一种高效的虚拟机快照管理系统①

邓换方[1,2], 吴 恒[1,2], 张文博[2], 许舒人[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院软件研究所 软件工程技术研究开发中心, 北京 100190)

摘 要: 链式虚拟磁盘快照技术是广泛实现的一种支持虚拟机失效恢复的技术手段, 针对链式结构磁盘快照技术引入多余 I/O 请求导致虚拟机性能低效问题, 分析和研究了支持高效虚拟机快照实现的无链式关联磁盘快照技术, 通过集成 ZFS 及 OCFS2 文件系统给出了无链式关联磁盘快照技术实现, 并设计和实现了虚拟机快照存储组织模型和快速检索算法, 提高了虚拟机快照检索效率. 设计试验验证了系统的有效性.

关键词: 虚拟化; 虚拟机; 虚拟磁盘快照; 高效; 链式关联快照; 无链式关联快照

Efficient Management System for Virtual Machine Snapshots

DENG Huan-Fang[1,2], WU Heng[1,2], ZHANG Wen-BO[2], XU Shu-Ren[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Virtual disk snapshot is a prevailing technology to support virtual machine failover. It's usually organized into linked structure, which may cause virtual machine performance degradation because this method would introduce additional disk I/O overhead. In this paper, we analyze the efficiency of disk snapshot technology with non-linked structure. By integrating ZFS and OCFS2 file system based on non-linked structure, we implement an efficient management system for virtual machine snapshots. In order to improve the efficiency of snapshot retrieval, we also propose a virtual machine snapshot storage organization model and optimize a snapshot search algorithm. The result of comparative experiment indicates our system's effectiveness.

Key words: virtualization; virtual machine; virtual disk snapshot; efficient; disk snapshot technology with linked structure; disk snapshot technology with non-linked structure

1 引言

随着虚拟化技术的快速发展和应用深入, 虚拟机逐渐替代物理机成为企业关键业务系统的运行支撑环境, 而保障虚拟机的可用性则成为衡量关键业务系统可靠性的重要度量指标之一, 因为虚拟机故障或宕机会给企业带来严重的经济和名誉损失. 如 Amazon 虚拟化平台 EC2 的虚拟机宕机事件. 2010 年 1 月, Ruby-on-Rails 应用托管商 Heroku 在 Amazon EC2 平台上的 22 台虚拟机发生故障, 影响了站点上 44000 个应用的正常使用[1]. 因此, 在虚拟化计算环境中, 支持虚拟机失效恢复逐渐成为学术界研究的热点和产业界关注的重点.

虚拟机快照技术是支持虚拟机失效恢复的一种常用技术手段, 快照本质是虚拟磁盘在某一时间点的状态和数据的副本, 这里虚拟磁盘是指虚拟机操作系统及其部署应用的逻辑抽象. 因此, 当虚拟机因内部或外部因素发生无法修复的故障时, 可通过回滚到快照时间点的方式实现虚拟机的快速恢复.

针对虚拟机占用空间大, 全拷贝策略的快照实现技术具有耗时长的不足(时间复杂度与虚拟机占用空间大小的关系是 O(n)). Citrix 的 XenServer, Microsoft 的 Hyper-v 等虚拟化产品实现了一种时间复杂度为 O(1)

① 基金项目:国家自然科学基金(61173003,61363003)
收稿时间:2014-03-01;收到修改稿时间:2014-03-31

的虚拟机快照技术, 这种技术通过共享虚拟磁盘, 不复制数据本身仅修改磁盘索引来实现快照地瞬时生成. 然而此种快照技术生成的每个磁盘快照都仅保留了上一快照时间点之后的修改数据, 快照间具有链式关联, 导致虚拟机数据读写请求由无快照状态下的一次读一次写变为多次读一次写. 这些多余的 I/O 请求会严重影响虚拟机性能, 导致虚拟机低效, 且随着快照链长度增长而愈加明显. 因此, Vmware、Xenserver 等虚拟化厂商建议虚拟机快照链的长度不超过 3 个, 每个虚拟机快照的保留时间不超过 72 个小时[2].

如何降低快照引起的虚拟机性能低效问题逐渐成为学术界的一个关注重点. LVHPsnap[3]通过构建混合快照链表和压缩快照索引实现了高频度快照生成时系统能保持较高的性能和存储空间利用率; 文献 4[4]则通过在已有虚拟化平台配置持续的快照收集和合并策略, 缩短快照链长度, 减少不必要的快照创建操作, 降低快照对虚拟机性能影响并保证虚拟机高可用性. 这些方法一定程度上降低了快照对虚拟机性能的影响, 但并未打破快照间的链式关联, 快照间的链式结构依然是影响虚拟机性能的主要因素.

针对上述问题, 本文首先采用无链式关联快照技术, 用以打破快照间链式关联, 降低快照对虚拟机性能影响; 在此基础之上, 优化虚拟机快照存储组织模型, 定义虚拟机快照命名规则, 设计和实现快照排序和查找算法, 以提高虚拟机快照检索效率; 同时, 为快照生命周期管理中的某些操作定义约束限制, 以保证系统中各对象的正常运行. 此外, 系统未来也将设计和实现虚拟机快照策略管理和快照监测分析, 以进一步降低快照对虚拟机性能影响, 帮助用户更好地管理快照.

本文的组织结构如下: 首先详细分析不同的虚拟机快照技术, 接着重点介绍虚拟机快照管理系统架构设计和模块实现, 最后给出基于不同快照技术实现的快照功能对虚拟机性能影响的对比实验结果, 以及下一步工作.

2　系统技术原理

链式关联虚拟磁盘快照技术是目前大多数虚拟化平台都广泛实现的一种虚拟机快照技术. 它提供了一种通用的快照实现, 不考虑底层文件系统的具体实现, 通过在磁盘文件之上封装, 虚拟化平台按照既定文件格式重新组织了文件, 如 XenServer 的 VHD 格式、VMware 的 VMDK 格式等. 但是由于封装, 不能感知具体哪些数据块进行了修改, 每个快照都仅保留被修改数据, 都只记录了部分磁盘数据索引, 磁盘某些数据或磁盘完整状态的获取需要读取之前一个甚至多个连续快照, 快照间具有链式关联. 这将无快照状态下数据读写请求由一次读一次写变为多次读一次写, 引入大量多余的磁盘 I/O, 会严重影响虚拟机 I/O 性能. 随着链式增长, 对虚拟机性能影响会越来越明显, 进而导致虚拟机低效.

为降低链式结构虚拟磁盘快照引起的虚拟机性能低效问题, 本文分析和研究了无链式关联虚拟磁盘快照技术. 这是一种细粒度的快照实现, 依赖于底层文件系统暴露的接口, 可感知具体数据块修改, 通过将源数据块索引替换为修改数据块索引, 未修改数据块索引不变, 可保证每个快照都记录了其对应时间点所有数据块索引, 都对应一个完整的磁盘状态; 任一数据都可通过索引直接获取, 与其他快照无关, 数据的读写请求依然是一次读一次写. 这种快照实现打破链式关联结构, 避免了多余 I/O 请求, 对虚拟机 I/O 性能影响较小. 但是快照功能实现与具体文件系统相关.

无链式关联虚拟磁盘快照技术的技术原理示意如下. 图 1 所示为虚拟机随时间所进行的磁盘数据修改. 时间点 0 为初始状态, 磁盘数据由 A、B、C、D 组成; 在时间点 0 创建磁盘第一个快照, 快照通过引用 A、B、C、D 数据块来保留时间点 0 的磁盘状态. 从时间点 0 到时间点 1, 虚拟机修改数据块 B 为 B', 删除数据块 C 以及写入新数据 E, 时间点 1 磁盘数据由 B'、D、A、E 组成; 之后再次创建磁盘快照, 快照 2 也通过引用新数据 B'、E 及旧数据 D、A 保留时间点 1 的磁盘状态. 快照 2 之后虚拟机修改数据块 D 以及写入新数据 F, 那么时间点 2 磁盘状态包含 B'、A、E、D'、F.

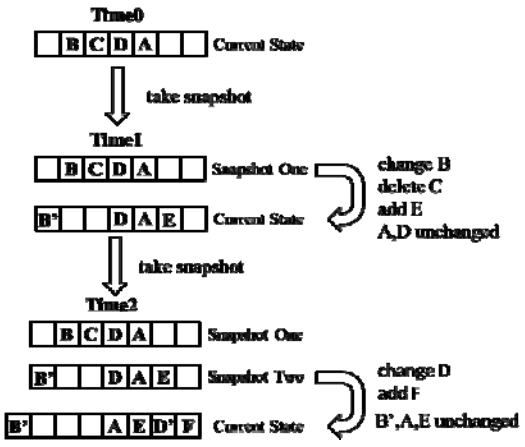

图 1　数据变化过程图

链式关联快照技术由于屏蔽了底层文件系统的异构性, 不能感知具体哪些数据块进行了修改, 每个快照中只记录了前一个快照时间点之后新写入的数据的索引, 未发生更改的旧数据索引需读取之前多个连续快照才能获取, 导致快照间具有链式关联. 如图 2 链式关联快照技术中所示, 对应于时间点 1 的快照 2 的完整状态应包括数据 A、D、B’、E, 但是由于快照 2 中只记录新写入的数据 B’、E, 读取旧数据 A、D 就需要在读取快照 2 发现未命中后读取快照 1 来获取数据 A、D. 数据请求也就由一次读一次写变为多次读一次写, 引入多余 I/O 请求. 同样地, 当前状态需要两次读请求才能获取数据 B’、E, 而三次读请求获取数据 A. 这些由于快照链式关联所带来的多余 I/O, 将会严重影响虚拟机的性能, 且性能影响会随着快照链的增长而愈加明显.

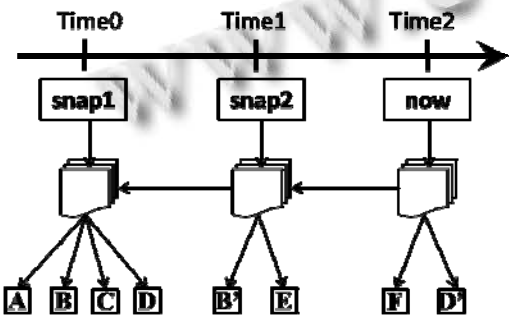

图 2 快照链式关联结构图

针对链式关联快照技术性能问题, 无链式关联快照技术通过收集数据块修改请求, 将被修改数据块索引替换为新数据索引, 不更改未修改数据块索引, 每个快照都记录了对应时间点的所有数据块索引, 都是对应时间点的完整状态. 如图 3 所示, 快照 2 将旧数据 B 索引替换为新数据 B’索引, 新增数据块 E 索引, 且数据块 A、D 索引不变; 快照 2 记录了时间点 1 所有数据索引. 无论是旧数据 A、D 还是新数据 B’、E 都仅需要一次读请求即可, 避免了链式结构造成的多余 I/O 请求, 对虚拟机 I/O 性能影响较小.

OCFS2[5,6]及 ZFS[7] 文件系统都支持无链式关联磁盘快照技术, 它们的区别是: ZFS 不支持集群, OCFS2 支持集群. 以 ZFS 文件系统为例[8-10], 系统以包含诸多叶子节点的树形结构组织磁盘数据. 当发生数据更改请求时, 系统不覆盖磁盘上的原始数据, 而是通过将新数据写入磁盘新位置并修改旧数据块父节点索引, 父节点的数据更改也同样写入新位置并引起其上节点数据更改, 持续以上过程直到到达树根节点来实现数据更新. 所生成快照为对应时间点的全快照, 通过快照根节点即可访问快照时间点任一数据. 因此快照间不存在相关关联, 不会引入多余 I/O 请求, 不影响虚拟机性能.

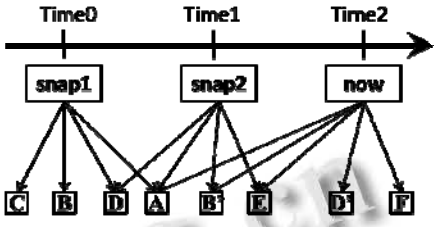

图 3 快照无链式关联结构图

3 系统设计与实现

为设计和实现一个高效的虚拟机快照管理系统, 本文集成了 ZFS、OCFS2 文件系统以实现无链式关联虚拟磁盘快照技术, 即适应单机环境也适应无链式集群环境. 下面将从系统的架构设计、关键模块设计与实现等方面来具体介绍系统的设计与实现.

3.1 系统架构设计

从系统架构设计角度, 系统总体上可分成以下三大部分: 快照基础功能模块(Basic Function)、快照 API 功能接口(Snapshot API)以及快照功能集成模块(Snapshot Integration). 系统体系架构如图 4 所示.

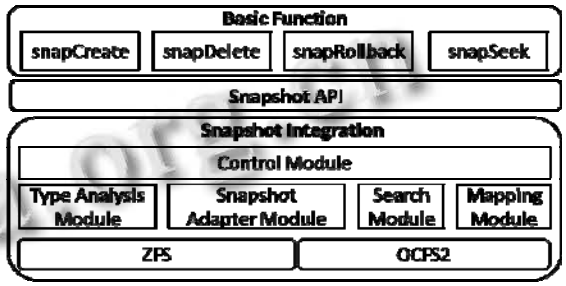

图 4 系统架构图

①快照基础功能模块(Basic Function): 依赖于快照 API 功能接口模块提供的 API 接口, 支持虚拟机基本快照操作, 如虚拟机快照创建、快照删除、快照回滚以及快照查看等.

②快照 API 功能接口(Snapshot API): 以底层各种不同的快照实现为基础, 为高层功能模块的实现提供统一交互接口.

③快照功能集成模块(Snapshot Integration): 封装 ZFS 和 OCFS2 文件系统暴露的快照命令接口, 并设计和实现内部各子模块: 控制模块、SR 类型分析模块, 快照适配模块、检索模块以及映射模块等. 通过各子

模块之间地相互协作, 支持快照各功能高效具体实现, 以及在不同快照实现间相互切换.

3.2 模块设计

为了支持系统中各快照功能的高效实现, 需设计虚拟机快照存储组织方式, 定义快照命名规则和操作约束限制, 记录存储库(SR)中存储对象间相互关系, 以及设计和实现高效的快照检索算法. 下面将详细阐述功能实现所涉及的上述设计要点.

3.2.1 存储对象映射关系

存储库(SR)中存在多种存储对象: 虚拟机(VM)、虚拟块设备(VBD)以及虚拟磁盘映像(VDI). VBD 用于在 VDI 和 VM 之间进行映射. 各存储对象都用 UUID 来唯一标识, 对象间相互关系如图 5 所示.

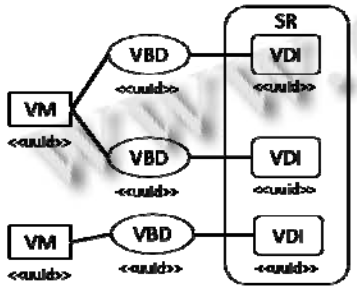

图 5　存储对象间映射关系

为支持虚拟机所在存储类型分析功能以及获取虚拟机存储路径功能, 配置文件(Config file)以<VM,VBD,VDI >和<SR,VDI>形式记录了这些存储对象间的依赖关系.

3.2.2 快照存储组织模型

虚拟机快照统一集中式存储由于快照数量众多而不易检索. 为提高快照检索效率, 系统中虚拟机快照不使用单独后备存储, 而是直接占用虚拟机所在存储库(SR)中的磁盘空间, 并以树形结构组织快照存储. 快照存储组织模型如图 6 所示.

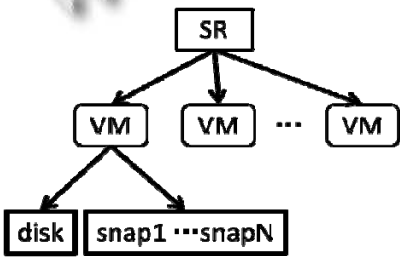

图 6　快照树形存储组织模型

每个存储库中都部署有大量虚拟机 VM, 每个虚拟机对应有虚拟磁盘 disk 和磁盘快照. 快照树形存储组织模型依据虚拟机和存储库之间相互关系, 将虚拟机所有磁盘快照都存储到其在存储库所在位置. 依赖于存储对象的依赖关系: VM->VDB->VDI->SR, 通过映射模块可以快速定位到所检索快照在存储库中位置.

3.2.3 快照命名规则和检索算法

为便于快照管理和提高快照检索速度, 为快照定义命名规则. 从系统使用者角度, 定义快照命名规则如下: 快照名只能包括字母、数字和下划线, 建议字母全部小写, 如 snapshot_1; 从系统实现角度, 为支持特殊功能实现, 系统会在用户命名的快照名加上特殊标签, 如为支持快照按时间点排序功能, 系统为所有快照打上时间标签, 规则为<Y, M, D, T, H:M:S>, 如 snapshot_1_20140219T02:52:44.

为支持快照快速检索功能, 系统事先对虚拟机下的所有快照都按照时间点排序, 排序功能由快速排序算法完成. 获取虚拟机快照时, 以二分查找算法完成虚拟机快照的快速检索, 查找操作时间复杂度由 O(n) 降为 O(logn). 快速检索功能所用算法如下所示.

算法 1. 快照检索算法

```
Input: sr,vdi of VM, snapshot’s name
Output: snapshot info
GETSNAPSHOTS(vdi)
    l ← get_filesystem_name(vdi,sr)
    S ← get_snapshot_list(l)
QUICKSORT(S,p,r)
    if p < r
      then q PARTITION(S,p,r)
           QUICKSORT(S,p,q-1)
           QUICKSORT(S,q+1,r)
PARTITION(S,p,r)
    x ← S[r]
    i ← p - 1
    for j ← p to r-1
      do if A[j]<=x
        then i ← i + 1
             swap A[i]↔A[j]
    swap A[i+1]↔A[r]
    return i+1
BINARYSEARCH(S,name)
    p ← 0, r ← S.length
    while( p <= r)
      m ← (p+r)/2
      if(eq(name.time,S[mid].time)) return m
      else if(less(name.time,S[mid].time) r ← m - 1
      else p ← m + 1
     return -1
```

3.2.4 删除操作约束限制

通过虚拟机快照可以由此虚拟机克隆(clone())出来新的虚拟机, 如图 7 所示. 由虚拟机 VM1 的快照 snap1 和 snapN 分别克隆出虚拟机 VM2 和 VM3. 由此虚拟机 VM1 与 VM2 及 VM3 之间因快照克隆建立起树形关联关系.

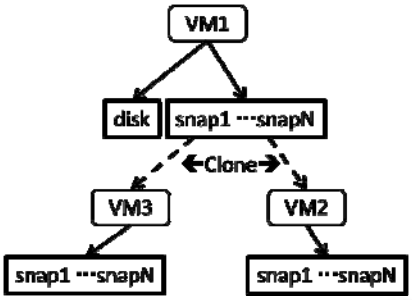

图 7 虚拟机克隆

在虚拟机生命周期管理中, 由于虚拟机之间的这种依赖关系, 删除被依赖虚拟机或快照, 将导致依赖虚拟机不可用. 因此, 需设计管理机制来合理释放虚拟机. 本文首先为被删除虚拟机打标签, 将标记值 isDeleted 标记 true, 同时采用引用计数机制, 用标记值 parent 和 children 分别记录虚拟机引用对象和引用虚拟机的对象. 只有当虚拟机标签 isDeleted 为 true 且引用计数 children 的值为 null 时, 才真正删除虚拟机对象.

3.3 模块实现

系统快照功能的实现需要各模块间的相互协作, 而快照功能集成模块是支撑系统中其他模块快照功能实现的基石, 下面将详细介绍此模块的实现. 通过封装 ZFS 及 OCFS2 文件系统对外暴露的快照功能接口以及设计实现内部各功能模块, 快照功能集成模块为高层模块操作请求适配对应的快照实现. 模块内部实现如图 5 所示. 内部各模块功能和相互间协作方式如下:

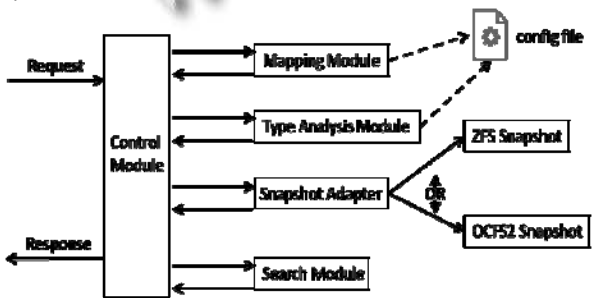

图 8 内部模块协作图

①控制模块(Control Module)接收高层模块发来的请求信息, 并依据请求信息调度其他各功能模块, 共同完整高层模块操作请求.

②类型分析模块(Type Analysis Module)接收来自控制模块的分析请求, 依据配置文件中存储对象间映射关系, 获取虚拟机所在存储库(SR), 得到存储库(SR)类型信息, 并将类型信息返回给控制模块. 存储库类型决定了虚拟机应进行 ZFS 快照还是 OCFS2 快照.

③映射模块(Mapping Module)访问配置文件, 依据存储对象间的映射关系: VM->VDB->VDI->SR, 获取虚拟机在存储库中的存储路径信息.

④适配模块(Snapshot Adapter)设计和实现了策略模式, 依据虚拟机所在存储库类型, 判断高层快照操作应执行何种快照实现; 依据虚拟机存储路径信息, 判断具体需做快照对象. 策略模式由抽象接口、封装类以及快照功能具体实现类组成.

⑤ZFS 快照模块和 OCFS2 快照模块继承统一操作接口, 分别封装了 ZFS 和 OCFS2 文件系统所暴露的快照命令接口, 支持快照功能的具体实现.

⑥检索模块(Search Module)设计和实现虚拟机快照按时间点排序, 并以二分查找算法完成具体快照高效快速检索.

4 实验验证

本文通过在虚拟化平台 XenServer6.2 和本系统所在 OnceCloud 虚拟化平台上的对比实验验证了系统无链式关联快照实现的有效性. 实验配置及实验结果如下文所述.

4.1 实验配置

对比实验的物理硬件环境配置了两台 Inspur NX580 系列的服务器. 每台服务器都配有 8 核的 Intel(R)的处理器, 16GB 的内存, 272GB 的硬盘空间以及两个千兆以太网网卡. 其中一台服务器作为计算节点, 其上运行服务器虚拟化平台; 另一台作为存储节点. 计算节点利用以太网通过 NFS 协议挂载远程存储节点. 实验用虚拟机配有 1 个 vCpu、512MB 内存以及 win server 2003 操作系统.

本文利用基准测试工具 Iometer 产生 I/O 负载, 测试不同虚拟化平台快照功能对虚拟机性能影响. 实验模拟数据库负载, 设置如下表 1 所示: 每次测试读写 4000000 个扇区, 即读写 2G 大小的 ibow.tst 文件; 读写

数据大小为 2KB, 读操作比例为 67%, 100%随机读写.

表 1　数据库负载实验条件设置

数据块大小	扇区数	读操作比例	顺序比例	随机比例
2KB	4000, 000	67%	0%	100%

4.2 实验结果

实验采用的虚拟化平台均支持多个快照同时存在. XenServer 虚拟化平台生成的快照间具有链式关联关联, 且快照的数量大小直接影响虚拟机的性能, 虚拟机开销随着快照数量增多而加大. 而 OnceCloud 虚拟化平台生成的快照间无链式关联关联, 快照个数对系统性能无影响. 本实验的关注点就是随着快照个数增多虚拟机性能变化. 虚拟机性能以其每秒平均 I/O 次数、每秒平均吞吐量以及平均响应时间来衡量. 实验结果如图 9、10、11 所示:

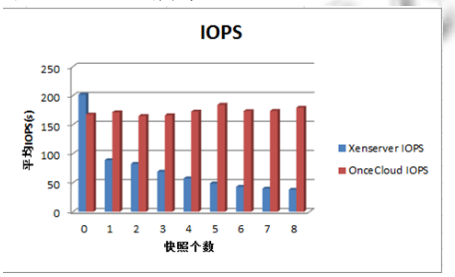

图 9　每秒平均 I/O 次数

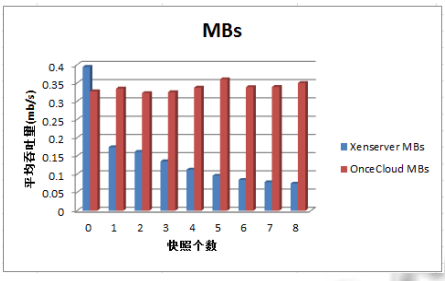

图 10　每秒平均吞吐量

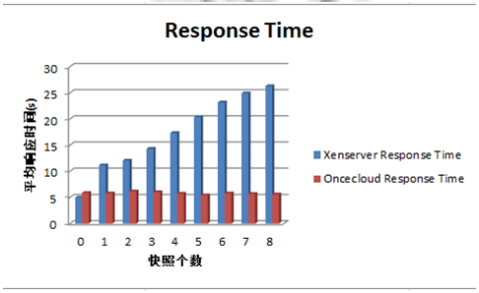

图 11　每次 I/O 平均响应时间

图 9 所示为虚拟机平均每秒的 I/O 次数, 即 IOPS, 图 10 表示虚拟机平均每秒的吞吐量(MB/s), 图 11 表示每次 I/O 操作的平均响应时间. 从实验结果可知, 随着虚拟机快照个数的增多, XenServer 虚拟化平台的链式结构快照, 由于引入多余 I/O, 读写请求延迟导致虚拟机的 IOPS 和平均吞吐量越来越低, 而平均响应时间越来越长. 而 OnceCloud 虚拟化平台的无链式关联关联的快照实现, 随着虚拟机快照个数的增多, 对虚拟机性能影响很小.

5　总结与展望

通过集成 ZFS 和 OCFS2 文件系统设计和实现了一种高效的虚拟机快照管理系统, 它采用无链式关联虚拟磁盘快照技术, 通过记录快照对应时间点的所有数据索引, 消除了链式关联快照所带来多余读写请求, 降低了快照对虚拟机性能的影响. 在此基础之上, 定义快照命名规则和操作约束限制, 设计和实现快照存储组织模型和快速检索算法, 优化了系统功能. 最后以不同虚拟化平台的对比实验验证了系统所实现的无链式关联快照在降低快照对虚拟机性能影响方面的有效性. 未来将进一步优化和完善系统功能, 设计和实现快照策略管理和快照信息监测分析, 并以实验验证系统快照策略管理在优化虚拟机性能方面的有效性.

参考文献

1 Rajagopalan S, Cully B, O'Connor R, et al. SecondSite: disaster tolerance as a service. ACM SIGPLAN Notices. ACM, 2012, 47(7): 97–108.
2 Best practices for virtual machine snapshots in the VMware environment. http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1025279.
3 周炜,谭怀亮,易乐天,等.基于带外存储虚拟化的逻辑卷高性能快照.计算机研究与发展,2012,49(3):636–645.
4 Chan H, Chieu T. An approach to high availability for cloud servers with snapshot mechanism. Proc. of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference. ACM. 2012, 6.
5 Fasheh M. OCFS2: The Oracle Clustered File System, version 2. Ottawa Linux Symposium, 2006.
6 Sudhakar GNVS. Clustered snapshots in networks: U.S. Patent 7,774,568. 2010-8-10.
7 Strobl R. ZFS: Revolution in File Systems. Sun Tech Days 2008-2009, 2008.
8 Dawidek PJ. Porting the ZFS file system to the FreeBSD operating system. Proc. of AsiaBSDCon. 2007. 97–103.
9 Ahrens. Snapshot, is it magic? http: //blogs.sun.com/ahrens/entry/is_it_magic. [2009-04-10].
10 ZFS 源代码. https://github.com/zfsonlinux/zfs.