HeterPS: Distributed deep learning with reinforcement learning based scheduling in heterogeneous environments

研究背景

深度神经网络（DNNs）具有多层和大量参数，从而实现卓越的性能。然而，其训练过程需要处理大规模稀疏输入数据，因而面临高输入/输出（I/O）成本和计算密集型挑战。为了加速训练过程，通常采用分布式计算资源。然而，在异构计算资源（如多类型的CPU、GPU等）下，如何高效调度多层任务到不同的计算资源上，成为分布式训练中的关键问题。

为此，这篇文章提出了一个分布式框架——**异构参数服务器（HeterPS）**，其核心由分布式架构和基于强化学习（RL）的调度方法组成。HeterPS具有以下三大优势：

1. **高效处理异构资源下多样化负载的训练任务**；
2. **采用基于强化学习的方法高效调度各层的任务，降低成本同时满足吞吐量要求**；
3. **管理分布式计算资源中的数据存储和通信。**

研究挑战

1. 随着计算单元（CPU、GPU等）种类的多样化，充分利用异构计算资源变得关键。
2. 不同的计算资源对任务适配性不同：CPU适合数据密集型任务，而GPU适合计算密集型任务。
3. 任务调度是分布式训练中的核心问题，但这是一个典型的NP-hard问题，现有方法（如遗传算法、贪心算法、贝叶斯优化等）有一定效果，但可能存在高成本或次优解。

研究想法

1. 数据并行（Data Parallelism）：通过分割数据集，让所有计算资源同时处理不同的数据块。
2. 管道并行（Pipeline Parallelism）：将DNN模型按层分割，每个计算资源负责一段连续的层。
3. 数据并行与管道并行可结合使用，实现更细粒度的并行化以提高效率。

实践困难：**尽管并行化减少了训练时间，但可能会显著增加成本。优化训练需要在吞吐量和成本之间找到平衡。**

研究贡献

- **框架创新**：提出了一个用于弹性异构计算资源的分布式训练框架，管理分布式资源中的数据存储和通信。
- **RL 调度方法**：设计了一种基于强化学习（RL）的调度方法，利用 LSTM 模型为每一层选择合适的资源，优化总成本并保证吞吐量。同时提出了一种方法确定分布式训练所需的资源数量。
- **实验验证**：在多种结构的 DNN 模型上进行了广泛实验，验证了 HeterPS 的优越性，与基线方法相比表现更好。

系统架构

设施架构

- **训练数据集群**：存储训练数据，例如使用 HDFS 集群。
- **协调器（Coordinator）**：连接每个工作节点和训练数据集群，负责集中式控制。
- **计算资源**：
  - **CPU 工作节点**：适合处理数据密集型任务。
  - **GPU/XPU 工作节点**：适合处理计算密集型任务。XPU 包括适用于 DNN 训练的多种优化处理器（如 Kunlun AI 处理器）。
- **参数服务器**：管理模型参数。

问题建模

成本模型

成本模型，用于估算分布式训练过程中**吞吐量**和**货币成本**，并以此为基础定义调度计划和资源分配计划。

核心假设

1. 模型分区：将一个 DNN 模型划分为K个阶段 $(S_1, S_2, \ldots, S_K)$，每个阶段包含若干连续的层，且这些层被调度到相同类型的计算资源上。每个阶段的计算资源在其内部是同构的，但不同阶段之间是不同构的。
2. 批量大小与训练过程：假设训练集有M个样本，批量大小为B，则每个 epoch 包含M / B个批次。

计算公式

对每个阶段 S_i，在分配 k_i个计算资源后，单次迭代的计算时间（ CT_i）和数据通信时间 (DT_i)为

$$CT_i = \frac{OCT_i}{B_0} \times \left(1 - \alpha_i + \frac{\alpha_i}{k_i}\right),$$

$$DT_i = \frac{ODT_i}{B_0} \times \left(1 - \beta_i + \frac{\beta_i}{k_i}\right),$$

- α_i 和 β_i：分别表示计算和通信任务中可以并行化的部分。
- $1 - \alpha_i$ 和 $1 - \beta_i$：表示计算和通信中不可并行化的部分（如同步开销、参数服务器负载等）。
- k_i：分配到阶段 S_i 的计算资源数量。

上述公式基于 **Amdahl 定律** 推导，反映了并行化和资源分配对计算和通信时间的影响。

计算阶段的执行时间可以通过下式计算

$$ET_i = max(CT_i, DT_i)$$

即，每个阶段的执行时间由计算时间（ CT_i）和数据通信时间 (DT_i)中较大的一个决定；

每个阶段的吞吐量定义为

$$Throughput_i = B/(ET_i)$$

其中，B为批量大小，表示每次迭代处理的样本数；

当训练过程的epoch数为 L 时，整个训练过程的**总执行时间**为

$$ET = L \times M / Throughput$$

货币成本：用于衡量执行整个训练任务的花费，定义如下：

$$Cost = ET \times \sum_{t=1}^{T} p_t \times k_t$$

- p_t：计算资源类型 t 的单位价格，例如每分钟使用 V100 GPU 的成本。
- k_t：计算资源类型 t 的数量。
- T：计算资源的种类总数。

问题构建

目标：满足吞吐量约束的同时最小化货币成本；

决策变量：

定义一个调度矩阵，用来表示层和计算资源类型之间的分配关系

$$Schedule(l,t) = \begin{cases} 1, & \text{第 } l \text{ 层分配到类型 } t \text{ 的计算资源上} \\ 0, & \text{否则} \end{cases}$$

- 每一层只能分配给一种类型的计算资源。
- 分配到同一类型计算资源的连续层构成一个阶段（stage）。

该问题可以形式化为以下优化问题

$$\min_{SP \in S} \text{Cost}(SP)$$

- 目标：最小化成本 $Cost(SP)$

约束条件

1. 吞吐量约束

$$Throughput(SP) >= Throughput_{limit}$$

- $Throughput_{limit}$：系统规定的最低吞吐量；

2. 资源限制

$$N_t(SP) <= N_{t,limit}$$

- $N_t(SP)$：调度中分配给类型t的资源数量；
- $N_{t,limit}$：类型t的资源数量上限；

基于强化学习的调度

目标：

通过为每个阶段分配适当数量的计算资源，使多个阶段的执行负载达到平衡。负载均衡以每个阶段的吞吐量（计算和数据通信的组合）为依据，尽量使所有阶段的吞吐量相等，从而避免因某个阶段成为瓶颈而影响整个系统性能。

执行时间与批量大小的关系

- 当批量大小较小时，阶段的执行时间主要受 **数据通信时间** 控制，称为通信密集型阶段。
- 当批量大小较大时，阶段的执行时间主要受 **计算时间** 控制，称为计算密集型阶段。
- 在分布式训练中，不同阶段使用数据并行和流水线并行来实现资源利用。

平衡吞吐量的推导

假设所有阶段 S_i 的吞吐量相等：

$$Throughput_i = Throughput_1$$

$$k_i = \frac{\alpha_i \cdot OCT_1}{OCT_i \cdot (1 - \alpha_i + \frac{\alpha_1}{k_1})} \cdot (1 - \alpha_1).$$

初始资源分配的约束条件

$$k_1 > \min\left\{\frac{\alpha_1 \cdot OCT_1}{Throughput_{limit} \cdot B_0 - (1 - \alpha_1) \cdot OCT_1}, \frac{\beta_1 \cdot ODT_1}{Throughput_{limit} \cdot B_0 - (1 - \beta_1) \cdot ODT_1}\right\}$$

强化学习设计

- 模型设计：
  - 构建一个基于 LSTM 的强化学习模型，每个 LSTM 单元（cell）的长度与 DNN 层数相同。
  - 每个单元的输入是一个五维特征向量，表示每一层的特性：
    1. 层的索引（one-hot 编码）。
    2. 层的类型（如全连接层、嵌入层等，one-hot 编码）。
    3. 输入数据的大小（浮点数）。
    4. 权重的大小（浮点数）。
    5. 数据通信时间（浮点数）。
  - LSTM 输出一个维度为T 的向量，其中 T 为计算资源类型的数量。
- 输出与决策：
  - 将 LSTM 的输出通过 softmax 函数，得到每一层分配到每种计算资源的概率分布。
  - 根据最高概率值选择每一层的计算资源类型，从而生成调度计划。
- 训练过程：
  - 奖励函数与训练成本直接关联
  - 通过REINFORCE算法进行优化；
  - 两阶段训练：
    - 预训练阶段：随机生成调度计划，计算其成本，作为初始奖励进行训练；
    - 分布式训练：基于更新后的 LSTM 模型生成调度计划，计算真实吞吐量和成本，再用作奖励更新模型。

实验设置

- HeterPS实现：基于C++和Python编写；
- **硬件配置**：
  - **CPU服务器**：
    - 每台服务器配备 2 颗 Intel Gold 6271C CPU，每颗 CPU 有 24 个物理核心。
    - 每台服务器配有两块 NVMe SSD，用于支持数据加载。

- **GPU服务器**：
  - 配置与 CPU 服务器相同，但额外配备 **8 块 Nvidia Tesla V100 GPU**。
  - GPU 服务器和 CPU 服务器均配备 512 GB 内存。
- **网络**：
  - 所有服务器通过 InfiniBand NIC 连接，带宽为 100 Gbps。

- **集群规模**：
  - 默认配置为 **10 台 CPU 服务器和 4 台 GPU 服务器**。
  - 可以根据需要扩展更多服务器。
- **成本设置**：
  - CPU 的费用为 **0.04 美元/小时**。
  - GPU 的费用为 **2.42 美元/小时**。