****

**操作系统原理**

**CFS调度算法分析**

|  |  |
| --- | --- |
| **学生姓名:** | **熊术文** |
| **学生学号：** | **2017051107** |
| **班级：** | **计应173** |
| **指导教师：** | **张欢** |
| **分数：** |  |

2019年12月04日

成都信息工程大学计算机学院

CFS调度算法分析

**摘 要：**一个操作系统的核心是进程调度。那么操作系统最重要的程序之一则是进程调度程序，同时进程调度程序也是多任务操作系统执行频度最高的一部分，操作系统的整体性能也决定于进程调度程序的性能。本文剖析从O(1)算法到CFS算法的演变。最后用测试工具对CFS的稳定性和计算速度进行了分析。

**关键词：**Linux调度CFS调度器红黑树

1 引言

操作系统内核调度算法历来是人们改进系统性能的研究热点。作为主流操作系统之一的linux，它的调度算法几经改进，表现出优异的性能，在越来越多的邻域逐渐占据重要地位。纵观linux调度器的发展，大致经历了三个阶段：最早期的0.11内核中的O(N)调度算法，并一直到2.4内核都没有大的改变；随后在2.6内核中发布了由IngoMolnar设计并实现的O(1)调度器，该调度器与过去的调度相比获得了长足的进步；它采用了与以往调度器完全不同的设计理念,具有革命性的意义。其调度复杂度依然为O(1),并使得进程更加公平的共享处理器资源,这也是CFS调度器最终被采纳的主要原因。

2 linux进程调度的概述

所谓进程调度就是指操作系统正确的匹配CPU时间之后来准确的执行等待中的进程。怎样从若干个可运行的进程里面找到其中最优先的进程执行的同时又能保证响应时间短、吞吐量高是进程调度的核心所在。进程调度有何时启动调度器和调度器执行什么调度算法两部分。进程调度的要求就是吞吐量大、Ⅱ向应时间快、周转时间短以及效率高。由进程的Ⅱ向应时间Linux内核可以把进程分为三类：实时进程、批处理进程和交互进程。根据这三类进程内核又产生了三种不同的调度方法：先进先出策略(SCHED\_FIFD)、轮转策略(SCHED\_RR)和适合交互分时的程序(SCHED\_OTHER)。

3 CFS调度器

3.1 CFS调度器概述

CFS调度器由IngoMolnar提出，采用了完全公平的思想对待所有任务。但是“公平”不等于同等对待所有任务，而是指相对长程的、统计上的公平。在每个小的时间区间很可能看起来并不公平，原因可能是需要对以往的不公平作出补偿、系统的负载变化等。

CFS还增加了一个调度器模块管理器，各种不同的调度算法可以作为一个模块注册到该管理器中。不同的任务可以选择使用不同的调度器模块，例如实时任务可以选择实时任务调度模块。Linux2.6.24内核中，CFS实现了两个调度算法，CFS算法模块和实时调度模块。对应实时任务，将使用实时调度模块；对应普通任务则使用CFS算法。

CFS的算法和实现都更简单，现有的测试表明其性能也比0(1)更好。相关数据表明CFSrc6-devel的性能比0(1)提高3.1％；二进制文件的尺寸SMP版CFS比0(1)小8.5％。

3.2 算法的主要思想

CFS的总体设计可以用一句话来总结：在真实的硬件上模拟“理想的多任务处理器”，使每个进程都尽可能公平的获得CPU。为此，CFS引入了一个新概念“virtualruntime”，它描述了进程在CPU上的执行时问。在调度的过程中，CFS为了使每个进程都获得相近的执行时间，总是选取vruntime最小，也就是执行时间最短的进程来运行，以达到各个任务执行时间的平衡。这就是CFS的核心思想，即每个进程都被公平对待。

CFS调度算法相比于0(1)算法有了很大的变化：不再跟踪进程的睡眠时间、区分交互式进程，因此代码中没有那么多难以理解的经验性公式，思路清晰简单；新版本中增加了组调度功能，实现了对用户和组的公平性；引入了调度类sched\_class和调度实体sched\_entity的概念——调度类使不同的进程选择不同的调度模块，调度实体则实现了组调度的功能；不再使用优先级数组，它将所有就绪态的进程都插入红黑树，用红黑树来选择下一个被调度的进程。

图1描述了进程的调度模型。类似于以往的调度器，它的主要工作仍是在就绪态的进程队列中选择最合适的进程来运行，不同的是，新内核中有了调度类的概念，将不同的进程放入不同的调度模块中执行，例如：普通进程进入CFS调度模块，实时进程进入实时调度模块。因此，每个调度模块都要执行调度类为它指定的一组相应的函数。这样做的好处是，当需要修改相应进程的调度算法时，并不需要修改整个scheduler()函数，只需要修改对应的函数。

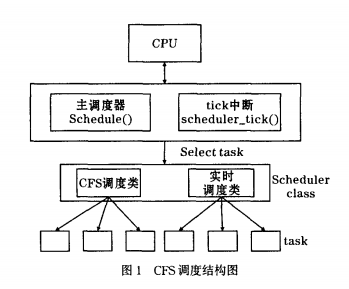


图1 调度结构图

3.3 红黑树

与RSDL和SD算法相比，CFS不再使用优先级队列，而是为每个CPU使用一个按时间排序的红黑树结构，所有可运行进程都被插入到红黑树中。红黑树是一种自平衡二叉搜索树，对红黑树的操作时问复杂度为O(1ogN)。对于每个可运行进程，在红黑树上都有一个节点。红黑树上位于最左边的节点表示下一次将要被调度的进程。红黑树如图2：



图2 红黑树

红黑树的键值是由三个因子计算而得：一是进程已经占用的CPU时间；二是当前进程的nice值；三是当前CPU的负载。很大程度上可以简单地认为等于当前的虚拟时钟减去进程的等待时间，当进程等待CPU时，进程的等待时间开始增加。当进程获得CPU时，进程的等待时间开始减少，有可能是负值。

3.4CFS算法在Linux上的具体实现

3.4.1重要数据结构

重要数据结构伴随着CFS算法的加入，随之而来的是添加了一些新的数据结构，去掉了一些O(1)算法残留的数据结构，例如优先级队列和指向这些队列的指针。CFS算法核心数据结构关系图如图3：



图3 核心数据结构关系图

下面根据各个结构关系，从上到下分析主要数据结构：

task\_struct：进程描述符，描述了进程的当前状态、堆栈、进程标识、优先级等。在CFS算法中新增加了sched\_class和sched\_entity。

sched\_class：调度类结构是CFS调度算法的核心结构。它采用了面向对象的方法，提供了一系列调度过程所必需使用的接口，将调度模块封装并链接起来。在具体的调度过程中，经常使用这些接口。其中，enqueue\_task()是把一个进程加入运行队列，同时作为调度实体插入到红黑树中。dequeue\_task()是将进程从就绪队列中移除，同时将对应的调度实体从红黑树删除。pick\_next\_task()是挑选下一个要运行的最佳进程。put\_prev\_task()是让当前运行的进程停止运行，返回到运行队列。

sched\_entity：调度实体，每一个进程都作为一个调度实体，它包含进程用于调度的一些相关信息。其中，wait\_runtime是进程等待CPU的时间，fair\_key是进程插入红黑树的键值。

cfs\_rq：对于每一个运行队列，都有一个与红黑树相关联的数据结构，用于描述运行队列。其中，fair\_clock是该队列虚拟时钟，exec\_clock是该队列的实际时钟，wait\_runtime是该队列等待CPU的时间，rb\_leftmost是该运行队列对应的红黑树最左边的节点，也是下一个运行的最佳进程。

rb\_node：红黑树上的节点，不包括根节点和叶子节点，因为根节点结构为rb\_root，而叶子节点不含任何信息。

3.4.2进程调度过程

进程调度时机主要发生在中断时。当时钟中断发生时，系统会调用函数scheduler\_tick()，更新相应的调度信息。

如果内核检查到需要调度，就调用schedule()函数。schedule()函数会首先定义一些局部变量，包括prev指向当前进程，也就是即将被切换出CPU的进程；next指向下一个要运行的进程，switch\_count表示进程切换的次数，rq是运行队列指针，cpu指本地CPU。

接着，函数检查当前进程的状态，如果不是可运行状态而且没有在内核态抢占，就从运行队列中删除当前进程。如果它是非阻塞挂起信号，而且状态为TASK\_INTERRUPTIBLE，函数就把该进程的状态设置为TASK\_RUNNING，并把它插入到运行队列。

如果运行队列中没有可运行的进程存在，函数就调用idle\_balance()函数进行负载平衡，它会从另外一个运行队列迁移一些可运行进程到本地运行队列。

然后，函数调用调度类中的函数put\_prev\_task\_fair()停止当前运行进程，调用pick\_next\_task\_fair()选择下一个新进程，即从红黑树中选取其最左边的节点。

最后，如果选择的下一个进程不是当前进程，就调用context\_switch()进行上下文切换。否则直接运行新选择的进程。

4 总结：

以上的讨论看出CFS对以前的调度器进行了很大改动。以完全公平为核心思想，通过追踪进程的执行时间来调度任务；使用红黑树代替优先级数组来选择下一个被调度的进程；引入调度类，显著增强了内核调度程序的可扩展性和代码的可维护性；代码中不再有难以理解的经验性公式，思路清晰简单、结构灵活、算法适应性更高。

当然，任何调度算法都还无法满足所有的应用需要，CFS也有一些负面的测试报告，由于红黑树的查找执行时问为O(IgN)，当调度任务大幅增加时，性能会有所下降，但CFS在总体性能上还是比O(1)调度算法有了显著的提高。随着Linux内核的不断发展，Linux调度算法会进一步完善，新的调度算法更令人期待。

**参考文献：**

[1] 张桂兰，王飞超. Linux内核完全公平调度器的分析及模拟. 《中国科技信息》 2009年第4期 134-135,137,共3页

[2] 简岩. Linux CFS调度算法分析. 《中小企业管理与科技》2014年第18期 236-237页

[3] 朱旭，杨斌，刘海涛. 完全公平调度算法分析. 《成都信息工程学院学报》2010年第1期 18-21,共4页

[4] 杜慧江，王云光. Linux内核2．6．24的CFS调度器分析. 《计算机应用与软件》 2010年第2期 166-168,共3页

[5] 田振. Linux2.6进程调度算法分析与仿真研究[D].中南大学,2009.

[6] 李建萍,陆建德.基于公平策略的Linux2.6调度算法与应用分析.微计算机信息,2010(33):176-178.

[7] 刘婷,王华军,王光辉.基于Linux内核的CFS调度算法研究.电脑与电信,2010(4):61-63.