Pytorch DDP（DistributedDataParallel）术语

1. 术语：

Node：server / 机器节点

Process: 进程

World: `world` as a group containing all the processes

World size: 所有节点上的进程总数量

Rank： 进程ID/index， [0, World_size - 1]

Local Rank：当前 node 上进程 ID/index，[0, nproc_per_node - 1]

GPUs & devices: 指显卡设备。经常的显卡数量等于进程数量但是在 model parallel 下不是：

> The recommended way to use DDP is to spawn one process for each model replica, where a model replica can span multiple devices. DDP processes can be placed on the same machine or across machines, but GPU devices cannot be shared across processes. This tutorial starts from a basic DDP use case and then demonstrates more advanced use cases including checkpointing models and combining DDP with model parallel.

Data parallel: 类似 DP（ `DataParallel` ） 的方式，每个 GPU 上一个 model replica， 每一个进行 loss 计算，其中一个进行梯度更新。

Model parallel：单张卡无法塞进整个 model，model replica 分布在几个卡就是 Model parallel。Model parallel 是用户代码定义的，不是 `DistributedDataParallel` 自动完成的

2. 启动 DDP

```
torchrun --nnodes=2 --nproc_per_node=8 --rdzv_id=100 --rdzv_backend=c10d --
rdzv_endpoint=$MASTER_ADDR:29400 elastic_ddp.py
```

torch.distributed.launch 是旧方式，应用 torchrun

> ## torch.distributed.launch vs. torchrun
>
> Both `torch.distributed.launch` and `torchrun` are used for distributed training, but `torchrun` is newer and generally simpler to use.
>
> `torch.distributed.launch`: It's an older utility and requires you to pass the `--local_rank` argument manually to your script.
>
> `torchrun`: Provides a simpler interface and doesn't require manual handling of `--local_rank`.

3. 其他资料

- GETTING STARTED WITH DISTRIBUTED DATA PARALLEL
- SINGLE-MACHINE MODEL PARALLEL BEST PRACTICES
- nanoGPT/train.py
- HF blog DDP vs DP
- Pytorch 分布式训练 (DP, DDP)

4. 总结 DDP 之优点

多进程，多机器，支持模型分片。（DP 做不到）

在单机器多卡环境下，因 DDP 及时（每一层流式的同步无需整体的同步梯度，ring-reduce 减少通信量）同步梯度数据，且只同步梯度数据，其他模型参数更新、优化器参数更新在每一个机器上自然“计算” 同步，无需 “通信“ 同步。 （所以一般推荐 DDP，比 DP 更快即使在单机）

实验看 DDP 比 DP 快 5+ 倍，每个 GPU 几乎 100% 的最大载荷，且显存占用也都很均衡。