大模型-AI集群 (存)

CheckPoint 存储优化

ZOMI

大模型业务全流程

数据 & 模型算法

2. 数据处理

- 开源数据
- 数据预处理
- 向量数据库

3. 模型算法

- LLM模型架构
- 多模态一切皆Tokens

模型训练 & 微调

4. 模型训练

- 混合精度
- 梯度检查
- 梯度累积
- …

5. 分布式并行

训练集群稳定性

6. 模型微调

- 全参微调
- 低参微调
- 指令微调

模型验证 & 推理部署

7. 模型验证

- 下游任务
- 测评标准
- bench mark

8. 推理与智能体

- 量化压缩
- 推理加速
- 9. Agent 智能体

集群算力准备

1. AI 集群建设

- 计算、存储、网络
- AI 集群机房建设
- AI 集群上线与运维

关于本内容

1. 内容背景

- AI 集群 + 大模型

2. 具体内容

- **数据存储现状和场景：**存储软件类型、存储硬件类型的发展
- **大模型对存储的挑战：**存储性能指标、存储遇到大模型挑战与新机会点
- **大模型训练CKPT优化：**大模型训练过程、CKPT过程分解、CKPT优化
- **大模型时代对存储的思考：**什么样的存储架构才是AI大模型时代的选择？

Why key point to Checkpoint ?

1. LLMs 中训练数据加载开销远小于整个训练过程开销，该优化收益有限（而且可能影响用户使用习惯和降低易用性），因此很多存储优化方案专门针对Checkpoint。

1. 大模型CKPT 保存流程

频繁保存CKPT原因

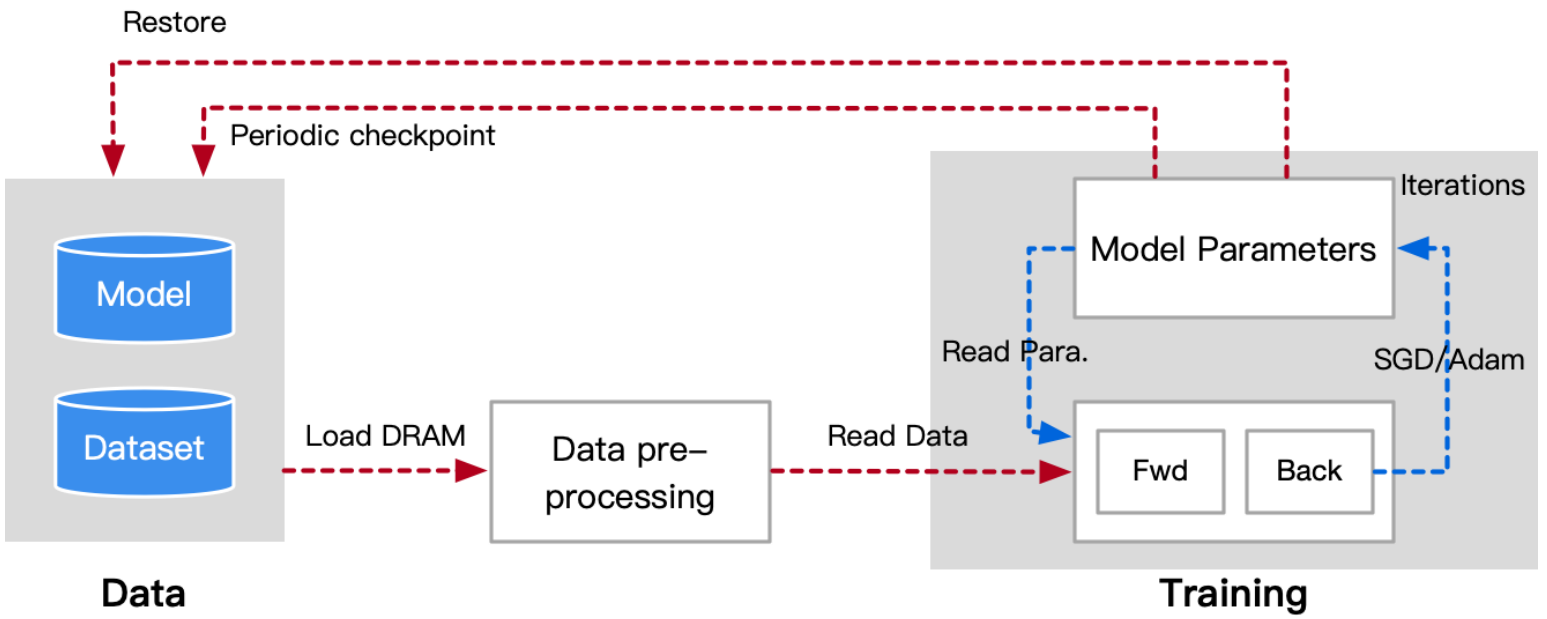
- 大模型在训练过程中，可能会遇到硬件故障、系统问题、网络错误、液冷供给等问题。千卡AI集群 2~3天，频繁中断会导致大模型训练难以丝滑持续进行，因此需要通过 Checkpoint（保存模型权重参数）来保存和恢复进度。
- 对于百/千亿参数 LLMs 大模型，Checkpoint 时间开销从分钟级到小时级不等（Checkpoint 存储 read/write 耗时与模型大小成正比）。Save Checkpoint 时大模型训练任务需要暂停。
- 一旦大模型训练过程出现中断，之前迭代的 Epoch 次数在恢复时需要重新计算，通常会花费数小时的时间。大模型训练过程一般使用 千卡规模 AI 集群，因此总体损失数千个 NPU 卡时间。

Disaster

- 大模型训练过程频繁保存和加载 Checkpoint（CKPT）都是个灾难
- Checkpoint 配置的频率，决定了 AI 集群中故障恢复后的重训成本

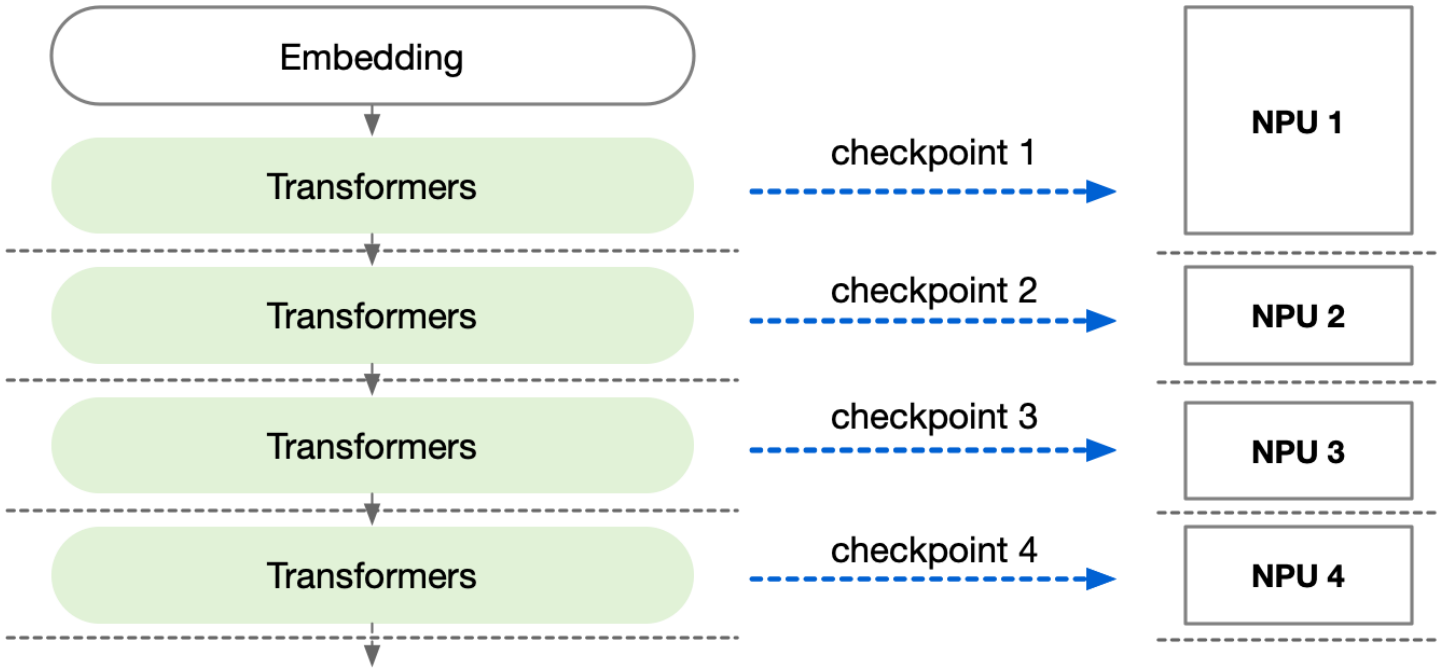
```
# trainging Epoch
for each in epoch:
    # List files in the datasets
    # pre-processing of datasets
    for mbs in batch:
        # Read files of the batch
        # Training
        pass

    checkpoint.save()
```



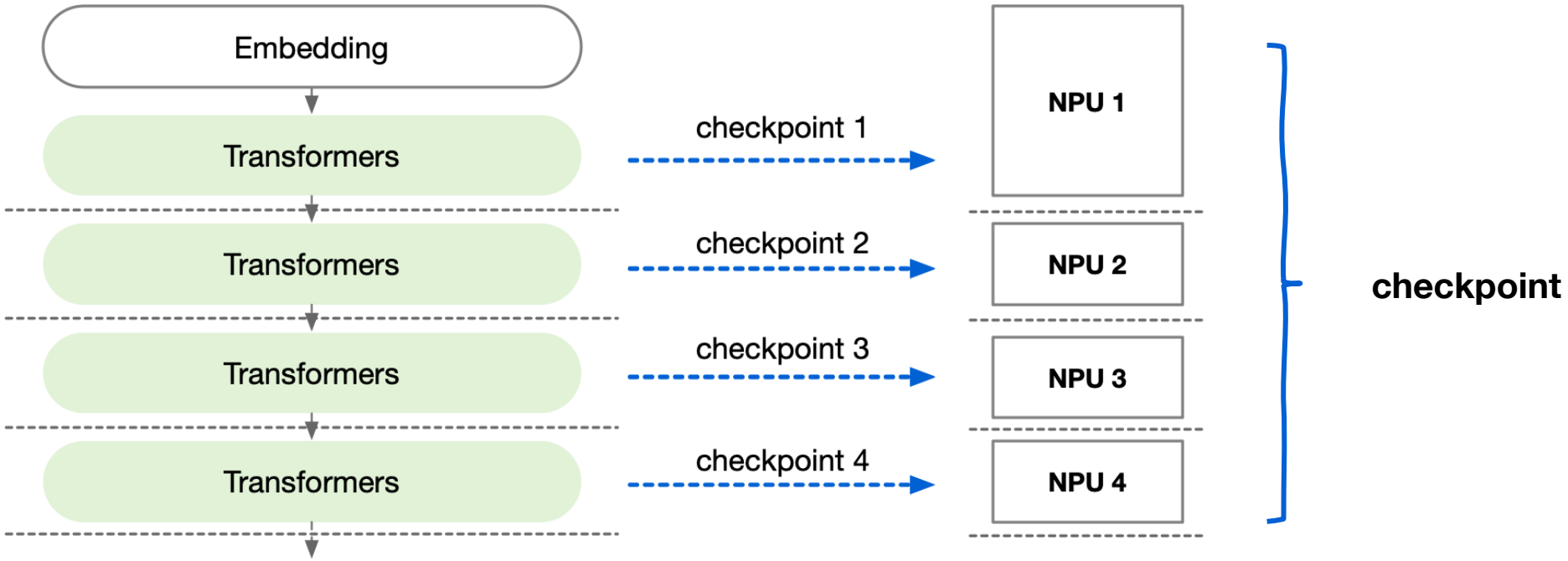
Why Disaster ?

- 大模型因为模型参数量太大，需要切分到不同 NPU，每个 NPU 有独立的模型权重参数。
  - **Save**：保存一次完整 Checkpoint 需要聚合所有 NPU 的权重信息，然后再回传对象存储服务器。
  - **Write**：加载一次 Checkpoint 需要把聚合后的模型权重按照 AI 集群分布进行切分，逐个 NPU 加载。



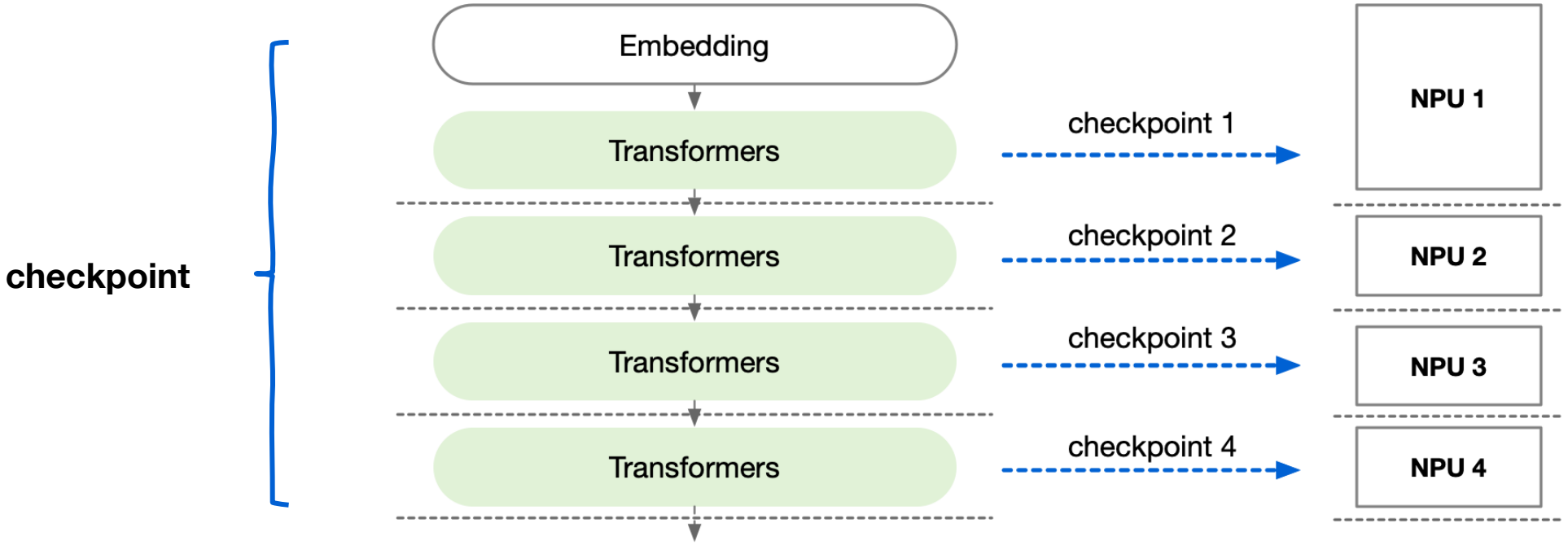
Why Disaster - Save

- 大模型因为模型参数量太大，需要切分到不同 NPU，每个 NPU 有独立的模型权重参数。
  - **Save**：保存一次完整 Checkpoint 需要聚合所有 NPU 的权重信息，然后再回传对象存储服务器。
  - **Write**：加载一次 Checkpoint 需要把聚合后的模型权重按照 AI 集群分布进行切分，逐个 NPU 加载。



Why Disaster - Write

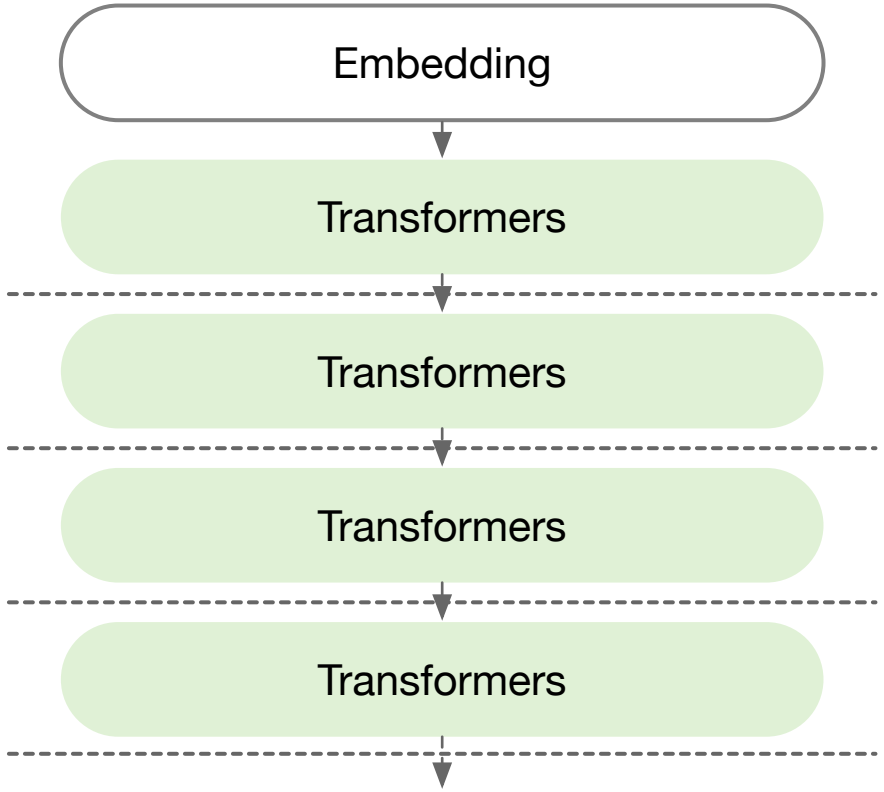
- 大模型因为模型参数量太大，需要切分到不同 NPU，每个 NPU 有独立的模型权重参数。
  - **Save**：保存一次完整 Checkpoint 需要聚合所有 NPU 的权重信息，然后再回传对象存储服务器。
  - **Write**：加载一次 Checkpoint 需要把聚合后的模型权重按照 AI 集群分布进行切分，逐个 NPU 加载。



Save and Write case?

1. 什么时候应该完整保存 Checkpoint、什么时候直接保存单个 NPU Checkpoint？
2. 加载时什么时候对完整 Checkpoint 进行切分后加载，什么时候直接加载？

Save and Write case? 以千亿模型为例



- 数据湖保存分片的 CKPT 和完整的 CKPT
- 程序定期保存分片 CKPT，定期汇聚分片 CKPT
- 如果没有节点坏掉，中断恢复直接拉起分片存储的 CKPT，便于加载
- 如果节点坏掉，使用完整的 CKPT 根据新的分布式策略进行切分后重新拉起

2. 大模型CKPT 优化分析

Checkpoint 文件分析

- Checkpoint 主要存储是网络模型状态的元数据
  - 网络模型权重参数 Model parameters
  - 网络模型梯度参数 Model gradients
  - 优化器状态信息 Optimizer Status
  - 训练状态信息 Training Status
  - 配置信息 Configure data

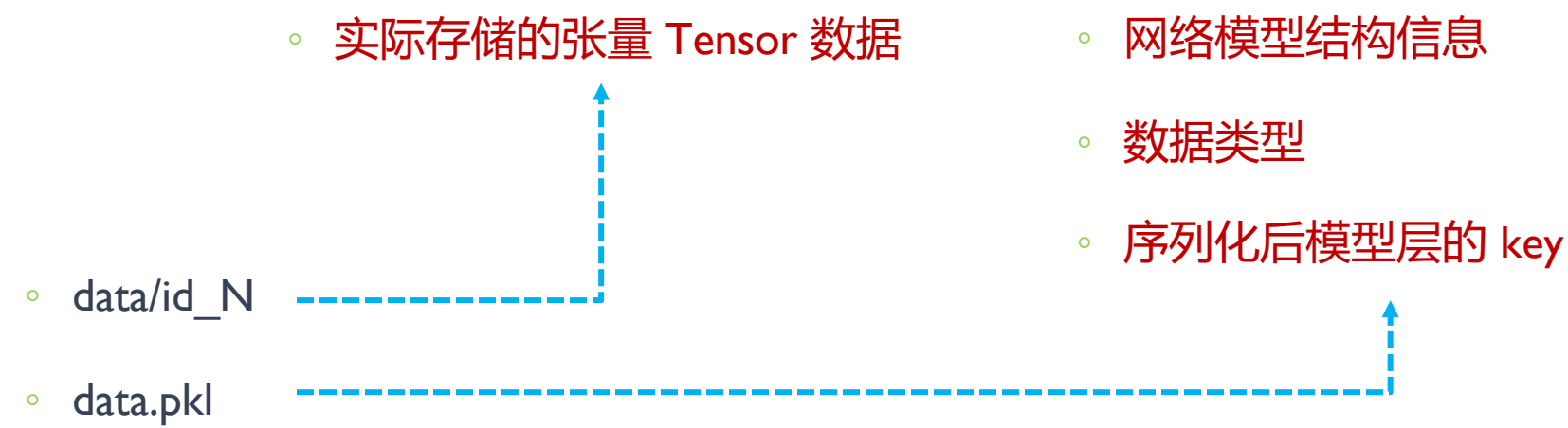
- Checkpoint 用途：故障快速恢复、二次训练、模型微调
- Checkpoint 保存加载：torch.save() and torch.load()

torch.save() 过程

- 网络模型在训练过程中会间隔一定时间保存一次 Checkpoint，用来记录当前训练的状态信息。
- save() 数据序列化过程主要步骤：
  - pickler 元数据序列化
  - save 存储数据
    - 实际存储的张量 Tensor 数据
    - 网络模型结构信息
    - 数据类型
    - 序列化后模型层的 key
  - data.pkl
  - data/id_N

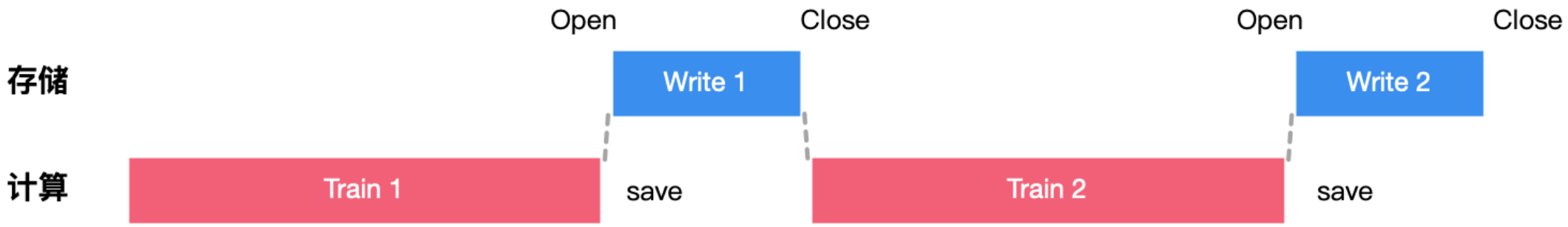
torch.load() 过程

- save() 的逆操作过程，数据根据 I/O 顺序读，不做过度的分析。
- load() 数据反序列化过程主要步骤：
  - pickler 元数据反序列化
  - read 读取 key 对应的value 数据



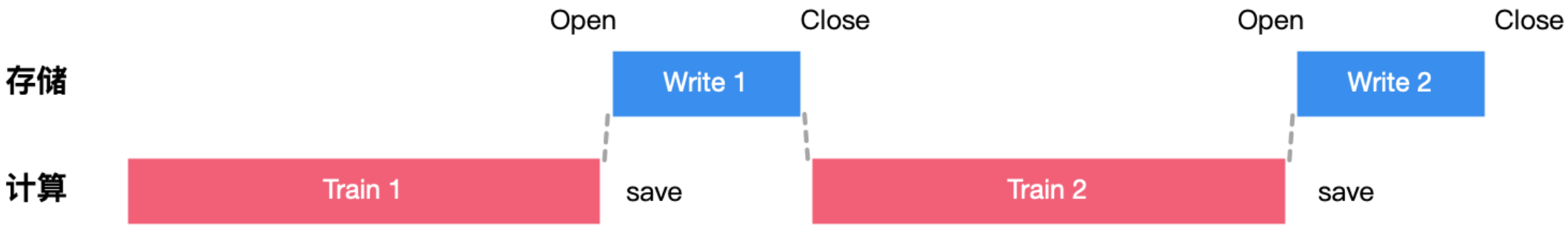
训练时候 Checkpoint 过程

- 优化 Checkpoint 保存和加载过程，缩短 Checkpoint 耗时，减少训练中断时间。



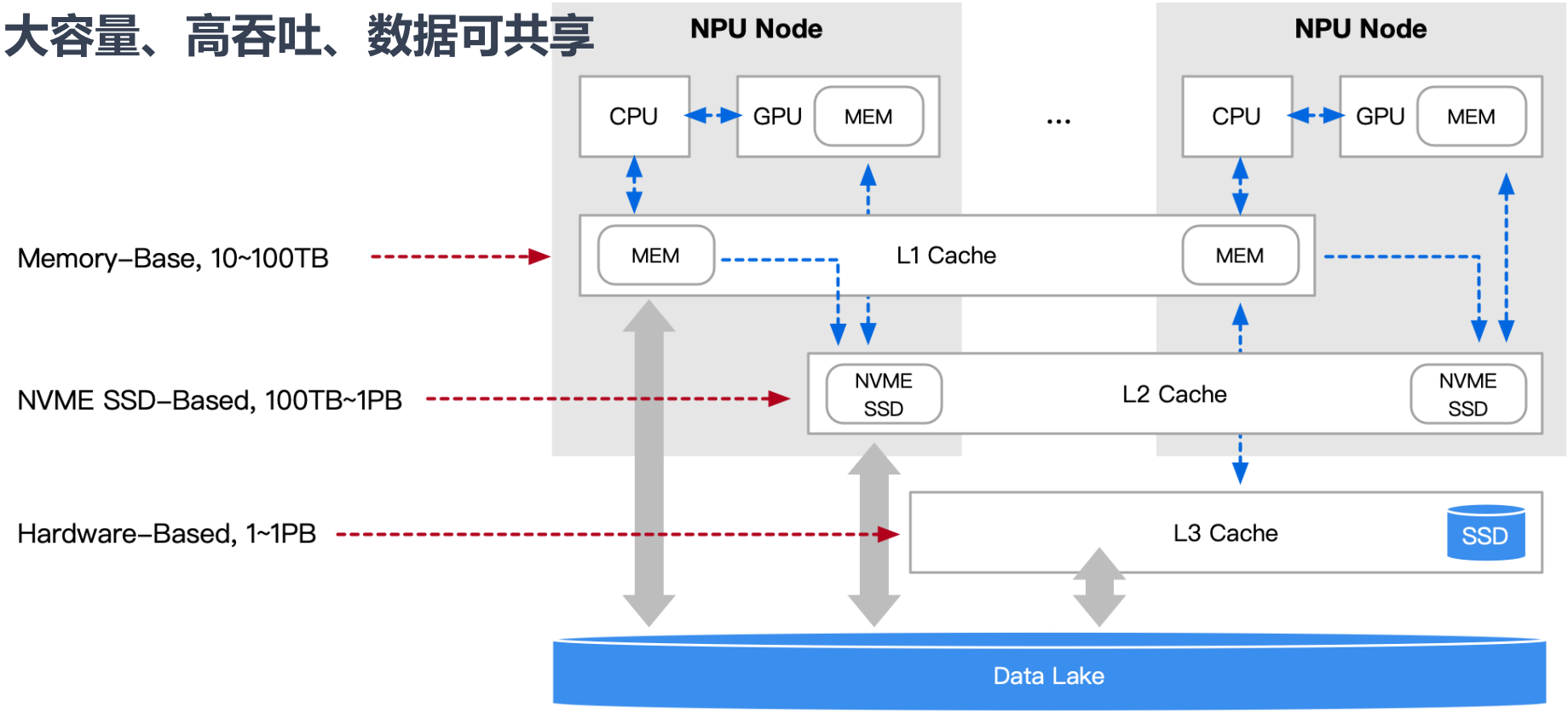
训练 Checkpoint 优化方案

优化思路	优化路径	成熟度
将CKPT模型数据存放在数据湖	save() and/or load()	*****
CKPT的save过程从同步到异步	save()	**
CKPT流式分块存储	save()	**
多文件加速聚合	save() and/or load()	****
本地内存缓存，同步写内存	save() and/or load()	****
数据拷贝过程使用零拷贝	save() and/or load()	



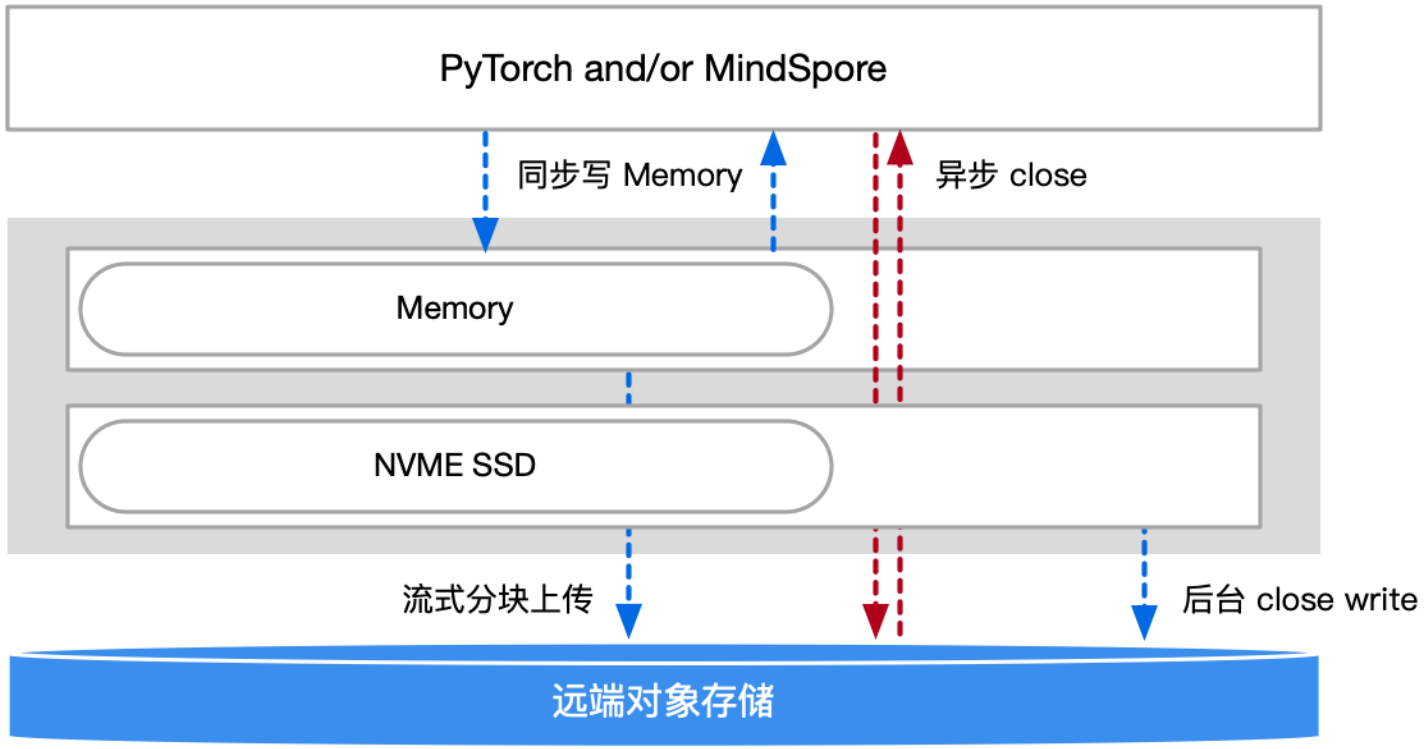
将CKPT模型数据存放在数据湖

- 单节点：每个训练节点使用更快的NVME SSD代替HDD进行存储，再分层回传数据湖；
- 多节点：一个节点挂了，其CKPT被其他节点代替继续进行训练；
- **优点：大容量、高吞吐、数据可共享**



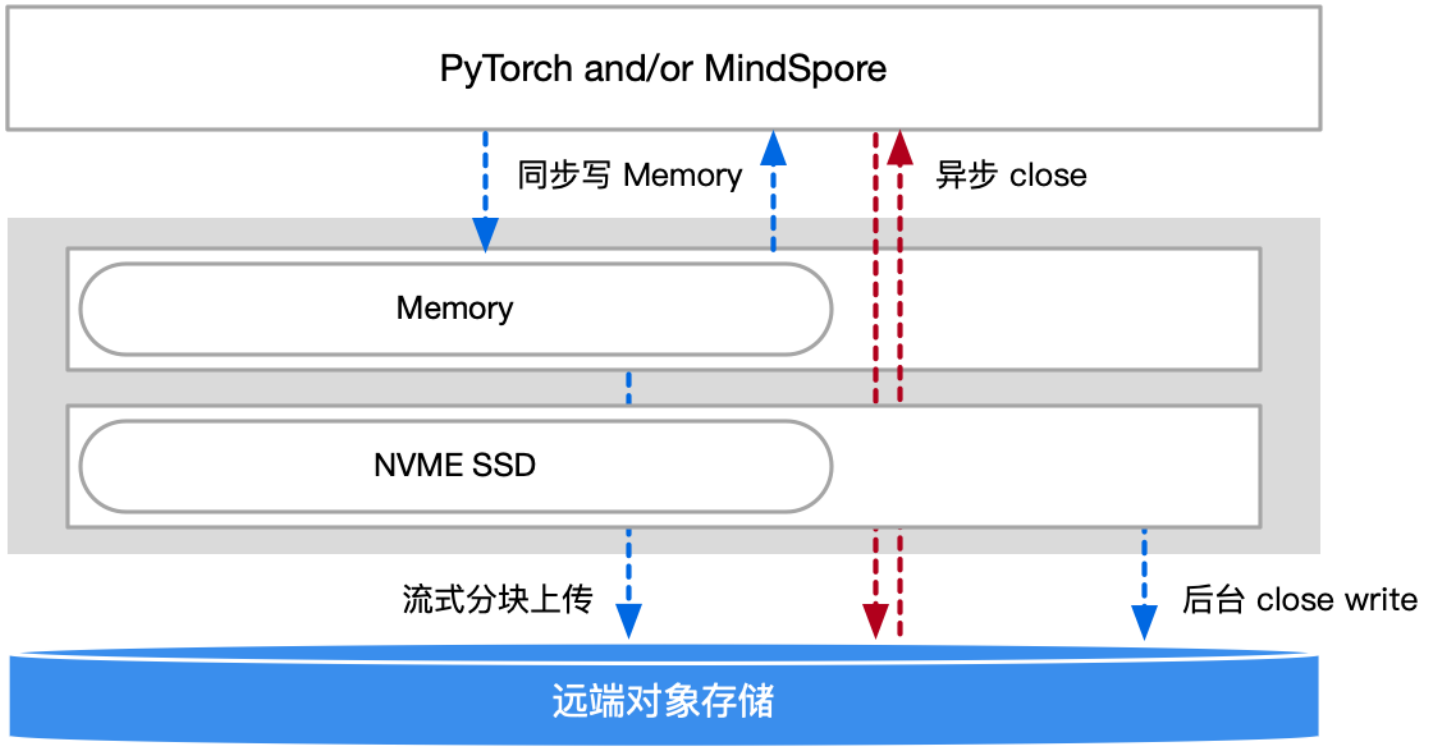
2 save() 过程由同步到异步

- 方式：将 save() 过程与下一轮的训练迭代并行，计算掩盖存储耗时。
- 问题：系统崩溃、节点挂掉时需要处理数据部分写入逻辑。
- 方案：CKPT 添加 success 标记位便于恢复



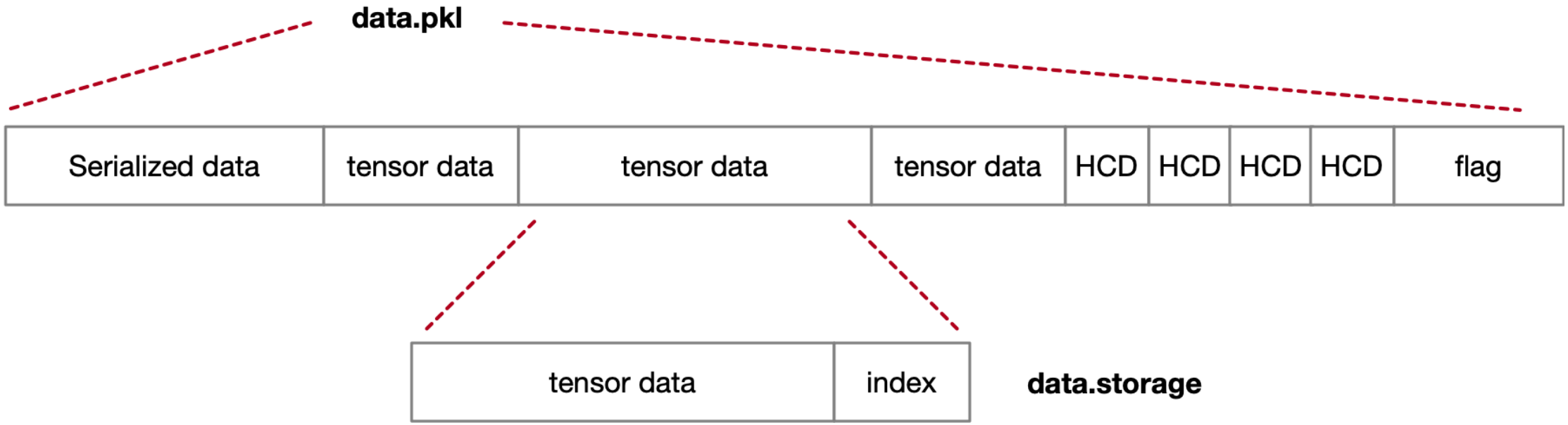
3 CKPT流式分块存储

- 采用流式 & 分块上传的方式，无需等 Checkpoint 全部写完到 Memory/SSD 就开始向数据湖上传。
- 问题：系统崩溃、节点挂掉时需要处理数据部分写入逻辑。
- 方案：CKPT 添加 success 标记位便于恢复。



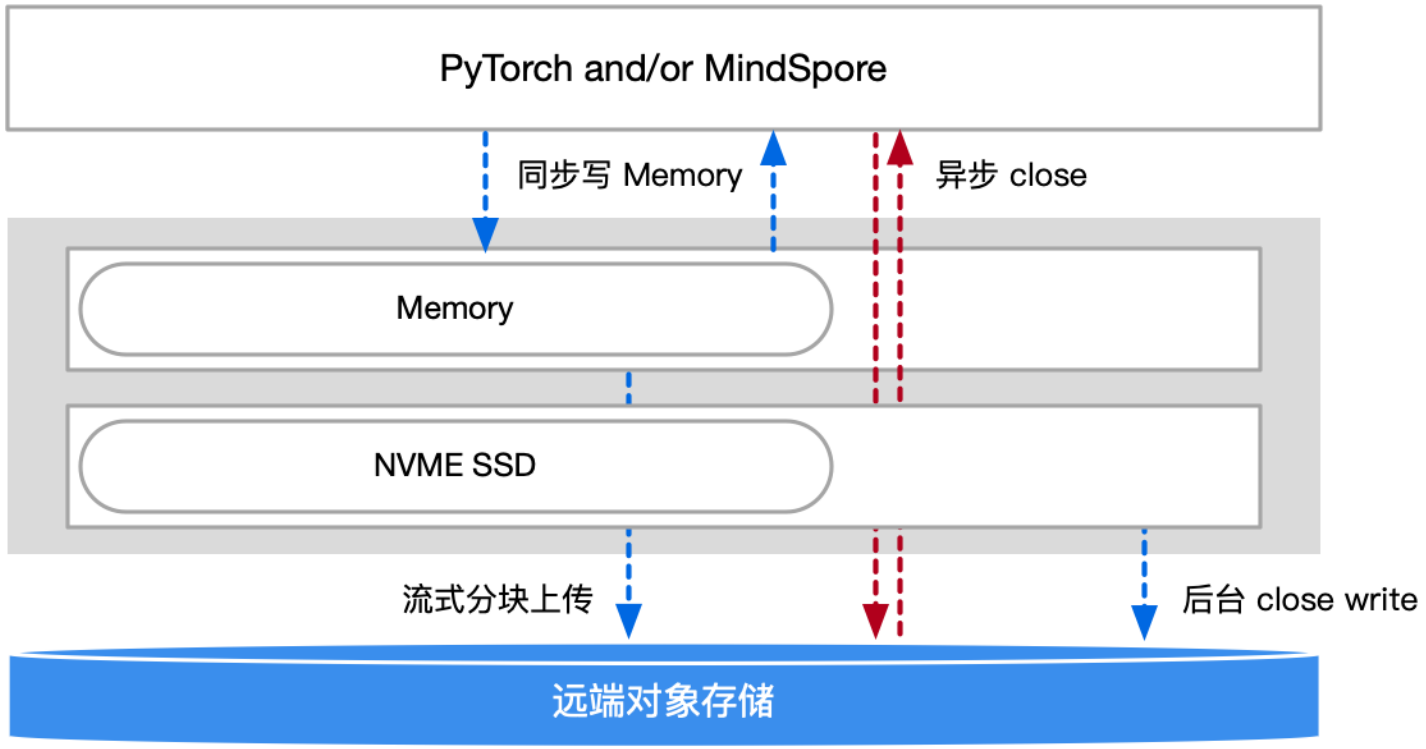
4 多文件加速聚合

- torch.save() 将多个 tensor 分别写入各自文件 idx，对meta data要求高，写入开销大。
- 重写 torch.save()/load() 方法定义数据存储流程，聚合多个 tensor 数据文件；为了 load 过程中数据读取，保留每份数据的索引 index。



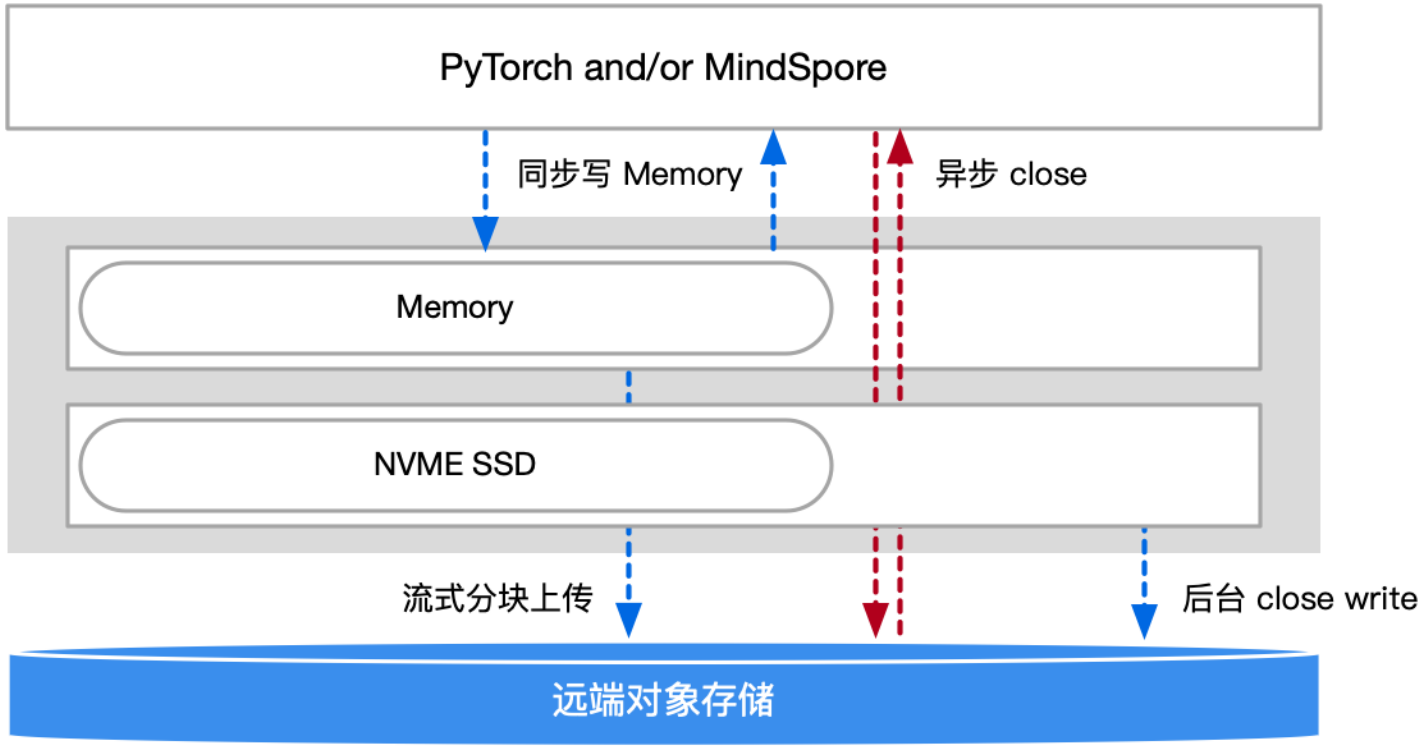
5 本地内存缓存，同步写内存

- 对于 Latest Checkpoint 采用异步写的同时，CKPT save() 时驻留在 CPU 内存，当训练需要恢复时 load() 直接读取，再从数据湖中读取备份到内存，解决 Checkpoint 快速加载问题。



6 数据拷贝过程使用零拷贝

- 通过操作系统的内核技术，实现用户 buffer 间的数据传递，达到数据零拷贝、内存节省的目的。此时 CKPT 数据要求存放在节点内存中。



小结&思考

小结

1. 大模型 CKPT 因为集群训练的不稳定性需要频繁保存
2. 集群的保存和加载需要跟分布式并行的策略相结合对 CKPT 进行切分
3. 了解了大模型 CKPT 保存的文件内容除了模型权重以外还有很多元数据
4. 针对 CKPT 保存提出了 6 个可能得方案，这些方案都在各大厂商进行深度实践

Thank you

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem