一、论文解读

1. 论文基本信息

- **标题**：DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving （DistServe：以实现吞吐量优化的大型语言模型服务的预填充和解码分解）
- **作者**：Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, Hao Zhang
- **期刊/会议** 18th USENIX Symposium on Operating Systems Design and Implementation [OSDI 24]
- **发表年份**：2024
- **DOI/链接**：10.48550/arXiv.2401.09670

2. 论文总结

- **研究背景**：
  - 现有的LLM服务系统通常将**预填充**和**解码**阶段集中在一起处理，这会导致资源分配和并行计划的耦合，以及预填充与解码之间的干扰，影响服务性能
  - LLM应用对延迟有严格要求，如预填充阶段的首次token时间（TTFT）和解码阶段的每个输出token时间（TPOT），现有系统难以同时优化这两个指标
- 大型语言模型（LLM）推理通常分为**预填充**和**解码（Decoding）**两个阶段。
- 现有系统（如vLLM、TensorRT-LLM）将两阶段共置于同一GPU，导致**预填充-解码干扰**，影响延迟目标（TTFT和TPOT）。
- **研究问题**：
  - 如何优化LLM推理的**有效吞吐量（Goodput）**（即满足SLO的请求率），避免两阶段干扰？
  - 如何解耦资源分配和并行策略，以适应不同计算特性？
- **研究方法**：
  - **分解架构**：将预填充和解码分配到不同GPU，消除干扰。
  - **定制并行策略**：为预填充（计算密集型）和解码（内存带宽受限）分别优化资源分配。
  - **KV缓存传输优化**：利用高速网络（如NVLink）最小化因解耦引起的通信开销。
- **研究结果**：
  - 相比现有系统，DistServe可提升**4.48倍请求率**或支持**10.2倍更严格的SLO**，同时保持90%请求在延迟限制内510。
- **贡献点**：
  1. 首次系统分析预填充-解码干扰问题，并提出分解架构。
  2. 设计**自动placement算法**，优化资源分配和并行策略。
  3. 实验验证在多种LLM和负载下均显著提升性能。
- **创新点**

  (1) 异构设备感知
  - **计算型GPU**（如A100）：优先分配给预填充组。
  - **高带宽GPU**（如H100）：优先分配给解码组。

(2) 细粒度流水线

- **预填充完成后的KV缓存** 直接迁移到解码组的GPU显存，避免重复计算。
- **零拷贝传输**：使用NVIDIA GPUDirect RDMA技术降低传输开销。

(3) 负载均衡

- **基于DAG的调度**：将请求依赖关系建模为有向无环图，避免死锁。
- **优先级队列**：解码请求优先调度（因实时性要求更高）。

3. 研究问题

问题阐述：

- **预填充-解码干扰**：
  - 预填充（计算密集型）和解码（内存带宽受限）在共置GPU时相互影响：
    - 预填充任务延长解码TPOT（需等待预填充完成）。
    - 解码任务增加预填充TTFT（抢占GPU资源）。
- **资源耦合**：
  - 两阶段共享并行策略（如张量并行TP vs. 流水线并行PP），无法分别优化。
  - 现有系统（如连续批处理）需过度配置资源以满足SLO，成本高。

4. 研究方法

研究设计：

- **分解架构**：
  - **预填充实例**：处理新请求，生成首个token和KV缓存。
  - **解码实例**：接收KV缓存，自回归生成后续token。

技术方法：

1. **延迟建模**：
   - 预填充延迟：$A + B * bs * l_in + C * \Sigma(l_in_i^2)$
   - 解码延迟：$A + B * bs * l_in + C * bs$（区分大/小batch size）。
2. **并行策略优化**：
   - 预填充：优先**张量并行（TP）**减少TTFT（计算加速）。
   - 解码：**流水线并行（PP）**提高吞吐量（线性扩展）。
3. **KV缓存传输优化**：
   - 利用高速网络（如PCIe 5.0）减少跨GPU通信开销。

5. 研究结果

实验平台：

- GPU：NVIDIA A100-80GB
- 模型：OPT-13B、OPT-66B、OPT-175B
- 负载：合成请求（符合泊松分布）。

对比方法：

- **基线系统**：vLLM、 DeepSpeed-MII（连续批处理）。
- **改进方案**：块预填充搭载（Chunked Prefill with Piggyback）。

实验结果：

- **吞吐量提升**：
  - 在13B模型上，DistServe（2P1D配置）达**3.3 reqs/GPU**，比基线（1.6 reqs/GPU）提升2.1倍。
- **SLO满足率**：
  - 在TTFT<0.4s、TPOT<0.04s约束下，90%请求达标，优于基线的50%10。

6. 研究总结与个人理解

研究总结：

- DistServe通过**分解架构**和**定制并行策略**，显著提升LLM推理的有效吞吐量。
- 该工作为LLM服务优化提供了新思路，已被工业界（如Mooncake、NVIDIA）关注。

个人理解：

- **优势**：
  - 分解复杂系统以解耦，通过提高每个部分的效率来提高系统总效率，符合并行思想
  - 分解架构简单有效，适合多GPU环境。
  - 延迟建模和并行策略优化具有普适性。
- **局限**：
  - 需至少2块GPU，资源受限场景不适用。
  - 跨机通信（如非NVLink环境）可能成为瓶颈。

评价与建议：

- **未来方向**：
  - 优化通信环境，当前需要NVLink环境来传递KV缓存，费用较为高昂，如果没有很好的通信环境，通信费用可能成为制约模型吞吐量的因素。

二、算法解释

1. Placement 算法的目标

- **解耦预填充与解码**：将两个阶段的任务调度到不同的GPU设备组，避免资源竞争。
- **最大化 Goodput**：优化系统吞吐量（每秒完成的请求数），同时满足SLA（如TTFT和TPOT延迟要求）。
- **动态适应负载**：根据实时请求流量调整资源分配（如突发大量预填充请求时自动扩容）。

---

2. 算法核心步骤

(1) 资源划分

- **预填充组（Prefill Group）**：专用GPU集群处理高并行、计算密集的预填充任务。
- **解码组（Decode Group）**：专用GPU集群处理低并行、内存密集的解码任务。
- **弹性资源池**：部分GPU可动态切换角色（通过轻量级上下文切换）。

(2) 成本建模

为每个请求类型建立资源消耗模型：

- **预填充成本**：`C_prefill = α × (输入长度)^2`（计算复杂度）
- **解码成本**：`C_decode = β × (输出长度)`（内存带宽受限）

(3) 动态调度策略

```
def auto_placement(requests):
    # 实时监控
    prefill_queue = get_prefill_queue_length()
    decode_queue = get_decode_queue_length()

    # 计算资源需求比
    ratio = (prefill_queue * C_prefill_avg) / (decode_queue * C_decode_avg)

    # 调整GPU分配
    if ratio > threshold_high:
        move_gpus_from_decode_to_prefill()
    elif ratio < threshold_low:
        move_gpus_from_prefill_to_decode()

    # 保证最小资源（避免饥饿）
    enforce_minimum_allocation()
```

---

3. 与现有方案的对比

特性	常规Pipeline	Orca	DistServe
预填充/解码解耦	❌	部分	✅
动态资源分配	❌	❌	✅

特性	常规Pipeline	Orca	DistServe
KV缓存迁移优化	❌	✅	✅（零拷贝）

---

4. 应用场景建议

- **高并发场景**：适合ChatGPT类服务，突发流量时自动扩展预填充资源。
- **长上下文场景**：预填充组专用大显存GPU处理长输入（如32K tokens）。
- **混合负载**：同时处理交互式请求（低TTFT）和批处理任务（高吞吐）。

总结：
DistServe 的自动 Placement 算法通过 **解耦+动态调度**，解决了传统LLM服务中预填充与解码的资源竞争问题。其核心是通过 **实时监控+成本模型** 实现资源的最优分配，属于系统级创新的典范。这一设计对构建高性能LLM推理服务（如大规模对话系统）具有重要参考价值。

三、重点名词解释

1. TTFT (Time To First Token)

- **定义**：从用户发送请求到收到LLM生成的**第一个token**的时间延迟。
- **重要性**：直接影响用户体验的"响应速度感知"，尤其是交互式应用（如聊天机器人）。
- **影响因素**：
  - 预填充阶段的计算效率（需完整处理输入序列）。
  - 批处理中其他任务的资源竞争。
- **论文关联**：DistServe通过**预填充专用GPU**和**张量并行优化**显著降低TTFT。

---

2. TPOT (Time Per Output Token)

- **定义**：生成**每个后续token**的平均时间（即解码阶段的单步延迟）。
- **重要性**：决定输出流畅度，影响长文本生成的用户体验。
- **影响因素**：
  - 解码阶段的内存带宽限制（需频繁访问KV缓存）。
  - 自回归过程的串行特性。
- **论文关联**：DistServe通过**解码专用GPU**和**流水线并行**优化TPOT。

---

3. SLO (Service Level Objective)

- **定义**：服务承诺的**可量化性能目标**，例如：
  - "90%请求的TTFT < 0.5秒"
  - "TPOT < 0.05秒/token"
- **作用**：衡量系统是否满足服务质量要求，是**Goodput优化**的核心指标。
- **论文关联**：DistServe的目标是**在满足SLO的前提下最大化请求吞吐量**（Goodput）。

---

4. Goodput (有效吞吐量)

- **定义**：**符合SLO的请求处理速率**（单位：reqs/sec），区别于单纯计算吞吐量（Throughput）。
- **关键点**：
  - 仅统计满足延迟约束的请求。
  - 体现系统实际可用性能。
- **论文关联**：传统系统因预填充-解码干扰导致Goodput低下，DistServe通过解耦提升4.48倍。

---

5. KV Cache (Key-Value缓存)

- **定义**：推理过程中存储的**历史token的Key-Value矩阵**，用于避免重复计算（Attention机制依赖）。
- **挑战**：
  - 解码阶段需频繁访问，导致内存带宽瓶颈。
  - 预填充阶段生成后需传输给解码实例。
- **论文关联**：DistServe优化了**跨GPU的KV缓存传输效率**（利用NVLink）。

---

6. Prefill (预填充) vs Decoding (解码)

特性	Prefill阶段	Decoding阶段
计算类型	计算密集型（全序列并行处理）	内存带宽受限（逐token自回归）
延迟敏感度	影响TTFT（用户等待首响应）	影响TPOT（输出流畅度）
优化策略	张量并行（TP）加速计算	流水线并行（PP）提高吞吐量

---

7. 张量并行 (TP) vs 流水线并行 (PP)

- **张量并行（Tensor Parallelism, TP）**：
  - **原理**：将模型参数**横向拆分**到多个GPU，单层计算由多卡协同完成。
  - **适合场景**：预填充阶段（计算密集型，需降低单请求延迟）。
- **流水线并行（Pipeline Parallelism, PP）**：
  - **原理**：将模型**按层拆分**到不同GPU，请求流水线式处理。
  - **适合场景**：解码阶段（提高吞吐量，支持更多并发请求）。

其他术语

- **Chunked Prefill**：将长输入序列分块处理，减少TTFT波动。
- **Piggyback**：在解码传输中捎带预填充结果，降低通信开销。