8.2 异步计算

此节内容对应PyTorch的版本本人没怎么用过，网上参考资料也比较少，所以略:)，有兴趣的可以去看看**原文**。

关于PyTorch的异步执行我只在**官方文档**找到了一段:

> **By default, GPU operations are asynchronous. When you call a function that uses the GPU, the operations are enqueued to the particular device, but not necessarily executed until later. This allows us to execute more computations in parallel, including operations on CPU or other GPUs. In general, the effect of asynchronous computation is invisible to the caller, because (1) each device executes operations in the order they are queued, and (2) PyTorch automatically performs necessary synchronization when copying data between CPU and GPU or between two GPUs. Hence, computation will proceed as if every operation was executed synchronously. You can force synchronous computation by setting environment variable CUDA_LAUNCH_BLOCKING=1. This can be handy when an error occurs on the GPU. (With asynchronous execution, such an error isn't reported until after the operation is actually executed, so the stack trace does not show where it was requested.)**

大致翻译一下就是: 默认情况下，PyTorch中的 GPU 操作是异步的。当调用一个使用 GPU 的函数时，这些操作会在特定的设备上排队但不一定会在稍后立即执行。这就使我们可以并行更多的计算，包括 CPU 或其他 GPU 上的操作。 一般情况下，异步计算的效果对调用者是不可见的，因为（1）每个设备按照它们排队的顺序执行操作，（2）在 CPU 和 GPU 之间或两个 GPU 之间复制数据时，PyTorch会自动执行必要的同步操作。因此，计算将按每个操作同步执行的方式进行。 可以通过设置环境变量 `CUDA_LAUNCH_BLOCKING = 1` 来强制进行同步计算。当 GPU 产生error时，这可能非常有用。（异步执行时，只有在实际执行操作之后才会报告此类错误，因此堆栈跟踪不会显示请求的位置。）