网络文件系统中的元数据存取优化研究

Research for Meta-data Operation Optimizing in Network File Systems

中国科学院计算技术研究所智能中心　贺劲　吴思宁　孟丹　徐志伟　（北京 100080）

摘　要：文章研究了客户机/元数据服务器/存储服务器三层结构网络文件系统中，针对改善元数据服务效率的优化策略，包括客户节点接口中的路径解析加速、元数据服务器上的元数据存取优化及元数据服务器组的结构支持等几部分，作者通过建立分析模型与实际系统模拟的方法，证明这些策略的确可改善系统性能，优化系统效率。

关键词：机群文件系统，文件元数据，全路径解析

1　引言

计算机外存储系统已经成为现代计算系统进一步提高性能与可靠性的主要障碍。网络技术的高速发展使网络文件系统成为解决此问题的有效手段之一。

图 1 给出了一种三层结构的网络文件系统的结构示意图，该结构包含了文件系统客户节点、文件服务器组、存储单元组及配置管理节点等四部分。这几部分之间通过 IP 网络或者其他网络（如光纤通道与 IP 网络的混合链路）等连接在一起。

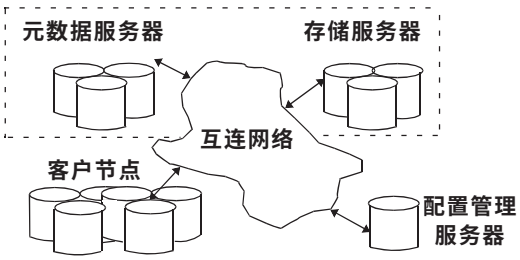


图 1　网络系统结构示意图

图 1 中的存储服务器组可以是连接高性能磁盘子系统的一组文件服务器节点，由这些文件服务器来管理网络文件系统中所有数据文件的存储。存储服务器也可以是一组高性能存储设备，如一组链接到光纤通道交换机上的光纤通道存储设备，此时网络文件系统也就是一个 SAN1 文件系统（如 IBM 的 SANergy[1]等）。

在网络文件系统中，独立的元数据管理器组负责管理此文件系统的文件数据分布/位置信息、网络文件系统目录文件及普通网络文件系统中文件元数据（包括文件长度、访问控制列表、权限、日期及其它属性信息）的存储。

网络文件系统中的配置管理服务器则负责监控系统中所有部件的状态以及响应系统管理员发出的各种文件系统维护或控制命令。

以上三种节点上对应的服务进程通称为文件服务器，它们运行在机群节点上并为网络文件系统的客户节点提供文件共享服务。

图 1 中的虚框包括的元数据与存储两类服务器节点，在具体实现中，这些服务器节点指运行了相关系统服务进程（或者核心模块，如 NFS 协议中的服务器端核心服务模块 NFSD[2,3]）的节点机。当两种服务进程同时运行在同一个节点上时，元数据服务器同时也是存储服务器。

之所以需要将服务器进一步分类的根据在于：

(1) 有些文件操作无需涉及它所包含的数据，而只与文件属性或者目录数据等元数据信息有关，因此可以将这两部分任务分派给两类不同的服务器进程，通过减轻服务器的负载以提高客户节点文件操作性能；

(2) 通过分离文件元数据与数据，使得两种类型的服务器都可以根据需要进行扩展，从而增强系统的可扩展能力；

(3) 在服务器的磁盘上，由于文件元数据信息与文件数据具有不同的存取特性与尺寸，比如一般元数据信息要远远小于文件数据等，因此可以在服务器方设计不同的本地存储策略，通过优化服务器性能来提高客户节点文件操作性能。

在网络文件系统中，客户节点上的用户或应用存取文件的路径与本地文件系统有很大差异，在本文后续小节将对这两种文件系统中文件存取的关键路径进行全面的分析。

2　网络文件系统的元数据存取

在 POSIX 标准[4]的定义中，进程只有获得文件句柄或者文件名字之后才能操纵存储在文件系统中的文件数据或文件属性。在 Linux 进程上下文中，每个文件句柄唯一对应一个文件对象是进程核心部分的 file 结构，而文件名在所有进程上下文中都唯一对应某个确定的文件[3, 5]。

收稿日期：2002-06-27

在文件系统中,对某个特定文件的标识是通过同样存储在文件系统中的 i 节点来进行的，在 i 节点结构中还存放着文件存取权限、长度及日期等重要信息。当应用进程以文件名字为参数,发出文件存取操作请求后,文件系统核心首先需要找到文件名所对应的唯一 i 节点,然后再进行后续处理。从文件名到对应 i 节点定位的过程一般称之为文件名字的 Lookup 过程，它类似于 Internet 上的域名到 IP 地址的 DNS 域名解析过程。

常用的文件系统元数据操作主要有以下几类：

(1) getattr,取文件属性；

(2) setattr,设置文件属性；

(3) lookup,进行文件名到 i 节点的路径查找；

(4) readlink,取符号链接的实际路径名；

(5) fstat,取文件系统状态统计数据；

(6) readdir,读取目录文件内容；

(7) create,创建文件。

研究人员在为网络文件系统 NFS 构造一个新的 SPEC 基准(SFS 2.0)前,对 750 台运行 NFS V2(版本 2)的 Auspex 服务器进行专门的负载研究，希望能从收集的负载数据中获得一些文件负载共性。这些研究人员发现，根据收集的数据，负载基本上可以分为两大类,如图 2 所示[6]。在图 2 的两组负载模式中，元数据相关的存取操作比例分别达到了 81%与 51%。可见在某些应用模式中，元数据操作的数量要远远超过普通文件数据读写操作的数量。这种频繁发生的元数据服务请求是否能及时得到响应，在很多应用场合决定了系统的整体性能。

Operation	Cluster1	Cluster2
null	1%	0%
getattr	25	15
setattr	1	1
lookup	40	23
readlink	7	2
read	11	32
write	5	17
create	2	1
remove	1	0
readdir	7	5
fsstat	1	2

图 2 数据存取操作比例统计结果

3 元数据存取关键路径分析

本节将分析网络文件系统中元数据操作的关键路径，并着重剖析了元数据操作中最重要的一个操作:文件名字查找 Lookup 操作的流程，然后指出目前该流程中存在的一些严重制约系统性能的缺陷。

3.1 本地节点 VFS 层

VFS(Virtual File System)是 Sun 微系统公司提出的在单个操作系统上支持多种不同类型文件系统的文件操作抽象层[7]。图 3 是其实现原理图。

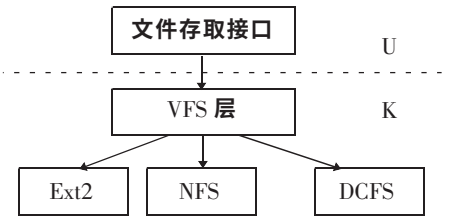


图 3 VFS 层接口示意图(U:用户空间,K:核心空间)

当用户程序通过标准文件系统调用接口操纵文件时,首先将陷入操作系统核心,相关的文件系统调用参数被传递到操作系统的 VFS 层,然后 VFS 根据被操纵对象所属的具体物理文件系统,将请求转发给底层物理文件系统。

VFS 层定义了一系列标准操作，主要划分为如下几个部分：

(1) 文件系统信息收集与操作；

(2) 数据文件读写操作；

(3) 目录相关操作；

(4) 文件 (数据与目录文件) 属性操作等。

在具体系统实现中，每个 POSIX 标准中定义的系统调用由一个或几个 VFS 层操作组合而成。不同操作系统 VFS 层的具体实现差异较大，以开放源码的 freeBSD 与 Linux 为例，从它们的核心源代码中就可以清楚地发现这一点。

在众多 POSIX 标准定义的文件系统调用中,有许多调用需要被操作对象的文件名字。前面已经介绍过,VFS 层所能处理的对象都必须有对应的 i 节点，因此需要一个从文件名字到某个特定 i 节点的转换接口，文件系统名字空间的 Lookup 操作就是来完成这一任务的。在常用的操作系统,如 Linux、FreeBSD 以及 AIX，都在 VFS 层上定义了一个 lookup 操作接口,底层文件系统可以通过此接口实现其特殊的 Lookup 函数。

3.2 VFS 层 Lookup 策略缺陷

一般 Lookup 接口需要两个参数，分别是父目录的 i 节点以及本对象的名字,也就是路径名。在许多操作系统上(如 AIX、Linux 及 FreeBSD 等),传递给 VFS 层 Lookup 接口的文件名字并非整个路径

名，而是要分多次将每个路径分量依次送交给底层文件系统的 Lookup 接口。

对于 open("/x/y/z/a/b/c")，即打开目录“/x/y/z/a/b”下的文件 c 这个调用，一般 VFS 层需要对“/x”，“y”，“z”，“a”，“b”，“c”调用六次底层文件系统的 Lookup 接口才能最终获得对象 c 的 inode。

本地文件系统通过使用名字缓存来进行 Lookup 查找优化，因此这种操作性能基本可以接受，但对于通过 VFS 层扩充实现的网络/网络文件系统，由于需要频繁地穿越连接网络而引入严重地网络访问延迟，这种 Lookup 操作的开销将随网络文件系统目录层次深度的增加线性上升，因此这种实现方式将严重地影响系统中元数据操作性能。

4　普通 Lookup 操作优化策略

由于 Lookup 操作在某种程度上决定了分布/本地文件系统元数据操作的性能，因此大多数分布/本地文件系统都对此操作进行了优化。本小节就目前分布/本地文件系统中已经广泛使用的名字空间 Lookup 操作的优化策略进行分析。

4.1　名字解析缓存

网络文件系统 NFS[2,8]与 COSMOS[9]在客户端节点中使用了 Lookup 结果缓存，即缓存了部分文件系统对象名字与 i 节点对应表。这样将来对相同对象名字的 Lookup 操作将不必穿越网络去获取相应的结果。

另外 NFS 在服务器端也使用了所谓“响应缓存”(response cache)[2,8]，即将以前名字解析的结果缓存在文件服务器内部，从而加速服务器对客户节点发出的 Lookup 请求的响应速度。

从实际效果来看，使用 Lookup 缓存在一定程度上改善了 Lookup 操作的性能，但时它仍然避免不了分由 VFS 层 Lookup 实现策略带来的多级路径分量查询的开销。

4.2　绕开 VFS 层限制

Dartmouth 学院的 Galley[10]采用了避开 VFS 层限制的办法，即使用“平板”名字策略，即所有 Galley 文件系统中的文件都在根目录下，这样所有 Galley 文件系统名字空间中对象的路径分量都不会超过 1，这样，即使当元数据操作需要穿越网络时，也只增加了一次通信开销。

虽然 Galley 的设计思路并不是针对优化元数据操作，不过对于一个可使用系统来说，这种方式并不可取[10]，因为网络文件系统的 I/O 节点常常使用某种本地文件系统，如 Linux 上的 Ext2、AIX 上的 JFS 等，来存放实际的文件数据，因此单个目录下的文件个数总被限制在一个较小的数值以内，实际上使得系统可用性受到很大的制约。

4.3　通信网络性能优化

网络文件系统中的 Lookup 操作只需要传输少量数据，因此机群内部互连网络延迟性能对它有较大的影响。

一般基于商品化部件的机群系统使用传统的 100Mbps 或者 1Gbps 以太网以及普通的 TCP/IP 协议栈作为内部连接网络。但是在这种网络环境中，端到端的延迟较大，约在 100μs 左右。而如果采用高性能机群内部互连网络与高性能通信协议，如 COSMOS[9]中曾经尝试将 Myricom 公司的 Myrinet[11](延迟）与国家智能计算机研发中心研制的 BCL-3 协议[9,12]相结合，点到点延迟可以降低到 20μs 以内，这种网络环境已使 Lookup 性能得到明显改善。

5　FPLS——基于目录子树的路径解析优化

FPLS(Full Path Lookup Strategy)是对以上优化策略的一个综合与改进，它直接针对文件系统客户端本地 VFS 层的限制，提出了一种新型的网络文件系统动态部署策略，突破了路径分量分析的制约；同时通过相应元数据服务器在结构组织的调整，结合传统的优化策略，以期望获得更好的 Lookup 及其他元数据操作性能。

5.1　基本策略

首先，本文作者认为：Lookup 操作产生大量延迟的本质原因是传统 VFS 层的多级目录解析限制与元数据服务器方的磁盘访问延迟，但首先需要打破 VFS 层这种制约，才能从在当本地缓存缺失情况下提高该操作的性能。这可以采取直接修改操作系统 VFS 层源码的方式进行，但对于无法获取源码的系统这种方式无法实现，因此作者尝试了直接修改操作系统核心符号表的做法，原理实验在 Linux 上已经获得成功。

其次，当本地 VFS 层如果可以一次发出所有的路径分量，则若希望避免多次网络通信，则元数据服务器的组织方式也需要进行相应的调整。本文作者在系统研制过程中借鉴了参考文献[13]中的有关思路，将管理单个网络文件系统中文件系统卷的所有元数据服务器组织成为一个二级树结构，如图 4 所示。其中超级管理器管理整个文件系统卷的超级块以及根目录，其它管理器则负责管理根目录下的

一个或几个子目录下所有文件对象的元数据。这种策略与服务器端 Lookup 缓存结合起来，共同优化服务器端的 Lookup 性能。

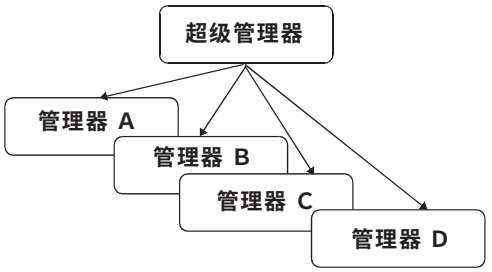


图 4 网络文件系统元数据服务器结构

网络文件系统客户端可充分利用这一特性，通过在本地缓存整个根目录文件来减少访问管理器中元数据的次数。假设客户端的文件系统卷根目录缓存采用表 1 的结构来记录信息：

表 1 网络文件系统客户端根目录缓存格式

目录名称	管理器编号
Abc	2
Def	4

此时当应用通过 VFS 层发出对此网络文件系统分区上名字为‘/abc/123’的对象的 Lookup 操作时，分布文件 VFS 层核心模块将迅速确定此对象所在管理器位置，再直接向此管理器发出路径解析。

管理器进程接收到来自客户代理的路径解析请求及相应参数后，其搜索算法还是沿路径分量层次逐级查找，如果在该路径中包含了符号链接且该链接所对应物理目录没有存储在该管理器中，则管理器将发回给客户代理进程已解析完路径分量的描述。

5.2 Lookup 缓存策略

网络文件系统可以同时在客户端与管理器进程中设置了数量较大的路径解析缓存。在查找时，首先在客户端的路径解析缓存中搜寻 ;如果搜寻成功，则立即返回该结果 ;否则将向管理器进程发出路径解析请求。

由于存在于多个不同客户节点上的进程可能同时对某一相同父目录下的文件或目录进行操作，这样可能导致在多个客户节点上看到了不同的目录内容，即带来了元数据缓存的不一致性。

为了尽量减少由于这种不一致所带来的问题，网络文件系统的客户端路径解析缓存可以对于目录入口的修改操作采用一种简单的策略 :即直接将修改结果发送到管理器 ;而对于其它客户节点上对此目录的读操作，此网络文件系统则确保在某一设定时间后可以看到修改结果，这也就是所谓的基于时间的路径解析缓存策略。

另外，在网络文件系统管理器进程内部也设置了更大容量的路径解析缓存，其基本单元就是一个目录文件的内存映象，但通过 hash 表等快速查找数据结构组织在一起，以便通过以路径名为参数的 hash 算法快速获得所对应项的位置指针。

6 元数据缓存服务器结构

6.1 FPLS 策略中的负载平衡

当网络文件系统的元数据服务器组织为图 4 中的结构时，如果此文件系统分区的某个子目录下的元数据成为热点数据且客户端内部缓存也无法容纳这些数据时，这些 Lookup 操作将集中到单个元数据服务器上，这样将造成严重的负载失衡。

采用元数据服务器复制的方式，即将热点数据复制到多个元数据服务器中，可以减轻单个元数据服务器的负载，此时网络文件系统客户端可以从多个元数据服务器中选择一个进行 Lookup 操作，但这种策略同时将带来一个问题，即 :多个服务器上的副本必须保持一致，否则将出现元数据的不一致问题。这种情况下的一种简单解决策略是采用单服务器副本可写、多服务器可读（即 Multi-Reader Single-Writer）的方式来工作，则当负载模式变成大量的对元数据的写操作时，这种简单的元数据分布策略仍然将导致负载失衡。

6.2 FPLS+: 改进的 FPLS 结构

因此针对以上问题，本文作者提出了改进的 FPLS 结构。图 5 给出了 FPLS 与 FPLS+（改进的 FPLS）结构比较的示意图。在 FPLS+结构中，网络文件系统的元数据服务由前端的元数据缓存服务器组与后端的元数据服务器协作来提供。

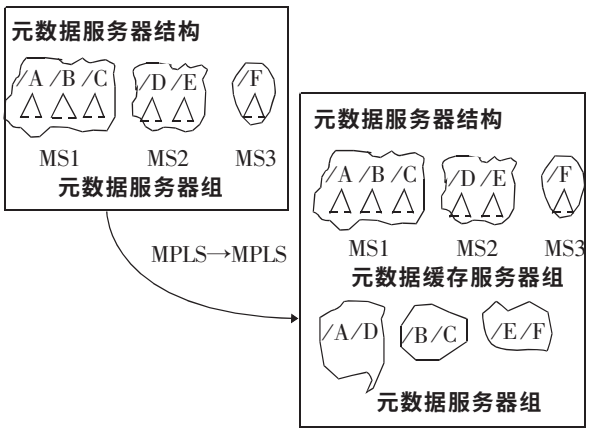


图 5 改进的 FPLS算法与元数据服务器结构

网络文件系统客户端的元数据请求全部由前端的元数据缓存服务器（图中的 MCS——Metadata Cache Server）完成，但是这些元数据缓存服务器本地不存储任何网络文件系统分区中的元数据，只是在内存中存在元数据副本。

后端的元数据服务器在与其连接的物理存储设备中都保存真正的网络文件系统元数据，同时也在内存中提供这些元数据的缓存以提高响应速度。

在具体实现中，由于后端的元数据服务器对网络文件系统客户节点不可见，因此在 FPLS 基础上，网络文件系统客户端无需进行更多的修改。另外，由于引入 FPLS+的目的主要是为了实现元数据服务器上的存储设备负载平衡，因此可以将 MS（Metadata Server）与 MCS 实现为运行在相同服务器上的两个进程甚至单个进程内的两个线程。

7　性能模型

7.1　FPLS .vs. NFS

本小节提出了一个分析模型来比较基于 FPLS 策略的网络文件系统与 NFS V3 的文件名字 Lookup 路径解析操作效率。文中给出了该分析模型中 Lookup 路径解析模拟负载的产生与模拟测试流程：

(1) 首先，根据用户设定参数生成一定的目录结构；

(2) 然后，产生类似某种应用的负载，即某些路径解析请求发送给模拟的文件系统客户端；

(3) 在模拟的客户本地端进行缓存查找，如果命中则返回；

(4) 否则发送给模拟的网络传输系统；

(5) 在服务器端的缓存中进行查找，若命中则发送给模拟的网络传输系统返回结果；

(6) 否则将请求发送给模拟的 I/O 子系统；

(7) 通过网络返回给客户端。

假定客户端缓存共有为 n 项，平均命中率为 λ_1，访问此缓存的平均延迟为 d_1；系统中存在 M 个元数据服务器，每个服务器端的缓存为 $(100\times n)/M$ 项，其平均命中率为 λ_2，访问此缓存的平均延迟为 d_2。

设基于 Ethernet 的 TCP/IP 协议在传输短消息（小于 300 字节）网络传输延迟为 td_1。

设元数据服务器 I/O 子系统访问时延为 td_2，根据参考文献[2,6]，它服从如下一般分布，这里为了简化起见，将 td_2 定义为一个常量（可取磁盘寻道与旋转延迟的上限）。

在解析一个路径深度为 S 的目录时，NFS V3 与 FPLS 网络文件系统所需时间 T_1 与 T_2 分别由如下等式得到：

$$T_1=S\times(\lambda_2\times d_2+2\times td_1+(1-\lambda_2)\times td_2) \quad (1)$$

$$T'_1=S\times(\lambda_1\times d_1+(1-\lambda_1)(\lambda_2\times d_2+2\times td_1+(1-\lambda_2)\times td_2)) \quad (2)$$

$$T_2=\lambda_1\times d_1+(1-\lambda_1)(\lambda_2\times d_2+2\times td_1+S\times(1-\lambda_2)\times td_2) \quad (3)$$

式(1)是屏蔽客户端元数据缓存的 NFS 路径查找时间，式(2)与式(3)分别是使用相同客户端元数据缓存算法，且缓存大小相同的 NFS 与多元数据服务器的网络文件系统的路径查找时间。

由于内存访问延迟在纳秒级，而 Ethernet 网络延迟与磁盘 I/O 延迟都在一百微秒到几毫秒的范围内，因此可以简单认为内存访问延迟可以忽略不计。因此式 1 到三可以简化为如下：

$$T_1=S\times(2\times d_1+(1-\lambda_2)\times td_2) \quad (4)$$

$$T'_1=S\times(1-\lambda_1)(2\times td_1+(1-\lambda_2)\times td_2) \quad (5)$$

$$T_2=(1-\lambda_1)(2\times td_1+S\times(1-\lambda_2)\times td_2) \quad (6)$$

设 α 与 β 分别为 T_2 与 T_1 及 T'_1 的差，且 $\lambda'_1=1-\lambda_1$，$\lambda'_2=1-\lambda_2$，则：

$$\alpha=2\cdot td_1\cdot(S-\lambda'_1)+S\cdot\lambda'_2\cdot td_2\cdot(1-\lambda'_1)$$

$$\beta=2\cdot\lambda'_1\cdot td_1\cdot(S-1)+td_2\cdot S\cdot\lambda'_1\cdot\lambda'_2$$

由于 $\forall S, S\geq 1$，因此 α 与 β 恒大于等于 0，因此可见，基于 FPLS 的多元数据服务器结构与单元数据服务器且需要多次解析的算法有效。

7.2　FPLS+ .vs. FPLS

由于在 FPLS+中增加了一级缓存，因此需要在 6.1 的路径解析分析模型上进行一些修改工作。

首先，当使用 FPLS+策略的网络文件系统路径解析 Lookup 在本地缓存缺失的情况下的关键路径如下：

(1) 在 MCS 缓存中查找，若命中，则返回路径解析结果回客户端；

(2) 否则在 MS 的缓存中继续查找，若命中，则将 MCS 缺少的内容返回，并由 MCS 更新本地缓存，最后返回给客户端；

(3) 如果该解析操作在 MS 的缓存中未命中，则在本地存储设备上寻找相应的名字解析结果，并更新本地缓存，同时转为执行(2)。

由于此时 MCS 中的虚拟路径树中的路径分量可以存放在多个 MS 中，因此在最差情况下，当整条路径分量完全不在 MCS 的缓存中，且该需查找的路径分量全部位于不同的 MS 中时，MCS 必须依次与所有 MS 发出路径解析请求，直到所有的路径分量被解析完成。

因此在解析 0 中的路径时，使用 FPLS+策略的

网络文件系统所需要的最大时间 T_3 由式(7)得到：

$$T_3=S\times(3\times td_1+td_2) \quad (7)$$

相比 T_1，当客户端缓存完全未命中时，T_3 仅增加了一次网络开销，但相对于 FPLS+结构带来的负载均衡优势，这些开销是可以容忍的。

8 仿真性能比较

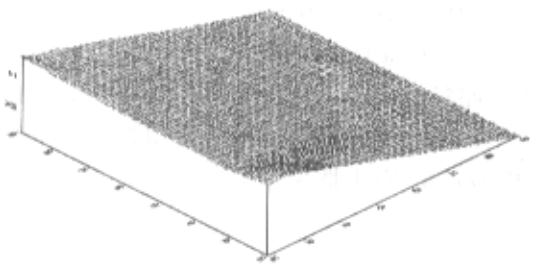
本小节将通过多组模拟真实负载，仿真比较 NFS 与基于 FPLS 策略、多元数据服务器的网络文件系统在处理路径解析的效率。

模拟环境由客户端与服务器端组成，为了便于调试，将客户端实现为一个服务器进程，模拟本地应用的进程通过向它发送消息来获取文件服务。

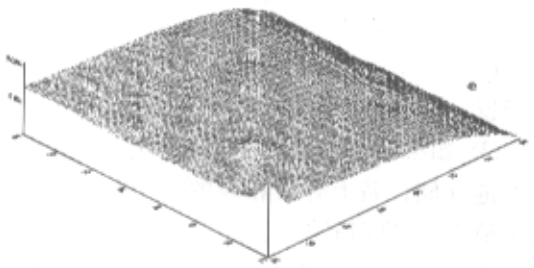
服务器端也实现为一个用户级的进程，它管理一个树形的内存数据结构，当出现缓存不命中时，将产生一系列的磁盘 I/O 操作。

被模拟的网络文件系统目录层次深度为最大为 12 级，在模拟系统中的网络 100Mbps Ethernet，I/O 子系统为 SCSI 硬盘。

图 6 给出了模拟系统的测试结果，从中可见应用 FPLS 策略的网络文件系统与标准 NFS 相比，目录解析操作效率确实得到了提高(NFS 路径查找操作的时间是 FPLS 策略的 1.46~4.96 倍)。



(a) NFS Lookup 性能模拟结果



(b) FPLS Lookup 性能模拟结果

图 6 不同缓存命中率下的 Lookup 效率比较

9 结束语

本文全面分析了改善网络文件系统中的元数据处理效率的策略，提出了基于客户端优化的 FPLS 策略，同时针对它带来的负载失衡问题进行了改进，最后通过模拟实验，验证了有关的想法。但是由于目前实际系统尚未完全建立，因此还不能对此系统进行全面的实际性能测试，因此今后将对这些策略进行实际应用环境下的性能测试与评价。

参考文献

[1] Alessandro Rubini. Linux Device Driver. 1998 O'Reilly & Associaties, Inc.

[2] Brent Callaghan. NFS Illustrated, Addison-Wesley. April, 2000.

[3] Charlotte Brooks, Ron Henkhaus, Udo Rauch, Daniel Thompson. A Practical Guide to Tivoli SANergy. IBM Red-books, SG24-6146-00.

[4] Dino Quintero, Zbigniew Borgosz, Andre Botura, Darren Gilchrist, Stefan Kister, Octavian Lascu, Kenneth So. RS/6000 SP Cluster: The Path to Universal Clustering. IBM Redbooks, SG24-5374-01.

[5] D Robinson. The advancement of NFS benchmarking: SFS 2.0. In Proceedings of the 13th USENIX Systems Admin-istration Conference (LISA '99), pages 175~185.

[6] Haskin, R L. Tiger Shark-a scalable file system for multimedia. IBM Journal of Research and Development, March, 1998, 42(2): 185~197.

[7] (ISO/IEC) [IEEE/ANSI Std 1003 1, 1996 Edition] Information Technology-Portable Operating System Interface (POSIX (r))-Part 1: System Application: Program Interface (API) [C Language]. ISBN 1-55937-573-6.

[8] Jason Barkes, Marcelo R Barrios, Francis Cougard, Paul G. Crumley, Didac Marin, Hari Reddy, Theeraphong Thi-tayanun. GPFS: A Parallel File System. SG24-5165-00. April 1998.

[9] Kleiman, S Vnodes: An Architecture for Multiple File System Types in Sun UNIX. In Summer Usenix Conference Proceedings, Altanta(1986).

[10] Linux Source Code. http://www.kernel.org/pub/.

[11] MA Jie, HE Jin, MENG Dan and LI Guojie. BCL-3: A High Performance Basic Communication Protocol for Com-modity Superserver DAWNING-3000. Journal of Com-puter Science & Technology, November, 2001, 16 (6): 522~530.

[12] N J Boden, D Cohen, R E Felderman, A E Ku-lawik, C L Seitz, J N Seizovic, and W Su. Myrinet: A Gigabit-per-second Local Area Network. IEEE Micro, February, 1995, 15(1): 29~36.

[13] Nils Nieuwejaar and David Kotz. The Galley parallel file system. In Proceedings of the 10th ACM International Conference on Supercomputing. ACM Press. Philadelphia, PA, May, 1996, 374~381.

[14] Randolph Y Wang. Improving the I/O Performance and Correctness of Network File Systems. Ph D thesis. UC Berkeley, Spring, 1999.
[15] RFC 3010, December, 2000.
[16] W Liu, X Ou, M Wu, W Zheng, M Shen. A Distributed Naming Mechianism in Scalable Cluster File Sys-tem, Proceedings of 4th HPC-ASIA, Beijing, China, 14-17 May, 2000, 1: 37~41.
[17] 贺劲, 徐志伟, 孟丹, 马捷, 冯军. 基于高速通信协议的 COSMOS 机群文件系统性能研究. 计算机研究与发展, 2002, 2, 129~135.
[18] 王建勇. 可扩展的单一映象的文件系统. 博士学位论文, 中科院计算所, 1999, 6.
[19] M Y Kim and A N Tantawi. Asynchronous Disk Interleaving: Approximating Access Delays. IEEE Trans. On Computer, July, 1991, 40, 801~810.

HE Jin, WU Si-Ning, MENG Dan, XU Zhi-Wei
(National Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

Abstract: In network file systems with client/meta-data severs/storage severs architecture, it is importance to improve the performance of meta-data operations. We have studied the optimizing strategies to improve the performance of meta-data operations, which include a strategy of meta-data layout structure adjusting and an improved the pathname lookup algorithm. In this paper, we also put up with an analytical model and a simulate model to analyse these optimizing strategies. The result showed that these optimizing strategies is effective.

Key words: Network file systems, Meta-data, Pathname lookup

(上接第 40 页)

Grid. In: proc. of Int'l Conference on High Performance Computing in Asia-Pacifc Region (HPC Asia 2000), May, 2000.
[8] L Boloni and D C Marinescu. An Object-oriented Framework for Building Collaborative Network Agents. In: A Kandel, K Homann, D Mlynek, and N H Teodorescu, editors. Intelligent Systems and Interfaces, Kluwer Publishing, 2000, 31~64.
[9] H Casanova and J Dongarra. Netsolve: A Network-enabled Server for Solving Computational Science Problems. Int'l Journal of Supercomputer Applications and High Performance Computing, Fall, 1997, 11(3): 212~223.
[10] Fran Berman and Rich Wolski. The AppLeS Project: A Status Report. The 8th NEC Research Symposium, Berlin, Germany, May, 1997. http://apples.ucsd.edu
[11] Spyros Lalis and Alexandros Karipidis. JaWS: An Open Market-Based Framework for Distributed Computing over the Internet. IEEE/ACM International Workshop on Grid Computing (GRID 2000), Dec. 2000. http://roadrunner.ics.forth.gr:8080/
[12] H Nakada, M Sato, and S Sekiguchi. Design and Implementation of Ninf: Towards a Global Computing Infrastructure. Future Generation Computing Systems (Metacomputing Special Issue), October, 1999.
[13] D Carvalho, F Kon, F Ballesteros, M Romn, R Campbell, and D Mickunas. Management of Execution Environments in 2K. In: proc. of the 7th Int'l Conference on Parallel and Distributed Systems (ICPADS '00), July, 2000, 479~485.
[14] Andy Oram. Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly & Associates, Inc. USA, March, 2001.
[15] http://www.platform.com, July, 2002.
[16] http://www.gridforum.org, July, 2002.
[17] M Wahl, T Howes, S Kille. Lightweight Directory Access Protocol (v3), RFC2251, December, 1997.

JIA Ming-fei (Xi'an Jiaotong University, Xi 'an 710049)

Abstract: Grid resource management system is the important component of a grid computing system which is now being investigated and developed widely and deeply. This paper has analyzed the functional requirements for grid resource management systems and classified them into three basic models according to their architecture. Based on the above analysis and classification, a general and abstract model for grid resource management systems has been proposed and also a prototype system for the model has been constructed based on the campus scale grid environment. Finally, the related works have been discussed.

Key words: Grid resource management, Grid computing, Resource allocation, Resource reservation, Job scheduling

贾明飞 男,(1974–),硕士研究生。主要研究方向为计算机系统结构、高性能计算和并行编译。
董渭清 男,(1953–),副教授。主要研究方向为计算机系统结构、高性能计算和网络安全认证。
桂小林 男,(1966–),博士研究生,副教授。主要研究方向为计算机系统结构、高性能计算等。
白雪柏 女,(1978–),硕士研究生。主要研究方向为计算机系统结构、网络存储等。

网络文件系统中的元数据存取优化研究

作者： 贺劲， 吴思宁， 孟丹， 徐志伟
作者单位： 中国科学院计算技术研究所智能中心,北京,100080
刊名： 微电子学与计算机 ISTIC PKU
英文刊名： MICROELECTRONICS & COMPUTER
年，卷(期)： 2003，20(3)
被引用次数： 0次

参考文献(19条)

1. Alessandro Rubini Linux Device Driver 1998

2. Brent Callaghan NFS Illustrated Addison-Wesley April2000

3. Charlotte Brooks. Ron Henkhaus. Udo Rauch. Daniel Thompson A Practical Guide to Tivoli SANergy

4. Dino Quintero. ZbigniewBorgosz. Andre Botura RS/6000 SP Cluster:The Path to Universal ClusteringIBM Redbooks SG24-5374-01

5. D Robinson The advancement of NFS benchmarking:SFS 20 In Proceedings of the 13th USENIX Systems dmin-istration Conference (LISA'99) pages

6. Haskin R L Tiger Shark-a scalable file system for multimedia IBM Journal of Research and DevelopmentMarch 1998 42(2): 185～197

7. (ISO/IEC) [IEEE/ANSI Std 1003 1 1996 Edition] Information Technology-Portable Operating System Interface (POSIX (r-Part 1 System Application Program Interface (AP1) [C Language]

8. Jason Barkes. Marcelo R Barrios. Francis Cougard Theeraphong Thi-tayanun GPFS: A Parallel File System SG24-5165-00 1998

9. Kleiman S Vnodes: An Architecture for Multiple File System Types in Sun UNIX In Summer Usenix Conference Proceedings Altanta 1986

10. Linux Source Code 查看详情

11. Ma Jie. HE Jin. Meng Dan. LI Guojie BCL-3: A High Performance Basic Communication Protocol for Com-modity Superserver DAWNING-3000[期刊论文]-Journal of Computer Science and Technology 2001(06)

12. N J Boden. D Cohen. R E Felderman. A E Ku-lawik, C L Seitz, J N Seizovic, and W Su. Myrinet A Gigabitper-second Local Area Network 1995(01)

13. Nils Nieuwejaar. David Kotz The Galley parallel file system 1996

14. Randolph Y Wang Improving the Ⅰ/O Performance and Correctness of Network File Systems 1999

15. 查看详情 2000

16. W Liu. X Ou. M Wu A Distributed Naming Mechianism in Scalable Cluster File Sys-tem 2000

17. 贺劲. 徐志伟. 孟丹. 马捷. 冯军 基于高速通信协议的COSMOS机群文件系统性能研究[期刊论文]-计算机研究与发展 2002(02)

18. 王建勇 可扩展的单一映象文件系统[学位论文] 1998

19. M Y Kim. A N Tantawi Asynchronous Disk Interleaving: Approximating Access Delays 1991

相似文献(1条)

1. 会议论文 金翊. 洪学海. 詹剑峰. 孙凝晖 Spreader:一种基于元数据的机群文件管理系统 2007

随着机群计算能力的增长和应用的深入,机群系统中的文件数量也随之急剧增加。在拥有海量文件的大规模机群中快速搜索进而操作所需要的文件,已成为机群研究面临的重要问题。本文提出一种新颖的基于元数据的智能机群文件管理系统--Spreader。Spreader以对单机文件系统透明的方式实现了单一系统映像的机群文件管理,利用成熟的关系型数据库技术实现海量文件的快速定位,和传统的文件系统相比提供了更丰富的描述文件语义的扩展文件属性机制。实验表明,Spreader在提供了方便、灵活的机群文件查询和管理机制的同时,能够将上亿规模的文件查找性能提高到数百毫秒。