GeePS代码说明

一、综述

1、GeePS是一个适用于GPU深度学习的分布式参数服务器。它建立了GPU内存、CPU内存和P2P网络之间的统一管理。在Caffe实践中，它接管了caffe的GPU管理，有效的降低了大模型对GPU内存容量的要求。

2、GeePS的通讯使用了ZMQ。ZMQ全称是ZeroMQ，是一个分布式MQ系统。

3、对Caffe进行了修改，集成了GeePS模块。同时包括GeePS Server和Client。

3、GeePS\_Caffe运行时的结构如下图：

控制端（发起者）

通过pdsh来分发任务给Worker

Worker0

Worker1

Worker2

Worker3

ZeroMQ Network

Caffe

pdsh

二、GeePS文件结构

1、include/geeps.hpp。geeps核心头文件，应用只要包含此头文件即可。

2、src/client/clientlib.hpp。geeps提供的接口函数的实现都在此处。

具体实现分散在clientlib.cpp、clientlib-\*.cpp中.

启动通讯服务端端口，创建客户端，并且建立连接，启动后台服务线程都在构造函数里面。因其使用了ZMQ，故不用深究其Socket通讯，而要先去了解一下ZMQ的封装。

对src/server/中的代码的调用，也都在这里。

3、encoder-decoder.hpp。内含ClientServerEncode（客户端到服务端的数据编码）和ServerClientDecode（服务端到客户端的数据解码）

三、GeePS接口

0、GeePsConfig：GeePS运行环境配置

1. num\_tables：需要GeePS管理的表数目。Caffe应用中相当于非本地参数的网络层数（或者+1），代码在Solver.cpp line 273。
2. host\_list：Work主机列表。
3. num\_comm\_channels：通道数。Work将会启动从tcp\_base\_port开始的通道数个端口。
4. tcp\_base\_port。开始端口，默认9090.
5. gpu\_memory\_capacity。GPU内存容量限制，此值应该跟GeePS显存控制有关。

1、 Virtual\_系列：对table操作的包装，方便后续实际对table的read和update的操作。

应用初始化的时候，使用Virtual函数来包装操作（Operation），预先分配好table\_id，row\_ids，slack（同步异步的控制变量）。返回值是Operation的handle，Caffe应用中将handle保存在每一层里。同时也提前绑定下操作顺序，Release内存，避免内存泄漏问题。

2、实际访问接口。

输入参数为上述的handle和内存指针的指针。内存的实际分配在GeePS内部，Release操作在下一个接口中。所谓下一个接口，对于Read来说是PostRead，对于PreUpdate来说是Update。

3、GeePS额外提供了PostRead和PreUpdate两个接口，当应用需要读取参数时，参数服务器在GPU内存中分配一块缓冲区，并返回指针，读取结束后，由PostRead负责释放缓冲区，该接口是非阻塞操作；当应用需要更新参数时，首先通过PreUpdate获得缓冲区，而非阻塞的Update接口负责更新参数和释放缓冲区。因此，思路很简单，利用非阻塞IO在不同内存之间交换数据，提升吞吐量，避免单条记录读取的同步开销引起的性能问题。GeePS的这种缓冲区设计，实际上也管理起了整个GPU内存，因为GPU的内存尺寸总是有限的，把参数服务器的Key-Value都放到GPU内存中显然并不现实，通过在后台不停歇的在CPU和GPU之间进行数据交换，从而实现了在少量GPU内存存放巨大模型参数的可能。

附录：

zmq（ZeroMQ）http://zguide.zeromq.org/page:all