商业银行容灾系统建设方案①

王 刚

(陕西省汉中市邮政局 信息技术局, 汉中 723000)

摘 要: 根据商业银行对容灾中心的定位描述, 容灾系统不仅具有基本的容灾功能, 而且是连接生产核心系统与辅助系统的桥梁. 因此, 建设容灾系统具有十分重要的现实意义. 鉴于容灾系统建设是一个涉及面广、专业性强的系统工程, 本文对容灾系统建设的总体原则和思路进行了研究, 探讨各种容灾可用技术, 并进行分析对比, 提出了系统构架, 对商业银行容灾系统的建设具有参考价值.

关键词: 商业银行; 容灾系统; 总体原则; 系统构架; 建设方案

Commercial Bank Disaster Recover System in the Construction Plan

WANG Gang

(The postal bureau of hanzhong, shannxi the bureau of Information technology, Hanzhong 723000, China)

Abstract: According to the commercial Banks located in the disaster center of the description, the disaster recover system disaster not only has basic functionality, but also is a bridge connecting production core system and auxiliary system. Therefore, construction of the disaster recover system has very important practical significance. Given disaster recover system construction is a wide range and specialized system engineering, this paper general principles and ideas for the construction of the disaster recover system are studied, discussed all kinds of disaster available technology, and carries on the analysis comparison, proposed the system architecture,it has the reference value the commercial bank disaster recover system construction.

Key words: commercial bank; disaster recover system; general principle; system architecture; construction plan

由于银行业务对计算机系统的依赖性越来越大, 金融数据处理的高可靠性显得非常关键, 一旦数据由于某种灾难发生永久性丢失, 其社会影响和经济损失不可估量, 后果不堪设想. 银行系统的灾难主要有自然灾难和人为灾难, 为应对灾难的发生, 商业银行必须制定和建立完备的灾难恢复系统. 容灾恢复系统建设的核心目标是在尽最大可能地保证容灾系统中的数据与生产系统中的数据一致性的基础上, 确保备份数据安全、可用. 该系统必须建立在成熟、稳定的软硬件平台基础上, 保证数据备份的完整和灾难发生时应用接管的可靠[1]. 通过建立完善的容灾安全控制体系和可靠的运行管理机制, 保障容灾系统稳定、安全运行. 按照各业务系统及子系统的重要性和各业务系统的恢复等级, 制定合理的数据备份与容灾策略, 优先保障关键业务数据的完整和关键业务的快速恢复. 目前容灾中心所采用的技术方案主要有两类, 即基于存储系统的同步数据容灾方案和基于数据库的归档日志追加数据级容灾方案.

1 容灾建设可用技术分析

在应用系统层、数据库管理层、操作系统卷管理层、存储网络(SAN)层和存储设备层, 均有相关的数据复制技术方案, 可用于容灾系统建设中[2].

1.1 基于应用层的容灾技术

1.1.1 实现原理

生产系统中的应用程序通过数据复制转发的模式, 将交易数据传送到部署于容灾中心的灾备系统. 由生产系统和容灾系统处理相同的交易数据, 以确保两边

① 收稿时间:2013-04-25;收到修改稿时间:2013-06-13

数据的一致性. 其原理如图 1.

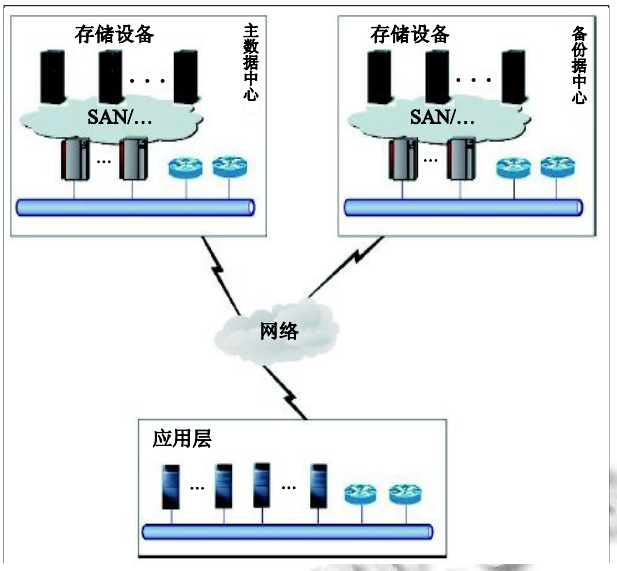

图 1 基于应用层的容灾原理

中心为主数据库, 容灾中心为备用数据库. 当修改主数据库时, 生成的更新数据发送到备用数据库, 其原理如图 2. 如果主数据库出现了故障, 备用数据库即可以被激活并接管生产数据库的工作.

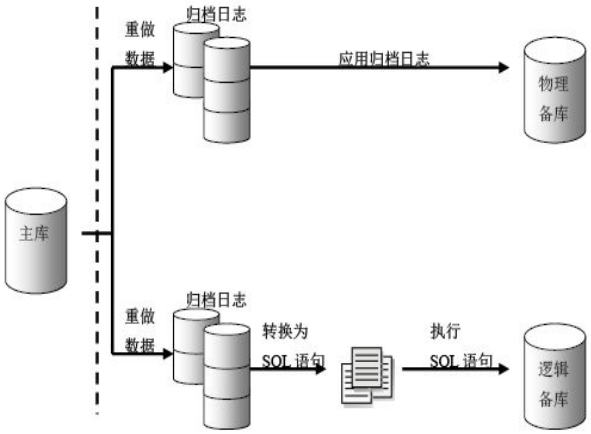

图 2 基于数据库的备份原理

1.1.2 实现方式

应用层容灾在异地容灾中心建立一套完整的应用系统, 当灾难发生时, 异地容灾系统会检测到灾难的发生, 在保证用户数据的完整性、可靠性、一致性的前提下, 实时进行切换, 由异地应用层容灾系统向用户提供不间断的业务服务. 基于应用层的容灾方案有同步和异步两种实现方式. 同步方式是指生产中心应用系统交易处理完成后, 同时将交易请求转发给容灾中心应用系统[3]. 只有当两个系统都处理完成后, 才向客户端返回交易处理结果. 异步方式是指生产中心应用系统交易处理完成后, 立刻向客户端返回交易处理结果. 生产中心应用系统可以在一定时间之后, 以报文或者批量文件的方式, 将交易转发到容灾中心应用系统重新执行. 异步方式对应用程序性能的影响小, 但容灾中心数据更新方面与生产中心相比会有延迟.

1.1.3 优缺点分析

(1) 优点: 容灾系统处于热备用状态, 随时可以提供服务; 传输交易发起数据, 对网络带宽要求较低.

(2) 缺点: 如果采用同步方式, 数据处理速度较慢; 数据一致性完全由应用软件控制, 软件开发面临新挑战, 技术实现难度大.

1.2 基于数据库的容灾技术

1.2.1 实现原理

在生产中心和容灾中心采用相同的数据库, 生产

1.2.2 实现方式

备用数据库可以是物理备用数据库, 也可以是逻辑备用数据库.

(1) 物理备用数据库: 物理备用数据库通过接收主数据库日志并应用日志的方式与主数据库保持同步. 物理备用数据库在应用日志时, 是基于数据块级别来操作. 因此, 要求备用数据库和主数据库具有相同的物理结构, 而且备用数据库只能处在恢复状态和只读打开两种状态中的一种[4].

(2) 逻辑备用数据库: 逻辑备用数据库与主数据库只要求逻辑结构相同, 物理结构可以不同. 它通过接收主数据库的日志, 并转化为 SQL 语句, 在备用数据库中运行的方式, 与主数据库保持同步. 逻辑数据库除了用于灾难恢复之外, 也可以用于其他的用途, 它允许用户根据需要随时进行查询以及随时生成报表, 还可以建立自己的数据库对象, 进行读写操作. 逻辑备用数据库与物理备用数据库相比, 其优点在于数据库可以一直处于打开状态, 以提供查询、统计等功能, 但是, 逻辑备用数据库目前具有一些无法克服的缺点: 由于需要将归档日志转换为 SQL 语句再重新执行, 对系统的性能要求很高; 逻辑备用数据库不支持主数据库表中使用一些特殊的数据类型; 逻辑备用数据库要求主数据库表中的数据必须唯一标识, 也就是说必须具有主键或者唯一索引.

1.2.3 优缺点分析

(1) 优点: 对存储设备透明; 灾难发生时, 备用数据库系统可快速就绪.

(2) 缺点: 最大保护和最高可用模式, 对于主数据库系统资源占用很高; 最大性能模式在灾难发生时, 有数据丢失; 要求主数据库和备用数据库的操作系统和数据库版本一致.

1.3 基于操作系统卷管理层的容灾技术

1.3.1 实现原理

生产中心的主机同时识别来自于生产中心和容灾中心的存储设备上的磁盘, 利用操作系统所配置的卷管理器, 实现生产中心的逻辑卷和容灾中心的逻辑卷之间的实时同步数据复制. 其原理如图 3. 当生产中心发生突发性灾难时, 可以在容灾中心服务器上激活相应的卷组和逻辑卷, 进而启动数据库和应用系统, 实现业务系统快速恢复.

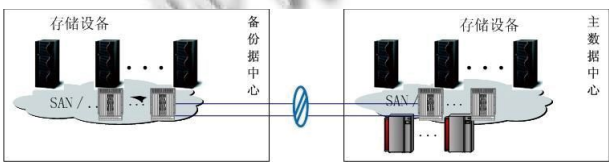

图 3　基于操作系统卷管理层的容灾原理

1.3.2 实现方式

基于操作系统卷管理器的数据复制方式分为同步复制和异步复制. 同步复制是指操作系统将更新数据写往本地连接的磁盘系统, 同时将数据转发给远端连接的磁盘系统. 这两个写操作是同时进行, 只有当两个系统都拥有数据的拷贝以后, 本地系统才会向处理器返回一个 I/O 完成指示[5]. 应用进程会等待写 I/O 操作完成, 才能够进行下一个数据写操作. 同步复制方式工作原理见图 4.

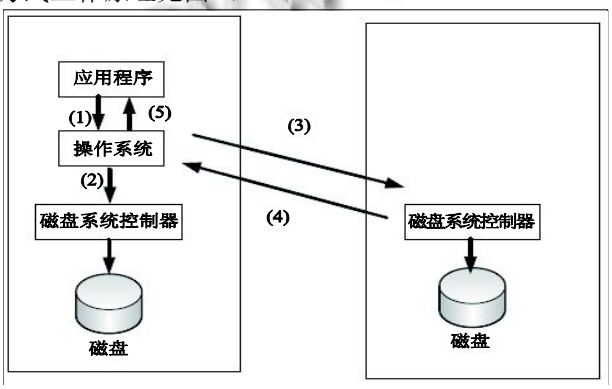

图 4　基于操作系统的同步复制方式

远程容灾异步方式下, 通过时间戳、分组号可以保证数据的一致性和完整性, 并在灾难发生时的数据丢失最少, 恢复时间短. 生产系统所发出的 I/O 操作至本地存储系统, 本地存储系统处理结束后即通知主机本次 I/O 结束. 然后, 本地生产存储系统将多个累计的写 I/O 异步的, 不一定按顺序的传送到备份中心的存储系统中, 由于 I/O 操作不是同步的传送到备份中心, 在异步方式下, 就存在数据的传送顺序与实际的数据的操作顺序不一致问题. 为了解决这一问题, 容灾系统对每个写入生产中心存储系统的 I/O 都打上一个时间戳(Time Stamp)并进行一致性分组, 在数据传输至备份中心时, 备份中心存储系统严格按照此时间戳的时间顺序重新排列并写入相应的逻辑卷中, 从而保证了备份数据的逻辑一致性与完整性. 异步复制方式工作原理见图 5.

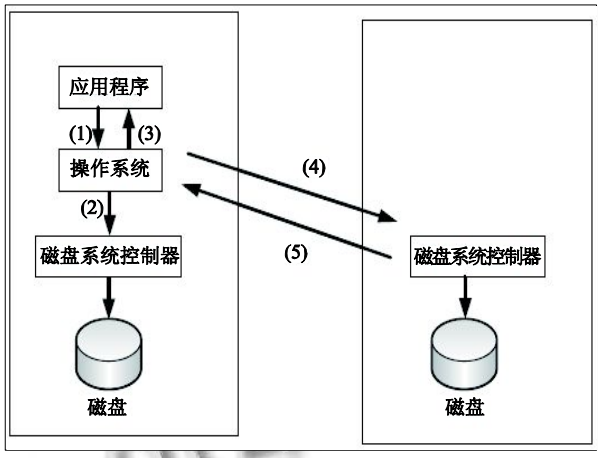

图 5　基于操作系统的异步复制方式

1.3.3 优缺点分析

(1) 优点: 对存储子系统透明, 生产和容灾中心可部署不同的磁盘阵列, 容灾结构相对简单.

(2) 缺点: 远程数据复制操作占用较多主机的资源; 生产中心应用系统的写操作性能受生产和容灾中心传输距离影响较大.

1.4 基于存储系统的容灾技术

1.4.1 实现原理

基于存储系统的容灾技术, 是利用存储设备控制器中嵌入的远程复制功能, 配合数据复制软件、卷管理软件, 在相同类型的存储子系统之间进行同步复制和异步数据复制, 可以实现生产中心和容灾中心数据备份, 尽量保持生产系统的存储数据逻辑卷与备份系

统存储数据逻辑卷的一致性[6]. 这种技术可以不依赖于主机和应用软件, 而靠存储系统的大量硬件技术来实现.

1.4.2 实现方式

基于存储系统的远程拷贝形式分为同步复制和异步复制两种[7].

(1) 同步复制(Synchronous Writes): 来自主机的数据被写往本地连接的磁盘系统, 该系统将数据转发给远端连接的磁盘系统. 只有当两个系统都拥有数据的拷贝以后, 本地系统才会向主机返回一个 I/O 完成指示. 同步远程拷贝能够在远端提供最新的数据, 但应用程序会因等待所有 I/O 操作的完成而被延迟. 工作原理如图 6:

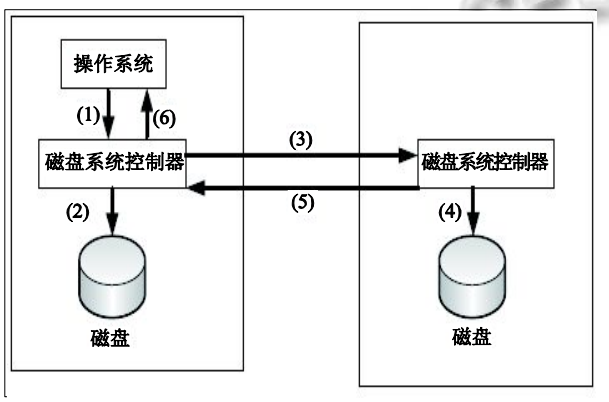

图 6 基于存储系统的同步复制原理

(2) 异步复制(Asynchronous Write): 来自主机的数据被写往本地连接的磁盘系统后, 该系统立即向主机返回一个 I/O 完成指示. 数据在很短的一段时间以后被送往一个远端磁盘系统. 异步远程拷贝对应用系统的性能影响最小, 但远端磁盘系统在数据的更新程度上与本地系统相比会有一个延迟. 工作原理如图 7.

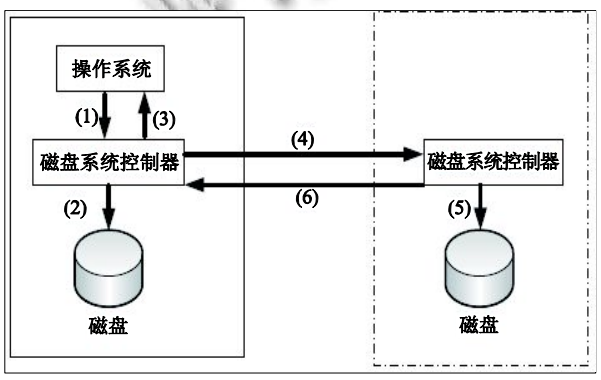

图 7 基于存储系统的异步复制原理

如果在异步复制方式中, 远端磁盘在写入数据块时不考虑发生的先后顺序, 这种方式也称全局拷贝(Global Copy), 在全局拷贝的情况下, 比如某一时刻主机对本地磁盘发起了 A、B、C、D、E 五个块的写盘请求, 本地的磁盘系统的写顺序是 A, B, C, D, E. 但是由于线路等原, 远端磁盘系统的接收顺序可能是 A, C, B, D, E, 并按此循序写盘. 为了解决本地磁盘和远端磁盘可能存在的数据块在读写顺序的差异, 磁盘系统提供有一致性组的异步远程数据拷贝.

1.4.3 优缺点分析

(1) 优点: 对系统性能影响很小, 数据一致性较好.

(2) 缺点: 两边的存储设备必须是同构的, 对线路带宽的要求通常也较高; 灾难发生时, 可能导致灾备中心数据不一致, 严重时有可能造成数据库无法启动.

1.5 基于 SAN 的容灾技术

1.5.1 实现原理

在存储交换机与存储磁盘阵列之间增加一个存储数据逻辑管理层, 用以管理其后面的所有磁盘阵列并可形成存储池, 同时, 通过这一层设置数据级容灾保护机制和数据备份策略, 以增量的方式或持续复制的方式将生产中数据通过高速网络复制到远端的容灾中心的存储设备中. 具有存储数据逻辑管理层功能的设备即可以是 SAN 交换机和高级存储设备磁盘控制器, 也可以是专用于 SAN 卷管理的设备.

1.5.2 实现方式

基于 SAN 的容灾技术方案实现方式也有同步复制和异步复制两种, 其实现原理基本与基于存储系统的远程拷贝方式基本相同, 如图 8.

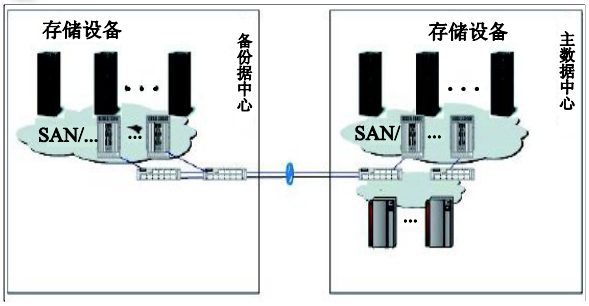

图 8 基于 SAN 的容灾原理

1.5.3 优缺点分析

(1) 优点: 数据复制过程对于应用系统透明, 对主机性能无影响, 生产中心和容灾中心可部署异构磁盘阵列; 采用专用的 SAN 卷管理设备, 可减小磁盘

阵列控制器处理压力.

(2) 缺点: 采用专用的 SAN 卷管理设备, 在数据存储路径上增加一层, 对线路带宽的要求也较高.

1.6 容灾技术对比分析

综上所述, 每种容灾技术都有其各自的特点[8-11], 具体情况如表 1:

表 1　容灾系统对比分析

项目	基于应用系统	基于数据库系统	基于操作系统	基于存储系统	基于SAN
备份内容	数据库数据	数据库数据	整个磁盘数据	整个磁盘数据	整个磁盘数据
传输的内容	交易数据	数据库归档日志	整个磁盘的增量数据块	整个磁盘增量数据	整个磁盘增量数据
传输的安全性	中	高	高	高	高
异步方式对数据丢失影响	较少	较多	由数据复制缓冲区及数据块大小决定	少	少
对生产主机性能影响	有影响	影响较大	影响较大	影响较小	影响较小
对数据库的要求	无	版本一致	版本一致	版本一致	版本一致
网络带宽的要求	较高	高	很高	很高	很高
实现的难易程度	很复杂	复杂	较容易	容易	容易
维护工作的难易	很复杂	复杂	较容易	容易	容易
恢复时间	短	长	较短	较短	较短
对存储设备要求	无	无	无	无	无

2　容灾系统架构

为了保证业务系统关键功能的不间断运行, 以及所有生产数据的完整和安全, 容灾中心在商业银行 IT 总体规划中具有重大意义. 依据 IT 总体规划中“两地三中心”的建设模式, 容灾系统建设包括同城灾备中心建设和异地灾备中心建设两部分内容, 见图 9. 其中, 同城灾备中心与生产全国中心的所有生产数据完全一致. 在生产全国中心发生灾难时, 金融关键业务的关键功能应能实时切换到同城灾备中心, 实现关键业务功能的不间断运行, 其它相关业务系统可在短时间内恢复运行[12]. 异地灾备中心与生产全国中心的所有生产数据存在一定差异. 在生产全国中心与同城灾备中心同时发生灾难时, 生产数据应该能够恢复到要求的时间点, 关键业务系统的关键功能可在有限时间内恢复运行.

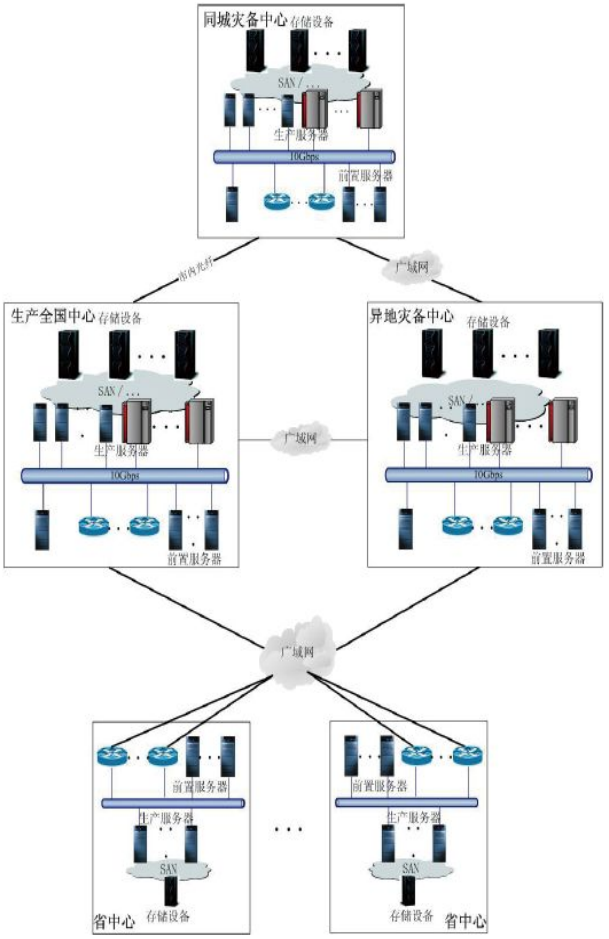

图 9　容灾系统目标架构

2.1 容灾系统方案的研究

容灾不是简单的备份, 而是当灾难发生时, 能全面及时地恢复整个系统. 容灾系统可以分为三个级别: 数据级容灾、系统级容灾、应用级容灾. 针对不同信息系统的应用、特点和需求, 各种容灾技术既可以单独使用, 也可以组合使用[13]. 根据银行业务系统的具体情况, 选择合适的技术或技术组合, 是容灾系统建设方案的基础.

2.1.1 基本概念

衡量容灾系统性能的有两个指标: (1)恢复时间目

标(RTO), 是系统所能容忍的应用停止服务的最长时间, RTO 值越小就表示容灾系统恢复能力越强; (2)恢复时间点目标(RPO), 是反映恢复数据完整性的指标, 表示业务系统所能容忍的数据丢失量.

建设容灾系统的最基本目标是保证业务的连续性, 业务的连续性研究包括两个重要方面: (1)业务连续性管理(BCM), BCM 是一项综合管理流程, 包括基础数据、应用系统、业务的灾难备份与恢复计划; (2)业务连续性计划(BCP), BCP 是为避免关键业务功能中断, 减少业务风险而建立的控制过程, 包括支持关键业务功能的人力、物力、和所需要的基本服务的连续性保证[14].

2.1.2 同城容灾

综合考虑各种技术实现方案对生产系统性能影响程度及实施难度、容灾系统同步要求、各专业银行同城容灾技术方案及同城容灾业务指标的要求等因素, 在同城容灾方案中选用基于存储系统的同步技术方案是比较合适的. 如果同城容灾中心与生产中心距离超过 50km, 将使用基于存储系统的异步技术方案. 基于存储系统的方案可实现:

(1) 恢复点目标很小, 业务系统基本没有数据丢失;

(2) 数据同步工作由存储设备完成, 对应用系统主机透明;

(3) 商业银行的实际使用表明, 基于存储系统的数据同步方案实施简单.

2.1.3 异地容灾

在生产全国中心与同城灾备中心同时发生灾难时, 生产数据应该能够恢复到要求的时间点, 关键业务系统的关键功能可在有限时间内恢复运行. 根据技术成熟度、对生产应用系统性能影响度, 基于存储系统的异步容灾技术和基于数据库的最大性能容灾方案均能满足要求.

2.1.4 多中心连接模式的选择

在“两地三中心”间实现基于存储系统的异步数据传输模式, 有多跳、分发两种基本模式及多跳与分发相结合的模式, 如图 10、图 11.

2.1.5 两种模式的区别

多跳模式中, 生产中心存储设备上的变化数据同步更新到同城容灾中心相应的存储设备上, 而异地容灾中心的数据则来自于同城容灾中心, 通过异步模式以减小对同城容灾设备的影响. 同城容灾中心和异地

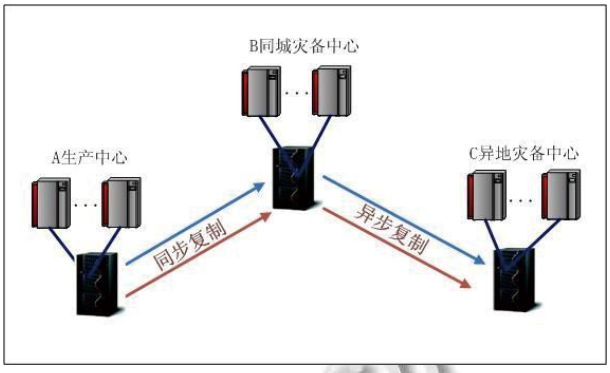

图 10 多跳模式

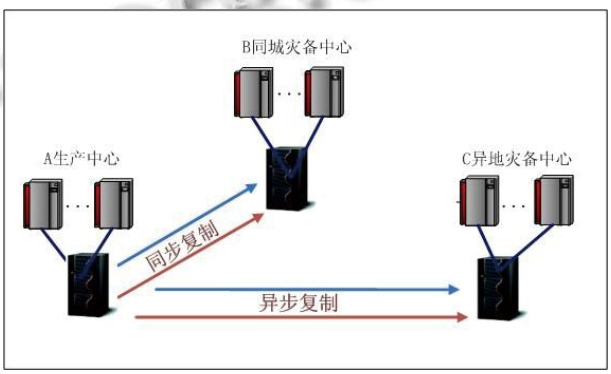

图 11 分发模式

容灾中心的数据均来自于生产中心磁盘, 同城容灾中心与生产中心保持存储设备同步连接, 而异地容灾中心与生产中心设备保持异步连接模式. 通过原理分析可发现两种模式存在如下区别:

(1) 异地容灾同步数据时间不同

多跳模式中, 异地容灾数据需要经同城中心进行中转而得到, 而分发模式中异地容灾中心数据直接来自于生产中心.

(2) 压力点不同

多跳模式的压力主要存在于同城容灾中心存储设备上, 而分发模式的压力则在生产中心的存储设备上.

(3) 抗灾难性能力

对于多跳模式, 如果同城容灾中心存储设备出现故障, 则异地容灾中心与生产中心同步断开; 对于分发模式, 如果生产中心存储设备出现问题, 同样存在异地生产中心与同城容灾中心无法同步的问题.

2.1.6 两种模式的融合

多跳与分发相融合的模式, 结合了多跳与分发的技术优势, 解决了数据中心间潜在的无法数据同步的风险, 其原理如图 12.

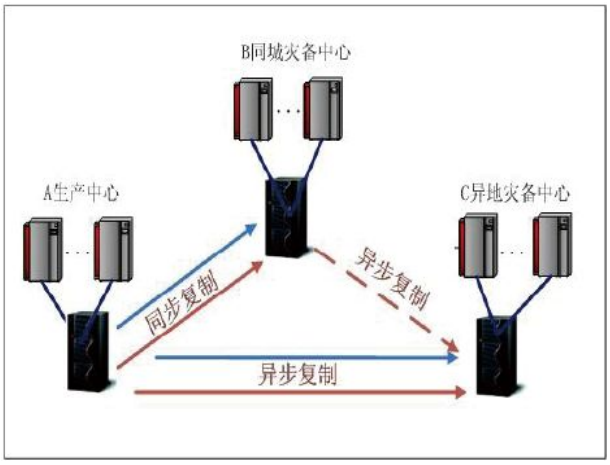

图 12 多跳与分发相结合的模式

3 结束语

基于存储系统的容灾方案, 无论采用同步模式还是异步模式, 均需要一个前提条件: 两边必须部署同品牌、同级别的存储设备. 这一特点与成本预算有着直接的关联关系. 在现有可行方案中, 基于 SAN 的容灾技术可以从底层化解同构问题. 为了将存储设备从数据同步的压力中解放出来, 可在 SAN 交换机上或 SAN 交换机这一层采用专用的硬件设备, 完成数据同步策略管理及异构设备的支持. 其不足之处是增加了底层数据传输层次和维护难度.

参考文献

1 郭继君.大集中模式下存储容灾系统设计方案.华南金融电脑,2002,(9):42–46.
2 施蓓莉.基于 HP 主机平台加 EMC/EVA 存储的银行业务系统灾备实现.上海交通大学,2008:4–16.
3 杨柳明.论商业银行容灾系统建设.广西金融电脑,2008,(2):56–59.
4 王德军,王丽娜.容灾系统研究.计算机工程,2005,31(6):43–46.
5 刘迎风.容灾技术及其应用.计算机应用研究,2002,19(6):11–15.
6 徐鹏,薛建新.数据中心容灾系统研究.计算机工程与设计,2007,28(33):22–27.
7 许福忠.银行交易系统灾备技术研究与应用.国防科技大学,2006:7–17.
8 叶嘉铭,胡晓勤.多数据库容灾系统的设计与实现.计算机工程与设计,2012,33(12):241–245.
9 蔡皖东,何得勇.一种网络容灾系统的设计与实现.计算机工程,2004,30(7):116–118.
10 于新华,刘川意.远程容灾系统设计框架.计算机工程与应用,2006,42(24):31–34.
11 武鲁,李钟华.基于集群服务器的容灾系统的副本管理研究.计算机应用研究,2006,23(6):76–78.
12 陈汶滨,吕曼曼.容灾备份系统研究.计算机安全,2009,(7):70–71.
13 施云龙,韩广琳.浅谈银行系统柜面服务器集群的容灾备份方案.福建电脑,2006,(5):69–71.
14 杨景发.商业银行四地容灾系统的设计与实现.上海交通大学,2011:6–43.

(上接第 6 页)

actions on Software Engineering, SE-12(7): 744–751.
14 Maranzano JF, Rozsypal SA, Zimmerman GH. Architecture reviews: practice and experience. IEEE Software, 22(2): 34–43.
15 Arkin B, Stender S, McGraw G. Software penetration testing. IEEE Security & Privacy, 2005, 3: 84–87.
16 Madan BB, Gogeva-Popstojanova K, Vaidyanathan K. Modeling and quantification of security attributes of software systems. Proc. Int'l Conf. Dependable Systems and Networks. 2002.
17 Ouchani S, Jarraya Y, Ait Mohamed O. Model-based systems security quantification. Privacy, Security and Trust(PST), 2011 Ninth Annual International Conference on. IEEE. 2011. 142–149.
18 Barnum S, McGraw G. Knowledge for software security. Security and Privacy, IEEE, 2005, 3(2): 74–78.
19 MITRE.Common Attack Pattern Enumeration and Classification,Version 1.7.1.http://capec.mitre.org/.2012.
20 MITRE.Common Weakness Enumeration.Version2.4.http://cwe. mitre.org/.2013.
21 杜晶,杨叶,王青.基于证据的可信软件过程评估方法.计算机科学与探索,2011,5(6):501–512.
22 Yang Y, Wang Q, Li M. Process trustworthiness as a capability indicator for measuring and improving software trustworthiness. Trustworthy Software Development Processes. Springer. Berlin, Heidelberg. 2009. 389–401.