

CUDA编程入门（二）GPU硬件基础



ZihaoZhao
深度学习算法|AI芯片|无人机

+ 关注他

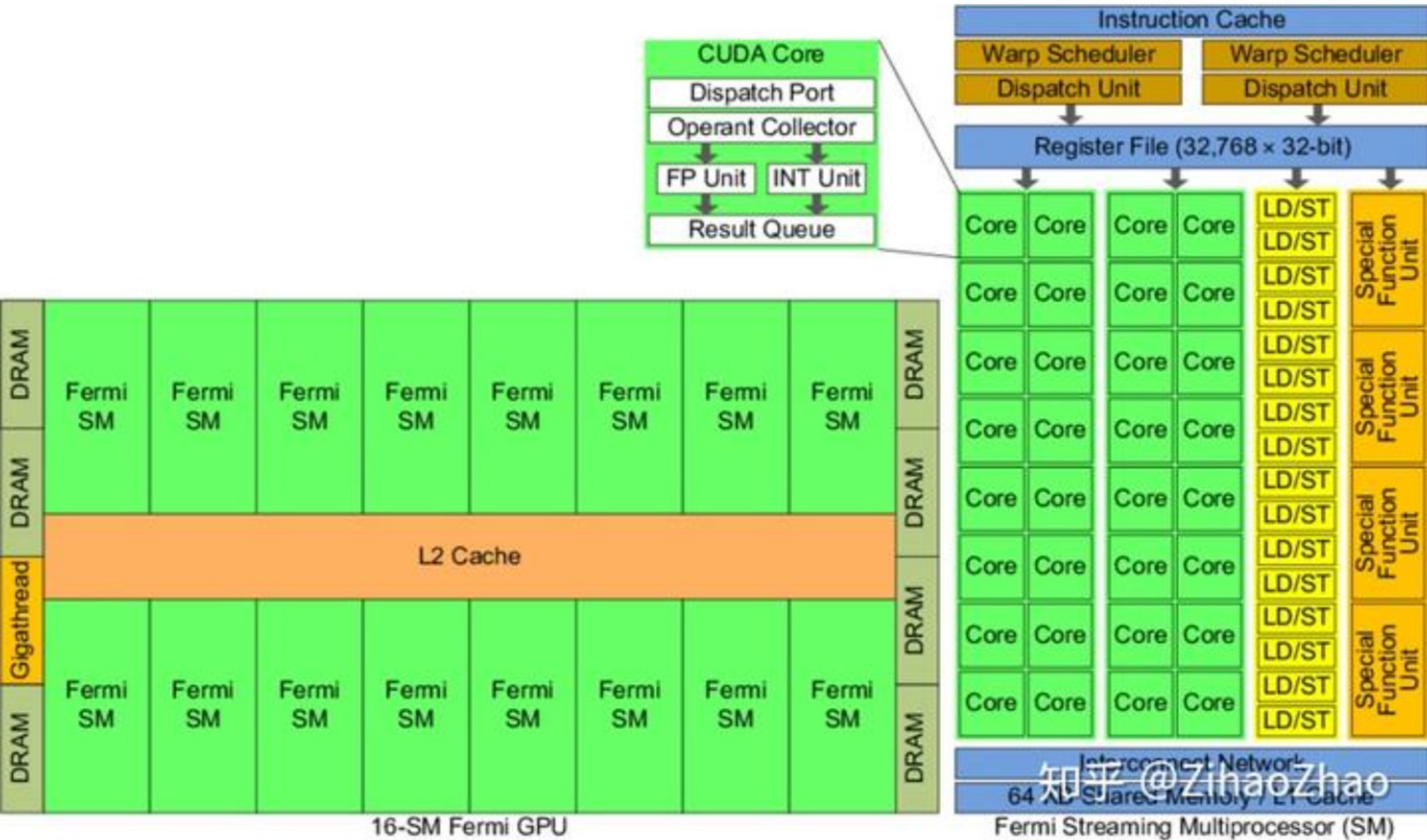
151 人赞同了该文章

要写出高效率的CUDA代码，还必须对GPU的硬件系统有整体的了解，不能只停留在软件层面。所以这一篇，我们来介绍一下GPU的硬件结构相关知识，再把软件逻辑层面和硬件底层结构结合起来，深入了解一下GPU。

GPU硬件结构

GPU实际上是一个SM的阵列，每个SM包含N个计算核，现在我们的常用GPU中这个数量一般为128或192。一个GPU设备中包含一个或多个SM，这是处理器具有可扩展性的关键因素。如果向设备中增加更多的SM，GPU就可以在同一时刻处理更多的任务，或者对于同一任务，如果有足够的并行性的话，GPU可以更快完成它。

具体而言，以Fermi架构的GPU为例，其结构如下图。



左边是GPU的整体结构，其主要是由大量的SM（Streaming-Multiprocessor）和DRAM存储等构成的。右图是对单个SM进行放大，可以看到SM由大量计算核（有时也称SP或CUDA核）、LDU（Load-Store Units）、SFU（Special-Function Units）、寄存器、共享内存等构成。这种结构正是GPU具有高并行度计算能力的基础。通过一定的层级结构组织大量计算核，并给各级都配有相应的内存系统，GPU获得了出色的计算能力。

其中流式多处理器（SM）是GPU架构的核心。GPU中的每一个SM都能支持数百个线程并发执行，每个GPU通常有多个SM，所以在一个GPU上并发执行数千个线程是有可能的。当启动一个内核网络时，它的线程块会被分布在可用的SM上来执行。当线程块一旦被调度到一个SM上，其中的线程只会在那个指定的SM上并发执行。多个线程块可能会被分配到同一个SM上，而且是根据SM资源的可用性进行调度的。

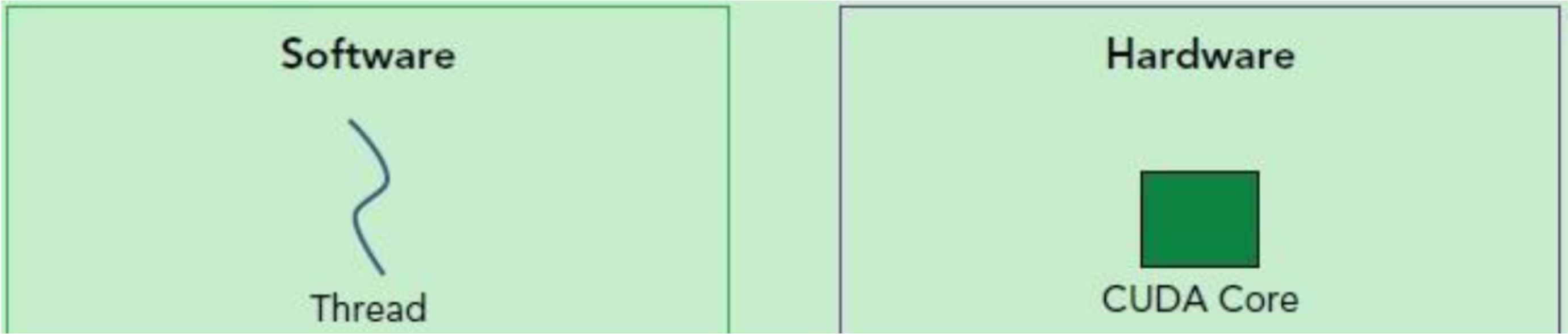
再多提一嘴，稍微说说计算核以外的部分。线程束调度器（Warp Scheduler）顾名思义是进行线程束的调度，负责将软件线程分配到计算核上；LDU（Load-Store Units）负责将值加载到内存或从内存中加载值；SFU（Special-Function Units）用来处理sin、cos、求倒数、开平方特殊函数。

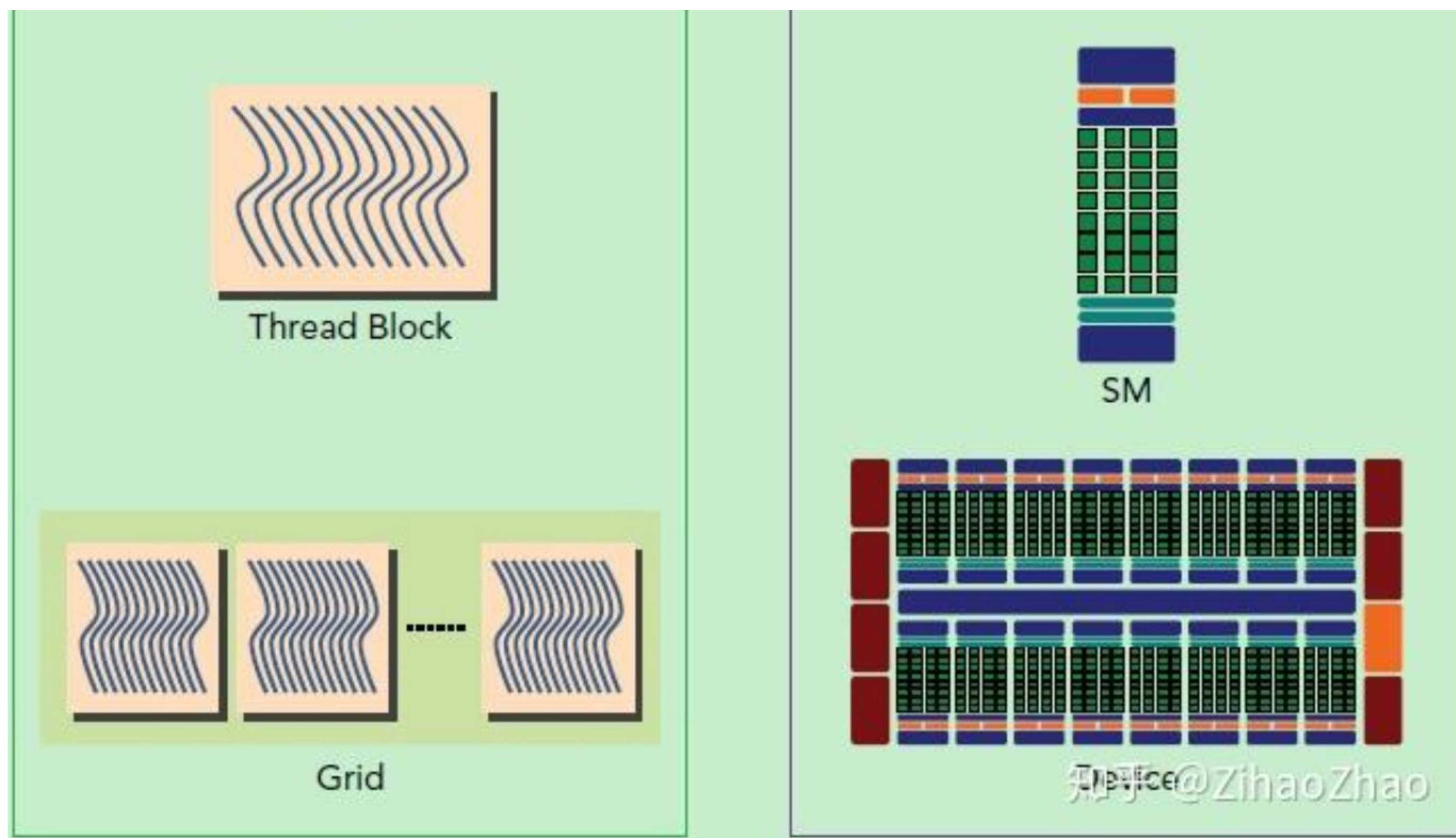
软硬件组织结构对比

CUDA采用单指令多线程（SIMT）架构来管理和执行线程，每32个线程为一组，被称为线程束（Warp）。线程束中所有线程同时执行相同的指令。每个线程都有自己的指令地址计数器和寄存器状态，利用自身的数据执行当前的指令。每个SM都将分配给它的线程块划分到包含32个线程的线程束中，然后在可用的硬件资源上调度执行。

SIMT架构和CPU编程中常见的SIMD（单指令多数据）架构相似。两者都是将相同的指令广播给多个执行单元来实现并行。一个关键的区别就是SIMD要求同一个向量中的所有元素要在一个统一的同步组中一起执行，而SIMT允许同一线程束的多个线程独立执行。尽管一个线程数中的所有线程在相同的程序地址上同时开始执行，但是单独的线程仍有可能有不同的行为。

下面这幅图分别从逻辑视图和硬件视图描述了CUDA编程对应的组件。





左侧是逻辑视图，自上而下，从线程构成线程块再构成线程网络。对应右侧的硬件就是CUDA core、SM、GPU。一个线程块只能在一个SM上被调度，而且一旦线程块在一个SM上被调度，就会保存在该SM上直到执行完成。需要注意的是，这两种层级并不是完全一一对应的，比如在同一时间，一个SM可以容纳多个线程块。

在SM中，共享内存和寄存器是非常重要的资源。共享内存被分配在SM上的常驻线程块中，寄存器在线程中被分配。线程块中的线程通过这些资源可以进行相互的合作和通信。尽管线程块里的所有线程都可以逻辑地并行运行，但并不是所有线程都可以同时在物理层面执行。因此线程块里的不同线程可能会以不同速度前进。我们可以使用CUDA语句在需要的时候进行线程的同步。

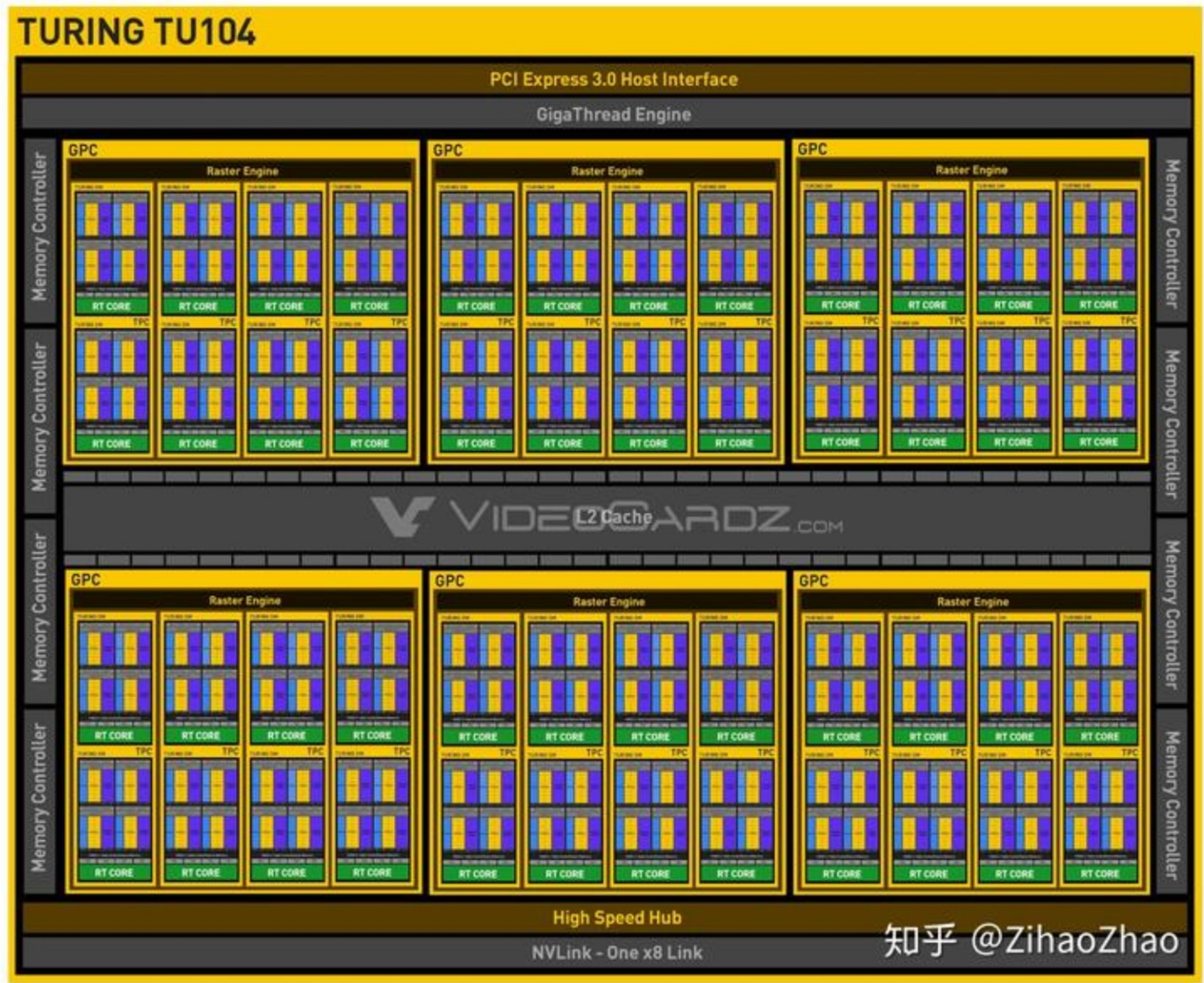
尽管线程块里的线程束可以任意顺序调度，但活跃的线程束数量还是会由SM的资源所限制。当线程数由于任何理由闲置的时候（比如等待从设备内存中读取数值）这时SM可以从同一SM上的常驻线程块中调度其他可用的线程束。在并发的线程束间切换并没有额外开销，因为硬件资源已经被分配到了SM上的所有线程和线程块中。这种策略有效地帮助GPU隐藏了访存的延时，因为随时有大量线程束可供调度，理想状态下计算核将一直处于忙碌状态，从而可以获得很高的吞吐量。

总结一下，SM是GPU架构的核心，而寄存器和共享内存是SM中的稀缺资源。CUDA将这些资源分配到SM中的所有常驻线程里。因此，这些有限的资源限制了在SM上活跃的线程束数量，而活跃的线程束数量对应于SM上的并行量。

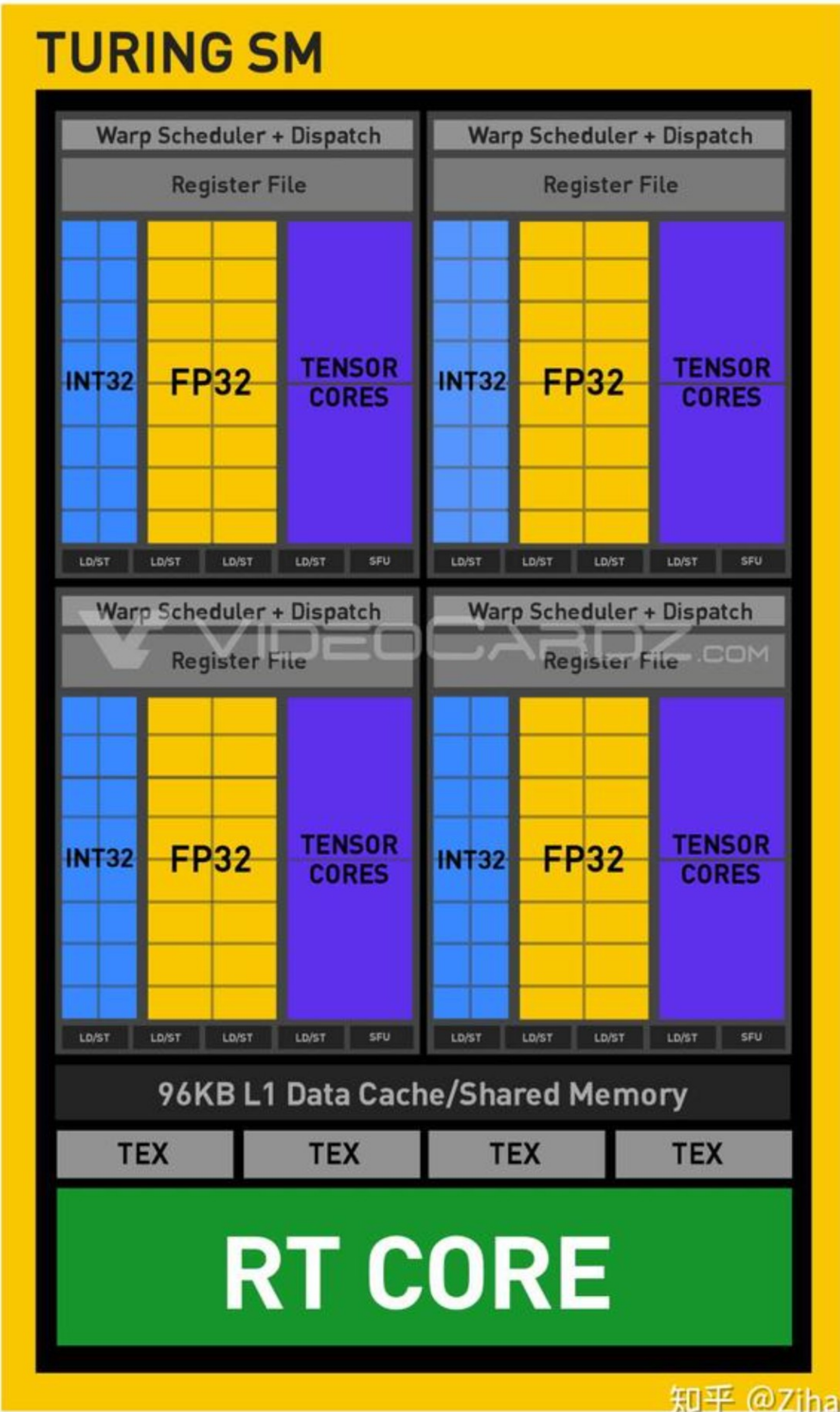
补充：Turing架构

上面那些是比较通用的GPU结构知识，那么我们再来看看现在最新一代GPU是什么结构。

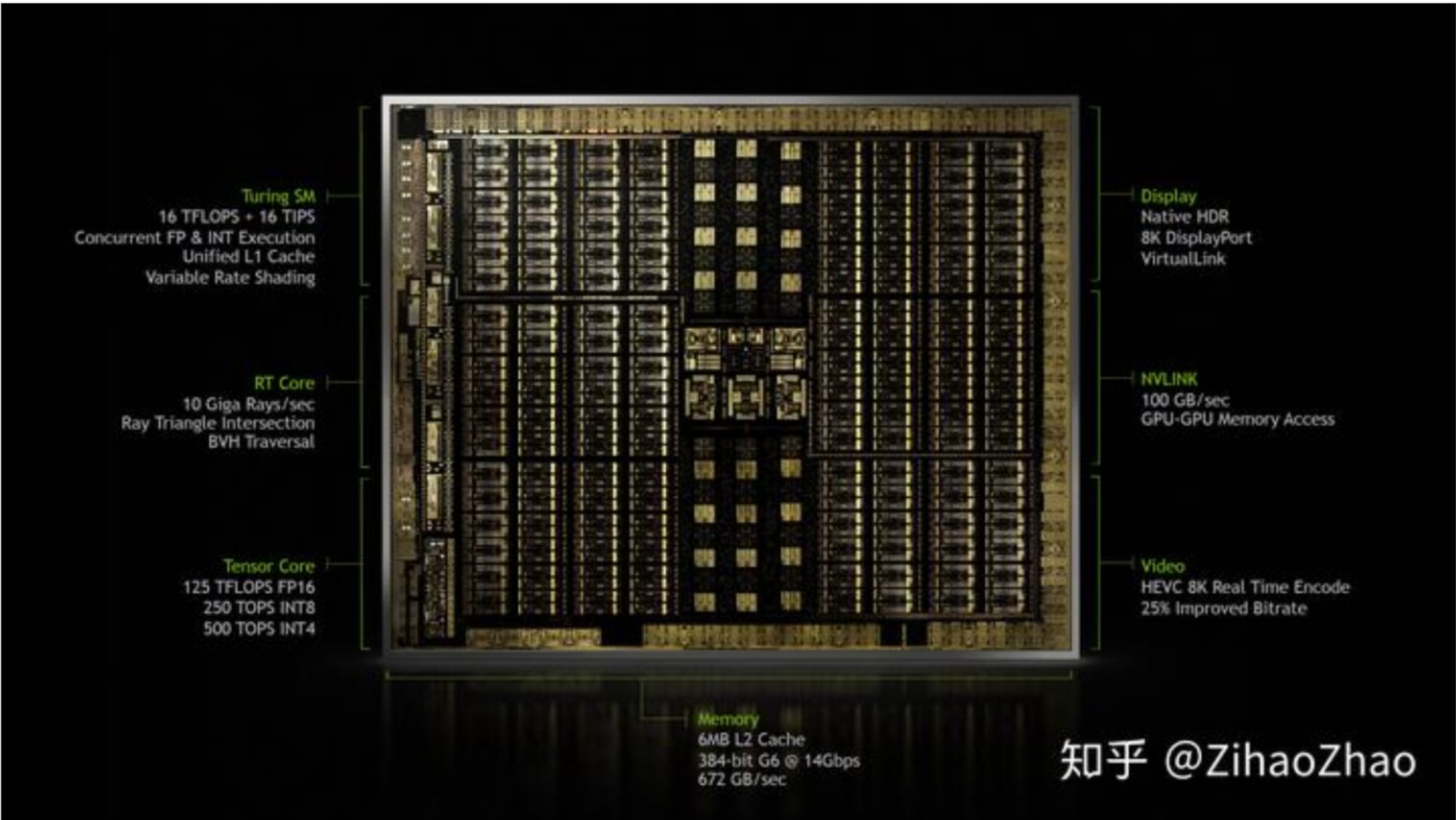
最新的NVIDIA GPU为Turing架构，继上一代的Pascal架构有了不小的改变。以Turing架构中我们最常见的RTX2080为例，RTX2080采用的是TU104核心，其架构如下图。与上文介绍的略有不同，我们能看到一个叫GPC（Graphics Processing Cluster）的层次，TU104共有6个GPC，其中每个GPC包括4个TPC（Texture Processing Cluster），每个TPC包含2个SM，总计有48个SM可供使用。而每个SM包含64个CUDA核（一般指FP32的数量），总计有3072个CUDA核可供使用。（感谢评论区幽玄大佬指正错误！）



再具体看其中每个SM的结构，如下图。Turing架构在SM中增加了新的处理单元，INT32。这让Turing架构的GPU有能力并行处理浮点数和非浮点数。官方宣称，理论上这将提高GPU在浮点操作时36%的吞吐量。还有每个SM上的L1缓存/共享内存也增加到了96KB。



最后来欣赏一下芯片内部结构，12nmFFN工艺制造的。官方渲染图也搞得十分精致，黑金范。



这一篇就这样吧，如果有什么地方没讲清或者漏掉的话，大家帮忙评论订正下。下一篇开始编程实践，通过一些小例程来巩固一下这两篇我们学到的基础知识。

欢迎各位留言交流，谢谢大家！

编辑于 2019-12-20 15:44

GPU 通用计算 CUDA 图形处理器（GPU）



发布一条带图评论吧

23 条评论

默认 最新



幽玄

gpc不是新概念，老早就有了，比如fermi gf100是4个gpc。
tu104每个gpc是4个tpc，每个tpc 2 个sm，所以每个gpc是8个sm，每个sm是64个cuda core（一般是指fp32的数量）。

2019-12-20

回复 4



ZihaoZhao 作者

感谢大佬指出，文章已修改，我是最近才知道这个概念

2019-12-20

回复 4



头像是狐狸吗

不愧是叔

2019-12-20

回复 喜欢



发布一条带图评论吧



汪阙

...

您好，我想问一下，如果在GPU的运算过程中，不可避免的使用了大量的if...else判断，在性能和结果上会出现一些什么状况？（例如我现在遇到的问题是相同的过程在CPU和GPU上的计算结果完全不同）

2020-11-16

回复 3



汪阙 ▸ 甜她个大头鬼

...

解决了，我的问题是随机数函数用错了

2021-07-17

回复 1



甜她个大头鬼

...

相同的过程在CPU和GPU上的计算结果完全不同
你好，这个问题你解决了吗？

2021-07-17

回复 1



dong

...

“当线程数由于任何理由闲置的时候”里的“线程数”应该说的是“线程束”吧？

2021-09-30

回复 3



王先森

...

1个cuda core运行1个thread？


2021-09-10

回复 2



sazc   ▸ 图楠

...

1个CUDA Core可以对应1个Thread，但因为流水线，里面可能正在运行着几个Thread


2022-08-31

回复 1



王先森 ▸ sazc

...

同时运行多个thread？

01-13

回复 喜欢

展开其他 1 条回复 >



铁锤哥哥

...

一个cuda core 上运行一个THREAD？如果那样，THREAD不就是固定数量 的吗？是不是一个CUDA CORE运行多个线程？

2020-05-19

回复 2



XingjingLu 

...

cuda的线程调度单元按照warp来理解，同一个时刻，每个stream上会有一个block在执行（含多个warp，每个warp32个线程），当所有warp执行完，则该block执行完。物理核数有限，但是线程数目可以远大于物理核数。

2021-10-13

回复 4



极致的混沌 ▶ XingjingLu

...

“同一个时刻，每个stream上会有一个block在执行”这里应该是这样，同一个时刻，每个stream上可能会有多个block在执行。

2023-07-31

● 回复 ♥ 喜欢



肉盾局特工

...

2080有46个SM，一共2944个CUDA core

2021-03-19

● 回复 ♥ 1



铁锤哥哥

...

一个BLOCK可以定义512个THREAD,为什么GPU硬件结构中, 一个SM只有32个CORE?

2020-05-19

● 回复 ♥ 1



孙培钦

...

有个问题请教一下: SM中的资源有shared memory和register, 文中写到 shared memory分配给常驻线程块, register分配给thread, 那理论上actived warp应该不受限制才对啊, 不然就是这些资源只能分配给actived warp. 那这样就有个问题: 当执行actived-warp与non-actived warp切换时, 资源难道可以释放? 释放数据不就丢失了吗?

2022-05-01

● 回复 ♥ 喜欢



BabyDog



...

在给SM分配常驻block的时候，会预计算所需寄存器文件大小。保证每个thread都有可用的寄存器，warp调度时候资源不会被释放。

2023-11-03

● 回复 ♥ 1



李嵘

...

感谢分享

2021-02-02

● 回复 ♥ 喜欢



Octopus

...

指令地址计数器和寄存器状态是指程序计算器和堆栈么，是的话是不是Volta架构才开始有独立的

2020-11-17

● 回复 ♥ 喜欢