Cuda Tensor Core

要保持张量核心持续运行并不容易。

研究人员发现GPU硬件具有一些特性，对于保持矩阵乘法的运行非常重要：

- WGMMA指令虽然是必要的，但使用起来颇为麻烦。
- 共享内存的速度并不如预期的快，使用时还需格外注意。
- 生成地址的成本较高。
- 保持高占用率对于提升性能是有益的，寄存器至关重要

早期GPU中的张量核心指令如wmma.mma.sync和mma.sync，要求SM一个子单元内的32个线程的一个warp同步传输数据块至张量核心并等待结果。

wgmma.mma_async指令则不同。它允许128个连续线程跨SM所有子单元协作同步，并从共享内存及寄存器（可选）异步启动矩阵乘法。这使得这些warp在等待矩阵乘法结果时可以处理其他任务。

然而，这些指令的内存布局极其复杂。未重排的共享内存布局合并性差，需要额外的L2带宽。重排的内存布局记录不准确，研究人员花费了大量时间才弄明白。最终发现，这些布局只适用于特定矩阵形状，并与wgmma.mma_async指令的其他部分不兼容，例如硬件仅在未重排的布局下转置子矩阵。

此外，未重排的wgmma布局内存合并性差且有bank conflicts。尽管TMA和L2缓存在如flash attention这类内核上能较好地掩盖这些问题，但要充分利用硬件，必须精心控制内存请求的合并和避免bank conflicts。

TMA允许用户在全局和共享内存中指定多维张量布局，命令其异步提取张量的一部分，并在完成后触发一个屏障。这大大节省了地址生成的开销，并简化了pipelines的构建。

	Volta	**Turing**	**Ampere**	**Hopper**
Base Size*	64个 F16(4x4x4)	64个 F16(4x4x4)	256个 F16(8x4x8)	512个 F16(8x4x16)
tensorCore个数	672(8/SM)	576(8/SM)	512(4/SM)	576(4/SM)
SM内算力 *	512个 F16	512个 F16	1024个 F16	2048个 F16
总算力 FOPS(F16)	125T	130.5T	312T	1000T
boost clock(MHz)	1530	1455	1410	≈A100x1.3
FP64	-	-	√	√
TF32	-	-	√	√
BF16	-	-	√	√
FP16	√	√	√	√

	Volta	Turing	Ampere	Hopper
FP8	-	-	-	√
INT8/UINT8	-	√	√	√
INT4/UINT4	-	√	√	-
INT1	-	√	√	-

	Volta	Turing	Ampere	Hopper
SM/TPC	2	2	2	2
process blocks/SM	4	4	4	4
FP64/PB	8	-	8	16
FP32/PB	16	16	16	32
INT32/PB	16	16	16	16
tensorCore/PB	2	2	1	1
LSU/PB	8	4	8	8
register file	64KB(16384*32bit)	64KB(16384*32bit)	64KB(16384*32bit)	64KB(16384*32bit)
L0 ICache/PB	1	1	1	1
L1/SHM/SM	128KB	96KB	192KB	256KB
warp scheduler/PB	1(32thread/clk)	1(32thread/clk)	1(32thread/clk)	1(32thread/clk)
dispatch uint/PB	1(32thread/clk)	1(32thread/clk)	1(32thread/clk)	1(32thread/clk)

[![image.png](Cuda Tensor Core/Ubsimage-png.png)](Cuda Tensor Core/Ubsimage-png.png)

tensorCore的启动是依靠 SM内的 thread（ warp）共同完成的，每个 thread
能提供的数据个数是相同的，因此统计这个算力能有助于分析 SM内部的 thread和 tensorCore之间的调度方式

Revision #2
Created 2025-01-11 09:46:27 UTC by Colin
Updated 2026-04-29 07:33:23 UTC by Colin