Java 芯片操作系统的模型和实现*

陈 虎[1], 戴 葵[2], 杨晓东[2], 胡守仁[2]

[1](第一军医大学 医学工程系,广东 广州 510515);
[2](国防科学技术大学 计算机科学与技术系,湖南 长沙 410073)
E-mail: tommychen74@yahoo.com
http://www.fimmu.edu.cn

摘要: 提出的 Java 芯片操作系统是一个基于 Java 芯片系统的、针对嵌入式应用的面向对象操作系统.它由服务器、内核和转换层等 3 种组件构成.这些组件的抽象特征构成了其基本框架.特定组件继承了上述的特征,并进行特化.与经典操作系统相比,该操作系统更容易开发和维护,并具有更好的可伸缩性、代码的可重用性和可移植性.介绍了该操作系统的结构、运行模型以及实现技术,并与相关的工作进行了比较.

关 键 词: Java;操作系统;Java 芯片
中图法分类号: TP316 文献标识码: A

Java 芯片操作系统(Java chip operating system,简称 JCOS)是一个基于 Java 芯片的针对嵌入式应用的面向对象操作系统.在结构上,它由多个相对独立的服务器、转换层和内核等组件构成.在设计方法上,采用抽象出各种组件共同特征(包括外部接口、内部结构、运行模式和交互关系等)构成系统运行模型的方法.这些特征由一些基础类实现.特定的组件可以通过继承基本类而利用共同特征,其特性通过重载实现.在此结构和设计方法的指导下,Java 芯片操作系统具有更好的可维护性、可伸缩性和代码可重用性,并使组件的设计更加容易.目前该系统已经具有对象存储管理、线程和锁、文件系统和基本 I/O 等功能.

本文第 1 节介绍 JCOS 的运行环境和结构.第 2 节描述 JCOS 的运行模型.第 3 节讨论 JCOS 的具体实现.最后与相关的工作进行了比较.

1 JCOS 的运行环境和结构

Java 芯片系统包括改进的 Java 开发包(Java development kit,简称 JDK)[1]、Java 芯片和 Java 芯片操作系统,结构如图 1 所示.JCOS 与外部的接口包括硬件接口和 JDK 接口.

・硬件的接口

Java 芯片是一个能直接运行 Java 虚拟机(Java virtual machine,简称 JVM)[2]指令的微处理器.JCOS 与它的接口包括:

JCOS 通过调用硬件实现的本地方法控制硬件;

硬件通过外部中断、异常、指令自陷等方式激活操作系统.

* 收稿日期: 1999-11-12; 修改日期: 2000-10-16
基金项目: 国家自然科学基金资助项目(69803013)

作者简介: 陈虎(1974－),男,江苏南京人,博士,讲师,主要研究领域为计算机体系结构,Java 实现技术;戴葵(1968－),男,湖北恩施人,博士,副教授,主要研究领域为计算机体系结构,神经网络;杨晓东(1936－),男,吉林长春人,教授,博士生导师,主要研究领域为并行计算机体系结构,故障检测;胡守仁(1926－),男,江西江山人,教授,博士生导师,主要研究领域为新一代计算机体系结构.

当发生中断(异常或自陷)时,硬件的动作包括关闭处理器的外部中断、查找系统处理例程入口、保存当前方法的运行框架,并在当前线程的堆栈空间和局部变量空间中创建新的运行框架.当中断处理程序运行结束时,使用 return 指令返回到当前运行方法的框架.

・JDK 接口

应用程序通过特定的应用程序接口(application program interface,简称 API)调用操作系统.JDK 提供的 API 是 Java 语言 API 的事实标准.JDK 中实现了部分 API 功能,其余功能将由 JCOS 实现.JCOS 实现的 API 可以分成两种类型:

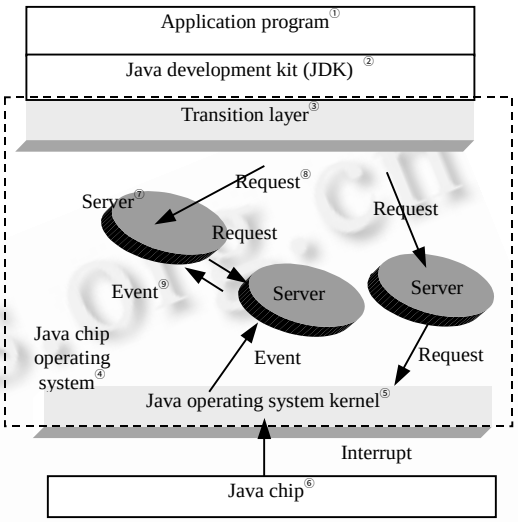
(1) 非阻塞类型.它与其他线程或硬件事件无关,无需等待即可直接完成;

(2) 阻塞类型.它与其他线程或硬件中断相关,可能会使当前线程转入等待状态.例如从外部设备中读取数据、线程之间的同步等.这类 API 涉及操作系统中各种资源和线程状态的转换,实现机制复杂.

Java 芯片操作系统由 3 种组件组成:转换层、若干服务器和内核,并通过请求和事件进行交互.

・转换层

应用程序在发出 API 调用时并不需要知道系统中服务器的结构和功能,即转换层保证了操作系统的内部结构对应用程序透明.转换层将 API 调用转化为对内部服务器的请求,还负责监控相应请求是否被满足或阻塞,并控制线程状态的转换.



①应用程序,②Java 开发包,③转换层,④Java 芯片操作系统,⑤Java 操作系统内核,⑥Java 芯片微处理器,⑦服务器,⑧请求,⑨事件,⑩中断.

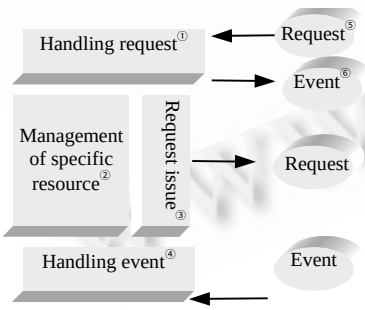
Fig.1 The environment and internal structure of Java chip operating system

图 1 Java 芯片操作系统的运行环境和内部结构

・服务器

系统中包括存储器、定时器、文件系统、锁系统等软硬件资源.这些资源分别由相应的服务器管理.它们能接受对所管理资源的请求,并为之服务,必要时还可以向其他服务器提出请求.这些服务器具有相同的内部结构、外部特征、运行模式和交互关系.

服务器的内部结构如图 2 所示,包括:



①请求处理,②预处理调用 API 函数,③提交请求,④事件处理,⑤请求,⑥事件.

Fig.2 The structure of sever

图 2 服务器的结构

・管理特定资源;

・处理转换层或其他服务器发送的请求;

・向其他服务器或内核提出请求;

・处理内核和其他服务器发送来的事件;

・向其他服务器或转换层发送事件.

在不同的服务器中,管理资源的部分并不相同,但它们被相同的外部特征所封装.这些外部特征包括接受请求、回送事件、发送请求和接受事件.所有服务器都遵循接受请求→处理请求→回送事件的运行模式.

服务器之间的关系包括 3 个方面:

(1) 它们是相互独立的.服务器按照约定接受外界发来的请求,完成请求的细节则被封装在服务器内部;

(2) 服务器之间通过请求和事件进行交互.即由源服务器向目标服务器发出特定的请求,当目标服务器完成请求后,将向源服务器回送事件;

(3) 服务器之间的请求关系可以迭代.服务器在完成请求的过程中可以向其他服务器发出请求,而且允许

服务器并发地或串行地向一个或多个服务器提出请求.

・内核

内核是一种特殊的服务器.它的主要功能是与硬件进行交互:

(1) 根据其他服务器发送来的请求直接控制硬件;

(2) 处理硬件产生的中断,并将它转换成事件发送到相应的服务器中.

2 JCOS 的运行模型

在 JCOS 中,各种服务器、转换层、内核和请求/事件分别在内部结构、外部接口、控制流程和交互关系等方面具有共同特征.下述的 JCOS 运行模型[3]描述了这些特征和相应的算法.特定组件可以继承上述特征,并进行特化.

JCOS 运行模型包括线程模型、服务和请求、请求满足过程、服务器模型、内核模型和转换层模型等部分.

2.1 线程模型

Java 芯片系统中的线程有 3 种状态:当前运行状态、就绪状态和阻塞状态.具有相同状态的线程分别组成了就绪队列、阻塞队列和当前线程.线程模型提供了 3 种方法以实现线程状态的转换:

(1) Schedule:从就绪队列中选择线程运行;

(2) Giveup 当前线程放弃 CPU,加入阻塞队列,并由 Schedule 从就绪队列中选择线程运行;

(3) Resume(Thread *t*)将线程 *t* 从阻塞队列中移出,并加入就绪队列.

2.2 请求与事件

请求与事件是 JCOS 中服务器之间交互的基础.当源服务器需要目标服务器完成特定功能时,将向目标服务器发出请求.目标服务器完成请求后,将回送事件通知源服务器.

请求中包含提出该请求的线程和源服务器、服务号、请求完成的当前状态、接受请求的目标服务器以及为完成该请求而由其他服务器提出的子请求集合.

事件有两种类型:与特定请求对应的事件和没有请求对应的事件(无主事件).无主事件可以表示外部的异步输入(如键盘输入)和系统中的特殊事件(如系统退出事件).事件中包括对应的请求(无主事件为空)、事件类型和发送该请求的目标服务器.

2.3 请求满足过程

由于服务器的请求迭代特性,它在完成特定请求的过程中可能还需要向其他服务器发出请求.服务器之间的交互特性可以用包含在服务器中的请求满足过程描述.请求满足过程类似于有限状态自动机,包括状态、状态转移函数、每个状态的服务器内部方法(向其他服务器提出请求 IssueRequest 和服务结束 Final)和参数(目标服务器和请求号)以及回送事件类型.其中,自动机的状态转移由服务器接收的事件所决定.

该自动机还应满足下述约束条件:在终结状态不会有提交请求的方法,且在终结状态才会有结束动作.

根据是否等待事件和是否向其他服务器发送请求,请求满足过程可以分成直接过程、等待过程和迭代请求过程:

・直接过程的特点是无须等待事件,也不向其他服务器发送请求;

・等待过程的特点是需要等待事件的发生,但不需要再向其他服务器提出请求;

・迭代请求过程的特点是需要向其他服务器提出请求才能完成的请求服务过程.

2.4 服务器

服务器提供的服务通过请求号区别,并有特定的请求满足过程与之对应.它还能并发地处理多个线程提出的多个请求和多个事件,其内部结构包括正在处理的请求集合 Rset 和等待处理的事件集合 Eset.

服务器 S 提供给外界的接口有 4 种:

(1) 接受请求 Rrequest.当服务器接受服务请求 r 后,它将根据 r 中的服务号找到相应的请求满足过程.如果该过程是直接过程,则处理后立即返回;否则将 r 加入 Rset 中,并将初始状态填入 y 的状态域中,还将执行初始状态的服务器内部方法.

(2) 撤消请求 Crequest.当服务器被要求撤消请求 r 时,它将 r 从 Rset 中删除,并调用其子请求所对应服务器的撤消请求方法.

(3) 接受事件 Revent.当服务器接收到事件 e 后,它将 e 加入 Eset 中,找到和 e 对应的请求 r,并用 resume 方法唤醒 r 对应的线程.

(4) 处理事件 HEvent.当服务器处理事件 e 时,将首先找到和 e 对应的请求 r,随后根据 e 和 r 的请求满足过程转移到新的状态,并执行该状态的服务器内部方法.如果已经到达结束状态,将向服务器回送相应的事件.最后它将从 Eset 中撤消请求 e.

2.5 服务器的内部方法

服务器的内部方法有两种:

(1) 向其他服务器提出请求(issue request).此时将创建新的请求对象 r1,将其加入源请求 r 的子请求集合,并以 r1 为参数调用 r1 对应目标服务器的接受请求方法 Rrequest.

(2) 结束(Final).此方法的功能是撤消请求 r 中所有未完成的子请求.

2.6 内核模型

内核是一种特殊的服务器,结构与服务器基本相似,具有接受或撤消请求的功能,但有 3 点区别:

(1) 内核不再向其他服务器提出请求,即请求满足过程都是等待过程;

(2) 硬件中断对应于事件,且中断处理功能取代了服务器中接受和处理事件的功能;

(3) 增加了缺省中断处理服务器.

其中断处理方法 InterruptHandle 的处理流程为:在接收到硬件中断之后,将首先检查是否有等待该事件的请求,如果有,则回送相应的事件,并从 Rset 中撤消该请求,否则将向缺省服务器发送事件.

2.7 转换层模型

转换层也是一种特殊的服务器,有请求满足过程,并可以接受事件,但有以下不同:

- 接受应用程序 API 调用的接口(RAPI)取代了接受或撤消请求的外部接口;
- 服务结束时不是发送事件而是返回应用程序的运行环境;
- 未处理事件的外部接口;
- 控制当前线程的状态并查询相关的服务器;
- 其请求集合是与线程相关联的.

转换层的 IssueRequest 方法与一般服务器有所不同:向提交请求 r 时,并不将 r 加入子请求集合中,而是加入请求集合 Rset 中.Final 中的撤消请求操作也是针对 Rset 进行的.

由于服务器的请求迭代功能与一个 API 相关的所有请求可以通过请求中的子请求域构成一个请求树.请求树的初始状态仅有一个根结点,即用户提出的 API 调用.因此,当转换层向其他服务器发出请求时,新的请求就成为根结点的子结点.当服务器再向其他服务器发出请求时,新的请求也将成为原请求的子结点.当处于子结点的请求得到满足时,该结点将从请求树中删除.当服务器调用撤消请求方法(CRequest)时,以该请求为根结点的子请求树也将从请求树中删除.请求树的上层请求是否能够得到满足取决于下层请求的完成情况,所以事件的发生、传播和处理在请求树中是自下而上的.这就要求调用子结点所在服务器的 HEvent 方法要早于父结点.JCOS 在转换层构造一个包含所有请求的双向请求树链表:当每个请求向其他服务器提出子请求时,将子请求加入请求树链表尾部.

与一般服务器的 Rrequest 方法相比,转换层的 RAPI 方法也有相应的请求满足过程自动机,并且也将按照接收到的事件控制自动机的状态转移.另外,它还要控制线程的状态.

当RAPI方法向其他服务器发送的请求不能立刻得到满足时,它将调用Giveup方法使当前线程进入阻塞状态.请求树中相关服务器在接受到与本线程相关的事件(REvent 方法)时,将唤醒本线程.当该线程重新运行时,按照请求树链表的逆序依次调用各个请求对应的服务器的处理事件方法.当转换层的请求满足自动机达到结束状态时,它将撤消 Rset 中该线程所有未完成的请求.

从上述介绍可以看出,一般的服务器提供 4 种对外的接口(RRequest,CRequest,REvent 和 HEvent)和两种内部方法(IsseuRequest,Final).转换层和内核虽然也是服务器,但在上述的接口和内部方法上有所不同,详见表 1.明确它们的区别将有助于确定实现它们的类之间的继承关系.

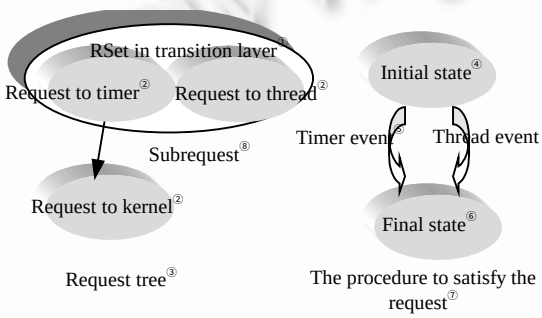
Table 1 The comparison of the external interface and internal methods among sever, kernel and transition layer

表 **1** 服务器、内核和转换层的外部接口和内部方法比较

Common sever①	Transition layer②	Kernel③
Rrequest method⑦	RAPI method⑦	The same⑤
Crequest method⑦	No exist④	The same
Revent method⑦	The same	InterruptHandle method⑦
Hevent method⑦	No exist	
IssueRequest method⑦	Modified⑥	No exist
Final method⑦	Modified	The same

①一般服务器,②转换层,③内核,④不存在,⑤与一般服务器相同,⑥有改动,⑦上面已介绍的方法.

2.8 一个实例



①转换层的 RSet,②对 Timer(Thread,Kernel)的请求,③请求树,④初始状态,⑤Timer(Thread)事件,⑥结束状态,⑦转换层的请求满足过程⑧子请求.

Fig. 3 The implementation of wait (time)

图 3 Wait(time)调用示意图

以 JDK 中的 wait(time)[1]为例.该调用的含义是,当前线程处于等待状态,直到其他线程唤醒它或线程等待了 time 指定的时间为止.在用户调用该系统调用后,转换层将其转化成两个并发请求,分别交给线程服务器(thread)和定时服务器(timer).其中定时器还将向内核提出请求.然后线程放弃 CPU 进入等待状态.

当定时器请求首先满足时,该线程将被唤醒.被唤醒后的线程将依次调用 Kenel,Timer 和 Thread 的事件处理方法,并根据转换层的请求满足过程判断是否为终结状态,随后将撤消 Thread 服务器中的请求并返回应用程序.它的请求树和转换层的请求满足过程如图 3 所示.

2.9 Java芯片操作系统模型的特性

JCOS 运行模型具有以下特性:

·整个操作系统由独立的服务器、转换层和内核等组件组成.其中服务器负责管理特定的资源,转换层负责启动请求序列、监控请求状态和控制线程状态,内核负责与硬件交互和处理硬件中断.

· 系统中各种组件都具有公共的外部接口、内部结构、运行模式和交互机制.

· 事件接受与处理的功能分开,只有简单的事件接受过程占用其他线程的时间,处理请求的主要过程都在本线程的时间片内完成,线程之间具有更好的公平性.

·服务器之间的交互采用类似于有限自动机的方式描述,服务器可以并发或串行地向其他服务器提出请求,增强了服务器之间交互的灵活性.

3 JCOS 的实现

3.1 类的层次结构

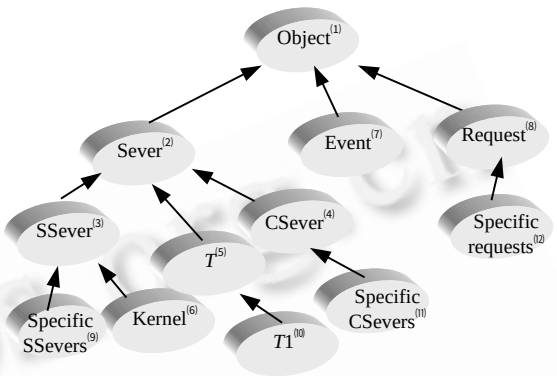
JCOS 中类的层次结构如图 4 所示.

· Object 类是所有类的超类.

· 服务器(sever)类是一个抽象类.说明了服务器的外部接口和内部方法,以及请求满足过程自动机.

· 简单服务器(SSever)类和复杂服务器(CSever)类继承了 Sever 类.其中 SSever 类中的请求满足过程都是直接过程或等待过程,CSever 类中的请求满足过程包含迭代请求过程.它们中实现了 Sever 类中的服务器外部接口和内部方法(SSever 类中进行了简化).各种特定服务器类将根据其请求满足过程的特点分别继承 SSever 类或 CSever 类,并实现请求满足过程自动机以及管理特定资源的功能.

· 转换层(T)类继承 Sever 类.它实现了 RAPI 方法,重载了 IssueRequest 和 Final 方法.各种转换层类都是 T 类的子类.



(1)Object 类,(2)服务器类,(3)简单服务器类,(4)复杂服务器类,(5)转换层,(6)内核类,(7)事件类,(8)请求类,(9)特定简单服务器类,(10)特定转换层类,(11)特定复杂服务器类,(12)特定请求类.

Fig 4 The inheritance tree of the classes in JCOS

图 4 JCOS 中类的层次关系

· 核心(kernel)类继承了 SSever 类,并实现了 InterruptHandle 方法.

· 各种类型的请求和事件类分别是请求(request)类和事件(event)类的子类.各种特定请求类中增加了各种相应的请求参数和返回结果.

JCOS 中 Sever,SSever,CSever,T,Event,Request 等类成为各种组件的基本“模板”,可以为各种特定组件继承.在特定服务器类、转换层类中需要实现的内容只包括请求满足过程自动机和特定资源的管理.系统运行时,各种特定服务器都被创建为对象.虽然它们的类型不同,但都按照 Sever 类的外部接口被调用,由对象方法的重载机制自动实现其特化.

3.2 JCOS中的服务器、转换层和内核

目前已实现的 JCOS 结构如图 5 所示(连线表示组件之间的调用关系),主要包括对象存储管理[4]、线程和锁、文件系统和基本 I/O 等子系统.

对象存储管理子系统

对象存储管理主要由对象创建服务器、垃圾回收服务器和相应的转换层构成:

· 对象创建服务器.它将接受对象创建请求、管理物理存储器、填写对象头和初始化对象体.其中采用分页的方式管理物理存储器.当物理存储空间不足,它将向垃圾回收服务器提出垃圾回收请求.当填写对象头中的对象类型时,还需要与类服务器交互.

· 垃圾回收服务器,采用“三色法”标记对象和同步垃圾回收策略,并支持快速精确扫描.

文件子系统

文件系统的磁盘存储格式和系统内数据结构类似于 UNIX 系统,可支持文件打开、关闭、读/写等基本操作,由文件系统服务器、磁盘缓冲服务器和磁盘服务器等组件组成.

· 文件系统服务器.它用来维护文件表和索引结点表,并将用户的请求转化为相应的磁盘扇区号和偏移量交给磁盘缓冲服务器.

· 磁盘缓冲服务器.它管理数据缓冲和索引节点缓冲.如果文件系统服务器提交请求的扇区不在当前缓冲

中,它以扇区号为参数向磁盘服务器发出请求.

· 磁盘服务器.它将根据扇区号操纵磁盘的 I/O 操作,并处理磁盘 I/O 中断.

线程和锁子系统

· 类服务器.它将维护系统类表,并支持对类特性(例如超类、名字)的查询.在需要装入新类时,它还要与文件服务器交互.

· 线程服务器.它管理各种状态下的线程队列,接受线程状态改变请求,并在当前线程挂起时根据优先级选择合适的线程运行.

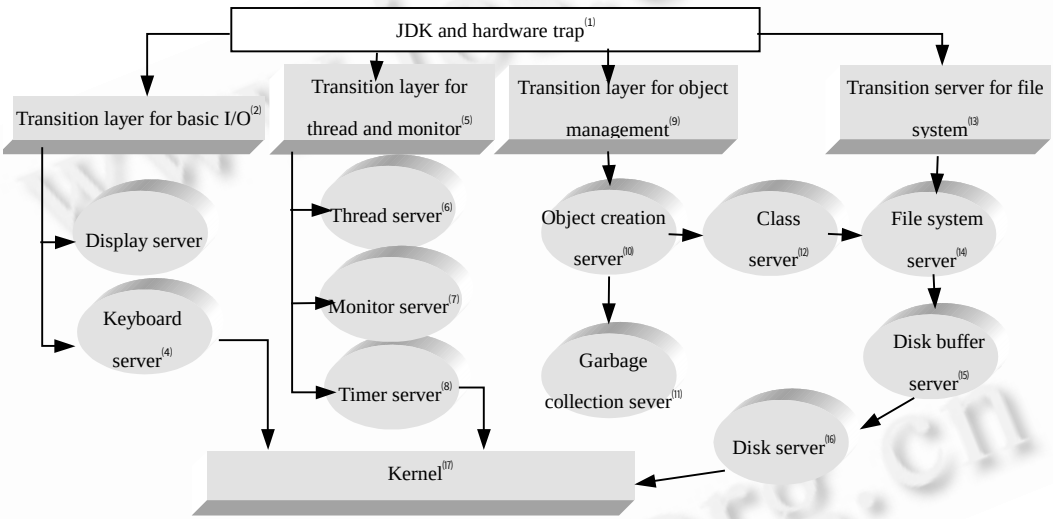
· 锁服务器.它维护当前上锁对象表,支持对对象上锁和解锁的操作.

· 定时器服务器.它接受定时请求,维护相应的定时队列,并设置硬件定时器和处理定时器中断.

基本 **I/O** 子系统

· 显示服务器.它维护当前 40 列/20 行的显示缓冲区,并将用户的输出显示到屏幕.

· 键盘服务器.它能处理键盘输入中断,读出键盘的输入值,并将其加入键盘输入队列,还可以根据输入顺序向其他服务器提供键盘输入.



(1)JDK 和硬件自陷,(2)基本 I/O 转换层,(3)显示服务器,(4)键盘服务器,(5)线程和锁转换层,(6)线程服务器,(7)锁服务器,(8)定时服务器,(9)对象存储管理转换层,(10)对象创建服务器,(11)垃圾回收服务器,(12)类服务器,(13)文件系统转换层,(14)文件系统服务器,(15)磁盘缓冲服务器,(16)磁盘服务器,(17)内核.

Fig.5 The realization of JCOS

图 5 JCOS 的实际构成

内核除了完成模型中所述功能以外,还负责指令运行时的异常处理,并管理系统中的服务器.上述这些已实现的组件构成了一个可完成 Java 语言基本 API 的操作系统.

3.3 JCOS中的其他问题

上述模型需要经常创建各种请求对象和事件对象,会带来较大的开销.JCOS 从两个方面解决这个问题:首先,所有的直接过程都直接调用相关的方法.其次,在服务器中创建相应的请求对象池和事件对象池,采用一次分配多次使用的方法以减少创建对象的开销.

Java 语言中没有进程的概念.而且所有线程共享一个堆空间.JCOS 中线程之间的数据共享通过共享堆空间中的对象实现.

在 Java 芯片系统中,硬件不仅提供 JVM 指令的执行功能,还提供了一种新的接口——本地方法.它用面向对象中的类方法将硬件提供的非标准指令功能封装起来.这样,整个操作系统都可以用 Java 语言完成,具有良好的可移植性,而且不用开发特殊的编译器.

引导程序的主要功能是初始化内存、内核、服务器、转换层、中断/异常处理程序、调用类初始化方法、创建基本线程等.一个特殊的链接程序将系统引导程序转化成 ROM 内存映像.微处理器开始执行时将自动调入固化在约定地址上的线程运行环境,执行引导程序.

4 相关工作比较

经典的操作系统一般为层次结构或微内核结构[5~7],采用面向过程语言实现(如 C 语言).它们一般以功能为中心划分组件,即某方面的 API 功能都是由相应的组件实现.在这样的结构中,至少有以下两方面因素导致组件复杂,难以设计、维护和修改:

首先,某些 API 需要使用多种资源(例如前面所述的 wait()功能),或者同一种资源可能由不同组件中的 API 使用,这样使得同一资源的管理被分布在不同的组件中.其次,具体 API 的实现过程、资源的管理和线程(进程)状态的转换都混合在同一组件内部.

在设计方法上,实现经典操作系统的面向过程语言不能提供有效的系统抽象方法,不能抽象出各种组件在外部接口、运行模式和交互关系等方面的公共特征.这样就会导致系统内各种组件之间的接口复杂且不"规则",一般用户难以理解、修改和扩充.

一些面向对象操作系统(例如 Choices[8]系统)利用面向对象中继承的特性,用一组虚类构成一个特定子系统的框架.但是它只提供子系统级抽象,并且基于传统的层次结构.

已有的 Java 操作系统[9,10]都是在经典操作系统之上通过扩展 Java 虚拟机中的本地方法库实现(例如 SUN 公司的 JavaOS[11,12]中包含了三分之一的 C 语言代码和少量汇编语言代码).在结构上,它们都采用了面向功能的层次结构或微内核结构,与一般的操作系统无异.在设计方法上,也没有体现出面向对象设计中对系统进行抽象的优越性.

与上述工作相比,JCOS 采用基于面向资源的服务器结构,其组件的共同特征确定,而且完全采用 Java 语言实现.所以它具有以下特点:

· 由于面向资源划分组件,服务器的功能比较单纯.另外,API 和服务器接口的差别处理以及线程状态转换都被剥离到转换层,所以服务器更容易实现、维护和修改;

· 系统中各种组件的外部接口、内部结构、运行模式和交互关系等公共特征已经由系统中若干基本类描述和实现,组件的设计者只要实现服务器内部资源的管理和请求满足过程的自动机,不仅减轻了设计者的负担,而且提高了代码的可重用性;

· 服务器之间利用面向对象机制进行封装,而且是通过标准的外部接口进行交互,服务器之间具有更好的独立性,也便于根据系统的需要对其进行裁剪;

· 整个操作系统都用 Java 语言完成,并使用标准的 Java 编译器编译,保持了 Java 程序易于设计、维护和二进制代码级可移植性等优点;

· 服务器的功能与 API 分离,使得用户可以更加容易地增加新的 API.

5 总 结

本文提出了 Java 芯片操作系统的结构、运行模型和实现技术.该操作系统的组件易于开发,并具有更好的可维护性、可伸缩性、可移植性和代码可重用性.目前,JCOS 已经能在我们设计的 Java 芯片模拟器上运行,并能支持 JDK1.1.2 中的基本类库(包括对象存储管理、线程和锁、简单的文件操作等).我们将在原有模型的基础上开发出更多的操作系统组件,以满足更多应用场合的需求.

References:

[1] Gosling, J., Joy, B., Steele, G. The Java Language Specification. New York: Addison-Wesley, 1996.

[2] Lindolm, T., Yellin, F. The Java Virtual Machine Specification. New York: Addison-Wesley, 1996.

[3] Chen, Hu. The research of the crucial technologies in JE-Java chip system [Ph.D. Thesis]. Changsha: National University of Defence Technology, 2000 (in Chinese).

[4] Qin, Ying. The research and implementation of the object storage management in JE-Java chip system [MS. Thesis]. Changsha: National University of Defence Technology, 2000 (in Chinese).

[5] Lieilthe, J. Toward real microkernels. Communications of the ACM, 1996,9(39):70~77.

[6] Fleisch, B.D. Workplace microkernel and OS: a case study. Technical Report, TRCS-98-4, University of California, Riverside, 1998.

[7] Ghormley, D.P. Slic: an extensibility system for commodity operating system. In: Proceedings of the USENIX Annual Technical Conference. New Orleans, Louisiana, 1998. 56~62.

[8] Campbell, R., Islam, N., Raila, D., *et al*. Designing and implementing CHOICES: an object oriented system in C++. Communications of the ACM, 1993,9(36):117~126.

[9] Back, G., Tullmann, P., Stoller, L., *et al*. Java operating system: designs and implementation. Technical Report, UUCS-98015, University of Utah, 1998.

[10] Montague, B.R. JN: an operating system for an embedded Java netwrk computer. Technical Report, UCSC-TRL-96-29, University of California, Santa Cruz,1996.

[11] SUN Microsystem. Java OS Whitepaper, http://www.java.com.

[12] O'Connor, J. M., Tremblacy, M. Pico-Java: the Java virtual machine in hardware. IEEE Micro, 1997,2(17):45~53.

附中文参考文献:

[3] 陈虎.JE-Java 芯片系统中关键技术的研究[博士学位论文].长沙:国防科学技术大学,2000.

[4] 秦莹.JE-Java 芯片系统对象存储管理的研究和实现[硕士学位论文].长沙:国防科学技术大学,2000.

Model and the Implementation of a Java Chip Operating System*

CHEN Hu[1], DAI Kui[2], YANG Xiao-dong[2], HU Shou-ren[2]

[1](*Department of Medical Engineering, The First Military Medical University, Guangzhou*, 510515, *China*);

[2](*Department of Computer Science and Technology, National University of Defence Technology, Changsha*, 410073, *China*)

E-mail: tommychen74@yahoo.com

http://www.fimmu.edu.cn

Abstract: A Java chip operating system (JCOS) is presented in this paper, which is an object-oriented operating system for embedded applications based on Java chip. JCOS is composed of three kinds of components including severs, kernel and transition layers. The model of JCOSis constituted by the abstract characteristics of these components. The specific components inherit above characteristics and specialize their features. By this technology, JCOS is developed and maintained easily, and has better scalability, portability and code reuse ability. Architecture , model and implementation of JCOS are introduced in this paper. At last, JCOS is compared with some related work.

Key words: Java; operating system; Java chip

* Received November 12, 1999; accepted October 16, 2000
Supported by the National Natural Science Foundation of China under Grant No.69803013