特立独行MVP

【第三章：BAT等名企面试真题解析8讲】第15节：字节面试真题解析之协程 已购

来自【《收割BAT：C++校招学习路线总结》】 | 114 浏览 | 0 回复 | 2020-02-21

特立独行MVP 专栏作者 +关注

前言

在第12节当中我们介绍了面试当中常考的进程和线程相关的问题，除了进程和线程之外，协程相关的问题在面试当中出现的频次越来越高，尤其是在一些使用Golang作为主要后端开发语言的大厂当中。本节我将以字节面试真题为例讲解协程相关知识。

并发模型

要理解协程的出现，首先需要了解网络服务器并发模型有哪些：

1.简单多线程模型：

该模型采用一个连接一个线程的模式，对于每一个连接都需要一个单独的线程去处理业务逻辑。

2.半同步半异步模型：

单独一个IO线程来异步处理各种网络IO，比如使用IO多路复用技术。同时使用线程池来同步处理每个请求，业务逻辑部分交给线程池当中的工作线程处理。

3.全异步模型：

该模型在网络IO部分和业务逻辑处理部分都是异步的，不会因为IO而导致程序阻塞，通过一个线程就处理所有的任务，由于避免了多线程的开销问题，能够**最大程度利用计算机的性能**，但是将给程序员带来编程的巨大困难，因为业务逻辑的编写将非常难以理解。

协程

我们知道线程可以看做轻量级进程，而协程简单来说可以看做更轻量级的线程。不同的语言有不同的协程实现，比如Golang原生支持协程，Python的Greenlet，C++的第三方库libgo等等。如同一个进程可以拥有多个线程，一个线程也可以拥有多个协程。如同线程是进程的一种优化，协程也可以看做线程的进一步优化。

协程内存开销：

以Golang为例，协程初始化创建的时候为其分配的栈有2KB，而线程栈一般为8M左右。如果对每个连接创建一个协程去处理，100万并发请求只需要2G内存，而如果用线程模型则需要8T，所以对于相同的内存而言，使用协程可以支持的并发量比线程多很多。

用户级协程和操作系统线程的多对多的映射关系，Golang抽象出G，P，M的概念，将协程的调度完全实现在用户态的runtime当中：

M: Machine，简单理解为操作系统线程；

G: Goroutine，协程；

P: Processor，处理器抽象，协调多个个G在某一个M上执行。

有兴趣的话可以去看看Golang在协程方面的设计和实现。

字节跳动面试真题解析

1.协程的切换在什么时候？

答：1.当Golang当中协程在执行一些阻塞调用，例如网络IO，磁盘IO函数会发生协程切换；

2.Golang当中协程的切换时间片是10ms，当协程连续执行超过10ms时会被runtime调度器切换;

3.当协程主动交出执行权限时也会发生协程切换，比如协程主动sleep。

分析：该问题是考察对于协程切换的理解，协程切换和操作系统线程切换有所不同，线程的切换由操作系统在内核态切换，同时操作系统支持时间片以及优先级调度，但是协程并不支持优先级调度。

2.如何实现协程的自动切换？说出你的想法。

答：可以使用任务队列维护协程和操作系统线程的映射关系，封装各种io函数，当这些封装的函数被程序使用的时候，内部使用操作系统的异步io函数，当异步io函数表示阻塞时，将当前协程放入任务队列，切换执行任务其他可执行的协程。

分析：这个问题是一个发散性问题，如果不了解协程实现可以通过类比的方式进行回答，言之有理即可。

3.Golang当中的协程是占用一个CPU还是会被调度到不同的CPU？

答：会被调度到不同的CPU。

分析：协程和线程是多对多的关系，一个协程的运行实际还是会绑定到一个操作系统线程，而线程的调度是操作系统决定的，任何一个线程在没有特殊限制的情况下可能会被调度到任何一个CPU上运行，而协程和线程是由runtime动态绑定的，所以协程也可能在任何一个CPU上运行。

4.协程和线程的区别？

特立独行MVP

2.相比创建一个线程而言，创建一个协程的开销非常小。

3.协程之间的切换开销相比线程间切换开销低很多。

4.协程比线程支持更大的并发量。

5.协程不支持抢占，因为操作系统有时间片概念，所以线程是支持优先级和抢占的，但是协程是在用户态实 现，协程是非抢占式调度。在任务调度上，协程是弱于线程的。

分析：本题属于概念性问题，回答全面即可。

5.协程适合哪些场景？

答：协程适合以下特点的场景

1.高并发场景：每秒钟需要处理成千上万的用户请求；

2.高网络IO场景：服务经常需要从其它机器获取数据，进行网络IO；

3.低计算场景：CPU密集型计算比较少。

分析：本题考查协程的应用场景。无论是网络IO，还是磁盘IO，远远慢于CPU的操作速度，所以往往需要CPU去等待IO操作完成。同步IO下系统需要切换线程，但是由于大量的线程切换带来了大量的性能的浪费，尤其是IO密集型的程序。而异步IO可以减少线程切换带来性能损失，但是编程模式并不符合人类思维。所以使用协程既可以解决线程切换开销大的问题，也可以符合人类编程的思维。

6.说一说你对于协程的理解？

答：由于现在系统的并发量越来越大，一开始的简单多线程模型已经不能很好的应对高并发场景，因为线程的切换开销已经成为了瓶颈。由于epoll的出现，IO多路复用的并发模式可以极大的提升系统的性能，但是这种IO多路复用的方式虽然提升的运行效率但是对于程序员而言这种编程模型非常难以理解，开发效率低且代码调试困难。此时协程就成了2者的一种折中方案，协程解决了操作系统线程切换和创建销毁开销大的问题，同时又可以以简单的编程模式开发，提升系统的效率的同时也没有降低太多编程难度。

分析：本题也是发散性问题，对一个概念的理解，我认为从其产生的原因和其能够解决的问题这两个角度进行回答是比较合适的。

总结

本节总结了协程的概念，结合我在面试字节跳动过程当中遇到的问题进行解答。

特立独行MVP

感谢阅读，如果文中有任何错误欢迎你给我留言指出，也欢迎分享给更多的朋友一起阅读。

举报

收藏 赞 1

相关专栏

《收割BAT：C++校招学习路线总结》

19篇文章 | 95订阅

已订阅

0条评论 默认排序

没有回复

请留下你的观点吧~

发布

专栏推荐

《收割BAT：C++校招学习路线总结》

《收割BAT：C++校招学习路线总结》，专栏共计17节。专栏分为五大主要内容，包括后台开发学习...

19篇文章 | 95阅读

特立独行MVP