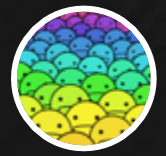
AI 芯片 – GPU 详解

Tensor Core 架构演进



ZOMI

Talk Overview

1. **硬件基础**
   - GPU 工作原理
   - GPU AI编程本质
2. **英伟达 GPU 架构**
   - GPU基础概念
   - 从 Fermi 到 Volta 架构
   - Turing 到 Hopper 架构
   - Tensor Code 和 NVLink 详解
3. **GPU 图形处理**
   - GPU 逻辑模块划分
   - 算法到 GPU 硬件
   - GPU 的软件栈
   - 图形流水线基础
   - 流水线不可编译单元
   - 光线跟踪流水线

Talk Overview

I. 基本工作原理

- Conv and TC Relationship - 卷积与Tensor Core关系
- TC Basic Principles - Tensor Core的基本原理
- TC Architecture Evolution - Tensor Core架构演进

Talk Overview

1. 基本工作原理

- Conv and TC Relationship - 卷积与Tensor Core关系
- TC Basic Principles - Tensor Core的基本原理
- TC Architecture Evolution - Tensor Core架构演进

2. 深入Tensor Core

- XXX
- XXX

NVIDIA GPU架构发展

架构名称	Volta	Turing	Ampere	Hopper
中文名字	伏特	图灵	安培	赫柏
发布时间	2017	2018	2020	2022
核心参数	80个SM，每个SM包括32个FP64+64 Int32+64 FP32+8个Tensor Cores	102核心92个SM，SM重新设计，每个SM包含64个Int32+64个FP32+8个Tensor Cores	108个SM，每个SM包含64个FP32+64个INT32+32个FP64+4个Tensor Cores	132个SM，每个SM包含128个FP32+64个INT32+64个FP64+4个Tensor Cores
特点&优势	NVLink2.0，Tensor Cores第一代，支持AI运算	Tensor Core2.0，RT Core第一代	Tensor Core3.0，RT Core2.0，NVLink3.0，结构稀疏性矩阵MIG1.0	Tensor Core4.0，NVlink4.0，结构稀疏性矩阵MIG2.0
纳米制程	12nm 211亿晶体管	12nm 186亿晶体管	7nm 283亿晶体管	4nm 800亿晶体管
代表型号	V100 TiTan V	T4，2080TI RTX 5000	A100 A30系列	H100

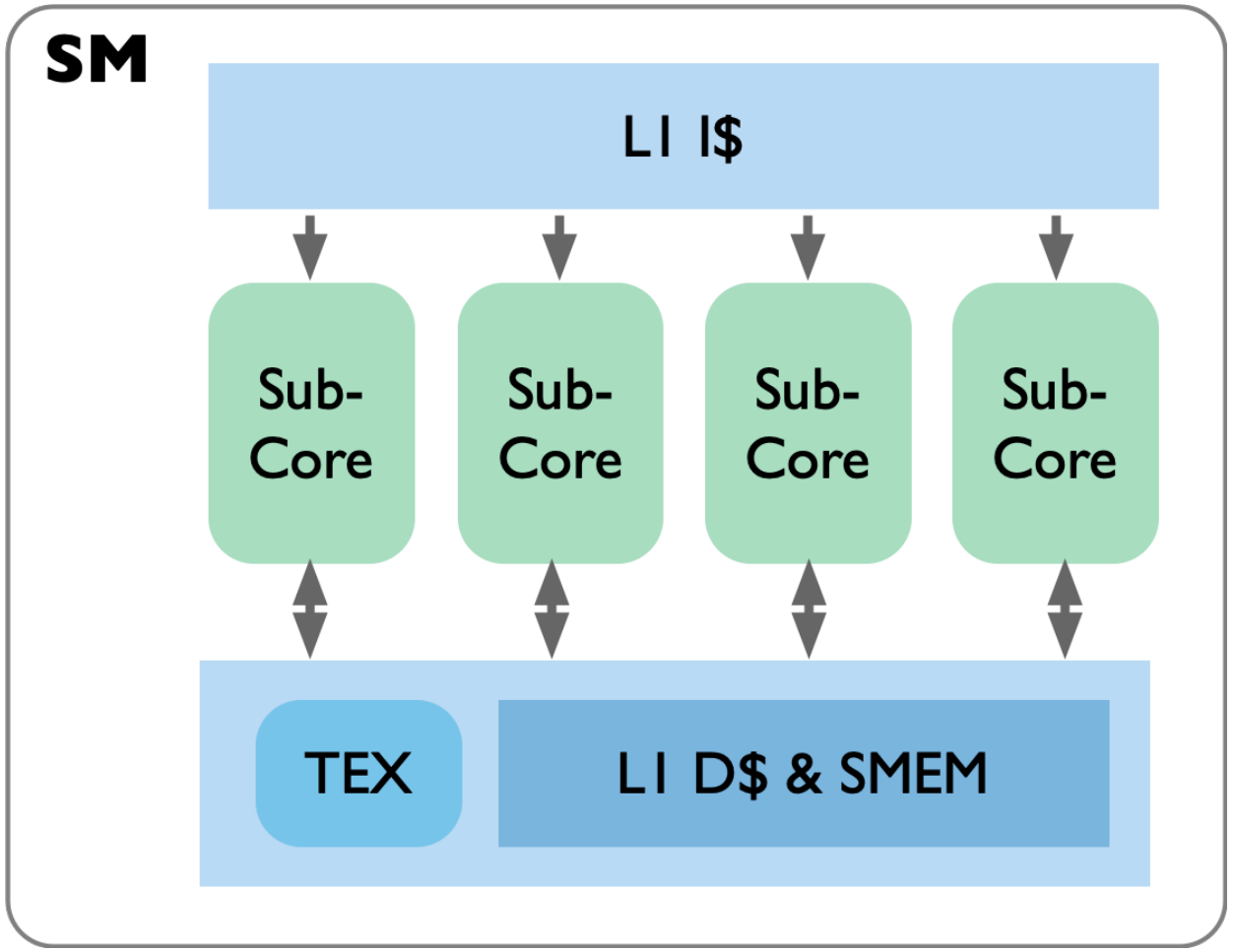
Tensor Core
历代发展

Tensor Core 历代发展

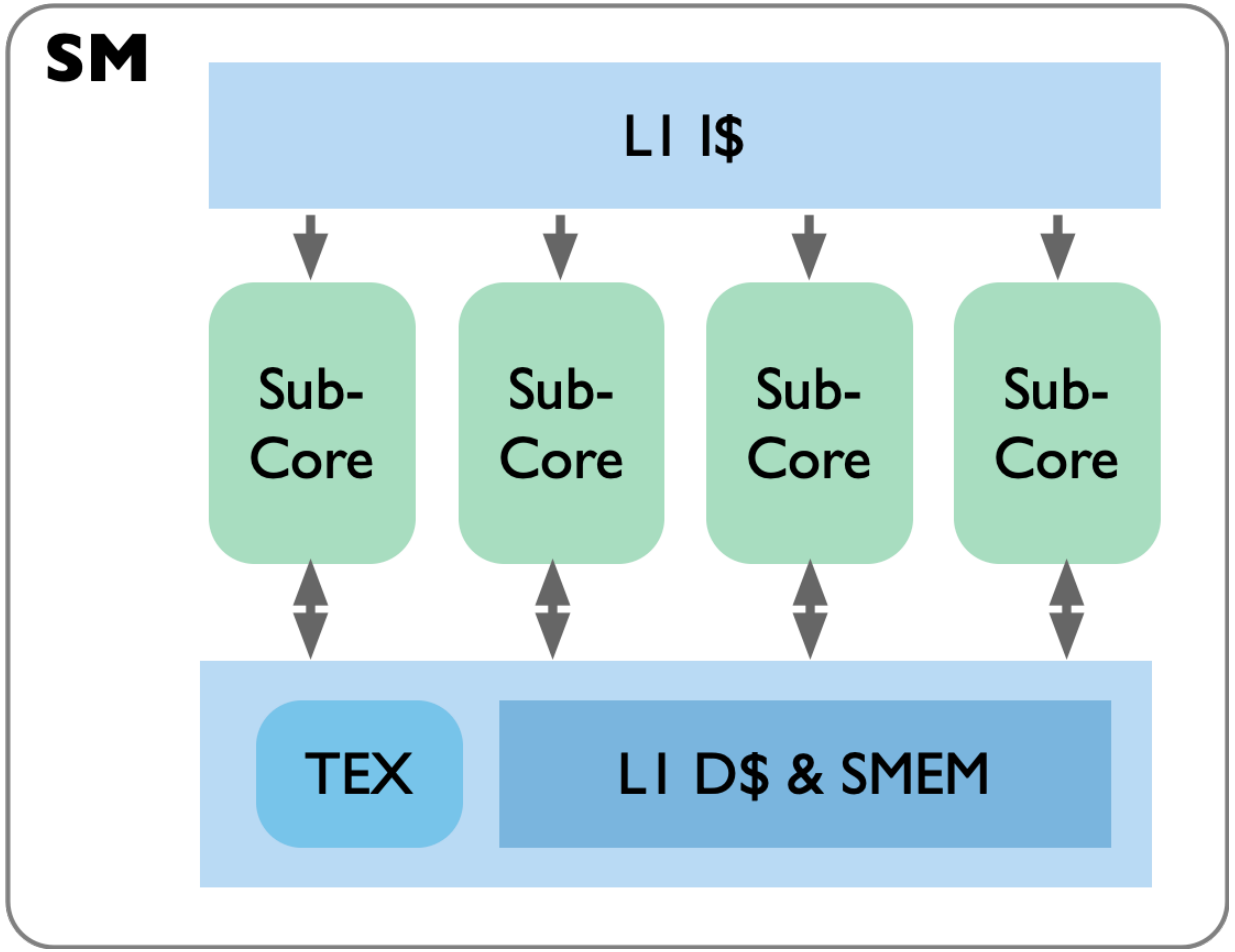
	Supported CUDA Core Precisions									Supported Tensor Core Precisions								
	FP8	FP16	FP32	FP64	INT1	INT4	INT8	TF32	BF16	FP8	FP16	FP32	FP64	INT1	INT4	INT8	TF32	BF16
NVIDIA Tesla P4	No	No	Yes	Yes	No	No	Yes	No	No	No	No	No	No	No	No	No	No	No
NVIDIA P100	No	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No
NVIDIA Volta	No	Yes	Yes	Yes	No	No	Yes	No	No	No	Yes	No	No	No	No	No	No	No
NVIDIA Turing	No	Yes	Yes	Yes	No	No	Yes	No	No	No	Yes	No	No	Yes	Yes	Yes	No	No
NVIDIA A100	No	Yes	Yes	Yes	No	No	Yes	No	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
NVIDIA H100	No	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	Yes	No	Yes	No	No	Yes	Yes	Yes

VOLTA SM


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
FP64
INT
FP32
TENSOR CORE
LD/ST
SFU
128KB L1 Data Cache / Shared Memory
Tex

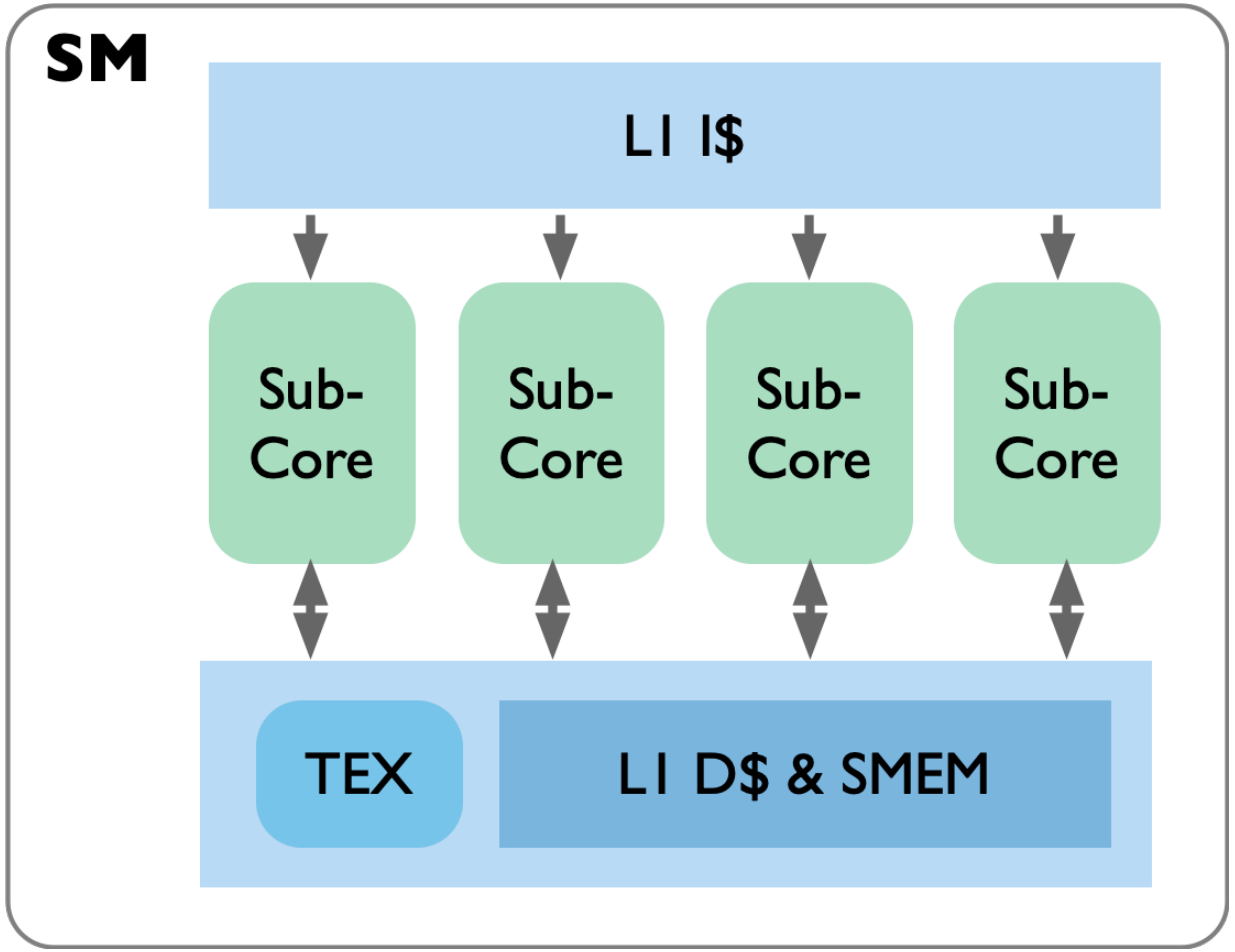

SM
L1 I$
Sub-Core
Sub-Core
Sub-Core
Sub-Core
TEX
L1 D$ & SMEM

VOLTA SM 整体架构



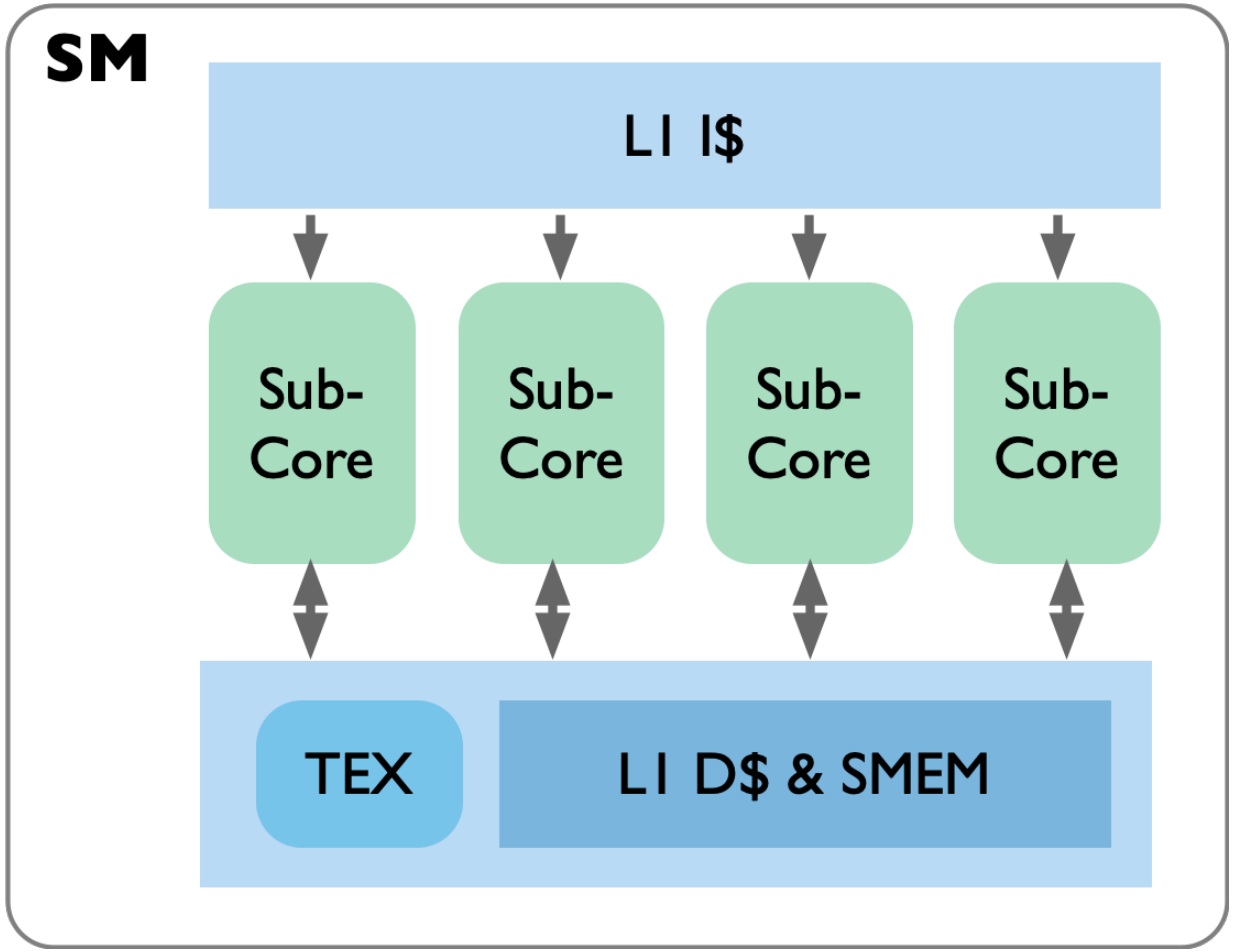
- Twice The Schedulers
- Simplified issue Logic
- Large, Fast L1 Cache
- Improved SIMT Model
- Tensor Acceleration
- +50% energy efficiency vs GP100 SM

第一代 Tensor Core (Volta)



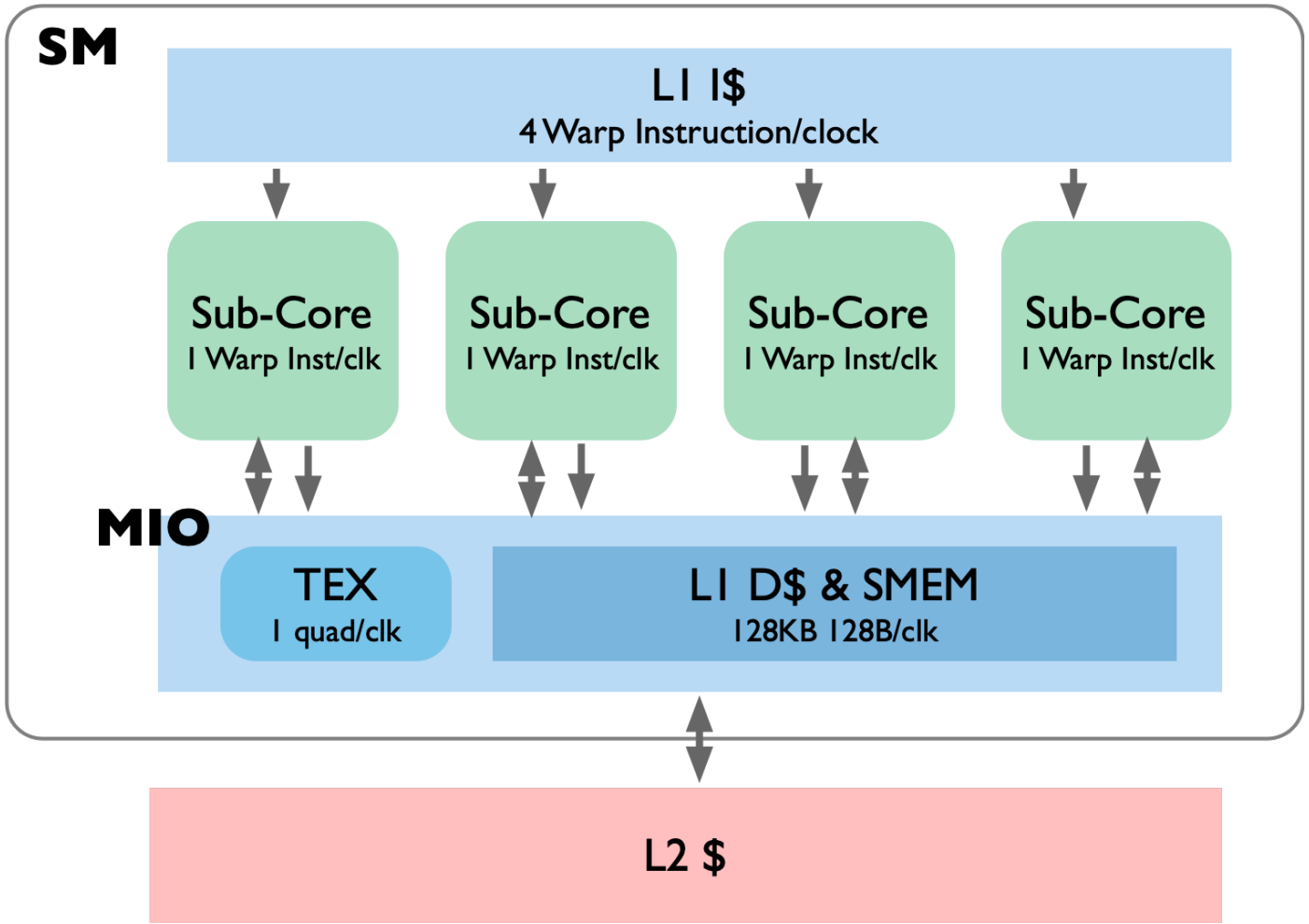
- **SM**：Sub-Core 上计算单元，负责对于寄存器整体的读写逻辑；
- **Sub-Core**： 包括 Tensor Core + FP64 + FP32 + INT8 + 特殊函数处理单元 MFU；
- **Warp Schedule**：计算单元都由 Warp Scheduler 调度执行，数据存储在寄存器；

第一代 Tensor Core (Volta)



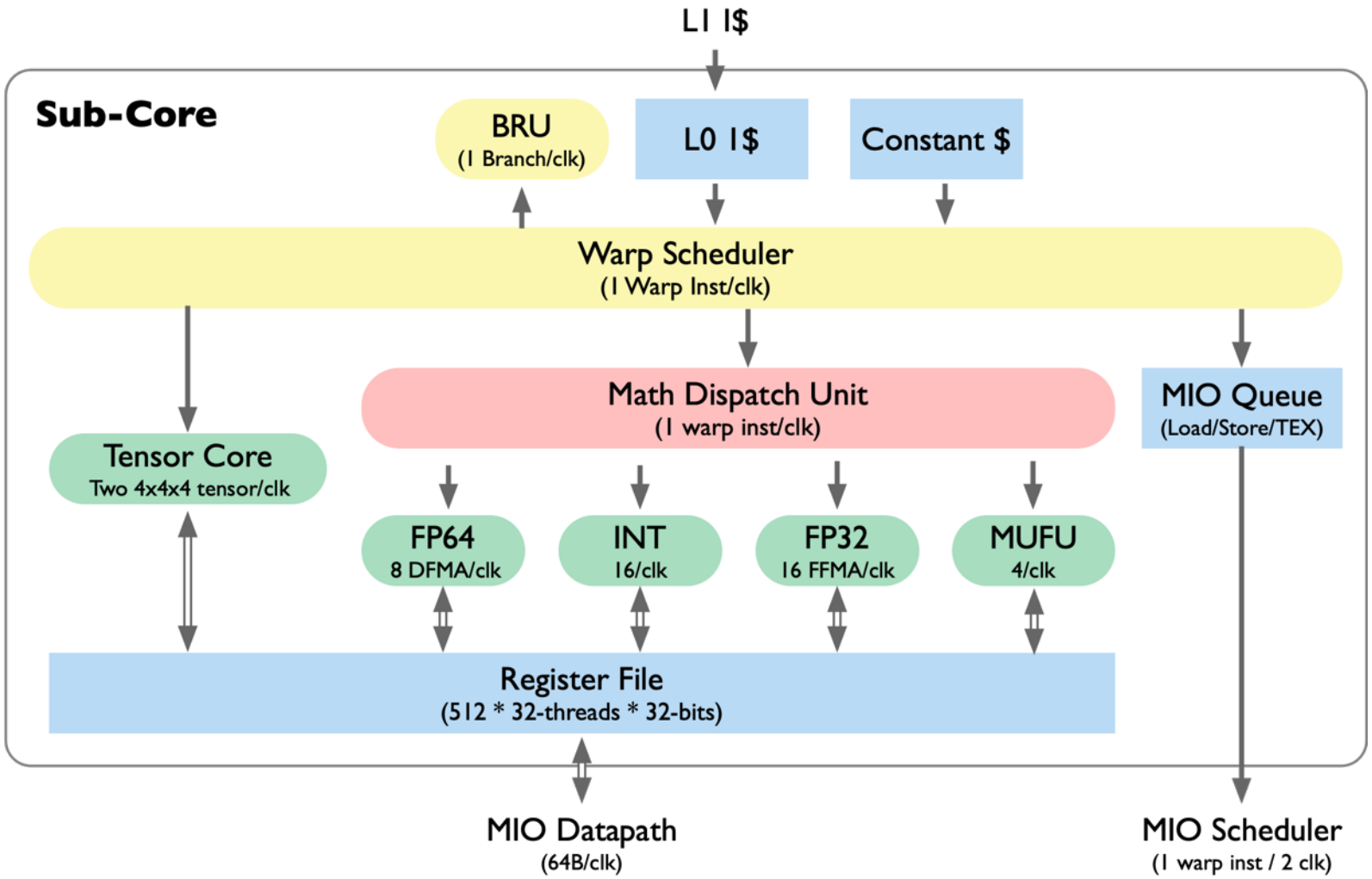
- 每个 SM Sub-Core 包含两个 4x4x4 Tensor Core，其中 Warp Scheduler 向 Tensor Core 发送矩阵乘法GEMM 运算指令。
- Tensor Core 从寄存器 (Register File) 接收输入矩阵 (A、B、C)，执行多次 4x4x4 矩阵乘法，直到完成完整的矩阵乘法，并将所得矩阵写回寄存器堆 (Register File)；

VOLTA SM 微架构



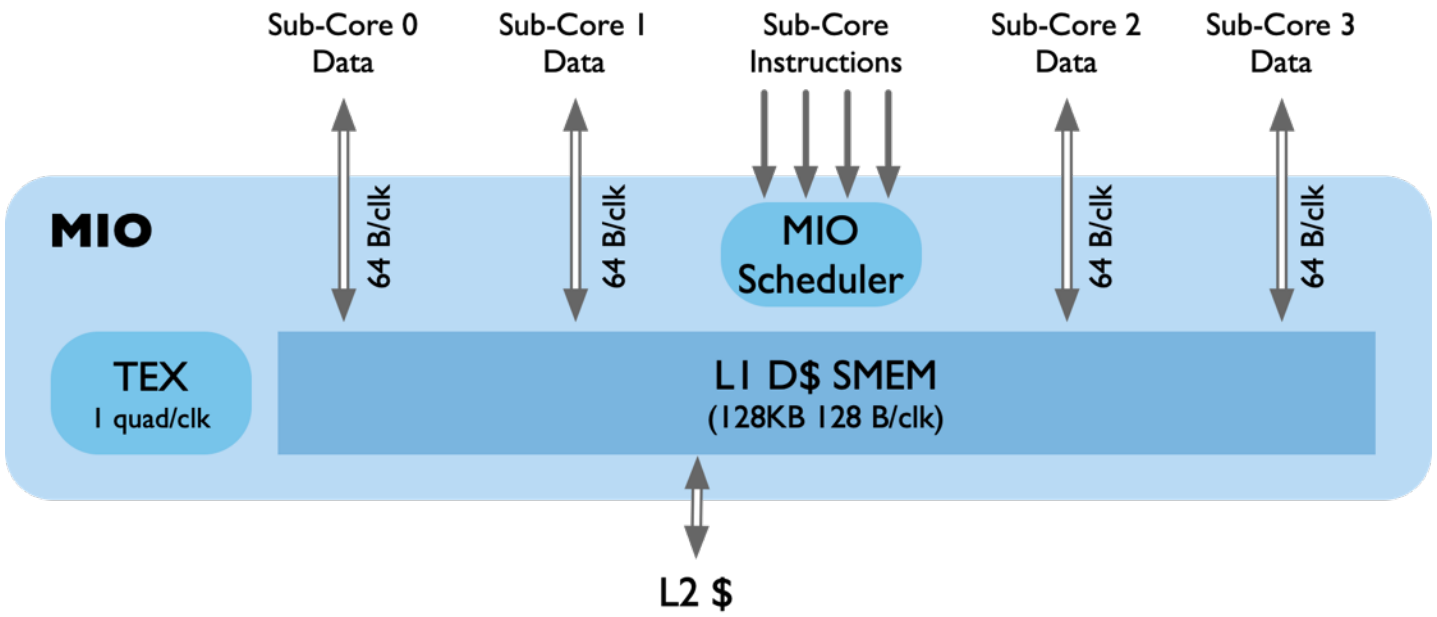
- Shared L1 I$ 共享缓存
- 4 个独立的 Sub-Cores
- Shared MIO (TEX/L1$/SMEM)

VOLTA Sub-Core 微架构



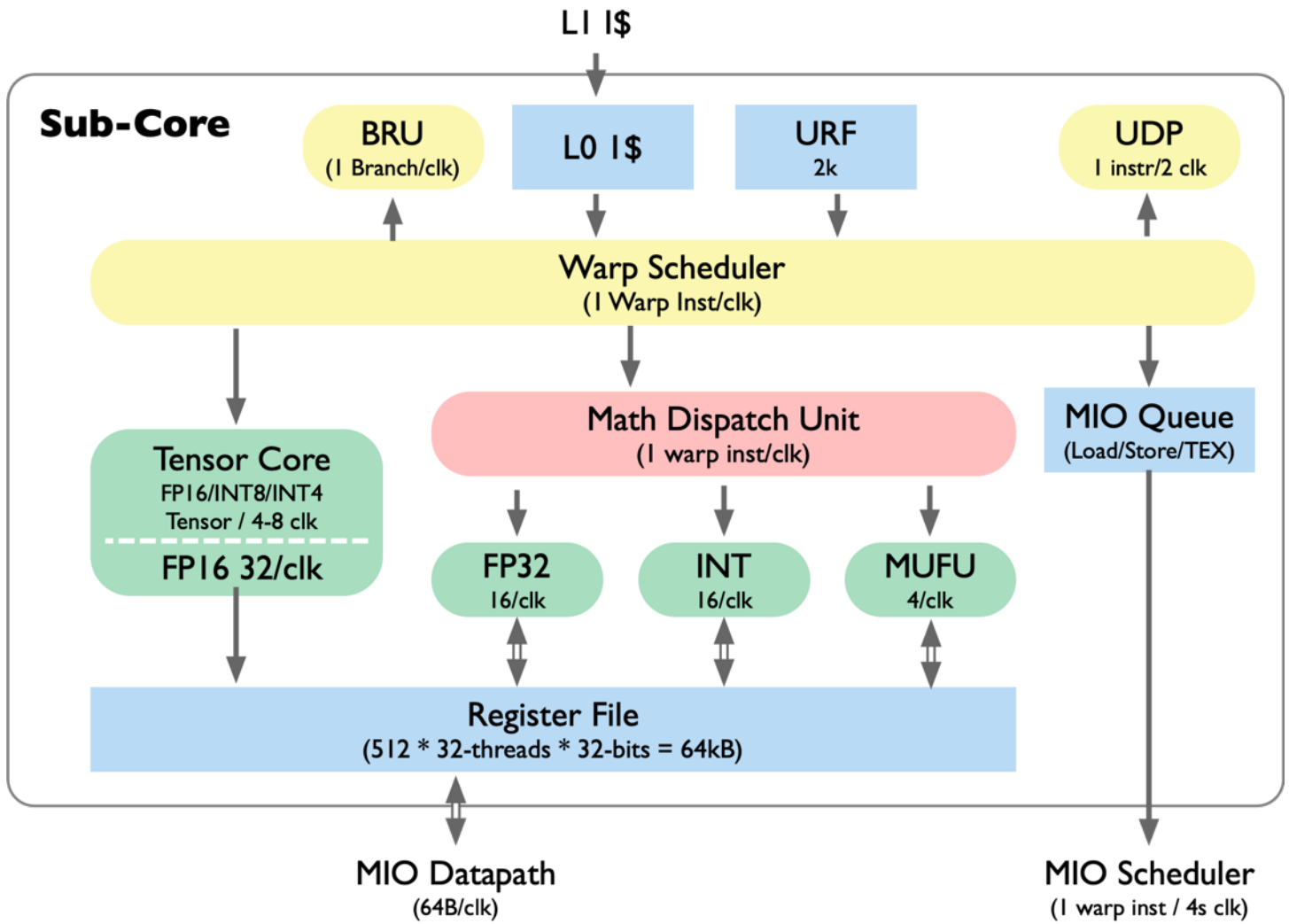
- **Warp Scheduler**
  - 1 Warp instr/clk
  - L0 I$, branch unit
- **Math Dispatch Unit**
  - Keeps 2+ Datapaths Busy
- **MIO Instruction Queue**
  - Hold for later Scheduling
- **Tensor Cores**
  - Two 4x4x4 matric

L1缓存和共享内存



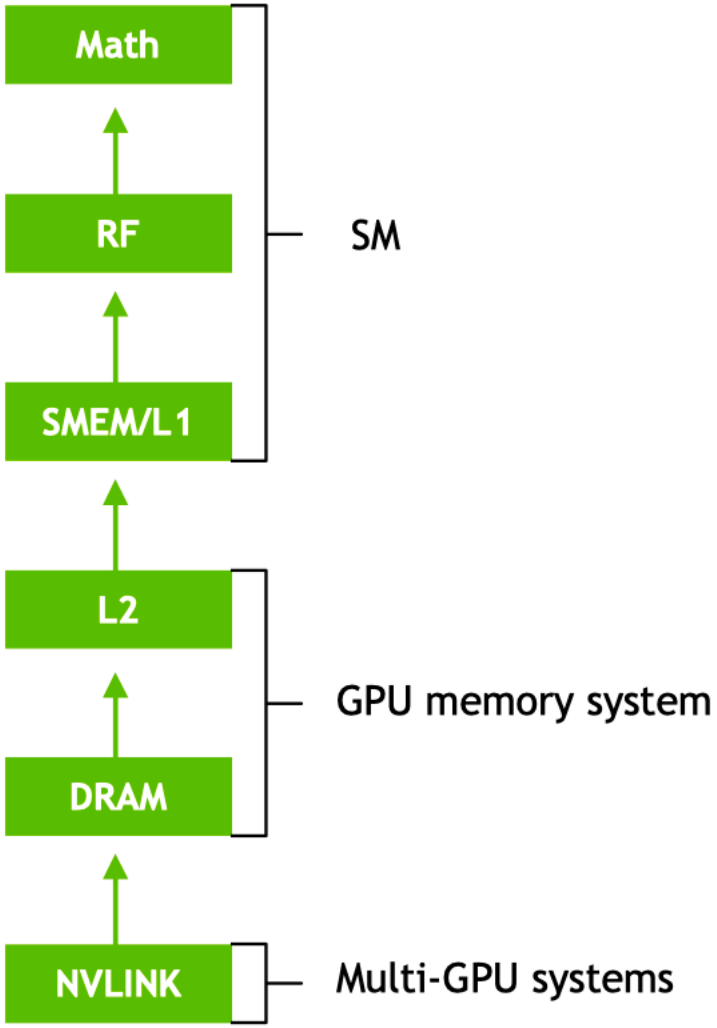
- **Streaming L1$**
  - Unified Cache misses in fight
  - Low cache hit latency
  - 4x bandwidth vs P100
  - 4x capacity vs P100
- **Shared Memory**
  - Unified Storage with L1$
  - Configurable up to 96KB

第二代 Tensor Core (Turing)



- Tensor Core 中增加 Int8 和 Int4 类型支持，多了 FP16 Fast path；
- 在 4-8 个时钟周期内可以执行单个多线程 (multi-thread) GEMM 数学运算；
- 通过线程 Thread 共享数据 local memory 来节省数据搬运操作和内存带宽；

第三代 Tensor Core (Ampere)



- A100 SM DATA MOVEMENT EFFICIENCY
- 3x SMEM/L1 Bandwidth, 2x In-flight Capacity

第三代 Tensor Core (Ampere)

- Ampere 之前的 GPU 架构中，如果要使用共享内存（Shared Memory），必须先把数据从全局内存（Global Memory）加载到寄存器中，然后再写入共享内存。

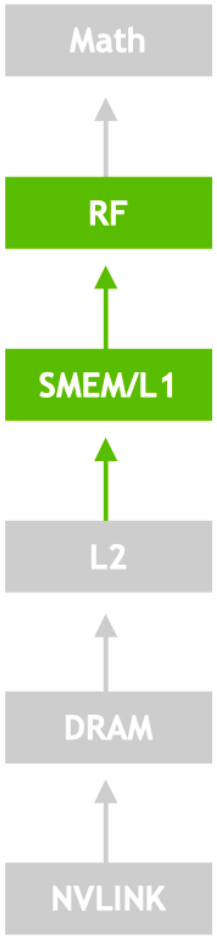
Pros：

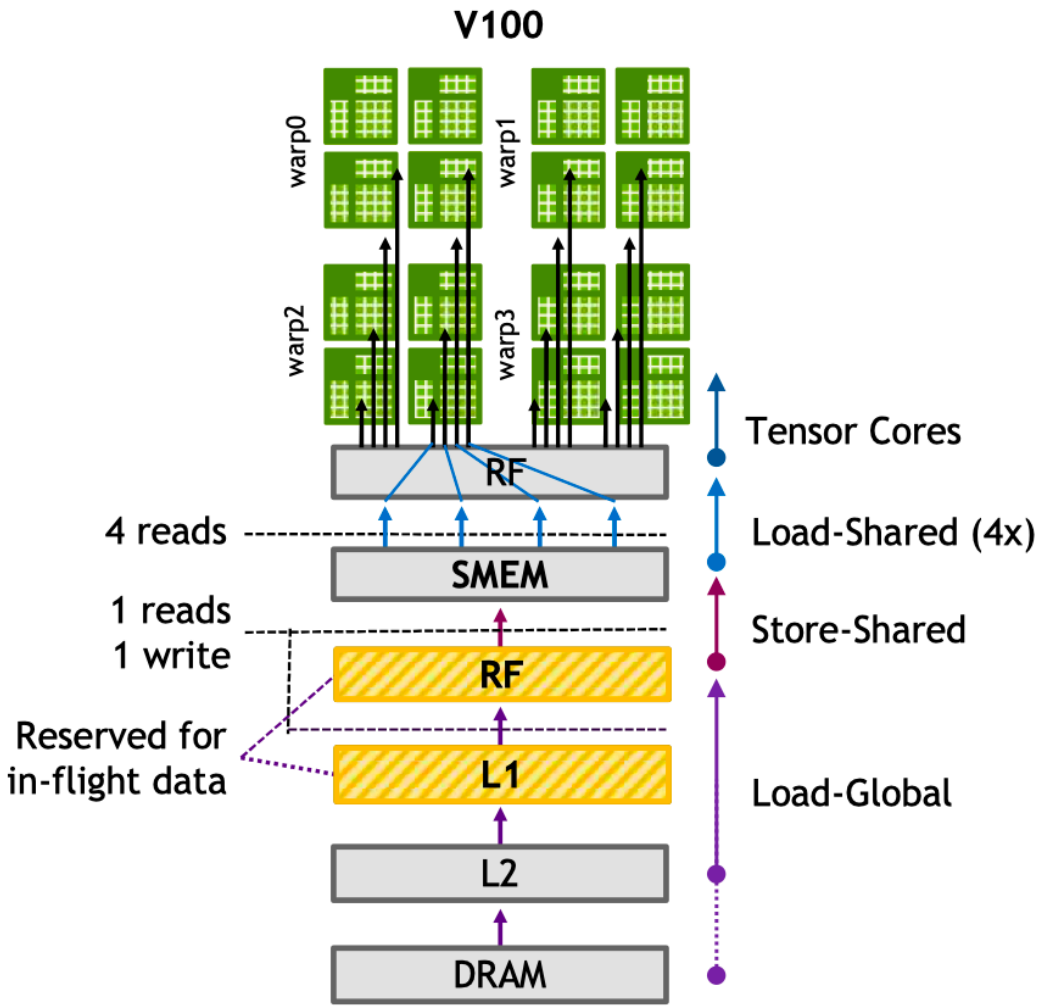
- 浪费部分寄存器资源，同时增加了数据搬运的时延 latency；

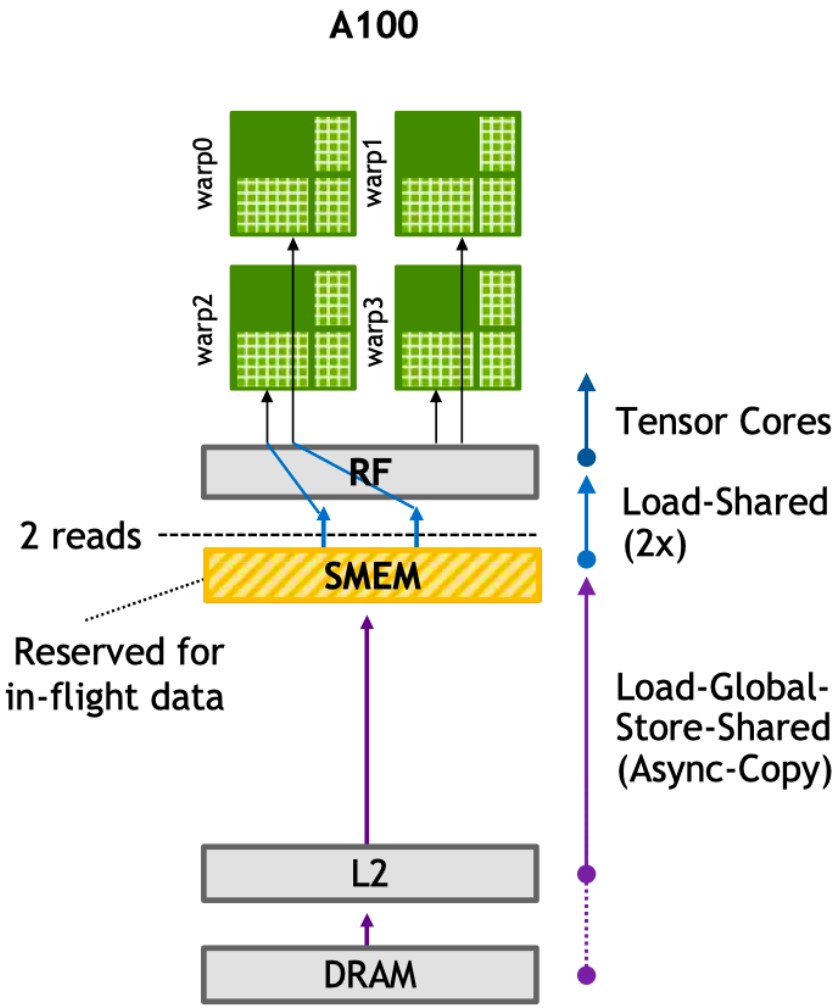
Cons：

- Ampere 架构中提供异步内存拷贝机制，通过 LDGSTS SASS 指令（Load Global Store Shared）实现全局内存不经过寄存器直接到共享内存的数据加载；

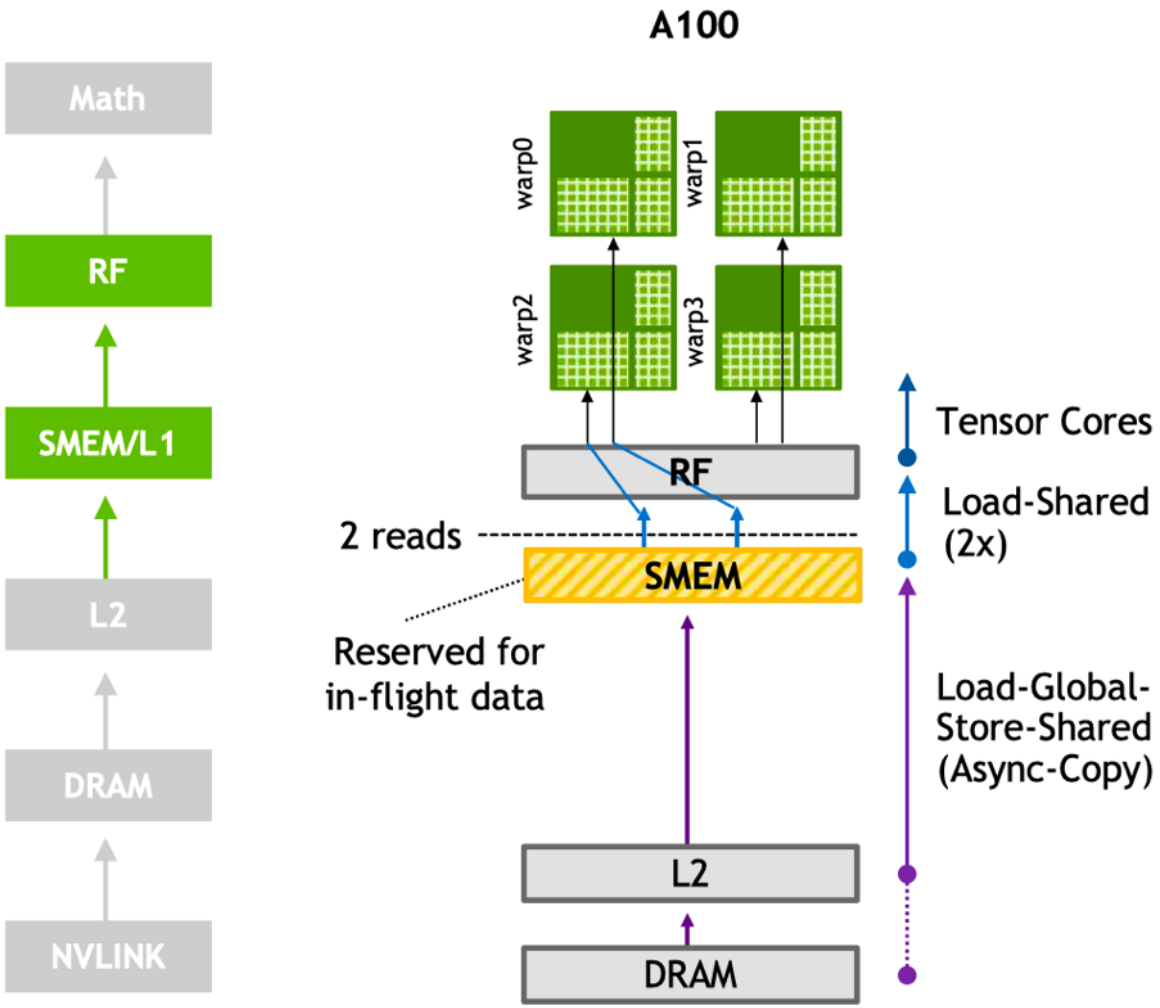
第三代 Tensor Core (Ampere)







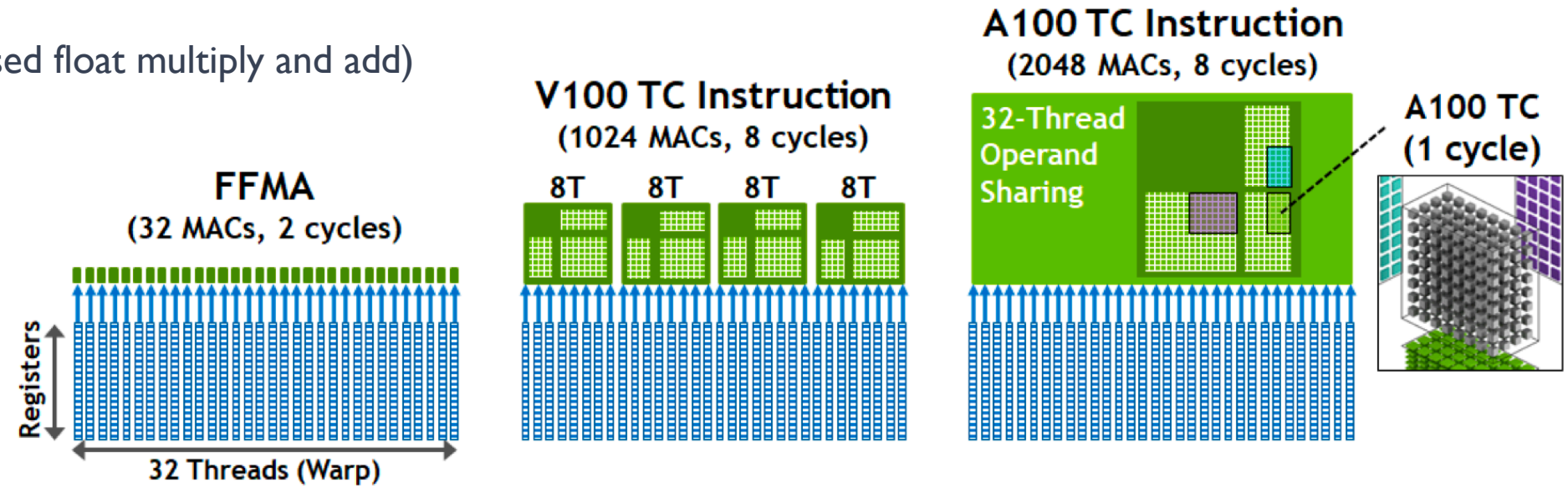
第三代 Tensor Core (Ampere)



- Ampere架构的 Tensor Core 一个 warp 中 32 个线程间共享数据，而 Volta 架构 Tensor Core 只有8个线程；
- 可以更好地在线程间减少矩阵的数据搬运。

第三代 Tensor Core (Ampere)

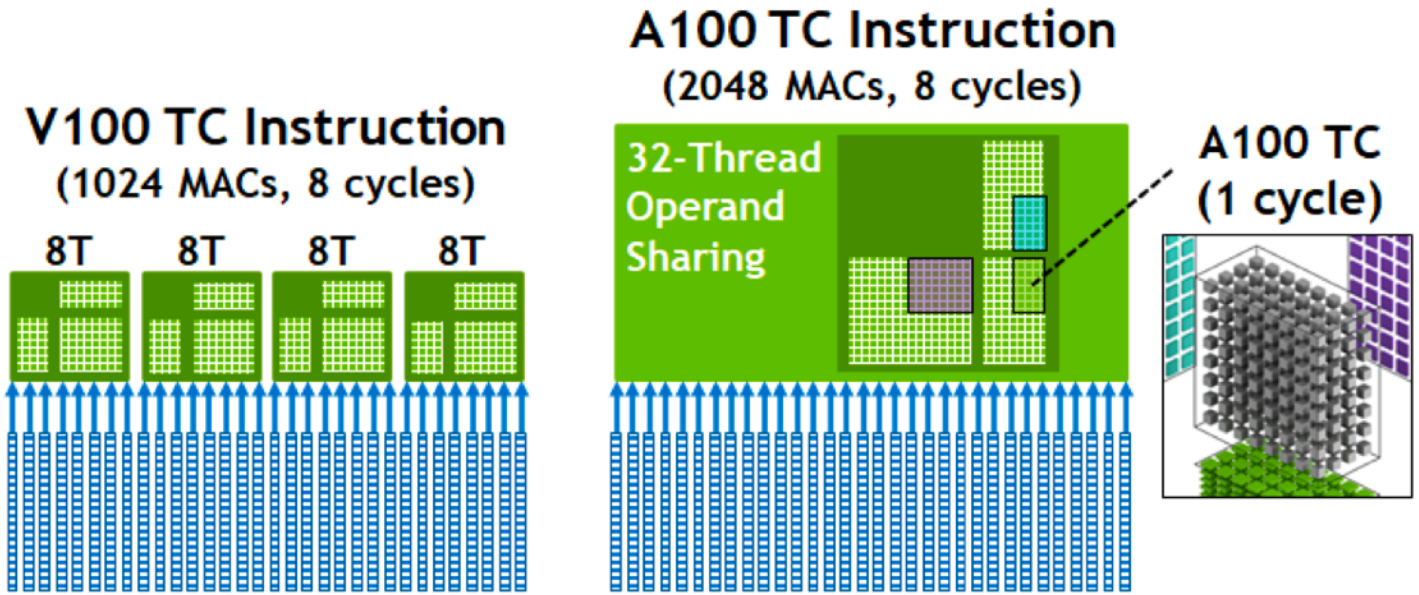
FFMA (fused float multiply and add)



16x16x16 matrix multiply	FFMA	V100 TC	A100 TC	A100 vs. V100 (improvement)	A100 vs. FFMA (improvement)
Thread sharing	1	8	32	4x	32x
Hardware instructions	128	16	2	8x	64x
Register reads+writes (warp)	512	80	28	2.9x	18x
Cycles	256	32	16	2x	16x

绿色小块为Sub core中浮点计算单元，连续蓝色框表示寄存器

第三代 Tensor Core (Ampere)



- V100 Tensor Core 可以聚合的操作八个线程的寄存器，A100则可以操作32个线程的所有寄存器。
- A100 的Tensor Core 增强了16x8x16 的计算指令，将寄存器次数访问从 80 减少到 28，硬件指令从16 减少到 2。

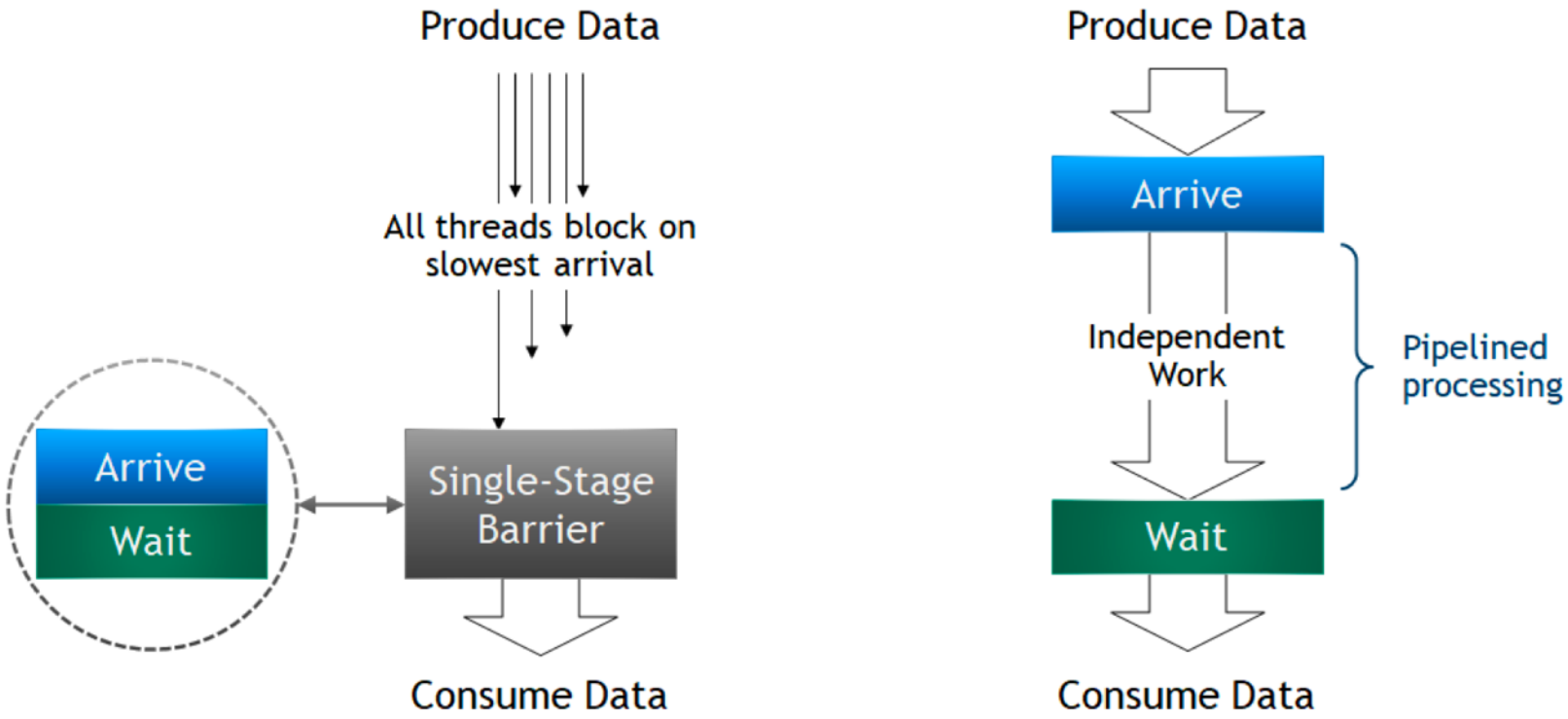
第四代 Tensor Core (Hopper)

- 前三代 Tensor Core 基于 warp Level 进行编程：通过 SIMT 完成矩阵计算，将数据从全局内存加载到寄存器上，再通过 Warp Scheduler 调用 Tensor Core 完成矩阵乘法，最后将结果写出到寄存器。

Pros：

1. **数据搬运和计算耦合：**Tensor Core 准备数据时，warp 内线程分别加载矩阵数据 Data Tile，每一个线程都会获取独立矩阵块地址；为了隐藏数据加载的延时（全局内存到共享内存，共享内存到寄存器的数据加载），会构建多层级软流水（software pipeline），消耗寄存器数量及存储带宽；
2. **可扩展性受约束：**由于多级缓存Cache的存储空间限制，单个 warp 的矩阵计算规格有上限。

第四代 Tensor Core (Hopper)



Single-Stage barriers combine back-to-back arrive & wait

Asynchronous barriers enable pipelined processing

Hopper 赫柏架构


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32 INT32 FP32 FP32 FP64
TENSOR CORE
LD/ ST
SFU
192KB L1 Data Cache / Shared Memory
Tex


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32 FP32 FP32 FP64
TENSOR CORE
4th GENERATION
LD/ ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

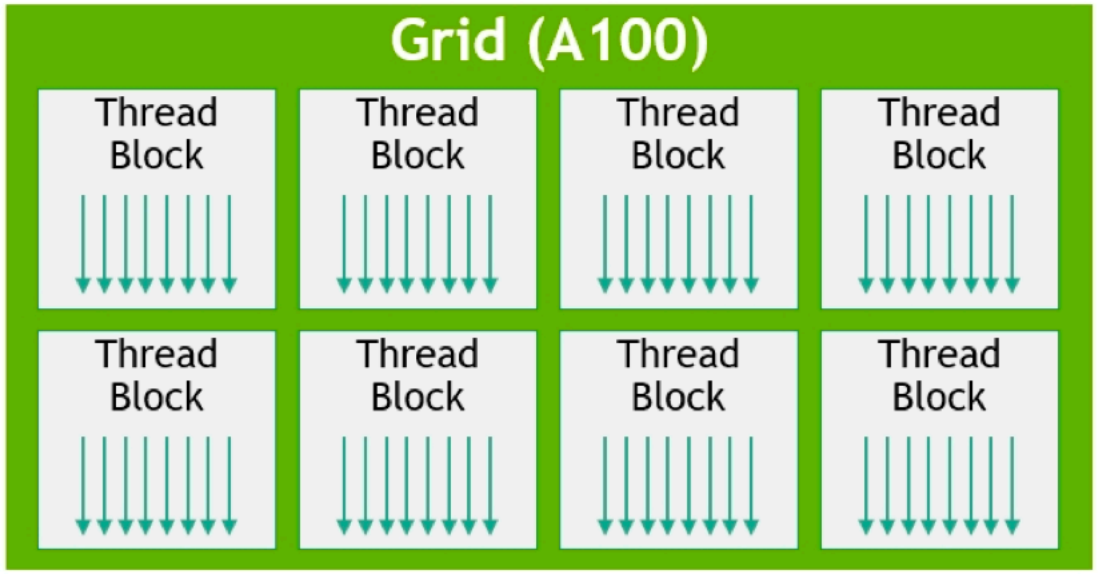
Hopper 赫柏架构

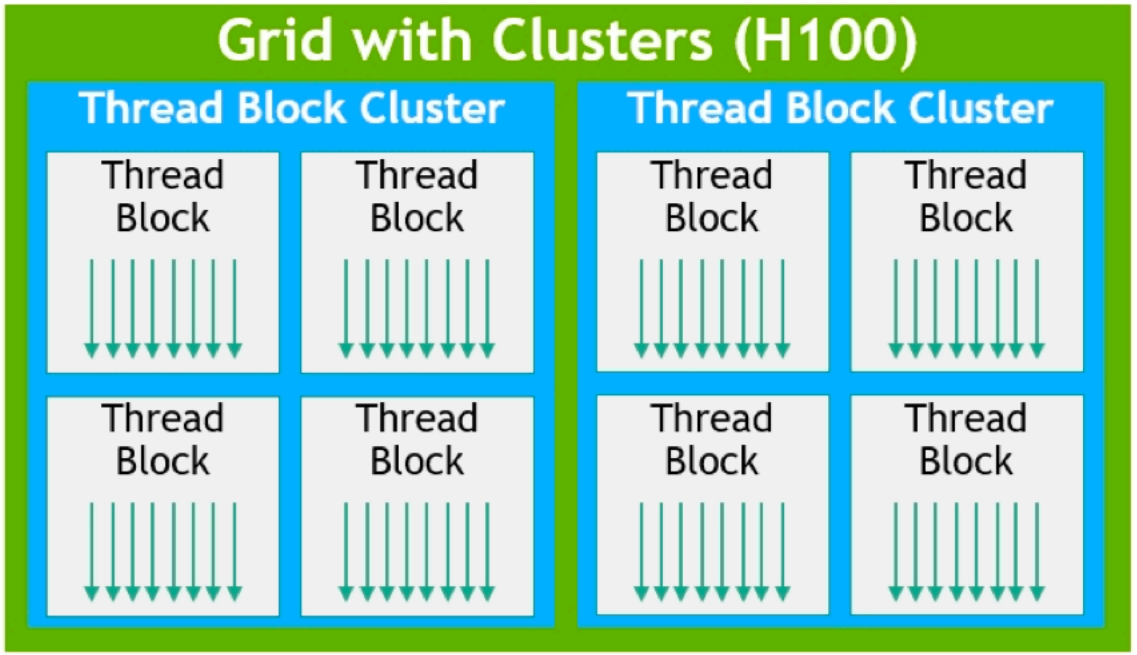
- **TMA（Tensor Memory Accelerator）：**
  硬件异步数据加载，即全局内存中的数据可以被异步地加载到共享内存。

- 采用单线程schedule模型，不需要所有线程都参与，同时使用 TMA 之前只需要一次性配置好首地址、偏移量、等Tensor描述信息即可。

第四代 Tensor Core (Hopper)

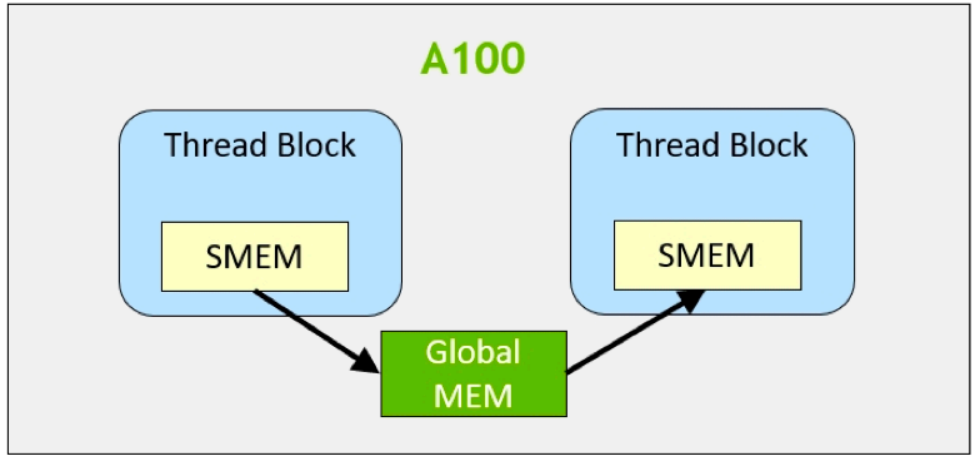
- 只有block和grid，分别对应硬件 SM 和 Device，局部数据只能通过 shared memory 限制在SM内部，不能跨SM。
- hopper架构在 GPC 内部通过引入交叉互连网络以及同步元语将数据共享层次扩展到4个 SMs，GPC 内 SM 可以高效访问彼此的共享内存。

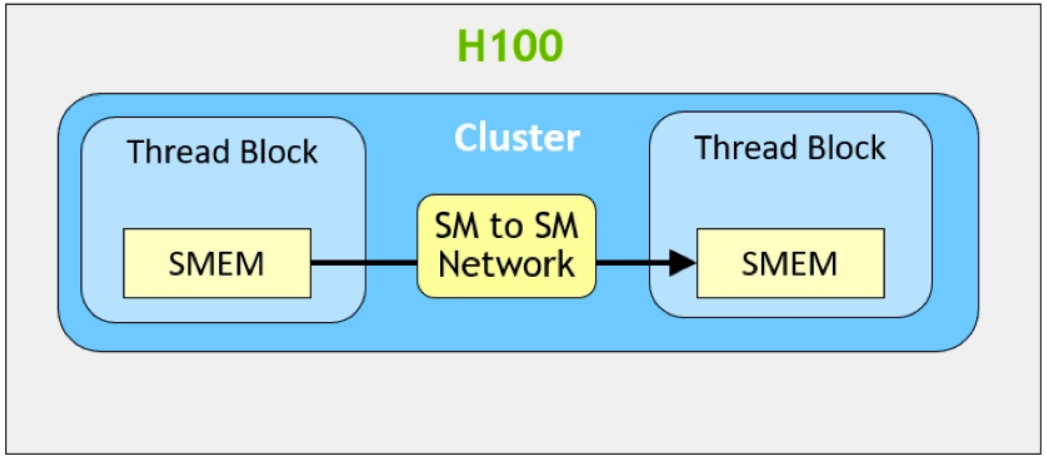




第四代 Tensor Core (Hopper)

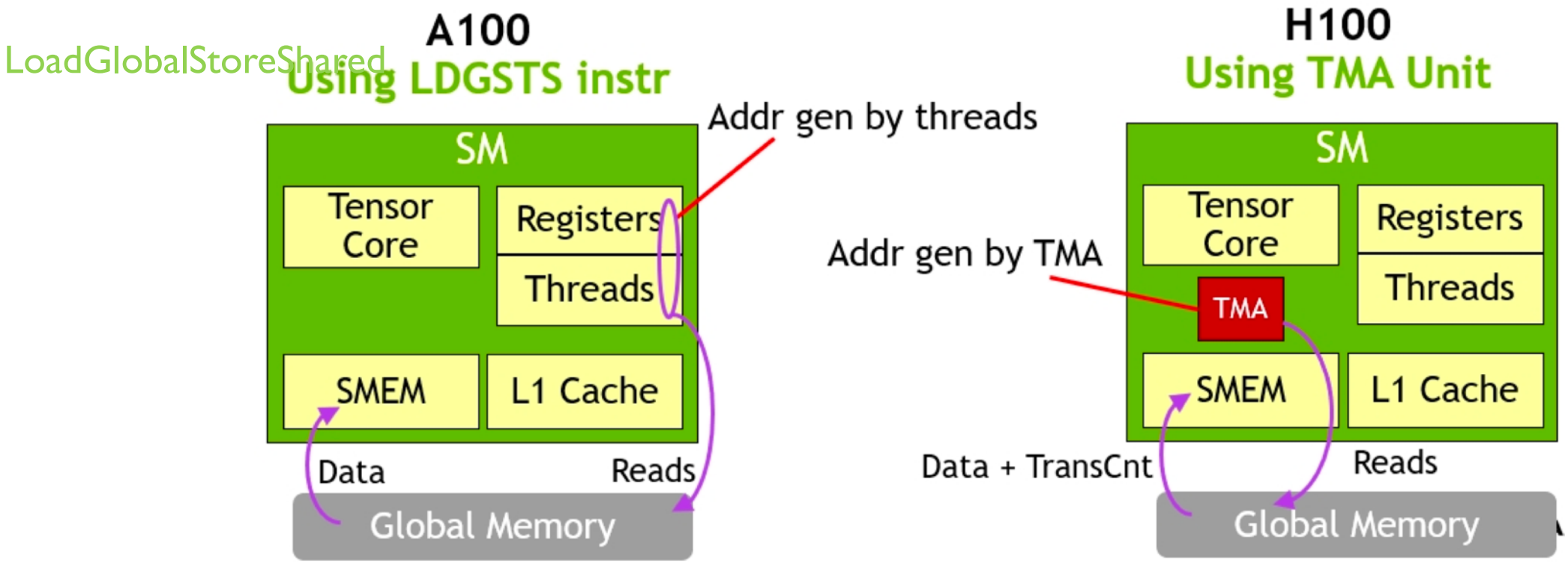
- **分布式共享内存和warp group编程模式**：这种结构很好的提升了数据的复用性，为了能编程使用该结构，CUDA 中也对应的引入了 Thread Block Cluster 的概念。





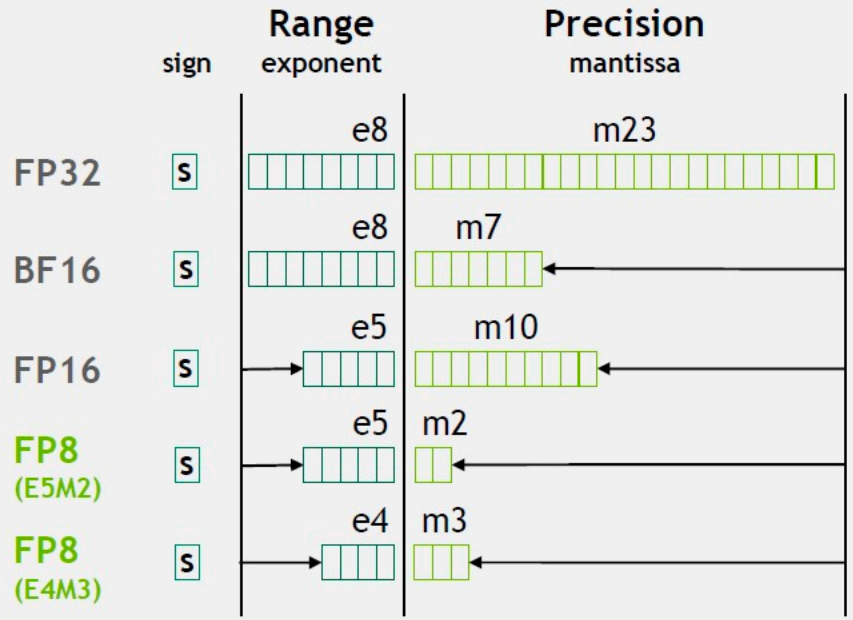
第四代 Tensor Core (Hopper)

- **直接读取共享内存异步Tensor Core**：TMA 将 SM 组织成一个更大的计算和存储单元，完成数据从 global 到 shared memory 的异步加载，数据到寄存器的计算，最后通过硬件实现了矩阵乘法的流水线。



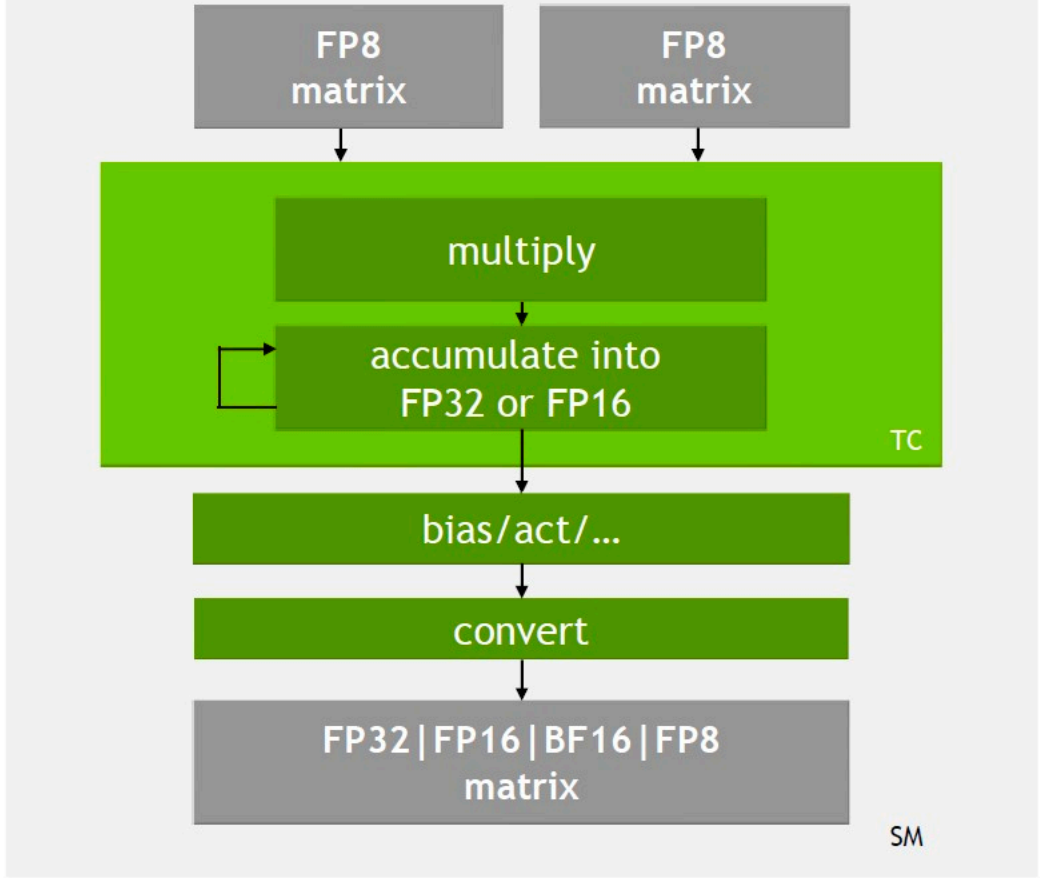
第四代 Tensor Core (Hopper)

Allocate 1 bit to either range or precision



2x throughput & half footprint of FP16/BF16

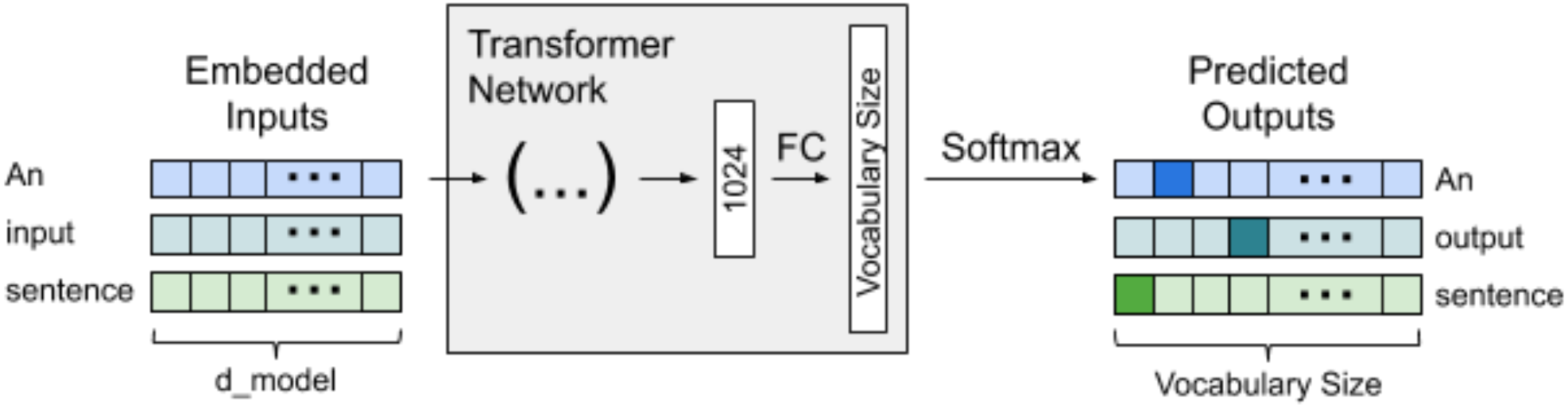
Support for multiple accumulator and output types



Tensor Core 的应用

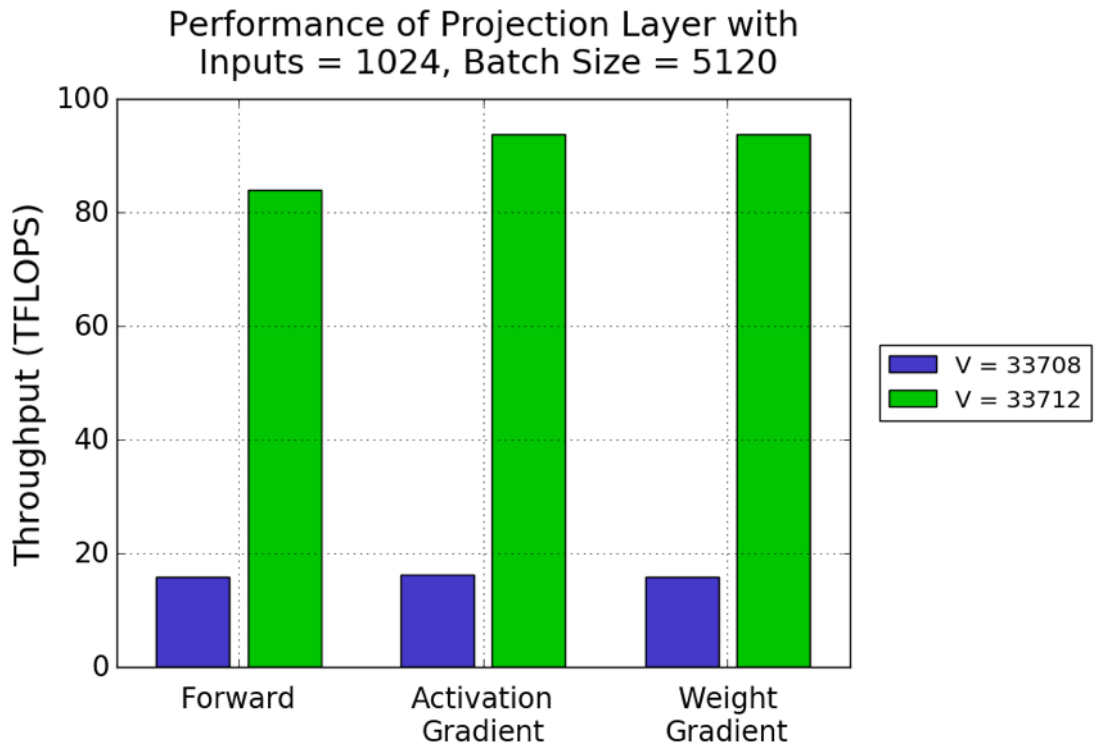
Padding Vocabulary Size – Projection Layer Example

- 在 Transformer 架构中，通过 softmax 层对 FC 输出的特征向量进行计算，得到包含词汇表中每个 Token 概率的向量。Softmax 层的输出数量等于词汇表大小，通常超过 30000。



Padding Vocabulary Size – Projection Layer Example

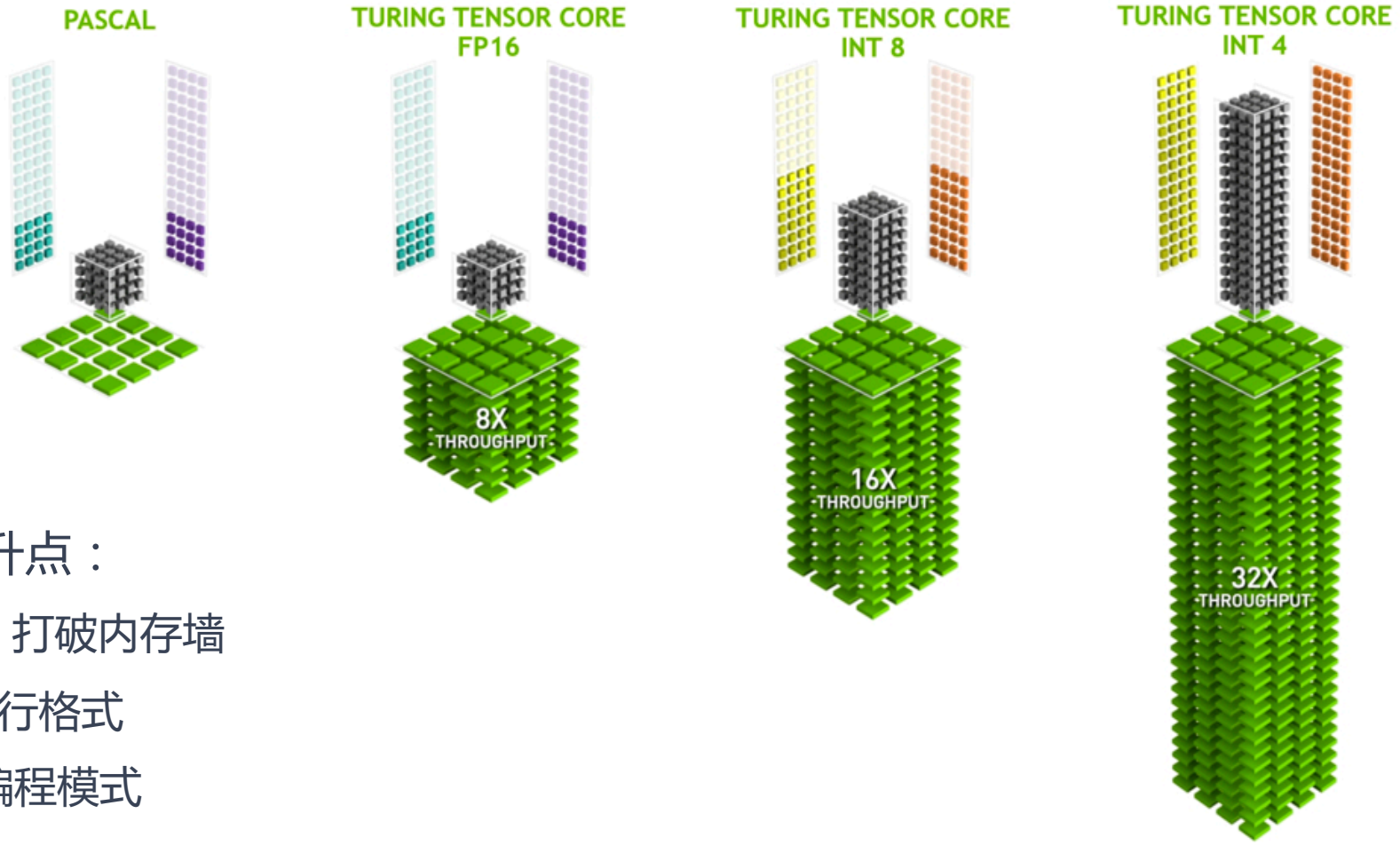
- Input Size = 1024， Batch Size = 5120，在Volta Tesla V100 使用 FP16 数据进行训练。假设使用英语-德语训练数据集进行 WMT14 任务，词汇大小为 33708，将词汇表大小填充到下一个 8 的倍数，即可激活张量核心并显著提高吞吐量。



33708/8=4213.5

33712/8=4214

Summary 总结



- 历代 Tensor Core 主要提升点：
  1. **Memory Improvements** 打破内存墙
  2. **SM Changes** 提供更多执行格式
  3. **Program Model** CUDA编程模式

Reference 引用&参考

1. https://zhuanlan.zhihu.com/p/620257581
2. https://developer.nvidia.com/blog/programming-tensor-cores-cuda-9/
3. https://developer.nvidia.com/blog/accelerating-winml-and-nvidia-tensor-cores/
4. https://developer.nvidia.com/blog/optimizing-gpu-performance-tensor-cores/
5. https://www.anandtech.com/Gallery/Album/6494#15
6. https://www.anandtech.com/Gallery/Album/6493#33
7. https://www.anandtech.com/show/12673/titan-v-deep-learning-deep-dive/3
8. https://www.anandtech.com/show/12673/titan-v-deep-learning-deep-dive/4
9. https://www.cnblogs.com/wujianming-110117/p/12993096.html
10. https://www.cnblogs.com/wujianming-110117/p/12993096.html
11. https://aijishu.com/a/1060000000286803#item-2
12. https://learnopencv.com/demystifying-gpu-architectures-for-deep-learning-part-2/