[Cinder] Support LVM on a shared LU

July/30/2014

Mitsuhiro Tanino <mitsuhiro.tanino@hds.com>

1.1 Background and Goal

- Background of this proposal
  - Currently, Cinder supports both commodity storage driver and enterprise 3rd party storage driver.
  - I'm focusing both type of storage drivers because I think it is better for Cinder to provide appropriate driver for corresponding to user's workload in cloud.
  - For example;
    - ✓ 3rd party storage driver
      ⇒ For "High performance" type storage volume.
      ex. I/O intensive workload, Required rich snapshot and backup features.
    - ✓ Commodity driver
      ⇒ For "Standard" type storage volume,
      ex. CPU intensive workload, Create and delete a large number of volumes for temporally usage.

1.1 Background and Goal

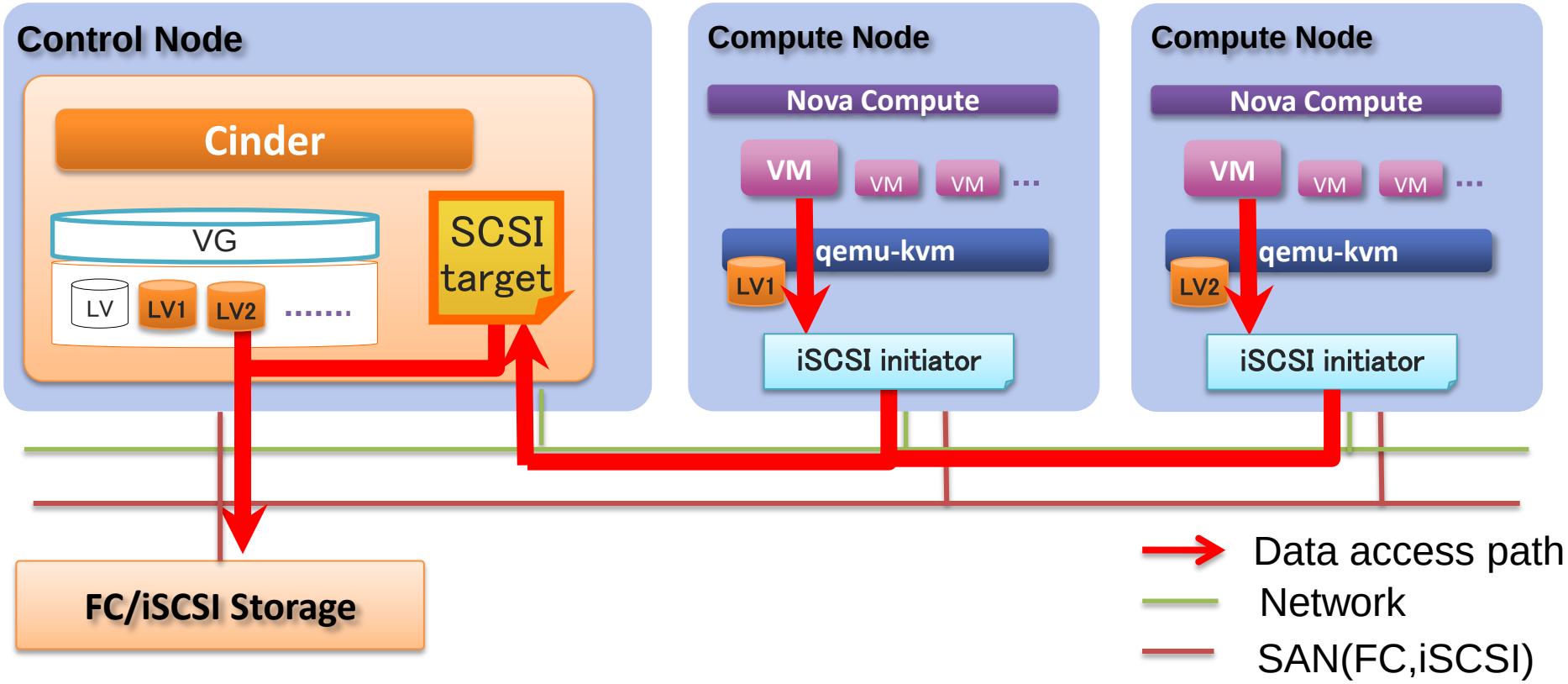
- Goal of my proposal
  - **Provide stable data access to backend storage from instances using commodity driver.**
  - Stable data access is must be provided in enterprise level cloud.
  - To provide stable data access, these are important points;
    1) Redundant data access path using dm-multipath, bonding, etc
    2) Use simple/direct data access path to back-end storage

Target of this proposal is 2).

- Current LVM based drivers
  - As for commodity storage driver, Cinder has LVM based drivers using tgtd or Linux-IO Target. Both of them requires SCSI targets on control node.
  - Using these targets, guest VM **can't issue I/O to their storage directly** via FC/iSCSI even if each node has own FC/iSCSI access.
  - All I/Os concentrate to a control node. If the target stops accidentally, all of I/O from guest VM also stops.

1.2 Stable data access to a storage

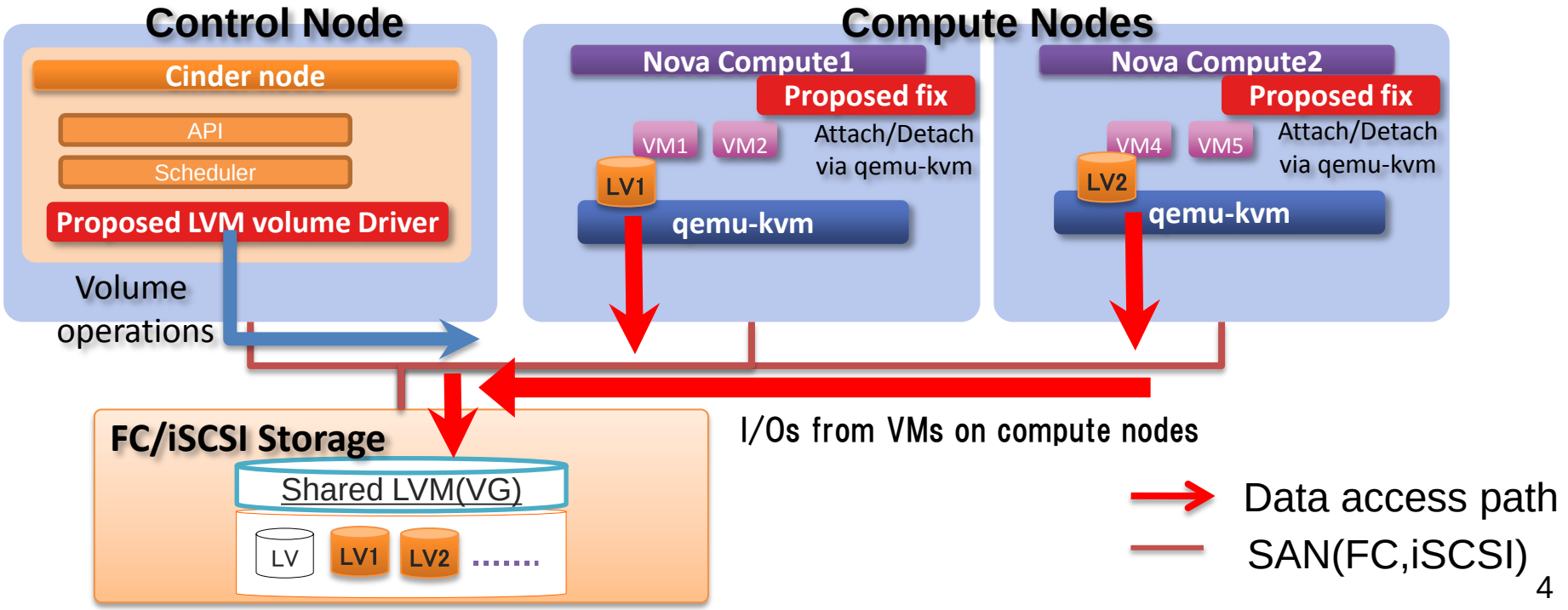
- Current LVM based driver(tgtd/LIO)



- The guest VM can't issue I/O to their storage directly via FC/iSCSi even if each node has own FC/iSCSI access
- If FC connector for LIO target will be implemented, we will able to use FC path from compute node, however we still need SCSI target on data path.

1.2 Stable data access to a storage

- Shared LVM driver(Proposed driver)
  - Purpose of this driver is to provide alternative LVM based commodity storage driver to support direct access to a back-end storage using shared LVM.
  - To profit a connectivity from FC or iSCSI, it is better to issue I/O from guest VMs to backend storage using direct data access path.
  - This will improve I/O bandwidth and response of guest VM, provide stable data access path and simplifies data access path to a storage.



1.3 Shared LVM driver

- Benefits of shared LVM driver
  - Comparison with current LVM based drivers;
    - ✓ I/O bandwidth and response of guest VM will be improved, because this driver use a direct access path via SAN(FC or iSCSI)
    - ✓ Stability of data path will be improved, because this driver does not use SCSI target.(reduction of access layer)
    - ✓ Reduce workloads of the control node because iSCSI tgtd or Linux-IO Target are unnecessary.

1.3 Shared LVM driver

- Use case of shared LVM driver
  - Enable cinder to any kinds of storages with shared volume
    - ✓ Currently, cinder supports limited model of storages and many old model of storages don't have cinder driver. This driver enables a storage which has a feature of shared volume. Using this driver, user does not need specific cinder 3rd party driver to enable their storage.
    - ✓ Some of old model of storages are not cloud ready, therefore it may take few minutes for each volume operation such as volume create/snapshot, etc. In this case, user can get merits of LVM such as quicker volume creation and snapshot creation within few seconds for these storages by using this driver.

1.3 Shared LVM driver

- Use case of shared LVM driver
  - Reduce hardware based storage workload
    - ✓ To use enterprise storage more efficiently, it is better to reduce hardware based storage workload by offloading it to software based volume operation on control node by using multiple drivers.

    - ➢ 3rd party driver
      ⇒ For "High performance" type storage. ex. I/O intensive workload
    - ➢ Shared LVM driver
      ⇒ For "Standard" type storage, ex. CPU intensive workload , create and delete a large number of volumes for temporally usage.

    This enables that user can select an appropriate storage type on a case by-case-basis.

1.4 Comparison with other drivers

	LVMiSCSI	Shared LVM Driver	FC/iSCSI (3rd partyDriver)
Implementation of volume	LV (managed by LVM)	LV (managed by LVM)	LU (managed by storage)
Volume Operation	By software (LVM)	By software (LVM)	Good By hardware (Storage)
Work Load	Server side	Server side	Storage side
Supported Storage	Good Any storage (Storage independent)	Good Any storage (Storage independent)	Specific storage (Requires specific plugin)
Volume Access Path	Via software iSCSI target	Good Direct from fibre channel	Good Direct from fibre channel
Performance	Not enough	Better I/O performance Good	Better I/O performance Good
HA	Active/Passive：○ Active/Active ：—	Active/Passive：○ Active/Active ：—	Active/Passive：○ Active/Active ：○ Good

Rich storage features such as backup, COW snapshot are supported.

Better support coverage.

Better I/O performance.

Both A/A and A/P are supported.

1.5 Performance measurement

❑ Test environment

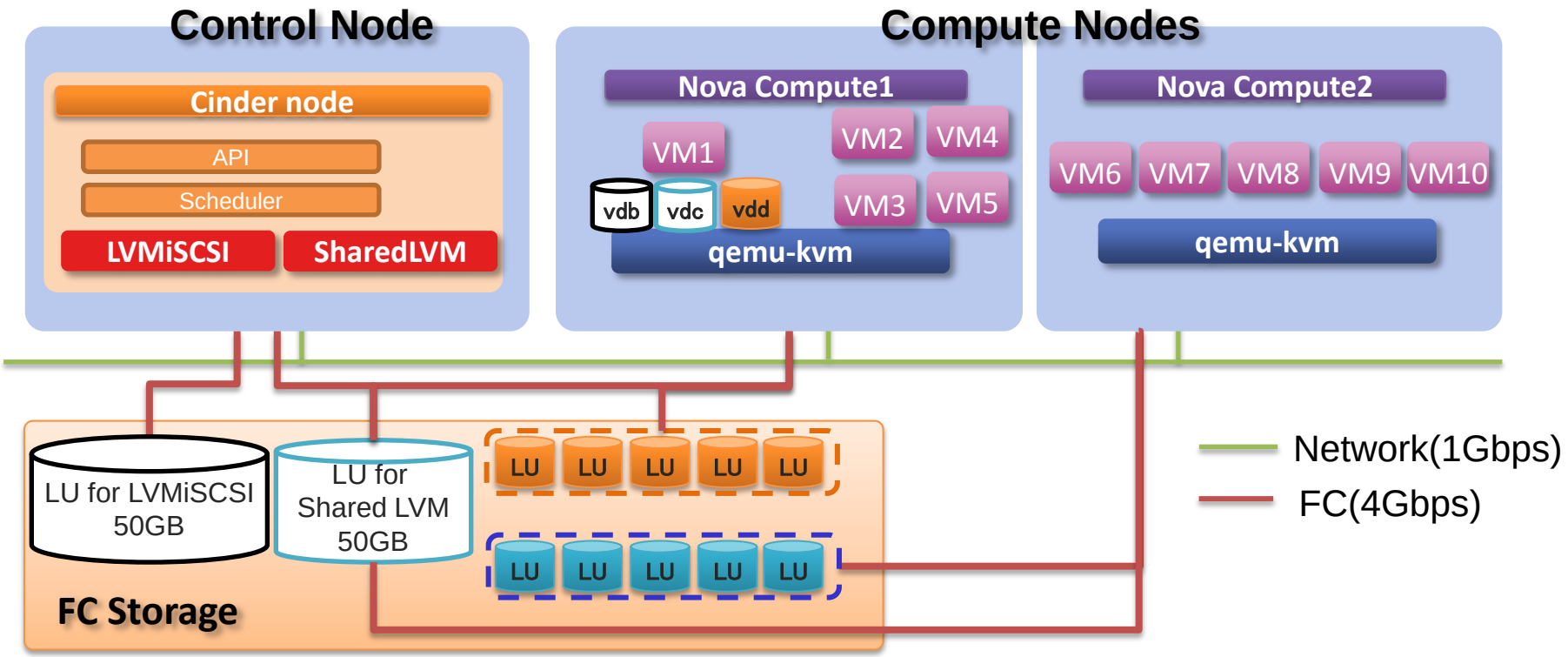
- Control node x 1(CPU:4core x2, mem:8GB)
  - ✓ Service: nova, cinder, glance, keystone, horizon, etc
  - ✓ Storage
    - ➢ LVMiSCSi: 50GB LU via FC
    - ➢ ShredLVM: 50GB LU via FC(shared between Control and Computes)

- Compute node x 2(CPU:4core x2, mem:8GB)
  - ✓ Service: nova-compute
  - ✓ Storage
    - ➢ LVMiSCSi:
    - ➢ ShredLVM: 50GB LU via FC(shared between Control and Computes)
    - ➢ 5 x FC LU for Raw FC volume attach.
  - ✓ Guest VM: 5 VM per each compute node

1.5 Performance measurement

❑ Test environment

- Guest VM(m1.small flavor): 10 VM
  - Each VM has 3 types of volumes
    - ➢ vdb: Volume created by LVMiSCSI
    - ➢ vdc: Volume created by SharedLVM
    - ➢ vdd: Attach RAW FC volume via virsh attach-device using same config of “nova volume-attach”.



1.5 Performance measurement

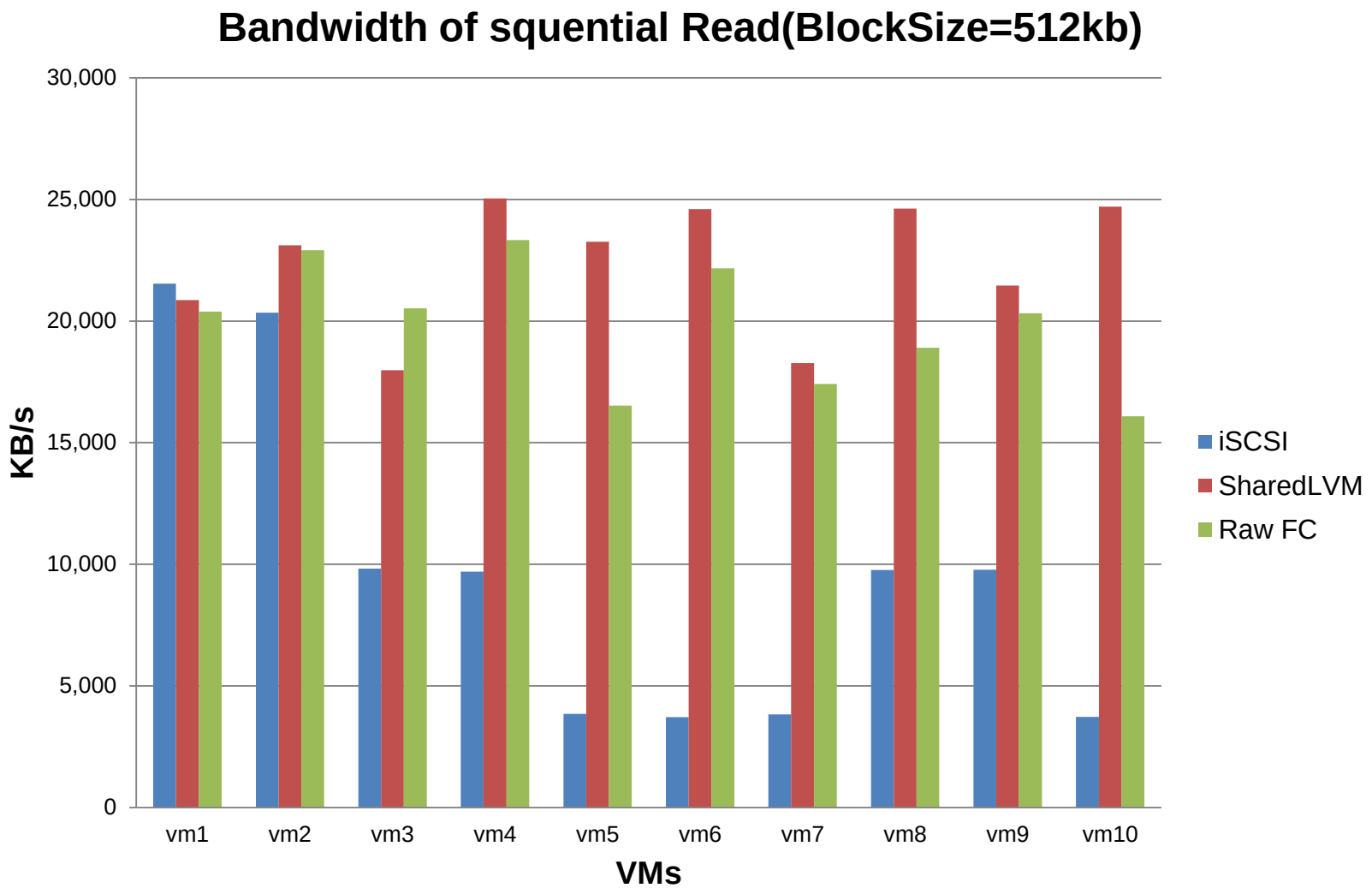
❑ Test tool

- fio(http://freecode.com/projects/fio/)
- Test patterns

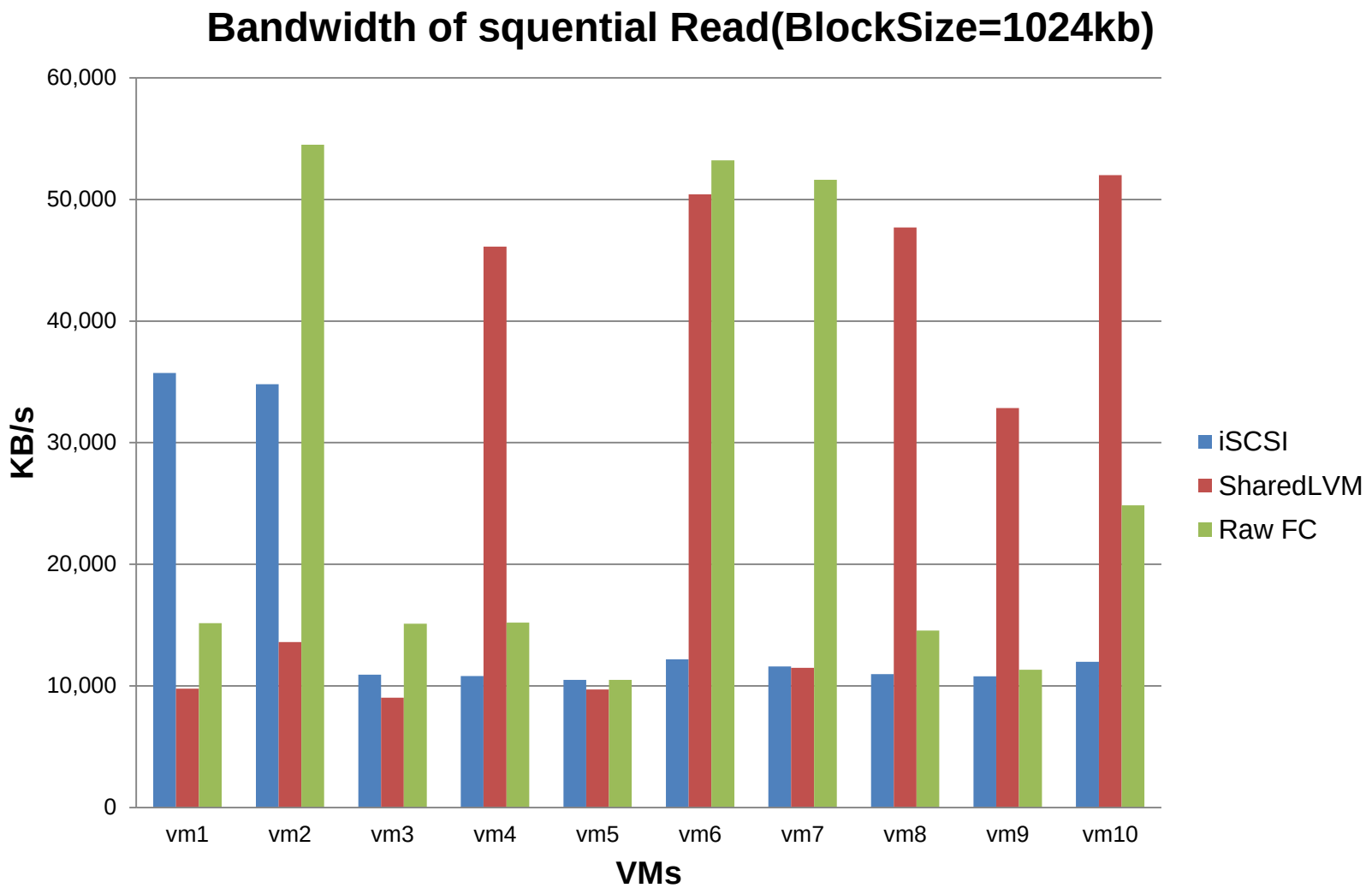
  Run fio using these I/O patterns inside guest VM
  - ✓ Sequential read
  - ✓ Sequential write
  - ✓ Random read
  - ✓ Random write

- Test command example;
  - ✓ fio --name=lvm_seqread -filename=/dev/vdc -direct=1 -rw=read -ramp_time=20 -bs=512k -size=4G --output=lvm_rd_512k.txt -runtime=120 -iodepth=128 –numjobs=5 -group_reporting -ioengine=libaio

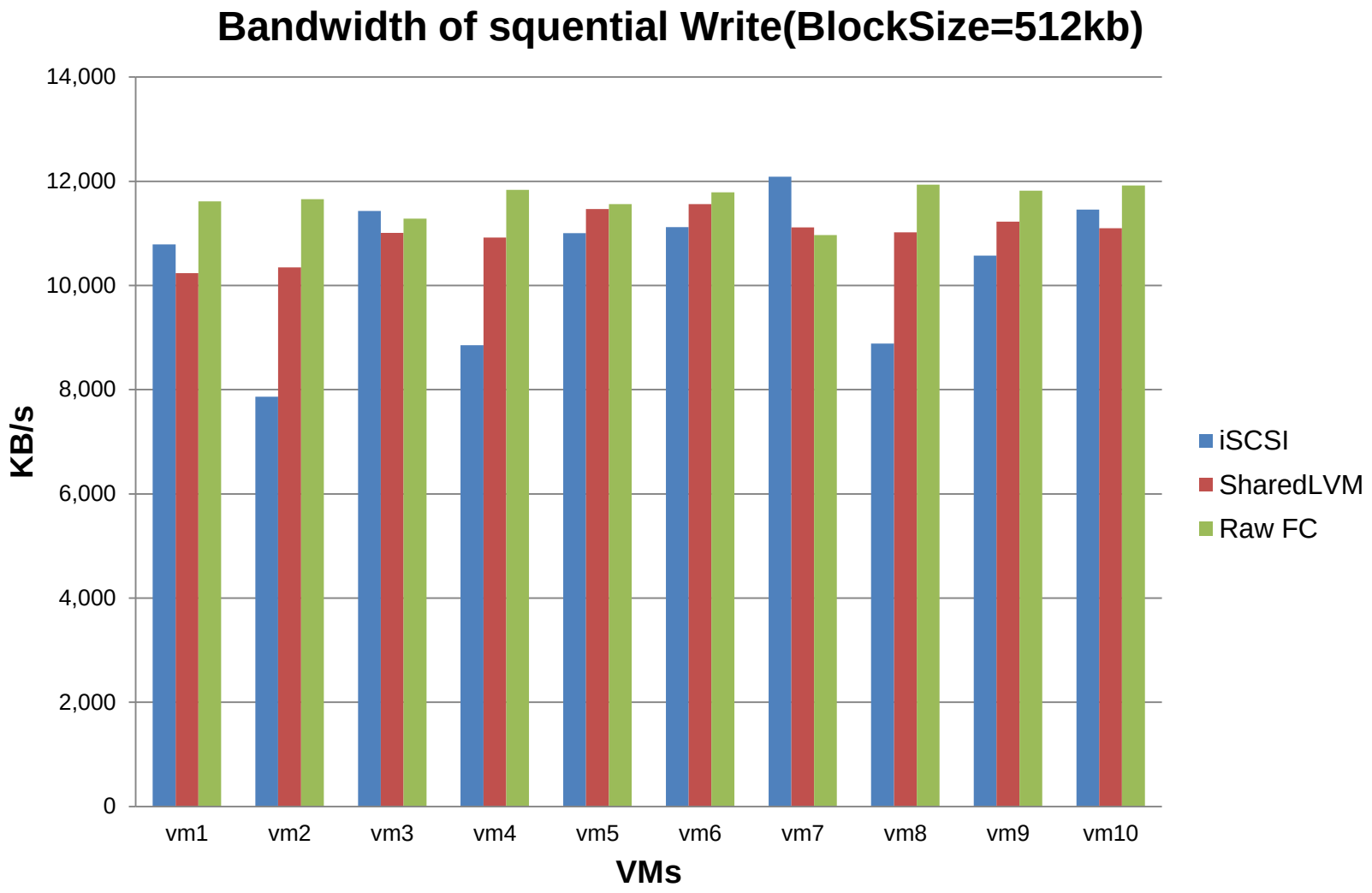
1.5 Performance measurement



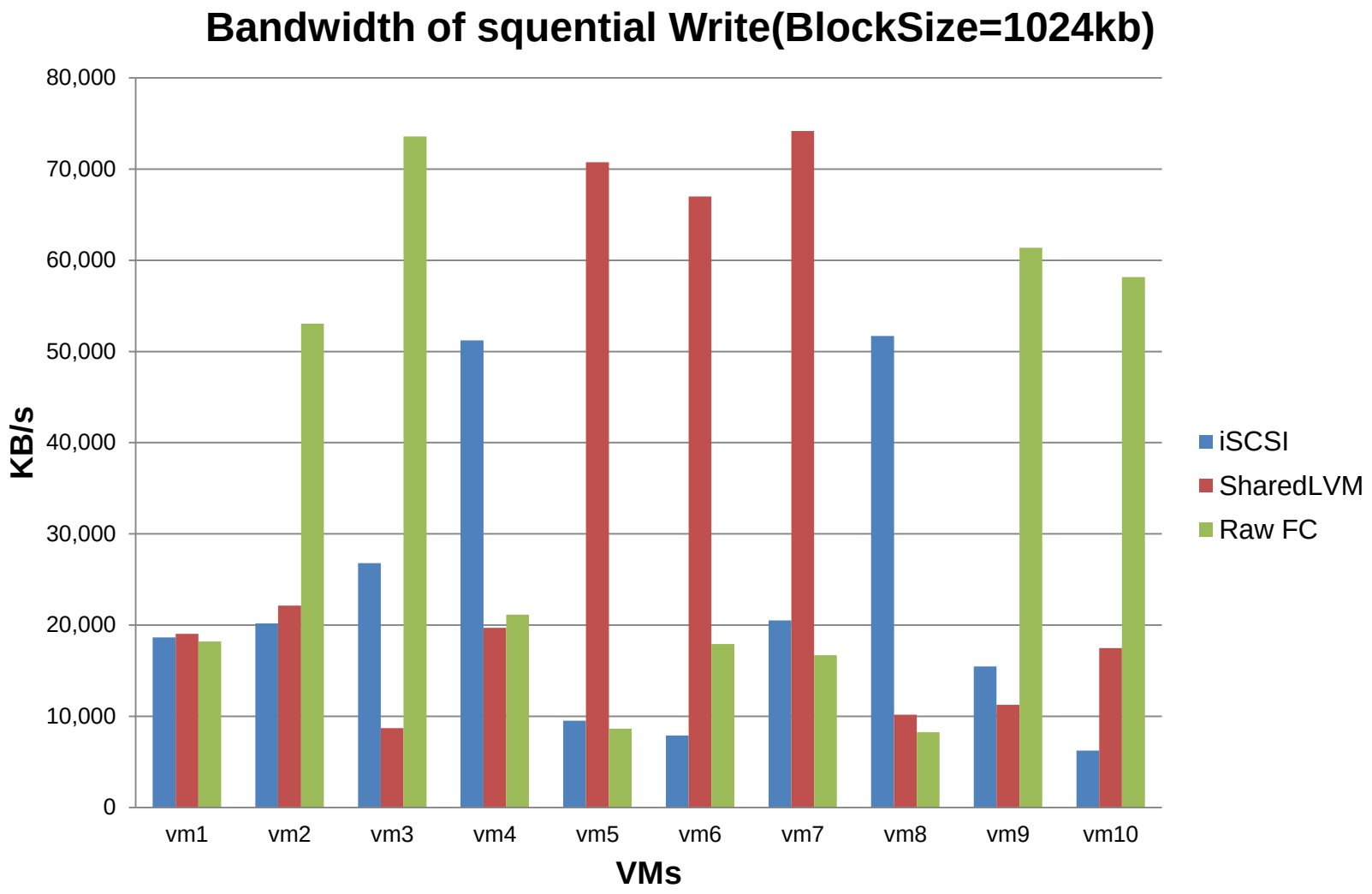
1.5 Performance measurement



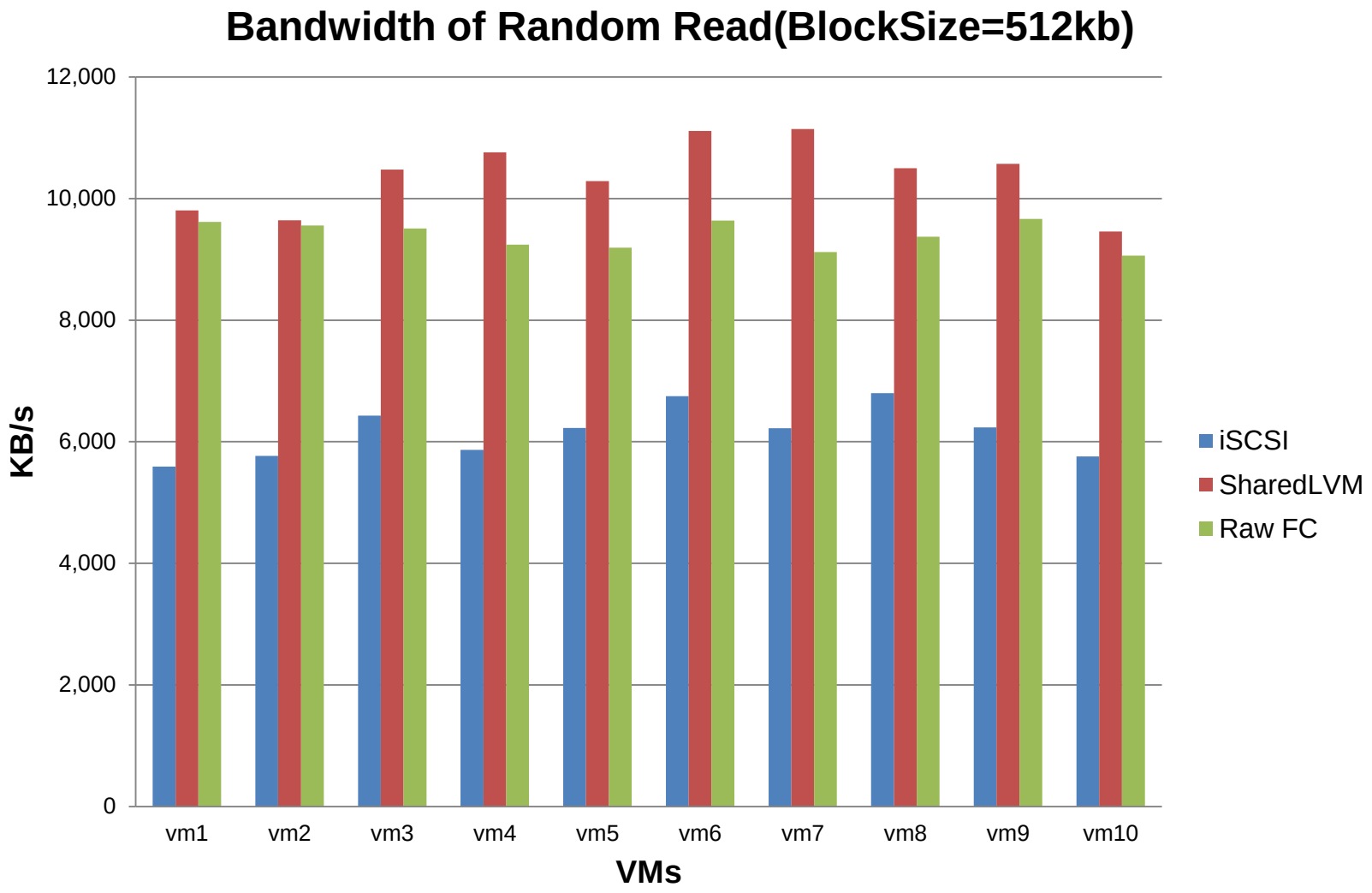
1.5 Performance measurement



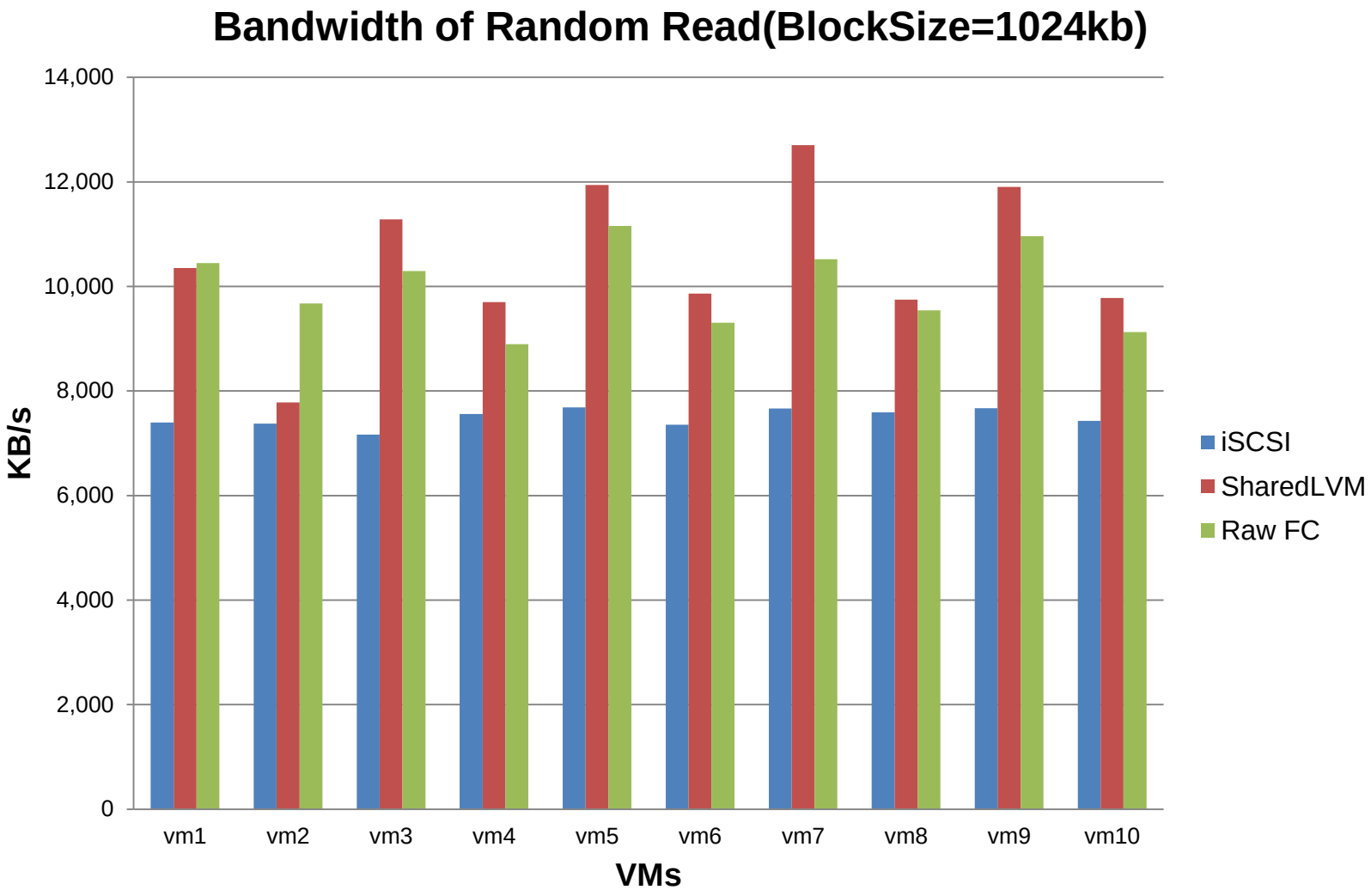
1.5 Performance measurement



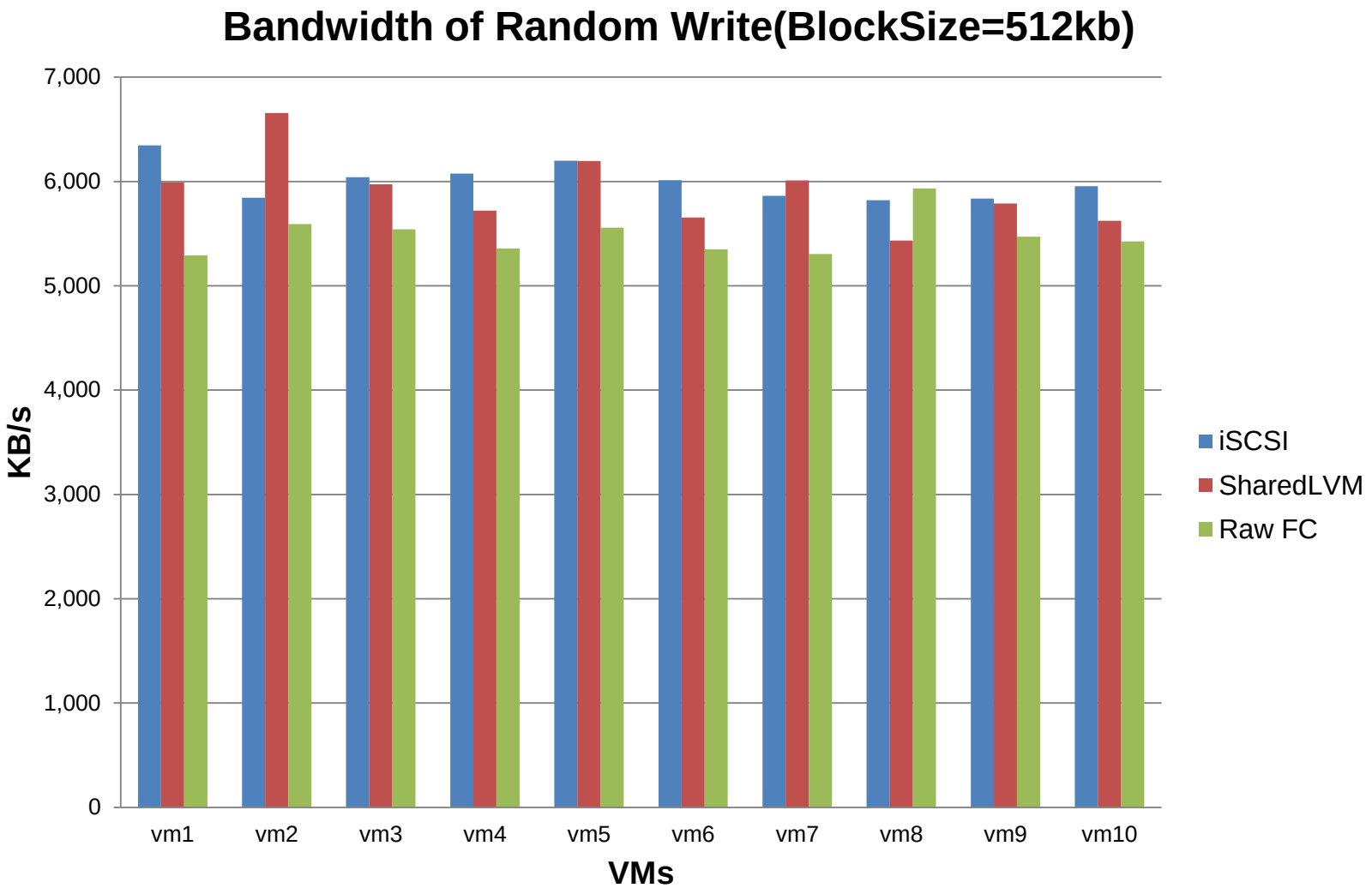
1.5 Performance measurement



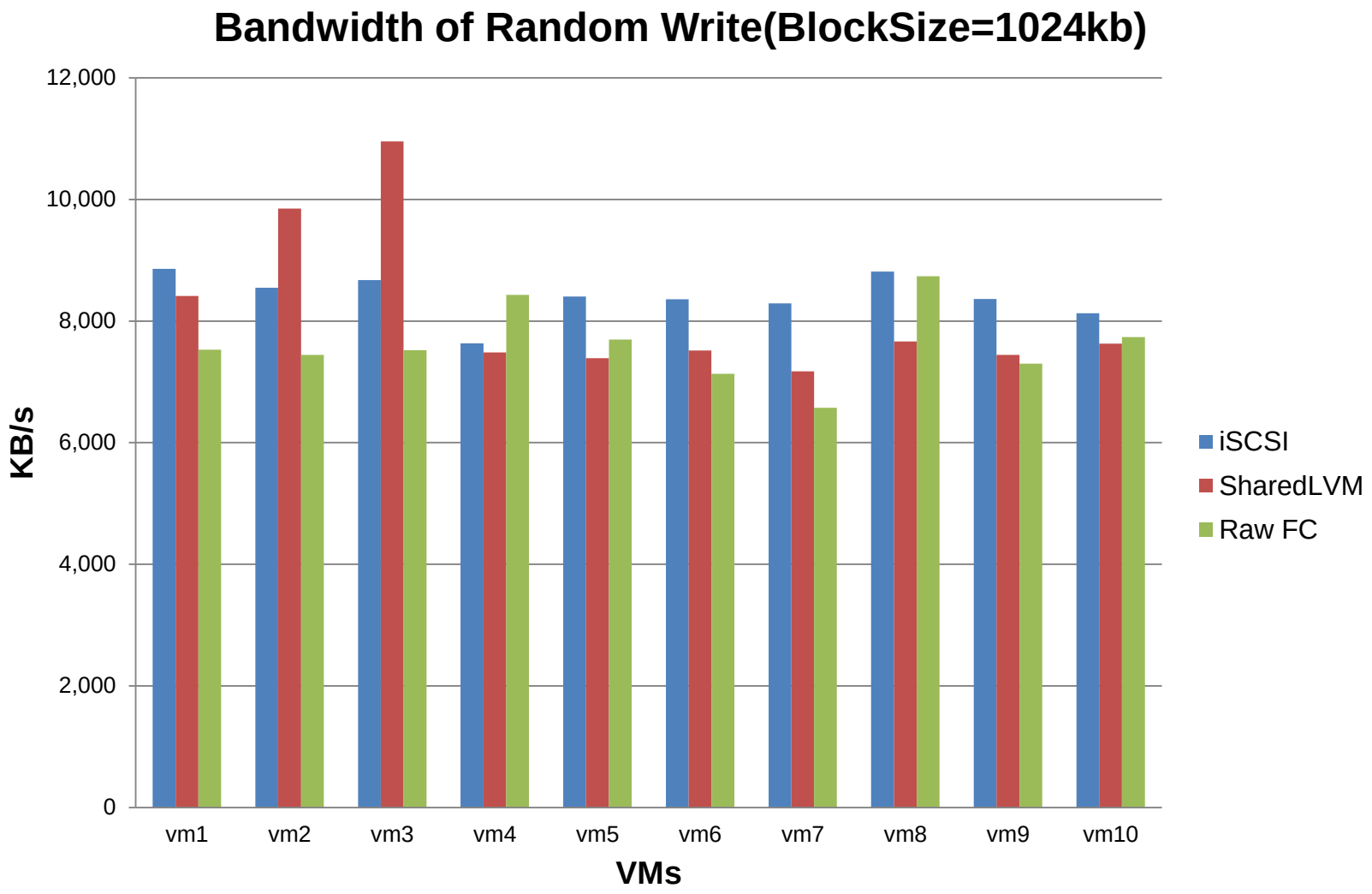
1.5 Performance measurement



1.5 Performance measurement



1.5 Performance measurement



1.5 Performance measurement

- Cinder driver settings

```
enabled_backends=LVM_iscsi_drv,LVM_shared_drv

[LVM_iscsi_drv]
volume_group=stack-volumes
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name=LVM_iscsi
volume_clear=none
iscsi_write_cache = off
[LVM_shared_drv]
volume_group=shared_lvm
volume_driver=cinder.volume.drivers.lvm.SharedLVMDriver
volume_backend_name=LVM_shared
volume_clear=none
```

1.5 Performance measurement

- Example of guest VM XML file

```
<disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none'/>
    <source dev='/dev/disk/by-path/ip-xx.xx.xx.xx:3260-iscsi-iqn.2010-
10.org.openstack:volume-247ce1bf-8463-458b-9402-f23f929c50fa-lun-1'/>
    <target dev='vdb' bus='virtio'/>
    <serial>247ce1bf-8463-458b-9402-f23f929c50fa</serial>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
</disk>
<disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none'/>
    <source dev='/dev/shared_lvm/volume-a57f8601-911f-4023-9ad9-6a63d2d455bc'/>
    <target dev='vdc' bus='virtio'/>
    <serial>a57f8601-911f-4023-9ad9-6a63d2d455bc</serial>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
</disk>
<disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none' io='threads'/>
    <source dev='/dev/disk/by-path/pci-0000:04:04.0-scsi-0:0:0:11'/>
    <target dev='vdd' bus='virtio'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
</disk>
```

1.6 Conclusion

- Current LVM based drivers provide a volume via software SCSI target even though each server node has direct access path to storage via FC or iSCSI.
- To profit a connectivity from FC or iSCSI, I proposed Shared LVM driver in order to use direct access path to a storage via SAN or iSCSI.
- This will improve I/O bandwidth and response of guest VM compared to current LVM driver.

❑ Links

[Blueprint] LVM: Support a volume-group on shared storage

Cinder:

https://blueprints.launchpad.net/cinder/+spec/lvm-driver-for-shared-storage
https://review.openstack.org/#/c/92479/

Nova

https://blueprints.launchpad.net/nova/+spec/lvm-driver-for-shared-storage
https://review.openstack.org/#/c/97602/
https://review.openstack.org/#/c/92443/