分类号：_____ 密级：_____
编号：_____

桂林理工大学

硕士研究生学位论文

面向多核处理器的嵌入式操作系统微内核任务调度机制的研究

专　　业：计算机应用技术

研究方向：嵌入式系统应用技术

研 究 生：沈凡凡

指导教师：黎卫红 高级工程师

论文起止日期：2012 年 4 月至 2013 年 4 月

Research of Embedded Operating System Micro-kernel Task Scheduling Mechanism based on Multi-core Processor

Major：Computer Application Technology

Direction of Study：Embedded System Applied Technology

Graduate Student： Shen Fanfan

Supervisor： Prof. Li Weihong

College of Information Science and Engineering
Guilin University of Technology

April,2012 to April,2013

研究生学位论文独创性声明和版权使用授权书

独 创 性 声 明

本人声明：所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含他人已经发表或撰写过的研究成果，也不包含为获得其它教育机构的学位或证书而使用过的材料。对论文的完成提供过帮助的有关人员已在论文中作了明确的说明并表示谢意。

学位论文作者（签字）：沈凡凡

签字日期：2013年6月4日

学位论文版权使用授权书

本学位论文作者完全了解(学校)有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的印刷本和电子版本，允许论文被查阅和借阅。本人授权(学校)可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。同时授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。(保密的学位论文在解密后适用本授权书)

本论文是否保密： 是 否✓ 如需保密，保密期限为：

学位论文作者签名：沈凡凡 导师签字：黎卫红

签字日期：2013 年 6 月 4 日 签字日期：2013年 6 月 4 日

摘　要

近年来，多核处理器的应用已经越来越广泛，为了充分发挥多核处理器的效率，提高嵌入式操作系统微内核性能已成为一个重要的研究课题，而多核处理器的任务调度机制的好坏是影响嵌入式操作系统微内核性能的关键因素。目前，国内外的许多研究人员都对多核处理器的任务调度机制进行了研究，主要通过改变任务的执行次序、减少任务间的通信开销及减少任务调度的次数等。然而当前多核处理器环境下，嵌入式操作系统微内核的任务调度效率仍然不高，存在任务间通信开销大、任务分配不合理及处理器的负载不均衡等问题，因此为进一步提高系统的性能，需要更加深入的研究多核处理器的任务分配模型和任务调度机制。

本文在多核处理器环境下，对嵌入式操作系统微内核的任务调度算法和调度机制进行了研究。分析了主流的任务调度机制的优缺点，并对其进行了对比分析，在此基础上，针对多核处理器任务调度机制进行了改进，提出了一种高效的任务分配模型和任务调度机制。（1）任务分配模型。任务分配模型的好坏直接影响任务的执行次序及系统的实时性，因为当高优先级任务到来时，如果分配的不合理，就必然会影响任务执行的实时性。因此本文给出了多核处理器下任务分配的通用规则，提高了任务分配的效率。接着给出了改进的基于优先级的抢占式任务分配机制，使高优先级的任务能够实时快速执行，提高了系统的实时性。（2）任务调度机制。本文用全局队列调度和局部队列调度相结合的方法实现了任务的调度，提出了基于缓存队列的任务调度机制，将全局队列中的任务调度到每个处理器的缓存队列，大大提高了缓存的命中率，提高了处理器的利用率，为了使处理器的负载更加均衡，对全局队列的调度方式进行了改进，融合了快内核优先的动态迁移策略，这些都保证了系统的实时性和负载均衡性。本文设计的改进多核处理器任务调度机制提高了嵌入式操作系统微内核的性能，满足了当前人们对系统的性能要求。

为了验证本文设计的任务分配模型和任务调度机制的高效性和可行性，本文在Simics 仿真平台上进行了测试，搭建了多核处理器的硬件平台和嵌入式操作系统微内核，根据本文的任务分配模型和任务调度机制设计原理，对嵌入式操作系统微内核中的相关的部分进行修改，接着在嵌入式操作系统微内核上运行一段多任务的测试程序，对改进的方法和机制的性能进行了验证。实验结果表明，本文设计的多核处理器任务分配模型和任务调度机制具有良好的性能，系统的实时性得到了提高。

关键词：多核处理器，嵌入式操作系统，微内核，任务调度机制，任务分配模型

Abstract

In recent years, the application of multi-core processor has been used more and more widely, in order to exert the efficiency of multicore processors, improve the performance of operating system has become an important research topic, and task scheduling mechanism of multi-core processor is the key factor affecting the performance of operating system. At present, many researchers both at home and abroad has researched the mechanism of task scheduling for multi-core processors. They discuss it mainly by changing the task execution order, reduce the communication overhead between tasks and reduce the number of task scheduling. However, in the current multi-core processor environment, task scheduling efficiency of embedded operating system microkernel is still not high, there is communication overhead between tasks, task allocation is not reasonable and the processor load imbalances and other issues, so as to further improve the performance of the system, we need research task allocation model and task scheduling mechanism more deeply on multi-core processors.

In this paper, in the multi-core processor environment, we research the embedded operating system microkernel task scheduling algorithm and scheduling mechanism. Analysis the advantages and disadvantages of the task scheduling mechanism of the mainstream, and compare them together, on the basis of this, some improvements are made for the multiprocessor task scheduling mechanism, this paper presents an efficient task allocation model and task scheduling mechanism. (1) The task allocation model. Task allocation model will directly affect the real-time execution order and the system tasks, because when a high priority task arrives, if the distribution is not reasonable, it will affect the real-time task execution. This paper puts forward a preemptive task allocation mechanism based on priority, so that the high priority task can execute quickly in real-time, improve the real-time performance of the system. Then it gives the general rules of task assignment multi-core processor, improves the efficiency of task allocation. (2) The task scheduling mechanism. This paper combines the global queue scheduling and local scheduling to achieve the task scheduling, then pretends the task scheduling mechanism based on cache queue, greatly improving the cache hit rate, improves the processor utilization, in order to load more balanced for the processor, the global queue scheduling method was modified, fusion of dynamic migration strategy of fast kernel priority, which are ensured the system's real-time and load balancing. Improved

multiprocessor task scheduling mechanism is designed to improve the performance of embedded operating system micro-kernel, it meets the demands for the performance of the system.

In order to validate the feasibility and efficiency of the design task assignment model and task scheduling mechanism, this paper tested in the Simics simulation platform, we built a hardware platform of multi-core processor and operating system, according to the design principle of task scheduling mechanism, we modified the task scheduling mechanism of the embedded operating system which is related to part of it, and then running a multitasking program on the operating system, we verify the improvement in the performance of the task scheduling mechanism. The experimental results show that, the multi-core processor task scheduling mechanism we designed in this paper has a good performance. The real-time performance of the system is improved.

Key words: Multi-core processor, Embedded operating system, Micro-kernel, Task scheduling mechanism, Task allocation model

目 录

摘　要 I
Abstract II
目 录 IV
第 1 章 引言 1
1.1 研究背景和意义 1
1.1.1 课题研究的背景 1
1.1.2 课题研究的意义 2
1.2 国内外研究现状 3
1.3 本论文主要内容 4
1.4 论文章节安排 5
第 2 章 多核处理器和微内核操作系统的技术概况 7
2.1 多核处理器技术及其架构 7
2.1.1 多核处理器的概念 7
2.1.2 多核处理器的架构 8
2.2 微内核操作系统概述 9
2.2.1 微内核操作系统 9
2.2.2 微内核操作系统的优点 10
2.2 常见的微内核操作系统介绍 11
2.2.1 第一代微内核操作系统Mach 11
2.2.2 第二代微内核操作系统L4 12
2.2.3 第三代微内核操作系统seL4 12
2.2.4 其它微内核操作系统 13
2.3 本章小结 14
第 3 章 多核处理器下微内核任务调度机制的分析 15
3.1 多核处理器任务调度基本原理 15
3.1.1 多核处理器进程原理 16
3.1.2 多核处理器任务的调度 17
3.1.3 多核处理器任务间的通信 18
3.2 多核处理器的任务调度算法 20
3.2.1 静态任务调度算法 20
3.2.2 动态任务调度算法 22
3.2.3 两种算法的性能比较 23
3.3 现有的任务调度机制和策略分析 23
3.3.1 单核任务调度策略 24
3.3.2 多处理器任务调度机制和策略 25
3.4 多核处理器任务调度机制的评价标准 26
3.5 本章小结 27

第 4 章 多核处理器下微内核任务调度机制的设计 28
4.1 多核处理器任务调度机制问题描述 28
4.2 任务分配模型的设计 29
4.2.1 现有的任务分配模型 29
4.2.2 改进的任务分配模型设计 30
4.2.2.1 基于多核处理器的任务分配规则 30
4.2.2.2 基于优先级的抢占式任务分配机制 31
4.3 多核处理器任务调度机制的设计 32
4.3.1 改进的任务调度机制设计 33
4.3.1.1 基于缓存队列的任务调度机制 33
4.3.1.2 快内核优先的任务动态迁移策略 34
4.3.2 改进的任务调度机制评价 34
4.4 本章小结 34
第 5 章 实验测试与结果分析 35
5.1 多核处理器实验平台 35
5.1.1 Simics仿真平台的介绍 35
5.1.2 Simics多核仿真平台的搭建 37
5.2 多核处理器任务调度机制性能测试结果与分析 40
5.2.1 改进的任务调度机制实现 40
5.2.2 性能测试程序设计 41
5.2.3 性能测试结果对比分析 43
5.3 本章小结 45
第 6 章 总结与展望 46
6.1 工作总结 46
6.2 研究展望 47
参考文献 48
个人简介、申请学位期间的研究成果及发表的学术论文
致 谢

第 1 章 引言

1.1 研究背景和意义

1.1.1 课题研究的背景

随着科学技术的不断发展与进步，信息技术和计算机技术的发展日新月异。近年来，社会的各个领域都逐步向信息化方向发展，信息技术已经深深的改变了我们的生活方式，从日常的个人应用软件，到高端通信、航空航天、医疗成像等大型高科技应用，计算机技术扮演着非常重要的角色，它起着非常重要的作用。计算机科学技术的迅速发展促进了现代应用的发展，现代应用的新需求又推动着计算机技术的进步。

随着科技的发展，人们不断追求更快、更高效及更节能的计算机，对高性能的处理器也是非常的渴望。20 世纪中期，半导体工艺的日益崛起，传统提升处理器性能的方法是提高处理器的主频，处理器的主频得到极大的提升，有些已经达到 4GHZ，但是，到了 21 世纪，由于主频的不断提升，芯片集成的复杂度越来越大，处理器的散热和功耗问题日益突出，这使得处理器的主频几乎达到了其物理极限，通过纯半导体工艺已经无法再提高处理器的主频。同时，传统的提高处理器并行性能的方式是使用超标量和超流水技术，但是这种设计的成本非常高，单核处理器的性能提高因此受到了很大的影响。然而不断变化的新应用需求需要超高性能的处理器来提供服务，单核处理器的性能提高与新应用对处理器的高速运转的矛盾，成为单核处理器发展的主要矛盾，这一矛盾推动了人们对处理器体系结构的改进和设计，从而出现了多核处理器。

如今，多核处理器已经发展成为时代的主流了，主要有两类多核处理器，分别为同构多核处理器及异构多核处理器，同构多核处理器即内核的类型、功能和各内核的地位都完全相同，而同构多核处理器不同，异构多核处理器中各内核的类型、功能并不完全相同，地位也不均等。在实际应用中，我们应该选择适合项目的多核处理器的结构。

多核处理器是通过并行技术实现的，它给用户提供了高速的硬件计算平台，但是光有硬件是无法完全发挥他的作用的，需要软件的支持，需要操作系统微内核支持多处理器任务调度，支持应用程序的并行执行。然而传统的软件设计是遵循串行计算标准的，不能适应多核处理器结构的性能要求。所以，随着市场上广泛的使用多核处理器，为了充分发挥它的性能，需要对操作系统微内核的任务调度、程序设计方法及优化技术做出相应的改进，这些都成为了研究的重点。

计算机系统软件的最重要组成部分是操作系统，它管理着整个计算机的系统资源。包括硬件资源和所有的软件资源。操作系统能支持多处理器架构的平台，并为所有的软件提供统一的基础平台，是需要在多核环境下解决的一个基本问题，此外，微内核操作

系统的任务调度机制的效率是多处理器结构的性能最重要的决定因素。因此，多核环境下，操作系统的微内核任务调度机制的研究是一个热点和发展趋势。

1.1.2 课题研究的意义

近年来，计算机研究领域的一大热点是多核处理器，在计算机科学技术发展的过程中，在业界有一条非常著名的定律，即摩尔定律。戈登·摩尔(GordonMoore)（英特尔(Intel)创始人之一）提出了这条著名的定律。在半导体产业和信息技术发展的过程中，这一定律一直指引着它们的发展。在单核处理器环境下，人们一般是使用超流水线的方法来提高主频，它能使处理器的性能得到很大的提升，另外一种提升性能的方法是用superscalar(超标量)的并行技术，处理器在一个时钟周期内执行多条指令。但是随着科技的进步，一个芯片里面集成的晶体管越来越多，有些芯片里面已经达到数亿个晶体管，这样的集成方式使得处理器的设计难度以及它的功耗都非常大，同时它的发热量也会变得非常大，那么芯片的稳定性就会变得越来越差。因此在单核处理器设计中出现了这样的瓶颈，处理器的功耗、电路的复杂性及内存的消耗等问题都非常突出，这些因素都严重的阻碍了单核处理器性能的提升，目前对单核处理器的功能已经发挥到了极限，性能已经无法再提高了。这些发展状况都印证了摩尔定律的权威性和指导性，同时对人们提出了新的命题和挑战。在 2007 年的英特尔技术峰会上，戈登摩尔说：“大家都知道，摩尔定律指出，半导体技术在物理上是有极限的，每隔 10 年，它就会遇上这类问题，而促进新技术的产生。”也就是说，戈登摩尔认为，半导体科技的发展会继续发展 10 年左右，但是它的发展方向不会一直停留在单核处理器上，信息技术界都认为目前应该将发展眼光投向蓬勃发展的多核处理器。多核处理器的使用和深入研究已经成为了处理器发展的必然趋势和重点方向。

如今嵌入式领域、PC 领域及移动互联网领域，人们都迫切希望系统具有高效的计算能力，需求比过去要高几十甚至几百倍的计算速度。然而当前的单核处理器在处理能力上根本没法满足这种需求，同时现在单核处理器的性能已经达到了最高，仅通过提高其主频来提高系统的整体性能的代价是非常大的，也是无法突破极限的。为了使计算机的计算速度更快，计算机行业的人士都认为多核处理器是潜在的发展方向。多核处理器是集成在片上的多个处理器，多个低频率的处理器合成在一个芯片上，通过并行运算来提高计算机的性能和速度。当前的所有计算机及嵌入式产品架构都开始投向多核心处理器，并没有停留在提升单核处理器的性能上。因此多核技术的开发是个很好方向。

当前，多核处理器已经广泛的应用到个人计算机上，使计算机的性能有了很大幅度的提升，操作系统能充分的利用这些多核资源，但是，在嵌入式系统环境下，其中的操作系统一般是各种裁剪过的的 Linux 系统或者 WinCE 系统，在嵌入式系统行业中，使用最多的操作系统是 Linux 系统，但是 WinCE 及 Linux 系统由单内核构成的，并且其结

构非常复杂，不适合在嵌入式平台使用。又由已知的μC/Os-II 和 eCos 操作系统过于简单，不能提供完整的服务功能给用户。目前，微内核操作系统的发展和技术水平已经非常成熟，并且能够提供足够的服务功能，极适合在嵌入式系统环境下使用，是一个非常优秀的系统。微内核操作系统是指将内核中必要的功能放在内核态，而其它非必要的功能和服务则交由用户态来管理和执行，这些必要的服务包含进程/线程的调度，内存映射和进程间通信，像内存分配、文件系统和驱动程序则由用户态的程序来管理。

由上面可知，微内核操作系统适用于嵌入式系统平台，但是现有的微内核操作系统并不能很好的支持多核架构，多核架构是指在一个处理器上集成多个计算内核。但随着半导体技术和计算机技术的发展，如今多核处理器已经在个人电脑中广泛的使用，人们已经有能力购买这样的计算机，而不是像服务器、工作站等可望而不可及的高端机，在今后多核处理器的使用会普及到各个领域。目前平板电脑、智能手机等数码产品已经在广泛的使用多核技术了。

如今，嵌入式微内核操作系统在多核处理器环境下的使用已经成为了一种趋势，因此，本论文研究并设计一套嵌入式操作系统微内核任务调度机制和策略，通过这来保证多核的负载平衡及最少完成时间，并充分发挥各个内核的性能特性，提高整个系统的吞吐量和实时性，这对于嵌入式微内核操作系统在多核环境下的发展具有极其重要的意义。

1.2 国内外研究现状

针对多核处理器资源利用的问题，国外学者 Andrew Baumann 认为，在设计操作系统时需要考虑分布式系统的原理，将网络的基本功能融入到系统中，将多核处理器系统虚拟成通过网络连接的各个节点，并且各个节点之间没有公共资源。他给出了微内核操作系统的基本设计原则，包括操作系统与底层硬件的无关性、以资源复制的方式看待资源及多个内核间显示通信。同时，Chao Wang 也认可这种设计方式，认为这是一个很好的想法可发展方向，与分布式系统的思想相结合，通过网络来传递数据，即可认为多核处理器系统是通过分布式芯片组成实现的。动态任务调度方式已经在这种系统下广泛的研究，其资源的设计和使用方式不同，因此将宏观上的分布式系统理论应用到当前的系统是很难实现的。

通过将集成的通信资源整合起来应用到并行操作系统中，通过映射的方式将任务分配到节点上，将操作系统控制系统自适应路由、动态注入率控制和动态数据信息的统计等功能加入到操作系统中。Vincent Nollet 认为这种设计方式将操作系统的功能分为多个小部分，在主节点上运行操作系统的内核，而从节点运行各个模块的功能，当操作系统需要调用其它功能时，则使用远程调用功能，通过消息机制把信号传送到相应的从节点，

它完成对应的功能后，在将信息以消息的形式返回给主节点。这样就完成了一次完整的通信和信息交互。

学者 Wei Hu 认为当前多核处理器设计方式存在一定的问题，通过集中式的方式控制整个系统会使主节点及从节点间的负载不均衡。使得多核处理器系统无法充分利用资源，因此，他指出，可以将微内核操作系统和一些软件模块组合起来，微内核操作系统维护这些软件模块，这样将它们各自的功能分离开来，使得它非常适合在分布式系统中使用。在功能划分中，内核和软件模块控制的系统功能不同，每个核上有一个基本微内核功能，其它功能则在对应的软件模块中执行。系统中的应用程序则通过系统调用的方式访问微内核操作系统，如果它无法处理，则有将消息发送给软件模块，用软件模块来处理这个请求。这种方式设计的操作系统将多核功能分配到各个软件模块执行，非常方便，但是也存在一定的问题，就是文件系统模块会大量的接收系统消息，会导致文件系统间的节点通信成为热点，会导致性能问题。

然而，在国内针对多核操作系统的研究尚不多，主要是一些学术机构和科研院所在研究，同时研究的相对较少，发展比较慢，技术还不够成熟，这和当今蓬勃发展的多核处理器技术还有差距。因此国内的相关学者和研究人员应逐渐转向这方面的研究。

在 2008 年，一个嵌入式实时操作系统 SmartOSEK OS-M 由浙江大学的研究人员设计和设计出来，这款操作系统支持多核异构的系统，这样多个处理器核之间可以相互通信和交互，同时他们在这种多核操作系统上设计了高效的编程模型。西安交通大学和复旦大学及麻省理工大学在同年也实现了一款多核操作系统 Corey，它的设计思想是由应用程序自动控制共享数据，由应用程序自主选择多核资源及共享资源，这样可以大量减少多核处理器的资源传送及更新，可以高效的利用多个内核的资源。经过实践检验，这种方式对系统的整体性能提升效果非常明显，几乎可以提升四分之一的性能。在 2009 年，电子科技大学设计和实现了操作系统 aCoral，它架构在四核处理器上，同时改进了任务的调度方式，使用位图映射的方式，将系统的复杂度控制在一个常量内，这样充分保证了多核处理器系统的实时性能。

综上所述，如何提高多核操作系统的实时性能和吞吐量已成为研究的热点与重点。而现有的嵌入式操作系统微内核任务调度机制存在负载不均衡，调度时间长等问题，难以适应高实时性、高稳定性和低延时的嵌入式系统的需要。因此研究并设计一套新的多核环境下嵌入式操作系统微内核任务调度机制以保证系统的负载平衡及最少完成时间对于嵌入式系统的发展具有极其重要的意义。

1.3 本论文主要内容

多核处理器如今已经走入了我们的生活，它也是未来的重要发展方向，其上搭载的

嵌入式微内核操作系统体积小、性能卓越，是未来操作系统向小型化、轻量化发展的重要方向，近几年多核处理器及微内核操作系统得到了众多科研人员的密切关注，与此同时也获得了迅猛的发展。嵌入式微内核操作系统中的任务都必须通过任务调度机制和策略来分配执行，因此微内核任务调度机制性能的好坏是多核环境下嵌入式微内核操作系统性能的好坏的决定因素，由此分析，本文对多核处理器环境下的嵌入式操作系统微内核的任务调度机制进行了改进。本文主要研究内容如下：

（1）对多核处理器的架构、微内核操作系统以及流行的嵌入式微内核操作系统进行了研究，分析了当前主流的的多核嵌入式微内核操作系统的任务调度机制并进行了比较分析。在对多核处理器下嵌入式微内核操作系统的任务调度机制进行改进之前，我们必须要先对多核处理器架构和微内核操作系统的特点有一个详细的了解，同时要对当前主流的任务调度机制有清晰的认识，这样我们才能够对多核嵌入式微内核操作系统的发展有个整体上的思路，才能够改进出符合当前多核嵌入式微内核操作系统发展方向的任务调度机制。基于以上基础，进一步分析主流的任务调度机制的优点及不足。这些分析结果都将是多核嵌入式微内核操作系统任务调度机制改进的理论依据，为改进的机制和策略提供了有力的理论基础。

（2）对多核处理器环境下嵌入式微内核操作系统的任务调度机制进行了改进。多核嵌入式微内核操作系统的任务调度机制主要包括两个方面：任务分配模型、任务调度机制。任务分配模型的好坏直接影响任务的执行次序及系统的实时性，因为当高优先级任务到来时，如果分配的位置不对，就必然会影响任务执行的实时性。因此本文提出了基于优先级的抢占式任务分配机制，使高优先级的任务能够实时快速执行，同时给出了多核处理器下任务分配的通用规则。接着用全局队列调度和局部队列调度相结合的方法实现了任务的调度，提出了基于缓存队列的任务调度机制，为了使处理器的效率更高，对全局队列调度进行了改进，融合了快内核优先的动态迁移策略。这些都保证了系统的实时性和负载均衡性。

（3）通过在本文改进的任务分配模型和任务调度机制基础上，对微内核操作系统中与任务调度机制相关的部分进行了改进，然后在 Simics 仿真平台上模拟了多核处理器硬件平台和操作系统，并通过测试程序进行了对比测试。性能测试实验结果表明，本文设计实现的多核处理器下嵌入式操作系统微内核的任务分配模型和任务调度机制提高了系统的效率和实时性，使多核处理器负载均衡，达到了本文的研究目标。

1.4 论文章节安排

本文的各章节主要内容如下：

第一章：介绍了本文的研究背景和研究意义，阐述了多核处理器和嵌入式微内核操

作系统的发展历程以及发展状况，分析了国内外在多核嵌入式微内核操作系统任务调度机制方面的研究状况。在本章的最后介绍了本文的主要研究内容和各章节内容的安排。

第二章：对多核处理器的概念及架构做了简单的介绍，描述了微内核操作系统的概念和其优点，然后介绍了常见的微内核操作系统的整个发展历程及其特性，并对当前最成功的嵌入式微内核操作系统 Minix3 和 QNX 进行了简单的介绍。

第三章：介绍了多核处理器任务调度的基本原理及通信方法，分析了当前主流的多核处理器静态任务调度算法和动态任务调度算法，并对其优缺点进行了比较分析，接着分析了单核处理器和多核处理器中的主要任务调度机制和策略的优缺点，最后给出了多核处理器任务调度机制的评价标准。

第四章：本章是全文的重点章节，本章对通过两方面对任务调度机制进行了改进。分别对任务的分配模型和任务的调度机制进行了改进。提出了基于优先级的抢占式任务分配机制，给出了多核处理器环境下任务分配的通用规则，接着将全局队列调度和局部队列调度想结合，提出了基于缓存队列的任务调度机制，同时为了进一步提高内核的利用率，给出了快内核优先的任务动态迁移策略。

第五章：根据第四章的任务分配模型和任务调度机制的改进思想，本章首先介绍了功能强大的 Simics 仿真平台及其使用，接着通过修改 Simics 配置文件搭建了本文所需的多核处理器环境，接着将改进方法加入到仿真平台中，最后通过仿真实验实现了本文设计的任务调度机制，同时进行了性能对比测试。并对性能测试结果进行了详细的分析。

第六章：描述了本文作者所做的主要工作，总结了本文的主要内容，并对以后的进一步工作进行了展望。

第 2 章 多核处理器和微内核操作系统的技术概况

在多核环境下，嵌入式操作系统微内核的任务调度机制性能的好坏直接影响到了系统的效率，直接决定了嵌入式操作系统微内核的性能的高低。因此，在了解任务调度机制之前，我们必须要先对多核处理器架构和微内核操作系统的特点有一个详细的了解。本章着重介绍了多核处理器的概念和架构，微内核操作系统的概念和优点，并对几代微内核操作系统的发展和特点进行了简单的介绍。

2.1 多核处理器技术及其架构

2.1.1 多核处理器的概念

随着单核处理器出现的性能瓶颈，并伴有许多的局限性，多核处理器的概率也就应运而生了，多核处理器的性能大大提高，能很好的满足用户的需求。同时它在系统的安全性、虚拟技术等方面表现的非常出色，给用户提供了更快、更高效、更安全的体验，提高了人们的工作效率。

多核处理器是指将多个功能完整的计算内核集成在处理器上，使其能并行处理任务。多核处理器技术的发展是由当前处理器技术发展的瓶颈而产生的，因为单芯片的处理能力已经达到了上限，提高主频使得芯片的热量非常大，性能无法再提升和改善了，因此研究人员和工程师们认为如果继续提高芯片主频，那么那热量足以烧坏芯片了，其制作工艺也会非常复杂，性价比非常低，用户很难接受。

Intel 的工程师及研究人员开发并实现了多款多核处理器芯片，它通过并行的使用各个内核，提供处理器的整体性能，对多个任务采取了分而治之的方法。将任务划分出来，分别分派给各个内核执行，并且在同一时间内可以执行非常多的任务。多核处理器是一个芯片，它能放入处理器插槽中，使操作系统可以充分的利用这些多核资源，从逻辑上看，处理器的各个部分是独立的内核，能并行的执行。多个核获得任务后可以在各自的时钟周期内执行，这种架构方式使得应用程序运行效率大大提高，同时也给出了一个非常完善的软件编程模型供开发人员使用，目前很多软件厂商及设计人员都在研究和探索更好的任务并发出来模型。随着科技的发展及多核技术的成熟，现有的应用程序可以顺利的迁移到多核平台上。操作系统也需要满足多核处理器设计，能平滑的迁移到多处理器上，为了更好的利用多核处理器技术，应用程序设计中应该融合各种思想和方法，其设计流程与多核处理器保持一致，单线程和多线程都能同时运行。多线程技术能在多核处理器上表现出优越的性能，并且拥有可扩展性。多核处理器技术使得系统能并行的处理多个任务，以前都使用多个处理器，现在的多核系统可以很方便的扩充，能在外面融

入许多强大的功能，这种方式使的功耗更加低，处理器产生的热量也非常少，非常实用。有此可知，多核处理器技术的未来发展的必然趋势，是推动处理器处理速度的关键因素。促进多核处理器技术发展的主要原因是半导体工艺的高速发展和设计方法的成熟。由摩尔定律可知，每当半导体技术进步与发展一步，处理器的处理方式也会发生变化，这样新的研究领域就出现了。设计方法的成熟又推动了新的半导体工艺的发展，性能得到很大的提高。这些因素都是相互影响和相互推动的，每当工艺技术提高 10 倍，那么处理器的体系结构和编译技术都会大幅度的提高。科技飞速发展的如今，这种规律也在动态的变化着，总的来说多核技术的发展与应用是技术发展的必然选择和需求。

人们采用多核处理器能体验高性能的计算机，并且能够获得非常合适的主频，每个时钟周期内多核处理器能运行多个内核有效单元，内核之间拥有高效的通信机制，这使得系统的性能达到了最大，主频达到了最佳。

影响这些的因素主要有以下四点：

第一、由摩尔定律可知，单核处理器不能无限提升主频，而集成电路、晶体管等生产工艺的发展，其工艺水准达到 90nm，那么多核处理器的芯片能发挥其重要的性能。

第二、随着多核处理器核与核之间的通信技术逐步发展起来后，核间的通信标准逐渐规范和统一了，它们相互协作、具有高效的通信机制，同时线程与线程间的通信延迟也越来越小。

第三、多核处理器的能耗非常小，和传统的单核处理器相比，其功耗小很多。当多核处理器与单核处理器达到同一主频时，单核处理器的发热量一定非常大，它将会消耗非常大的功率，同时发出巨大的热量，要是没有足够有效的冷却措施，比如风扇等，处理器就不能正常稳定保持工作。然而多核处理器的芯片针对这类问题做了非常大的优化，在功耗非常低，性能也表现的非常卓越。

2.1.2 多核处理器的架构

多核处理器的架构可以分为同构多核处理器和异构多核处理器，同构多核是指计算机的内核相同，地位对等，而异构多核处理器的内核地位是不对等的，它们之间的功能和作用差异很大，各个内核发挥自己最擅长的能力来提高整个系统的性能，异构多核处理器一般采用“主从式”设计思想，他给多核的负载均衡和资源如何管理带来了新的解决方案，它的主要设计思想是当系统启动时，一个主内核首先启动，它负责整个系统的初始化以及管理其它内核。在系统运行后，只有主内核可以访问整个系统的资源，其它内核只能通过主内核调度来访问系统资源。异构多核处理器的另一种系统模型是每个处理器里面都有一个操作系统，这样能更好的支持内核的异构特点。既然异构多核处理器有如此多的优点，能极大的提高系统的性能，目前许多研究者都开始研究着手异构多核处理器架构，它是未来的多核处理器技术的发展趋势，因此，本论文的的重点也是研究

异构多核处理器环境下的操作系统微内核的任务调度机制。

多核处理器架构的多个处理器间不共享前端，多个处理器对内存中的地址和资源的访问不会产生影响，那么这种架构方式的带宽非常大，核与核之间的通信效率非常高，性能非常高，其架构方式如图 2.1 所示，各个内核有自己的一级缓存和二级缓存。

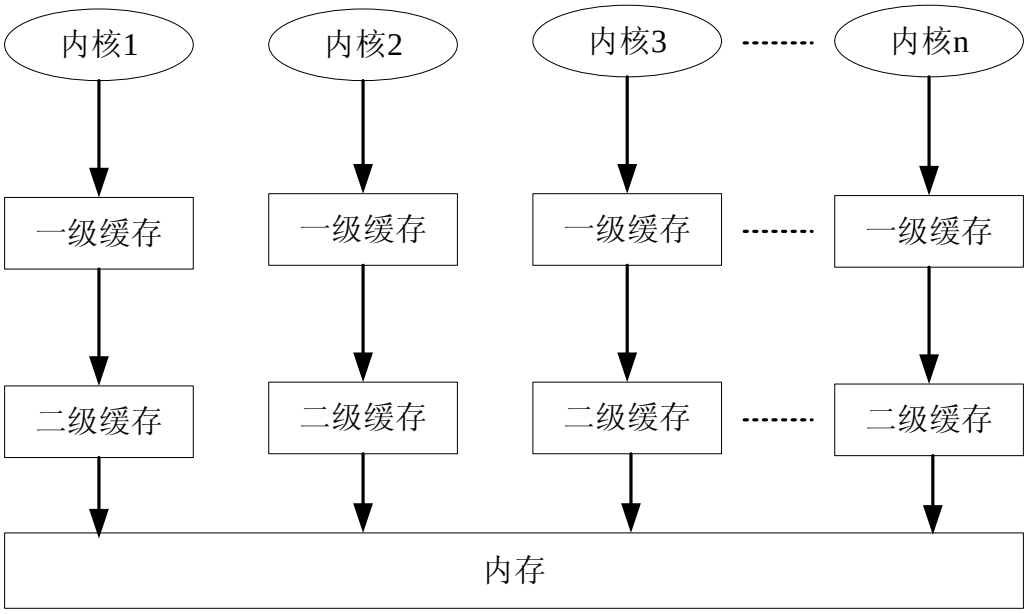


图 2.1 多核处理器架构

2.2 微内核操作系统概述

2.2.1 微内核操作系统

Richard Rashid 是最早提出了微内核操作系统概念的人，基于微内核的理念，他带领卡内基梅隆大学研究员一起研究并开发第一代微内核操作系统---Mach，目的是开发一个非常小的内核，它是利用消息传送机制来实现的。操作系统的关键核心是微内核，微内核中使用了系统中必备的功能模块，比如线程管理，地址空间控制，线程间通信一些必要功能，其余的非必要功能模块则从系统中分离出来，由用户层来实现，那么微内核的结构就简单多了，复杂度降低了不少。这样的设计使得微内核操作系统比传统的内核操作系统更加灵活，有着更好的扩展性，因此，微内核操作系统是现代操作系统发展的主流趋势。

微内核操作系统提供了系统所必须的基本服务，其它服务由用户层提供，包含文件系统、网络服务及设备驱动等功能模块，当系统启动后，这些服务也随之加入进来了，微内核与这些服务是通过消息传递机制来实现通信的，它就是各种服务间的桥梁，紧紧的把他们联系在了一起，它们一起工作就构成了操作系统。其它应用程序可以顺利的在这个系统上运行起来。

在内核态下，微内核直接访问硬件，用户态下，各个功能模块被分为独立的进程，

每个进程提供单独的服务，这种服务器模型将多个进程联系在一起，相互协同工作，这样用户可以享用各种类型的服务。服务器通过相应的接口与调用微内核，各个模块之间相互独立，通过消息机制进行通信，从而保证了系统能对硬件的完整访问。

微内核与传统的宏内核相比，其具有更灵活的配置方式和扩展性，宏内核操作系统同样也给用户提供了了各种各样的服务，它管理着整个电脑的硬件资源和软件资源，全局控制着系统的运行。系统的所有功能都融为一体了，因此它能更快的响应用户的请求，系统开销也很小，但是它的这种设计也带来了不少问题，由于它的所有功能模块都整合在了一起，每个模块间的耦合性很大，每个模块不能单独的使用和运行，研究人员很难对系统功能进行增加、修改或者删除操作，具有很差的扩展性及可升级性。而微内核则恰恰相反，只需要增加相应的服务模块，其它的地方不需要修改便可以直接运行了。

微内核操作系统的体系结构如图 2.2 所示。

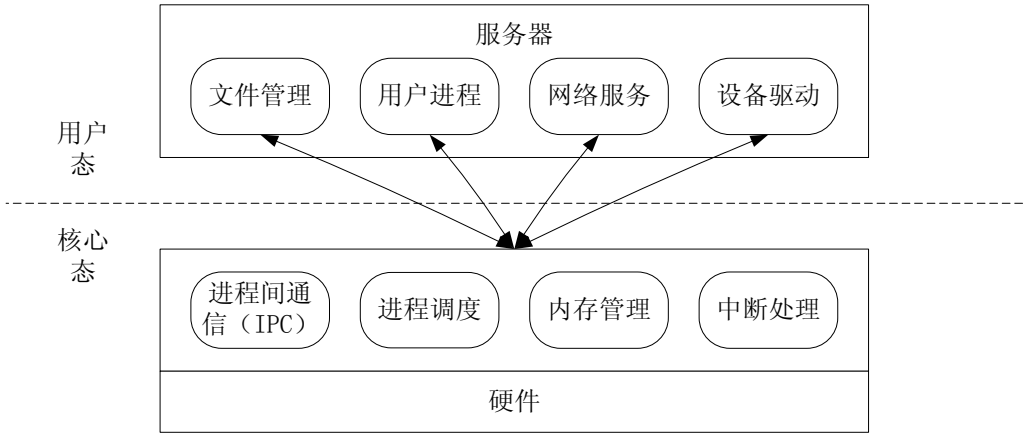


图 2.2 微内核操作系统的体系结构

2.2.2 微内核操作系统的优点

微内核操作系统的架构方式有着非常多的优点，这些特点也非常适合嵌入式环境，同时，它又通过模块化的设计思想，将操作系统的复杂性降低到了最低，其原则就是要使内核的结构绝对的简单，不把非必要的模块放到内核中，因此微内核操作系统具有如下优点：

一、可扩展性。微内核操作系统有着高度的动态可扩展性，即操作系统新增加的服务或功能模块在不用修改内核的情况下，能快速的安装到操作系统上，操作系统像拥有多个插槽一样，每放一个他遍自动适应了系统，然后通过微内核预设的消息通信机制与其它模块进行通信。由于微内核的设计使得所有的服务在编译后都拥有独立的二进制可执行代码，那么增加、修改和删除服务都不会对微内核操作系统的功能有任何影响。

二、可配置性。微内核操作系统可以根据具体应用的具体需求进行定制，它能迅速的适应各种要求，并立刻配置完成。目前 Linux 操作系统的内核也有 make config 工具对特定的应用进行编译配置，各个模块在内核中运行，当应用程序的功能变化后，则需

要从新编译配置内核，而微内核操作系统没有这些问题，能很快的适应系统功能的变化。因此，它的这个特性很适合需要根据各种环境配置的嵌入式系统。

三、可靠性。传统操作系统的内核设计中，将所有的服务整合在了一起，一旦某一个服务出错了，那么整个系统就会崩溃。然而，微内核的设计思想是使各个模块的耦合性降到最低，内核中只存放系统必要的服务，而把其它具体的服务实现放到了用户态中执行，这样内核崩溃的概率就大大的降低了。目前操作系统错误大多出在驱动程序的稳定性上，一旦错误执行，整个系统就崩溃了。因此，微内核的设计让系统的整体可靠性有了很大的提升。

四、健壮性。微内核操作系统的健壮性非常强，因为传统的操作系统内核的结构非常复杂，代码量大，可维护性非常差，错误也比较多，系统相对比较脆弱，容易崩溃。而微内核操作系统的所有服务模块都在自己的地址空间中运行，那么，即使某个模块出错了，也不会影响整个系统进程，因此微内核操作系统比传统的操作系统更加健壮。

这些是微内核操作系统优点的几个主要方面，当然，微内核操作系统的架构并不是完满无缺的，它最大的缺点就是性能，传统的操作系统内核是通过系统调用的方式调用的，只需要在内核态和用户态进行转换，每个进程有自己的用户空间和内核栈，能存放系统执行的整个过程，然而微内核操作系统却需要通过消息机制来传递各种信息，服务器完成请求再传回数据信息，那么系统进程间会有上下文切换的过程，并行系统内部没有内核栈来缓存信息，因此在消息传递时会消耗较大的拷贝信息的开销，性能就受到了很大的影响。最初研发的 Mach 系统的性能是非常糟糕的，让人很难接受，因此有待提高微内核系统的服务性能。提升系统任务调度的效率。

2.2 常见的微内核操作系统介绍

电脑发展的最初阶段，操作系统的内核都是非常小的，随着对电脑功能的需求越来越多，同时软件和硬件技术的高速发展，使得操作系统的内核越来越大，BSD Unix 带来了大内核时代，然而内核的结构越来越大越来越复杂使得 Bug 也越来越多，不方便用户对其进行管理。伴随着微内核技术的产生，它很好的解决了这个问题，很多研究人员都从事到微内核操作系统的研发工作中来，这样许多性能高效的微内核操作系统便诞生了，这些系统都有着各自的优点，但也都存在不少缺点，下面介绍常见的微内核操作系统。

2.2.1 第一代微内核操作系统 Mach

最早的微内核操作系统是 Mach，它是卡耐基梅隆大学的 Richard Rashid 教授为了研究操作系统，主要是分散与平行运算，对 BSD Unix 改进而设计出的操作系统，Mach

替代了早期的 Unix 系统。早期的 Unix 系统设计原则是一切皆文件，其扩展性非常差，系统运行效率不高，因此 Mach 的设计试图改变这些弊端，其设计理念主要有以下几点：

一、任务。即拥有一组系统资源的对象，它允许“线程”在其中执行。

二、线程。是执行的基本单位，拥有一个任务的上下文，并且共享任务中的资源。

三、port。是任务间通讯的一组受保护的消息队列，任务可以对任何 port 发送或接收数据。

四、消息。是某些有类型的数据对象的集合，它们只可以发送至 port 而非某特定任务或线程。

当然 Mach 系统的缺点是是进程间的通信机制。Mach 采用 port 机制实现进程间通信，但是这种机制被证明是导致系统性能低下的主要原因，在后续的微内核操作系统开发过程中，研究人员普遍摒弃了这种机制。

2.2.2 第二代微内核操作系统 L4

L4 微内核操作系统是约亨·李德克在 L3 的基础上研究设计的第二代微内核操作系统。由于意识到 Mach 微内核操作系统在设计和性能上的缺陷，约亨·李德克对第一代微内核进行了细致的分析，并希望证明存在更加轻量级的进程间通信机制，并且性能更高，平台无关系更强，更加贴近现实世界中的性能改进。L4 的运行时间比 Mach 提升了 20 倍，同时，L4 也在许多方面对 L3 进行了改进，进一步完善了系统。L4 提供了两种抽象：线程和地址空间，同时也提供了两种机制：映射机制和进程间通信策略。因此，L4 的设计方法也被大多数学者所接受，在后续的设计中争相采用了这种设计理念，出现了很多微内核操作系统。

2.2.3 第三代微内核操作系统 seL4

seL4 是一组基于微内核构架的操作系统内核，他是由澳大利亚研究组织 NICTA 研发的一个新的 L4 版本，名叫 Secure Embedded L4（简写 seL4），同时，他们宣布他们在世界上率先开发出第一个形式化机器证明的（formal machine-checked proof）通用操作系统。

它提供了最小，高效的软件水平级别，它是微内核系统的主要部分，它可以在硬件层享有优先执行的权利，它是在微内核操作系统 L4 架构上发展起来的第三代微内核操作系统，它拥有非常小的体积、高效的性能和很大的自由度。同时它内建了高性能的模型，在微内核和应用程序级别，这个模型提供了一种机制来增强系统的安全保证。seL4 系统的重点是在系统的安全性和和健壮性上做了很大的提升。

2.2.4 其它微内核操作系统

随着软件和硬件技术的发展，微内核操作系统后期的发展方向主要是围绕微内核性能的提升和结构的优化方向发展。其中非常成功的微内核操作系统有 Minix3 和 QNX 等。

Minix3 操作系统是一个免费的、开源的系统，它被设计的高可靠性、高灵活性和高安全性。它是基于微内核运行在内核态，其它的功能作为一个固定的、受保护的进程运行在用户态。它有这些特征 NetBSD 用户空间遵循 POSIX 标准，开源的并且拥有 BSD 许可证，同时符合 TCP/IP 网络协议，语言采用了 C、Perl、Python 等编写，系统只有几千行代码编写，方便研究人员进行代码移植及研究工作。Minix3 的设计思想完全是借鉴于微内核的，采用模块化的方式组织内核，而在服务器端，则放着文件系统、存储管理等功能。

由于 Minix3 操作系统是采用了全新的设计方法，重新构建了一个新型的内核结构，使得微内核操作系统更加可靠、安全，同时也加强了系统的模块化。研究表明，每一千行代码中就可能存在有 6~16 个错误。而 Minix3 的内核仅仅包含不到 4000 行的代码，这就减少了系统内核出现错误的几率，增加了系统的稳定性。

QNX 是一个遵守 POSIX 标准的、支持多任务的、抢占式的实时微内核操作系统。它是面向嵌入式系统的。其微内核核心只提供 4 种基础服务：进程调度、进程间通信、底层网络通信和中断处理等，它的进程在独立的地址空间运行，驱动程序、网络协议和应用程序处于用户空间中。所有其它 OS 服务，都实现为协作的用户进程，因此 QNX 的内核体积非常小巧（约为 12Kb），而且它的运行速度极快。

QNX 所采用的这些机制使得它的的运行速度非常的快，它也是 Unix 系统的同胞异构体，和 Unix 系统有着极高的相似性，同时，QNX 系统上不会存在计算机病毒，和 Linux 系统一样，这就不存在系统的文件资料被病毒破坏的危险，QNX 的用户权限管理也非常严格，任何未正确登录的用户是无法访问系统的，这在很大程度上提高了系统的安全性，而这些特点也是其他的操作系统所不具备的。

QNX 的任务调度策略采用的是 POSIX 标准的进程调度方式，拥有 255 个进程优先级，抢占式策略和基于优先级的正文切换方式，同时用户也可以根据自己的需求来选择调度其它调度方式，比如先进先出策略、时间片轮转策略及系统自适应调度策略等。

QNX 给用户带来了前所未有的稳定性、安全性以及良好的兼容性。QNX 优越的性能使得它深入到了在生活的方方面面，尤其在汽车行业，市场份额达 75%以上，在手机通信行业，黑莓手机也开始采用 QNX 系统作为其手机系统。由此可见，QNX 是目前最成功的微内核操作系统。

2.3 本章小结

本章简单阐述了多核处理器的概念和架构及微内核操作系统的概念和优点，接着详细介绍了第一代、第二代和第三代微内核操作系统的发展和特点，最后介绍了当前最成功的微内核操作系统 Minix3 和 QNX 的特性及优缺点，为后续章节分析嵌入式操作系统微内核的任务调度机制提供了一些理论基础。

第 3 章 多核处理器下微内核任务调度机制的分析

通过前文对多核处理器和微内核操作系统的研究分析，多核处理器是提高计算机性能的重要组成部分，它起着非常重要的作用。微内核操作系统从第一代、第二代到第三代以及到发展到现在流行的微内核操作系统的过程中，性能都有了极大的提升，其中，进程间的通信机制和任务的调度机制的好坏是阻碍系统效率提高的关键因素。由此可见，任务调度机制的优劣对于微内核操作系统的重要性。多核环境下，研究任务调度机制有利于发挥多核处理器的最大性能。如今，随着硬件技术和软件技术的发展，多核架构的电子产品已经真实的出现在我们眼前，如智能手机、平板电脑、台式电脑等等，这些设备上的操作系统必须支持多核体系结构的架构，因此，我们要设计一种既能实现系统任务的高效调度，同时要保证嵌入式操作系统的微内核能在多核架构的硬件环境下运行。

但是，在设计一种任务调度机制之前，我们必须要了解多核处理器下微内核的任务调度机制的基本原理以及多核处理器任务调度算。本章 分析研究了多核处理器任务调度的基本原理，然后分析了多核处理器的任务调度算法的优缺点，并对现有的任务调度机制和策略进行了对比分析，最后给出了多核处理器任务调度机制的评价标准，并对如何设计一种高效的任务调度机制进行了分析。

3.1 多核处理器任务调度基本原理

微内核操作系统中的任务是指系统它完成了一个具体的操作，任务是可以用进程或线程来描述。程序的执行过程叫做进程，它是一个数据集合的运行过程，在内存中有完整的数据空间及代码空间，操作系统进行资源分配和调度的基本单位是进程。

线程是进程中单一的控制流，也被称作轻量级进程，它是存在于进程之中的，一个或多个线程构成了进程，一个进程内的所有线程共享对应的全局数据，同时，每个线程又有自己的计数器、寄存器、栈等基本功能，它是比进程更加小的能独立运行的独立单位，同时也是操作系统调度的基本单位。

目前，大多数操作系统都是以进程为资源分配的基本单位的，虽然进程和线程他们之间有着区别，但在各类处理器任务调度中，它们都是运行的基本单位，本文将进程和线程统称为任务。

任何操作系统都有任务调度功能，它是最基础的功能模块，它主要负责将任务分配给处理器，提高任务的响应时间、提高处理器利用率及防止系统进入饿死状态或者死锁状态。任务的调度其实主要是对处理器进行管理，管理处理器的所有资源。

任务调度的功能实现主要是任务调度程序或算法完成的，调度程序或算法主要将待分配的任务按照算法规则分给处理器，然后进行上下文切换，最后再返回到用户程序。调度程序应该短小并且效率高，同时要减少调度延时。

由于多核处理器中的多个任务也需要交换数据，即时通讯，同时任务间需要协同工作，为了提高系统的整体性能，我们需要对多核处理器的进程基本原理、任务的调度和通信过程做深入的研究。

3.1.1 多核处理器进程原理

由操作系统的基本原理可知，程序是一段静止的代码，是大量的指令通过一定的逻辑组成的，它和进程的不同之处是程序是一个静止概念，而进程是动态的，那么我们可以把程序的一次完整执行过程看作是进程，那么有了进程这个概念，我们可以把整个程序的所有模块分给多个进程，这些进程同时工作，协同完成这个任务，可见，并发程序执行的效率要远远高于顺序执行的程序。

进程是人们抽象出来的一个概念，是程序执行一次的内存映像，它主要由进程控制块、正文段及数据块组成。一个进程有唯一的进程控制块，它能唯一标识进程块，这样操作系统就可以根据这个标识知道进程的存在，然后可以对其进行管理和控制。正文段是用于共享的重入码，而数据段是进程运行时所使用的数据。进程有三种基本状态，即就绪状态、运行状态和阻塞状态，如图 3.1 所示：

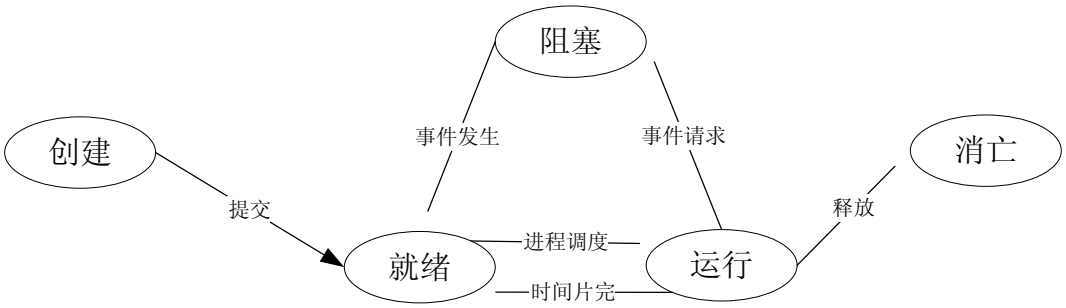


图 3.1 进程基本状态转换图

操作系统通常都提供创建进程的系统调用，创建进程时，操作系统给每个进程生成了唯一的标识符和控制块，它记录了进程的所有属性，如：进程标识符、运行状态、现场区及优先级等一系列基本属性信息，一个进程可以创建多个子进程，而新生成的进程又能创建新的子进程。

进程是系统资源调度的最基本单位。在多个处理器上同时运行多个任务，操作系统的微内核负责所有任务的调度工作，让每个进程都有自己的时间片，同时可以快速切换到另外一个进程。微内核根据系统的调度算法来决定任务在哪个处理器上运行，并负责管理整个系统中的所有任务。

由于多个任务在同时执行，那么它们之间存在信息的交换，同时也会存在公共的共

享资源，那么进程间会有同步和互斥机制，它是最原始的进程间通信方式，互斥是指某一个共享资源在某一个时间点只允许一个访问者对其进行访问，它具有唯一性和排它性。访问进程对互斥资源的访问是没有顺序的，是根据进程的优先级进行排它访问的。而同步是指在互斥的情况下，进程对资源的访问有序访问，并且对于写入资源一定是互斥的，进程访问资源可以允许多个人同时访问。

同步和互斥机制在进程访问中具有非常重要的意义，多个进程要保持相互同步，那么它们会同时对一个文件执行写操作，而对于一个文件，在某一个时间内只让一个进程进行写入操作，以防止文件因不同步而产生不可预料的错误。目前，已经有的同步方法有互斥锁、临界区、信号量及事件等。

进程在执行的过程中还存在僵死进程。它是指出现多个进程时，子进程退出后要对它的相关信息进行清除，如果一个进程在退出后未清空进程表中对应的信息，那么其父进程在查找子进程时，会发现子进程的信息还存在，但它有无法执行，那么这样的进程就是僵死进程。对于长期运行的服务器来说，每个僵死的进程都会占用一定的空间和进程表空间，那么随着时间的推移，僵死进程越来越多，会耗费大量的资源，直到资源耗尽时，其它新进程便无法创建运行了。因此，使用专门的方法来清理这些僵死进程。

3.1.2 多核处理器任务的调度

多核处理器的任务调度是由调度器来决定的，它决定分配时间片给进程，根据进程拥有的优先级来确定谁先运行，优先级高的会多次切换运行，优先级低的任务就切换比较少，进程运行在分配给他的时间片里，任务大多数采用的抢先式多任务处理，任务的挂起及运行无需彼此协作。

进程一旦拥有了 CPU 及所需要的资源，那么系统调度机制就可以使进程运行起来，这时，它会选择进程池中最有可能运行的进程进入运行态，调度程序选择每个进程主要是依据如下几个策略：

1、进程的调度策略和机制。主要是对实时进程及普通进程进行区分，以期获得最大的进程相应和系统最大利用率。调度策略就是在寻找最佳的解决方案。

2、进程调度的优先级。通过对进程的运行次序分级，可以满足进程对价值和处理器时间的需求。调度程序一定是选择时间片未用完的进程，同时它的优先级也是最高的。

3、剩下的时间片。它是进程的静态优先级的决定因素之一。它用于计算进程运行到什么状态有着重要作用，因此，它也决定了进程运行的动态优先级。

4、实时进程的优先级。是用于实时进程间选择和调度的，是实时进程所特有的调度方式。

任务调度算法可以知道所有实时程序的状态，目前，多核处理器下任务的调度主要需要解决的问题有：在什么情况下使用进程抢占策略；如何确定时间片的大小，它对系

统的性能影响非常大，这也非常关键；如何选择高效的调度机制和策略来提供系统的性能等。

目前，普遍采用的进程抢占策略是当一个新进程来的时候，它会有一个动态优先级，内核会检查当前正在运行的进程的优先级，如果新进程的优先级大于当前正在执行的进程，那么当前进程被中断，新进程抢占时间片并运行起来，当其执行完毕再唤醒被中断的进程。这种方式存在的问题是，如果当前进程马上就要执行完了，这时却中断了它，等新进程完了再调度这个进程执行，这样系统的开销就会比较大，花费了大量的调度时间。因此，这种方式需要进一步的优化。

时间片的长短选择非常重要，如果时间片分配的太短，那么进程间就会频繁的切换，系统因此引起的额外开销非常大，比如，分配给进程的时间片是 1ms，进程切换的时间也是 1ms，这样 CPU 就要将一半以上的时间用在进程切换上，这样就额外的耗费的大量的系统时间。如果时间片分配的太大，进程的执行就不像是在并发执行，一个进程运行好长时间后，另一进程才有机会运行，比如，每个进程分配的时间片是 10s，这样每个进程就要执行大概 10s，执行完后，需要很长时间才能再次获得时间片。这样任务执行的效率就会降低很多。当然，如果分配的时间片很长时，系统允许交互式进程抢占机制的话，那么系统的效率会大大的提高。

3.1.3 多核处理器任务间的通信

多核处理器任务间通信的主要方式有管道机制、信号、消息队列、共享内存、信号量机制和套接字，任务间通信即进程间的通信，它是指多个进程以及多个线程相互间通信，每种进程间通信方法都有其擅长的功能和弱点，进程间通信允许用户态的进程执行以下操作，如通过信号量和其它进程进行通信、给其它进程发送消息或者接收其它进程发送过来的消息及和其它进程共享一段内存块。下面分别介绍多核处理器任务间的主要通信方式：

1、管道和命名管道

管道（pipe）是进程间通信的一种方式，它的数据流是单向的，一个进程写入管道的所有内容，通过内核的调度而使其流向另外一个进程，另外一个进程就可以在管道中读取传过来的数据。管道非常适合进程间满足生产者/消费者关系模式的交互。管道是一种类似打开的文件，在系统中无法看到其表现形式，当需要创建新的管道时，我们可以用 pipe()系统调用来实现，它会返回一些文件描述符，进程再通过 fork()系统调用将文件描述符传给它的子进程，那么它的子进程和它共享管道。这时进程使用 read()调用可以从管道中读取信息，同时也可以用 write()系统调用向管道中写入数据。

有此可见，管道通信机制非常的方便有效，且十分简单和灵活，但是它的缺点也十分明显，就是不知道打开已经存在的管道，那么任意两个进程就没法共享一个管道，除

非它们是从一个共同的父进程产生的子进程。在 Linux 操作系统中使用了命名管道(FIFO)机制，它是代表“先进先出”，命名管道一旦创建，所有进程就可以用系统调用访问它，这样就方便所有进程间相互传递数据。

2、信号

信号是一种很短的消息，它可以在进程间进行传递，可以发送给单个进程或多个进程，每个信号都有一个唯一的标识，它通常都是一个常数，使用信号的主要目的有两个，一个是通知进程已经发生了某一个的事件，二是通知进程执行信号处理程序。信号的主要任务不是传送数据，而是通知特定的事件要发生，如果进程不需要此信息，便可以忽略这个信号。当用户进程接收到信号后，可使进程执行特定的信号处理程序，这段代码是一定会执行的，有时，用户不希望程序马上执行，信号可以延迟传送给进程，那么这时可以先阻塞信号。

3、消息队列

消息队列提供了一种任何进程间通信和传送数据的方法，它非常简单和有效，它位于内核地址空间的一段链表上，多个进程可以通过这个公共的消息来交换数据。使用此队列的进程可以发送消息及接收消息，每个消息队列都有唯一的 IPC 标识。消息队列有如下优点：有基于文件的应用接口、支持消息到达的异步通知、支持消息间的优先级及用于阻塞消息发送和接收消息操作的超时机制等等。

4、共享内存

共享内存即是任何不同的进程可以访问同一个逻辑内存区。这是一种非常有用的通信机制，它是多个进程把公共数据结构及信息放入到一个共享内存区，其它进程都可以访问这段信息。那么在进程的地址空间中加入新的内存区，并将它与共享内存中对应的区域映射出来，共享内存通信机制有如下优点：访问速度非常快、可以指定共享区域数据空间大小及自定义数据区的结构。

5、信号量

信号量主要是为了解决多个线程同时访问共享资源时产生的问题，对资源的同步与互斥访问。信号量主要有两种情形，一个是互斥，它是防止多个进程在访问同一共享资源时发生数据不一致性的，通常采用 P，V 操作原语来实现的；另一个是同步，即常见的生产者与消费者问题，当消费者进程开始执行时，它会执行一个 P 操作，表示此进程要进入关键区域。当生产者进程生产出数据后，将执行 V 操作，此时，消费进程可以处理这些数据。

6、套接字

套接字通信方式是一种非常有用的机制，它和前面的几种通信方式不一样，前面的都是在单台计算机上通信，而套接字方式可以在多台计算机之间进行通信。从程序进程中向外界发送的一条数据时，数据信息会被自动的划分为很小的数据段进行传输，这些数据段中包含不少控制信息，包括发给哪个计算机的哪个端口，还有数据校验等等。其

它计算机的进程接收到数据之后，再将其解析为对应的数据，套接字机制很好的解决了多计算机及多个进程之间的通信问题，为计算机之间的交互和资源共享提供了有利条件。

3.2 多核处理器的任务调度算法

多核处理器任务调度是指将系统中运行的多个任务调度分配到多个处理器上，使得处理器相互之间的负载平衡，提高所有内核的资源利用率，使系统的执行任务的时间最短，并使系统的吞吐量最大化。因此，任务调度算法和策略的选择对系统最终执行的效果好坏有着直接的影响，任务调度算法和策略对于提高系统的并行计算能力有着非常重要的作用。目前比较流行的调度方式有静态任务调度和动态任务调度机制，下面将分别介绍这两种机制。

3.2.1 静态任务调度算法

多核处理器静态任务调度算法就是指系统中的任务在运行前就知道如何分配任务，在任务运行前就估算了程序的计算量、通信开销、任务间的各种依赖关系及各个硬件模块间的连接和处理能力。系统编译器就通过这些基本信息将任务分配到多个处理器，任务一旦确定分配到某个处理器上，它就只能在该处理器上执行。多核处理器静态任务调度算法根据其调度的方法的不同可以分为 4 种类型：

1、有向无环图(DAG)

所谓有向无环图就是通过无向图的各个节点来表示程序的结构，节点就是各个任务进程，而边则表示任务间的相互依赖关系以及相互之间的通信关系。多核处理器的结构方式和任务类似，也被抽象为处理器模型，节点表示多个处理器节点，边表示多个处理器节点相互间的拓扑结构和通信关系。有向无环图如图 3.2 所示，这个图 P 抽象成数学模型就是，二元组 P=(T, E)，其中 T 表示单个节点任务，E 表示任务相互间的依赖关系和通信开销，那么通过有向无环图的的理论就可以描述整个系统的通信和调度开销了。

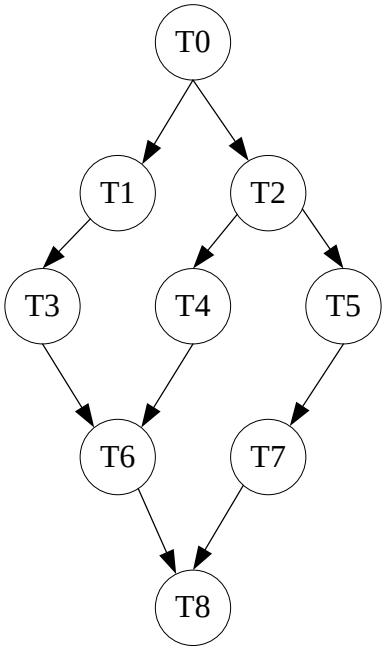


图 3.2 任务的有向无环图

2、Petri 网结构

Petri 网是对离散并行模型的数学表示形式，它是一个经典的控制数据流模型，它对上层任务的并行计算能力提供了很好的解决方案。因此，在硬件的设计过程中广泛的使用了这种结构，也有不少研究人员在尝试使用 Petri 网结构来解决多处理器的任务调度和分配问题，但是由于 Petri 网的结构非常复杂，因此，目前很少有人用这个结构来解决多核处理器任务调度和分配的问题。

3、分层任务图

一个分层任务图是由多个复合节点构成，复合结构又由其它的分级任务图构成，它是在“自动调度”理论中诞生的，编译器要求用户显式的写入驱动代码。

4、正规计算模型

这个模型是按照一定的规则去设计和实现网络系统，此设计方法给出了并行通信系统的通信方式，也描述了各个组件间的相互作用模型，组件都是用虚拟的方式描述的，这个模型的重点是把计算模型的功能分割为了控制端口和通信端口，有专门的设计工具帮助设计者来构建各个虚拟组件，当生成完成后，他们能相互协同工作。它给用户提供了应用程序设计的方法，但具体怎么实现的需要用户去手动操作。

以上是几种常见的多核处理器任务调度方法，都是为了提高内核的利用率。由于任何任务的运行时间是不能精确计算出来的，任务与任务间的的相互依赖程度比较大，调度比较复杂，由已有的理论可知，相互依赖的任务静态调度方案是一个 NP 完全问题，只具有次优解。任务模型及环境模型对静态任务调度算法的影响很大，多核任务调度算法为了得到更多的时间解，我们往往需要通过启发性算法做多个假设，例如针对每个任务建立单独的模型而忽略任务间的依赖关系，或者假设每个任务的运行时间基本一样，

任务间通信时间为零等。但实际上，每个任务都很复杂，且相应的环境也非常多变，因此这些假设情况很难实现，只有特殊情况才可能成立。这样很难求得最优解，且不具有通用性。

目前，评价一个静态调度算法的好坏主要有：

(l)调度性能：指任务调度算法执行完后，系统整体执行时间是否减小，算法是否具有通用性，能够用于各种不同的任务调度模型，是否能适应不同的环境模型，算法是否能适应不同的内核个数。

(2)实用性：是指算法是否实用，能适应千变万化的实际需求或内核结构的变化等等，它都能自动的匹配上，这一点对于任务调度算法非常重要。

(3)时间复杂度：指算法执行后，任务所消耗的时间情况，时间复杂度越高，性能越差，如果任务的规模非常大，那么它所花费的时间是用户无法接受的。

综合以上影响因素，我们需要在性能和时间上找到合理的平衡点，这样才能提高静态任务调度算法的整体性能。

3.2.2 动态任务调度算法

多核处理器的动态任务调度是指系统的任务调度器根据各个内核的负载情况，进行的动态的任务调整和分配，按照策略将任务分配到多个处理器的内核中，动态任务调度主要是研究整个系统的负载均衡策略及任务的迁移策略，系统的负载均衡是指每个处理器上处理的任务大致相当，总体负载状况相当，当某个处理器负载非常大时，动态任务调度策略会调整内核负载情况，将部分任务迁移到负载较轻的内核上。但是动态任务调度策略如何知道系统各个任务的状况以及如何调整任务到所需的处理器，如何知道任务在处理器间切换的次数，这些都是要明确的。要提高动态任务的调度性能，需要选择合理的系统负载指标、系统负载均衡区间及如何迁移任务。

负载指标的选择主要是根据现有的内核利用率、系统资源利用率、内核功耗等影响因素设计的，根据负载均衡的经验理论可知，在系统状态决策中系统指标是当前现有的，而不是已经过期的或不准确信息。因此使用合适的负载指标能提供系统的性能，在实际情况中，负载指标大多数是使用可计算的任务运行队列作为参考指标。

系统的负载均衡区间是指任务的调度在一个合理的负载下消耗的调度时间较短，通常是将内核分区调度，形成一个调度区间。这个区间要开辟的合理，如果太大，那么系统会花费非常大的时间查找调度，如果给的太小，就达不到系统的负载均衡的效果。因此需要设计一个非常适合的均衡负载区域。

任务迁移的主要作用是将负载较大的内核上的任务分配到较轻的内核上，但是在任务迁移的过程中，会消耗很多的系统代价和资源，因此在任务要被迁移前，需要计算任务所付出的代价。如重载内核和轻载内核间任务频繁的切换调度、不能迁移当前正在运

行的任务等等。

目前比较流行的动态任务调度算法主要有最早截止期优先算法（EDF）、最迟开始时间算法（LST）、多处理器截止期优先算法（DOOM）。

总之，动态任务调度机制有着很多优点，它的灵活性很高，能根据当前处理器中的任务执行状态进行动态的调度，使整个系统的总负载达到均衡，能取得很好的调度性能，当然，动态任务调度机制也有着明显的缺点，就是调度算法比较复杂，运行的开销比较大，任务调度及算法的执行都会耗费一定的时间，因此它也严重的影响着系统的性能。

3.2.3 两种算法的性能比较

通过对多核处理器任务静态调度算法和任务动态调度算法的整体优缺点进行了研究，在早期，研究人员解决多核处理器任务调度的方法主要是用静态任务调度算法，这种方法简单方便快捷，任务在调用前就已经通过静态调度算法预先估算好了，即任务在运行前，它会被分配一个优先级，任务再按照调度算法规则进行计算，这样整个系统的调度很清晰，复杂度小且开销低，也不用考虑系统的均衡负载问题，这样用户使用也很方便，不需要考虑这些问题，系统程序开发的难度降低了不少。当然它的确定也很明显，有一定的局限性，不能动态的适应需求的变化和处理器的处理性能，因此，它在某些情况下不如动态调度策略，动态调度策略是在任务运行过程中对任务进行动态的分配和调整，能高效的解决全局系统的负载问题，因此它的使用的非常广泛，当然也有很明显的确定，就是在动态调度过程中额外的增加了处理器和系统的开销。有此可见，设计一种高效的任务调度机制且又能兼顾这些调度性能缺陷是非常有必要的。

3.3 现有的任务调度机制和策略分析

任务是完成某个功能的程序集合，它是系统调度中最基本的单位。从任务的实时性看，任务有到达时间、就绪时间、运行时间、截止时间及到达频率。根据其频率不同又有周期任务及非周期任务的区别。周期任务是指在连续两次到达系统所花的时间是相等的。知道了任务第一次到的时间就知道以后每次任务到来的时间。而非周期的任务是指连续到达的两次任务所花的时间是不固定的，也不知道它什么时候到来。当一个新任务到来时，如果其没有获得时间片，它就需要等待该任务运行完再执行，这都是由调度器控制的。

目前，众多研究人员在单核处理器上的任务调度机制方面做了研究了很多，其调度机制和方法的研究已经非常的成熟，并且被广泛的应用到各类应用系统中，本节将介绍单核处理器的任务调度机制及常见的多核处理器任务调度机制，然后比较它们的不同之处以及多核处理器任务调度的难点。

3.3.1 单核任务调度策略

在单核处理器环境下，由于处理器只有一个，系统运行后每个时间点只有一个任务处于运行状态，剩余的任务则均处于等待状态，当处理器空闲后，等待的任务才被唤醒。单核处理器的任务调度机制的总体设计思想是处理器一直处于运行状态，时刻有任务在上面交替执行，这样整个处理器资源的利用率最大，系统的整体性能及吞吐量也是最大。因此良好的单核处理器的任务调度机制可以使计算机的性能达到最大化。目前在任务调度策略上比较成熟的算法如下：

1、最短作业优先调度策略（SJF）。这个策略是根据任务在处理器中运行所需的时间来衡量的，如果某个任务运行所要的时间最短，那么调度器就会优先选择这个任务运行，可采用抢占式或者非抢占式机制来调度。这种调度策略的优点是所有任务的平均等待时间很短，是很理想的调度算法。但是它有个难点就是在任务运行前无法快速确定每个任务在处理器上运行所需要的时间，因此无法知道最短作业是哪个。

2、时间片轮转调度法。这种调度方法是最古老、最公平、最简单且被使用最广泛。它的设计思想是使每个进程都分配一个时间片，即是每个进程运行的时间，当一个进程用完时间片后，CPU 便会剥夺这个进程的资源，如果进程执行完后时间片还未用完，则 CPU 直接切换进程。由此可知，对于时间片大小的设计非常重要，它关系到整个任务调度算法的性能，如果时间片分配的太小，那么进程上下文切换就会很频繁，这样系统开销会非常大，如果时间片分配的太大，那么 CPU 会浪费不少时间片。因此合理的分配时间片是非常重要的。

3、多级反馈队列。多级反馈队列是指任务在调度运行时其可在队列之间移动，这样可以解决任务分配到某个队列后，任务再也不会改变调度位置，灵活度不够的问题。多级反馈队列的调度策略是如果较低优先级的任务在队列中等待了很长时间，系统就会改变该任务的优先级，使其移到较高优先级的队列中，这样可以防止任务由于优先级太低而一直得不到运行而饿死，如果任务过多的使用处理器，那么该任务就会被移动到低优先级队列中，但正在交互的高优先级的任务保持不变。多级反馈队列根据这些信息来调度整个系统的任务，用以确认任务何时升高和降低任务的优先权。

4、先到先服务调度策略。这个策略的调度思想是新来的任务将会加入到任务就绪队列尾部中，处理器依次处理就绪队列中的任务，先来的任务先得到处理器资源。这种调度方式的优点是代码编写简单并且容易被用户所理解，但是它的缺点是所有任务的平均等待时间非常长，使处理器和系统硬件资源利用率非常低。

5、优先权调度策略。优先权调度是指根据任务所拥有的优先权进行调度，高优先权的任务优先分配处理器并调度，如果多个任务具有相同优先级，则根据先来先服务策略进行调度。优先权的设置方式有两种，一种是内部优先权设置，即系统根据任务的整体运行情况来计算任务的优先权，比如内存运行状况、打开的文件数、任务的运行状况

及处理器的资源使用情况。另一种是外部优先权设置，即用户可以手动的设置任务的优先级，如提供任务的优先等级，设置任务的重要性等。然而这种调度算法的缺点是容易造成低优先级的任务饿死，低优先级的任务无线的等待。因此在使用这种策略时可以适当的根据任务的等待时间来增加任务的优先级。

6、多级队列调度。多级队列调度是指运行的任务被分层多个单独的队列，每个队列内部采用适当的调度策略，队列与队列之间又采用合适的调度策略进行协同工作，这样任务将会有序的执行，并且能提高整体任务执行的并行度。

3.3.2 多处理器任务调度机制和策略

和单核处理器任务调度机制和策略相比较，多核处理器的任务调度机制没有比较成熟的调度机制和策略。一些研究人员纷纷的提出了多核处理器的任务调度机制，比如有异构多核处理器的自适应算法，其设计思想是系统根据当前系统的整体运行情况，不断的寻找和学习更优的调度方法，能随时适应系统的任务和内核任务量的变化；动态调度优于静态调度的机制，其设计思想是尽可能多的使用动态调度，动态调度符合任务的动态变化特点，它能最大可能的利用内核的资源；根据任务对内核依赖的属性进行调度任务，其设计思想是根据任务的特点，该任务最适合在哪个核上运行的原理，将任务自动分配到该处理器上，那么内核就会发挥其最大的性能；更快内核优先调度策略，其调度思想是让更快的内核优先执行任务，其它内核上的任务也可以动态的调度到这个内核上执行，这种设计思想会带来负载均衡和任务迁移的额外开销。这些调度机制的设计思想和方法给我提供了很好的参考，然而目前主要使用的多核处理器任务调度机制如下：

一、全局队列任务调度机制。

全局队列任务调度机制是指操作系统在运行过程中会从这个任务调度队列中取出任务，操作系统中拥有一个全局任务调度队列，当处理器有空闲资源时，它便从这个队列中取出任务并执行，这样各个处理器都有比较好的利用率。这种设计方式非常简单，也容易维护，各个处理器也能获得不错的资源，目前大多数多核操作系统都使用的这种任务调度机制。

二、局部队列任务调度机制。

局部队列任务调度是指每个处理器中都有一个局部的任务调度队列，当处理器有空闲资源时它便从自己的局部任务队列中取出任务，这样任务就不用在各个内核间进行调度了，各个内核对任务的命中率也就大大的提高了，但是它也有致命的缺点就是一旦某几个内核的局部队列中没有任务，那么内核的资源就大大的浪费了，这样使得内核的利用率非常的低。

三、共生队列任务调度机制。

共生队列任务调度机制是指把访问共享资源较多的任务及访问共享资源较少的任

务均衡的放在调度队列中，然后让其在一个时刻执行，这样任务间就会减小冲突，提高任务调度的效率。

当然，前面提到的单核处理器任务调度机制和策略都适用于多核处理器的任务调度，但是单核处理器的任务调度主要关注的是多样性的任务，而在多核处理器环境下，不仅任务是多样的，处理器也是多样的，因此为了提高多核处理器的资源利用率和系统的性能，需要任务调度机制也是多样的。因此设计一种保证多核负载均衡，发挥各个内核的性能特长，提高整体系统吞吐量和实时性的任务调度机制具有非常重要的研究意义。

3.4 多核处理器任务调度机制的评价标准

针对多核处理器任务调度机制的处理方法和方式，其性能的评价标准可以从以下几点讨论，任务的单位吞吐量、任务的响应时间、系统资源的利用率、任务的可调度性、任务调度的复杂性、数据的有效生存时间、系统的容错性、任务的接受率等。这些基本特性很好的反映了一种调度机制的好坏，其具体表现如下：

1、任务的单位吞吐量。是指系统在单位时间内接收的任务总数，若单位时间内接收的任务多说明其吞吐量大，性能好；反之，其吞吐量小，性能差。

2、任务的响应时间。是指系统接收到这个任务后到系统对其做出反应的时间，若反应时间越短，说明系统响应越快，实时性更高，性能更好。

3、系统资源的利用率。是指系统在运行过程中，所有运行任务所占用的资源总数，它能显示资源的整体利用率。它对单位时间内任务的调度数量有着影响。

4、任务的可调度性。是指任务调度序列中的所有任务都是可以被调度的，都可以被执行，调度算法能在单位时间内使用它们，并保证系统的需求。

5、任务调度的复杂性。是指任务在调度过程中，任务的复杂程度及调度的复杂性集合。

6、数据的有效生存时间。是指数据只在这段时间内有效。

7、系统的容错性。是指处理器或系统硬件出现故障后，任务可以正常的恢复到有效的状态。

8、任务的接受率。是指所有到达就绪队列的任务被顺利执行调度的比率，即调度任务的总共数量和到达的全部任务之比。

以上基本评价指标对多核处理器中的任务调度性能有着非常重要的作用。这些标准的比例大小将直接影响整个操作系统的全局性能，对系统实时性要求很高的产品将大幅度提升任务的响应时间、数据生存时间及任务的吞吐量。其中响应的时间非常重要，这也是实时系统的一个非常重要的指标。这与任务的整个运行过程都密切相关的。

3.5 本章小结

本章首先分析了多核处理器任务调度的基本原理，其中就多核处理器的进程原理、任务调度及任务间通信做了详细介绍，然后分析了多核处理器的静态任务调度算法和动态任务调度算法的优缺点，接着对现有的任务调度机制和策略进行了对比分析，最后给出了多核处理器任务调度机制的评价标准，并对如何设计一种高效的任务调度机制进行了分析。为后续章节的任务调度机制的设计提供了理论基础。

第 4 章 多核处理器下微内核任务调度机制的设计

通过第 3 章对多核处理器下微内核任务调度机制的整体研究和分析，使我们对目前多核环境下，已有的微内核操作系统任务调度机制有了进一步的了解，我们发现，现有的多核处理器任务调度机制存在一定的缺点，调度效率仍然有待提高，因此我们可以通过对任务分配模型和任务调度机制进行改进，克服现有多核处理器任务调度机制的不足，以提高多核环境下微内核任务调度的性能。

4.1 多核处理器任务调度机制问题描述

由前面的分析可知，目前多核处理器的主流设计是将处理器的核放在芯片上成倍的增加，然而随着时间的发展，这种发展方式会受芯片体积和技术等因素而阻碍其发展，处理器核的增加可以适当的降低时钟频率及系统的功耗，系统的性能会得到一定的提升，这种不断复制核的策略确实是个不错的选择，但是我们是复制很多小而简单的核还是复制少数复杂又多样的核呢？第一种设计使线程有了很高的并行性，第二种设计线程的并行性比较低，但是它处理复杂的任务非常的快捷。这两种设计有着各自的优点，本文通过结合这两种设计思想，采用折中的办法，即在异构多核处理器上实现任务的调度，这种设计思想是多个核的功能不同，各自的处理能力很专一，很符合线程对硬件的要求，那么就可以根据处理器的特性，将任务分配给不同的处理器，这样就可以提高处理器的资源利用率，同时可以提高线程间的并行度，因此设计一种良好的任务调度机制是提高多核处理器性能的关键因素。

为了发挥出异构多核处理器的最大性能，我们要设计合理的任务分配模型和任务调度机制。由已有的理论可知任务在分配给处理器时，其主要开销是任务执行的开销和处理器核与核之间的通信开销。如果这两个开销的整体都比较小的话，那么性能是最优的。然而这两个因素是相互依存的，如果将大量任务分配在单处理器中，那么核间通信开销会大大减小，但是会导致单核的负载过大；如果将任务均匀的分配给处理器，这样可以提高任务的并行度及处理器的利用率，但是会增加不少处理器间的通信。

那么要提高异构多核处理器的性能，对于任务的分配和调度既要考虑任务的执行开销也要考虑处理器间的通信开销。在使用任务调度机制前加入任务分配策略会使系统的并行性非常高，任务调度算法在多核处理器上将执行的更加高效。目前比较流行的提高系统性能的方法是先对任务进行分组，然后再用任务调度算法进行调度。根据这个思想，本章提出了改进的任务分配模型和改进的任务调度机制。

4.2 任务分配模型的设计

4.2.1 现有的任务分配模型

由现有的任务分配模型定义可知，假如一个多核处理器系统平台是由 NP 个处理器组成的，分别为 P0，P1，P2，…；PN，假如这个系统中有 NT 个任务进程 T0，T1，T2，…；TN，且每个进程分别由 TD0，TD1，TD2，…，TDN 组成，任务的集合可以用有向无环图 DAG 表示，它可以由一个五元组构成，D=（V，E，RT，C，W），其中：

1、V 是顶点的集合对，即一个二元组（Ti，TDi）,其中 Ti 是任务进程，TDi 是相应的线程；

2、E 是有向边的集合，即线程 TDi 与线程 TDj 间的联系；

3、RT 是运行时间的集合，即线程 TDi 的运行时间 RT（TDi）；

4、C 表示线程间的通信的次数，它越大就表示两个进程间通信的非常频繁，相反就表示没通信；

5、W 表示线程间的通信开销集合，即线程间的通信开销 W（TDi，TDj）；

例如图 4.1 为一个 DAG 图，它包含四个进程，分别为 T0，T1，T2，T3，T0 包含的线程有 TD0，TD1；T1 包含的线程有 TD2，TD3，TD4，TD5；T2 包含的线程有 TD6，TD7，TD8；T3 包含的线程有 TD9，TD10；其中边上的数据表示线程间的通信开销。

当有新任务到来时，任务进入就绪队列，根据现有的任务分配模型，队列中的任务建立任务关系图，然后任务根据全局队列调度方式将任务分配给各个处理器，然后处理器再根据相应的调度策略来执行任务。

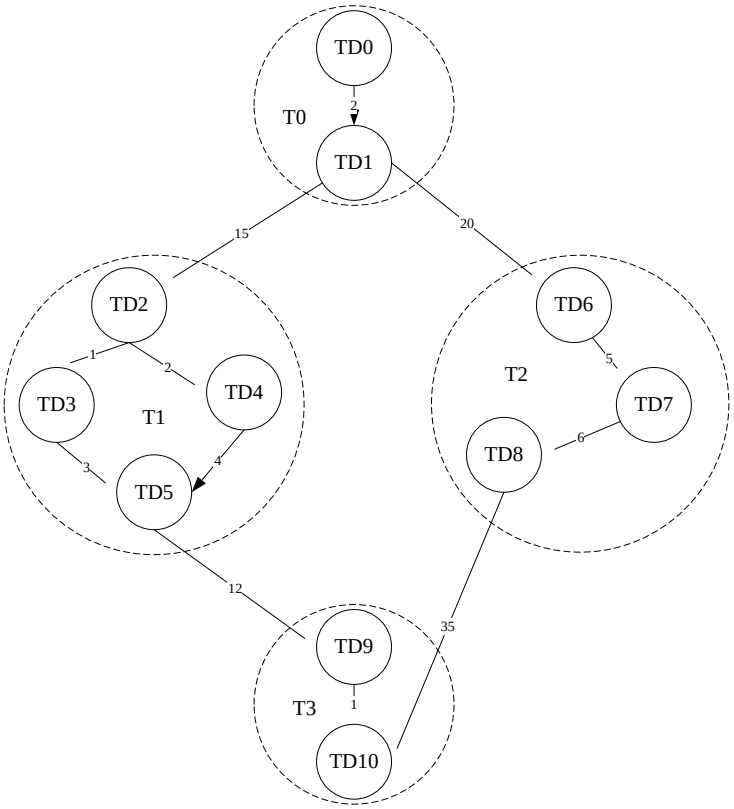


图 4.1 任务分配关系图

4.2.2 改进的任务分配模型设计

4.2.2.1 基于多核处理器的任务分配规则

在多核处理器系统中，多个处理器通过共享缓存或共享高速信道进行通信，不同处理器间的联系并不紧密，它们之间的通信往往要耗费大量的通信时间。通常，在一个应用程序中，程序包含了多个任务，他们之间的通信是通过消息机制来实现的，即相互间发送和接收消息。在多核处理器中，为了使有效的资源利用率最大化，任务可能会划分为多个线程，它们会在多个内核上来回的运行，同一个任务的各个线程是通过共享内存来通信的，不同的任务有各自独立的地址空间，它们通过消息来通信的。

通过上面的分析，我们可以得到基于多核处理器任务分配的基本规则：

一、将同一个任务的多个线程分配在一起，因为同一任务中的线程共享内存空间，将它们分配在一起，可以减少不必要的通信开销。

二、不要将多个任务同时分配给同一个处理器，这样不仅处理器的负载大，而且任务间频繁的切换执行也会带来很大的开销。

三、将相互间通信非常频繁的任务分配给一个处理器，这样可以减少多任务间的通信开销。

四、保证多个处理器和内核间的负载均衡，防止一些处理器负载很重，一些处理器很空闲的情况。提高处理器的并行执行能力。

4.2.2.2 基于优先级的抢占式任务分配机制

由前面的分析可知，目前主要采用的任务分配策略是直接随机分配任务，这种任务分配策略简单、方便和快捷，一旦任务到来就能立刻被分配执行，在一定条件下能提高系统的效率。然而这种快捷的分配方式也恰恰是其弱点，当所有处理器都在执行任务时，如果一个高优先级的任务到来，由于没有空闲的资源，那么它只能等待处理器释放资源，这就使得高优先级的任务延迟执行；当多个依赖程度很大的进程到来时，系统随机的将任务分配给处理器，那么这些任务间会进行频繁的通信，这样就加大了任务间的通信开销。这些问题会经常出现在任务分配的过程中，都极大的影响了系统的性能。

针对目前的任务分配机制存在的问题，本文提出了一种改进的基于优先级的抢占式任务分配机制，在多核处理器系统中，每个任务都赋有优先级，优先级高的任务享有优先被分配的权利，任务进入系统后，系统根据任务的优先级，将任务分配到各自优先级的就绪队列，那么每个就绪队列中的任务具有相同的优先级，为了提高任务分配的效率，本文采用了位图法来描述任务的状态，通过位图的标志位快速定位就绪队列中的任务，这样提高了系统的实时性。

基于优先级的抢占式任务分配机制中的任务优先级是通过如下的方法指定的，任务在初始创建时，系统会默认的给该任务指定一个优先级，用户可以更改这个优先级，同时也可以在任务运行过程中动态的改变它的优先级。在任务的优先级就绪队列中，相同优先级的任务按照先进先出的规则依次进入队列中，系统根据任务的优先级状况，将队列中的任务依次分配到多个处理器中。多级优先级任务就绪队列如图 4.2 所示，其中 0 到 255 表示所有的优先级，0 的优先级最高，255 的优先级最低，处理器根据任务的优先级得到分配的任务。

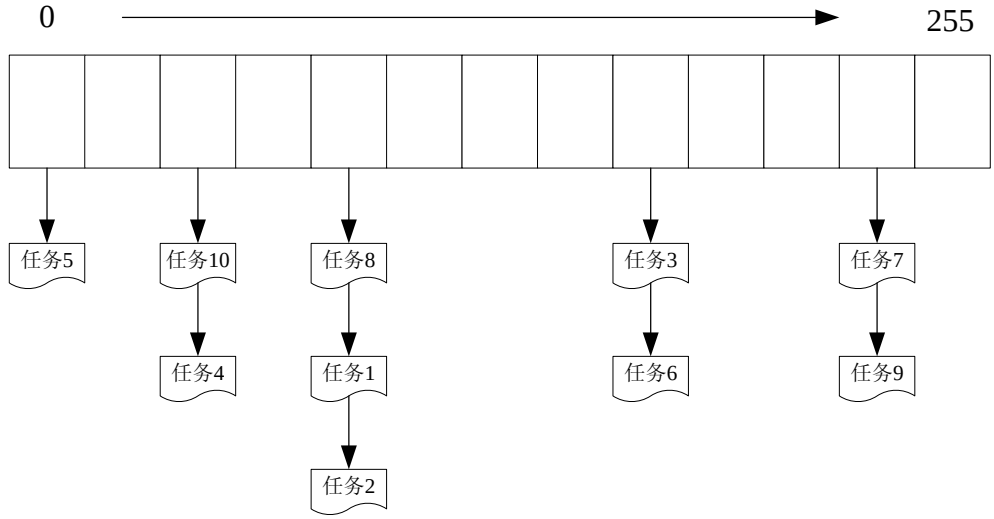


图 4.2 多级优先级任务就绪队列

本文设计的任务分配机制支持任务的抢占模式，即在当前正常运行的系统中，如果有更高更紧急的任务进入就绪队列时，当前运行的任务就会被抢占，那么系统就进入了

任务抢占模式，当前的任务被阻塞中止，直到高优先级任务运行完为止才唤醒当前任务，这样提高了系统的实时性，使紧急任务能得到快速分配和处理。

根据这种改进的任务分配机制，本文设计了如下任务分配模型，如图 4.3 所示，多个任务集中包含所有新来的任务，系统通过基于优先级的抢占式任务分配机制将任务分配到就绪队列中，接着任务调度器根据改进的任务调度机制将任务就绪队列中的任务分配给处理器，处理器处理完任务后，信息采集器根据任务调度的结果实时更新任务的优先级，这样便可动态的调度所有任务，使任务的调度的效率更高，更快，更准，任务的执行与当时的实时环境变化密切相关。

这种改进的任务分配机制相比于现有的任务分配机制，在一定程度上减小了任务在多个处理器间频繁的切换，提高了任务处理的效率，进一步提高了系统的性能。但是，只改变任务的分配模型是无法解决调度过程中的所有问题，因此本文将改进任务的调度机制来进一步提高任务调度的效率。

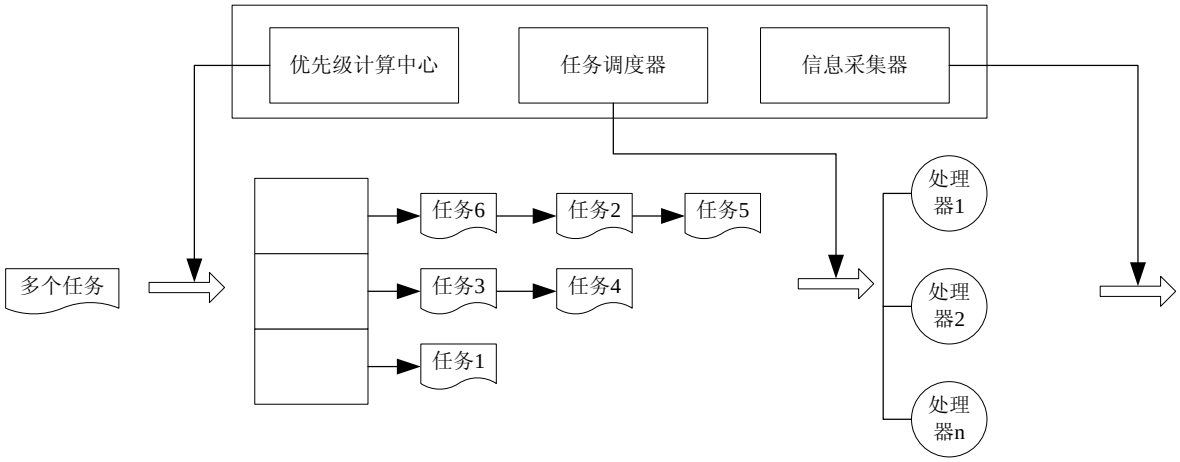


图 4.3 改进的任务分配模型

4.3 多核处理器任务调度机制的设计

由改进的任务分配模型可知，按照多核处理器任务的分配规则及基于优先级的抢占式任务分配机制这两种方法，在一定程度上能够提高任务的执行效率和系统的整体性能，但是，为了满足人们对系统效率的更高要求，就必须对多核处理器的任务调度机制进行改进，任务调度机制效率的高低决定了整个系统的性能的好坏。因此本节着重研究多核处理器下，嵌入式操作系统微内核的任务调度机制。

4.3.1 改进的任务调度机制设计

4.3.1.1 基于缓存队列的任务调度机制

由第三章的分析，我们了解到，目前比较流行的多核嵌入式操作系统微内核的任务

调度机制主要是全局队列调度、局部队列调度和共生队列调度。全局队列调度中的任务可以调度到任何一个可用的处理器上，所有的处理器都共享一个队列，任务的调度实现了负载均衡，这种方法虽然简单，但是有着很明显的缺点，一是系统的并行性受到了很大的限制，因为只有一个全局队列，所有处理器都要互斥的访问它，不能并行的访问，这样极大的影响了多核并行性能的发挥。二是全局队列调度不能有效的利用缓存，不能发挥缓存的作用。

相反，局部队列调度在一定程度上能提高系统的性能，每个处理器都有自己的局部队列，所有处理器可以同时执行任务，系统的并行性得到了很大的提升，它能充分的利用本地的缓存提升性能，但是它也有个很大的弱点就是无法保证系统的负载均衡，可能造成一些处理器的局部队列任务非常多，一些处理器的局部队列任务非常少，要均衡负载所需要的开销非常大，对系统的整体性能有着很大的影响。

基于以上分析，本文提出了基于缓存队列的任务调度机制，此调度机制结合了全局队列调度和局部队列调度的优点，同时又能保证高优先级的任务优先执行，实现系统的负载均衡。此调度机制设计的思想是，首先，全局队列从任务分配模型中获得任务，其获取规则是先获取高优先级的任务，然后再获取低优先级的任务；其次，全局队列将获得的任务调度到各个处理器的缓存队列，并负责所有处理器缓存队列的负载平衡；最后，处理器从缓存队列中读取任务并执行，这样系统的实时性、任务吞吐量及任务的并发执行都得到了很大的提升，是非常合理的调度方式。其调度过程如图 4.4 所示，左边的队列是任务优先级就绪队列，0 表示优先级最高的任务队列，其中有任务 6、任务 2 和任务 5，经过任务调度器后，0 优先级的任务被调入全局队列，然后全局队列再将任务调度到各个处理器的缓存队列，最后各个处理器并行执行任务，这样整个任务调度过程就完成了。

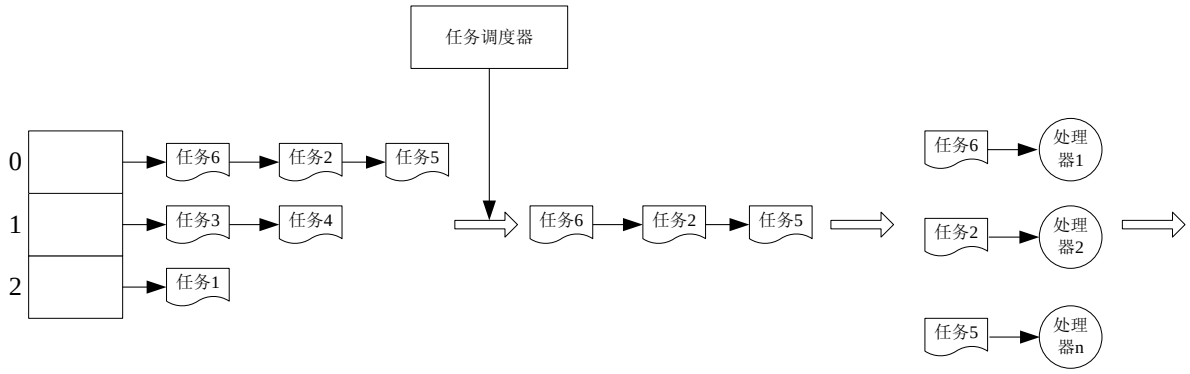


图 4.4 基于缓存队列的任务调度

4.3.1.2 快内核优先的任务动态迁移策略

我们知道，嵌入式操作系统微内核中的任务具有多样性和复杂性，同时，多个处理器的性能也有着很大的区别，处理任务的快慢不尽相同，那么可能在某个时间段，一些

处理器在执行任务，另一些处理器则空闲着，这样就浪费了系统的资源，因此，需要允许任务中的线程可以动态的迁移到空闲处理器上并执行。

本文为了进一步提高处理器的利用率，设计了快内核优先的任务动态迁移策略，就是全局队列将任务优先调度给性能好的内核，同时信息采集器实时的检测各个处理器的任务缓存队列执行情况，一旦发现有处理器空闲时，就根据调度策略，将其它处理器上的任务迁移到空闲处理器上。这样不仅提供了处理器的效率，还可以使每个内核的负载均衡，降低了任务执行的延时，提高了系统的实时性。

4.3.2 改进的任务调度机制评价

改进的任务调度机制具有简单高效的特点，同时拥有以下优点：

1、该机制能保证高优先级的任务优先执行，减少任务延时，提高系统的实时性；

2、能保证每个处理器都能充分发挥其性能，并且处理器的整体负载均衡；

3、该机制简单高效，任务调度器的开销恒定，复杂度低；

4、支持任务抢占模式，让紧急任务能优先执行；

4.4 本章小结

本章是本文的核心章节，介绍了作者的主要工作。本章主要从两个大的方面介绍了在多核环境下，对嵌入式操作系统微内核的任务分配模型和任务调度机制进行了的改进。首先描述了当前多核处理器任务调度机制面临的问题，介绍了现有的任务分配模型，在此基础上，提出了基于优先级的抢占式任务分配改进方案，同时给出了多核处理器环境下任务分配的通用规则，然后对多核嵌入式操作系统微内核的任务调度机制的改进，通过利用全局队列调度与局部队列调度相结合的方法实现了任务的调度，提出了基于缓存队列的任务调度机制，为了进一步提高处理器的效率，给出快内核优先的任务迁移策略，使处理器的任务负载均衡，最后，对改进的任务调度机制的优点进行了评价。

第 5 章 实验测试与结果分析

5.1 多核处理器实验平台

5.1.1 Simics 仿真平台的介绍

Virtutech Simics 是一款非常优秀的模拟器软件，它的最初版本是由瑞典计算机科学研究院开发的，后来交给 Virtutech 公司进行的商业化开发。Simics 可以在标准的 PC 机上快速的建立一个虚拟的目标系统，这个完全的虚拟平台实际上是一个真正意义上的物理硬件模型。每个 Simics 模型都对应了一个真实的操作系统和硬件平台，它可以模拟单片机和嵌入式平台，也可以模拟几个到几千个的芯片、DSP 处理器、CPU、网络集群系统和多核架构等等。

正因为 Simics 有如此强大的模拟仿真功能，在科学研究领域它常用来模拟计算机的体系结构，如内核的源码调试。同时，Simics 可以模拟各种处理器内核和多种处理器架构。它支持 SMP、CMP 和超线程等多核处理器技术，通过 Simics 的配置文件，我们可以配置内核及处理器的各种参数，如一级缓存、二级缓存、系统频率和内核结构等。用户自定义一个系统时，可以通过配置系统的硬件，像组装一台电脑一样，可以放内存、处理器、南桥与北桥芯片以及外接设备等，按照 Simics 规范配置好这个系统后，就可以在 Simics 仿真平台上运行和测试系统。

Simics 仿真平台可以模拟多种处理器指令架构，如 x86、ARM、Alpha、MIPS、PowerPC、EM64T 等；也可以模拟各种外接设备，如软盘、磁盘、鼠标、键盘、网络、图形加速器等；这个平台上可运行多种类型的操作系统，如 Linux、VxWorks、Solaris、QNX、FreeBSD、Windows、RTEMS 等；也可以模拟交换机、嵌入式系统、机顶盒、多核处理器等，由此可见 Simics 的功能是非常丰富和强大的。

本文选择使用 Simics 为实验平台，是因为它支持多核体系的架构，可通过配置即实现多核处理器的功能，能完成本文的任务调度功能，因此非常适合本文的实验测试。

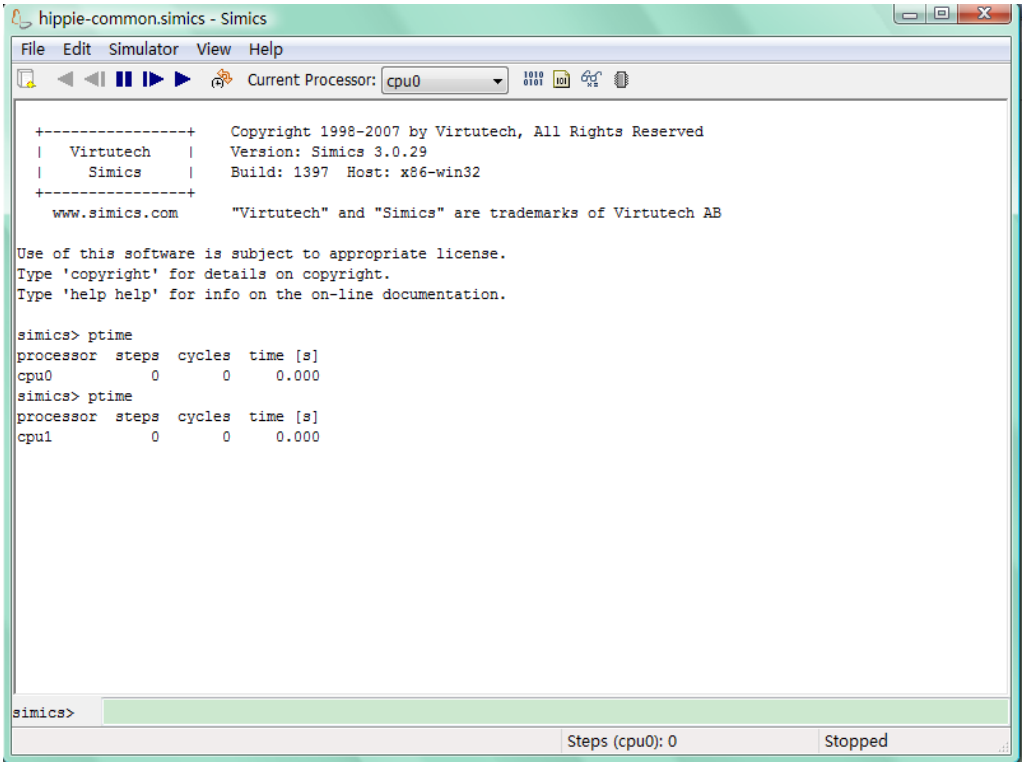


图 5.1 Simics 命令输入界面

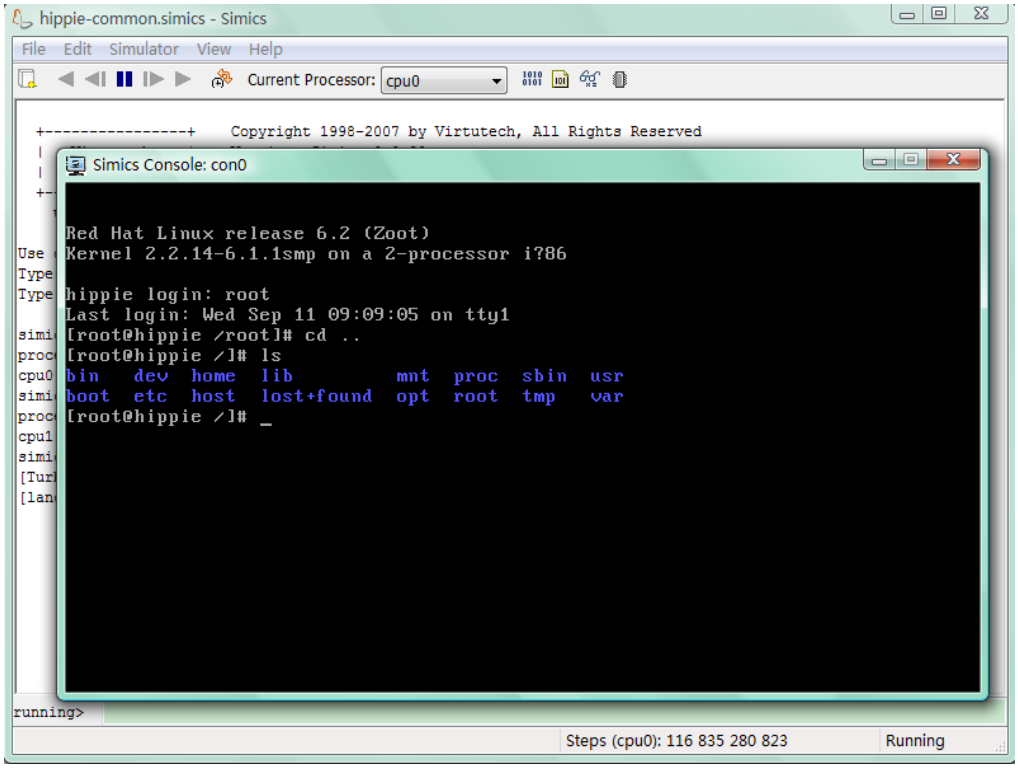


图 5.2 Simics 模拟系统界面

如图 5.1 所示，是 Simics 的命令输入界面，Simics 运行时会加载 target 目录中的 simics 文件，根据文件中配置的硬件信息，系统会启动模拟系统，我们可以编写 simics 文件来定义多核处理器架构，然后在上面运行测试程序。如图 5.2 所示，是 Simics 的模拟系统界面，它将启动多核处理器架构下的 linux 系统。

5.1.2 Simics 多核仿真平台的搭建

多核处理器平台是指具有 2 个或者 2 个以上处理器的计算机。搭建多核处理器可以提高系统的可靠性及计算机的处理能力，保证计算机高效的运行。Simics 平台是通过 Simics 脚本或 python 脚本来实现多核平台的。其搭建方式比较清晰，只需要配置好相应的硬件及操作系统，然后就可以启动自己定义的多核，其启动过程和真实机器的过程基本一样。首先配置多核处理器的硬件环境，通过编写 hippie-common.simics 脚本实现，配置了机器的物理地址、网络 IP 及 2 个处理器其具体实现过程如下：

```
if not defined machine_count    {$machine_count = 3}
if not defined mac_address      {$mac_address = "00:04:ac:00:50:00"}
if not defined ip_address       {$ip_address = "10.10.0.50"}
if not defined host_name        {$host_name = "hippie"}
if not defined id               {$id = 0}
if not defined num_cpus         {$num_cpus =2}

@from components import mac_as_list, mac_from_list
@from components import ip_as_list, ip_from_list
@mac = mac_as_list(simenv.mac_address)
@ip = ip_as_list(simenv.ip_address)

$host = $host_name
$create_network = "yes"

if not defined eth_link {
    import-std-components
    $eth_link = (create-std-ethernet-link)
}

$count = 0
while $count < $machine_count {
    set-component-prefix "processor" + $id + "_"
    $mac_address = `mac_from_list(mac)`
    $ip_address = `ip_from_list(ip)`
    $host_name = $host + $id
    run-command-file "%script%/multiprocessor.simics"
```

```
        python "mac[5] += 1"
        python "ip[3] += 1"
        $id += 1
        $count += 1
    }

    sim->cpus_may_share_memory = 10
```

接着，搭建基于 Simics 多核架构的操作系统环境，本文选用 Linux 操作系统环境，配置了操作系统的镜像文件、启动环境和启动参数，其具体实现过程如下：

```
    if not defined enter_in_boot_menu {$enter_in_boot_menu = "yes"}
    if not defined ip_address {$ip_address = "10.10.0.9"}
    if not defined host_name {$host_name = "hippie"}
    if not defined service_node {}

    if $service_node {
        local $sn = ($service_node.get-component-object sn)
        ($sn.add-host name = $host_name
        ip = $ip_address domain = network.sim
        mac = $mac_address)
    }

    default-port-forward-target $ip_address

    @image = sim.objects[simenv.disk].object_list['hd_image']
    @image.files = [["hippie3-rh62.craff", "ro", 0, 1056964608, 0]]
    @if 'linux_diffs' in dir():
    image.files += linux_diffs

    if $enter_in_boot_menu == "yes" {
        script-branch {
            local $con = ($console.get-component-object con)
            local $smp = (($num_cpus > 1) or ($cpu[0]->cpu_threads > 1))
            wait-for-hap Core_Step_Count 10000000
            if ($smp) {
                $con.input "linux-smp\n"
```

```
        } else {
            $con.input "linux-up\n"
        }
    }
}
```

通过以上方法，Simics 多核仿真平台就搭建完成了，在命令输入界面中选择 File->New Session，接着选择 workspace 工作目录中选择 hippie-common.simics 脚本文件，便可看到 Simics 平台初始化了 2 个处理器，cpu0，cpu1，如图 5.3 所示，接着点击启动按钮，便可启动操作系统了，如图 5.4 所示。

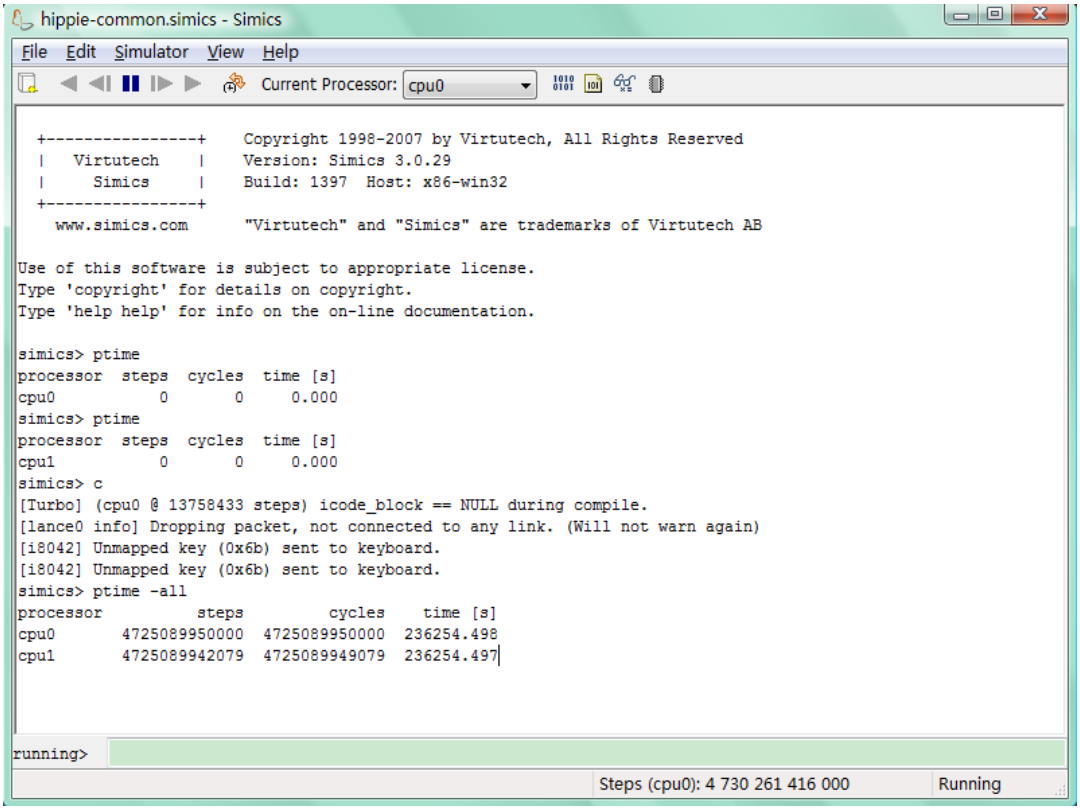


图 5.3 多核处理器初始化

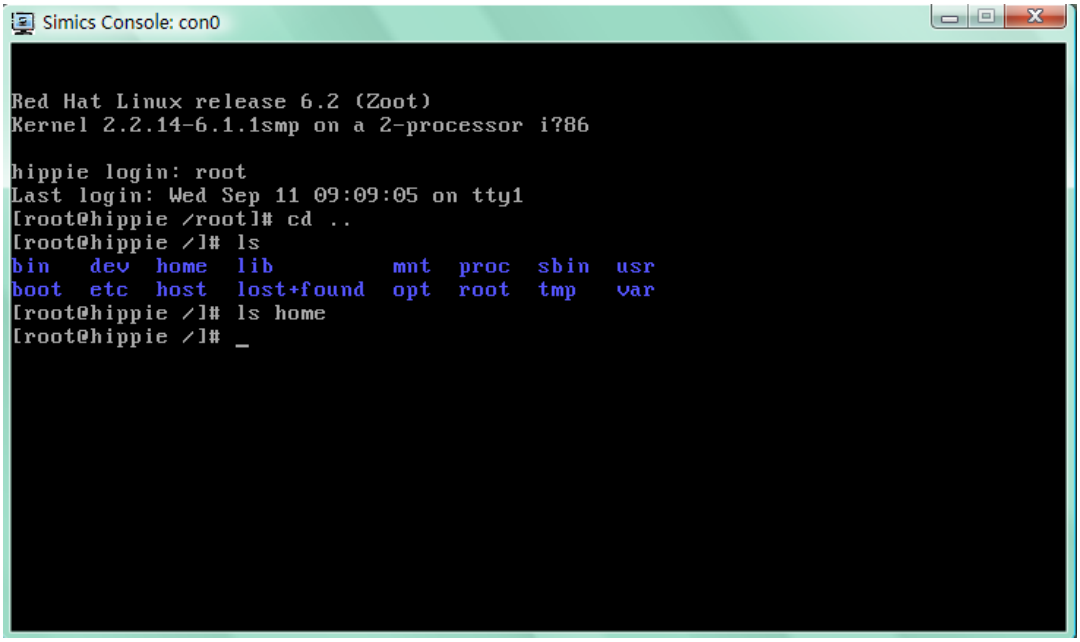


图 5.4 启动操作系统

5.2 多核处理器任务调度机制性能测试结果与分析

5.2.1 改进的任务调度机制实现

操作系统中的任务调度是由 schedule(void)函数实现的，系统中的调度策略是由 rt_policy(int policy)函数控制的，policy 从整体上区分普通进程和高优先级任务，因为针对它们的调度方式是不同的，高优先级的任务要优于低优先级任务执行，用户可以通过 sched_setscheduler()来调整调度策略。默认情况下，操作系统采用的是任务先来先服务和时间片轮转调度方法，其代码如下：

```
static inline int rt_policy(int policy)
{
    if (policy == SCHED_FIFO || policy == SCHED_RR)
        return 1;
    return 0;
 }
```

根据本文的改进任务调度机制思想，任务先通过任务分配模型，按照优先级对任务分组，接着按照优先级先后顺序将任务调度到全局队列，接着再由全局队列将任务分配到处理器的缓存队列，其改进的任务调度机制代码如下所示，将改动的代码放入 Simics 仿真操作系统中，能顺利运行。

```
#define SCHED_PRIORITY 0
#define SCHED_GQUEUE 0
```

```
#define SCHED_BQUEUE 0
static inline int rt_policy(int policy)
{
    if (policy == SCHED_PRIORITY || policy == SCHED_GQUEUE || policy == SCHED_BQUEUE)
        return 1;
    return 0;
 }
```

5.2.2 性能测试程序设计

本文主要针对多核环境下嵌入式操作系统微内核的任务调度机制进行了设计，通过对任务调度机制进行了改进，然后需要对这种改进机制的性能进行测试，然后与改进前的任务调度效率进行对比。

本文设计的测试程序的思想是：为了使任务具有随机性，系统随机产生了 2 个任务，分别是 task0 和 task1，通过执行一段循环累加程序来测试性能，任务调度完成后即可知道任务的调度效率，任务测试程序如下：

```
#include <pthread.h>
#include <stdio.h>

#define NUM_THREADS 2

void *task(void * p_thread_id) {
    int i;
    double sum=0.0;
    int thread_id = (int) p_thread_id;
    int RUN_TIMES = 1000; //设置循环次数

    for(i=0; i<RUN_TIMES; i++) {
        sum += i;
    }

    printf("task%d run times = %d\n", thread_id, RUN_TIMES);
    fflush(NULL);
    pthread_exit(NULL);
```

```
}

int main() {
    pthread_t threads[NUM_THREADS];
    int t;

    //创建线程
    for(t=0; t<NUM_THREADS; t++) {
        int ret;

        ret = pthread_create(&threads[t], NULL, task, (void *) t);
        if(ret != 0) {
            printf("pthread_create: ERROR: t=%d return code=%d\n",t, ret);
            exit(1);
        }

        fflush(NULL);
    }

    //等待线程结束
    for(t=0; t<NUM_THREADS; t++) {
        int ret, status;

        ret = pthread_join(threads[t], (void **)&status);
        if(ret != 0) {
            printf("pthread_join: ERROR: t=%d return code=%d\n",t, ret);
            exit(1);
        }

        fflush(NULL);
    }
    pthread_exit(NULL);
    return 0;
}
```

将性能测试程序复制到 c 盘根目录，在操作系统中执行 mount /host 命令，就可将 c

盘挂载到 linux 操作系统下，然后编译该程序即可以运行了，如图 5.5 所示。

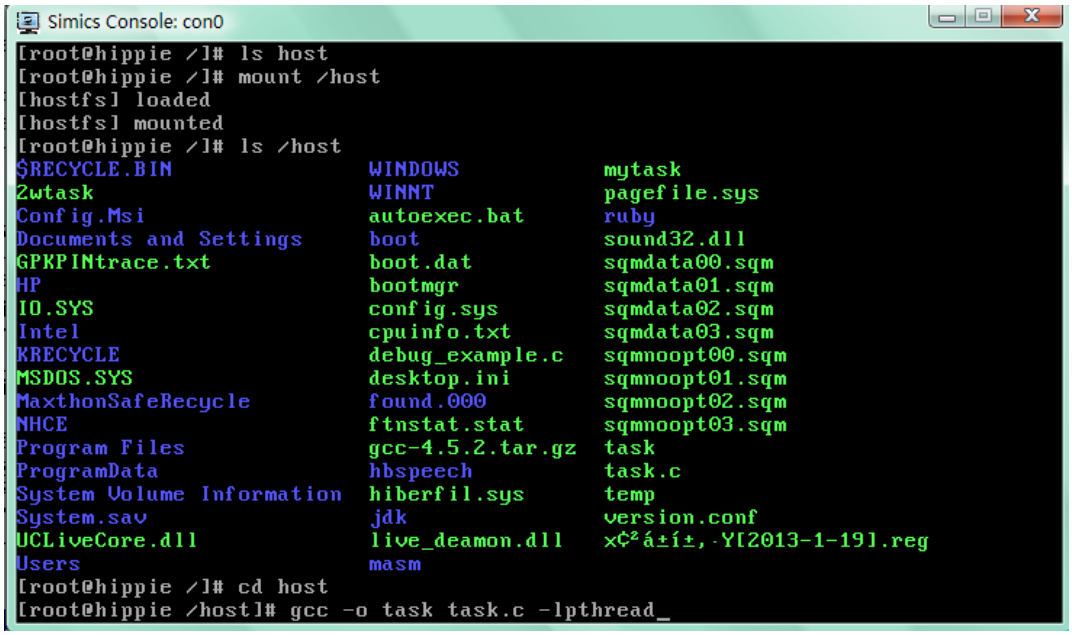
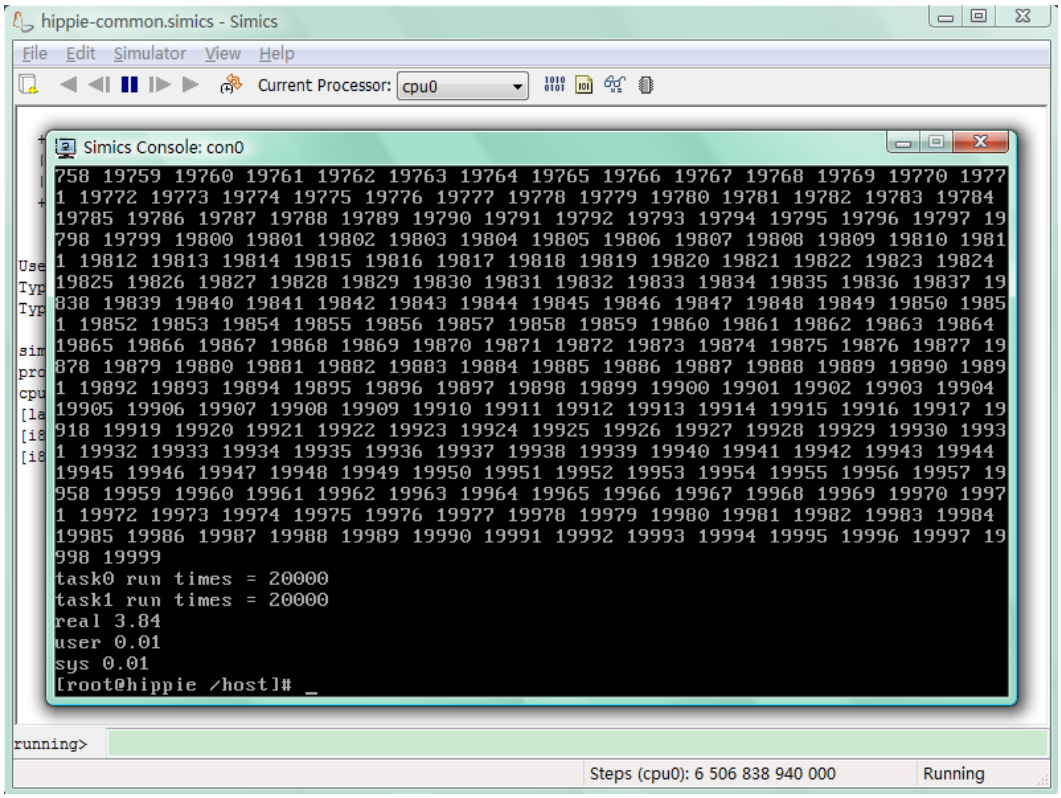


图 5.5 性能测试程序编译

5.2.3 性能测试结果对比分析

根据多核处理器操作系统微内核原有的调度机制和改进后的微内核任务调度机制分别调用测试程序，测试程序循环执行 20000 次，性能测试仿真实验结果对比如图 5.6 所示。



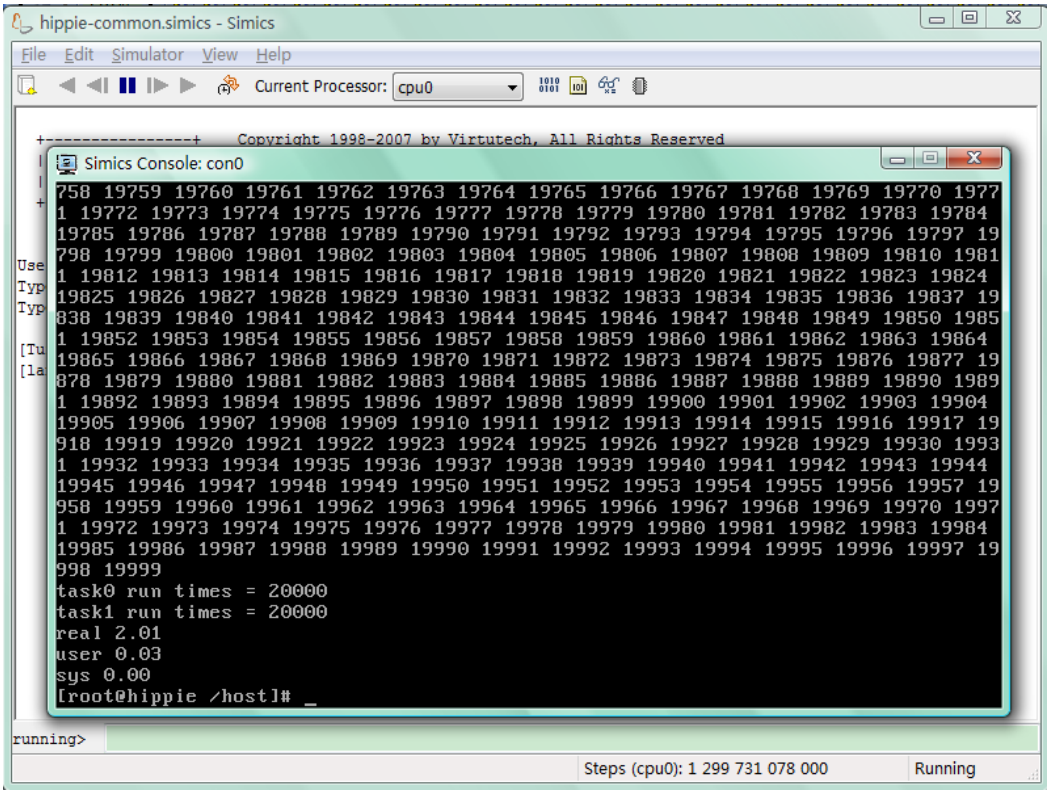


图 5.6 性能测试运行结果对比图

如图 5.6 所示，分别为性能测试程序在微内核原有的任务调度机制运行结果（上图）和改进后的微内核任务调度机制运行结果（下图），测试程序中的两个任务在微内核操作系统中调度了 20000 次，其中 real 表示实际时间，user 表示用户 CPU 时间，sys 表示系统 CPU 时间。

接着针对循环次数为 4000，8000，12000，16000，20000 进行测试，得到如表 5.1 所示的结果数据。

表 5.1 性能测试结果对比数据

调度机制	4000 次	8000 次	12000 次	16000 次	20000 次
改进前	0.78	1.48	2.17	3.03	3.84
改进后	0.39	0.76	1.16	1.64	2.01

（注：表中的结果是用系统命令 time –p 测试的时间，其结果在每次运行中会有略微的变化，与当时的系统运行状况有关，但它们的总体趋势是保持一致的，因此从量化角度看可以进行对比实验）

根据表 5.1 中的性能测试结果对比数据可以看出，随着任务调度的次数越来越多，改进的任务调度机制所用的时间远远小于改进前所用的时间，由此可见，改进的任务调度机制随着任务调度的次数增加，总体减少的调度时间趋势是一致的，改进的任务调度机制效率会越来越高，效果会越来越明显。因此可以知道，本文设计的多核处理器任务调度机制提高了系统的效率，增强了系统的实时性，达到了处理器的负载均衡的效果。

5.3 本章小结

本章完成了本文设计的任务调度机制的测试。在 Simics 仿真平台上搭建了多核处理器硬件平台和嵌入式操作系统平台，实现了测试程序任务调度的过程。实验平台为 Simics 仿真平台，在通过配置 Simics 文件和 include 文件，完整的搭建了仿真平台，接着对与任务调度策略相关的代码进行了修改，实现了改进的多核处理器嵌入式操作系统任务的调度机制。最后用测试程序测试了改进前后的任务调度机制性能，通过测试结果对比分析可知，基于本文设计的多核操作系统微内核任务分配模型和任务调度调度机制具有良好的性能。

第 6 章 总结与展望

6.1 工作总结

如今，多核处理器越来越普及，多核处理技术的应用也逐渐成熟，它使得系统的性能有了极大的提升，同时，微内核操作系统比一般的宏内核操作系统体积更小安全性更高，越来越多的研究人员开始从事这方面的研究，将微内核操作系统的功能引入到多核处理器上，因此目前多核处理器环境下的操作系统微内核功能的研究已经成为了主流发展方向。

尽管目前有不少研究人员对多核处理器上的任务调度算法和机制进行了研究，然而却没有形成标准的规范和成型的体系，因此多核处理器下操作系统的任务调度机制的方法和策略有待进一步研究和探索。

在嵌入式微内核操作系统中，系统的任务调度机制是核心功能，它直接关系着整个系统的运行性能，所有任务调度机制的好坏就决定了微内核操作系统在多核处理器上的运行性能。因此，本文针对这个问题，提出了一种多核处理器环境下嵌入式操作系统微内核任务调度机制的改进策略。本文首先介绍了多核处理器、嵌入式操作系统和微内核操作系统的发展趋势，并对当今国内外在多核操作系统方面的研究情况进行了介绍，接着，详细介绍了多核处理器的架构方式和调度原理，然后分析了常见的微内核操作系统 Mach、L4、seL4 的优缺点，在此基础上，本文进一步研究了当前主流的多核处理器任务调度算法和策略，基于这些理论基础，本文对多核操作系统的任务调度机制进行了两方面的改进，分别是任务分配模型和任务调度机制。任务分配模型的好坏直接影响任务的执行次序及系统的实时性，因为当高优先级任务到来时，如果分配的位置不对，就必然会影响任务执行的实时性。因此本文提出了基于优先级的抢占式任务分配机制，使高优先级的任务能够实时快速执行，同时给出了多核处理器下任务分配的通用规则。接着用全局队列调度和局部队列调度相结合的方法实现了任务的调度，提出了基于缓存队列的任务调度机制，为了使处理器的效率更高，对全局队列调度进行了改进，融合了快内核优先的动态迁移策略。这些都保证了系统的实时性和负载均衡性。

通过改进的任务调度机制思想，本文在 Simics 仿真平台中搭建了多核处理器平台和嵌入式操作系统，然后将改进的设计方法在这个平台上模拟实现，验证了本文设计方法的有效性。实验证明，改进的任务调度机制是系统的运行效率有了大大的提高，性能也提升了不少。

6.2 研究展望

本文通过对多核处理器环境下微内核操作系统的任务调度机制进行了研究，提出了改进的任务调度机制，它提高了系统的性能和实时性，满足了目前人们对操作系统的性能要求。但是随着科技的进步，高清视频、高清游戏及大型程序的出现，会对性能的要求更高，如果能进一步使用更加优越的改进微内核任务调度机制，就有必要对其进行深入的研究。本文认为，下一步的研究工作主要有以下两点：

（1）对多核处理器架构下，如何保证提升系统性能的同时减少处理器的能量消耗；

（2）研究多核处理器架构下，如何提高任务调度机制和任务间通信及传送数据的安全性，提高整个系统的安全性和可靠性。

参考文献

[1] Xiao-Hui Cheng, You-min Gong, Xin-zheng Wang. Study of Embedded Operating System Memory Management. ETCS'09: Computer Science, Published by the IEEE Computer Society, 2009:962～965

[2] Rafal Kolanski, Gerwin Klein. Formalising the L4 microkernel API. Proceedings of the 12th Computing: The Australasian Theroy Symposium-Volume 51. 2006

[3] Xiao-hui Cheng, Zeqiu Li, Fuyuan Xiao. An advanced inter-process communication model of micro-kernel[J]. In ICCDA'11: Proceedings of IEEE 3rd International Conference on Computer Design and Applications. 2011

[4] Gernot Heiser, Kevin Elphinstone, Ihor Kuz, Gerwin Klein, Stefan M. Petters. Towards Trustworthy Computing Systems: Taking Microkernels to the Next Level. ACM SIGOPS Operating Systems Review, 2007, 41 (4)

[5] Philipp Kupferschmied. Design and Implementation of a Microkernel-Based Operating System for NUMA Machines[D]. Karlsruhe: University Karlsruhe, 2006

[6] Sven Schneider. Multiprocessor Support for the Fiasco Microkernel[D]. Chemnitz: Technische Universität Chemnitz, 2006

[7] D. Kim, S. Lee, D. Shin. Design of the operating system virtualization on L4 microkernel. 2008 Fourth International Conference on Networked Computing and Advanced Information Management. School of Information and Communication Engineering Sungkyunkwan University, 2008.223～229

[8] Jan Stoess. Towards Effective User-Controlled Scheduling for Microkernel-Based Systems. ACM SIGOPS Operating Systems Review, 2007, 41 (4): 1～10

[9] Wang Chengjun. Research on the Microkernel Technology. 2009 Second International Workshop on Computer Science and Engineering. 2009(J):199-202

[10] Xiao-hui Cheng, Liang Zhang, A research of inter-process communication based on shared memory and Address-mapping. In ICCSNT 2011: Proceedings of Int. Conference on Computer Science and Network Technology, Harbin, China, 24-26, Dem 2011

[11] Jorrit N.Herder. Towards a true microkernel operating system[D]. Amsterdam: Vrije Universiteit, 2005

[12] Wang Guoqin, Xu Meihua. Research on Tightly Coupled Multi-Robot Architecture Using Microkernelbased, Real-Time, Distributed Operating System. 2008 International Symposium on Information Science and Engieering, 2008, 8～13

[13] Xiao-hui Cheng, Yong-jian Hu, You-min Gong. Design methodology of control system with mutual exclusion. PACIIA'08: Intelligent control, Published by the IEEE Computer Society, 2008:403～407

[14] Rami Matarneh. Multi Microkernel Operating Systems for Multi-Core Processors. Journal of Computer Science, 2009, 5 (7): 493～500

[15] 俞建德. 支持异构多核的嵌入式实时操作系统 SmartOSEK OS-M[硕士学位论文]. 杭州：浙江大学，2008

[16] 顾宝刚. 基于 VxWorks 的异构多核处理器软件系统的研究与设计[硕士学位论文]. 长沙：国防科学技术大学，2008

[17] 张荫芾. 基于多核处理器架构的嵌入式微内核操作系统的研究与设计[硕士学位论文]. 上海：上海交通大学，2009

[18] 张荫芾，应忍冬，周玲玲. 支持多核架构的微内核操作系统设计. 计算机工程，2009，35（23）:249～251

[19] 蒋建春，汪同庆. 一种异构多核处理器嵌入式实时操作系统构架设计. 计算机科学，2011，38（6）:298～303

[20] 肖学甲. 基于 AMP 架构的多核间任务同步与通信的设计与实现[硕士学位论文]. 西安：西安电子科技大学，2011

[21] 谢子光. 多核处理器间通信技术研究[硕士学位论文]. 成都：电子科技大学，2009

[22] 赵懿. 多核嵌入式系统的实时性研究[硕士学位论文]. 杭州：浙江大学，2007

[23] 李静. 基于多核的任务调度策略研究[硕士学位论文]. 哈尔滨：哈尔滨工程大学，2010

[24] 高金华. 基于多核的任务调度研究与实现[硕士学位论文]. 长沙：中南大学，2010

[25] 胡泊. 基于可配置处理器的异构多核线程级动态调度模型[硕士学位论文]. 上海：上海交通大学，2008

[26] 李彦冬，雷航. 多核操作系统发展综述. 计算机应用研究，2011，28（9）:3215～3218

[27] 董珍. 异构多核处理器的任务调度分配问题及算法研究[硕士学位论文]. 哈尔滨：哈尔滨工程大学，2010

[28] 彭蔓蔓，黄亮. 多核处理器中任务调度与负载均衡的研究. 微电子学与计算机，2011，28（11）:35～39

[29] 寇立涛. Linux 多核调度算法的优化与扩展[硕士学位论文]. 西安：西安工业大学，2010

[30] 黄志刚，陶旭东，潘振杰. 一种异构多核处理器体系机构的仿真软件. 沈阳理工大学学报，2010，29（6）：5～9

[31] 周本海，乔建忠，林树宽. 多核平台的并行实时调度与内存分配算法. 东北大学学报，2012，33（3）：357～376

[32] 刘俊海. 多核系统内存管理算法的设计与实现[硕士学位论文]. 天津：天津大学，2008

[33] 周怀洋. 多核系统上的调度策略建模与分析[硕士学位论文]. 上海：上海交通大学，2010

[34] 杜建军. 共享高速缓存多核处理器的关键技术研究[博士学位论文]. 重庆：重庆大学，2011

[35] 任胜兵，张万利，卢念，潘震宇. 基于 OVPsim 嵌入式系统多核仿真教学研究. 电子电气学报，2010，32（1）：92～95

[36] 林明亮，祝永新. 基于 SimpleScalar 的异构多核仿真. 微电子学与计算机，2007，24（7）：204～208

[37] 袁云. 基于多核处理器并行系统的任务调度算法的研究[硕士学位论文]. 上海：华东师范大学，2008

[38] 章承科. 基于多核处理器的实时操作系统的扩展[硕士学位论文]. 成都：电子科技大学，2006

[39] 徐子川. 基于多核处理器的温度感知实时调度算法研究[硕士学位论文]. 大连：大连理工大学，2010

[40] 杨帆. 基于多核平台的网络流量监测研究与优化[博士学位论文]. 北京：北京邮电大学，2011

[41] 覃中. 基于多核系统的线程调度[硕士学位论文]. 成都：电子科技大学，2009

[42] 吴佳骏. 多核多线程处理器上任务调度技术研究[博士学位论文]. 北京：中国科学院，2006

[43] 王旭涛. 基于异构多核处理器系统的任务调度算法研究[硕士学位论文]. 南京：南京邮电大学，2011

[44] 戴鸿君. 基于异构多核体系与组件化软件的嵌入式系统研究[博士学位论文]. 杭州：浙江大学，2007

[45] 徐恒阳. 龙芯多核平台上性能分析工具的设计与实现[硕士学位论文]. 合肥：中国科学技术大学，2011

[46] 陈云川. 面向多核处理器的嵌入式操作系统研究[硕士学位论文]. 成都：电子科技大学，2009

[47] 郭秀岩. 面向多核的多层次实时网络数据流调度技术研究[博士学位论文]. 合肥：中国科学技术大学，2011

[48] 黄振宝. 面向嵌入式异构多核体系的软硬件协同任务调度方法[硕士学位论文]. 杭州：浙江大学，2007

[49] 申建晶. 嵌入式多核实时操作系统研究与实现[硕士学位论文]. 成都：电子科技大学，2011

[50] 赵磊. 适应多核处理器的任务调度研究[硕士学位论文]. 哈尔滨：哈尔滨理工大学，2010

[51] 刘近光. 虚拟操作系统研究及在多核环境下的实现[硕士学位论文]. 北京：北京交通大学，2007

[52] 朱文俊. 一种新的多核体系结构中的线程调度算法极其模拟[硕士学位论文]. 兰州：兰州大学，2010

[53] 曹皓. 多核处理器体系结构下 Linux 调度机制的研究[硕士学位论文]. 内蒙古：内蒙古大学，2011

[54] 徐立超. 异构多核处理器的任务分配及能耗研究[硕士学位论文]. 长沙：湖南大学，2010

[55] 蒋建春，汪同庆. 异构多核处理器的任务调度算法. 计算机工程与应用，2009，45（33）：52～56

[56] 刘彦. 异构多核片上系统的任务调度及应用研究[博士学位论文]. 长沙：湖南大学，2009

[57] 胡新安. 支持动态任务调度的多核分布式操作系统设计[硕士学位论文]. 哈尔滨：哈尔滨工业大学，2011

[58] 诸利勇. 支持多核处理器的星载嵌入式操作系统的研究与实现[硕士学位论文]. 长沙：国防科学技术大学，2008

[59] 黄国睿，张平，魏广博. 多核处理器的关键技术极其发展趋势. 计算机工程与设计，2009，30（10）：2414～2418

[60] 唐伟杰. 微内核进程间通信的优化与实现[硕士学位论文]. 杭州：浙江工业大学，2008

个人简介、申请学位期间的研究成果及发表的学术论文

沈凡凡，男，1987 年 10 月出生于湖北省荆门市，2010 年 7 月毕业于三峡大学理学院信息与计算科学专业，获理学学士学位。2010 年 9 月考入桂林理工大学攻读硕士学位，师从黎卫红高级工程师，研究方向为嵌入式系统应用技术。

一、申请学位期间参加的研究项目：

1、项目来源：国家自然科学基金项目（61063001/F020207）

项目名称：嵌入式实时操作系统微内核体系结构及 IP 核方法研究

申请人： 程小辉

2、项目来源：广西研究生教育创新计划项目

项目名称：基于事件调度的嵌入式操作系统内核的研究与设计（2010105960812M33）

申请人：李泽球

二、在攻读硕士研究生期间获奖情况

（1）荣获 2011 年桂林理工大学“优秀研究生干部”

（2）荣获 2011 年广西壮族自治区“优秀硕士研究生”

（3）荣获 2012 年桂林理工大学优秀硕士研究生“二等奖学金”

（4）荣获 2012 年桂林理工大学“物联网下设备间通信关键技术研究”技成果“一等奖”

（5）荣获 2012 年“研究生国家奖学金”

（6）荣获 2013 年校“优秀毕业生”和广西壮族自治区“优秀毕业生”

三、读研期间发表论文：

（1）Xiaohui Cheng, Fanfan Shen, Design of the Wireless Sensor Network Communication Terminal Based on Embedded Linux, The 2nd IEEE International Conference on Software Engineering and Service Sciences, Beijing, China, 15-17, July 2011.

（2）Xiaohui Cheng, Fanfan Shen, Morihino KUGA, Design of the wireless temperature supervisory control system based on eZ430-RF2500, 2012 IEEE International Conference on Computer Science and Automation Engineering, Zhangjiajie, China, 25-27, May 2012.

（3）Weihong Li, Fanfan Shen, Xiaohui Cheng, Research on Node Repair Mechanisms in Wireless Sensor Networks, 2012 IEEE International Conference on Computer Science and Automation Engineering, Zhangjiajie, China, 25-27, May 2012.

致 谢

弹指一挥间，研究生的生活已经接近尾声，在这人生最重要的阶段，我学会了独立思考，独立解决问题的能力，开阔了视野，增长了知识，动手能力和学术研究水平都有了质的飞跃，在此，我衷心的感谢所有帮助过我的老师、同学和亲人。

首先我要感谢我的导师黎卫红老师，他治学态度严谨，学识渊博，对研究生指导认真仔细。同时我要特别感谢程小辉教授三年来细心的指导和深切的关怀，他为人随和、平易近人，对工作精益求精、认真负责，学术造诣很深，对我影响非常深刻。程老师不仅教会了我做科研的基本方法，还使我学会了为人处事的道理。本论文从选题、开题到写作的每个环节，程老师都细心指导，严格把关，付出了大量的心血。在这里，向我所有恩师表示最崇高的敬意和最深切的感谢！

其次我要特别的感谢邓昀、李泽球、陈伟、肖富元、周茂杰、张一哲、张良、巫湘林等师兄师姐，感谢你们平时对我的帮助和关怀，给了我前进的方向和动力。同时感谢同门许安明、王小烨、窦萌萌和吴名欢，跟他们一起科研、一起学习，非常的开心和快乐，这份感情非常值得怀念。

最后我要感谢我的家人，感谢爸爸、妈妈在物质上和精神上的支持，是你们给了我前进的动力。感谢我的朋友，是你们的帮助是我顺利完成学业！

愿所有帮助我、关心我、支持我、爱护我的人永远幸福快乐，祝福他们一切称心如意！

面向多核处理器的嵌入式操作系统微内核任务调度机制的研究

作者： 沈凡凡
学位授予单位： 桂林理工大学