Tensor Core技术解析（上） - 吴建明wujianming - 博客园

https://www.cnblogs.com/wujianming-110117/p/12992932.html

None Mon Jul, 05 01:23

Tensor Core技术解析（上）

NVIDIA在SIGGRAPH 2018上正式发布了新一代GPU架构——Turing（图灵），黄仁勋称Turing架构是自2006年CUDA GPU发明以来最大的飞跃。Turing架构的两大重要特性便是集成了用于光线追踪的RT Core以及用于AI计算的Tensor Core，使其成为了全球首款支持实时光线追踪的GPU。

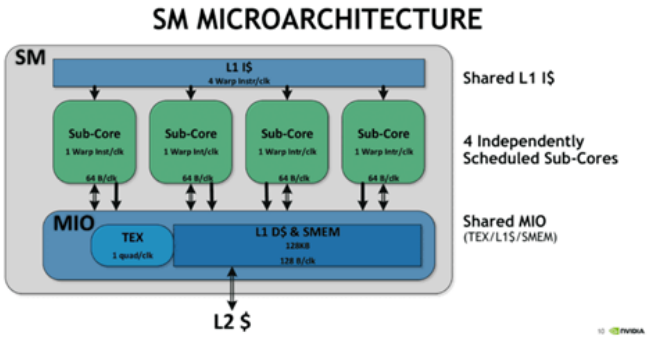
不过说到AI计算，NVIDIA GPU成为最好的加速器早已是公认的事实，但将Tensor Core印上GPU名片的并不是这次的Turing，而是他的上任前辈——Volta。

在关于Volta混合精度Tensor Core的几个谜团中，一个比较烦人的问题是4×4矩阵乘法的能力。Tensor Core是一种新型处理核心，它执行一种专门的矩阵数学运算，适用于深度学习和某些类型的HPC。Tensor Core执行融合乘法加法，其中两个4*4 FP16矩阵相乘，然后将结果添加到4*4 FP16或FP32矩阵中，最终输出新的4*4 FP16或FP32矩阵。

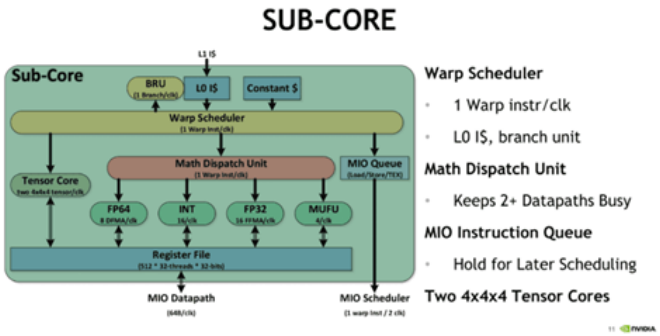
NVIDIA将Tensor Core进行的这种运算称为混合精度数学，因为输入矩阵的精度为半精度，但乘积可以达到完全精度。碰巧的是，Tensor Core所做的这种运算在深度学习训练和推理中很常见。

Tensor Core虽然在GPU里是全新的运算单元，但其实它与标准的ALU流水线并没有太大差别，只不过Tensor Core处理的是大型矩阵运算，而不是简单地单指令流多数据流标量运算。Tensor Core是灵活性和吞吐量权衡的选择，它在执行标量运算时的表现很糟糕，但它可以将更多的操作打包到同一个芯片区域。

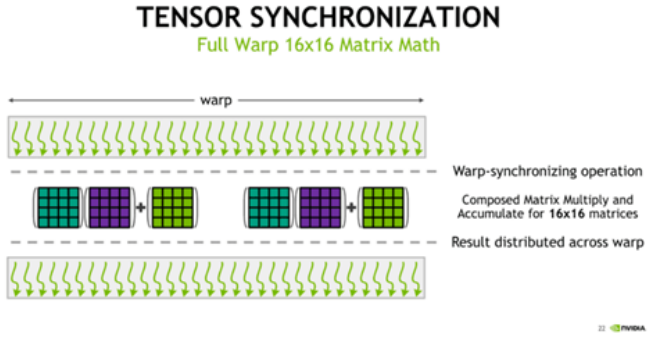
Tensor Core虽然有一定的可编程性，但仍然停留在4*4矩阵乘法累加层面上，并且不清楚累积步骤是如何以及何时发生的。尽管被描述为进行4*4矩阵数学运算，但实际上Tensor Core运算似乎总是使用16*16矩阵，并且操作一次跨两个Tensor Core进行处理。这似乎与Volta架构中的其他变化有关，更具体地说，与这些Tensor Core是如何集成进SM中有关。



对于Volta架构，SM被划分为四个处理块或子核。对于每个子核，调度器每个时钟向本地分支单元（BRU）、Tensor Core阵列、数学分派单元或共享MIO单元发出一个warp指令，这就首先阻止了Tensor运算和其他数学运算同时进行。在利用两个Tensor Core时，warp调度器直接发出矩阵乘法运算，并且在从寄存器接收输入矩阵之后，执行4*4*4矩阵乘法。待完成矩阵乘法后，Tensor Core再将得到的矩阵写回寄存器。

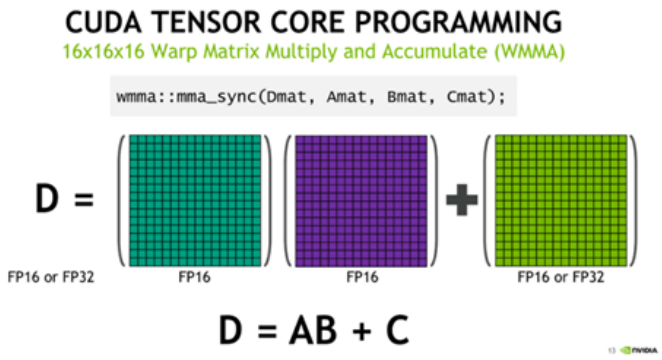


在Tensor Core执行实际指令时，即使在使用NVVM IR（LLVM）的编译器级别上，也仅存在用于warp级矩阵操作的本征，对于CUDA++和PTX ISA，warp级别仍然是唯一级别。加载输入矩阵的形式是每个扭曲线程持有一个片段，其分布和身份均未指定。从广义上讲，它遵循标准CUDA核心的基于线程级别拼接的GEMM计算的相同模式。



一般而言，给定A*B+C Tensor Core操作，片段由A的8个FP16*2元素（即16个FP16元素）和B的另外8个FP16*2元素，以及FP16累加器的4个FP16*2元素或FP32累加器的8个FP32元素组成。

在矩阵乘法累加运算之后，计算结果会分散在每个线程的目标寄存器片段中，需要在整个范围内统一，如果其中一个warp线程退出，这些低级操作基本上就会失败。



Citadel LLC团队的低级微基准测试揭示了许多Volta微体系结构细节，包括Tensor Core操作和相关的片段，与输入矩阵相比，它们都位于寄存器和标识中。他们观察到，子核核心以特定的拼接模式计算矩阵乘法，其中所有32个warp线程都在运行。

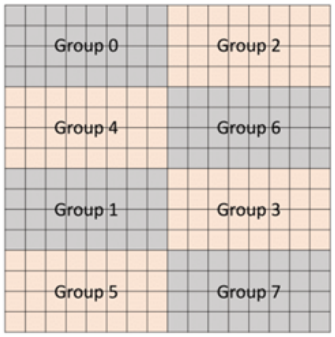


Figure 4.5: Mapping between positions in matrix *C* and thread group indices.

从概念上讲，Tensor Core在4*4子矩阵上运行，以计算更大的16*16矩阵。warp线程被分成8组，每组4个线程，每个线程组连续计算一个8*4块，总共要经过4组的过程，每一个线程组都处理了目标矩阵的1/8。

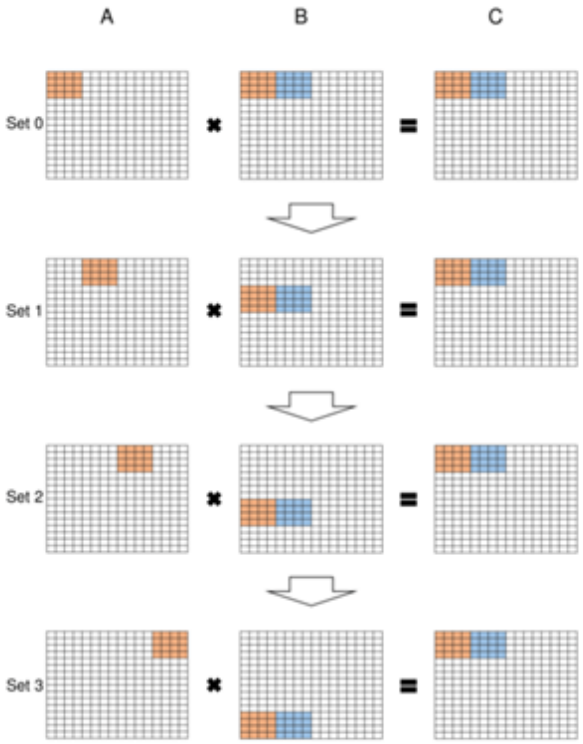


Figure 4.6: Four sets of HMMA instructions complete 4×8 results in matrix C within thread group 0. Different sets use different elements in A and B. The instructions in set 0 execute first, then the instructions in set 1, set 2 and set 3. This way, the 16 HMMA instructions can correctly compute the 4×8 elements in matrix C.

在一个集合中，可以并行完成四个HMMA步骤，每个步骤适用于4*2子块。这四个线程直接链接到寄存器中的那些矩阵值，因此线程组可以处理单个Step 0 HMMA指令，从而一次性计算子块。

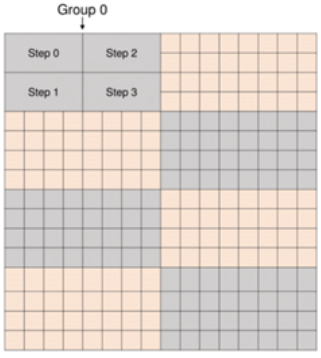
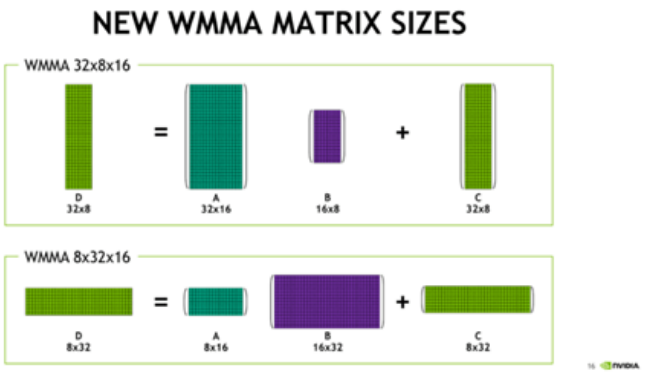


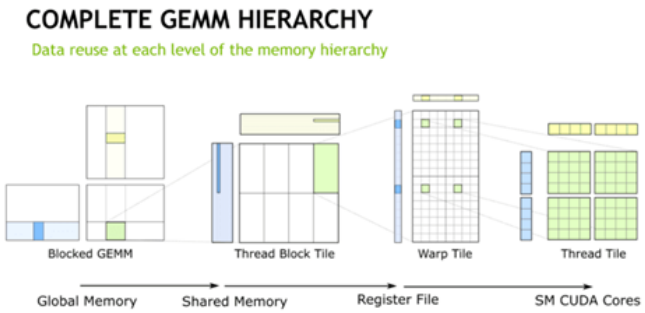
Figure 4.4: 4 steps of HMMA instructions within one set compute different elements in matrix C.

由于矩阵乘法在数学上需要对某些行列进行复用，以允许跨所有8*4块并行执行，每个4*4矩阵被映射到两个线程的寄存器。在计算16*16父矩阵的4*4次子矩阵运算中，这将包括将连续计算的集合相加，形成16*16矩阵中4*8个元素的相应块。尽管Citadel没有对FP16进行测试，但它们发现FP16 HMMA指令只产生2个步骤，而不是4个步骤，这或许与FP16只占用的较小的寄存器空间有关。



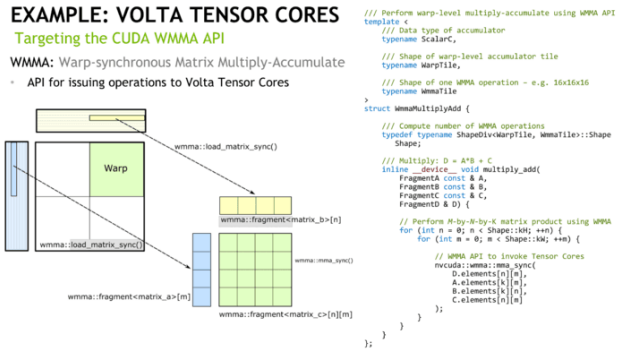
通过独立的线程调度和执行，以及warp同步和warp-wide结果分配，基本的4*4*4 Tensor Core操作转换为半可编程16*16*16混合精度矩阵乘法累加。虽然CUDA 9.1支持32*8*16 and 8*32*16矩阵，但相乘的矩阵都需要相应的列和行为16，最终矩阵为32*8或8*32。

Tensor Core的运行方式似乎是NVIDIA GEMM计算层次结构的一个硬件实现的步骤，如CUTLASS（用于GEMM操作的CUDA C ++模板库）中所示。对于传统的CUDA核心，最后一步需要将warp tile结构分解为由各个线程拥有的标量和向量元素。使用WMMA API（现在表示张量核），所有这些都被抽象掉了，只剩下了需要处理的合作矩阵片段加载/存储和多重积累。积累发生在一个FMA类型的操作中。



在寄存器级别上，NVIDIA在他们的Hot Chips 2017论文中提到“使用三个相对较小的乘法和累加器数据的4*4矩阵，可以执行64次乘加运算。”而增强的Volta SIMT模型的每线程程序计数器（能够支持张量核）通常需要每个线程2个寄存器槽。HMMA指令本身会尽可能多复用寄存器，所以无法想象寄存器在大多数情况下不会出现瓶颈。

对于独立的4*4矩阵乘法累加，Tensor Core阵列在寄存器、数据路径和调度方面很有核能并没有物理设计，它只能用于特定的子矩阵乘法。



无论如何，从NVIDIA的角度来看，Volta不是一颗深度学习的专用ASIC，它仍然覆盖GPGPU的领域，因此保持CUDA可编程Tensor Core适用于GEMM / cuBLAS和HPC是合乎逻辑的。对于CUDA c++的CUTLASS来说，情况更是如此，因为它的WMMA API支持旨在为广泛的应用程序启用Tensor CoreGEMM操作。从根本上说，NVIDIA深度学习硬件加速的发展与cuDNN（以及cuBLAS）的发展有很大关系。