基于缓存竞争优化的 Linux 进程调度策略

夏 厦，李 俊

(中国科学技术大学自动化系，合肥 230027)

摘 要：分析 Linux 经典内核版本 2.6.22 的进程调度算法，利用性能监测单元的监测信息，给出 3 个性能指标 CMR、CRR、OCIP 对进程的缓存竞争性强弱进行刻画，以此为依据，采用轮询算法优化 Linux 下的进程调度顺序，尽量避免在 CPU 上同时运行多个缓存竞争力强的进程，减小系统因缓存竞争产生的性能损耗。在 benchmark 上的测试结果表明，该方法能够提升系统在中、高负载下运行时的性能，在高负载下运行时的性能提升比例可达 6%左右。

关键词：Linux 内核；片上多处理器；进程调度；性能监测单元；进程行为；缓存竞争

Linux Process Scheduling Strategy Based on Cache Contention Optimization

XIA Sha, LI Jun

(Department of Automation, University of Science and Technology of China, Hefei 230027, China)

【Abstract】According to the analysis of the process scheduling algorithm of Linux kernel 2.6.22, through the information getting from Performance Monitor Unit(PMU), this paper puts forward three performance indexes CMR, CRR and OCIP to describe the process behavior of cache contention. On the basis of this, it uses polling algorithm to optimize the process scheduling in order to reduce the cache contention of Last Level Cache(LLC). Benchmark test results show that this algorithm can improve about 6% performance when the system load is high.

【Key words】Linux kernel; Chip Multi-processor(CMP); process scheduling; Performance Monitor Unit(PMU); process behavior; cache contention

DOI: 10.3969/j.issn.1000-3428.2013.04.014

1 概述

Linux 是当今世界上最流行的开源操作系统，其具有多用户、多线程、实时性好等多方位的优势。随着时间的推移，Linux 经历了多个版本的更新，其性能也在日益完善，最新发布的 Linux 内核版本为 2.6.24。由于对该版本的研究还有待加强，以及考虑到开发和运行的效率，本文将针对 Linux 比较经典的、也是目前研究的最多的版本 Linux2.6.22 的内核进程调度策略进行分析，并提出在多核处理器环境下的改进策略。

Linux 中的进程主要可分为实时进程和普通进程 2 种，实时进程要求响应的速度快且可靠性高，因此要比普通进程优先得到调度，在 Linux 中用优先级 0~99 代表实时进程，其优先级在 setscheduler()中设定，一经设定就不再改变。实时进程的调度策略主要有时间片轮转调度法(SCHED_RR)和先进先出调度法(SCHED_FIFO)这 2 种。普通进程主要采用 SCHED_NORMAL 策略，其优先级通过动态计算得到，优先级 100~139 代表普通进程[1]。

在 Linux 2.6.22 中，调度器为每一个 CPU 维护了 2 个进程队列数组，active 数组和 expire 数组，统称为就绪队列。其中 active 数组由时间片(time_slice)尚未用完的进程组成，而 expired 队列中存放着已用完时间片的进程。具有相同优先级 i 的进程都被插入到头指针为 queue[i]的链表中。由于系统共有 140 个不同优先级进程，因此 2 个数组大小都是 140。

当需要进行进程调度时，调度器可以通过 active 数组对应的 bitmap，选出 active 数组中优先级最高的队列中的第 1 个进程，作为候选进程 next，这种算法复杂度为 $O(1)$，因此，该调度器又称为 $O(1)$调度器。

Linux2.6 通过 $O(1)$调度器，给每个 CPU 设置了单独的

运行队列，尽量避免进程在不同核上的频繁迁移，有效平衡各个 CPU 的负载，同时兼顾了 Cache 的连续性，因此能较好的支持多核系统。但是对于 CMP 结构而言，Linux 还有所不足，例如在文献[2]中提到的 Linux 内核不能够识别有关联的进程，从而将它们分配到一个核上，提高 Cache 的命中率。

如何提高 Cache 的命中率，一直是研究的热点。文献[3]从 Cache 替换算法角度分析，提出了优化 Cache 命中率的方法；文献[4]中提出 3 种预测 CMP 上线程间竞争的模型，但这些模型都需要离线统计信息，难以应用于实时操作系统中。本文针对 CMP 架构下进程之间的缓存竞争问题，通过在线获取的进程参数，对进程的行为特征进行分析，并以此为指导，优化 Linux 的进程调度顺序，降低进程之间因共享缓存的竞争带来的性能下降。

2 改进算法

2.1 缓存资源的竞争

CMP 架构是将多个相对简单的超标量处理器核集成到一个芯片上，从而提高吞吐量，典型代表是 Intel Core 微架构处理器，每个核都拥有独立的 L1 Cache，并且共享 2 MB 或 4 MB 的 L2 Cache，图 1 是典型的 Intel Core 微处理器的结构示意图。

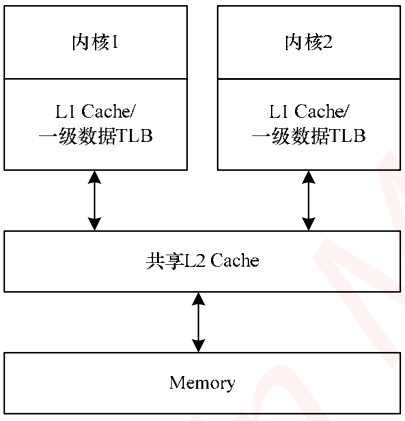

图 1 Intel Core 微处理器结构

当核 1 和核 2 上同时有进程运行时，就会向共享缓存中读/写入数据，从而形成对 Cache 的竞争。文献[5-6]详细地分析了共享 Cache 竞争对于系统性能的影响。根据进程共享 Cache 资源的占用程度，可以将进程划分为 Cache 竞争性强、中、弱这 3 类。当核 1 和核 2 同时运行 Cache 竞争性强的进程时，将会导致 L2 Cache Miss 数急剧上升，影响 CPU 的性能。

2.2 进程特性的刻画

2.2.1 事件监控单元

在现代处理器中，一般都会在芯片上集成一种硬件事件的监测单元(Performance Monitor Unit, PMU)，能对底层的硬件的各种指标进行监测[7]，表 1 为性能计数器可以监测的事件(部分)。

表 1 PMU 单元功能

周期	时钟周期
Load instructions	装载内存指令数
Store instructions	写入内存指令数
L1 misses	L1 Cache 缺失数
L2 misses	L2 Cache 缺失数
L2 cache reference	L2 Cache 访问数
Total cache misses	内部 Cache 的未命中数
Instructions	已完成的指令数

2.2.2 进程特征的选取

利用 PMU 单元读取的数据，可以对进程的一些行为特征做出判断，例如文献[8]利用 PMU 计数单元获取的数据，提出了对于进程发热量的刻画标准。采用类似思想，可以利用 PMU 单元的在线统计功能，分析进程对共享 Cache 资源竞争性的强弱。下面将从理论上提出 3 个性能指标，对进程进行刻画。

指标 1 CMR(L2 Cache Miss Rate)

$$C_{\mathrm{CMR}} = \frac{\text{L2 Cache Misses}}{Instructions} \tag{1}$$

当处理器访问数据的时候，会首先从 Cache 中寻求数据，如果 L2 Cache 中没有则从内存中寻找。L2 Cache Miss 越多，说明进程将从内存读取更多的数据，替换 Cache 中原有的数据(现代 Cache 数据替换常见的是采用 LRU 算法)，从而有可能污染另一个核上进程在 Cache 的数据。CMR 越大，表示对共享 Cache 竞争性越强。

指标 2 CRR(L2 Cache Reference Rate)

$$C_{\mathrm{CRR}} = \frac{\text{L2 Cache Reference}}{Instructions} \tag{2}$$

CRR 反映了一个进程对 L2 Cache 访问的频繁程度。CRR 越高，表示进程对 L2 Cache 的访问越频繁，从而消耗更多的 L2 Cache 资源，因此 CRR 越大，该进程的共享 Cache 竞争性越强。

指标 3 OCIP(Off-chip Instruction Proportion)

$$O_{\mathrm{OCIP}} = \frac{I_{\mathrm{load}} + I_{\mathrm{store}}}{Instructions} \tag{3}$$

其中，$I_{\mathrm{load}}+I_{\mathrm{store}}$ 表示装载和写入内存指令数之和。OCIP 反映缓存与内存交互的频繁程度。I_{load} 越大，表示由内存写入缓存的数据越多，I_{store} 越大，表示由 CPU 写回缓存的数据越多，从而占用更多的共享缓存空间，因此，OCIP 越大，表示该进程越倾向于 Cache 竞争性强的进程。

2.2.3 实验验证

为证实上述理论，采用 SPEC CPU2006 基准测试程序集[9]，测试各程序对于 L2 Cache 竞争性的强弱与上述 3 个指标之间的关系。

测试环境：ubuntu7.10、Intel Core2 Duo E4600 CPU(主频 2.4 GHz，L1 cache 256 KB，L2 cache 2 MB)、DDR2 1 GB 内存、Linux 内核版本 2.6.22。

实验方法如下：

(1)在核 1 上运行基准测试程序 mcf，测试它的运行时

间。之所以选择 mcf 作为基准测试程序，是因为 mcf 运行速度与 L2 cache 有很大的关系，能直观地反映另一个核上的测试程序对共享 cache 竞争性的强弱。

(2)在核 2 上分别运行 bzip、gamess 等 15 个测试程序，同时测试核 1 上 mcf 的运行时间。通过核 1 的性能下降程度(以 mcf 运行时间为标准)，作为度量 15 个测试程序的缓存竞争性强弱的标准。

(3)通过 PMU 监测单元信息，统计平均每执行 100 万条指令 L2 cache misses、L2 cache reference、$I_{\text{load}}+I_{\text{store}}$ 的值。实验结果如图 2 所示。

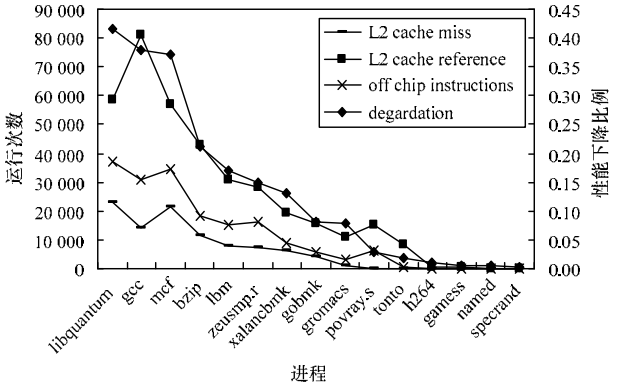

图 2　3 个进程特征值对性能的影响

图中左边的主坐标轴表示 100 万条指令中 L2 cache miss、L2 cache reference、off chip instructions 的数量，右边的副坐标轴表示测试进程 mcf 的性能下降幅度。

2.3　目标函数的提出和实现

从图 2 可以看出，以上 3 个进程特征与缓存竞争性的强弱有明显的相关性，为了能更准确地刻画进程对共享缓存的影响，以测试进程 mcf 的性能下降程度作为目标函数：

$$E=\lambda_1\times C_{\text{CMR}}+\lambda_2\times C_{\text{CRR}}+\lambda_3\times O_{\text{OPIP}} \tag{4}$$

为了确定 λ_1、λ_2、λ_3 的最佳取值，本文根据 15 个测试进程的结果，选取当 $\sum_{n=1}^{15}|\lambda_1\times CMR_n+\lambda_2\times CRR_n+\lambda_3\times OPIP_n-E_n|$ 取最小值时 λ_1、λ_2、λ_3 的值，同时考虑到运算的效率(尽量减少浮点数运算)，最终取 $\lambda_1=10$，$\lambda_2=2$，$\lambda_3=2$。图 3 反映了目标函数 E 的值与实际值的对比。

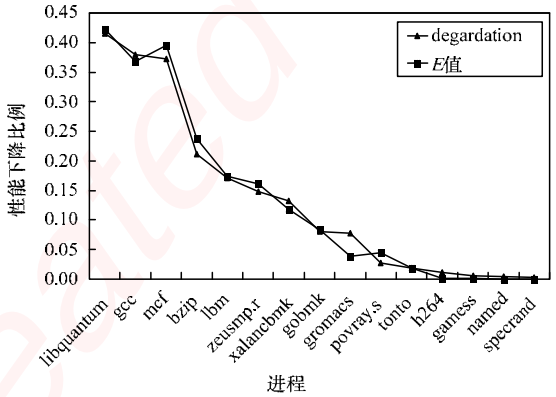

图 3　目标函数值与实际性能下降之间的关系

可以看到选取的进程目标函数与该进程对测试程序 mcf 造成的实际影响吻合得非常好，从而能准确地反映该进程对于共享缓存竞争性的强弱。

为了实现上述功能，在 schedule()函数中添加如下模块：

```
Void_sched schedule(void)
 {…
   if (likely(prev!=next))
  {   if(stat_cmr_switch==1)
          cmr_stat_end(prev);//统计 cmr
      if(stat_crr_switch==1)
          crr_stat_end(prev)//统计 crr
      if(stat_ocip_switch==1)
          ocip_stat_end(prev)//统计 ocip
prev->character=task_io_charactrer(cmr_count,crr_count,  ocip_
count)
//计算特征函数
prev=context_switch(rq,prev,next) //进程切换
barrier();
if(stat_cmr_switch==1)
        cmr_stat_start();//打开 cmr 统计
          …//打开 crr 统计
          …//打开 ocip 统计
finish_task_switch(this_rq(),prev);
}else
      spin_unlock_irq(&rq->lock)
…}
```

即在进程发生切换时对旧任务的任务特征值进行现场保存，计算任务特征函数，并在新任务开始时打开任务特征值的统计。

2.4　改进算法

通过上述分析，可以在线统计进程对共享 Cache 资源竞争性的强弱。2 个核上运行进程的 Cache 竞争性组合可能为(强，弱)、(中，弱)、(弱，弱)、(中，中)、(中，强)、(强，强)这 6 种情况，其中，前 4 种不会对处理器的性能产生大的影响，而后 2 种情况则应该尽量避免。界定特征值 $E\in(0,0.1)$ 的进程为 Cache 竞争性为“弱”；$E\in(0.1,0.2)$ 的进程 Cache 竞争性为“中”；$E>0.2$ 的进程 Cache 竞争性为“强”。基于以上思想，对 schedule()函数做出修改，增加 task_choose_from_queue()单元：

```
Void task_choose_from_queue()
       { sched_get_info(p,cpu_id)
//发生进程切换时，获取另一个核上当前运行的进程信息；
   if (p->character >0.2)
//另一个核上运行的进程的共享 cache 竞争性“强”；
next=search_min_character(queue[idx].next,task_t,run_list)
//搜寻当前最高优先级队列下特征函数值最小的进程，即
//else if(p->character <0.1)
```

```
//另一个核上运行的是 cache 竞争性弱的进程；
next=list_entry(queue[idx].next,task_t,run_list)
//遵循 Linux 原有的调度策略；
else  if(p->character > 0.1&& p->character < 0.2)
//cache 竞争性为“中”；
{if(next=search_fit_charactor(queue[idx].next,task_t,run_list))==NULL
//搜寻队列中第 1 个 cache 竞争性非“强”的进程；
next=search_min_character(queue[idx].next,task_t, run_list)
//如果没有则取队列中特征值最小的进程；
}
}
```

3 调度器改进后的测试

测试环境：ubuntu7.10、Intel Core2 Duo E4600CPU(主频 2.4 GHz，L1 cache 256 KB，L2 cache 2 MB)、DDR2 1 GB 内存、Linux 内核版本 2.6.22。

比较对象：Linux 内核 2.6.22、改进的 Linux 2.6.22。

测试程序：从 Mibench[10]和 SPEC CPU2006 中选取了 20 个基准测试程序(benchmark)，通过 sched_setaffinity()函数，可以将其分配到指定的核上运行(主要是防止 Linux 中的负载均衡函数将进程迁移到其他核上，对试验结果造成影响)。

分别使用改进后的 Linux 内核和原版对比，测试 CPU 在低、中、高负载下(低负载时每个核上运行 2 个测试程序，中负载时每个核上运行 4 个测试程序，高负载时每个核上运行 8 个测试程序)，每周期执行指令数值(Instructions Per Cycle, IPC)的变化，定义加速比 $speedup=\frac{IPC_{改进}}{IPC_{原}}-1$，得到结果如图 4 所示。

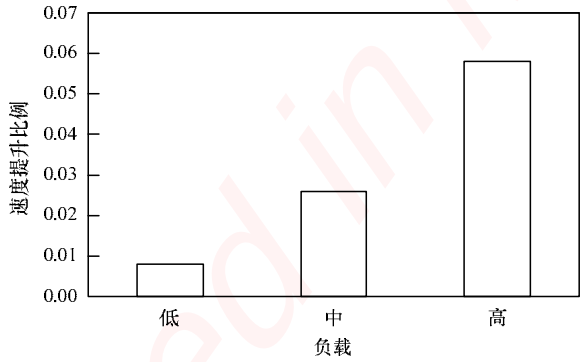

图 4 改进后的内核性能测试

从分析结果可以看出，本文的算法在低负载是提升效果并不明显，原因是相同优先级队列中可供选择的进程较少，进程的调度方式相较于原来的内核差异不大；在中、高负载时，尤其是在高负载下，性能提升较为明显。

4 结束语

本文提出一种利用 PMU 在线监测数据对进程的缓存竞争性强弱进行判定的方法，利用该方法，对 Linux 2.6 内核的进程调度顺序做出调整。实验结果表明，经过修改后的内核比原内核的性能有所提升。由于现在的内核普遍采用完全公平调度器(CFS)，因此今后主要的研究方向是如何将缓存竞争优化策略与完全公平调度器相结合，提升系统性能。

参考文献

[1] Bovet D P. 深入了解 Linux 内核[M]. 陈莉君, 张琼声, 张宏伟, 译. 北京: 中国电力出版社, 2008.

[2] 覃 中. 基于多核系统的线程调度[D]. 成都: 电子科技大学, 2009.

[3] 林晓敏, 桂 婷, 乔福明. 多核系统中共享 Cache 的冒泡替换算法[J]. 微电子学与计算机, 2011, 28(4): 118-121.

[4] Chandra D, Guo F, Kim S. Predicting Inter-thread Cache Contention on a Chip Multi-processor Architecture[C]// Proceedings of the 11th International Symposium on High-performance Computer Architecture. San Francisco, USA: [s. n.]. 2005: 340-351.

[5] Zhang Xiao, Dwarkadas S, Shen K. Towards Practical Page Coloring-based Multi-core Cache Management[C]// Proceedings of the 4th ACM European Conference on Computer Systems. [S. 1.]: ACM Press, 2009: 89-102.

[6] Jaleel A, Hasenplaugh W, Qureshi M, et al. Adaptive Insertion Policies for Managing Shared Caches on CMPs[C]//Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. Toronto, Canada: [s. n.], 2008: 208-219.

[7] 夏 亮. 温度感知的调度算法研究与实现[D]. 上海: 上海交通大学, 2009.

[8] 王中飞. 系统级处理器热量控制的研究和设计[D]. 合肥: 中国科学技术大学, 2011.

[9] Spec[EB/OL]. (2012-04-20). http://www.spec.org/cpu2006/.

[10] Mibench[EB/OL]. (2012-04-20). http://www.eecs.umich.edu/mibench/.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现
51. IEEE 1588 协议在工业以太网中的实现
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现
57. 一种基于 IPv6 的流媒体传送方案研究与实现
58. 基于 IPv4-IPv6 双栈的 MODBUS-TCP 协议实现
59. RS485CAN 网关设计与实现
60. MVB 周期信息的实时调度
61. RS485 和 PROFINET 网关设计
62. 基于 IPv6 的 Socket 通信的实现
63. MVB 网络重复器的设计
64. 一种新型 MVB 通信板的探究

VxWorks:

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统

7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法
40. 基于 Tilcon 的 IO 控制板可视化测试软件的设计和实现
41. 基于 VxWorks 的通信服务器实时多任务软件设计
42. 基于 VXWORKS 的 RS485MVB 网关的设计与实现

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现

43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计
47. 基于 Android 平台的软件自动化监控工具的设计开发
48. 基于 Android 的视频软硬解码及渲染的对比研究与实现
49. 基于 Android 移动设备的加速度传感器技术研究
50. 基于 Android 系统振动测试仪研究

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现
25. 虚拟仪器的 Windows XP Embedded 操作系统开发
26. 基于 EVC 的嵌入式导航电子地图设计

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统
19. 基于 PowerPC 的控制器研究与设计
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法
23. 基于 PowerPC440GP 型微控制器的嵌入式系统设计与研究
24. 基于双 PowerPC 7447A 处理器的嵌入式系统硬件设计
25. 基于 PowerPC603e 通用处理模块的设计与实现

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计

6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植
28. 基于 ARM11 的嵌入式无线视频终端的设计
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 上的移植
32. 基于 ARM11 的嵌入式 Linux NAND FLASH 模拟 U 盘挂载分析与实现
33. 基于 ARM11 的电源完整性分析
34. 基于 ARM S3C6410 的 uboot 分析与移植

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备

8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发
24. 基于 X86 平台的嵌入式 BIOS 可配置设计
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法
31. 一种基于 CPUGPU 异构计算的混合编程模型
32. 面向 OpenCL 模型的 GPU 性能优化

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案