第 33 卷 增刊　　华 中 科 技 大 学 学 报(自然科学版)　　Vol. 33　Sup.
2005 年　12 月　　J. Huazhong Univ. of Sci. & Tech. (Nature Science Edition)　　Dec.　2005

HANDY 集群文件系统

程 斌　金 海

(华中科技大学 计算机科学与技术学院, 湖北 武汉 430074)

摘要: 针对当前集群文件系统缺乏动态扩展性和高可用性的不足,提出了一种基于逻辑矢量环结构的集群文件系统 HANDY. 系统利用逻辑矢量环实现动态扩展和资源的动态管理,采用了分布式元数据管理策略,引入邻接复制技术实现元数据的容错,通过可定制的数据存放策略满足不同应用的数据存储需求. 测试结果表明,基于环状结构的系统 HANDY 具有良好的动态扩展性和高可用性.

关　键　词: 集群; 文件系统; 动态扩展; 高可用

中图分类号: TP393　　**文献标识码**: A　　**文章编号**: 1671-4512(2005)S1-0034-05

HANDY: a cluster file system with dynamic scalability and high availability

Cheng Bin　Jin Hai

Abstract: Most of existing cluster file systems are shortage of dynamic scalability and high availability, so we provide a new cluster file system based on logical vector rings, named HANDY. It adopts logical vector rings to implement distributed management of metadata and dynamic management of storage resource. Through neighbor replication mechanism, it provides high availability of metadata. In addition, a flexible, customized data distribution scheme is employed in HANDY, which meets storage requirements of all kinds of different applications. The experimental results show that HANDY cluster file system based on logical vector rings achieves good dynamic scalability and high availability.

Key words: cluster; file system; dynamic scalability; high availability

Cheng Bin　Doctoral Candidate; College of Computer Sci. & Tech., Huazhong Univ. of Sci. & Tech., Wuhan 430074, China.

根据调查发现,当前多数集群文件系统[1]都只能实现静态扩展,由于静态扩展不需要考虑数据容错、分布式的元数据管理、资源动态分配等问题,因而在实现上相对容易,但由此也带来一些其他的问题,如系统扩展起来复杂、可用性不高、管理和维护困难等. 当节点数增加到几十或几百时,任何节点的失效都可能导致整个系统出错,系统的可用性受到严重影响.

针对上述问题,本文提出一种新的基于逻辑矢量环的集群文件系统——HANDY. 该系统的设计目标是提供一种基于普通存储设备的、具有高可用性和动态可扩展性的集群文件系统. 它采用分布式的元数据管理,实现节点动态管理和容错机制,能够自动屏蔽部分节点失效的故障,消除单一失效点,减小整个系统配置和管理复杂性.

1　系统结构与动态扩展

如图 1 所示,整个系统分为四大组成部分:元数据服务器池、数据服务器池、客户端、管理控制台. 各部分之间通过高速网络互联,如 1 000 Mbyte 的以太网、Myrinet 等,其中元数据服务器、数据服务器通过动态扩展协议各自维护一个逻辑矢量环,以表示节点之间的逻辑关系. 每个元数据服务器通过 Berkeley DB 存放元数据信息,包括文件和目录的访问权限、创建修改时间、目录项、存放策略、分片文件的 Handle 列表等,每个数据服务器主要存放文件的数据分片. 客户端为上层应

收稿日期: 2005-08-25.

作者简介: 程　斌(1979-),男,博士研究生;武汉,华中科技大学计算机科学与技术学院 (430074).

E-mail: showersky@hust.edu.cn

基金项目: 国家高技术研究发展计划资助项目 (2002AA1Z02).

用程序提供集群文件系统的访问接口,包括三种形式的接口:VFS 内核虚拟文件系统的访问接口、库函数访问、ROMIO 并行程序访问接口;管理控制台部分提供 HANDY 文件系统的监控管理工具,查看各个服务器的资源利用情况、节点逻辑关系、统计信息以及数据和元数据的读写带宽等.

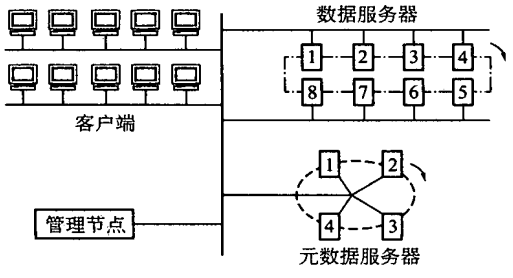


图 1　系统结构图

HANDY 系统在设计上具有如下特征:通过动态扩展协议实现节点的动态扩展;基于逻辑环实现数据和元数据的邻接容错,满足集群文件系统高可用方面的需求;实现元数据服务的分布式管理,消除集中管理所带来的性能瓶颈;实现可定制的数据放置策略的管理,适应不同应用程序的存储需求.

动态扩展是指系统在运行中能够自适应地加入一个新节点或去掉一个已存在节点. HANDY 系统采用一种基于逻辑矢量环的方法来维护节点间的逻辑关系. 它给环上的每个节点分配一个唯一的 ID 号,按从小到大顺序排列,然后将首尾相连,形成一个闭合的环,环的方向由 ID 号小的节点指向 ID 号大的节点,每个节点有一个前续和一个后续. 如图 2 所示,每个节点向后续节点发送心跳消息,同时检测前续节点的失效. 当接收线程

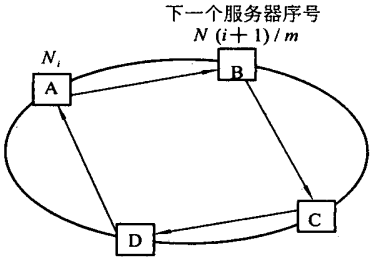


图 2　逻辑矢量环

在预定的时间内没有收到来自前续节点 $N_{(i-1)}$ 的心跳信息时,就断定前续节点已经发生故障,对外发送组播消息,宣告 $N_{(i-1)}$ 已经失效. 当出现节点失效和加入时,通过组播消息来通知所有节点.

为了保证系统在动态扩展的过程中不影响系统的整体可用性,需要对系统的资源进行动态管理. HANDY 系统中所有的存储对象都由一个唯一的 Handle 值来标识,它是一个 64 位的数值. 为了便于实现对象的寻址,将 Handle 的 64 个 bit 位进行分段,用于表示不同的意思. 如图 3 所示,前 1 位表示管理该 Handle 的服务器节点类型,0 表示元数据服务器,1 表示数据服务器;接下来的 9 位表示逻辑环的逻辑 ID,因此数据和元数据的逻辑环最大可以支持 $2^9 = 512$ 个节点;最后的 54 位表示节点内部的 Handle 地址空间,也就是说每个节点可以管理分配的 Handle 数目大小为 2^{55}. 系统中存在如下几种的存储对象:目录和文件的元数据信息,目录项信息,分片文件.

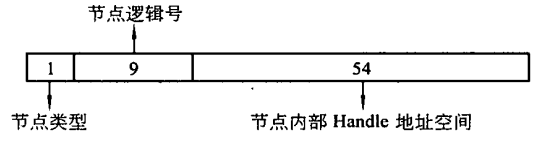


图 3　Handle 位段划分

ServerMap 表是一个全局的数据结构,由有序链表构成,它位于系统所有节点上,记录每个服务器节点负责的 Handle 范围,用于定位某个 Handle 对应的服务器节点的地址信息. 设有 Node1、Node2、Node3 三个节点,Handle 空间为 $[0, 2^3 - 1]$,每个节点负责的 Handle 范围如图 4 所示. 当某个节点需要读取 6 所对应的存储对象时,它首先去查找 ServerMap 表,查出 6 由 Node3 负责,然后才向 Node6 发送请求.

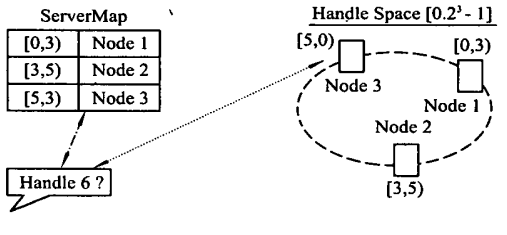


图 4　客户端访问 ServerMap 的过程图

客户端任何时候访问资源都必须首先知道资源对应的 Handle 值,然后才能定位对应的服务器节点并发送请求,因此动态扩展核心功能就是要实现资源的有效分布和快速定位. 为了实现资源的有效分布,HANDY 系统为每个加入的节点分配一个唯一的逻辑 ID 号,根据逻辑 ID 号的大小在元数据和数据服务器节点之间各自形成一个逻辑有向环,每个节点占据环上的一个位置 V_i,负责管理前续节点 $V_{(i-1)}$ 到自身的这一段 Handle 值所表示的对象. 为了实现资源的快速定位,HANDY 系统通过组播消息维护一个全局 Hash

表(ServerMap),客户端通过查询自己的ServerMap表来实现Handle到节点的定位.由于ServerMap表在每个节点上都有一个副本,所以基于HANDY资源定位只需要一次本地查找操作即可完成,开销非常小.

2 分布式元数据管理

2.1 邻接复制技术

单独的动态扩展协议本身并不能保证系统在节点离开或失效时能够继续工作,所以HANDY系统在逻辑矢量环的基础上,设计了一种邻接复制技术[2]来备份数据,保证扩展时元数据的高可用性[3,4].

邻接复制是指将节点的数据备份到自己的后续节点.对于元数据服务器采用的是服务器端复制的方式,即每个元数据服务器会将自己的数据存放在自己的后续节点,同时也将所有的元数据更新操作传递到自己的后续节点,完成后备数据的更新和同步.HANDY系统提供了基于UPDATE-ACK方式的邻接复制策略,提供同步和异步两种数据复制方法.只有UPDATE没有ACK的方式称为异步复制;有UPDATE和ACK的方式称为同步复制.

每个元数据服务器上的元数据存放在两个目录下,一个是主目录,一个是备份目录.每个元数据服务器将自己负责的元数据存放在主目录上,同时存放在后续节点的备份目录上.正常情况下,元数据的读操作仅仅访问主目录里面的元数据;写操作时,不但写本地主目录的元数据,还将构造备份操作将命令与数据发送至后续节点,后续节点协助在备份目录里完成相同的写操作.

实现时每个元数据服务器运行三个线程:一个发送线程,一个接收线程,一个处理线程,同时维护两个队列,一个发送队列,一个接收队列.元数据服务器每次对本地元数据的操作完成后,构造一个相应的备份操作的消息(包含操作类型与相应数据等信息),放到发送队列里面.发送线程不停的将发送队列里面的消息传至后续节点的接受线程.接收线程接收到消息后,也构造一个操作放入接收队列.处理线程逐个从接受队列里面取出消息,解析成有用的信息(操作类型和响应数据),对本地备份元数据执行相应操作.

2.2 元数据管理

HANDY系统实现基于动态扩展协议的分布式元数据管理策略.由于所有的存储对象都是由Handle标识,所以分布式管理的核心就是实现基于逻辑矢量环的Handle空间管理.HANDY根据各个元数据节点在环上的位置,将$[0,2^{63}-1]$的元数据Handle空间分成n个前后连续Handle Range,每个元数据服务器负责其中的一段.每个节点可分配的Handle Range空间大小为2^{54},如何对这段连续的空间进行有效地管理和分配是本节关注的重点.即使用最小的bit数组表示每个Handle的是否已分配,也需要消耗4096 TB的内存大小,显然不可行.为此本文用<first, last>二元组来表示某一个连续段的Handle状态,同时将这些二元组用平衡二叉树组织起来,加快平均查找时间.另外,在整个操作过程中既有插入操作也有删除操作,所以不断会有Handle需要分配和回收,如果每次操作都去执行二叉树的查找和更新操作,其效率势必很低,于是引入聚合回收算法来改进Handle的管理效率.算法的基本思想是先将释放的Handle组织在一个较小的二叉树中,待总数积累到一定程度或时间超过某个预定值才让其真正加入到可分配的二次树中.

3 可定制的数据放置策略

HANDY系统中实现了多种数据放置策略,如适合存取小文件的SIMPLE策略,适合存取可靠性要求不高的大文件的raid0策略,适合存取可靠性要求很高的大文件的raid1策略等.如图5所示,它通过引入数据放置策略属性来实现文件或目录的放置策略的定制和继承.文件系统创建时文件系统根目录具有一个缺省的放置策略(一般是raid0),根下面的目录和文件默认继承根的放置策略属性(如raid0目录),也可以在创建时指定自己的放置策略(如simple,raid1目录).当进行文件操作时,通过读取当前目录的属性,即可知道相应的数据放置策略,进行相应的数据放置操作.通过文件继承目录放置策略,子目录继承父目录放置策略,用参数修改目录策略属性等方法,用户可以根据自己的应用需求,定制相应的数据放置策略,从而实现集群文件系统数据可定制管理.

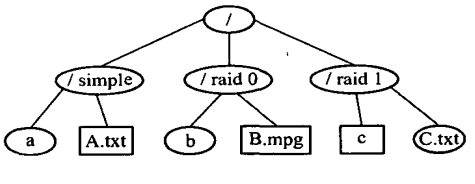


图5 放置属性的继承树

4　性能测试

4.1　测试环境

如表1所示(表中CPU均为P4 1.6 GHz,测试环境是一个36个节点的Cluster系统,所有节点均安装Redhat Linux9.0的系统,采用100 Mbyte Fast Ethernet互连.由IOZONE测得本地EXT2文件系统的读带宽为37.5 Mbit/s,写带宽为23.5 Mbit/s,由NETPIPE测得网络有效传输带宽为88.56 Mbit/s.测试工具采用通用的文件系统基准测试程序[5]: IOZONE, PostMark, Lmbench.

表1　测试环境

	内存/Mbyte	DISK/Gbyte	网卡/Mbyte	数目
元数据服务器	256	40	100	8
数据服务器	256	40	100	12
测试客户端	256	40	100	16

4.2　测试结果及分析

HANDY系统将元数据和数据服务器分离,利用动态扩展协议形成两个独立的逻辑环状结构,实现节点的动态扩展.为了便于分析,本文分别测试了元数据服务器和数据服务器的可扩展性以及系统的可用性.

4.2.1　元数据的扩展性

从图6测得的结果看,当元数据服务器数目

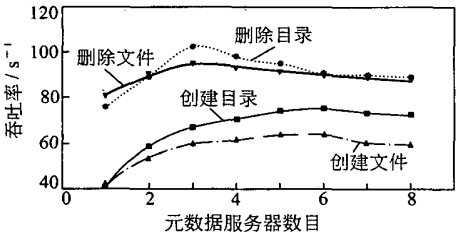


图6　元数据的吞吐率随元数据节点的变化

到达3时,文件和目录删除操作的吞吐率已经达到最大94.73/s⁻¹和102.04/s⁻¹,之后随着元数据节点的增加吞吐率反而减小;对应文件和目录的创建操作也同样遇到这种情况,当元数据数目达到6时,创建操作的吞吐率也达到最大63.78/s⁻¹和75.29/s⁻¹.究其原因,发现这种限制主要是因为在同一个目录下进行创建和删除操作,任何创建和删除操作都会涉及当前目录元数据的修改,因此存储当前目录元数据信息的元数据服务器成为扩展时的瓶颈.为了证明这种分析,测试了客户端在不同目录下运行结果.如图7所示(图中m为每秒完成的操作数),在8个节点以内时元数据服务器的扩展性非常明显,但随着节点数的增多,增长的幅度会有所减小,因为分布式元数据会导致用户访问时需要和多个元数据服务器通信才能完成文件路径的解析.

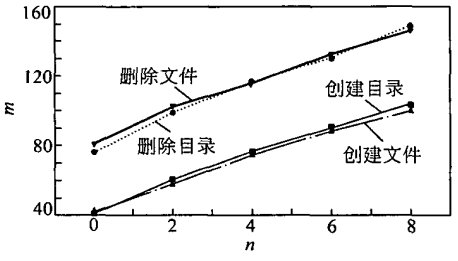


图7　不同目录下创建和删除的吞吐率

4.2.2　数据服务器的扩展性

图8是用IOZONE测得的不同数目数据服务器情况下聚合读写带宽的变化情况.测试时只配置了一个元数据服务器,数据服务器的数目N分别为1,2,4,8,客户端最多有10个,HANDY的放置策略为raid0.可以看到,当数据服务器增加时HANDY系统的聚合读写带宽也随之增加.图9显示数据服务器数目分别为1,2,4,6,8时HANDY系统测得的最大聚合读写带宽.1个服务器的最大读带宽为11.20 Mbit/s,写带宽为10.75 Mbit/s;2个服务器的最大读带宽为22.12 Mbit/s,写带宽为21.32 Mbit/s;当数据服务器数目增为8时,最大读带宽达到65.25 Mbit/s,写带宽为63.29 Mbit/s,这已经大大超过用IOZONE测得的本地文件系统的读写带宽.

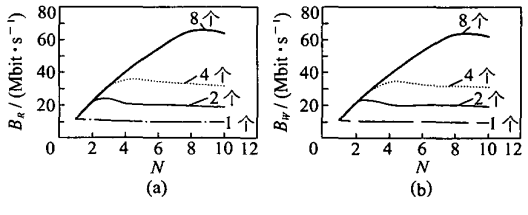


图8　raid0读写带宽

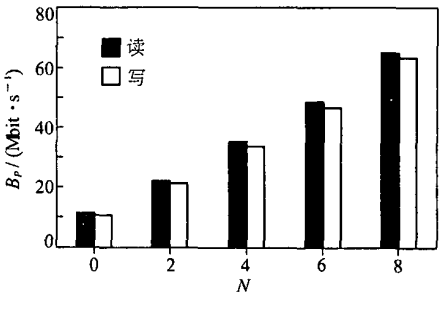


图9　最大读写带宽的变化

综上所得结果,可以发现:不管是元数据还是数据服务器,随着节点数的增多,系统吞吐率和聚合IO读写带宽也随之提高.但相比较而言,元数

据服务器的扩展幅度要小于数据服务器,我们认为这主要是由元数据服务器分布后客户端解析路径时的通信开销导致的.

4.2.3 可用性

为了达到高可用性,系统必须具有失效发现和失效恢复的能力. HANDY 系统的故障发现时间

$$T_{\text{found}} = T_{\text{detect}} + T_{\text{announce}} + T_{\text{update}},$$

式中: T_{detect}为心跳检测时间, $T_{\text{detect}} \leqslant 4$ s; T_{announce}为 Crash 消息通告时间; T_{update}为 ServerMap 表更新时间,这后两个时间都非常短,因此 HANDY 系统可以在 4 s 左右发现节点失效,这一点可以从图 10 的测试结果中看出来. 20 s 以前,系统中有 3 个数据服务其正常工作,聚合读带宽大约在 25 Mbit/s,20 s 种的时候,我们人为地去掉一个数据服务器,发现之后的 4 s 内读带宽降低很厉害,这主要是因为系统要读取失效节点的数据,又不知道该节点失效,于是不断地去连接失效节点. 4 s 后,客户端收到节点失效的消息,很快更新自己的 ServerMap 表,将请求定向到自己的后续节点,完成数据的读取. 但此时只有 2 个数据节点在工作,所有整体的聚合读带宽在 17～20 Mbit/s 之间波动,相对于原来的聚合读带宽有所下降.

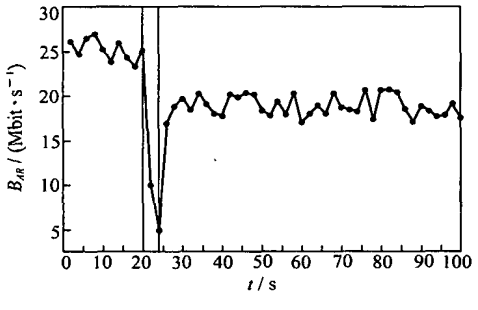
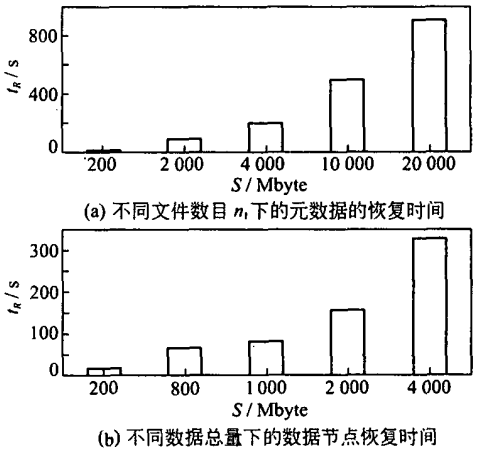


图 10 节点失效时的聚合读写带宽变化

当节点失效后重新加入时需要完成恢复操作,其恢复时间是个不定值,取决于需要恢复的数据量的大小. 图 11(a)给出 4 个元数据节点,系统总的文件数(n_1)分别在 200,2 000,4 000,10 000,20 000 时某个节点失效后重新恢复的时间,可以看到当文件数据总数到达几万个时,元数据的恢复时间比较长,说明系统在元数据的恢复策略上效率有待进一步提高和优化. 图 11(b)给出 4 个数据服务器,文件总量(S)分别为 200,800,1 000,2 000,4 000 Mbyte 情况下一个数据节点失效后又重新恢复起来的时间.



(a) 不同文件数目 n_1 下的元数据的恢复时间

(b) 不同数据总量下的数据节点恢复时间

图 11 元数据服务器和数据服务器的恢复时间

参 考 文 献

[1] 章隆兵,陈意云. 基于分布式共享存储系统的并行文件子系统 DPFS[J]. 计算机研究与发展, 2002, 39(3): 360—365

[2] 赵 东,姚绍文,周明天. 一种适应性复制协议的研究与设计[J]. 电子学报, 2002, 12(A): 45—50

[3] 庞丽萍,何飞跃,岳建辉等. 并行文件系统集中式元数据管理高可用系统设计[J]. 计算机工程与科学, 2004, 26(11): 86—89

[4] 葛江伟,田 捷,崔伟东. 一种集群环境下高可用的 NFS 服务器[J]. 工业控制计算机, 2002, 30(8): 222—224

[5] 胡雨壮,孟 丹. 一个提高机群文件系统吞吐率的方法及其实现[J]. 计算机工程与应用, 2003, 18(2): 87—89