个人资料

lingerlanlan

访问： 219483次
积分： 3991
等级： BLOG 5
排名： 第3738名

原创： 156篇 转载： 0篇
译文： 2篇 评论： 260条

文章搜索

文章分类

深度学习（deep learning） (28)
机器学习 (16)
cuda(GPU programming) (20)
文本挖掘 (5)
c/c++ (15)
dota外挂 (5)
hack programming (6)
web2.0 (5)
工具源码 (7)
语言学习 (22)
设计模式 (2)
读书笔记 (1)
翻译 (1)
足球大数据 (2)
大杂烩 (18)
Hadoop (12)
Spark (2)
sklearn (1)

文章存档

2015年08月 (1)

caffe源码分析--SyncedMemory类代码研究

分类： 深度学习（deep learning） 2014-04-24 19:42 2347人阅读 评论(1) 收藏 举报

神经网络 深度学习 机器学习 cuda caffe

数据成员：

void*cpu_ptr_;数据 针

void*gpu_ptr_;数据在gpu的指针

size_tsize_;数据的大小

SyncedHeadhead_;表示数据的状态，有四种状态，分别是未初始化，数据在cpu中，数据在gpu中，数据在cpu和gpu中都有

enumSyncedHead { UNINITIALIZED,HEAD_AT_CPU,HEAD_AT_GPU,SYNCED};

构造函数

SyncedMemory()

:cpu_ptr_(NULL),gpu_ptr_(NULL),size_(0),head_(UNINITIALIZED){}

简单的初始化

explicitSyncedMemory(size_tsize)

:cpu_ptr_(NULL),gpu_ptr_(NULL),size_(size),head_(UNINITIALIZED){}

只是把size（大小）设置了，并未申请内存

析构函数

SyncedMemory::~SyncedMemory(){

//如果cpu有数据，则释放

if(cpu_ptr_){

CaffeFreeHost(cpu_ptr_);

}

```
//如果gpu有数据，则释放
if(gpu_ptr_){
CUDA_CHECK(cudaFree(gpu_ptr_));
}
}
```

函数voidto_cpu()

功能：把数据放到cpu上

1数据未初始化，则在cpu申请内存。此时状态为HEAD_AT_CPU

2数据本来在gpu，则从gpu拷贝内存到cpu。此时状态为SYNCED

3数据本来在cpu，不做处理

4数据在cpu和gpu都有，不做处理

```
inlinevoidSyncedMemory::to_cpu(){
switch(head_){
caseUNINITIALIZED:
CaffeMallocHost(&cpu_ptr_,size_);
memset(cpu_ptr_,0,size_);
head_=HEAD_AT_CPU;
break;
caseHEAD_AT_GPU:
if(cpu_ptr_==NULL){
CaffeMallocHost(&cpu_ptr_,size_);
}
CUDA_CHECK(cudaMemcpy(cpu_ptr_,gpu_ptr_,size_,cudaMemcpyDeviceToHost));
head_=SYNCED;
break;
caseHEAD_AT_CPU:
caseSYNCED:
break;
}
}
```

函数voidto_gpu();

功能：把数据放到gpu上

1数据未初始化，在gpu申请内存。此时状态为HEAD_AT_GPU

2数据在cpu，从cpu拷贝到gpu。此时状态为SYNCED

3数据在gpu，不做操作。

4数据在cpu和gpu都有，不做操作。

```
inlinevoidSyncedMemory::to_gpu(){
switch(head_){
caseUNINITIALIZED:
CUDA_CHECK(cudaMalloc(&gpu_ptr_,size_));
CUDA_CHECK(cudaMemset(gpu_ptr_,0,size_));
head_=HEAD_AT_GPU;
break;
caseHEAD_AT_CPU:
if(gpu_ptr_==NULL){
CUDA_CHECK(cudaMalloc(&gpu_ptr_,size_));
}
CUDA_CHECK(cudaMemcpy(gpu_ptr_,cpu_ptr_,size_,cudaMemcpyHostToDevice));
head_=SYNCED;
break;
caseHEAD_AT_GPU:
caseSYNCED:
break;
}
}
```

函数constvoid*cpu_data();

功能：返回数据在cpu的指针

```
constvoid*SyncedMemory::cpu_data(){
to_cpu();
return(constvoid*)cpu_ptr_;
}
```

函数constvoid*gpu_data();

功能：返回数据在gpu的指针

```
constvoid*SyncedMemory::gpu_data(){
to_gpu();
return(constvoid*)gpu_ptr_;
}
```

函数void*mutable_cpu_data();

功能：返回数据在cpu的指针，并改变数据的状态为HEAD_AT_CPU

```
void*SyncedMemory::mutable_cpu_data(){
to_cpu();
head_=HEAD_AT_CPU;
returncpu_ptr_;
}
```

函数void*mutable_gpu_data();

功能：返回数据在cpu的指针，并改变数据的状态为HEAD_AT_GPU

```
void*SyncedMemory::mutable_gpu_data(){
to_gpu();
head_=HEAD_AT_GPU;
returngpu_ptr_;
}
```

函数SyncedHeadhead(){returnhead_;}

功能：返回数据的状态

函数size_tsize(){returnsize_;}

功能：返回数据的大小

DISABLE_COPY_AND_ASSIGN(SyncedMemory);

一个宏，把该类的拷贝函数和等号操作符给禁止掉

其实就是

```
private:\
SyncedMemory(constSyncedMemory&);\
SyncedMemory&operator=(constSyncedMemory&)
```

如果想让你的类不能使用copy构造函数和赋值操作符，只要将该类的copy构造函数和赋值操作符函数定义为private即可，并且只是声明，不用实现.

上一篇 caffe源码分析--math_functions.cu代码研究
下一篇 给cuda核函数传递二维数组的一种方法