分布式训练中的组通信

计算机学院 2211290 姚知言

并行程序设计 Open Topic

2024.06

Contents

01 什么是分布式训练

02 组通信在分布式训练的应用

03 相关工具及研究

04 总结与展望

01.
什么是分布式训练

2211290 姚知言
计算机学院

STAR_

2024.06

分布式训练

分布式训练（Distributed Training）是指将机器学习或深度学习模型训练任务分解成多个子任务，并在多个计算设备上并行地进行训练。根据数据和模型在多台机器上的分布方式，分布式训练可以分为数据并行和模型并行两种模式。

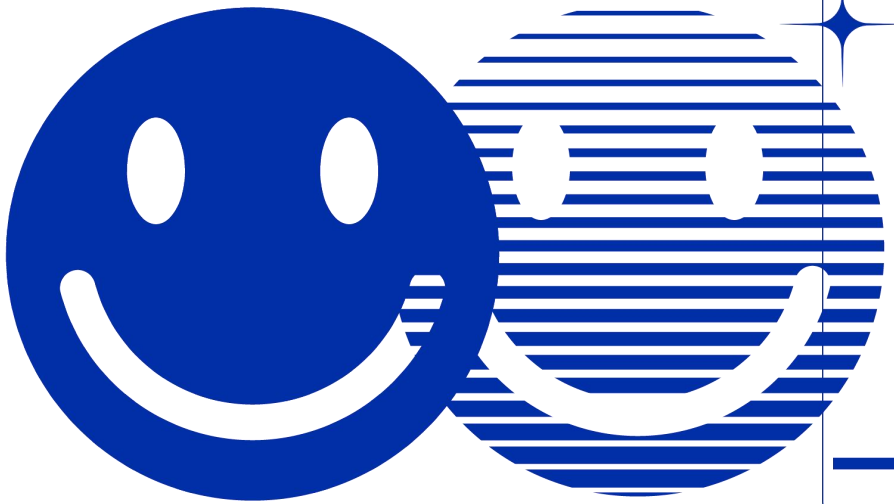
由于同一个服务器内部的多个计算设备之间内存也可能并不共享，因此无论这些计算设备是否处于一个服务器还是多个服务器中，其系统架构都属于分布式系统范畴。一个模型训练任务往往会有大量的训练样本作为输入，可以利用一个计算设备完成，也可以将整个模型的训练任务拆分成子任务，分发给不同的计算设备，实现并行计算。此后，还需要对每个计算设备的输出进行合并，最终得到与单个计算设备等价的计算结果。由于每个计算设备只需要负责子任务，并且多个计算设备可以并行执行，因此其可以更快速地完成整体计算，并最终实现对整个计算过程的加速。

分布式训练的总体目标就是提升总的训练速度，减少模型训练的总体时间。总训练速度可以用如下公式简略估计：

$$\text{总训练速度} \propto \text{单设备计算速度} \times \text{计算设备总量} \times \text{多设备加速比}$$

2024.06

为什么我们需要分布式训练



DELICIOUS ✦ DELICIOUS ✦ DELICIOUS ✦ DELICIOUS ✦ DELICIOUS

HIDDEN VISION

随着机器学习、深度学习领域的发展，单个计算设备的算力已经不足以支撑模型训练。

从2013 年AlexNet 开始，到2022 年拥有5400 亿参数的PalM 模型，机器学习模型以每18 个月增长56 倍的速度发展。模型参数规模增大的同时，对训练数据量的要求也指数级增长，这更加剧了对算力的需求。然而，近几年CPU 的算力增加已经远低于摩尔定律，虽然计算加速设备（如GPU、TPU 等）为机器学习模型提供了大量的算力，但是其增长速度仍然没有突破每18 个月翻倍的摩尔定律。为了能够满足大模型等技术的发展，只有通过分布式训练系统才可以匹配模型不断增长的算力需求。

02.
组通信在分布式训练的应用

2211290 姚知言
计算机学院

STAR_

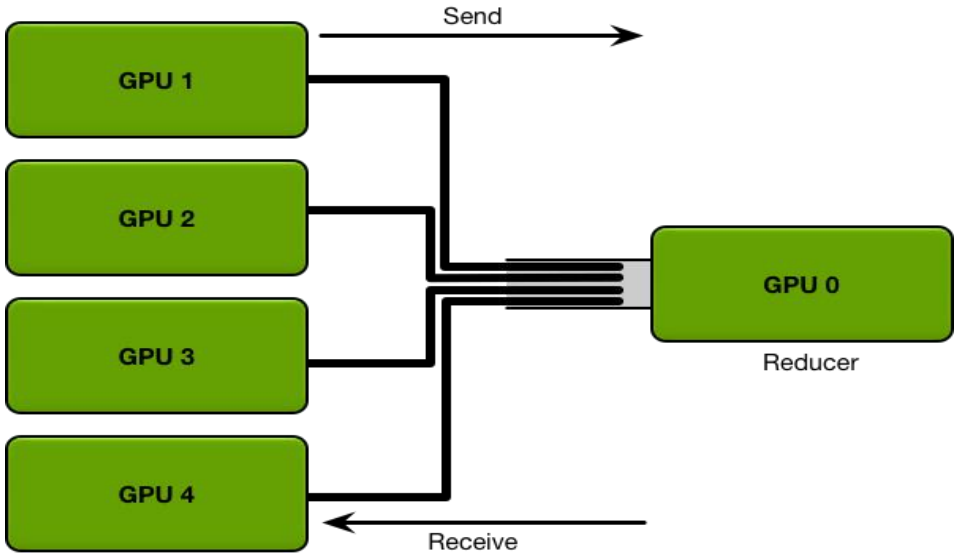
起源：Bringing HPC Techniques to Deep Learning-baidu

背景

在过去的几年里，神经网络已被证明是解决各种问题的非常有效的工具，并且在规模和计算要求方面迅速增长。随着神经网络参数数量和计算需求的增长，在许多节点和许多 GPU 上有效地并行进行神经网络训练变得越来越重要，因为等待数月的大型网络训练会减慢实验速度并限制进一步的开发。

过往GPU通信方式（Parameter server）

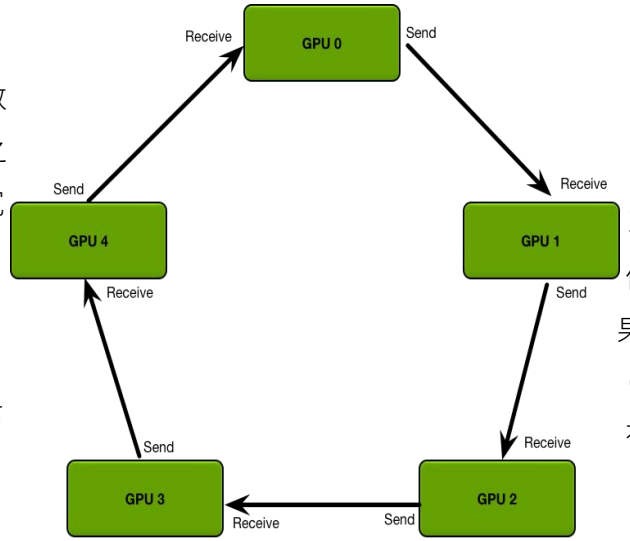
将神经网络训练并行到多个 GPU 上时，必须选择如何将不同的操作分配到可用的不同 GPU 上。在本实验中应用并行随机梯度下降（SGD）的技术。每个 GPU 都有整个神经网络模型的完整副本，并且每次迭代都只分配小批量样本的子集。对于每次迭代，每个 GPU 都会在其数据上运行网络正向传播，然后进行误差反向传播，以计算相对于网络参数的损失梯度。最后，GPU相互通信，对不同GPU计算的梯度进行平均，将平均梯度应用于权重以获得新的权重。



如图所示，每个 GPU 都会在其小批量的子集上计算梯度。然后，每个 GPU 将其梯度发送到单个 GPU，该 GPU 获取所有梯度的平均值，并将平均值发送回所有其他 GPU。注意：为保证正确更新权重，先完成计算的GPU需要进行锁步。

在从单个 GPU 发送和接收数据的直接机制中，该单个 GPU 必须接收来自所有 GPU 的所有参数，并将所有参数发送到所有 GPU。系统中的 GPU 越多，通信成本就越高。随着 GPU 数量的增加，每次迭代所需的时间呈线性增长。即使每次迭代需要几秒钟的时间，通信成本的这种线性增长使进一步的并行化变得不切实际，也会降低效率。

Ring Allreduce 是一种通信成本恒定的算法，与系统中的 GPU 数量无关，并且仅由系统中 GPU 之间最慢的连接决定;在只考虑带宽而不考虑延迟的情况下，那么 Ring AllReduce 是一种最佳通信算法。将 GPU 排列在逻辑环中。每个 GPU有且只有一个左邻和右邻，它只会将数据发送到其右邻，并从其左邻接收数据。



该算法分两步进行：计算之前需要分散减少，完成之后需要全聚集。在分散减少步骤中，GPU 通过数据交换，便每个 GPU 最终获得最终结果的一块。在全聚集步骤中，GPU 通过数据交换，以便所有 GPU 最终获得完整的最终结果。

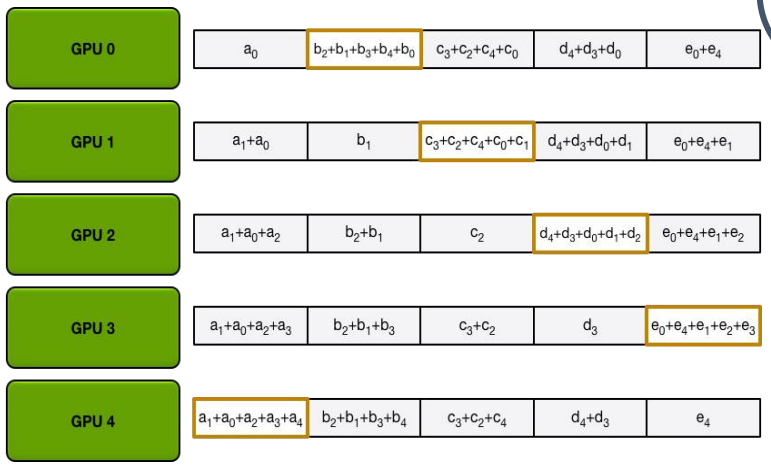
存在的问题

The Ring Allreduce

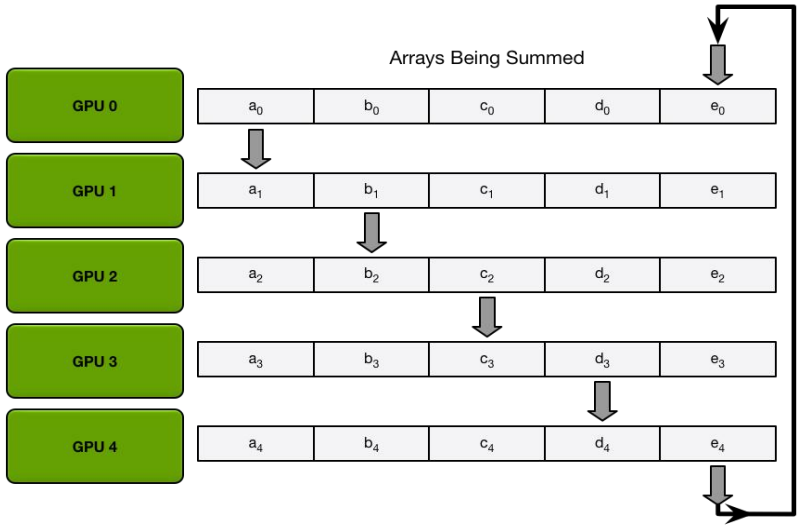
2024.06

一个例子：分散减少Scatter-Reduce过程图解

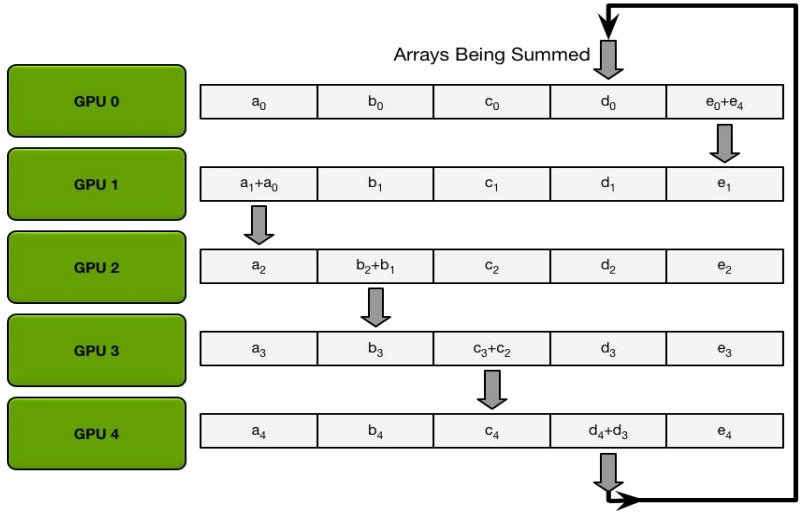
5



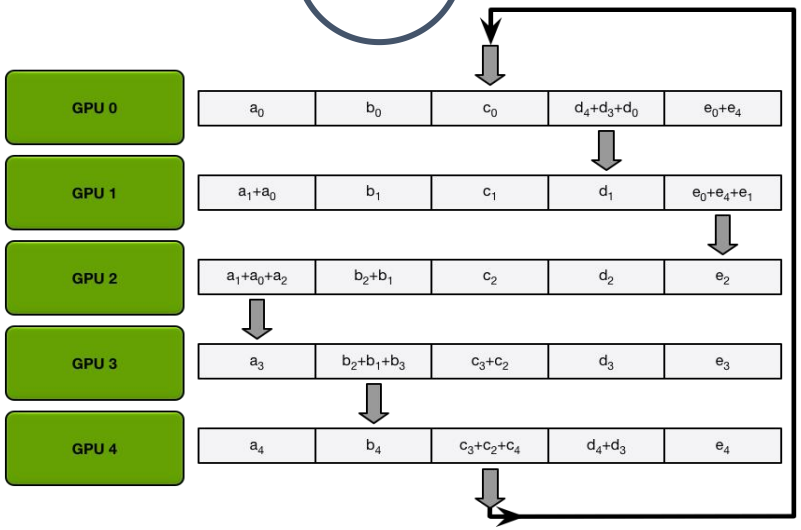
1



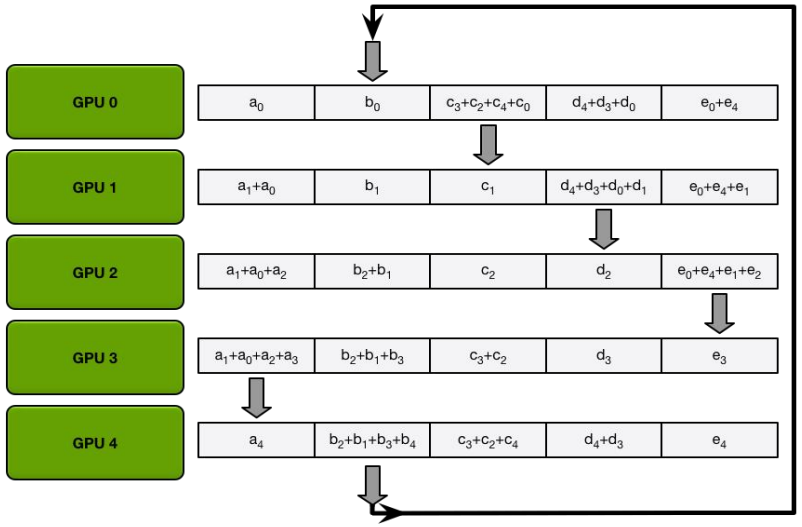
2



3



4



2024.06

一个例子：Allgather全聚集过程图解

5

GPU 0	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 1	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 2	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 3	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 4	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$

1

GPU 0	a_0	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0$	$d_4+d_3+d_0$	e_0+e_4
GPU 1	a_1+a_0	b_1	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1$	$e_0+e_4+e_1$
GPU 2	$a_1+a_0+a_2$	b_2+b_1	c_2	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2$
GPU 3	$a_1+a_0+a_2+a_3$	$b_2+b_1+b_3$	c_3+c_2	d_3	$e_0+e_4+e_1+e_2+e_3$
GPU 4	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4$	$c_3+c_2+c_4$	d_4+d_3	e_4

2

GPU 0	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0$	$d_4+d_3+d_0$	e_0+e_4
GPU 1	a_1+a_0	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1$	$e_0+e_4+e_1$
GPU 2	$a_1+a_0+a_2$	b_2+b_1	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2$
GPU 3	$a_1+a_0+a_2+a_3$	$b_2+b_1+b_3$	c_3+c_2	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 4	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4$	$c_3+c_2+c_4$	d_4+d_3	$e_0+e_4+e_1+e_2+e_3$

3

GPU 0	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 1	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1$	$e_0+e_4+e_1+e_2+e_3$
GPU 2	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2$
GPU 3	$a_1+a_0+a_2+a_3$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 4	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$

4

GPU 0	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0$	$d_4+d_3+d_0$	$e_0+e_4+e_1+e_2+e_3$
GPU 1	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1$	$e_0+e_4+e_1$
GPU 2	$a_1+a_0+a_2$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2$
GPU 3	$a_1+a_0+a_2+a_3$	$b_2+b_1+b_3$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 4	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4$	$c_3+c_2+c_4$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$

5

GPU 0	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 1	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1$	$e_0+e_4+e_1+e_2+e_3$
GPU 2	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2$
GPU 3	$a_1+a_0+a_2+a_3$	$b_2+b_1+b_3+b_4+b_0$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$
GPU 4	$a_1+a_0+a_2+a_3+a_4$	$b_2+b_1+b_3+b_4$	$c_3+c_2+c_4+c_0+c_1$	$d_4+d_3+d_0+d_1+d_2$	$e_0+e_4+e_1+e_2+e_3$

优势

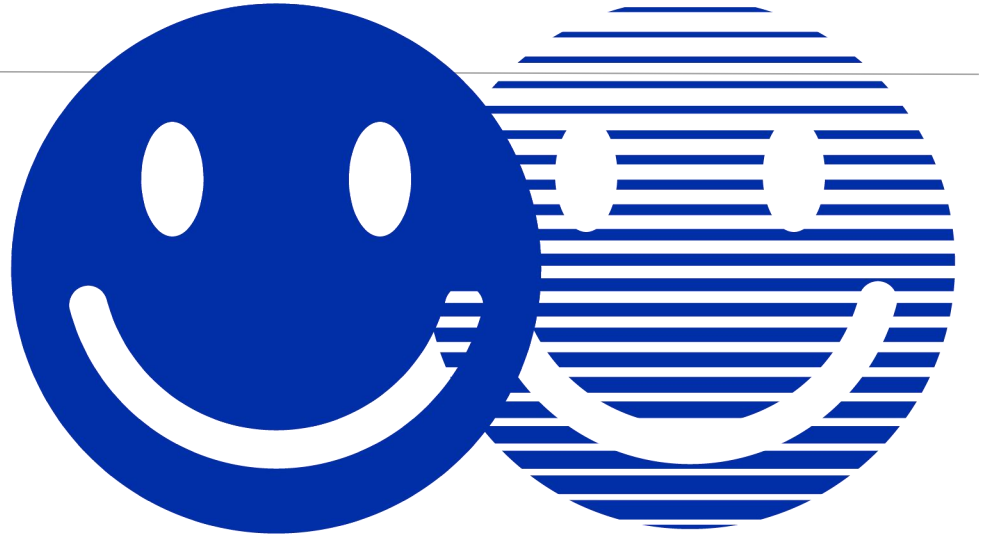
深度学习中的应用

假设我们有N个CPU，不同CPU需要计算的数据总量为K
在该算法中，每个 GPU 将发送和接收值 N-1 次（分散减少）和 N-1 次（全聚集）。每次，GPU 都会发送 K/N 个值，其中 K 是中不同 GPU 计算的计算总数。因此，传入和传出每个 GPU 的数据总量为

$$2(N-1)\frac{K}{N}$$

至关重要的是，这与 GPU 的数量无关。
由于不同GPU之间的传输都是在迭代中同步进行的，因此 Ring Allreduce 的速度受到环中相邻 GPU 之间最低带宽连接的限制。
当然，要保证所有GPU在环中彼此相邻，才能使得算法发挥最好效果。

为了最小化通信开销，可以利用神经网络的结构。在每次迭代中，每个 GPU 都会正向传播以计算误差，然后反向传播以计算神经网络每个参数的梯度。反向传播计算从输出层开始并移入前面层的梯度，这意味着输出层参数的梯度在先前层的梯度之前可用。
由于 Allreduce 可以一次对网络参数的子集进行操作，因此我们可以在计算其他梯度时对输出层参数启动 Allreduce。这样做会将通信与反向传播步骤中的其余计算叠加在一起，从而减少每个 GPU 最终等待通信完成的总时间。



03.
相关工具及研究

2211290 姚知言
计算机学院

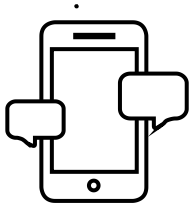
STAR_

2024.06

Ring Allreduce in Tenserflow

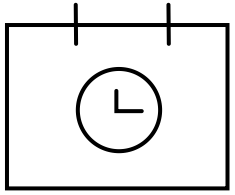
Feb 2017

百度在PaddlePaddle平台上首次引入了Ring Allreduce的架构，并将其提交到tensorflow的contrib package中



Aug 2017

Uber为tensorflow平台开源了一个更加易用和高效的Ring Allreduce分布式训练库Horovod



Sep 2018

tensorflow官方在1. 11版本中支持了Ring Allreduce的分布式训练策略CollectiveAllReduceStrategy

2024.06

NVIDIA NCCL

01

引入

2017年，NCCL 2. 0开始使用
Ring-AllReduce

02

简介

NCCL是 GPU 标准通信例程的独立库，实现了 多种基于发送/接收的通信模式。NCCL 支持在单个节点或跨多个节点安装任意数量的 GPU，可用于单进程或多进程（例如 MPI）应用程序。

03

优点

高性能
易于编程
兼容性强

04

进一步的组通信

2D环
双二叉树

04.
总结与展望

2211290 姚知言
计算机学院

STAR_

总结

组通信是指在并行计算和分布式系统中，多个进程或节点协同进行数据传输和处理的操作。

截至今日，常见的组通信结构有：Flat Ring，2D Ring，Binary Tree等。

组通信在分布式训练中的应用是非常有意义的，对于保证数据的同步性和高速提升性能都很有价值。

展望

在未来，可能会对组通信的拓扑结构进行进一步的优化，以使得带宽被更充分的使用，提升性能。

此外，通过硬件层面进行GPU之间的通信加速，也会对通信效率进行更大帮助。

2024.06

REFERENCE

大模型LLM之分布式训练

https://bbs.huaweicloud.com/blogs/426944?utm_source=cnblog&utm_medium=bbs-ex&utm_campaign=other&utm_content=content

Bringing HPC Techniques to Deep Learning

from Baidu

https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/

Tenserflow

https://zhoujm-blog.readthedocs.io/en/latest/%E7%A8%8B%E5%BA%8F%E4%BA%BA%E7%94%9F/TensorFlow/

NCCL

https://developer.nvidia.com/nccl

Massively Scale Your Deep Learning Training with NCCL 2.4

https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/

THANK YOU

计算机学院 2211290 姚知言

并行程序设计 Open Topic

2024.06