Tesla GPU 架构分析

一、 GPU 简介

GPU 全称是 Graphics Processing Unit，图形处理单元。它的功能最初与名字一致，是专门用于绘制图像和处理图元数据的特定芯片，后来渐渐加入了其它功能。现代 GPU 除了绘制图形外，还担当了很多额外的功能，综合起来如下几方面：

1、图形绘制。这是 GPU 传统的拿手好戏，也是最基础、最核心的功能。为大多数 PC 桌面、移动设备、图形工作站提供图形处理和绘制功能。

2、物理模拟。GPU 硬件集成的物理引擎（PhysX、Havok），为游戏、电影、教育、科学模拟等领域提供了成百上千倍性能的物理模拟，使得以前需要长时间计算的物理模拟得以实时呈现。

3、海量计算。计算着色器及流输出的出现，为各种可以并行计算的海量需求得以实现，CUDA 就是最好的例证。

4、AI 运算。近年来，人工智能的崛起推动了 GPU 集成了 AI Core 运算单元，反哺 AI 运算能力的提升，给各行各业带来了计算能力的提升。

5、其它计算。音视频编解码、加解密、科学计算、离线渲染等等都离不开现代 GPU 的并行计算能力和海量吞吐能力。

二、 GPU 硬件架构

Tesla 微观架构总览图如下。

- 拥有 7 组 TPC（Texture/Processor Cluster，纹理处理簇）
- 每个 TPC 有两组 SM（Stream Multiprocessor，流多处理器）
- 每个 SM 包含：

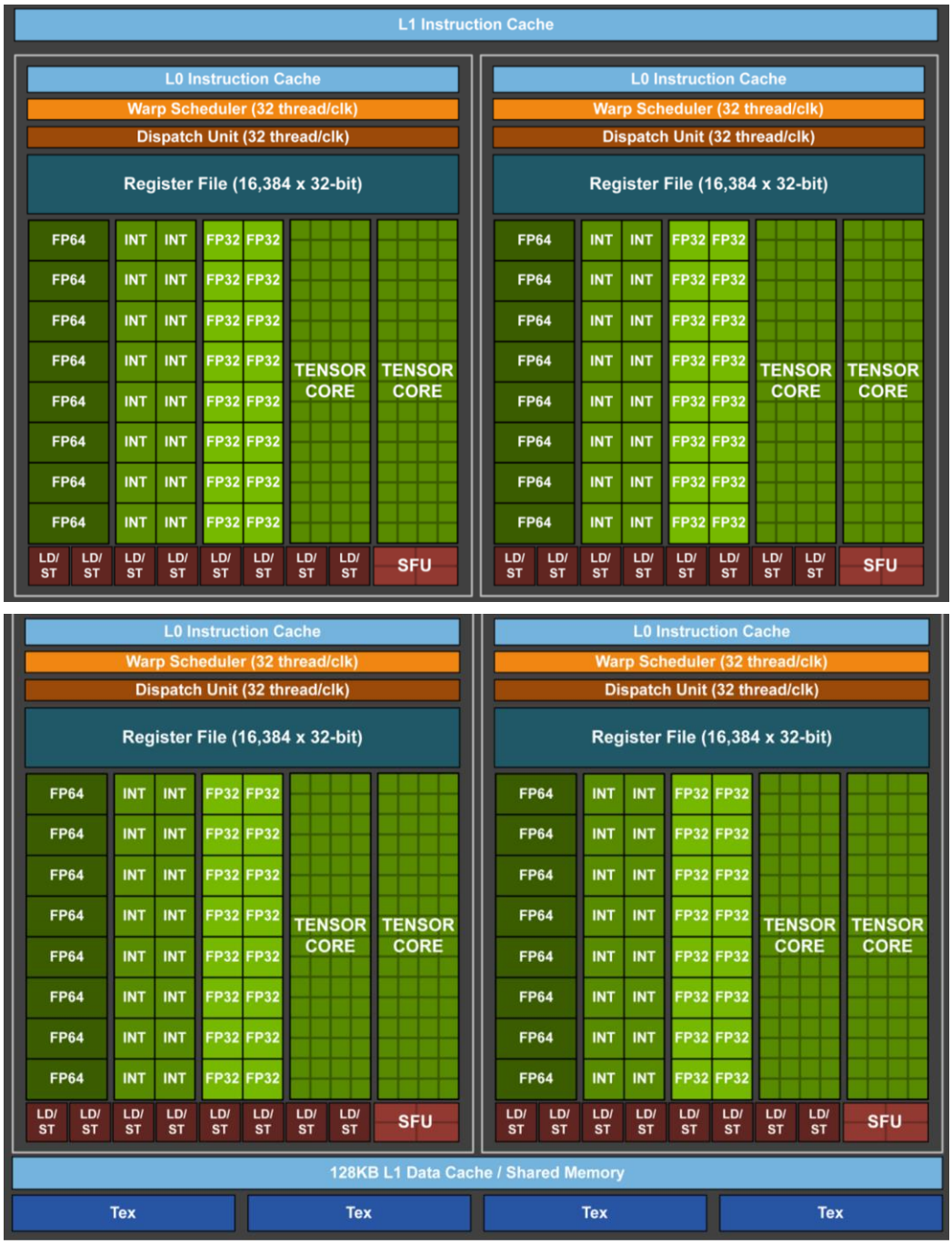
  6 个 SP（Streaming Processor，流处理器）

  2 个 SFU（Special Function Unit，特殊函数单元）

  L1 缓存、MT Issue（多线程指令获取）、C-Cache（常量缓存）、共享内存
- 除了 TPC 核心单元，还有与显存、CPU、系统内存交互的各种部件。

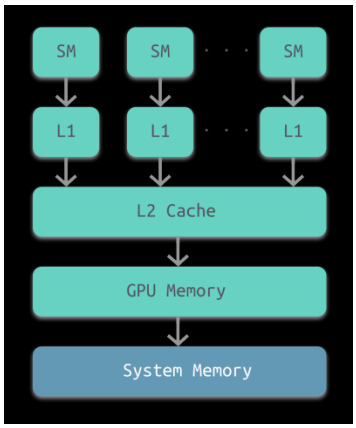

PCI Express 3.0 Host Interface
GigaThread Engine
GPC
TPC
SM
L2 Cache
Memory Controller
HBM2
High-Speed Hub
NVLink

V100 SM 结构图如下：



- 每个 SM 包含 64 个 FP32 内核和 32 个 FP64 内核。
- GV100 SM 被划分为四个处理块，每个处理块具有 16 个 FP32 内核、8 个 FP64 内核、16 个 INT32 内核、两个用于深度学习矩阵算术的混合精度的 Tensor 核心、一个新的 L0 指令缓存、一个变形调度程序、一个调度单元和一个 64 KB 注册文件。
- 共享内存和 L1 资源的合并使共享内存容量增加到每个 Volta SM 96 KB。

三、 GPU 资源运行机制



部分 GPU 的架构与 CPU 类似，也有多级缓存结构：寄存器、L1 缓存、L2 缓存、GPU 显存、系统显存。

各存储结构访问速度如下：

存储类型	寄存器	共享内存	L1缓存	L2缓存	纹理、常量缓存	全局内存
访问周期	1	1~32	1~32	32~64	400~600	400~600

由此可见，shader 直接访问寄存器、L1、L2 缓存还是比较快的，但访问纹理、常量缓存和全局内存非常慢，会造成很高的延迟。

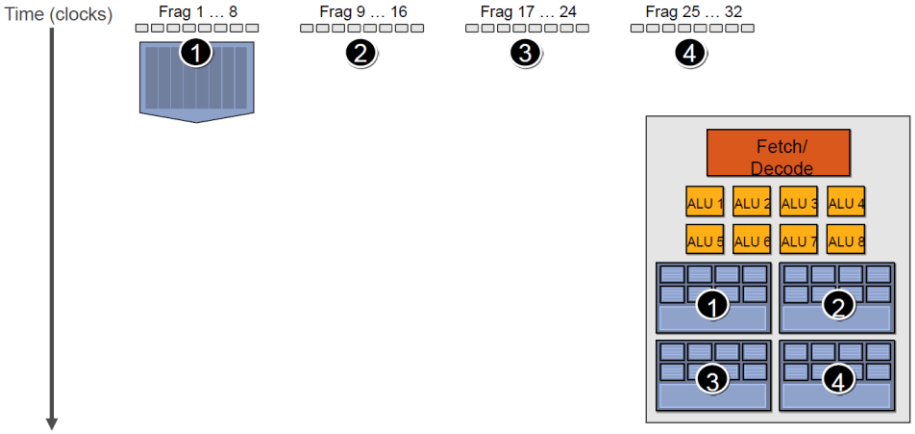
还有另外一部分 GPU-Style 的内存架构

More ALUs, no large traditional cache hierarchy:
Need high-bandwidth connection to memory

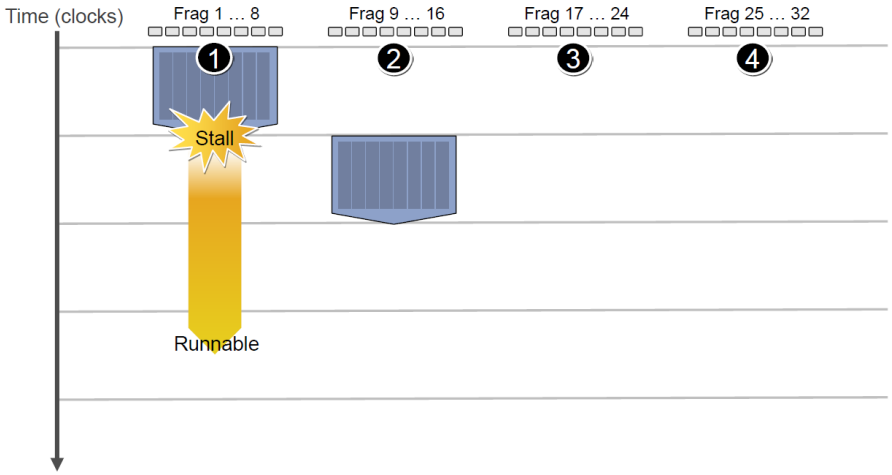
ALU 多，GPU 上下文（Context）多，吞吐量高，依赖高带宽与系统内存交换数据。

由于 SIMT 技术的引入，导致很多同一个 SM 内的很多 Core 并不是独立的，当它们当中有部分 Core 需要访问到纹理、常量缓存和全局内存时，就会导致非常大的卡顿（Stall）。

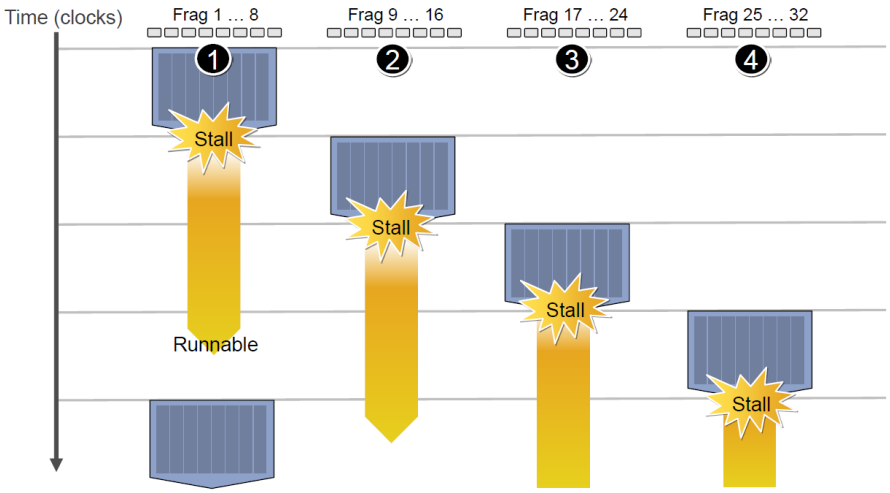
例如下图中，有 4 组上下文（Context），它们共用同一组运算单元 ALU。



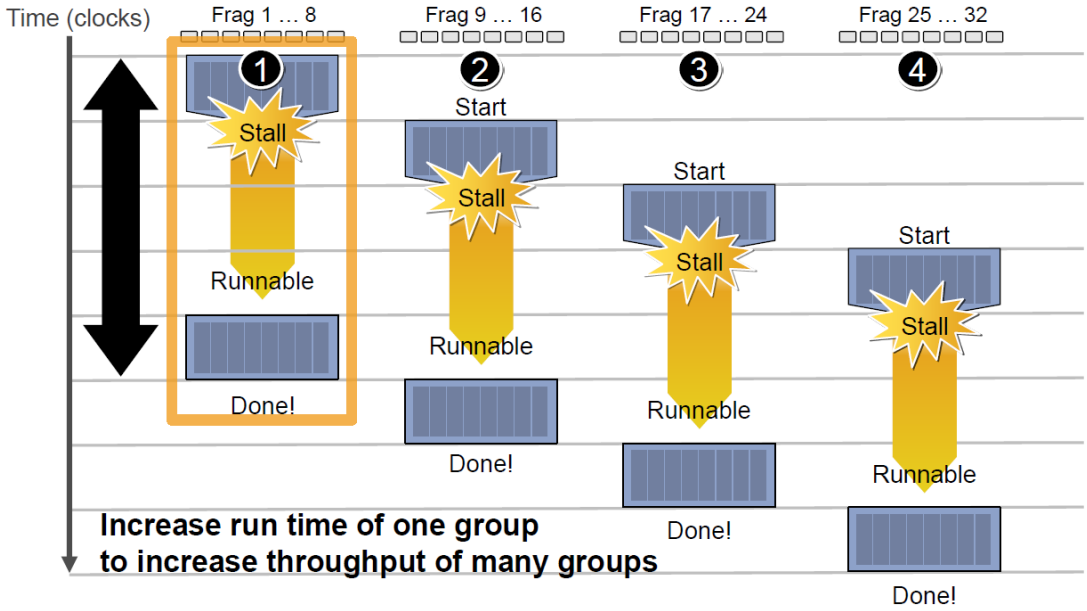
假设第一组 Context 需要访问缓存或内存，会导致 2~3 个周期的延迟，此时调度器会激活第二组 Context 以利用 ALU：



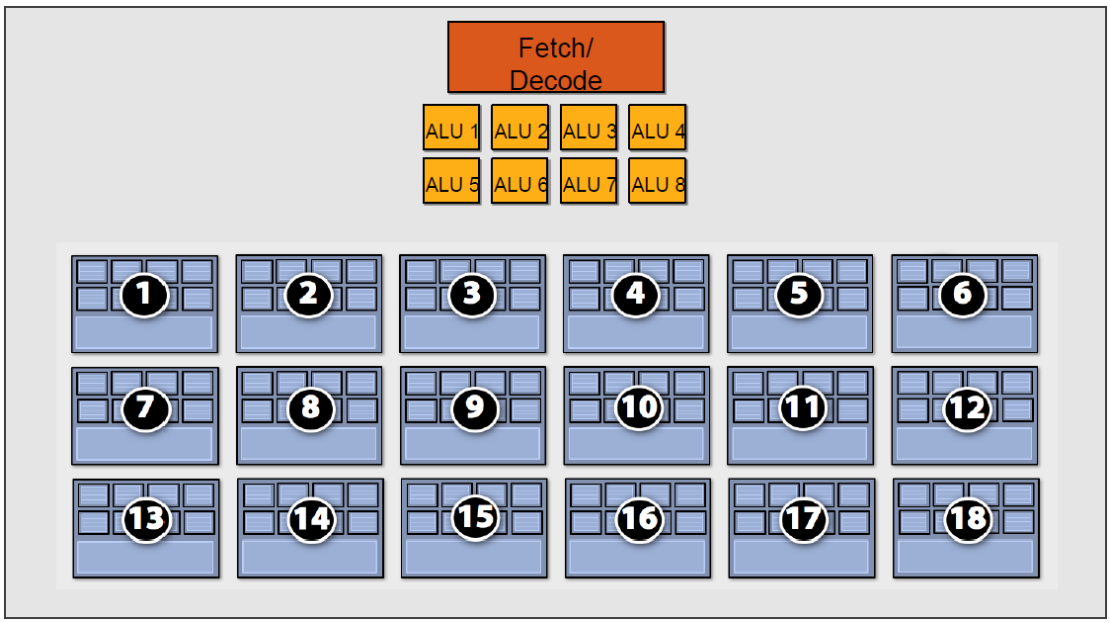
当第二组 Context 访问缓存或内存又卡住，会依次激活第三、第四组 Context，直到第一组 Context 恢复运行或所有都被激活：



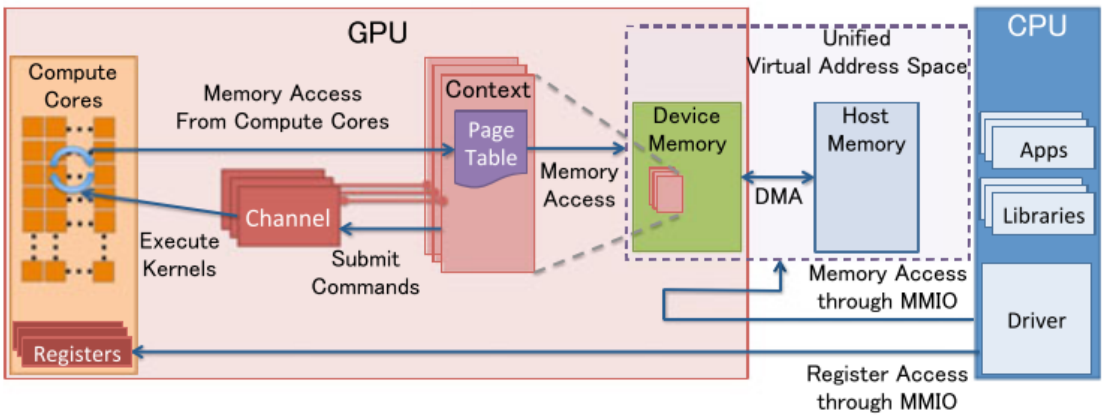
延迟的后果是每组 Context 的总体执行时间被拉长了：



但是，越多 Context 可用就越可以提升运算单元的吞吐量，比如下图的 18 组 Context 的架构可以最大化地提升吞吐量：



下图是 GPU 资源管理模型图



- MMIO（Memory Mapped IO）

  CPU 与 GPU 的交流就是通过 MMIO 进行的。CPU 通过 MMIO 访问 GPU 的寄存器状态。

  DMA 传输大量的数据就是通过 MMIO 进行命令控制的。

  I/O 端口可用于间接访问 MMIO 区域，像 Nouveau 等开源软件从来不访问它。

- GPU Context

  GPU Context 代表了 GPU 计算的状态。

  在 GPU 中拥有自己的虚拟地址。

  GPU 中可以并存多个活跃态下的 Context。

- GPU Channel

  任何命令都是由 CPU 发出。

  命令流（command stream）被提交到硬件单元，也就是 GPU Channel。

  每个 GPU Channel 关联一个 context，而一个 GPU Context 可以有多个 GPU channel。

  每个 GPU Context 包含相关 channel 的 GPU Channel Descriptors ， 每个 Descriptor 都是 GPU 内存中的一个对象。

  每个 GPU Channel Descriptor 存储了 Channel 的设置，其中就包括 Page Table 。

  每个 GPU Channel 在 GPU 内存中分配了唯一的命令缓存，这通过 MMIO 对 CPU 可见。

  GPU Context Switching 和命令执行都在 GPU 硬件内部调度。

- GPU Page Table

  GPU Context 在虚拟基地空间由 Page Table 隔离其它的 Context 。

  GPU Page Table 隔离 CPU Page Table，位于 GPU 内存中。

  GPU Page Table 的物理地址位于 GPU Channel Descriptor 中。

  GPU Page Table 不仅仅将 GPU 虚拟地址转换成 GPU 内存的物理地址，也可以转换成 CPU 的物理地址。因此，GPU Page Table 可以将 GPU 虚拟地址和 CPU 内存地址统一到 GPU 统一虚拟地址空间来。

- PCI-e BAR

GPU 设备通过 PCI-e 总线接入到主机上。 Base Address Registers(BARs) 是 MMIO 的窗口，在 GPU 启动时候配置。

GPU 的控制寄存器和内存都映射到了 BARs 中。

GPU 设备内存通过映射的 MMIO 窗口去配置 GPU 和访问 GPU 内存。

- PFIFO Engine

  PFIFO 是 GPU 命令提交通过的一个特殊的部件。

  PFIFO 维护了一些独立命令队列，也就是 Channel。

  此命令队列是 Ring Buffer，有 PUT 和 GET 的指针。

  所有访问 Channel 控制区域的执行指令都被 PFIFO 拦截下来。

  GPU 驱动使用 Channel Descriptor 来存储相关的 Channel 设定。

  PFIFO 将读取的命令转交给 PGRAPH Engine。

- BO

  Buffer Object (BO)，内存的一块(Block)，能够用于存储纹理（Texture）、渲染目标（Render Target）、着色代码（shader code）等等。

附录：

Table 1. Comparison of NVIDIA Tesla GPUs

Tesla Product	Tesla K40	Tesla M40	Tesla P100	Tesla V100
GPU	GK180 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)	GV100 (Volta)
SMs	15	24	56	80
TPCs	15	24	28	40
FP32 Cores / SM	192	128	64	64
FP32 Cores / GPU	2880	3072	3584	5120
FP64 Cores / SM	64	4	32	32
FP64 Cores / GPU	960	96	1792	2560
Tensor Cores / SM	NA	NA	NA	8
Tensor Cores / GPU	NA	NA	NA	640
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz	1530 MHz
Peak FP32 TFLOPS[1]	5	6.8	10.6	15.7
Peak FP64 TFLOPS[1]	1.7	.21	5.3	7.8
Peak Tensor TFLOPS[1]	NA	NA	NA	125
Texture Units	240	192	224	320
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB	6144 KB
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB	Configurable up to 96 KB
Register File Size / SM	256 KB	256 KB	256 KB	256KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB	20480 KB
TDP	235 Watts	250 Watts	300 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion	21.1 billion
GPU Die Size	551 mm^2	601 mm^2	610 mm^2	815 mm^2
Manufacturing Process	28 nm	28 nm	16 nm FinFET+	12 nm FFN

[1] Peak TFLOPS rates are based on GPU Boost Clock