How GPU Computing works

1、Compute Intensity

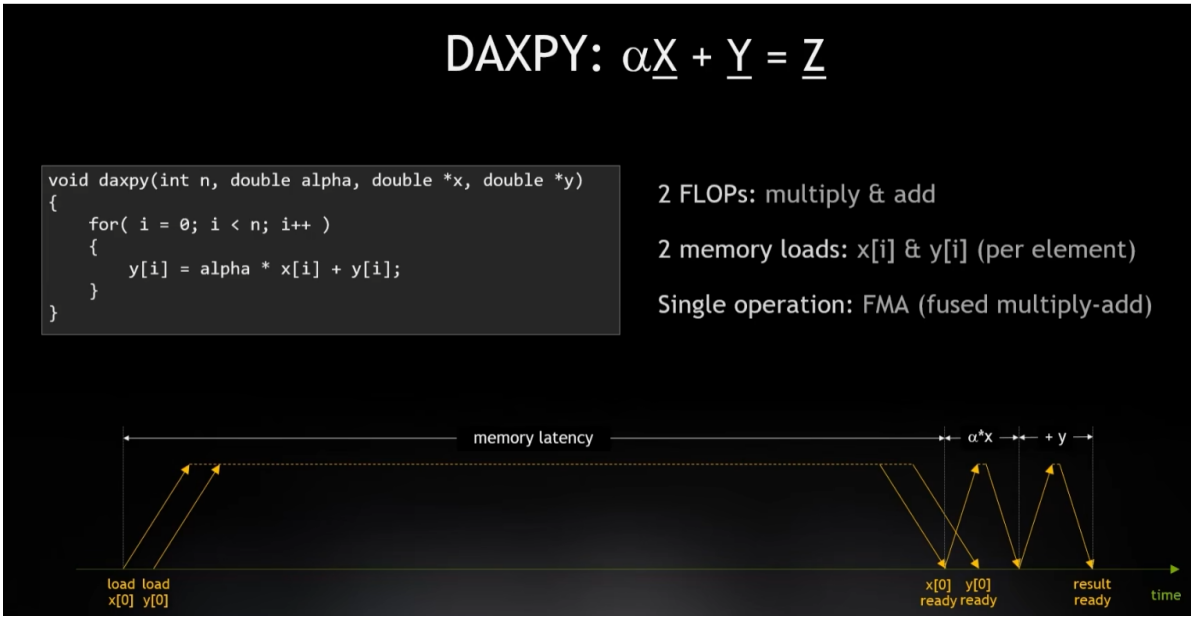
计算强度，我们知道内存提供数据的速率与CPU处理数据的速率是不同的，所以很多时计算的瓶颈并不是CPU而是高速缓存， 假设高速缓存的速率是 200Gbytes/sec， 对于双精度浮点数来说，相当于25Giga-FP64/sec， 此时CPU的处理数据速率是2000G-FP64/sec，那么此时的计算强度是200/25=80。 那么对于我来说，内存中load的一个数据，我CPU中执行80次计算，我才算很好的利用该CPU的资源。

	NVIDIA A100	Intel Xeon 8280	AMD Rome 7742
Peak FP64 GigaFLOPs	19500	2190	2300
Memory B/W (GB/sec)	1555	131	204
Compute Intensity	100	134	90

所以，几乎没有人需要真正关心CPU的FLOPS处理能力，因为内存的宽带限制 ，我们更应该关心的是LATENCY，延迟。

2、LATENCY

延迟是如何发生的?

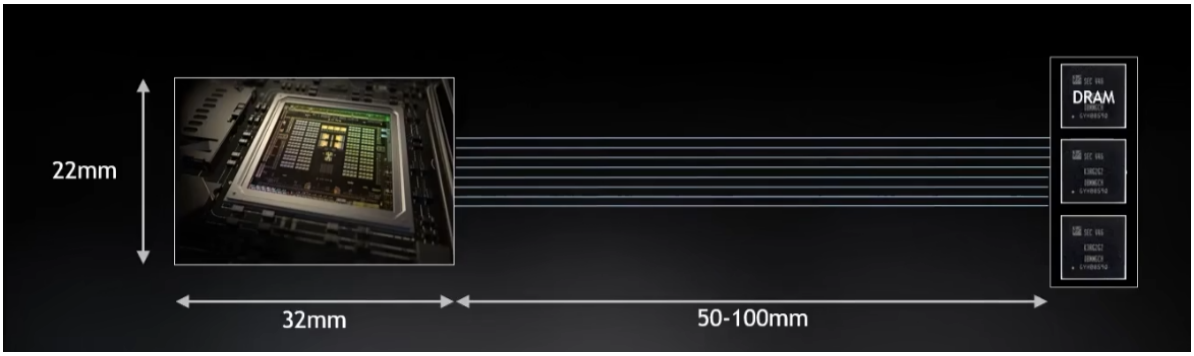


以当前的例子距离，以上的简单程序中需要用到两个能一个，一个是CPU的乘法和加法，一个是内存加载 x、y数组。

图片的下方是运算的pipeline，首先从内存加载数据，后面到CPU中进行计算。 虽然我们编程的过程中不关心这个pipeline，但是编译器是花了很大的经历来保证CPU指令pipeline的正确性，以数据加载的正确性。

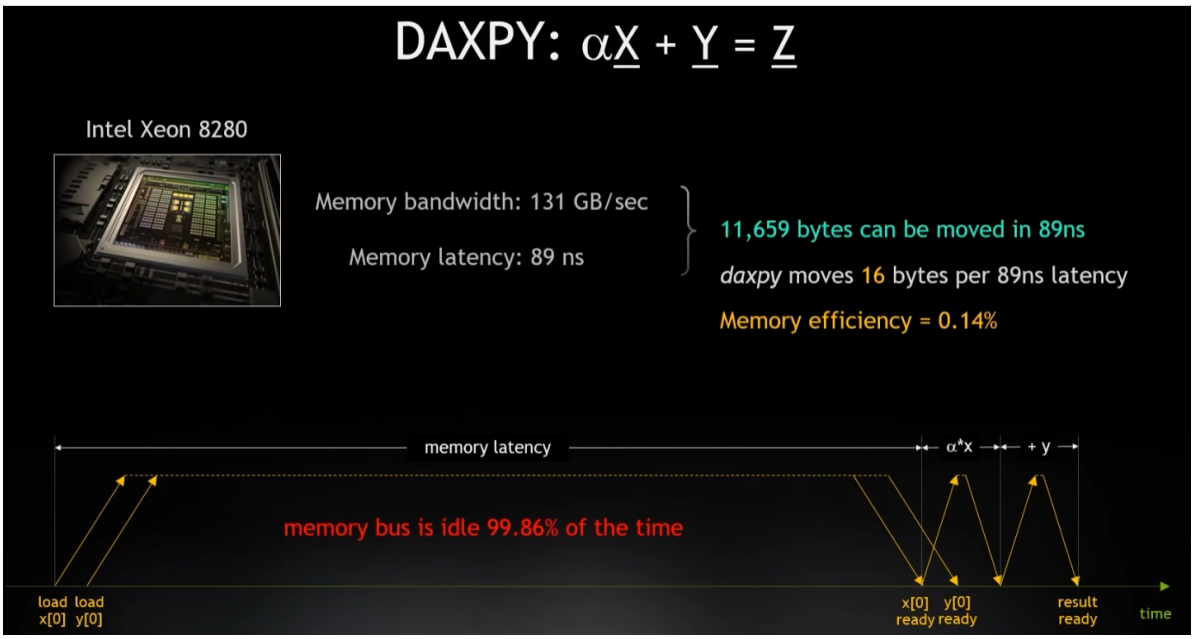
光的传播速度是非常快的 3 * 10^8m/s, 而CPU的频率也很高，能到到 3 * 10^9 HZ, 那么在一次时钟周期内，光的传播距离只有10cm。时钟周期是非常快的，一个时钟周期内光的距离也很短。 而电流在硅中的传播速度只有光速的五分之一，即6 * 10^7m/s，那么一个时钟周期上，电流只能移动2cm。

那么我们看关注下芯片的设计尺寸，假设一个时钟周期让电流从一端达到另一端，以直线上最快的速度移动，处理器就有5-7个时钟周期的延迟(芯片设计的距离大概10-14cm)， 所以从这里人知道，电的速度是正在和CPU的计算速度赛跑的。因为物理上的原因，当我们从内存中提取数据时，内存需要5-10个时钟周期才能返回。



而且，不完全是我们离内存的距离，这个距离并不是延迟影响最大的因素。

电路的工作方式是把信号从一组晶体管传递到另一组晶体管。晶体管的流水线的深度才是最大的因素。



以Intel Xeon 8280举例， 上面是他的内存宽带与内存延迟，该程序中，每个时钟周期内只加载了x、y的某个值，一共16个字节，内存使用效率为0.14%, 内存总线99.86%的时间是空闲的。

lantency bound: 内存限制类别一个自己，它主要发生在不需要立即从内存中检索太多的数据，但是在内存层次结构的上层，必须等待很长的事件才能将数据发送到处理器的情况。

为了充分利用内存，按照全面的数据，内存宽带为131 / 1 * 10^9 bytes/ns=11659 bytes/ns，上面的程序一次加载数据为16bytes， 所以对于同一组数据 我们需要 迭代 11659/16 = 729才能让内存保持满负荷运转。

我们有一些简单的方式可以提高内存利用率，比如一个计算多个值，批量从内存中获取数据，即多线程和并发。

3、GPU与CPU的设计差异

COMPARISON OF DAXPY* EFFICIENCY ON DIFFERENT CHIPS

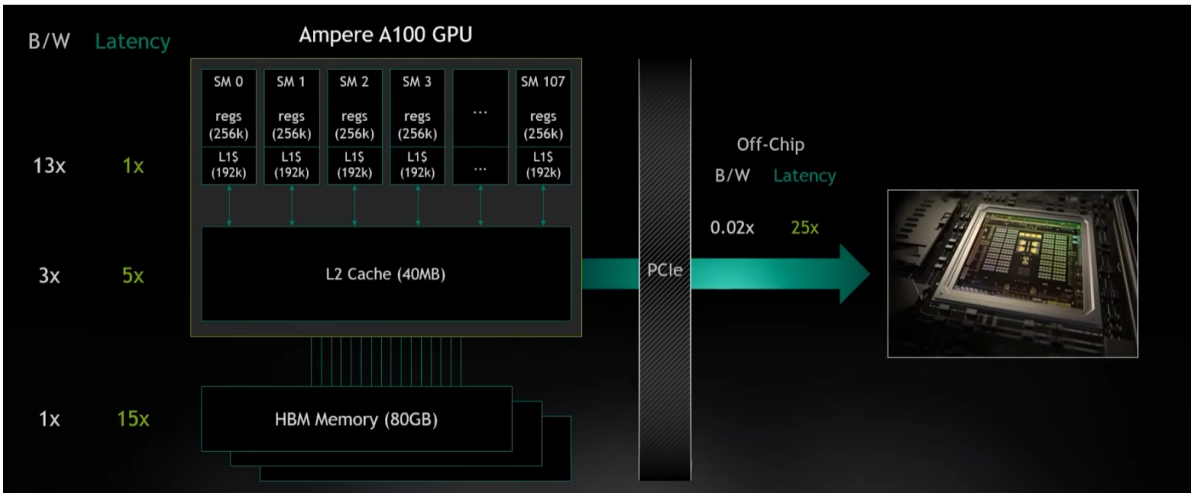
	NVIDIA A100	AMD Rome 7742	Intel Xeon 8280
Memory B/W (GB/sec)	1555	204	143
DRAM Latency (ns)	404	122	89
Peak bytes per latency	628,220	24,888	12,727
Memory efficiency	0.0025%	0.064%	0.13%
Threads required	39,264	1,556	729
Threads available	221,184	2048	896
Thread ratio	5.6x	1.3x	1.2x

GPU拥有更高的延迟和更多的线程， GPU的可用线程数量是其他处理器的100倍以上。从上面的图，GPU的线程比达到了5.6， 这也是GPU最重要的设计要点，他拥有比你需要的多得多的线程，是为大量的任务而设计的。 GPU的设计将所有的资源投入更多的线程中，而不是减少延迟。

CPU的期望是一个线程基本完成所有的工作，因为CPU时间片调度，切换上下文成本是很高的，所以CPU的设计者把所有的资源投入到减少延迟上，而不是增加线程。 这也是CPU与GPU 工作方式的根本区别。

4、GPU工作原理

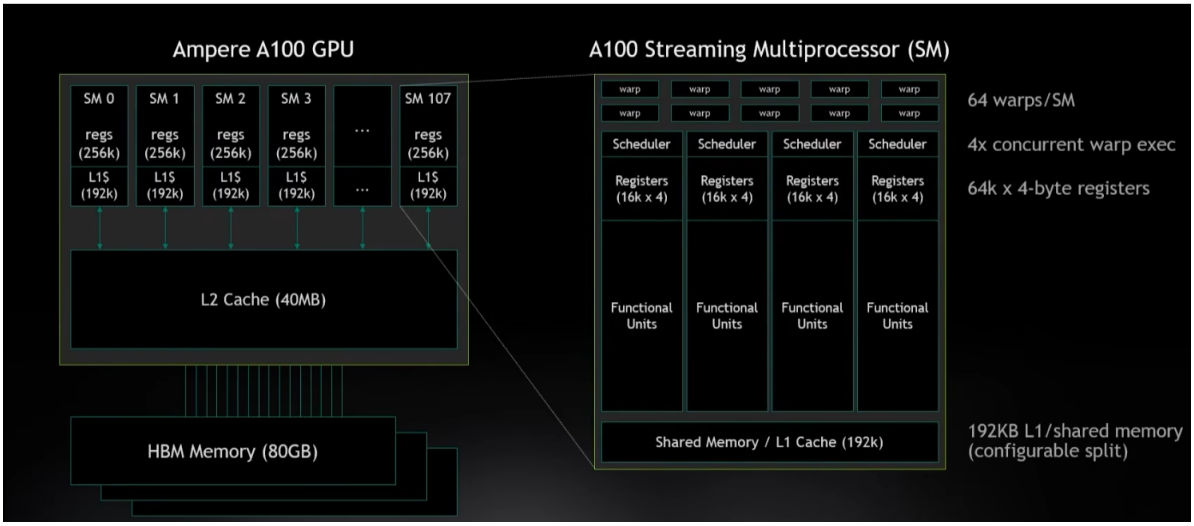
GPU使用大量的寄存器，不同类型的缓存延迟差距很大，寄存器是其中最优秀的，能够以很低的延迟来保存数据活动。



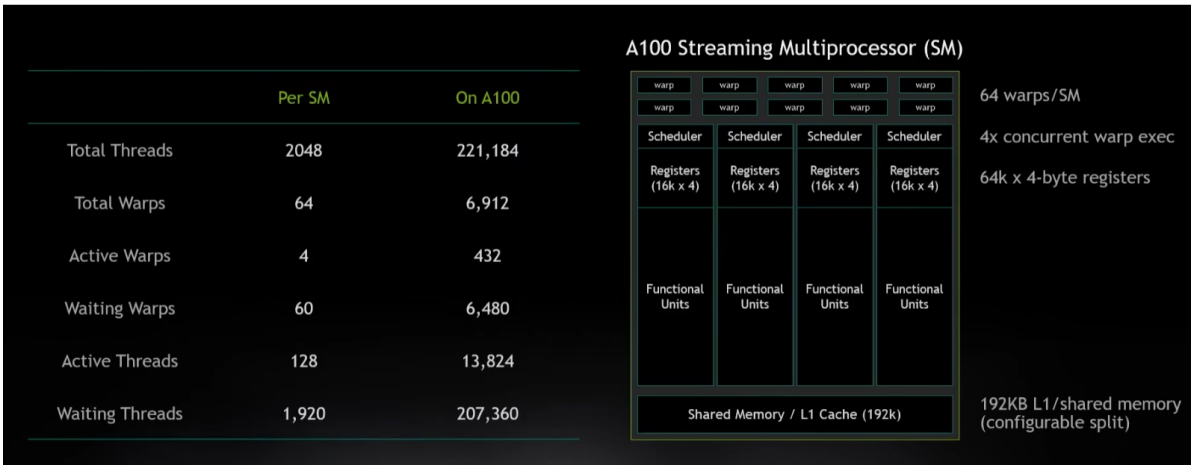
在Ampere中，寄存器的总量为27M，那么该GPU原则上可以维持27M的高速数据。GPU使用寄存器缓存数据， 即通过靠近数据来解决高延迟问题，这是与CPU不相同的。

Data Location	Bandwidth (GB/sec)	Compute Intensity
L1 Cache	19,400	8
L2 Cache	4,000	39
HBM	1,555	100
NVLink	300	520
PCIe	25	6240

可以看到L1缓存，计算强度是8，意味着同一个数据，我们只要能进行8次操作，那么就能保证GPU计算能力的饱和。8次操作这是很容易做的。



该GPU存在108个SM，每个SM中存在64个WARP，warp由32个线程组成一组，WRAP是GPU调度的基本单元。一个时钟周期内，我可以运行多个WRAP。该芯片一个SM中4个WARPS可以并行。

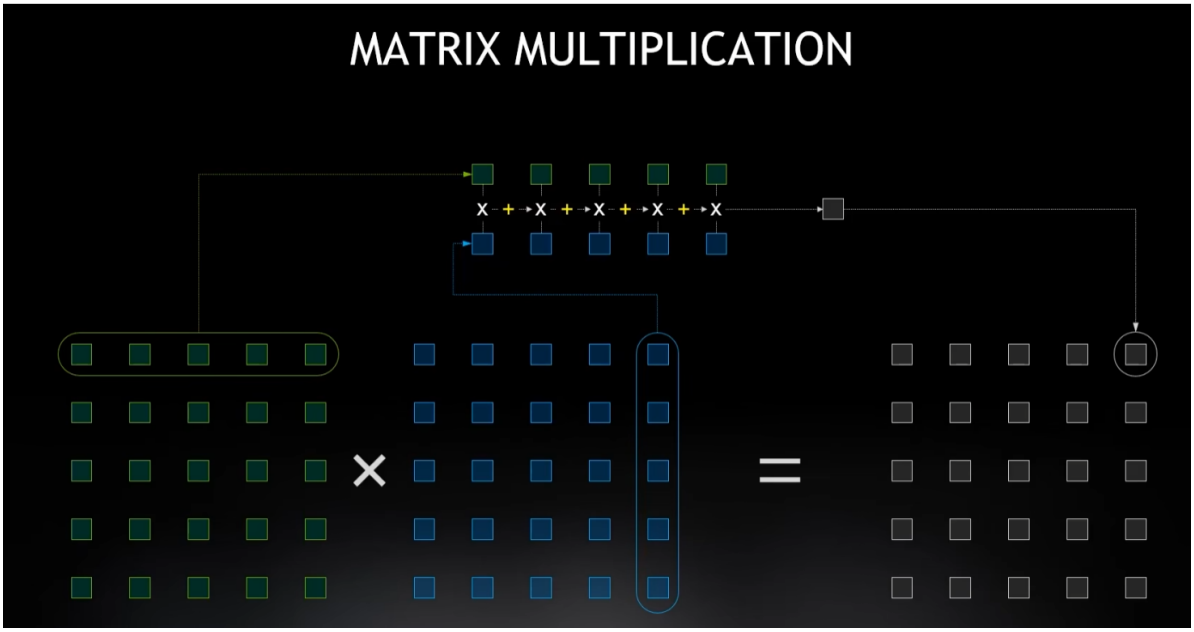


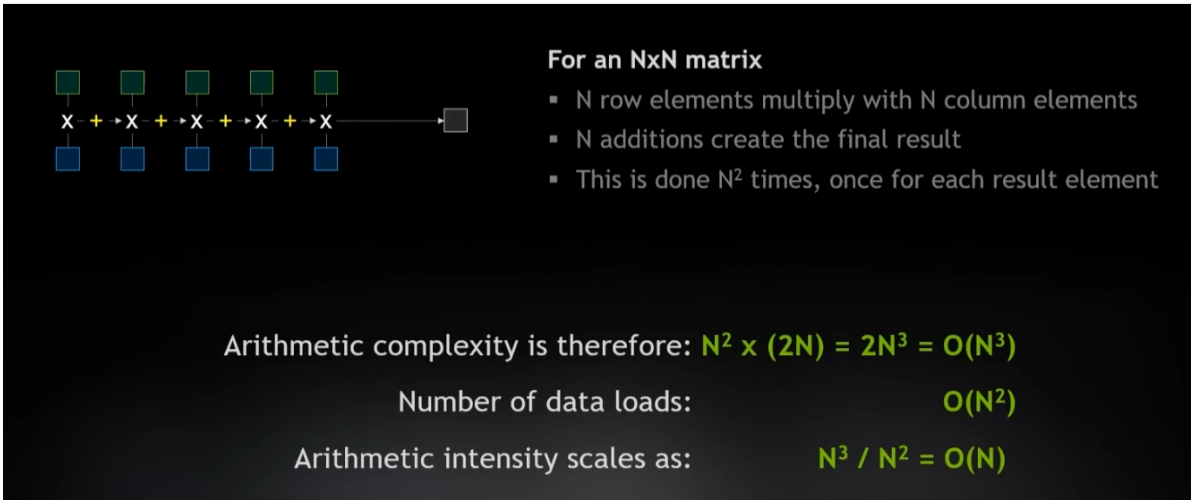
而且CPU在一个时钟周期内完成 WARP之前的切换，简单来说，就是基本不存在上下文切换开销(实际存在，但是在一个时钟周期内可以完成，并运行，所以我们根本不用关心其中的花销)。 一个系统在任何时候能够运行更多的线程是非常重要的，因为这是减少延迟最简单的方法。 这也是GPU与CPU的差异点。

GPU是一个吞吐机，而CPU是一个延迟机。CPU中切换线程开销很大，所以希望一个线程可以尽快的运行。当遇到堵塞是，一切都会停滞不前，他的目标是尽快完成所有的任务，然后未下一个任务让路。

回到之前提到的计算强度的问题，我们想合理的利用GPU或者CPU资源的话，都需要我们的算法能合理的去适配计算强度，GPU只是提供给我们了一个更优异的计算强度，换句话说，就是GPU提供给我们了一个相对CPU来说，实现一个合理利用计算强度的算法程序 更简单的平台。

深度学习中大量的矩阵运算就合理利用GPU的资源，在GPU强大吞吐之下相互加持。





以上是对于不同的缓存，Tensor Core的最佳计算强度。 所以对于小矩阵计算，我们需要提供更多的告诉缓存来降低延迟，而对于大矩阵运算，我们对内存宽带的要求反而没那么高。 对于计算 和 数据加载之间，两者要达到某种平衡，才能最大的发挥相关的效率。

5、总结

- 处理器的FLOPs浮点数计算能不并不重要，因为延迟、计算强度等相关因素。
- 内存宽带也不像延迟那么重要，因为延迟很长，为了修复延迟，我们需要更多的线程。
- GPU架构基于上面的因素简历起来，通过大量的线程和超量分配来解决延迟问题。GPU是需要超量分配的吞吐机，而不是具有固定工作量的延迟机。
- 尽管存在这么多独立线程，有时他们仍需要一起工作，并非一切都是Element-wise，所以GPU运行时，线程是一个层次结构，一大堆工作被分解成吞吐量的模式运行的block, block中的线程可以一起工作并合并一些操作。
- 在解决延迟问题后，要开始平衡计算和宽带，在小型、计算密集型的工作上获得更好效率的方法实际上就是需要更有层次的缓存。
- GPU可以用线程消除延迟，也可用局部性解决低宽带问题，以获取所有的FLOPS能力。

参考

GPU工作原理