2012 IEEE 18th International Conference on Parallel and Distributed Systems

iROW: An Efficient Live Snapshot System for Virtual Machine Disk

Jianxin Li[12], Hanqing Liu[2], Lei Cui[2], Bo Li[12], Tianyu Wo[12]

1 State Key Lab of Software Development Environment
Beihang University
Beijing, China
{lijx, libo,woty}@buaa.edu.cn

2 School of Computer Science and Engineering
Beihang University
Beijing, China
{liuhq,cuilei}@act.buaa.edu.cn

***Abstract*—The high-availiablity of mission-critical data and services hosted in a virtual machine (VM) is one of the top concerns in a cloud computing environment. The live disk snapshot is an emerging technology to save the whole state and the data of a VM at a specific point of time, and be used for quick disaster recovery. However, the existing VM disk snapshot systems suffer from long operation time and I/O performance degradation problems during snapshots creating and managing, and thereby affecting the performance of the VM and its services. To address such issues, we designed an efficient VM disk snapshot system, named `iROW` (improved Redirect-on-Write). In `iROW`, a bitmap based light-weight index scheme is adopted to replace the existing multi-level index tree structure to reduce query cost. Additionally, through a combination of Redirect-on-Write (ROW) and Copy-on-Demand (COD) schema to avoid extra copy operation on the first write after snapshot with Copy-on-Write (COW) schema, and the file fragmentation problem caused by ROW snapshot after long-term using. Finally, `iROW` gives a unified disk space allocation function by the host machine's file system. We have implemented `iROW` in qemu-kvm 0.12.5 and conducted some experiments. The implementation of `iROW` completely obey the interfaces of the block device driver in QEMU, so it is transparent to the upper system or applications and original disk image formats can be also supported. The experimental results show that `iROW` has obvious performance advantages in snapshot creating and management operations. Compared with the existing `qcow2` disk image in KVM, when the VM disk size is 50GB, and the cluster size is 64KB (the default cluster size of `qcow2`), the snapshot creation and rollback time is only about 6% and 3% of original `qcow2`'s. With the increasing of the VM disk size, `iROW` has more performance advantages on snapshot creation and rollback operations. In addition, the I/O performance of `iROW` is better than `qcow2`. When the cluster size is 64 KB, typically the `iROW`'s performance loss is 10% less than `qcow2`'s, and its first write performance after snapshot creation is about 250% of `qcow2`'s.**

Keywords-Cloud Computing*; *Reliability*; *Virtual Machine; Snapshot*; *Virtual Disk Image

I. INTRODUCTION

With the development of computer technology and information technology, users and businesses are increasingly dependent on digital information. The requirements of the data storage's reliability are also increased. Especially in special industries like banking, electricity, medical care and others, the loss of stored data would cause significant loss, or results in serious accidents. With the increased complexity of the computer system, its design inevitably has some defects. In some special conditions, these defects will cause the system failure, and result in the loss of important data. Besides, the increased complexity of the computer system puts forward higher requirements on the users. The users' misoperation may also cause the loss of data. Finally, the Internet is a completely open environment. The computer system connected to it may crash due to the malicious attacks etc. Therefore effective measures are needed to ensure the reliability of data in the computer system.

Usually, the reliability of data relies on backing up multiple copies of the data. When the current data are lost or corrupted, they can be restored to the pervious copies. Disk snapshots technology is an important way of data backup and recovery. When they perform the snapshot operations, most of the snapshot technologies do not have data copy operations. Only the metadata is changed during the snapshot operations. So snapshot operation can be completed in a short time.

Virtualization is an essential technology that separates computing environment from physical hardware, to support the delivery of computing and storage capacity as a service in a cloud computing paradigm. Virtual machine can pack both the VM's operation system and its fully configured applications together in a VM disk image. There are several features and benefits: ① Only a disk file is used to preserve the whole state, the disk data, and the configuration of a virtual machine at a specific point in time; ② A VM disk snapshot can used as a whole to back up and recover; ③ The VM disk snapshot mechanism is independent of VM's operating system and the internal file system. Therefore, a unified snapshot and recovery mechanism can be used on different systems. Moreover, a live snapshot method can, not only save the VM disk data, but also the running states and data of the software inside the VM. So that the VM can be reverted to a snapshot moment, and continue to run. This greatly improves the reliability and availability of the services provided by the VM.

The VM disk snapshot function is integrated in most of the existing system virtualization solutions. However, the existing VM disk snapshot mechanisms use high-cost multi-level tree index structure as their metadata, and use single snapshot method, and implement disk space allocation function which overlaps with the similar function of the host

1521-9097/12 $26.00 © 2012 IEEE
DOI 10.1109/ICPADS.2012.59

machine's file system. Therefore, the creating and managing of a VM snapshot will take a long time, which is not suitable for continuous cloud service in a virtual machine. Besides, these issues also have adverse effect on the normal I/O performance of the VM disk.

The qcow2 image format is one of the disk image formats supported by the QEMU processor emulator, which used in KVM [1], and it uses a two-level tree index table to store its metadata, and some extra I/O operations will be make when the first write after snapshot creation. Therefore, when the VM disk size is large, performance of qcow2's snapshot key operations will be dramatically reduced. It has great performance loss on the first writing operation after snapshot creation. As our motivating research goals, two experiments results are showed here. Figure 1 shows some results found during our experiments, where the time of snapshot creation grows dramatically with the increasing of VM disk size. When the VM disk size is 50GB, the qcow2's snapshot creation time is over 3 seconds, which seriously impacts on the live VM snapshot feature. Figure 2 shows the performance loss of qcow2 compared with the raw format. The raw is another VM disk format that KVM supports. It is actually a "plain" format. The VM disk image addresses of the data clusters are identical with the physical addresses of these clusters in non-virtualized environment. The raw format does not have any metadata, so its I/O efficiency is high. However, the raw format does not support snapshot function. In Figure 2 "qcow2" and "qcow2-s" denotes the performance of qcow2 before and after snapshot creation respectively. According to the different VM disk cluster size, the performance loss of qcow2 before snapshot is about 15%~45%, and the performance loss after snapshot is up to 60%~90%. The performance loss is very obvious.

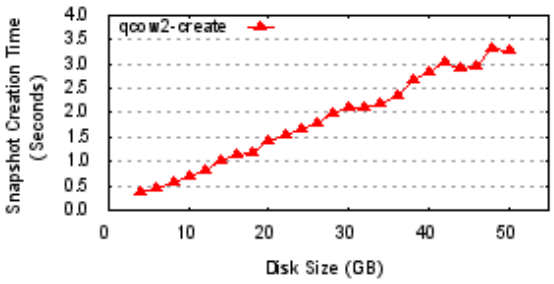

Figure 1 The snapshot creation time of qcow2

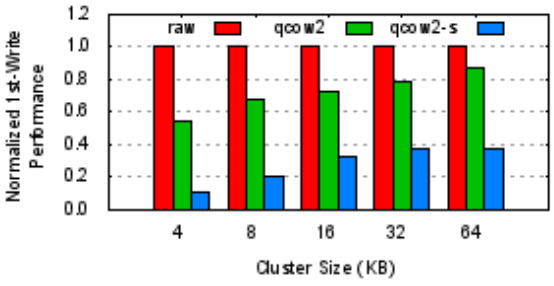

Figure 2 The performance loss of qcow2

In order to solve the problems of existing VM disk snapshot systems, we design a novel high-efficiency VM disk snapshot system: iROW. We implement a prototype system, based on the virtual block device driver framework of KVM. The main contributions of iROW are as follows:

- On the design of snapshot metadata, iROW uses bitmap to replace multilevel tree index structure which is commonly used in existing solutions. This replacement greatly reduces the amount of metadata, so that both the VM disk snapshot key operations performance and the VM disk I/O performance would be enhanced at the same time.
- On the selection of snapshot methods, iROW combines Redirect-on-Write (ROW) with Copy-on-Demand (COD). This method not only avoids the extra I/O operations of Copy-on-Write (COW) snapshot method on the first write after snapshot, but also alleviates the file fragmentation problem cause by ROW snapshot method after long-term using.
- On the setting of features, in order to avoid duplicate work, iROW does not decide the position of the data cluster in the VM disk image, and the disk space allocation function still depends on the host machine's file system. Because of supporting of sparse files based on host machine's file system, iROW also achieves the gradual growth of the VM disk image size with the actual disk usage.

The efficiency of iROW mainly reflects in the following two aspects:

- The snapshot key operations of iROW are very efficient. When the VM disk size is 50GB and cluster size is 64KB, the snapshot creation, rollback time is less than 6% and 3% of qcow2's respectively. With the increase of VM disk size, iROW has more advantages in snapshot creation and rollback operation.
- The iROW's I/O performance is also better than the qcow2's. When VM disk cluster size is 64KB, the iROW's performance loss is 10% less than qcow2's typically; the iROW's first write performance after snapshot is close to 250% of the qcow2's; the iROW's worst case I/O rate is 20% higher than the worst case of qcow2.

This paper is organized as follows: In Section II, we briefly describe the background and some related work. Section III describes the design and implementation of iROW in detail. The iROW's performance evaluation results and discussion of existing problems in iROW are presented in Section IV. The last section is our conclusion and future work.

II. BACKGROUND AND RELATED WORK

This section begins with a brief introduction to the most commonly used two snapshot methods; then discusses some related work, as well as their problems.

A. *Snapshot Methods*

There are two most commonly used snapshot methods: The Copy-on-Write (COW) snapshot and the Redirect-on-Write (ROW) snapshot [2][3].

After the creation of the COW snapshot, on the first write to a block, the original data in the block are copied to another block, and then the new data are written into the original block. The data copying operation introduces extra I/O operations, thus the I/O performance of the first write operation after snapshot is decreased. This is the main problem of the COW snapshot.

In contrast, after creating a ROW snapshot, on the first write to a data block, the original data in the block are maintained intact and the write operation is performed directly on another block. The ROW snapshot avoids the extra I/O operations of the COW snapshot. However, as the new data are written to other blocks, the changed data after snapshot are discontinuous with the unchanged data, which results in the fragmentation of the data. As the system running over a long time, the fragmentation will become increasingly serious, and thus may result in the degradation of the I/O performance.

B. *Related Work and Their Problems*

The Parallax [4][5] is a virtual storage system designed for Xen [6]. It uses three-level radix tree to map virtual block address to physical block address. The radix tree is a high-cost structure. A complete radix tree will take more than 1GB storage space (1 first-level page, 512 second-level pages, and 262144 third-level pages, each page is 4KB). Although it can be cached in memory, this strategy lacks of scalability; if only part of the radix tree is cached, then extra I/O operations are needed to read radix tree from disk. The I/O performance will be decreased by the extra I/O operations.

In order to solve the problem of high-cost metadata of Parallax, SNPdisk [7] replaces radix tree with sparse tree. SNPdisk reduces the cost of the metadata storage by the merger of index nodes, which makes more metadata can be cached in memory. This methods improves the I/O performance, however, SNPdisk uses COW snapshot method, which inevitably have extra I/O operations on the first write after snapshot.

KVM is another important system virtualization solution in Linux environment. It supports multiple VM disk formats, `qcow2` (QEMU copy-on-write version 2) is the most important one. It supports snapshot, compression, encryption and others functions. The `qcow2` uses two-level index tree to map virtual block address to VM disk image address. Besides, the `qcow2` uses reference-count values to record the number of snapshots that share the same data clusters. Its architecture is shown in Figure 3.

The main problems of the `qcow2` are as follows:

- When it performs snapshot key operations, `qcow2` needs update all of the clusters' reference-count values. This decreases the efficiency of its VM disk snapshot key operations, especially when the VM disk size is large.

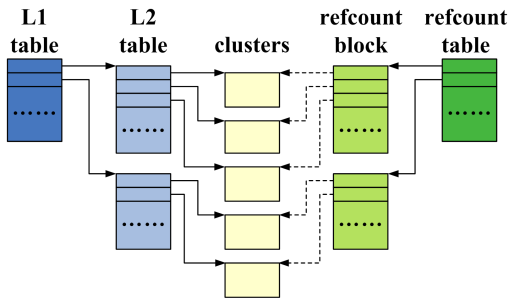

Figure 3 The architecture of `qcow2`'s metadata

- Due to the large size of its metadata, `qcow2` only caches part of the index table, which needs constantly read index table from the VM disk image. This causes the performance loss of the VM disk I/O.
- There are extra I/O operations on the first write after snapshot. Therefore, `qcow2` has great performance loss on the first writing operation after snapshot.

III. DESIGN AND IMPLEMENTATION

In order to solve the problems in existing VM disk snapshot systems, based on the KVM virtual block device driver framework, we design a high-efficiency VM disk snapshot system: `iROW`. Its position is equivalent to `qcow2` and others format in the KVM virtual block device driver framework (as shown in Figure 4). It does not have any impact on the application that uses the existing virtual block device driver.

`iROW` replaces the high-cost multi-level tree structure with the bitmap. On one hand, this replacement greatly improves the snapshot key operations performance; on the other hand, it reduces the performance loss of VM disk I/O operations. Besides, `iROW` combines ROW with COD. The combination not only avoids extra I/O operation of COW on the first write after snapshot, but also alleviates the I/O performance degradation cased by data fragmentation due to long-term using of ROW.

A. *`iROW` VM Disk Snapshot System Architecture*

The `iROW`'s architecture is shown in Figure 5. In Figure 5, the VM monitor (VMM) runs in root mode; the VM runs in non-root mode [9] [10]. When the VM has an I/O request, a `VMExit` is caused. Then the VM's I/O request is intercepted by the VMM. VMM sends this I/O request to the general virtual block device driver (Step 1 in Figure 5). General virtual block device driver sends the I/O request to the `iROW` driver (Step 2 in Figure 5). The `iROW` diver operates on the VM disk image, and obtains the return value or data (Step 3 and Step 4 in Figure 5,). Then the `iROW` driver sends the return value or data to general virtual block device driver (Step 5 in Figure 5). VMM begins to execute VM code (Step 6 in Figure 5). This is a complete I/O simulation procedure. VM does not know how KVM and QEMU simulate the I/O, it feels just like to initiate I/O requests and obtain data directly on a physical machine.

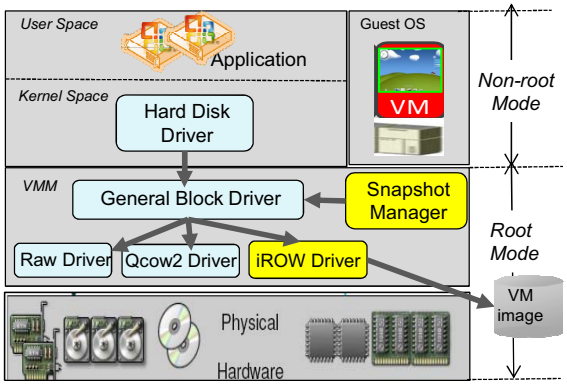

Figure 4 iROW in the KVM

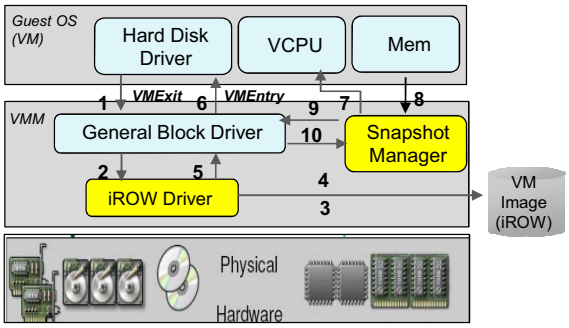

Figure 5 The workflow of iROW

When creating online snapshot, the user initiates the `savevm` command in the QEMU console. Snapshot manager first stops the virtual CPU (Figure 5, step 7). Then it collects the VM state information, and sends these data to the general virtual block device driver (Figure 5, step 8, 9). The general virtual block device driver makes the `iROW` driver save the VM state to the VM disk image (Figure 5, step 2, 3), and then returns the result to the snapshot manger (Figure 5, step 4, 5, 10). After the VM state is successfully saved, the snapshot manager initiates the `snapshot-create` command to the general virtual block device driver (Figure 5, step 9). Then the general virtual block device driver makes the `iROW` driver create a VM disk snapshot (Figure 5, step 2, 3), and sends the result to snapshot manger (Figure 5, step 4, 5, 10). After the VM disk snapshot is successfully created, the snapshot manager restores the virtual CPU (Figure 5, step 7). Then the VM resumes the work that has been stopped.

When creating offline snapshot, the user initiates the `snapshot -c` command in the `qemu-img` program. The procedure of the offline snapshot is the same as the procedure of the disk snapshot creation in online snapshot.

B. *iROW VM Disk Image Architecture*

The `iROW` VM disk image consists of a `meta` file and several `snapshots`. A `snapshot` consists of 2 files: a bitmap file (`btmp` file) and a VM disk data file (`irvd` file). The current state of the `iROW` VM disk also occupies a snapshot. Figure 6 illustrates an `iROW` VM disk image with 3 snapshots.

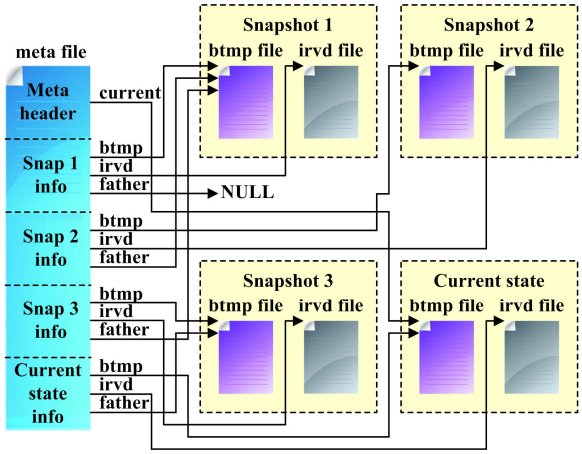

Figure 6 The structure of iROW image

The `meta` file consists of the `meta header` and the `snapshots information`. The `meta header` is used to store basic information of VM disk image. The `snapshots information` sequentially stores every snapshot's name, id and others related information. The `iROW`'s snapshot id is a 32-bits unsigned integer, so it can have up to 2^{32}-1 snapshots (The current state of the VM disk occupies a snapshot, which id is fixed to 0). For most applications, this can be seen as unlimited number of snapshots.

The `btmp` file consists of a `bitmap` and the `VM state data`. The `bitmap` is used to indicate whether the clusters exist in corresponding `irvd` file. Each cluster in the VM disk image is mapped to a bit in the `bitmap`. The `VM state data` includes the VM memory data, the virtual CPU registers data etc. The `VM state data` is generated when online snapshot is created. Offline snapshot does not have the `VM state data`.

The `irvd` file is used to store the actual data of the VM disk image. The smallest unit of storage is `cluster`. The `cluster` size can be specified by the user when creating an `iROW` disk image. `iROW` does not decide the address of the data clusters. It just writes the clusters to the same VM disk image addresses as the virtual addresses of the clusters. Because of host machine's file system support sparse files, `iROW` also achieves the gradual growth of the VM disk image size with the actual disk usage.

C. *iROW Implementation*

Each virtual block device driver in the KVM virtual block device driver framework corresponds to a `BlockDriver` structure, which contains more than 30 function-pointers and other information. These function-pointers point to the methods that are implemented by the specific virtual block driver. Later in this section, we describe part of these methods that `iROW` implemented.

1) ***Open Operation***

When it opens the VM disk, `iROW` first obtains the current state pointer form the `meta header`, and then it

loads snapshots information to the snapshot cache. The btmp file and the irvd file are opened according to the current state pointer. The bitmap of current state is cached in bitmap cache. The file descriptors of the btmp file and the irvd file are stored in the BDRVIrowState structure. In order to reduce the I/O requests and to enhance the VM disk I/O performance, the snapshots information and bitmap are completely cached in the memory.

2) ***Snapshot Create Operation***

When it creates a snapshot, iROW generates a btmp file and an irvd file as the new current state. The new current state information is added to the snapshot cache. The father pointer of the new current state points to the old current state. Then the new snapshot information is updated to the meta file. At last, the old current state is close, and the new current state is opened. The VM disk size and cluster size do not affect the file creation time, the meta updating operation time and the file opening/closing operations time. Therefore, the snapshot creation time of iROW is neither affected by the increase of the VM disk size, nor the decrease of the cluster size.

3) ***Snapshot Rollback Operation***

The iROW snapshot rollback operation is very simple, it is just needed to change the father pointer of the current state to the rollback target, and clear the current bitmap. The VM disk size and cluster size also do not affect the snapshot rollback time of iROW.

4) ***Snapshot Delete Operation***

Snapshot deletion is the most complex snapshot operation of iROW. When it deletes the snapshot, iROW needs to merge the disk data from the target snapshot to its children snapshots. After the data have been merged, corresponding files are deleted from host machine's file system. Therefore, iROW has some disadvantages in snapshot deletion. We will discuss this issue in later section.

5) ***Read Operation***

When it receives the read request from the VM, iROW first checks the bitmap and determines whether the requested data are present in the current irvd file. If the requested data are present, then the data are read from current irvd file (step 1 in Figure 7). If the requested data are not present, then the father snapshots are recursively opened according to the father pointer. The data are read from father snapshots (Figure 7, step 2).

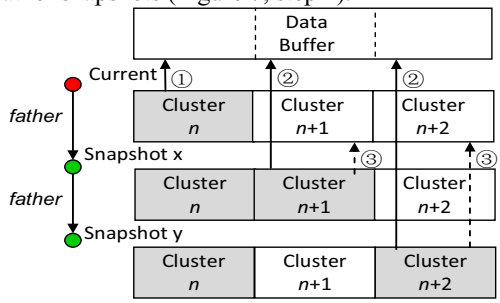

Figure 7 The reading operation of iROW

In order to alleviate the I/O performance degradation cause by file fragmentation after long-term use of ROW, iROW adds Copy-on-Demand (COD) function. If COD is enabled, the data that are read from father snapshot will be written to corresponding position in current irvd file (Figure 7, step 3). By moderate data redundancy, this method reduces the fragmentation of data, which improves the VM disk I/O performance. The state of COD is specified by the user when the VM disk is created, it also could be changed after the VM disk is created.

6) ***Write Operation***

When it receives the write request from the VM, iROW first determines whether the data are aligned with the cluster boundaries, and whether the target clusters are present (as shown in Figure 8). If the data are aligned with the cluster boundaries, or the target clusters are present, then the data can be written to the clusters directly. If the data are not aligned, and the target clusters are not present, then iROW reads these clusters from the father snapshot, and merges them with the data to be written. After the merging, iROW writes the whole clusters data to the target clusters.

IV. EVALUATION AND DISCUSSION

A. Evaluation

We conducted our experiments on a DELL Precision T1500 workstation with Intel Core i7-860 2.8GHz CPU, 4GB DDR3 memory, 500G SATAII hard disk. The OS is Debian 6.

1) *Snapshot Performance*

We write a script to evaluate the snapshot performance of iROW. This script creates a VM disk image with specified disk size, cluster size and format. Then the script writes data to the VM disk image. When the VM disk is full, a snapshot is created. Then the script writes data to the VM disk image again until the VM disk is full. Then the VM disk is rollback to the previous snapshot. After that, the script writes data to the VM disk image until it is full once again. At last, the script deletes the snapshot. For convenience, we use $T_{iROW\text{-}c}$, $T_{iROW\text{-}r}$ and $T_{iROW\text{-}d}$ to denote the snapshot creation time, rollback time and deletion time of iROW respectively, and use $T_{qcow2\text{-}c}$, $T_{qcow2\text{-}r}$ and $T_{qcow2\text{-}d}$ to denote the snapshot creation time, rollback time and deletion time of qcow2 respectively. The results of these experiments are shown in Figure 9, Figure 10, and Figure 11.

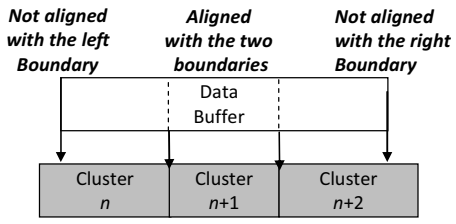

Figure 8 The relationship between the data buffer and clusters boundaries

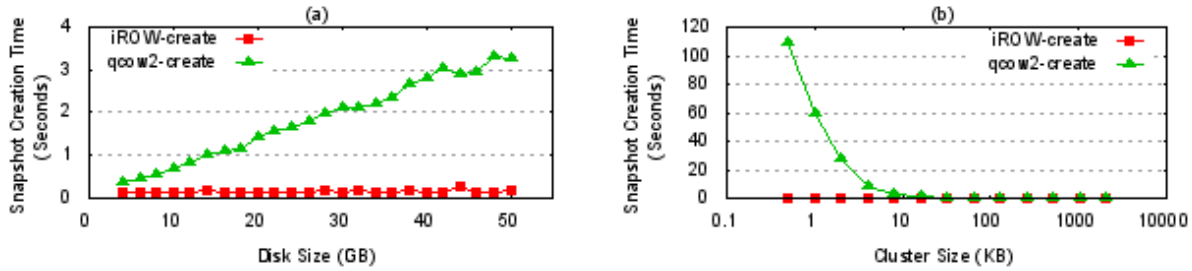

Figure 9 The snapshot creation time of iROW and qcow2: (a) the cluster size is 64KB (the default cluster size of qcow2); (b) the disk size is 1GB

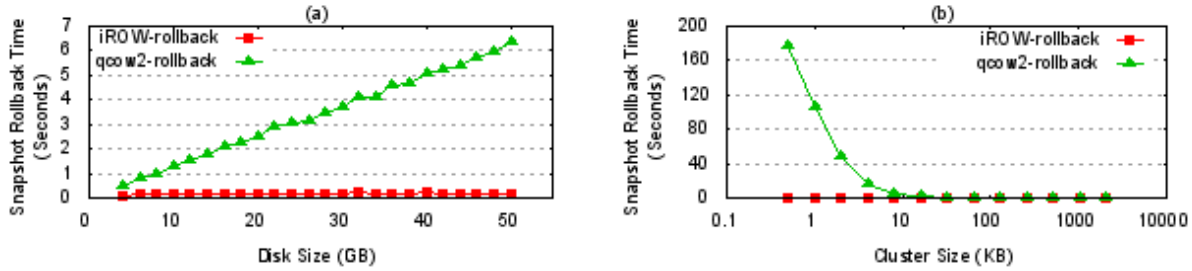

Figure 10 The snapshot rollback time of iROW and qcow2: (a) the cluster size is 64KB (the default cluster size of qcow2); (b) the disk size is 1GB

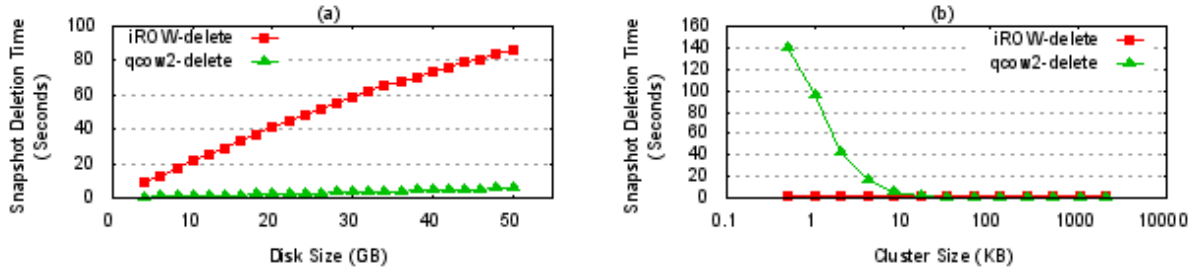

Figure 11 The snapshot deletion time of iROW and qcow2: (a) the cluster size is 64KB (the default cluster size of qcow2); (b) the disk size is 1GB

Figure 9 is the comparison of $T_{iROW\text{-}c}$ and $T_{qcow2\text{-}c}$. Figure 10 is the comparison of $T_{iROW\text{-}r}$ and $T_{qcow2\text{-}r}$. Figure 9 and Figure 10 show that $T_{iROW\text{-}c}$ and $T_{iROW\text{-}r}$ are independent of the VM disk size and the cluster size. However, the VM disk size and cluster size have great influence on $T_{qcow2\text{-}c}$, $T_{qcow2\text{-}r}$. Compared with qcow2, when the VM disk size is large or the cluster size is small, iROW has obvious advantages in both snapshot creation time and rollback time. In Figure 9-(a) and Figure 10-(a), when the disk size is 50GB, $T_{iROW\text{-}c}$ is 5.8% of the $T_{qcow2\text{-}c}$, and $T_{iROW\text{-}r}$ is only 2.8% of $T_{qcow2\text{-}r}$. In Figure 9-(b) and Figure 10-(b), when the cluster size is 0.5KB, $T_{iROW\text{-}c}$ is only 0.13% of the $T_{iROW\text{-}c}$, $T_{iROW\text{-}r}$ is only 0.06% of $T_{qcow2\text{-}r}$.

Figure 11 is the performance comparison of $T_{iROW\text{-}d}$ and $T_{qcow2\text{-}d}$. When it deletes a snapshot, iROW needs to merge the data and delete the corresponding files. So, compared with qcow2, iROW has some disadvantages in snapshot deletion time. We will discuss this issue in later section.

2) *I/O Performance*

We expand the qemu-io with I/O performance evaluation function. qemu-io is the I/O exerciser and diagnostic tools of KVM. It is mainly used to check the functionality of the virtual block driver. We add some I/O evaluation functions to the framework of qemu-io. These functions write random data with different block size to the VM disk image until the disk image is full. And then, these functions read data with different block size from the VM disk image. At the same time, these functions record the time that the writing and reading operation consumed. In order to avoid the influence of the host machine's system cache on the I/O performance, we open the VM disk image with O_DIRECT flag. The results of these experiments are shown in Figure 12, Figure 13, and Figure 14.

In Figure 12, "iROW" denotes the performance of iROW before snapshot, "iROW-s" denotes the performance of iROW after snapshot, "qcow2" denotes the performance of qcow2 before snapshot, and "qcow2-s" denotes the performance of qcow2 after snapshot. Figure 12 shows that the write performance of iROW is better than qcow2's. It

also shows that the write performance of iROW is independent of the cluster size.

Figure 13 shows that the read performance of iROW is better than qcow2's too. iROW caches the most recently read cluster, so in Figure 13-(b), the performance of iROW is even better than raw's, when the cluster size is lager than the data block size.

Figure 14 is the comparison of iROW and qcow2 in special conditions. Figure 14-(a) is the first write performance comparison. The metadata of iROW is much simpler than qcow2's metadata, so the first write performance loss of iROW is smaller than qcow2's. In addition, qcow2 has extra I/O operations on the first write after snapshot. So the first write performance of iROW is much better than qcow2's after snapshot. Figure 14-(b) is the worst case comparison. In Figure 14-(b), "iROW-COD" denotes the read performance of iROW when the first I/O operation is read after snapshot with the Copy-on-Demand enabled; "qcow2-s" denotes the first write performance of qcow2 after snapshot. Figure 14-(b) shows that the worst case of iROW still superior to the worst case of qcow2.

3) *Actual Disk Usage*

In order to prove the physical disk space that the iROW VM disk image takes up can gradually increase with the actual disk usage, we designed following experiment: First, we create a 10GB VM disk image with specified format (iROW or qcow2). Then we install Debian 6 and some necessary tools on it. After that, we download, decompress and compile the Linux Kernel. During this process, we use a script to record the real size of the VM disk image in every 100 seconds. The results are shown in Figure 15.

In Figure 15, stage A is the OS installing process; stage B is the VM powering up and tools installing process; stage C is the Kernel source code downloading and decompressing process; stage D is the Kernel compiling process. Figure 15 shows that the real size of iROW image can gradually increase with the actual disk usage; and its real size is almost the same as the real size of qcow2 image.

B. *Discussions*

In iROW, the time of a snapshot deletion is longer than qcow2 due to the data merge and the file deletions operations. However, this cost makes the largest benefits to the performance of critical snapshot operations. First, the efficient snapshot creation and rollback operations make a small impact on the VM and its applications, thereby maintaining business continuity in a cloud computing environment. Second, the frequency of snapshot creation and rollback will be much higher than the frequency of snapshot deletion. Third, the snapshot deletion operation can be performed in VM idle or non-busy period. Currently, in iROW, it has obvious advantages on two critical ones of the three snapshot operations, and it also has some advantages over qcow2 in I/O performance. Overall, iROW is an efficient disk image for snapshot of a VM, and has better performance than qcow2, and is able to achieve the reliability of the virtualized-based cloud environment.

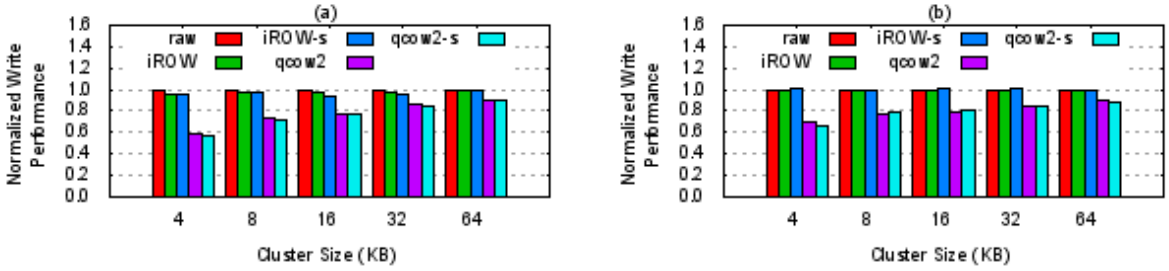

Figure 12 The normalized write performance: (a) the writing block size is 1MB; (b) the writing block size is 4KB

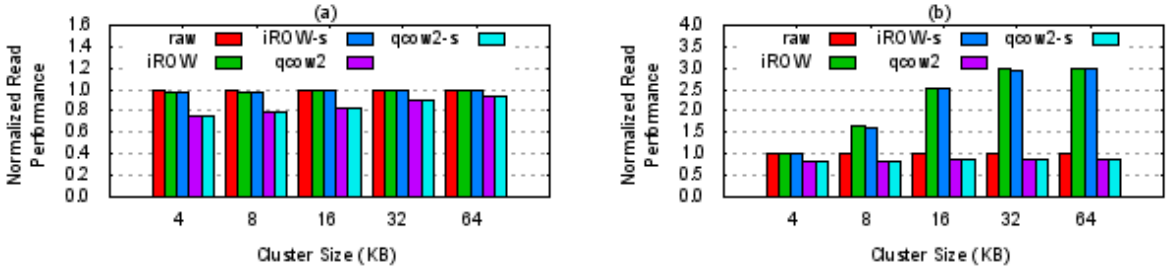

Figure 13 The normalized read performance: (a) the reading block size is 1MB; (b) the reading block size is 4KB

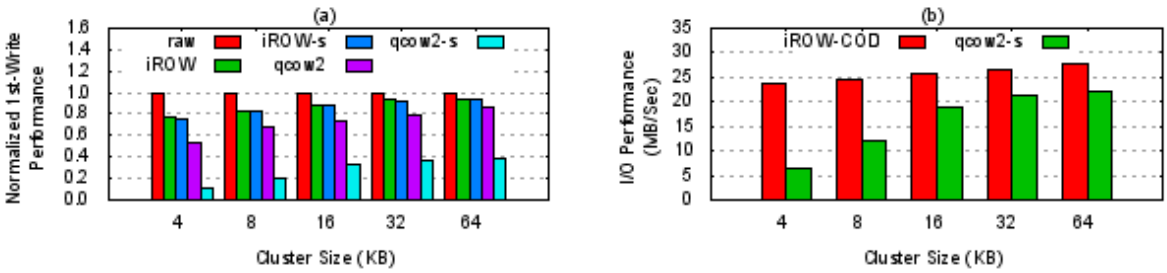

Figure 14 The I/O performance in special conditions: (a) the first write performance; (b) the worst case performance

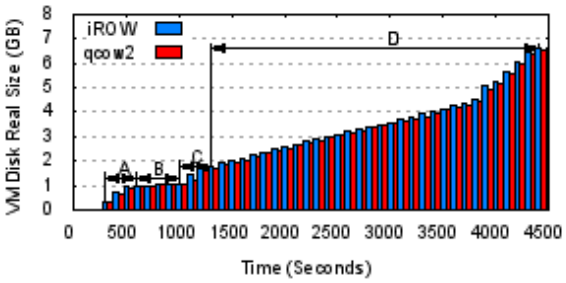

Figure 15 The real size of `iROW` and `qcow2` image

V. Conclusion and Future Work

The existing VM disk snapshot system solutions have some performance penalties on snapshot key operations and I/O operations. `iROW` is designed to solve these problems. `iROW` uses bitmap to replace the high-cost multi-level index tree structure, which is commonly used in existing solutions; `iROW` combines redirect-on-write with copy-on-demand; `iROW` gives the disk space allocation function back to the host machine's file system. These measures have enhanced both snapshot key operations performance and I/O performance of `iROW`. In addition, because of the host machine's file system supports sparse file, `iROW` also achieve that the VM disk image gradually increases with the actual disk usage.

The experiments show that, compared with `qcow2`, `iROW` has very obvious advantages in snapshot creation and rollback performance. When the VM disk size is 50GB and the cluster size is 64KB (the default cluster size of `qcow2`), the snapshot creation and rollback time is less than 6% and 3% of `qcow2`'s respectively. In addition, these advantages will be more obvious when the VM disk size is larger. The experiments also show that the I/O performance of `iROW` is better than `qcow2`'s. According to the cluster size, typically `iROW`'s I/O performance loss is 10%~50% less than `qcow2`'s; the first-write operation performance of `iROW` after snapshot is close to 250%~700% of `qcow2`'s; In the worst case, `iROW`'s I/O performance is 20%~270% higher than the worst case of `qcow2`. The performance of `iROW` is very stable in variety of VM disk size and cluster size.

We are also working on optimization of iROW such as snaspshot delete operation, and system integration into our iVIC cloud platform [11].

Acknowledgments

We would also like to thank Shouyu Si, JinSheng Zheng et al. for their helps. This work is partially supported by Program for National Nature Science Foundation of China (No. 61272165. 60903149), China 863 Project (No. 2011AA01A202), National Key Technology R&D Program (No.2012BAH42B04), and New Century Excellent Talents in University 2010.

References

[1] Kivity, Y. Kamay and D. Laor "kvm: the Linux Virtual Machine Monitor", in Proc. Linux Symposium vol. 1, pp. 225-230, 2007

[2] W. Xiao, Q. Yang, "Design and Analysis of Block-Level Snapshots for Data Protection and Recovery", IEEE Transactions on Computers, vol. 58, no. 12, 2009

[3] N. Garimella, "Understanding and exploiting snapshot technology for data protection", IBM developer Works, IBM, 2006, http://www.ibm.com/developerworks/tivoli/library/t-snaptsm1/index.html retrieved November 8, 2011

[4] D. T. Meyer, G. Aggarwal, B. Cully, G. Lefebvre, M. J. Feeley, N. C. Hutchinson, and A. Warfield, "Parallax: Virtual Disks for Virtual Machines", in Proc. ACM SIGOPS /EuroSys. (EuroSys'08), 2008

[5] A. Warfield, R. Ross, K. Fraser, C. Limpach, S. Hand, "Parallax: Managing Storage for a Million Machines", in Proc. USENIX Hot Topics in Operating Systems, pp. 1-11, 2005

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization", in SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles Pages 164-177

[7] L. Yu, C. Weng, M. Li, and Y. Luo, "SNPdisk: An Efficient Para-Virtualization Snapshot Mechanism for Virtual Disks in Private Clouds", in IEEE Network, vol. 25, pp. 20-26, 2011

[8] F. Bellard, "QEMU, a Fast and Portable Dynamic Translator", in USEIX FREENIX Track, 2005

[9] Intel Open Source Technology Center, The Parrallel Processing Institure at Fudan University. System Virtualization——Principle and Implementation. Beijing. Tsinghua University Press. 2009

[10] Intel Corporation. Intel 64 and IA-32 Architectures Software Developer's Manual, Vol3B, System Programming Guide Part 2, 2011

[11] Jianxin Li, Bo Li, Tianyu Wo, Chunming Hu, Jinpeng Huai, Lu Liu &. Lam K.P. (2012). CyberGuarder: A virtualization security assurance architecture for green cloud computing, *Future Generation Computer Systems*(*FGCS*), 38(2), 379–390. doi:10.1016/j.future.2011.04.012