基于 P2P 的分布式推断系统

于松琦 11910910
计算机系
计算机科学与工程
指导教师：唐茗

May 13, 2024

摘要

P2P 网络具有优秀的可扩展性，非中心化，隐私性与高性价比等特点，这些特点使其具有分布式训练神经网络的能力，本文主要研究的是如何将 *P2P* 网络与神经网络训练相结合，搭建一个基于 *P2P* 的分布式训练推断系统，为此本文使用了 *ns3*，*pytorch,docker* 等工具进行实验，将 *docker* 与 *ns3* 联系起来搭建一个 *P2P* 网络，并在其中设置了几个包含 *GPU* 的特殊节点作为高算力节点使用。同时设置节点选择算法对节点进行选取。从实验结果可以看出，分布式训练模式却是可以收集 *p2p* 网络的冗余算力进行计算，这十分依赖网络本身设备质量以及节点选择算法。

摘要

P2P networks possess excellent scalability, decentralization, privacy, and cost-effectiveness, making them capable of distributed neural network training. This paper primarily investigates how to integrate P2P networks with neural network training to establish a distributed training and inference system based on P2P. For this purpose, tools such as ns3, PyTorch, and Docker were employed in experiments to connect Docker with ns3 to build a P2P network. Several special nodes equipped with GPUs were set up as high-computational nodes within the network. A node selection algorithm was also implemented for choosing nodes. The experimental results demonstrate that the distributed training mode can indeed utilize the redundant computational power of the P2P network, which heavily depends on the quality of the network devices and the node selection algorithm.

Contents

1 引言

目前，随着 AI 大模型的规模愈发庞大，模型对算力的要求也逐步升高。但考虑到目前 GPU 价格不低，而互联网上存在算力冗余的情况，如果能收集互联网的一部分冗余算力用于模型训练，或许可以降低一部分模型的训练成本。而 P2P 网络一直是一个具有优秀延展性，较高冗余算力，与较高隐私性的网络，而神经网络的分布式训练恰好对这几个特点有较高要求。本文主要探究基于 P2P 网络的神经网络分布式推断系统的特点。首先搭建一个基于 P2P 的分布式推断系统，系统中有很多节点（终端设备或者边缘服务器），由选定节点发送最初的训练模型。任意节点会缓存一个或多个模型，当一个有训练需求或推断需求的节点加入系统，向服务器注册，同时获取其他节点的信息。与其他节点握手，获得实际的每个节点的模型信息，可以选择传数据到其他节点进行推断，或将模型拆分并分散到这个系统上各个节点进行计算，或者直接接受模型自己进行推断，需要根据节点活跃情况，节点算力情况对每个节点进行选择，各个节点返回计算结果后将模型聚合，使用多个模型进行测试。将结果与单节点训练相比较。

2 相关工作

目前，分布式训练主要基于四个并行，数据并行，模型并行，流水线并行以及混合并行，其拥有多个框架 [1]DeepSpeed,Megatron-LM。2019 年 nividia 曾对

使用基于模型并行和数据并行的方法对多 GPU 集群训练模型的情况进行了实验 [2]，其实验结果如下：

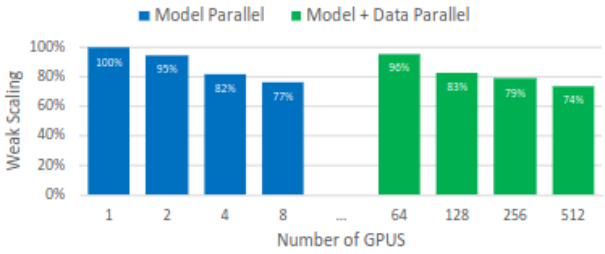

Figure 1: Model and model + data parallel weak scaling efficiency as a function of the number of GPUs

可以看出，随着 GPU 数量增加，使用 8 卡的模型并行方法中单个 GPU 训练效率在下降。本文采用了与其类似训练原理的 pytorch 的 DDP 训练模组来进行分布式训练，使用 docker 进行容器物理分割，并使用 ns3 来搭建 p2p 网络系统。从而在 P2P 网络上进行分布式训练。而 P2P 的节点选择算法则是被研究很多年了，有信任度算法 [3]，粒子群算法 [4]，蚁群算法 [5] 等多种算法进行节点选择。考虑到本文的模拟网络没有安全需求，故对其中的恶意节点概念进行了排除。

3 实验工具介绍

3.1 ns3

NS3 是一款离散事件驱动的网络仿真器，主要应用于研究和教育领域，旨在满足学术和教学的需求。NS3 项目是一个完全开源的开发工程，于 2006 年启动。NS3 对 linux 系统具有很好的支持。NS3 支持 C++ 和 PYTHON 的开发，本文使用 C++ 作为主要语言，因为 NS3 对 C++ 的支持比 python 要强很多，python 部分有很多功能不能正常使用

3.2 docker

Docker 是一种开源的容器化平台，它允许开发者将应用及其依赖环境打包在轻量级、可移植的容器中。容器在运行时，共享宿主机的操作系统内核，但在用户空间内部彼此隔离，保证了运行环境的一致性和安全性。Docker 容器的启动速度快，资源占用低，使得它成为开发和运维领域的首选技术。在分布式训练场景中，训练任务需要在多个计算节点上协同执行。这些节点可能分布在不同的地理位置，具有不同的操作系统和硬件配置。Docker 通过容器这一概念，成为单机模拟分布式学习不可或缺的存在。Docker 容器确保所有节点上的训练环境完全一致，包括操作系统、库版本及其他依赖，从而避免了因环境差异带来的兼容性问题。同时，Docker 容器可以在几秒钟内启动，相比于虚拟机，docker 轻量性的特点使其十分适合模拟分布式学习。Docker 提供了有效的资源隔离机

制，允许不同的容器在同一物理机上高效共存，同时保证各个容器所需资源的合理分配，这一点对分布式学习尤为重要。

3.3 ubuntu 操作系统

Ubuntu 是一个免费开源的基于 Debian 的开源操作系统，其构建在 Linux 内核之上。提供了直观的 GUI 界面，其配置，使用都较为容易，因此被广泛使用，从而形成了一个庞大的社区。

3.4 pytorch

PyTorch 是一个开源的机器学习库，广泛应用于计算机视觉、自然语言处理等人工智能领域。它由 Facebook 的人工智能研究团队（FAIR）首次发布于 2016 年，如今已成为学术界和工业界广泛使用的深度学习框架之一。

4 项目主体

4.1 搭建 P2P 网络

NS3 是一款离散事件驱动的网络仿真器，适用于 linux 系统，被广泛应用于网络模拟方面。本次实验选用 ns3 作为搭建工具，其有一个内置的可视化工具叫 Pyviz, 但它不能自定义节点位置，对于节点间信息传输也没有很好的表示。所以我最后选择 NetAnim 作为可视化组件。

具体模拟过程为，首先设计了一个 20 个节点，基于 csma 的网络。Ns3 主要有三种网络结构，分别是 peerTopeer,csma,wifi，其中 peerTopeer 的节点通信模式只支持两个预设的节点之间通信，并为这两个节点搭建信道，如果使用 peerTopeer 作为基础网络结构，则网络复杂度会随着节点数量增加而急速上升，不利于大型网络模拟。而 csma 则是将所有节点放在同一信道下，网络结构十分简单，原理上也更符合 P2P 网络定义，故选择 csma。接着创建网络节点和安装网卡设备。考虑到这个网络需要动态调整节点数量来模拟 P2P 网络中节点进出的场景，故选择对几个特定的节点单独安装网卡设备，用开启关闭设备的方法来模拟网络运行中节点退出与进入的过程。同时考虑到 gpu 数量有限，故只设定几个特定的节点拥有计算能力并配置 GPU。

docker 容器间通信可以通过 veth 来进行，而 ns3 存在一个 TapBridge 模块，这一模块是一个模拟的网络设备，安装了这一模块的节点可以从物理主机中接收分组，然后通过 Tun/Tap 等虚拟网络设备将分组发送到 NS3 中，从而实现 docker 容器和 ns3 虚拟网络的通信。这几个 ns3 节点就使用 docker 构建容器，同时为每个容器指定使用的 gpu 并使用网桥实现 docker 与 ns3 里特定节点之间通信。最后安装协议栈，安装应用层协议，并将模拟事件压入协议栈以执行网络模拟。

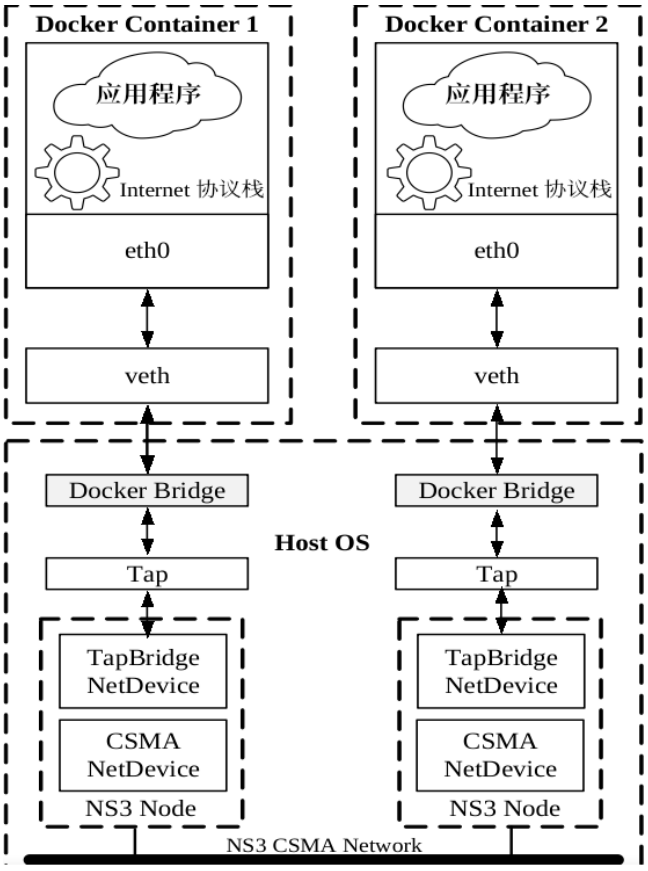

Figure 2: 连接原理

必须提到的一点是，DDP 的工作原理导致 DDP 所控制的每个分布式训练进程必须控制至少一个 GPU，如果要扩大网络规模，增加大量高算力的节点的话，GPU 数量不足的问题会导致 P2P 网络规模会受到极大影响。故此处采用了就加大量只具有存储和信息交换能力，不具有计算能力的网络节点，以模拟现实中 P2P 网络节点鱼龙混杂的情况的办法，而有 GPU 算力的节点作为特殊节点，参与分布式训练，这套解决方案的优点是可以更进一步模拟现实中 P2P 网络的情况，但缺点是分布式训练的基础算力节点没有任何改变，无法对大规模分布式训练的情况进行模拟。

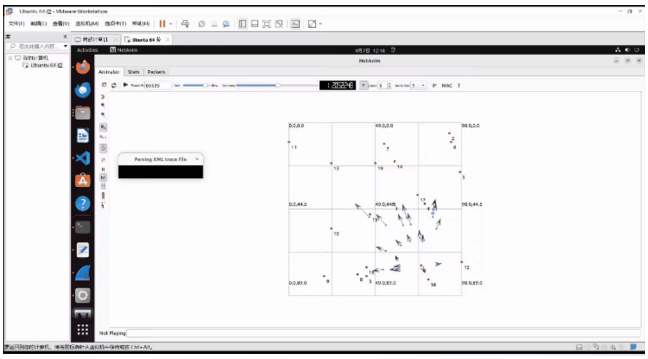

Figure 3: 已经搭好的网络

4.2 分布式训练构建

实验选用的是 pytorch 自带的 DPP 模块和 mutiprocessing 模块，以一起来实现神经网络的分布式训练，DPP 模块是 pytorch 官方推出的一个神经网络分布式训练模块，其基本原理是在每一 GPU 上加载一个进程，并使用 ring-reduce 模块来实现进程间相互通信，交换梯度。经过实际 i 测试，使用 NCCL 的集合通信库效率要比使用 gloo 高，但 gloo 支持 cpu 运行，nccl 不支持，本文并不打算使用 cpu 参与计算，所以选用 nccl 作为 i 集合信息库。同时，为了测试多个模型对于该网络的适应性，我选择了使用基于 minist 数据集和 cifar 数据集的两个不同模型，为了保证两个模型总训练时间可以基本相同，从而易于比对，也易于直观展示，我对两个模型训练过程的参数进行了调整。其中 minist 模型的 batchsize 为 32，epoch 为 3；cifar 模型 batchsize 为 4，epoch 为 1. 在不同数量的网络节点上进行了测试，每个网络节点都对应一个 GPU。模拟现实生活中具有冗余算力的节点，同时也匹配 DDP 在每个 GPU 上单独创建一个进程的要求。经过查询得知，目前分布式训练有很多现有的例子，如 VGG 模型和 googleNet 模型，两个模型并行组合情况卜 VGG 运行 100 次迭代大概需要 30 秒，单独运行 100 次迭代需要 25.13 秒，而 googlenet 则在组合情况下运行 100 次迭代需要 45 秒，单独运行需要 34.64 秒 [6]

4.3 节点选择机制

考虑到 P2P 网络中并不是所有节点都有冗余算力，因此对节点的选择就十分重要。理想的情况是在需要进行分布式推断时，能够找到算力最强的节点，将模型传输给他进行训练并接受训练好的模型，其次是在需要进行分布式训练时，能够找到较多拥有较强算力的节点，以在这些节点上进行分布式训练，在训练结束后接受返回结果。为了实现这个目的，有两个指标是不能忽略的，一个是节点稳定性，节点必须具有较强稳定性，不能频繁出现掉线的情况。一个是节点算力强度，由于需要进行分布式训练，节点本身必须要有一定的计算能力，否则无法参与训练。在阅读过其他人的论文成果后，我设计了一套基于信任度的节点选择机制。首先，由于这个网络目前只存在模型，数据集这两种主要资源

以及其他小型数据资源。所以信任度需要主要基于这两个部分进行计算。同时还需要考虑到存在稳定优秀的节点临时波动的可能，所以节点过往的信任度也需要纳入考量，这里我将模型传输成功次数，成功参与模型分布式训练的次数，与节点旧的信任度，传输延时作为四个指标，首先对一个节点进行打分。

$$Score_{new} = 0.3 * Score_{old} + 0.1 * Transmission + 0.5 * train - 0.1 * 10 * delay \quad (1)$$

利用这四个指标，可以首先对该节点作出初步评价，由于在测试过程中节点交互时延太低，故这里将其放大以纳入考量。在获得节点初步评分后，对节点存续时间进行考量，很明显，一个节点每次在网络中存在时间越长，这个节点的稳定性相对而言一定是越高的。因此，第二步的关键就是需要将节点以往稳定存在时间进行计算，理论上，在线时间越长，在线次数越多，该节点也越好。其中，在线时间与在线次数经常不属于一个数量级，一个节点一次可能在线几百几千秒，而这么长的在线时间需要与在线次数相绑定，为了尽量削减在线时间的影响，使用 ln() 函数来尽量减少其增长速率, 且每次在线时间单独计算，且该指标最多不能超过十。不能将所有在线时间统合计算。

$$\text{time}_{score} = \begin{cases} \sum ln(online_{last} + 1) & \sum ln(online_{last} + 1) <= 10 \\ 10 & \sum ln(online_{last} + 1) > 10 \end{cases} \quad (2)$$

上式中 $time_{score}$,$ln(online_{last}$ 分别代表在线时间考量因素和在线时间。第三步就是考虑节点稳定性的问题，如果一个节点的出现了中断的情况，那么这个节点信任度理应下降。同理，当节点收到请求时发现自己没有满足请求能力时，该节点信任度也需要下降

$$score_{reduce} = interruption + unfinished - time_{score} \quad (3)$$

综合起来，信任度公式可以表示为

$$belief = Score_{new} + time_{score} - score_{reduce} \quad (4)$$

4.4 实验结果

4.4.1 将 GPU 均分到 8 个节点进行实验

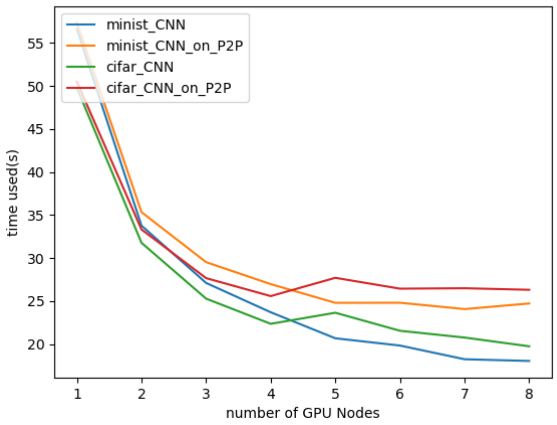

Figure 4: 在网络上的训练结果

这是我使用 minist 和 cifar 数据集分别在网络上使用不同 GPU 节点数量训练的两种不同模型。为了突显出时延的效果，同时降低成本，提高训练速度。我对两个模型的 epoch 和 batchsize 各自进行了减小。从图中可以看出，经过调整后两个模型的训练耗时曲线十分接近。在这两个模型中，minist 模型在网络上训练的结果大致符合预期，随着 GPU 节点的增加，虽然通信消耗也会上升，但总体消耗时间是一直下降的。但在 cifar 数据集上训练的模型却不同，cifar 模型在 GPU 节点数量在 4 到 5 时，其时间消耗反而上升了，这个现象可能是由于 cifar 本身数据大于 minist 数据集，导致在 GPU 节点在 4 到 5 个时，其节点间通信成本显著增加，抵消了 GPU 节点增加带来的算力提升。除此之外，从图中还可以看出，相比于单机多卡的情况，分布式训练在 P2P 网络上训练的时间成本随着节点增加而显著增加。导致这种情况的主要原因是分布式训练受网络延迟和网络节点间额外通信的影响，是可以预料到的。下图是随着 GPU 节点数量的增加，P2P 网络上的额外负担对应的时间开销增加的百分比。

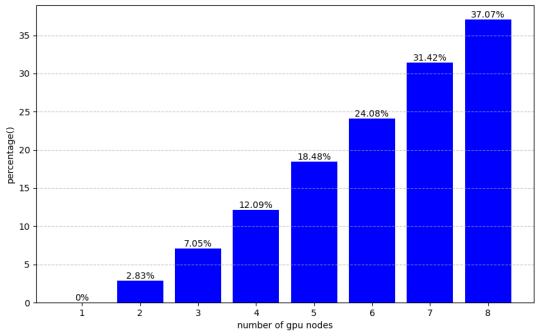

Figure 5: 时间开销增加百分比

从中可以看出，随着 GPU 节点数量的增加，总体时间开销所占百分比也越来越大，这也是符合预期的，由于 GPU 节点间通信频率随着 GPU 节点增加而增加，总体消耗的时间也会因此而上升。

4.4.2 节点选择算法

考虑到将 GPU 均分到节点上情况下，各个节点之间的算力只有 0 与 1 的区别，为了更真实的模拟 P2P 网络，本文将各个 docker 容器内部的 GPU 数量进行调整，调整为 3，2，1，1，1 的节点容器情况以进行进一步的测试。对于无算力节点，我提前设置了其模型传输次数等数据。但由于本文网络规模较小，在选择节点时，限制单个节点选择其他节点时的请求次数，这里我设置为 2，3，4,6 和 8，即初始节点最多只能向 2 个，3 个，4 个,6 个或 8 个节点发起请求，请求内容皆为算力请求，经过 50 次节点选择后的结果如下：

	node 5(3 GPU)	node 6(2 GPU)	node 7(2GPU)	node 11(1 GPU)	node 17(1 GPU)
2 requests	8	5	2	3	3
3 requests	4	12	13	7	7
4 requests	8	11	16	6	11
6 requests	16	11	7	17	17
8 requests	19	23	19	16	2

Figure 6: GPU 节点选取情况

同时经统计，在 50 次节点选择过程中，8 次和 6 次请求全部都请求到了有算力节点，3 次 4 次都共有 46 次请求到了有算力节点，2 次共有 35 次请求到了有算力节点。这说明在实际测试中，使用 3 次请求则可以以较高概率请求到有算力节点同时，又能节约请求次数造，减少时延问题，同时也能模拟出只能请求到一部分有算力节点的现实 P2P 网络情况。
使用 3 次请求作为请求数，进行分布式训练，其得到的结果如下

	time(s)
2	32.60697293281555
3	26.497018575668335
0	0.0003161022949218 75
3	25.902172803878784

Figure 7: minits 数据集，纵坐标为节点 GPU 组成，横坐标为训练时间

	time(s)
3 and 2	25.624929428100586
0	0.0003622604370117 1874
0	0.0002406669616699219
2	32.05440645217895

Figure 8: cifar 数据集，纵坐标为节点 GPU 组成，横坐标为训练时间

根据上图可以看出，虽然理论上，在网络上使用多 GPU 的单个节点理应比单 GPU 多个节点耗时更短，如 cifar 数据集的 2GPU 和 3GPU 组合训练的情况，但仍有部分节点未能达到这个标准，如 minist 数据集的 3 节点 GPU 情况，

我认为，这可能是因为训练时环境波动的原因。但总体而言，使用多 GPU 的单个节点确实能减少训练耗时，符合预期。但也存在因网络环境波动造成的数据不符合预期的情况。

5 未来展望

由于 GPU 价格昂贵，目前我只能租用 8 卡 GPU 进行训练，这导致我的模拟规模很小，在改进方面，我认为首先可以扩展网络规模，提高 GPU 数量。其次提高其他无算力节点数量，真实的 P2P 网络中往往存在大量低算力节点，这些节点不能作为分布式训练使用，而要从海量低算力节点中找到高算力节点，还需要更先进有效的节点选择算法。除此之外，我认为对于 P2P 网络中设备中断，不同设备算力估计，也需要进一步改进，真实网络中存在各种各样的突发情况，相比之下，我的模拟网络还是显得十分脆弱。

6 参考文献

[1] 蒋丰泽. 大模型分布式训练方法研究综述 [J]. 深圳信息职业技术学院学报,2023,21(06):9-15.
[2]Shoeybi M ,Patwary M ,Puri R , et al.Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.[J].CoRR,2019,abs/1909.08053
[3] 刘玉枚, 杨寿保, 陈万明, 等.P2P 系统中基于信誉感知的超级节点选择算法研究 [J]. 中国科学院研究生院学报,2008(02):
[4] 陈翔. 粒子群改进算法在 P2P 网络中的研究与应用 [D]. 西华大学,2015.197-203.
[5] 马丽芳, 陈伟峰, 兰世战, 等. 基于蚁群优化的移动 P2P 网络路由选择算法 [J]. 电信科学,2016,32(07):121-125.
[6] 文欢. 模型分布式训练方法研究及实现 [D]. 电子科技大学,2023.DOI:10.27005/d.cnki.gdzku.2022.004131.

7 致谢

我要感谢我的导师唐茗老师，感谢她在整个研究过程中给予我的耐心指导和宝贵建议。唐茗老师的严谨学风和执着的学术态度，一直是我学习和研究的楷模。我要感谢我的家人，特别是我的父母，他们无私的爱和理解支持了我整个学习和研究过程。没有他们的鼓励和支持，我无法完成这项工作。大学生活即将结束，我也将踏上下一步旅程，愿前途一片光明。