面向全系统毁坏后的服务即时恢复技术

郑纬民
（清华大学计算机科学与技术系，北京 100084）

［摘要］ 论述了一种全新的系统容灾保护方法，它脱离了传统基于数据保护的容灾保护思路，通过复制包括数据状态以及服务运行状态在内的全系统状态，并引入并行恢复的思想，最终达到了无论系统毁坏程度如何，都能保证服务即时恢复的目的。同时，与传统技术相比，该技术还可独立于具体设备和应用，容易做到容灾资源的共享从而节省了容灾系统的建设成本。

［关键词］ 容灾；虚拟化；并行
［中图分类号］ TN914 ［文献标识码］ A ［文章编号］ 1009－1742(2009)10－0032－07

1 前言

随着信息技术的快速发展，计算机在人类生活中扮演着越来越重要的角色。计算机系统的停机也越来越变得无法容忍，因为这将意味着无数人的生活将受到影响，同时也将导致巨大的经济损失。然而不幸的是，诸如飓风和地震这样的自然灾害，以及像恐怖袭击这样的人为灾难，都将对全世界的各种计算机系统的正常运行带来严重威胁。2001 年的“9· 11”恐怖袭击，2004 年的印度洋海啸，2005 年的卡特里娜飓风，2008 年美国艾奥瓦州的洪水泛滥，以及 2008 年中国汶川地震等等[1,2]。这些灾难不但对人类生活造成了困扰，更进一步影响到了商业活动的连续性，从而导致了巨大的经济损失。为了保护信息系统在灾难面前尽可能小的受到影响，特别是保证信息系统中的数据不丢失，容灾备份及恢复技术（简称容灾技术）[2~4]也就应运而生了。

目前各种容灾技术最为重要的问题，一个是要尽量缩短两次备份之间的间隔（RPO），以保证尽可能少的数据丢失，另一个是要在发生灾难后，系统需要进行恢复之时，保证尽可能短的服务恢复时间（RTO），从而保证服务连续性。到目前为止，有两类容灾技术解决上述问题的效果最好：一是镜像技术（Mirror），二是连续数据保护技术（CDP）[5,6]。Mirror 的原理是通过使用两套冗余的数据存储设备，甚至两套完全一样的计算机系统，实现了在发生灾难导致主运行系统的数据存储设备或者全套系统发生错误时，服务由另一套冗余设备接管从而无缝连续运行。其代价巨大，且无法将服务恢复到一个历史状态，而在某些特定灾难（比如病毒攻击）下，历史状态的恢复是必须的。CDP 可以弥补镜像技术的不足，实际上这两种技术在实际部署中也往往是结合起来使用的。CDP 技术是将每次的数据变动都进行连续备份的一种数据保护技术，采用 CDP 技术的系统可以恢复到任何一个历史时刻的状态。因此从理论上来说，在发生灾难后需要进行恢复时，采用 CDP 技术的系统可以精确地恢复到灾难发生前最近一个时刻的备份点，由于这个时间差可能非常的小，因此往往也可以假设所需要恢复的数据量很小。从这个角度说，CDP 技术被认为是可以做到快速的灾难恢复。但如果这个极小的时间差所带来的数据变化非常的大，极端情况下比如全系统由于灾难而完全毁坏了的话，由于所需要恢复的数据量极大[7]，CDP 技术也就丧失了其优势，这种情况下

［收稿日期］ 2009－08－13；修回日期 2009－08－25
［作者简介］ 郑纬民（1946－），男，浙江宁波市人，清华大学教授，博士生导师，研究方向为计算机系统结构，并行与分布式系统等；E－mail：zwm－dcs@tsinghua.edu.cn

其恢复速度将非常慢。

笔者将论述一种全新的系统容灾保护方法,它脱离了传统基于数据保护的容灾保护思路,通过复制包括数据状态以及服务运行状态在内的全系统状态,并引入并行恢复的思想,最终达到了无论系统毁坏程度如何,都能保证服务即时恢复的目的。同时,与传统技术相比,该技术还可独立于具体设备和应用,容易做到容灾资源的共享从而节省了容灾系统的建设成本。

2 对传统数据保护技术的反思

笔者分析传统基于数据保护的容灾技术,并提出一种一致备份模型以及并行恢复模型对数据保护技术所产生的问题进行弥补,并讨论传统的 RTO 定义所存在的问题。

2.1 应用程序一致性

现存的容灾技术主要采用的都是基于数据保护的方法。通过将所保护的计算机系统中的持久性存储状态(磁盘状态)复制到一个物理隔离的备份站点,实现了对数据的保护,而计算机系统则通过读取这些数据而恢复到灾难发生之前的状态。这个过程中,存在一个应用程序自身状态与数据存储状态之间的一致性问题。这个问题在某些特殊场合越发突出,比如一些系统为了提高效率,将非结构化的数据文件存储在文件服务器中,而将这些数据文件的索引存储在结构化数据库中[8],如果这种对应关系由于应用程序的一致性问题没有得到维持,将带来严重的错误。

在传统数据保护技术中,一致性维护的方法可分为两类。第一类方法常用于周期性备份的场合。当应用程序运行期间,如果要进行备份的话,需要在备份前采用各种不同的方法来确保所备份的数据具有一个应用一致的状态。例如飞康备份软件采用的数据库 DB Snapshot Agent[9],该程序的任务就是确保将要备份的数据库具有事务完整性,也就是不能有一些数据在内存中没有处理完,而必须将状态持久化到存储系统中去。这样,在对该备份状态进行恢复时,数据库就不必经历一个长时间的状态检查过程,从而缩短了恢复时间。显然,为了达到这种应用一致性,所采用的方法必须是要建立在对应用程序语义的了解之上,具有强烈的应用相关性。

第二类一致性维护的方法常用于 CDP[5] 系统中。CDP 由于将所有的数据更新都进行了即时的备份,因此可以做到零数据丢失。但不幸的是,由于备份操作过于频繁,CDP 很难在数据复制时找到一个合适的时间点进行应用程序一致性的确认[6]。这可以通过一个最简单的应用场景进行解释,假设有一个应用程序,它不断从网络上收到数据并将这些数据写入磁盘。当 CDP 系统要对这样一个应用进行保护时,它必须将所收到的任何数据都记录下来,并将它们发送给备份站点进行存储。这期间,为了保证应用程序一致性,任何收到的数据都必须立即更新到(flush)的磁盘上去,以确保这些数据没有被文件系统所缓存住。显然,由于对每一个接收到的数据都必须执行缓存刷新操作,而后才能写入磁盘,实际上也就相当于文件系统的缓存不存在了,这将大大降低整个系统的性能。因此,CDP 系统实际上在进行备份的时候是不保证应用一致性的,无论当时的磁盘状态与应用程序的状态是否一致,CDP 系统都将当时的磁盘状态进行远程备份,而在发生灾难或错误需要进行恢复时,CDP 系统才对以前存储的大量备份点进行验证,以期找到一个最接近于错误发生之前的干净的备份点,也就是具有应用程序一致性的备份点。由于备份点验证是一个非常耗时的过程,这就使得 CDP 系统的实际错误恢复实际上往往很长。

从以上分析可以看出,应用程序一致性问题对数据保护技术造成了很大困扰,它或者对数据备份过程造成了影响,或者对数据恢复过程造成了损害。这一切的结果是导致了一种应用程序依赖性强,而且非常耗时的容灾解决方案。上述问题产生的根源就在于我们要确保的是应用程序与磁盘存储具有一致的状态,但在实际备份中却仅仅备份了磁盘的状态,而这也是所有数据保护技术的基础。以此相对应的是近年来逐渐兴起的基于虚拟机的全系统复制技术[10~16],这类技术是从整个运行系统的角度出发,通过同时复制包括持久化数据存储状态以及程序运行状态,应用程序一致性可以随时都得到保证,为容灾技术提供了新的思路。笔者等人所提出的结构无关的容灾理论[17],也正是建立在数据保护技术与全系统复制技术的融合基础之上的。

2.2 恢复时间目标(RTO)

作为在容灾规划中经常使用的指标,恢复点目标(RTO)[18,19]指定了发生灾难后,在应用服务恢复之间的最大允许延迟。换言之 RTO 指定了被保护服务的最大当机时间。在基于数据保护技术的容灾

方案中,这包括了数据的恢复时间以及服务重启和恢复的时间。在实际系统中,往往采用一个测量指标,服务恢复时间 TTR(time to recovery)对实际的服务恢复时间进行测量,并与预先定义的 RTO 指标进行比较。从数据保护的角度而言,TTR 指标有时会被错误测量。比如 Keeton 将一个系统恢复的时间定义为"从离线磁带库中获取磁带,并按照最后一次全备份以及最近的增量备份进行恢复的时间"[19]。据此定义,TTR 将仅仅包含数据恢复时间,而不包含服务重启和恢复的时间。

另一个常用的指标是服务修复时间(time to repair)[20,21],它是从 Jim Gray[22]所定义的平均修复时间(mean time to repair)引申而来的。服务修复时间表示将服务恢复到其正常执行状态所需要的时间,即服务修复时间是通过计算服务恢复到其 100 % 服务质量而得到的。很显然,它与服务恢复时间(time to recovery)具有同样的缩写,也很容易混淆。因此,往往使用另一个指标 BTN(back to normal time)来替代它表示同样的意思[20]。

综上所述,BTN 可能会比 TTR 要小。在基于数据保护技术的容灾恢复过程中,服务只能在所有数据都恢复完成后才能被重新启动,而其在启动后就将达到其应有的全部性能。因此,在此情况下,TTR 和 BTN 是相等的。如果 BTN 在某种情况下比 TTR 要大,就必然意味着在服务重启后存在一定时间的降级服务过程。笔者等人提出一种新的指标 SL(service level),作为对可能存在的服务降级的描述。SL 可能根据应用的不同,以服务响应时间、吞吐率等形式进行计算,但 SL 的提出意味着 RTO 是不能像以往一样单独描述容灾恢复的性能,它必须与 SL 相结合对容灾过程进行完整的描述,因为基于数据保护技术的容灾方案已经不能代表容灾技术的全部了,在结构无关的容灾方案中,服务降级是恢复过程中的常态。

3 并行容灾恢复

传统的容灾过程实际上是一种串行的恢复过程,这可以从 Keeton 的恢复图理论(recovery graph)[23]中明显看到。该过程可以大致分为两个阶段:数据恢复以及服务重建。这两个阶段是串行完成的,在结构无关的容灾体系下,我们所关注的不仅仅是数据存储状态的备份问题,同时也关心系统运行状态的备份问题,这就为缩短容灾恢复时间提供了新的思路。笔者等人提出一种能够实现快速恢复的容灾恢复方法,该方法通过将数据存储状态的恢复与系统运行状态的恢复并行执行,可以得到更短的容灾恢复时间。

如果我们将整个恢复过程进行细粒度划分,比如将整个恢复过程划分为 N 个片段,按照执行次序依次为:S_1,S_2,$\cdots$ S_N。对于每个指令片段来说,并不需要等待所有数据 D_{ALL} 全部被恢复完成后才可以执行。S_i 的执行仅仅依赖于完成相应工作所需要的最小数据集。以总共 3 个指令片段为例,在数据集 D_1^S 已经恢复完毕后,S_1 就可以开始执行了,而 D_2^S 恢复完成后,S_2 也可开始执行,以此类推。在数据获取以及服务重建之间重合的那部分也就是实际可以并行恢复的部分。

将其与传统容灾恢复过程相比,并行容灾恢复过程加速了服务重建过程,然而也带来了可能随之而来的服务降级问题。我们可以选择一种数据恢复次序,该次序可以得到最小化的 RTO 和 BTN,同时也尽可能将服务等级 SL 维持在一个合理水平。

为了简化问题,首先做以下假设:a. 系统中总是存在足够的资源,即在数据集与指令集之间没有因为资源竞争而产生的依赖关系。b. 每一指令片段的执行时间以及数据传输带宽均为常数。该问题可定义如下:

问题:在以上假设条件下,给定一个串行恢复指令序列,目标是找到一个有效的数据恢复顺序,使得在限定条件 1 之下,目标 1 可以达成。

目标 1:TTR 和 BTN 应该达到最小化。

限制 1:S_i 所依赖的最小数据集 D_i 必须在 S_i 执行之前恢复就绪,$\sum_{k=1}^{i} D_k \subseteq \sum_{k=1}^{i} D_k^S$

对该问题可以采用一种快速恢复算法进行求解,其基本原理就是顺序执行指令,当遇到指令需要取的数据还没有被恢复时,冻结该指令以及与该指令有关的相关指令,并针对该部分需要的数据进行远程数据恢复,而后恢复指令的执行。在整个恢复过程中,还可结合不同的数据预取策略以提高网络带宽利用率。

4 即时恢复系统简介

结合并行恢复策略简要介绍所构建的容灾系统,其基本结构如图 4 所示。其中主要思路是在进行容灾备份时,同时备份进程运行状态以及存储空间状态,而在进行容灾恢复时,则首先恢复系统运

行,然后通过数据的按需获取,实现服务的即时启动和按序执行。

上述策略实现的前提是要能对进程空间进行控制,这可以通过 3 种可能的方式实现:进程检查点(process checkpoint)、基于容器的虚拟化(container based virtualization)以及虚拟机技术(virtual machine)[24~27]。由于容灾的过程中通常对资源隔离的需求较小,而对备份的额外开销较为敏感,笔者等人选择采用基于容器的虚拟化技术来对所需要保护的进程进行隔离和保护,实际操作中可以将整个操作系统的所有进程都放在容器中[25],以实现对整个操作系统的保护。我们将所保护的进程集合称为隔离进程组(IPG)。当然,在进行备份的时候,增量备份机制也可以很方便地在该框架下实现。

另一个实现结构无关容灾的前提是原有待保护系统已带有基于容器的虚拟化能力。由于虚拟化技术需要操作系统内核的支持,这就意味着一个待保护系统必须重新安装带虚拟化能力内核的操作系统,才可能获得这种能力,这对于实际的容灾过程是不可行的。比如:不可能要求一个已安装配置好的一个大型 Oracle 数据库系统为了容灾而重新进行操作系统安装及应用程序配置。因此,笔者等人引入了一种操作系统无缝迁移技术。

图 1 的左半部分是操作系统无缝迁移技术的示意图,待容灾的系统可能起初并没有安装上虚拟化的支持。首先制作一个包含虚拟化支持的 USB 闪存盘,但其中并没有任何虚拟子系统,然后将该 USB 盘装到待容灾系统之上并以它启动,并用一组转换程序将原待容灾系统转换为 USB 盘上的一个虚拟子系统,这样原待容灾系统的所有行为就完全受 USB 盘上操作系统控制了,从而能够实现结构无关的容灾。

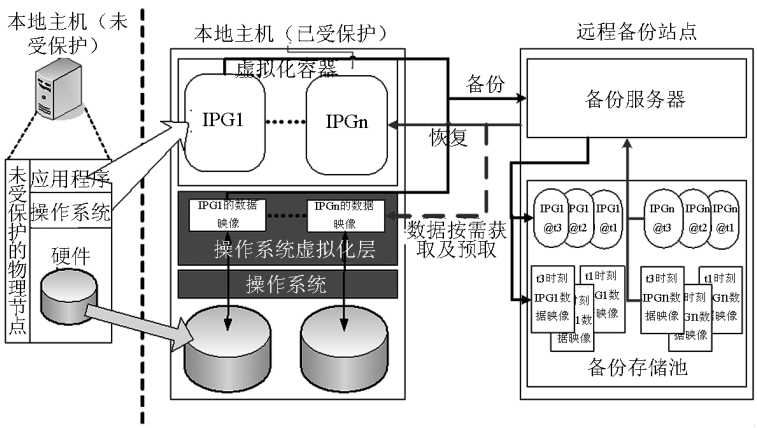


图 1　系统结构

Fig. 1　System architecture

具体的容灾恢复过程包括 3 个步骤:a. 通过冻结和解冻操作停止和触发 IPG 进程组的执行,这样就可以随时将 IPG 的运行进行分片和调度;b. 在恢复中的 IPG 以及远程数据备份之间建立一个映射关系,这样就可以在 IPG 运行时,为下一个恢复操作获取所需要的数据;c. 通过分析 IPG 的 I/O 请求,对当前数据的使用情况进行探测,并且预测最近所需要的数据有哪些,并以此来修正数据获取的顺序。

在原型系统中,笔者等人使用了两种恢复策略:带数据预取的策略(aggressive)以及不带数据预取的策略(lazy)。在不带数据预取的策略中,数据仅当 IPG 由于该数据被阻塞执行时才被获取,后台没有任何额外的传输活动;而在带数据预取的策略中,除了将传输当前所急需的数据之外,后台将始终存在一个数据预取的进程,该进程在网络带宽有空闲时将进行数据预取工作。

5　实验验证

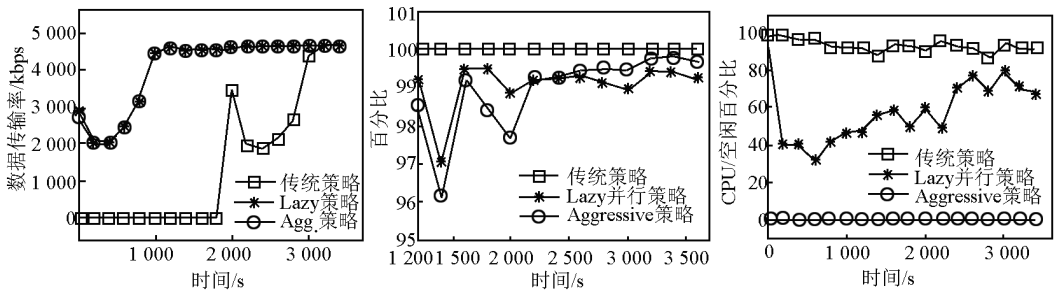
面向前面介绍的系统,笔者等人使用多种实际应用进行实验验证。实验环境包括 3 台计算机:A,B和 C。它们由 100 Mbps 的以太网进行连接。每台机器的配置都包括 Xeon 2.33G CPU,16 GB 内存,以及 6TB 的 SATA 磁盘。机器 A 运行笔者等人修改过的 Linux 2.6.18 内核,具备进行并行容灾恢复的能力,机器 B 和 C 都运行 Vanilla 2.6.18 系统。在所有实验中,机器 A 被用做受保护的应用服务器,机器 B 只充当 A 客户端的角色。机器 C 则被设置为容灾中心,存储机器 A 的备份状态。为了模拟

更真实的网络,我们对所有网络延迟都另外加上2 ms的延迟,以模拟容灾中心与受保护服务器距离100 km,并通过光纤连接的场景。

5.1 真实应用服务恢复实验

笔者等人通过一个真实的应用服务来展示即时服务恢复技术的好处,在此使用一个流媒体点播服务(VOD)。首先从 Internet 上得到了一个真实的 VOD 点播的 LOG[28],将该 LOG 在实验环境中进行重放,对每次点播事务,客户端都将产生一个线程并向服务器端发出视频请求,服务器端收到请求后将立即开始把视频数据以特定的码流速度发送给客户端。

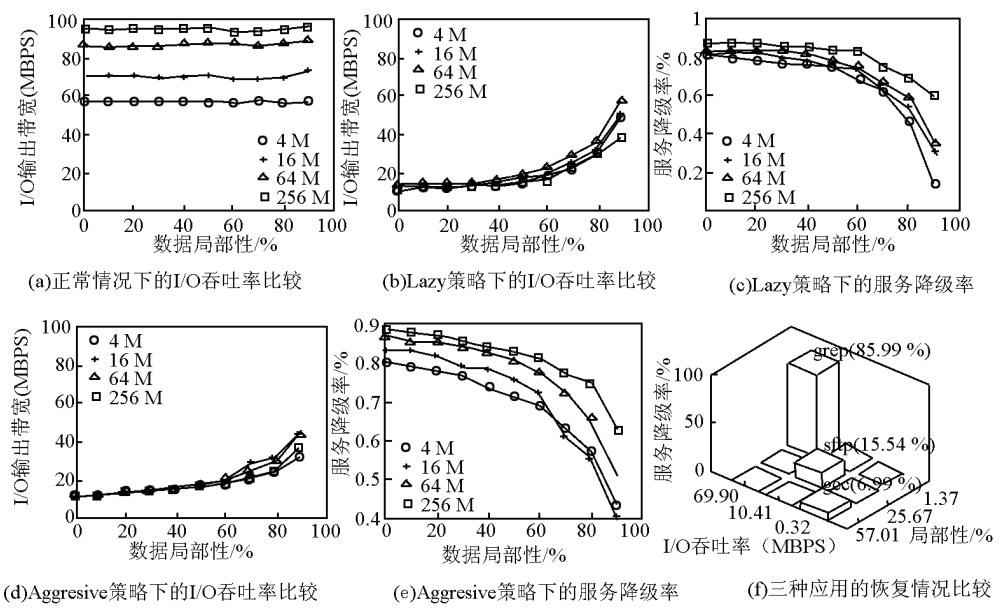
图2(a)将传统恢复过程与即时服务恢复进行了对比。其中平均数据传输率(ASDR)被定义为:$ASDR = \sum_{k=1}^{N} SDR_k/N$, 其中 N 是在此时间点的事务数,SDR_k 是第 k 个事务的当前数据传输率。我们使用 ASDR 来对 VOD 系统的服务等级 SL 进行测量。就像图中所显示的那样,我们所使用的两种策略都可以将服务能力立即恢复,而传统的恢复过程则需要 1 800 s 以后才能恢复提供服务能力。实际上,VOD 服务器上所存储的数据量越大,两者间的实际差距也会越大。此外,可以看到,VOD 服务等级 SL 几乎在此过程中没有受到什么显著影响。



(a) ASDR　(b)服务等级SL　(c)CPU消耗情况

图2　VOD 服务恢复结果

Fig. 2　VOD service recovery results



(a)正常情况下的I/O吞吐率比较　(b)Lazy策略下的I/O吞吐率比较　(c)Lazy策略下的服务降级率

(d)Aggresive策略下的I/O吞吐率比较　(e)Aggresive策略下的服务降级率　(f)三种应用的恢复情况比较

图3　应用特性对即时恢复的影响

Fig. 3　Effect of application characteristics to recovery performance

图2(b)中给出了传统容灾恢复技术在数据恢复之后的归一化服务等级,并将其与即时恢复中的服务等级进行了比较。为了使这种比较更加清晰,笔者等人将这种比较聚焦于服务能力已经稳定了的这一段,而去除了服务预热阶段的结果。与传统服务恢复过程相比,即时恢复的服务降级大约在0.6 %~4 %。图2(c)中还列出了不同恢复过程的CPU消耗情况。对于传统恢复而言,数据恢复过程几乎耗尽了全部CPU,但数据一旦恢复完,恢复过程就不会再占用CPU了;即时恢复技术则需要边提供服务,边进行恢复,因此恢复过程带来的CPU消耗是始终存在的。为了让比较更为合理,笔者等人将传统恢复的曲线左移并仅仅使用了其在数据恢复后的CPU消耗数据,这样可以更容易看出差别。

5.2 不同应用程序特性对恢复过程的影响

笔者等人设计并实现了一个基于磁盘复制工具dd的工具,命名为dd-util。以此模拟不同I/O特性应用程序的行为,研究的特性包括I/O吞吐率特性及I/O局部性。在每次实验中,使用一个包含20 %写操作以及80 %读操作的workload来对共64 GB的数据进行访问,这些数据在不同实验中将被配置为不同的数据块大小以及数据局部性。这里数据局部性被定义为重复访问的数据与总数据之比。在此基础上,对3个实际的应用程序GCC,SFTP以及GREP的即时恢复性能进行了测试。

图3(a)给出了不同数据块大小以及I/O局部性下的I/O吞吐率。从图中可以看出,I/O吞吐率随着数据块大小的增长而增长,而数据局部性则对性能没有明显影响,这主要由于dd-util是以一种随机访问的形式运行的,从而屏蔽磁盘缓存的作用。图3(b)中显示了使用Lazy策略下的I/O吞吐率对比,很明显随着数据局部性的增强,I/O吞吐率从10 Mbps提高到了接近50 Mbps。图3(c)显示了SL的服务降级情况,此处SL被定义为$(Throughput_{normal}-Throughput_{lazy})/Throughput_{normal}$。该比率随着局部性的提高而降低,并且数据块越大,在正常情况下的I/O吞吐率越高,该比率也越高。综上可得出结论:当被保护的应用不是I/O密集型应用并且I/O局部性好的时候,即时恢复的服务降级问题是最小的。图3(d)和图3(e)显示了在aggressive策略下的相似结果。上述情况的根源在于后台预取所引发的资源竞争,预取策略越贪婪,资源竞争越明显。

为了验证这些结论,笔者等人采用3个实际应用任务进行了测试:a.用GCC编译一个Linux 2.6 Kernel;b.用SFTP客户端下载16 GB数据;c.使用GREP在一个包含40 GB的文件系统中查找字符串。上述任务执行过程的服务降级情况见图3(f),可以再次验证笔者等人的结论。

6 结语

容灾技术通过构建可靠的存储系统,实现在各种自然或人为灾难下,数据的安全保证和关键业务的持续运行,全系统毁坏对容灾技术的挑战更多。传统容灾技术下,全系统毁坏后除了使用镜像系统能快速进行恢复外,很难做到服务的即时恢复,而这种灾难发生后,历史状态的即时恢复就更无法实现。文章提出了一种即时服务恢复技术,无论被毁坏的数据多少,以及要恢复回原系统的哪个状态,服务恢复均可即时完成,该技术的基础是要备份和恢复整个系统而不仅是数据,并采用了一种并行恢复策略,通过虚拟化技术以数据按需恢复的方式缩短了容灾恢复时间。此外,应用该技术时原系统不需要进行任何修改,对用户十分友好,应用前景广阔。

参考文献

[1] Brett J L, Landry. Dispelling 10 common disaster recovery myths: lessons learned from hurricane katrina and other disasters[J]. ACM Journal on Educational Resources in Computing, 2006, 6(4)

[2] Khalid Saleem, Steven Luis, Yi Deng, et al. Towards a Business Continuity Information Network for Rapid Disaster Recovery, DGO 2008[C]. Montreal, Canada, 2008

[3] Ken Fong. Contingency Planning and Disaster Recovery[A]. ACM 1984 Annual Conference[C]. 1984

[4] Hector Garcia Molina, Christos A. Polyzois, Issues in disaster recovery[A]. the Thirty-Fifth IEEE Computer Society International Conference[C]. 1990

[5] Laden G, TaShma P, Yaffe E, et al. Fienblit, Architectures for Controller Based CDP[A]. FAST 2007[C]. San Jose, CA, 2007

[6] Akshat Verma, Kaladhar Voruganti, Ramani Routray, et al. SWEEPER: An Efficient Disaster Recovery Point Identification Mechanism[A]. FAST 2008[C]. San Jose, CA, 2008

[7] Weijun Xiao, Qing Yang. Can We Really Recover Data If Storage Subsystem Fails[A]. ICDCS 2008[C]. Beijing, China, 2008

[8] Suparna Bhattacharya, Mohan C, Karen W, et al. Coordinating Backup/Recovery and Data Consistency Between Database and File Systems[A]. SIGMOD 2002[C]. Madison, Wisconsin, 2002

[9] IPStor Snapshot Agents for Databases - At a Glance[R]. FalconStor Software, 2007, http://www.ids-g.com/Whitepapers/FalconStor/Brochures/Appdatabasesnapshots.pdf

[10] Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, et al. Remus:

High Availability via Asynchronous Virtual Machine Replication, NSDI 2008[C]. San Francisco, CA,2008
[11] Constantine P, Sapuntzakis, Ramesh Chandra, et al. Optimizing the Migration of Virtual Computers[A]. OSDI 2002[C]. Boston, MA,2002
[12] Steven Osman, Dinesh Subhraveti, Gong Su, et al, The Design and Implementation of Zap: A System for Migrating Computing Environments[A]. OSDI 2002[C]. Boston, MA, 2002
[13] Wei Huang, Jiuxing Liu, Matthew Koop, et al. Nomad: Migrating OS - bypass Networks in Virtual Machines[A]. VEE 2007 [C]. San Diego, CA,2007
[14] Rob Bradford, Evangelos Kotsovinos, Anja Feldmann et al. Live Wide - Area Migration of Virtual Machines Including Local Persistent State[A]. VEE 2007[C]. San Diego, CA,2007
[15] Sanjay Kumar, Karsten Schwan. Netchannel: A VMM - level Mechanism for Continuous, Transparent Device Access During VM Migration[A]. VEE 2008[C]. Seattle, WA,2008
[16] Thomas C, Bressoud, Fred B. Schneider: Hypervisor - based Fault - tolerance[A]. SOSP 1995[C]. Colorado, US, 1995
[17] Weimin Zheng, Binxing Fang. Structure - independent disaster recovery: concept, architecture and implementations[J]. Science in China Series F, 2009, 52(5):813 - 823
[18] Recovery time objective, Wikipedia[EB/OL]. http://en.wikipedia.org/wiki/Recovery_time_objective
[19] Kimberly Keeton, Cipriano Santos, Dirk Beyer, et al. Designing for disasters[A]. FAST 2004[C]. San Francisco, CA,2004
[20] Erik Vanden Meersch. Designing Highly Available Architectures: A Methodology[A]. Sun BluePrints Online,2002
[21] Jim Metzler, Rethinking MTTR. IT Impact Breifs[A]. Issue 4, May 2007, http://www.netscout.com/docs/itimpactbriefs/NetScout iib Metzler 0407 Rethinking MTTR.pdf
[22] Jim Gray. Why do computers stop and what can be done about it? [R]. Technical Report 85.7, PN87614,1985
[23] Kimberly Keeton, Dirk Beyer, Ernesto Brau, et al, On the Road to Recovery: Restoring Data after Disasters, In Eurosys 2006, Leuven, Belgium,2006
[24] Oren Laaden, Jason Nieh. Transparent Checkpoint - Restart of Multiple Processes on Commodity Operating Systems, USENIX 2007[C]. Santa Clara, CA,2007
[25] Stephen Soltesz, Herbert Potzl, Marc E, Containerbased Operating System Virtualization: A Scalable, High - performance Alternative to Hypervisors[A]. Eurosys 2007[C]. Lisbon, Portugal, 2007
[26] Michael Nelson, Beng Hong Lim, Greg Hutchins. Fast Transparent Migration for Virtual Machines[A]. USENIX 2005[C]. Anaheim, CA, 2005
[27] Christopher Clark, Keir Fraser, Steven Hand, et al. Andrew Warfield, Live Migration of Virtual Machines[A]. NSDI 2005 [C]. Boston, MA,2005
[28] Hongliang Yu, Dongdong Zheng, Ben Y, et al. Understanding user behavior in large - scale video - on - demand systems[A]. Eurosys 2006[C]. Leuven, Belgium,2006, http://hpc.cs.tsinghua.edu.cn/granary

Towards instant service rebuilding after site corruptions

Zheng Weimin

(*Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*)

[**Abstract**] The paper describes a new system protection method which is used for disaster recovery. It abandons the traditional data protection based DR(disaster recovery) achieves, replicates the whole system states, including data states together with service running states, and further introduces the method of parallel recovery to restore the interrupted services instantly. Compared with traditional techniques, the method can be independent from specific devices and applications. Different systems can share a unique DR system which is resource - saving.

[**Key words**] disaster recovery; virtualization; parallel