CPU 和 GPU 的区别是什么？

中央处理器 (CPU) 图形处理器（GPU）

关注者 2,713 被浏览 926,323

CPU 和 GPU 的区别是什么？

关注问题 写回答 1 条评论 分享 邀请回答 ···

37 个回答 默认排序

知乎用户

收录于编辑推荐 · 1,474 人赞同了该回答

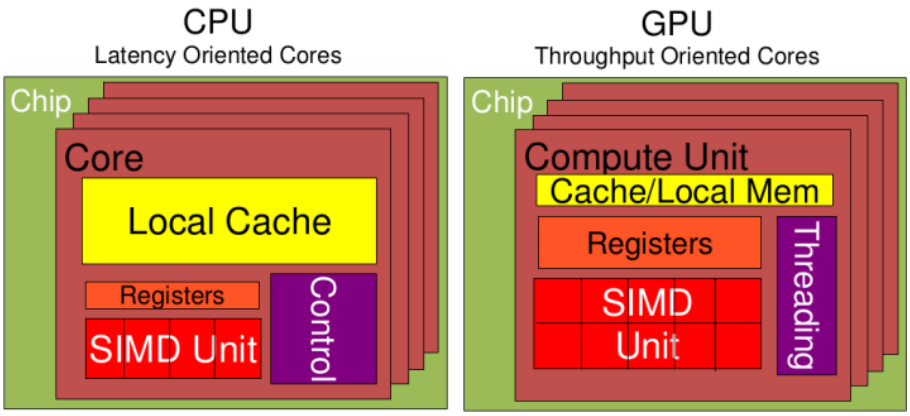
看了好多，觉得下面这个介绍才是我想要的以及能看明白的，转载自：
1.2CPU和GPU的设计区别

CPU和GPU之所以大不相同，是由于其设计目标的不同，它们分别针对了两种不同的应用场景。CPU需要很强的通用性来处理各种不同的数据类型，同时又要逻辑判断又会引入大量的分支跳转和中断的处理。这些都使得CPU的内部结构异常复杂。而GPU面对的则是类型高度统一的、相互无依赖的大规模数据和不需要被打断的纯净的计算环境。

于是CPU和GPU就呈现出非常不同的架构（示意图）：

图片来自nVidia CUDA文档。其中绿色的是计算单元，橙红色的是存储单元，橙黄色的是控制单元。

GPU采用了数量众多的计算单元和超长的流水线，但只有非常简单的控制逻辑并省去了Cache。而CPU不仅被Cache占据了大量空间，而且还有有复杂的控制逻辑和诸多优化电路，相比之下计算能力只是CPU很小的一部分



从上图可以看出：

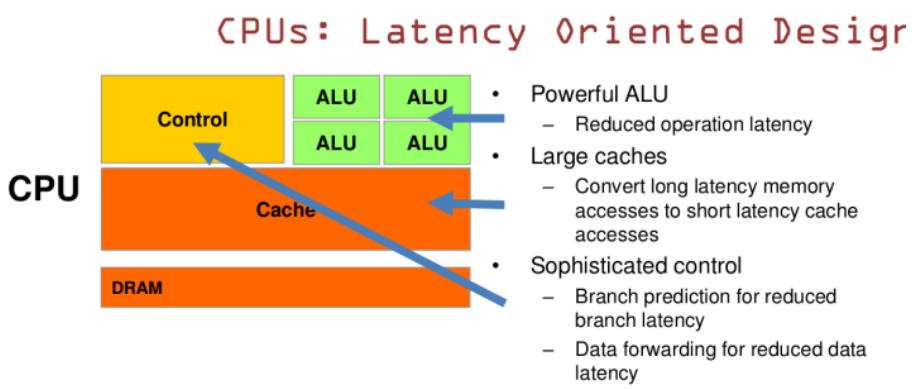
Cache, local memory： CPU > GPU

Threads(线程数): GPU > CPU

Registers: GPU > CPU 多寄存器可以支持非常多的Thread,thread需要用到register,thread数目大，register也必须得跟着很大才行。

SIMD Unit(单指令多数据流,以同步方式，在同一时间内执行同一条指令): GPU > CPU。

CPU 基于低延时的设计：



CPU有强大的ALU（算术运算单元），它可以在很少的时钟周期内完成算术计算。

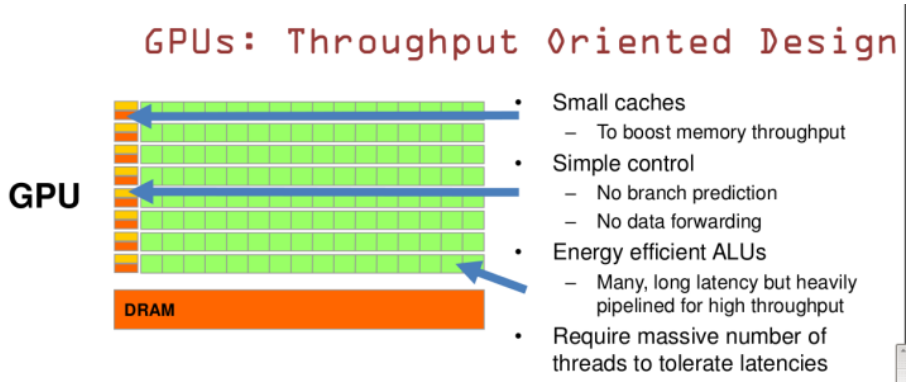
当今的CPU可以达到64bit 双精度。执行双精度浮点源算的加法和乘法只需要1～3个时钟周期。

CPU的时钟周期的频率是非常高的，达到1.532～3gigahertz(千兆HZ, 10的9次方).

大的缓存也可以降低延时。保存很多的数据放在缓存里面，当需要访问的这些数据，只要在之前访问过的，如今直接在缓存里面取即可。

复杂的逻辑控制单元。当程序含有多个分支的时候，它通过提供分支预测的能力来降低延时。

数据转发。 当一些指令依赖前面的指令结果时，数据转发的逻辑控制单元决定这些指令在pipeline中的位置并且尽可能快的转发一个指令的结果给后续的指令。这些动作需要很多的对比电路单元和转发电路单元。



GPU是基于大的吞吐量设计。

GPU的特点是有很多的ALU和很少的cache. 缓存的目的不是保存后面需要访问的数据的，这点和CPU不同，而是为thread提高服务的。如果有很多线程需要访问同一个相同的数据，缓存会合并这些访问，然后再去访问dram（因为需要访问的数据保存在dram中而不是cache里面），获取数据后cache会转发这个数据给对应的线程，这个时候是数据转发的角色。但是由于需要访问dram，自然会带来延时的问题。

GPU的控制单元（左边黄色区域块）可以把多个的访问合并成少的访问。

GPU的虽然有dram延时，却有非常多的ALU和非常多的thread. 为啦平衡内存延时的问题，我们可以中充分利用多的ALU的特性达到一个非常大的吞吐量的效果。尽可能多的分配多的Threads.通常来看GPU ALU会有非常重的pipeline就是因为这样。

所以与CPU擅长逻辑控制，串行的运算。和通用类型数据运算不同，GPU擅长的是大规模并发计算，这也正是密码破解等所需要的。所以GPU除了图像处理，也越来越多的参与到计算当中来。

GPU的工作大部分就是这样，计算量大，但没什么技术含量，而且要重复很多很多次。就像你有个工作需要算几亿次一百以内加减乘除一样，最好的办法就是雇上几十个小学生一起算，一人算一部分，反正这些计算也没什么技术含量，纯粹体力活而已。而CPU就像老教授，积分微分都会算，就是工资高，一个老教授资顶二十个小学生，你要是富士康你雇哪个？GPU就是这样，用很多简单的计算单元去完成大量的计算任务，纯粹的人海战术。这种策略基于一个前提，就是小学生A和小学

生B的工作没有什么依赖性，是互相独立的。很多涉及到大量计算的问题基本都有这种特性，比如你说的破解密码，挖矿和很多图形学的计算。这些计算可以分解为多个相同的简单小任务，每个任务就可以分给一个小学生去做。但还有一些任务涉及到“流”的问题。比如你去相亲，双方看着顺眼才能继续发展。总不能你这边还没见面呢，那边找人把证都给领了。这种比较复杂的问题都是CPU来做的。

总而言之，CPU和GPU因为最初用来处理的任务就不同，所以设计上有不小的区别。而某些任务和GPU最初用来解决的问题比较相似，所以用GPU来算了。GPU的运算速度取决于雇了多少小学生，CPU的运算速度取决于请了多么厉害的教授。教授处理复杂任务的能力是碾压小学生的，但是对于没那么复杂的任务，还是顶不住人多。当然现在的GPU也能做一些稍微复杂的工作了，相当于升级成初中生高中生的水平。但还需要CPU来把数据喂到嘴边才能开始干活，究竟还是靠CPU来管的。

什么类型的程序适合在GPU上运行？

（1）计算密集型的程序。所谓计算密集型(Compute-intensive)的程序，就是其大部分运行时间花在了寄存器运算上，寄存器的速度和处理器的速度相当，从寄存器读写数据几乎没有延时。可以做一下对比，读内存的延迟大概是几百个时钟周期；读硬盘的速度就不说了，即便是SSD, 也实在是太慢了。

（2）易于并行的程序。GPU其实是一种SIMD(Single Instruction Multiple Data)架构， 他有成百上千个核，每一个核在同一时间最好能做同样的事情。

编辑于 2016-04-20

王洋子豪
软件工程师, 关注图形学，并行计算以及深度学习

815 人赞同了该回答

首先需要解释CPU和GPU这两个缩写分别代表什么。CPU即中央处理器，GPU即图形处理器。其次，要解释两者的区别，要先明白两者的相同之处：两者都有总线和外界联系，有自己的缓存体系，以及数字和逻辑运算单元。一句话，两者都为了完成计算任务而设计。

两者的区别在于存在于片内的缓存体系和数字逻辑运算单元的结构差异：CPU虽然有多核，但总数没有超过两位数，每个核都有足够大的缓存和足够多的数字和逻辑运算单元，并辅助有很多加速分支判断甚至更复杂的逻辑判断的硬件；GPU的核数远超CPU，被称为众核（NVIDIA Fermi有512个核）。每个核拥有的缓存大小相对小，数字逻辑运算单元也少而简单（GPU初始时在浮点计算上一直弱于CPU）。从结果上导致CPU擅长处理具有复杂计算步骤和复杂数据依赖的计算任务，如分布式计算，数据压缩，人工智能，物理模拟，以及其他很多很多计算任务等。GPU由于历史原因，是为了视频游戏而产生的（至今其主要驱动力还是不断增长的视频游戏市场），在三维游戏中常常出现的一类操作是对海量数据进行相同的操作，如：对每一个顶点进行同样的坐标变换，对每一个顶点按照同样的光照模型计算颜色值。GPU的众核架构非常适合把同样的指令流并行发送到众核上，采用不同的输入数据执行。在2003-2004年左右，图形学之外的领域专家开始注意到GPU与众不同的计算能力，开始尝试把GPU用于通用计算（即GPGPU）。之后NVIDIA发布了CUDA，AMD和Apple等公司也发布了OpenCL，GPU开始在通用计算领域得到广泛应用，包括：数值分析，海量数据处理（排序，Map-Reduce等），金融分析等等。

简而言之，当程序员为CPU编写程序时，他们倾向于利用复杂的逻辑结构优化算法从而减少计算任务的运行时间，即Latency。当程序员为GPU编写程序时，则利用其处理海量数据的优势，通过提高总的数据吞吐量（Throughput）来掩盖Lantency。目前，CPU和GPU的区别正在逐渐缩小，因为GPU也在处理不规则任务和线程间通信方面有了长足的进步。另外，功耗问题对于GPU比CPU更严重。

总的来讲，GPU和CPU的区别是个很大的话题，甚至可以花一个学期用32个学时十几次讲座来讲，所以如果提问者有更具体的问题，可以进一步提出。我会在我的知识范围内尝试回答。

发布于 2012-02-01

3MSummer

169 人赞同了该回答

CPU 力气大啥P事都能干，还要协调。

GPU 上面那家伙的小弟，老大让他处理图形，这方面处理简单，但是量大，老大虽然能处理，可是

老大只有那么几个兄弟，所以不如交给小弟处理了，小弟兄弟多，有数百至数千个，而且是专门只干这行和只能干这行。

编辑于 2015-08-05

▲ 赞同 169 ▼ 8 条评论 分享 收藏 感谢

feng sam
还好动态里看不到我评论了什么√

141 人赞同了该回答

当你操作电脑的时候，为了完成某项工作，需要电脑帮你工作，就像计算某个题目那样。
计算题目，理解题目并且整理出解题的步骤以及解法，那是CPU的事情。
但是解题的过程需要用到的众多计算，则需要一帮不需要很高逻辑理解力的计算者完成，他们只需要负责其中很简单但是数量又很大的简单运算就行了，最后他们把各自运算的结果交出来给CPU整理，那么这群计算者就是GPU。
这就是一个博士带着100个小学生的意思了

发布于 2014-04-09

▲ 赞同 141 ▼ 14 条评论 分享 收藏 感谢

OF小工
智能，可穿戴

16 人赞同了该回答

GPU是显示卡的“心脏”，也就相当于CPU在电脑中的作用，它决定了该显卡的档次和大部分性能，同时也是2D显示卡和3D显示卡的区别依据。2D 显示芯片在处理3D图像和特效时主要依赖CPU的处理能力，称为“软加速”。3D显示芯片是将三维图像和特效处理功能集中在显示芯片内，也即所谓的“硬件加速”功能。显示芯片通常是显示卡上最大的芯片(也是引脚最多的)。GPU使显卡减少了对CPU的依赖，并进行部分原本CPU的工作，尤其是在3D图形处理时。GPU所采用的核心技术有硬体T&L、立方环境材质贴图和顶点混合、纹理压缩和凹凸映射贴图、双重纹理四像素256位渲染引擎等，而硬体 T&L技术可以说是GPU的标志。

GPU 能够从硬件上支持T&L(TransformandLighting，多边形转换与光源处理)的显示芯片，因为T&L是3D渲染中的一个重要部分，其作用是计算多边形的3D位置和处理动态光线效果，也可以称为“几何处理”。一个好的T&L单元，可以提供细致的3D物体和高级的光线特效;只不过大多数PC中，T&L的大部分运算是交由CPU处理的(这就也就是所谓的软件T&L)，由于CPU的任务繁多，除了 T&L之外，还要做内存管理、输入响应等非3D图形处理工作，因此在实际运算的时候性能会大打折扣，常常出现显卡等待CPU数据的情况，其运算速度远跟不上今天复杂三维游戏的要求。即使CPU的工作频率超过1GHz或更高，对它的帮助也不大，由于这是PC本身设计造成的问题，与CPU的速度无太大关系。

主要作用

今天，GPU已经不再局限于3D图形处理了，GPU通用计算技术发展已经引起业界不少的关注，事实也证明在浮点运算、并行计算等部分计算方面，GPU 可以提供数十倍乃至于上百倍于CPU的性能，如此强悍的“新星”难免会让CPU厂商老大英特尔为未来而紧张，NVIDIA和英特尔也经常为CPU和GPU 谁更重要而展开口水战。GPU通用计算方面的标准目前有 OPEN CL、CUDA、ATI STREAM。其中，OpenCL(全称Open Computing Language，开放运算语言)是第一个面向异构系统通用目的并行编程的开放式、免费标准，也是一个统一的编程环境，便于软件开发人员为高性能计算服务器、桌面计算系统、手持设备编写高效轻便的代码，而且广泛适用于多核心处理器(CPU)、图形处理器(GPU)、Cell类型架构以及数字信号处理器(DSP) 等其他并行处理器，在游戏、娱乐、科研、医疗等各种领域都有广阔的发展前景，AMD-ATI、NVIDIA现在的产品都支持OPEN CL。 NVIDIA公司在1999年发布GeForce 256图形处理芯片时首先提出GPU的概念。从此NV显卡的芯就用这个新名字GPU来称呼。GPU使显卡减少了对CPU的依赖，并进行部分原本CPU的工作，尤其是在3D图形处理时。GPU所采用的核心技术有硬体T&L、立方环境材质贴图和顶点混合、纹理压缩和凹凸映射贴图、双重纹理四像素256 位渲染引擎等，而硬体T&L技术可以说是GPU的标志。

工作原理

简单说GPU就是能够从硬件上支持T&L(Transform and Lighting，多边形转换与光源处理)的显示芯片，因为T&L是3D渲染中的一个重要部分，其作用是计算多边形的3D位置和处理动态光线效果，也可以称为“几何处理”。一个好的T&L单元，可以提供细致的3D物体和高级的光线特效;只不过大多数PC中，T&L的大部分运算是交由CPU处理的(这就也就是所谓的软件T&L)，由于CPU的任务繁多，除了T&L之外，还要做内存管理、输入响应等非3D图形处理工作，因此在实际运算的时候性能会大打折扣，常常出现显卡等待CPU数据的情况，其运算速度远跟不上今天复杂三维游戏的要求。即使CPU的工作频率超过 1GHz或更高，对它的帮助也不大，由于这是PC本身设计造成的问题，与CPU的速度无太大关系。

GPU与DSP区别

GPU在几个主要方面有别于DSP(Digital Signal Processing，简称DSP(数字信号处理)架构。其所有计算均使用浮点算法，而且目前还没有位或整数运算指令。此外，由于GPU专为图像处理设计，因此存储系统实际上是一个二维的分段存储空间，包括一个区段号(从中读取图像)和二维地址(图像中的X、Y坐标)。此外，没有任何间接写指令。输出写地址由光栅处理器确定，而且不能由程序改变。这对于自然分布在存储器之中的算法而言是极大的挑战。最后一点，不同碎片的处理过程间不允许通信。实际上，碎片处理器是一个SIMD数据并行执行单元，在所有碎片中独立执行代码。

尽管有上述约束，但是GPU还是可以有效地执行多种运算，从线性代数和信号处理到数值仿真。虽然概念简单，但新用户在使用GPU计算时还是会感到迷惑，因为GPU需要专有的图形知识。这种情况下，一些软件工具可以提供帮助。两种高级描影语言CG和HLSL能够让用户编写类似C的代码，随后编译成碎片程序汇编语言。Brook是专为GPU计算设计，且不需要图形知识的高级语言。因此对第一次使用GPU进行开发的工作人员而言，它可以算是一个很好的起点。Brook是C语言的延伸，整合了可以直接映射到 GPU的简单数据并行编程构造。经 GPU存储和操作的数据被形象地比喻成“流”(stream)，类似于标准C中的数组。核心(Kernel)是在流上操作的函数。在一系列输入流上调用一个核心函数意味着在流元素上实施了隐含的循环，即对每一个流元素调用核心体。Brook还提供了约简机制，例如对一个流中所有的元素进行和、最大值或乘积计算。Brook还完全隐藏了图形API的所有细节，并把GPU中类似二维存储器系统这样许多用户不熟悉的部分进行了虚拟化处理。用Brook编写的应用程序包括线性代数子程序、快速傅立叶转换、光线追踪和图像处理。利用ATI的X800XT和Nvidia的GeForce 6800 Ultra型GPU，在相同高速缓存、SSE汇编优化Pentium 4执行条件下，许多此类应用的速度提升高达7倍之多。

对GPU计算感兴趣的用户努力将算法映射到图形基本元素。类似Brook这样的高级编程语言的问世使编程新手也能够很容易就掌握GPU的性能优势。访问GPU计算功能的便利性也使得GPU的演变将继续下去，不仅仅作为绘制引擎，而是会成为个人电脑的主要计算引擎。

GPU和CPU的区别是什么?

要解释两者的区别，要先明白两者的相同之处：两者都有总线和外界联系，有自己的缓存体系，以及数字和逻辑运算单元。一句话，两者都为了完成计算任务而设计。

两者的区别在于存在于片内的缓存体系和数字逻辑运算单元的结构差异：CPU虽然有多核，但总数没有超过两位数，每个核都有足够大的缓存和足够多的数字和逻辑运算单元，并辅助有很多加速分支判断甚至更复杂的逻辑判断的硬件;GPU的核数远超CPU，被称为众核(NVIDIA Fermi有512个核)。每个核拥有的缓存大小相对小，数字逻辑运算单元也少而简单(GPU初始时在浮点计算上一直弱于CPU)。从结果上导致CPU擅长处理具有复杂计算步骤和复杂数据依赖的计算任务，如分布式计算，数据压缩，人工智能，物理模拟，以及其他很多很多计算任务等。GPU由于历史原因，是为了视频游戏而产生的(至今其主要驱动力还是不断增长的视频游戏市场)，在三维游戏中常常出现的一

类操作是对海量数据进行相同的操作，如：对每一个顶点进行同样的坐标变换，对每一个顶点按照同样的光照模型计算颜色值。GPU的众核架构非常适合把同样的指令流并行发送到众核上，采用不同的输入数据执行。在 2003-2004年左右，图形学之外的领域专家开始注意到GPU与众不同的计算能力，开始尝试把GPU用于通用计算(即GPGPU)。之后NVIDIA 发布了CUDA，AMD和Apple等公司也发布了OpenCL，GPU开始在通用计算领域得到广泛应用，包括：数值分析，海量数据处理(排序，Map- Reduce等)，金融分析等等。

简而言之，当程序员为CPU编写程序时，他们倾向于利用复杂的逻辑结构优化算法从而减少计算任务的运行时间，即Latency。当程序员为GPU编写程序时，则利用其处理海量数据的优势，通过提高总的数据吞吐量(Throughput)来掩盖 Lantency。目前，CPU和GPU的区别正在逐渐缩小，因为GPU也在处理不规则任务和线程间通信方面有了长足的进步。另外，功耗问题对于GPU比 CPU更严重。

总的来讲，GPU和CPU的区别是个很大的话题，甚至可以花一个学期用32个学时十几次讲座来讲。

发布于 2016-03-22