第35卷 第20期 计 算 机 工 程 2009年10月
Vol.35 No.20 Computer Engineering October 2009
·工程应用技术与实现· 文章编号：1000—3428(2009)20—0222—03 文献标识码：A 中图分类号：TP311

远程跨平台水下声场并行计算系统的实现

范培勤，笪良龙，谢 骏
(海军潜艇学院二系，青岛 266071)

摘 要：介绍一种跨平台远程水下声场并行计算系统。该系统支持从Windows平台到Linux平台的远程跨平台操作，为用户提供一种统一的使用模式，用户无须了解并行系统的具体结构即可使用并行系统，完成所需要的计算或仿真。应用结果表明，该系统可降低开发并行程序的复杂度，提高并行系统的使用效率，实现服务端与客户端的实时数据交换，具有很好的实用价值。
关键词：并行计算；PC集群；跨平台；远程；水下声场传播模型

Realization of Remote Underwater Acoustics Parallel Computing System on Cross Platform

FAN Pei-qin, DA Liang-long, XIE Jun
(The Second Department, Navy Submarine University, Qingdao 266071)

【Abstract】This paper introduces a remote underwater acoustics parallel computing system. On cross platform, this system supports remote operating and running Linux system parallel program on Windows system and provids a kind of uniform mode for the applied customer to operation. Customer does not need to know the concrete structure of the parallel computing system, and can operate the parallel system and complete the computation or simulation what they need. Application results show that this system can reduce the complication on developing parallel programs, improve the efficiency on using parallel computing system, and solve the problem of real-time data transmission between server and client machine.
【Key words】parallel computing; PC cluster; cross platform; remote; underwater acoustics transmission model

1 概述

目前并行机的软件开发环境还未成熟，限制了并行计算的广泛应用，并且在一定程度上阻碍了并行计算从高性能到高效能的转化。如何在现有的高性能硬件平台和软件支撑环境上，结合各个应用部门的具体应用和数值方法的特点，通过移植、改进和设计高效的并行应用软件，并开发相应的辅助软件工具，以简化并行应用程序的开发，缩短软件的开发周期，提高软件的计算效率，是从事并行科学计算与工程计算研究的人员必须面对的一个关键问题。

本文在上述背景下设计开发一个客户/服务模式的水下声场并行计算系统，通过该用户既可以在Windows平台远程调用Linux平台上的并行应用程序，也可以上传自己的计算程序来完成计算。这样可以使Windows程序开发人员能方便地利用并行系统解决水下声场并行计算的问题，减轻程序开发人员的工作量，提高程序、集群的利用率，降低并行系统操作的复杂性[1]。

2 系统结构

系统基于客户端/服务端结构，由服务器端、客户端和Socket[2]通信组成，客户端在Window、平台，服务器端在Linux平台[3]。客户端主要由图形用户界面、远程服务器等模块组成，主要工作是实现图形界面，支持用户完成对被调试程序的控制，发送相应命令给服务器端，接收从服务器端返回的计算结果，并以图形化方式展现给用户；服务器端主要由远程服务器、作业调度器、节点执行器、并行程序调试器组成，其中，服务器端的远程服务器、作业调度器、节点执行器为守护进程，它们之间相互配合，共同实现作业管理、作业调度等功能。服务器端的主要工作是实现对多用户的支持，根据客户端发来的请求命令，完成相应的并行计算任务和并行程序的调试、入库工作，并返回运行结果。客户端、服务器端的功能如下：

(1)客户端图形用户界面，它接收用户指定的操作，并通过远程服务器传给服务端。同时通过远程服务器接收服务器端返回的信息。客户端的远程服务器用于传输用户指令、参数及接收和显示计算结果。

(2)服务端远程服务器负责程序的全局控制。它根据接收的客户端传来的指令，向对应的模块发出命令，控制并行程序的运行或调试。如果有结果返回，它负责通过服务器的远程服务器传给客户端。

使用层次化的设计思想实现该系统，将系统分为3层[4]：第1层是图形用户界面层，提供友好的图形界面，接收用户指令并返回结果，并提供与远程服务器的对接接口，实现远程连接；第2层是通信层，主要由高速交换机及其他网络设备构成，负责网络连接的硬件实现；第3层是服务器端，负责控制并行系统的运行和通信，分别由并行编译模块、计算

基金项目：国家部委基金资助项目；教育部新世纪优秀人才支持计划基金资助项目(ZL200510000885.6)
作者简介：范培勤(1981－)，男，博士研究生，主研方向：水下声场并行算法及应用；笪良龙，教授、博士生导师；谢 骏，讲师、博士研究生
收稿日期：2009-02-10 **E-mail：**similaroly05@163.com

模块和声场计算并行程序库和远程通信接口组成。系统结构如图 1 所示。

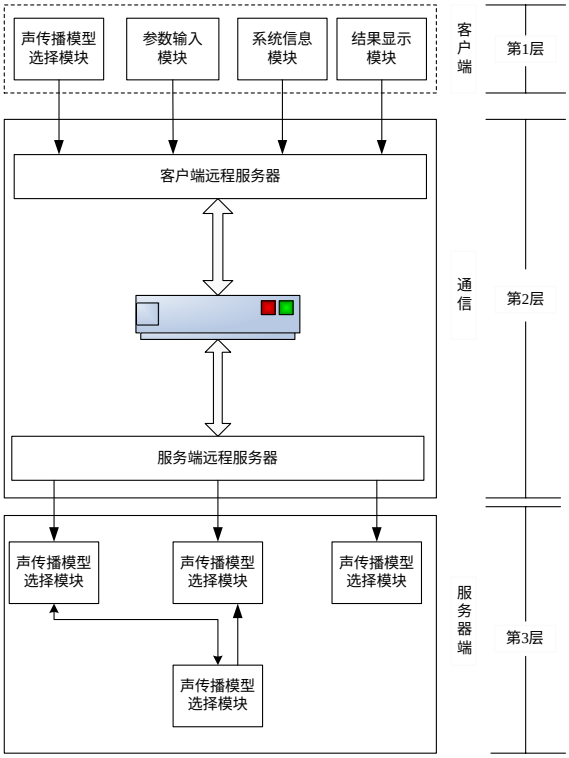

图 1　水下声场并行计算系统结构

这样的层次化设计使水下声场并行计算系统与并行系统的具体结构分割开来，从而不用考虑命令的具体执行，有利于系统整体性能的提高。

3　系统实现

3.1　客户端具体实现

客户端实现了一个友好的图形用户界面，主要由水下声场计算模型选择、计算参数输入、并行程序调试、计算结果显示和作业管理等模块组成(见图 2~图 5)。

图 2　客户端系统信息模块

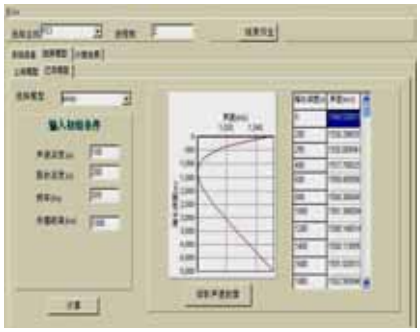

图 3　客户端模型选择模块

图 4　客户端上传模型模块

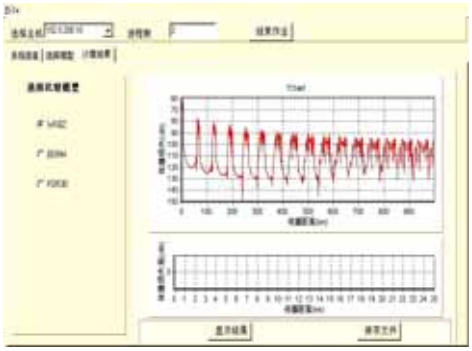

图 5　客户端结果显示模块

在系统中将所有的功能封装在模块中，简化了系统的开发过程，提高了系统的开发效率。为让用户及时了解程序当前程序的运行情况，本文在服务器端实时地将并行程序的执行情况发送到客户端远程服务器上，客户端根据这些结果将并行程序的实际运行情况显示给用户，例如，显示程序名、源程序、进程所在的节点和当前执行等，帮助用户进行下一步操作。作业管理模块主要完成作业的提交、查看、修改、删除、挂起/恢复、移动和传送等功能。客户端的实现向用户屏蔽了服务器端底层的所有实现细节，其内部通信对用户完全透明。

客户端的实现比较简单，主要技术难点在于如何与服务器端进行通信，以完成参数、指令的传递。由于在于服务器端进行通信时需要传递大量的指令、参数及计算结果，所需的通信量很大，因此本文选用效率较高的 Socket 完成通信。

3.2　服务器端具体实现

服务器端主要有服务器端远程服务器、水下声场并行计算模型库、作业调度模块组成。

3.2.1　服务器端远程服务器的实现

为了能及时地支持客户端的临界，将服务器端的远程服务器做成一个守护进程，守护进程将在服务器中不间断的运行，时时监测来自客户端连接请求。为了简化服务器端的远程服务器，将所有与计算有关的程序都剥离出来，服务器端的远程服务器只负责接收来自客户端的请求和参数，然后根据接收到的指令激发应用进程来完成水下声场的并行计算。守护进程具体实现如下：

(1)调用 *fork*()，创建一个子进程。

(2)调用 *setsid* 使子进程成为新会话的领头进程，从而与原有会话进程组脱离。为避免作为新会话领头进程的子进程在打开一个终端设备时拥有一个控制终端再次派生，并结束父进程，让子进程成为非会话领头进程。

(3)忽略信号 *S1GHU*1，第 2 次调用 *fork*()。第 2 次调用 *fork*()后，进程已经脱离了控制终端，但由于它与退出的父进

程属于同一个进程组，因此发给原来进程组的信号仍然可能发给这个进程，调用 *stpgrg*()，使自身脱离原来的进程组。

(4)关闭不再需要的文件描述符，并为标准输入、标准输出和标准错误输出打开新的文件描述符。

(5)调用函数 *chdir*("/")将当前工作目录更改为根目录。这是为了保证的进程不使用任何目录。否则守护进程将一直占用某个目录，这可能会造成超级用户不能卸载一个文件系统。

(6)完成以上的准备工作后，守护进程启动，开始循环等待用户的连接。

当有用户连接后，再调用 *fork*()，子进程完成与此用户的所有操作，父进程继续等待其他用户的连接，这样服务器端便可支持多用户连接。建立连接后的服务器端，将阻塞在用于远程同步传输的 Socket 上，当有客户端发来调试命令和参数后，执行此命令，并将经同步传输返回的结果由用于远程同步传输的 Socket 返回给客户端，将经异步传输返回的结果由用于远程异步传输的 Socket 返回给客户端。

3.2.2　水下声场并行计算模型库实现

为减少系统对内存的占用并同时为了简化系统的开发，将水下声场计算模型的并行程序做成一个动态链接库[5]，以供执行程序调用。Linux 与 Windows 的动态连接库概念相比，它引入了 GOT 表和 PLT 表的概念，综合使用了多种重定位项，实现了“浮动代码”，比 Windows 环境下的动态链接库有更好的共享性能。这样用户可以选择使用动态链接库中已有的水下声场计算的并行程序，也可以提交自己的并行程序，这时只需将并行程序载入到动态链接库就可以方便的调用。这样给用户提供了一个统一的编程模式，便于系统的拓展和升级。随着模型的增加，系统可以处理的问题的范围和能力将逐渐增大。

为了让执行程序顺利找到动态链接库，有 3 种方法：

(1)将动态链接库拷贝到/usr/lib 和/lib 目录下。

(2)在 LD_LIBRARY_PATH 环境变量中加上动态链接库所在路径。例如库 libhello.so 在/home/ting/lib 目录下，以 bash 为例，使用命令：

$export LD_LIBRARY_PATH=$LD_LIBRARY_PATH：/home/ting/lib

(3)修改/etc/ld.so.conf 文件，将动态链接库所在的路径加到文件末尾，并执行 ldconfig 刷新。这样，加入的目录下的所有库文件都可见。

服务器端的其他功能主要通过对 Socket 通信内容的处理来完成，客户端和服务器端都采用 Socket 通信。

3.3　客户端与服务器端 Socket 通信具体实现

Socket 是一进程与另一进程进行双向通讯的最终的套接点，Socket 是伯克利在 Unix 系统中推出的计算机系统的 IPC 和异种机网络互联的基本机制。网络 Socket 数据传输是一种特殊的 I/O，Socket 也具有一个类似于打开文件的函数调用 *Socket*()，该函数返回一个整型的 Socket 描述符，随后的连接建立、数据传输等操作都是通过该 Socket 实现的。常用的 Socket 类型有 2 种：流式 Socket(SOCK_STREAM)和数据报式 Socket(SOCK_DGRAM)。流式是一种面向连接的 Socket，针对于面向连接的 TCP 服务应用；数据报式 Socket 是一种无连接的 Socket，对应于无连接的 UDP 服务应用。在本系统中本文主要使用流式 Socket。

Socket 在进行数据传输通信时，主要有 2 种传输方式：远程同步传输和远程异步传输。远程同步传输的 Socket 通道主要传输调试命令和调试附带参数，以及服务器端经同步传输返回的结果。

在同步命令模式下，命令的接收处于待命状态，随时准备接收命令，当被控进程处于停止状态时，调试命令按同步命令模式发出，例如，设置断点，主控制器直接发送命令给从控制器，并等待经执行后通过从控制器返回的结果。

在异步响应模式下，命令的接收对象处于工作状态，不能接收命令，此时命令采用异步中断的方式发出，例如，用户产生一个新进程，这时要通知主控制器创建一个新的从控制器去控制它，这种通知就是异步响应。远程异步传输不同于本地异步传输，不能使用 signal。由于用于远程异步传输的 Socket 通道要传输服务器端异步返回的结果，因此使用 *select*()实现此 Socket 非阻塞。在客户端每隔固定时间扫描一次端口，如有服务器端的结果返回则响应并接收，如没有也不在此阻塞。在客户端，数据传输前，在每一个数据的最后标上一个结束符，然后将数据一起传送，当服务器端接收到一组数据后，可以根据结束符将数据分离开。这样做降低了系统的通信的次数，减少了由通信带来的延迟。

4　结束语

本文介绍一种跨平台远程水下声场并行计算系统，远程跨平台技术降低了用户操作并行系统的难度，提高了并行系统的使用效率。模块化和分层设计的思想便于系统的维护和拓展。为将并行计算系统应用于水下声场及其相关领域的研究提供了一种思路。该系统的进一步优化和服务器端的并行程序动态链接库的扩展是下一步工作的重点。

参考文献

[1] 李蔚泽. Linux 与 Windows 整合——跨平台操作、资源共享、数据转移[M]. 北京: 机械工业出版社, 2007.

[2] 李　洋, 王虎松. Red Hat Linux 9 系统与网络管理教程[M]. 北京: 电子工业出版社, 2006.

[3] 刘　建, 徐宏亮, 沈美明, 等. Linux 机群系统并行程序调试器的设计与实现[J]. 计算机工程, 2002, 28(4): 7-9.

[4] 黄　萍, 藩荫荣, 胡幼华. 基于 Java 和 XML 跨平台数据迁移的设计与实现[J]. 计算机工程, 2005, 31(17): 74-75.

[5] 杨树青, 王　欢. Linux 环境下 C 编程指南[M]. 北京: 清华大学出版社, 2007.

编辑　金胡考