CUDA编程——GPU架构，由sp，sm，thread，block，grid，warp说起_Zhang Junior 的博客-CSDN博客

https://blog.csdn.net/junparadox/article/details/50540602

None Sat Nov, 21 18:18

ZhangJunior 2016-01-19 10:04:36

最后发布:2016-01-19 10:04:36首次发布:2016-01-19 10:04:36

掌握部分硬件知识，有助于程序员编写更好的CUDA程序，提升CUDA程序性能，本文目的是理清sp，sm，thread，block，grid，warp之间的关系。由于作者能力有限，难免有疏漏，恳请读者批评指正。

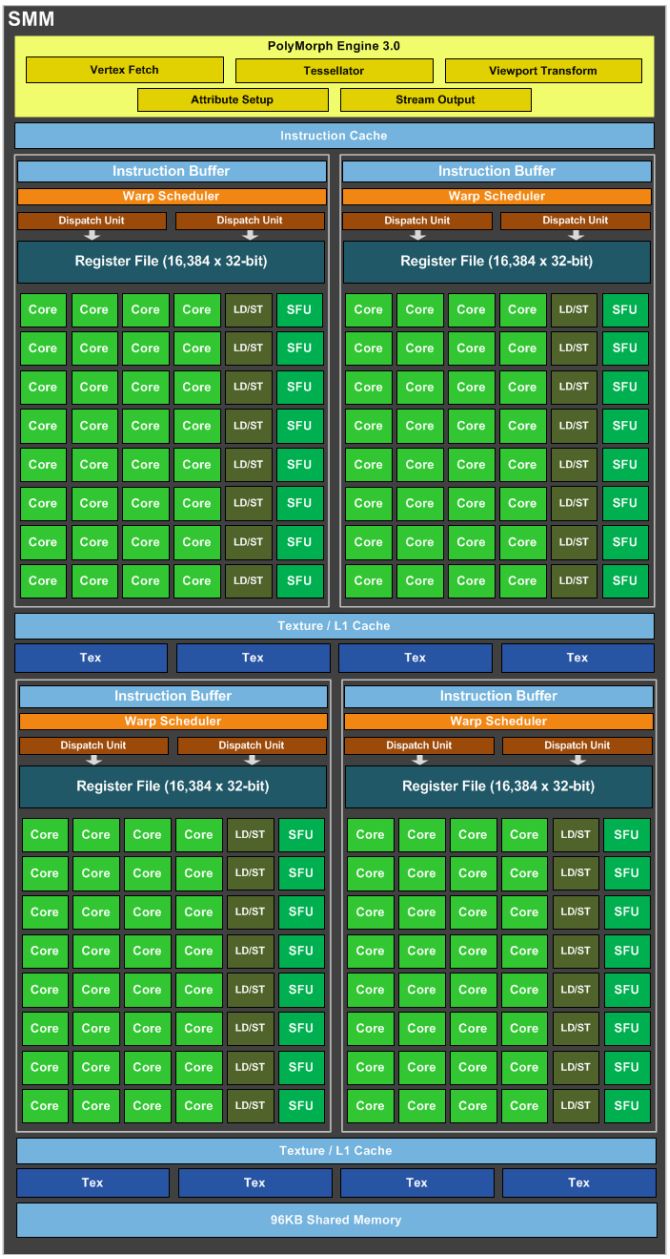
首先我们要明确：SP（streaming Process），SM（streaming multiprocessor）是硬件（GPU hardware）概念。而thread，block，grid，warp是软件上的（CUDA）概念。

从硬件看

- **SP**：最基本的处理单元，streaming processor，也称为CUDA core。最后具体的指令和任务都是在SP上处理的。GPU进行并行计算，也就是很多个SP同时做处理。
- **SM**：多个SP加上其他的一些资源组成一个streaming multiprocessor。也叫GPU大核，其他资源如：warp scheduler，register，shared memory等。SM可以看做GPU的心脏（对比CPU核心），register和shared memory是SM的稀缺资源。CUDA将这些资源分配给所有驻留在

SM中的threads。因此，这些有限的资源就使每个SM中active warps有非常严格的限制，也就限制了并行能力。

需要指出，每个SM包含的SP数量依据GPU架构而不同，Fermi架构GF100是32个，GF10X是48个，Kepler架构都是192个，Maxwell都是128个。相同架构的GPU包含的SM数量则根据GPU的中高低端来定。下图给出Nvidia GTX980 的一个SM示意图，图中每个绿色框框表示一个SP。注意，在Maxwell架构中，Nvidia已经把SM改叫SMM。下图表示的仅仅是一个SMM，一个GPU可以有多个SM（比如16个），最终一个GPU可能包含有上千个SP。这么多核心“同时运行”，速度可想而知，这个引号只是想表明实际上，软件逻辑上是所有SP是并行的，但是物理上并不是所有SP都能同时执行计算，因为有些会处于挂起，就绪等其他状态，这有关GPU的线程调度，以后再写了。
原图

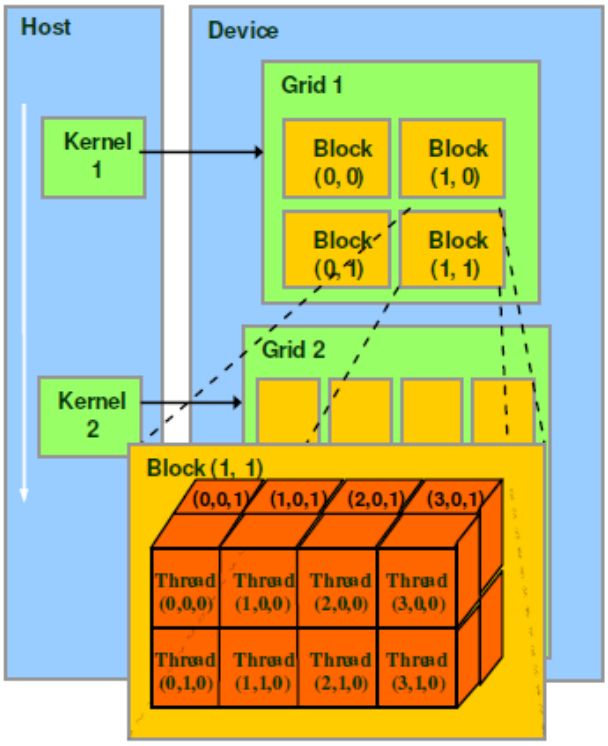


从软件看

thread，block，grid，warp是CUDA编程上的概念，以方便程序员软件设计，组织线程，同样的我们给出一个示意图来表示。

- **thread**：一个CUDA的并行程序会被以许多个threads来执行。
- **block**：数个threads会被群组成一个block，同一个block中的threads可以同步，也可以通过shared memory通信。
- **grid**：多个blocks则会再构成grid。

- **warp**：GPU执行程序时的调度单位，目前cuda的warp的大小为32，同在一个warp的线程，以不同数据资源执行相同的指令,这就是所谓 SIMT。



对应关系

从软件上看，SM更像一个独立的CPU core。SM（Streaming Multiprocessors）是GPU架构中非常重要的部分，GPU硬件的并行性就是由SM决定的。以Fermi架构为例，其包含以下主要组成部分：

- CUDA cores
- Shared Memory/L1Cache
- Register File
- Load/Store Units
- Special Function Units
- Warp Scheduler

GPU中每个sm都设计成支持数以百计的线程并行执行，并且每个GPU都包含了很多的SM，所以GPU支持成百上千的线程并行执行。当一个kernel启动后，thread会被分配到这些SM中执行。大量的thread可能会被分配到不同的SM，**同一个block中的threads必然在同一个SM中并行**

（SIMT）执行。每个thread拥有它自己的程序计数器和状态寄存器，并且用该线程自己的数据执行指令，这就是所谓的Single Instruction Multiple Thread。

一个SP可以执行一个thread，但是实际上并不是所有的thread能够在同一时刻执行。Nvidia把32个threads组成一个warp，**warp是调度和运行的基本单元**。warp中所有threads并行的执行相同的指令。一个warp需要占用一个SM运行，多个warps需要轮流进入SM。由SM的硬件warp scheduler负责调度。目前每个warp包含32个threads（Nvidia保留修改数量的权利）。所以，一个GPU上resident thread最多只有 SM*warp个。

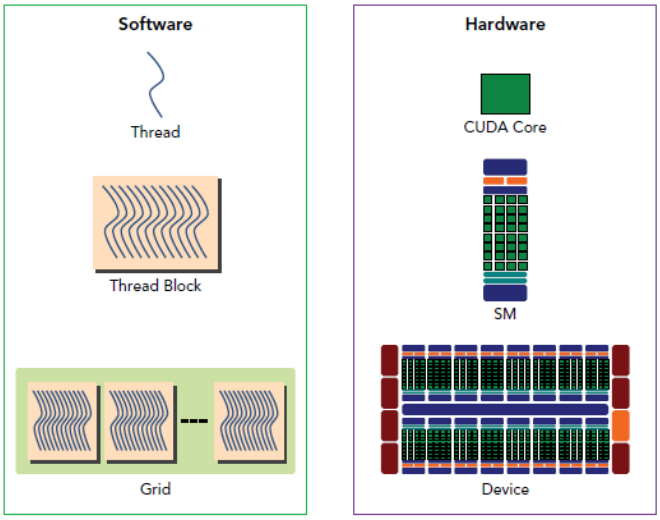
SIMT和SIMD

CUDA是一种典型的SIMT架构（单指令多线程架构），SIMT和SIMD（Single Instruction, Multiple Data）类似，SIMT应该算是SIMD的升级版，更灵活，但效率略低，SIMT是NVIDIA提出的GPU新概念。二者都通过将同样的指令广播给多个执行官单元来实现并行。一个主要的不同就是，SIMD要求所有的vector element在一个统一的同步组里同步的执行，而SIMT允许线程们在一个warp中独立的执行。SIMT有三个SIMD没有的主要特征：

- 每个thread拥有自己的instruction address counter
- 每个thread拥有自己的状态寄存器
- 每个thread可以有自己独立的执行路径

更细节的差异可以看这里。

前面已经说block是软件概念，一个block只会由一个sm调度，程序员在开发时，通过设定block的属性，**“告诉”**GPU硬件，我有多少个线程，线程怎么组织。而具体怎么调度由sm的warps scheduler负责，block一旦被分配好SM，该block就会一直驻留在该SM中，直到执行结束。一个SM可以同时拥有多个blocks，但需要序列执行。下图显示了软件硬件方面的术语对应关系：



需要注意的是，大部分threads只是逻辑上并行，并不是所有的thread可以在物理上同时执行。例如，遇到分支语句（if else，while，for等）时，各个thread的执行条件不一样必然产生分支执行，这就导致同一个block中的线程可能会有不同步调。另外，并行thread之间的共享数据会导致竞态：多个线程请求同一个数据会导致未定义行为。CUDA提供了cudaThreadSynchronize()来同步同一个block的thread以保证在进行下一步处理之前，所有thread都到达某个时间点。

同一个warp中的thread可以以任意顺序执行，active warps被sm资源限制。当一个warp空闲时，SM就可以调度驻留在该SM中另一个可用warp。在并发的warp之间切换是没什么消耗的，因为硬件资源早就被分配到所有thread和block，所以该新调度的warp的状态已经存储在SM中了。不同于CPU，CPU切换线程需要保存/读取线程上下文（register内容），这是非常耗时的，而GPU为每个threads提供物理register，无需保存/读取上下文。

总结

掌握部分硬件知识，有助于CUDA性能提升。