**Tesla GPU架构分析**

**GPU架构**

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，但是同一个block中的thread必然在同一个SM中并行执行。

CUDA采用Single Instruction Multiple Thread（SIMT）的架构来管理和执行thread，这些thread以32个为单位组成一个单元，称作warps。warp中所有线程并行的执行相同的指令。每个thread拥有它自己的instruction address counter和状态寄存器，并且用该线程自己的数据执行指令。

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：

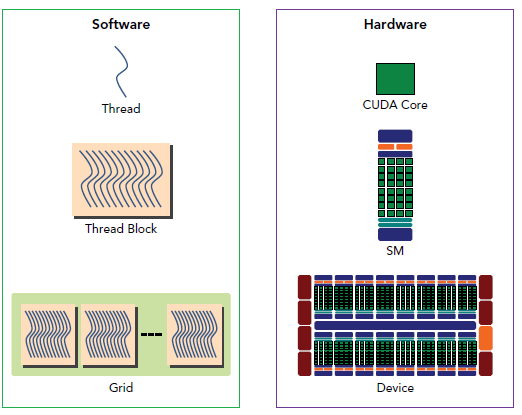
每个thread拥有自己的instruction address counter

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。](http://yosefk.com/blog/simd-simt-smt-parallelism-in-nvidia-gpus.html" \t "https://www.cnblogs.com/1024incn/p/_blank)

一个block只会由一个SM调度，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block。下图显示了软件硬件方面的术语：



需要注意的是，大部分thread只是逻辑上并行，并不是所有的thread可以在物理上同时执行。这就导致，同一个block中的线程可能会有不同步调。

并行thread之间的共享数据回导致竞态：多个线程请求同一个数据会导致未定义行为。CUDA提供了API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。不过，我们是没有什么原子操作来保证block之间的同步的。

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。

SM可以看做GPU的心脏，寄存器和共享内存是SM的稀缺资源。CUDA将这些资源分配给所有驻留在SM中的thread。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。所以，掌握部分硬件知识，有助于CUDA性能提升。

**Fermi架构**

Fermi是第一个完整的GPU计算架构。

512个accelerator cores即所谓CUDA cores（包含ALU和FPU）

16个SM，每个SM包含32个CUDA  core

六个384位 GDDR5 DRAM，支持6GB global on-board memory

GigaThread engine（图左侧）将thread blocks分配给SM调度

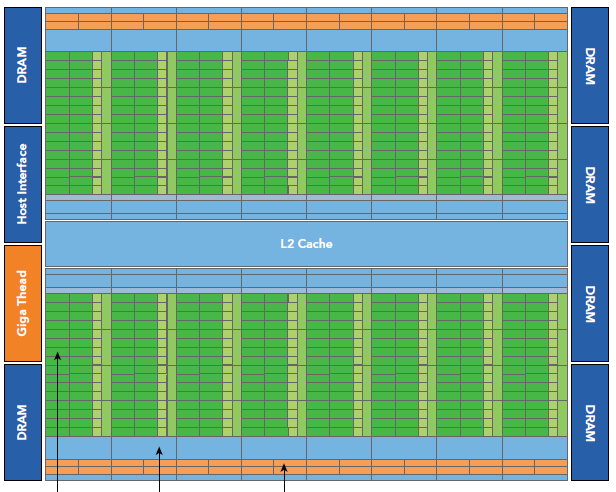
768KB L2 cache

每个SM有16个load/store单元，允许每个clock cycle为16个thread（即所谓half-warp，不过现在不提这个东西了）计算源地址和目的地址

Special function units（SFU）用来执行sin cosine 等

每个SM两个warp scheduler两个instruction dispatch unit，当一个block被分配到一个SM中后，所有该block中的thread会被分到不同的warp中。

Fermi（compute capability 2.x）每个SM同时可处理48个warp共计1536个thread。



每个SM由一下几部分组成：

执行单元（CUDA cores）

调度分配warp的单元

shared memory，register file，L1 cache

**Kepler 架构**

Kepler相较于Fermi更快，效率更高，性能更好。

15个SM

6个64位memory controller

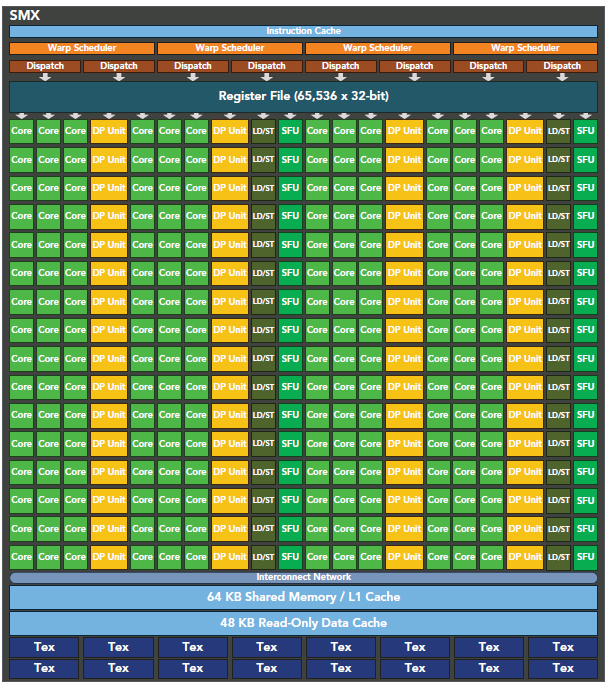
192个单精度CUDA cores，64个双精度单元，32个SFU，32个load/store单元（LD/ST）

增加register file到64K

每个Kepler的SM包含四个warp scheduler、八个instruction dispatchers，使得每个SM可以同时issue和执行四个warp。

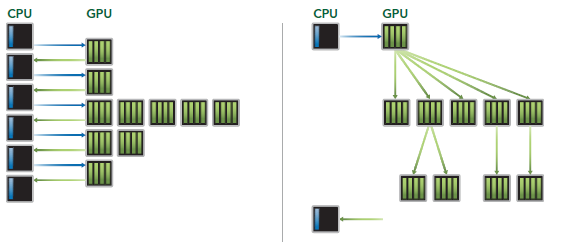
Kepler K20X（compute capability 3.5）每个SM可以同时调度64个warp共计2048个thread。





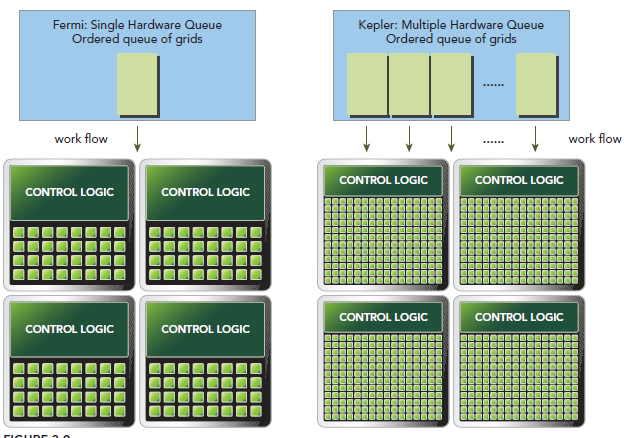
**Dynamic Parallelism**

Dynamic Parallelism是Kepler的新特性，允许GPU动态的启动新的Grid。有了这个特性，任何kernel内都可以启动其它的kernel了。这样直接实现了kernel的递归以及解决了kernel之间数据的依赖问题。也许D3D中光的散射可以用这个实现。



**Hyper-Q**

Hyper-Q是Kepler的另一个新特性，增加了CPU和GPU之间硬件上的联系，使CPU可以在GPU上同时运行更多的任务。这样就可以增加GPU的利用率减少CPU的闲置时间。Fermi依赖一个单独的硬件上的工作队列来从CPU传递任务给GPU，这样在某个任务阻塞时，会导致之后的任务无法得到处理，Hyper-Q解决了这个问题。相应的，Kepler为GPU和CPU提供了32个工作队列。



不同arch的主要参数对比

