计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(5):131−136 [doi: 10.15888/j.cnki.csa.008452]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向 Linux 非逻辑卷块设备的快照系统①

宋东平[1], 胡晓勤[1], 谢俊峰[2], 钱禹航[2]

[1](四川大学 网络空间安全学院, 成都 610207)
[2](成都云祺科技有限公司, 成都 610041)
通信作者: 胡晓勤, E-mail: huxiaoqin@scu.edu.cn

摘　要: 为满足 Linux 操作系统下非逻辑卷块设备需要在不添加额外块设备存储数据的场景下创建临时快照的需求, 设计实现了一种针对 Linux 非逻辑卷块设备的快照系统. 系统基于写时拷贝 (COW), 分为应用层和内核层中的通用块层两部分. 应用层部分对用户的快照创建或删除命令分析并传递到通用块层部分, 通用块层部分创建或删除快照设备, 在快照创建后截获快照源设备的通用块层 I/O (bio) 请求并做 COW. 实验结果表明系统能正确创建快照, 其最佳拷贝块大小为 4 MB, 对于快照源设备的新增写性能最低影响低于 10%.

关键词: 非逻辑卷; 块设备; 快照; 写时拷贝; 通用块层; Linux

引用格式: 宋东平,胡晓勤,谢俊峰,钱禹航.面向 Linux 非逻辑卷块设备的快照系统.计算机系统应用,2022,31(5):131–136. http://www.c-s-a.org.cn/1003-3254/8452.html

Snapshot System for Linux Non-logical Volume Block Devices

SONG Dong-Ping[1], HU Xiao-Qin[1], XIE Jun-Feng[2], QIAN Yu-Hang[2]

[1](School of Cyber Science and Engineering, Sichuan University, Chengdu 610207, China)
[2](Chengdu Vinchin Technology Co. Ltd., Chengdu 610041, China)

Abstract: Given that non-logical volume block devices under the Linux operating system need to create temporary snapshots without additional block devices being added to store data, this study designs and implements a snapshot system for Linux non-logical volume block devices. The system has an application layer and a generic block layer of the inner nuclear layer and is based on copy-on-write (COW). The application layer analyzes the user's creation or deletion commands and transmits them to the generic block layer. The general block layer creates or deletes the snapshot devices and intercepts the general block layer I/O (bio) request of the source device and performs COW after the snapshot is created. Experimental results show that the system can create snapshots correctly. The optimal copy block size is 4 MB. The minimum impact on the write performance of the source device is less than 10%.

Key words: non-logical volume; block device; snapshot; copy-on-write (COW); generic block layer; Linux

快照是关于指定数据集在某个时间点的完全可拷贝映像, 可以作为一种备份方法使系统中的数据得到有效保护[1−3]. 常用的快照实现方法有写时拷贝 (copy-on-write, COW) 和写重定向 (redirect-on-write, ROW). 两种快照方法均可实现在线快照并生成快照卷[4,5]. COW[6] 不改变源卷的数据分布, 在即将写入新数据时将源卷中指定区域的数据拷贝至快照卷, 因此 COW 不影响源卷的读效率, 而写效率会造成额外的两倍开销. ROW[7] 将新写入的数据存储于快照卷, 随着写入次数增多, 源卷的数据分布发生变化, 读效率下降.

现有 Linux 操作系统可对基于逻辑卷 (logical-volume) 的块设备在不添加额外块设备的场景下创建

① 基金项目: 国家自然科学基金 (61872255)
收稿时间: 2021-07-12; 修改时间: 2021-08-11; 采用时间: 2021-08-17; csa 在线出版时间: 2022-04-11

快照[8,9], 但非逻辑卷块设备缺少成熟的快照技术支撑, 定时备份缺少一致性支持. 为解决这一问题, 有学者[10,11]指出非逻辑卷块设备快照可在文件系统层或块设备层实现. 文件系统层快照对于源数据的获取以及快照数据的存取和解析有着较高的效率, 但需要针对特定的文件系统实现; 块设备层快照可分为在通用块层和物理块层实现, 二者均可屏蔽上层文件系统的差异, 较前者更具灵活性, 但物理块层快照不可屏蔽具体磁盘协议, 而通用块层快照无需考虑磁盘协议, 较物理块层快照更加灵活. 耿芳忠[12] 提出了一种基于 ROW 的 Linux 存储卷的快照方法, 但是该方法需要添加真实块设备作为快照卷存储数据. 刘志勇[13] 利用添加缓冲卷的方法改进了 COW 效率低下的问题, 但并未解决需要额外块设备的问题. 张权等[14] 提出了一种 Linux 标准分区的快照方法, 但方法仍然有上述需要添加额外块设备存储数据的问题. 开源软件 dattobd[15] 基于块层利用虚拟文件系统读写方法实现了非逻辑卷块设备的快照并将快照数据存储于快照源设备, 但是该软件只能创建一个快照, 并且创建快照后快照源设备的写效率低下, 对于实际生产的影响较大. 逻辑卷快照[16] 的实现方案同样会在快照创建后对于原始块设备的写效率影响比较大.

本文基于 COW, 结合通用块层的优势, 设计实现了一种面向 Linux 非逻辑卷块设备的快照系统, 能够在不添加额外块设备的场景下为 Linux 非逻辑卷块设备创建快照, 并且快照对于快照源设备的写速率影响较低.

1 系统设计

1.1 快照节点存储结构设计

为对不同的块设备创建多个快照, 本文设计了一种链式快照存储结构, 如图 1 所示. 建立一个 head 节点作为全局快照链的头节点, 仅用于快照链寻址. 建立 device_head 链作为主块设备链 (例如: sdx1 的主设备为 sdx), 通过访问该链上的节点实现对具体快照头结点的查找. 建立 snap_head 链作为快照链头结点链, 通过访问该链上的节点实现对快照源设备 (具体到分区, 例如: sdx1) 的确定和快照链的确定. 建立 snap 链作为快照链, 通过访问该链上的节点实现相同块设备的不同快照的确定. 该结构实现了对于不同块设备、同一块设备内部不同分区以及不同快照的记录. 快照节点信息将根据该结构进行保存.

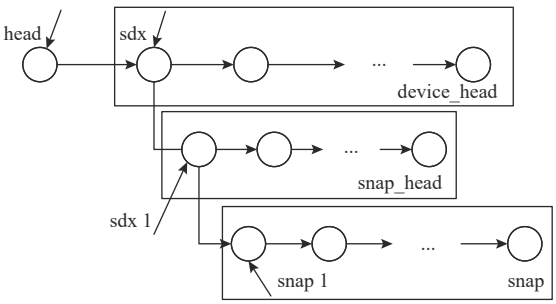

图 1 快照存储结构图

1.2 Bitmap 存储结构设计

为记录块设备的拷贝映射信息, 本文设计了一种广义 Bitmap 结构进行存储, 如图 2 所示. 一条 Bitmap 记录占 18 字节, 从左起第 1 字节作为标志位, 用于标记当前拷贝块 (一次 COW 拷贝多少数据, 自定义大小, 不同于操作系统的数据块) 有无拷贝记录, 第 2–9 字节用于记录该拷贝块在快照源设备中的逻辑扇区号即拷贝源地址, 第 10 字节为保留位, 第 11–18 字节用于记录该拷贝块拷贝后的逻辑扇区号即拷贝目的地址. 通过该 Bitmap 存储结构可以记录原始拷贝块的拷贝标记和拷贝块的拷贝前后地址映射.

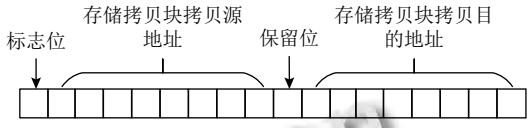

图 2 Bitmap 存储结构图

1.3 架构设计

快照系统设计为前后端模式, 分别对应应用层程序和内核层程序, 内核层程序主要位于通用块层. 前端程序主要处理用户命令以及文件操作, 后端程序主要执行快照底层逻辑, 系统模块组成及其关系如图 3 所示. 快照处理模块由命令接口和快照管理两部分构成. 命令接口属于前端程序, 用于接收用户的快照创建或删除命令, 然后依照该命令对快照文件进行相应的操作 (快照文件由位图文件.bit 和数据文件.data 组成), 完成后将该命令发送到快照管理. 快照管理属于后端程序, 用于接收并解析命令接口发送的命令, 然后依照该命令进行快照设备及过滤器的创建或删除, 快照设备为内存块设备, 其数据存储于快照源设备上. 过滤器、COW 模块以及快照设备的读模块属于后端程序. 过滤器在创建后对快照源设备进行通用块层的 I/O 请求 (block input outpu, bio) 截获, 并将截获到的写 bio 转发

到 COW 模块, 将读 bio 转发回快照源设备. COW 模块根据接收到的 bio 进行数据拷贝操作, 然后将该 bio 转发回快照源设备以完成 COW. 快照设备的读模块将发送到快照设备的读 bio 重定向到快照源设备以完成快照设备的读请求处理.

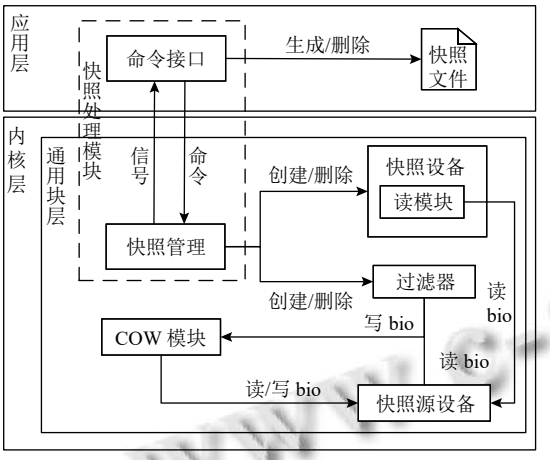

图 3 快照系统架构图

2 系统实现

2.1 快照创建与删除

快照创建与删除工作主要由快照处理模块完成. 模块包含命令接口和快照管理两部分. 命令接口用于对用户传入的命令进行解析, 快照管理用于对命令接口发送的命令进行通用块层的处理, 快照创建与删除流程如图 4 所示.

创建流程如图 4(a) 所示. 命令接口首先查询设备是否存在, 不存在则结束创建, 存在则查询快照源设备的设备大小、设备挂载点以及设备的主次设备号等有关该设备的基本信息, 随后在设备挂载点下创建一组空洞文件作为快照文件并获取该组文件在块设备上的逻辑扇区分布 (其中.bit 文件用于存储 Bitmap 数据, .data 文件用于存储 COW 过程中拷贝的数据), 目的是为了在通用块层进行数据拷贝时从通用块层存取 Bitmap 记录以及定位数据的存储位置. 然后注册信号处理回调, 回调用于处理快照管理发送的创建信号以及删除信号. 随后发送创建命令到快照管理. 快照管理在接收到创建快照命令后依次创建一个快照设备和一个快照源设备 bio 过滤器 (一个快照源设备只对应一个过滤器, 如果存在则不创建), 随后将其添加到全局快照链中, 然后将块设备大小按照拷贝块划分, 将各个拷贝块的起始扇区号作为原始数据所在扇区号按照 Bitmap 存储结构存储于.bit 文件所对应的逻辑扇区上, 将数据拷贝后所在扇区号置 0, 随后发送创建结束信号到命令接口触发该信号的处理回调, 创建快照过程结束.

删除流程如图 4(b) 所示. 命令接口首先查询该快照是否存在, 不存在则结束删除, 存在则发送删除命令到快照管理. 快照管理接收到删除命令后将过滤器和快照设备删除 (如果该快照源设备还存在快照, 则不删除过滤器), 随后将其从全局快照链中移除, 然后发送删除结束信号到命令接口触发该信号的处理回调, 删除快照过程结束.

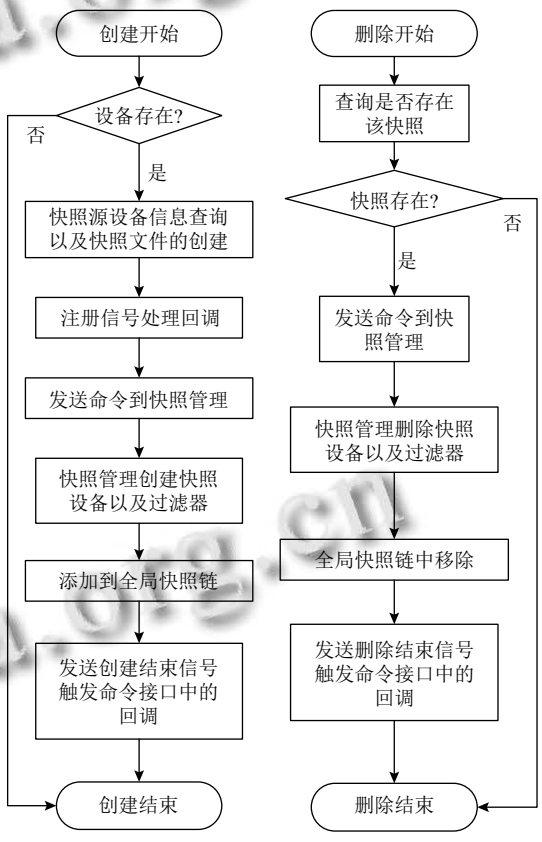

图 4 快照创建与删除流程图

2.2 快照写

快照写工作主要由过滤器和 COW 模块完成, 其中过滤器用于快照源设备 bio 的截获, 然后对其按读写类型进行分类, COW 模块主要进行数据拷贝操作. 流程如图 5 所示.

过滤器的创建方法为首先获取快照源设备的设备对象, 根据设备对象获取通用块层入口函数指针, 使用

自定义的函数对其进行替换, 至此过滤器创建成功. 在过滤流程中, 对于读 bio, 处理方式为转发到快照源设备; 对于写 bio, 处理方式为转发到 COW 模块. 过滤流程如图 5(a) 所示.

COW 模块在接收到 bio 后从 Bitmap 中查询该 bio 所指向的数据块所在的拷贝块有无拷贝记录. 对于有拷贝记录的拷贝块, 处理方法为转发该 bio 到快照源设备; 对于没有拷贝记录的拷贝块, 处理方法为先进行逻辑上的拷贝再进行物理上的拷贝. 逻辑上的拷贝首先根据截获到的 bio 和已拷贝的数据块大小计算出拷贝块的拷贝源地址和拷贝目的地址, 然后对 Bitmap 进行更新. 物理上的拷贝首先对该拷贝块进行通用块层的数据读取, 然后将数据在通用块层写入拷贝目的地址中. 通用块层的数据读写方法即为构造 bio 进行数据读写. 拷贝完成后再将该 bio 转发回快照源设备. COW 流程如图 5(b) 所示. 其中, 拷贝源地址可通过式 (1) 计算得出:

$$S_{org} = \lfloor S_{bio}/B_{size} \rfloor \times B_{size} \tag{1}$$

其中, $\lfloor S_{bio}/B_{size} \rfloor$为对$S_{bio}/B_{size}$向下取整, S_{bio}为 bio 指向的数据块的块设备地址, B_{size}为自定义的拷贝块大小. 拷贝目的地址可通过式 (2) 计算得出:

$$S_{dst} = C_{done} \times B_{size} + S_{start} \tag{2}$$

其中, C_{done}为已经拷贝的拷贝块的个数, B_{size}为自定义的拷贝块大小, S_{start}为数据文件的起始地址.

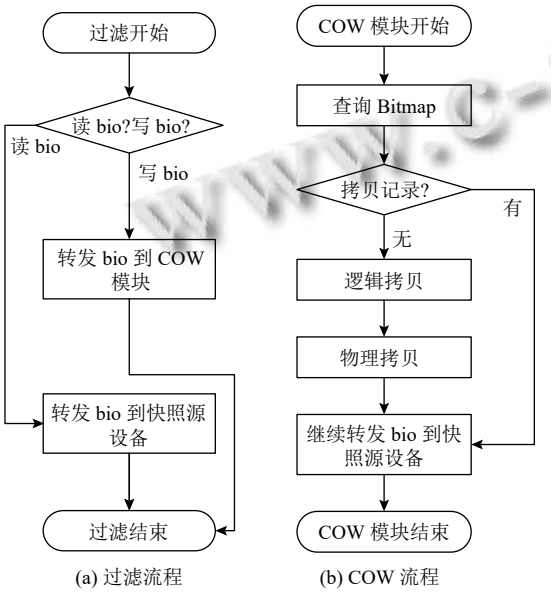

图 5 快照写流程图

2.3 快照读

快照读工作主要由快照设备中的读模块完成, 读流程如图 6. 模块在接收到读 bio 后从中获取该 bio 指向的数据块在块设备中的偏移, 依据偏移查询 Bitmap 以获得该数据块所在拷贝块的拷贝记录. 对于没有拷贝记录的拷贝块, 处理方法为重定向该 bio 到快照源设备. 对于有拷贝记录的拷贝块, 处理方法为先从 Bitmap 中获取该拷贝块的拷贝目的地址, 然后将 bio 重定向到该拷贝目的地址, 因此读模块获取到的数据即快照之前的数据.

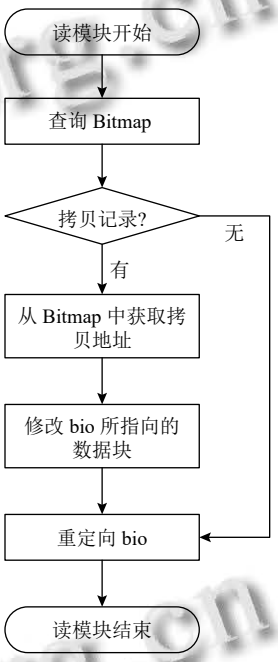

图 6 快照读流程图

3 实验分析

由于 COW 对快照源设备的读速率没有影响, 并且已经进行了 COW 的数据块在覆盖写过程中不会产生性能损耗, 因此本文实验仅依据快照源设备的新增写速率损耗便可衡量系统性能, 损耗量与系统性能成反比. 其中新增写为每创建一个快照就向快照源设备写一个指定大小的全新文件. 本文设计了 3 个实验, 分别为快照数据的正确性验证实验、拷贝块大小对系统性能的影响测试实验以及快照个数对比实验. 所有实验均在虚拟化环境中进行, 采用直接 I/O 的方式将数据写入块设备上的文件, 单次 I/O 块文件大小为 4 KB, 具体实验环境如表 1 所示. 其中块设备 1 名为/dev/sdb1, 为非逻辑卷块设备, 挂载点为/sdb1. 块设备 2 名为/dev/sdc1, 为逻辑卷块设备, 逻辑卷为/dev/vg_1/lv_test.

表 1 实验环境配置表

类型	名称	配置
宿主机	处理器	Intel Core i5-5200U 2.20 GHz
	操作系统	Windows 10
	硬盘	SATA SSD 480 GB
	硬盘缓存	512 MB
	硬盘读速率	500 MB/s
	硬盘写速率	450 MB/s
	内存大小	8 GB
虚拟机	虚拟化环境	VMWare workstation 15
	操作系统	CentOS-7.9
	内核版本	Linux-3.10.0
	内存大小	4 GB
	块设备1大小	120 GB
	块设备2大小	120 GB

3.1 正确性

本实验在对块设备/dev/sdb1 创建快照前在其挂载点/sdb1 目录下创建一个 1 GB 的文件 test_1G, 计算该文件的 MD5. 在对/dev/sdb1 创建快照后修改 test_1G 文件, 随后将生成的快照设备/dev/sdb1-snapshot1 挂载于/snap 目录下, 计算/snap 目录下的 test_1G 文件的 MD5. MD5 记录如表 2 所示.

表 2 文件 MD5 表

文件名	MD5值
/sdb1/test_1G	b2e0c41ec875aec1ec538935590c3c7b
/snap/test_1G	b2e0c41ec875aec1ec538935590c3c7b

表 2 表明, 对/dev/sdb1 创建快照后快照设备/dev/sdb1-snapshot1 中的 test_1G 文件的 MD5 值与快照前快照源设备中的 test_1G 文件的 MD5 值一致, 这说明快照设备中的文件内容与快照源设备快照前的文件内容一致, 进而证明了本系统创建的非逻辑卷块设备快照能够确保数据的正确性.

3.2 拷贝块大小对系统性能的影响

本实验选用了 1 MB、2 MB、4 MB 和 8 MB 大小的拷贝块进行快照源设备的写速率比较进而衡量系统性能, 针对每一种拷贝块大小, 均在拷贝后向快照源设备写入一个 1 GB 的全新文件, 测试其平均写速率. 平均写速率与拷贝块大小的关系如图 7 所示.

图 7 表明当拷贝块为 4 MB 大小时快照源设备写速率最高, 同等条件下其速率损耗最小, 而随着拷贝块增大或减小都会使速率损耗增大. 这是因为拷贝块粒度过细会导致 COW 操作过于频繁, 从而引起真实 I/O 操作过多耗时, 拷贝块粒度过粗会导致单次 COW 过程中读写数据量过大而引起耗时, 二者都会成为影响快照源设备写速率的因素. 因此本系统 4 MB 拷贝块大小下对于快照源设备新增写速率影响最小, 性能表现最佳.

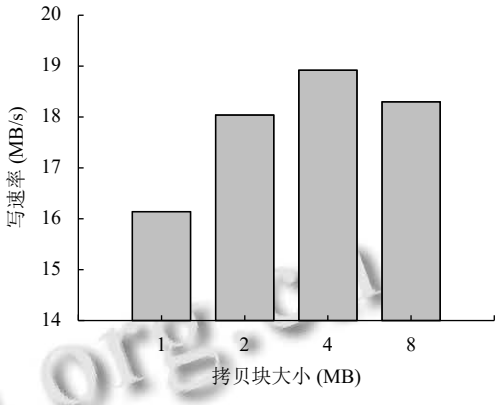

图 7 拷贝块大小对写速率的影响

3.3 快照个数对比

本实验分为单快照子实验和多快照子实验. 在单快照子实验中将本系统与其他同类系统均创建 1 个快照进行新增写速率损耗对比. 在多快照实验中利用本系统创建多个快照, 然后进行速率损耗自我对比. 实验均采用 4 MB 作为本系统的拷贝块大小.

3.3.1 单快照

本实验对块设备/dev/sdb1 分别利用本系统和开源软件 dattobd 创建单个快照, 然后向其中新增写 1 GB 文件, 对逻辑卷/dev/vg_1/lv_test 利用逻辑卷的快照方法创建单个快照, 然后同样向其中新增写 1 GB 文件, 记录三者所创建快照存在的情况下快照源设备的新增写速率并计算出速率损耗, 实验结果如表 3 所示.

表 3 不同系统下快照对新增写速率的影响

系统	0个快照写速率 (MB/s)	单个快照写速率 (MB/s)	速率损耗 (%)
本系统	20.25	18.86	6.68
dattobd	20.25	12.79	36.83
逻辑卷	19.79	8.73	55.89

表 3 表明, 在只创建一个快照的情况下, 本系统对于快照源设备的新增写速率损耗低于 7%, 开源软件 dattobd 的速率损耗高于 35%, 逻辑卷的速率损耗高于 55%. 这是因为本系统在通用块层进行 COW, 而 dattobd 和逻辑卷分别在虚拟文件系统层和逻辑卷管理层进行 COW, 二者在 Linux 内核中的层次均高于通用块层, 因此通用块层 I/O 落盘速率远高于上层, 因此本系统对于快照源设备的新增写速率影响较已经存在的两种系统有明显下降, 本系统性能优于已有的两种快照方案.

3.3.2 多快照

本实验对块设备/dev/sdb1 创建 9 个快照并进行新增写, 新增写文件大小为 1 GB, 记录不同快照个数下快照源设备的新增写速率及其损耗百分比, 实验结果如表 4 所示.

表 4 新增写速率及损耗

快照数 (个)	块设备新增写 (MB/s)	块设备新增写损耗 (%)
0	20.25	0
1	18.86	6.68
2	17.71	12.54
3	15.39	24
4	14.04	30.67
5	11.03	45.53
6	9.43	53.43
7	7.23	64.3
8	6.9	69.53
9	6.09	69.93

表 4 表明, 随着快照个数增多, 快照源设备新增写速率降低, 这是由于快照个数增多, COW 次数增多导致的, 属于正常衰减. 当快照个数为 1 时, 新增写速率损耗低于 7%; 当快照个数为 4 时, 新增写速率损耗低于 31%; 当快照个数为 9 时, 新增写速率损耗接近 70%. 因此在快照个数为 1 时, 本系统性能表现最优, 快照个数为 5 以下时性能表现良好.

4 结束语

本文设计实现了一种面向 Linux 非逻辑卷块设备的快照系统. 系统依赖于 COW 技术, 结合通用块层的优势, 实现了在不添加额外块设备的场景下对 Linux 非逻辑卷块设备创建快照, 并且较已有的快照方案有明显的新增写速率损耗降低, 满足了 Linux 非逻辑卷块设备定时备份过程中对于快照的需求. 系统性能在拷贝块为 4 MB 大小, 快照个数为 1 时最优, 快照个数少于 5 时表现良好. 但本系统存在当快照个数过多时性能变差, 对于快照源设备的新增写性能损耗偏大问题, 后续可对系统结构和数据存储方案进行优化.

参考文献

1 Aldossari K. Efficient data protection for enterprise data centers. International Journal of Computer Science & Information Technology, 2015, 7(1): 119–129.

2 Chuang PJ, Huang YC. Efficient snapshot mechanisms for Xen virtual machines. 2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA). Kanazawa: IEEE, 2017. 112–115.

3 Raju BKSP, Geethakumari G. SNAPS: Towards building snapshot based provenance system for virtual machines in the cloud environment. Computers & Security, 2019, 86: 92–111.

4 Li JX, Zhang YY, Zheng JS, *et al*. Towards an efficient snapshot approach for virtual machines in clouds. Information Sciences, 2017, 379: 3–22.

5 Qian ZW, Zhang XD, Ju XM, *et al*. An online data deduplication approach for virtual machine clusters. 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation. Guangzhou: IEEE, 2018. 2057–2062.

6 Yu X, Tan YA, Zhang CY, *et al*. A high-performance hierarchical snapshot scheme for hybrid storage systems. Chinese Journal of Electronics, 2018, 27(1): 76–85. [doi: 10.1049/cje.2017.10.008]

7 Taha H, Aknin N, El Kadiri KE. A novel model of data storage service in the architecture cloud storage. International Journal of Online and Biomedical Engineering, 2019, 15(7): 66–83.

8 Xu J, Zhang L, Memaripour A, *et al*. NOVA-fortis: A fault-tolerant non-volatile main memory file system. Proceedings of the 26th Symposium on Operating Systems Principles. Shanghai: ACM, 2017. 478–496.

9 Son Y, Choi J, Jeon J, *et al*. SSD-assisted backup and recovery for database systems. Proceedings of the IEEE 33rd International Conference on Data Engineering (ICDE). San Diego: IEEE, 2017. 285–296.

10 Zhang JP, Li HM. Research and implementation of a data backup and recovery system for important business areas. Proceedings of the 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). Hangzhou: IEEE, 2017. 432–437.

11 Liu J, Chen SY. HVF: An optimized data backup and recovery system for hard disk based consumer electronics. Optik, 2015, 126(2): 251–257.

12 耿芳忠. 一种基于 Linux 内核的存储卷的快照方法及装置: 中国, 107491363A. 2017-12-19.

13 刘志勇. 一种快照实现方法及快照系统: 中国, 105260264A. 2016-01-20.

14 张权, 胡晓勤. 一种基于 Linux 标准分区的快照方法. 现代计算机, 2017, (7): 29–33. [doi: 10.3969/j.issn.1007-1423.2017.07.008]

15 Datto Inc. Dattobd. https://github.com/datto/dattobd. (2021-04-26).

16 Nayak P, Ricci R. Detailed study on linux logical volume manager. Technical Report, Salt Lake City: Flux Research Group University of Utah, 2013.