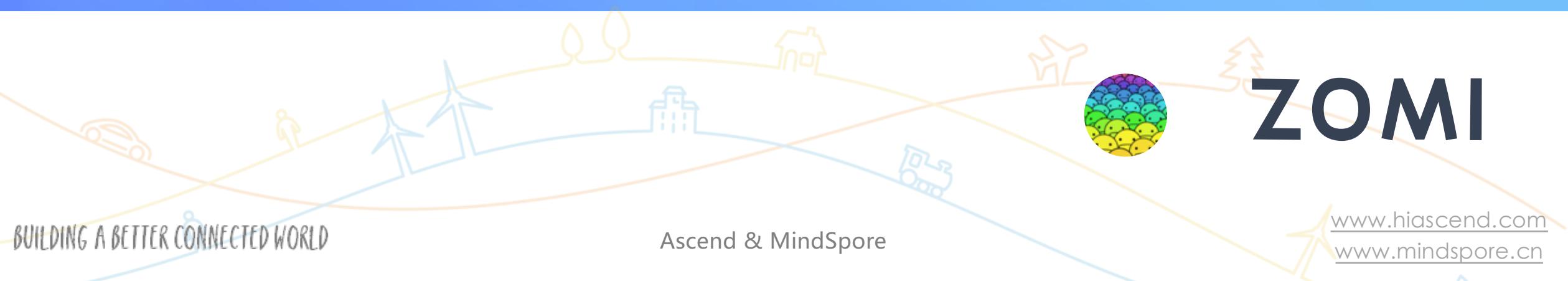


分布式训练系列

AI框架中的分布式



关于本内容

1. 内容背景

- AI集群+大模型+分布式训练系统

2. 具体内容

- **AI集群服务器架构**：参数服务器模式 – 同步与异步并行 - 环同步算法
- **AI集群软硬件通信**：通信软硬件实现 - 通信实现方式
- **分布式通信原语**：通信原语
- **框架分布式功能**：并行处理硬件架构 – AI框架中的分布式训练

- **大模型算法**：挑战 – 算法结构 – SOTA大模型
- **分布式并行**：数据并行 – 张量并行 – 自动并行 – 多维混合并行

并行处理硬件架构

并行处理硬件架构

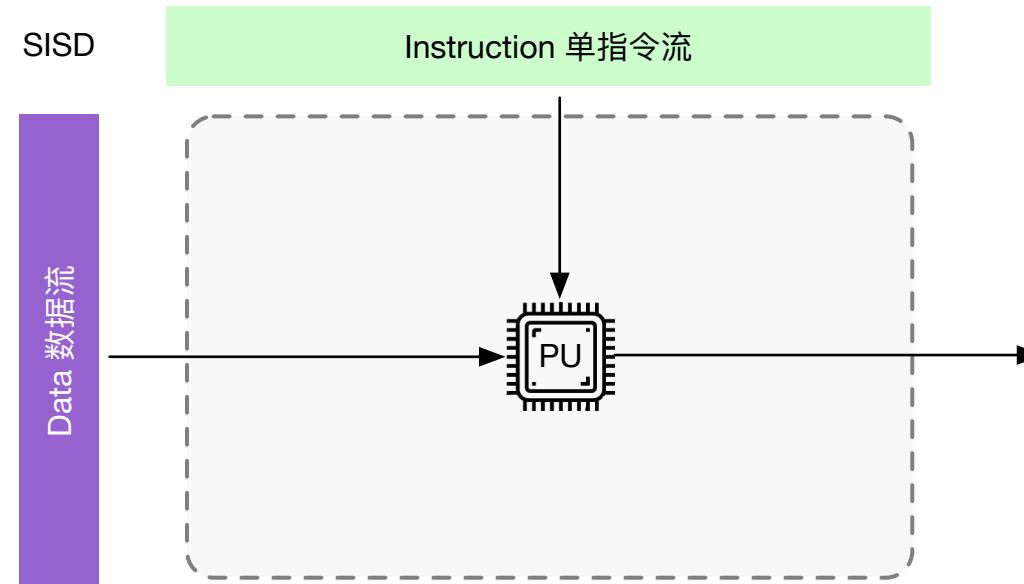
- 单指令单数据流 (SISD) 系统。
- 单指令多数据流 (SIMD) 系统。
- 多指令单数据流 (MISD) 系统。
- 多指令多数据流 (MIMD) 系统。

		Data stream	
		Single	Multiple
Instruction stream	Single	SISD $a_1 + b_1$	SIMD $a_1 + b_1$ $a_2 + b_2$ $a_3 + b_3$
	Multiple	MISD $a_1 + b_1$ $a_1 - b_1$ $a_1 * b_1$	MIMD $a_1 + b_1$ $a_2 - b_2$ $a_3 * b_3$

并行计算处理硬件架构 (I) : SISD 系统

- 每个指令部件每次仅译码一条指令，而且在执行时仅为操作部件提供一份数据
- 串行计算，硬件不支持并行计算；在时钟周期内，CPU只能处理一个数据流。

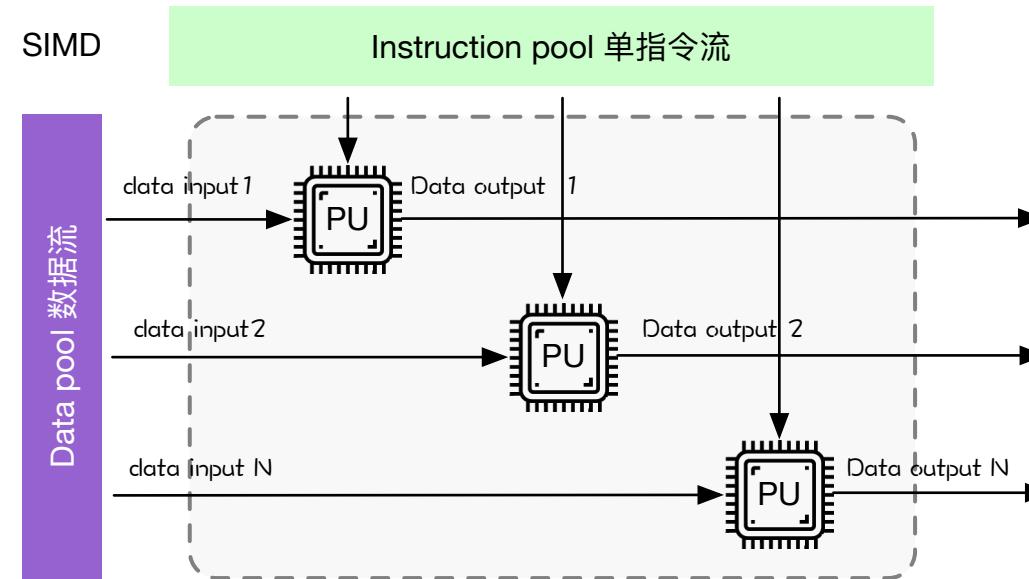
		Data stream	
		Single	Multiple
Instruction stream	Single	SISD [$a_1 + b_1$]	SIMD [$a_1 + b_1$, $a_2 + b_2$, $a_3 + b_3$]
	Multiple	MISD [$a_1 + b_1$, $a_1 - b_1$, $a_1 * b_1$]	MIMD [$a_1 + b_1$, $a_2 - b_2$, $a_3 * b_3$]



并行计算处理硬件架构 (II) : SIMD 系统

- 一个控制器控制多个处理器，同时对一组数据中每一个分别执行相同操作
- SIMD主要执行向量、矩阵等数组运算，处理单元数目固定，适用于科学计算
- 特点是处理单元数量很多，但处理单元速度受计算机通讯带宽传递速率的限制

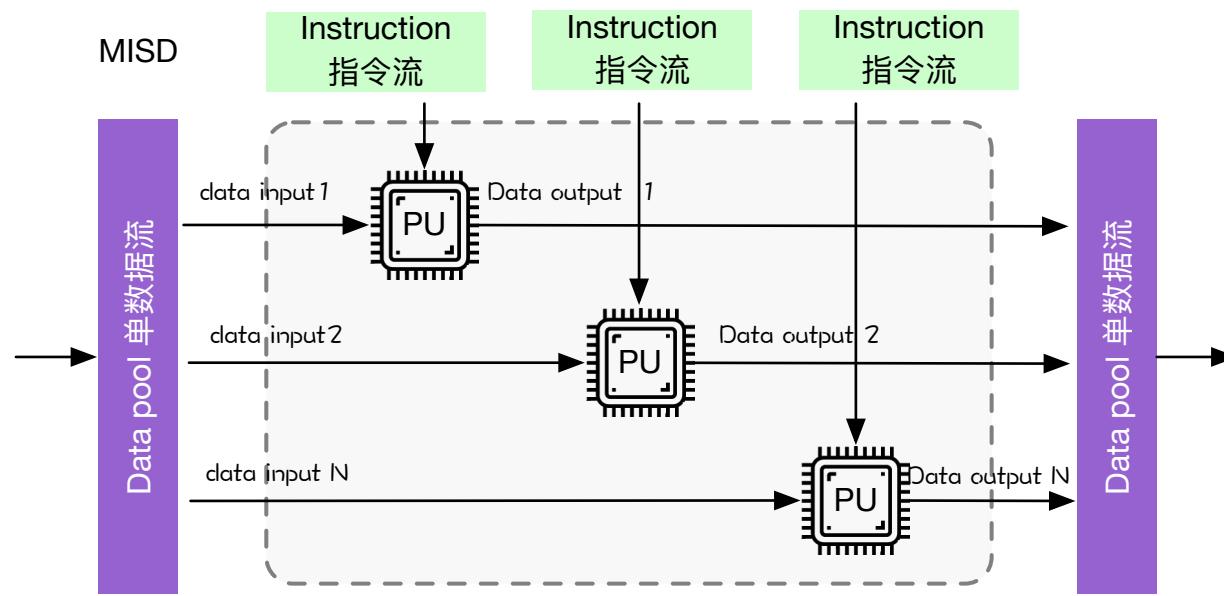
		Data stream	
		Single	Multiple
		SISD	SIMD
Instruction stream	Single	$a_1 + b_1$	$a_1 + b_1$ $a_2 + b_2$ $a_3 + b_3$
	Multiple	MISD	$a_1 + b_1$ $a_1 - b_1$ $a_1 * b_1$
		MIMD	$a_1 + b_1$ $a_2 - b_2$ $a_3 * b_3$



并行计算处理硬件架构 (I) : MISD 系统

- 多指令流单数据流机器，采用多个指令流来处理单个数据流
- 作为理论模型出现，没有投入到实际应用之中

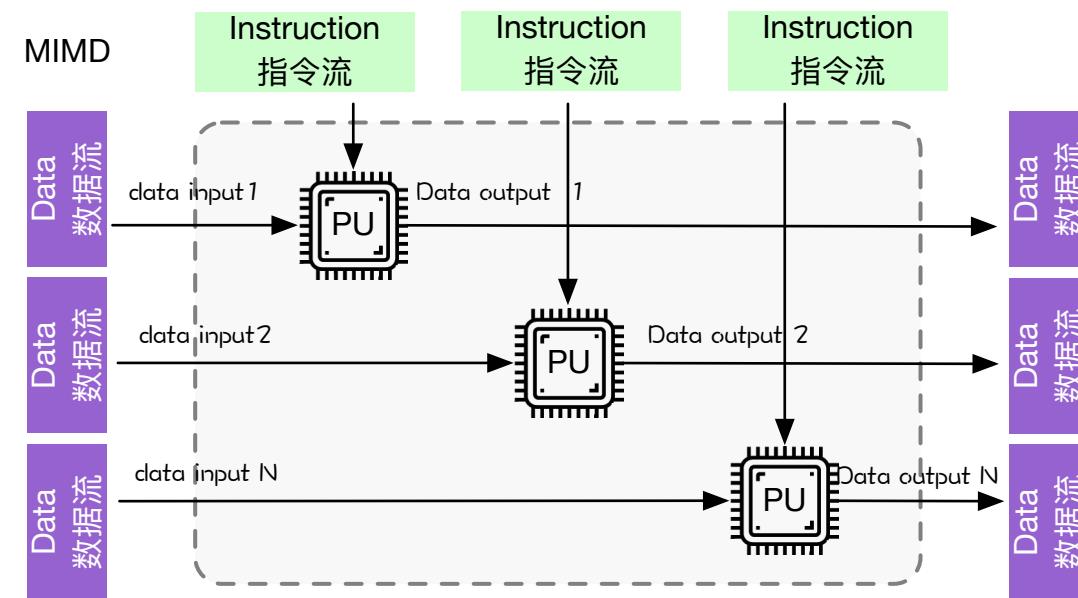
		Data stream	
		Single	Multiple
Instruction stream	Single	SISD $[a_1 + b_1]$	SIMD $[a_1 + b_1]$ $[a_2 + b_2]$ $[a_3 + b_3]$
	Multiple	MISD $[a_1 + b_1]$ $[a_1 - b_1]$ $[a_1 * b_1]$	MIMD $[a_1 + b_1]$ $[a_2 - b_2]$ $[a_3 * b_3]$



并行计算处理硬件架构 (III) : MIMD 系统

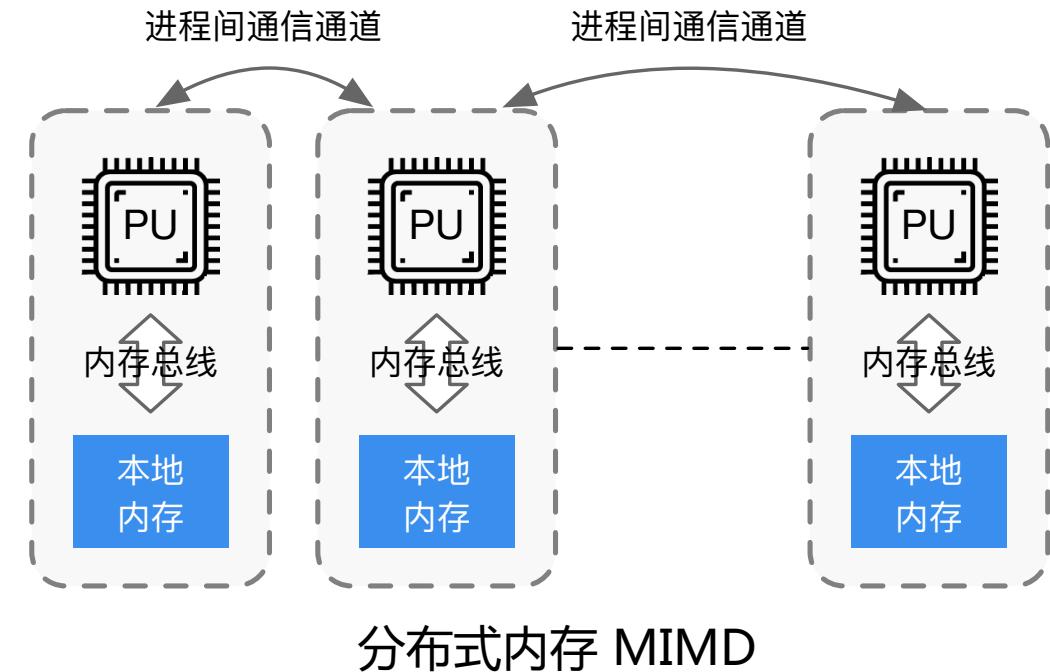
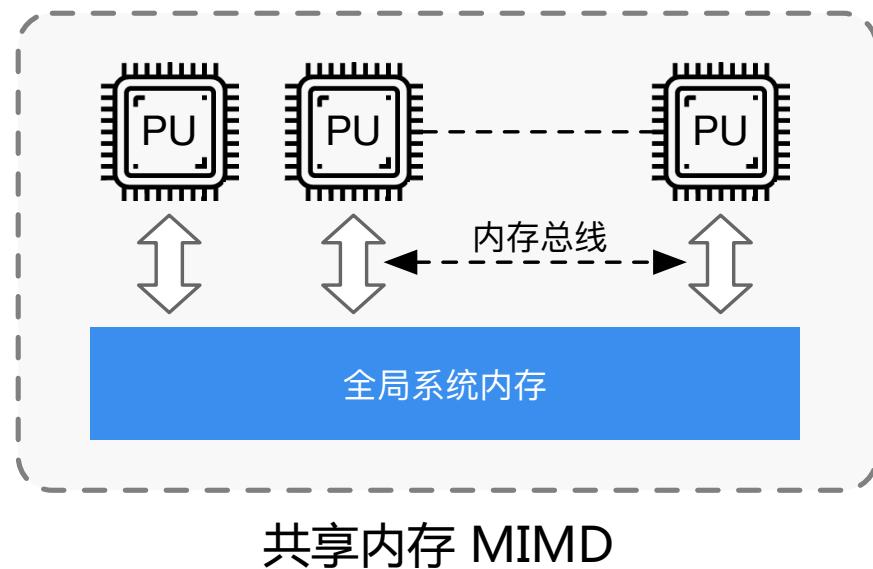
- 在多个数据集上执行多个指令的多处理器机器
- 共享内存 MIMD 和分布式内存

		Data stream	
		Single	Multiple
Instruction stream	Single	SISD $a_1 + b_1$	SIMD $a_1 + b_1$ $a_2 + b_2$ $a_3 + b_3$
	Multiple	MISD $a_1 + b_1$ $a_1 - b_1$ $a_1 * b_1$	MIMD $a_1 + b_1$ $a_2 - b_2$ $a_3 * b_3$



并行计算处理硬件架构 (III) : MIMD 系统

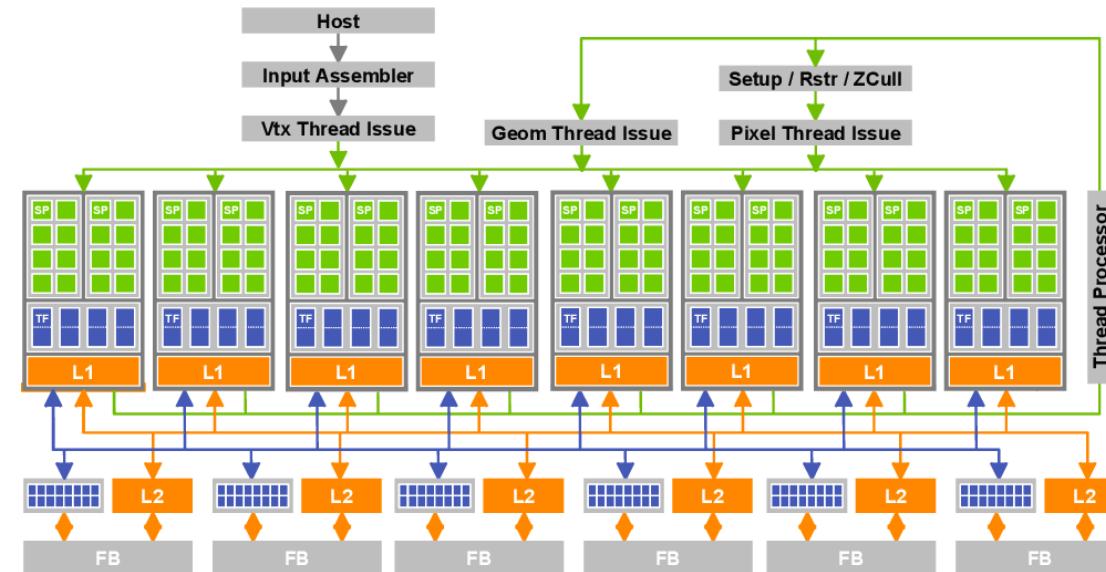
- 在多个数据集上执行多个指令的多处理器机器
- 共享内存 MIMD 和分布式内存 MIMD



并行计算处理硬件架构 (III) : SIMT 系统

- 单指令多线程，有效地管理和执行多个单线程，允许一条指令的多数据分开寻址
- 无需开发者把数据凑成合适的矢量长度，并且SIMT允许每个线程有不同的分支
- 条件跳转会根据输入数据不同在不同的线程中有不同表现

		Data stream	
		Single	Multiple
Instruction stream	Single	SISD $a_1 + b_1$	SIMD $a_1 + b_1$ $a_2 + b_2$ $a_3 + b_3$
	Multiple	MISD $a_1 + b_1$ $a_1 - b_1$ $a_1 * b_1$	MIMD $a_1 + b_1$ $a_2 - b_2$ $a_3 * b_3$



分布式训练系统

分布式训练系统

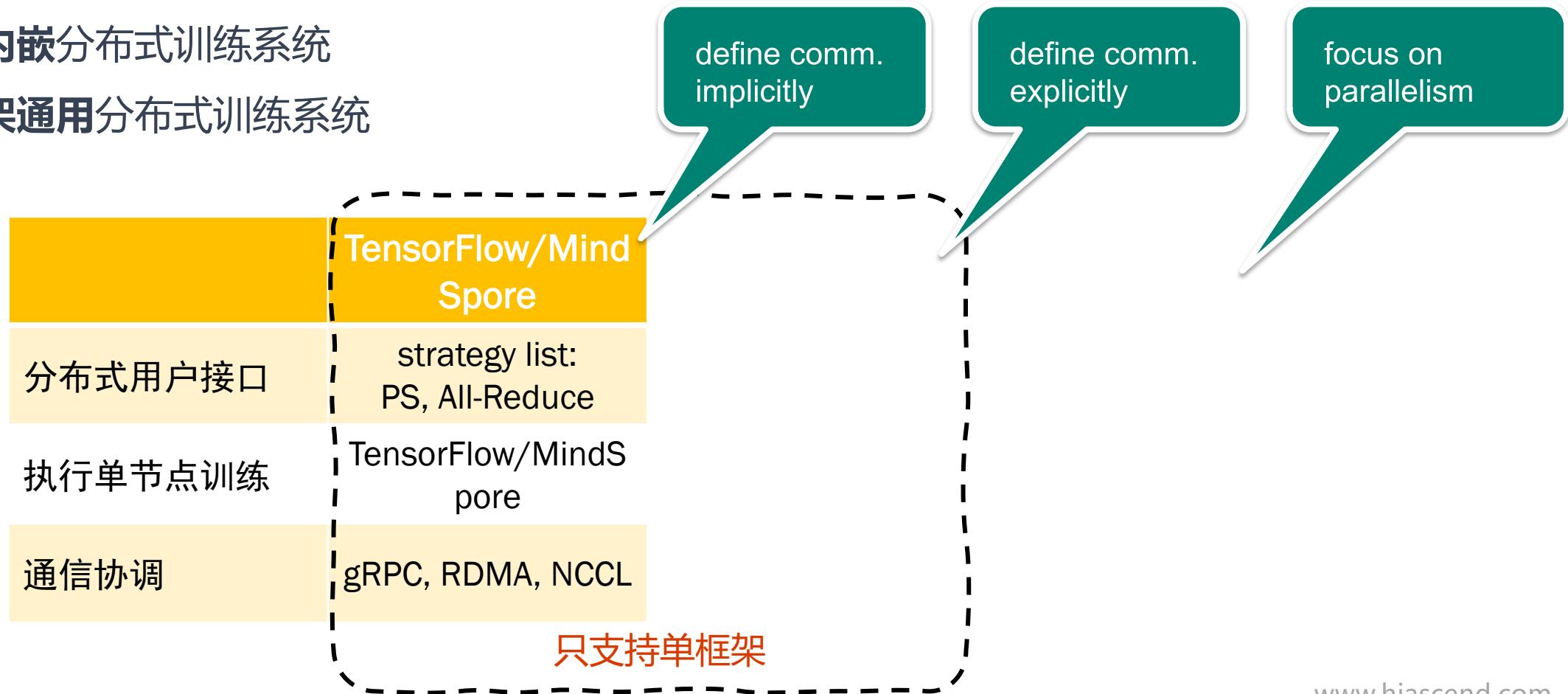
定义：能够分布式地执行深度学习的训练的系统

- **分布式用户接口**
 - 用户通过接口，实现模型的分布化
- **执行单节点训练**
 - 产生本地执行的逻辑
- **通信协调**
 - 实现多节点之间的通信协调

意义：提供易于使用，高效率的分布式训练

分布式训练系统分类对比

- 按照实现方式的不同，分为：
- **框架内嵌分布式训练系统**
- **跨框架通用分布式训练系统**



分布式训练系统 (I)



- TensorFlow通过不同的API支持多种分布式策略(distributed strategies)

Training API	MirroredStrategy	TPUStrategy	MultiWorker-MirroredStrategy	CentralStorage-Strategy	ParameterServer-Strategy	OneDeviceStrategy
Keras API	Supported	Experimental support	Experimental support	Experimental support	Supported planned post 2.0	Supported
Custom training loop	Experimental support	Experimental support	Support planned post 2.0	Support planned post 2.0	No support yet	Supported
Estimator API	Limited Support	Not supported	Limited Support	Limited Support	Limited Support	Limited Support

分布式训练系统（I）



- TensorFlow 在版本（ v0.8 ）中加入基于**参数服务器** “Parameter Server” 的分布式训练
- 思路：多worker独立进行本地计算，分布式共享参数

分布式训练系统 (I)



TensorFlow参数服务器用户接口

● 定义模型

- 指定节点信息 (PS , Worker)
- Worker 包含 “原模型” 逻辑

● 执行模型

- 指定角色 : PS / Worker
- 指定rank : 第几个 PS / Worker

```
tf.train.ClusterSpec({  
    "worker": [  
        "worker0.example.com:2222",  
        "worker1.example.com:2222",  
        "worker2.example.com:2222",  
    ],  
    "ps": [  
        "ps0.example.com:2222",  
        "ps1.example.com:2222",  
    ]})
```

```
if job_name == "ps":  
    server.join()  
elif job_name == "worker":  
    ...
```

分布式训练系统（I）

- 底层实现：基于计算图的分布式切分



计算图系列

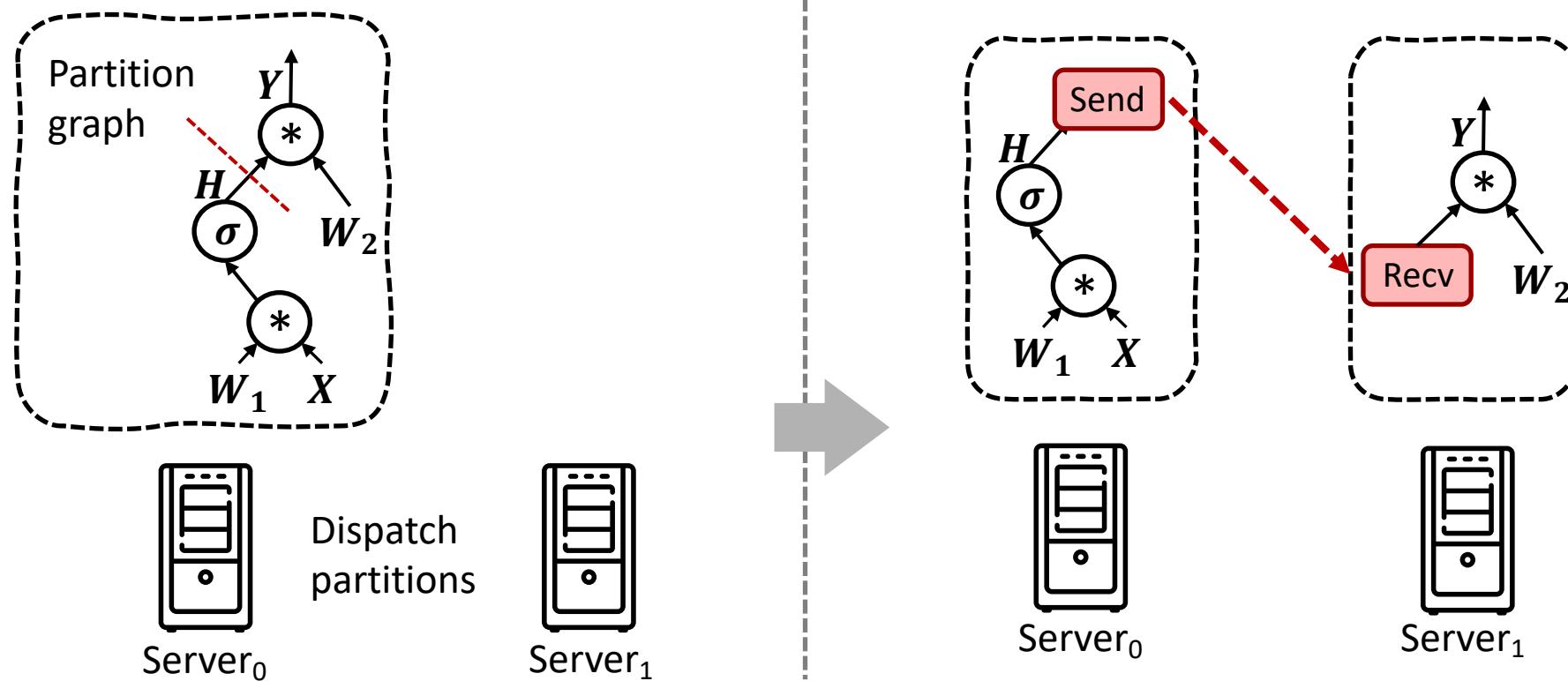
<https://space.bilibili.com/517221395/channel/collectiondetail?sid=765278>

 计算图系列 01. 内容介绍 01:55	【计算图】系列第一篇！计算图有哪些内容知识？ 123 10-6	 计算图系列 02. 什么是计算图 10:31	【计算图】系列第二篇！为什么AI框架都用计算图？什么是 117 10-8	 计算图系列 03. 跟自动微分什么关系？ 16:45	【计算图】系列第三篇！计算图跟微分什么关系？怎么用计 127 10-9
 计算图系列 04. 图优化与执行调度 10:30	【计算图】第四篇！图优化与执行调度！计算图优化！单算 109 10-10	 计算图系列 05. 怎么表示控制流 19:27	【计算图】第五篇！PyTorch 控制流如何实现？动静统一原 167 10-11	 计算图系列 06. 挑战和未来 06:42	【计算图】第六篇！计算图未来将会走向何方？ 133 10-12

分布式训练系统 (I)



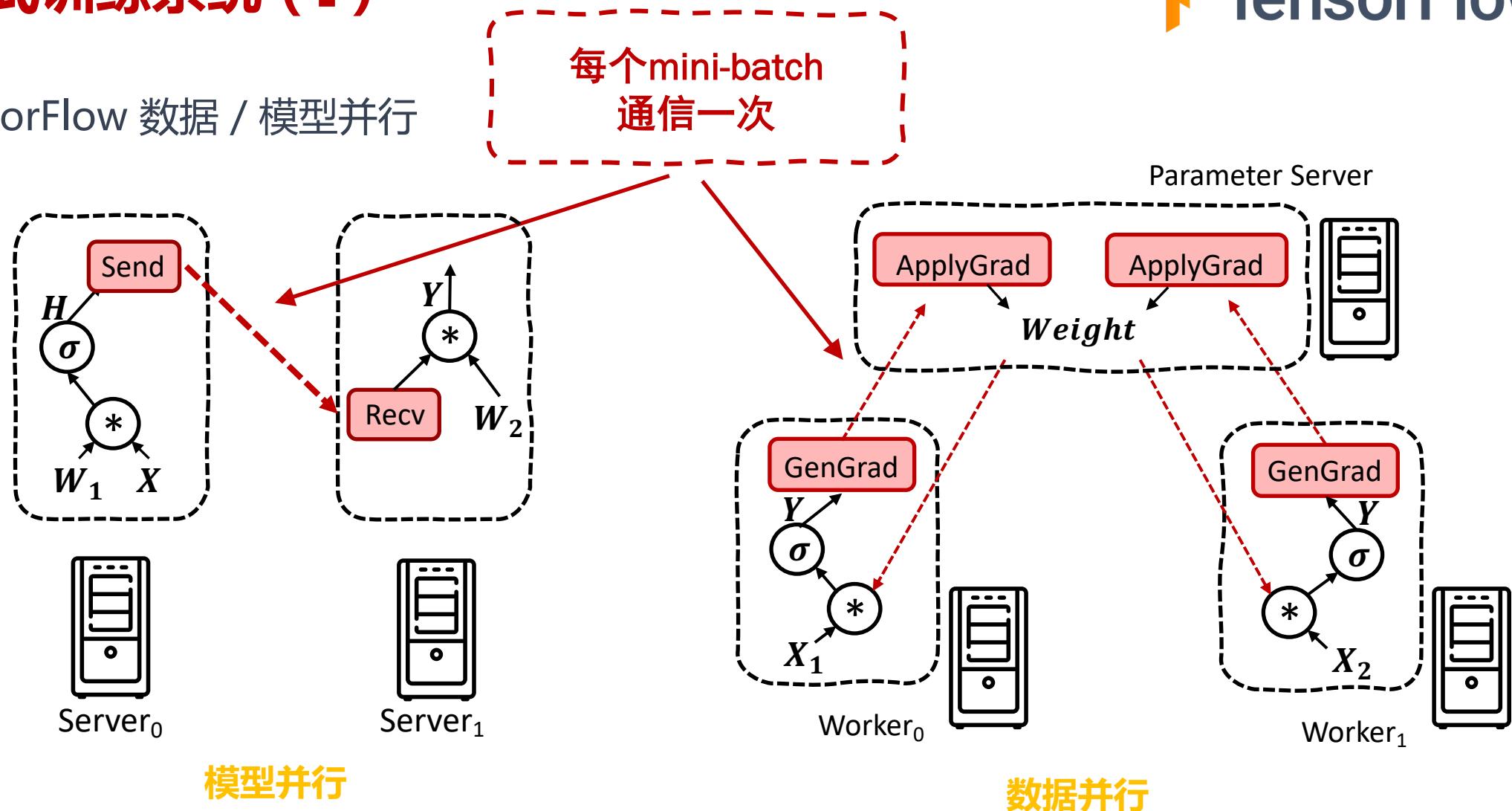
- 计算图跨节点切分



分布式训练系统 (I)



- TensorFlow 数据 / 模型并行



模型并行

数据并行

分布式训练系统 (II)

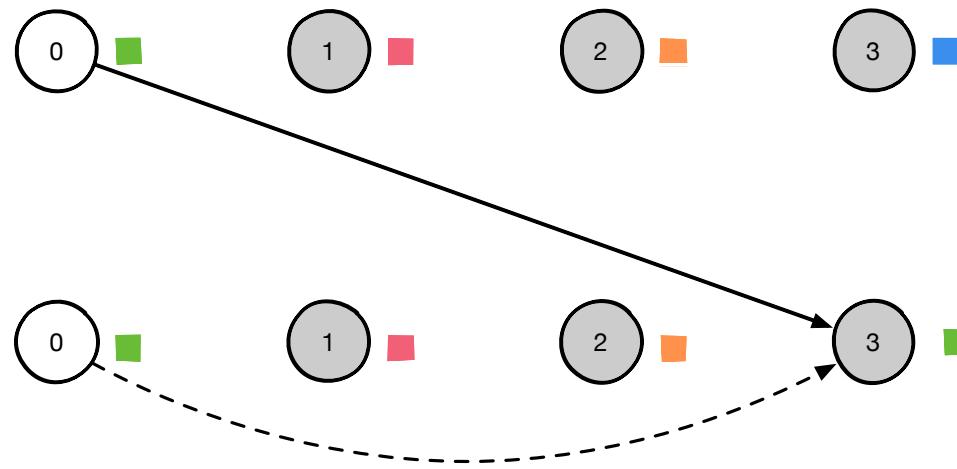


- PyTorch 同样包含 点对点通信 和 集中式通信

分布式训练系统 (II)

PyTorch

- PyTorch 同样包含 点对点通信 和 集合式通信



分布式训练系统 (II)



PyTorch 点对点通信 (同步)

可以实现用户指定的同步send/recv

例如：rank 0 send -> rank 1 recv

```
"""Blocking point-to-point communication."""

def run(rank, size):
    tensor = torch.zeros(1)
    if rank == 0:
        tensor += 1
        # Send the tensor to process 1
        dist.send(tensor=tensor, dst=1)
    else:
        # Receive tensor from process 0
        dist.recv(tensor=tensor, src=0)
    print('Rank ', rank, ' has data ', tensor[0])
```

分布式训练系统 (II)



PyTorch 点对点通信 (异步)

可以实现用户指定的异步send/recv

例如 : rank 0 send -> rank 1 recv

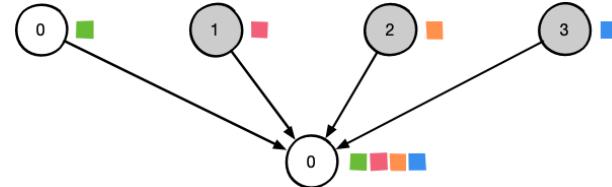
```
"""Non-blocking point-to-point communication."""

def run(rank, size):
    tensor = torch.zeros(1)
    req = None
    if rank == 0:
        tensor += 1
        # Send the tensor to process 1
        req = dist.isend(tensor=tensor, dst=1)
        print('Rank 0 started sending')
    else:
        # Receive tensor from process 0
        req = dist.irecv(tensor=tensor, src=0)
        print('Rank 1 started receiving')
    req.wait()
    print('Rank ', rank, ' has data ', tensor[0])
```

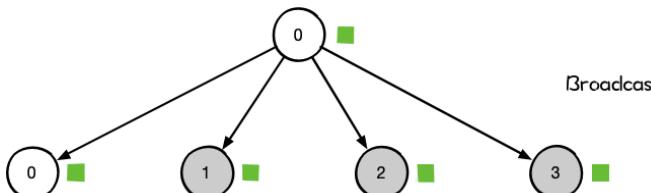
分布式训练系统 (II)

PyTorch

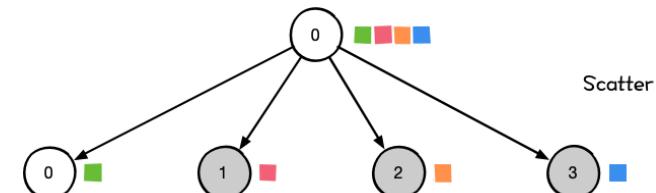
- PyTorch 集合式通信



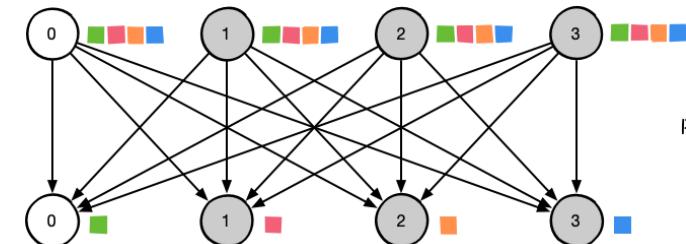
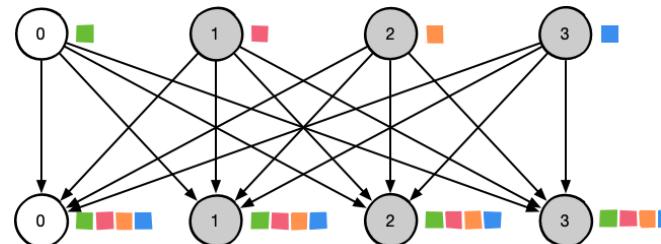
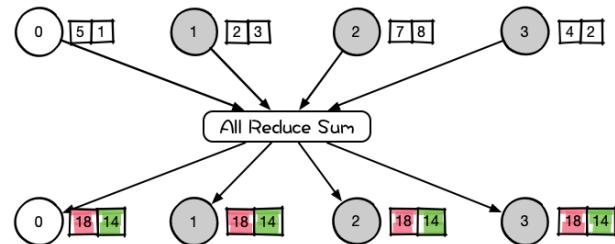
Gather



Broadcast



Scatter



PyTorch 集中式通信

多对多: All-Reduce

```
""" All-Reduce example."""
def run(rank, size):
    """ Simple point-to-point communication. """
    group = dist.new_group([0, 1])
    tensor = torch.ones(1)
    dist.all_reduce(tensor, op=dist.reduce_op.SUM, group=group)
    print('Rank ', rank, ' has data ', tensor[0])
```

分布式训练系统 (II)



- 示例：分布式MNIST

```
""" Gradient averaging. """
def average_gradients(model):
    size = float(dist.get_world_size())
    for param in model.parameters():
        dist.all_reduce(param.grad.data, op=dist.reduce_op.SUM)
        param.grad.data /= size
```

```
""" Distributed Synchronous SGD Example """
def run(rank, size):
    torch.manual_seed(1234)
    train_set, bsz = partition_dataset()
    model = Net()
    optimizer = optim.SGD(model.parameters(),
                          lr=0.01, momentum=0.5)

    num_batches = ceil(len(train_set.dataset) / float(bsz))
    for epoch in range(10):
        epoch_loss = 0.0
        for data, target in train_set:
            optimizer.zero_grad()
            output = model(data)
            loss = F.nll_loss(output, target)
            epoch_loss += loss.item()
            loss.backward()
            average_gradients(model)
            optimizer.step()
        print('Rank ', dist.get_rank(), ', epoch ',
              epoch, ': ', epoch_loss / num_batches)
```

分布式训练系统 (III)

1. 环境依赖

- 并行训练前 communication.init 接口初始化通信资源，自动创建全局通信组 WORLD_COMM_GROUP；

2. 数据分发

- dataset模块将数据集拆分为多份并循环采样的方式，采集batch大小的数据到各自的卡上；

3. 网络构图

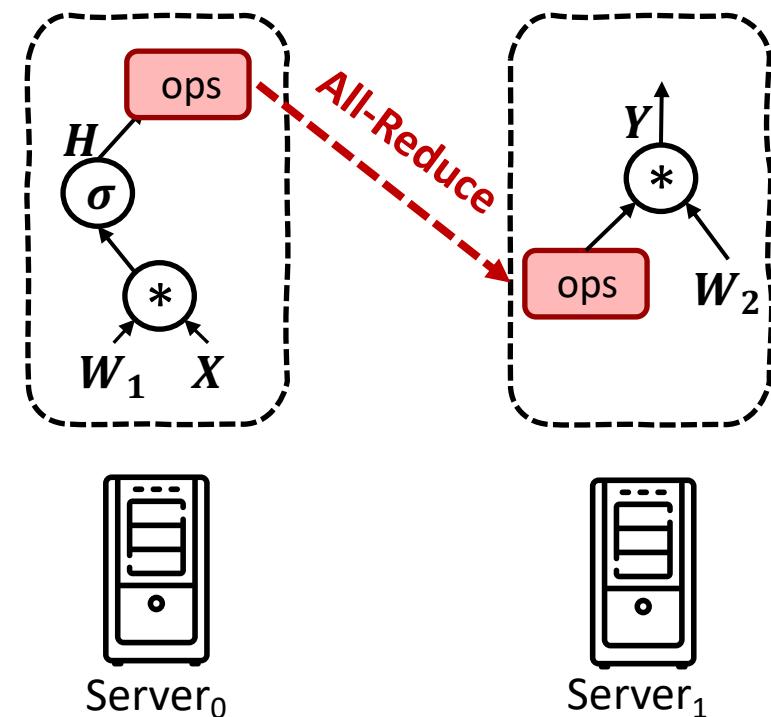
- 数据并行网络的书写方式与单机网络没有差别，保持相同的网络结构。DATA_PARALLEL和HYBRID_PARALLEL模式下通过broadcast达到权重广播的目的；

4. 梯度聚合

- MindSpore设置了mean开关，可以选择是否要对求和后的梯度值进行求平均操作；

5. 参数更新

- MindSpore实现的是一种同步数据并行训练。



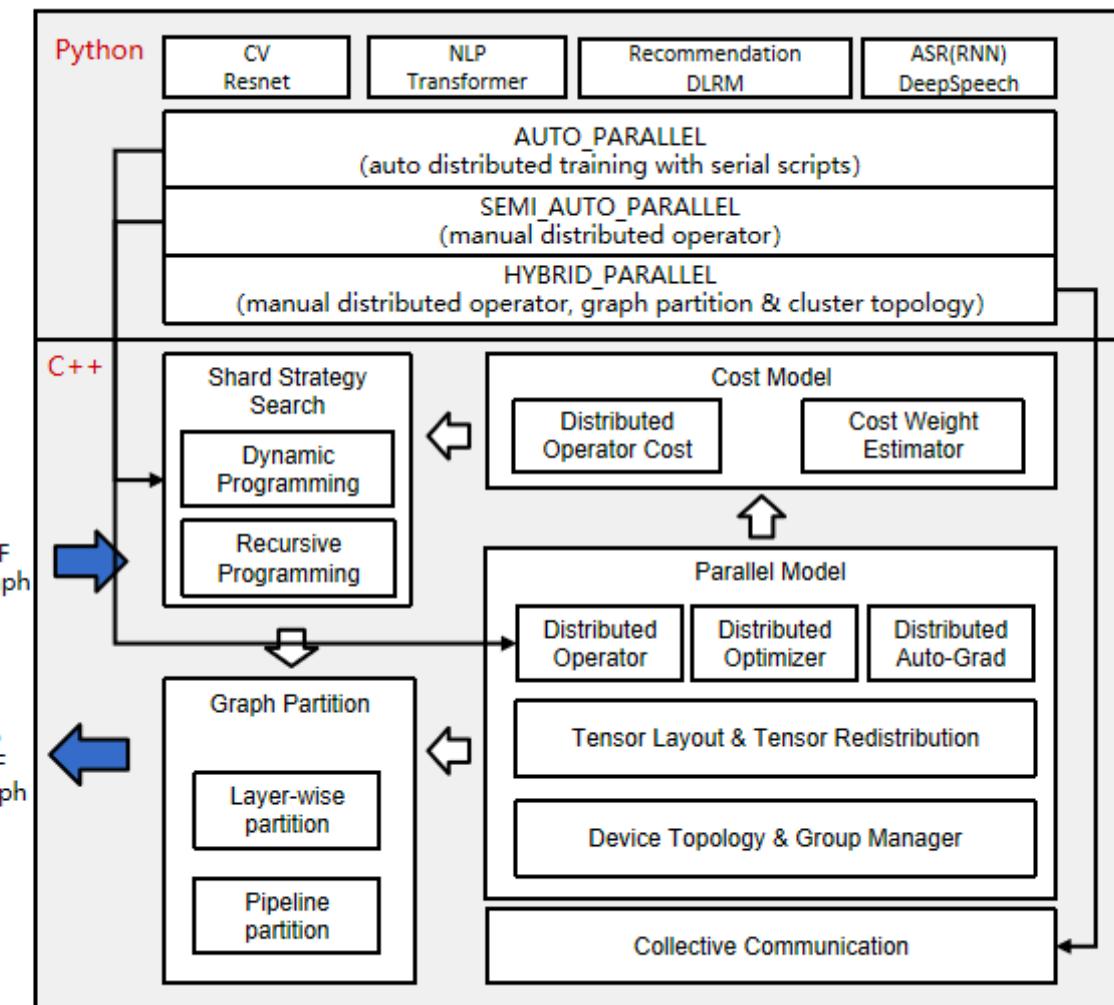
数据并行

分布式训练系统 (III)



• 自动并行原理

1. 张量排布模型 [tensor layout](#)
2. 分布式算子 [ops info](#)
3. 张量排布变换
4. 切分策略搜索算法 [auto_parallel](#)
5. 分布式自动微分



其它

- **数据读取与预处理**
 - 本地读取：预先推送数据片到worker
 - 分布式文件系统：框架内嵌并行读取功能，多worker分片读取自己的部分
- **容错处理和故障恢复**
 - 机器学习的内禀特性：训练数据丢弃不敏感（部分worker更新）
 - 模型checkpoint：save/load

思考题

- Q : 为什么模型训练通常需要分布式进行，而分布式模型预测并不常见？
- 计算模式不同：
 - 训练需要各个worker保持通信，从而协调统一地更新模型参数；
 - 预测中的模型参数是固定的，各个worker分别使用只读副本，无需相互通信协调；

本章小结

1. 单服务器执行从串行处理到并行处理，硬件架构从SISD到MIMD发展，实际上目前应用最广的是支持SIMT编程的GPU硬件架构。
2. 基于内嵌分布式策略的训练系统：针对TensorFlow/MindSpore为代表的基于计算图的AI框架，根据内置规则，自动完成分布式并行训练
3. 基于提供通信原语分布式训练系统：针对以解释执行 AI 框架 PyTorch ，能支持用户自定义的灵活分布式策略



BUILDING A BETTER CONNECTED WORLD

THANK YOU

Copyright©2014 Huawei Technologies Co., Ltd. All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.