4.2第一次协会技术交流会

显卡

英伟达

1. 迭代

架构名称	Volta-伏特	Turing-图灵	Ampere-安培	Hopper-赫柏
发布时间	2017	2018	2020	2022
核心参数	80个SM，每个SM包括32个FP64+64个Int32+64个FP32+8个Tensor Cores	102核心92个SM，SM重新设计每个SM包含64个Int32+64个FP32+8个Tensor Cores	108个SM，每个SM包含64个FP32+64个INT32+32个FP64+4个Tensor Cores	132个SM，每个SM包含128个FP32+64个INT32+64个FP64+4个Tensor Cores
特点	NVLink2.0，Tensor Cores第一代，支持AI运算	Tensor Core2.0，RT Core第一代	Tensor Core3.0，RT Core2.0 NVLink3.0，结构稀疏性矩阵MIG1.0	Tensor Core4.0，NVlink4.0结构稀疏性矩阵MIG2.0
纳米制程	12nm 211亿晶体管	12nm 186亿晶体管	7nm 283亿晶体管	4nm 800亿晶体管
代表产品	V100TiTan V	T4，2080TI RTX 5000	A100、A800 A30系列	H100、H800

2. 关键指标
    a. 显存
    b. CUDA Core：FP32计算单元数量
    c. Tensor Core：用矩阵乘加的张量计算单元，将整个矩阵载入寄存器

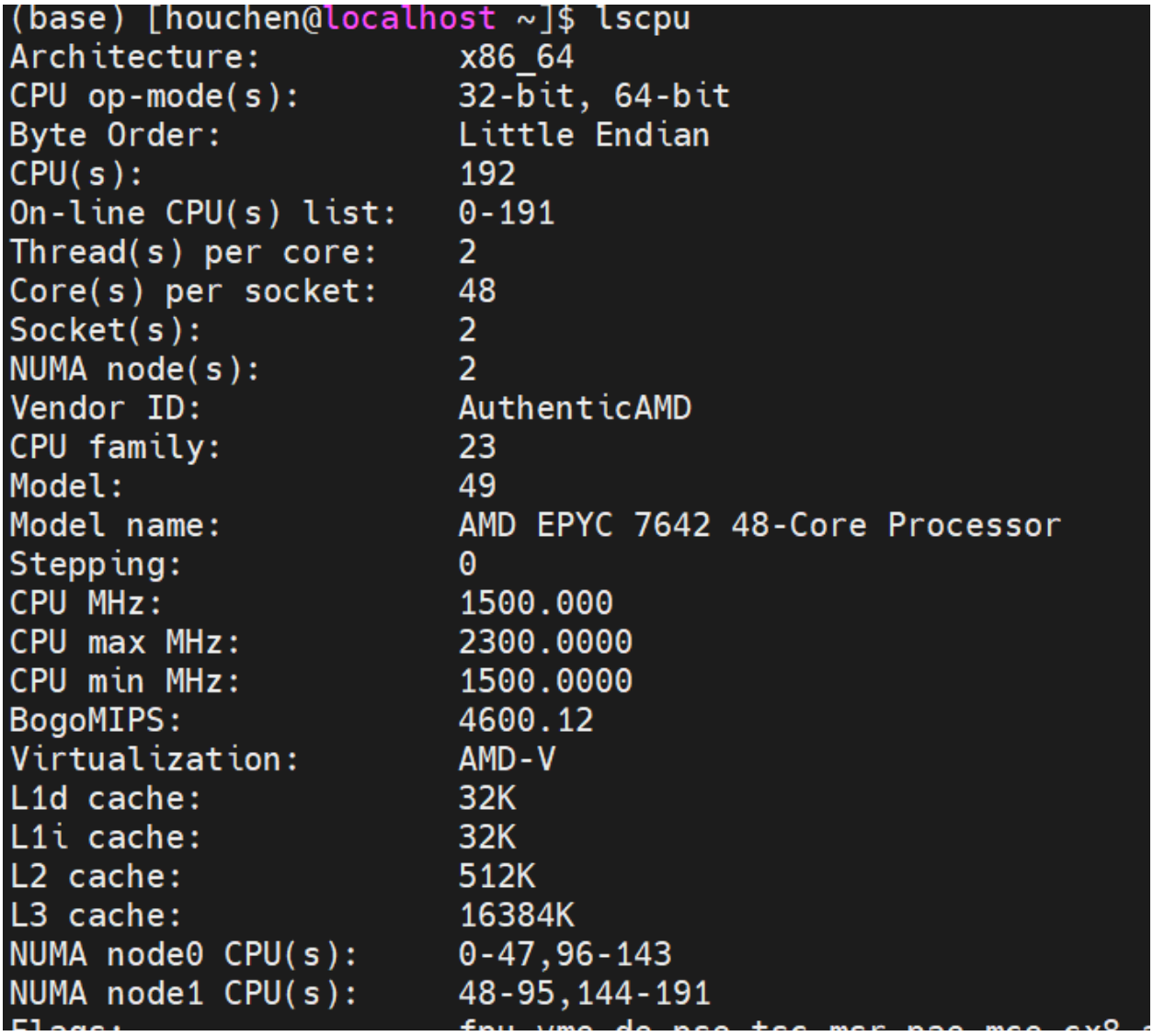
性能参数	V100 PCIe	A100 80GB PCIe	A800 80GB PCIe	H100 80GB PCIe
微架构	Volta	Ampere		Hopper
FP64	7TFLOPS	9.7TFLOPS		26 TFLOPS
FP32	14TFLOPS	19.5TFLOPS		51 TFLOPS
FP16 Tensor Core		312TFLOPS		756.5 TFLOPS
INT8 Tensor Core	62 TOPS	624 TOPS		1513 TOPS
GPU 显存	32/16GB HBM2	80GB HBM2e		80GB
GPU 显存带宽	900 GB/s	1935GB/s		2TB/s
最大热设计功耗 (TDP)	250 瓦	300 瓦		300-350W
多实例 GPU		最多 7 个 MIG 每个 10GB		
外形规格		PCIe 双插槽风冷式 或单插槽液冷式		PCIe 双插槽风冷式
互连技术	NVLink：300 GB/s PCIE：32 GB/s	搭载 2 个 GPU 的 NVIDIA" NVLink" 桥接器：600GB/s PCIe 4.0：64GB/s	搭载 2 个 GPU 的 NVIDIA" NVLink" 桥 接器：400GB/s PCIe 4.0：64GB/s	NVLink：600GB/s PCIe 5.0：128GB/s
服务器选项		搭载 1 至 8 个 GPU 的合作伙伴认证系统和 NVIDIA 认证系统		

3. PCIE5.0单向带宽63g,NVLink单向300G，因为NVLink是自家标准，不考虑功耗可以10000G

others

1. 华为昇腾910B/920
    a. 910B约等于A100
2. 谷歌TPU:Tensor Processing Unit
    a. V4=A100*1.7倍

CPU



1. NUMA：非一致性内存访问, 能通过**绑核（进程绑定在某个 CPU 或 NUMA Node 的内存上执行）**实现访存加速
2. Socket：主板上的CPU插槽

并行训练

DP：

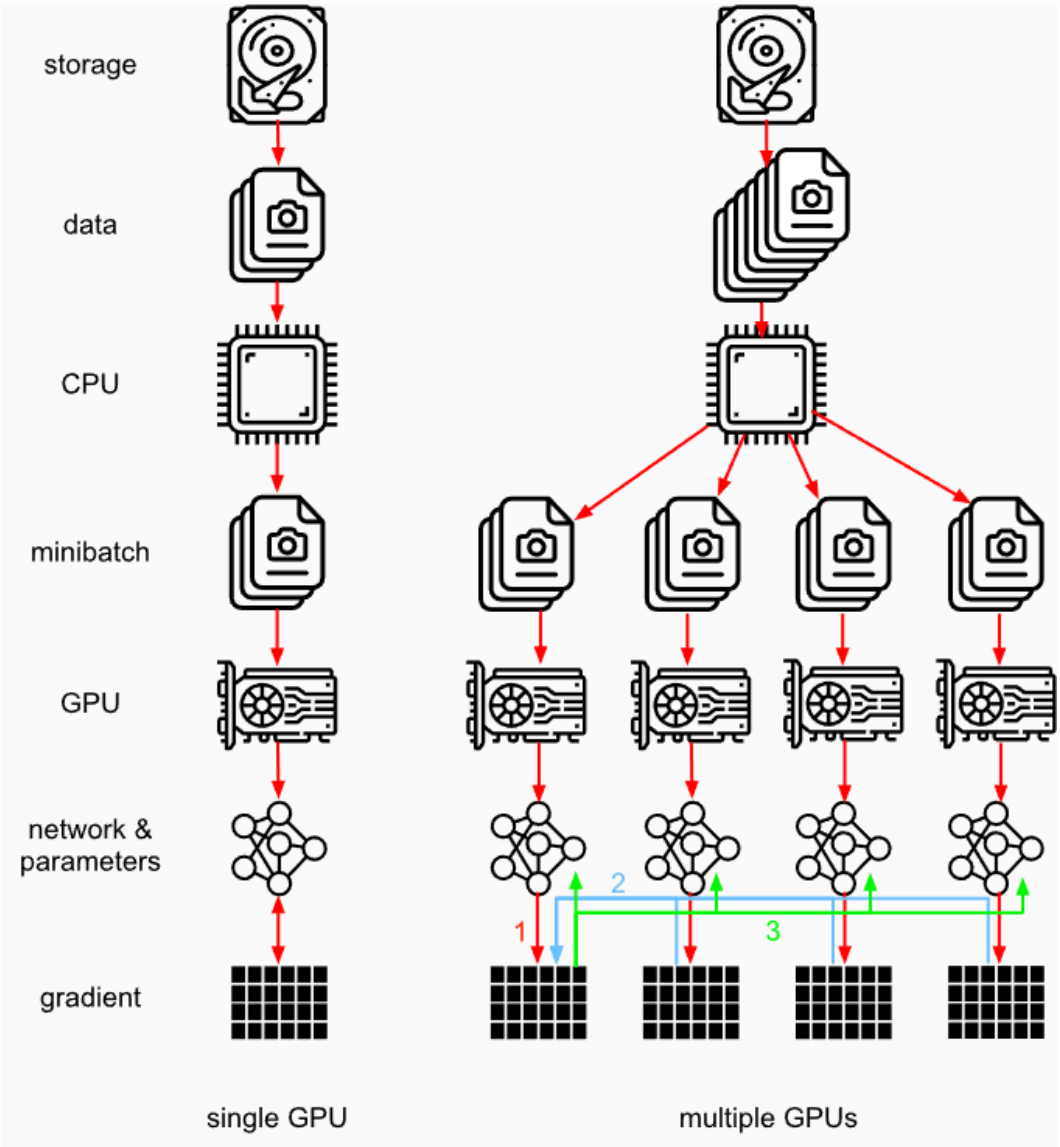
1. 每个 GPU 复制一份模型，将一个batch的样本分为多个小份输入各个模型并行计算

2. 使用简单, 直接调用

```
>>> net = torch.nn.DataParallel(model, device_ids=[0, 1, 2])
>>> output = net(input_var)  # input_var can be on any device, including CPU
```

3. 只能单机多卡

4. device[0] 负责整合梯度，更新参数，其他节点负责前向传播，反向传播算梯度，然后将梯度传给主卡，主卡将**梯度相加÷卡数，然后自己更新参数**



5. 缺点

a. 通信开销大，随着卡数增加，通信代价线性增长

b. 更新后的模型参数需要再次从主 GPU 广播到其他 GPU

c. 不能多机多卡

d. 单进程控制

DDP

1. 使用稍微复杂

```
import torch
import torch.distributed as dist
import torch.multiprocessing as mp
import torch.nn as nn
import torch.optim as optim
import os
from torch.nn.parallel import DistributedDataParallel as DDP

# 示例函数，每个进程都会执行这个函数
def example(rank, world_size):
    # 初始化进程组，设置了进程间通信的方式，并且定义了每个进程在进程组中
    dist.init_process_group("gloo", rank=rank, world_size=wor
    # 创建本地模型
    model = nn.Linear(10, 10).to(rank) # to函数一般用在将数目转移
    # 构建DDP模型
    ddp_model = DDP(model, device_ids=[rank])
    # 定义损失函数和优化器
    loss_fn = nn.MSELoss()
    optimizer = optim.SGD(ddp_model.parameters(), lr=0.001)

    # 前向传播
    outputs = ddp_model(torch.randn(20, 10).to(rank))
    labels = torch.randn(20, 10).to(rank)
    # 反向传播
    loss_fn(outputs, labels).backward()
    # 更新参数
```

```
    optimizer.step()

# 主函数
def main():
    world_size = 2
    # 一定要用mp.spawn拉起整个训练,多次调用拉起整个训练
    mp.spawn(example,
        args=(world_size,),
        nprocs=world_size,
        join=True)

if __name__=="__main__":
    # 当使用c10d的默认"env"初始化模式时，需要设置的环境变量
    # 使用单机多卡就这样设置
    os.environ["MASTER_ADDR"] = "localhost"
    os.environ["MASTER_PORT"] = "29500"
    main()
```

```
Tue Apr  2 10:46:28 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100 80G...  Off  | 00000000:01:00.0 Off |                    0 |
| N/A   40C    P0    70W / 300W |   5840MiB / 81920MiB |    100%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A100 80G...  Off  | 00000000:41:00.0 Off |                    0 |
| N/A   41C    P0    69W / 300W |   5001MiB / 81920MiB |    100%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     38875      C   ...s/Pytorch_test/bin/python     4999MiB |
|    0   N/A  N/A     38876      C   ...s/Pytorch_test/bin/python      839MiB |
|    1   N/A  N/A     38876      C   ...s/Pytorch_test/bin/python     4999MiB |
+-----------------------------------------------------------------------------+
```

2. DDP 支持 Ring AllReduce，其通信成本是相对恒定的，与 GPU 数量无关。
3. 每个 GPU 上的梯度被同时汇总并分发给所有 GPU，这样每个 GPU 都会得到**完整的梯度信息**，然后**自己更新参数**。
4. 缺点
    a. 还是数据并行，当**一块卡装不下模型**时就废了，所以有了后来的magatron和deepspeed

Deepspeed

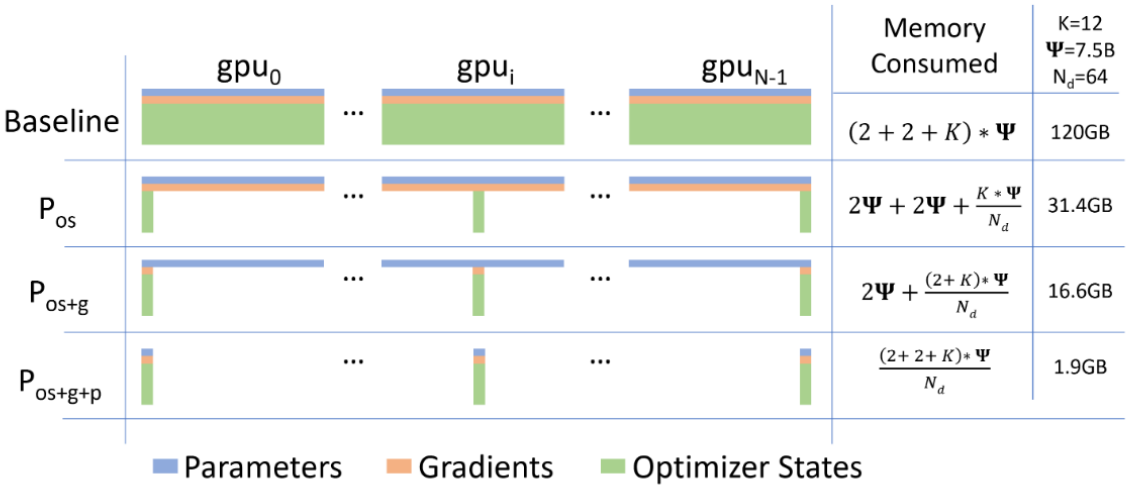
https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/

1. 模型训练显存占用分析，假设参数量为a
    a. 16a = 参数保存2a+梯度2a+（Adam的momentum和variance 8a+参数的FP32版本4a）
2. ZeRO的三个stage
    a. https://zhuanlan.zhihu.com/p/394064174

    ZeRO-1：分割`Optimizer States`；

    ZeRO-2：分割`Optimizer States`与`Gradients`；

    ZeRO-3：分割`Optimizer States`、`Gradients`与`Parameters`



3. 再不够还可以offload到host，再不够还可以从主存offload到硬盘的对换区
    a. 一般为了不让缓存到硬盘，会在内存中写死不让换出，然后将id存在一个list，等需要加载到GPU时用hook勾进去，而且会offload到绑核后的近端内存
4. 4路模型并行，4路数据并行