# 面试官问：单核CPU支持Java多线程吗?

点击关注👉 Java精选 2022-10-02 12:00 发表于河北

收录于合集
#Java基础 69 #多线程 1

由于现在大多计算机都是多核CPU，多线程往往会比单线程更快，更能够提高并发，但提高并发并不意味着启动更多的线程来执行。更多的线程意味着线程创建销毁开销加大、上下文非常频繁，你的程序反而不能支持更高的TPS。

## 时间片

多任务系统往往需要同时执行多道作业。作业数往往大于机器的CPU数，然而一颗CPU同时只能执行一项任务，如何让用户感觉这些任务正在同时进行呢？操作系统的设计者 巧妙地利用了时间片轮转的方式

时间片是CPU分配给各个任务（线程）的时间!

思考：单核CPU为何也支持多线程呢?

线程上下文是指某一时间点 CPU 寄存器和程序计数器的内容，CPU通过时间片分配算法来循环执行任务（线程），因为时间片非常短，所以CPU通过不停地切换线程执行。

换言之，单CPU这么频繁，多核CPU一定程度上可以减少上下文切换。面试宝典：https://www.yoodb.com

## 超线程

现代CPU除了处理器核心之外还包括寄存器、L1L2缓存这些存储设备、浮点运算单元、整数运算单元等一些辅助运算设备以及内部总线等。一个多核的CPU也就是一个CPU上有多个处理器核心，就意味着程序的不同线程需要经常在CPU之间的外部总线上通信，同时还要处理不同CPU之间不同缓存导致数据不一致的问题。

超线程这个概念是Intel提出的，简单来说是在一个CPU上真正的并发两个线程，由于CPU都是分时的（如果两个线程A和B，A正在使用处理器核心，B正在使用缓存或者其他设备，那AB两个线程就可以并发执行，但是如果AB都在访问同一个设备，那就只能等前一个线程执行完后一个线程才能执行）。实现这种并发的原理是 在CPU里加了一个协调辅助核心，根据Intel提供的数据，这样一个设备会使得设备面积增大5%，但是性能提高15%~30%。

## 上下文切换

- 线程切换，同一进程中的两个线程之间的切换
- 进程切换，两个进程之间的切换
- 模式切换，在给定线程中，用户模式和内核模式的切换
- 地址空间切换，将虚拟内存切换到物理内存

CPU切换前把当前任务的状态保存下来，以便下次切换回这个任务时可以再次加载这个任务的状态，然后加载下一任务的状态并执行。任务的状态保存及再加载, 这段过程就叫做上下文切换。

每个线程都有一个程序计数器（记录要执行的下一条指令），一组寄存器（保存当前线程的工作变量），堆栈（记录执行历史，其中每一帧保存了一个已经调用但未返回的过程）。

寄存器 是 CPU 内部的数量较少但是速度很快的内存（与之对应的是 CPU 外部相对较慢的 RAM 主内存）。寄存器通过对常用值（通常是运算的中间值）的快速访问来提高计算机程序运行的速度。

程序计数器是一个专用的寄存器，用于表明指令序列中 CPU 正在执行的位置，存的值为正在执行的指令的位置或者下一个将要被执行的指令的位置。

- 挂起当前任务（线程/进程），将这个任务在 CPU 中的状态（上下文）存储于内存中的某处
- 恢复一个任务（线程/进程），在内存中检索下一个任务的上下文并将其在 CPU 的寄存器中恢复
- 跳转到程序计数器所指向的位置（即跳转到任务被中断时的代码行），以恢复该进程在程序中

**线程上下文切换会有什么问题呢?**

上下文切换会导致额外的开销，常常表现为高并发执行时速度会慢串行，因此减少上下文切换次数便可以提高多线程程序的运行效率。

- **直接消耗**：指的是CPU寄存器需要保存和加载, 系统调度器的代码需要执行, TLB实例需要重新加载, CPU 的pipeline需要刷掉

- **间接消耗**：指的是多核的cache之间得共享数据, 间接消耗对于程序的影响要看线程工作区操作数据的大小

## 切换查看

Linux系统下可以使用vmstat命令来查看上下文切换的次数， 其中cs列就是指上下文切换的数目（一般情况下, 空闲系统的上下文切换每秒大概在1500以下）

## 线程调度

### 抢占式调度

指的是每条线程执行的时间、线程的切换都由系统控制，系统控制指的是在系统某种运行机制下，可能每条线程都分同样的执行时间片，也可能是某些线程执行的时间片较长，甚至某些线程得不到执行的时间片。在这种机制下，一个线程的堵塞不会导致整个进程堵塞。

java使用的线程调使用抢占式调度，Java中线程会按优先级分配CPU时间片运行，且优先级越高越优先执行，但优先级高并不代表能独自占用执行时间片，可能是优先级高得到越多的执行时间片，反之，优先级低的分到的执行时间少但不会分配不到执行时间。

### 协同式调度

指某一线程执行完后主动通知系统切换到另一线程上执行，这种模式就像接力赛一样，一个人跑完自己的路程就把接力棒交接给下一个人，下个人继续往下跑。线程的执行时间由线程本身控制，线程切换可以预知，不存在多线程同步问题，但它有一个致命弱点：如果一个线程编写有问题，运行到一半就一直堵塞，那么可能导致整个系统崩溃。

**线程让出cpu的情况**

- 当前运行线程主动放弃CPU，JVM暂时放弃CPU操作（基于时间片轮转调度的JVM操作系统不会让线程永久放弃CPU，或者说放弃本次时间片的执行权），例如调用 `yield()` 方法。
- 当前运行线程因为某些原因进入阻塞状态，例如阻塞在I/O上
- 当前运行线程结束，即运行完 `run()` 方法里面的任务

## 引起线程上下文切换的因素

- 当前执行任务（线程）的时间片用完之后，系统CPU正常调度下一个任务
- 中断处理，在中断处理中，其他程序"打断"了当前正在运行的程序。当CPU接收到中断请求时，会在正在运行的程序和发起中断请求的程序之间进行一次上下文切换。中断分为硬件中断和软件中断，软件中断包括因为IO阻塞、未抢到资源或者用户代码等原因，线程被挂起。
- 用户态切换，对于一些操作系统，当进行用户态切换时也会进行一次上下文切换，虽然这不是必须的。

- 多个任务抢占锁资源，在多任务处理中，CPU会在不同程序之间来回切换，每个程序都有相应的处理时间片，CPU在两个时间片的间隔中进行上下文切换

**因此优化手段有：**

- 无锁并发编程，多线程处理数据时，可以用一些办法来避免使用锁，如将数据的ID按照Hash取模分段，不同的线程处理不同段的数据
- CAS算法，Java的Atomic包使用CAS算法来更新数据，而不需要加锁
- 使用最少线程
- 协程，单线程里实现多任务的调度，并在单线程里维持多个任务间的切换

合理设置线程数目既可以最大化利用CPU，又可以减少线程切换的开销。

- 高并发，低耗时的情况，建议少线程。
- 低并发，高耗时的情况：建议多线程。
- 高并发高耗时，要分析任务类型、增加排队、加大线程数

> 作者：布道
> https://blog.csdn.net/alex_xfboy/article/details/90722654

公众号"Java精选"所发表内容注明来源的，版权归原出处所有（无法查证版权的或者未注明出处的均来自网络，系转载，转载的目的在于传递更多信息，版权属于原作者。如有侵权，请联系，笔者会第一时间删除处理！

最近有很多人问，有没有**读者**交流群！加入方式很简单，公众号**Java精选**，回复"**加群**"，即可入群！

**Java精选面试题**（微信小程序）：**3000+**道面试题，包含Java基础、并发、JVM、线程、MQ系列、Redis、Spring系列、Elasticsearch、Docker、K8s、Flink、Spark、架构设计等，在线随时刷题！

------ 特别推荐 ------

特别推荐：专注分享最前沿的技术与资讯，为弯道超车做好准备及各种开源项目与高效率软件的公众号，**「大咖笔记」**，专注挖掘好东西，非常值得大家关注。**点击下方公众号卡片关注**。

大咖笔记
关注最前沿的技术与资讯，为弯道超车做好准备！
2篇原创内容

公众号

点击“阅读原文”，了解更多精彩内容！文章有帮助的话，点在看，转发吧！

收录于合集 #Java基础 69

上一篇
Java 19 已至，虚拟线程 = 王炸！！

下一篇
别再写 main 方法测试了，太 Low！这才是专业 Java 测试方法！

阅读原文

喜欢此内容的人还喜欢

Python岗面试官最喜欢问的问题，没有之一！【#147】
麦叔编程


面试官让我重构 Kafka，懵了……
labuladong


微软有史以来最大的软件产品：超36斤的C/C++编译器
码小辫