第 1 章 引言

1.1 简介

在深度学习中，使用图形处理单元（GPU）进行计算可以加快训练和推断的速度。然而，GPU 内存通常有限，当处理大型模型或大规模数据集时，内存限制可能会成为一个瓶颈。

本软件提供了一种解决方案实现，通过将部分张量从 GPU 内存异步迁移到主机内存，从而减轻了 GPU 显存的压力。这有助于提高训练和推理的整体性能，并且使得在资源受限的环境中更容易运行大型深度学习任务。

删除[X]: 补丁

1.2 编写目的

本文档为该软件的使用说明文档，提供软件的设计和使用方式信息

1.3 项目背景

在对深度神经网络的研究中，我们发现大模型的训练和推理需要使用到巨大的 GPU 内存空间，往往是超过现有单卡显存容量的。在研究中我们发现目前 Pytorch 深度学习框架下没有在模型训练中可将 Tensor 从 GPU 异步迁移到 CPU 的底层实现。本软件针对该问题提供 Tensor 的异步迁移操作。

删除[X]: 补丁

第 2 章 总体设计

2.1 模块概述

本软件包含两个主要模块：

（1）异步换出模块：根据用户指定，将 Tensor 异步从 GPU Memory 迁移到 Host Memory 上；

（2）被动/主动模块：根据数据访问请求被动将 Tensor 从 Host Memory 迁移到 GPU Memory 上，或在反向传播时主动预取。

第 3 章 软件安装说明

3.1 系统需求

1. NVIDIA CUDA 10.2 或更高
2. NVIDIA cuDNN v7 或更高
3. 支持 C++14 的 C++编译器

3.3 安装准备

Ubuntu:

```
1. sudo apt install -y git
```

RHEL/CentOS:

```
1. sudo yum install git
```

3.2 安装

本软件适配 Pytorch-1.12 使用，安装指令如下:

```
1. git clone https://github.com/pytorch/pytorch
2. git clone https://github.com/Nayaco/mahoshojos-large-model-support mlms
```

```
3. cd pytorch
4. git checkout v1.12.0
5. git am ../mlms/patches/pytorch_v1.12.0_large_model_support.patch
```

补丁安装完成后按照 Pytorch 官方仓库文档说明编译软件，Pytorch 官方仓库地址：https://github.com/pytorch/pytorch/tree/v1.12.0

第 4 章 系统功能设计

4.1 异步换出设计

DNN 训练可以分为两个阶段：前向传播阶段和反向传播阶段，其中前向传播阶段中产生的持久数据称为 feature map。feature map 的生命周期一般从对应层的前向传播产生到反向传播梯度计算结束，前向传播产生后这部分数据被暂存在显存中并长时间无访问，一种降低显存占用做法是将 feature map 数据暂时转移到 CPU 内存中。

本设计采用队列维护换出任务，通过底层的换出线程异步执行换出任务。在换出结束后通过同步接口同步换出线程和主线程。

4.2 被动/主动换入设计

被动/主动换入同异步换出的设计在数据结构上保持一致，均采用队列维护对应的任务，被动换入和主动换入共享同一个 FIFO 队列。

1. 4.2.1 被动换入

被动换入操作在 pytorch 底层数据访问接口 *data*/*data_ptr*/*unsafe_data* 中触发，触发前首先检查是否已经存在于 GPU 内存中，若不存在 GPU 内存中，则一个换入任务会添加到任务队列中。在本软件中，被动换入的优先级定义为最高，因此被动换入任务将会添加到任务队列的首部。

1. 4.2.2 主动换入

预取通过用户指定, 使用 *Tensor.need_prefetch()*方法将任务添加到任务队列的尾部，并使用用户接口开始主动换入任务。

第 5 章 函数名称功能

5.1 Tensor Native 函数

2. 5.1.1 Tensor.pageout_manual

将 Tensor 添加到换出任务队列尾部。

3. 5.1.2 Tensor.need_prefech

将 Tensor 添加到换入队列尾部。

5.2 数据转移上下文

4. 5.2.1 torch.cuda.create_swap_env

初始化数据转移上下文。

5. 5.2.2 torch.cuda.prefetch_init

初始化数据换入队列。

1. 5.2.3 torch.cuda.before_prefetch_wait_all

等待所有数据换出操作结束。

2. 5.2.4 torch.cuda.prefetch_all

启动换入任务队列执行主动换入操作。

1. 5.2.5 torch.cuda.close_swap_env

结束数据转移上下文。

5.3 数据转移任务队列

1. 5.3.1 CudaEntityTransferQueue::enqueue

添加数据传输任务到队列尾部。

1. 5.3.2 CudaEntityTransferQueue::erase

从队列中删除指定任务。

2. 5.3.3 CudaEntityTransferQueue::dequeue

返回并删除队列头部任务。

3. 5.3.4 CudaEntityTransferQueue::start_actions

开始队列传输任务。

4. 5.3.5 CudaEntityTransferQueue::wait_and_stop_actions

锁定队列（阻止添加任务到队列）并等待所有任务结束。

5. 5.3.6 CudaEntityTransferQueue::wait_actions

锁定队列（阻止添加任务到队列）等待所有任务结束后重新解锁队列。

5.4 数据转移操作

1. 5.4.1 CudaEntityStorageImpl::pageout_internal_sync

执行换出操作，将 Tensor 数据从 GPU 内存换出到 CPU 内存。

2. 5.4.2 CudaEntityStorageImpl::pagein_internal_sync

执行换入操作，将 Tensor 数据从 CPU 内存换入到 GPU 内存。

第 6 章 接口设计

提供给用户的接口见表 1。

表 1 接口设计说明表

功能简介	接口名	输入参数
添加 Tensor 到换出任务队列尾部	Tensor.pageout_manual	/
添加 Tensor 到换入队列尾部	Tensor.need_prefech	/
初始化数据转移上下文	torch.cuda.create_swap_env	/
初始化数据换入队列	torch.cuda.prefetch_init	/
等待数据换出操作结束	torch.cuda.before_prefetch_wait_all	/
执行主动换入操作	torch.cuda.prefetch_all	/
结束数据转移上下文	torch.cuda.close_swap_env	/