FTDSS:高容错分布式共享存储机制*)

魏青松　卢显良　雷　宇
(电子科技大学计算机学院　成都610054)

FTDSS:High Fault-Tolerance Distributed Shared Storage Mechanism

WEI Qing-Song　LU Xian-Liang　LEI Yu
(Department of Computer Science of UEST of China , ChengDu, China 610054)

Abstract Computerized data has become critical to the survival of an enterprise. Company must have a strategy for recovering their data should data lose. RAID is a popular mechanism to offer good fault-tolerance. But RAID can not work well when one more disks fail. In this paper, we present an efficient Network-based high High-Tolerance Distributed Shared Storage mechanism called FTDSS. FTDSS makes use of disk space of node in Network to build a large public shared storage space. Users can read/write their file from/to the public storage space from node of network. Physically File is stored in each node in form of data fragment or XOR verify fragment. Because of redundant XOR fragments, file is available even when two more nodes fail. FTDSS realize distant redundant srorage. At last, this paper use experiment to prove that FTDSS can offer high fault-tolerance and advanced performance.

Keywords Network, Fault-tolerance, Distributed, Storage, Fragment, XOR

1　引言

随着近年来网络系统的普及和应用,数据在企业全部资产中所占的比例越来越大,在企业的网络系统中,最宝贵的不是各种硬件设备,而是企业在长期发展过程中所积累下来的业务数据.数据的高可用性要求存储的高可靠性,因此,发展数据容错和恢复技术是极其重要的.

RAID是广泛采用的保证数据可靠性的技术,但RAID[1]具有如下缺点:1、它只允许单个磁盘失败,如果系统中任意两个或两个以上磁盘失败,将导致数据不可用.2、无法实现分布式远程存储.3、价格昂贵,需要单独的硬件支持.SAN虽然实现了网络存储,但其容错性依赖RAID的容错性[2].

本文提出一种高容错的基于网络的分布式共享存储机制—FTDSS(High Fault-Tolerance Distributed Shared Storage Mechanism),FTDSS利用网络中各节点的磁盘空间,构建一个大的虚拟空间,该空间对所有的网络授权用户开放,用户可以在全局空间读写文件,从而实现文件的分布式共享存储,FTDSS将文件分片,然后将数据分片和它们之间的校验分片存储在网络中的各节点,只要一定数目的节点在线,即使其它的机器不在线或崩溃,也可以从全局空间完整读出数据文件,从而获得高的容错性.FTDSS通过网络分布式存储,实现了数据远程容灾.同时,FTDSS将文件分片并行读写,能获得比单机高的读写性能.

2　FTDSS原理简介

网络中包含多个节点主机,每个主机不仅是网络中的一员,而且是独立的计算机,拥有独立的存储空间.利用这个特点,本文提出基于网络的高容错的分布式共享存储机制FTDSS,其结构如图1所示.FTDSS利用网络中各主机的磁盘空间,构建一个大的虚拟空间,该空间对所有的网络授权用户都是完全开放的,用户可以从全局空间读写文件,从而实现文件的分布式共享存储.

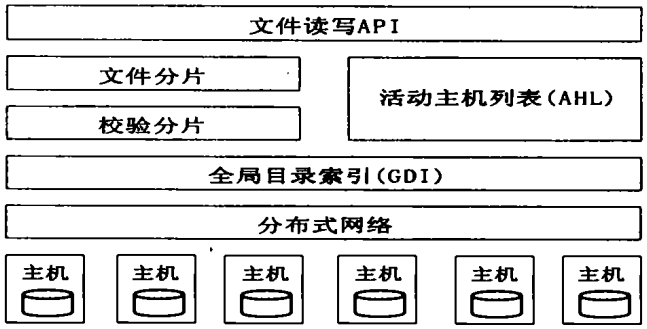

图1　FTDSS结构图

全局存储空间通过全局目录索引(Global Directory Index,简称GDI)来管理,全局目录索引GDI类似Unix文件系统中的inode,inode实现了文件名和磁盘块之间的映射,而GDI实现了文件名和存储节点主机之间的映射,它保存文件分片在网络中的存储情况,在每个主机上都有一份GDI,作为一个分布式系统,系统中所有主机都没有主从之分,每个节点都参与信息同步,GDI同步机制对各节点的GDI进行信息同步,从而保证全局空间数据的一致性.

活动主机列表(Active Hosts List,简称AHL)维护分布式网络中当前活动的主机列表,随着节点加入或退出网络,活动主机列表要增加或删除相应的列表项,从而保证活动主机列表信息的实时性.

当用户向全局空间写文件时,首先查询活动主机列表AHL,根据活动主机的网络通信情况和磁盘剩余空间,选择可用主机阵列(Available Hosts Array,简称AHA),根据分片存储算法将文件分成N个分片,并计算分片之间的F个XOR校验分片,然后按可用主机阵列AHA将N个分片和F个校验分片分别存储到网络中的N+F被选择的主机节点并更新全局目录索引GDI.

*)本文受国家95重点攻关项目支持.魏青松　博士研究生,主要研究方向为计算机网络、网络存储、分布式操作系统等.卢显良　教授,博士生导师,主要研究方向为计算机网络、操作系统.雷　宇　硕士研究生,主要研究方向为计算机网络.

读取文件时，先查询全局目录索引 GDI，获得文件的数据分片和校验分片的存储信息，得到文件存储节点表，然后与活动主机列表 AHL 取交集得到可用的存储节点表，在从各存储节点读分片时，尽量先读原始分片，如原始分片无法重组，再读校验分片，通过 XOR 计算，得到完整的分片数据，完成文件重组。

3 FTDSS 的实现机制

3.1 全局目录索引 GDI 及其同步机制

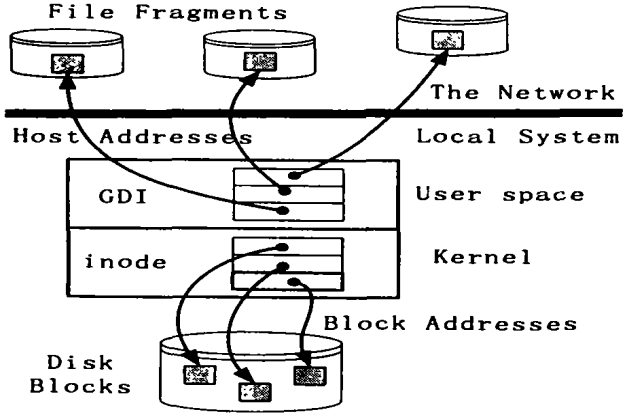

图2 GDI 和 Inode 对比图

为了将各节点主机的磁盘空间构建成一个大的全局共享存储空间，并实现全局名字空间的统一，在每个主机上有一份全局目录索引(Global Directory Index，简称 GDI)，用于维护全局存储空间的文件信息，它保存文件的分片情况以及分片和校验分片在网络各节点主机中的存储情况。全局目录索引 GDI 类似 Unix 文件系统中的 Inode，Inode 实现了文件名和磁盘块之间的映射，而 GDI 实现了文件名和存储节点主机之间的映射，通过文件名查询全局目录索引 GDI，就可以知道该文件的分片情况，以及该文件分片和校验分片存储在网络中的哪些节点主机上。图2为全局目录索引 GDI 和 Inode 的对比图[3]。GDI 和 Inode 之间的区别在于：inode 将一个磁盘卷上的磁盘块聚集成一个文件，而 GDI 将网络上各节点主机上的文件片聚集成一个完整的文件。

在每个主机节点上都有一份全局目录索引 GDI，作为一个分布式系统，系统中所有主机都没有主从之分，为了保持全局目录节点表 GDI 的一致性，每个节点都必须参与信息同步，GDI 采用事件驱动的信息同步机制 EDSM(Event Driven Synchronization Mechanism)，对各节点主机上的全局目录索引 GDI 进行信息同步，从而保证全局空间数据的一致性。

所谓事件驱动的同步机制 EDSM 是指当文件写、文件显示等操作以及定时器等事件发生时，驱动各节点广播同步信息，更新全局目录索引 GDI，从而维护全局空间元数据内容的一致性。当一段时间没有文件操作发生时，而定时器超时将驱动 GDI 的信息同步。

EDSM 分需要应答的同步和不需要应答的同步。需要应答的同步要求发送方和接收方同时更新本地 GDI 信息，用于发送方强制所有在线主机同步 GDI 信息，如图3(a)所示，发送方节点在事件驱动下广播要求应答的 GDI 同步报文 SYN(GDI,ANS)，接受方接收 GDI 同步广播后，更新本节点的 GDI 信息，并向发送方发送不要求应答的 GDI 同步应答报文 SYN(GDI,NOANS)，发送方接收 GDI 同步应答报文，更新本机 GDI[4]。

不需要应答的同步只要求各接收方更新其 GDI 信息，发送方不需要根据接收方的回答再次更新，用于某个主机写文件成功之后，通知其他所有在线主机同步此记录，如图3(b)所示，发送方节点在事件驱动下广播不要求应答的 GDI 同步报文 SYN(GDI,NOANS)，接受方接收 GDI 同步广播后，更新本节点 GDI 信息。

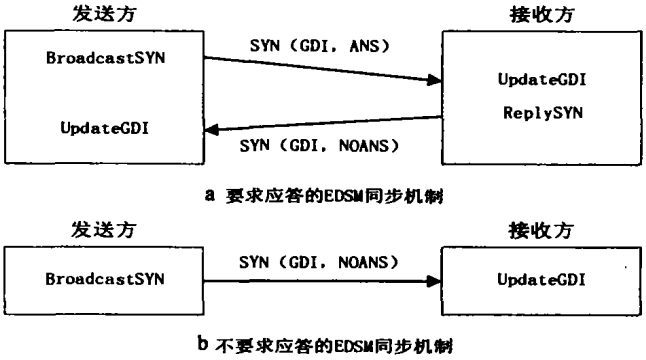

图3 事件驱动的 GDI 同步机制 EDSM

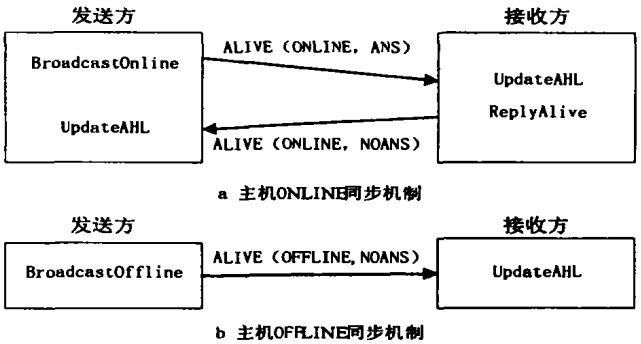

图4 活动主机列表 AHL 同步机制

3.2 活动主机列表 AHL 及其同步机制

作为一个分布式系统，为了避免广播报文频繁出现，影响网络通信质量，每一个在线节点主机维护一个活动主机列表(Active Hosts List，简称 AHL)，活动主机列表 AHL 维护网络中当前活动的节点主机列表，AHL 列表项包括各主机的 IP 信息、剩余磁盘空间及时间戳。IP 信息用于单播通信时确定对方是否在线，节点主机剩余磁盘空间用于文件写时的节点选择，时间戳记录活动主机列表项的更新时间。

当用户向全局空间写文件时，首先查询活动主机列表 AHL，根据活动主机的网络状况和磁盘剩余空间，选择可用的主机阵列，然后决定文件分片情况；当用户读文件时，根据全局目录索引 GDI 与活动主机列表 AHL，得到可用的存储节点表，然后从各存储节点读出文件。活动主机列表 AHL 是文件读写的关键。

鉴于活动主机列表 AHL 的重要性，须保持 AHL 的及时性。也就是，新节点主机加入到系统中时，所有在线主机均须将该主机添加到 AHL；某节点主机正常退出时，所有在线主机也须从 AHL 删除该主机；某节点异常退出时，其他在线主机须检测到此事件，并更新各自的 AHL。

图4显示活动主机列表 AHL 的同步机制。活动主机列表 AHL 的同步分 AHL 上线同步和 AHL 离线同步。AHL 上线同步要求发送方和接收方同时更新本地 AHL，用于上线主机

强制所有在线主机同步 AHL，如图4(a)所示，发送方节点在上线时广播要求应答的 AHL 上线同步报文 ALIVE(ONLINE,ANS)，接收方接收 AHL 上线同步广播后，更新本节点 AHL，并向发送方发送不要求应答的 AHL 在线同步应答报文 ALIVE(ONLINE,NOANS)，发送方接收 AHL 同步应答报文，更新本机 AHL[5]。

AHL 离线同步用于某主机正常退出时通知其他在线主机同步 AHL 记录，只要求各接收方更新其 AHL，发送方不需根据接收方应答再次更新，因为发送方即将离线。如图4(b)所示，离线节点广播不要求应答的 AHL 离线同步报文 ALIVE(OFFLINE,NOANS)，接受方接收 AHL 离线同步广播后，将该主机从 AHL 中删除。当主机异常退出后，其它节点的定时 AHL 同步广播，将保证该主机从在线节点的 AHL 中删除。

3.3 文件写

当用户向全局空间写文件时，为了将文件分布式保存到网络各节点的存储设备上，需要分片存储，其基本步骤是：1、查询活动主机列表 AHL，获得当前活动主机总数 M；2、根据分片算法计算出存储节点数 N(N<M)；3、将文件分成 N_d 个数据片，并计算分片之间的 XOR 校验，得到 N_m 个校验片($N=N_d+N_m$)；4、将 N_d 个数据片和 N_m 个校验片分别存储到 N 个节点主机中；5、更新全局目录索引 GDI。

当保存分片数据的 N 个节点中部分节点不在线或崩溃时，我们仍希望数据文件可用，为了达到这个目标，我们采用了冗余校验分片算法 RVSA (Redundant Verify Segment Arithmetic)。其思想是：在分布保存文件时，不仅保存文件分片，而且保存分片之间的校验分片。这样当部分分片不可用时，可以通过校验分片恢复数据。

冗余校验分片算法 RVSA 可以用如下公式表示：

$$\begin{cases} N=n+C_n^2 & \{n\mid n\geqslant 2, n\in Z\} \\ N_n\leqslant M\leqslant N_{n+1} & (M\geqslant 3) \end{cases}$$

其中，M：当前活动主机数；n：文件分片数；C_n^2：校验分片数；N：存储节点数。

写文件时，通过活动主机数 M 计算出文件分片数 n，进而获得存储节点数 N，从 M 个活动主机中选择 N 个存储节点，将文件等分成 n 片，计算出 C_n^2 校验分片，将分片和校验分片分别存储到 N 个存储节点。

如当前活动主机数为7，通过 RVSA 算法计算出文件分片数 n=3，存储节点数 N=6，并从7个主机中选择6个存储主机，写文件时，将文件等分为3片，计算相互之间的 XOR 校验，获得3个校验片，然后将6个分片存储到6个存储主机上，更新全局目录索引 GDI，如图5所示。

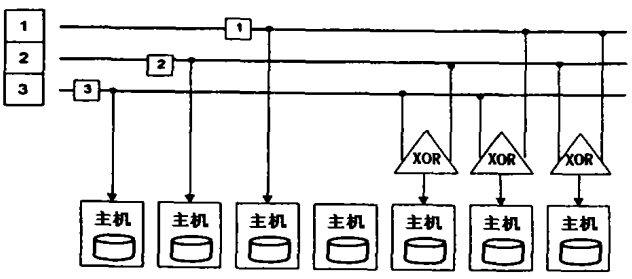

图5 活动主机数 M=7时的文件写过程

3.4 文件读

用户从全局空间读文件时，需从网络中各节点读出相应的分片，然后将各分片重新组合，恢复成原始文件。其基本步骤是：1、通过文件名查询全局目录索引 GDI，获得该文件的数据分片和校验分片的存储信息，得到文件存储节点表；2、与活动主机列表 AHL 进行比较，取交集得到可用的存储节点表；3、从各存储节点读分片或校验分片；4、将分片重新组合成原始文件[6]。

读取文件时，为了减少校验计算量，提高文件读的速度，采用尽量先读原始分片数据的策略，在原始分片数据无法重组的情况下，再读校验分片数据，利用 XOR 的逆运算是 XOR 的特性，通过 XOR 计算，得到完整的分片数据，完成文件重组。

由冗余校验分片算法 RVSA 公式可知，考虑最坏的情况，当在线节点数最少为 C_n^2+1，最多 n−1个节点不在线时，都可以100%读出文件，这样可以得到文件可读最低依赖率 X：

$$X=\frac{C_n^2+1}{n+C_n^2}=1-2\frac{1-1/n}{n+1} \quad n=2,3,4,\cdots$$

当写文件节点数为 N，在线节点数 M>=X*N 时，文件可以完整读出。图6为文件写到6个节点，在线节点为3时的文件读过程。

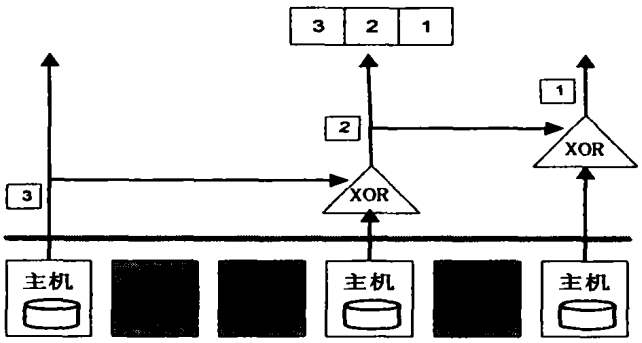

图6 文件写到6个主机，在线节点数为3的读过程

4 试验结果

4.1 容错性测试

我们使用有6台主机相联的以太网作为测试环境对 FTDSS 进行测试。试验中分别按当前节点数 n(n=3,4,5,6)进行文件写试验，然后以在线节点数为1,2,…,n−1的情况进行文件读测试。测试数据如下表。从测试数据看来，FTDSS 具有高容错性，只要2/3节点在线，文件都可读，克服了 RAID 的缺点。

文件写节点数	当前在线节点数	文件可读性	文件写节点数	当前在线节点数	文件可读性
3	1	0		3	0.5
	2	1		4	1
	3	1		5	1
4	1	0	6	1	0
	2	0.5		2	0.3
	3	1		3	0.9
	4	1		4	1
5	1	0		5	1
	2	0.3		6	1

4.2 性能测试

分别按当前节点数 n(n=3,4,5,6)对16M 文件进行读写试验，测试结果如图7，可以看出，文件读写性能随当前节点数

增多而提高，因为节点数越多，并发性越高。

以当前节点数为6对不同大小的文件进行读写试验，测试结果如图8，可以看出 FTDSS 读写性能比单机读写性能要高，并且文件越大，读写性能增幅越大，说明并发读写对大文件更有利。

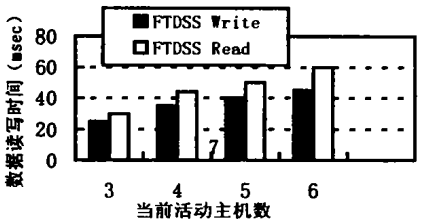

图7 FTDSS 读写随当前活动主机数的变化情况

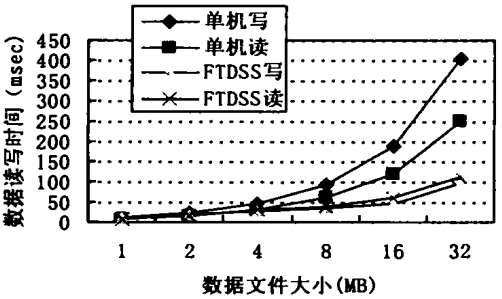

图8 FTDSS 的读写随文件大小的变化情况

结束语 本文提出了基于网络的分布式容错共享存储 FTDSS。FTDSS 将网络中各节点的磁盘空间组织成全局共享的存储空间，FTDSS 将文件以数据分片和冗余的校验分片形式分布式存储，只要一定数目的节点在线，就可以从全局空间完整读出文件，获得高的容错性，FTDSS 通过网络存储实现了数据远程容灾，FTDSS 将文件分片并行读写，获得比单机高的读写性能。因此，FTDSS 有很大的实用价值，FTDSS 可以在不添加任何硬件设备的基础上，在网络中实现高容错的共享存储。

参 考 文 献

1 Plank J S. A tutorial for Fault-Tolerate in RAID-like systems: [Technical Report UT-CS-96-332]. Department of Computer Science University of Tennessee, July 1996. 10～22

2 Mplero X, Silla F, Santonia V. Performance analysis of storage area networks using high-speed LAN interconnects. In: Proc. of the 8th ISPAN Conf. IEEE Computer Society. Sept. 2000. 5～9

3 Atchley S, et al. Fault-Tolerance in the Network Storage Stack. In: Proc. of the FAST 2002 Conf. Monterey, California, USA, Jan. 2002. 3～5

4 Tierney B L, Lee J, Crowley B, Holding M. A Network-Aware Distributed Storage Cache for Data Intensive Environments. In: Proc. of the FAST 2002 Conf. Monterey, California, USA, Jan. 2002. 15～18

5 Sobti S, et al. PersonalRAID: Mobile Storage for distributed and disconnected computers. In: Proc. of the FAST 2002 Conf. Monterey, California, USA, Jan. 2002. 5～8

6 Gibson G. Cost-effective high-bandwidth storage architecture. In: Proc. of ACM ASPLOS, Oct. 1998. 4～5

（上接第58页）

2. 秘密恢复时，一个用户只能知道其他合作用户的 Set_u，而不能知道其他人的 K_u，因此一个用户不能从一次秘密恢复中得到关于其他人手中的密钥信息，也不能从一次秘密恢复中得到有关另一次秘密分享中的秘密。

5.2 特点

1. 每个用户手中持有的是固定的 Set_u 和 K_u，因此进行多次秘密分享时不需要重新生成用户手中的信息，从而实现了动态的秘密分享。

2. 方案可以实现(k,n)门限的秘密分享，并且对最常见的(n,n)门限的情况的实现非常简单(使用单位矩阵形式的分配矩阵)。

3. 提出了一种同 Stinson 访问结构等价的，但更适合本方案的描述一般性不对称权限秘密分享的方法。

4. 方案可以对参与秘密恢复的用户动态地增加和删除，而不影响其他用户的信息。

5.3 性能分析

算法的时空复杂度是线性的，在秘密分享阶段需要 O(r×t)的时间和空间要求，在秘密恢复阶段每个用户最多进行 r 次解密运算，而恢复秘密只进行 r 次异或运算。

5.4 和其他秘密分享方案的比较

1. 文[1～3]的方案是一次性的秘密分享，本文的方案是可以反复使用而不需要更改用户手中的信息的，即所谓动态秘密分享。

2. 文[4～6]的方案可以实现(k,n)门限秘密分享，但是进行(n,n)形式是太过复杂，而本文的方案在(n,n)的情况下非常简单，使用的是单位矩阵型的分配矩阵(见第4节的分析)。

3. 文[7～8]的方案可以实现动态秘密分享，但是只适合于(n,n)形式，本文的方案可以同时满足这两种要求，并且文[7]中的幂指数运算计算量太大，而本文使用异或操作进行秘密的分享和恢复，计算量更易接受。

结论 本文提出的基于密钥矩阵的秘密分享方案，不但实现了秘密分享中各种基本要求，包括动态秘密分享，(k,n)门限，还提出并实现了不对称权限的秘密分享，可以动态增加删除参与秘密恢复的用户，并且要求比较小的时空复杂度。根据实际应用的需要，可以设计出更满足某种需要的秘密分享方案。

参 考 文 献

1 Shamir A. How to share a secret. Comm. ACM, 1979, 22(11): 612～613

2 Karnin E D, Greene J W. On secret sharing systems. IEEE Trans Inform Theory, 1983, IT-29(1): 35～41

3 Tompa M, Woll H. How to share a secret with cheaters. J Crypto, 1988, 1: 133～138

4 Tompa M, Blakley G R. Threshold schemes with disenrollment. In: Proc. Crypto 1992, Santa Barbara, CA, Aug. 1992. 13-1-13-4

5 He J, Dawson E. Multistage secret sharing based on one-way function. Electron Lett, 1994, 30(19): 1591～1592

6 Sun H M, Shieh S P. Construction of dynamic threshold schemes. Electron Lett, 1994, 30(24): 2023～2024

7 Pinch R G E. Online multiple secret sharing. Electron Letts, 1996, 32(12): 1087～1088

8 谭凯军，诸鸿文. 基于单向函数的动态秘密分享机制. 通信学报，1999. 7

9 Chor B, Fiat A, Naor M. Tracing Traitors. In: Proc. Advances in Cryptology-Crypto '94, Springr-Verlag LNCS 839, 1994. 257～270

10 Stinson D R. Cryptography: Theory and Practice, CRC Press, Boca Raton, 1995