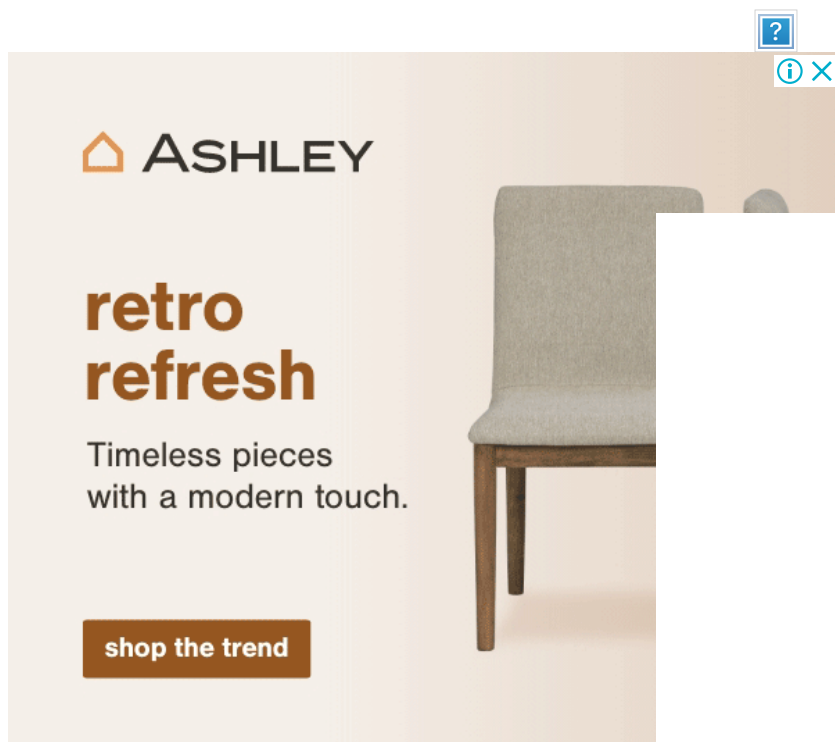




# 谭升的博客

人工智能基础



## 【CUDA 基础】4.5 使用统一内存的向量加法

📅 2018-05-14 | 📁 [CUDA](#) | [Freshman](#) | 💬 0 | 👁

**Abstract:** 使用统一内存的CUDA程序——向量加法

**Keywords:** 统一内存, Uniform Memory

### 使用统一内存的向量加法

本文是前面关于统一内存的补充

参考: <https://face2ai.com/CUDA-F-4-2-%E5%86%85%E5%AD%98%E7%AE%A1%E7%90%86/>

统一内存矩阵加法

统一内存的基本思路就是减少指向同一个地址的指针，比如我们经常见到的，在本地分配内存，然后传输到设备，然后在从设备传输回来，使用统一内存，就没有这些显式的需求了，而是驱动程序帮我们完成。具体的做法就是：

```
1  CHECK(cudaMallocManaged((float**) &a_d, nByte));
2  CHECK(cudaMallocManaged((float**) &b_d, nByte));
3  CHECK(cudaMallocManaged((float**) &res_d, nByte));
```

使用cudaMallocManaged 来分配内存，这种内存存在表面上看在设备和主机端都能访问，但是内部过程和我们前面手动copy过来copy过去是一样的，也就是memcpy是本质，而这个只是封装了一下。

我们来看看完整的代码：

```
1  #include <cuda_runtime.h>
2  #include <stdio.h>
3  #include "freshman.h"
4
5
6
7  void sumArrays(float * a, float * b, float * res, const int size)
8  {
9      for(int i=0; i<size; i+=4)
10     {
11         res[i]=a[i]+b[i];
12         res[i+1]=a[i+1]+b[i+1];
13         res[i+2]=a[i+2]+b[i+2];
14         res[i+3]=a[i+3]+b[i+3];
15     }
16 }
17 __global__ void sumArraysGPU(float*a, float*b, float*res, int N)
18 {
19     int i=blockIdx.x*blockDim.x+threadIdx.x;
20     if(i < N)
21         res[i]=a[i]+b[i];
22 }
23 int main(int argc, char **argv)
24 {
25     // set up device
26     initDevice(0);
27
28     int nElem=1<<24;
```

```

29     printf("Vector size:%d\n",nElem);
30     int nByte=sizeof(float)*nElem;
31     float *res_h=(float*)malloc(nByte);
32     memset(res_h,0,nByte);
33     memset(res_from_gpu_h,0,nByte);
34
35     float *a_d,*b_d,*res_d;
36     CHECK(cudaMallocManaged((float**) &a_d,nByte));
37     CHECK(cudaMallocManaged((float**) &b_d,nByte));
38     CHECK(cudaMallocManaged((float**) &res_d,nByte));
39
40     initialData(a_d,nElem);
41     initialData(b_d,nElem);
42
43     //CHECK(cudaMemcpy(a_d,a_h,nByte,cudaMemcpyHostToDevice));
44     //CHECK(cudaMemcpy(b_d,b_h,nByte,cudaMemcpyHostToDevice));
45
46     dim3 block(512);
47     dim3 grid((nElem-1)/block.x+1);
48
49     double iStart,iElaps;
50     iStart=cpuSecond();
51     sumArraysGPU<<<grid,block>>>(a_d,b_d,res_d,nElem);
52     cudaDeviceSynchronize();
53     iElaps=cpuSecond()-iStart;
54     printf("Execution configuration<<<%d,%d>>> Time elapsed %f sec\n",grid.x,block.x,
55
56     //CHECK(cudaMemcpy(res_from_gpu_h,res_d,nByte,cudaMemcpyDeviceToHost));
57     sumArrays(b_d,b_d,res_h,nElem);
58
59     checkResult(res_h,res_d,nElem);
60     cudaFree(a_d);
61     cudaFree(b_d);
62     cudaFree(res_d);
63
64     free(res_h);
65
66     return 0;
67 }

```

注意我们注释掉的，这就是省去的代码部分、

运行结果：

```
Tony — tony@tony-Lenovo: ~/Project/CUDA_Freshman/build/23_sum_array_uniform_memory — ssh tony@192.168.3.19 — 101x24
tony@tony-Lenovo:~/Project/CUDA_Freshman/build/23_sum_array_uniform_memory$ ./sum_arrays_uniform_memory
Using device 0: GeForce GTX 1050 Ti
Vector size:16777216
Execution configuration<<<32768,512>>> Time elapsed 0.042995 sec
Check result success!
tony@tony-Lenovo:~/Project/CUDA_Freshman/build/23_sum_array_uniform_memory$ ./sum_arrays_uniform_memory
Using device 0: GeForce GTX 1050 Ti
Vector size:16777216
Execution configuration<<<32768,512>>> Time elapsed 0.040515 sec
Check result success!
tony@tony-Lenovo:~/Project/CUDA_Freshman/build/23_sum_array_uniform_memory$ ./sum_arrays_uniform_memory
Using device 0: GeForce GTX 1050 Ti
Vector size:16777216
Execution configuration<<<32768,512>>> Time elapsed 0.042737 sec
Check result success!
tony@tony-Lenovo:~/Project/CUDA_Freshman/build/23_sum_array_uniform_memory$
```

就这个代码而言，使用统一内存还是手动控制，运行速度差不多。

这里有一个新概念叫页面故障，我们分配的这个统一内存地址是个虚拟地址，对应了主机地址和GPU地址，当我们的主机访问这个虚拟地址的时候，会出现一个页面故障，当CPU要访问位于GPU上的托管内存时，统一内存使用CPU页面故障来出发设备到CPU的数据传输，这里的故障不是坏掉了，而是一种通信方式，类似于中断。

故障数和传输数据的大小直接相关。

使用

```
1 nvprof --unified-memory-profiling per-process-device ./sum_arrays_uniform_memory
```

可以查看到实际参数

```
tony@tony-Lenovo: ~/Project/CUDA_Freshman/build/23_sum_array_uniform_memory — ssh tony@192.168.3.19 — 101x35
tony@tony-Lenovo:~/Project/CUDA_Freshman/build/23_sum_array_uniform_memory$ sudo /usr/local/cuda/bin/
nvprof --unified-memory-profiling per-process-device ./sum_arrays_uniform_memory
==8309== NVPROF is profiling process 8309, command: ./sum_arrays_uniform_memory
Using device 0: GeForce GTX 1050 Ti
Vector size:16777216
Execution configuration<<<32768,512>>> Time elapsed 0.044016 sec
Results don't match!
110.089996(hostRef[0] )!= 62.320999(gpuRef[0])
==8309== Profiling application: ./sum_arrays_uniform_memory
==8309== Profiling result:
      Type  Time(%)   Time      Calls      Avg      Min      Max  Name
GPU activities: 100.00% 43.911ms      1 43.911ms 43.911ms 43.911ms sumArraysGPU(float*, flo
at*, float*, int)
  API calls: 72.96% 142.83ms      3 47.609ms 175.80us 142.45ms cudaMallocManaged
              22.43% 43.919ms      1 43.919ms 43.919ms 43.919ms cudaDeviceSynchronize
              4.12% 8.0694ms      3 2.6898ms 2.3654ms 3.1853ms cudaFree
              0.20% 395.69us      1 395.69us 395.69us 395.69us cudaGetDeviceProperties
              0.18% 359.18us     94 3.8210us  125ns 157.83us cuDeviceGetAttribute
              0.04% 82.985us      1 82.985us 82.985us 82.985us cudaLaunch
              0.03% 54.063us      1 54.063us 54.063us 54.063us cuDeviceTotalMem
              0.03% 51.315us      1 51.315us 51.315us 51.315us cuDeviceGetName
              0.00% 4.6360us      1 4.6360us 4.6360us 4.6360us cudaSetDevice
              0.00% 3.7910us      1 3.7910us 3.7910us 3.7910us cudaConfigureCall
              0.00% 1.8340us      3    611ns  166ns 1.4630us cuDeviceGetCount
              0.00% 1.2530us      4    313ns  141ns   614ns cudaSetupArgument
              0.00% 957ns      2    460ns  155ns   802ns cuDeviceGet

==8309== Unified Memory profiling result:
Device "GeForce GTX 1050 Ti (0)"
      Count  Avg Size  Min Size  Max Size  Total Size  Total Time  Name
      1284  102.08KB  4.0000KB  0.9844MB  128.0000MB  21.85574ms  Host To Device
      386  169.95KB  4.0000KB  0.9961MB  64.06250MB  10.43146ms  Device To Host
      389  -         -         -         -         42.93462ms  Gpu page fault groups
Total CPU Page faults: 577
tony@tony-Lenovo:~/Project/CUDA_Freshman/build/23_sum_array_uniform_memory$
```

也可以使用 nvvp来查看，效果类似。

## 总结

虽然统一内存管理给我们写代码带来了方便而且速度也很快，但是实验表明，手动控制还是要优于统一内存管理，换句话说，人脑的控制比编译器和目前的设备更有效，所以，为了效率，大家还是手动控制内存吧，把命运掌握在自己手里。

本文作者： 谭升

本文链接：<https://face2ai.com/CUDA-F-4-5-使用统一内存的向量加法/>

版权声明： 本博客所有文章除特别声明外，均采用 [CC BY-NC-SA 4.0](#) 许可协议。转载请注明出处！

 相关文章

---

© 2022 谭升

