操作系统『五』：CPU 上下文切换

原文链接

操作系统『五』：CPU 上下文切换

发表于2021-04-16更新于2021-04-17阅读次数：988
本文字数：3.9k阅读时长 ≈4 分钟

进程 CPU 调度和资源分配的单位，是一个程序在一个数据集上的一次执行；线程是CPU上运行和调度的基本执行单位，进程内的一个代码片段可创建为线程，线程由进程创建，寄生在进程身上。

CPU 从一个**任务**切换到另一个**任务**，俗称上下文切换。理论上而言，进程负载了程序运行所有内容，开销较大。线程较轻量，和父进程共享各类资源，上下文切换的成本较低。可实际情况中具体如何？今天一探究竟。

多任务系统与上下文切换

我们都知道，现代操作系统是一个多任务操作系统，它支持远大于 CPU 数量的任务同时运行。当然，这些任务实际上并不是真的在同时运行，而是因为系统在很短的时间内，将 CPU 轮流分配给它们，每秒可能发生几百次上下文切换，造成多任务同时运行的错觉。

其中上下文的意思是指：某一时刻，寄存器内的数据和程序计数器的值；寄存器保留了CPU的指令，程序计数器表明了指令所在的地址，即当前指令之后立即执行的指令的地址。它们都是 CPU 在运行任何任务前，必须的依赖环境。

而在每个任务运行前，CPU 都需要知道任务从哪里加载、又从哪里开始运行，也就是说，需要系统事先帮它设置好 CPU 寄存器和程序计数器，也就是上下文。涉及到具体的切换，就是先把前一个任务的 CPU 上下文（也就是 CPU 寄存器和程序计数器）保存起来，然后加载新任务的上下文到这些寄存器和程序计数器，最后再跳转到程序计数器所指的位置，运行新任务。而这些保存下来的上下文，会存储在系统内核中，并在任务重新调度执行时再次加载进来。这样就能保证任务原来的状态不受影响，让任务看起来还是连续运行。

从上面的描述中可以知道，CPU 上下文切换无非就是更新了一些寄存器的值嘛，但这些寄存器，本身就是为了快速运行任务而设计的，为什么会影响系统的性能呢？在回答这个问题前，先得搞清楚操作系统管理的这些『任务』到底是什么呢？

进程和线程正是最常见的任务。但是除此之外，硬件通过触发信号，会导致中断处理程序的调用，也是一种常见的任务。所以，根据任务的不同，CPU上下文切换可分为几种场景：

- 进程上下文切换
- 线程上下文切换
- 中断上下文切换
- 系统调用上下文切换

系统调用

现代操作系统按照特权等级，把进程的运行空间分为内核空间和用户空间，内核模式运行特权指令，用户程序在用户模式执行，主要是为了保护系统不受一些故障程序的影响。进程既可以在用户空间运行，又可以在内核空间中运行。进程在用户空间运行时，被称为进程的用户态，而陷入内核空间的时候，被称为进程的内核态。

因为特权指令只能在内核模式下运行，所以无法在用户模式运行特权指令。只能在用户模式下发出特权指令的系统调用，此时用户程序暂停，转换为内核模式执行系统调用，得到结果反馈给用户，在转换为用户模式，进程继续执行。具体流程为：

1. 保存 CPU 寄存器里原来用户态的指令位
2. 为了执行内核态代码，CPU 寄存器需要更新为内核态指令的新位置。
3. 跳转到内核态运行内核任务。
4. 当系统调用结束后，CPU 寄存器需要恢复原来保存的用户态，然后再切换到用户空间，继续运行进程。

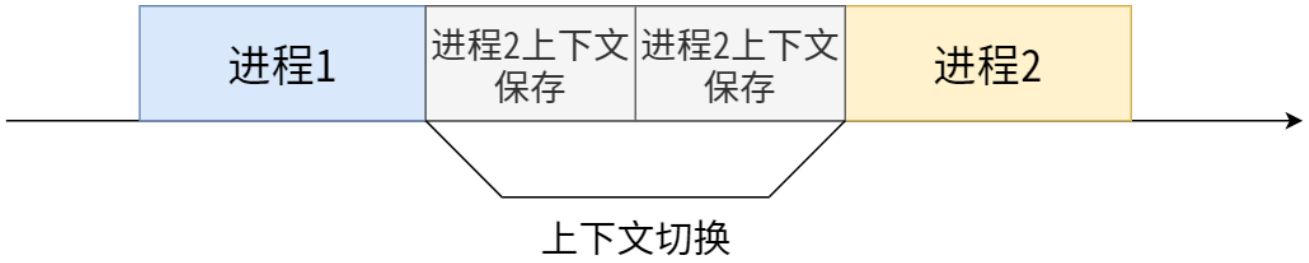
所以，一次系统调用的过程，其实是发生了两次 CPU 上下文切换。（用户态`->`内核态`->`用户态）

不过，需要注意的是，系统调用过程中，并不会涉及到虚拟内存等进程用户态的资源，也不会切换进程。这跟我们通常所说的进程上下文切换是不一样的：进程上下文切换，是指从一个进程切换到另一个进程运行；而系统调用过程中一直是同一个进程在运行。所以，系统调用过程通常称为特权模式切换，而不是上下文切换，系统调用属于同进程内的 CPU 上下文切换。

进程上下文切换与系统调用

首先，进程是由内核来管理和调度的，进程的切换只能发生在内核态。所以，进程的上下文不仅包括了内核堆栈、寄存器等内核空间的状态，还包括了虚拟内存、栈、全局变量等用户空间的资源。

因此，进程的上下文切换就比系统调用时多了一步：在保存内核态资源（当前进程的内核状态和 CPU 寄存器）之前，需要先把该进程的用户态资源（虚拟内存、栈等）保存下来；而加载了下一进程的内核态后，还需要刷新进程的虚拟内存和用户栈。如下图所示，保存上下文和恢复上下文的过程开销是很大的，此外，考虑到进程之间的相互的保护，只有操作系统能操作其它进程和了解其它进程的状态，因此这个过程需要在内核态完成。



另外，我们知道，现代操作系统通过 TLB（Translation Lookaside Buffer）来管理虚拟内存到物理内存的映射关系。如果发生运行时动态链接（进入系统态）、内存紧缩等情况，即虚拟内存更新后，TLB 也需要刷新，内存的访问也会随之变慢，不过这是进程上下文切换带来的副作用了。

发生进程上下文切换的场景

1. 基于时间片的操作系统，CPU 时间被划分为一段段的时间片，这些时间片再被轮流分配给各个进程。这样，当某个进程的时间片耗尽了，就会被系统挂起，切换到其它正在等

待 CPU 的进程运行。
2. 进程在系统资源不足（比如内存不足）时，要等到资源满足后才可以运行，这个时候进程也会被挂起，并由系统调度其他进程运行。
3. 当进程通过睡眠函数 sleep 这样的方法将自己主动挂起时，自然也会重新调度。
4. 当有优先级更高的进程运行时，为了保证高优先级进程的运行，当前进程会被挂起，由高优先级进程来运行
5. 发生硬件中断时，CPU 上的进程会被中断挂起，转而执行内核中的中断服务程序。

线程上下文切换

线程与进程最大的区别在于：线程是调度的基本单位，而进程则是资源拥有的基本单位。说白了，所谓内核中的任务调度，实际上的调度对象是线程；而进程只是给线程提供了虚拟内存、全局变量等资源。线程也有自己的私有数据，比如栈和寄存器等，这些在上下文切换时也是需要保存的。

发生线程上下文切换的场景：

- 前后两个线程属于不同进程。此时，因为资源不共享，所以切换过程就跟进程上下文切换是一样。切换线程，CPU 的各种寄存器都要重新刷一遍，从这个角度而言，你可以把进程和线程当作一种东西，只是共享度不同，其他没区别的。
- 前后两个线程属于同一个进程。此时，因为虚拟内存是共享的，所以在切换时，虚拟内存这些资源就保持不动，内存命中率会高一些，只需要切换线程的私有数据、寄存器等不共享的数据。

虽然同为上下文切换，但同进程内的线程切换，要比多进程间的切换消耗更少的资源，这也正是多线程代替多进程的一个优势。

中断上下文切换

中断处理程序运行于内核态。中断发生时CPU可能处于内核态（如执行系统调用的过程中）也可能处于用户态（执行应用空间代码）。所以前者不涉及特权级转换，后者涉及。

不涉及特权级转换的情况：

- 压入寄存器现场、错误代码等
- 执行中断处理程序
- 恢复寄存器现场

可以看到这里并没有发生堆栈的切换——因为本来就运行在内核栈上嘛！中断处理程序借用了应用程序的内核栈。说『借用』是因为进程的内核栈是给进程执行内核空间代码使用的（通常就是系统调用），由于中断并不一定和正在运行的进程有什么关联。

但是对于后者，也就是用户态中被中断，有一个用户 `->` 内核 `->` 用户的切换过程，伴随着相关栈的切换。具体过程：

- 找到内核栈
- 压入寄存器现场、错误代码
- 转入中断处理程序
- 恢复第二步保存的现场
- 切换换回用户栈

为了快速响应硬件的事件，中断处理会打断进程的正常调度和执行，转而调用中断处理程序，响应设备事件，如断电和设备损坏等。而在打断其他进程时，就需要将进程当前的状态保存下来，这样在中断结束后，进程仍然可以从原来的状态恢复运行。

中断上下文，其实只包括内核态中断服务程序执行所必需的状态，包括 CPU 寄存器、内核堆栈、硬件中断参数等。跟进程上下文不同，中断上下文切换的东西不需要涉及到用户态。所以，即便中断过程打断了一个正处在用户态的进程，也不需要保存和恢复这个进程的虚拟内存、全局变量等用户态资源；只需在内核态记录：当前进程的寄存器和程序计数器。如喜闻乐见的，硬盘传输文件时突然拔掉硬盘，传输文件进程的虚拟内存、全局变量等用户态资源都没有被保存，相当于白传输了。~~例子有点生硬。~~

由于中断会打断正常进程的调度和执行，所以大部分中断处理程序都短小精悍，以便尽可能快的执行结束。另外，跟进程上下文切换一样，中断上下文切换也需要消耗 CPU，切换次数过多也会耗费大量的 CPU，甚至严重降低系统的整体性能。所以，当你发现中断次数过多时，就需要注意去排查它是否会给你的系统带来严重的性能问题。

一些讨论

- CPU 上下文切换，是保证现代系统正常工作的核心功能之一，一般情况下不需要我们特别关注。根据Tsuna的测试报告，每次上下文切换都需要几十纳秒到数微秒的 CPU 时间。这个时间还是相当可观的，特别是在进程上下文切换次数较多的情况下，会把 CPU 时间消耗在寄存器、内核栈以及虚拟内存等数据的保存和恢复上，从而缩短进程真正运行的时间，导致系统的整体性能大幅下降。
- CPU 执行的最小逻辑单元是线程，并不是一个 CPU 核心。所以切换进程，只是切换一个进程里的一个线程到另一个进程里的一个线程，说白了还是线程切换。如果对比的是单个进程和进程内线程的切换，线程共享资源，cache命中率会高很多；进程切换，不共享资源，cache命中率低。
- 上下文切换说的是 CPU 寄存器的切换，跟 cache 和内存没啥关系，后者只是上下文切换带来的副作用。也许你会有疑问，当切换到新的进程，进程所需的资源不在内存里，这不就开始调度了，调度的开销呢？但实际是，新建进程不会分配 CPU，进程就绪后等等待分配处理器，此时资源已经进内存了。
- 切换进程可能要刷 TLB，进程内线程切换不需要。进程切换要切页表，所以可能同时要刷 TLB，这个根据实现定。

参考

1. 本文大量参考，CPU切换
2. 什么是上下文
3. 中断上下文