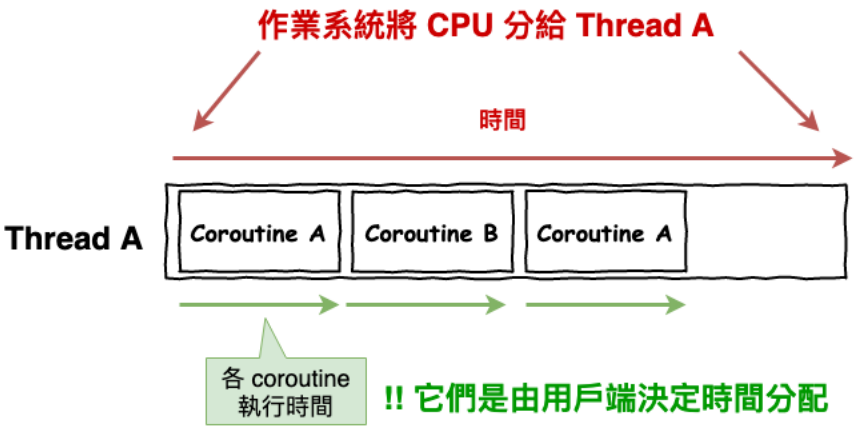
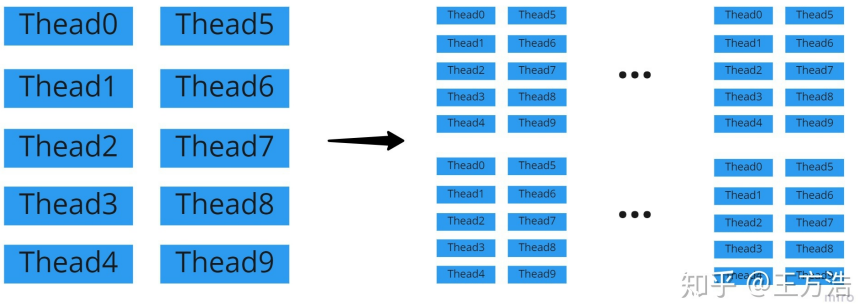
什么是协程？



为什么需要协程？

我们都知道多线程，当需要同时执行多项任务的时候，就会采用多线程并发执行。拿手机支付举例子，当收到付款信息的时候，需要查询数据库来判断余额是否充足，然后再进行付款。

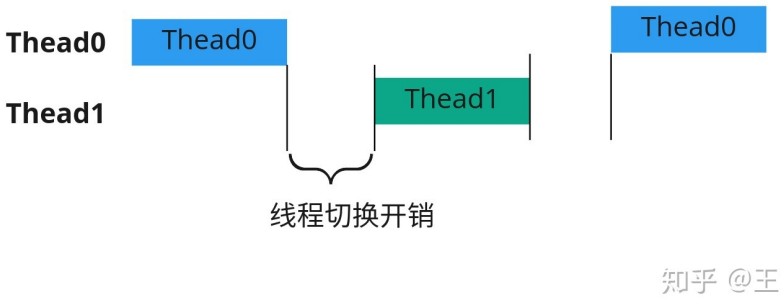
假设最开始我们只有可怜的10个用户，收到10条付款消息之后，我们开启启动10个线程去查询数据库，由于用户量很少，结果马上就返回了。第2天用户增加到了100人，你选择增加100个线程去查询数据库，等到第三天，你们加大了优惠力度，这时候有1000人同时在线付款，你按照之前的方法，继续采用1000个线程去查询数据库，并且隐隐觉察到有什么不对。



不断增长的线程

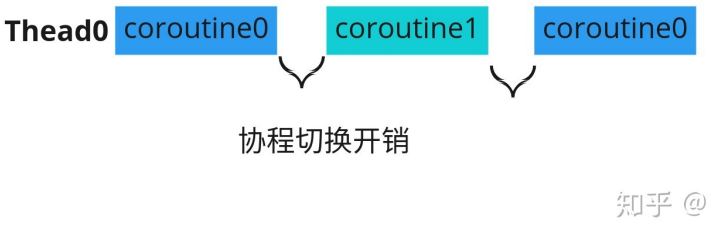
几天之后，见势头大好，运营部门开始不停的补贴消费券，展开了史无前例的大促销，你们的用户开始爆炸增长，这时候有10000人同时在线付款，你打算启动10000个线程来处理任务。等等，问题来了，因为每个线程至少会占用4M的内存空间，10000个线程会消耗39G的内存，而服务器的内存配置只有区区8G，这时候你有2种选择，一是选择增加服务器，二是选择提高代码效率。那么是否有方法能够提高效率呢？

我们知道操作系统在线程等待IO的时候，会阻塞当前线程，切换到其它线程，这样在当前线程等待IO的过程中，其它线程可以继续执行。当系统线程较少的时候没有什么问题，但是当线程数量非常多的时候，却产生了问题。**一是系统线程会占用非常多的内存空间，二是过多的线程切换会占用大量的系统时间。**



线程切换

协程刚好可以解决上述2个问题。协程运行在线程之上，当一个协程执行完成后，可以选择主动让出，让另一个协程运行在当前线程之上。**协程并没有增加线程数量，只是在线程的基础之上通过分时复用的方式运行多个协程**，而且协程的切换在用户态完成，切换的代价比线程从用户态到内核态的代价小很多。



协程切换

回到上面的问题，我们只需要启动100个线程，每个线程上运行100个协程，这样不仅减少了线程切换开销，而且还能够同时处理10000个读取数据库的任务，很好的解决了上述任务。

知道了协程的工作方式，那么我们再看下使用协程有哪些注意事项。

协程的注意事项

实际上协程并不是什么银弹，协程只有在等待IO的过程中才能重复利用线程，上面我们已经讲过了，线程在等待IO的过程中会陷入阻塞状态，意识到问题没有？

假设协程运行在线程之上，并且协程调用了一个阻塞IO操作，这时候会发生什么？实际上操作系统并不知道协程的存在，它只知道线程，**因此在协程调用阻塞IO操作的时候，操作系统会让线程进入阻塞状态，当前的协程和其它绑定在该线程之上的协程都会陷入阻塞而得不到调度，这往往是不能接受的。**

因此在协程中不能调用导致线程阻塞的操作。也就是说，协程只有和异步IO结合起来，才能发挥最大的威力。

那么如何处理在协程中调用阻塞IO的操作呢？一般有2种处理方式：

1. **在调用阻塞IO操作的时候，重新启动一个线程去执行这个操作，等执行完成后，协程再去读取结果。这其实和多线程没有太大区别。**
2. **对系统的IO进行封装，改成异步调用的方式，这需要大量的工作，最好寄希望于编程语言原生支持。**

协程对计算密集型的任务也没有太大的好处，计算密集型的任务本身不需要大量的线程切换，因此协程的作用也十分有限，反而还增加了协程切换的开销。

以上就是协程的注意事项。这里顺带一提JavaScript的异步变同步的调用方式，如果协程能够实现该类型的语法，不仅可以把异步操作变为同步，同时在IO操作的时候还能够不占用CPU，写起来非常方便。

异步变同步的调用方式只是一种编程方式，不管是用线程还是用协程都可以实现这种编程方式，好处是不用在处理非常多的回调。

```
async function getProcessedData(url) {
  let v;
  try {
    v = await downloadData(url);
  } catch(e) {
```

```
    v = await downloadFallbackData(url);
  }
  try {
    return await processDataInWorker(v);
  } catch (e) {
    return null;
  }
}
```

总结

在有大量IO操作业务的情况下，我们采用协程替换线程，可以到达很好的效果，一是降低了系统内存，二是减少了系统切换开销，因此系统的性能也会提升。

在协程中尽量不要调用阻塞IO的方法，比如打印，读取文件，Socket接口等，除非改为异步调用的方式，并且协程只有在IO密集型的任务中才会发挥作用。

协程只有和异步IO结合起来才能发挥出最大的威力。
