基于 Linux 内核的 CFS 调度算法研究

刘 婷 王华军 王光辉
(成都理工大学信息工程学院，四川 成都 610059)

[摘 要] 通过分析 Linux 2.4 和 2.6 内核调度算法的缺点，介绍了基于公平思想的 CFS 调度算法的优点。深入分析了 CFS 算法的思想和核心结构，并研究了 CFS 算法的调度过程、具体实现的数据结构和细节。

[关键词] 调度算法；Linux 内核；CFS；红黑树

1. 引言

随着 Linux 内核的不断发展，Linux 已经被许多人视为是未来最有前途的操作系统之一。进程调度是 Linux 操作系统的核心功能，在 Linux 的发展史上起着非常重要的作用，不同的算法对系统性能的影响也是不一样的。2.4 内核的 O(n)算法和早期 2.6 内核的 O(1)算法都是以往调度算法中的经典算法，但是存在着一些缺点。2.6.23 内核发布的 CFS 算法将公平的理念用到调度算法中，具有划时代的意义。本文将在研究以往调度算法优缺点的基础上重点介绍 CFS 调度算法。

2. Linux 2.4 与 Linux 2.6 调度算法简介

2.1 Linux 2.4 内核调度算法

2.4 内核采用的调度算法是基于优先级的设计。它在每次进程切换时，内核扫描可运行进程的链表，计算进程的优先级，然后选择“最佳”进程来运行[1]。

通过对 2.4 调度算法研究，发现存在以下几个缺点：

(1) 开销太大：选择最佳进程所要消耗的时间与可运行的进程数量相关，因此达不到实时性的要求。

(2) 内核不支持抢占：当进程处于内核态时不会发生抢占，这对真正的实时应用是不能接受的。

(3) SMP 效率：在 2.4 内核中，就绪进程队列是一个全局数据结构，调度算法对它的所有操作都会因全局自旋锁而导致系统各个处理机之间的等待，这将导致大部分的 CPU 处于空闲状态，从而影响 SMP 效率[2]。

2.2 Linux 2.6 内核调度算法

针对 2.4 内核的缺点，Ingo Molnar 设计并实现了早期 2.6 内核的调度算法，它的算法复杂度为 O(1)，所以又叫 O(1)算法。它继承和发扬了 2.4 内核调度算法的特点，极大地改善了 O(n)调度算法的缺陷。首先，O(1)调度算法所花费的时间为常数，与当前系统中的进程个数无关。其次，它支持内核态抢占，更好地支持了实时进程。最后，增强了 SMP 的可扩展性，每个处理器拥有独立的运行队列和自旋锁，不会产生互斥[3]，极大地提高了 SMP 效率。

在 O(1)算法中，每个 CPU 都有一个运行队列，每个运行队列是一个二元数组 arrays，按时间片是否用完分为两个队列和两个队列指针[4]，指针 active 指向时间片没用完、当前可被调度的就绪进程队列，指针 expired 指向时间片已用完的就绪进程队列。当 active 队列中的一个进程用完自己的时间片之后，它就被移动到 expired 队列中。在移动过程中，调度算法会对其时间片重新进行计算。当 active 队列中没有就绪进程时，active 和 expired 两个指针将执行互换操作。

相比 O(n)算法，O(1)算法在交互性方面有很大的改进，它实现了基于进程过去行为的启发式算法，以确定进程应该被当作交互式进程还是批处理进程。但是，它对于交互式应用仍然存在响应性差的问题，对 NUMA 支持也不完善。为了解决这些问题，大量代码被加入调度算法中，虽然很多性能问题得到了解决，但是代码很复杂且难以维护和阅读。

3. CFS 算法

在 Linux 2.6.23 内核中发布了一种新的调度算法——CFS 算法，CFS 是“Completely Fair Scheduler”的缩写，它主要是改进早期 2.6 内核调度算法的代码复杂度的问题，使调度算法变得更加简单有效。它的出现彻底改变了内核看待进程调度的方式，不再依赖划分时间片和增加抢占点。

O(1)算法的主要复杂性来自动态优先级的计算，调度算法根据平均睡眠时间和一些很难理解的经验公式来修正进程的优先级以区分交互式进程和批处理进程。这样的代码很难阅读和维护。相比 O(1)算法，CFS 算法思路简单，抛弃了时间片和复杂的算法，它总是试图按照对 CPU 时间的最大需求运行进程，这有助于确保每个进程可以获得对 CPU 的公平共享。

3.1 CFS 算法基本理论

通过研究 Linux 2.6.23 内核发现，CFS 算法从 RSDL/SD 中吸取了完全公平的思想，不再跟踪进程的睡眠时间，也不再企图区分交互式进程。它将所有的进程都统一对待，这就是公平的含义。但是所谓的公平并不是绝对公平，而是相对公平。在以往的调度算法中，当给一个进程分配 40ms 的 CPU 计算时间时，它会一直占有 CPU，其它进程必须等待，这就产生不公平。但是对于 CFS 算法，它会将 40ms 的 CPU 计算时间根据优先级比例分配给不同的进程，使每个进程都能相对公平地获得 CPU 的执行时间。

3.1.1 CFS 算法主要思想

CFS 算法的核心思想是围绕着虚拟时钟展开的。虚拟时钟区别于硬件的实际时钟，虚拟时钟可以根据权重调节步调，

作者简介：刘婷，女，安徽蚌埠人，硕士研究生。

按照不同的步调前进。而实际时钟是按着固定的步调前进，不能改变。每个处理器不仅维护实际的时钟，还在对应的运行队列维护着一个虚拟时钟，它的前进步伐与该处理器上的进程总权重成反比。总权重越大，虚拟时钟前进的步伐越慢。同时，每个进程也有自己的虚拟时钟，它记录着进程已经占用 CPU 的虚拟时间。它的前进步伐与进程的权重成反比。进程的权重对应进程的优先级，对于不同优先级的进程，一般来说，优先级低的进程其虚拟时钟的步调比较快，而优先级高的步调比较慢。CFS 调度算法会选择最落后于运行队列虚拟时钟的进程来执行。如果某个进程的虚拟时钟快于运行队列的虚拟时钟，则在将来的调度过程受到“惩罚”，相反，如果比运行队列虚拟时钟慢的进程，将在调度过程受到“奖励”[5]。

3.1.2 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，对红黑树的操作时间复杂度为 O(log n)。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 1：

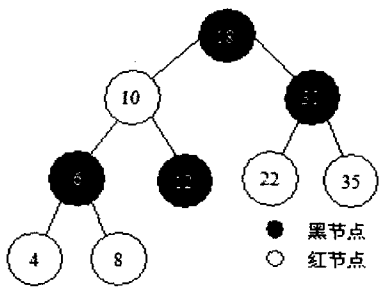

图 1 红黑树

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[6]。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。

3.2 CFS 算法在 Linux 上的具体实现

3.2.1 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 2：

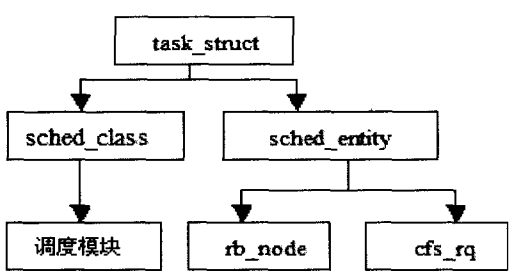

图 2 核心数据结构关系图

下面根据各个结构关系，从上到下分析主要数据结构：

(1) 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。

(2) 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。

(3) 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。

(4) 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。

(5) 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。

3.2.2 进程调度过程

2.6.23 内核进程调度时机和以往的内核调度算法一样，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. 结束语

调度算法是内核中的一个重要的组成部分，它在一定程

度上决定了内核的性能。调度算法的优劣直接反应出内核的好坏。纵观调度算法的发展，不难发现，它总是朝着代码简单、高效、交互性和实时性好的方向发展，而在这方面起决定性作用的就是算法的数据结构，从简单的全局队列到优先级队列再到本文提到的红黑树，不断将调度算法的性能进行改进。CFS 算法是 Linux 的最新的调度算法，它大大改善了 Linux 调度算法之前存在的问题，使代码更另简单，进程更加公平高效。

参考文献：

[1] 陈莉君，张琼声等. 深入理解 LINUX 内核(第三版)[M]. 北京：中国电力出版社，2007.

[2] 何克右，周彩贞. Linux 2.6 进程调度机制的剖析[J]. 华中师范大学学报(自然科学版)，2007，41(4)：520.

[3] 张永选，姚远耀. Linux 2.6 内核 O(1)调度算法剖析[J]. 韶关学院学报(自然科学版)，2009，30(6)：5.

[4] 王刚，余兆等. Linux 2.6 内核进程调度分析[J]. 计算机应用与软件，2007，24(9)：163.

[5] 李云华编著. 独辟蹊径品内核：Linux 内核源代码导读[M]. 北京：电子工业出版社，2009.

[6] 张桂兰，王飞超. Linux 内核完全公平调度器的分析和模拟[J]. 中国科技信息，2009，(4)：134.

[7] Linux 2.6.23 内核源代码[EB/OL]. http://www. kernel. org.

Research of CFS Scheduling Algorithm Based on Linux Kernel

Liu Ting Wang Huajun Wang Guanghui

(Chengdu University of Technology, Chengdu 610059, Sichuan)

【**Abstract**】 The paper introduces CFS scheduling algorithm based on the fair thought by analyzing the disadvantage of Linux 2.4 and 2.6 kernel scheduling algorithm. The paper analyses the idea and core structure of CFS algorithm in depth, and researches the scheduling process, concrete realization of the data structure and details of CFS algorithm.

【**Keywords**】 scheduling algorithm; Linux kernel; CFS; red-black tree

(上接第 60 页)

出现个别不一致的结果，我们可以单独找出，分析其在某项指标上是否存在异常现象。另外，聚类的结果与原始分数的顺序基本一致，这说明通过聚类分析的方法将教师划分成不同的类别是可行的。

5. 结论

通过以上分析，我们可以得出如下结论：

对全体教师评价数据进行快速聚类以及对不同院系教师的评价数据进行层次聚类是可行的；在此基础上给出不同类教师单项指标平均得分的分布图，对于学校从整体上了解院系情况、教师的教学情况和教师的教学水平等具有一定的参考价值；同时，针对不同院系和不同类教师给出相应的建议对于教师有针对性地提高自己的能力有重要的作用。

基于表 1 形式的数据，可向不同人员提供相应的信息。如对于学校和院系的管理人员而言，可以给出整个表的信息，也可基于此表作进一步的统计分析；而对于教师个人而言，可以提供其所属类别及每个类别中包含的教师人数等信息，这样教师对自己的差距有一个较详细的了解。

参考文献：

[1] 杨红等. 基于网络的教师教学评价系统[J]. 教育信息化，2005，4：51.

[2] 苏金明. 统计软件 SPSS12.0 for Windows 应用及开发指南[M]. 北京：电子工业出版社，2004.

[3] 黄润龙. 数据统计与分析技术：SPSS 软件实用教程[M]. 北京：高等教育出版社，2004.

[4] 杨晓明. SPSS 在教育统计中的应用[M]. 北京：高等教育出版社，2004.

Clustering Analysis Technology in the Application of Teaching Evaluation

Yu Chengmin Zhang Huaiwei Ling Haiyun

(Liaocheng University, Liaocheng 252059, Shandong)

【**Abstract**】 We analyze the teaching evaluation data by the clustering analysis technology of SPSS. The results show that it is feasible and very necessary that all the teachers are divided into the appropriate categories according to the evaluation data and then the general characteristics of each category are analyzed by clustering. It can not only play a diagnostic role in teaching evaluation, but also be conducive to promoting the development of teachers.

【**Keywords**】 clustering analysis; teaching evaluation; SPSS