7.自动并行

1.简述

自动并行的目标就是**用户给定一个模型和所使用的机器资源后，能够自动地帮用户选择一个比较好或者最优的并行策略来高效执行。**可以说，自动并行是分布式并行的终极目标，它能够解放工程师去手动设置分布式并行策略。

而自动并行可以分为**全自动并行**和**半自动并行**模式。

- **半自动模式**下用户可以根据自己需要指定某些tensor和operator的切分方式。如：Mesh-TensorFlow、GShard、GSPMD 等提到的自动并行切分方案。
- **全自动模式**下所有 tensor 和 operator 都由框架自适应选择最优切分策略。如：OptCNN、Flexflow、Unity、Alpa 等提到的全自动并行切分方案。

目前，很多的通用AI框架（如：PaddlePaddle、OneFlow、PyTorch、MindSpore、TensorFlow、JAX等）都对自动并行(全自动或半自动)进行了实现。

下面将分享一些典型的分布式训练自动并行方案。

2.Mesh-TensorFlow

2.1 背景

在深度学习中，由于数据量和计算量的庞大，往往会使用到分布式计算。而最常用的分布式模式是SPMD(Single-Program-Multiple-Data)，即数据并行，这种模式相当于在数据的batch维去做拆分；然后，进行并行。Mesh-Tensorflow对这种模式做了泛化，即**除了batch维外的其他维度也可做并行。**

2.2 SPMD 的 batch 切分

首先，回顾下之前的数据并行，每个设备上都有全部模型参数的备份，在每一次迭代中，数据首先被切分分发到各个设备上；然后，各个设备分别进行计算，得到的梯度再通过AllReduce进行聚合，然后再更新参数。

2.3 Mesh-tensorflow 的切分

分布式依赖的是数据分发和聚合，这点上面讲解的batch切分也是，但 Mesh-tensorflow 做了更泛化的抽象。

1. 给 Tensor 的每个维度命名

在传统的深度学习框架中，Tensor 只是一个 **多维数组**，通常只依赖于 **形状（shape）** 来描述它的维度。比如，一个输入 `x` 的维度可以是 `[batch, d_io]`，其中：

- `batch` 是样本的数量（批量大小），
- `d_io` 是每个样本的维度（输入特征的大小）。

然而，在 **Mesh-TensorFlow** 中，每个维度都被赋予了 **更具描述性的名字**，这有助于理解每个维度的角色。例如：

- `batch`：表示训练数据中的样本批次。
- `d_io`：表示输入数据的特征维度。
- 这种命名方式能帮助开发者更清晰地理解各个维度的含义，特别是在 **分布式训练** 中，多个维度可能会被分配到不同的计算单元上。

2. 将处理器集群表示为一个矩阵

在分布式训练中，通常有多个 **计算单元（如 GPU）**。这些计算单元可以组织成一个 **二维结构**，比如：

- `rows`：表示计算单元的行数，
- `cols`：表示计算单元的列数。

通过这种方式，计算单元（即 **处理器集群**）被 **映射** 成一个矩阵结构，帮助理解如何将数据并行地分发到这些计算单元上。举个例子，假设你有一个 `2x2` 的计算单元矩阵，那么这个矩阵就可以用 `[rows, cols]` 来表示，其中 `rows=2`，`cols=2`。

3. 定义 Computation Layout（计算布局）

Computation layout 是 **Mesh-TensorFlow** 的一个核心概念，它定义了从 **tensor 维度** 到 **计算集群维度** 的映射关系。换句话说，计算布局描述了每个 **tensor 的维度** 如何在 **多个计算单元（如 GPU、处理器集群）** 上进行分配和计算。

举个简单的例子，假设你有一个训练数据 `x`，它的维度是 `[batch, d_io]`，你想将 `batch` 维度分配到 **所有的计算单元** 上，并且让每个计算单元处理不同的样本批次。这时，你就可以定义一个 **computation layout**，比如：

- `computation_layout = [("batch", "all_processors")]`

这个布局的意思是，将 `batch` 维度的每个样本分配给 **所有的计算单元**（即每个 GPU），并在这些计算单元上并行计算。这种映射关系确保了每个计算单元都参与到整个数据集的训练过程中，同时也能充分利用 **分布式计算资源**。

2.4 Mesh-tensorflow 实现

每个操作都通过并行计算和 collective communication 来完成，这里，我们介绍几个 Mesh-Tensorflow 中比较重要的操作。

- **Component-wise Operations**: 所谓的component-wise，就是指输入和输出的维度相同。这一类的操作可以直接分布式的进行。
- **Reduction(reduce_sum, reduce_max, etc)**: Reduction操作是指会消减维度的操作，这一类操作可以先在每个切片上操作，然后用MPI-allreduce来聚合。
- **Einstin Summation(max multiplication, etc)**: Einstin操作是一组矩阵计算的统称，在TensorFlow 中被实现成了一个可以配置的API，配置的方式就是用维度的名字来表达计算，这点其实和 Mesh-Tensorflow 异曲同工，所以可以很方便的实现。同样的，实现的方式就是**首先本地计算**每个分片的结果，然后通过 **MPI-allreduce** 等操作来聚合结果，最终确保所有设备的计算一致。
- **Reshape**: Reshape虽然简单，但是在分布式环境下却需要网络通信才能完成，不同的reshape需要的操作不同，涉及到的MPI通信包括MPI-allgather，MPI-alltoall等。

> **Note**
>
> 在分布式计算中，**MPI（Message Passing Interface）** 是一种常用的 **消息传递机制**，用于在多台计算机或者多个计算单元（如多个 GPU）之间进行高效的通信和数据共享。MPI 提供了多种不同的操作，这里提到的 `MPI-allreduce`、`MPI-allgather` 和 `MPI-alltoall` 是其中的几种重要操作，分别用于不同的通信需求：
>
> **MPI-allreduce**：
>
> - **作用**：它用于 **在所有计算单元之间聚合数据**（如加法、最大值等），并将结果 **广播回所有计算单元**。

- **应用场景**：在分布式训练中，通常会使用 **MPI-allreduce** 来 **同步梯度**。每个计算单元（如 GPU）计算出自己的梯度，然后通过 `MPI-allreduce` 聚合所有计算单元的梯度（例如：对梯度进行加和），然后广播给所有计算单元，以确保每个计算单元的梯度一致。

MPI-allgather：

- **作用**：在所有计算单元之间，**收集每个计算单元的数据**，并将其聚合成一个大的数据块，最终每个计算单元都获得这个聚合后的数据。
- **应用场景**：当你需要在分布式系统中收集来自不同计算单元的数据时，使用 `MPI-allgather`。例如，在并行训练中，如果你有多个 GPU，每个 GPU 处理数据的一部分，**MPI-allgather** 会将所有 GPU 的数据聚合起来，使得每个 GPU 都能获取完整的数据。

MPI-alltoall：

- **作用**：每个计算单元将自己的数据发送到所有其他计算单元，每个计算单元 **接收来自所有其他计算单元的数据。**
- **应用场景**：当你需要 **不同计算单元之间交换数据** 时使用 `MPI-alltoall`。例如，如果你在进行模型并行的计算，不同 GPU 存储着不同的模型部分，你可能需要交换这些部分，以便每个 GPU 计算模型的不同部分。

特性	MPI-allreduce	MPI-allgather
功能	聚合数据（如求和、最大值等），并广播结果给所有计算单元	收集每个计算单元的数据，并广播给所有计算单元
数据处理	每个计算单元计算本地数据的某种操作（如梯度），然后将结果聚合	每个计算单元收集其他计算单元的数据，所有计算单元获得完整数据
应用场景	梯度同步、全局统计（如求和、最大值等）	数据收集、所有计算单元共享完整数据
输出	所有计算单元得到 **聚合后的结果**	所有计算单元得到 **所有计算单元的数据**

总结：

- **MPI-allreduce**：首先进行 **数据聚合**，然后将结果 **广播** 给所有计算单元。
- **MPI-allgather**：只是 **收集数据**，没有进行聚合，最后将 **所有数据广播** 给所有计算单元。

2.5 小结

Mesh-Tensorflow 定义了一套DSL语法，用于描述模型的维度和布局，你用它重写你的整个Model后，它自动帮你把模型和数据分割到多个TPU上。

另外，Mesh-Tensorflow 没有实现并行的卷积操作，因此，只适合 Language Model 这个领域。

除此之外，需要用 Mesh-Tensorflow 的语法重写你的整个模型，仔细思考维度，不仅工作量大，同时对代码侵入性强。

不同的 layout 会带来不同的性能，因此，可以考虑自动搜索最优的layout，但 Mesh-Tensorflow不支持。