设计一个实用系统容错虚拟机

丹尼尔·J·斯凯尔斯、迈克·尼尔森和加内什·文基塔查拉姆
威睿公司
{scales,mnelson,ganesh}@vmware.com

抽象的

我们已经实现了一个商业企业级系统，用于提供容错虚拟机，基于通过另一台服务器上的备份虚拟机复制主虚拟机 (VM) 执行的方法。我们在VMware vSphere 4.0中设计了一个完整的系统，该系统易于使用，在商用服务器上运行，通常会使实际应用程序的性能降低不到10%。此外，对于一些实际应用程序来说，保持主虚拟机和辅助虚拟机同步执行所需的数据带宽小于 20 Mbit/s，这使得实现长距离容错成为可能。一个易于使用的商业系统，可以在故障后自动恢复冗余，除了复制虚拟机执行之外，还需要许多其他组件。我们设计并实现了这些额外的组件，并解决了在支持运行企业应用程序的虚拟机时遇到的许多实际问题。在本文中，我们描述了我们的基本设计，讨论了替代设计选择和许多实现细节，并提供了微基准测试和实际应用程序的性能结果。

1. 简介

实现容错服务器的一种常见方法是主/备份方法 [1]，其中，如果主服务器发生故障，备份服务器始终可以接管。备份服务器的状态必须始终与主服务器保持几乎相同，以便当主服务器发生故障时，备份服务器可以立即接管，从而对外部客户端隐藏故障，并且不会传输任何数据。丢失的。在备份服务器上复制状态的一种方法是将主服务器的所有状态（包括 CPU、内存和 I/O 设备）的更改几乎连续地传送到备份服务器。然而，发送此状态所需的带宽，特别是内存中的变化，可能非常大。

另一种可以使用更少带宽的复制服务器的方法有时被称为状态机方法 [13]。这个想法是将服务器建模为确定性状态机，通过从相同的初始状态启动它们并确保它们以相同的顺序接收相同的输入请求来保持同步。由于大多数服务器或服务都有一些不确定的操作，因此必须使用额外的协调来确保主服务器和备份保持同步。但是，保持主数据库和备份数据库同步所需的额外信息量远远少于主数据库中正在更改的状态量（主要是内存更新）。

实施协调以确保物理服务器 [14] 的确定性执行很困难，特别是随着处理器频率的增加。相比之下，运行在虚拟机管理程序之上的虚拟机 (VM) 是实现状态机方法的绝佳平台。VM 可以被认为是一个定义良好的状态机，其操作是被虚拟化的机器（包括其所有设备）的操作。与物理服务器一样，虚拟机有一些非确定性操作（例如读取时钟或发送中断），因此必须将额外信息发送到备份以确保其保持同步。由于虚拟机管理程序可以完全控制虚拟机的执行，包括所有输入的传递，

因此，状态机方法可以在商用硬件上为虚拟机实现，无需修改硬件，从而可以立即为最新的微处理器实现容错。此外，状态机方法所需的低带宽允许主设备和备份设备之间更大程度的物理分离。例如，复制的虚拟机可以在分布在整个园区的物理机上运行， 这比在同一建筑物中运行的虚拟机提供更高的可靠性。

我们在VMware vSphere 4.0平台上采用主备方式实现了容错虚拟机，高效运行全虚拟化的x86虚拟机。由于 VMware vSphere 实现了完整的 x86 虚拟机，因此我们能够自动为任何 x86 操作系统和应用程序提供容错能力。允许我们记录主数据库的执行并确保备份执行相同的基本技术被称为确定性重放[15]。VMware vSphere 容错 (FT) 基于确定性重放，但添加了必要的额外协议和功能来构建完整的容错系统。除了提供硬件容错之外，我们的系统通过在本地集群中的任何可用服务器上启动新的备份虚拟机，在发生故障后自动恢复冗余。目前，确定性重放和 VMware FT 的生产版本仅支持单处理器虚拟机。记录和重放多处理器虚拟机的执行仍在进行中，存在严重的性能问题，因为几乎每次对共享内存的访问都可能是非确定性操作。

Bressoud 和 Schneider [3] 描述了一个原型实现

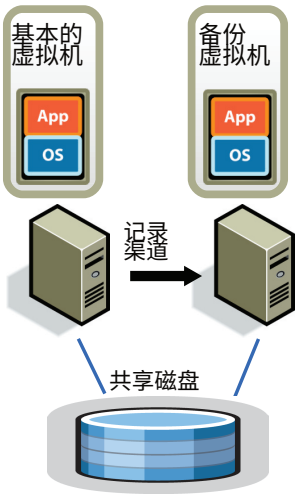

图 1：基本 FT 配置。

HP PA-RISC 平台的容错虚拟机的管理。我们的方法是相似的，但出于性能原因，我们做了一些根本性的改变，并研究了一些设计替代方案。此外，我们还必须在系统中设计和实现许多附加组件，并处理许多实际问题，以构建一个高效且可供运行企业应用程序的客户使用的完整系统。与讨论的大多数其他实用系统类似，我们只尝试处理故障停止故障[12]，这些故障是在故障服务器导致不正确的外部可见操作之前可以检测到的服务器故障。

本文的其余部分安排如下。首先，我们描述我们的基本设计并详细介绍我们的基本协议，以确保在主虚拟机发生故障后备份虚拟机接管时不会丢失数据。然后，我们详细描述了构建强大、完整和自动化系统必须解决的许多实际问题。我们还描述了为实现容错虚拟机而出现的几种设计选择，并讨论了这些选择中的权衡。接下来，我们给出一些基准测试和一些实际企业应用程序的实施结果。最后，我们描述了相关工作并得出结论。

2. 基本 FT 设计

图 1 显示了我们的容错虚拟机系统的基本设置。对于我们希望提供容错功能的给定虚拟机（*基本的*VM），我们运行*备份*不同物理服务器上的虚拟机与主虚拟机保持同步并以相同的方式执行，但存在较小的时间延迟。我们说这两个虚拟机位于*虚拟锁步*。VM 的虚拟磁盘位于共享存储（例如光纤通道或 iSCSI 磁盘阵列）上，因此主 VM 和备份 VM 可以访问它们以进行输入和输出。（我们将在第 4.1 节中讨论主虚拟机和备份虚拟机具有单独的非共享虚拟磁盘的设计。）只有主虚拟机在网络上通告其存在，因此所有网络输入都会到达主虚拟机。同样，所有其他输入（例如键盘和鼠标）仅发送至主虚拟机。

主虚拟机接收到的所有输入都会发送到通过网络连接备份虚拟机*记录通道*。对于服务器工作负载，主要的输入流量是网络和磁盘。如下文第 2.1 节中讨论的附加信息根据需要进行传输，以确保备份 VM 以与主 VM 相同的方式执行非确定性操作。结果是备份虚拟机始终以与主虚拟机相同的方式执行。但是，备份虚拟机的输出会被虚拟机管理程序删除，因此只有主虚拟机会生成返回给客户端的实际输出。如第 2.2 节所述，主虚拟机和备份虚拟机遵循特定协议，包括备份虚拟机的显式确认，以确保在主虚拟机发生故障时不会丢失数据。

为了检测主虚拟机或备份虚拟机是否发生故障，我们的系统结合使用相关服务器之间的检测信号和监控日志通道上的流量。此外，我们必须确保只有一个主虚拟机或备份虚拟机接管执行，即使存在主服务器和备份服务器彼此失去通信的裂脑情况。

在以下部分中，我们将提供有关几个重要领域的更多详细信息。在第 2.1 节中，我们详细介绍了确定性重放技术，该技术确保主虚拟机和备份虚拟机通过日志通道发送的信息保持同步。在第 2.2 节中，我们描述了 FT 协议的基本规则，该规则确保在主节点发生故障时不会丢失数据。在第 2.3 节中，我们描述了以正确方式检测和响应故障的方法。

2.1 确定性重放的实现

正如我们所提到的，复制服务器（或虚拟机）执行可以建模为确定性状态机的复制。如果两个确定性状态机以相同的初始状态启动并以相同的顺序提供完全相同的输入，那么它们将经历相同的状态序列并产生相同的输出。虚拟机具有广泛的输入，包括传入的网络数据包、磁盘读取以及来自键盘和鼠标的输入。非确定性事件（例如虚拟中断）和非确定性操作（例如读取处理器的时钟周期计数器）也会影响VM的状态。这给复制运行任何操作系统和工作负载的任何虚拟机的执行带来了三个挑战：(1) 正确捕获确保备份虚拟机确定性执行所需的所有输入和非确定性，(2) 正确地将输入和非确定性应用于备份虚拟机，以及 (3) 以不影响备份虚拟机的方式执行此操作。 t 降低性能。此外，x86 微处理器中的许多复杂操作具有未定义的、因此不确定的副作用。捕获这些未定义的副作用并重播它们以产生相同的状态提出了额外的挑战。

VMware 确定性重放 [15] 为 VMware vSphere 平台上的 x86 虚拟机提供了这一功能。确定性重放将 VM 的输入以及与写入日志文件的日志条目流中的 VM 执行相关的所有可能的非确定性记录下来。稍后可以通过从文件中读取日志条目来准确地重播VM执行。对于非确定性操作，记录足够的信息以允许使用相同的状态改变和输出来再现该操作。对于非确定性事件，例如定时器或 IO 完成输入

中断时，事件发生的确切指令也会被记录。在重放期间，事件在指令流中的同一点传递。VMware 确定性重播实现了高效的事件记录和事件传递机制，该机制采用了各种技术，包括使用与 AMD [2] 和 Intel [8] 联合开发的硬件性能计数器。

Bressoud 和 Schneider [3] 提到将 VM 的执行划分为 epoch，其中诸如中断之类的非确定性事件仅在 epoch 结束时传递。纪元的概念似乎被用作批处理机制，因为在每个中断发生的确切指令处单独传递每个中断的成本太高。然而，我们的事件传递机制足够高效，VMware 确定性重放不需要使用纪元。每个中断发生时都会被记录下来，并在重放时有效地传递到适当的指令。

2.2 FT协议

对于VMware FT，我们使用确定性重播来生成必要的日志条目来记录主VM的执行情况，但我们不是将日志条目写入磁盘，而是通过日志记录通道将它们发送到备份VM。备份虚拟机实时重放条目，因此执行方式与主虚拟机相同。然而，我们必须在日志通道上使用严格的 FT 协议来增加日志条目，以确保我们实现容错。我们的基本要求如下：

> 输出要求：如果备份虚拟机在主虚拟机发生故障后接管，则备份虚拟机将以与主虚拟机发送到外部世界的所有输出完全一致的方式继续执行。

请注意，经过*故障转移*发生（即，备份虚拟机在主虚拟机故障后接管）时，备份虚拟机可能会以与主虚拟机继续执行的方式完全不同的方式开始执行，因为在执行期间会发生许多非确定性事件。但是，只要备份 VM 满足输出要求，在故障转移到备份 VM 期间就不会丢失外部可见的状态或数据，并且客户端将注意到其服务中不会出现中断或不一致。

可以通过延迟任何外部输出（通常是网络数据包）直到备份虚拟机接收到允许其重播执行至少到该输出操作点的所有信息来确保输出要求。一个必要条件是备份虚拟机必须已接收到输出操作之前生成的所有日志条目。这些日志条目将允许它执行到最后一个日志条目的位置。然而，假设在主节点执行输出操作后立即发生故障。备份虚拟机必须知道它必须保持重放直到输出操作点，并且仅在该点“上线”（停止重放并接管为主虚拟机，如第 2.3 节所述）。如果备份要在输出操作之前的最后一个日志条目处生效，

考虑到上述限制，执行输出要求的最简单方法是在以下位置创建一个特殊的日志条目：

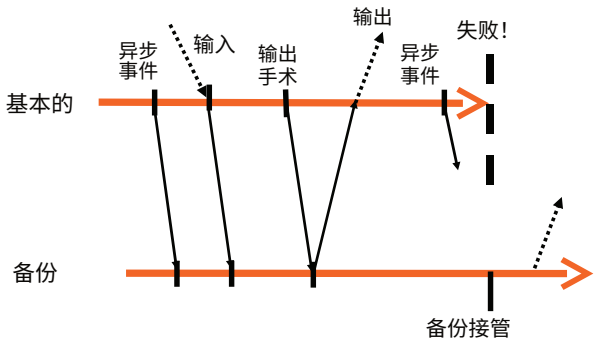

图 2：FT 协议。

每个输出操作。然后，输出要求可以通过以下特定规则强制执行：

> 输出规则：主VM可能不会向外部世界发送输出，直到备份VM接收并确认与产生输出的操作相关联的日志条目。

如果备份虚拟机已接收到所有日志条目，包括用于输出生成操作的日志条目，则备份虚拟机将能够准确地重现主虚拟机在该输出点的状态，因此如果主虚拟机死亡，备份将正确达到与该输出一致的状态。相反，如果备份虚拟机在没有接收到所有必要的日志条目的情况下接管，则其状态可能会很快出现分歧，从而与主虚拟机的输出不一致。输出规则在某些方面类似于[11]中描述的方法，其中“外部同步”IO实际上可以被缓冲，只要它在下一次外部通信之前实际写入磁盘即可。

请注意，输出规则没有说明有关停止主虚拟机执行的任何内容。我们只需要延迟输出的发送，但VM本身可以继续执行。由于操作系统通过异步中断进行非阻塞网络和磁盘输出以指示完成，因此虚拟机可以轻松地继续执行，并且不一定会立即受到输出延迟的影响。相比之下，之前的工作 [3, 9] 通常表明，在执行输出之前，主 VM 必须完全停止，直到备份 VM 确认来自主 VM 的所有必要信息。

作为示例，我们在图 2 中显示了一张图表，说明了 FT 协议的要求。该图显示了主虚拟机和备份虚拟机上的事件时间线。从主线到备份线的箭头表示日志条目的传输，从备份线到主线的箭头表示确认。有关异步事件、输入和输出操作的信息必须作为日志条目发送到备份并得到确认。如图所示，到外部世界的输出被延迟，直到主VM从备份VM接收到其已接收到与输出操作相关联的日志条目的确认为止。如果遵循输出规则，备份虚拟机将能够以与主虚拟机最后输出一致的状态进行接管。

我们不能保证在故障转移情况下所有输出都恰好生成一次。当主节点打算发送输出时，如果不使用具有两阶段提交的事务，则备份节点无法确定主节点是否在发送其最后一个输出之前或之后立即崩溃。幸运的是，网络基础设施（包括 TCP 的常见使用）旨在处理丢失的数据包和相同（重复）的数据包。请注意，在主服务器发生故障期间，传入主服务器的数据包也可能会丢失，因此不会传送到备份服务器。然而，传入的数据包可能会由于与服务器故障无关的多种原因而被丢弃，因此网络基础设施、操作系统和应用程序都经过编写，以确保它们可以补偿丢失的数据包。

2.3 检测和响应故障

如上所述，如果另一个虚拟机出现故障，主虚拟机和备份虚拟机必须快速响应。如果备份虚拟机出现故障，主虚拟机将*上线*–也就是说，离开记录模式（并因此停止在日志记录通道上发送条目）并开始正常执行。如果主虚拟机发生故障，备份虚拟机也应类似*上线*，但过程稍微复杂一些。由于其执行滞后，备份虚拟机可能会有许多已接收并确认但尚未被消耗的日志条目，因为备份虚拟机尚未达到其执行的适当点。备份虚拟机必须继续从日志条目重播其执行，直到消耗完最后一个日志条目。此时，备份虚拟机将停止重放模式并开始像普通虚拟机一样执行。实质上，备份虚拟机已升级为主虚拟机（并且现在缺少备份虚拟机）。由于它不再是备份虚拟机，因此当来宾操作系统执行输出操作时，新的主虚拟机现在将向外部世界产生输出。在转换到正常模式期间，可能需要一些特定于设备的操作才能正确发生此输出。特别是，出于网络目的，VMware FT 自动在网络上通告新主虚拟机的 MAC 地址，以便物理网络交换机知道新主虚拟机位于哪台服务器上。此外，新升级的主VM可能需要重新发出一些磁盘IO（如3.4节所述）。

有许多可能的方法可以尝试检测主虚拟机和备份虚拟机的故障。VMware FT 在运行容错虚拟机的服务器之间使用 UDP 检测信号来检测服务器何时可能崩溃。此外，VMware FT 还监控从主虚拟机发送到备份虚拟机的日志流量以及从备份虚拟机发送到主虚拟机的确认。由于定期的计时器中断，日志记录流量应该是有规律的，并且对于正常运行的客户操作系统来说永远不会停止。因此，日志条目或确认流的停止可能表明虚拟机发生故障。如果检测信号或日志记录流量停止时间超过特定超时（大约几秒），则声明失败。

然而，任何此类故障检测方法都容易受到裂脑问题的影响。如果备份服务器停止从主服务器接收心跳，则可能表明主服务器已发生故障，或者可能仅意味着仍在运行的服务器之间的所有网络连接已丢失。如果备份虚拟机随后上线，而主虚拟机则上线 VM 实际上仍在运行，可能会出现数据损坏以及客户端与 VM 通信的问题。因此，我们必须确保在检测到故障时只有主虚拟机或备份虚拟机之一上线。为了避免脑裂问题，我们使用共享存储来存储虚拟机的虚拟磁盘。当主虚拟机或备份虚拟机想要上线时，它会在共享存储上执行原子测试和设置操作。如果操作成功，则允许虚拟机上线。如果操作失败，则另一个虚拟机一定已经上线，因此当前虚拟机实际上会自行停止（“自杀”）。如果虚拟机在尝试执行原子操作时无法访问共享存储，那么它只会等待，直到可以访问为止。请注意，如果由于存储网络出现故障而无法访问共享存储，那么虚拟机可能无论如何都无法执行有用的工作，因为虚拟磁盘驻留在同一共享存储上。因此，使用共享存储来解决裂脑情况不会带来任何额外的不可用性。

该设计的最后一个方面是，一旦发生故障并且其中一个虚拟机上线，VMware FT 就会通过在另一台主机上启动新的备份虚拟机来自动恢复冗余。尽管之前的大多数工作并未涵盖此过程，但它对于使容错虚拟机发挥作用至关重要，并且需要仔细设计。第 3.1 节给出了更多详细信息。

3. FT的实际实施

第 2 部分描述了我们的 FT 基本设计和协议。然而，为了创建一个可用的、健壮的和自动化的系统，还必须设计和实现许多其他组件。

3.1 启动和重新启动 FT 虚拟机

必须设计的最大附加组件之一是在与主虚拟机相同的状态下启动备份虚拟机的机制。当发生故障后重新启动备份虚拟机时，也会使用此机制。因此，该机制必须可用于处于任意状态（即不仅仅是启动）的正在运行的主VM。此外，我们希望该机制不会显着中断主虚拟机的执行，因为这会影响虚拟机的任何当前客户端。

对于 VMware FT，我们调整了 VMware vSphere 的现有 VMotion 功能。VMware VMotion [10] 允许将正在运行的虚拟机从一台服务器迁移到另一台服务器，同时将中断降至最低——虚拟机暂停时间通常不到一秒。我们创建了一种修改形式的 VMotion，它可以在远程服务器上创建虚拟机的精确运行副本，但不会破坏本地服务器上的虚拟机。也就是我们修改后的*FT VMotion*将虚拟机克隆到远程主机而不是迁移它。FT VMotion 还设置日志记录通道，并使源 VM 作为主 VM 进入日志记录模式，并使目标 VM 作为新备份进入重放模式。与普通 VMotion 一样，FT VMotion 通常会中断主虚拟机的执行不到一秒。因此，在正在运行的虚拟机上启用 FT 是一项简单、无中断的操作。

启动备份虚拟机的另一个方面是选择运行它的服务器。容错虚拟机在可以访问共享存储的服务器集群中运行，因此所有虚拟机通常可以在集群中的任何服务器上运行。这种灵活性使 VMware vSphere 能够恢复 FT 冗余，即使在

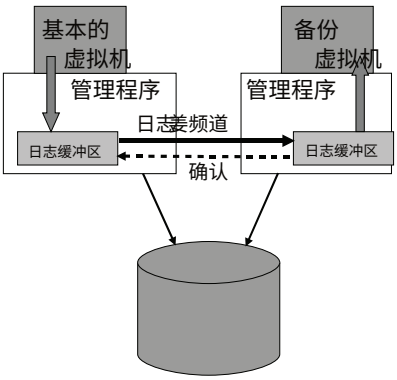

图 3：FT 记录缓冲区和通道。

一台或多台服务器出现故障。VMware vSphere 实施维护管理和资源信息的集群服务。当发生故障并且主虚拟机现在需要新的备份虚拟机来重新建立冗余时，主虚拟机会通知集群服务它需要新的备份。集群服务根据资源使用情况和其他限制确定运行备份虚拟机的最佳服务器，并调用 FT VMotion 来创建新的备份虚拟机。结果是，VMware FT 通常可以在服务器发生故障后几分钟内重新建立虚拟机冗余，并且容错虚拟机的执行不会出现任何明显中断。

3.2 管理日志通道

管理日志通道上的流量有许多有趣的实现细节。在我们的实现中，虚拟机管理程序维护一个大缓冲区来记录主虚拟机和备份虚拟机的条目。当主虚拟机执行时，它会在日志缓冲区中生成日志条目，同样，备份虚拟机会消耗其日志缓冲区中的日志条目。主数据库的日志缓冲区的内容会尽快刷新到日志记录通道，日志条目一到达就会从日志通道读取到备份的日志缓冲区。每次备份从网络将一些日志条目读入其日志缓冲区时，都会将确认发送回主服务器。这些确认允许 VMware FT 确定何时可以发送因输出规则而延迟的输出。图 3 说明了此过程。

如果备份虚拟机在需要读取下一个日志条目时遇到空日志缓冲区，它将停止执行，直到有新的日志条目可用。由于备份虚拟机不与外部通信，因此此暂停不会影响虚拟机的任何客户端。类似地，如果主VM在需要写入日志条目时遇到日志缓冲区已满，则它必须停止执行，直到可以刷新日志条目。这种执行停止是一种自然的流量控制机制，当主虚拟机以过快的速度生成日志条目时，它会减慢主虚拟机的速度。但是，此暂停可能会影响 VM 的客户端，因为主 VM 将完全停止且无响应，直到它可以记录其条目并继续执行。因此，我们的实现必须设计成最大限度地减少主日志缓冲区被填满的可能性。

主日志缓冲区可能填满的原因之一是备份虚拟机执行速度太慢，因此消耗日志条目的速度太慢。一般来说，备份虚拟机必须能够以与主虚拟机记录执行的速度大致相同的速度重播执行。幸运的是，VMware 确定性重放中的记录和重放的开销大致相同。但是，如果托管备份虚拟机的服务器负载其他虚拟机很重（因此资源过度使用），则尽管尽了最大努力，备份虚拟机可能无法获得足够的 CPU 和内存资源来与主虚拟机一样快地执行备份虚拟机管理程序的 VM 调度程序。

除了避免日志缓冲区填满时出现意外暂停之外，我们不希望执行延迟变得太大还有另一个原因。如果主虚拟机发生故障，备份虚拟机必须通过重播它在上线并开始与外部世界通信之前已确认的所有日志条目来“赶上”。完成重放的时间基本上是故障点的执行延迟时间，因此备份上线的时间大致等于故障检测时间加上当前执行延迟时间。因此，我们不希望执行延迟时间过长（超过一秒），因为这会显着增加故障转移时间。

因此，我们有一个额外的机制来减慢主虚拟机的速度，以防止备份虚拟机落后太多。在发送和确认日志条目的协议中，我们发送附加信息来确定主虚拟机和备份虚拟机之间的实时执行延迟。通常执行延迟小于 100 毫秒。如果备份虚拟机开始出现显着的执行延迟（例如，超过 1 秒），VMware FT 会通知调度程序为其提供稍少的 CPU 量（最初仅几个百分点），从而开始减慢主虚拟机的速度。我们使用缓慢的反馈循环，它将尝试逐渐确定主虚拟机的适当 CPU 限制，从而允许备份虚拟机与其执行相匹配。如果备份虚拟机继续落后，我们将继续逐步减少主虚拟机的 CPU 限制。

请注意，主虚拟机的这种减速情况非常罕见，通常仅在系统承受极大压力时才会发生。第 5 节的所有性能数据都包括任何此类减速的成本。

3.3 FT虚拟机上的操作

另一个实际问题是处理可应用于主VM的各种控制操作。例如，如果主虚拟机显式关闭，则备份虚拟机也应停止，而不是尝试上线。作为另一个示例，主服务器上的任何资源管理更改（例如增加的 CPU 份额）也应应用于备份服务器。对于这些类型的操作，特殊的控制条目在日志通道上从主服务器发送到备份服务器，以便对备份服务器执行适当的操作。

通常，VM 上的大多数操作应仅在主 VM 上启动。然后，VMware FT 发送任何必要的控制条目，以在备份虚拟机上引起适当的更改。唯一可以在主虚拟机和备份虚拟机上独立完成的操作是 VMotion。也就是说，主虚拟机和备份虚拟机可以独立于其他主机进行虚拟移动。请注意，VMware FT 确保两个虚拟机都不会移动到另一个虚拟机所在的服务器，

因为这种情况将不再提供容错能力。

与普通 VMotion 相比，主 VM 的 VMotion 增加了一些复杂性，因为备份 VM 必须与源主 VM 断开连接，并在适当的时间重新连接到目标主 VM。备份虚拟机的 VMotion 也有类似的问题，但增加了额外的复杂性。对于正常的 VMotion，我们要求所有未完成的磁盘 IO 在 VMotion 上发生最终切换时停顿（即完成）。对于主 VM，可以通过等待物理 IO 完成并将这些完成传递给 VM 来轻松处理这种停顿。然而，对于备份VM来说，没有简单的方法可以使所有IO在任何需要的点完成，因为备份VM必须重放主VM的执行并在同一执行点完成IO。主VM可能正在运行一个工作负载，其中在正常执行期间始终存在磁盘IO。VMware FT 有一个独特的方法来解决这个问题。当备份虚拟机处于 VMotion 的最终切换点时，它会通过日志记录通道请求主虚拟机暂时停顿其所有IO。然后，备份 VM 的 IO 将在单个执行点自然停顿，因为它会重放主 VM 的停顿操作执行。

3.4 磁盘IO的实现问题

有许多与磁盘 IO 相关的微妙实现问题。首先，考虑到磁盘操作是非阻塞的，因此可以并行执行，访问同一磁盘位置的同时磁盘操作可能会导致不确定性。此外，我们的磁盘 IO 实现直接使用 DMA 往返于虚拟机的内存，因此访问相同内存页面的同时磁盘操作也可能导致不确定性。我们的解决方案通常是检测任何此类 IO 竞争（这种情况很少见），并强制此类竞争磁盘操作在主数据库和备份数据库上以相同的方式顺序执行。

其次，磁盘操作还可能与 VM 中的应用程序（或操作系统）的内存访问竞争，因为磁盘操作通过 DMA 直接访问 VM 的内存。例如，如果虚拟机中的应用程序/操作系统正在读取内存块，同时对该块进行磁盘读取，则可能会出现不确定的结果。这种情况也不太可能发生，但是如果发生了我们一定要及时发现并处理。一种解决方案是在作为磁盘操作目标的页面上临时设置页面保护。如果 VM 碰巧访问的页面也是未完成的磁盘操作的目标，则页面保护会导致陷阱，并且可以暂停 VM，直到磁盘操作完成。由于更改页面上的 MMU 保护是一项昂贵的操作，因此我们选择使用*反弹缓冲区*。反弹缓冲区是一个临时缓冲区，其大小与磁盘操作访问的内存相同。修改磁盘读取操作以将指定数据读取到反弹缓冲区，并且仅在 IO 完成交付时才将数据复制到客户内存。类似地，对于磁盘写操作，首先将要发送的数据复制到反弹缓冲区，然后修改磁盘写入以从反弹缓冲区写入数据。使用反弹缓冲区会减慢磁盘操作速度，但我们还没有看到它导致任何明显的性能损失。

第三，当发生故障并且备份接管时，存在一些与主数据库上未完成（即未完成）的磁盘 IO 相关的问题。不可能对于新升级的主虚拟机，确定磁盘 IO 是否已发出到磁盘或成功完成。此外，由于备份虚拟机上的磁盘 IO 不是从外部发出的，因此当新升级的主虚拟机继续运行时，它们不会有显式的 IO 完成，这最终会导致虚拟机中的来宾操作系统启动一个新的虚拟机。中止或重置程序。我们可以发送一个错误完成来指示每个 IO 失败，因为即使 IO 成功完成，返回错误也是可以接受的。但是，来宾操作系统可能无法很好地响应来自其本地磁盘的错误。相反，我们在备份虚拟机上线过程中重新发出挂起的 IO。因为我们已经消除了所有竞争，并且所有 IO 都直接指定访问哪些内存和磁盘块，

3.5 网络IO的实现问题

VMware vSphere 为虚拟机网络提供了许多性能优化。其中一些优化基于管理程序异步更新虚拟机网络设备的状态。例如，在 VM 执行时，管理程序可以直接更新接收缓冲区。不幸的是，这些对虚拟机状态的异步更新增加了不确定性。除非我们能够保证所有更新都发生在主设备和备份设备上指令流中的同一点，否则备份设备的执行可能会与主设备的执行有所不同。

FT 网络仿真代码的最大变化是禁用了异步网络优化。使用传入数据包异步更新 VM 环形缓冲区的代码已被修改，以强制来宾捕获到虚拟机管理程序，它可以在其中记录更新，然后将其应用到 VM。同样，通常从传输队列中异步提取数据包的代码对于 FT 被禁用，而是通过陷阱完成到虚拟机管理程序的传输（除非如下所述）。

网络设备异步更新的消除以及 2.2 节中描述的发送数据包的延迟给网络带来了一些性能挑战。我们采取了两种方法来提高运行 FT 时的 VMnetwork 性能。首先，我们实施了集群优化以减少虚拟机陷阱和中断。当VM以足够的比特率传输数据时，管理程序可以为每组数据包执行一个传输陷阱，并且在最好的情况下为零陷阱，因为它可以将数据包作为接收新数据包的一部分来传输。同样，管理程序可以通过仅发布一组数据包的中断来减少 VM 因传入数据包而产生的中断数量。

我们的第二个网络性能优化涉及减少传输数据包的延迟。如前所述，虚拟机管理程序必须延迟所有传输的数据包，直到从备份中获得相应日志条目的确认。减少传输延迟的关键是减少将日志消息发送到备份并获得确认所需的时间。我们在这方面的主要优化包括确保发送和接收日志条目和确认都可以在没有任何线程上下文切换的情况下完成。VMware vSphere 虚拟机管理程序允许向 TCP 堆栈注册函数，每当收到 TCP 数据时，就会从延迟执行上下文（类似于 Linux 中的 tasklet）调用这些函数。这阿尔-

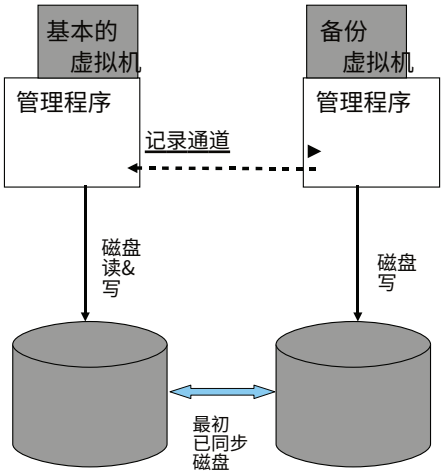

图 4：FT 非共享磁盘配置。

使我们能够快速处理备份上的任何传入日志消息以及主服务器收到的任何确认，而无需任何线程上下文切换。此外，当主虚拟机将要传输的数据包排入队列时，我们通过调度延迟执行上下文来强制立即刷新关联的输出日志条目（如第 2.2 节中所述）。

4. 设计方案

在实施 VMware FT 的过程中，我们探索了许多有趣的设计替代方案。在本节中，我们将探讨其中一些替代方案。

4.1 共享磁盘与非共享磁盘

在我们的默认设计中，主虚拟机和备份虚拟机共享相同的虚拟磁盘。因此，如果发生故障转移，共享磁盘的内容自然是正确且可用的。本质上，共享磁盘被视为主虚拟机和备份虚拟机的外部，因此对共享磁盘的任何写入都被视为与外部世界的通信。因此，只有主虚拟机才会实际写入磁盘，并且必须根据输出规则延迟写入共享磁盘。

另一种设计是让主虚拟机和备份虚拟机拥有单独的（非共享）虚拟磁盘。在此设计中，备份VM确实将所有磁盘写入其虚拟磁盘，并且在这样做时，它自然地保持其虚拟磁盘的内容与主VM的虚拟磁盘的内容同步。图 4 说明了此配置。对于非共享磁盘，虚拟磁盘本质上被视为每个 VM 内部状态的一部分。因此，根据输出规则，主节点的磁盘写入不必延迟。当主虚拟机和备份虚拟机无法访问共享存储时，非共享设计非常有用。出现这种情况的原因可能是共享存储不可用或太昂贵，或者运行主虚拟机和备份虚拟机的服务器相距较远（“远距离 FT”）。非共享设计的一个缺点是，首次启用容错时，必须以某种方式显式同步虚拟磁盘的两个副本。此外，磁盘在发生故障后可能会不同步，因此当备份虚拟机发生故障后重新启动时，必须显式重新同步它们。也就是说，FT VMotion 不仅要同步主备虚拟机的运行状态，还要同步它们的磁盘状态。

在非共享磁盘配置中，可能没有可用于处理裂脑情况的共享存储。在这种情况下，系统可以使用其他一些外部仲裁器，例如两个服务器都可以与之通信的第三方服务器。如果服务器是具有两个以上节点的集群的一部分，则系统可以选择使用基于集群成员资格的多数算法。在这种情况下，只有当虚拟机运行在属于包含大多数原始节点的通信子集群的一部分的服务器上时，才允许该虚拟机上线。

4.2 在备份虚拟机上执行磁盘读取

在我们的默认设计中，备份虚拟机从不读取其虚拟磁盘（无论是共享还是非共享）。由于磁盘读取被视为输入，因此很自然地将磁盘读取的结果通过日志记录通道发送到备份虚拟机。

另一种设计是让备份虚拟机执行磁盘读取，从而消除磁盘读取数据的记录。对于进行大量磁盘读取的工作负载，此方法可以大大减少日志记录通道上的流量。然而，这种方法有许多微妙之处。它可能会减慢备份虚拟机的执行速度，因为备份虚拟机必须执行所有磁盘读取，并等待（如果这些读取在主虚拟机上完成的虚拟机执行点时尚未物理完成）。

此外，还必须完成一些额外的工作来处理失败的磁盘读取操作。如果主数据库读取磁盘成功，但备份数据库读取相应磁盘失败，则必须重试备份数据库读取磁盘，直到成功，因为备份数据库必须在内存中获取与主数据库相同的数据。相反，如果主虚拟机读取的磁盘发生故障，则目标内存的内容必须通过日志记录通道发送到备份虚拟机，因为内存内容将是不确定的，并且备份虚拟机成功读取的磁盘不一定会复制该内存内容。

最后，如果将此磁盘读取替代方案与共享磁盘配置一起使用，则有一个微妙之处。如果主虚拟机对特定磁盘位置进行读取，然后很快对同一磁盘位置进行写入，则必须延迟磁盘写入，直到备份虚拟机执行第一次磁盘读取。这种依赖性可以被正确检测和处理，但会增加实现的额外复杂性。

在第 5.1 节中，我们给出了一些性能结果，表明在备份上执行磁盘读取可能会导致实际应用程序的吞吐量略有下降 (1-4%)，但也会显着减少日志记录带宽。因此，在日志通道的带宽非常有限的情况下，在备份虚拟机上执行磁盘读取可能会很有用。

5. 绩效评估

在本节中，我们对 VMware FT 在许多应用程序工作负载和网络基准测试中的性能进行基本评估。为了获得这些结果，我们在相同的服务器上运行主虚拟机和备份虚拟机，每个服务器都配备 8 个 Intel Xeon 2.8 Ghz CPU 和 8 GB RAM。服务器通过 10 Gbit/s 交叉网络连接，但在所有情况下都可以看到，使用的网络带宽远小于 1 Gbit/s。两台服务器都访问其共享的虚拟

	表现 （金融时报/非金融时报）	记录 带宽
规格Jbb2005	0.98	1.5 兆比特/秒
内核编译	0.95	3.0 兆比特/秒
Oracle 摇摆台	0.99	12 兆比特/秒
MS-SQL DVD 商店	0.94	18 兆比特/秒

表 1：基本性能结果

	根据 带宽	金融时报 带宽	记录 带宽
接收 (1Gb)	940	604	第730章
传输 (1Gb)	940	第855章	42
接收 (10Gb)	940	860	990
传输 (10Gb)	940	935	60

表 2：1Gb 和 10Gb 日志通道的网络传输和客户端接收性能（全部以 Mbit/s 为单位）

来自通过标准 4 Gbit/s 光纤通道网络连接的 EMC Clariion 的磁盘。用于驱动某些工作负载的客户端通过 1 Gbit/s 网络连接到服务器。

我们在性能结果中评估的应用程序如下。SPECJbb2005 是一个行业标准的 Java 应用程序基准测试，它非常消耗 CPU 和内存，并且执行的 IO 操作很少。内核编译是运行 Linux 内核编译的工作负载。此工作负载执行一些磁盘读取和写入操作，并且由于许多编译进程的创建和销毁，因此非常消耗 CPU 和 MMU。Oracle Swingbench 是一种工作负载，其中 Oracle 11g 数据库由 Swingbench OLTP（在线事务处理）工作负载驱动。此工作负载执行大量磁盘和网络 IO，并且有 80 个并发数据库会话。MS-SQL DVD 存储是一种工作负载，其中 Microsoft SQL Server 2005 数据库由 DVD 存储基准驱动，该基准有 16 个并发客户端。

5.1 基本性能结果

表 1 给出了基本性能结果。对于列出的每个应用程序，第二列给出了在运行服务器工作负载的 VM 上启用 FT 时的应用程序性能与在同一 VM 上未启用 FT 时的性能之比。计算性能比时，值小于 1 表示 FT 工作负载较慢。显然，在这些代表性工作负载上启用 FT 的开销不到 10%。SPECJbb2005 完全受计算限制，没有空闲时间，但性能良好，因为除了计时器中断之外，它具有最少的不确定性事件。其他工作负载执行磁盘 IO 并有一些空闲时间，因此 FT VM 的空闲时间较少，因此可能会隐藏一些 FT 开销。然而，

在表的第三列中，我们给出了运行这些应用程序时在日志记录通道上发送的数据的平均带宽。对于这些应用，记录带宽相当合理，1 Gbit/s 网络很容易满足。事实上，低带宽要求表明多个 FT 工作负载可以共享同一个 1 Gbit/s 网络，而不会产生任何负面性能影响。

对于运行 Linux 和 Windows 等常见来宾操作系统的 VM，我们发现来宾操作系统空闲时的典型日志记录带宽为 0.5-1.5 Mbits/sec。“空闲”带宽主要是记录定时器中断传递的结果。对于具有活动工作负载的虚拟机，日志记录带宽主要由必须发送到备份的网络和磁盘输入决定——接收的网络数据包和从磁盘读取的磁盘块。因此，记录带宽可能会很大

对于具有非常高网络接收或磁盘读取带宽的应用程序，该值高于表 1 中测量的值。对于这些类型的应用程序，日志记录通道的带宽可能是瓶颈，尤其是在日志通道有其他用途的情况下。

许多实际应用程序的日志通道所需的带宽相对较低，这使得基于重放的容错对于使用非共享磁盘的长距离配置非常有吸引力。对于主备之间可能相距 1-100 公里的长距离配置，光纤可以轻松支持 100-1000 Mbit/s 的带宽，且延迟小于 10 毫秒。对于表 1 中的应用，100-1000 Mbit/s 的带宽应该足以获得良好的性能。但请注意，主数据库和备份数据库之间的额外往返延迟可能会导致网络和磁盘输出延迟最多 20 毫秒。长距离配置仅适用于客户端可以容忍每个请求的额外延迟的应用程序。

对于两个磁盘最密集的应用程序，我们测量了在备份虚拟机上执行磁盘读取（如第 4.2 节中所述）与通过日志记录通道发送磁盘读取数据的性能影响。对于 Oracle Swingbench，在备份 VM 上执行磁盘读取时吞吐量大约降低 4%；对于 MS-SQL DVD 存储，吞吐量大约低 1%。同时，Oracle Swingbench 的日志记录带宽从 12 Mbits/秒减少到 3 Mbits/秒，MS-SQL DVD Store 的日志记录带宽从 18 Mbits/秒减少到 8 Mbits/秒。显然，对于磁盘读取带宽更大的应用程序来说，带宽节省可能会更大。正如 4.2 节中提到的，当在备份虚拟机上执行磁盘读取时，预计性能可能会稍差一些。但是，对于日志通道带宽有限的情况（例如，

5.2 网络基准测试

由于多种原因，网络基准测试对于我们的系统来说可能相当具有挑战性。首先，高速网络可能具有非常高的中断率，这需要以非常高的速率记录和重放异步事件。其次，以高速率接收数据包的基准测试将导致高速率的日志流量，因为所有此类数据包都必须通过日志记录通道发送到备份。第三，发送数据包的基准测试将受到输出规则的约束，该规则会延迟网络数据包的发送，直到收到备份的适当确认为止。此延迟将增加测量到的客户端延迟。这种延迟还可能会减少客户端的网络带宽，因为网络协议（例如 TCP）可能会

随着往返延迟的增加而降低网络传输速率。

表 2 给出了我们根据标准进行的多项测量的结果网络性能基准。在所有这些测量中，客户端 VM 和主 VM 通过 1 Gbit/s 网络连接。前两行给出了主主机和备份主机通过 1 Gbit/s 日志通道连接时的发送和接收性能。第三行和第四行给出了主备服务器通过10Gbit/s日志通道连接时的发送和接收性能，与1Gbit/s网络相比，不仅具有更高的带宽，而且延迟更低。作为粗略测量，1 Gbit/s 连接的虚拟机管理程序之间的 ping 时间约为 150 微秒，而 10 Gbit/s 连接的 ping 时间约为 90 微秒。

当未启用 FT 时，主 VM 可以实现接近 (940 Mbit/s) 的传输和接收线路速率 (940 Mbit/s)。当为接收工作负载启用 FT 时，日志记录带宽非常大，因为所有传入网络数据包都必须在日志记录通道上发送。因此，日志记录通道可能成为瓶颈，如 1 Gbit/s 日志记录网络的结果所示。对于 10 Gbit/s 日志网络来说，影响要小得多。当为传输工作负载启用 FT 时，不会记录传输数据包的数据，但仍必须记录网络中断。记录带宽要低得多，因此可实现的网络传输带宽高于网络接收带宽。总的来说，我们看到 FT 可以在非常高的传输和接收速率下显着限制网络带宽，

六、相关工作

Bressoud 和 Schneider [3] 描述了通过完全包含在管理程序级别的软件实现虚拟机容错的最初想法。他们展示了通过配备 HP PA-RISC 处理器的服务器原型保持备份虚拟机与主虚拟机同步的可行性。然而，由于PA-RISC架构的限制，他们无法实现完全安全、隔离的虚拟机。此外，他们没有实施任何故障检测方法，也没有尝试解决第 3 节中描述的任何实际问题。更重要的是，他们对 FT 协议施加了许多不必要的限制。首先，他们强加了纪元的概念，其中异步事件被延迟直到设定的时间间隔结束。纪元的概念是不必要的——他们可能强加了它，因为他们无法足够有效地重播单个异步事件。其次，他们要求主虚拟机基本上停止执行，直到备份收到并确认所有先前的日志条目为止。然而，只有输出本身（例如网络数据包）必须被延迟——主虚拟机本身可以继续执行。

Bressoud [4] 描述了一种在操作系统 (Unixware) 中实现容错的系统，因此为在该操作系统上运行的所有应用程序提供容错。系统调用接口成为必须确定性复制的操作集。这项工作与基于管理程序的工作具有类似的限制和设计选择。

纳珀等人。[9] 以及 Friedman 和 Kama [7] 描述了容错 Java 虚拟机的实现。他们在发送信息时遵循与我们类似的设计日志通道上的输入和非确定性操作。与布雷苏德一样，他们似乎并不专注于检测故障并在故障后重新建立容错能力。此外，它们的实现仅限于为在 Java 虚拟机中运行的应用程序提供容错能力。这些系统尝试处理多线程 Java 应用程序的问题，但要求所有数据都受到锁的正确保护，或者在访问共享内存时强制执行序列化。

邓拉普等人。[6]描述了一种确定性重放的实现，其目标是在半虚拟化系统上调试应用程序软件。我们的工作支持在虚拟机内运行的任意操作系统，并为这些虚拟机实现容错支持，这需要更高级别的稳定性和性能。

库利等人。[5] 描述了支持容错虚拟机的替代方法及其在名为 Remus 的项目中的实现。通过这种方法，主虚拟机的状态在执行期间被重复设置检查点，并将其发送到备份服务器，备份服务器收集检查点信息。检查点必须非常频繁地执行（每秒多次），因为外部输出必须延迟，直到发送并确认下一个检查点。这种方法的优点是它同样适用于单处理器和多处理器虚拟机。主要问题是，这种方法需要非常高的网络带宽才能在每个检查点将增量更改发送到内存状态。[5] 中提供的 Remus 结果显示内核编译和 SPECweb 基准测试速度降低了 100% 到 225%，当尝试使用 1 Gbit/s 网络连接每秒执行 40 个检查点来传输内存状态的更改时。有许多优化可能有助于降低所需的网络带宽，但尚不清楚 1 Gbit/s 连接能否实现合理的性能。相比之下，我们基于确定性重放的方法可以实现低于 10% 的开销，并且多个实际应用程序的主主机和备份主机之间所需的带宽低于 20 Mbit/s。

7 结论和未来工作

我们在 VMware vSphere 中设计并实施了一个高效、完整的系统，为集群中服务器上运行的虚拟机提供容错 (FT)。我们的设计基于使用 VMware 确定性重放通过另一台主机上的备份虚拟机复制主虚拟机的执行。如果运行主虚拟机的服务器出现故障，备份虚拟机会立即接管，不会造成中断或数据丢失。

总体而言，VMware FT 下的容错虚拟机在商用硬件上的性能非常出色，并且对于某些典型应用程序的开销不到 10%。VMware FT 的大部分性能成本来自使用 VMware 确定性重放来保持主虚拟机和备份虚拟机同步的开销。因此，VMware FT 的低开销源自 VMware 确定性重放的效率。此外，保持主数据库和备份数据库同步所需的日志记录带宽通常非常小，通常小于 20 Mbit/s。由于大多数情况下日志记录带宽相当小，因此实现主虚拟机和备份虚拟机相距较长距离（1-100 公里）的配置似乎是可行的。因此，VMware FT 可以用于以下场景：

还可以防止整个站点发生故障的灾难。值得注意的是，日志流通常是相当可压缩的，简单的压缩技术可以显着降低日志记录带宽，同时增加少量的额外 CPU 开销。

我们对 VMware FT 的结果表明，可以在确定性重放的基础上构建容错虚拟机的高效实施。这样的系统可以以最小的开销透明地为运行任何操作系统和应用程序的虚拟机提供容错能力。然而，要使容错虚拟机系统对客户有用，它还必须健壮、易于使用且高度自动化。除了虚拟机的复制执行之外，一个可用的系统还需要许多其他组件。特别是，VMware FT 通过在本地集群中查找合适的服务器并在该服务器上创建新的备份虚拟机，在发生故障后自动恢复冗余。通过解决所有必要的问题，我们展示了一个可用于客户数据中心实际应用程序的系统。

通过确定性重放实现容错的权衡之一是，目前确定性重放仅针对单处理器虚拟机有效实现。然而，单处理器虚拟机对于各种工作负载来说绰绰有余，特别是因为物理处理器不断变得更强大。此外，许多工作负载可以通过使用许多单处理器虚拟机来横向扩展，而不是通过使用一个更大的多处理器虚拟机来纵向扩展。多处理器虚拟机的高性能重放是一个活跃的研究领域，并且可以通过微处理器中的一些额外硬件支持来实现。一个有趣的方向可能是扩展事务内存模型以促进多处理器重放。

未来，我们还有兴趣扩展我们的系统以处理部分硬件故障。部分硬件故障是指服务器中部分功能或冗余丢失，但不会导致数据损坏或丢失。一个例子是与虚拟机的所有网络连接丢失，或者物理服务器中的冗余电源丢失。如果运行主虚拟机的服务器上发生部分硬件故障，在许多情况下（但不是全部），立即故障转移到备份虚拟机会更有利。此类故障转移可以立即恢复关键虚拟机的全部服务，并确保虚拟机快速从可能不可靠的服务器上移走。

致谢

我们要感谢 Krishna Raja，他创造了许多性能结果。有很多人参与了 VMware FT 的实施。确定性重放（包括对各种虚拟设备的支持）和基本 FT 功能的核心实现者包括 Lan Huang、Eric Lowe、Slava Malyugin、Alex Mirgorodskiy、Kaustubh Patil、Boris Weissman、Petr Vandrovec 和 Min Xu。此外，还有许多其他人员参与VMware vCenter中FT的高层管理。Karyn Ritter 出色地管理了大部分工作。

8. 参考文献

[1]阿尔斯伯格，P.，和戴，J.分布式资源弹性共享的原则。在*第二届国际软件工程会议论文集*（1976），第 627–644 页。

[2]AMD 公司。*AMD64架构程序员手册*。加利福尼亚州桑尼维尔

[3]布雷苏德，T. 和施奈德，F.基于管理程序的容错。在*SOSP 15 会议记录*（1995 年 12 月）。

[4]TC 布雷苏德TFT：用于应用程序透明容错的软件系统。在 *第二十八届容错计算国际研讨会论文集*（1998 年 6 月），第 128-137 页。

[5]Cully, B.、Lefebvre, G.、Meyer, D.、Feeley, M.、Hutchison, N. 和 Warfield, A.Remus：通过异步虚拟机复制实现高可用性。在*第五届 USENIX 网络系统设计与实现研讨会论文集*（2008 年 4 月），第 161-174 页。

[6]邓拉普 (GW)、金 (ST)、西纳尔 (S.)、巴斯莱 (Basrai)、M.和陈PMReVirt：通过虚拟机日志记录和重放启用入侵分析。在*2002年操作系统设计与实现研讨会论文集*（2002 年 12 月）。

[7]弗里德曼，R. 和卡马，A.透明的容错 Java 虚拟机。在 *可靠分布式系统论文集*（2003 年 10 月），第 319-328 页。

[8]英特尔公司。*英特尔- 右64 和 IA-32 架构软件开发人员手册*。加利福尼亚州圣克拉拉。

[9]Napper, J.、Alvisi, L. 和 Vin, H.容错 Java 虚拟机。在*国际可靠系统和网络会议论文集*（2002 年 6 月），第 425-434 页。

[10]Nelson, M.、Lim, B.-H. 和 Hutchins, G.虚拟机快速透明迁移。在 *2005 年度 USENIX 技术会议论文集*（2005 年 4 月）。

[11]南丁格尔，EB，Veeraraghavan，K.，陈，PM 和 Flinn，J.重新考虑同步。在 *2006年操作系统设计与实现研讨会论文集*（2006 年 11 月）。

[12]Schlicting, R. 和 Schneider, FB故障停止处理器：一种设计容错计算系统的方法。*ACM 计算调查 1*，3（1983 年 8 月），222–238。

[13]施耐德，FB使用状态机方法实现容错服务：教程。 *ACM 计算调查 22*，4（1990 年 12 月），299-319。

[14]斯特拉图斯技术公司。受益于 Stratus 持续处理技术：Microsoft Windows Server 环境自动实现 99.999% 的正常运行时间。位于 http://www.stratus.com/~/media/-Stratus/Files/Resources/WhitePapers/continuousprocessing-for-windows.pdf，六月 2009年。

[15]Xu, M.、Malyugin, V.、Sheldon, J.、Venkitachalam, G. 和 Weissman, B.ReTrace：通过虚拟机确定性重放收集执行跟踪。在*2007 年建模、基准测试和仿真研讨会论文集*（2007 年 6 月）。