·计算机工程与应用·

内核级并发通信的研究

丘志杰，刘心松，刘 丹，刘 谐

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】提出了一种内核级并发消息通信机制。该机制采用对象传送协议和动态线程池技术，并通过会话控制完成数据的收发过程，将线程池设计为二级阻塞队列来暂缓线程的撤销过程，从而保证既能够快速响应客户请求，又可减少因频繁创建和撤销线程而消耗系统的资源和时间，为分布式操作系统提供了高效可靠的通信服务。

关 键 词 并发通信； 内核线程； 会话； 动态线程池

中图分类号 TP393； TP316.4 文献标识码 A

Study of Kernel-Level Concurrent Communication

QIU Zhi-jie，LIU Xin-song，LIU Dan，LIU Xie

(School of Computer Science and Engineering, Univ. of Elec.Sci.& Tech.of China Chengdu 610054)

Abstract This paper presents a new concurrent communication mechanism, which uses object Transfer protocol, dynamic thread-pool technology and session mechanism to send and receive data. By designing a two-level blocked queue for the thread pool to delay the process of destroying thread, this mechanism can response the client request quickly, and the system resource and time consumed by creating and destroying thread can be brought down. An efficient and reliable communication service for the distributed operating system can be provided by this mechanism.

Key words concurrent communication; kernel thread; session; dynamic thread pool

在分布式系统中，各节点之间通过交换信息而相互协作，构成一个有机整体。通信是分布式系统的基础，其效率高低将决定系统的整体性能。目前有各种各样的消息传递模型，例如发布-订阅模型[1]、FIFO通道模型[2]、消息中间件通信平台。但是，这些消息模型效率较低，实现相对复杂，而且都是面向应用的，不适合内核下开发的分布式操作系统使用。

本文提出了一种Linux内核级基于会话的并发消息通信机制(Linux Kernel Concurrent Communication，LKCC)。LKCC以面向非连接的用户数据报协议(User Datagram Protocol，UDP)为基础，在UDP协议之上采用对象传输协议(Object Transfer Protocol，OTP)[3]，并通过会话控制来完成数据的可靠传输；采用动态线程池技术并发处理客户请求，极大地提高了处理能力；封装底层通信细节，为分布式操作系统的其他模块提供了一个统一透明的调用接口；兼顾效率和可靠性的要求，可以为分布式系统提供高效和可靠的通信服务，在分布式操作系统中得到了很好的应用。

1 LKCC的设计基础

作为分布式系统底层的通信平台，LKCC为分布式操作系统的其他内核模块提供通信服务。在内核下实现LKCC，可以减少在用户态与内核态之间切换所产生的系统开销和数据的拷贝次数，因而比在用户层实现速度更快；且LKCC采用多线程的方式，效率更高。LKCC以OTP作为数据传送的基础协议，并结合使用会话机制，能够变异步为同步，变不可靠为可靠，使其他模块从烦琐复杂的通信过程中解脱出来。

1.1 Linux内核线程

Linux内核线程是一个能被独立调度的可执行上下文[4]，它在内核空间被创建，不执行用户程序，完全

收稿日期：2003 – 06 – 10

作者简介：丘志杰(1979 –)，男，硕士，助教，主要从事分布式并行操作系统的方面的研究.

运行在内核态下，只执行内核函数。调用kernel_thread()可以创建内核线程，该函数定义为int kernel_thread(int (*fn)(void*)，void*arg，unsigned long flags)，其中fn是线程要执行的内核函数；arg是传递给fn的参数；flags主要是设置一些标志位，规定子线程具有怎样的特性，例如共享父线程的哪些资源。子线程一旦创建成功，就可以接受内核的调度而运行。

1.2 OTP协议

OTP协议是以简单文件传输协议(Trivial File Transfer Protocol，TFTP)为原型[5]，并对TFTP进行改进的通信协议。它定义了RRQ、WRQ、DATA、DATAEX、ACK、ERR和PACKET7种报文格式。利用该协议可以实现SendData，GetData和PublishData 3种数据传输方式。客户机在调用SendData向服务器发送数据时，首先发送WRQ请求报文。服务器收到WRQ报文后，给客户机返回ACK应答报文，同意客户机发送数据。此后，通信双方通过发送DADA和ACK报文完成数据的发送－应答过程，如图1a所示。客户机在调用GetData()从服务器读取数据时，首先发送RRQ请求报文。服务器收到RRQ报文后，根据报文中的信息准备好客户机要读取的数据，然后向客户机发回DATAEX报文，其中包含第一块数据。此后，通信双方通过发送DADA和ACK报文完成数据的发送－应答过程，如图1b所示。客户机调用PublishData()可以向所有服务器广播数据，数据封装在PACKET报文中。服务器收到PACKET报文后，从中提取数据做处理。如图1c所示。

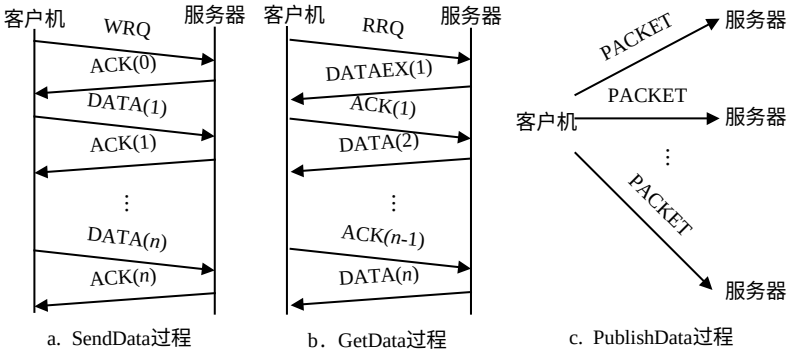

图1　发送数据、读取数据、发布数据过程

2 LKCC的设计

2.1 基于会话的通信

LKCC采用会话方式来控制点对点的数据传送。会话描述了通信双方的一次通信过程，其中记录相关数据信息和控制信息，通信双方通过维护各自会话中的信息来控制数据收发过程。LKCC引入会话机制，用户不必关心数据收发的控制细节，通信过程对用户完全透明，用户只需调用相应的通信接口。基于会话的点对点通信过程如图2所示。(1) 节点*a*调用SendData或GetData发起一次通信过程从而在节点a上建立客户端会话a；(2) 节点*a*向*b*发送RRQ或WRQ请求报文；(3) 节点*b*收到RRQ或WRQ请求报文时，同样建立对等的服务器端会话*b*；(4) 节点*a*与*b*通过维护各自会话中的信息，按照图1中所示的过程发送DATAEX、DATA、ACK等报文来完成数据收发。数据传送完毕后，通信的两个节点撤销各自的会话，通信过程结束。在通信过程中，通信双方都存在着超时重发机制。LKCC定义了超时值*T*和重发次数*N*。任何一方在发出报文后，都会启动周期为*T*的定时器，如果在时间T内没有收发对方的报文，则重发报文并且重发计数加1。如果连续*N*次重发后对方仍无回应，则认为不可到达对方，立即撤销会话，数据传送失败。

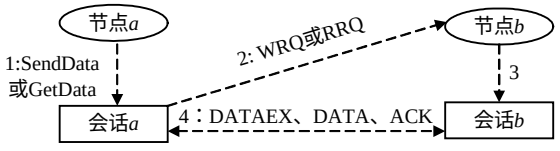

图2　会话通信过程

2.2 LKCC的组成

在客户机－服务器模型中，按照服务器处理客户请求的方式可以将服务器模型划分为：循环服务器、多路复用I/O并发服务器、动态创建进程并发服务器、固定进程数并发服务器。LKCC采用了动态线程池的方式来处理客户请求，其特点是规定线程数的上限值*M*和下限值*m*，在系统运行期间线程数在*M*与*m*之间动

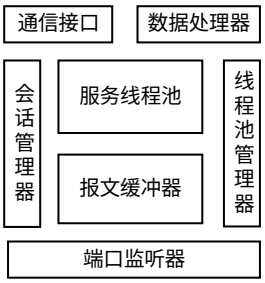

图3 LKCC结构

态变化。利用动态线程池技术，可以快速响应客户请求，能够减少频繁创建和撤销线程所消耗的系统资源和时间，也可以解决因过多的服务线程而导致的频繁上下文切换和系统资源不足的问题。图3给出了LKCC的结构，LKCC由端口监听器、报文缓冲器、线程池管理器、服务线程池、会话管理器、通信接口、数据处理器等7部分构成，如图3所示。(1) 端口监听器是一个守护线程，通过监听一个固定端口来接收RRQ、WRQ、PACKET报文，将合法报文放入报文缓冲池。并通知线程池管理器有报文到达。(2) 报文缓冲器维护一个报文缓冲池，为报文的接收提供一种缓冲机制。在服务线程都处于“忙”状态时缓存暂时不能被处理的RRQ、WRQ或PACKET报文。(3) 线程池管理器作为LKCC的核心控制部件，负责创建和撤销服务线程、控制线程池的线程数、协调服务线程的协同工作。(4) 服务线程池是所有服务线程的集合，在处理报文时线程处于“忙”状态；无报文处理时线程阻塞，处于“空闲”状态。(5) 会话管理器负责建立会话、启动会话、维护会话的状态信息、撤销会话并回收会话占有的系统资源。(6) 通信接口实现SendData、GetData、PublishData 3个通信接口，供其他内核模块调用。(7) 数据处理器为其他内核模块提供注册和注销回调函数的接口，这些函数负责处理收到的数据。

2.3 服务线程池设计

频繁创建和撤销服务线程会消耗大量系统资源，为了减少这种情况的出现，LKCC将线程池设计成为二级阻塞队列形式，如图4所示。

该线程池设计思想是：先将第1级阻塞队列用于保持系统启动时预先创建的那*m*个线程，该阻塞队列称作R_QUEUE队列，即保持(Retain Queue)队列；再将第2级阻塞队列用于延缓撤销那些在系统运行期间动态创建的线程，该阻塞队列称作E_QUEUE队列，即退出(Exit Queue)队列。当缓冲池中没有要处理的报文时，那些预先创建的线程将阻塞在R_QUEUE队列上，而那些动态创建的线程将暂时阻塞在E_QUEUE队列上，线程池管理器则会定期统计报文的处理情况以决定是否需要撤销E_QUEUE队列中的线程。如果在规定的时间内没有报文要处理，就唤醒E_QUEUE队列中的一个线程，使其终止。在经过若干个时间周期后如果仍然没有报文要处理，那么E_QUEUE队列上的线程最终会逐个被唤醒而退出运行。E_QUEUE队列中的线程在两种情况下会被唤醒：(1) 被线程池管理器唤醒去处理报文，(2) 线程池管理器为了撤销线程而唤醒它。为区分这两种情况，LKCC定义了变量need_exit，第(1)种情况need_exit被置0；第(2)种情况need_exit被置1。

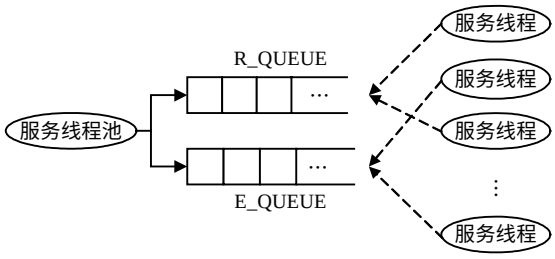

图4 服务线程池

2.4 报文处理过程

图5给出了RRQ，WRQ和PACKET报文的接收处理过程，包括接收报文、唤醒服务线程、处理报文3个步骤。(1) 接收报文。端口监听器将收到的合法的RRQ，WRQ或PACKET报文加入报文缓冲池，并通知线程池管理器有报文到达，如图5a所示。(2) 唤醒服务线程。线程管理器得到通知后唤醒服务线程去处理报文。如果要唤醒的线程是在E_QUEUE队列中，则将need_exit置0。如果两个阻塞队列中均没有空闲线程，则线程管理器需要决定是否再创建新线程来提供服务，其依据是线程数是否达到了上限值*M*，如图5b所示。(3) 处理报文。服务线程最初运行时均挂在阻塞队列上，被唤醒后从缓冲池中被取出来处理报文。如果缓冲池中已经没有要处理的报文，那么该服务线程必须决定是否退出。如果need_exit为1，且线程属于E_QUEUE队列，那么该线程终止运行，其他情况下线程仍然进入阻塞队列。在处理报文时，对于PACKET报文，可以直接从报文中提取数据来处理；对于RRQ或RRQ报文，它是客户方的请求报文，因此需要创建服务器端会话，并通过会话控制完成与客户方的数据交互。服务线程将收到的数据交给数据处理器处理后，继续查看下一个报文，如图5c所示。

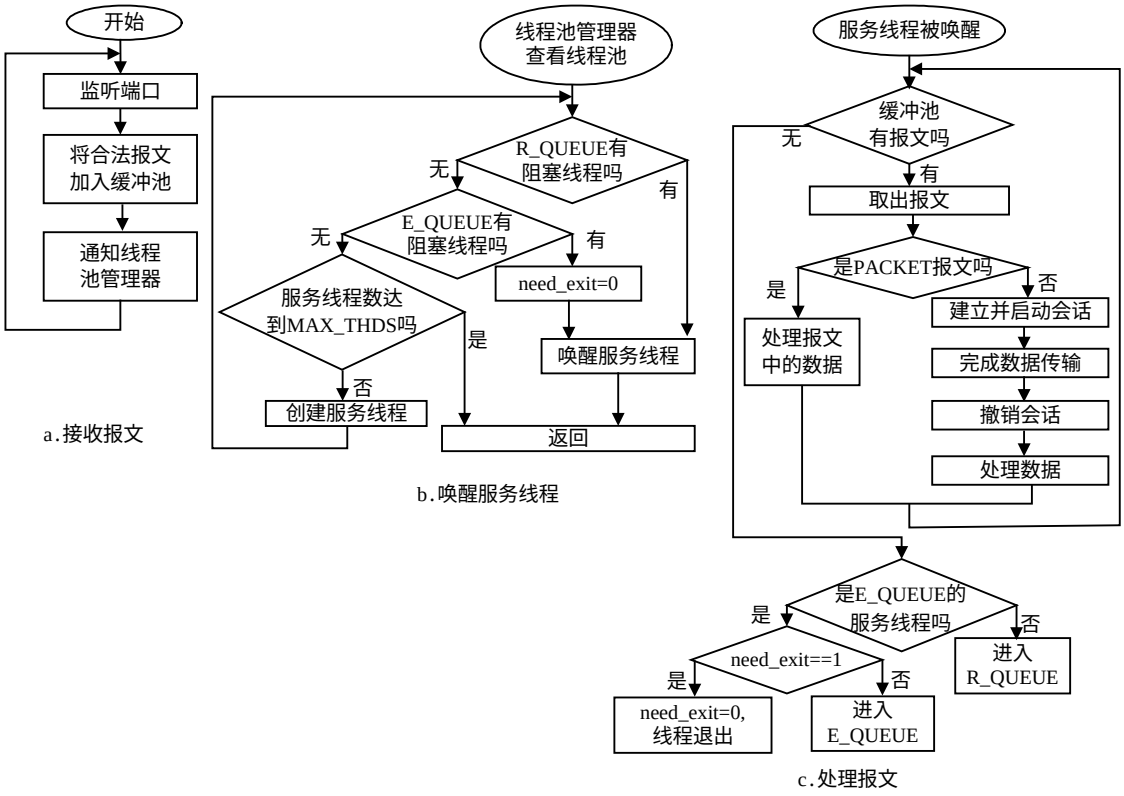

图5　处理报文过程

3　性能测试

为了测试LKCC服务器的整体性能，将LKCC服务器与另外两种功能相同但实现机制不同的动态创建线程TCP并发服务器和固定线程TCP并发服务器做比较。这3种机制中用来模拟客户请求的客户端测试程序均在应用层实现，所不同的是测试LKCC服务器的程序是通过调用LKCC提供的系统调用来向服务器发送数据；而其他两种测试程序是先进行TCP连接，然后才发送数据。LKCC服务器是一个内核模块，它向LKCC注册回调函数，LKCC收到客户数据后调用该回调函数来处理数据；其他两种服务器响应客户的TCP连接、接收和处理客户数据均在应用层实现。表1给出了3种数据量条件下服务器处理客户请求的平均处理时间。

表1　平均处理时间(单位：ms)

数据量/k	动态创建线程TCP并发服务器	固定线程TCP并发服务器	LKCC服务器
10	59	63	34
40	72	75	47
80	91	97	65

从表1可以看出，LKCC服务器整体性能比其他两种服务器要好，这主要是因为LKCC服务器减少了TCP连接过程，而且数据接收和处理均在内核下完成，从而减少了在内核态与用户态之间切换的时间开销。

4　结 束 语

本文提出了一种内核级基于会话的并发消息通信机制，该机制在分布式操作系统中得到很好的使用。

参　考　文　献

[1] Gamma E, Helm R, Johnson R. 设计模式－可复用面向对象软件的基础[M]. 李英军, 马晓星, 蔡　敏 等译. 北京: 机械工业出版社, 2001.
[2] 何炎祥, 宋文欣, 彭　锋, 等. 一个异步分布进程通信模型[J]. 计算机研究与发展, 1999, 36(2): 170-174.
[3] 江　科, 刘心松, 刘克剑. 一种应用于分布式操作系统的消息通信机制[J]. 高技术通讯, 2002增刊, 12: 339-344.
[4] 陈莉君, 冯　锐, 牛欣源, 译. 深入LINUX内核[M]. 北京: 中国电力出版社, 2001.
[5] Stevens W R. TCP/IP Illustrated(The Protocols Vol.1) [M]. 北京: 机械工业出版社, 2002.

编　辑　熊思亮