当的）。



解决方案：将合并推理和未合并推理相结合。两种批处理方法的组合可能会在排队延迟和计算开销之间提供更合理的权衡。

challenge2

确定最佳推理方式的时间以及切换带来的开销：

- 需要在迭代的粒度上确定推理方式
- 切换推理方式会带来不可忽略的开销

解决方案：Dynamic Batching算法

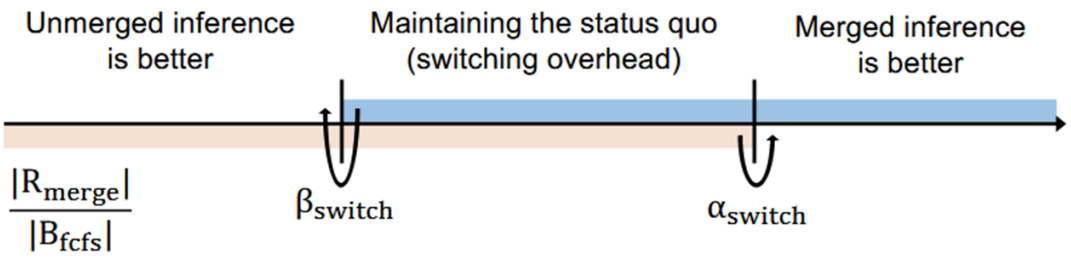
Algorithm 1 Dynamic Batching

function DYNAMICBATCHING(B_{fcfs}, R, S, L)
  Input: FCFS requests B_{fcfs}, Request $R = \{r_1, r_2, ..., r_n\}$
    Replica state S, LoRA adapters $L = \{l_1, l_2, ..., l_m\}$
  Output: The batch of requests to be executed B_{next}
  // Adaptive switching between different modes
  if $S.state ==$ unmerge **then**
    $R_{merge} = \arg\max_{l_i \in L} |\{r_i \in R \mid r_i.type == l_i\}|$
    if $|R_{merge}|/|B_{fcfs}| > \alpha_{switch}$ **then**
      $S.state$, $S.type =$ merge, $R_{merge}.type$
      return $B_{next} = R_{merge}[: max\_bs]$
    else
      return $B_{next} = B_{fcfs}$
  else
    $R_{merge} = \{r_i \in R \mid r_i.type == S.type\}|$
    if $|R_{merge}|/|B_{fcfs}| < \beta_{switch}$ **then**
      $S.state =$ unmerge
      return $B_{next} = B_{fcfs}$
    else
      return $B_{next} = R_{merge}[: max\_bs]$

- 设定两个阈值α*switch*和*β*switch
- Bfcfs表示fcfs的请求集合
- 当前状态为未合并时：|Rmerge|/|Bfcfs|>α_switch，切换为合并，此时Rmerge指的是当前请求中数量最多的类型的请求集合
- 当前状态为合并时：|Rmerge|/|Bfcfs|<β_switch，切换为未合并，此时Rmerge指的是符合当前处理的请求类型的请求集合
- 核心思想：

  相同类型请求多—>采用合并推理方式

  相同类型请求少—>采用未合并推理方式



自适应阈值调整算法

Algorithm 2 Adaptive Threshold Tuning

Input: Candidate period N_I, Merged batches $B_1, B_2, ..., B_I$, Switching overhead t_M, Current switching threshold α_{switch}
Output: New switching threshold α_{switch}
function ADAPTIVETUNING(N_I, $\{B_i\}$, t_M, α_{switch})
    $T_{merge} = \frac{\sum_{i=1}^{N_I} |B_i|}{\sum_{i=1}^{N_I} IterationTime(B_i) + t_M}$
    $T_{unmerge} = \frac{\sum_{i=1}^{N_I} |B'_i|}{\sum_{i=1}^{N_I} IterationTime(B'_i)}$
    if $T_{merge} > T_{unmerge}$ **then**
        $\alpha_{switch} = \alpha_{switch} - \gamma_{dec}$
    else
        $\alpha_{switch} = \alpha_{switch} \times \gamma_{mul}$
    return α_{switch}

保证两个阈值对不同请求特征的适应性

- 当前状态为unmerged：调α
    - 上一周期合并推理的吞吐量Tmerge较高—>α减少一个递减因子
    - 上一周期未合并推理的吞吐量Tunmerge较高—>α增加一个乘法因子
- 当前状态为merged：调β（同理）

challenge3

动态批处理倾向于处理负载最多的 LoRA 适配器类型请求

解决方案：

饥饿预防：使用credit-based方法，为每个 LoRA 适配器分配一个credit。当适配器被抢占，相应credit有所增加，当某些适配器的积分超过一定阈值，该算法会优先处理使用这些适配器的请求。

Inter-replica：Proactive mechanism

根据请求到达模式的特点：

- 长期：可预测性和周期性
- 短期：不可预测性和突发性

解决方案

- 适配器预取：优先选择突发容忍度低的适配器（表明该适配器在突发状况下更可能成为系统的瓶颈），加载到没有该适配器的副本
    - 突发容忍度=峰值容量 / 平均负载，

      LoRA 适配器的峰值容量定义为相应 LoRA 适配器所在的未分配 GPU 内存可以提供的请求数
- 估计等待时间：将请求分派到估计等待时间最短的副本
    - 等待时间包括加载 LoRA 适配器的时间（如果尚未加载）和此副本上的估计排队时间

Inter-replica：Reactive Migration

challenge：Proactive Mechanism无法有效解决可变请求长度导致的负载不均衡问题。

解决方案：Dynamic adapter-request co-migration 动态迁移适配器和请求

建模：将该迁移问题建模为ILP问题，求解得到适配器和请求的最佳放置

- 优化目标：最小化所有副本之间的总体运行时间。
- 约束：
  - 适配器和请求相匹配时才能迁移
  - 内存约束
  - 存在约束：每个请求只能放在一个副本

为了避免巨大的搜索空间带来的巨大开销，采用**具有约束放宽的选择性迁移**：

- dLoRA 仅在副本的可用 GPU 内存超过内存阈值或副本中请求的排队延迟超过计算阈值时触发迁移算法
- 只考虑前 K 个重载副本和前 K 个低载副本之间的迁移
- 将请求的迁移简化为token transfer，接着进行中间状态重建，比直接迁移所有的中间状态更快
- 将ILP问题中的变量矩阵放宽到实数范围加快求解速度，再从实数解中选取最大值对应的列来决定请求的分配位置

实验设置

Testbed：4×8 NVIDIA A800 80GB GPU，每个节点配备 128 个 CPU、2048 GB 主机内存和一个 200 Gbps InfiniBand NIC

Models：Llama-2（7B、13B 、70B），LoRA 适配器的等级设置为 8

Workloads：

Dataset：ShareGPT

Trace：Microsoft Azure 的无服务器函数追踪记录，2019 年（MAF1）和 2021 年（MAF2）

Baseline：Vllm、PEFT

Metrics：平均延迟

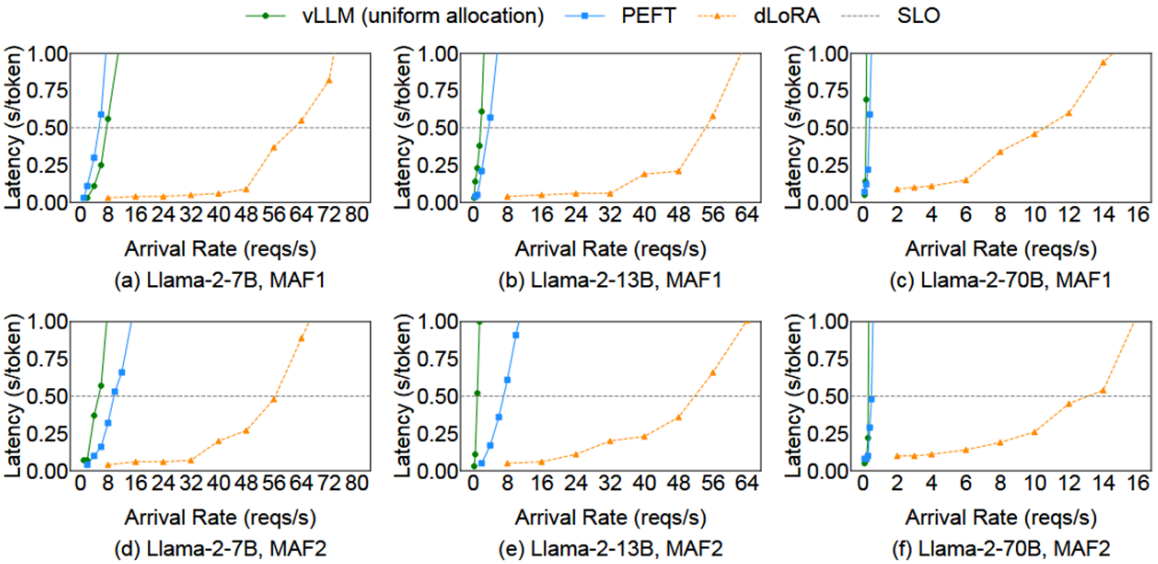
实验一：端到端性能评估

Llama-2-7B：单个GPU可以容纳整个模型，然而vllm和peft不能动态批处理

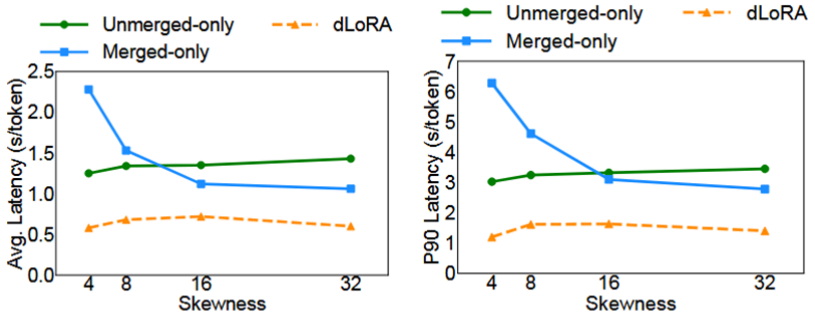
Llama-2-13B：单个GPU无法容纳整个模型

- Vllm将不同的lora llm视为单个llm，因此需要在GPU和主机之间交换模型参数
- PEFT在不同的lora llm之间共享模型基本权重，但没有请求适配器迁移

Llama-2-70B：dlora与现有的并行策略相结合，在4个GPU上并行



实验二：动态批处理的有效性



(a) Average Latency. (b) P90 Latency.

消融实验：将dLora与unmerged-only、merged-only推理方式进行对比

实验配置：单NVIDIA A800 80GB GPU ， 8×LoRA Llama-2-7B

实验结论：

- 相比merged-only ， dLoRA 将时延加快至 3.9×；相比unmerged-only，dLoRA将时延加快至2.4×
- merged-only： skewness低时，系统无法同时处理不同类型的请求排队时延长。
- unmerged-only： 平均时延基本保持不变，但当skewness比较高时，无法发挥合并推理的优势

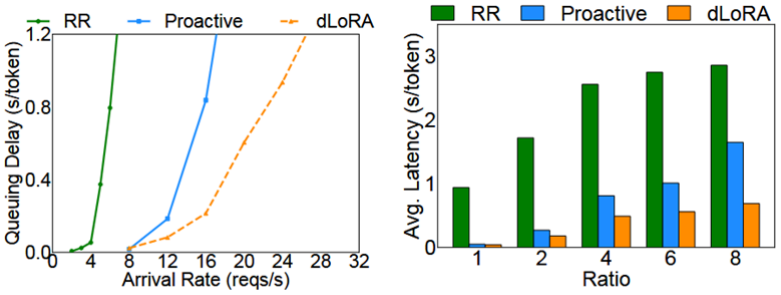
实验三：动态负载均衡的有效性

消融实验：与RR、 Proactive mechanism进行对比

实验配置：8 × NVIDIA A800 80GB GPUs

实验结论：

- RR 不考虑突发负载不平衡。
- 仅仅采用Proactive mechanism无法应对请求长度不平衡导致的负载不均衡问题。



(a) Reduction in Queuing Delay. (b) Stability under Different Ratios.

总结

dLora通过采用动态批处理，将合并推理和未合并推理相结合，以在排队时延和计算开销之间找到最好的平衡，提高GPU的利用率。

通过Proactive mechanism根据请求到达模式的长短期特点，采用适配器预取和将请求分配到等待时间最短的副本上，主动将LoRA 适配器加载到 GPU并将请求调度到副本，一定程度上缓解负载不均衡的问题 。

此外，Reactive Migrationt通过求解ILP问题，得到LoRA 适配器和请求的最佳放置计划，将 LoRA 适配器和请求从过载的副本迁移到其他副本，缓解由于可变请求长度导致的负载不均衡问题。