NFS 在分布式数控系统中的应用与改进①

杨 阳[1,2], 王 品[2,3], 杜少华[2,3]

[1](中国科学院大学, 北京 100049)
[2](中国科学院沈阳计算技术研究所, 沈阳 110168)
[3](沈阳高精数控技术有限公司, 沈阳 110168)

摘 要: 随着数控技术的进步, 数控系统由单机系统向分布式系统发展, 机床存储介质的容量不能满足复杂数控程序存储的需求. 在分布式数控系统中应用 NFS 可以解决存储空间限制的问题, 而且便于用户管理共享文件. 介绍了 NFS 技术, 并结合实际需求将 NFS 服务由传统的使用 IP 分配资源改进为依据用户信息分配资源.

关键词: 网络文件系统; NFS; 分布式数控系统; 资源分配

Application and Improvement of NFS in Distributed CNC System

YANG Yang[1,2], WANG Pin[2,3], DU Shao-Hua[2,3]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)
[3](Shenyang Institute of Computing Technology Co.Ltd., Chinese Academy of Science, Shenyang 110168, China)

Abstract: With the progress of CNC(Computer Numerical Control) technology, CNC system has developed from a stand-alone system to a distributed system. The capacity of the machine storage can't meet the needs of complex numerical control program. Applying NFS(Network File System) service to the distributed CNC system can solve the problem of limited storage space, and is convenient for users to manage the shared files. This paper introduces the key technology of NFS and changes the resource allocation strategy from using IP address to using user information.

Key words: network file system; NFS; distributed CNC system; resource allocation

数控技术的发展导致数控程序日益复杂, 程序体积迅速膨胀, 机床的存储空间往往不能满足存储需求. 为了解决机床存储容量的限制并方便集中对程序进行管理, 有必要在分布式数控系统中部署 NFS 服务器提供文件服务[1]. 本文的研究重点就是 NFS 在分布式数控系统中的应用与改进.

使用 NFS 时, 需读取 exports 文件配置访问权限, 在 exports 的配置中会限定访问者的 IP, 只有 IP 地址符合的客户端才可以挂载 NFS[2]. 一个常见的 exports 文件配置如下: /home/brian 192.168.1.111(rw), 意味着 /home/brian 文件夹下的资源只能被 IP 为 192.168.1.111 的客户机挂载. 这种使用 IP 进行资源分配的机制导致不同用户使用相同 IP 地址挂载 NFS 时, 得到的是相同的路径. 这种策略导致在实际应用中, 服务器无法根据用户的信息定制不同的挂载路径.

本文针对上述问题, 对已有的 NFS 系统进行改进, 使其可以根据用户信息进行资源分配. 具体思路是, 当客户机进行挂载时, 服务器根据传递的用户名为客户机生成私有的文件结构并返回文件句柄.

1 NFS基本原理与关键技术

NFS 不提供数据传输的协议, 却能通过网络进行数据的共享, 这是因为 NFS 本身就是一个 RPC 应用. 只有先启动了 RPC 服务, 才可以启动 NFS, 而且客户端与服务器端都必须启动 RPC 服务, 这样才能使得双方通过 RPC 实现通信. 所以 NFS 实现的基础是 RPC,

① 基金项目: "高档数控机床与基础制造装备"国家科技重大专项(2013ZX04007-011)
收稿时间:2014-10-21;收到修改稿时间:2014-12-22

首先介绍一下什么是 RPC.

1.1 RPC 介绍

远程过程调用(RPC)是一个协议, 程序可以通过这个协议请求网络上另一台主机上的某个程序而不需要考虑网络的细节. RPC 是一种服务器/客户端模型, 请求程序是一个客户端, 而服务提供程序是一个服务器. 在服务器运行守护进程等待客户端发送的请求, 客户端的请求会包含请求的参数, 服务器在接收到请求时, 将请求分配给指定的线程进行处理, 并向客户端返回计算结果, 继续等待下一个请求[3]. 客户端在接收到远程过程调用的执行结果后继续执行.

NFS 功能复杂, 包含多个 RPC 应用, 常用的有 rpc.nfsd, rpc.mountd, portmap 等, 这些应用与客户端进行通信时的端口都是不确定的. RPC 随机为这些应用分配端口, 这样就导致在传输前客户端不知道这些 RPC 应用的通信端口, 从而导致无法进行通信. RPC 的解决方法是, 任何一种 RPC 应用启动时需要向 RPC 注册, RPC 为该应用分配一个随机端口, 将该应用和对应的端口保存在 PORTMAP 中. 在通信时, 客户端只需知道 RPC 协议的固定端口 111, 在请求具体的 RPC 应用时, 服务器根据 PORTMAP 获取为该应用随机分配的端口号并返回给客户端, 客户端得到端口号之后就可以与 RPC 应用进行通信. 所以, RPC 一定要在 NFS 之前启动. 图 1 是 NFS 请求的处理过程:

(1)启动 nfsd,moutd 等 NFS 相关的 RPC 应用, 为这些应用分配随机端口, 并在 RPC 中注册应用, 将应用与端口的信息保存在 PORTMAP 映射表中.

(2)客户端的请求通过 RPC 客户端提交给 RPC 服务器.

(3)服务器查询 PORTMAP 找到 RPC 应用对应端口并返回给客户端.

(4)客户端得到 RPC 应用端口后, 与 RPC 应用通信, 完成后续操作.

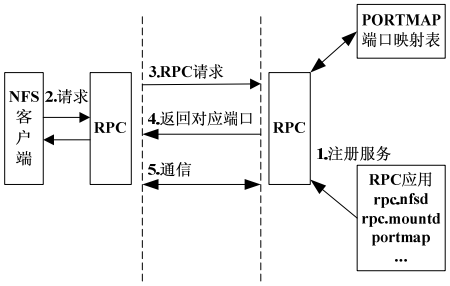

图 1 NFS 工作流

1.2 NFS 工作原理

NFS 基于 RPC 进行通信, RPC 通信时使用 XDR 格式表示数据, 在网络层则使用 TCP 或者 UDP 进行传输[4]. NFS 的层次结构如图 2 所示.

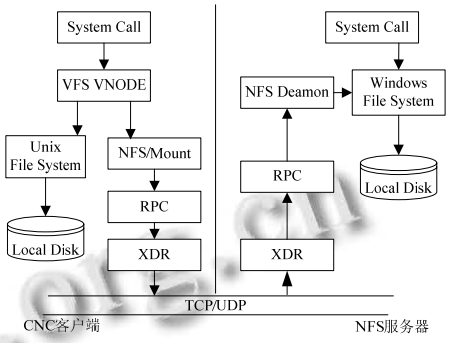

图 2 经典的 NFS 模型

NFS 提供透明访问文件的功能. 所谓的透明, 就是操作 NFS 中的文件与操作本地磁盘的文件对使用者而言是无差别的. FTP 也是广泛使用的一种文件传输协议, NFS 与 FTP 最大的区别在于, NFS 返回的是一个文件的句柄, 对文件的操作都是通过这个句柄完成的, 而 FTP 则返回了服务器文件完整的副本. 使用句柄操作文件, 使得任何一个可以操作本地句柄的文件系统都可以方便的操作从 NFS 服务器获取的句柄.

1.3 文件句柄

NFS 中的文件操作都是通过文件句柄这种数据结构实现的, 文件句柄包含了唯一标识文件或目录所需要的信息. 客户端对服务器发送 mount 请求时, 服务器会返回请求路径根节点的文件句柄. 对文件句柄的操作是: 解析句柄并在需要时通过这个句柄与服务器进行交互并获取新的句柄, 直到操作结束, 释放句柄. 图 3 描述了当客户端通过 NFS 获取路径为 /brian/home/test 的目录时, 客户端与服务器通过句柄的交互过程.

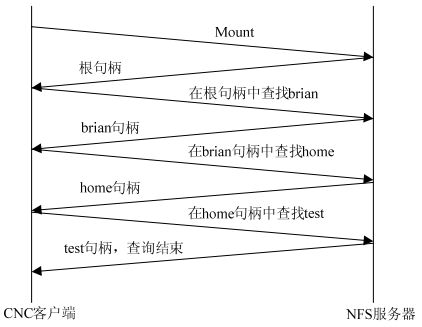

图 3 通过文件句柄获取路径/brian/home/test 的过程

2　改进资源分配策略

NFS 应用在数控系统中后, 不同级别的用户可以访问的文件路径有所不同, 不同数控机床由于功能不同需要挂载的程序路径也可能不同, 因此 NFS 使用 IP 管理文件系统的访问权限不能满足实际需求. 本文主要改进内容是将 NFS 服务由传统的使用 IP 分配资源改进为依据用户信息分配资源.

当用户在 Linux 客户端登录后, 如果客户端的 IP 地址在 exports 文件中, 那么该用户就可以对 NFS 服务器上的文件进行挂载操作. 使用 mount 命令进行挂载时, 客户端会将当前用户的用户名隐式传递给服务器.

既然用户名可以隐式传递给服务器, 在原有 NFS 结构中增加一层负责读取用户信息并进行差异化的资源分配就可以实现按用户分配资源. 也就是说, 需要在服务器 NFS 层上增加 UserManager 层来控制用户的访问行为. 图 4 是修改之后系统的整体结构:

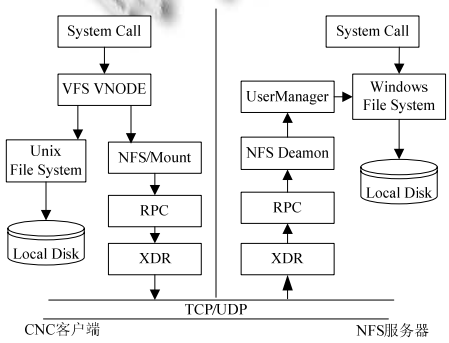

图 4　增加 UserManager 层之后的 NFS 模型

改进后的 NFS 服务器处理请求的过程如下:

(1) 客户端 Mount 请求, 传递给服务器的数据中包含了当前用户需要挂载的目录、用户信息等.

(2) 客户端传来的数据经过 UserManager 层, 经过处理后生成的文件句柄将通过 RPC, XDR, TCP/UDP 传递给客户端. NFS 的文件操作也是如此, 所有的信息都要通过 UserManager 层.

具体来说是通过读取 manage 文件中该用户的私有目录构造用户私有的 Inodetree, 通过读取公共目录构造所有可访问路径的 Inodetree, 最后合成该用户的 Inodetable 数据结构, 同时把结构为 Inodetable 的文件句柄传给客户端, 这样客户端就可以根据该句柄对服务器上的文件系统进行操作.

Inodetable 的结构设计为:

```
typedef struct DIRNODE{
    char name[MAXFILENAMELEN];
    u_char fsid;
    u_short inode;
    struct DIRNODE *next;
    struct DIRNODE *subdir;
    struct DIRNODE *parent;
    Acache_t *attr;
}Dir_t;
typedef Dir_t *Inode_t;
Inode_t *InodeTable;
```

文件句柄的结构设计为:

```
typedef struct{
    struct fhs f;
    struct fhs p;
    char fh_pn[FH_PN];
}fhandle_t;
```

其中 struct fhs 表示句柄的状态. 结构如下:

```
struct fhs{
int fh_fsid;
u_long fh_fno;
};
```

在改进的设计中, 当用户登录服务器时, UserManager 获得用户名, 读取 exports 获得共享文件的路径, 随后通过读取 manage 中该用户的访问路径并在共享文件路径中进行标示, 最后将标示的路径整理成虚拟的树形结构并返回根句柄. 例如, exports 的配置如下:

```
c:\doc ro
c:\test rw
```

manage 文件的定义如下:

```
Brian:c:\test\p1\src,c:\doc\java
Lucy:c:\test\p1\bin,c:\test\p1\src
```

mount 命令为:

```
mount -t nfs 192.168.1.131(服务器 IP):\c:\test \home\tmp
```

NFS 目录的树形结构如图 5 所示.

假设当前用户为 Brian, UserManager 根据 exports 和 manage 的标示出 Brian 可以访问的路径, 如图 6 所示, 图中着色部分就是用户 Brian 私有的访问路径.

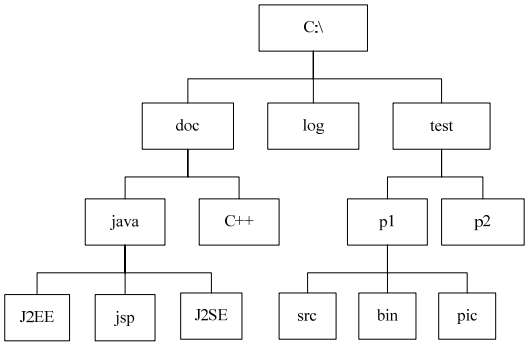

图 5 NFS 目录的树形结构

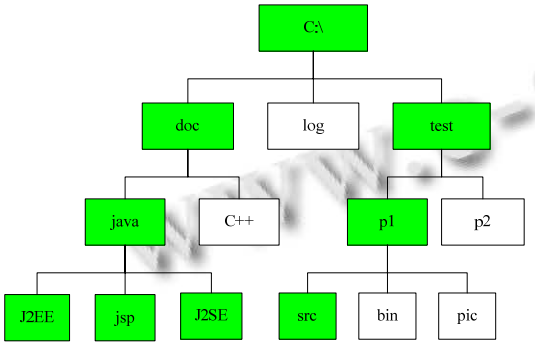

图 6 Brian 可以访问的路径

得到访问路径后, UserManager 对路径整理生成虚拟的树形结构, 新的树形结构去掉了图 6 中的非着色部分,Root 指针指向了树形结构的根节点. 如图 7 所示.

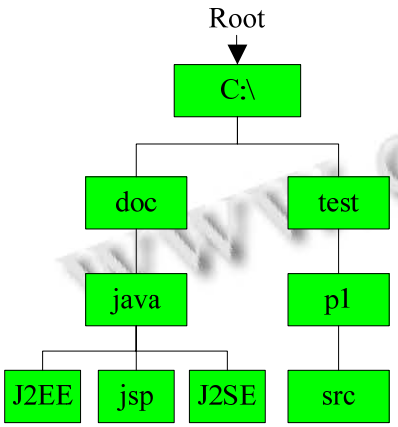

图 7 整理后的树形结构

UserManager 层将图 7 中的 Root 句柄返回给 NFS 层, 并最终返回到客户端. 返回的文件句柄是根据用户的信息定制生成的, 与实际的文件文件结构并不相同, 客户端在操作文件句柄时, 只能沿着虚拟的路径进行操作, 这样就实现了客户端访问路径的差异化管理.

3 实验验证

3.1 实验环境

本测试的客户端采用 CNC 中运行的 Linux 操作系统, 开发平台为 QT, 开发语言为 C++.

服务器为 Windows 主机, 开发平台为 QT, 开发语言为 C++, 服务器 IP 为 192.168.1.132.

客户端和服务器位于同一局域网内.

3.2 测试方法

根据要求需要测试同一用户在不同 IP 客户机挂载时得到的路径是否相同以及不同用户使用相同 IP 挂载时得到的路径是否有差异, 设计对照实验. 实验中的 exports 配置为:

c:\doc\ro

c:\test\rw

manage 配置为:

Brian:c:\test\p1\src,c:\doc\java

Lucy:c:\test\p1\bin,c:\test\p1\src

测试步骤及结果如表 1 所示(注: 表 1 中的目录是指客户端挂载文件系统后, 在挂载目录下包含的目录名)

表 1 测试步骤及结果

步骤	操作	目录	结果
1	在 192.168.1.132 的服务器启动服务		
2	在 192.168.1.11 上的 CNC 上登录 Brian		
3	执行 mount -t nfs 192.168.1.132:\c:\test \mnt\tmp	p1	正确
4	切换为用户 Lucy		
5	执行 mount -t nfs 192.168.1.132:\c:\doc \mnt\tmp2	空	正确
6	执行 mount -t nfs 192.168.1.132:\c:\test\p1 \mnt\tmp3	bin,src	正确
7	在 192.168.1.13 的 CNC 上登录 Brian		
8	执行 mount -t nfs 192.168.1.132:\c:\doc \mnt\tmp	java	正确
9	执行 mount -t nfs 192.168.1.132:\c:\test \mnt\tmp	p1	正确

3.3 测试结果分析

实验结果表明, 服务器为 Brian 和 Lucy 分配的资源实现了基于用户身份的定制化, 在步骤 5 和步骤 8 中, 虽然两次实验的挂载命令和客户机 IP 都相同, 但由于登陆用户不同, 得到的文件句柄不同, 而步骤 3 和步骤 9 说明同一用户在 IP 不同的客户机挂载相同路径时, 可得到一致的文件结构. 这说明在传统 NFS 模

型中增加 UserManager 层和 manage 配置文件后, 满足了按照用户标识分配资源的需求. 实现了不同用户使用相同 IP 地址挂载时可以得到不同的访问路径, 而同一用户在不同 IP 的 CNC 上挂载时可以得到相同的访问路径.

4 结语

在分布式数控系统应用 NFS, 解决了数控机床存储容量限制的问题. 传统的 NFS 服务使用 IP 管理文件资源, 导致了不同用户在相同 IP 的客户机挂载时得到的是相同的文件结构. 在实际应用中, 不同权限的用户应该拥有不同的访问路径, 不同功能的 CNC 需要挂载的程序路径也不同. 为了增加访问文件时的安全性和定制性, 在原有 NFS 结构中增加 UserManager 层和 manage 配置文件,保证了用户挂载的是一个私有的虚拟文件结构.

参考文献

1 陶林,蒋廷彪,张向利.基于以太网的分布式数控系统实时通信网络研究.制造技术与机床,2009,(11):114–119.

2 王冬,王超,韩永,等.iSCSI 与 NFS 的协议开销对比. 计算机工程与应用, 2012, 48(36).

3 叶军,朱华生.嵌入式 LinuxNFS 方式下应用程序的实现.微计算机信息,2007(3Z):74–75.

4 陈道喜.VM 中 Fedora8 系统下 NFS 的安装与配置研究.智能计算机与应用,2012,2(2): 7–29.

5 李丽娜.网络存储系统性能优化的设计和实现.微型电脑应用,2010,26(8):3–5.

6 李颖,刘金刚,李锦涛,等.一种面向用户的网络文件系统远程访问控制方法.计算机工程,2005,30(22):21–23.

7 Muthitacharoen A, Chen B, Mazieres D. A low–bandwidth network file system. ACM SIGOPS Operating Systems Review, 2001, 35(5): 174–187.

8 Lombard P, Denneulin Y. nfsp: a distributed NFS server for clusters of workstations. Proc. International Symposium of Parallel and Distributed Processing(IPDPS 2002). Abstracts and CD-ROM. IEEE. 2001. 6.