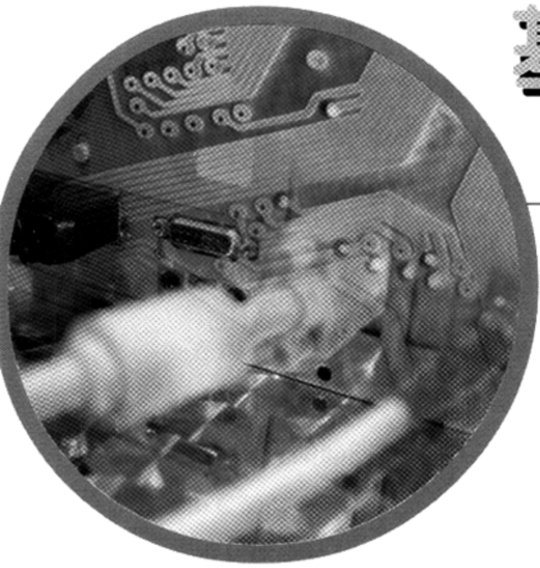

基于信号量竞争调度的多端口通信服务

王 斌 曾广平 蒋外文 (长沙中南大学信息工程学院 410083)

摘要：在代理业务平台中，需要支持多端口的tcp通信服务，并且必须把他们作为一个整体来对待。本文在预启服务进程的基础上，提出了一种基于信号量竞争调度的服务模式。

关键词：信号量 socket 端口 预启 并发服务

1 引言

代理业务平台需要提供多种不同的服务，每个服务通过不同的端口接入。所以，在其通信服务中支持多个端口是必需的，并且需要把他们作为一个整体来对待。本文论及的通信服务都是指面向连接的tcp通信服务。

多端口的通信服务和单端口的通信服务差别主要在于，从监听一个端口到监听多个端口，并把它们作为一个整体来设计。考虑的关键仍在于并发性、高效性和程序实现的复杂程度。

在对通信服务的并发性支持设计中，传统的做法是这样的：父进程等待连接请求，连接建立后，父进程生成一个子进程，然后回到等待连接请求状态。子进程(服务子进程)继承连接套接字，执行具体的通信服务，最后死亡退出。这种方式的优点是逻辑简单，程序实现容易。但是在Unix系统中，生成子进程是一个比较耗时的过程，所以如果通信压力大，这种方式是不适合的。

本文论及的方式采用了预启子进程(服务子进程)的方式：预先生成一定数目的子进程，等待被调度服务。当一个服务到达时，通过某种调度方案，某个子进程被调度，在获得连接后，执行服务。服务完毕后，又回到等待被调度服务的状态。

在这种方式下，就需要考虑一个合理的调度方案。比较常见的调度方案是通过父进程来完成的：每个子进程在父进程中有一个状态标志，表明“闲”或“忙”。父进程等待连接请求，当一个服务连接建立后，父进程扫描子进程的状态标志，找到一个“闲”的子进程，置其为“忙”，然后将连接套接字传送给这个服务子进程，自己仍回到等待连接请求的状态。被调度的服务子进程执行服务。服务完毕后，回到等待被调度状态，然后通知父进程，让父进程置自己的状态标志为“闲”。这种方案的缺点是：父进程建立连接，子进程服务，这就存在一个传递来连接套接字的问题，这在unix系统中实现比较复杂。同时，父子进程之间，有多次信息交互，关系紧密，影响系统的稳定，而且程序实现比较复杂。

本文论及的调度是通过多个子进程对一系列信号量的平等竞争完成。在这种调度设计中，父进程没有起任何作用。

2 设计方案

2.1 基础知识

(1) tcp通信服务是通过通信套接字实现的。通信套接字又称为socket。

(2) 套接字上可以绑定端口(对应于服务)。如果一个绑定了端口的套接字listen调用成功(本文称为监听套接字)，表明这个套接字(服务)可以接受连接请求。

(3) 如果一个监听套接字accept调用，表明在等待连接。如果accept成功返回，表明一个连接建立，返回新的套接字(本文称为连接套接字，唯一的指示这个连接)。如果没有服务请求，调用进程将会被阻塞(本文称为等待连接请求)。任何时候如果有服务请求到达，系统唤醒这个进程，accept将成功返回，获得连接套接字。在等待连接请求状态下，本文定义该监听套接字的状态为’满’，否则’空’。

(4) 在多进程的tcp通信服务中，怎样使真正服务进程获得连接套接字是一个最基础问题。

2.2 设计思想

(1) 父进程listen调用所有已经绑定服务端口的套接字，使其变成监听套接字。子进程(服务子进程)继承所有的监听套接字，通过动态的改变accept调用的监听套接字，就能服务任何一种服务。子进程accept调用成功，在得到连接套接字后，就可以执行具体通信服务。子进程不仅完成具体的通信服务，同时也完成accept调用，这样就避免了在进程之间传递套接字。这是能利用简单的信号量完成调度的一个关键因素。

(2) 子进程有以下三种情况：空闲，阻塞在对信号量的P操作上；阻塞在accept调用；已经从accept调用中返回，正在执行具体的通信服务。

(3) 任何时候，对任何服务监听套接字，有且只能有一个进程在这个监听套接字上等待连接请求。这是有些系统的要求。

(4) 当某个进程A从accept调用中返回，进入具体服务后，立刻能够有一个后备空闲子进程B被调度，并且B能自己确定accept调用的监听套接字，以填补进程A走后的空缺。这表明，如果一个监听套接字状态由'满'变为'空'，必须能够调度一个进程，使它在最短的时间内重新变为'满'，保证并发服务。

2.3 调度方案的实现

(1) 信号量的定义及其意义。SemDoor：监听套接字为'空'的数目，也即需要调入空闲子进程数目。因为每个监听套接字在一个时刻只能有一个子进程等待连接请求，所以初始值为所需服务端口的数目。

Sem[j](Sem信号量组)：编号j的监听套接字的状态，每个监听套接字对应一个信号量，所以这是一个信号量组。1表示'空'，需要调入子进程来使之变为'满'；0表示'满'初始值置1。

SemChoice:互斥扫描Sem信号量组的控制信号量。不保证互斥操作，可能出现不可预料的情况。

(2) 信号量原子操作

```
P(s)                    // 申请资源操作
    if(s==0) block      // 没有资源，阻塞
    s--                 // 获得资源，资源数目减1
V(s)                    // 释放资源操作
    s++                 // 释放资源，资源数目加1
```

(3) 竞争调度伪语描述

```
ChildMain()
while(1)
[   P(SemDoor)          // 是否要调入子进程
    P(SemChoice)        // 扫描Sem信号量组
    c=GetPosi()         // 得到空监听套接字
    P(Sem [c])          // 改变监听套接字状态
    V(SemChoice)        // 扫描Sem [j] 组完毕
    accept(listenfd [c] // 等待客户连接请求
    V(Sem [c]           // 改变监听套接字状态
    V(SemDoor)          // 需要调入一个子进程
    ServAction()        // 具体通信服务
]
    GetPosi()           // 返回'空'的监听套接字
[   for(j=1; j<= 服务端口数; j++)
        if(Sem [j]>0) return(j)
]
```

3 效率测试

为了比较信号量竞争调度与父进程负责调度的效率，做如下的测试。

测试用例：TCP连接，两个服务端口。Server端：预启不同数目的子进程，对于每次请求，接收20字节数据，返回4000字节的数据。Client端：对每个端口，启动五个客户进程，每个进程发起250次连接，每次连接发送20字节的数据，接收4000字节的数据。

测试环境：SCOUNIX，10M局域网，ANSIC。服务机，CPU P2MMX-200，内存64M，网卡D_Link DE 220。客户机CPU 赛扬-333，内存128M，网卡NE200。

测试结果：取server端服务完成总时间(系统时间+用户时间)，做为比较数据。见表1。

表1 测试结果(单位：秒)

预启进程数	4	8	16	24	32	48	64	96
信号量调度	19.38	19.22	20.20	19.57	20.59	22.12	23.27	25.76
父进程调度	20.55	21.31	23.72	25.92	28.27	32.11	36.29	41.58

4 结论和问题

(1) 进程之间没有通信，调度方案简单可靠，简化了程序结构，实现简单。不过调度方案用户不介入，也就

(下转第63页)

(上接第 30 页)

使它缺乏一些灵活性，在某些情况下可能不适合。

(2) 从测试数据中可以看出，与父进程负责调度比较，调度效率具有很大优势。而且当预启子进程的数目越多时，调度效率的优势越明显。

(3) 在有些 unix 系统中，支持一个监听套接字可以同时有多个进程等待连接请求。当没有服务请求时，所有进程都被阻塞。如果一个连接请求到达，系统会唤醒所有的阻塞进程，但当在其中的一个进程在获得连接后，其余的进程又会被重新阻塞。在这种系统中，调度策略可以变得非常简单，可以去掉所有的信号量操作。不过简单测试表明，在预启超过60个子进程的时候，这种方案的效率会略低与竞争信号量的调度策略。

(4) 这种设计最担心的问题是当某个子进程完成了一个P操作后，异常死亡，这将会造成整个系统的死等。不过，大多数 unix 系统提供一种功能，当一个进程退出时，如果存在以上的情况，系统会先修正上述信号量的值，使其恢复到 P 操作之前的状态。■

参考文献

1 *W.Richard Stevens.UNIX network Programming：sockets and XTI.清华大学出版社，1997。*

2 *杨继张译，UNIX 网络编程：进程间通信，清华大学出版社，1997。*

3 *曾广平，Unix 环境下 C/S 模型编程与事例，计算技术与自动化，1998.3。*

4 *曾广平、阳绿云，电信业务自动开机系统中实时文件传输和跨平台进程通信，计算机系统应用，1999.8。*