SQL Server 数据库的应用级持续数据保护系统①

王蒙蒙[1], 邬玉良[2], 张铁峰[1]

[1](华北电力大学 电气与电子工程学院, 保定 071003)
[2](北京中科同向信息技术有限公司, 北京 100086)

摘 要: 针对传统数据容灾系统备份窗口大、容灾等级低等问题, 提出并设计了与 SQL Server 数据库相结合的应用级持续数据保护容灾系统. 系统通过 SQL Server 提供的应用程序接口 API, 读取数据库事务日志变化, 从而实时捕获数据库数据变化, 将捕获的数据变化以日志的形式缓存并传送到异地进行操作重放, 实现异地远程容灾. 当发生灾难时利用心跳检测、IP 浮动等技术进行远程异地接管; 当发生误操作时可根据日志文件恢复到之前的任意时间点, 有效应对人为或意外造成的数据丢失和损坏

关键词: 事务日志; 变动数据捕获; 持续数据保护; 容灾

Application-Level Continuous Data Protection System for SQL Server Database

WANG Meng-Meng[1], WU Yu-Liang[2], ZHANG Tie-Feng[1]

[1](School of Electric and Electronic Engineering, North China Electric Power University, Baoding 071003, China)
[2](Beijing HeartsOne Technology Company, Beijing 100086, China)

Abstract: To solve the problems of traditional data disaster recovery system with large backup window and low disaster immunity, an application-level continuous data protection disaster recovery system combined with SQL Server database is introduced and designed. Application program interface provided by SQL Server is used to read the database transaction log file in the system, then date changes are captured online. The captured data change is recorded in log file and stored in cache-quene. It will be sent to other places to achieve remote offsite disaster recovery. When a disaster occurs, the system uses heartbeat detection, floating IP technologies to take over the broken system. When a malfunction occurs, the system can return to any time point according to the log file. So the system can effectively avoid data loss which caused by man-made or accident.

Key words: transaction log file; change data capture; continuous data protection; disaster recovery

随着计算机技术的发展、信息化程度的不断深入, 数据库作为企事业单位存放数据的主要管理系统逐渐普及, 其对数据安全的保障关系着企业的利益和生命. 目前造成数据丢失的原因[1]主要有以下几种: 硬件故障、软件故障、人为误操作、病毒入侵以及不可抗拒的自然灾害. 针对硬件故障, 目前主要采取 RAID[2]磁盘保护技术避免磁盘故障带来的数据丢失; 针对软件故障、人为误操作以及病毒入侵, 目前主要通过传统的数据保护技术如备份、复制、快照等方式进行; 针对人为不可抗拒的自然灾害, 目前主要利用现有的备份手段结合网络或人工方式建立异地容灾中心, 以防止自然灾害发生时的数据丢失及应用中断.

传统的数据库保护技术, 无论备份方式是热备份, 还是冷备份; 备份策略是全备份、增量备份、异地备份或备份策略的组合; 备份地点无论是本机备份还是异地远程备份; 备份手段是采用镜像或快照等技术, 备份的基本思想都是周期性备份[3]. 当灾难发生时, 备份只能恢复到前一次备份的时间点, 造成数据或多或少的丢失. 定时周期太长灾难发生时, 会丢失大量的数据. 定时周期太小, 又会因为备份动作的频繁操

① 收稿时间:2014-06-09;收到修改稿时间:2014-07-07

作, 造成系统的负担和内存资源的浪费.

针对以上缺点, 本文将 SQL Server 数据库管理系统与持续数据保护系统相结合, 利用数据库事务日志捕获本机数据库变动, 将数据实时传送到备份机, 并实时写入所备份的数据库日志中, 利用数据库管理系统重做日志文件更改机制, 实现数据的实时同步; 当本机宕机或网络中断时, 备份机利用 IP 浮动技术, 接管主机 IP 继续对外提供服务, 有效保证了业务的连续性, 提高了系统容灾的等级.

1　持续数据保护系统概述

1.1 CDP 的概念及发展历程

CDP 是针对传统数据备份容灾系统无法对数据进行持续的保护, 细粒度的备份和恢复而提出的一种可持续的数据备份和恢复技术, 这个概念最初产生于 2005 年左右. 全球网络存储工业协会(SNIA)作为研究 CDP 的重要机构, 从 2005 年开始着手研究 CDP, 历时几年后, 出台了一份研究文档[4], 明确给出了 CDP 的定义. CDP 至少要满足三个条件, 第一、数据有变化就备份. 第二、能够备份到本机以外的其他地方. 第三、能够恢复到任意时间点, 实现数据的零丢失, 零窗口备份. CDP 突破了传统数据保护技术, 为恢复对象提供足够细的备份和恢复粒度, 实现几乎任意时间点的数据恢复, 管理者不必关心备份过程中数据的变化, 从而实现对动态数据的自动保护.

1.2 CDP 的研究现状及目前存在的问题

CDP 从诞生之日迅速成为研究的热点, CDP 系统按实现层次可以分为文件级 CDP、块级 CDP 和应用级 CDP[5]. 文件级 CDP[6], 部署在文件系统上, 利用文件过滤驱动程序捕获被保护文件或目录文件数据的变化; 块级 CDP 部署在系统的块设备驱动层, 利用磁盘过滤驱动程序实时捕获磁盘的每一个写操作; 应用级 CDP 利用所保护应用自身的机制, 如数据库可利用事务日志文件捕获变动数据.

由于文件级和块级 CDP 通用性较好, 目前国内外的研究主要集中在文件级和块级实现 CDP.

文件级 CDP 目前存在的问题在于无法跨越不同的系统平台, 而且对于数据库文件而言, 由于数据库系统自身的特点, 使得其与普通文件在本质上存在着区别, 利用普通文件分析方式分析数据库文件, 难以保证数据库元数据与备份数据的一致性以及备份数据的可用性.

块级 CDP 系统采取的备份策略和机制需要在本机和备份机之间传输大量的数据, 其对存储系统的开销巨大, 恢复时间长. 目前为了减少存储系统开销的问题, 研究集中在优化存储策略, 改进压缩算法(如异或算法或引入重复数据删除技术)[7], 但采用压缩机制后虽然减少了存储开销, 但系统恢复时需要对压缩数据进行反操作, 这在一定程度上增加了数据恢复的时间, 无法达到两全其美的效果, 使得目前块级 CDP 研究趋于瓶颈.

应用级 CDP 将 CDP 系统与具体的应用相结合, 利用应用自身的特点建立相应的变动数据捕获和恢复机制, 有效避免了文件级和块级 CDP 存在的问题, 并保证了备份数据的一致性. 与具体应用紧密结合, 用户管理更加方便灵活同时降低了部署和实施的难度. 由于应用级 CDP 仅对某一具体应用有效且通用性较差, 目前研究较少, 且就数据库系统而言, 有深刻的研究意义.

2　系统架构

本论文提出的 CDP 容灾系统建立在 SQL Server 这一具体应用之上, 利用数据库事务日志, 捕获数据库数据变化; 通过日志记录数据库数据变化, 为数据恢复提供依据; 将数据实时传送到异地重放, 实现应用级容灾; 通过 IP 浮动保证灾难后业务的不中断. 系统结构如图 1 所示.

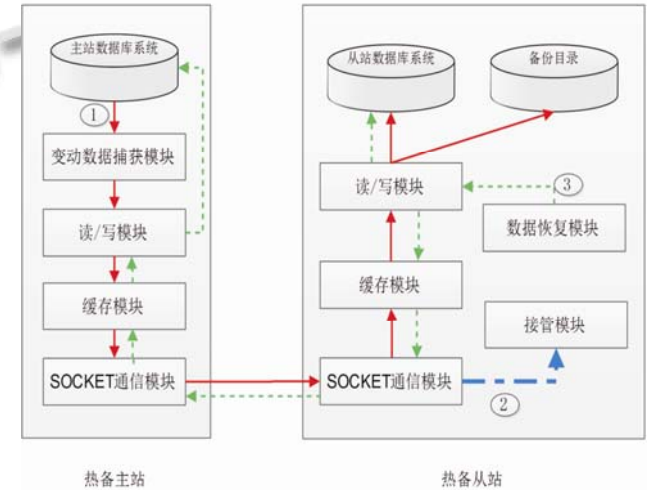

图 1　数据库容灾系统框图

该方案应用场景为双机场景, 一台为主机, 一台

为从机, 两台机子上装有相同的系统和数据库. 主机上主要部署的模块为变动数据捕获模块、读/写模块、缓存模块和 SOCKET 通信模块; 从机上部署的模块有 SOCKET 通信模块、缓存模块、读/写模块、接管模块和数据恢复模块. 正常情况下, 主机通过变动数据捕获模块定位主机上数据库数据的变化, 利用读/写模块读取变化, 并将其存入缓存队列等待传送, 传送模块将缓存模块中的数据打包经 TCP/IP 链路传送到异地的从机上, 从机接收数据后, 一方面将数据实时写入与主站路径一致的数据库管理系统默认数据目录中的数据库文件中, 另一方面将数据库初始完整备份写入设定的备份目录中, 将后续捕获的数据变动写入 cdp_change, 并在 cdp_log 文件中记录相关信息, 供数据恢复时使用. 备份过程数据流向如图 1 中①号线路所示. 当发生人为误操作或软件故障、病毒入侵等造成的错误数据传入时, 可利用从机上的数据恢复模块将数据恢复到错误之前的时间点. 数据恢复和回退过程如图 1 中③号线路所示. 当发生不可抗拒的自然灾害, 如地震、火灾时, 可利用从机上的接管模块, 通过图 1 中的②号线路实时监听主从通信状态, 当主机由于上述等原因宕机或网络中断后, 从机可在设定的时间内接管主机 IP 使原来向主机传送的数据传送到从机, 从而保证应用的连续性, 避免主机故障造成的应用中断.

3 系统的设计和实现

3.1 变动数据捕获模块

数据库变动数据捕获的主要方法有以下几种[8,9]:

基于快照差分的捕获方法. 基于快照差分的变动数据捕获方法是利用快照技术首先生成两个数据源的快照, 通过比较两次快照的差异计算出变动数据. 这种方式捕获效率低, 且无法达到持续保护的设计目标.

基于触发器的捕获方法. 基于触发器的捕获方法是利用数据库提供的触发机制捕获变动数据, 实时性好, 但对主机数据库目前正在运行的事务影响较大.

基于事务日志的捕获方法. 基于事务日志的捕获方法是利用数据库的日志机制, 分析数据库的事务日志, 得出变动数据.

利用数据库自身提供的变动数据捕获方法. 这种方法仅限于部分提供该功能的数据库, 如 SQL Server 2008,但之前的数据库版本并不提供该功能, 因此具有局限性, 且该方式将提取的变动数据通过在对应数据库如 TEST 中新建表, 在表中保存变更信息, 这同样会影响系统目前正在运行的事务.

综上分析, 我们采用基于事务日志的变动数据捕获方法实时捕获数据库变动信息. 由于对于几乎所有的数据库而言, 数据写入到库过程中, 会先将操作记录在日志文件中, 随后记录到数据文件中, 所以针对数据库的应用级 CDP 的研究, 可以从数据库日志着手捕获变动数据. 对于 SQL Server 数据库管理系统, 微软虽然不提供事务日志的内部格式文档, 但是可通过应用程序接口 API 访问数据库事务日志信息, 如可用 fn_dblog 或 DBCC LOG 命令查看数据库的联机事务日志.

基于数据库事务日志捕获数据库数据变化, 有两种方法: 一种可从 DBCC[10]相关命令或函数返回的日志信息中, 得到每条日志记录信息, 分析整个事务过程中的所有的日志记录[11], 转换成对应的操作语句, 利用异地数据库加载该操作语句, 实现数据库数据同步; 另一种可从捕获到的事务记录信息上定位整个事务所在的日志段, 以及该日志段所在的数据块, 将整个事务所涉及到的数据块中的信息读取出来, 在异地将这些信息写入数据库事务日志文件的对应位置, 从而实现数据库数据同步.

第一种方法由于 SQL Server 提供的日志信息有限, 在没有微软支持的情况下很难从捕获的十六进制的日志记录信息上分析出其对应的 SQL 语句, 而且由于 SQL 操作语句种类很多, 需要分析的工作量很大, 因此很难实现. 而第二种方法相对第一种方法可行性要强, 无需分析数据库具体进行了什么操作, 当数据库数据变化时, 只要定位准确, 将变动数据从事务日志文件中提取出来, 实时写入异地数据库日志文件中, 利用数据库管理系统自身的重做和回滚特性, 即可有效保证主从数据库数据同步的高效性和一致性. 本设计中变动数据捕获模块采用的即是第二种方法.

变动数据捕获模块又可细化如图 2.

图 2 变动数据捕获模块结构图

3.1.1 数据库连接

通过 ADO 方式连接数据库. ADO 是 Microsoft 数据库应用程序开发的新接口, 是微软最新的数据访问技术. 其简化了数据访问的过程, 增加了数据访问的灵活性, 是当前数据访问接口的主流技术. ADO 库包含三个基本接口:_ConnectionPtr 接口、_CommandPtr 接口和_RecordsetPtr 接口.

在 VC 中使用 ADO 访问数据库基本过程是:

1) 初始化 COM 库, 引入 ADO 库文件.

2) 创建连接对象.

m_pConn.CreateInstance("ADODB.Connection");

3)连接数据库.

m_pConn->Open((_bstr_t)strConn,"","", adModeUnknown);

4)执行 SQL 语句, 返回查询信息.

_RecordsetPtr& CconsqlDlg::GetRS(CString strSQL) ;

5)释放连接.

if(m_pConnection)m_pConnection->Close();

m_pConnection = NULL;

3.1.2 变动监测

SQL Server 事务日志中的每个记录都由一个日志序列号 (LSN) 唯一标识. 利用 SQL Server 提供的 fn_dblog()函数获得日志记录的 LSN 信息, 返回大于 lastLSN 的记录, 若返回为空则循环比较, 判定为无数据变化, 若不为空, 则判定为有数据变化, 然后根据得到的 LSN 信息, 定位事务日志文件的变化, 得到此事务包含的日志记录数目, 重新设置 lastLSN. 变动监测流程如图 3 所示.

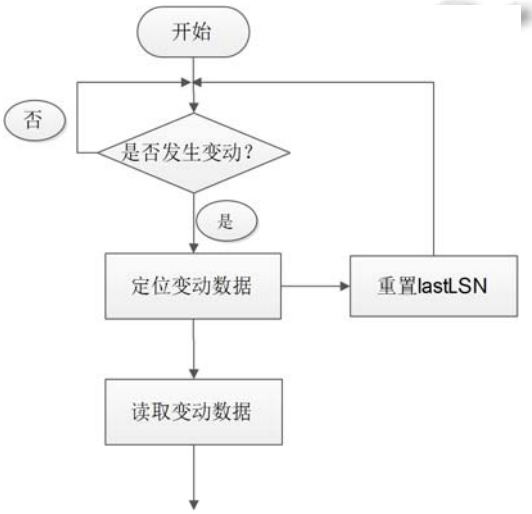

图 3　变动监测流程图

3.1.3 定位数据变动

利用::fn_dblog(@StartingLSN, @EndingLSN), 以及 dbcc loginfo 命令提取需要的信息, 定位变动数据在事务日志中的具体位置.

SELECT [Current LSN],Operation, Context, [Transaction ID] from::fn_dblog (null, null) where [Current LSN] > '00000011:00000134:0009';

dbcc loginfo;

捕获信息如图 4 所示.

	Current LSN	Operation	Context	Transaction ID
152	00000011:00000135:0001	LOP_BEGIN_XACT	LCX_NULL	0000:00000213
153	00000011:00000135:0002	LOP_INSERT_ROWS	LCX_HEAP	0000:00000213
154	00000011:00000135:0003	LOP_COMMIT_XACT	LCX_NULL	0000:00000213

	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	2	253952	8192	17	2	64	0
2	2	253952	262144	0	0	0	0
3	2	253952	516096	0	0	0	0
4	2	278528	770048	0	0	0	0

图 4　变动捕获实例

由捕获的上述信息可定位事务日志变化部分的具体位置, 具体定位过程如下:

1)首先由 LSN 的 3 部分信息, 提取其所在的 VLF 编号、起始数据块号、日志段中的 SLOT 序号.

2)由 dbcc loginfo 命令得到正在使用的 VLF 的 StartOffset, 定位到变化事务所在的 VLF 的起始位置, 可记为 vlf_begin.

3)由 1)、2)步骤得到的信息, 将得到的数据块号记为 bk_number, 则日志记录所在日志段的起始位置可用以下公式计算得到.

$$vlf\_begin + bk\_number * 512$$

4)在日志段的起始处记录了此日志段的大小记为 $block\_size$, 由公式 $block\_size \backslash 512 + 1$ 得到该日志段占居的数据块, 从而成功定位变动信息, 之后利用读/写模块读取变动信息.

3.2 读/写模块

读/写模块主要分为两部分, 一部分为读, 一部分为写. 对于热备主机主要用于初始时刻的完整备份, 以及在变动捕获模块定位到事务日志文件中数据变化的日志段后, 利用得到的 vlf_begin, 以及 block_size 的两个参数, 将变化事务所在的日志段读取出来, 并写入缓存队列, 传送到异地.

对于热备从机, 对于初始的完整备份, 一方面写

入到设定的备份目录中, 另一方面写入从机数据库管理系统的默认数据目录中; 对于后续的变动数据一方面实时写入数据管理系统默认数据目录中相应数据库事务日志文件的对应位置, 一方面写入备份目录的 cdp_change 文件中, 并将写入的对应位置、时间等相关信息保存在 cdp_log 文件中. 备份过程的数据流向如图 5 所示.

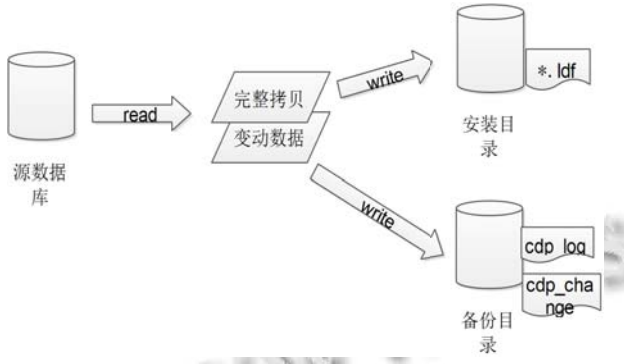

图 5 备份数据流向图

其中 cdp_change 文件为只追加文件, 将变动数据依次写入文件中. cdp_log 目录记录变动数据的对应关系的相应信息, 具体包含内容如表 1 所示.

表 1 cdp_log 记录项

cdp_log 项	各项属性
dbname	备份数据库名称
Ch_time	变动数据捕获时间点
step	步数, 累加
Cl_offset	捕获的变动数据在 cdp_log 中的偏移量
ldf_offset	捕获的变动数据在源数据库事务日志中的偏移量
block_size	变动数据所占数据块大小

3.3 缓存模块

在主机和从机两端都设有缓存模块. 主从数据同步有两种方式, 一种采用同步方式传输, 一种采用异步传输机制. 由于在捕获日志信息后, 需要先对日志信息进行分析后, 再进行传输, 故分析后选用异步传输机制. 异步传输过程中涉及到数据不一致和断点续传问题, 为了保证数据一致性, 设置数据缓存模块. 在缓存模块中采用堆栈结构, 遵循先进先出的原则.

3.4 数据传输模块

数据传输模块可具体细化为传送模块和接受模块两部分. 该设计适用于局域网和广域网环境, 利用 TCP/IP 协议, 设置软件监听端口, 利用 SOCKET 编程技术, 实现数据的主从同步.

3.5 恢复模块

CDP 系统恢复有三种恢复策略, 对应于三种参考模型, 分别为基准参考数据模型、复制参考数据模型和合成参考数据模型. 这三种方式分别通过正向、逆向、正逆向相结合的方式记录和保存变化的数据来实现, 各自的优缺点如下:

基准参考数据模型原理简单, 实现容易, 但越是靠近当前的时间点, 恢复时间越长.

复制参考模型恢复的时间点越靠近当前恢复时间越短, 但是由于数据存储过程中需要同时进行数据和日志记录的同步, 需要较多的系统资源.

合成参考数据模式, 将前两者存在的问题折衷, 在资源占用和恢复效果上都不错, 但是实现起来复杂, 需要复杂的软件管理和数据处理功能.

对于数据库管理系统, 由于其自身的日志重做机制, 采用的恢复策略类似于基准参考数据模型.

本设计的恢复原理: 在备份目录存放有备份数据库初始的完整备份, 恢复过程中首先比对从站数据库目录中的数据库数据文件是否为初始的完整备份, 如果不是则将备份目录的初始完整备份拷贝到数据库指定的数据目录中, 直至一致. 之后按照要恢复到的时间点和具体的回退步数, 依照 CDP_Change 以及 CDP_Log 中记录的信息将初始时刻到该时刻对应步数的变动写入到数据库事务日志文件的相应位置. 数据库管理系统重新启动后会重做事务日志文件中记录的所有变动, 从而实现对任意时间点的回退, 有效避免了由于人为误操作或病毒入侵等造成的数据损坏或丢失. 恢复流程如图 6 所示.

3.6 接管模块

接管模块主要利用心跳检测技术和 IP 浮动技术, 实现对主机数据库的接管, 避免数据库支持的业务中断以及数据丢失. 通过在主从之间发送心跳检测包, 探测主从的连接情况, 判断主机状态. 本设计中从机周期性的发送心跳检测包, 根据主机反馈回来的信息, 判定主机状态, 可根据情况人为调节发送周期. 如果主机网络中断或宕机, 则从站利用 IP 浮动技术, 接管主机 IP 继续对外提供服务,并启动从机数据库相应服务, 具体流程如下图 7.

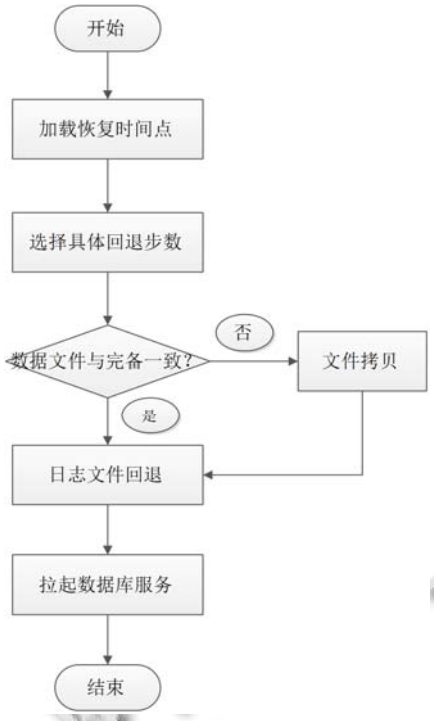

图 6　恢复流程图

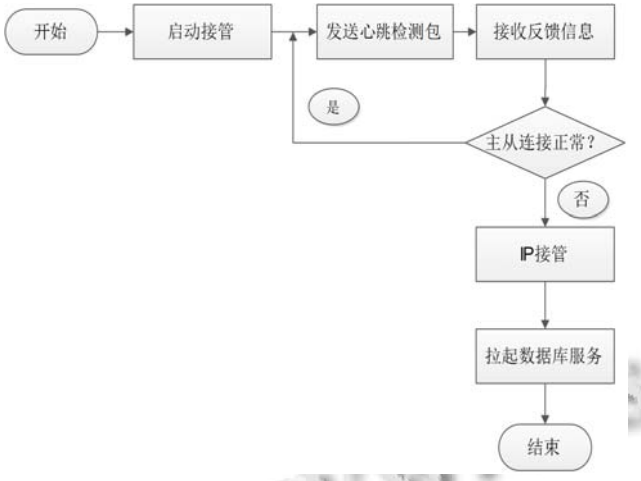

图 7　接管模块流程

4　结语

CDP 系统相对于传统数据保护策略存在很多优势, 但实现过程也比较复杂, 目前针对 CDP 的研究主要集中在基于文件级以及块级实现, 而针对具体应用的研究寥寥无几. 对于目前企事业单位存储关键数据常用的数据库管理系统而言, 虽然文件级 CDP 和块级 CDP 系统也能够实现对其数据的持续保护, 但都存在着一定的问题, 本文从数据库事务日志入手, 针对具体数据库建立一套应用级的持续数据保护系统, 分析了如何捕获数据库任意时刻变化及恢复到任意时间点, 建立瞬时接管机制, 提高了系统的容灾等级. 但对于 CDP 系统而言, 基于应用的研究还有很长的路要走, 本文仅对人们常用 SQL Server 数据库管理系统提出了详细的设计方案和实现方法, 还有待更为广泛的扩展, 以及更为深入的研究.

参考文献

1 Wallis J. Common causes for data loss. http://www.isnare.com/?aid=222505&ca=Computers+and+Technology. [2008-10-08].

2 Patterson DA, Gibson G, Katz RH. A case for redundant arrays of inexpensive disks (RAID). ACM, 1988: 109–116.

3 Rock M, Poresky P. Shorten your backup window. Special Issue on Managing the Information that Drives the Enterprise, 2005: 28-34.

4 Continuous data protection. https://snia.org/sites/default/files/SNIA_DMF_V6_DPI_Guide_4WEB.pdf.

5 Flouris M, Bilas A. Clotho: transparent data versioning at the block I/O level. Proc. of the 21st IEEE/12th NASA Goddard Conference on Mass Storage Systems and Technologies. Washington, DC, USA. IEEE Computer Society. 2004: 315–328.

6 黄志刚,郭玉东,杨宗博.文件级 CDP 的设计和实现.计算机应用,2008,S2:277–279.

7 Wang C, Li ZH, Hu N, Nie YM. S-TRAP: Optimization and evaluation of timely recovery to any point-in-time (TRAP). Computer Science and Information Systems/ComSIS, 2012, 9(1): 431–454.

8 黄建平.基于 SQL Server 数据库日志的信息源监测方法的研究与实现[学位论文].广州:暨南大学,2007.

9 邹先霞,贾维嘉,潘久辉.基于数据库日志的变化数据捕获研究.小型微型计算机系统,2012,3:531–536.

10 向猛,谢立靖.SQL SERVER 2005 基于事务日志的备份与恢复深入研究.计算机系统应用,2013,22(6):18–23,74.

11 李爱武.SQL Server 实例恢复中重做日志记录定位机制研究.现代计算机(专业版),2009,9:107–109,138.