8.3 自动并行计算

上一节提到，默认情况下，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，但不一定会在稍后执行。这允许我们并行更多的计算，包括 CPU 或其他 GPU 上的操作。 下面看一个简单的例子。

首先导入本节中实验所需的包或模块。注意，需要至少2块GPU才能运行本节实验。

```
import torch
import time

assert torch.cuda.device_count() >= 2
```

我们先实现一个简单的计时类。

```
class Benchmark():  # 本类已保存在d2lzh_pytorch包中方便以后使用
    def __init__(self, prefix=None):
        self.prefix = prefix + ' ' if prefix else ''

    def __enter__(self):
        self.start = time.time()

    def __exit__(self, *args):
        print('%stime: %.4f sec' % (self.prefix, time.time() - self.start))
```

再定义 `run` 函数，令它做20000次矩阵乘法。

```
def run(x):
    for _ in range(20000):
        y = torch.mm(x, x)
```

接下来，分别在两块GPU上创建 `Tensor` 。

```
x_gpu1 = torch.rand(size=(100, 100), device='cuda:0')
x_gpu2 = torch.rand(size=(100, 100), device='cuda:1')
```

然后，分别使用它们运行 `run` 函数并打印运行所需时间。

```
with Benchmark('Run on GPU1.'):
    run(x_gpu1)
    torch.cuda.synchronize()

with Benchmark('Then run on GPU2.'):
    run(x_gpu2)
    torch.cuda.synchronize()
```

输出：

```
Run on GPU1. time: 0.2989 sec
Then run on GPU2. time: 0.3518 sec
```

尝试系统能自动并行这两个任务：

```
with Benchmark('Run on both GPU1 and GPU2 in parallel.'):
    run(x_gpu1)
    run(x_gpu2)
    torch.cuda.synchronize()
```

输出：

```
Run on both GPU1 and GPU2 in parallel. time: 0.5076 sec
```

可以看到，当两个计算任务一起执行时，执行总时间小于它们分开执行的总和。这表明，PyTorch能有效地实现在不同设备上自动并行计算。