****异构计算（Heterogeneous Computing）是指在同一计算系统中，使用不同类型的处理单元来完成任务的计算方法****

****异构计算的基本原理****

[什么是异构计算？-CSDN博客](https://blog.csdn.net/weixin_49007164/article/details/143061229?ops_request_misc=%7B%22request%5Fid%22%3A%221d9ae8af3241af6fa8e67e6d7b2283f5%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=1d9ae8af3241af6fa8e67e6d7b2283f5&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-143061229-null-null.142^v100^pc_search_result_base1&utm_term=%E5%BC%82%E6%9E%84%E8%AE%A1%E7%AE%97&spm=1018.2226.3001.4187)

****异构计算的基本原理主要包括以下几个方面：****

****任务分配：将计算任务根据不同处理单元的特点进行合理分配。例如，将并行计算密集型任务分配给GPU，而将串行计算任务留给CPU。****

****并行处理：利用不同处理单元的并行处理能力，加速计算过程。GPU特别适合处理大规模并行计算任务，因此在图像处理、机器学习等领域应用广泛。****

****数据传输：在不同处理单元之间高效传输数据，确保计算的流畅性。数据传输的效率直接影响到整个系统的性能。****

****协调与管理：通过适当的软件框架和编程模型，协调不同处理单元的工作，管理资源，以实现最佳性能。****

****异构计算的关键技术****

****编程模型：异构计算需要适合不同硬件的编程模型，例如OpenCL、CUDA等。这些模型允许开发者以统一的方式编写代码，同时针对不同硬件优化性能。****

****任务调度：合理的任务调度策略可以有效提高计算效率。通过动态分配任务，系统能够根据实时负载情况进行自我调整，确保各处理单元的高效利用。****

****内存管理：在异构计算中，不同处理单元可能使用不同的内存架构。高效的内存管理策略可以减少数据传输的延迟，提高计算性能。****

****性能监控：实时监控系统的性能，收集各处理单元的运行状态数据，以便进行性能优化和故障诊断****

[异构计算--CUDA架构-CSDN博客](https://blog.csdn.net/qq_44924694/article/details/126202388?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_utm_term~default-1-126202388-blog-143061229.235^v43^control&spm=1001.2101.3001.4242.2&utm_relevant_index=4)

****与CPU相比，CPU芯片空间的不到20%是ALU，而GPU芯片空间的80%以上是ALU。即GPU拥有更多的ALU用于数据并行处理。****

****从硬件架构分析来看，CPU和GPU似乎很像，都有内存、Cache、ALU、CU，都有着很多的核心，但是CPU的核心占比比较重，相对计算单元ALU很少，可以用来处理非常复杂的控制逻辑，预测分支、乱序执行、多级流水任务等等。相对而言GPU的核心就是比较轻，用于优化具有简单控制逻辑的数据并行任务，注重并行程序的吞吐量。****

****简单来说就是CPU的核心擅长完成多重复杂任务，重在逻辑，重在串行程序；GPU的核心擅长完成具有简单的控制逻辑的 任务，重在计算，重在并行****

### **3.NVIDIA显卡硬件架构：SM、SP、Warp**

****1.流处理器SP（streaming processor）****  
  
****2****.**S**M（streaming multiprocessor）****  
  
  
****Warp调度****  
一个SP可以执行一个thread，但是实际上并不是所有的thread能够在同一时刻执行。Nvidia把32个threads组成一个warp，warp是调度和运行的基本单元。warp中所有threads并行的执行相同的指令。warp由SM的硬件warp scheduler负责调度，一个SM同一个时刻可以执行多个warp，这取决于warp scheduler的数量。目前每个warp包含32个threads（Nvidia保留修改数量的权利）。

### **4.软件架构：Kernel、Grid、Block**

具体到我们如何调用GPU上的线程实现我们的算法，则是通过Kernel实现的。在GPU上调用的[函数](https://marketing.csdn.net/p/3127db09a98e0723b83b2914d9256174?pId=2782&utm_source=glcblog&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_44924694/article/details/_blank)成为CUDA核函数（Kernel function），核函数会被GPU上的多个线程执行。我们可以通过如下方式来定义一个kernel：

kernel\_function<<<grid, block>>>(param1, param2, param3....);

* 1

如下图所示为kernel、grid、block的组织方式以及对应的硬件部分。  
  
****Grid：**** 由一个单独的kernel启动的所有线程组成一个grid，grid中所有线程共享global memory。Grid由很多Block组成，可以是一维二维或三维。

****Block：**** 一个grid由许多block组成，block由许多线程组成，同样可以有一维、二维或者三维。block内部的多个线程可以同步（synchronize），可访问共享内存（share memory）。