Ex3 profile report

environment: ubuntu 14.04
Cuda compilation tools, release 7.5, V7.5.17
NVIDIA Visual Profiler, Version: 7.5

班级：F1303023
姓名：董一哲
学号：5130309692

1. 实验介绍

本实验中，我们将设计 GPU 并行计算程序，实现一个长数组的翻转。比较使用 shared memory 的方法和一般方法，并进行 profile。

sharedmemory 在 GPU 体系结构中的位置如下图所示：

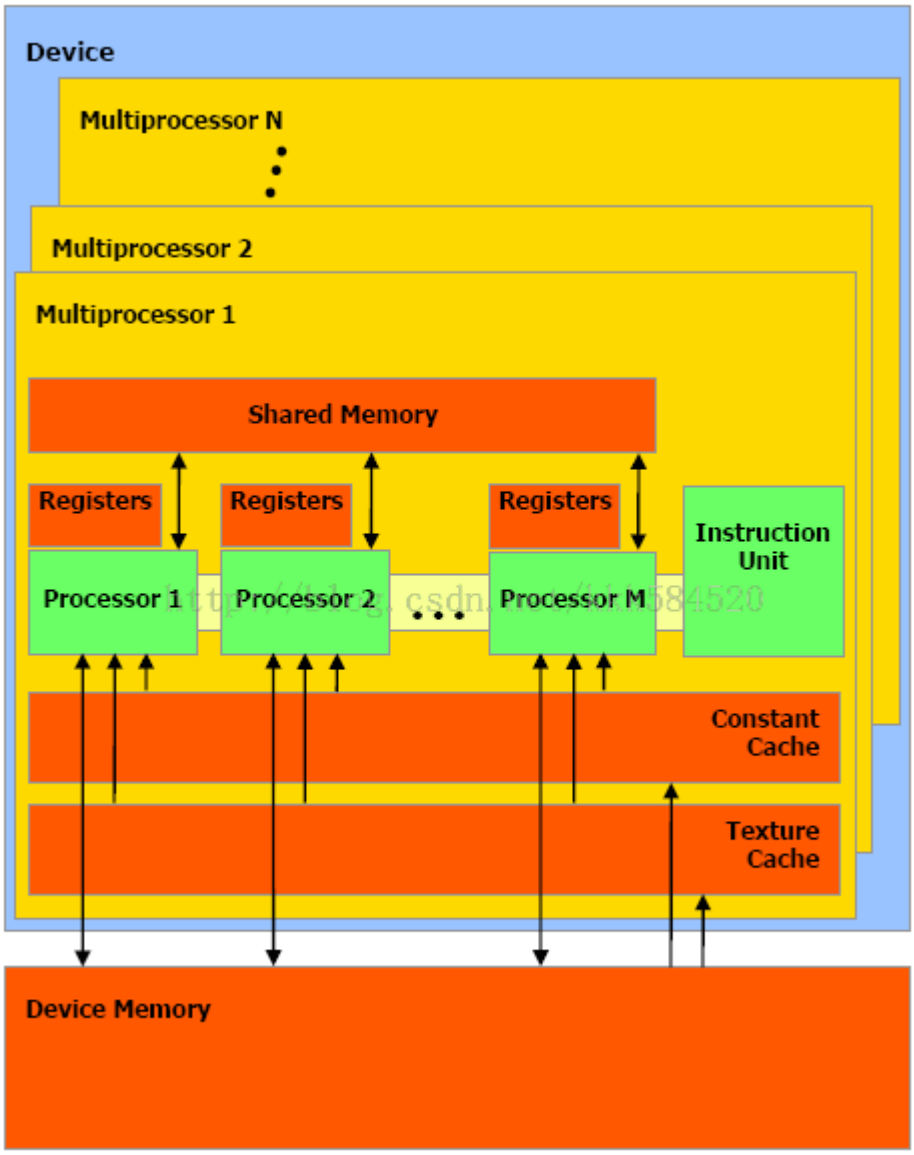

图 1.1 shared memory 在 GPU 架构中的位置

它实际上充当了 device memory 和 multiprocessor 之间的类似于 cache 的角色，同一个 block 中的各个 thread 共享一块 shared memory，因此，当同一个 block 中，各个 thread 对相同数据有访存请求时，将该数据放在这个 block 的 shared memory 中，可以有效减少方寸延迟。

2. 实验设计

数组规模设置为 int [256 * 1024] 共计 1MB。

方法一：

直接利用 multiBlock, multiThread 的方式。步骤如下：

1.在内存中申请空间，生成源数组;
2.在显存中申请数组两倍大小的空间，再将数组拷贝进入显存;
3.利用核函数并行地将数组翻转;
4.将结果拷贝回内存，并进行正确性检测。

方法二:

步骤基本与方法一相同，但是在核函数中，显式地在读取显存时将数据拿到 sharedmemory 中，同一个 block 中的其它 thread 直接访 问 sharedmemory 即可。

3. 实验分析

根据 share memory 的原理，我们推断：share memory 适用于单数据多次访问的场合。对应地，如果在一个 block 中所需要的数据没有被多个 thread 同时发起访存请求的话，sharedmemory 无法达到类似 cache 的时间节省效果。

在本实验中，由于每一个放到 shared memory 中的数据不会被其它 thread 所利用（每个 thread 负责把源数组中的一个数据放到目标数组的对应位置）
因此 sharedmemory 不会起到明显的加速作用，相反由于操作数增加，可能会增大开销。
在本实验中，我们假设方法一、二之间的开销相差不大。

4. profile 测试

利用 NVIDIA Visual Profiler, Version: 7.5 在源代码末尾中加入 cudaProfilerStop()函数，运行图形化 profile IDE，分别将两种方法所编译生成的.out 文件进行 profile。结果如下：

方法一 profile 结果

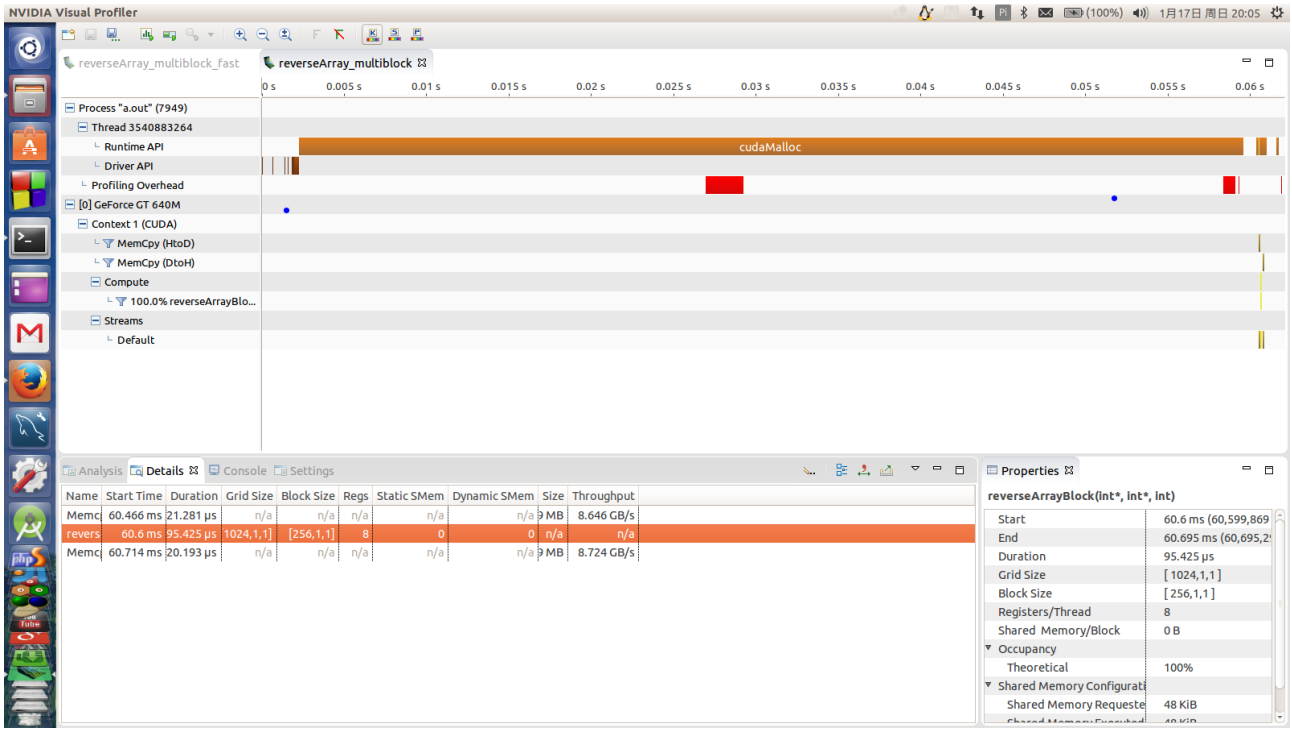

图 4.1 non_shared 方法 profile 结果

核函数 duration 为 95.425 微秒

方法二 profile 结果

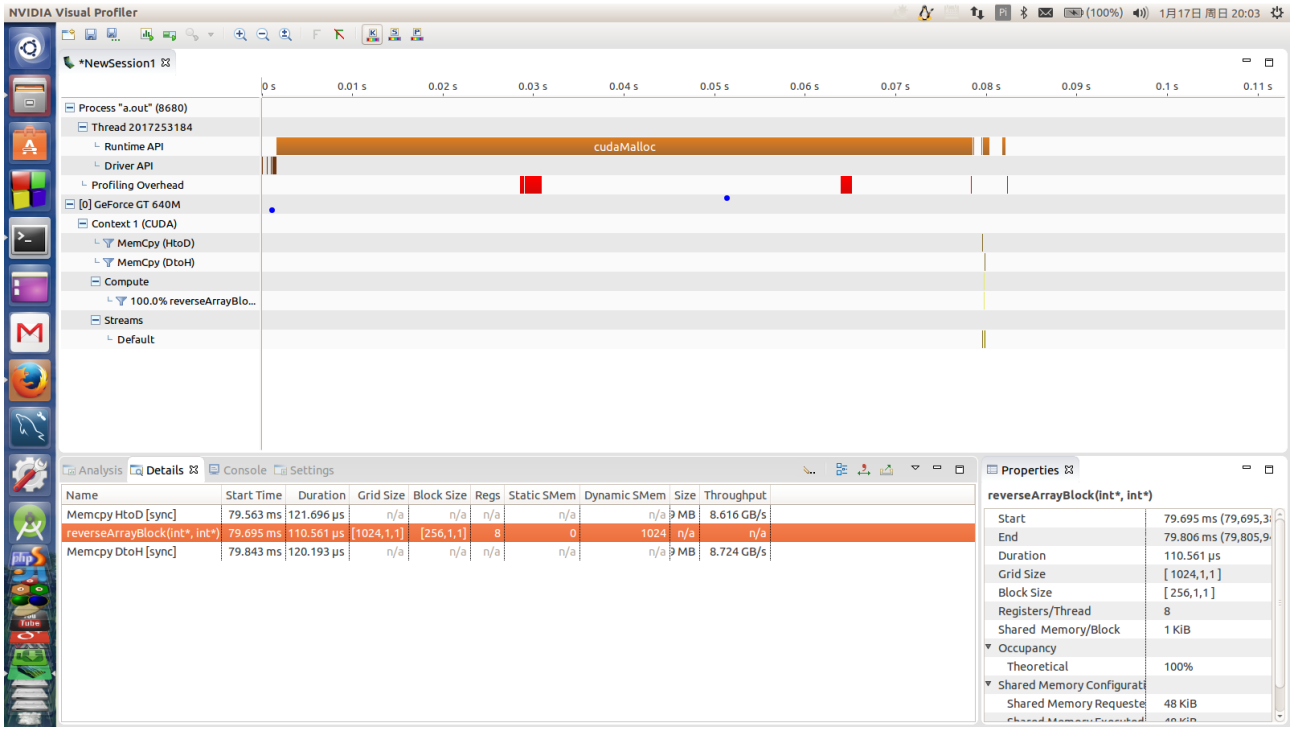

图 4.2 sharedmemory 方法 profile 结果

核函数 duration 为 110.561 微秒。

5. 结论

根据 profile 结果，加入了 sharedmemory 优化的程序，其核函数运行时间反而略为增加，这印证了我们的分析，在一个 block 中所需要的数据没有被多个 thread 同时发起访存请求的话，sharedmemory 无法达到类似 cache 的时间节省效果。