一种基于日志的异步远程镜像协议的设计*)

刘卫平　蔡皖东
(西北工业大学计算机学院　西安 710072)

摘　要　远程镜像是一种有效的数据容灾技术。本文设计了一种基于日志的异步远程镜像协议,通过写请求分批传播的机制,减少了通信链路传输的数据量。同时给出了一种批请求的原子提交机制,从而避免了写顺序不一致引起的主存储系统与从存储系统之间数据视图不一致。该协议在保证对应用的写请求具有较好响应速度的时候,也能够很好地保持镜像系统的数据一致性。

关键词　异步镜像,日志,批请求,原子更新

A Design of Asynchronous Remote Mirroring Protocol Based on Log

LIU Wei-Ping　CAI Wan-Dong
(College of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072)

Abstract　Remote Mirroring is an effective technology for disaster tolerance. This paper describes an asynchronous remote mirroring protocol based on log, which decreases the workload of network by using write coalescing and asynchronous propagation. This paper also designs an atomic update schema that ensures uniform date snapshot between primary storage system and secondary storage system. The protocol ensures that storage system can rapidly response I/O request, while ensures data consistency between primary and secondary storage system.

Keywords　Asynchronous mirroring, Log, Batch request, Atomic update

1　引言

远程镜像是一种有效的数据容灾技术。在一个远程镜像系统中,至少需要有一个主存储系统(主端)和一个从存储系统(从端)。主端与从端之间通过通信链路进行连接,主端负责响应上层应用的读写请求以及与从端进行镜像通信。远程镜像主要有两种运行模式[1]:(1)同步模式。应用每发出一个写请求,必须等到数据完全写入主端存储设备和从端存储设备之后,才能继续执行。该模式能使系统保持每一个 I/O 的精确状态信息,可以保证主端和从端的数据一致性。但由于通信链路传输的延时,使得对写请求的响应性能较差,因此适用于通信链路较短的情况。(2)异步模式。应用发出的写请求,只需等待数据写入主端存储设备之后即可继续进行,此后再由主端与从端进行数据同步,即从端和主端之间允许滞后多个写请求。该模式对写请求的响应性能较好,适用于通信链路较长的情况,但是在某些故障情况下会出现主端与从端数据不一致的状况。综合考虑系统性能和数据可用性,异步镜像模式更适合于远程数据容灾。

目前各主要的存储厂商都提供了不同类型的远程镜像产品,如 EMC 的 SRDF[2]、VERITAS 的 VVR[3]、IBM 的 PPRC 和 XRC[4]等。同时针对远程镜像协议的研究也有很多,文[5]对现有的远程镜像技术指标进行了分类和规范,阐明了其中涉及的一些模糊的概念,同时提出了一种远程镜像协议 Seneca。通过对上述产品和技术分析,本文设计了一种基于日志的异步远程镜像协议,使用日志机制来保证数据一致性,使用批请求传播来降低传输负载,同时设计了批请求的原子提交机制来避免写顺序不一致引起的主端与从端的数据视图不一致。

2　异步镜像协议

2.1　设计要求

在异步镜像系统中,连续多个写请求只需等待数据写入主端存储设备之后即可继续进行,此后再由主端与从端之间按照某种机制进行数据同步。该模式对写请求的响应性能较好,但是可能出现两端存储设备之间数据不一致的情况。所以设计异步镜像协议需要考虑以下几个问题,从而保证镜像协议在实现高性能镜像的同时,也应该尽可能地保证两端存储设备的数据一致性[6]。

首先是可恢复性。在任意时刻 t,如果存在时刻 $t'<t$,使得从端在时刻 t 所呈现的数据视图等于主端在时刻 t' 的数据视图,即从端的数据视图必然为主端在过去某一时刻的数据视图,则认为镜像系统的数据是可恢复的。其次是收敛性。在任意时刻,如果停止主端的写请求,而镜像系统的其它组件正常运行,那么镜像系统必须能够在有限的时间内把主端已经发生的数据更新反映到从端,从而使两端存储设备中的数据完全一致。同步所需要的时间越短,说明镜像系统的收敛性越好。最后是故障时的数据丢失量。异步镜像对从端的更新总是滞后于主端。因此当主端由于某种灾难而失效时,会有一部分已经在主端更新的写请求没有传播到从端,导致从端的数据视图与失效前主端的数据视图不一致。如果主端不能恢复,意味着有一部分数据将永远丢失。所以,在采用某种机制将写请求传播到从端时,要尽量减少可能的数据丢失量。

*)航空基础科学基金项目(项目号:03F53031)和西安市工业攻关项目(项目号:GG200312)。**刘卫平**　在读博士生,主要从事网络容灾、网络存储、信息安全等方面研究;**蔡皖东**　教授,博士生导师,主要从事计算机网络、网络信息安全、网络容灾等方面研究。

2.2 协议设计

应用程序的访问请求最终表现为对存储设备逻辑块的I/O请求。当执行读请求时，系统直接对主端存储设备进行读取并迅速得到响应，访问请求不需要经过通信链路传输到从端，因此不存在访问延时。而执行写请求时，系统在对主端存储设备进行写操作时，也要对从端存储设备进行相同的操作，即该写请求要经过某种通信链路传输到从端。异步镜像只要求写请求得到主端存储设备的确认，就可执行下一个写请求，保证了系统对写请求的快速响应。但主端与从端之间写请求的异步执行，不能够完全保证两个存储系统之间的数据一致性。因此异步远程镜像协议需要实现写请求的远程传播以及数据同步。本文设计了一种基于日志的异步远程镜像协议，该协议在主端和从端都采用基于日志机制的两阶段提交思想，较好地实现了性能和数据一致性的折衷。该协议的写请求执行以及传播流程如图1所示。

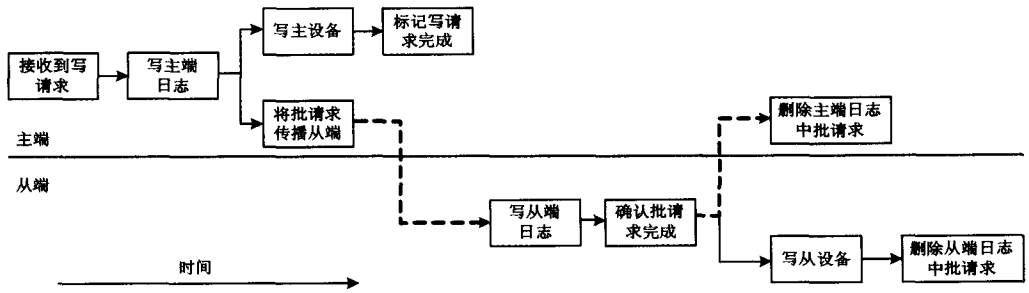


图1 半同步镜像协议数据更新流程

主端收到来自应用的写请求，首先将该写请求记录到主端日志中。接着在主端存储设备上执行该写请求，将数据写入相应区域，然后将写请求执行完成的确认信息返回给应用。此后，系统可继续处理来自应用的下一个写请求。与Seneca协议类似，为了减少向从端传播写请求所产生的网络数据量，本文给出的协议也采用批请求提交的方式来向从端传播写请求。即并不是在主端执行的每一个写请求都会触发网络传输，而是根据某种机制将记录在主端日志中的写请求集中为一批发送到从端。这种批请求传播的机制可以把一批写请求中的重复写请求进行合并，可以减少传播负载。

从端在收到批请求之后，首先将其记录在从端日志中。确认批请求接收完成后，即可向主端发出接收完成的应答，同时依次执行批请求中的每一个写请求。批请求执行完成后，即可删除从端日志中的相应记录。此外，主端收到来自从端的批请求，完成应答后，从主端日志中删除相应的批请求记录。至此，一批写请求更新至从端，且从端此时的数据视图满足可恢复性条件。此外，如果主端在某一时刻停止接收写请求，之前的写请求可以集中成批传播至从端并顺序执行，可以达到与主端存储设备完全相同的数据视图，因此该协议也满足收敛性条件。

3 协议中的几个关键问题

本协议中用到了日志、写请求分批传播等机制，因此复杂度比传统的异步镜像协议有所增加。为了保证协议的正常运行，需要对其中的一些关键技术进行具体的定义和规范，使得协议在实现高性能的同时也具有高可靠性。

3.1 日志机制

日志是一组记录构成的序列，用于保存以往访问和更新的信息，为在将来需要查阅或者在系统崩溃时恢复系统所用。日志机制在操作系统、数据库系统、文件系统等不同领域已得到广泛的应用。本文提出的镜像协议中的日志机制借鉴了传统的日志机制的一些思想，同时具有以下一些特点。

首先，本协议在主端和从端都采用了基于日志的两阶段提交机制。即在收到写请求后，不仅需要把写请求的描述信息（设备号、SCSI CBD、起始地址、长度等）记录到日志中，还要记录写请求的数据内容，以便充当数据重建时的数据源，等到上述内容在日志中记录完成后才提交给具体设备。而日志要等到写请求在具体设备上执行完成后才可以删除。由此可知，本协议的日志容量远大于文件系统或数据库系统的日志容量，所以本协议要求日志记录占用一个完整的设备（如一个逻辑盘或一个磁盘分区）。同时，为了提高系统的性能，记录日志的设备与存储数据设备不应该位于同一个物理设备上。

其次，本协议对写请求分批进行传播。在日志中记录写请求的时候，还需要采取一定的机制，将写请求序列划分批次。因此，将日志空间分为两部分：一部分为控制信息区，存储目前日志中已经划分的批次及相关信息，每一条记录应该包括为每个批次设定的唯一标识，及每个批次中包含的写请求个数。另一部分是写请求信息区，保存具体的写请求，每一条记录应该包括一个写请求的描述信息和具体的数据内容，同时标明该写请求分别属于哪一个批次。如图2所示，控制信息区中有两个已经划分好的批，分别用a、b来标识，其中批a包含4个写请求，批b包含3个写请求。而在写请求信息区中，写请求以队列的形式顺序排列，并标明各自属于哪一个批次。

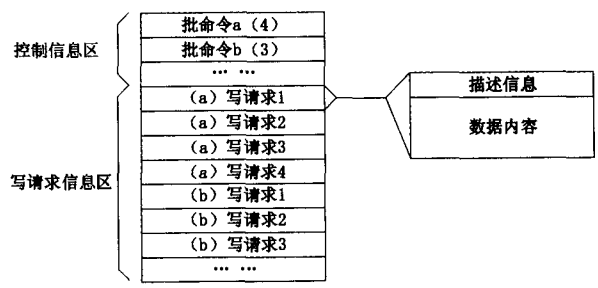


图2 日志结构图

每一个写请求都按照图2的格式记录在日志盘中。根据日志盘的容量或者用户的要求设定日志空间的大小。日志空间中可保存的最大记录数就是可在得到从端应答之前可并发执行的写命令的最大个数。如果日志空间已经满，那么写命令将被阻塞。直到有命令执行完后，删除日志中的相应条目，才可继续执行。

3.2 写请求分批传播机制

异步镜像中存在这样一种情况：主端存储设备中的一个逻辑块已经被写请求更新，但是这个更新还没有传播到从端。如果此时主机又生成一个写请求，再次更新该逻辑块，这样的操作被称为写冲突。对于存在写冲突的两个或者多个写请求，只传播其中最后一个写请求到从端，可以减少通过通信链路传输的数据量。而传统的异步镜像协议不考虑写冲突现象的存在，严格地将每一个写请求传播到从端，对于存在写冲突的多个写请求，也需要依次进行传播，浪费了通信链路资源。

写请求分批传播机制可以用来解决这个问题。将连续的写请求按照某种机制划分为批请求，这样的一个批请求中会存在着不同程度的写冲突。对于存在写冲突的写请求，进行覆盖写，即只保留其中的最后一个。经过覆盖写处理后的批请求中包含的写请求个数少于原始批请求包含的写请求个数。将经过优化的批请求传播到从端，可以减少通过通信链路传输的数据量。当然，对于不同批请求中出现对同一个块的重复写操作，不允许进行覆盖写，否则会导致数据的不可恢复。

通常根据时间划分批请求，即将一定间隔时间内产生的写请求集中为一个批请求[5]。Patterson 等人对覆盖写减少的数据传输量做过深入研究[7]，他们对各种文件系统环境下的应用(Web 页、数据库等)进行测试，结果表明，当时间间隔为 1min 时，能够减少的数据传输量为 10%～20%；当时间间隔为 15min 时，减少的数据传输量为 30%～80%；当时间间隔超过 60min 时，数据传输量的减少幅度趋于平缓。

但是主机生成写请求的速率是动态的。如果按照固定的时间间隔 T 来划分批请求，会导致以下两种情况：当主机生成写请求的速率较高或者时间间隔 T 过大时，在 T 内产生的写请求数量 S 也较大，其中包含多个写冲突，覆盖写后，传输给从端的批请求中所包含的写请求个数会大大地小于 S，因此减轻了链路负载。但是，该批请求在传播至从端的过程中，如果主端因为某种灾难而失效，会丢失较大数量的数据。当主机写请求的速率较低或者时间间隔 T 过小时，在 T 内会产生的写请求数量 S 会较小，其中包含的写冲突也较少，覆盖写后，传输给从端的批请求中所包含的写请求个数仅略小于 S，减轻链路负载的效果并不明显。但是，该批请求在传播至从端的过程中，如果主端失效，丢失的数据量也较少。这说明采用固定的时间间隔来划分批请求，减轻的链路负载量和故障时的数据丢失量都是动态的，与生成写请求的速率成正比，这不利于系统的评价和分析。

本文按照写请求的数量来划分批次。即将连续产生的 S 个写请求集中为一个批请求，其中包含写冲突，覆盖写后，传输给从端的批请求中所包含的写请求个数会小于 S。下面根据一个实际的应用系统 cello2002 来分析在 S 不同取值情况下，链路负载和数据丢失量的分布情况。Cello2002 在正常运行状态下，每分钟产生约 4000 个写请求，每 30min 产生约 120000 个写请求。当 S 的取值为 0～120000 时，链路负载和数据丢失量的变化如图 3 所示。随着 S 取值的增大，需要传播到从端的数据量大幅减少，减幅达原始数据量的一半以上。但同时如果发生故障，数据丢失量也会大幅增加。这两项指标的变化幅度与传统的固定时间间隔分批机制下所得到的结果完全一致。但在基于写请求个数的分批机制下，如果主机生成写请求的速率较高，可在较短的时间内生成一个批请求；如果主机生成写请求的速率较低，则会在较长时间内生成一个批请求。即使任何时刻主端因为某种灾难而失效，丢失的数据量都是比较固定的。这表明在 S 确定的情况下，无论主机生成写请求的速率有何变化，系统的链路负载和数据丢失量都是稳定的。

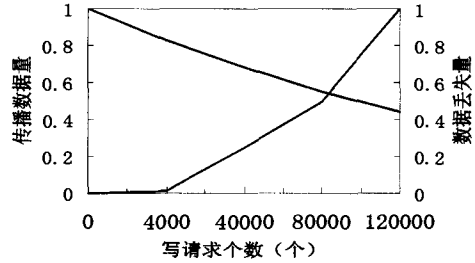


图 3 批请求效率示意图

异步镜像协议要能够在通信链路负载和可能丢失的数据量之间作出动态的权衡。因此，基于写请求数量的分批机制有助于在较低的网络传输成本和可接受的数据丢失量之间作出平衡。

3.3 批请求的原子提交机制

写请求分批传播的机制虽然节约了通信链路带宽，但也使得从端与主端数据视图之间会滞后多个写请求。这些写请求经过覆盖写后，在从端存储设备被提交的顺序可能会不同于主端，从而可能导致两端数据不一致，如图 4 所示。主端应用产生了一个写请求序列依次写逻辑块 A、B、C、D、B、E、D、B。该写请求序列完全按照实际产生的次序提交给主存储设备执行。初始的写请求序列经过覆盖写后可以变为一个优化的写请求序列 A、C、E、D、B，然后传播给从存储系统。但是这个优化的写请求序列在传播至从端的过程中，如果主端停机，则该批请求只会有部分传播到从端。此时从端所呈现的数据视图有可能是主端在任何时刻都不曾出现过的。

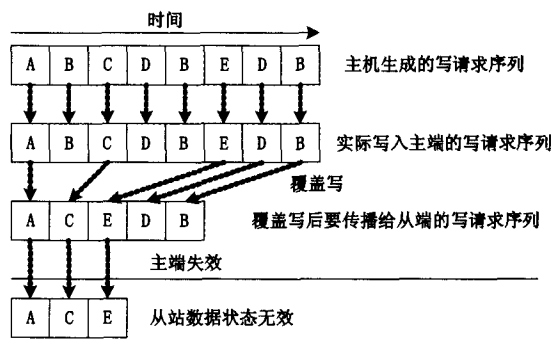


图 4 写顺序不一致导致的数据视图不一致

解决从端写顺序不一致问题需要满足以下几个条件：首先要保证批请求的传播是原子操作，即同一批内的写请求到达从端的顺序可以与提交的顺序不同，但要保证这些命令或者全部提交成功，或者一个也不提交。其次是从端收到的批请求在提交给存储设备执行时也要与主端保持一致，即前一个批请求在从端完成全部写盘操作之前，后一个批请求不允许在从端写盘。本文设计了满足上述条件的主端与从端之间的协商机制，如下所述。

在主端，规定将连续产生的 S 个写请求组成一个批请求。每开始一个批请求，生成一个时间戳 Stamp 来标识该批请求。所有写请求都依次写入到主端日志的写请求信息区，每个写请求为一个日志记录。同时为写入的每一个记录添加一个批请求标识字段，内容为当前批次的时间戳 Stamp，标识

该写请求属于某一个批请求。同一个批请求中的多个写请求在出现写冲突时可以按照覆盖写机制进行合并，只保留最新的一。合并后的批请求按照原来的时间顺序存放在写请求队列中。连续产生 S 个写请求后，统计该批请求中覆盖写后生成的写请求队列中的写请求个数，并将批请求的标识时间戳 Stamp 和该批请求中写请求的个数存入主端日志的控制信息区中，然后将该批请求中的全部写请求以及控制信息区中的相关信息传播至从端。见图 5(a)所示。

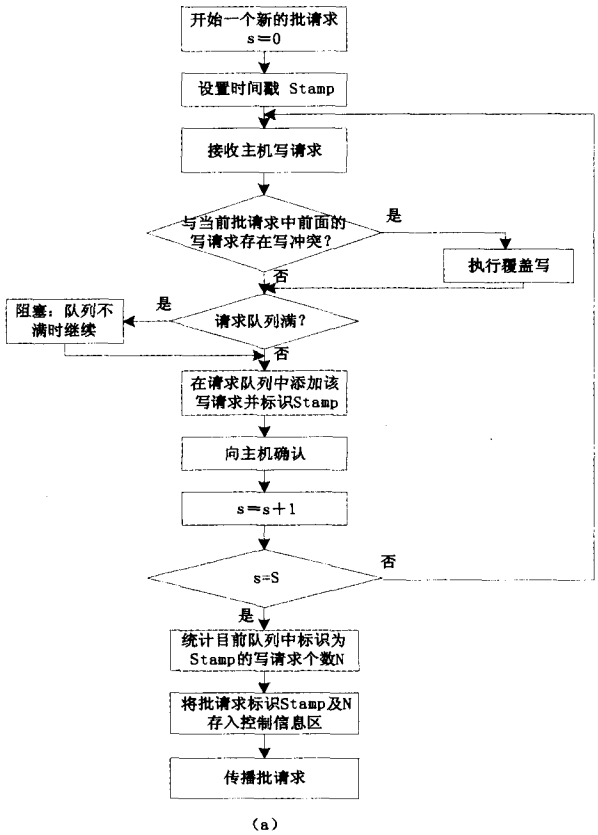
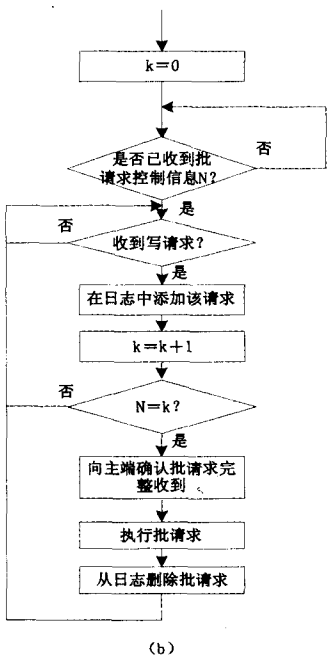


图 5 (a)主端原子提交流程



(b)从端原子提交流程

在从端，从端日志与主端日志具有同样的格式。由主端传播而来的批请求在从端首先写入从端日志，批请求中的控制信息写入控制信息区，写请求写入写请求数据区。从端根据每条日志记录的时间戳来判断收到的日志记录分别属于哪一个批请求。对于每一个批请求，从端建立一个变量 k 来记录已经接收到的该批命令中写请求的个数。从端依次接收由主端传播来的写请求记录，属于哪一个批命令，就将与该批请求对应的变量值加 1。若该值等于控制信息区中记录的该批请求中包含的写请求个数，则表明该批命令中的写请求记录已全部收到，则向主端发出批命令收取结束标识，否则继续接收。见图 5(b)所示。

原子提交机制可以保证故障发生时从端数据视图是主端在过去某一时刻所呈现的数据视图，但不能防止主端失效引起的数据丢失。因为主端失效时，正在传播的批请求只有部分传播到了从端。这种情况由于不具有原子性，因此从端不会执行收到的这一部分写请求。这一批请求在主端都已经执行完成，而此时从端的数据视图仍然保持在主端执行这一批请求之前。这意味着该批请求写入主端存储系统的数据丢失，这是异步镜像协议的一个共同问题。

结束语 远程镜像作为一种有效的数据容灾技术得到了广泛的应用。但作为一项技术，它的复杂性并没有得到相应的关注[5]。本文对异步镜像协议做了深入的研究，通过分析和综合现有产品和技术的特点，设计了一种基于日志和批请求传播机制的远程异步镜像协议，同时设计了一种批请求的原子提交机制，从而避免了写顺序不一致引起的主存储系统与从存储系统之间数据视图不一致。研究表明，本协议在保证对应用的写请求具有较好响应速度时，也能够很好地保持镜像系统的数据一致性，在数据丢失量以及通信链路传输效率之间找到一个好的平衡。

参 考 文 献

1 Fujitsu Software Technology Corporation. Data Mirroring Alternatives for Network Storage Pools:[Technical White Paper]

2 EMC Corporation. Symmetrix Remote Data Facility (SRDF) Product Description Guide. 2000

3 Massiglia P. VERITAS Volume Replication and Oracle Databases. VERITAS Corporation, 2000

4 IBM Corporation. Method system and program for maintaining data consistency among updates across groups of storage areas using up-date times. United States Patent, 6463501. 2002. 10

5 Ji M, Veitch V, Wilkes J. Seneca: remote mirroring done write. In: Proceedings of the 2003 USENIX Annual Technical Conference, San Antonio, TX, June 2003. 253～268

6 董欢庆，李站怀，张延园. 异步主从式设备复制协议研究[C]. 第13届全国信息存储技术学术会议论文集，2004. 8

7 Patterson H, Manley S, Federwisch M, et al. SnapMirror: File System Based Asynchronous Mirroring for Disaster Recovery [C]. In: Proceedings of the Fast 2002 Conference on File and Storage Technologies