

# 第八讲 多处理器调度

## 第五节 **Linux/FreeBSD BFS** 调度

-- From [Analysis of the BFS Scheduler in FreeBSD](#)

向勇 陈渝 李国良 任炬

2023年春季

## 提纲

### **1. BFS调度器**

### 2. BFS 与 CFS 的性能对比 (2012)

# BFS 的思路

BFS全称： Brain Fuck Scheduler，脑残调度器

- BFS 调度算法是一种时间片轮转算法的变种。
- 在多处理机时使用单就绪队列（双向链表）
  - 增加了队列互斥访问的开销
  - 减少了负载均衡算法开销

# BFS 的进程优先级

- 进程有 103 个优先级
  - 100 个静态的实时优先级;
  - 3 个普通优先级
    - SCHED\_ISO (isochronous) : 交互式任务
    - SCHED\_NORMAL : 普通任务
    - SCHED\_IDLEPRIO : 低优先级任务

# BFS 的就绪队列

- 就绪队列
  - 所有 CPU 共享一个双向链表结构的**单就绪队列**;
  - 所有进程按优先级排队;
  - 相同优先级的每个进程有一个时间片长度和虚拟截止时间;

## BFS 的时间片

- 时间片大小：由算法参数指定，可在 1ms 到 1000ms 间选择，缺省设置为 6ms；
- 虚拟截止时间（**Virtual Deadline**）：关于就绪队列中进程等待 CPU 最长时间的排序，并不是真实的截止时间；
  - 进程**时间片用完**时，重新计算虚拟截止时间；
  - **事件等待结束**时，虚拟截止时间保持不变，以抢先相同优先级的就绪进程；
  - 为了让进程在**上次运行的 CPU** 上运行（亲和性），不同 CPU 对进程的虚拟截止时间加一个权重；

## BFS 的虚拟截止时间计算

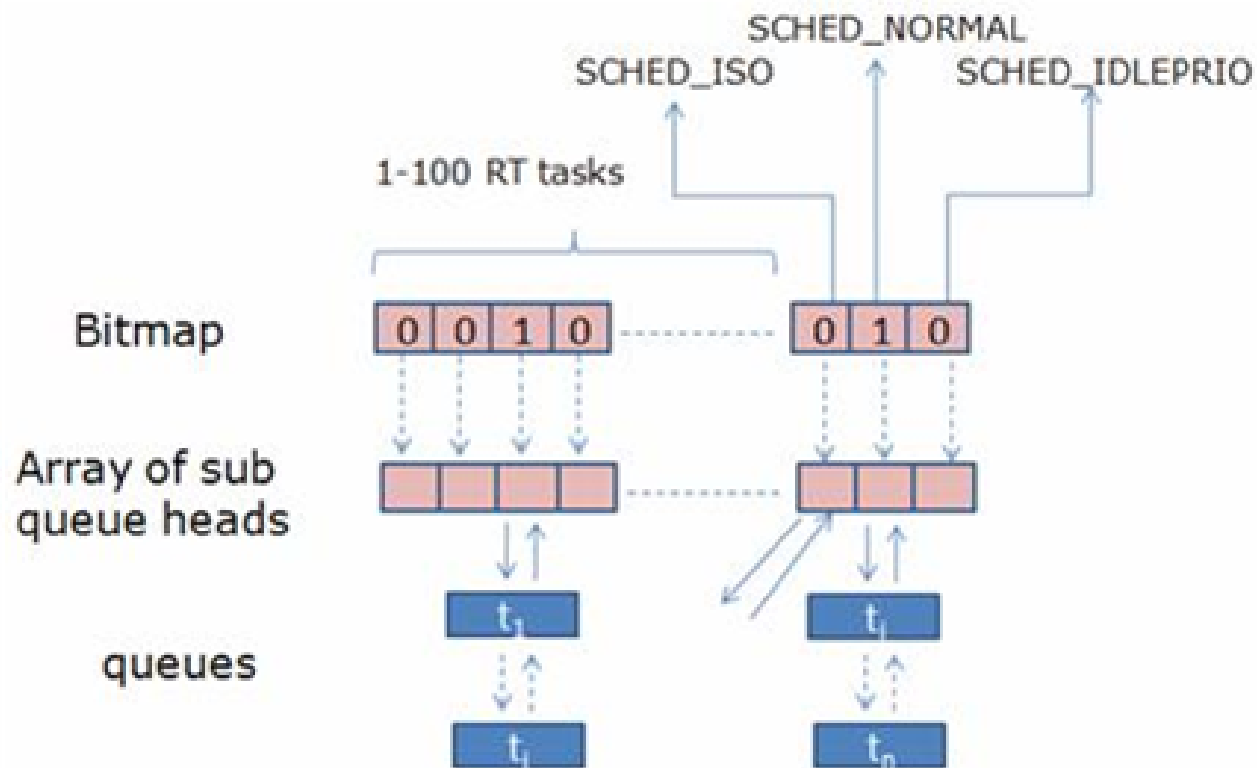
- 依据当前时间、进程优先级和时间片设置计算;

```
offset = niffies + (prio_ratio * rr_interval)
prio_ratio increases by 10% for every nice level
```

- niffies是当前时间; prio\_ratios[priority]是一个常量数组, 不同的priority对应不同的prio\_ratios[priority]; rr\_interval是timeslice, 是CPU分配给每个任务的时间片, 是一个常数
- 虚拟截止时间计算结果:  
[https://wikimili.com/en/Brain Fuck Scheduler](https://wikimili.com/en/Brain_Fuck_Scheduler)

# BFS 的调度思路

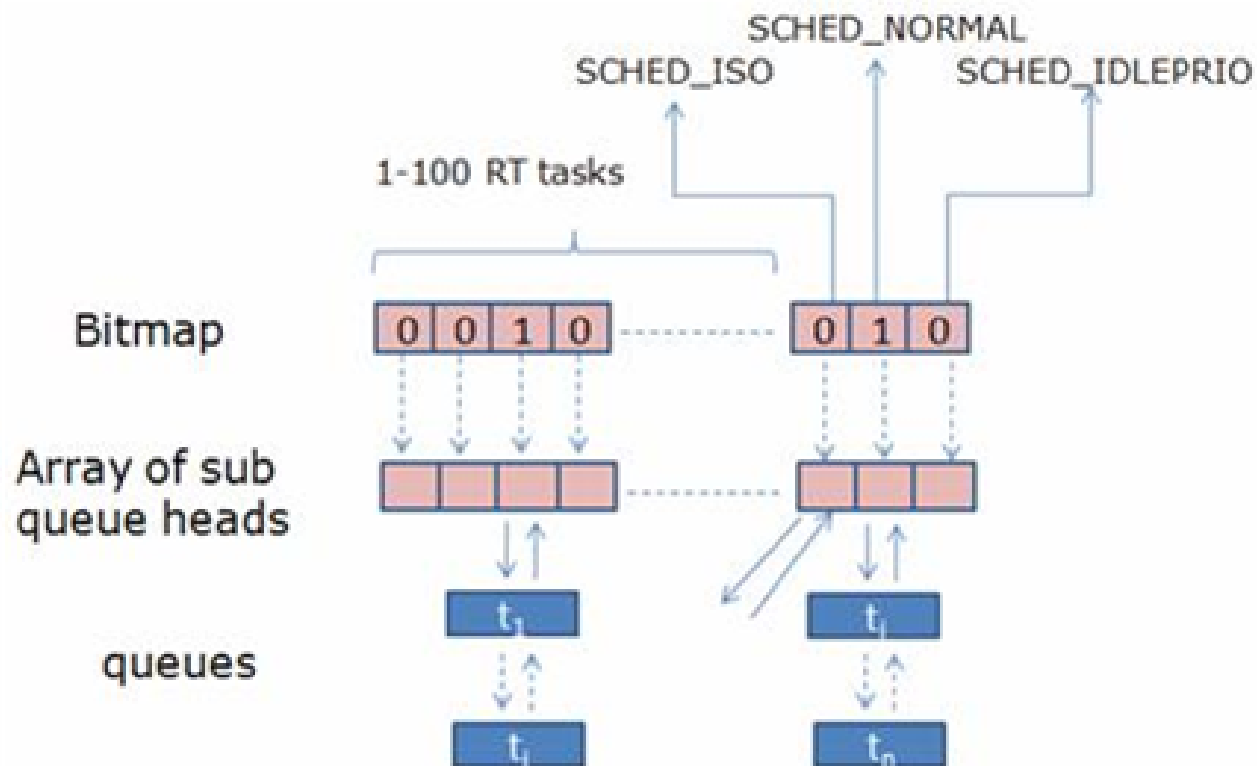
使用 $O(1)$ 调度器中的**位图**概念，所有进程被安排到**103**个**queue**中，各个进程不是按照优先级而是按照优先级区间被排列到各自所在的区间，每一个区间拥有一个**queue**。





## BFS 的调度思路

按照 $O(1)$ 调度器的方式首先查找位图中不为0的那个queue，然后在该queue中执行 $O(n)$ 查找，查找到virtual deadline最小的那个进程投入执行。



## BFS 的就绪队列插入

- 时间片用完：重新设置虚拟截止时间后，插入就绪队列；
- 等待事件出现：虚拟截止时间保持不变，抢先低优先级进程或插入就绪队列；

# 提纲

## 1. BFS调度器

## **2. BFS 与 CFS 的性能对比 (2012)**

# BFS 与 CFS 的性能对比 (2012)

## 测试用例集

- Linux kernel v3.6.2.2 的 GCC 编译
- Linux kernel v3.6.2 内核源代码树的 lrzip 压缩
- 从 720p 到 360p 的 MPEG2 视频 ffmpeg 压缩

Athlon XP 3200+

Intel E5200



Intel Atom 330

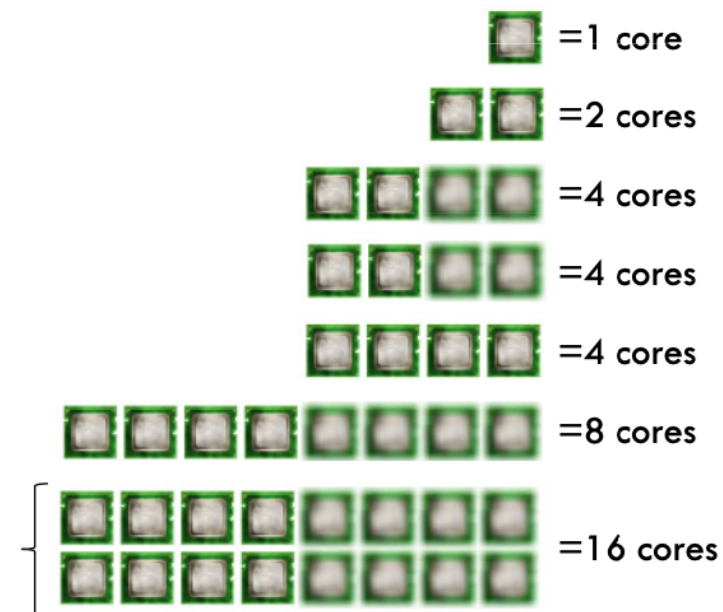
Intel i7-2620M

Intel X3360

Intel i7-3770K

Dual Intel E5620

 = Physical core  
 = Hyperthreaded core



Each test machine ran Arch Linux x86\_64 except for the Athlon XP which ran Arch i686 due to its lack of 64-bit support.

## BFS 与 CFS 的性能对比: 压缩测试

CPU	Average Time (sec)		CK Kernel is...	
	Vanilla	CK-Patched	Difference	Result
AMD Athlon XP	558.3320	<b>547.7577</b>	<b>-10.5753</b>	<b>1.9 % faster</b>
Intel E5200	166.3141	<b>165.7622</b>	<b>-0.5519</b>	<b>0.3 % faster</b>
Intel Atom 330	470.0283	<b>454.8185</b>	<b>-15.2098</b>	<b>3.2 % faster</b>
Intel i7-2620M	81.2978	<b>79.4898</b>	<b>-1.808</b>	<b>2.2 % faster</b>
Intel X3360	68.2635	<b>67.7987</b>	<b>-0.4648</b>	<b>0.7 % faster</b>
Intel i7-3770K	35.2904	<b>34.3310</b>	<b>-0.9594</b>	<b>2.7 % faster</b>
Dual Intel E5620	<b>27.7919</b>	28.2716	<b>+0.4797</b>	<b>1.7 % slower</b>

## BFS 与 CFS 的性能对比: 测试编译

CPU	Average Time (sec)		CK Kernel is...	
	Vanilla	CK-Patched	Difference	Result
AMD Athlon XP	1,120.6486	<b>1,095.6486</b>	-25.4671	2.3 % faster
Intel E5200	374.0274	<b>366.0912</b>	-7.9362	2.1 % faster
Intel Atom 330	1,568.5016	<b>1,546.4804</b>	-22.0212	1.4 % faster
Intel i7-2620M	192.5477	<b>190.6712</b>	-1.8765	1.4 % faster
Intel X3360	127.6179	<b>127.2340</b>	-0.3839	0.3 % faster
Intel i7-3770K	68.9835	<b>67.9671</b>	-1.0164	1.5 % faster
Dual Intel E5620	74.1218	<b>68.2665</b>	-5.8553	7.9 % faster

## BFS 与 CFS 的性能对比: 视频编码测试

CPU	Average Time (sec)		CK Kernel is...	
	Vanilla	CK-Patched	Difference	Result
AMD Athlon XP	380.8340	386.4206	+5.5866	1.5 % slower
Intel E5200	102.3183	99.8744	-2.4439	2.4 % faster
Intel Atom 330	471.0781	443.4450	-27.6331	5.9 % faster
Intel i7-2620M	50.1631	48.7489	-1.4142	2.8 % faster
Intel X3360	39.3656	37.6724	-1.7232	4.4 % faster
Intel i7-3770K	19.6863	18.1044	-1.5819	8.0 % faster
Dual Intel E5620	30.9037	30.7141	-0.1896	0.6 % faster

## 参考文献

- [http://repo-ck.com/bench/cpu\\_schedulers\\_compared.pdf](http://repo-ck.com/bench/cpu_schedulers_compared.pdf)
- <https://zhuanlan.zhihu.com/p/351876567>
- <https://blog.csdn.net/dog250/article/details/7459533>
- <https://www.cnblogs.com/dragonsuc/p/7144265.html>