### 异步编程之我所见

记得2年前的一次面试中面试官问我什么是“异步（编程）”，当时我不确定自己是不是对这个概念有没有准确的理解，我就给他打了个比方：假如你要烧一壶水，你不可能一直在旁边等着水开，你肯定得在水开之前去干别的事情，等水开了之后你再回头继续处理它（比如把它灌进热水瓶）。记得当时面试官对我的这个解释很满意。直到现在我也还是不确定到底对这个概念有没有更深的理解，至少现在再拿这个比喻说事情还是很合适的。

说到“异步”，那必须要说的就是“同步”，所谓同步就是一直在旁边等水开了呗。我有一段时间一直简单的以为以异步的方式处理事情在速度上肯定会比同步的方式快，但是如果仔细想，这样理解并不准确。就拿烧水这件事本身来说，无论在同步还是异步的情况下，水从常温到烧开再到它进入热水瓶的这段时间几乎是一样的，可能在异步的情况下还不如同步呢（因为水开的时候你有可能正在处理其他的事情要耽搁下，而如果你是一直守在旁边就不一样了，自然这个时间相对烧水这件事本身来说可以忽略不计）。那既然这样，异步的优势在哪？再拿这个例子来说，如果你是一直守着等水开（同步），就在这时你老妈突然又叫你去门口小店买瓶酱油，你说：我在等水烧开呐~。然后你老妈不得不叫你妹妹或者弟弟去买。而如果你不是那么傻（异步），那就同时可以替你妈去买酱油，回来指不定水刚好开了。我们不是傻子，肯定不会这么干，否则就太浪费劳动力了，对于计算机来说就是资源利用率太低了（CPU）。所以异步的好处往往并不是提高做一件事情的速度，而是提高做很多事情的整体效率。

说到异步很多人都会想到多线程，觉得没有多线程肯定就没法异步了。异步是否一定需要多线程？我觉得两者没有必然的联系，你仍然可以在一个线程的情况下完成异步操作，就像你一个人仍然可以在烧水的时候干很多其他事情。典型地拿Node来说，它不就是只有一个线程么。我们不妨拿Node处理Http请求的例子来做一个简单的计算，比如有一个Http请求的流程是这样的：

1. 请求到达Node，Node解析请求（请求到达Node前的网络传输时间不计算在内，因为网络的好坏不影响Node处理请求的速度）；（1s）
2. Node发出数据库查询请求（IO操作）；（98s）
3. Node得到数据库查询结果将其发送给客户端（假设这个结果很短，从数据库读取值的时间忽略不计，当然发送响应网络传输时间也不计在内）；（1s）

我在上面的三个步骤中分别标记了处理时间，这个请求的时间消耗主要是在发出数据库查询请求到得到结果的这段时间（IO等待），需要98秒，其他两步只需要1秒时间，因为只是简单cpu计算，不需要IO操作。这样数量级的差距是很平常的。那么对于一个客户端来说，从请求到达Node到得到Node发回响应至少需要100s（中间减去了客户端和Node之间的网络传输时间）。现在有100个这样的请求先后到达了Node（假设是每秒1个），我们先假设Node是以同步方式处理这些请求，也就是说，Node在发出数据库查询请求之后一直等在那边什么事也不干。那么完成这100个请求需要100\*100=10000s，对于客户端来说，排的越后的请求响应时间越长，最长的需要100\*100 - 100=9900s，平均下每个客户端的响应时间是（9900+100）/ 2=5000s简直不能忍。但是Node采用的是异步IO，也就是说，在等待数据库查询结果返回之前它并不是啥事也不干干等在那边，在发出第1个数据库请求之后，继续发第2个，第3个......，经过在100s后发送完所有这100数据库查询请求，完了之后，在理想情况下这时Node的事件队列中已经有最早的那2个请求的数据库查询结果返回了，此时Node继续处理后续的数据库返回的结果，将其发送给客户端。现在来算一下每个客户端的响应时间：1s（Node解析请求时间）+ 98s（数据库查询时间）+ 1s（返回结果时间）+2s（在数据库返回结果队列中等待的时间）=102s，几乎和只有1个请求的时候一样！拿这个例子其实是要说明2点：

1. 异步和多线程没有直接关系；
2. 异步的好处是提高系统的伸缩性，响应性，资源利用率（上面的例子Node的CPU几乎的全负荷运行的，看看自己的电脑吧，CPU 90%的时间都躺在那啥事也不干！）；

某些情况下我甚至可以得出这样的结论：异步的目的就是要提高cpu的利用率，不要让它闲着！这个结论一般适用于IO限制的操作中（需要长时间IO等待），在计算限制的操作中（CPU要进行一个长时间的计算），比如在GUI程序中，我要通过一个关键字在选定的目录中查找所有出现过该关键字的文件以及出现的次数，这个操作可能会因为文件数量过多耗时很久，所以这个计算操作不能在主线程中进行，必须另外创建另外的线程去执行，在将来计算完成后告诉主线程计算结果。所以一般在计算限制的操作中，异步操作才需要涉及到多线程，对于主线程来说，操作是以异步方式进行的。当然这个时候cpu的利用率也就说不上是什么好处了，在计算限制的操作中，异步主要是防止主线程卡死。虽然这里用到了多线程技术，但是这里的多线程只是为了能够让主线程以异步的方式处理事情，和异步本身并没有太大关系。还有关于多线程的使用就是让多个线程同时处理一个事情，提高处理速度，比如当前的例子中子线程可以又开启多个线程并行地查找关键字，然而这又是另一码事，和异步一点关系都没有。