I/O就是一种错误的编程模型。

这个问题很早就为人所知，但是一直没有什么进展，直到Birrell 和Nelson在其论文（Birrell和Nelson，1984）中引进了一种完全不同的方法来解决这个问题。尽管其思想是令人吃惊的简单（曾经有人想到过），但其含义却相当精妙。在本节中，我们将讨论其概念、实现、优点以及缺点。

简言之，Birrell 和Nelson所建议的是，允许程序调用位于其他CPU中的过程。当机器1的进程调用机器2的过程时，在机器1中的调用进程被挂起，在机器2中被调用的过程执行。可以在参数中传递从调用者到被调用者的信息，并且可在过程的处理结果中返回信息。根本不存在对程序员可见的消息传递或I/O。这种技术即是所谓的远程过程调用（Remote Procedure Call，RPC），并且已经成为大量多计算机的软件的基础。习惯上，称发出调用的过程为客户机，而称被调用的过程为服务器，我们在这里也将采用这些名称。

RPC背后的思想是尽可能使远程过程调用像本地调用。在最简单的情形下，要调用一个远程过程，客户程序必须被绑定在一个称为客户端桩（client stub）的小型库过程上，它在客户机地址空间中代表服务器过程。类似地，服务器程序也绑定在一个称为服务器端桩（server stub）的过程上。这些过程隐藏了这样一个事实，即从客户机到服务器的过程调用并不是本地调用。

进行RPC的实际步骤如图8-20所示。第1步是客户机调用客户端桩。该调用是一个本地调用，其参数以通常方式压入栈内。第2步是客户端桩将有关参数打包成一条消息，并进行系统调用来发出该消息。这个将参数打包的过程称为编排（marshaling）。第3步是内核将该消息从客户机发给服务器。第4步是内核将接收进来的消息传送给服务器端桩（通常服务器端桩已经提前调用了receive）。最后，第5步是服务器端桩调用服务器过程。应答则是在相反的方向沿着同一步骤进行。

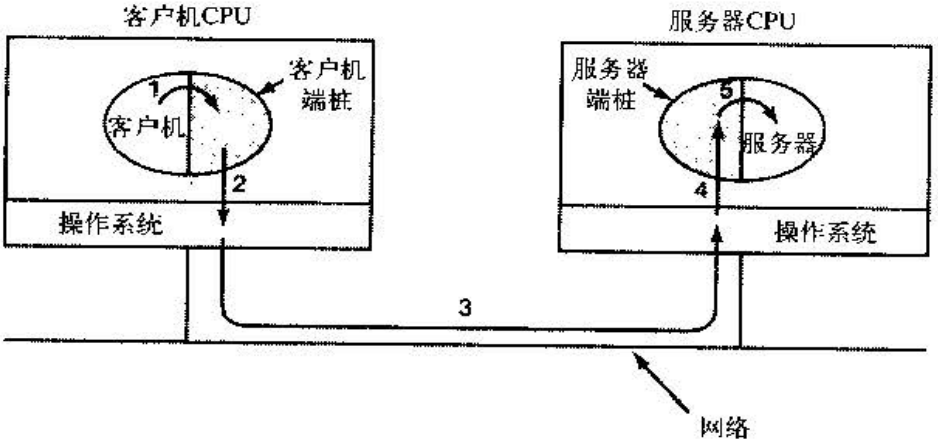

图8-20 进行远程过程调用的步骤。桩用灰色表示

这里需要说明的关键是由用户编写的客户机过程，只进行对客户端桩的正常（本地）调用，而客户端桩与服务器过程同名。由于客户机过程和客户端桩在同一个地址空间，所以有关参数以正常方式传递。类似地，服务器过程由其所在的地址空间中的一个过程用它所期望的参数进行调用。对服务器过程而言，一切都很正常。通过这种方式，不采用带有send和receive的I/O，通过伪造一个普通的过程调用而实现了远程通信。

实现相关的问题

尤论RPC的概念是如何优雅，但是“在草丛中仍然有几条蛇隐藏着”。一大条就是有关指针参数的使用。通常，给过程传递一个指针是不存在问题的。由于两个过程都在同一个虚拟地址空间中，所以被调用的过程可以使用和调用者同样的方式来运用指针。但是，由于客户机和服务器在不同的地址空间中，所以用RPC传递指针是不可能的。

在某些情形下，可以使用一些技巧使得传递指针成为可能。假设第一个参数是一个指针，它指向一个整数k。客户端桩可以编排k并把它发送给服务器。然后服务器端桩创建一个指向k的指针并把它传递给服务器过程，这正如服务器所期望的一样。当服务器过程把控制返回给服务器端桩后，后者把k送回