第 32 卷 第 21 期 计 算 机 工 程 2006 年 11 月
Vol.32 No.21 Computer Engineering November 2006
·软件技术与数据库· 文章编号：1000—3428(2006)21—0067—03 文献标识码：A 中图分类号：TP303

多级集群文件系统的全局命名空间的设计与实现

封仲淹，万继光，李锡武，李 旭
(华中科技大学计算机学院信息存储系统教育部重点实验室，武汉 430074)

摘 要：随着存储数据的爆炸性增长和集群技术的快速发展，集群文件系统的研究越来越成为一个焦点，该文设计并实现了一个多级集群文件系统（MCFS）的全局命名空间，MCFS 系统采用元数据分层管理思想，建立一个统一的元数据树，从而使整个系统运行在一个松耦合、异构的环境下实现全局命名空间，提高了系统的并行性和可扩展性。在介绍了 MCFS 命名空间管理的同时，进行了相应的试验测试和性能分析。

关键词：并行文件系统；分布文件系统；多级集群文件系统；全局命名空间；元数据树

Design and Implementation of Global Name Space in Multilevel Cluster File System

FENG Zhongyan, WAN Jiguang, LI Xiwu, LI Xu
(National Storage System Laboratory, School of Computer, Huazhong University of Science and Technology, Wuhan 430074)

【Abstract】 With the quick development of storage and cluster technology, the cluster file system research becomes a more and more important issue. This paper presents the design and implements the global name space in the multilevel cluster file system(MCFS). It brings forward a metadata multilevel policy, thus forms a uniform metadata tree, so multilevel file cluster system can implement the global name space in a loose coupling, isomerous circumstances, improving the system's parallelism and scalability. After introducing the global name space in the multilevel cluster file system, some experiments is made to analyze the performance and scalability.

【Key words】 Parallel file system; Distribute file system; Multilevel cluster file system(MCFS); Global name space; Metadata tree

随着信息技术的发展，科学技术、工程计算对各种数据存储技术提出更大容量、更高性能的要求。虽然现有的分布式文件系统如 NFS、AFS 和 Coda 等已得到了广泛的应用，但它们并不能完全满足机群系统的需要。它们的主要目标是为网络用户提供共享数据的手段，并不能在单一系统映像、可扩展性和高性能等方面达到统一，为了适应集群系统应用对文件系统的要求，我们设计并实现了多级集群文件系统(Multilevel Cluster File System, MCFS)。本文重点研究了 MCFS 的元数据管理，采用元数据分层管理思想，建立了一个统一的元数据树，从而使整个系统运行在一个松耦合、异构的环境下实现全局命名空间。

1 元数据管理的设计与实现

为了实现 MCFS 全局命名空间，让用户都可以用相同的名字来访问该文件或者目录而无须关心文件的实际存储位置和给其提供服务的元数据服务器的位置，并且为了避免单一服务器瓶颈问题，采用了元数据的分布式管理，在元数据分布式管理的条件下，通过元数据的分层思想，减少了相互间的通信量，提升了性能，充分保证了跨平台的特性，并允许针对具体应用而采用特定的优化措施。

1.1 元数据的分布式管理

目前主流的并行文件系统或分布式文件系统通常采用集中式元数据管理，例如虚拟并行文件系统(pvfs)[1]、General Parallel File System(GPFS)[2]、Slice File system(SFS)[3]、蓝鲸分布式文件系统(BWFS)[4]等，这种集中式元数据管理方式通常会带来一些问题：

(1)所有对文件的访问至少要分为 2 个步骤，首先从元数据服务器获得元数据，然后定位到具体存储节点上具体位置；元数据的获得会增加网络的负荷，而且增加文件访问的步骤，同样会引起风险的增加。

(2)集中式的元数据管理，很容易出现瓶颈和单点失效，所有的计算节点在访问文件之前，都必须发送请求到元数据服务器，从而元数据服务器就成为一个系统的瓶颈，而且如果元数据服务器一旦崩溃，很有可能导致整个系统陷入瘫痪状态。

(3)为让系统具备扩展性，这种集中式元数据都采用了复杂的管理技术，以便让新增加的节点，尤其是元数据服务器的节点能够正常融入旧的系统中去。MCFS 系统如图 1 所示。

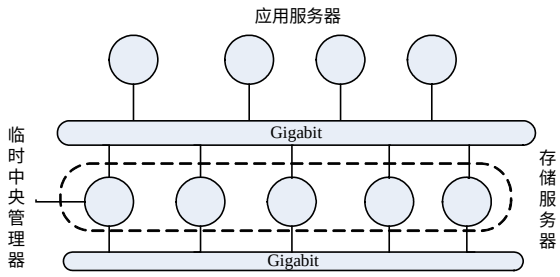

图 1 MCFS 系统的架构

基金项目：国家“973”计划基金资助重大基础研究项目(2004CB318200)
作者简介：封仲淹(1981－)，男，硕士生，主研方向：网络存储系统；万继光，博士生；李锡武，硕士生；李 旭，博士生
收稿日期：2005-12-01 **E-mail：**longdafeng@yahoo.com.cn

所有的存储服务器的地位是等同的，当系统运行时将选举产生一个临时中央管理器，但当中央管理器失效时，立即从存储节点中选举产生一个新的临时中央管理器，而临时中央管理器并不负责元数据进行定位，而是提供普通的存储服务和额外附属操作，如提供对系统的热点文件的处理和备份操作的管理，根据各个存储服务器反馈的负载情况，采用对应的策略来处理热点文件，然后将处理结果通知给所有的存储服务器，从而刷新整个系统的元数据，而存储服务器采用高自治的分布式策略，各个存储节点能够独立地维护自己的存储资源和文件的元数据及数据本身，并能够独立通过文件服务。整个系统没有专门负责元数据的服务器，整个系统的元数据是通过把不同存储服务器上的局部配置信息整合成一个全局配置信息。每一个存储服务器只负责管理自己的局部配置信息，应用服务器在第 1 次连接存储服务器时，会自动请求全局配置信息，从而也会构建一颗完整的元数据树。

1.2 元数据的分层管理

整个多级集群文件系统的元数据分为 2 个层次，如图 2 所示，分别由底层的宿主元数据管理和上层的 MCFS 元数据模块管理。通过一个例子来从理论和具体操作上说明 MCFS 元数据的层次模型。

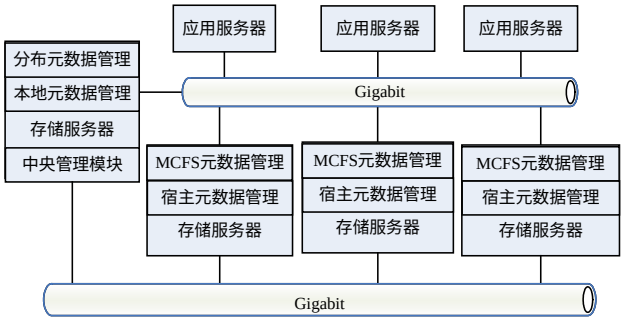

图 2 元数据分层管理

先解释 2 个特有的概念：本地节点，分布节点。假设在逻辑视图中(用户所看到的视图)如图 3 所示，节点 2 是节点 b 的子节点；如果节点 2 对应的物理节点是节点 b 对应物理节点的子节点，并且子节点 2 的控制属性(主要是指涉及安全的元数据信息)与节点 b 对应的物理节点的控制属性相同，则节点 2 就称为本地节点，反之，则称为分布节点。多级文件系统的元数据管理器只负责管理分布节点的元数据。对于本地节点，其主要的元数据主要是由宿主文件系统管理。

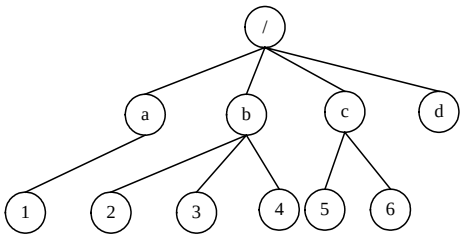

图 3 元数据的层次模型

通过协调宿主文件系统和 MCFS 构成一棵完整的元数据树，并使整个系统具有全局名字空间，仍以图 3 的用户逻辑视图为例，在该例中，假设根节点 a、b 、c 和 5 节点是分布节点，而 d、1、2、3、4 和 6 节点是本地节点，再假设根节点和 a 节点属于存储服务器 1，b 和 c 节点属于存储服务器 2，而 5 节点属于存储服务器 3，则在存储服务器 1 中的元数据配置信息中就会存储根节点和 a 节点的配置信息，同理，文件服务 2 含有 b 和 c 节点配置属性，而存储服务器 3 含有 5 节点的配置文件信息，每一个节点的配置信息均可以构建一个节点数据结构，然后，每个存储服务器的心跳管理器接收其他存储服务器的配置信息并发送自己的配置信息，形成一个统一的全局配置信息，通过该全局配置文件系统，每个存储服务器均构建一个统一的全局元数据树，其模型如图 4。

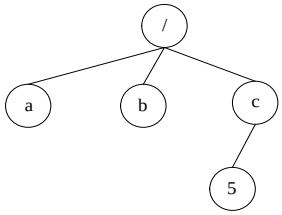

图 4 多级文件系统形成的元数据树

这种元数据的分层思想将会拥有如下优点：

(1)可以大幅度地减少元数据的管理信息，减少元数据的管理控制和资源消耗(例如元数据树通常是在内存中构建)，从而可以提高整个系统的性能和减少失效的概率。从理论上，根据局部性原理，一个节点很有可能会与他相邻的节点属性相同，一个子节点很有可能就会与其父节点的相同。

(2)兼容性强，宿主是一个健全的文件系统，在为多级文件系统提供服务的同时，还可以为其自身的 OS 提供文件服务。这一点不会像 pvfs 那样，pvfs 的 I/O 节点中的很多文件理论上是可以为其本身 OS 提供服务，但是实际中，其文件分块了，而且元数据不健全很难为其本身 OS 提供文件服务。并且存储服务器之间可以是异构的，不需要强制为同构的。

(3)本地文件系统可以是任何随意的文件系统，可以是NTFS、FAT32、EXT2、EXT3 等，可以针对系统的具体应用有一些特殊优化，例如，如果使用环境用于提供多媒体服务的环境下，那可以大量使用对顺序大块读操作有优势的文件系统，如集成多媒体文件系统[5]。

2 分布元数据同步的设计与实现

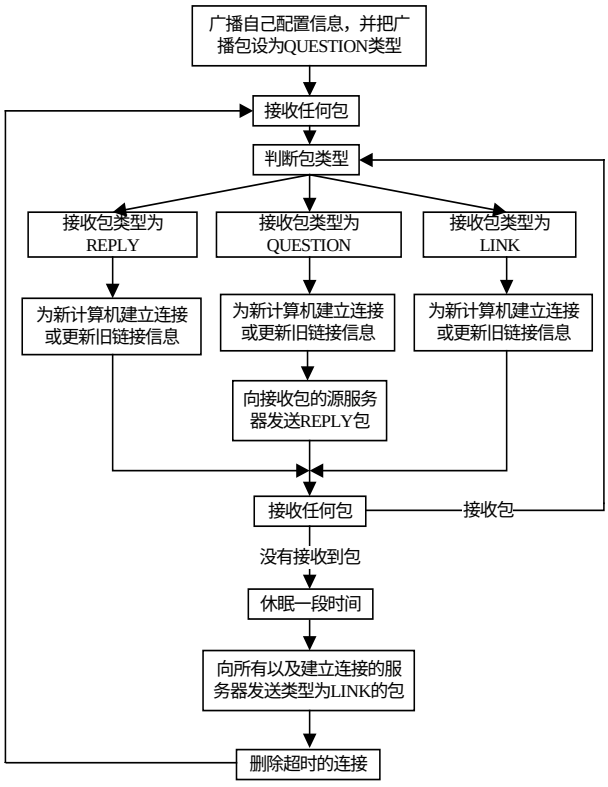

图 5 心跳管理程序核心流程

在多级文件系统中，所有的异构服务器都是通过心跳管理器发送自己的局部配置信息，接收其他服务器的配置信息，从而构建成为一个完整的全局配置信息，最终根据这些全局配置信息形成一个统一的元数据树，实现全局命名空间。

心跳管理器的所有收发包都是基于 TCP/IP，并通过高速专用网络，避免占用高速存储网络带宽而影响整体的性能。自定义的心跳管理协议不像通常的心跳协议那样仅仅采用广播的方式发送信息，采用广播和定点通信的方式进行收发信息。通过这种方式，可以避免大量的广播占用带宽，并且容易形成广播风暴；其次，提高各服务器间通信的效率，避免重复发送无用包。试验证明：这种广播和定点链接方式效果理想，十分稳定，并且在运行过程中可任意动态添加或删除节点。整个心跳管理器的流程如图 5。

全局配置文件主要是在更新链接或创建链接模块中更新，当出现一个新的链接或是一个旧的链接版本发生变化时，更新全局配置文件，当出现超时的链接时，同样更新全局配置文件。

3 系统测试与分析

测试说明：在测试中屏蔽了内存并行文件系统这一级别，因为内存并行文件系统的使用，会存在一个命中率的问题，命中率高时和命中率低时二者系统性能相差十分巨大，为了真实挖掘 MCFS 的潜力，所以屏蔽了内存文件系统这一级别。我们的测试环境是国家外存储重点实验室(武汉)，如表 1。

表 1 测试环境

	CPU	内存	文件系统	网卡	硬盘
应用服务器	至强 1.8GHz	512MB	MCFS	Intel PRO/1000 MT Server Adapter	Maxtor 6y060L0
存储服务器	至强 1.8GHz	512MB	NTFS 或 EXT3	Intel PRO/1000 MT Server Adapter	Maxtor 6y060L0

使用目前比较流行的 Intel 的 iometer-2004.07.30-post.DS1.common-src.tar(Linux)和 iometer-2004.07.30-post.DS1.win32-bin(Windows)。进行顺序读吞吐率测试。

3.1 跨平台测试

此项主要测试各种平台下性能：(1)都采用 Windows 2003，在图 6 中用 MCFS_Cw_Sw 曲线表示；(2)应用服务器采用 Windows2003，存储服务器采用 Linux redhat 9.0，图 6 中用 MCFS_Cw_Sl 曲线表示；(3)应用服务器采用 Linux redhat 9.0，存储服务器采用 Windows 2003，图 6 中用 MCFS_Cl_Sw；(4)都采用 Linux redhat 9.0，图 6 中用 MCFS_Cl_Sl 曲线表示。

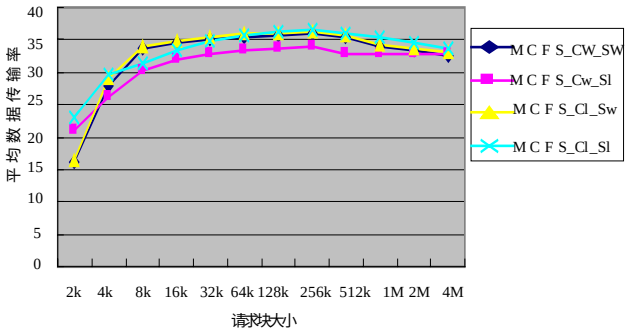

图 6 跨平台顺序读

测试结果分析：

(1)整个系统可以成功实现跨平台运行，并且元数据分层思想可以有效地在各种本地文件系统上实现。

(2)各平台组合的性能相差无几，但应用服务器基于 Windows，而存储服务器基于 Linux 时，性能最差，而同时基于 Linux 平台的性能最佳。对源代码进行分析，其原因如下：当应用服务器和存储服务器使用相同的系统时，其对命令处理步骤更为精简，减去了不少进行转化的操作，命令格式更为和谐一致，相同平台下性能会更好；其次，应用服务器使用 Linux 性能要优于使用 Windows，Linux 下 MCFS 更为接近底层驱动层，减少了类似 Windows 的转发消息，整体表现要优于基于 Windows 的。

(3)当 request size 为 256KB 时，整个系统的性能最佳，主要原因归结于应用服务器端和存储服务器端的 Cache，在应用服务器端 Cache 大小为 256KB，而存储服务器端为 512KB，应用服务器端一次 Cache 命中失败后，除了请求对应的数据外，还会预读几块 256KB 大小的数据快，这直接导致当 request size 为 256KB 时，系统性能最佳。

(4)因为宿主文件系统的原因，当请求块较小时(<64KB)，存储服务器基于 Linux 平台的性能要优于基于 Windows 平台的性能，而当请求块≥64KB 后，二者相差无几。

3.2 与 samba 和网络邻居的性能对比

此项测试主要是通过测试本地的 EXT3 文件系统，本地的 NTFS 文件系统，基于 NTFS 文件系统的网络邻居，基于 EXT3 的 Samba，来对比 MCFS 应用服务器和存储服务器均使用 Linux 的性能。如图 7 所示。

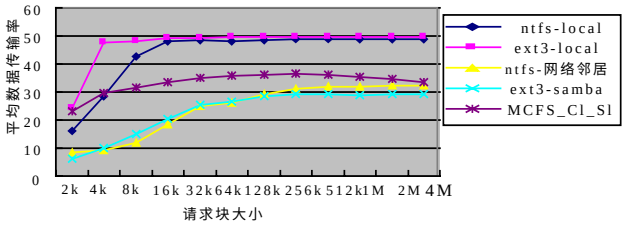

图 7 性能对比

测试结果分析：(1)各平台下的 MCFS 系统性能均优于基于 EXT3 的 samba 性能和基于 NTFS 的 Windows 网络邻居性能。尤其是当 request size 大小很小时(低于 64KB 时)，性能差距更大，分析其原因在于：相对 samba 和网络邻居，MCFS 较少受 request size 影响，当 MFS 应用服务器端 Cache 一次命中失败，它会立即请求对应的数据外，还会预读几块 256KB 大小的数据，因此，它本身就减少了 4KB 请求包的数量。(2)相对于 samba 和网络邻居，MCFS 的性能曲线更为平滑，其原因主要归结于 Cache，Cache 的存在使整个请求包序列中还是以应用服务器端缓存块大小为主，减少了受 request size 的影响，从一定的程度上减少了性能颠簸的可能性。

4 结束语

本文从无集中式服务器的角度，探讨了一种适合集群技术的多级集群文件系统，该文件系统通过使用元数据分层思想，建立了一棵统一的元数据树，使整个系统运行在一个松耦合、异构的环境下实现全局命名空间。性能测试结果表明，该模型使得多级集群文件系统具有很好的性能以及很好的可扩展性。

参考文献

1 Ligon W B, Ross R B. Implementation and Performance of a Parallel File System for High Performance Distributed Applications[C]. Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing, 1996.

(下转第 72 页)