NVIDIA中国特供GPU又被禁了，你还不知道GPU的原理和前景

Original Pulsar planet Tim在路上 2023-11-11 08:25 北京

Tim在路上
大模型、Spark、LakeHouse 欢迎关注知乎账号"Tim在路上"
132篇原创内容

公众号

大家好，我是Tim。

在去年的时候，美国就出台了对高性能计算芯片出口中国限制措施。

当时是以NVIDIA的A100芯片为标准进行限制的，直接导致的结果就是A100及H100 GPU国内没法买了。

NVIDIA立马针对中国大陆市场推出特供版的A800及H800芯片，主要是将NVLink的传输速率进行了限制。

A800和A100单机测试性能基本区别不大，可以说是完美绕过了美国的限制措施。

然而美国拜登政府于上个月，出台一系列新的限制措施，新增了”性能密度阈值”，直接封住了之前的缺口。

而且放出话来，未来可能“至少每年”更新一次，基本上从政策上堵住了“漏洞”。

目前A800和H800也都买不着了，而替代产品在性能上还是差点儿意思，美国这“老贼”确实影响了我国AI技术领域的迭代。

未来GPU国产适配一定是“在路上”。

那么问题来了，高性能GPU 被禁，普通GPU不能用吗？CPU和GPU差在哪里？

在解答这些问题之前，我们先来了解下目前哪些场景使用了GPU，以及为什么说大多数程序员都应该了解GPU。

GPU和CPU全方位协同的时代

说起GPU的应用起源，就不得不说打游戏。可以说在GPU发展的前期，游戏发展占半边天。PS5和Xbox游戏机都内置了GPU芯片。

如果你是一位游戏爱好者，你大概率会知道购买游戏设备，会更看重其GPU性能。即所谓的“CPU决定下限，GPU决定上限”。

GPU大体决定了游戏分辨率、特效能开多高，对于用户的游戏体验起到关键性作用。

其次就是前几年还比较流行的加密货币、“挖矿”。比特币等加密货币的火爆带动矿卡GPU需求，Nvidia的股票也是成倍的翻。矿机算力的大小决定挖矿的速度，算力越大，挖矿越快。

当时有句话叫，“上班是为了给我的矿机挣电费”。

其次是自动驾驶场景，自动驾驶场景一般在处理分析实时数据后，需要在毫秒的时间精度下对行车路径、车速进行规划，保障行车过程安全，对处理器的计算速度要求也较高。

而GPU采用流式并行计算模式，可对每个数据行独立的并行计算,擅长大规模并发计算，正是自动驾驶所需要的。

再其次就是深度学习，随着Transformer的出世，AI大模型其参数量从亿级飙升到万亿级，训练时间从小时到天级，使得大模型对算力提升提出要求，而这时高性能GPU集群正好可以满足这个要求。

当然，除此以外手机、平板等移动设备，音响，手环，监控等都已被嵌入并应用了GPU。

未来GPU将不仅仅停留在这些领域，云计算和大数据分析、机器人技术、虚拟现实和医疗目前也有很多开源的Demo已经在探索并应用。

过去的十多年，GPU依靠着上述的热点领域不断地拓展，而大多数的程序员依然是在CPU和顺序编程中成长。

未来的十多年，每一个程序员都需要像了解CPU一样了解GPU工作原理，以及并发编程的工作方式。

未来会是GPU和CPU全方位协同的时代。

GPU与CPU的比较

首先，我们借用一个经典的比喻。

GPU犹如一群小学生，CPU是一个大学教授，相比能力来说CPU完胜GPU，但对于大量浮点计算来说，还是GPU算的快。

我们可以说CPU是全能选手，GPU是专精选手。GPU一般无法独立工作，其工作都是由CPU触发的。

其次，对于设计目标来说。

CPU 被设计为顺序执行指令，为了提高顺序执行性能，多年来 CPU 设计中引入了许多功能，包括指令流水线、乱序执行、推测执行和多级缓存等。

CPU的重点是减少指令执行延迟，以便 CPU 能够尽快执行指令序列。

GPU 专为大规模并行性和高吞吐量而设计，但代价是较高的指令延迟。

这一设计方向受到了它们在视频游戏、图形、数值计算和现在深度学习中的使用的影响。

所有这些应用程序都需要以非常快的速度执行大量线性代数和数值计算，因此人们对提高这些设备的吞吐量投入了大量注意力。

让我们举一个具体的例子。

由于CPU指令延迟较低，CPU 可以比 GPU 更快地执行两个数字相加。

他们将能够以比 GPU 更快的速度连续执行多项类似的计算。

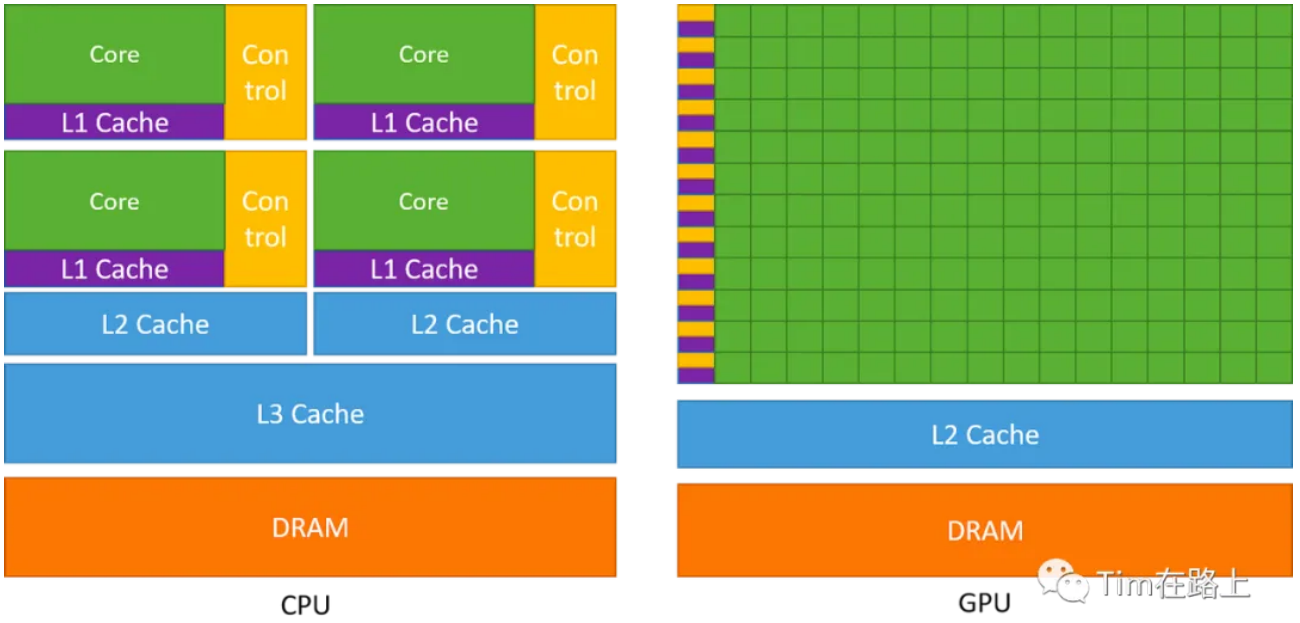
然而，当进行数百万或数十亿次此类计算时，GPU 由于其巨大的并行性而比 CPU 更快地完成这些计算。

如果以具体的数值进行衡量。

Nvidia A100 在 32 位精度下提供 19.5 TFLOPS 的吞吐量。（每秒可以执行多少次浮点运算）

英特尔 24 核处理器的 32 位精度吞吐量为 0.66 TFLOPS。而且，GPU 和 CPU 之间的吞吐量性能差距逐年扩大。

最后，我们对比下CPU和GPU的架构。



上图左边为CPU架构，右边为GPU架构。

CPU 将大量芯片面积专门用于可减少指令延迟的功能，例如大缓存、更少的 ALU 和更多的控制单元。

相比之下，GPU 使用大量 ALU 来最大化其计算能力和吞吐量，它们使用非常少量的芯片区域作为缓存和控制单元，使得其具有很高的延迟。

GPU真没那么复杂

要真正的用好GPU，或者使用GPU进行性能加速，你就必须切实的了解GPU的架构，尤其是不断升级迭代的GPU产品，需要做好软硬一体的开发。

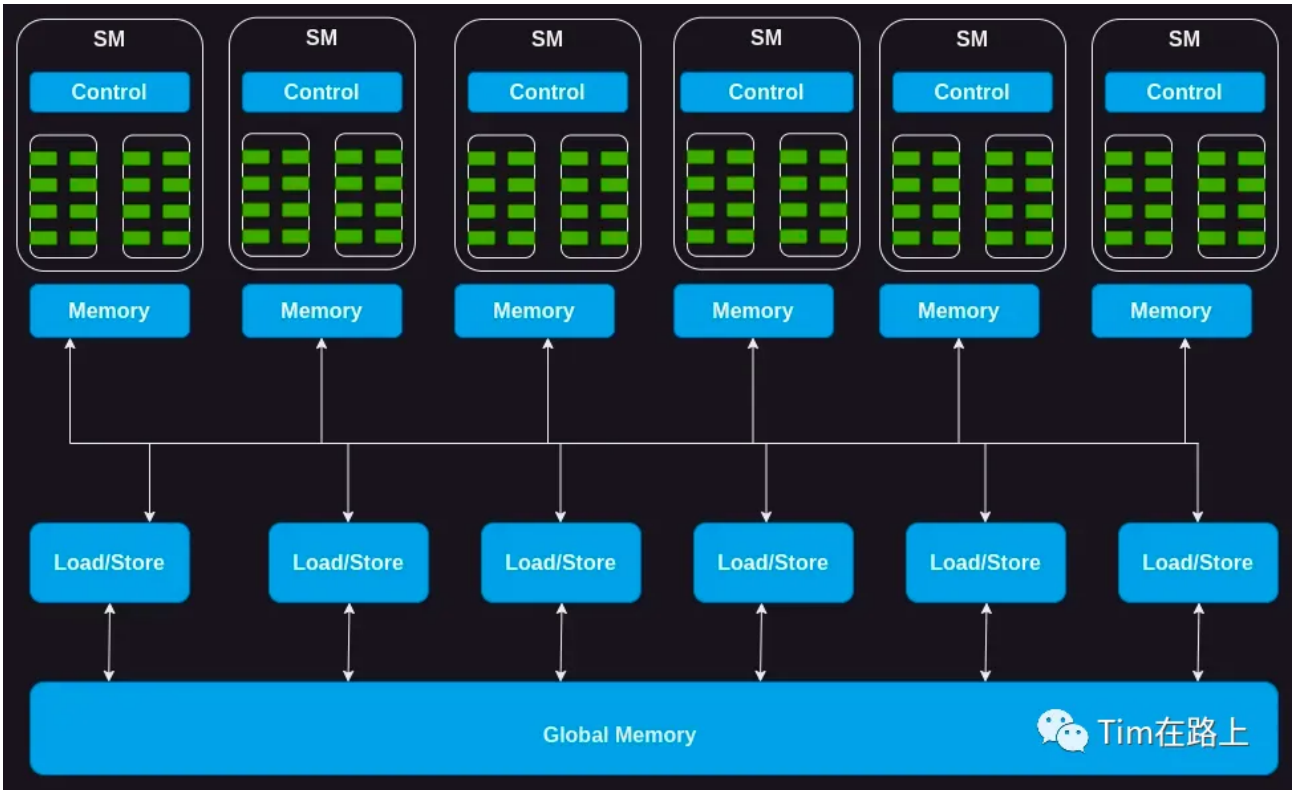
这不像很多基于CPU的软件工程师一样，即使我不了解不同CPU产品的特点，也能完美高效的完成运行。

但GPU硬件真的那么复杂吗？其实也没有，建议未来所有工程师都应该深入的了解GPU的架构原理，这样才能更好的适配和应用。

我们都知道 GPU 偏爱高吞吐量，那么什么样的架构才能实现这一目标？

1. GPU计算架构

如下图所示，GPU 是由一系列流式多处理器 (SM) 组成的。



SM是什么呢？说人话就对GPU核心进行分组管理，其中一组GPU core（线程组）就是SM。

例如，Nvidia H100 GPU 有 132 个 SM，每个 SM 有 64 个Core，总共有 8448 个Core。

当然了每一个分组的SM，也会分配一定的存储器，如上图所示的Memory。

通常称为共享存储器或暂存器，在所有内核之间共享。同样，SM 上的控制单元资源由所有核共享。

此外，每个 SM 都配备了基于硬件的线程调度程序来执行线程。

除此之外，每个 SM 还具有多个功能单元或其他加速计算单元，例如张量核心或光线追踪单元，以满足 GPU 所满足的工作负载的特定计算需求。

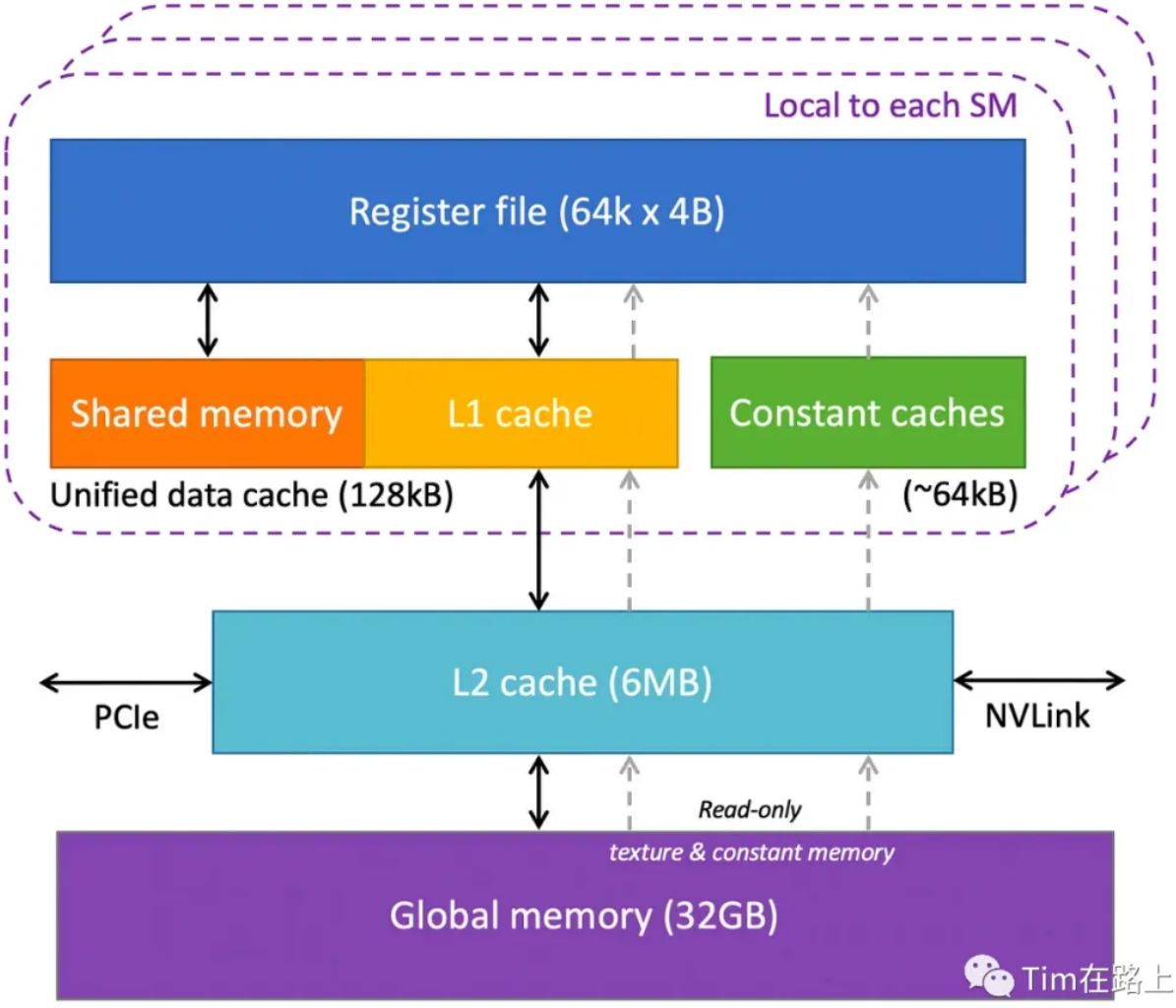
此外，每个SM又被分为多个（Block）即处理单元，每个Block中包含一个独立的Warp Scheduler，一个独立的Dispatch Unit，一个一定大小的Register File，一个独立的L0 Instruction Cache，多个INT32处理单元，多个FP32处理单元，以及多个TENSOR CORES单元。

这就好像，SM为一个班级，Block是一个小组一样。

2. GPU存储架构

GPU 有多层不同类型的存储器，从访问速度来看，这些存储结构按照从高到低排序依次是：

RMEM（寄存器）> SMEM（共享存储）> CMEM（常量存储）> TMEM（类常量存储）> LMEM（本地存储）> GMEM（全局存储）



上面中虚线中表示的就是一个SM中的存储结构，每一个SM都包括寄存器、共享内存、常量内存和L1 cache。

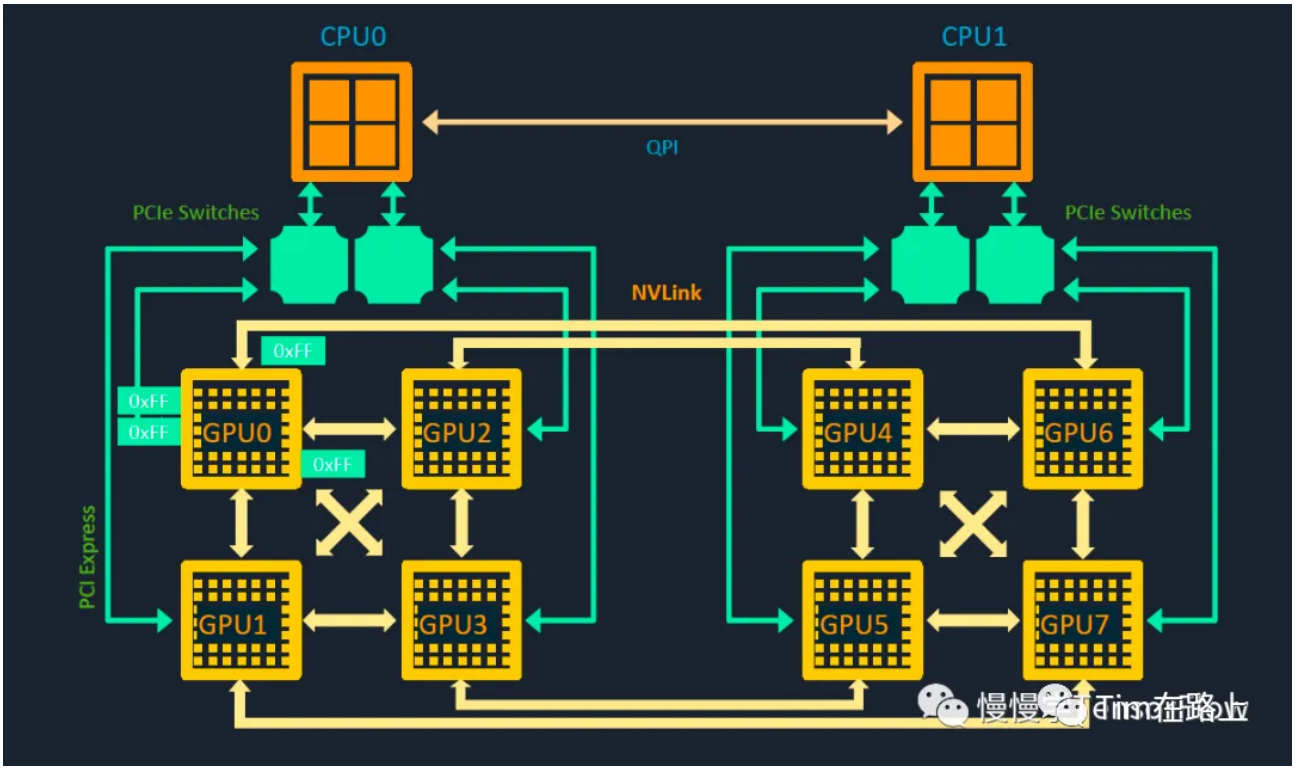
而在SM之外，还存在L2 Cache，全局存储，由所有SM 共享。

值得注意的是，我们平时所说的Nvidia H100 拥有 80 GB 显存，指的就是**Global Memory全局存储**，它是一种高容量、高带宽的DRAM。

由于距离SM较远，全局内存的延迟相当高。然而，片上存储器的几个附加层和大量计算单元有助于隐藏这种延迟。

3. GPU通信架构

一般一台服务器中一般不超过 16 张卡，最为常见的就是单机8 卡 GPU 服务器。



单机内部存在多组通信与拓扑关系，包括CPU-CPU、CPU-GPU、GPU-GPU。

如上图所示，其中 CPU0 和 CPU1 通过 QPI 互联，QPI（QuickPath Interconnect）是一种用于高性能处理器间通信的接口技术，提供了一个高速、点对点的数据通路，可以实现低延迟和高带宽的处理器之间的通信。

CPU 与 GPU 则通过 PCIe Switch 互联，如图所示，一个CPU与4个GPU间都是通过PCIe Switch通信，类似于交换机，每个PCIe Switch可以有多个PCIe插槽。

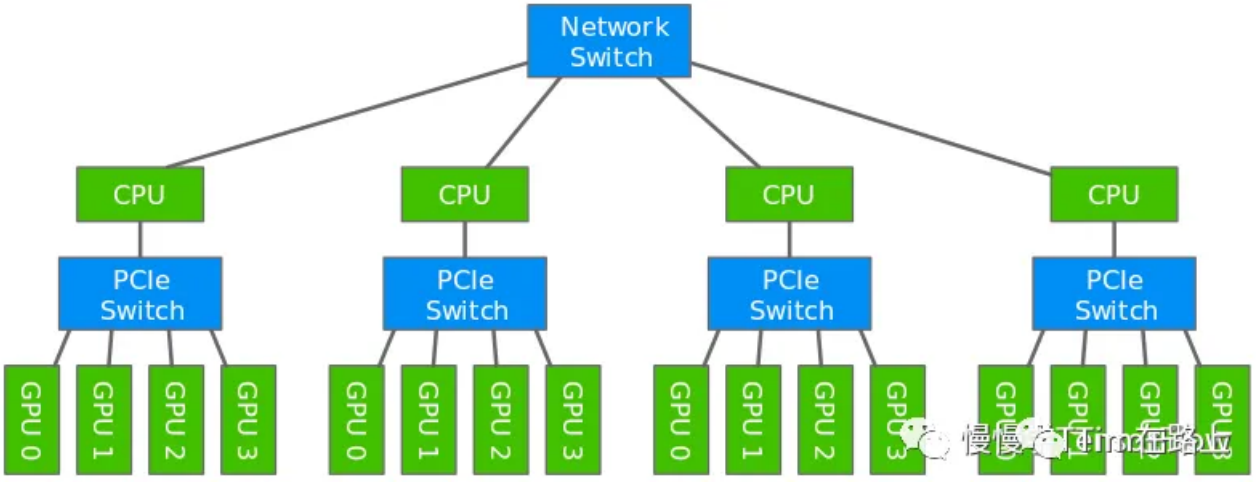
GPU 0~3 两两之间可以既可以通过 NVLink 通信又可以走 PCIe Switch，同样 GPU 4~7 也是如此。NVLink专门用于GPU之间的通信。

如图所示，GPU 0 和 GPU 4 之间没有直接通路，需要借助其他途径间接通信。

除此之外，NVIDIA 提供了用于多 GPU 通信库 NCCL（NVIDIA Collective Communications Libary），实现了 AllReduce、Reduce、Broadcast、ReduceScatter、AllGather 等常用通信原语，面向 PCIe 和 NVLink 做了专门优化，具有更高带宽、更快速度。

那么多机GPU间是如何通信的？

多台机器一般通过交换机相连，多机间的CPU与CPU通信一般需要经过Network Switch通信，通常基于socket原理进行通信，需要经过用户态与内核态的多次数据拷贝。



Nvidia 实现了一套IB网络技术，支持远程直接数据存取（Remote Direct Memory Access，简称 RDMA）技术。

可以将数据直接从一台计算机的内存传输到另一台计算机，无需双方操作系统以及 CPU 的介入，IB 单网口速度支持从 10Gb/s（SDR） 到 56Gbps（FDR）。

其缺点是IB需要专门支持 IB 的网卡，还要采购价格高昂的交换机设备。

如果想在以太网环境支持 RDMA 的标准，可以采用RoCE（RDMA over Converged Ethernet）通信技术，其无需复杂和低效的 TCP 传输，可以实现多机通信间的“零拷贝”，大幅降低通信延迟。

Nvidia GPU CUDA编程原理拆解

CUDA是Nvidia花了十年时间打造的一套编程生态，经历了投资人各种不理解，现在已经成为Nv最强护城河。

在 CUDA 中，以类似于 C 函数的形式表达要在 GPU 上运行的计算，该函数称为内核。

我们以向量加法为例子，来讲解下CUDA编程范式是如何执行的？

在这个例子中我们以一个内核将两个数组向量作为输入，将它们按元素相加并将结果写入另一个向量数组中。

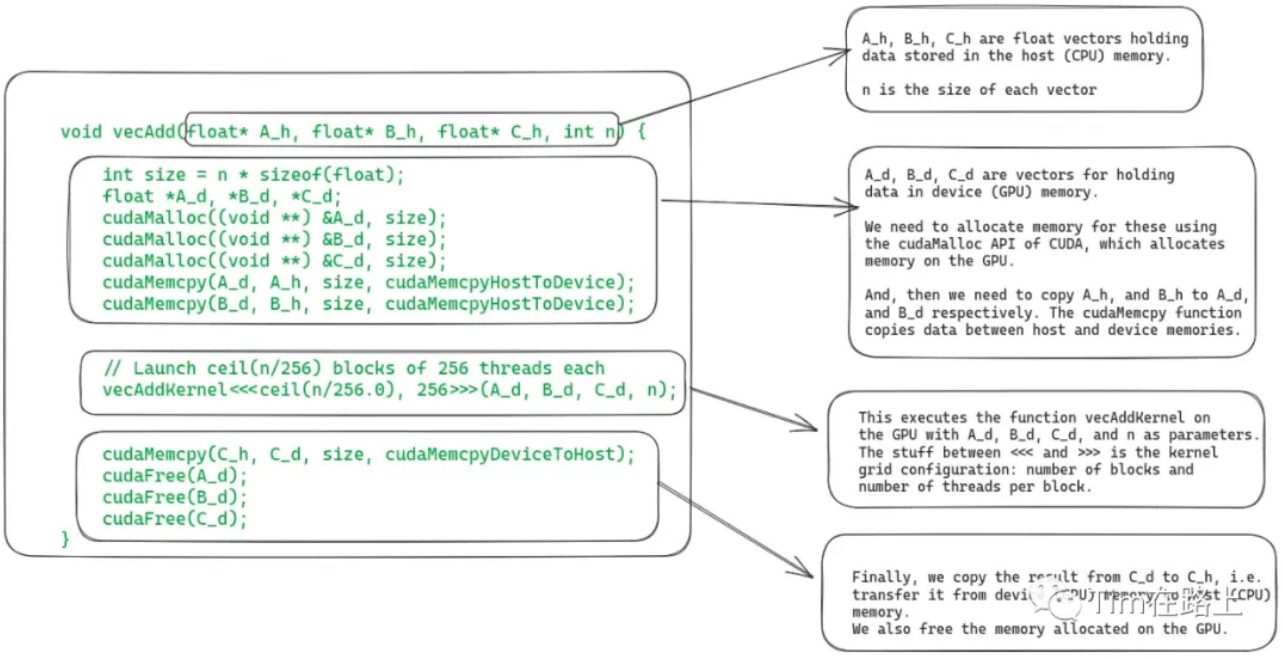
原本在CPU上的实现，我们可能需要实现一个for循环，依次遍历数组，并将数据相加写出到结果数组中。

在GPU上，最简单的实现就是为数组中的每个加法运算分配一个线程，分配的总线程数为数组的长度，这样同时进行计算，一次即可计算完成。

这就需要GPU可以实现调度管理和合理的分配线程，在Nvidia GPU上为管理所有的线程，线程首先被分为多个Grid（网格），每一个Grid中又被分割为多个Block（线程块），每一个Block块由一个或多个线程组成。

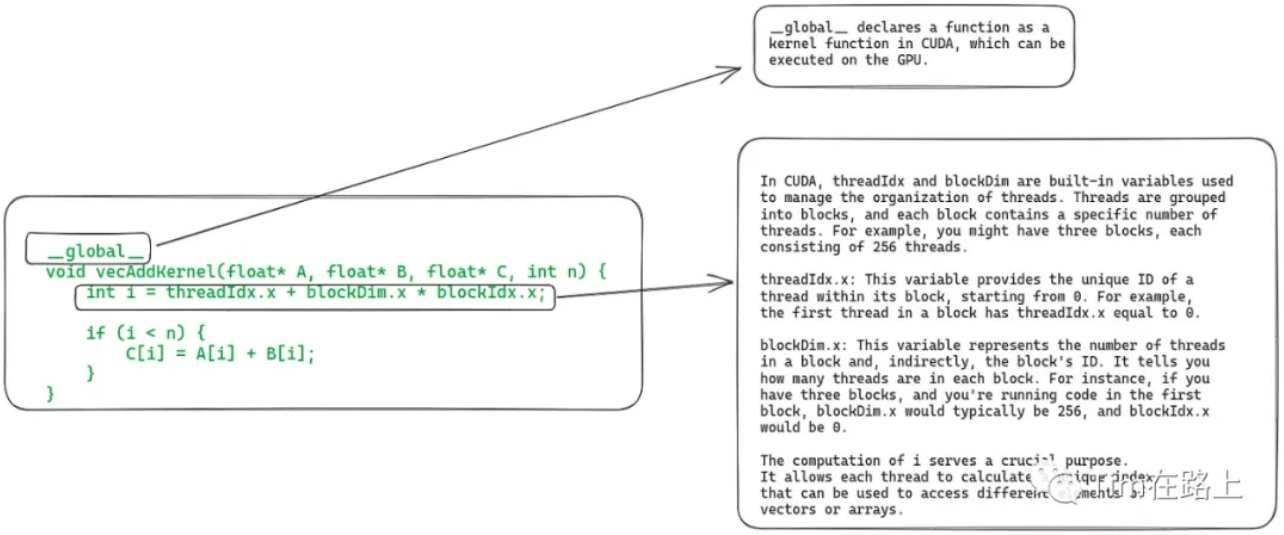
在申请线程数，按照网格和线程块进行申请，如果没有足够的可用线程，就得希望每个线程处理多个数据。

在实现上，GPU编程包括两部分，CPU上执行的主机代码（Host），GPU上执行的设备代码（GPU）。



如上图，展示的是CPU上执行的代码，它包括读取CPU上数组数据，申请GPU上的数组空间，将CPU数组拷贝到GPU以及使用配置的线程网格启动GPU内核。

在申请线程总数时，填写到<<<n/256, 256>>>中，其中n为数组长度，即总线程数，256为设置每block线程数。



上面是GPU设备上执行的代码，通过下面的代码可以拿到每一个线程id。

```
int i = threadIdx.x + blockDim.x * blockIdx.x;
```

如果这个线程id小于数组的长度n，就执行该线程所对应位置的加法，这样讲O(n)的计算转换为了O(1)。

下面我们总结下GPU的编程范式：

1. **将数据从主机复制到GPU设备；（当然在最新的 GPU 设备中，我们还可以使用统一虚拟内存直接从主机内存中读取数据）**
2. 申请和**调度SM上的线程块；**
3. **SM 通过获取并向Warp所有线程发出相同的指令来一起执行（单指令多线程）；**
4. **Warp线程组调度和延迟容忍，最大限度提供高吞吐量；**
5. **将结果数据从设备复制到主机内存；**

总结

据传Nvidia针对中国区已开发出最新改良版系列芯片：HGX H20、L20 PCIe和L2 PCIe。

然而这次会将进一步在算力上做性能阉割，这显然已不符合国内大模型长远训练的需求。

英伟达首席科学家Bill Dally曾表示：「**随着训练需求每6到12个月翻一番，这一差距将随着时间的推移而迅速扩大。**」

百度内部已经下令其AI系统“文心一言”使用的芯片，改向华为采购昇腾（Ascend）910B系列AI芯片。

Ascend 910B芯片最大算力为256TFLOPS，相当于Nvidia A100，虽然其目前生态上还不如CUDA。

目前GPU已得到普遍使用，虽然其架构和执行模型与CPU有很大不同。

通过深入理解Nvidia GPU的原理和CUDA生态，我们可以借鉴其优秀之处，实现'**它山之石可以攻玉**'，从而提升我们对GPU硬件架构的理解。