基于分散式体系结构的高可靠文件存储系统的研究*)

蔡 斌[1] 谢长生[1] 朱光喜[2]

(华中科技大学武汉国家光电实验室(筹)信息存储研究部 武汉 430074)[1]

(华中科技大学电子与信息工程学院 武汉 430074)[2]

摘 要 提出了一种分散式体系结构的高可靠文件存储系统(DHAFS),各个存储节点相互协作,将本地的存储资源虚拟化为一个全局的存储空间,实现统一的文件名字空间,向客户端提供文件接口,存储、缓存、数据/元数据的管理功能分布在各个存储节点中。相对于现有的集群存储系统而言,DHAFS一方面弥补了单一元数据节点的单点失效,另一方面消除了单一元数据节点的性能瓶颈,提高了系统的动态可扩展性。测试实验结果证明DHAFS能够高效、稳定地提供文件存储服务。

关键词 容错,可靠性,分散式存储,全局存储空间,元数据

Research of High-available File Storage System Based on Decentralized Architecture

CAI Bin[1] XIE Chang-sheng[1] ZHU Guang-xi[2]

(Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China)[1]

(Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)[2]

Abstract This paper proposes a decentralized high-available file storage (DHAFS). Each storage node connects via high-speed network, and global virtual storage space is constructed using each node's local storage resources. Storage, cache, data/metadata management are distributed among each node, and nodes cooperate with each other to present a unified file name space to clients. DRSFSS provides high availability, eliminates the bottleneck of single metadata server and enhances the extensibility due to decentralized metadata. The performance measurements indicate that decentralized storage system has huge potential on the architecture.

Keywords Fault-tolerance, Availability, Decentralized storage system, Global storage space, Metadata

1 引言

信息时代,数据的价值越来越高,作为信息载体的存储系统已成为企业业务应用系统的关键组成部分。存储系统的主要任务就是要保证数据的可靠性,提高数据访问的性能,即使在存储设备发生故障时依然能够提供良好的数据访问服务。2001年的统计调查显示,1/4的被调查者估计数据丢失导致的损失达到了$250,000/h,8%的被调查者估计损失达到了$1,000,000/h[1]。Gartner组织估计大约2/5的企业都经历过数据灾难,而灾难之后的重新恢复则需要大约5年的时间。

随着磁盘容量的不断增大和价格的不断下降以及网络带宽的不断提高,通过低廉的磁盘设备和高速局域网组建高可靠、高性能的集群存储系统是网络存储领域的重要研究内容。现有的许多集群存储系统(如PVFS[2],GFS[3],zFS[4]等)是集中控制的体系结构,根据一个全局中心控制点(又称为元数据节点)来进行数据的分布和请求调度决策。由于这种单一元数据节点的体系结构,系统不可避免地存在元数据节点的单点失效和单一元数据节点的性能瓶颈。

针对文件存储系统,本文提出了一种分散式体系结构的高可靠文件存储系统(DHAFS),系统中没有专用的元数据节点,各个存储节点通过高速局域网相互连接,每个存储节点的本地存储资源虚拟化为一个全局的存储空间,存储、缓存、数据/元数据的管理功能则分布在各个存储节点中,存储节点相互协作实现统一的文件名字空间,向客户端提供文件接口。相对于现有的集群存储系统,DHAFS不仅弥补了单一元数据节点的单点失效,提高了存储系统的可用性,而且还能够支持客户端的并行访问数据,避免了元数据节点的性能瓶颈,提高了系统的动态可扩展性。

最后实现了DHAFS的原型系统并进行了性能测试,实验结果证明DHAFS能够高效、稳定地提供文件存储服务。

2 DHAFS的体系结构

2.1 分散式存储系统的特点

分散式体系结构的存储系统和集中控制的存储系统相比主要有以下几方面的特点。

(1)存储节点是基本的组成部件,它的价格比较低廉,性能比较适中,系统通常由许多存储节点组成,数据的访问是并行方式。

(2)每个存储节点只完成系统中的一部分功能,整个存储系统中没有集中控制点,系统的存储功能(如存储、缓存以及数据和元数据的管理)分布在各个存储节点中。

(3)能够提供很高的数据可靠性。数据在系统中的各个

*)基金项目:国家重点基础研究发展计划(973)(基金项目号2004CB318200)和华中科技大学博士后基金资助。蔡 斌 博士,目前从事博士后研究工作,研究方向为网络存储系统、存储管理技术、分布式系统;谢长生 教授,博导,研究方向为网络存储系统、计算机高速接口与通道、采用新原理的超高密度超高速存储技术;朱光喜 教授,博导,研究方向为现代移动通信网络与技术、图形图像处理等。

存储节点之间冗余存放，即使出现较高的节点故障率，也能够保证数据的恢复。

(4)系统具有动态可扩展性。随着时间的推移，不断有各种异构的存储节点加入或者退出，系统能够对它们自动进行管理，将它们各自的本地存储空间加入到全局的虚拟存储空间中，或者从全局的虚拟存储空间中移走，并且能够根据存储节点的特性，自动在节点之间平衡工作负载。

(5)简化了容量部署和配置，提高了存储系统的性价比。如果需要增加存储系统的容量，只需要向系统中增加更多的存储节点，系统将自动进行配置，不会中断用户的数据访问。

分散式体系结构的存储系统如图 1 所示[5]。

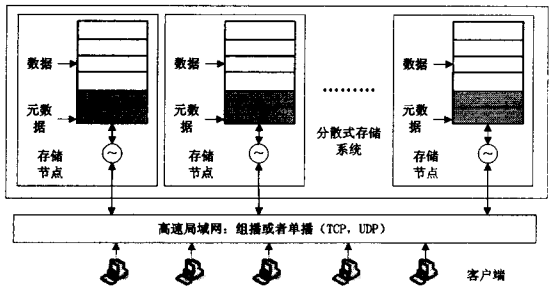

图 1　分散式存储系统的体系结构

2.2　DHAFS 的体系结构

DHAFS 系统是一个高可靠的、分散式的文件存储系统，对用户提供文件访问的接口。每个存储节点上都运行着一个后台进程，后台进程主要由 3 个部分组成：分布式文件系统层、本地文件系统层以及磁盘数据块层，如图 2 所示。分布式文件系统层对客户端呈现文件接口；本地文件系统层管理全局名字空间中的数据块资源(数据块资源可以是裸磁盘、磁盘分区，或者是本地文件系统中的大文件)，通过磁盘数据块层提供的块接口访问具有线性地址的数据块资源；最底层的是磁盘数据块层，提供数据块接口，本地文件系统和磁盘数据块层是节点之间的对等层，而分布式文件系统层则和客户端通信。

DHAFS 的本地文件系统层的作用是管理磁盘设备的线性地址空间。为了不失一般性，可以采用一种新的数据块寻址和 inode 节点编号机制来统一系统中所有的存储节点的本地文件系统，如图 3 所示。每个存储节点上的数据块和 inode 节点用一个 32 位的整数来表示，其中高 8 位表示存储节点的 ID 号。32 位的整数使得整个系统的存储容量可以达到 16TBytes，如果为了实现更大的存储空间，可以扩展到 64 位。通过这种机制，多个存储节点的本地文件系统被统一到一个全局的层次结构的文件名字空间中。

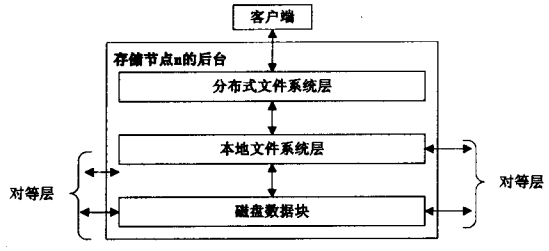

图 2　存储节点的后台进程结构

DHAFS 的每个文件或者目录都具有各自的文件句柄。可以使用 32 位的 inode 节点编号作为文件句柄，它能够定位到数据所在的存储节点。整个系统的“根目录”的句柄是非常特殊的，它是一个全 0 的 32 位的整数，称之为公共句柄。客户端能够从这个知名句柄开始执行查找(LOOKUP)操作。

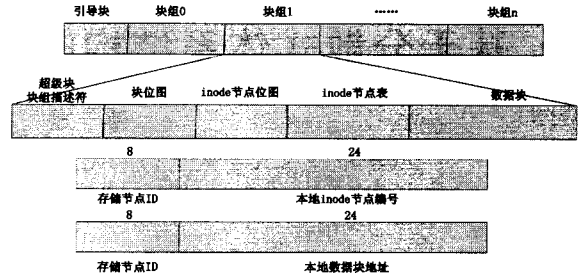

图 3　inode 编号与数据块寻址

在系统启动的初始阶段，每个存储节点从本地配置文件中读取配置信息，包括存储节点的 ID 号、组播 IP 地址、TCP 和 UDP 端口号以及各种缓存的大小等等。组播通信在系统的初始化阶段时使用，系统中所有的存储节点加入到这个组播组中，相互交换信息，并且建立起一个存储节点 ID 号到 IP 地址(ID-to-IP)的映射表，由所有加入的存储节点共享。当初始化阶段完成后，每个存储节点监听组播 UDP 端口以及各自的 UDP 端口，等待客户端的请求。组播组的存储节点也通过 TCP 协议建立一个全连接，方便内部的通信。

当客户端第一次进行数据访问时，它首先需要获得根目录。它通过公共句柄向组播地址发出一个 LOOKUP 请求，所有的存储节点将会收到这个请求，然后其中一个存储节点会响应客户端的请求。它会返回给客户端一个 ID-to-IP 表，通过文件句柄中的存储节点 ID 号，客户端就能够将请求直接发送到相应的存储节点上。

在提高数据的可用性和可靠性方面有许多可用的方法。最通用的一种方法是进行数据的镜像。这种方法虽然简单、易于实现，但是需要的存储空间的开销却是相当大的。数据校验是另一种非常流行的方法，如 RAID 机制，但是它只能够处理单个节点的故障，因此这种方法不适用于多个存储节点同时发生故障的情况。

Erasure coding 编码机制是一种前向纠错编码的方法(Forward Error Correcting FEC)，它将数据对象划分成 M 个数据段，然后将这 M 个片段重新编码为 N 个具有和前 M 片段大小相同的片段，并且满足 $N>M$。擦除编码机制的一个重要的特性就是：原始的数据对象能够从编码后的 N 个片段中的任意 M 个片段中重新构造出来，即可以支持 $N-M$ 个存储节点的同时故障。DHAFS 能够检测到发生故障的存储节点，使用 FEC 编码机制来选择需要的冗余级别，同时使用相应的纠错算法来恢复故障。

3　DHAFS 的设计与实现

DHAFS 设计时的主要目标如下：

(1)DHAFS 是一个无集中控制的分散式管理的文件系统，能够适应系统的动态扩展。

(2)DHAFS 能够提供很高的可靠性，即使在节点故障率很高的情况下，也能够很好的工作。

(3)DHAFS 能够利用每个存储节点的磁盘空间虚拟化为一个全局的可扩展的存储空间。

(4)DHAFS 对客户端提供文件接口。

(5)DHAFS 的网络是局域网环境，可以采用 IP 级的组播和基于 UDP 的流量控制对这种网络流量进行优化。

DHAFS划分为几个独立的模块，包括数据冗余模块、网络模块、数据存储模块、安全模块、组管理模块以及图形用户界面(GUI)模块。下面给出较为关键的读/写流程、纠错编码方法和基于组播的流量控制方法。

3.1 读流程

对于读请求，客户端首先向服务器组发送一个 GetCurrentFileList( )的组播消息，查询它所需要的版本和全局一致版本之间发生改变的数据。服务器组响应客户端的请求，通过服务器端的组管理模块来选择服务器。被选定的服务器向客户端发送一个 UpdateFileList( )消息，将最近发生改变的数据发送给客户端。然后客户端向服务器组发送一个 ReadFile( )的组播消息，这个消息中包含有一个全局唯一的事务ID号，事务 ID号是由文件内容的哈希值和用户的公有密钥来产生的。每个活动的服务器收到客户端的这个组播消息后，向客户端发送一个 ReturnFragment( )消息，将数据段返回给客户端。在读取数据段的过程时，客户端使用 TrafficControl( )流量控制消息来控制数据的传输率。

3.2 写流程

对于写请求，客户端首先向服务器组发送一个 GetCurrentFileList( )的组播消息，查询它所需要的版本和全局一致版本之间发生改变的数据。服务器组响应客户端的请求，通过服务器端的组管理模块来选择服务器。被选定的服务器向客户端发送一个 UpdateFileList( )消息，将最近发生改变的数据发送给客户端。然后客户端向服务器组发送一个 WriteFile( )的包含有事务 ID号的组播消息。客户端在写操作之前必须获得文件的写锁以避免多个并发的更新操作，因此在服务器端必须对文件加锁。如果在选定的服务器上对同样的事务 ID号没有其它的写操作发生，则选定的服务器发送一个 LockedFile( )消息给客户端，表明客户端可以开始进行写操作了。当客户端收到这个消息后，它就开始对服务器组中的文件数据段进行写操作。在服务器组中，后台运行着一个两阶段提交机制[6,7]的进程，以保证写入到服务器中的数据的顺序一致性。同时，服务器端也运行着流量控制进程，控制服务器和客户端之间数据传输的速率。

3.3 纠错编码

图 4 表明了编码机制。DHAFS 选择 Bose-Chaudhuri 编码机制中的一种，称为 Reed-Solomon 编码，来实现“数据冗余”模块。在具体实现中，采用 Luigi Rizzo 所实现的 Reed-Solomon 编码机制，将文件划分为一系列大小为 64kB 的数据条带，然后计算每个条带的校验数据。

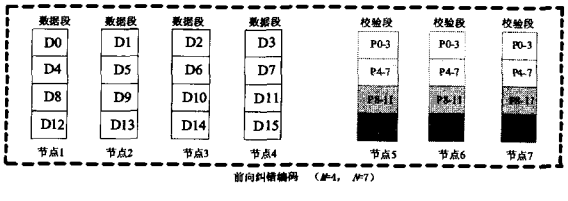

图 4 DHAFS的纠错编码

3.4 基于组播的流量控制

组播流量控制机制有两种工作模式：读工作模式和写工作模式。读工作模式在数据从服务器组传输到客户端的情况下；写工作模式在数据从客户端传输到服务器组的情况下。这两种工作模式都利用了系统中的纠错功能。通过这种方法，DHAFS能够在处理器的占用率和网络传输率之间达到平衡。下面的伪代码描述了读工作模式：

```
1) Receive(fragment, stripeID, from);
2) if(stripe is not yet processed){
3}  StoreFragmentInQueue();
4)   CheckQueue();
5) } else{
6} Drop(fragment);
7) }
8) if(the Queue occupation is over 20%){
9}   SendFlowControlInformation();
10) }
```

CheckQueue 函数的伪代码如下：

```
1) if(there are more than N data fragments for the same stripe){
2}  if(we have every data fragments){
3}  SendAlertToControler();
4)   SetTheProcessedFlag(stripeID);
5)  } else{
6}   StartErrorCorrection(stripeID);
7)  }
8) }
```

在读模式中，客户端从服务器接收文件的数据段，并且将它们保存在一个接收队列中。如果客户端接收到足够的数据段能够进行前向纠错，则客户端立即开始纠错处理(如CheckQueue( )函数中的第 6 行)。当客户端完成了纠错之后，它向控制模块发送一个“接收完成”的消息(RECEIVE_COMPLETE)，并且标记该条带为“已处理”(PROCESSED)的状态(如 CheckQueue( )函数中的第 3 和第 4 行)，条带中的其它的数据段就停止传送了(如读工作模式伪代码中的第 6 行)。使用这样的方法，能够避免瓶颈节点不断下降的数据传送速率对整个文件读取操作的影响，并且能够进行节点数据段的纠错处理，提高文件的可用性。

写模式的工作原理和读模式的基本相同，它也能够处理服务器故障，并且也能够避免瓶颈节点不断下降的数据传送速率对整个文件的写操作的影响。客户端使用单播 UDP 数据报将数据段写入到不同的服务器中，每个服务器都有一个接收队列。当服务器发现它本地的接收队列的剩余空间过小时，该服务器就向客户端发送流量控制信息，该信息中包含有它期望的传输速率。客户端收到服务器发送的流量控制信息后，它就决定是否接受服务器的流量控制请求，还是以更高的速率进行传送。客户端对服务器的流量控制请求作出的决定是在其它的服务器的响应的基础之上的，即客户端选择的传输速率对于大多数服务器而言是可以接受的。

3.5 DHAFS的实现

在实现 DHAFS 的原型时，选择 Window 作为操作系统平台。图 5 描述了 DHAFS 客户端的实现框架结构。在客户端运行着两个线程：网络线程和综合线程。网络模块单独运行于网络线程之中，客户端中的其它模块则运行于综合线程之中。网络线程的任务是接收数据包并且将它们保存在一个接收队列中(同步等待队列)。客户端综合线程则从这个同步等待队列中取出数据包。如果队列为空，则客户端综合线程就处于等待状态，网络线程在将数据包放入到同步等待队列中将唤醒客户端综合线程。客户端进行读/写请求时，对每个服务器使用异步方法调用的方式进行，文件的数据段的传输将会以并行的方式进行。

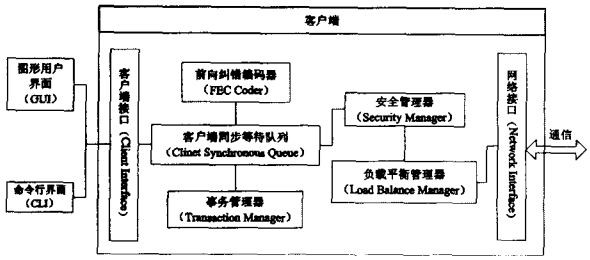

图 5　客户端的实现

图 6 描述了服务器端的实现框架结构。服务器端使用了一个数据库引擎和服务器/服务器智能模块。在服务器端运行着 4 个线程：网络线程、服务器/服务器线程、服务器/客户端线程以及一个心跳线程。前 3 个线程和客户端上的基本上是类似的，只是在服务器端设置了两个等待队列：一个队列是为服务器/服务器线程设置的，另一个是为服务器/客户端线程设置的。网络线程根据收到的数据包的目的地址的类型，分别将它们放入到不同的等待队列中。心跳线程则周期性地发送心跳信号。

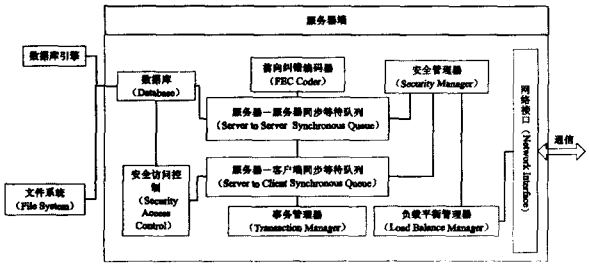

图 6　服务器的实现

4　读/写性能与可靠性评测

为了测试 DHAFS 的性能，我们使用 5 台 PC 机作为存储节点组成存储系统，每台 PC 机的配置是相同的，通过 100Mbit/s 的以太网卡连接，其中一台作为客户端。每台机器的配置为：Intel Celeron 1.8GHz、256MB 内存、Segate IDE 80GB 硬盘、Windows 2000 操作系统。

有关 UNIX 文件系统的研究表明，文件大小是服从 $a=1.05$ 和 $k=3800$ 的有限排列分布[8]。有关 Windows 文件系统的研究表明通过一个对数正态分布和一个两步对数正态分布的修正可以建立起文件大小的分布模型[9]。为了简化而又不失一般性，在进行测试时，只使用了重尾分布的经验函数，参数 $a=1.05$，$k=3800$。

Andrew benchmark[10]是一种用于测量 MakeDir，Copy，ScanDir，ReadAll，Compile 等文件访问任务所需要的时间。由于在 Windows 的图形用户界面中不能够直接使用 Andrew benchmark 的脚本，因此使用下面的替代方法：首先创建一个应用程序，由它来生成文件和目录，使得这些文件大小符合有限排列分布，目录深度符合线形分布，然后通过客户端的 GUI 界面对这些文件和目录进行读/写操作。使用 Windows 的 CIFS 网络文件系统作为对比的对象。在 DHAFS 中使用的 Erasure coding 编码率为 3/4。在 CIFS 中，使用一台服务器节点作为 CIFS 文件服务器。

下面进行的实验是 DHAFS 单个 I/O 入口节点情况下与 CIFS 的性能比较。由于分散式存储系统节点之间的通讯开销，单个节点入口的系统性能必然会下降，但这个下降的程度对我们了解整个系统的潜力是非常必要的，一旦采用多个节点，性能就会有成倍的提高。在进行测试时，由于考虑的是动态的环境，服务器组向客户端发送了所有的数据段（包括校验数据段），会占用一部分网络带宽，但对了解纠错处理对系统性能的影响也是非常有必要的。

首先测试在正常情况下（即 4 台存储服务器都正常运行），分三组测试 DHAFS 和 CIFS 文件系统的性能：

(1)创建目录的延时和删除目录的延时。在这个测试中，使用替代的方法创建 1,000 个目录，然后通过客户端 GUI 图形界面在 DHAFS 中执行，以及通过 CIFS 文件系统在 CIFS 服务器中执行。比较的结果如表 1 所示。

表 1　目录操作的性能比较

操作	DHAFS 文件系统延时(秒)	CIFS 文件系统延时(秒)
创建目录	1082	602
删除目录	203	131

(2)小文件的读/写的延时，以及吞吐量。使用替代的方法产生 10,000 个文件，文件大小服从参数 $a=1.05$，$k=3800$ 的有限排列分布，聚合文件的大小为 36.5MB。比较的结果如表 2 所示。

表 2　小文件的读/写性能比较

	DHAFS 文件系统		CIFS 文件系统	
操作	延时(秒)	吞吐量(MB/s)	延时(秒)	吞吐量(MB/s)
文件读	2641	0.013	339	0.108
文件写	13568	0.003	422	0.085

从上面的测试结果可以得到下面的结论：对于小文件，DHAFS 比网络文件系统要慢 2 个数量级，这是因为 DHAFS 中每个操作是在独立的事务中处理的，并且在每个事务处理完成之后，还需要表决机制来表明处理的成功与否。但是，这种差距是可以在很大程度上减小的。第一，当进一步利用分散式的结构提供多个入口节点时，性能会成倍数的提高；第二，可以采用成批处理操作方法：当写一个目录时，对整个操作分配一个事务，而不用对每个单独的操作分配一个事务。如果使用成批处理以及客户端缓存，小文件的读/写性能将会进一步提高；第三，当多个客户端并发进行数据存取时，性能会有大幅度的提高；第四，如果在稳定的网络环境中，当需要进行纠错处理时才发送纠错数据段，性能还会进一步提高。

(3)大文件的读/写的延时，以及吞吐量。使用 $a=1.05$ 和 $k=114,000,000$ 来生成文件。总共产生 20 个文件，聚合文件的大小为 2.14GB。和前面的测试步骤相同，测试了文件写和文件读时的性能，结果如表 3 所示。

表 3　大文件的读/写性能比较

	DHAFS 文件系统		CIFS 文件系统	
操作	延时(秒)	吞吐量(MB/s)	延时(秒)	吞吐量(MB/s)
文件读	627	3.51	294	7.45
文件写	985	2.23	434	5.05

从上面结果可以看出，在分散式结构下，采用单个入口节点，在单个客户端访问的情况下，其性能与传统文件系统差距已经不大，这里的差距主要是节点之间的通信开销造成的。在通讯开销已经发生的情况下，如果采用多个入口节点，分散式存储系统的大文件存取性能必将大大超过传统的分布式文件系统。

（下转第 99 页）

$|\Psi_{12}^{+}\rangle \rightarrow -a|0_3\rangle + b|1_3\rangle = -Z|\phi_i\rangle$

$|\Psi_{12}^{-}\rangle \rightarrow -a|0_3\rangle - b|1_3\rangle = -I|\phi_i\rangle$

然后 Alice 将测量结果通过经典信道(例如打电话)告诉 Bob,Bob 根据得到的消息分析自己手中的粒子 3 处于哪个状态,最后根据相应的么正变换得到 $|\phi_i\rangle$。我们把根据已知消息进行相应么正变换的过程称为量子解调。这样,量子比特 $|\phi_i\rangle$ 就传给 Bob 了。Bob 再进行译码操作,最后还原为初始的量子信息。在整个隐形传态过程中,原来的量子状态 $|\phi_i\rangle$ 通过 Bell 基测量已经塌缩到 Bell 态中的一个,因此隐形传态并不违反未知量子态不可克隆原理。

结束语 本文提出了两种基本的量子通信系统模型:量子直接通信模型和量子隐形传态模型。在量子直接通信模型中,整个通信过程被划分成不同的模块,并描述了每个模块的功能与作用。在量子隐形传态通信模型中,描述了用纠缠对作为信息载体传送消息的过程。量子通信系统在安全性及效率方面具有经典系统无法比拟的优势,在不远的将来,量子通信系统必将走向实用化的道路。

参 考 文 献

[1] Nielsen M A, Chuang I L. Quantum Computation and Quantum Information. Last modified. Cambridge: Cambridge University Press, 2000: 206-270

[2] 李承祖. 量子通信和量子计算. 第一版. 长沙:国防科技大学出版社, 2000: 73-143

[3] Shor P W. Scheme for reducing decoherence in quantum memory. Phys. Rev. A, 1995, 52(4): 2493-2496

[4] Calderbank A R, Shor P W. Good quantum error - correcting codes exist. Phys. Rev. A, 1996, 54(2): 1098-1105

[5] Steane A M. Multiple particle interference and quantum error correction// Proc. Roy. Soc. Lond. A, 1996, 452(1954): 2551-2577

[6] Gottesman D. Class of quantum error - correcting codes saturating the quantum Hamming bound. Phys. Rev. A, 1996, 54(3): 1862-1868

[7] Calderbank A R, Rains E M, Shor P W, et al. Quantum error correction and orthogonal geometry. Phys. Rev. Lett., 1997, 78(3): 405-408

[8] 王新梅,肖国镇. 纠错码——原理与方法. 修订版. 西安:西安电子科技大学出版社, 2001: 378-415

[9] Grassl M, Geiselmann W, Beth Th. Quantum Reed - Solomon Codes. AAECC-13. Honolulu, Hawaii, USA, 1999

[10] MacKay D J C, Mitchison G, McFadden P L. Sparse-graph codes for quantum error correction. IEEE Transactions on Information Theory, 2004, 50(10): 2315-2330

[11] van Loock P, Ladd T D, Sanaka K, et al. Hybrid Quantum Repeater Using Bright Coherent Light. Physical Review Letters, 2006, 96(24): 501-505

[12] Holevo A S. The capacity of the quantum channel with general signal states. IEEE Transactions on Information Theory, 1998, 44(1): 269-273

[13] Gordon G, Regolin G. Generalized Teleportation Protocol. Physical Review A, 2006, 73(4): 2309-2312

(上接第 90 页)

DHAFS 也能够提供很好的数据可用性,能够容忍服务器发生故障。以上的三组测试中使用的编码率为 3/4,在此基础之上测试一台服务器宕机的情况下,DHAFS 的纠错性能,结果如表 4 所示。从这个测试结果可以看到系统对大文件的读性能和传统网络文件系统已经很接近了,如果在稳定的网络环境中,当需要进行纠错处理时才发送纠错数据段,能够达到比传统网络文件系统更高的吞吐量。

表 4 DHAFS 的容错性能

	DHAFS 文件系统	
操作	延时(秒)	吞吐量(MB/s)
小文件读	1256	0.03
大文件读	312	7.03

结束语 随着磁盘容量的不断增大和价格的不断下降以及网络带宽的不断提高,通过低廉的磁盘设备和高速局域网组建高可靠、高性能的集群存储系统是网络存储领域的重要研究内容。现有的许多集群存储系统是集中控制的体系结构,根据元数据节点来进行数据的分布和请求调度决策。由于这种单一元数据节点的体系结构,系统不可避免地存在元数据节点的单点失效和单一元数据节点的性能瓶颈。

针对文件存储系统,本文提出了一种分散式体系结构的高可靠文件存储系统(DHAFS),系统中没有专用的元数据节点,各个存储节点通过高速局域网相互连接,每个存储节点的本地存储资源虚拟化为一个全局的存储空间,存储、缓存、数据/元数据的管理功能则分布在各个存储节点中,存储节点相互协作实现统一的文件名字空间,向客户端提供文件接口。相对于现有的集群存储系统而言,DHAFS 不仅弥补了单一元数据节点的单点失效,提高了存储系统的可用性,而且还能够支持客户端的并行访问数据,避免了元数据节点的性能瓶颈,提高了系统的动态可扩展性。

参 考 文 献

[1] Online Survey Results: 2001 Cost of Downtime. Eagle Rock Alliance Ltd., August 2001, accessed May 2003. http://contingencyplanningresearch.com/2001%20Survey.pdf

[2] Carns P H, Ligon W B. PVFS: A Parallel File System for Linux Cluster. Linux Journal, November 2000

[3] Matthew T, Keefe O. Shared File Systems and Fibre Channel// Proceeding of the 15th IEEE/6th NASA Goddard Conference on Mass Storage Systems and Technologies. College Park, Maryland, USA, March 1998: 1-16

[4] Rodeh O, TepermanA. zFS-A Scalable Distributed File System Using Object Disks// Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies. San Diego, California, USA, April 2003: 207-218

[5] Xie Changsheng, Cai Bin. A Decentralized Storage Cluster with High Reliability and Flexibility// Proceedings of the 14th Euromicro International Conference on Parallel, Distributed and Network-based Processing. Montbeliard, Sochaux, France, February 2006

[6] Lampson B, Lomet D. A New Presumed Commit Optimization for Two Phase Commit// Proceedings of the 19th International Conference on Very Large Data Base. August 1993: 630-640

[7] Liu M L, Agrawal D, El Abbadi A. The Performance of Two Phase Commit Protocols in the Presence of Site Failures. Distributed and Parallel Databases, 1998, 6(2): 157-182

[8] Crovela E, Taqqu M S, Bestavros A. Heavy-tailed Probability Distributions in the World Wide Web. a Practical Guide to Heavy Tails. New York: Chapman & Hall, 1998: 3-26

[9] Douceur R, Bolosky W J. A Large-scale Study of File System Contents// Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. Atlanta, Georgia, USA, May 1999: 59-70

[10] Kazar H M, Menees S, et al. Scale and Performance in a Distributed File System. ACM Transactions on Computer Systems, 1988, 6(1): 51-81