LiveBackup

Jagane Sundar
jagane@sundar.org
LiveBackup
A complete Backup Solution

• Create Full and Incremental Backups of running VMs
• A System Administrator or Backup Software can use `livebackup_client` to connect to the qemu process using a TCP socket and transfer dirty blocks over
• Once dirty blocks are transferred over:
  • They can be saved in an incremental backup file and written to tape, or
  • The dirty blocks can be applied to a full backup image and kept ready to restart the VM on the backup host
LiveBackup - Overview
LiveBackup

Design Principles

• Designed for cloud operator needs
• Must have minimal performance impact on the running VM while it is in normal operation
• Must work with all types of virtual disk types, qcow, qcow2, LVM volumes, etc.
• Under failure circumstances, the operation of the VM must not be impaired (the backup operation can be sacrificed, but the VM must continue to operate just fine)
• Extra disk requirements should not be onerous
LiveBackup
A Complete Backup Solution

- Solution consists of:
  - Functional improvements to the qemu block layer:
    - Track dirty blocks in memory since last full backup. Persist this across VM reboots
    - A custom network protocol to transfer dirty disk blocks (will possibly be replaced by enhancements to libvirt)
    - A snapshot mechanism to maintain a snapshot while the backup client transfers the dirty blocks over to the backup server
  - livebackup_client to transfer dirty blocks from qemu
Characteristics

• Most of the time, i.e. when a livebackup_client is not connected to qemu, LiveBackup merely sets a bit in the in-memory bitmap indicating that the block has been ‘dirtied’

• When a backup client wants to take a backup, it will to create a point-in-time snapshot of all the virtual disks, usually after inducing the Application and Guest OS to flush buffered data
LiveBackup Example
Backup a VM that uses mysql

• ssh into guest OS and run mysql command:
  • ‘flush tables with read lock’
• Run ‘sync’ on guest OS to flush blocks (wait for a few seconds)
• Connect to qemu livebackup and call ‘create snapshot’
• ssh into guest OS and run mysql command:
  • ‘unlock tables’
• Connect to qemu livebackup and copy blocks for snapshot (all blocks for full backup, just dirty blocks for incremental backup)
• Connect to qemu livebackup and call ‘destroy snapshot’
Timeline of backup operation

LiveBackup client

guest (with mysql, etc.)

qemu with LiveBackup, running in host OS

mysql: flush tables with read lock

OS 'sync'

create snap

mysql: unlock tables

transfer blocks

destroy snap

Time during which snapshot is active

Time during which mysql cannot write
Livebackup – Normal operation

vdisk0 – Each small rectangle is a 512 byte block

(1) VM requests write to a block
(2) Qemu sets a bit in the dirty blocks bitmap
(3) Qemu writes block to file

In-memory dirty blocks bitmap
Livebackup – `livebackup_client` calls snapshot

`vdisk0` – Each small rectangle is a 512 byte block

- `livebackup_client` connects to `qemu` and asks for snapshot
- `Qemu` moves existing dirty blocks bitmap to `livebackup_snapshot` struct (only one snapshot can exist)
- `Qemu` allocates new dirty blocks bitmap

---

In-memory dirty blocks bitmap

---

Struct `livebackup_snapshot`

- Dirty blocks bitmap at the time of snapshot
- Bitmap of blocks in the COW file
- COW file containing modified blocks
Livebackup – livebackup_client runs backup

vdisk0 – Each small rectangle is a 512 byte block

- VM calls qemu to write block(s)
- livebackup_interposer checks if any of these blocks are marked dirty in livebackup_snapshot’s dirty bitmap
- If so, the blocks that are going to be overwritten are read in and saved to the COW file in the livebackup_snapshot. Then the VM’s write is allowed to proceed
- In the meantime, livebackup_client is busy copying the blocks marked dirty in livebackup_snapshot over the network. The COW file in the livebackup_snapshot is checked for dirty blocks before reading from the base file
Livebackup – VM write while livebackup_client runs backup

Virtual Machine

1. Set dirty bit in current dirty blocks bitmap
2. Check whether block is in snapshot's dirty blocks bitmap
3. If block is in snapshot's dirty blocks map, then copy that block from vdisk0 into COW file, then perform VMs write

In-memory dirty blocks bitmap

Struct livebackup_snapshot

dirty blocks bitmap at the time of snapshot

COW file containing modified blocks

vdisk0 – Each small rectangle is a 512 byte block
Usage Example

Livebackup Server (built into qemu process for the VM)

```
# ./x86_64-softmmu/qemu-system-x86_64 \
-drive file=/dev/kvm_vol_group/kvm_root_part,boot=on,if=virtio,livebackup=on \
-drive file=/dev/kvm_vol_group/kvm_disk1,if=virtio,livebackup=on \
-vnc 0.0.0.0:1000 -usb -usbdevice tablet \
-livebackup_dir /root/kvm/livebackup \
-livebackup_port 7900 \
-m 512 -net nic,model=virtio,macaddr=52:54:00:00:00:01 \
-net tap,ifname=tap0,script=no,downscript=no
```

Livebackup Client (run on backup server)

```
# livebackup_client /root/kvm-backup 192.168.1.220 7900
```
LiveBackup

Failure Scenarios

• qemu crashes during normal operation of the VM
  • livebackup_client is forced to do a full backup the next time around

• qemu crashes while livebackup is in progress
  • livebackup_client is forced to do a full backup the next time around

• livebackup_client crashes while livebackup is in progress
  • a new livebackup_client can redo the last type of backup it was doing - an incremental backup or a full backup

In most failure scenarios, the backup is impacted, but the VM itself continues to run unimpaired
LiveBackup

Testing methodology

**Hypothesis:** LiveBackup creates a backup of all the virtual disks of a VM such that the backup virtual disk images will be a bit for bit match of the virtual disk images at the point in time when the livebackup_client program issues a 'create snapshot' command

- I added code to my implementation of do_snap in livebackup.c, such that after the livebackup snapshot is created, I would invoke `lvcreate` to create a LVM snapshot of the underlying LVM logical volume
  - `# /sbin/lvcreate -L1G -s -n kvm_root_part_backup /dev/kvm_vol_group/kvm_root_part`
- Run the livebackup_client on the same machine as the VM to create a backup image of the virtual disks
- Run ‘cmp’ to compare the backup disk image file and the snapshot logical volume created
  - `# cmp /dev/kvm_vol_group/kvm_root_part_backup kvm_root_part`
Git repositories

- git://github.com/jagane/qemu-livebackup.git
- git://github.com/jagane/qemu-kvm-livebackup.git
LiveBackup

Thank You

Question? Comments? Flames?

Jagane Sundar
jagane@sundar.org