

中国科学院计算技术研究所

硕士学位论文

蓝鲸分布式文件系统数据一致性语义研究

姓名：范勇

申请学位级别：硕士

专业：计算机系统结构

指导教师：许鲁

20060601

摘 要

随着并行处理技术及分布式应用的发展,传统存储模式在性能、容量、共享、可用性、可扩展性、可管理性等方面的局限性越来越多地暴露出来,特别是共享问题在很大程度上限制了系统的使用模式,进而限制了系统规模(可扩展性)及性能,是传统存储模式亟待解决的问题。蓝鲸网络存储系统在解决上述问题的同时,考虑对应用的兼容性,这种兼容性一方面取决于系统接口,另一方面与系统的文件共享语义密切相关,数据一致性语义作为文件共享语义的重要组成部分是决定分布式应用模式的关键所在。本论文研究蓝鲸网络存储系统的核心系统软件——蓝鲸分布式文件系统(BWFS)的数据一致性语义问题。

论文首先归纳数据一致性语义的主要类型,概要地介绍若干有代表性意义的分布式文件系统针对相关问题的解决方案;然后从描述数据一致性问题入手,分析蓝鲸分布式文件系统的数据一致性问题;在此基础上,从节点间数据传输保障、文件属性更新机制和数据一致性语义模型等三个方面对蓝鲸分布式文件系统数据一致性语义进行研究。取得如下主要成果:

1) 提出并实现基于软件的网络容错技术

节点间数据传输的正确性及完整性是蓝鲸分布式文件系统数据一致性的基础。针对蓝鲸分布式文件系统中影响较大的通道连接软故障中断异常采用独立于系统硬件支持的连接复制、通道切换、请求重构等软件技术加以容错,实现应用层透明的数据无损传输,提高了系统的可用性,使得蓝鲸分布式文件系统的数据一致性有所保障。

2) 设计并实现自适应的带外模式文件属性更新机制

文件属性更新是蓝鲸分布式文件系统数据一致性的前提。论文设计并实现一种自适应的带外模式文件属性更新机制,该机制结合机会更新、被动更新、周期更新等多种文件属性更新方式,能够根据当前的网络状态自适应地调整文件属性的更新周期,允许用户动态地设置更新周期基数及更新周期调整幅度。

3) 设计并实现基于授权机制的在线可调整的数据一致性语义模型

分布式文件系统的数据一致性语义是决定分布式应用模式的关键所在,是本论文的研究重点。论文设计并实现一种数据一致性语义模型,该模型基于授权(可剥夺的文件级粒度的多态文件锁)机制,允许用户在线设置蓝鲸分布式文件系统的数据一致性语义,以满足不同应用模式的需求。该模型提供超时一致性、释放一致性、写一致性、读写一致性等四种数据一致性语义,支持从类 NFS 语义到类 UNIX 语义等多种数据一致性语义的兼容性。

关键词: 分布式文件系统、BWFS、网络容错、文件属性更新、数据一致性

Research on Data Consistency of Blue Whale Distributed File System

Yong Fan (Computer Architecture)

Directed By Lu Xu

With the development of parallel processing and distributed application, more and more weaknesses of traditional storage modes in performance, capacity, sharing, availability, expansibility, manageability and so on are exposed, especially the sharing issue which restricts the application modes, further more the system size (expansibility) and performance. It is the pressing issue to be solved. Blue Whale Network Storage System overcomes these issues and also deals with the problem of application compatibility. The compatibility depends on the system interface, and the system file sharing semantics is correlative. Data consistency, which is an important component of system file sharing semantics, is the decisive factor of distributed application modes. This dissertation explores data consistency semantics of Blue Whale distributed File System (BWFS)—the kernel software of Blue Whale Network Storage System.

Firstly, the dissertation induces the essential types of data consistency semantics, introduces the solutions adopted by some representative distributed file systems. And then describes the issues of data consistency and analyze the problems of data consistency in BWFS. Based on these, the dissertation studies the data consistency semantics of BWFS in three aspects, including data transmission guarantee between nodes, file attribute update mechanism and data consistency semantics model. The essential achievements are as follows:

1) Propose and realize soft-based network fault-tolerant technology

The correctness and integrality of data transmission between nodes is the foundation of data consistency in BWFS. To handle the malfunction which is caused by soft failures of connection channel broken accidentally and affects system badly, BWFS adopts the soft-technology of connection-copy, channel-switch, and request-rebuild for achieving network fault-tolerant without extra hardware support. It achieves lossless data transmission which is transparent to application layer, enhances system availability, and ensures the data consistency of BWFS.

2) Design and implement the auto-adapted file attribute update mechanism under out-of-band mode

File attribute update is the premise of data consistency in BWFS. The dissertation designs and implements the auto-adapted file attribute update mechanism under out-of-band mode. This mechanism integrates chance updating, passive updating and periodic updating. It can automatically adjust the file attribute updating period according to the network state. The user can set the base period and the range of updating period adjustment dynamically.

3) Design and implement an online adjustable data consistency semantics model based on authorization mechanism

Data consistency semantics of distributed file system is the decisive factor of distributed application modes. It is the emphasis of this dissertation. The dissertation designs and implements a model of data consistency semantics. It bases on authorization (deprivable file-granularity polymorphic file lock) mechanism, and allows user to set data consistency semantics online to meet the requirements of different application modes. It offers Timeout Consistency, Release Consistency, Write Consistency and Read-Write Consistency, and supports semantic compatibility from NFS Analogy Semantics to UNIX Analogy Semantics.

Keywords: Distributed File System, BWFS, Network Fault-Tolerant, File Attribute Update, Data Consistency

图目录

图 1.1	BWFS系统架构示意图.....	2
图 1.2	Lustre系统结构示意图.....	7
图 2.1	BWFS数据一致性问题描述图.....	13
图 2.2	写操作区域伪交叠示意图.....	15
图 2.3	分布式应用模式关联图.....	18
图 3.1	热备示意图.....	20
图 3.2	BWFS连接复制示意图.....	22
图 3.3	BWFS通道切换示意图.....	23
图 4.1	数据与文件属性变更关系图.....	29
图 4.2	带内模式文件属性更新示意图.....	30
图 4.3	带外模式文件属性更新示意图.....	30
图 4.4	BWFS文件属性更新周期自适应调整曲线	37
图 5.1	超时一致性语义下数据获取流程图.....	40
图 5.2	数据一致性语义兼容关系图.....	42
图 5.3	应用节点授权记录数据结构.....	46
图 5.4	应用节点授权记录逻辑视图.....	47
图 5.5	元数据服务器节点授权记录数据结构	48
图 5.6	元数据服务器节点授权记录逻辑视图	50
图 5.7	授权申请的请求协议.....	51
图 5.8	授权申请的应答协议.....	51
图 5.9	授权主动释放的请求协议.....	52
图 5.10	授权主动释放的应答协议.....	52
图 5.11	授权剥夺的请求协议.....	53

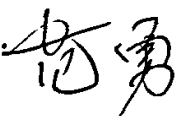
图 5.12	授权剥夺的应答协议.....	53
图 5.13	应用节点授权申请流程图.....	54
图 5.14	元数据服务器节点授权处理流程图	55
图 5.15	应用节点授权剥夺处理流程图	56
图 5.16	系统数据一致性语义模型授权机制的整体处理流程图	57
图 5.17	心跳请求协议.....	58
图 5.18	心跳应答协议.....	59
图 5.19	读写并发度与数据一致性语义关系图	71
图 5.20	读写并发度与授权申请次数关系图	72

表目录

表 3.1	BWFS基于软件的网络容错技术测试环境配置列表	25
表 3.2	BWFS基于软件的网络容错技术可行性测试结果	26
表 3.3	BWFS基于软件的网络容错技术性能测试结果	26
表 4.1	BWFS自适应的带外模式文件属性更新机制测试环境配置列表	36
表 4.2	BWFS自适应的带外模式文件属性更新机制性能测试结果	37
表 5.1	操作授权类型表	43
表 5.2	文件级授权粒度与数据块级授权粒度比较表	44
表 5.3	超时一致性授权兼容表	45
表 5.4	释放一致性授权兼容表	45
表 5.5	写一致性授权兼容表	45
表 5.6	读写一致性授权兼容表	45
表 5.7	模型异常处理与数据一致性安全策略对照表	68
表 5.8	BWFS基于授权机制的在线可调整的数据一致性语义模型测试环境配置列表...	69
表 5.9	超时一致性并发度测试结果	69
表 5.10	释放一致性并发度测试结果	70
表 5.11	写一致性并发度测试结果	70
表 5.12	读写一致性并发度测试结果	71


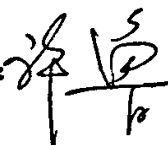
声 明

我声明本论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，本论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

作者签名:  日期: 2006.5.26

论文版权使用授权书

本人授权中国科学院计算技术研究所可以保留并向国家有关部门或机构送交本论文的复印件和电子文档，允许本论文被查阅和借阅，可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编本论文。

作者签名:  导师签名:  日期: 2006.5.26

第一章 引言

计算机系统在经历了从计算瓶颈向内存瓶颈的转变之后,正逐渐朝存储瓶颈(或称为 I/O 瓶颈)的方向发展,越来越多的应用,如视频点播、科学计算、医疗影像处理、地质数据分析等对存储系统提出越来越高的要求,传统存储模式在性能、容量、共享、可用性、可扩展性、可管理性等方面的局限性越来越多地暴露出来,特别是共享问题在很大程度上限制了系统的使用模式,进而限制了系统规模(可扩展性)及性能,是传统存储模式亟待解决的问题。

蓝鲸网络存储系统[1]是由国家高性能计算机工程技术研究中心(National Research Centre for High Performance Computers, NRCHPC)自主研发的用于解决系统存储瓶颈的海量网络存储系统,在解决上述问题的同时,考虑对应用的兼容性,尽可能降低已有应用的移植代价,使得基于传统存储模式(如 NFSv3 系统)的应用可以很方便地移植到蓝鲸网络存储系统上。这种兼容性一方面取决于系统接口,另一方面与系统的文件共享语义密切相关,数据一致性语义作为文件共享语义的重要组成部分是决定分布式应用模式的关键所在。本论文研究蓝鲸网络存储系统的核心系统软件——蓝鲸分布式文件系统(Blue Whale distributed File System, BWFS)的数据一致性语义问题。

1.1 蓝鲸分布式文件系统概述

蓝鲸分布式文件系统[2]是由国家高性能计算机工程技术研究中心自主研发的用于解决系统存储瓶颈的海量网络存储系统的核心系统软件,主要面向机群结构环境提供高速的、并发的、共享的、高可用的远程文件访问服务。该系统打破传统分布式文件系统单纯的客户/服务器模式(简称 C/S 模式),吸取 NAS 与 SAN 的优势特点[3],基于两者的融合架构,采用带外(Out-of-Band)数据传输模式(数据通道与控制通道分离,文件数据直接应用节点和存储节点之间传输,文件元数据经由元数据服务器节点传输处理),消除了传统带内模式(文件数据与文件元数据通过相同的通道在客户节点与服务器节点之间传输处理)的数据传输瓶颈,并且有效解决了传统 SAN 设备的数据共享问题(包括同构平台间共享和异构平台间共享)。

如图 1.1 所示,蓝鲸分布式文件系统的整体架构可以划分为应用服务器机群、元数据服务器机群[4]、存储节点机群,绑定服务器和管理服务器等几个部分,其中绑定服务器用于处理元数据的动态映射及负载均衡[5][6],管理服务器提供系统整体监控与管理。系统中各节点间彼此独立,通过高速交换网络互联在一起。蓝鲸分布式文件系统支持系统的在线扩展,用户可以根据系统的实际使用情况(应用模式、应用规模、数据容量、系统负载等)在线独立地扩展应用服务器机群、元数据服务器机群、存储节点机群,对

系统进行整体均衡。

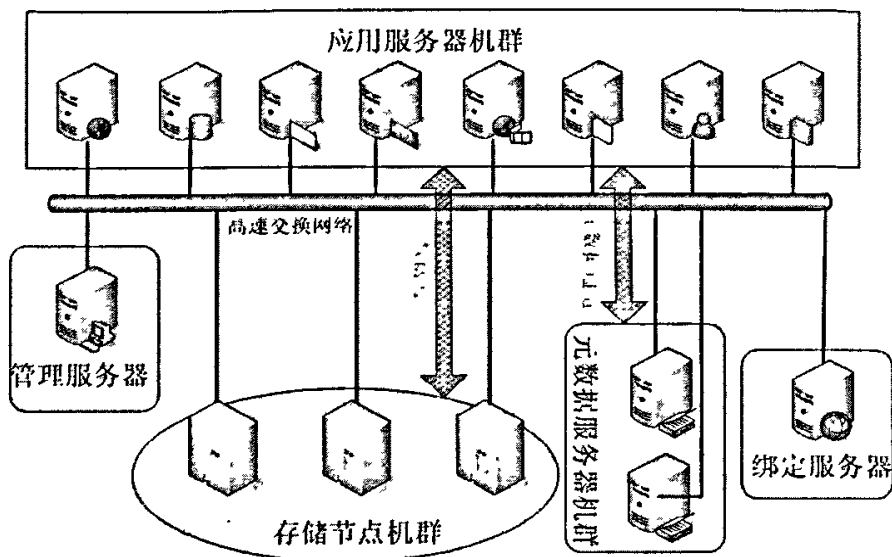


图 1.1 BWFS 系统架构示意图

1.2 相关领域研究

针对普遍关心的数据一致性语义，各种分布式文件系统有着不尽相同的解决方案[7][8]。在文件系统数据一致性语义方面存在如下几种主要类型[9][10]：

✧ UNIX 语义(UNIX Semantics)

这种语义源于 UNIX 的本地文件系统，基于本地时钟（物理的或逻辑的），任何互斥操作之间都有一个绝对的时间顺序（可能需要借助某些同步/互斥机制），操作结果由这个确定的操作顺序唯一决定。该语义规定：任何读写操作都应该基于之前最近的针对该共享文件写操作的结果。如果忽略网络延迟所带来的影响，上述 UNIX 语义可以推广到分布式文件系统中。

✧ 会话语义(Session Semantics)

会话语义规定对一个打开的共享文件所进行的修改操作只对操作进程（或客户节点）本身可见，当该共享文件被关闭之后其修改结果才对其他进程（或客户节点）可见。

✧ 类 NFS 语义(NFS Analogy Semantics)

会话语义面临这样的问题：当同一个共享文件被多个进程（或客户节点）同时独立打开并且修改，然后再各自独立的关闭时，则该文件的最终内容应该如何确定？——不论使用哪个进程（或客户节点）的修改结果都可能导致其他进程（或客户节点）修改结果的非交叠（或称为伪交叠）缺失。

类 NFS 语义对会话语义进行加强，该语义下对共享文件的修改操作与该操作结果对全局可见之间的时间间隔不能超过某个限定值（暂称为间隔时间），间

隔时间限定了节点 cache 的有效时间以及脏数据写回周期。通过修改该间隔时间,可以调整类 NFS 语义的强弱,其上限是零间隔时间,语义近似于 UNIX 语义,其下限是关闭时间间隔,语义等价于会话语义。

✧ 不可修改文件语义(Immutable Semantics)

不可修改文件语义规定文件一旦创建便不可修改,针对共享文件的修改操作将导致创建该文件的一个新版本,该文件的每个版本都被作为一个完整的文件对待。

✧ 事务语义(Transaction Semantics)

事务语义通过事务处理机制控制对共享文件的并发访问,隶属于相同事务的各个文件操作要么全都执行,要么全都不执行,其中间的状态结果对外不可见,以保证对共享文件操作的原子性和可串行性。

1.2.1 AFS

AFS[11][12][13][14](ANDREW File System)是 ANDREW 项目的产物,该项目于 1983 年在 CMU 由 IBM 资助发起,旨在建立一种校园范围(支持几千个工作站节点)的分布式共享环境。

AFS 采用 C/S 模式,实现了典型的会话语义。客户节点进程打开服务器节点共享文件的请求被客户节点的 ANDREW 缓存管理器发送到服务器节点,在通过必要的权限验证之后,该文件被拷贝至客户节点的本地存储,所有针对该文件的后继读写操作都基于文件本地副本;当该文件被关闭时,修改过的文件本地副本被写回服务器节点。服务器节点在处理打开权限验证时并不做多客户节点的共享检查,客户节点也不关心彼此之间的 cache 一致性。如果多于一个客户节点写操作打开同一个共享文件,则最后关闭该文件的客户节点的文件本地副本成为服务器节点上对应共享文件的最新版本。

1.2.2 NFS

网络文件系统[15][16][17][18](Network File System, NFS)最初由 Sun Microsystems, Inc.于 1985 年在 Solaris 系统上设计实现,目前已被广泛移植到几乎所有的操作系统上(包括诸多非类 UNIX 系统,如 OS/2、MS-DOS 等)。NFS 已为业界普遍接受,其协议标准现已更新到 NFSv4 版本。

NFS 采用 C/S 模式,客户节点与服务器节点之间通过 RPC 机制进行通讯。在 NFSv4 之前的版本中,NFS 实现为典型的无状态(stateless)类 NFS 语义。客户节点使用本地高速缓存以提高性能,服务器节点不向客户节点提供类似于本地文件系统的 OPEN 和 CLOSE 操作接口,不保存任何关于客户节点打开文件的状态信息(除非使用子协议 NLM[19]——Network Lock Manager 的显式锁机制)。这种无状态语义简化了系统的故障恢复(文件锁的恢复需要借助子协议 NLM 和 NSM[20]——Network Status Monitor 的支

持),但却降低了对客户节点 cache 有效性的控制,客户节点只能依靠超时机制主动检测以降低(不能从根本上消除)cache 一致性问题所带来的风险。在基于 Linux kernel-2.4.18-14 的实现中,一个 NFSv3 客户节点对普通文件的数据修改(是 WRITE-BEHIND 的)最迟可在 6 秒钟之后对另一个共享访问该文件 NFSv3 客户节点可见。

NFSv4 较 NFSv3 有较大的语义调整,其自身包含对文件锁的支持,无须再借助 NLM,文件锁的引入使得 NFSv4 变成有状态(stateful)协议。另一方面, NFSv4 在客户节点的 cache 管理上也做了较大的改动,在 NFSv3 的超时检测基础上引入打开授权机制:客户节点执行打开访问共享文件操作的时候检测本地是否拥有相关授权(read or write),若有则直接打开而不必与服务器节点交互,之后的相关操作也可以基于本地 cache 进行,而不必担心其他客户节点操作对本节点 cache 的影响;如果本地没有相关授权,则由服务器节点根据目标文件当前的共享状态及申请客户节点是否支持 callback 机制等因素决定是否产生新的授权;如果新的授权申请与已有授权发生冲突(读授权之间是相容的,写授权是排它的),则服务器节点通过某种 callback 机制对先前的授权进行 revoke;如果服务器不允许客户节点的授权,则客户节点将继续使用与 NFSv3 类似的超时检测机制来降低 cache 一致性问题带来的风险。授权机制的引入降低了客户节点与服务器节点之间的交互频率,有利于性能的提高,但却没有真正解决 cache 一致性问题,因此 NFSv4 实现的是一种改进的有状态的类 NFS 语义。

此外, NFSv4 还引入某种 lease 机制,该机制用于文件锁的有效性管理而不是客户节点间 cache 的一致性(这一点不同于后面将要提到的 ECHO 分布式文件系统 lease 机制)维护。超过 lease 时效期的客户节点所持有的文件锁被服务器节点视为无效而释放(lease 的时效期可以通过其他操作隐式更新以降低更新开销),这样就简化了系统故障恢复。

1.2.3 CODA

CODA[21][22][23]系统于 1987 年在 CMU 以 AFS-2 为原型开发而来,是一个容错的分布式文件系统。它继承 AFS 的会话语义,并对其进行扩展:在系统正常使用情况下, CODA 系统的客户节点会根据使用情况自动地把服务器节点上的共享文件尽可能多的保存为本地副本,以允许客户节点在与服务器节点无法通讯(如网络分割、服务器故障等)的情况下使用本地存储的文件副本进行工作;当通讯恢复之后,更新过的文件本地副本会自动写回服务器节点,如果出现写回冲突(针对相同共享文件的不同修改副本), CODA 系统将提示用户选择保留哪一个副本。

1.2.4 SPRITE

SPRITE[24][25]是 20 世纪 80 年代中期由 University of California at Berkeley 开发的

一个网络操作系统，其中包含分布式文件系统部分。

SPRITE 系统大量使用高速缓存（包括客户节点和服务器节点），针对写操作采用 WRITE-BEHIND 策略，用以提高性能。客户节点每 30 秒自动将最近 30 秒内没有修改过的脏数据写回服务器节点的高速缓存，每个修改最迟在 60 秒内写回服务器节点的高速缓存；服务器节点的高速缓存每 60 秒自动向持久存储刷新一次。这种策略提高了写操作的聚合度，降低了网络开销，但由于客户节点的修改最迟在 120 秒之后才能真正写到服务器节点的持久存储上，如果其间系统崩溃，那么 120 秒之内的修改可能全部丢失。

SPRITE 提供 UNIX 语义支持，不同于 NFSv3，它采用 stateful 的服务器模式，服务器节点保存所有客户节点对共享文件的操作状态信息，如果发生写冲突（多于一个客户节点打开相同的共享文件，其中包含写模式打开），那么相关的 cache 被禁止，所有读写操作直接与服务器节点交互，这样就保证了 UNIX 语义。SPRITE 的这种有状态服务器模式是以复杂的故障恢复为代价的。

1.2.5 ECHO

ECHO[26][27]分布式文件系统源于 1988 年 DEC Palo Alto 研究中心发起的一个大型分布式系统项目，是第一个使用 lease[28]机制的分布式文件系统。ECHO 系统采用 C/S 模式，客户节点使用本地高速缓存，并采用 WRITE-BEHIND 策略（包括目录文件在内），用以提高性能，同时使用 lease 机制维护 cache 的一致性。

ECHO 系统的 lease 是一种具有时效性的可剥夺的文件级粒度锁。当一个客户节点第一次读操作一个共享文件时，先向服务器节点申请该共享文件的读权限 lease，如果服务器允许该 lease，那么在该 lease 的时效期内该客户节点针对该共享文件的读操作可以使用本地高速缓存中的内容。相应的写操作需获得写权限 lease。读权限的 lease 之间是相容的，写权限 lease 是排它的。当欲申请的 lease 与其他节点已授权的 lease 发生权限冲突时，服务器通过一种 callback 机制取消之前的授权。借助 lease 机制 ECHO 系统既保证了很好的性能，又实现了 UNIX 语义支持，同时 lease 机制简化了系统的故障恢复，超过时效期的 lease 将被自动清除。其缺点是针对多客户节点共享写操作相同文件时彼此 revoke 所带来的抖动问题。

1.2.6 Storage Tank

Storage Tank[29][30]（简称 ST）是 IBM 公司在上个世纪 90 年代后期开发的分布式文件系统，该系统采用与蓝鲸分布式文件系统类似的数据通道与控制通道分离的模式：元数据处理通过元数据服务器节点进行（使用 IBM 自己开发的协议——IBM Storage Tank Protocol），数据读写直接访问存储节点。

ST 分布式文件系统提供复杂的锁机制，用于提高系统性能，维护客户节点 cache 一致性，提供 UNIX 语义支持。基于缓存数据和元数据的思想，ST 分布式文件系统引入

semi-preempted lock 机制——即使共享文件已被关闭（本客户节点无进程打开该共享文件），隶属于该共享文件的某些锁依然可以缓存在本节点，以便优化后继的打开操作，其流程如下：

- 1) 当客户节点准备打开一个 ST 系统的共享文件时，先在本地检测是否持有该共享文件相关权限（打开模式相关）的 session lock，如果有就不必与服务器节点交互，否则向服务器节点发出请求；
- 2) 服务器节点收到该请求，检测是否已经存在与该打开请求相冲突的 session lock 被其他客户节点所持有，若无冲突则允许请求，并返回相关信息，否则向相关冲突客户节点发出询问；
- 3) 相关客户节点收到服务器节点发来的询问，检测该 session lock 是否在用，若未用则释放该 session lock，否则给出拒绝信息；
- 4) 服务器节点根据询问结果答复申请的客户节点。

另一方面，ST 分布式文件系统引入 lease 机制，用于处理网络分割或节点失效等故障。元数据服务器节点为每个客户节点维护一个 lease，lease 对应一定的时效期，在 lease 的时效期内，客户节点可以通过与元数据服务器节点之间的交互隐式地或显式地更新 lease 的时效期，如果 lease 在其有时效之内不能被更新，那么元数据服务器节点将假定网络或客户节点发生故障，在一段指定的时间之后与该 lease 关联的客户节点的 lock 等信息将被清除；相应的客户节点发现 lease 超时之后，须把本地 cache 中的脏数据写回存储节点，在没有获取新的 lease 之前不能使用本地 cache。

1.2.7 Lustre

Cluster File System 公司开发的 Lustre[31][32]分布式文件系统使用“基于对象的存储设备” (Object based Storage Devices, OSD)[33]，以期获得更好的性能、安全性和可管理性。这种基于对象的存储设备，可以是软件模拟的，也可以是硬件直接支持的，具有较大的灵活性。

图 1.2 是取自 Lustre 白皮书中的系统结构示意图，其整体结构包括客户节点、元数据服务器节点和对象存储三部分。Lustre 的元数据服务器节点负责处理身份认证、创建、删除以及其他一些目录操作，文件数据块位置映射工作交由对象存储完成（该工作在蓝鲸分布式文件系统中由元数据服务器节点负责）。此外 Lustre 的对象存储还负责处理文件锁、维护系统数据一致性，并通过 VAX DLM 实现对 UNIX 语义的支持。

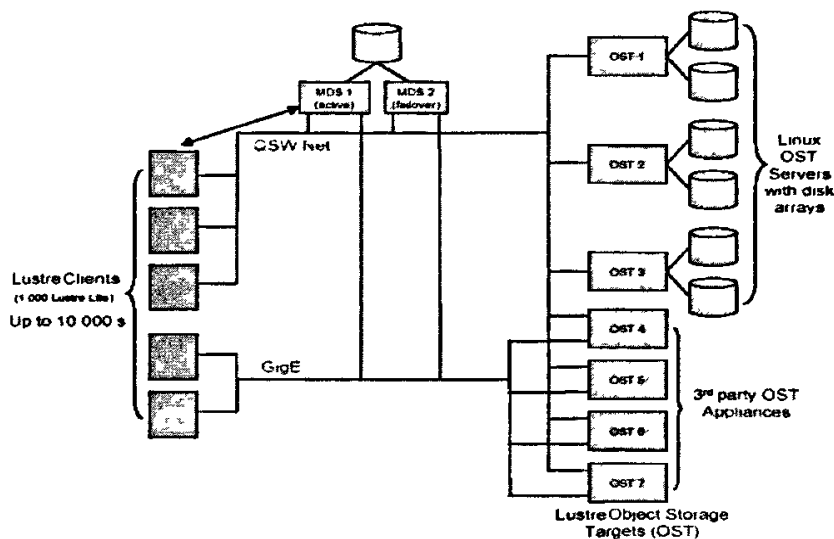


图 1.2 Lustr 系统结构示意图

1.2.8 小结

尽管各分布式文件系统的设计出发点及侧重点有所不同，其所实现的数据一致性语义差别较大，但其所采用的某些机制通用性较强，在一定程度上是可以相互借鉴的：

- ✧ 本地高速缓存提高读写性能
- ✧ WRITE-BEHIND 策略提高写操作聚合度，降低网络传输的协议开销
- ✧ 授权机制控制并发访问，维护客户节点 cache 一致性
- ✧ 时效机制处理系统故障（如节点宕机、网络分割等）恢复问题

1.3 论文的动机

从广义的角度看，分布式文件系统的数据一致性包括数据安全性（有别于涉密数据的受限访问）、数据完整性、cache 一致性、操作原子性、资源一致性等诸多方面。本论文并非就上述所有数据一致性问题进行研究，而是结合蓝鲸分布式文件系统从系统前端节点的应用兼容性角度研究相关的数据一致性语义问题，包括操作原子性、资源一致性等在内的系统后端相关问题不在本论文的研究之列。

关于分布式文件系统应该提供何种数据一致性语义的问题，人们往往认为系统提供的数据一致性语义越严格越好，实际上可能并非如此。数据一致性的维护是有代价的，越是严格的数据一致性其维护的代价越高。不同的应用模式对数据一致性的要求不尽相同，为那些并不需要很强数据一致性的应用维护高级别的数据一致性是不必要的，并且可能造成并发度降低等损失；另一方面，某些应用为提高平台兼容性，降低对底层系统数据一致性支持的依赖，应用本身通常提供适当的并发访问控制，为这种应用维护高级别的数据一致性不仅不必要，而且可能对应用原有的并发访问控制造成负面影响。基于

上述考虑,蓝鲸分布式文件系统应该提供何种数据一致性语义才能更好地兼容各种不同模式的分布式应用是本论文的研究重点。

本论文的研究工作基于单个元数据服务器节点环境完成,元数据服务器机群环境下的系统数据一致性语义有待进一步研究。

本论文的研究意义在于:

◆ 学术发展角度

- 1) 丰富对分布式文件系统数据一致性语义的研究
- 2) 明晰蓝鲸分布式文件系统的数据一致性语义

◆ 系统实用角度

- 1) 加强蓝鲸分布式文件系统的正确性与可用性
- 2) 提高蓝鲸分布式文件系统的应用兼容性

1.4 论文的贡献

本论文在借鉴已有分布式文件系统相关研究成果的基础之上,在调研当前的及潜在的分布式应用模式之后,对蓝鲸分布式文件系统的数据一致性语义及相关问题进行了较为深入地研究,做出如下主要贡献:

1) 提出并实现基于软件的网络容错技术

节点间数据传输的正确性及完整性是蓝鲸分布式文件系统可用性的重要体现,也是其数据一致性的基础。针对蓝鲸分布式文件系统中影响较大的通道连接软故障中断异常采用独立于系统硬件支持的连接复制、通道切换、请求重构等软件技术加以容错,实现应用层透明的数据无损传输,不仅提高了系统的可用性,也使得蓝鲸分布式文件系统的数据一致性有所保障。

2) 设计并实现自适应的带外模式文件属性更新机制

文件属性更新是蓝鲸分布式文件系统数据一致性的前提。通过对蓝鲸分布式文件系统架构和数据缓存策略的分析,设计并实现了一种自适应的带外模式文件属性更新机制。该机制结合机会更新、被动更新、周期更新等多种文件属性更新方式,能够根据当前的网络状态自适应地调整文件属性的更新周期,降低数据丢失风险,提高网络效率,允许用户动态地设置更新周期基数及更新周期调整幅度。所有这些调整都是应用层透明的,充分体现了蓝鲸分布式文件系统的灵活性。

3) 设计并实现基于授权机制的在线可调整的数据一致性语义模型

分布式文件系统的数据一致性语义是决定分布式应用模式的关键所在,是本论文的研究重点。针对蓝鲸分布式文件系统的架构特点,设计并实现了一种支持数据一致性语义在线调整的数据一致性语义模型,该模型通过授权(一种可剥夺的文件级粒度的多态文件锁)机制,允许用户在线调整蓝鲸分布式文件

系统的数据一致性语义,以满足不同应用模式的需求。该模型提供超时一致性、释放一致性、写一致性、读写一致性等四种数据一致性语义,支持从类 NFS 语义到类 UNIX 语义等多种数据一致性语义的兼容性。

◇ 超时一致性

通过类似于 NFSv3 系统的 cache 超时机制主动检测文件更新以降低应用节点间 cache 一致性问题所带来的风险,为提高系统对基于 NFSv3 系统应用的兼容性,其超时设置尽量与 NFSv3 系统保持一致。该语义下对共享文件的修改操作与该操作结果对全局可见之间的时间间隔取决于系统的文件属性更新策略。

◇ 释放一致性

在超时一致性的基础上加强对释放操作的并发访问控制,确保释放操作不会将正在被其他应用节点读写操作共享的数据块释放,实现蓝鲸分布式文件系统的数据安全性支持。

◇ 写一致性

在释放一致性的基础之上对多个应用节点并发更新共享文件进行控制,消除多个应用节点并发写操作区域伪交叠所导致的更新数据非正常缺失问题,实现蓝鲸分布式文件系统的数据完整性支持。该语义完全兼容 NFSv3 语义。

◇ 读写一致性

在写一致性的基础之上进一步加强对 cache 一致性的支持,确保应用节点读操作得到的数据是其他应用节点针对该文件的最新属性更新的同步数据,实现应用节点间的 cache 一致性支持。

1.5 论文的组织

论文第一章首先分析传统存储模式的缺陷,阐述数据一致性语义对分布式应用模式的重要意义;然后归纳数据一致性语义的主要类型,概要地介绍了若干有代表性意义的分布式文件系统针对相关问题的解决方案;在此基础之上指出本论文的研究动机及主要贡献。

第二章从描述数据一致性问题入手,分析蓝鲸分布式文件系统的数据一致性问题,并对分布式应用模式进行汇总分类。

第三章介绍节点间数据传输保障,它是蓝鲸分布式文件系统数据一致性的基础,描述蓝鲸分布式文件系统基于软件的网络容错技术。

第四章介绍文件属性更新,它是蓝鲸分布式文件系统数据一致性的前提,描述蓝鲸分布式文件系统自适应的带外模式文件属性更新机制。

第五章是全文重点,详细描述蓝鲸分布式文件系统基于授权机制的在线可调整的数

据一致性语义模型。

最后一章对通篇论文进行总结，并给出未来的研究方向。

第二章 数据一致性与应用模式

分布式文件系统的数据一致性语义是决定分布式应用模式的关键所在，分析数据一致性问题，了解不同类型分布式应用模式对数据一致性的需求对研究蓝鲸分布式文件系统数据一致性语义大有裨益。

本章从数据一致性问题入手，描述数据安全性、数据完整性、cache 一致性问题；然后结合蓝鲸分布式文件系统自身特点，分析其所面临的数据一致性问题；接下来对分布式环境下的应用模式进行汇总分类，分析各种类型应用模式对数据一致性的需求。

2.1 数据一致性问题描述

从广义的角度看，分布式文件系统的数据一致性包括数据安全性（有别于涉密数据的受限访问）、数据完整性、cache 一致性、操作原子性、资源一致性等诸多方面。前三者对分布式应用模式影响重大，是本章需要重点描述的部分，包括操作原子性、资源一致性等在内的问题通常由日志机制[34][35]加以保障，这里不作过多讨论。

2.1.1 数据安全性

数据安全性[36]通常包含两种含义：一种含义是涉密数据的受限访问，只有持特定权限的用户或进程才能访问特定的文件或数据，其他情况下对文件或数据的访问是非法的；另一种含义是记录到文件系统中的数据在其被合法修改或相应存储空间被释放之前是可访问的，所谓合法修改是指通过对相应文件的相应位置的修改操作，任何其他修改方式或空间释放操作导致的数据不可访问都是非法的。本论文中提及的数据安全性主要针对后者。数据安全性是分布式文件系统数据一致性的基本要求，也是分布式文件系统正确性的基本条件。

2.1.2 数据完整性

数据完整性[37]指应用节点的数据更新如果没有被其他应用节点针对相同文件相同位置的数据更新所覆盖，则该更新结果最终应该记录在相应文件的特定位置处。为保证数据完整性需要从两个层次入手：第一个层次是节点间的数据传输保障，使得非共享模式下的数据更新可以无损的更新到服务器节点；第二个层次是多个应用节点共享更新的并发控制，确保某个应用节点的更新结果不会被其他应用节点针对不同位置更新结果覆盖。数据完整性是以数据安全性为基础的，从这个意义上讲，数据安全性是数据完整性的特例。

2.1.3 cache 一致性

cache 一致性[38]指各应用节点本地 cache 中的数据保持一致,这种一致可以是同步的,也可以是异步的(保证应用节点真正使用的数据不是过时的数据)。通常有两种方式保持 cache 一致性:一种是某个应用节点本地 cache 中的数据更新会同步导致其他应用节点本地 cache 中数据的相应更新,这种方式的更新代价比较大;另一种方式是某个应用节点对本地 cache 中的数据更新会同步导致其他应用节点本地 cache 中相应数据失效,当其他应用节点需要再次使用相应数据时重新获取更新后的数据,这种按需更新的方式可以避免再次使用之前的多次不必要的 cache 数据更新,代价相对较低,是目前分布式文件系统普遍使用的方式。

cache 一致性是一种比较接近 UNIX 语义标准的数据一致性要求,维护的代价很高,因此有些分布式文件系统采用弱 cache 一致性支持。所谓弱 cache 一致性指不强制某个应用节点数据更新一定能被其他应用节点的相应操作及时获取,允许数据更新操作与更新结果被其他应用节点获取之间存在一定的时间间隔,在该间隔时间内,其他应用节点可能获取过时的数据。NFSv3 系统提供典型的弱 cache 一致性支持。

2.1.4 小结

尽管上述三个方面各有侧重,但其数据一致性的语义强度是逐渐提升的,数据安全性是数据完整性的基础,数据完整性是 cache 一致性的保障之一。

2.2 蓝鲸分布式文件系统的数据一致性问题

为消除传统分布式文件系统单纯客户/服务器模式(如 NFSv3 系统)下的数据传输瓶颈,蓝鲸分布式文件系统对数据通道与控制通道进行分离,这种带外模式给系统的数据一致性带来新的问题;另一方面,为提高系统对底层网络硬件环境(IP 网络、FC 网络[39]、InfiniBand 网络[40][41]等)支持的兼容性,蓝鲸分布式文件系统的应用节点与存储节点间的数据传输按块级(block)粒度进行,这种底层块级粒度传输与上层字节级(byte)粒度文件操作之间的差异也对系统的数据一致性造成影响。本节将针对蓝鲸分布式文件系统的数据一致性问题进行具体分析,图 2.1 给出蓝鲸分布式文件系统数据一致性问题的整体描述:

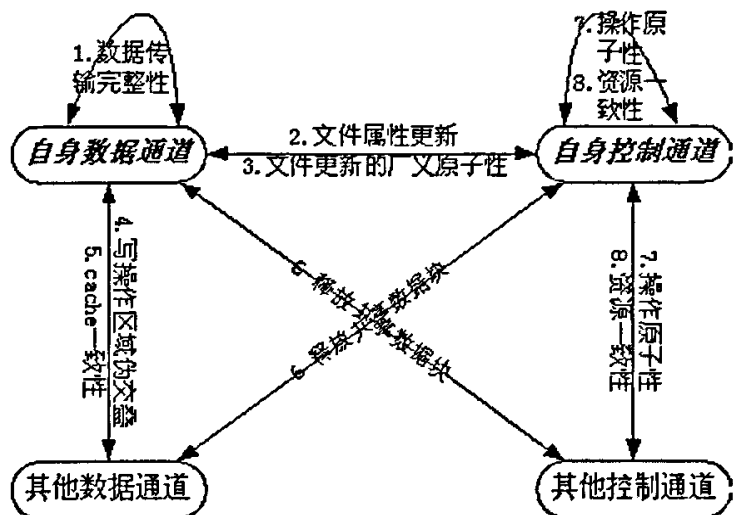


图 2.1 BWFS 数据一致性问题描述图

从图中可以看出，蓝鲸分布式文件系统的数据一致性问题可以从下述几个角度分析：

1) 自身数据通道的数据一致性问题

主要指节点间数据传输的正确性及完整性，这是蓝鲸分布式文件系统数据一致性的基础，缺少这一点保证，蓝鲸分布式文件系统的数据一致性将无从谈起。论文第三章对此进行专门论述。

2) 自身数据通道与自身控制通道之间的数据一致性问题

这是采用数据通道与控制通道分离机制所特有的，包括文件属性更新和文件更新的广义原子性两个方面。文件属性更新是蓝鲸分布式文件系统数据一致性的前提，它保证文件数据更新与文件元数据更新的逻辑关系，论文第四章专门讨论蓝鲸分布式文件的文件属性更新机制。文件更新的广义原子性涉及文件数据更新与文件元数据更新的事务性问题，用于解决文件更新过程中的宕机问题，需要适当的日志机制加以支持，论文并不对此作深入探讨，留待今后进一步研究。

3) 自身数据通道与其他数据通道之间的数据一致性通道

指不同应用节点的数据通道之间的数据一致性问题，包括写操作区域伪交叠问题和 cache 一致性问题，详见下文描述。

4) 自身数据通道与其他控制通道之间的数据一致性问题

指本节点数据通道与其他应用节点控制通道之间的数据一致性问题，主要针对释放共享数据块问题，对此下文有具体描述。

5) 自身控制通道的数据一致性问题

自身控制通道的数据一致性问题主要是蓝鲸分布式文件系统后端的操作原子性和资源一致性问题，本论文并不对此进行讨论。

6) 自身控制通道与其他控制通道之间的数据一致性问题

在单个元数据服务器节点情况下等效为自身控制通道的数据一致性问题。

7) 自身控制通道与其他数据通道之间的数据一致性问题

这是自身数据通道与其他控制通道之间的数据一致性问题的等价描述。

至于其他数据通道内部、其他数据通道之间、其他控制通道内部、其他控制通道之间以及其他数据通道与其他控制通道之间的数据一致性问题等都可以等效为上述某个角度的数据一致性问题。除了后文章节将要详细论述的节点间数据传输的正确性及完整性、文件属性更新机制外，本节主要讨论释放共享数据块问题、写操作区域伪交叠问题和 cache 一致性问题，如果未作特殊说明，下文有关蓝鲸分布式文件系统的数据一致性问题代指这些问题。

2.2.1 释放共享数据块问题

在蓝鲸分布式文件系统中，对文件占用数据块的释放操作由元数据服务器节点控制，数据块释放之后可以被再次分配给其他文件使用。在多个应用节点对共享文件进行并发操作的情况下，如果一些应用节点对目标文件进行数据块释放操作（如 truncate 操作、unlink 操作等），另一些应用节点对目标文件的相同区域进行读写操作，那么系统需要提供适当的并发访问控制，避免释放正被读写共享的数据块，否则可能产生如下问题：

◇ 释放被读共享的数据块

如果将一个正在被其他应用节点读共享的数据块释放，那么当读共享节点再次读取该数据块时可能得到无效数据；如果该数据块被再次分配，那么可能导致涉密数据的非法访问。该问题破坏了分布式文件系统的 cache 一致性。

◇ 释放被写共享的数据块

如果将一个正在被其他应用节点写共享的数据块释放，那么当写共享节点再次写该数据块时会把数据写到一个无效的存储空间中，这部分数据无法通过正常途径访问；如果该数据块被再次分配，那么写操作会将该块中的有效数据破坏，相关数据将变得不可访问。该问题破坏了分布式文件系统的数据安全性。

上述问题可以从另外一个角度考虑：按照 UNIX 语义，删除一个已被打开的文件不会导致该文件占用数据块的真正释放，基于已经打开文件的读写操作可以继续进行，直到最后的引用被关闭时该文件才真正被删除（占用的数据块随之被释放），采用 stateless 服务器的系统不能有效的兼容此种模式的应用。受数据通道与控制通道分离的带外模式的影响，蓝鲸分布式文件系统将释放共享数据块的问题进一步扩大：在传统的带内模式下（如 NFSv3 系统），释放共享数据块虽然可能破坏节点间的 cache 一致性，但不会造成其他有效数据的损失；而在蓝鲸分布式文件系统中则可能导致整个文件系统的损坏。

2.2.2 写操作区域伪交叠问题

蓝鲸分布式文件系统的底层数据传输基于块级粒度进行，块级粒度数据传输的优势在于对底层网络硬件环境的兼容性比较好，同时相对于字节级粒度更容易进行操作合

并，提高操作的聚合度。但块级粒度操作也带来了字节级粒度所没有的数据一致性问题——写操作区域伪交叠问题，如图 2.2 所示：

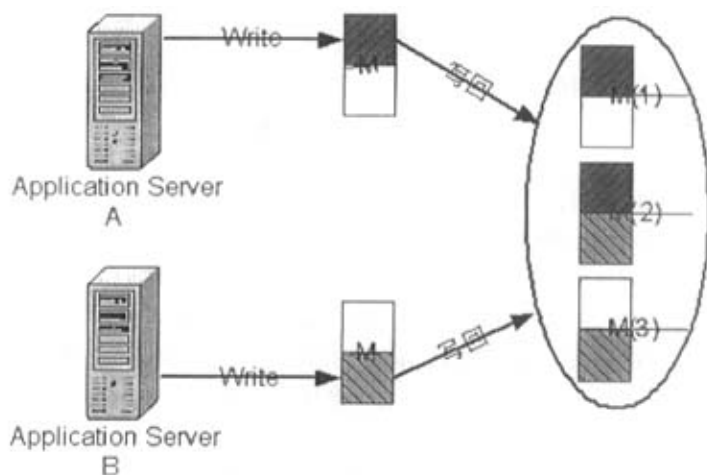


图 2.2 写操作区域伪交叠示意图

如上图所示，应用节点 A 与应用节点 B 同时对数据块 M 进行写操作，A 节点操作 M 块的前半部分，B 节点操作 M 块的后半部分，两者操作区域在字节级粒度上没有交叠，但表现在块级粒度上却出现交叠，这里称为伪交叠，在没有额外并发访问控制的情况下可能发生一个应用节点的写操作结果被另一个应用节点的非写操作结果（原始的过时数据）覆盖，如 M(1)或 M(3)，从而破坏文件蓝鲸分布式文件系统的数据完整性（M(2)是正确操作的结果）。此种模式的应用在 NFSv3 系统上运行不会发生上述问题。

2.2.3 cache 一致性问题

蓝鲸分布式文件系统的 cache 一致性问题是基于本地缓存策略的分布式文件系统所共有的问题，如何保证某个应用节点的数据更新结果及时为其他应用节点可用，避免其他应用节点使用本地 cache 中的过时数据是 cache 一致性所有解决的首要问题。

2.2.4 小结

上述几方面的问题是蓝鲸分布式文件系统数据一致性所面临的主要问题，其问题的严重性是不一样的，对分布式应用模式的影响也各不相同，蓝鲸分布式文件系统并非想通过一种手段解决所有的数据数据一致性问题，而是想实现一种能够将几个问题分解的数据一致性解决方案以满足不同的应用需求。

2.3 文件共享与应用模式

随着计算机系统结构的发展，应用模式也在发生变化，从最初的单机单处理器计算

到目前普遍的多机网络并行处理,应用模式正在从单一化的集中模式向多元化的分布模式发展。各种不同的应用模式对文件的共享访问差别很大,这里对之进行汇总分类,并简要分析各种类型应用模式对数据一致性的需求。

2.3.1 模式一:无共享模式

从数据一致性的角度看,无共享模式是一种最简单的应用模式。该模式下,各应用节点独自处理不同的文件,彼此之间没有文件共享访问,因此不必考虑多个应用节点之间的数据一致性问题。此种模式应用所需的数据一致性支持可以通过应用节点本地操作系统提供的相关机制(如 Linux 环境下 VFS[42]对数据一致性的支持、Windows 环境下 IFS 对数据一致性的支持等)加以保证,蓝鲸分布式文件系统无须为该模式应用提供额外的分布式环境下的数据一致性支持。单机应用可以看作这种模式应用的一个特例。

采用任务派发方式工作的应用具有该模式特征:每个文件都关联特定的任务,在任务被派发之前,文件不隶属于任何应用节点,应用会根据特定的规则(如节点空闲情况)将任务派发给某个应用节点进行处理,每个任务只能派发给一个应用节点,因此相应文件就只能被这个特定的应用节点处理,不能被其他应用节点共享访问。另外那些只对私有文件进行处理的应用(如邮件处理)也属于此种模式。

2.3.2 模式二:多点读共享模式

该模式下,对共享文件的访问仅限于读操作,允许多个应用节点并发读操作访问同一个文件,由于没有对数据的更新操作,应用节点不必考虑多个应用节点之间的数据一致性问题。从数据一致性的角度看,该模式与前面的无共享模式是一致的,蓝鲸分布式文件系统无须为该模式应用提供额外的分布式环境下的数据一致性支持;其差别在于该模式下文件是可以共享访问的,但只限于读操作,无共享模式下文件是非共享的,但读写等操作都可以。

视频点播服务(Video On Demand, VOD)[43]应用是一种典型的多点读共享模式应用(这里暂不考虑视频文件的制作过程),系统允许多个用户同时点播相同的影视文件。其他并发处理只读文件的应用也具有类似特性。

2.3.3 模式三:单点写多点读共享模式

该模式下,系统允许多个应用节点并发读操作访问同一个共享文件,并且允许最多一个应用节点对该共享文件进行并发写操作访问。写操作节点的引入使得共享文件可能发生数据更新,因此存在读写应用节点之间的 cache 一致性问题;由于最多只有一个应用节点对共享文件进行写操作,所以不必考虑多个应用节点对共享文件进行并发更新的情况,不存在类似于写操作区域伪交叠问题等的的数据完整性问题。从数据一致性语义的角度看,如果应用需要 cache 一致性支持,则蓝鲸分布式文件系统需要提供适当的机制

以保证读写应用节点之间的 cache 一致性, 否则蓝鲸分布式文件系统可以仅提供弱的 cache 一致性支持或者不提供额外的分布式环境下的数据一致性支持。

某些单点更新的信息查询系统具有该模式特征。

2.3.4 模式四：多点写共享模式

对于写操作而言可能包含必要的读操作（如针对不完整数据块的写操作需要先对该数据块进行读操作），因此多点写共享模式下是否包含读共享节点并不重要。该模式下，系统中存在多个应用节点对同一个共享文件进行并发的写操作，对于蓝鲸分布式文件系统而言，首要问题是以写操作区域伪交叠问题为代表的完整性问题，其他的关于应用节点之间的 cache 一致性问题同前述的单点写多点读共享模式。

Paradigm EPOS[44]软件是用于石油天然气勘探地质资料处理方面的专业应用工具软件，该软件运行在并行处理环境下，允许多个应用节点并发更新某些共享文件，是多点写共享模式应用的一个实例。

2.3.5 模式五：多点读与释放操作共享模式

该模式在前述的多点读共享模式的基础之上加入释放操作的共享节点，其数据一致性问题主要体现为释放读共享数据块问题，对系统的 cache 一致性有影响。如果应用需要 cache 一致性支持，则蓝鲸分布式文件系统需要提供适当的机制以保证读操作不会获取无效数据，否则蓝鲸分布式文件系统可以仅提供弱的 cache 一致性支持或不提供额外的分布式环境下的数据一致性支持。

此种模式的实际应用并不常见。

2.3.6 模式六：单点写多点读与释放操作共享模式

该模式下，系统允许多个应用节点针对同一个共享文件进行并发地读操作访问，允许最多一个应用节点对该共享文件进行并发地写操作访问，同时允许对该共享文件进行释放操作。对于蓝鲸分布式文件系统而言，允许对写共享文件进行释放操作可能带来以释放写共享数据块为代表的数据库安全性问题，这是涉及蓝鲸分布式文件系统正确性的问题，需要加以重点防范；对其他相关的数据一致性问题的处理同前述的单点写多点读共享模式。

FTP 应用是单点写多点读与释放操作共享模式的典型代表，任何时候最多有一个应用节点对共享文件进行写操作（上传文件），允许多个应用节点（通常 FTP 服务器限制同时在线的用户数量）并发读操作访问（下载文件）该共享文件，同时适当授权的用户可以对该共享文件进行释放操作。

2.3.7 模式七：多点写操作与释放操作共享模式

该模式在前述的多点写共享模式的基础之上进一步允许针对目标共享文件的释放操作。对于蓝鲸分布式文件系统而言，此种模式的应用可能引发前述的包括释放共享数据块问题、写操作区域伪交叠问题、cache 一致性问题等在内的蓝鲸分布式文件系统的所有数据一致性问题，是对蓝鲸分布式文件系统数据一致性维护机制的重大考验。

由于此种模式应用对数据一致性的要求比较高，为了提高应用的平台兼容性，应用本身通常提供适当的并发访问控制，用以降低对底层系统数据一致性支持的依赖。数据库系统具有明显的这种特征。

2.3.8 小结

上述关于分布式应用模式的划分基于应用的数据共享模式及操作类型。实际上，归属于上述相同应用模式的不同应用的操作方式可能会有很大差别，如针对共享文件的相同区域进行操作还是不同区域进行操作，读写操作粒度是数据块边界对齐还是数据块边界不对齐等，这些对分布式文件系统数据一致性要求都会有所不同。因此如果采取更加精细的划分标准，得到的应用模式会更加丰富，限于篇幅原因，这里不作更多讨论。

在上述七种分布式应用模式中，无共享模式是一种松散耦合的特殊应用模式，尽管系统中的数据可以共享，但应用节点之间没有进行并发共享访问，因此也没有额外的数据一致性需求；其余六种模式是存在数据并发共享访问的应用模式，各种应用模式之间通过限制访问共享文件的应用节点及操作类型加以区分，具有一定的关联性，图 2.3 给出这六种应用模式之间的关联。

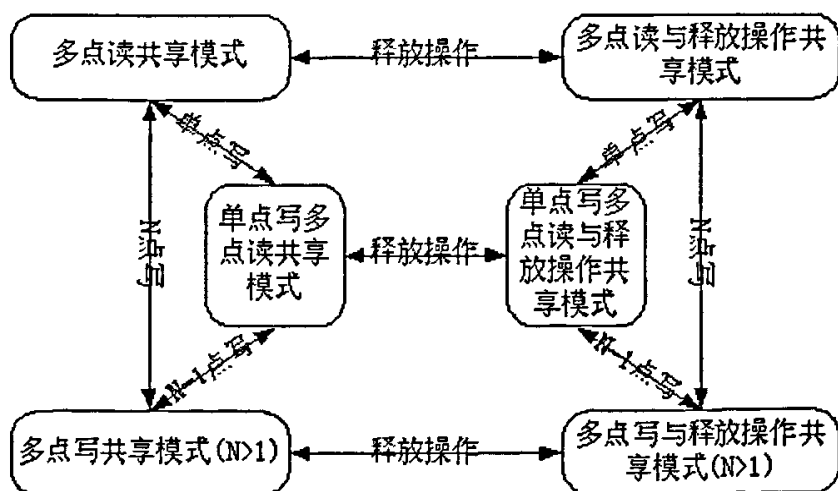


图 2.3 分布式应用模式关联图

第三章 基于软件的网络容错技术

节点间数据传输的正确性及完整性是蓝鲸分布式文件系统可用性的重要体现，也是其数据一致性的基础。在蓝鲸分布式文件系统中，网络故障是导致数据传输异常的主要原因，网络容错是节点间数据传输保障的关键所在。通过加强底层数据传输的可靠性，使得上层系统的数据一致性建立在可信的数据传输基础之上是蓝鲸分布式文件系统网络容错的重要目标。

本章从网络故障的分类入手，概要介绍热备容错、协议容错等常用的网络容错方式；然后结合蓝鲸分布式文件系统的实际情况，详细描述其基于软件的网络容错技术。

3.1 网络容错概览

3.1.1 网络故障分类

明确网络故障是进行网络容错的前提，在一个复杂的分布式系统中，可能发生的网络故障多种多样，其对系统所造成的影响也不尽相同。目前关于网络故障的划分有很多标准，依据网络故障的原因划分为：

- 网络硬（固）件故障

如网卡故障、网线故障、交换机故障等。

- 网络软件故障

如网卡驱动异常、系统协议栈异常、其他软件错误关联异常等。

依据网络故障对系统的影响程度划分为：

- 性能骤降

表现为某项（些）网络性能指标急剧下降，虽然可以收发数据，但与正常性能指标相去甚远。

- 连接中断

表现为某个（些）曾经建立的网络连接（如 TCP 连接）异常终止，但节点之间尚可进行通信。

- 网络分割

表现为系统中部分节点间的通信受阻（单向受阻或双向受阻），不能进行相应的数据接收或发送。

上述关于网络故障的划分并非绝对，不同的划分标准导致不同的划分结果，各种网络容错技术往往针对特定的网络故障，它们可能是某些其他标准（或交叉标准）的划分结果。

3.1.2 热备容错

所谓热备[45]指当系统中的某一部件发生故障时，使用该部件的替代品立即接替它继续工作。热备隐含着冗余的思想，但仅有冗余是不够的，热备需要软件或固件的支持，用于系统故障监测、工作切换以及故障恢复等。常用的热备模式有如下几种：（如图 3.1 所示，部件 A 与部件 B 实现相同的系统功能）

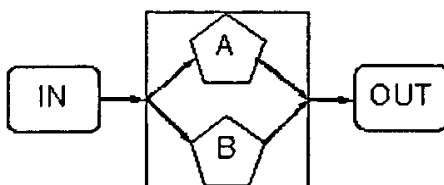


图 3.1 热备示意图

- 主从热备

系统正常运行时只有部件 A 参与系统工作，当且仅当部件 A 发生故障时，部件 B 才接替部件 A 参与系统工作。

- 镜像热备

系统正常运行时部件 A 与部件 B 各自完成相同的工作，当一个部件出现故障时，另一部件的工作不受影响。

- 互补热备

系统正常运行时部件 A 与部件 B 分担系统工作，当部件 A 发生故障时，部件 B 接管部件 A 的工作，即原先由两者分担的工作变成由部件 B 单独完成，反之亦是如此。

热备是计算机领域中十分常见的容错技术，从冗余电源到双机热备，从双通道磁盘到 RAID 阵列等都是热备容错的实例。对于网络容错而言，多网卡的容错方式绑定是较为普遍的网络容错手段。

3.1.3 协议容错

网络协议[46]定义了数据在节点间的传输管理规范，针对可能出现的网络传输异常，如误码、乱序、丢包等，部分协议采取相应的容错措施，主流的 TCP/IP 协议便是如此。

校验和是一种常见的协议容错手段。发送方基于一定的算法从待发送的部分（如 IP 协议[47]校验和仅覆盖 IP 首部）数据或整体（如 TCP 协议[48]校验和覆盖首部及有效载荷）数据中计算出若干校验信息附加到发送数据中，接收方根据接收到的数据再次计算校验信息，如果校验信息吻合则接受数据，否则丢弃数据（如 IP 协议）或要求重传（如 TCP 协议）。

通常而言，协议（尤其是物理网络层）规定最大传输单元——MTU (Maximum Transmission Unit)，不同的网络配置可能导致不同的 MTU，超长的上层数据包被分割成

较小的数据包分次传输,这些较小的数据包可能通过不同的路径以不同的次序达到接收方,接收方对应的协议层必须提供相应的排序重组功能来恢复原始数据包——不论这些碎片数据包以何种次序达到或以何种方式分割。IP 协议提供这种功能。

3.1.4 小结

协议容错作为多数系统的默认支持提供了基本的网络容错功能,对于提供更高可用性的分布式系统而言需要额外的网络容错支持。目前大多数分布式系统的网络容错通过复杂的硬件冗余热备实现,如 Caltech 的 RAIN System[49]。这类容错方式基于冗余度较高的网络拓扑模型,在很大程度上解决了硬件原因引起的网络故障,对处理网络分割尤为有效。其所带来的问题是较高的硬件成本及工程实施上的困难,且对硬件系统运行良好的网络故障(如网络性能骤降异常、逻辑连接中断异常等)处理欠佳,而这正是蓝鲸分布式文件系统基于软件的网络容错技术所要解决的问题。

3.2 蓝鲸分布式文件系统的网络容错

在蓝鲸分布式文件系统中,针对网络硬件故障,采取多网卡容错绑定等热备措施加以解决;针对误码、丢包等异常,通过系统的 TCP/IP 协议栈自动容错;本章主要针对各种软件因素或人为因素导致的非分割类网络故障进行基于软件的网络容错处理。

如前所述,网络故障可能导致系统性能骤降、连接中断、网络分割等,就蓝鲸分布式文件系统而言,对其节点间数据传输影响较大的网络故障是通道连接软故障中断异常:应用节点与存储节点之间的 TCP 连接通道在网络硬件运行良好的情况下异常中断(可能是系统软件故障,也可能是误操作所为,系统底层无法区分),导致基于该通道的数据传输失败。同样的问题存在于标准 NBD (Network Block Device)[50]系统等其他 TCP 相关系统中。在基于 TCP 的小规模简单分布式系统中,发生上述异常的概率不大,但对于蓝鲸分布式文件系统这样的大规模分布式系统而言,前端应用服务器机群和后端的存储节点机群之间存在几十上百甚至几千个 TCP 通道连接,这种通道连接软故障中断异常被放大若干个数量级,成为一个比较现实的问题。

针对通道连接软故障中断异常,蓝鲸分布式文件系统采用连接复制、通道切换、请求重构等软件技术[51]加以容错。

在蓝鲸分布式文件系统中,存储节点是数据传输通道的服务端,被动响应应用节点的操作请求。如果未作特殊说明,下述关于连接复制、通道切换、请求重构等技术的描述都从应用节点的角度出发。

3.2.1 连接复制

在蓝鲸分布式文件系统中,应用节点与存储节点之间的数据传输通过 TCP 连接通道进行,上述对蓝鲸分布式文件系统影响较大的网络故障最终表现为该通道的异常,加强

该通道的可靠性对于节点间数据传输而言至关重要。受热备容错思想的启发，可以对该通道做软件“热备”，以便异常发生时可以使用备份通道继续工作。

传输通道本身占用系统公共资源（内存、端口等），其建立时申请系统公共资源，其消亡时释放系统公共资源。在没有上述网络故障时，备份通道占用系统公共资源是不必要的，而当上述网络故障发生时，之前建立的备份通道可能亦受网络故障影响而不再可靠；另一方面在多存储节点的蓝鲸分布式文件系统中，应用节点与每一个存储节点之间都存在不同的连接通道，不区分情况地为每一个连接通道都建立一个备份通道是对系统公共资源的浪费。因此这种通道“热备”不同于通常意义上的硬件冗余热备，它不是静态持久分配的，而是动态按需分配的，也即监测到上述网络故障时才对故障通道进行“复制”备份。

所谓复制是因为对于每一个传输通道而言，其在建立时初始化了一些参数，包括存储节点 IP 地址、目标端口、认证、监控等蓝鲸分布式文件系统相关信息，这些参数因通道而异。为确保应用节点与存储节点之间可以建立并使用备份通道代替故障通道进行数据传输，须使用与故障通道一致的参数建立备份通道。

当应用节点监测到上述网络故障时，首先提取故障通道参数，然后试图使用相同参数建立一个备份通道，一旦备份通道建立完毕就可以使用后续的通道切换技术用备份通道替换故障通道。图 3.2 是蓝鲸分布式文件系统通道连接复制成功后的示意图。

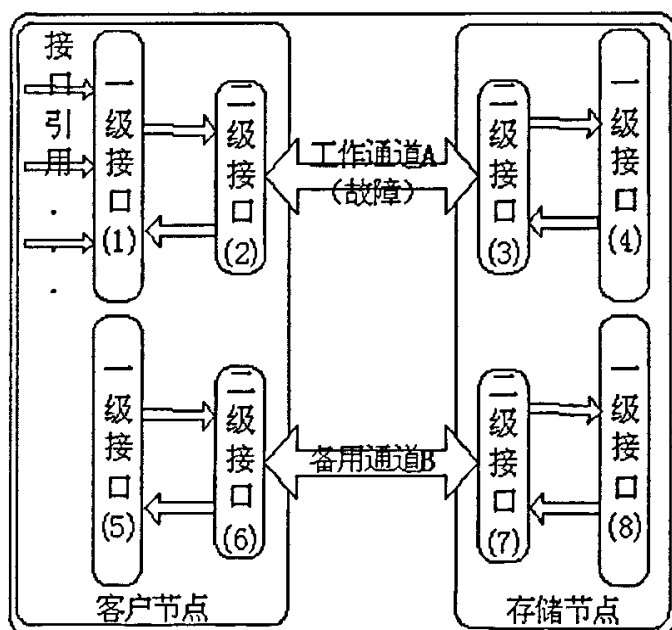


图 3.2 BWFS 连接复制示意图

3.2.2 通道切换

所谓通道切换就是把发生故障的工作通道上的数据流切换到备用通道上，利用备用通道进行数据传输。如图 3.2 所示，蓝鲸分布式文件系统中应用节点与存储节点之间的

数据传输通道包含十分复杂的数据结构：传输通道的每一端都提供两级逻辑接口，其中一级接口是与蓝鲸分布式文件系统耦合比较紧密的部分，二级接口是与操作系统网络协议栈耦合比较紧密的部分，两级接口之间通过众多相互引用（在图 3.2 中全部抽象为两个方向的引用箭头）建立关联。对于蓝鲸分布式文件系统上层而言，一级接口之下的结构是透明的，其不关心一级接口与哪个通道关联，也不关心数据如何传输，所有对通道的使用都通过对一级接口的引用进行，在应用节点端存在很多对一级接口的无记录引用，无法通过修改这些引用来切换通道；另一方面二级接口之下的部分与操作系统网络协议栈紧密耦合，其无须了解上层结构，通过修改这一部分来切换通道亦不合适。因此只有通过修改一级接口与二级接口之间的关联来实现通道切换。

通道切换是容错过程的关键所在，其难点在于处理一级接口与二级接口之间的复杂关联关系，切换过程必须确保切换双方的每个两级接口之间的相互引用全部按序修正到位，否则可能引起访问无效资源的严重系统错误，甚至导致系统崩溃。切换过程如图 3.3 所示：

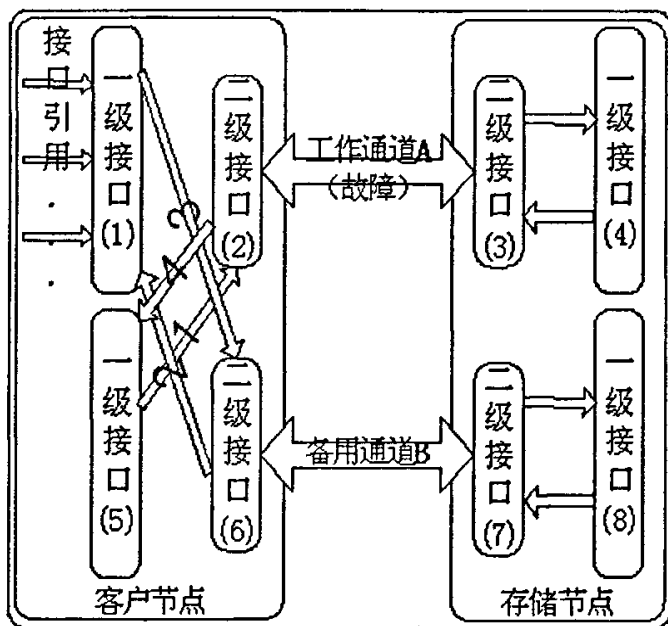


图 3.3 BWFS 通道切换示意图

1. 将备用通道 B 的一级接口 (5) 对二级接口的引用指向故障通道 A 的二级接口 (2)；
2. 将备用通道 B 的二级接口 (6) 对一级接口的引用指向故障通道 A 的一级接口 (1)；
3. 将故障通道 A 的一级接口 (1) 对二级接口的引用指向备用通道 B 的二级接口 (6)；
4. 将故障通道 A 的二级接口 (2) 对一级接口的引用指向备用通道 B 的一级接口 (5)。

整个切换过程中,系统暂停处理应用节点与故障通道对应存储节点之间的所有读写请求。待切换处理结束后,原工作通道 A 的一级接口(1)与备用通道 B 的二级接口(6)建立关联,而备用通道 B 的一级接口(5)则关联原工作通道 A 的二级接口(2),这样从上层看来,系统依旧使用原先的一级接口(1),但实际的工作通道已经切换到备用通道 B,备用通道 B 转变为工作通道,而故障通道 A 连同二级接口(2)可以通过一级接口(5)进行释放。

在蓝鲸分布式文件系统中,应用节点是主动方,存储节点是被动方,上述网络故障不论是应用节点问题,还是存储节点问题,或是其他原因引起,都由应用节点发起通道切换,当存储节点发现通道失效(如连接中断或应用节点主动销毁)时,将释放对应通道在本节点所持有的系统资源。

3.2.3 请求重构

经过连接复制和通道切换处理后,应用节点与存储节点之间的数据传输已经可以正常进行,但对于通道切换时刻尚未处理完毕的请求(不含尚未开始处理的请求)而言需要额外的重构工作,用以恢复网络故障时刻故障通道上的请求处理,使之处于可控状态。在蓝鲸分布式文件系统中,请求处理状态主要包括下述几种:

- 1) 读请求尚未处理,等待调度
- 2) 读请求正在发送
- 3) 读请求已经发出,等待存储节点数据
- 4) 读请求正在接收数据
- 5) 写请求尚未处理,等待调度
- 6) 写请求正在发送
- 7) 写请求已经发出,等待存储节点确认
- 8) 写请求正在接收确认

当系统发生上述网络故障而进行通道切换时,正在使用(状态 2、4、6、8)和即将使用(状态 3、7)故障工作通道进行数据传输的操作将失败,上述状态中只有状态 1 和状态 5 是可控的,其他状态下的数据是不完整(针对读请求)或不确定(针对写请求)的,需要重构。

所谓请求重构指当系统发生上述网络故障时,在必要的连接复制和通道切换完成之后,将故障通道上尚未处理结束的、处于非 1 且非 5 状态下的请求按照一定规则重新插入请求队列,等待被再次调度,以便使该通道恢复到网络故障发生之前某个时刻的可控状态(该通道上所有请求全部处于状态 1 或状态 5)。由于请求本身携带绝对位置信息,重复的读写操作不会对系统的数据有效性造成影响(此处暂不考虑多个应用节点并发操作所带来的影响)。重构规则如下:

- I. 将处于状态 2 的请求重新插入请求队列头部,恢复到状态 1;将处于状态 6 的请

求重新插入请求队列头部，恢复到 5 状态；

II. 将处于状态 3 的请求重新插入请求队列头部，恢复到状态 1；将处于状态 7 的请求重新插入请求队列头部，恢复到 5 状态；

III. 将处于状态 4 的请求重新插入请求队列头部，恢复到状态 1；将处于状态 8 的请求重新插入请求队列头部，恢复到 5 状态。

上述优先级依次递减，同一优先级确保其在重构后的请求队列中的顺序与重构前的相应顺序一致。经过这样的重构处理，系统可以恢复到网络故障发生之前某个时刻的可控状态，并且保持原系统请求之间的依赖关系，网络故障发生时刻的不完整或不确定数据将通过相应请求的再次调度处理而得到恢复。

3.2.4 效能测试

验证上述连接复制、通道切换、请求重构等基于软件的网络容错技术针对通道连接软故障中断异常能够实现高效的网络容错。测试环境如下：

表 3.1 BWFS 基于软件的网络容错技术测试环境配置列表

	应用节点	元数据服务器节点	存储节点	交换机
硬件	数量：2 CPU: Intel(R) Xeon (TM) 2.40GHz×2 MEM: 1024MB 网卡: Intel 82545EM 千兆以太网控制器×1	数量：1 CPU: Intel(R) Xeon (TM) 2.40GHz×1 MEM: 1024MB 网卡: Intel 82545EM 千兆以太网控制器×1	数量：1 CPU: Intel(R) Xeon (TM) 2.40GHz×1 MEM: 2048MB 网卡: Intel 82545EM 千兆以太网控制器×1 存储: 3ware 9000 for ATA-RAID 盘阵 (160GB×4, RAID0)	NETGEAR JGS524 千兆交换机
软件	Redhat8.0+BWFS3.0	Redhat8.0+BWFS3.0	Redhat8.0+BWFS3.0	

测试一：

通过从存储节点端强制关闭相关通道连接的方式模拟通道连接软故障中断异常，针对容错前后的蓝鲸分布式文件系统进行对比测试。测试方案如下：

在蓝鲸分布式文件系统两个应用节点的本地文件系统(ext3)中分别放置某数据体 (Linux 内核源代码复合体，包含目录文件 3420 个、普通文件 68360 个，占用空间 1323640KB) 的两个副本，其中一个应用节点将该数据体复制到测试目标系统，待复制操作结束之后由另外一个应用节点从测试目标系统中读出前一个应用节点复制的数据体，并与该节点本地的数据体副本进行比较。整个测试过程中，应用节点与存储节点之间的数据通道连接每隔 5 秒被强制关闭一次，确保在每个应用节点的测试过程中至少分

别出现一次模拟的通道连接软故障中断异常。

表 3.2 BWFS 基于软件的网络容错技术可行性测试结果

测试系统	测试结果
容错前的 BWFS	向 BWFS 复制数据体过程中系统提示 I/O error，复制操作异常退出；从 BWFS 读取数据体进行比较的过程系统提示 I/O error，比较操作异常退出。
容错后的 BWFS	向 BWFS 复制数据体成功，复制操作正常结束；从 BWFS 读取数据体成功；比较操作证实测试目标系统上的数据体与本地文件系统中的数据体在数据内容上完全一致。整个测试过程中应用未出现异常。

表 3.2 给出测试结果。实测表明，针对特定的网络故障——通道连接软故障中断异常，对于采用了基于软件的网络容错技术的蓝鲸分布式文件系统而言，其应用处理未受任何影响，实现了应用层透明的数据无损传输；而在采用该技术之前，蓝鲸分布式文件系统受数据通道连接中断的影响，出现数据丢失、应用异常退出等情况。表明相应的软件技术针对特定的网络故障真正到达网络容错目的，系统可用性大为提高，系统的数据一致性有所保障。

测试二：

通过从存储节点端强制关闭相关通道连接的方式模拟通道连接软故障中断异常，调整模拟故障频率，测试系统重负载情况下的性能变化。测试方案如下：

单个应用节点向目标测试系统写操作 16GB 数据体，其间存储节点与应用节点之间的相关数据通道连接以某种频率被关闭，每种情况测试三组数据取均值。

表 3.3 BWFS 基于软件的网络容错技术性能测试结果

测试情况	无容错无异常	有容错无异常	异常/5 秒	异常/3 秒	异常/1 秒
耗时（秒）	181.5	182.1	183.3	184.0	185.9
性能损失	0	0.33%	0.99%	1.38%	2.42%

表 3.3 给出在不同测试情况下向蓝鲸分布式文件系统写操作 16GB 数据体的耗时均值。可以看出在无通道连接软故障中断异常情况下，具有容错功能的蓝鲸分布式文件系统与无容错功能的蓝鲸分布式文件系统相比，其性能损失不超过 0.5%；故障频率不高于 1 次/5 秒情况的性能损失在 1%以下；当故障频率接近 1 次/1 秒时，其性能损失在 2.5%左右。对于系统负载不高的情况，通道连接软故障中断异常恢复的请求重构代价降低，性能损失相对较小。对于那些连接复制长时间不能成功的网络分割情况，其性能损失的主要因素不是相关容错技术。因此通常情况下蓝鲸分布式文件系统基于软件的网络容错技术的性能是可以接收的。

3.2.5 小结

对于网络分割类故障，上述容错技术将在连接复制阶段反复尝试，直至网络分割恢复后连接复制成功，然后继续进行通道切换与请求重构。因此上述网络容错技术可以用

于网络分割恢复后期的系统逻辑恢复工作。

本章未对网络故障期间发生多个应用节点并发访问冲突的情况进行处理，相关问题将在后继章节中进行讨论。

第四章 自适应的带外模式文件属性更新机制

文件是计算机系统记录信息的一种逻辑形式，通常包括数据与元数据两部分，其中数据是文件的实体信息，元数据是文件的控制信息（或称为描述信息）。尽管不同文件系统或不同文件类型对元数据定义有所不同，但普遍支持文件大小、文件修改时间等基本文件属性。如果没有特殊说明，本论文所描述的文件属性主要针对文件大小、文件修改时间等文件元数据信息。

本章首先概述文件属性更新模式，分析带内模式文件属性更新与带外模式文件属性更新的差异；然后介绍蓝鲸分布式文件系统自适应的带外模式文件属性更新机制，并对其文件属性更新周期的调整策略进行重点描述。

4.1 文件属性更新模式

通常而言，文件数据更新包括数据内容修改和数据增减，文件属性更新包括文件大小变化和文件修改时间变化，它们之间的变更关系如图 4.1 所示：

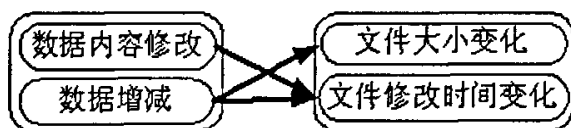


图 4.1 数据与文件属性变更关系图

上图表明这样一种关系：数据更新引发文件属性更新，文件属性更新隐含数据更新。进程通常根据文件属性的变化情况判断数据是否更新，进程能够从属性更新后的文件中获取更新后的数据。

上述变更关系在本地文件系统中比较容易满足。对于有服务器模式的分布式文件系统而言，这种变更关系不仅包括本地节点数据与文件属性的变更，还包括服务器节点的数据与文件属性的变更，并且可能涉及多个应用节点的并发访问，与数据一致性关系密切，相对比较复杂。本章主要针对有服务器模式的分布式文件系统讨论上述变更关系。

4.1.1 带内模式文件属性更新

所谓“带内”模式指分布式文件系统的服务器逻辑上既处理文件数据，又处理文件元数据，数据与元数据在应用节点与服务器节点间通过相同的逻辑通道进行传输。该模式的典型代表是 NFSv3 系统。

带内模式文件属性更新的特点是文件属性更新无须通过应用节点与服务器节点间的显式文件属性更新调用（如 `setattr`）进行，服务器节点能够根据本节点的数据更新对文件属性进行相应的更新，相当于文件属性更新信息隐含在待更新数据中一同从应用节点

传输到服务器节点，因此也称为文件属性的隐式更新。如图 4.2 所示：

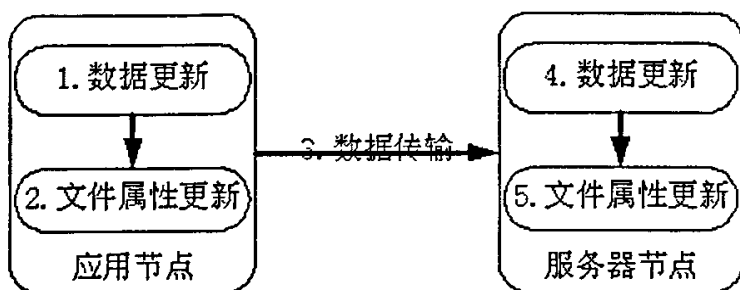


图 4.2 带内模式文件属性更新示意图

- 1) 应用节点数据更新
- 2) 应用节点文件属性更新
- 3) 应用节点向服务器节点写回脏数据
- 4) 服务器节点数据更新
- 5) 服务器节点文件属性更新

按照上述的更新顺序，能够保证其他应用节点的进程可以根据服务器节点上文件属性的变化情况来判断文件数据是否更新以及相应数据 cache 是否有效等，为后文将要讨论的数据一致性打下基础。

4.1.2 带外模式文件属性更新

所谓带外模式指分布式文件系统对服务器进行功能切分，数据处理与元数据处理通过不同的逻辑服务器完成，文件的数据与文件的元数据在应用节点与服务器节点间通过不同的逻辑通道进行传输。通常而言，逻辑服务器之间或逻辑通道之间是物理独立的，因此该模式带来的最大好处是数据带宽的可扩展性。

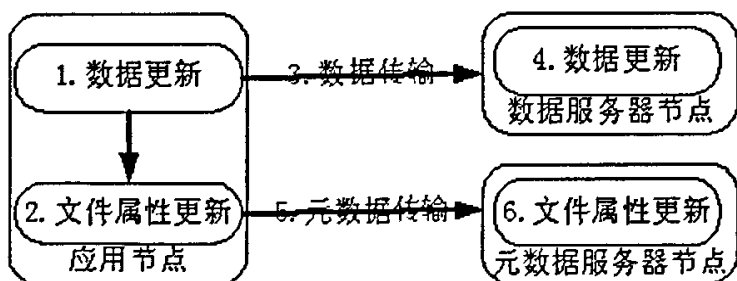


图 4.3 带外模式文件属性更新示意图

带外模式下，数据服务器节点的数据更新不会影响元数据服务器，与数据更新对应的文件属性更新信息不能像带内模式那样隐含于待更新的数据中，而是必须通过元数据通道显式传输，因此带外模式下文件属性更新必须通过应用节点与元数据服务器节点间的显式文件属性更新调用完成。见图 4.3。

- 1) 应用节点数据更新
- 2) 应用节点文件属性更新

- 3) 应用节点向数据服务器节点写回脏数据
- 4) 数据服务器节点数据更新
- 5) 应用节点向元数据服务器节点进行文件属性更新调用
- 6) 元数据服务器节点文件属性更新

上述更新顺序至关重要,与带内模式不同,带内模式下服务器节点的文件属性更新是隐式完成的,而带外模式下数据服务器节点的数据更新与元数据服务器节点的文件属性更新都是显式进行的,依照前述的数据与文件属性变更关系,两者之间存在拓扑顺序。如果元数据服务器节点的文件属性更新先于数据服务器节点的数据更新完成,则可能发生其他应用节点根据已更新的文件属性获取到过时数据的逻辑错误,与后文将要谈到的数据一致性相悖。

4.2 蓝鲸分布式文件系统的文件属性更新

蓝鲸分布式文件系统打破传统分布式文件系统单纯的客户/服务器模式,通过数据通道与控制通道的分离机制消除传统存储模式的数据传输瓶颈。受此影响,传统的带内模式文件属性更新机制失效,处理带外模式下的文件属性更新是蓝鲸分布式文件系统数据一致性的前提。

同多数分布式文件系统类似,蓝鲸分布式文件系统的应用节点使用本地高速缓存,针对数据更新采取 WRITE-BEHIND 策略,这些机制在提高蓝鲸分布式文件系统性能的同时带来了数据丢失的风险——如果发生应用节点宕机事件,那么该节点宕机前尚未写回的脏数据将丢失。蓝鲸分布式文件的文件属性更新作为脏数据写回的同步点,同时担负降低数据丢失风险的责任。

依照前述的数据与文件属性的变更关系,蓝鲸分布式文件系统的应用节点进程根据元数据服务器节点文件属性的变化情况判断文件数据是否更新以及该节点的数据 cache 是否有效,蓝鲸分布式文件的文件属性更新策略在保证上述变更关系的前提下,对提高系统性能与降低数据丢失风险进行权衡。为此蓝鲸分布式文件系统采用机会更新、被动更新、周期更新等多种方式相结合的文件属性更新策略。

4.2.1 机会更新

所谓机会更新指待更新的文件属性跟随系统上层的显式文件属性更新调用刷新到元数据服务器节点,或者在上层的显式脏数据写回后进行文件属性更新。机会更新是分布式文件系统中最基本的一种文件属性更新方式,其目的在于尽可能利用其他操作的网络传输机会,消除或降低文件属性更新的网络开销,提高系统性能。

在仅有机会更新方式的情况下,蓝鲸分布式文件系统应用节点进程对共享文件数据的更新与更新结果对其他应用节点进程可见的间隔时间是不确定的,取决于其间的机会更新点。在蓝鲸分布式文件系统中文件属性的机会更新点可以分成如下两类:

◇ 显式文件属性更新类

包括设置文件大小操作、设置文件时间操作、设置文件属主操作、设置文件权限操作等。

◇ 显式脏数据写回类

包括文件关闭操作、数据同步操作、同步写操作等。

上述文件属性的机会更新点在具体的操作系统平台下体现为若干系统调用，下文将基于 Linux-2.4.18-14 的操作系统平台对蓝鲸分布式文件系统的文件属性的机会更新点作具体描述。

4.2.1.1 显式文件属性更新类

在此类机会更新点处相应文件的脏数据可能尚未写回存储节点，因此必须先将目标文件的脏数据写回，然后才能向元数据服务器节点进行相应的文件属性更新调用。

◇ 设置文件大小操作

```
EXPORT_SYMBOL(sys_ftruncate);
EXPORT_SYMBOL(sys_truncate);
EXPORT_SYMBOL(sys_ftruncate64);
EXPORT_SYMBOL(sys_truncate64);
```

◇ 设置文件时间操作

```
EXPORT_SYMBOL(sys_utimes);
EXPORT_SYMBOL(sys_utime);
```

◇ 设置文件属主操作

```
EXPORT_SYMBOL(sys_fchown);
EXPORT_SYMBOL(sys_chown);
EXPORT_SYMBOL(sys_lchown);
```

◇ 设置文件权限操作

```
EXPORT_SYMBOL(sys_fchmod);
EXPORT_SYMBOL(sys_chmod);
```

4.2.1.2 显式脏数据写回类

在此类机会更新点处相应文件的脏数据已经写回存储节点，应用节点只需向元数据服务器节点进行相应的文件属性更新调用即可。

◇ 文件关闭操作

这种更新点的语义效果类似于会话语义，进程关闭文件或进程退出时将更新过的文件数据写回存储节点，并对元数据服务器节点的相应文件属性进行更新。

```
EXPORT_SYMBOL(sys_exit);
EXPORT_SYMBOL(svs_close);
```

✧ 数据同步操作

对于那些不愿付出同步写操作的性能代价,又想降低数据丢失风险的应用而言,通过显式的数据同步操作可以保证同步点上的数据安全性。

```
EXPORT_SYMBOL(sys_fsync);
EXPORT_SYMBOL(sys_sync);
EXPORT_SYMBOL(sys_fdatasync);
```

✧ 同步写操作

通常而言,系统默认的写操作是异步的,进行同步写操作需要指定特殊的标志,包括 O_SYNC、O_DIRECT 等,进程可以在打开文件时指定这些标志,也可以通过 fcntl 等调用进行设置。同步写操作使得蓝鲸分布式文件系统的 WRITE-BEHIND 策略失效,性能的代价较大。

```
EXPORT_SYMBOL(sys_write);
EXPORT_SYMBOL(sys_writev);
EXPORT_SYMBOL(sys_pwrite);
```

4.2.2 被动更新

对于机会更新而言,文件属性更新由应用节点进程控制,蓝鲸分布式文件系统只是提供这种控制的机制而已。这种单纯的文件属性更新机制所能提供的数据一致性十分有限。当多个应用节点对文件进行共享访问时,如果某个应用节点在本地对文件数据进行更新,其他应用节点想获取该更新结果,则必须等到更新节点的机会更新点出现时才能满足。为此蓝鲸分布式文件系统引入新的文件属性更新方式——被动更新。

所谓被动更新指元数据服务器节点强制应用节点将指定文件的脏数据写回存储节点并向元数据服务器节点进行相应的文件属性更新。对于应用节点而言,这种文件属性更新由元数据服务器节点发起,而非源自节点自身的需求;另一方面这种更新完全是异步随机的,应用节点进程无法预知何时发生这种更新,因此称为被动更新。

被动更新同样涉及文件属性更新点问题,在蓝鲸分布式文件系统中文件属性的被动更新点出现在授权剥夺处。授权是蓝鲸分布式文件系统针对数据一致性问题引入的一种机制,本章仅就其与文件属性被动更新相关部分作简单描述,关于授权的具体讨论留在后继章节进行。

应用节点进程在进行本地数据更新前须先从元数据服务器节点获取目标文件的相应授权,拥有授权后的相关操作可以不必考虑其他应用节点对该文件的共享访问。当这种授权被剥夺时,应用节点要把目标文件的脏数据写回存储节点,然后向元数据服务器节点进行相应的文件属性更新。

◇ 显式剥夺

元数据服务器节点在处理授权申请时会检测授权申请与已有授权之间的兼容性,如果发生冲突,那么元数据服务器节点将向已有授权的持有节点显式发出授权剥夺命令。

◇ 隐式剥夺

元数据服务器节点是授权的控制点,为防止元数据服务器节点宕机重启可能导致的冲突授权问题,应用节点一旦发现(或被告知)元数据服务器节点宕机重启,则默认该节点的所有授权被剥夺。

4.2.3 周期更新

如前所述,蓝鲸分布式文件系统的脏数据写回是异步的,文件属性更新作为脏数据写回的同步点担负降低数据丢失风险的责任。上述文件属性的机会更新和被动更新都带有相当程度的不确定性,应用节点本身对文件属性更新缺乏控制,应用节点宕机所带来的数据丢失风险受应用程序及文件并发访问状态的影响很大,难以评估。为此蓝鲸分布式文件系统提供周期更新方式加以解决。

所谓周期更新指蓝鲸分布式文件系统限定待更新文件属性在应用节点的滞留时间——文件属性更新周期:对于滞留时间超过更新周期的,系统将其进行强制性的脏数据写回,并向元数据服务器节点进行文件属性更新;对于滞留时间未超过更新周期的,系统允许其在该应用节点继续滞留,以增加脏数据写回的聚合度。这样在没有机会更新和被动更新的情况下,文件的脏数据将被周期性的同步回存储节点,元数据服务器节点的相应文件属性呈现周期性更新。机会更新或被动更新会导致滞留时间复位,因此周期更新能够与前述两种属性更新方式配合工作。

通过周期更新,应用节点加强了对文件属性更新的控制,另一方面应用节点宕机所带来的数据丢失风险变得明确化——在一个文件属性更新周期内,脏数据要么通过机会更新或被动更新方式写回,要么通过周期更新方式强制写回,应用节点宕机带来的数据损失被限制在一个文件属性更新周期内。

4.2.3.1 更新周期的设置

蓝鲸分布式文件系统文件属性更新周期对应用节点宕机的数据丢失风险及系统性能损失两方面造成影响,两者对系统的作用相反:文件属性的更新周期越短,应用节点宕机的数据丢失风险越小,数据损失越小,但系统性能的损失越大;反之更新周期越长,节点宕机的数据丢失风险越大,数据损失越大,但系统性能损失越小。因此蓝鲸分布式文件系统文件属性更新周期的选择需要在上述两个因素之间进行权衡,这种权衡是应用相关的:对数据安全敏感的应用可以适当减短文件属性的更新周期,对系统性能敏感的应用可以适当加长文件属性的更新周期,此外,如果应用自身提供适当的机会更新点或

者分布式应用的多个应用节点需要进行频繁的文件共享访问（引发被动更新），那么也可以适当加长文件属性的更新周期。

通常而言，文件系统本身并不了解应用模式，无法区分应用是数据安全敏感还是系统性能敏感，因此蓝鲸分布式文件系统向应用层提供接口，允许用户根据应用的实际情况动态设置文件属性的更新周期。这种设置可以在系统运行的任何时刻进行，而且可以为每个应用节点单独设置，以更好的满足各种不同应用模式的需求，提高了蓝鲸分布式文件系统的灵活性。

4.2.3.2 更新周期的自适应调整

上述关于文件属性更新策略的讨论以及对文件属性更新周期设置的讨论都没有涉及网络状态的问题，实际上网络状态对数据的安全性及系统性能都有很大影响，频繁的网络中断不仅严重影响系统性能，而且往往预示着系统崩溃的临近。因此很有必要结合网络状态讨论蓝鲸分布式文件系统文件属性更新问题。

文件属性的周期更新并不是蓝鲸分布式文件系统最先使用的，SPRITE 等系统中早有类似机制，蓝鲸分布式文件系统的特点在于其文件属性的更新周期能够根据当前的网络状况进行自适应调整，调整策略如下：

- 1) 如果网络连接发生中断，表明网络可能处于不稳定状态，系统崩溃的潜在可能性加大，此时需要减短文件属性的更新周期，也即加快脏数据写回的频度，以降低数据丢失的风险及损失。

$$T \leftarrow T - \Delta T$$

- 2) 当网络性能明显降低或服务器节点响应迟缓时，表明网络可能比较繁忙或者服务器节点比较繁忙，此时应加长文件属性的更新周期，也即降低脏数据写回的频度，降低本节点的当前数据流量，以避免网络拥塞恶化或者服务器节点过载加剧。

$$T \leftarrow T + \Delta T$$

- 3) 当网络状态恢复正常状态一段时间（文件属性更新周期基数的若干倍）后，文件属性更新周期会自动恢复为文件属性更新周期基数。

$$T \leftarrow T_0$$

在上述调整策略的作用下，蓝鲸分布式文件系统的文件属性更新周期（ T ）会在上述文件属性更新周期设置的基数（ T_0 ）之上，进行适当幅度（ $n \times \Delta T$ ）的调整，调整的幅度同样可以通过蓝鲸分布式文件系统提供给应用层的调用接口进行设置。

文件属性更新周期自适应调整特性的引入出于降低数据丢失风险与提高系统效率的双重考虑，使得网络流量趋于均稳，减少了抖动及拥塞。

4.2.4 效能测试

测试蓝鲸分布式文件系统自适应的带外模式文件属性更新机制的能效及其对系统性能的影响。测试环境如下：

表 4.1 BWFS 自适应的带外模式文件属性更新机制测试环境配置列表

	应用节点	元数据服务器节点	存储节点	交换机
硬件	数量：1 CPU：Intel(R) Xeon (TM) 2.40GHz×2 MEM：1024MB 网卡：Intel 82545EM 千兆以太网控制器×1	数量：1 CPU：Intel(R) Xeon (TM) 2.40GHz×1 MEM：1024MB 网卡：Intel 82545EM 千兆以太网控制器×1	数量：1 CPU：Intel(R) Xeon (TM) 2.40GHz×1 MEM：2048MB 网卡：Intel 82545EM 千兆以太网控制器×1 存储：3ware 9000 for ATA-RAID 盘阵 (160GB×6, RAID0)	NETGEAR JGS524 千兆交换机
软件	Redhat8.0+BWFS3.0	Redhat8.0+BWFS3.0	Redhat8.0+BWFS3.0	

测试一：

通过模拟网络异常测试蓝鲸分布式文件系统文件属性更新周期的自适应调整机制，测试中以从存储节点端强制关闭相关通道连接的方式模拟通道连接软故障中断异常。测试方案如下：

单个应用节点对蓝鲸分布式文件系统某个文件执行持续的限速写操作，文件属性更新周期基数设置为 30 秒，调整粒度为 10 秒；从元数据服务器节点实时监控目标文件的属性变化情况；其间应用节点与存储节点之间的数据通道连接不定时地被强制关闭一次，关闭间隔在 10 秒至 120 秒之间随机取值。

图 4.4 给出蓝鲸分布式文件系统文件属性更新周期的自适应调整曲线（截取前 5 分钟的数据），上方曲线代表文件属性的更新时间，下方曲线代表文件属性的更新周期。

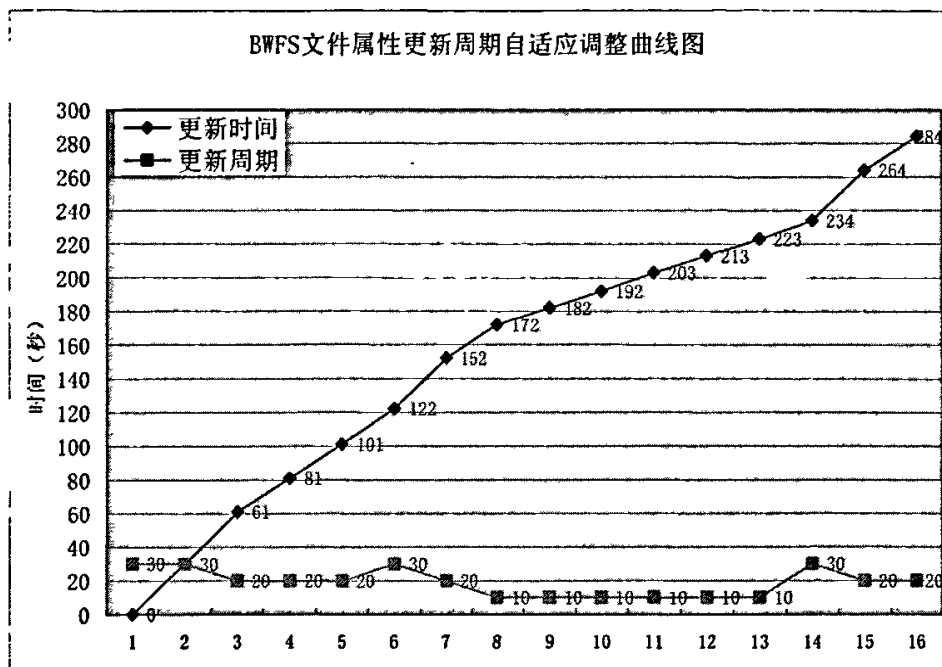


图 4.4 BWFS 文件属性更新周期自适应调整曲线

从图中可以看出，受通道连接软故障中断异常影响，蓝鲸分布式文件系统文件属性更新周期以更新周期基数为峰值作下波动。受系统调度及网络影响，图中文件属性更新的时间点与文件属性更新周期并不完全对应。

测试二：

关闭应用节点的文件属性更新周期自适应调整功能，通过设置不同的文件属性更新周期，测试文件属性更新频率对系统性能的影响。测试方案如下：

分别在文件属性更新周期为 1 秒、2 秒、4 秒、8 秒、16 秒、32 秒、64 秒以及文件被关闭时刷新等几种情况下，测试单个应用节点向目标测试系统写操作 320GB 单个数据体的耗时。

表 4.2 BWFS 自适应的带外模式文件属性更新机制性能测试结果

更新周期	关闭刷新	64 秒	32 秒	16 秒	8 秒	4 秒	2 秒	1 秒
耗时（秒）	3765	3824	3879	3961	4078	4218	4455	4892
性能损失	0	1.57%	3.03%	5.21%	8.31%	12.03%	18.33%	29.93%

从表 4.2 给出的测试结果中可以看出，在文件属性更新周期不小于 32 秒的情况下，文件属性更新对系统性能的影响在 3% 左右，当文件属性更新周期缩短至 4 秒以下时，文件属性更新所造成的系统额外开销超过 10%。此表可以作为蓝鲸分布式文件系统文件属性更新周期基数设定的参考，目前系统默认的文件属性更新周期基数为 30 秒。

4.3 小结

蓝鲸分布式文件系统的脏数据写回是异步的,文件属性更新是脏数据写回的同步点,在该点之前写回的脏数据对其他应用节点是否可见取决于蓝鲸分布式文件系统的数据一致性语义,针对该问题深入讨论将在后继章节中进行。

第五章 基于授权机制的在线可调整的数据一致性语义模型

关于分布式文件系统应该提供何种数据一致性语义的问题，人们往往认为系统提供的数据一致性语义越严格越好，实际上可能并非如此。数据一致性的维护是有代价的，越是严格的数据一致性其维护的代价越高。通过前面对分布式应用模式的分析可以知道，不同的应用模式对数据一致性的要求不尽相同，为那些并不需要很强数据一致性的应用维护高级别的数据一致性是不必要的，并且可能造成并发度降低等损失；另一方面，某些应用为提高平台兼容性，降低对底层系统数据一致性支持的依赖，应用本身通常提供适当的并发访问控制，为这种应用维护高级别的数据一致性不仅不必要，而且可能对应用原有的并发访问控制造成负面影响。

基于上述考虑，蓝鲸分布式文件系统采用一种基于授权机制的在线可调整的数据一致性语义模型，允许用户根据应用的实际需要在线调整蓝鲸分布式文件系统的数据一致性语义，以期用尽可能低的数据一致性维护代价获取尽可能高的应用兼容性。这是本论文的重点所在。

5.1 模型语义

蓝鲸分布式文件系统数据一致性语义模型的语义支持是该模型的外在表现，在详细介绍该模型的机理之前，先描述其所支持的数据一致性语义。作为一种支持数据一致性语义在线调整的模型，该模型提供超时一致性、释放一致性、写一致性、读写一致性等四种数据一致性语义，用户可以根据实际需要它们在它们之间进行动态切换。

```
#define CONSISTENCY_TIMEOUT      0x0001
#define CONSISTENCY_RELEASE      0x0010
#define CONSISTENCY_WRITE        0x0100
#define CONSISTENCY_READ_WRITE  0x1000
```

5.1.1 超时一致性 (Timeout Consistency)

超时一致性通过类似于 NFSv3 系统的 cache 超时机制主动检测文件更新以降低应用节点之间 cache 一致性问题所带来的风险，为加强对基于 NFSv3 系统应用的兼容性，其超时设置尽量与 NFSv3 系统保持一致。

超时一致性语义所对应的数据获取流程如图 5.1 所示（暂不考虑授权处理）：

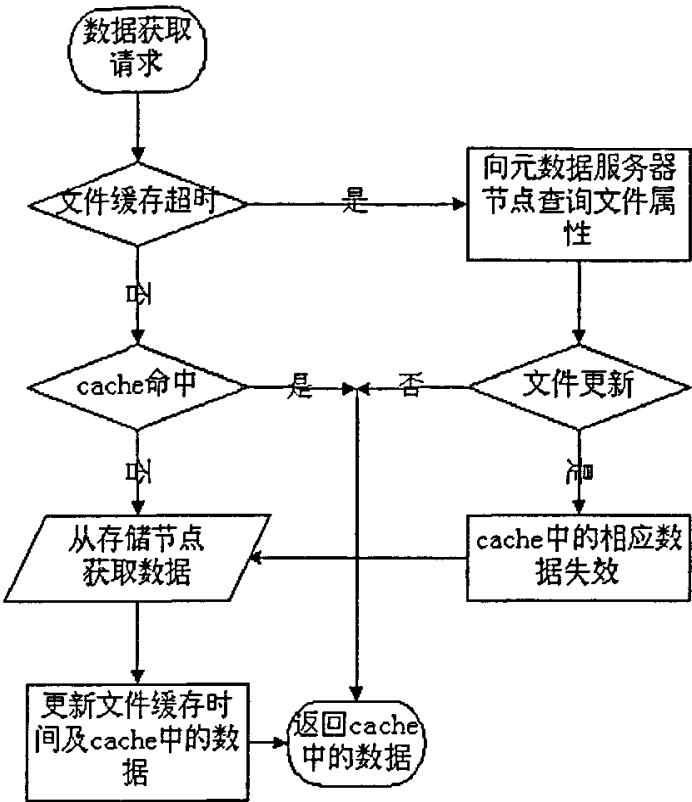


图 5.1 超时一致性语义下数据获取流程图

该数据一致性语义下对共享文件的修改操作与该操作结果对全局可见之间的时间间隔取决于蓝鲸分布式文件系统的文件属性更新策略，由于蓝鲸分布式文件系统采用机会更新、被动更新、周期更新相结合文件属性更新策略，所以该时间间隔最长不会超过一个文件属性更新周期。

蓝鲸分布式文件系统的文件属性更新是脏数据写回的同步点，该同步点之前异步写回的脏数据在超时一致性语义下对外可见，其他应用节点对相应数据的使用可能是本地 cache 中的过时数据，也可能是存储节点上的最新数据。因此超时一致性提供的是非严格的弱 cache 一致性支持。

超时一致性是蓝鲸分布式文件系统数据一致性语义模型所能提供的一种语义最弱的数据一致性语义，能够很好地支持无共享模式和多点读共享模式的分布式应用。

5.1.2 释放一致性 (Release Consistency)

释放一致性在超时一致性的基础上加强对释放操作的并发访问控制，确保释放操作不会将正在被其他应用节点读写操作所共享的数据块释放，用以支持蓝鲸分布式文件系统的数据安全性。

该数据一致性语义下，文件属性更新之前通过异步方式写回的脏数据对其他应用节点可见，系统所提供的仍然是非严格的弱 cache 一致性支持。应用节点可能使用本地

cache 中的过时数据, 但不会读到空闲 (或被重新分配) 数据块中的数据, 不会把数据写到无效的存储空间 (数据块空闲或被重新分配), 系统的数据安全性有所保证。

释放一致性主要针对那些对系统数据安全性要求比较高的分布式应用, 尤其是对 cache 一致性要求不高的情况, 能够很好的支持多点读与释放操作共享模式, 对单点写多点读与释放操作共享模式亦能提供适当的支持。

5.1.3 写一致性 (Write Consistency)

写一致性在释放一致性的基础之上对多个应用节点并发更新共享文件进行控制, 消除多个应用节点并发写操作区域伪交叠所导致的更新数据非正常缺失问题, 实现蓝鲸分布式文件系统的数据完整性支持。

该数据一致性语义对 cache 一致性的支持有所加强, 文件属性更新之前通过异步方式写回的脏数据对读操作节点可见, 对写操作节点不可见, 即该数据一致性语义保证写操作节点之间的 cache 一致性, 但不保证读写节点之间的 cache 一致性, 是一种完全兼容且略强于 NFSv3 语义的数据一致性语义。

实现对写一致性语义的支持是蓝鲸分布式文件系统第一次真正意义上的完全兼容 NFSv3 系统, 在该语义下, 基于 NFSv3 系统的分布式应用可以透明移植到蓝鲸分布式文件系统上。系统可以在很大程度上 (弱 cache 一致性支持) 支持多点写共享模式和多点写与释放操作共享模式的分布式应用。

5.1.4 读写一致性 (Read Write Consistency)

读写一致性在写一致性的基础之上进一步加强对 cache 一致性的支持, 文件属性更新之前通过异步方式写回的脏数据对其他应用节点不再可见, 也即应用节点读操作得到的数据是其他应用节点针对该文件的最新属性更新的同步数据, 确保读写节点之间的 cache 一致性, 是一种强 cache 一致性语义。

读写一致性是一种十分接近 UNIX 语义 (暂称为类 UNIX 语义) 的数据一致性语义, 消除了蓝鲸分布式文件系统的包括释放共享数据块问题、写操作区域伪交叠问题、cache 一致性问题等在内的前述所有数据一致性问题, 能够完全兼容前述七种模式的分布式应用, 特别适合于那些基于强 cache 一致性支持但本身并不提供相应并发访问控制的应用。

5.1.5 小结

蓝鲸分布式文件系统数据一致性语义模型提供从类 NFS 语义到类 UNIX 语义等多种数据一致性语义的兼容性, 图 5.2 给出各种数据一致性语义间的兼容关系。

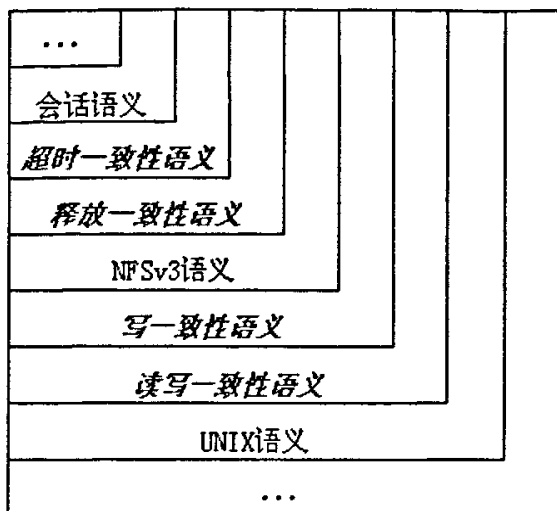


图 5.2 数据一致性语义兼容关系图

5.2 授权机制

通过前面的分析可以得出蓝鲸分布式文件系统数据一致性问题产生的主要原因：

- 1) 数据通道与控制通道的分离
- 2) 基于数据块级粒度的数据传输
- 3) 多个应用节点无协作地并发访问

原因 1) 是蓝鲸分布式文件系统相对于其他分布式文件系统的架构优势，是消除系统数据传输瓶颈的关键所在；原因 2) 是蓝鲸分布式文件系统平台兼容性的基础，是蓝鲸分布式文件系统潜在优势的一部分；原因 3) 虽然与系统的应用模式相关，但是通过建立多个应用节点之间的隐式（强制性的）协作关系可以控制多个应用节点的并发访问，是解决蓝鲸分布式文件系统数据一致性问题的出发点，为此蓝鲸分布式文件系统引入授权机制。

5.2.1 什么是授权

蓝鲸分布式文件系统的授权是一种可剥夺的文件级粒度的多态文件锁。机理如下：

- 1) 元数据服务器节点负责授权的集中管理，它依据一定的授权规则决定应用节点的相关操作是否可以执行，对允许执行的操作给予相应授权，并可以根据需要撤销先前批准的授权。
- 2) 应用节点在执行相关操作之前须先获得操作所对应的相关授权，未获得相关授权的操作不能被执行，应用节点可以把授权保存在本地，以便后继的相关操作可以直接使用而不必重复申请。

授权机制是蓝鲸分布式文件系统数据一致性语义模型的核心机制，通过元数据服务器节点对授权进行集中管理，建立并发访问共享文件的应用节点之间的隐式协作关系，

实现对应用节点的并发访问控制, 提供适当的数据一致性语义支持。

5.2.2 授权类型

蓝鲸分布式文件系统的授权是一种文件锁——持有授权的应用节点相当于持有对目标文件进行某种操作的锁。不同于普通的 0/1 锁——只有未锁与锁两种状态, 也不同于读写锁——未锁/读锁/写锁, 蓝鲸分布式文件系统的授权至少包含未授权/读授权/写授权/释放授权等四种锁状态, 是一种真正意义上的多态锁。

```
#define AUTHORIZATION_NONE      0x0000
#define AUTHORIZATION_READ      0x0001
#define AUTHORIZATION_WRITE     0x0010
#define AUTHORIZATION_RELEASE   0x0100
```

蓝鲸分布式文件系统需要授权的操作是那些可能导致蓝鲸分布式文件系统数据一致性问题操作, 主要涉及数据读写操作和数据块释放操作, 不同的操作需要的授权类型有所差别, 下表给出蓝鲸分布式文件系统中需要授权的操作及所需相关授权类型的对应关系:

表 5.1 操作授权类型表

操作类型	读操作(read)	写操作(write)	截短操作(truncate)	删除操作(unlink)
授权类型	读授权	写授权	释放授权	释放授权

从授权是否可以重复使用的角度可以将上述三种授权划分成两大类: 缓存类授权和非缓存类授权, 其主要差别在于授权是否可以被应用节点重复使用。

✧ 缓存类授权

包括读授权和写授权, 这类授权对应操作的频度较高, 应用节点将该类授权保存在应用节点本地, 在授权被释放或收回之前, 后继的相关操作可以使用该授权而不必进行重复申请。

✧ 非缓存类授权

包括释放授权, 此类授权对应操作的频度较低, 操作结束之后授权自动失效, 下次操作需要重新申请。

5.2.3 授权粒度

文件锁是有粒度的, 可以是字节级粒度, 或是数据块级粒度, 也可以是文件级粒度。以基于 Linux-2.4.18-14 内核的操作系统为例, 系统支持三种类型的文件锁: FL_LOCK 锁、FL_POSIX 锁、FL_LEASE 锁, 其中 FL_LOCK 锁和 FL_LEASE 锁是文件级粒度锁, FL_POSIX 锁是字节级粒度锁。

蓝鲸分布式文件系统的授权机制以解决其数据一致性问题为目标, 由于字节级粒度授权在处理写操作区域伪交叠问题时需要向数据块级粒度转化, 所以蓝鲸分布式文件系

统的授权粒度在文件级粒度和数据块级粒度中进行选择。下表是对两种授权粒度的优劣比较。

表 5.2 文件级授权粒度与数据块级授权粒度比较表

授权粒度 比较因素	数据块级粒度	文件级粒度
网络开销	操作相同文件的不同数据块区域需要多次授权申请, 对于大文件而言, 由此引发的额外网络开销不可忽视	对于缓存类授权, 如果没有授权剥夺, 那么针对相同文件的同类操作只需一次授权申请; 对于非缓存类授权, 其授权申请与操作次数相同
授权搜索	系统记录每个授权, 对于大文件而言, 这种记录很多, 搜索特定位置的授权并非易事; 元数据服务器节点进行授权兼容性测试的耗时更为可观	每个文件为每个应用节点的每类操作最多维护一个授权, 搜索简单高效
内存消耗	授权粒度较小, 相应的授权存储单元很多, 占用较多的系统内存资源, 在元数据服务器节点该问题被多个应用节点的共享访问进一步扩大	每个文件为每个应用节点的每类操作最多分配一个授权存储单元, 占用内存空间相对很少
系统并发度	多个应用节点针对相同文件不同数据块区域的互斥操作可以并发进行	多个应用节点针对相同文件的互斥操作须串行处理

表中关于系统并发度的比较与应用模式关系很大, 如果多个应用节点针对共享文件的操作区域相同, 那么两种授权粒度对系统并发度所造成的影响没有差异; 数据块级粒度在其他三个方面的劣势足以抵消其在系统并发度方面的优势。鉴于上述比较, 蓝鲸分布式文件系统的选取文件级粒度作为其数据一致性语义模型的授权粒度。

5.2.4 授权规则

授权规则是元数据服务器节点的授权依据, 元数据服务器节点据此判断多个应用节点之间的相关授权是否兼容, 新的授权申请是否可以满足, 已有授权是否需要撤销等。授权机制是蓝鲸分布式文件系统数据一致性语义模型的核心机制, 系统数据一致性语义依赖于授权规则, 授权规则的变化导致系统数据一致性语义的变化。蓝鲸分布式文件系统数据一致性语义模型的多种语义支持正是通过调整授权规则实现的。

下面给出蓝鲸分布式文件系统目前支持的四种数据一致性语义所对应的授权规则——授权兼容表。

1) 超时一致性

该语义下只有释放授权之间不兼容，其他类型授权之间都兼容。

表 5.3 超时一致性授权兼容表

授权申请 \ 已有授权	读授权	写授权	释放授权
读授权	兼容	兼容	兼容
写授权	兼容	兼容	兼容
释放授权	兼容	兼容	不兼容

2) 释放一致性

该语义下释放授权与其他类型授权之间不兼容，其他类型授权之间兼容。

表 5.4 释放一致性授权兼容表

授权申请 \ 已有授权	读授权	写授权	释放授权
读授权	兼容	兼容	不兼容
写授权	兼容	兼容	不兼容
释放授权	不兼容	不兼容	不兼容

3) 写一致性

该语义下释放授权与其他类型授权之间不兼容，写授权之间不兼容，其他类型授权之间兼容。

表 5.5 写一致性授权兼容表

授权申请 \ 已有授权	读授权	写授权	释放授权
读授权	兼容	兼容	不兼容
写授权	兼容	不兼容	不兼容
释放授权	不兼容	不兼容	不兼容

4) 读写一致性

该语义下只有读授权之间兼容，其他类型授权之间都不兼容。

表 5.6 读写一致性授权兼容表

授权申请 \ 已有授权	读授权	写授权	释放授权
读授权	兼容	不兼容	不兼容
写授权	不兼容	不兼容	不兼容
释放授权	不兼容	不兼容	不兼容

从上可以看出，随着各种类型授权之间兼容性的降低，系统的数据一致性语义逐渐增强，系统并发度也随之下降。

5.2.5 应用节点的授权管理

应用节点的授权管理主要负责授权信息在应用节点本地的存储及查询，存储空间及查询效率是其关心的两个重点，当两者产生冲突时采取以存储空间换查询效率的策略。

从存储的角度看，蓝鲸分布式文件系统应用节点对授权信息只作临时记录而非持久存储，其所记录的授权信息包含基本信息和辅助信息两部分：基本信息是授权本身的描述信息，通过这些信息应用节点可以对授权加以区分；扩展信息包括为方便查询而设置的链表、指针等信息。如图 5.3 所示：

```
struct authorization_client {
    struct double link    global_link;
    struct double link    hash_link;
    struct file pointer    file_pointer;
    unsigned long         file_identifier;
    unsigned long         authorization_type;
    unsigned long         consistency_type;
    unsigned long         consistency_set_time;
    ...
};
```

图 5.3 应用节点授权记录数据结构

1) 基本信息

- ✧ file_identifier
文件标识，描述授权属于哪个文件
- ✧ authorization_type
授权类型，描述授权属于何种类型
- ✧ consistency_type
数据一致性语义类型，描述授权是在何种数据一致性语义下获取的
- ✧ consistency_set_time
数据一致性语义设置时间，描述授权属于哪次数据一致性语义调整

2) 辅助信息

- ✧ global_link
授权记录的全局查询链
- ✧ hash_link
授权记录的 hash 查询链
- ✧ file_pointer
指向授权所属文件的指针，实际上通过 file_identifier 可以追踪授权所属文件，file_pointer 为加快查询等操作速度而设置

从查询的角度看，蓝鲸分布式文件系统应用节点的授权管理模块提供授权记录的文

件查询方式、全局查询方式、hash 查询方式等三种查询方式。

1) 文件查询方式

授权记录的文件查询方式是蓝鲸分布式文件系统应用节点授权管理最常使用的查询方式，主要用于执行读写操作前检测是否拥有相关授权的情况。为加快查询速度，蓝鲸分布式文件系统应用节点对文件属性的内存结构进行扩展，增设直接指向相关授权记录的指针，所以该查询方式简单且高效。

2) 全局查询方式

为了加强对授权记录的全局管理，蓝鲸分布式文件系统应用节点提供全局查询方式，通过追踪 `global_link` 双向链表，系统可以遍历本应用节点持有的所有授权记录。该查询方式主要用于数据一致性语义调整的历史授权处理、元数据服务器节点宕机所导致的授权清理等情况。

3) hash 查询方式

蓝鲸分布式文件系统应用节点还提供通过文件标识快速查询授权记录的 hash 查询方式，该查询方式主要用于元数据服务器节点根据文件标识进行授权撤销操作的情况，应用节点需要根据文件标识迅速定位特定的授权记录。实际上系统提供文件的快速定位方式，但由于授权记录往往比文件少，提供单独的 hash 查询会更加高效，并且降低了对系统其他机制的依赖，加强了模型的独立性。

图 5.4 给出蓝鲸分布式文件系统应用节点授权记录的内存逻辑视图。

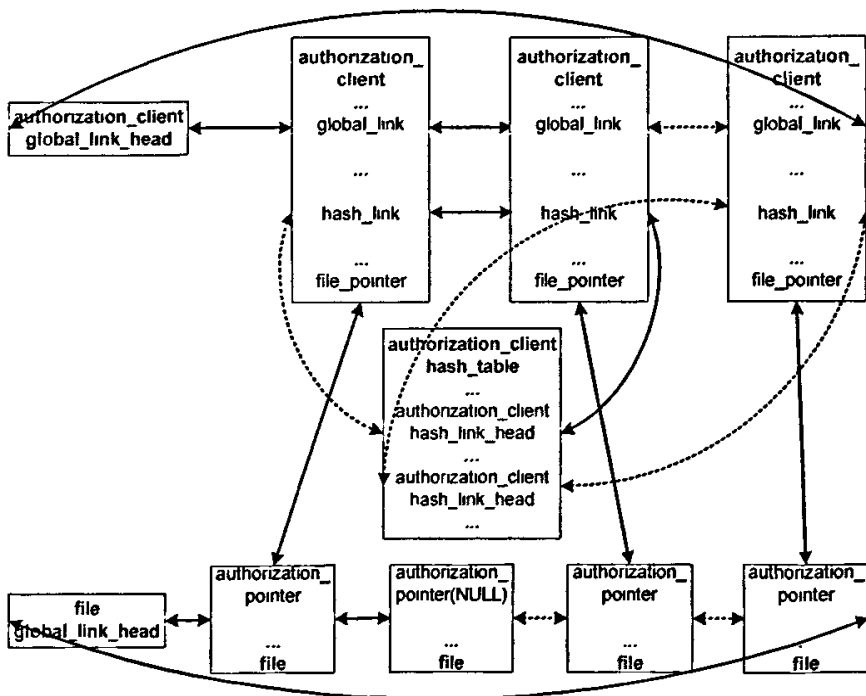


图 5.4 应用节点授权记录逻辑视图

5.2.6 元数据服务器节点的授权管理

元数据服务器节点的授权管理与应用节点的授权管理十分类似，不过由于元数据服务器节点是系统授权的集中控制点，所以授权记录的存储及查询压力比应用节点大得多。同样的，元数据服务器节点对授权信息也仅作临时记录而非持久存储，其所记录的授权信息包含基本信息和辅助信息两部分，如图 5.5 所示：

```

struct authorization_server {
    struct double link    global_link;
    struct double link    local_link;
    struct file pointer    file_pointer;
    unsigned long         client_identifier;
    unsigned long         file_identifier;
    unsigned long         authorization_type;
    unsigned long         consistency_type;
    unsigned long         consistency_set_time;
    ...
};

```

图 5.5 元数据服务器节点授权记录数据结构

1) 基本信息

- ✧ client_identifier
节点标识，描述授权属于哪个应用节点
- ✧ file_identifier
文件标识，描述授权属于哪个文件
- ✧ authorization_type
授权类型，描述授权属于何种类型
- ✧ consistency_type
数据一致性语义类型，描述授权是在何种数据一致性语义下获取的
- ✧ consistency_set_time
数据一致性语义设置时间，描述授权属于哪次数据一致性语义调整

2) 辅助信息

- ✧ global_link
授权记录的全局查询链
- ✧ local_link
授权记录的局部查询链，配合文件查询方式
- ✧ file_pointer
指向授权所属文件的指针，实际上通过 file_identifier 可以追踪授权所属文件，file_pointer 为加快查询等操作速度而设置

从查询的角度看,蓝鲸分布式文件系统元数据服务器节点的授权管理模块提供授权记录的文件查询方式和全局查询方式,但不支持类似于前述应用节点所提供的 hash 查询方式。与应用节点不同,元数据服务器节点是系统授权的集中控制点,内存中授权记录的数量很可能超过文件记录的数量,元数据服务器节点的授权记录查询更多的是为了授权兼容性测试,这种测试不是针对某个授权记录,而是针对某个文件的某种授权记录,此种情况下使用系统提供的通过文件标识快速定位文件的查询方式会更加便捷。

1) 文件查询方式

授权记录的文件查询方式是蓝鲸分布式文件系统元数据服务器节点授权管理最常使用的查询方式,系统中授权申请操作与大部分的授权释放操作都通过该方式进行。与应用节点不同,元数据服务器节点的单个文件可能对多个应用节点进行授权,即每个文件可能对应多个授权记录,元数据服务器节点不能像应用节点那样简单地对文件属性的内存结构进行扩展,增设直接指向授权记录的指针,而是需要另外的机制访问所有隶属于该文件的授权记录。一种可行的方案是对文件属性的内存结构进行扩展,增设指向授权记录的链表,把该文件对应的全部授权记录都链到该链表中,通过文件追踪该链表能够遍历系统中针对该文件的全部授权记录。

上述方案的一个缺陷是,系统每次进行授权兼容性测试时都要遍历该链表才能确认系统中目前没有与授权申请不兼容的授权存在,实际上这种兼容性测试的效率不高,原因是没有对授权记录进行分类存储。通过前面的授权兼容表可以发现,授权的兼容性测试只需要找出那些不兼容的授权即可,授权兼容表已经明确定义了各种数据一致性语义下各种类型授权之间的兼容性,那么在特定的数据一致性语义下只需要查询特定类型的授权即可。遵循上述思路,蓝鲸分布式文件系统元数据服务器节点对文件属性的内存结构进行如下扩展:增设三个指向授权记录的双向链表——读授权记录链表、写授权记录链表、释放授权记录链表,每个链表对应一种授权类型,每个授权记录根据授权类型通过 `local_link` 链入特定的授权记录链表。在进行授权兼容性测试时,系统只需要根据当前数据一致性语义对应的授权兼容表,搜索特定的授权记录链表即可,大大提高了查询效率。

2) 全局查询方式

蓝鲸分布式文件系统元数据服务器节点也提供授权记录的全局查询方式,通过追踪 `global_link` 双向链表,可以遍历整个蓝鲸分布式文件系统的所有授权记录。该查询方式主要用于内存压力下的系统无用授权回收,以及针对特定应用节点的授权进行全局搜索,如对宕机应用节点的授权清理等。

图 5.6 给出蓝鲸分布式文件系统元数据服务器授权记录的内存逻辑视图。

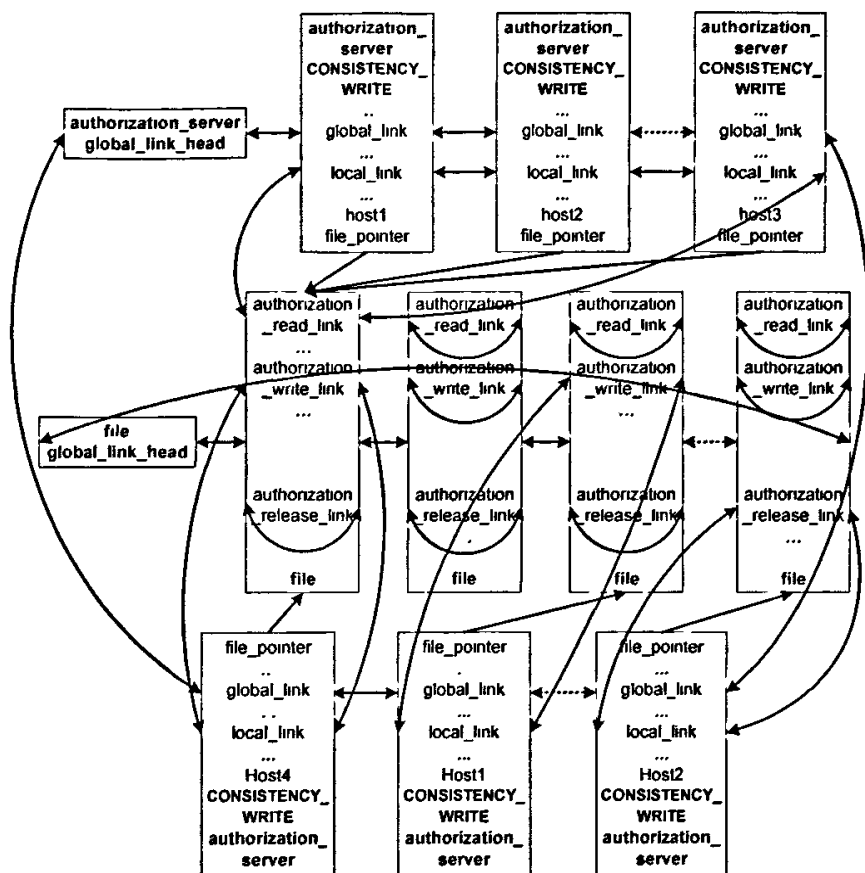


图 5.6 元数据服务器节点授权记录逻辑视图

5.2.7 授权申请

蓝鲸分布式文件系统的授权申请策略以提高系统并发度为目标，采用按需申请的方式，即仅在相关操作的调用点处根据操作需要进行授权申请，如读授权申请发生在读操作调用处，而不是发生在读操作对应的文件打开(open)操作处，这一点与 NFSv4 系统不同，NFSv4 系统使用的是打开授权（系统根据 open 操作模式决定授权类型）模式。蓝鲸分布式文件系统的授权机制使用按需申请而不是打开申请的原因如下：

- 1) 某些可能导致蓝鲸分布式文件系统数据一致性问题的操作并不对应打开操作，如 unlink 操作。
- 2) 相对于按需申请而言，打开申请不利于授权撤销，打开操作之后往往对应一系列的读写操作，在关闭操作之前相应授权是否可以撤销？如果不允许，那么系统并发度会受到很大影响，如果允许，则授权撤销后的读写操作还要涉及授权申请（此时无打开操作）的问题。
- 3) 打开申请可能导致授权冗余：文件打开模式与其后的数据读写操作并非精确对应，存在文件打开后未进行读写操作即关闭的情况，或者对以读写模式(rw)打开

的文件只进行读操作的情况；相对而言按需申请在相关操作调用处更清楚是否需要授权以及需要何种类型授权。

授权申请的请求协议定义如图 5.7 所示：

```
struct authorization_apply_request {
    unsigned long    consistency_set_time;
    unsigned long    client_identifier;
    unsigned long    file_identifier;
    unsigned long    authorization_type;
};
```

图 5.7 授权申请的请求协议

其中 consistency_set_time (Consistency Set Time) 是系统数据一致性语义的调整时间，应用节点的授权申请（及后文中的心跳申请）携带此标识作为认证信息，后继章节有针对 Consistency Set Time 的详细介绍。

授权申请的应答协议定义如图 5.8 所示：

```
struct authorization_apply_reply {
    int    status;
    union {
        struct {
            unsigned long    file_identifier;
            unsigned long    authorization_type;
            unsigned long    consistency_type;
            unsigned long    consistency_set_time;
        } apply_succ;
        struct {
            unsigned long    wait_time;
            unsigned long    reserved[3];
        } apply_try;
        struct {
            unsigned long    unused[4];
        } apply_failed;
    } apply_rep;
};
```

图 5.8 授权申请的应答协议

实际上写授权含盖的权限要大于读授权，拥有写授权的应用节点可以对相应文件执行读操作而不必再次申请读授权，反之则不成立。在持有读授权的基础上申请写授权称为授权升级，应用节点的授权申请包含授权升级的过程。如果应用节点已经持有某文件的读授权，当其申请该文件的写授权成功时，那么原先的读授权便升级为写授权，系统不再为之维持先前的读授权记录。

后文将对授权申请的处理流程作详细介绍。

5.2.8 授权释放

蓝鲸分布式文件系统的授权释放策略在不影响系统并发度的前提下以尽量降低系统开销（包括释放操作本身的处理开销、相关的文件属性更新开销、潜在的再次授权申请开销等）为目的。针对缓存类和非缓存类两类授权，蓝鲸分布式文件系统采取的释放策略有很大差异：对于非缓存类授权，其授权在相应操作结束后自动失效不能重复使用，是一种隐式释放方式；相对而言缓存类授权的释放比较复杂，主要包括主动释放、剥夺释放和故障释放三种方式。

✧ 主动释放

所谓主动释放指应用节点不再需要相关授权而主动向元数据服务器节点进行释放，主要发生在应用节点进程关闭目标文件的最后一个引用句柄（或文件描述符）时。主动释放前系统往往已经完成脏数据写回及文件属性更新工作（针对写授权），因此主动释放是缓存类授权释放方式中开销最小的一种。

授权主动释放的请求协议定义如图 5.9 所示：

```

1 struct authorization_release_request {
2     unsigned long    client_identifier;
3     unsigned long    file_identifier;
4     unsigned long    authorization_type;
5 };

```

图 5.9 授权主动释放的请求协议

授权主动释放的应答协议定义如图 5.10 所示：

```

1 struct authorization_release_reply {
2     int    status;
3 };

```

图 5.10 授权主动释放的应答协议

✧ 剥夺释放

蓝鲸分布式文件系统的授权是可剥夺的，为了提高系统并发度，元数据服务器节点可以根据授权的需要将先前批准的某些授权撤销，以满足新的授权申请。

应用节点收到元数据服务器节点的授权撤销命令后，如果对应读写操作尚未结束，则继续执行直到本次读写操作完成，然后写回脏数据，更新文件属性，交还授权。允许授权剥夺可能导致多个以非兼容共享方式操作的应用节点之间彼此争夺授权的 ping-pang 问题，该问题对系统的整体性能造成负面影响。为此元数据服务器节点记录授权时间，并规定授权的最短生存期，除非有授权的主动释放或隐式释放，否则那些处于最短生存期内的授权不予剥夺。

授权剥夺的请求协议定义如图 5.11 所示：

```

struct authorization_revoke_request {
    unsigned long    file_identifier;
    unsigned long    authorization_type;
};

```

图 5.11 授权剥夺的请求协议

授权主动释放的应答协议定义如图 5.12 所示:

```

struct authorization_revoke_reply {
    int    status;
};

```

图 5.12 授权剥夺的应答协议

✧ 故障释放

故障释放分为网络分割类释放和元数据服务器节点宕机类释放。

➤ 网络分割类释放

如果在元数据服务器节点进行权限撤销期间发生网络分割,应用节点将无法接收到元数据服务器节点的授权撤销命令,元数据服务器节点在授权撤销处理超时后强制剥夺相关授权并作异常记录;网络分割恢复后,应用节点将被告知相关授权已被强制剥夺,此时应用节点须丢弃无效授权。

➤ 元数据服务器节点宕机类释放

元数据服务器节点宕机会导致系统中所有授权信息丢失,为防止元数据服务器节点重启后向其他应用节点进行冲突授权,应用节点须将先前获取的授权全部释放。

故障释放是所有释放方式中开销最大的一种,需要尽量避免。详细讨论参见后文模型异常处理的相关部分。

5.2.9 授权处理流程

本节将针对系统正常运行状态下的授权处理流程进行详细讨论,包括应用节点的授权申请、元数据服务器节点的授权处理和应用节点的授权剥夺等三个部分。关于系统异常状态(如网络分割、节点宕机等)下的授权处理流程见后文模型异常处理的相关部分。

5.2.9.1 应用节点的授权申请流程

图 5.13 给出应用节点带授权申请的操作流程,图中数字用于流程顺序描述。

- 1) 应用节点系统执行授权相关操作调用,转到 2);
- 2) 系统检测该操作是否为缓存类授权相关操作,若是转到 3),否则转到 4);
- 3) 对于缓存类授权相关操作先在应用节点本地检测是否已持有对目标文件进行相关操作的有效授权,若有转到 6),否则转到 4);

- 4) 应用节点根据操作类型向元数据服务器节点进行相应授权申请, 转到 5);
- 5) 应用节点根据元数据服务器节点对授权申请的应答判断授权申请是否成功, 若成功转到 6), 否则转到 7);
- 6) 应用节点成功持有对目标文件进行相关操作的有效授权, 执行相关操作, 转到 9) (与操作执行是否成功无关);
- 7) 授权申请失败, 应用节点根据元数据服务器节点的返回信息进行必要的出错处理, 转到 8);
- 8) 元数据服务器节点的返回信息决定是否需要重试授权申请, 如需要则在适当的等待后转到 4), 否则未能获取相关授权, 操作不能执行, 转到 9);
- 9) 操作退出。

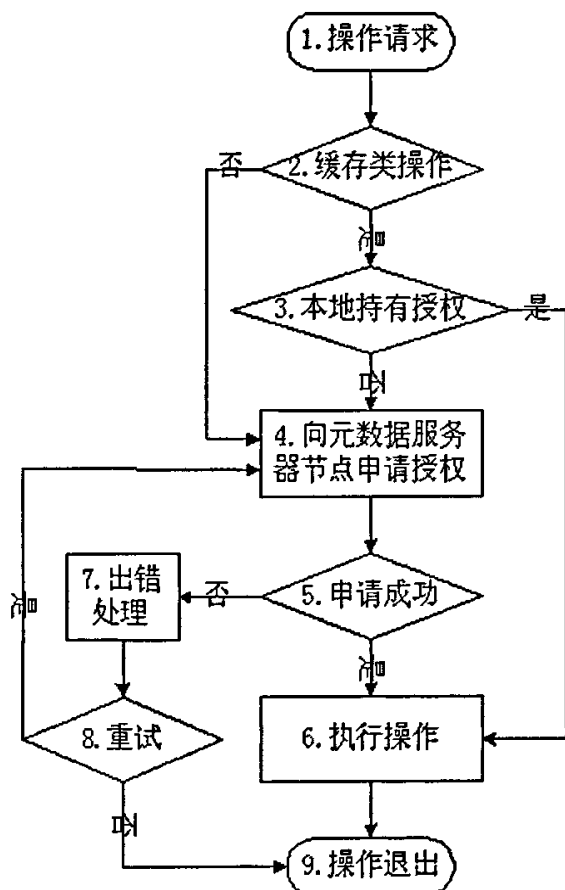


图 5.13 应用节点授权申请流程图

5.2.9.2 元数据服务器节点的授权处理流程

图 5.14 给出元数据服务器节点对应用节点授权申请的处理流程, 图中数字用于流程顺序描述。

- 1) 元数据服务器节点收到应用节点的授权申请, 转到 2);

- 2) 元数据服务器节点根据授权申请类型进行授权兼容性检测, 转到 3);
- 3) 如果没有不兼容授权, 转到 8); 否则转到 4);
- 4) 如果已有的不兼容授权为非缓存类授权则转到 6), 否则转到 5);
- 5) 元数据服务器节点向不兼容缓存类授权的持有节点发出授权撤销命令, 转到 7);
- 6) 对已有的不兼容的非缓存类授权须等待, 直到相关操作执行结束, 相应的不兼容授权自动失效, 转到 8);
- 7) 元数据服务器节点检测授权撤销是否成功, 如成功转到 8), 否则不能满足应用节点的授权申请, 转到 9);
- 8) 系统中目前没有与授权申请类型不兼容的授权, 元数据服务器节点进行相应的授权, 转到 9);
- 9) 授权处理结束返回。

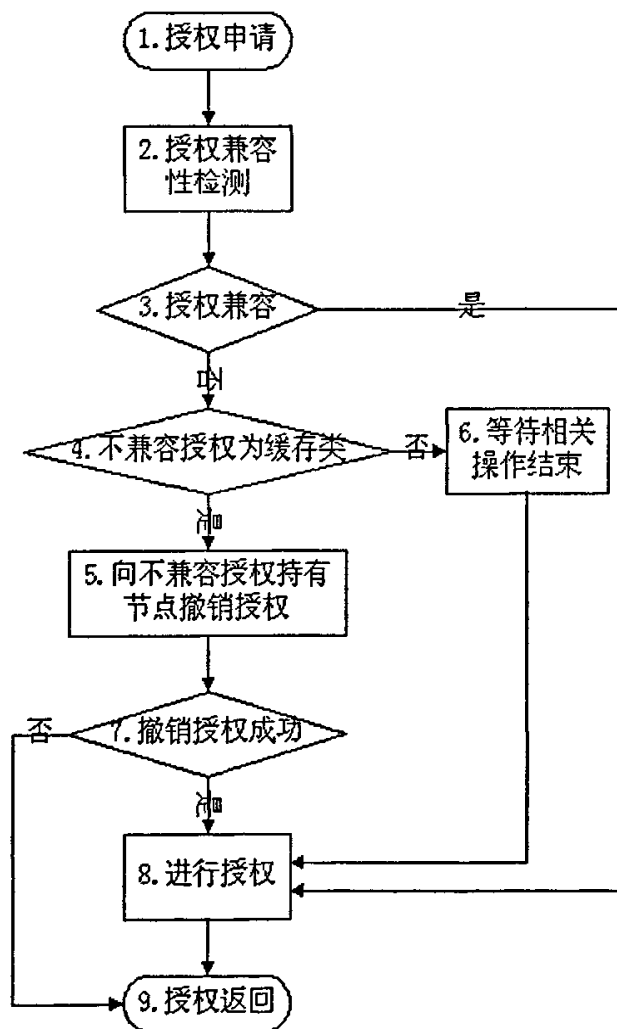


图 5.14 元数据服务器节点授权处理流程图

5.2.9.3 应用节点的授权剥夺流程

图 5.15 给出应用节点响应元数据服务器节点的授权撤销命令的处理流程，图中数字用于流程顺序描述。

- 1) 应用节点收到元数据服务器节点的授权撤销命令，转到 2)；
- 2) 应用节点检测目标授权是否正在使用，如在用转到 3)，否则转到 4)；
- 3) 系统等待目标授权对应的操作完成，转到 4)；
- 4) 应用节点在释放目标授权之前，先把授权所属文件对应的脏数据写回，并更新文件属性（此处是蓝鲸分布式文件系统文件属性被动更新点），转到 5)；
- 5) 目标授权已空闲，可以释放，向元数据服务器节点返回。

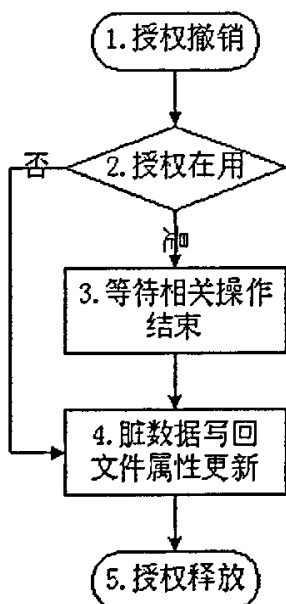


图 5.15 应用节点授权剥夺处理流程图

5.2.9.4 流程综述

综合上述三个部分，可以得到蓝鲸分布式文件系统数据一致性语义模型授权机制的整体处理流程，如图 5.16 所示：

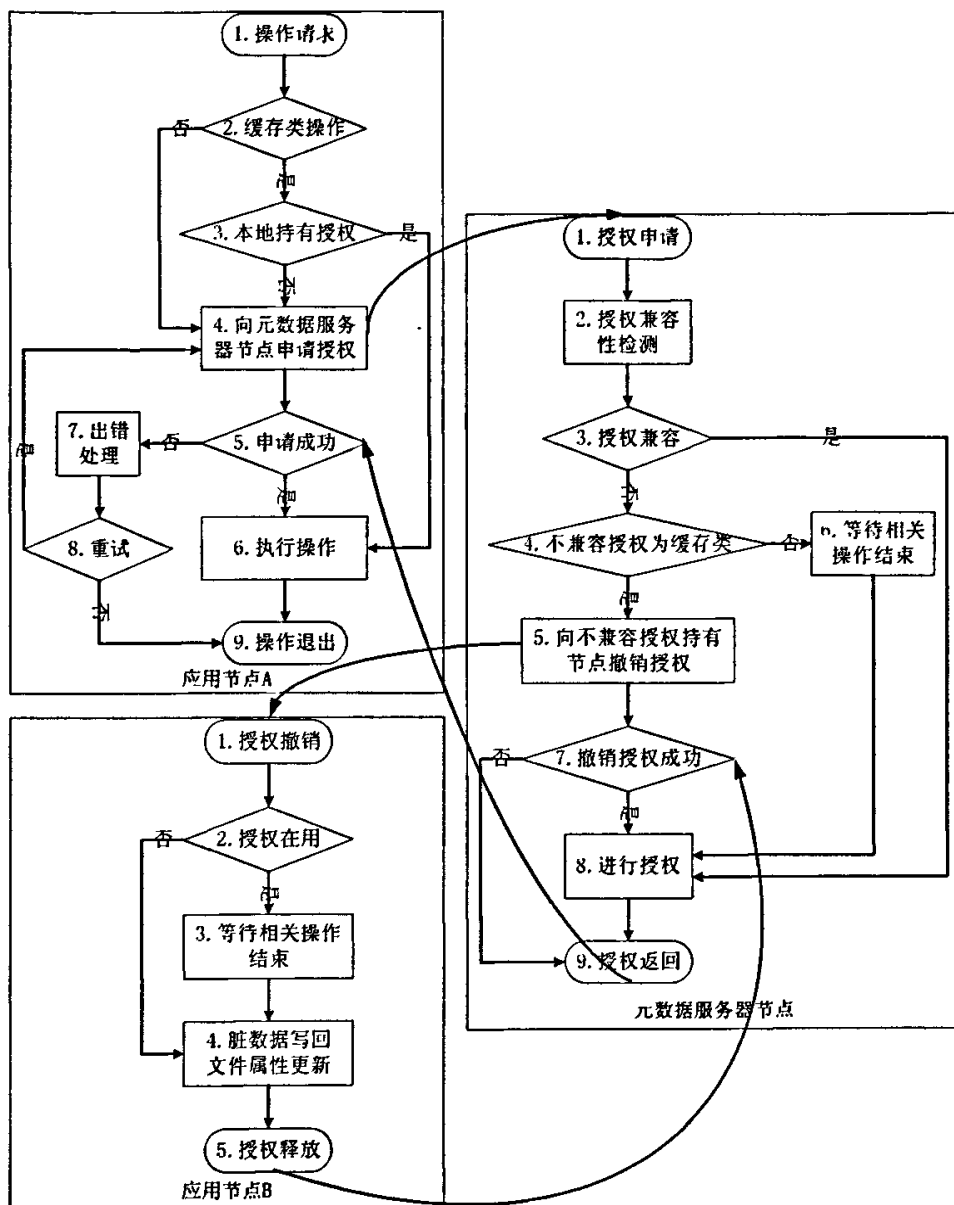


图 5.16 系统数据一致性语义模型授权机制的整体处理流程图

5.3 心跳机制

蓝鲸分布式文件系统数据一致性语义模型的心跳机制给应用节点和元数据服务器节点之间提供了一个定期交互的信息通道，同时也为模型的授权机制设定了一种超时——心跳周期。在每个心跳周期内，应用节点主动向元数据服务器节点发起心跳请求，以表

明自己的状态；元数据服务器节点被动接收应用节点的心跳请求，并把系统的某些信息通过心跳应答返回给应用节点。

心跳机制在蓝鲸分布式文件系统数据一致性语义模型中的主要作用如下：

- 1) 元数据服务器节点判定授权撤销操作超时的度量。如果授权撤销操作在一个心跳周期内不能完成，则元数据服务器节点将强制剥夺相应授权，并作异常记录。
- 2) 元数据服务器节点判断应用节点“可达”的手段。如果出现网络分割或者应用节点宕机，且这种故障不能在一个心跳周期内恢复，那么元数据服务器节点将认为对应节点不可达，在其恢复之前，与之相关的授权撤销操作将直接采用强制剥夺的方式。
- 3) 元数据服务器节点向应用节点通报事件的通道，如元数据服务器节点重启事件、授权在应用节点“不可达”期间被强制剥夺事件、系统数据一致性语义调整事件以及后文将要讨论的系统数据一致性安全策略等。

由此可见，蓝鲸分布式文件系统的心跳机制不仅是节点之间相互表明“活跃”的手段，也是通告系统事件的信息通道，因此不再是通常意义上的心跳机制，而是蓝鲸分布式文件系统数据一致性语义模型所特有的定时事件通道。

5.3.1 心跳请求

心跳请求协议定义如图 5.17 所示：

```

1 struct heartbeat_request {
2     unsigned long    client_identifier;
3     unsigned long    server_identifier;
4     unsigned long    consistency_set_time;
5 };

```

图 5.17 心跳请求协议

Server Identifier 是元数据服务器节点的启动标识，Consistency Set Time 是系统数据一致性语义的调整时间，应用节点心跳请求携带这两个标识信息用于表明自身状态。应用节点在首次心跳请求中将这两个标识设置为零，并通过心跳应答从元数据服务器节点获取相应标识的最新值，然后将其缓存在本地；在以后的心跳请求中，应用节点将使用本地缓存的标识，并根据心跳应答对其进行更新。

5.3.2 心跳应答

心跳应答协议定义如图 5.18 所示：

```

struct heartbeat_reply {
    int status;
    union {
        struct {
            unsigned long unused[4];
        } normal;
        struct {
            unsigned long server_identifier;
            unsigned long consistency_type;
            unsigned long consistency_set_time;
            unsigned long pad[1];
        } client_init;
        struct {
            unsigned long server_identifier;
            unsigned long consistency_type;
            unsigned long consistency_set_time;
            unsigned long security_policy;
        } server_init;
        struct {
            unsigned long server_identifier;
            unsigned long consistency_type;
            unsigned long consistency_set_time;
            unsigned long security_policy;
        } server_init_partition;
        struct {
            unsigned long file_identifier;
            unsigned long consistency_type;
            unsigned long consistency_set_time;
            unsigned long security_policy;
        } authorization_force_revoke;
        struct {
            unsigned long consistency_type;
            unsigned long consistency_set_time;
            unsigned long security_policy;
            unsigned long pad[1];
        } consistency_adjust;
        struct {
            unsigned long unused[4];
        } others;
    } heartbeat_rep;
};

```

图 5.18 心跳应答协议

心跳应答协议比较复杂,主要用于系统异常处理及数据一致性语义调整,相关讨论留待后继章节进行。

5.4 模型的数据一致性安全策略

元数据服务器节点处理授权申请时,对不兼容授权要向其持有者节点发出授权撤销命令,如果其间发生网络分割,那么相应的应用节点将无法接收授权撤销命令,于是在撤销操作超时之后,元数据服务器节点强制剥夺该授权并作异常记录,以防止单个应用节点异常造成系统“死等”问题。当网络分割恢复之后,元数据服务器节点将通过心跳应答通告该应用节点相应授权已被强制剥夺,此处涉及应用节点如何处理相关文件问题,如果不加以适当控制,可能出现授权被剥夺的应用节点非法访问数据问题(读操作可能使用过时数据,违反 cache 一致性;写操作可能破坏数据安全性或数据完整性)。

针对上述问题,蓝鲸分布式文件系统数据一致性语义模型提供数据一致性安全策略,元数据服务器节点通过心跳应答将授权被强制剥夺事件连同相关的数据一致性安全策略一起通告给应用节点,应用节点据此处理相应文件。数据一致性安全策略提供如下几种处理方式:

方式一:清空本地 cache (包括脏数据),清空驱动层缓冲数据,系统挂起,等待系统管理员处理。对网络分割采取同节点宕机的处理方式,以牺牲单个应用节点的数据为代价换取系统整体的安全性及可用性,是一种保守的处理方式。

方式二:清空本地 cache (包括脏数据),清空驱动层缓冲数据,系统继续运行。该方式同样以牺牲单个应用节点的数据为代价换取系统整体的安全性及可用性,也是一种保守的处理方式,但减少了人工干预。

方式三:清空授权被强制剥夺的文件所对应的本地 cache (包括脏数据),其他文件不受影响,系统继续运行。该方式不处理驱动层数据,是一种比较冒险的处理方式。

方式四:这种处理方式主要用于数据一致性语义调整的历史授权处理,以及元数据服务器节点宕机所导致的授权清理等情况。该方式下,应用节点将 cache 中所有的脏数据写回,并进行相应的文件属性更新,然后释放本节点持有的所有授权。

5.5 数据一致性语义的在线调整

5.5.1 在线调整的必要性

前面的章节中已经分析过,系统为应用提供的数据一致性语义支持并非越严格越好,数据一致性的维护是有代价的,越是严格的数据一致性其维护的代价越高,系统没必要花费代价去维护并不需要的数据一致性,而是只需提供应用所必须的数据一致性语义支持即可,过犹不及。

蓝鲸分布式文件系统基于授权机制的数据一致性语义模型可以提供多种不同强度的

数据一致性语义，以期更好地满足不同模式应用的需要。这种多语义特性需要相应的语义调整（或语义切换）机制加以支持，总的来讲可以有离线调整和在线调整两种方式。

所谓离线调整指蓝鲸分布式文件系统的数据一致性语义类型在系统加载或启动时通过特定的参数（或默认参数）设定，系统运行过程中其数据一致性语义始终保持不变，直到蓝鲸分布式文件系统被重新加载或启动时再次设定。这种数据一致性语义的调整方式比较简单，但缺乏灵活性，数据一致性语义的调整会导致一段时间（对于大型的分布式文件系统而言，系统重新加载或启动的时间通常都在分钟量级）的文件服务中断，对系统的影响比较大。蓝鲸分布式文件系统作为蓝鲸网络存储系统的核心组件，旨在面向大规模应用服务器机群提供高速的、并发的、共享的、高可用的远程文件访问服务，在任意时刻系统中可能有多种应用在同时运行，有些可能要求提供 7×24 小时的不间断服务，有些大规模作业可能需要连续运行数月，这种情况下，不论是贸然停止所有应用立即进行数据一致性语义调整，还是等待所有应用都运行结束后再进行数据一致性语义调整都是不可行的。

在线调整可以解决离线调整所带来的应用停止及文件服务中断等问题，允许用户根据实际需要动态调整蓝鲸分布式文件系统的数据一致性语义，这种调整不需要应用停止，也不需要蓝鲸分布式文件系统重新加载或启动，系统在应用透明的方式下完成数据一致性语义调整，满足蓝鲸分布式文件系统高可用的目的。

5.5.2 Consistency Set Time

Consistency Set Time 是元数据服务器节点进行数据一致性语义设置（或调整）的当前时间，元数据服务器节点记录该时间主要用于处理网络分割期间发生数据一致性语义回调（类似于 $A \rightarrow B \rightarrow \dots \rightarrow A$ 的调整模式）的情况，相关讨论参见后文数据一致性语义调整的异常处理部分。Consistency Set Time 初始取值为元数据服务器节点的启动时间，应用节点通过心跳应答获取该值，并在应用节点本地缓存（应用节点本地缓存的 Consistency Set Time 初始取值为零）。通过比较应用节点缓存的 Consistency Set Time 和元数据服务器节点当前的 Consistency Set Time，系统能够判断哪些应用节点需要响应系统的数据一致性语义调整。

5.5.3 调整机制

如前所述，蓝鲸分布式文件系统的数据一致性语义依赖于其数据一致性语义模型的核心机制——授权机制，不同的授权规则对应不同的数据一致性语义，蓝鲸分布式文件系统数据一致性语义的调整通过该模型授权规则的调整实现。在蓝鲸分布式文件系统中，与数据一致性语义调整相关的工作主要包括下述几个方面：

1) 授权规则调整

这是蓝鲸分布式文件系统数据一致性语义调整的核心工作，也是整个调整工

作的第一步。就工作量而言,这一步工作比较简单,仅涉及几种数据一致性语义所对应的授权兼容表的切换,使得系统当前的授权兼容表指针指向调整后的数据一致性语义所对应的授权兼容表。完成这一步工作后新的数据一致性语义就开始生效。

2) 语义调整导致的原有授权冲突处理

不同的数据一致性语义对应不同的授权规则,在弱的数据一致性语义下兼容的授权在强的数据一致性语义下可能是不兼容的。对于那些已经存在的但在新的数据一致性语义下不兼容的授权需要加以处理,以维护整个系统授权规则的完整性,这是整个调整工作的难点。对此元数据服务器节点采取“按需”方式加以处理,即不主动处理原有授权,当新的授权申请与原有授权不兼容时再通过授权撤销的方式加以剥夺;另一方面通过比较应用节点缓存的 Consistency Set Time 和元数据服务器节点当前的 Consistency Set Time,系统可以发现应用节点与元数据服务器节点间的数据一致性语义差异,元数据服务器节点将通过心跳应答把数据一致性语义调整事件并关联方式四的数据一致性安全策略一起通告给应用节点,应用节点据此处理本节点持有的历史授权。

3) 应用节点缓存策略调整

随着系统数据一致性语义的调整,应用节点的缓存策略也要随之发生变化。该工作同样通过心跳机制的异步通告方式完成:应用节点收到数据一致性语义调整的相关通告后,调整本地缓存策略,使得数据缓存策略与新的数据一致性语义相对应,清空过时的数据缓存,保持必要的数据刷新,维护适当的 cache 一致性。

由此可见蓝鲸分布式文件系统数据一致性语义调整是一个异步调整过程,整个调整过程在一个心跳周期内完成,即新的数据一致性语义可以马上生效,但原有数据一致性语义的影响可能要持续一个心跳周期才能完全消失。

5.5.4 调整模式

蓝鲸分布式文件系统数据一致性语义的调整存在语义强弱的调整方向问题,以此为依据可以分为降级调整(从强的数据一致性语义向弱的数据一致性语义调整)和升级调整(从弱的数据一致性语义向强的数据一致性语义调整)两种模式。

降级调整模式比较简单,因为蓝鲸分布式文件系统数据一致性语义模型所支持的多种数据一致性语义是向弱兼容的,从前面的授权兼容表可以看出,强的数据一致性语义所对应的授权兼容表中的兼容项在弱的数据一致性语义所对应的授权兼容表中依然是兼容项,即在强的数据一致性语义下的兼容授权在弱的数据一致性语义下也是兼容授权,所以降级调整模式中不存在原有兼容授权在调整后的数据一致性语义下变成不兼容授权的情况,不涉及语义调整导致的原有授权冲突处理,即上节中的工作 2) 可以省略。另一方面,应用节点的数据缓存策略需要调整,数据一致性语义的降级调整使得应用节

点之间的隐式协作规则变得更加宽松,应用节点可能需要通过更加积极主动的检测降低 cache 一致性问题所带来的风险。

相比之下,升级调整模式要复杂一些,因为弱的数据一致性语义不保证对强的数据一致性语义的兼容性,从弱的数据一致性语义向强的数据一致性语义的调整可能导致原有兼容授权在新的数据一致性语义下变成不兼容授权的情况,所以升级调整模式需要处理降级调整模式所不必处理的语义调整导致的原有授权冲突。另一方面,应用节点的缓存策略调整需要完成本地 cache 的清空工作,因为应用节点本地 cache 中的原有数据可能不满足新的数据一致性语义下 cache 一致性要求,这是升级调整模式较降级调整模式所特有的工作。

5.6 模型异常处理

在蓝鲸分布式文件系统这样的多组件(包括应用节点、元数据服务器节点、存储节点、绑定服务器节点、管理服务器节点以及节点之间的连接等)分布式环境中,任何一个组件的异常都可能对系统的整体运行造成影响,尤其是蓝鲸分布式文件系统数据一致性语义模型引入的应用节点间的隐式协作关系将单点异常进一步扩大。蓝鲸分布式文件系统针对个别组件异常提供适当的容错功能,允许系统从部分组件的异常甚至失效中加以恢复。本节主要针对与蓝鲸分布式文件系统数据一致性语义模型相关的异常处理加以讨论,包括网络分割、应用节点宕机、元数据服务器节点宕机等情况。

5.6.1 授权强制剥夺异常记录

在蓝鲸分布式文件系统的数据一致性语义模型中,对授权的强制剥夺需要进行异常记录,该记录用于将来某个时刻通告相关应用节点授权已被强制剥夺,以防止应用节点误以为持有相关授权而进行无授权数据访问。元数据服务器节点对这种异常记录作持久存储,原因如下:

- 1) 授权强制剥夺异常记录对系统异常恢复很重要,如果不作持久存储,那么元数据服务器节点宕机将导致应用节点和元数据服务器节点都不知道哪些授权是曾经被强制剥夺过的,给系统的故障恢复带来很大问题。
- 2) 与授权申请不同,授权强制剥夺是小概率事件,对授权强制剥夺异常记录作持久存储既不会占用太多的持久存储空间,也不会对系统性能造成很大影响。

每次元数据服务器节点启动后自动整理持久存储中的授权强制剥夺异常记录,并获取尚未通告给应用节点的授权强制剥夺异常记录,具体处理参见后文中关于元数据服务器节点宕机异常处理部分。

5.6.2 Server Identifier

Server Identifier 是蓝鲸分布式文件系统元数据服务器节点的启动标识,该标识随元

数据服务器节点重启而变化,并保持历史唯一性,一个可行的实现方案是使用元数据服务器节点启动的当前时间(精确到秒),这样就得到一个随元数据服务器节点重启而逐次递增的 Server Identifier。系统中的应用节点可以在节点本地缓存 Server Identifier,如果缓存的 Server Identifier 与元数据服务器节点当前的 Server Identifier 不一致,说明其间发生过元数据服务器节点重启事件,系统据此可以做出相应处理。关于 Server Identifier 的使用将在下文相关章节中详细讨论。

5.6.3 过渡期

过渡期指蓝鲸分布式文件系统元数据服务器节点重启后允许应用节点访问的最初一段特定的时间,该时间段内元数据服务器节点不处理任何应用节点的授权申请,防止新的授权与历史授权产生冲突。过渡期长于心跳周期,确保在过渡期内每个活跃的应用节点至少有一次与元数据服务器节点进行心跳交互的机会,元数据服务器节点通过心跳应答通告应用节点发生元数据服务器节点重启事件,应用节点据此作相应处理。

5.6.4 网络分割处理

在蓝鲸分布式文件系统中,网络分割是一种比较常见的网络故障,网络分割可能导致应用节点与其他所有节点都不可见,也可能导致应用节点仅与元数据服务器节点不可见,或者导致应用节点与除元数据服务器节点外的所有其他节点不可见,本节的讨论针对应用节点与元数据服务器节点不可见的网络分割情况。

如果网络分割期间没有其他应用节点的授权申请与本节点持有的授权产生冲突,那么当网络分割恢复后,前述的蓝鲸分布式文件系统基于软件的网络容错技术会自动恢复网络的逻辑连接通道,进行脏数据写回及文件属性更新,不会给系统带来数据损失。

如果网络分割期间发生其他应用节点的授权申请与本节点持有的授权产生冲突的情况,那么系统必须加以特殊处理,以防止授权撤销操作陷入“死等”以及持有过时授权的应用节点非法访问数据等问题的发生。处理过程如下:

- 1) 元数据服务器节点执行授权撤销操作时,如果在系统的心跳周期内不能得到应答,则表明要么出现网络分割,要么授权持有者节点发生异常,系统强制剥夺该授权,并做异常记录,对于写授权强制剥夺关联方式二的数据一致性安全策略,对于读授权强制剥夺关联方式三的数据一致性安全策略;
- 2) 由于网络分割时间长于系统的心跳周期(对于网络分割时间短于心跳周期的情况,元数据服务器节点的授权撤销操作能够处理),应用节点在此期间至少有一次与元数据服务器节点进行心跳交互的机会,受网络分割影响,心跳交互会失败,应用节点将不停的尝试,其间的访问数据暂停;
- 3) 系统网络分割恢复后,元数据服务器节点通过心跳应答将授权被强制剥夺事件连同相关的数据一致性安全策略一起通告给应用节点,在应用节点对此进行相应处

理之前，元数据服务器节点拒绝处理该应用节点的授权申请；

- 4) 应用节点根据心跳应答信息（包括数据一致性安全策略）对系统进行相应处理，处理结束之后通知元数据服务器节点；
- 5) 对于网络分割长期不能恢复的情况，如果没有其他应用节点的授权申请与该应用节点所持有的授权产生冲突，那么系统中就会留下很多长时间不使用的授权，下文将要讨论的应用节点宕机问题使得该问题更加严重，为此元数据服务器节点跟踪应用节点的心跳情况，在内存压力情况下，系统将默认那些失去心跳超过一定时间的应用节点无法恢复，对其作应用节点宕机情况处理，释放由其持有的所有授权，作异常记录，关联方式一的数据一致性安全策略；
- 6) 系统从长时间的网路分割中恢复后，会通过心跳与元数据服务器节点交互，可能被通告已被作为应用节点宕机情况处理，则应用节点将遵照心跳应答中的数据一致性安全策略进行相应处理。

5.6.5 应用节点宕机处理

应用节点对授权信息不作持久存储，应用节点宕机会导致本节点授权记录丢失，应用节点重启之后对授权信息进行恢复是不必要的（恢复的代价大于需要时再申请的代价），也无法通过主动释放方式将元数据服务器节点上对应的授权释放，如果其他应用节点的授权申请没有与该应用节点宕机前持有的授权发生冲突，则元数据服务器节点会保持很多无用的授权记录，造成授权“流失”，应用节点宕机事件的积累会将上述问题扩大。为此蓝鲸分布式文件系统试图通过简单的方式通知元数据服务器节点发生应用节点宕机事件，然后由元数据服务器节点自动清除该节点原先持有的无用授权。处理过程如下：

- 1) 元数据服务器节点针对宕机后尚未完成重启的应用节点执行授权撤销操作的情况类似于网络分割，如果在系统的心跳周期内不能得到应答，则系统强制剥夺该授权，并做异常记录，对于写授权强制剥夺关联方式二的数据一致性安全策略，对于读授权强制剥夺关联方式三的数据一致性安全策略；
- 2) 应用节点缓存元数据服务器节点的 Server Identifier，该值初始化为零，并在系统启动后通过心跳交互获取元数据服务器节点当前 Server Identifier，应用节点的心跳请求携带本节点缓存的 Server Identifier 作为认证信息；
- 3) 元数据服务器节点在处理心跳请求前检测相应的 Server Identifier 认证信息，如果该值为零，表明应用节点重启，元数据服务器节点将释放系统中所有由该应用节点持有的无用授权，并清除与该应用节点相关的授权被强制剥夺的异常记录；
- 4) 对于应用节点宕机之后长时间不重启的情况类似于网络分割长时间不恢复的情况，元数据服务器节点在内存压力情况下，默认那些失去心跳超过一定时间的应用节点无法恢复，释放由其持有的所有授权，作异常记录，关联方式一的数据一

致性安全策略。

5.6.6 元数据服务器节点宕机处理

元数据服务器节点对授权信息也不作持久存储,元数据服务器节点宕机会导致系统中所有授权记录丢失,元数据服务器节点重启后如果未对此作适当处理,则新的授权与历史授权可能产生冲突,造成不兼容授权,从而破坏系统的数据一致性。尽管通过类似于 NLM 协议的锁恢复机制从各应用节点对元数据服务器节点的授权信息进行恢复是有可能的,但考虑到元数据服务器节点宕机的随机性以及蓝鲸分布式文件系统数据一致性语义的在线可调整性,元数据服务器节点宕机时系统可能处于多数据一致性语义状态,授权恢复的复杂度及代价比较高,所以蓝鲸分布式文件系统采取更为简单的方式加以处理。处理过程如下:

- 1) 元数据服务器节点宕机重启后,在允许应用节点访问之前先处理保存在持久存储中的关于授权强制剥夺的异常记录,将那些已经通告给应用节点的异常记录删除,尚未通告给应用节点的保留并载入内存;
- 2) 在元数据服务器节点重启后可以处理新的授权申请前须要经历过渡期阶段,由于过渡期长于心跳周期,未发生网络分割的应用节点在此期间至少有一次与元数据服务器节点进行心跳交互的机会,元数据服务器节点将元数据服务器节点宕机重启事件并关联适当的数据一致性安全策略(宕机前针对相应应用节点有写授权强制剥夺的关联方式二,只有读授权强制剥夺的关联方式三,无授权强制剥夺的关联方式四)通过心跳应答通告给应用节点;
- 3) 如果某些应用节点在元数据服务器节点的过渡期内处于网络分割状态,那么元数据服务器节点将对这些应用节点作宕机情况处理,忽略这些应用节点的历史授权,即使有新的授权申请与历史授权发生冲突;
- 4) 应用节点的网络分割在元数据服务器节点过渡期结束后恢复,通过心跳与元数据服务器节点交互,元数据服务器节点在处理心跳请求前检测相应的 Server Identifier 认证信息,如果该值为不同于元数据服务器节点当前 Server Identifier 的非零值,表明在元数据服务器节点的过渡期内该应用节点未能及时释放历史授权,可能已发生与其他新的授权相冲突的情况,元数据服务器节点将元数据服务器节点宕机重启事件并关联方式一的数据一致性安全策略通过心跳应答通告给该应用节点;
- 5) 应用节点通过心跳应答得知系统发生元数据服务器节点宕机重启事件,遵照心跳应答中的数据一致性安全策略进行相应处理。

5.6.7 数据一致性语义调整的异常处理

如前所述,蓝鲸分布式文件系统数据一致性语义调整是一个异步调整过程,在此期

间可能发生系统异常。本节将针对蓝鲸分布式文件系统数据一致性语义调整的异常处理机制进行讨论,涉及网络分割、应用节点宕机、元数据服务器节点宕机等几种异常情况。

应用节点宕机的情况比较简单,发生宕机的应用节点所持有的授权全部失效,对系统的数据一致性语义调整不造成额外影响,前面关于应用节点宕机的异常处理已经覆盖数据一致性语义调整期间发生应用节点宕机的情况。

由于对元数据服务器节点宕机采取的是一种授权复位而非授权恢复的策略,所以元数据服务器节点宕机对数据一致性语义调整的影响仅限于在元数据服务器节点宕机重启后重新进行数据一致性语义设置(或调整),其他处理同前述的元数据服务器节点宕机处理。

网络分割对数据一致性语义调整的影响相对较大,如果数据一致性语义调整期间发生网络分割,则相关应用节点不能及时通过心跳交互得知数据一致性语义的调整,也不能对在原数据一致性语义下获取的授权进行处理,于是可能产生一些问题。考虑如下场景:

- 1) 应用节点 A 在读写一致性语义下获取文件 FILE 的读授权 M;
- 2) 发生网络分割,应用节点 A 无法与元数据服务器节点进行心跳交互;
- 3) 系统的数据一致性语义被调整为写一致性语义;
- 4) 应用节点 B 在写一致性语义下申请文件 FILE 的写授权 N,此时授权 N 不会与授权 M 产生冲突,因此授权 M 不会被强制剥夺;
- 5) 应用节点 B 持授权 N 对 FILE 进行修改,然后释放授权 N;
- 6) 在网络分割恢复前系统的数据一致性语义再次被调整为读写一致性语义,即发生数据一致性语义回调;
- 7) 网络分割恢复,根据读写一致性语义规定,如果授权 M 没有被剥夺,则应用节点 A 认为本地 cache 依然有效,于是可能产生与读写一致性语义相悖的 cache 一致性问题。

蓝鲸分布式文件系统数据一致性语义模型须提供适当的机制处理上述情况,单纯比较应用节点与元数据服务器节点的数据一致性语义类型是不行的,这种机制不能处理网络分割期间的数据一致性语义回调问题,前述的 Consistency Set Time 机制为此而设立。通过比较应用节点缓存的 Consistency Set Time 和元数据服务器节点当前的 Consistency Set Time,系统能够准确判断哪些应用节点需要响应系统的数据一致性语义调整——不管期间发生过多少次数据一致性语义调整,也不管是否发生数据一致性语义回调。

与之相关的另外一个问题是网络分割恢复后的数据一致性安全策略问题,蓝鲸分布式文件系统采取如下策略:如果网络分割期间其他授权申请与被分割节点持有的授权产生冲突,那么按照授权强制剥夺的方式处理;否则采用方式四的数据一致性安全策略。该策略看似比较冒险,但是其可能带来的风险都是数据一致性语义调整过程中相关语义所能容忍的风险。

5.6.8 小结

本节主要介绍了蓝鲸分布式文件系统数据一致性语义模型的异常处理机制，包括网络分割处理、应用节点宕机处理、元数据服务器节点宕机处理，以及数据一致性语义调整的异常处理等。涉及的一个核心问题是针对各种异常处理采取何种数据一致性安全策略，下表将数据一致性安全策略与各种异常的对应关系进行汇总。

表 5.7 模型异常处理与数据一致性安全策略对照表

数据一致性安全策略	相关异常描述
方式一	<ul style="list-style-type: none"> ◇ 元数据服务器节点宕机重启（过渡期外） ◇ 网络分割长期不能恢复 ◇ 应用节点宕机长期不重启
方式二	<ul style="list-style-type: none"> ◇ 网络分割状态下写授权强制剥夺 ◇ 元数据服务器节点宕机重启（过渡期内，宕机前有写授权强制剥夺）
方式三	<ul style="list-style-type: none"> ◇ 网络分割状态下读授权强制剥夺（无写授权强制剥夺） ◇ 元数据服务器节点宕机重启（过渡期内，宕机前有读授权强制剥夺而无写授权强制剥夺）
方式四	<ul style="list-style-type: none"> ◇ 元数据服务器节点宕机重启（过渡期内，宕机前无授权强制剥夺） ◇ 数据一致性语义调整（无授权强制剥夺）

5.7 并发度测试

如前所述，数据一致性的维护往往以降低系统并发度为代价，蓝鲸分布式文件系统基于授权机制的在线可调整的数据一致性语义模型亦是如此。通过对该模型的分析可知，影响蓝鲸分布式文件系统并发度的主要是模型中的不兼容授权，尤其是在操作统计中占较大比重的数据读写操作所对应的授权不兼容（与数据一致性语义相关）的影响更为可观。本节就蓝鲸分布式文件系统基于授权机制的在线可调整的数据一致性语义模型所提供的各种数据一致性语义的读写并发度进行测试。测试方案如下：

针对在操作统计中所占比重较大的数据读写操作，选取两个应用节点进行并发测试，两个应用节点在规定的时间内（120 秒）针对相同的目标文件分别执行数据读操作（以 4KB 粒度持续顺序读）或数据写操作（以 4KB 粒度持续顺序写），由元数据服务器节点控制两个应用节点同时开始执行相关操作。两个应用节点的读写操作组合包括如下几种情况：

- ◇ “读/读”组合
- ◇ “读/写”组合
- ◇ “写/写”组合

对模型提供的每种数据一致性语义分别进行上述几种组合测试，记录每组测试中各

个应用节点的相关操作执行次数、授权申请次数等并发度相关指标。

测试环境：

表 5.8 BWFS 基于授权机制的在线可调整的数据一致性语义模型测试环境配置列表

	应用节点	元数据服务器节点	存储节点	交换机
硬件	数量：2 CPU：Intel(R) Xeon (TM) 2.40GHz×2 MEM：1024MB 网卡：Intel 82545EM 千兆以太网控制器×1	数量：1 CPU：Intel(R) Xeon (TM) 2.40GHz×1 MEM：1024MB 网卡：Intel 82545EM 千兆以太网控制器×1	数量：1 CPU：Intel(R) Xeon (TM) 2.40GHz×1 MEM：2048MB 网卡：Intel 82545EM 千兆以太网控制器×1 存储：3ware 9000 for ATA-RAID 盘阵 (160GB×6, RAID0)	NETGEAR JGS524 千兆交换机
软件	Redhat8.0+BWFS3.0	Redhat8.0+BWFS3.0	Redhat8.0+BWFS3.0	

测试一：超时一致性并发度测试

在蓝鲸分布式文件系统基于授权机制的在线可调整的数据一致性语义模型中，超时一致性的语义最弱，该语义下的读写并发度最高，是一种完全并发的情况。表 5.9 给出的测试结果表明“写/写”组合(3)的并发度最高，这与写操作的异步性以及存储节点的写缓冲机制相关；受蓝鲸分布式文件系统顺序写性能高于顺序读性能以及超时一致性语义下应用节点 cache 策略的影响，“读/写”组合(2)的并发度最低，且写操作应用节点的操作次数高于读操作应用节点的操作次数。

表 5.9 超时一致性并发度测试结果

测试项目		操作次数	授权申请次数
“读/读”组合(1)	节点一（读）	1647282	1
	节点二（读）	1647333	1
	合计	3294615	2
“读/写”组合(2)	节点一（读）	1407019	1
	节点二（写）	1476615	1
	合计	2883634	2
“写/写”组合(3)	节点一（写）	1731840	1
	节点二（写）	1730166	1
	合计	3462006	2

测试二：释放一致性并发度测试

对于数据读写操作而言，释放一致性与超时一致性的语义是一致的，表 5.10 给出的

测试结果验证了这一点。

表 5.10 释放一致性并发度测试结果

测试项目		操作次数	授权申请次数
“读/读”组合(4)	节点一（读）	1607595	1
	节点二（读）	1607575	1
	合计	3215170	2
“读/写”组合(5)	节点一（读）	1392492	1
	节点二（写）	1441153	1
	合计	2833645	2
“写/写”组合(6)	节点一（写）	1726747	1
	节点二（写）	1715599	1
	合计	3442346	2

测试三：写一致性并发度测试

在写一致性语义下，写授权之间是不兼容的，这在很大程度上影响系统写操作的并发度。从表 5.11 给出的测试结果可以看出“写/写”组合(9)的并发度较相同数据一致性语义下其他两种组合的并发度有十分明显的下降，其下降幅度在 50%以上；另外在“写/写”组合(9)下，两个应用节点的操作次数也有明显差异，且与应用节点的授权申请次数正相关。

表 5.11 写一致性并发度测试结果

测试项目		操作次数	授权申请次数
“读/读”组合(7)	节点一（读）	1635603	1
	节点二（读）	1635625	1
	合计	3271228	2
“读/写”组合(8)	节点一（读）	1393304	1
	节点二（写）	1469795	1
	合计	2863099	2
“写/写”组合(9)	节点一（写）	612144	5278
	节点二（写）	741224	5292
	合计	1353368	10570

测试四：读写一致性并发度测试

读写一致性是蓝鲸分布式文件系统基于授权机制的在线可调整的数据一致性语义模型中语义最强的一种，也是读写并发度最低一种，该语义下写授权是排他的。表 5.12 给出的测试结果表明“读/写”组合(11)、“写/写”组合(12)的并发度明显低于“读/读”组合(10)，其差异幅度都超过 50%；对于“读/写”组合(11)而言，读写操作的不均衡被进一步放大，这与写授权的剥夺周期较长相关；“写/写”组合(12)的并发度最低，主要

因为写授权的剥夺周期较长，系统在授权剥夺操作中消耗了更多的时间；在每种涉及授权剥夺的测试组合内部，应用节点的相应操作次数与应用节点的授权申请次数正相关。

表 5.12 读写一致性并发度测试结果

测试项目		操作次数	授权申请次数
“读/读”组合(10)	节点一（读）	1645630	1
	节点二（读）	1645722	1
	合计	3291352	2
“读/写”组合(11)	节点一（读）	305160	2156
	节点二（写）	1252016	2166
	合计	1557176	4322
“写/写”组合(12)	节点一（写）	592031	6489
	节点二（写）	563688	6485
	合计	1155719	12974

5.7.1 并发度与数据一致性语义

图 5.19 对蓝鲸分布式文件系统基于授权机制的在线可调整的数据一致性语义模型中的各种数据一致性语义在每种测试组合下的读写并发度进行比较，可以看出，在没有产生授权冲突的情况下，不同数据一致性语义的读写并发度基本一致，当发生授权冲突的时候，系统的读写并发度急剧下降，其下降幅度在 40%至 60%之间。因此系统的数据一致性语义应该根据应用的实际需求适当选择，而不是盲目追求严格的数据一致性语义。

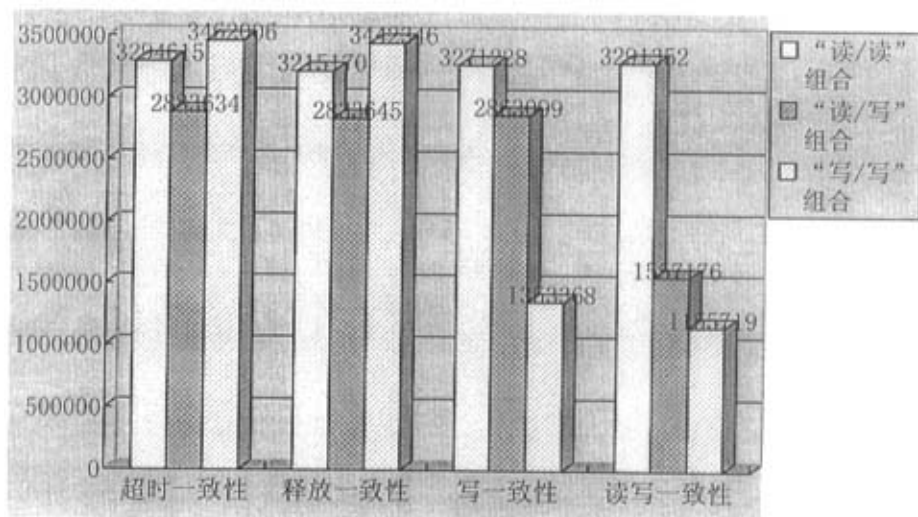


图 5.19 读写并发度与数据一致性语义关系图

5.7.2 并发度与授权申请

上述测试中涉及授权剥夺的测试组合包括写一致性语义下的“写/写”组合(9)、读写一致性语义下的“读/写”组合(11)和读写一致性语义下的“写/写”组合(12)等三组。如前所述,在这些测试组合的内部,应用节点的相应操作次数与应用节点的授权申请次数正相关,但在这些测试组合之间,系统整体的读写并发度却与系统授权申请次数反相关,如图 5.20 所示:

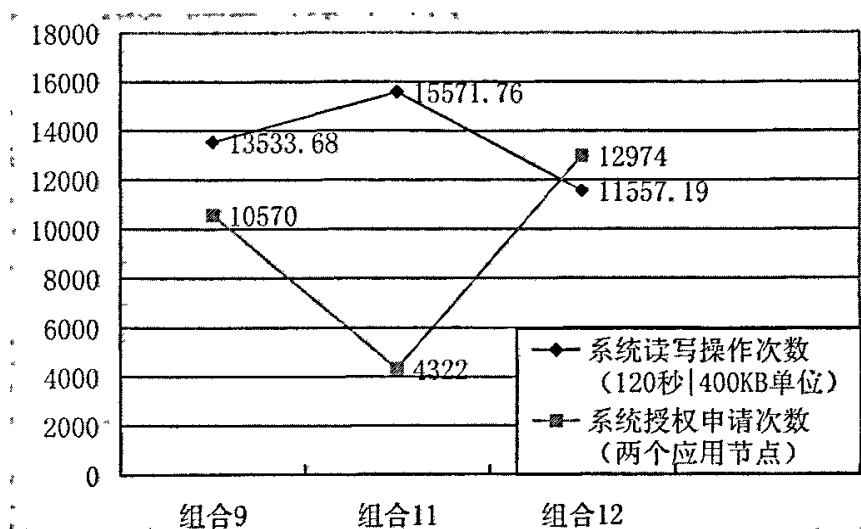


图 5.20 读写并发度与授权申请次数关系图

由此可见授权申请次数对系统的影响双向的,一方面发生数据共享冲突的应用节点之间授权申请次数占优的其吞吐率较高,另一方面系统的整体并发度随冲突授权申请次数的增加而下降。因此授权申请与剥夺策略选择以及相关参数的设定需要在两个方面之间进行适当的权衡。

第六章 结束语

本章首先概述蓝鲸分布式文件系统的研究现状, 然后就论文的研究成果及未来工作加以总结讨论。

6.1 研究现状

蓝鲸网络存储系统是国家“八六三”计划基金支持的重点科研项目, 到目前为止已经取得了诸多科研成果[52][53], 某些方面在国内外处于领先地位。蓝鲸分布式文件系统作为蓝鲸网络存储的核心系统软件, 管理海量存储空间, 提供机群结构环境下高速的、并发的、共享的、高可用的远程文件访问服务, 具有很好在线扩展性。蓝鲸分布式文件系统数据一致性语义模型允许用户根据实际需要在线设置系统的数据一致性语义, 以更好的兼容不同模式的分布式应用, 使得蓝鲸分布式文件系统在此方面具有其他分布式文件系统所没有的高度灵活性。

6.2 论文总结

如前所述, 对各种不同应用模式支持的兼容性是评价分布式文件系统优劣的重要标准之一, 分布式文件系统的数据一致性语义是决定分布式应用模式的关键所在。从这个意义上讲, 蓝鲸分布式文件系统数据一致性语义研究具有重要学术及实用价值, 它不仅丰富了对分布式文件系统数据一致性语义的研究, 明晰了蓝鲸分布式文件系统的数据一致性语义, 而且加强了蓝鲸分布式文件系统的正确性与可用性, 提高了蓝鲸分布式文件系统的应用兼容性。

本论文对蓝鲸分布式文件系统的数据一致性语义及相关问题进行了深入研究, 取得如下主要成果:

◇ 提出并实现基于软件的网络容错技术

节点间数据传输的正确性及完整性是蓝鲸分布式文件系统可用性的重要体现, 也是其数据一致性的基础。本论文提出并实现独立于系统硬件支持的连接复制、通道切换、请求重构等基于软件的网络容错技术, 用于解决蓝鲸分布式文件系统中影响较大的通道连接软故障中断异常以及网络分割故障恢复后期的系统逻辑恢复工作, 实现在特定网络故障下应用层透明的数据无损传输, 降低了蓝鲸分布式文件系统网络容错对系统硬件环境的依赖性, 提高了系统的可用性, 使得蓝鲸分布式文件系统的数据一致性有所保障。

◇ 设计并实现自适应的带外模式文件属性更新机制

文件属性更新是蓝鲸分布式文件系统数据一致性的前提。本论文在综合分析

蓝鲸分布式文件系统架构特点及数据缓存策略的基础之上,设计并实现了一种自适应的带外模式文件属性更新机制。该机制充分借鉴其他分布式文件系统在文件属性更新方面的经验,集机会更新、被动更新、周期更新等多种文件属性更新方式于一身,有效地配合了蓝鲸分布式文件系统系统的数据一致性语义模型。该机制的另一特点是其文件属性更新周期的在线调整能力:系统能够根据当前的网络状态对文件属性的更新周期进行自适应调整,用以降低数据丢失风险,提高网络的整体使用效率;并且允许用户动态地设置更新周期基数及更新周期调整幅度。所有这些调整都以应用层透明的方式进行,更好地体现了蓝鲸分布式文件系统机动灵活的特点。

✧ 设计并实现基于授权机制的在线可调整的数据一致性语义模型

本论文深入分析了分布式环境下的数据一致性问题,特别针对蓝鲸分布式文件系统的数据一致性问题进行了重点研究;并从数据一致性的角度出发,对当前的及潜在的分布式应用模式加以汇总分类;在此基础之上,设计并实现了一种支持数据一致性语义在线调整的数据一致性语义模型,该模型基于授权(一种可剥夺的文件级粒度的多态文件锁)机制,打破传统中文件系统数据一致性语义单一恒定的观念,创造性地提出数据一致性语义的在线调整思想,允许用户在线设置蓝鲸分布式文件系统的数据一致性语义,以满足不同应用模式的需求。该模型提供超时一致性、释放一致性、写一致性、读写一致性等四种数据一致性语义,支持从类 NFS 语义到类 UNIX 语义等多种数据一致性语义的兼容性。

6.3 未来工作

到目前为止,关于蓝鲸分布式文件系统数据一致性语义的研究工作已经取得了阶段性成果,蓝鲸分布式文件系统已经可以满足大多数分布式应用的数据一致性需求,尤其是历史上首次实现真正意义上的对 NFSv3 语义的完全兼容。但是作为一个支持并发共享访问和在线系统扩展的大型分布式文件系统,针对其数据一致性语义的研究工作远未就此结束,仍有许多问题有待研究。

➤ 进一步加强节点间数据传输保障

网络容错是复杂的系统问题,对于诸如网络驱动崩溃、网络连接硬中断等网络故障,蓝鲸分布式文件系统现有的基于软件的网络容错技术还不能做到应用层透明的数据无损传输。如何在各种网络故障下确保节点间数据传输的正确性及完整性,加强蓝鲸分布式文件系统数据一致性的基础,是值得深入研究的课题。

➤ 带外模式文件更新的广义原子性保障

蓝鲸分布式文件系统目前仍然存在的一个数据一致性问题如果是应用节点

在脏数据部分写回或全部写回而文件属性尚未更新时发生宕机,那么就会出现文件数据更新而文件属性未更新的不一致状态,蓝鲸分布式文件系统现有的数据一致性语义模型以及其他现有机制都不能有效处理该问题。从事务操作的原子性角度看,文件数据更新与文件属性更新是文件更新事务的两个阶段,系统应该保证两个阶段要么都完成,要么都不完成,至于两个阶段之间是否可中断(而非终止)取决于蓝鲸分布式文件系统的数据一致性语义(这一点与普通意义上的原子性有所不同,因此称为广义原子性)。通常而言,这种广义原子性操作需要日志机制加以支持,如何在蓝鲸分布式文件系统这种数据通道与控制通道分离的带外模式下实现日志支持(或文件更新的广义原子性保障机制)是一项颇具挑战性的工作。

数据一致性语义模型的测试与调优

论文描述的基于授权机制的在线可调整的数据一致性语义模型是蓝鲸分布式文件系统数据一致性语义模型的原型系统,在这个原型系统的设计实现中更多地考虑功能性问题,对性能问题的考虑相对较少。目前该原型系统的设计要点基本上已经全部实现,可以融入现有的蓝鲸分布式文件系统中运行,但是针对该模型的测试及调整优化工作却远未结束,尤其是某些性能相关的参数(如授权申请重试间隔、授权最小生存期等)需要结合具体的分布式应用的测试结果进行调整设定,关于模型异常处理机制的测试更是需要考虑众多的意外因素。这是一项浩繁的工作,其难度与工作量不亚于本文的已有工作,需要在今后的工作中针对性地加以解决。

模型调优的另一主要工作是数据一致性安全策略优化。蓝鲸分布式文件系统数据一致性语义模型提供数据一致性安全策略用于各种异常情况下的授权及数据处理。现有的数据一致性安全策略主要从两个层次对应用节点的相关数据进行处理:文件级的 cache 处理和设备驱动层的缓冲处理。文件级的 cache 处理比较简单,根据特定的授权信息可以准确定位相应文件在应用节点 cache 中的所有数据块,而设备驱动层的缓冲处理则缺少通过授权信息定位数据块的有效方式。因此现有的数据一致性安全策略更多的采用了保守的处理方式:以牺牲单个节点的数据为代价换取系统中其他节点的数据安全性和数据完整性。实际上,这种“宁错勿漏”的处理方式在合理性与效率上都存在很多问题。一种可能的替换方案是将数据一致性安全策略的部分处理工作转移到存储节点,从存储节点加强对特定数据块的读写控制,但该方式增加了存储节点对软件支持的依赖性,不利于存储节点的标准化,而且由于存储节点也可能发生网络分割,所以处理机制亦很复杂。蓝鲸分布式文件系统数据一致性语义模型需要更加合理高效的数据一致性安全策略。

➤ 元数据服务器机群环境下的系统数据一致性语义研究

本论文的研究工作基于单个元数据服务器节点环境完成,元数据服务器机

群环境下的系统数据一致性问题更加复杂, 如何确保多个元数据服务器节点之间的元数据的一致性是需要首先解决的问题。受蓝鲸分布式文件系统文件元数据动态负载均衡策略的影响, 蓝鲸分布式文件系统现有的数据一致性语义模型需要进行相应调整, 涉及诸多因素:

- 1) 是否允许已授权文件进行元数据动态迁移?
- 2) 如果不允许已授权文件进行元数据动态迁移, 那么如何避免系统元数据负载失衡?
- 3) 如果允许已授权文件进行元数据动态迁移, 那么是采取授权撤销的迁移方式? 还是采取授权保留的迁移方式?
- 4) 如果采取授权撤销的迁移方式, 那么如何避免元数据反复迁移造成的额外性能损失? 如何处理迁移过程中应用节点网络分割问题?
- 5) 如果采取授权保留的迁移方式, 那么是像普通元数据那样通过存储节点迁移? 还是直接通过网络迁移?
- 6) Server Identifier 机制是否需要调整? 如何调整?
- 7) 数据一致性语义调整如何在多个元数据服务器节点间广播? 如果其间元数据服务器节点发生网络分割怎么办?

类似上述问题还有很多, 都是现有的数据一致性语义模型向元数据服务器机群环境迁移所必须考虑处理的问题。另一个思路是考虑设计全新的数据一致性语义模型, 但同样面对前述所有的数据一致性问题。总之元数据服务器机群环境下的数据一致性语义研究是一个十分复杂的课题, 需要进一步的深入研究。

参考文献

- [1] 许鲁, 863 课题申请报告——新型网络服务器系统, 国家高性能计算机工程技术研究中心技术报告, 2002 年 1 月。
- [2] 杨德志, 黄华, 张建刚, 许鲁, 大容量、高性能、高扩展能力的蓝鲸分布式文件系统, *计算机研究与发展*, 第 42 卷, 第 6 期, 1028—1033 页, 2005 年。
- [3] 傅湘林, 谢长生, 曹强, 刘朝斌, 黄建忠, 一种融合NAS和SAN技术的存储网络系统, *中国存储会议*, 2002 年。
- [4] 杨德志, 元数据服务器集群设计报告, 国家高性能计算机工程技术研究中心内部技术文档, 2004 年。
- [5] 田颖, 许鲁, 分布式文件系统中的负载平衡技术, *计算机工程*, 第 29 卷, 第 19 期, 2003 年 11 月。
- [6] 田颖, 分布式文件系统中的负载平衡技术研究, *中国科学院计算技术研究所硕士学位论文*, 2003 年。
- [7] Fabio Kon. Distributed File Systems Past, Present and Future, A Distributed File System for 2006, March 1996.
- [8] Andrew S.Tanenbaum. *Distributed Operating Systems*, 1995 by Prentice-Hall, Inc. pp.245-285.
- [9] E. LEVY and A. SILBERSCHATZ. Distributed File Systems: Concepts and Examples. *ACM Computing Surveys*, Vol.22, No.4, pp.321-374, December 1990.
- [10] P. Triantafillou and C. Neilson. Achieving Strong Consistency in a Distributed File System. *IEEE Transactions on Software Engineering*, Vol.23, No.1, pp.35-55, January 1997.
- [11] John H. Howard. *An Overview of the Andrew File System*, CMU-ITC-062, <http://reports-archive.adm.cs.cmu.edu/itc85.html>
- [12] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in distributed file system. *ACM Transactions on Computer Systems*, Vol.6, No.1, pp.51-81, February 1988.
- [13] Mahadev Satyanarayanan. Scalable, Secure, and Highly Available Distributed File Access. *IEEE Computer*, pp.9-21, May 1990.

- [14] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1992.
- [15] Sun Microsystems, Inc. *NFS: Network File System Protocol Specification*, March 1989.
- [16] B. Callaghan, B. Pawlowski, P. Staubach, Sun Microsystems, Inc. *NFS Version 3 Protocol Specification*, June 1995.
- [17] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, Sun Microsystems, Inc. C. Beame, Hummingbird Ltd. M. Eisler, Zambel, Inc. D. Noveck, Network Appliance, Inc. *NFS Version 4 Protocol Specification*, December 2000.
- [18] R. Sandberg. The Sun Network File System: Design, Implementation and Experience. IN *Proceedings of USENIX Summer Conference*, Summer 1987. University of California Press, pp.300-313.
- [19] Protocols for Interworking: XNFS, Version 3W-Network Lock Manager Protocol. <http://www.opengroup.org/onlinepubs/009629799/chap10.htm>.
- [20] Protocols for Interworking: XNFS, Version 3W-Network Status Monitor Protocol. <http://www.opengroup.org/onlinepubs/009629799/chap11.htm>.
- [21] Mahadev Satyanarayanan, James J. Kistler, Puneet Kumar, Maria E. Okasai, Ellen H. Siegel, and David C. Steere. Coda: A Highly Available File System for a Distributed Workstation Environment, *IEEE Transactions on Computers*, Vol.39, No.4, pp.447-459, April 1990.
- [22] L. B. Mummert. Exploiting Weak Connectivity in a Distributed File System. PhD dissertation. Computer Science Division, School of Computer Science, Carnegie Mellon University, December 1996.
- [23] James J. Kistler and Mahadev Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, Vol.10, No.1, February 1992.
- [24] J. Ousterhout, A. Cherenon, F. Dougli, M. Nelson, B. Welch. The Sprite network operating system. *IEEE Computer*, Vol.21, No.2, pp.23-36, February 1988.
- [25] M. Baker and J. Ousterhout. Availability in the Sprite Distributed File System. *Operating Systems Review*, Vol.25, No.2, pp.95-98, April 1991.
- [26] Andy Hisgen, Andrew D. Birrell, Timothy Mann and Garret Swart. Availability and Consistency Tradeoffs in the Echo Distributed File System. *In Proceedings of the Workshop on Workstation Operating Systems*. Pacific Grove, CA, pp.49-53, September 1989.
- [27] Andrew D. Birrell, Andy Hisgen, Chuck Jerian, Timothy Mann, Garret Swart. The echo distributed file system, Technical Report #111, DIGITAL Equipment Corporation System Research Center, Palo Alto, CA, September 1993.

- [28] Cray G. Gray and David R. Cheriton. Lease: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency. *In Proceedings of 12th ACM Symposium on Operation System*, pp.202-210, December 1989.
- [29] Randal Chilton Burns. Data Management in a Distributed File System for Storage Area Networks. PhD thesis, University of California Santa Cruz, March 2000.
- [30] J. Menon, D. A. Pease, R. Rees, L. Duyanovich, B. Hillsberg. IBM Storage Tank-A heterogeneous scalable SAN file system. *IBM Systems Journal*, Vol.42, No.2, 2003.
- [31] Lustre: A Scalable, High-Performance File System, Cluster File Systems, Inc. Lustre Whitepaper Version 1.0, November 2002.
- [32] Peter Braam. The Lustre Storage Architecture. <http://www.lustre.org/docs/lustre.pdf>, 2004.
- [33] Garth A. Gibson, David F. Nagle, etc. NASD Scalable Storage Systems, Proceedings of USENIX 1999, Linux Workshop, Monterey, CA, June 1999.
- [34] Rosenblum and Ousterhout. The design and implementation of a log-structured file system. *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, October 1991.
- [35] Ganger and Patt. Soft Updates: A Solution to the Metadata Update Problem in File Systems. *Technical report CSE-TR-254-95, Computer Science and Engineering Division, University of Michigan*, 1995.
- [36] SAN Security Terminology, *Network Storage Group Technology Brief*, QLogic, Inc. January 2003.
http://www.qlogic.com/documents/datasheets/knowledge_data/whitepapers/tech_brief_sansecurity_term.pdf
- [37] H. Yoshida. LUN security considerations for storage area networks. *Technical Report, Hitachi Data Systems*, 1999.
- [38] James Archibald and Jean-Loup Baer. Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model. *ACM Transactions on Computer Systems*, Vol.4, November 1986.
- [39] The Fiber Channel Industry Association, <http://www.fibrechannel.org/>, 2004.
- [40] Mazin Yousif. InfiniBand 1.0 Architecture Overview, February 27, 2001. Intel Developer Forum, Spring 2001.
- [41] Andrew Lockey, Storage over the InfiniBand Fabric, March 1, 2001. Intel Developer Forum, Spring 2001.
- [42] Neil Brown, The Linux Virtual File-system Layer, 1999.
<http://www.cse.unsw.edu.au/~neilb/oss/linux-commentary/vfs.html>.

- [43] Video-on-Demand Overview, <http://www.cs.tut.fi/tlt/stuff/vod/VoDOverview/vod1.html>.
- [44] <http://www.paradigmgeo.com/>.
- [45] (美) Marc Farley 著, 孙功星, 蒋文保, 范勇等译, *SAN 存储区域网络 (第二版)*, 北京: 机械工业出版社, 2002 年 4 月。
- [46] W. Richard Stevens. *TCP/IP Illustrated Volume 1: The Protocols*, 1994 Pearson Education, Inc.
- [47] J. Postel, Information Sciences Institute, University of Southern California, *INTERNET PROTOCOL [RFC791]*, September 1981.
- [48] J. Postel, Information Sciences Institute, University of Southern California, *TRANSMISSION CONTROL PROTOCOL [RFC793]*, September 1981.
- [49] P. LeMahieu, V. Bohossian, J. Bruck, Fault-Tolerant Switched Local Area Networks, *Proc of the First Merged Int. Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, pp.747-751, 1998.
- [50] <http://nbd.sourceforge.net/>.
- [51] 范勇, 张建刚, 许鲁, 蓝鲸分布式文件系统网络容错的软件技术, *计算机工程*, 已录用。
- [52] 黄华, 张建刚, 许鲁, 蓝鲸分布式文件系统的分布式分层资源管理模型, *计算机研究与发展*, 第 42 卷, 第 6 期, 1034—1038 页, 2005 年。
- [53] 黄华, 张敬亮, 张建刚, 许鲁, 蓝鲸分布式文件系统的客户端元数据缓存模型, *计算机科学*, 已录用。

致 谢

谨以此文献给我的父母。为了我的学业，他们辛勤劳作，耗尽青春，今天当这篇论文完稿，我即将踏上人生另一起点之时，回首岁月，父母已老。二十年寒窗苦读，求学之路崎岖坎坷，他们给予我的不仅是贫寒中坚实的物质基础，更是逆境下强大的精神支持。这种恩情无以言表，今为此文，聊以慰藉。

感谢我的导师许鲁博士，本篇论文是在许老师的指导之下完成的，在论文的创作及撰写过程中，他的指导及建议使我少走了很多弯路。作为留学归国人员，他不仅带回先进的技术和思想，同时也带来创业的激情与动力，三年来我为之拼搏与奋斗，这份经历是我人生一笔宝贵的财富。

感谢蓝鲸分布式文件系统组组长张建刚博士，他在 Linux 内核及文件系统方面造诣颇深，对我研究及改进蓝鲸分布式文件系统的帮助很大。他如此之敬业，在他身上我看到了民族软件产业的希望。

感谢韩月、黄华、秦平、刘舸、田智勇、杨德志、张宏超、张敬亮、张军伟等蓝鲸分布式文件系统组的各位同事及同学，感谢他们在蓝鲸分布式文件系统项目组中的支持与合作。

感谢技术支持组的叶跃亮以及台州分所的冯严东等同事，感谢他们在测试环境搭建及测试系统维护等方面的配合与支持。

感谢国家高性能计算机工程技术研究中心的各位同仁，这里不乏优秀的人才，与这些人共事是一种幸运，感谢大家共同营造的环境。

感谢中科院计算所研究生部的靳晓明、李琳、宋守礼、张晓辉、周世佳等各位老师，感谢他们在我攻读硕士学位期间所给予的关心和帮助。

最后，谨向所有关心、支持、帮助过我的人们致以衷心的感谢。

作者简介

姓名：范勇 性别：男 出生日期：1979.03.25 籍贯：辽宁海城

2003.09 — 2006.07 中国科学院计算技术研究所 计算机系统结构专业 硕士

1998.09 — 2003.07 中国科学技术大学 信息学院 计算机科学与技术专业 学士

【攻读硕士学位期间发表的论文】

- [1] 范勇，张建刚，许鲁，“蓝鲸分布式文件系统网络容错的软件技术”，《计算机工程》
（已录用，2006年10月刊登）

【攻读硕士学位期间参加的科研项目】

- [1] 国家 863 软件专项：“虚拟化网络存储功能软件” (2004AA1Z2250)

【攻读硕士学位期间的获奖情况】

- [1] 2005 年度中国科学院计算技术研究所所长奖学金优秀奖
[2] 计算机软件著作权登记，“文件集群功能软件”，登记号：2005SR13703

蓝鲸分布式文件系统数据一致性语义研究

作者：[范勇](#)

学位授予单位：[中国科学院计算技术研究所](#)

本文链接：http://d.g.wanfangdata.com.cn/Thesis_Y1005425.aspx

授权使用：中科院计算所(zkyjsc)，授权号：e777fb1b-1c61-4b4d-9290-9e5c0120719a

下载时间：2010年12月30日