基于 Linux 集群的分布式进程通信系统的实现

肖 鹏[1] 李媛媛[2] 王云鹏[3]
[1](大连轻工业学院信息工程学院 大连 116022)
[2](大连交通大学软件学院 大连 116028)
[3](华为通讯公司 深圳 518129)

摘 要 本文提出了一种分布式进程通信系统的体系结构模型,并针对 Linux 集群环境,以 Linux 通用操作系统为基础实现了此分布式进程通信系统。

关键词 分布式系统 Linux 集群 进程间通信

Implementation of distributed processes communication system based on Linux

Xiao Peng[1] Li Yuanyuan[2] Wang Yunpeng[3]
[1](*Dailian Institute of Light Industry, Dalian 116022, China*)
[2](*College of Software, Dailian Jiaotong University, Dalian 116028, China*)
[3](*HuaWei Communication Company, Shenzhen 518129, China*)

Abstract The paper proposes a detailed model of the distributed processes communication system. According to Linux Cluster, we design and implement this model.

Key words distributed system Linux cluster processes communication

1 引 言

随着网络应用的日益广泛,基于 Linux 集群的分布式操作系统的研究得到了迅速发展。进程通信是分布式操作系统的一项重要功能,也是分布式计算环境中不可缺少的主要内容。对于在分布式环境下开发软件的程序员来说,一个实用的分布式进程通信系统应具有以下特征:①透明性[1,2],用户通过调用分布式操作系统提供的接口编写应用程序,而无须考虑任何网络细节;②支持分布式环境下的进程通信;③提供类似于单机下进程通信的C 语言调用接口。针对上述 3 种需求,本文提出一种实用的分布式进程通信系统模型,并基于 Linux 集群环境实现了此模型。

2 结构设计

2.1 软件体系结构

图 1 描绘了分布式进程通信系统的软件体系结构模型:

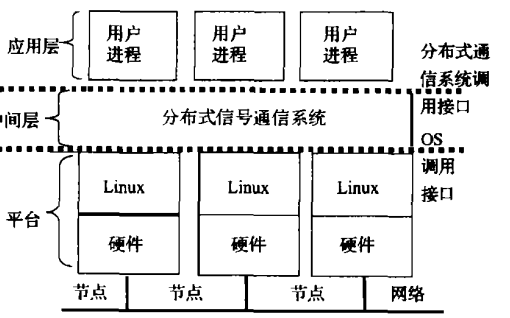

图 1 分布式进程通信系统的软件体系结构模型

(1)最底层的硬件和 Linux 操作系统共同构成了进程通信系统的工作平台。分布式进程通信系统通过在操作系统和应用层之间加入中间层实现。它由一组分布在不同节点上的进程利用操作系统提供的服务,相互协作,共同完成信号发送工作。

(2)从用户程序的角度看,这个结构屏蔽了分布式进程通信系统的分布特性,只需要象使用单机信号发送调用接口那样就能完成分布式环境下的信号发送,

保证了分布式系统的透明性。

2.2　系统模型

客户/服务器模型是分布式系统最常用的体系结构之一。分布式进程通信系统也基于这种结构为用户进程提供服务，如图 2 所示。

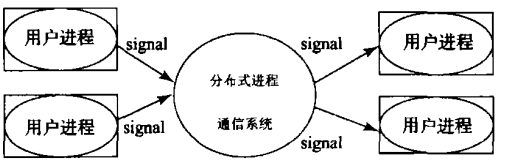

图 2　分布式进程通信系统的工作模型

实现分布式系统透明性的关键部分是命名[3]。为了达到名字透明，每个资源应该使用唯一的标识符。在单机系统中进程号(pid)唯一地标识进程。在网络通信中，可以用 IP 地址和端口号来定位该进程。然而，如果在分布式环境下使用{IP 地址，端口号}地址结构作为名字就会破坏分布式系统的透明性。我们可以模拟单机系统中的进程号，在分布式进程通信系统中建立分布式环境下的进程号唯一标识，简称为分布式进程号(表示为 pid_d)。对于系统来说，要对用户进程进行操作或访问，还需要将进程的分布式进程号(pid_d)转换为它的地址{IP，pid}，转换的过程需要由一个服务程序来完成，称为名字服务器(Name Server)，简称 NS。

由以上讨论可以确定如图 3 的体系结构模型。图中的虚线框就表示分布式进程通信系统。系统由两部分组成：

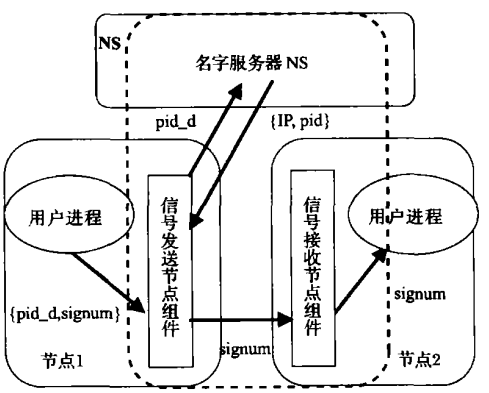

图 3　分布式进程通信系统体系结构模型

(1)名字服务器：维护进程名表，为整个系统提供名字服务，包括：注册、注销、名字查询。名为 NS 的守护进程监听来自用户节点组件的请求，并对进程名表作出相应操作。

(2)用户节点组件：负责接收用户的请求，它由守护进程 Daemon 和一组服务子进程(包括注册、注销、信号发送等)构成。Daemon 负责接收用户请求如注册、注销、信号发送请求，以及来自远端节点组件的请求等。然后根据请求类型利用 fork()调用创建相应子进程为用户服务。

2.3　系统的工作流程

首先分析分布式进程通信系统中进程的注册和注销，然后描述一个典型的通信过程。

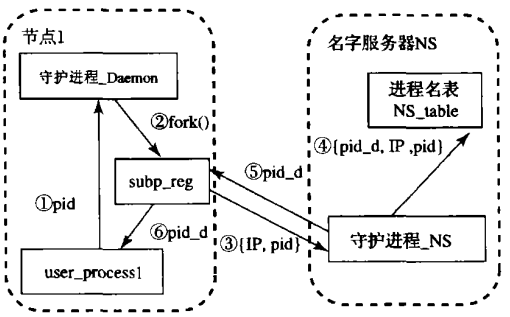

图 4　系统的工作流程－注册

(1)注册过程如图 4 所示：

①节点 1 上的用户进程 1 向本地用户节点组件的守护进程 Daemon 提出成为系统用户的申请，并提供自身的 pid；

②Daemon 创建一个名为 subp_reg 的服务子进程；

③subp_reg 服务子进程将本节点的 IP 地址和用户进程 1 的 pid 发送给名字服务器NS 上的守护进程_NS；

④守护进程_NS 进程分配一个全局唯一的 pid_d，并把三元组{pid_d，IP，pid}写入进程名表，用于将来 pid_d 与进程地址的相互转换；

⑤然后守护进程_NS 将 pid_d 返回给节点 1 上的 subp_reg 服务子进程；

⑥subp_reg 服务子进程将 pid_d 传递给用户进程 1。

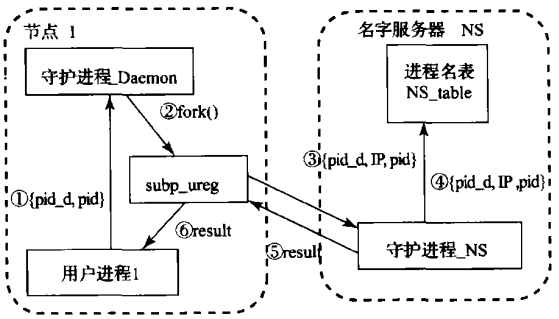

图 5　系统的工作流程－注销

(2)注销过程如图 5 所示：

①节点 1 上的用户进程 1 向 Daemon 提供{pid_d，

pid}, 申请注销;

②Daemon 创建一个服务子进程 subp_ureg;

③subp_ureg 将三元组{ pid_d, IP, pid} 发送给名字服务器 NS 上的守护进程_NS;

④守护进程_NS 将根据 pid_d 找到进程名表的相关记录, 并比较两个三元组是否相同。如果相同, 则将记录的三元组删除; 如果不同, 不做任何操作;

⑤⑥依次返回注销结果。用户进程只有完成注销, 才解除了与分布式进程通信系统的关系。

(3)图 6 描述了一个典型的通信过程, 两个在本系统注册过的用户进程 1 给另一节点上的用户进程 2 发 kill 信号中止其运行。

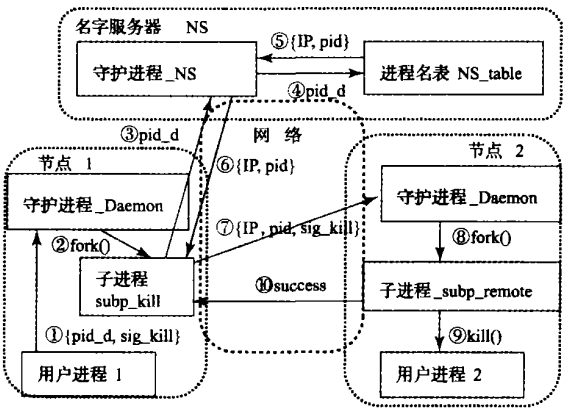

图 6 系统的工作流程 – 不同节点的进程通信

2.4 用户调用接口

针对以上通信过程, 定义出几个用户调用接口:

(1)注册接口: pid_d regist( )

用户进程在使用本系统接收信号前, 首先要进行注册, 系统会将用户的注册信息保存在进程名表中。具体流程如图 4 所示。

(2)注销接口: int uregist(pid_d)

当用户进程不再通过本系统接收信号时, 要通过注销操作将自己的 pid_d 释放给系统。具体流程如图 5 所示。

(3)信号发送接口: intkill_d(pid_d, sign)

用于分布式系统中一个进程通过信号发送接口向目标进程发送各种信号量进行通信。

3 系统的实现与测试

最后我们在安装有 Redhat Linux 9.2 的 Linux 集群中用 GCC 编程实现了本系统。并设计了一个用户程序和一个目标程序, 让用户程序连续向目标程序发送 100 个信号, 计算发送 100 个信号的时间。这里共做 10 次测试, 前 5 次测试结果见表 1。测试采用<sys/time.h> 中的 gettimeofday( ), 得到微秒级的当前时间(与 1900 01 01 00:00:00 的时间差)。经过计算, 每发送 100 个信号, 平均花费 3s, 发送一个信号约 0.03s。本系统运行平稳, 保证了分布式系统的透明性, 并且简单的用户调用接口使用户的编码数量显著减少。

表 1 分布式进程通信系统通信时间测试结果

开始时间		结束时间		通信时间	
秒	微秒	秒	微秒	秒	微秒
1113461689	380522	1113461692	522712	3	142190
1113461821	423068	1113461824	90152	2	667084
1113461869	871327	1113461873	196536	3	325209
1113461933	312543	1113461936	536394	3	223851
1113462047	3347	1113462050	368161	3	364814

4 结束语

与众多分布式操作系统实现方案相比, 该系统具有以下特点:

(1)很好地解决了分布式进程通信问题;

(2)基于现有的 Linux 集群网络实现分布式处理, 造价成本低, 易推广;

(3)提供了 C 调用接口, 实用性强且易移植;

(4)具有可实现性, 为分布式操作系统前瞻性研究提供了实验环境。

该系统可广泛应用于分布式进程处理环境和分布式操作系统研究中。

参考文献

[1] 胡贯荣. 分布式 IPC 实时性能的研究[J]. 华中理工大学学报, 1999, 27(4): 1-3.

[2] 郭锐锋. 分布式实时操作系统中实时通信的研究[J]. 小型微型计算机系统, 1995, 16(8): 1-3.

[3] William Stallings. 操作系统 内核与设计原理[M]. 北京: 电子工业出版社, 2001.

[4] George Coulouris. 分布式系统概念与设计[M]. 北京: 机械工业出版社, 2004.

[5] 毛德操. Linux 内核源代码情景分析(上、下卷)[M]. 杭州: 浙江大学出版社, 2001.

[6] 郑彦兴. 深刻理解 Linux 进程间通信(IPC)[M]. 长沙: 国防科技大学出版社, 2002.