Sheepdog is ready:
~Distributed block storage is turning from experiment to commercial use~

Teruaki Ishizaki
NTT Software Innovation Center

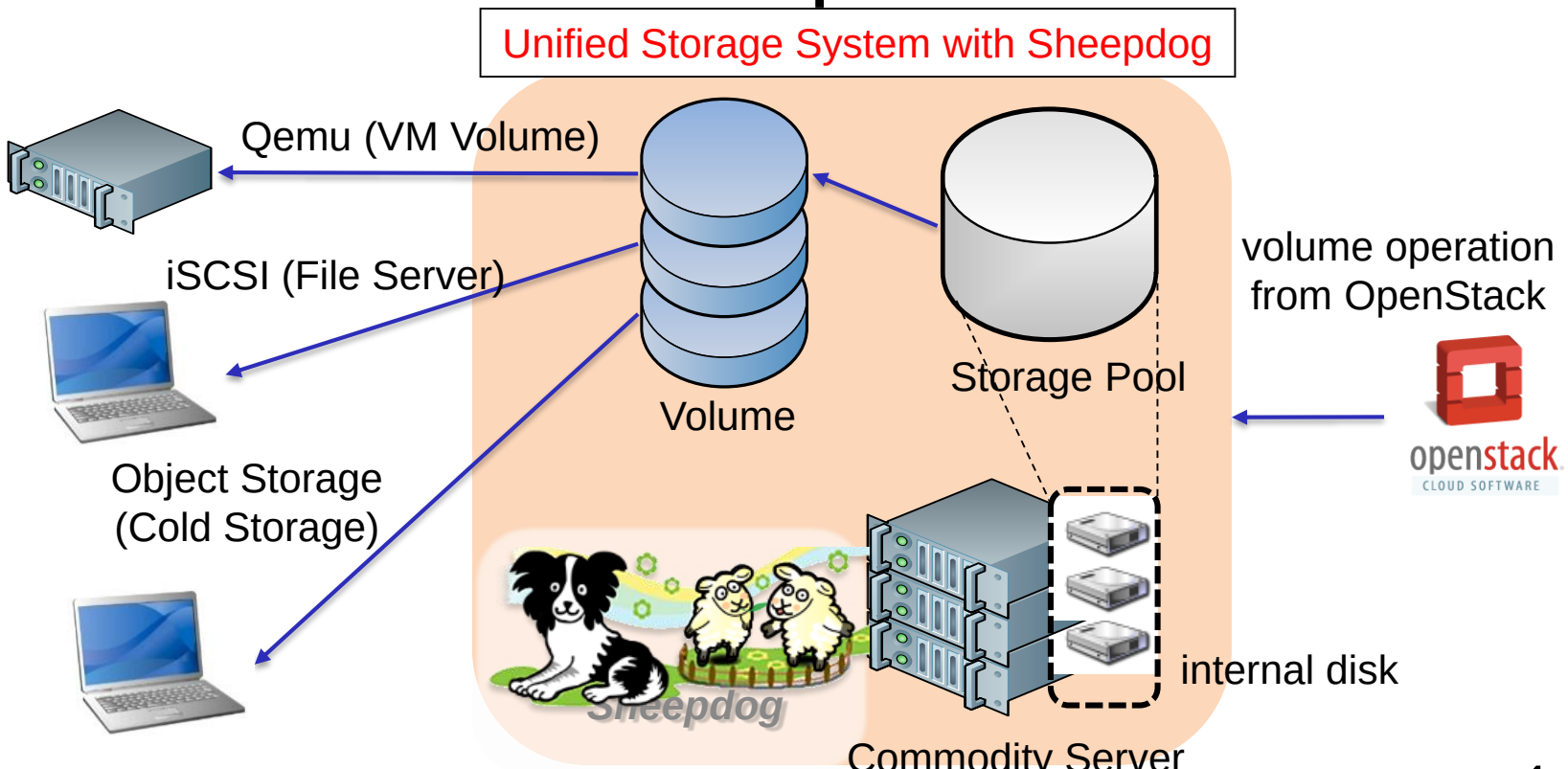
Outline

1. What is Sheepdog?
2. What are required for distributed storage?
3. Developments for commercial use
4. Performance test with SSD
5. Use Cases
6. Conclusions

Outline

What is Sheepdog?

- **First developed as distributed block storage**
- **Sheepdog currently covers many use cases**
  - IaaS backend, FileServer, Cold Storage
- **OpenStack Cinder/Glance cooperation**



Why was Sheepdog developed?

Innovative R&D by NTT

- **NTT Laboratory released Sheepdog first as OSS for experimental use (2009)**

- **Purpose was to verify that it was feasible to make distributed block storage having following features**
  - Durability
  - Scalability
  - Manageability
  - Availability

NTT

Outline

1. What is Sheepdog?
2. **What are required for distributed storage**
    1. Durability
    2. Scalability
    3. Manageability
    4. Availability
3. Developments for commercial use
4. Performance test with SSD
5. Use Cases
6. Conclusions

What are required for distributed storage?

- Durability
- Scalability
- Manageability
- Availability

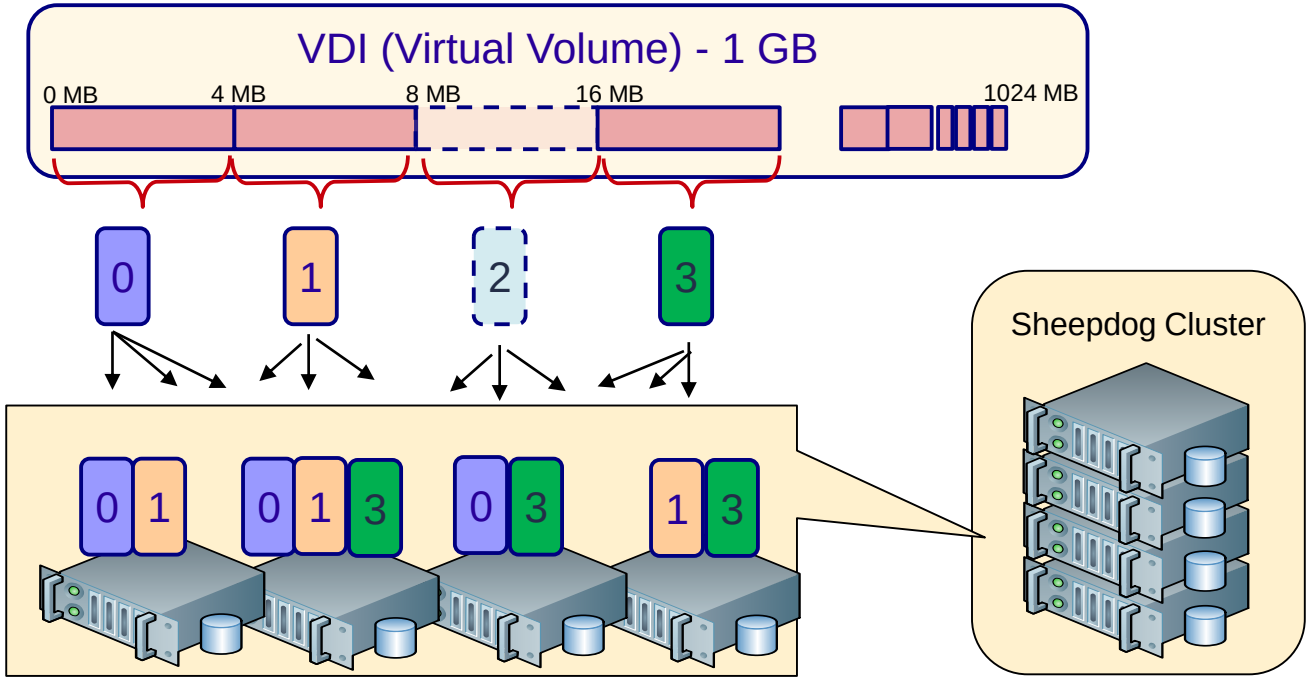
We tried to use Sheepdog as commercial product

What are required for distributed storage?

- **Durability**
- Scalability
- Manageability
- Availability

Sheepdog's Durability (1/2)

- **Virtual Volume is divided into fixed-size objects**
- **Each fixed-size objects is saved in multiple servers as file, so each object is durable**
  - Replica and ErasureCode are available



Sheepdog's Durability (2/2)

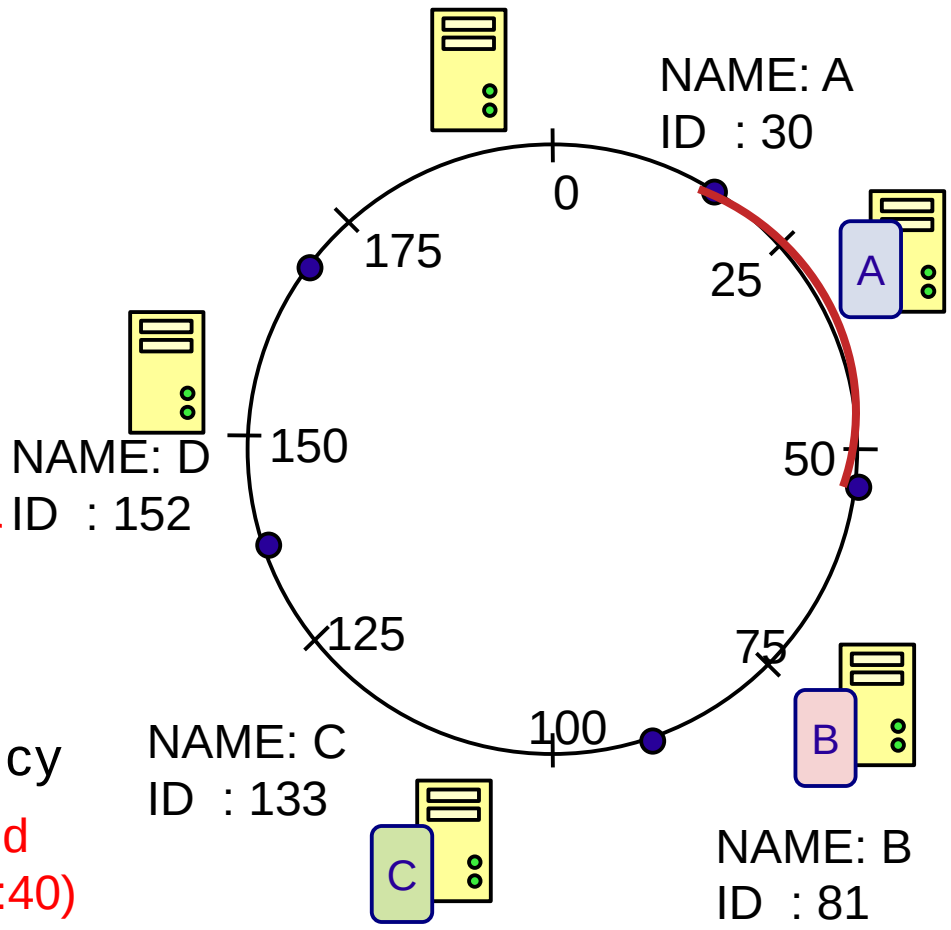
- **Object placement Policy**
  - Consistent Hashing
- **Hash key**
  - Calculated using server IP or Volume Name/Index

OBJ ID : 40

When its redundancy is three, Servers A, B, C are selected for storing object

- **Replacement Cost**
  - Low in recovering redundancy

When its redundancy is three and Servers A is down, the object(ID:40) is copied to Server D from Server B or C

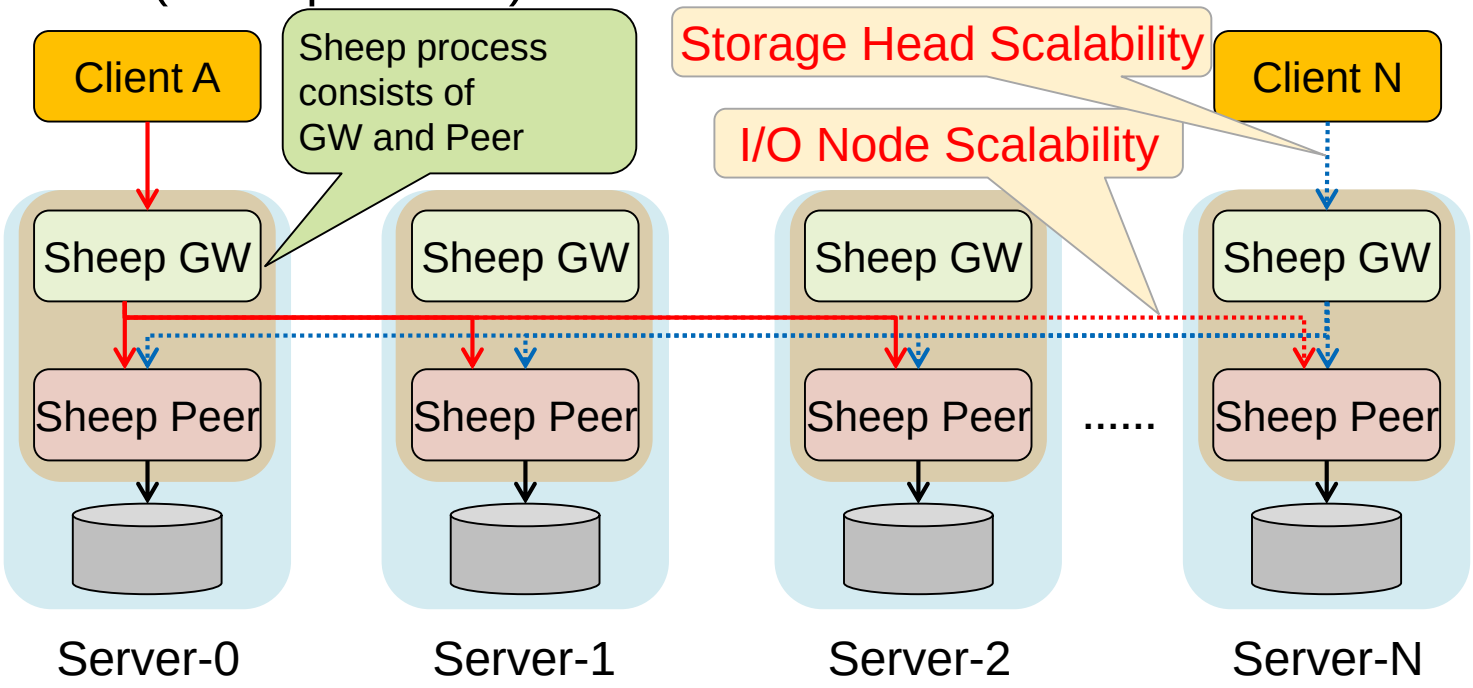


What are required for distributed storage?

- Durability
- **Scalability**
- Manageability
- Availability

Sheepdog's Scalability

- **Load Balance**
  - Clients using a different Volume can connect to the respective Sheep Gateway (GW)
- **I/O Performance**
  - If add servers, Sheep GW can send I/O requests to more servers (Sheep Peers)



What are required for distributed storage?

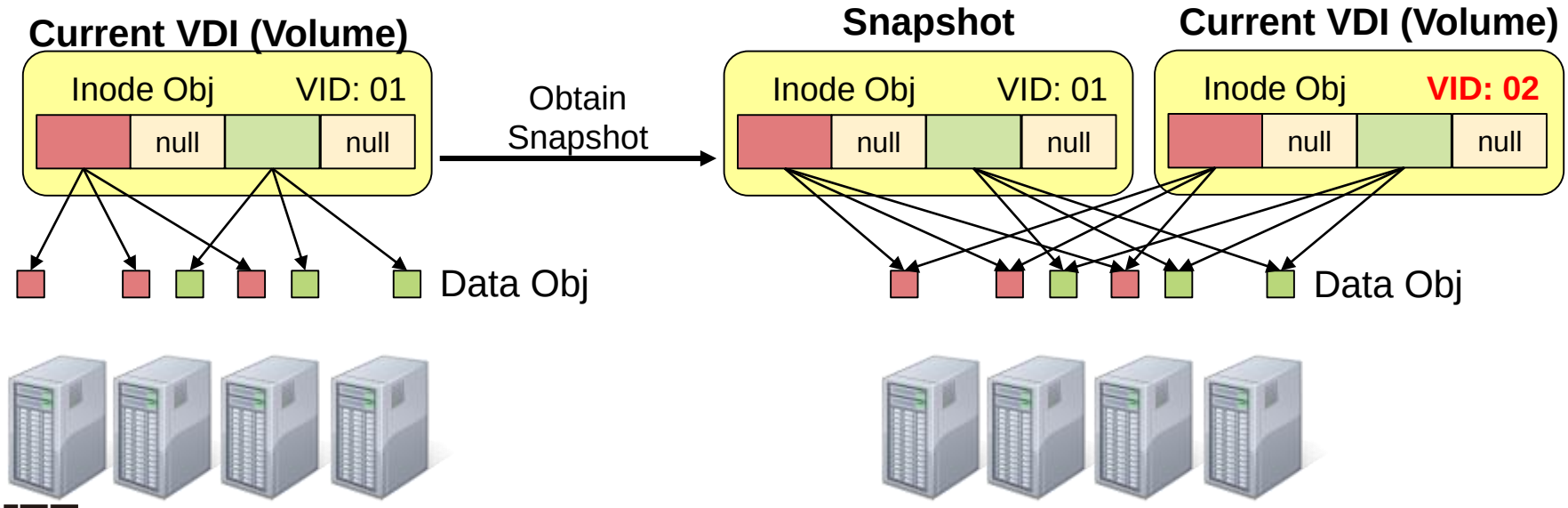
- Durability
- Scalability
- **Manageability**
- Availability

Sheepdog's Manageability

- **Auto Operation**
  - Relocation after adding a new server
  - Recovering data redundancy after a server crash
  - Returning data placement to the original server after repairing a crashed server
- **Volume Management**
  - Snapshot
  - Clone
  - Rollback
  - Export Volume
  - Import Volume
  - Full/Incremental Backup

Internal Data of Sheepdog: Snapshot/Clone

- **It requires little time to get Snapshot/Clone**
  - Volume Metadata are stored in 'Inode Object (Obj)' file
  - Inode Obj includes pointers to Data Obj
  - Inode Obj is copied for obtaining Snapshot/Clone and ID of Current VDI is changed
  - Copy on Write (COW) is done after obtaining Snapshot/Clone



What are required for distributed storage?

- Durability
- Scalability
- Manageability
- **Availability**

Sheepdog had Durability, Scalability and Manageability
But, Availability was not sufficient

Sheepdog's Availability – Problems

Innovative R&D by NTT

- **No SPOF**
  - Single point of failure (SPOF) does not exist for using Sheepdog as Qemu block
  - SPOF exists as iSCSI (tgtd)

    The iSCSI storage head feature was an additional feature.
    So, it was difficult to satisfy "No SPOF" feature for iSCSI head without breaking Sheepdog's initial design policy for Qemu block.

- **Appropriate Volume's Data Lifecycle**
  - Data Obj cannot be deleted until all VDIs of same family have been deleted

    It is the reason that we wanted to make Sheepdog's implementation simple first

NTT

Outline

1. What is Sheepdog?
2. What are required for distributed storage?
3. **Developments for commercial use**
   - iSCSI multipath support
   - Improved Garbage-Collection(GC) implementation of Data Obj
4. Performance test with SSD
5. Use Cases
6. Conclusions

Developments for commercial use

Innovative R&D by NTT

We have developed for solving problems to satisfy 'Availability'

- **iSCSI multipath support (v0.9.1~)**
- **Improved GC implementation of Data Obj (v0.9.0~)**

NTT

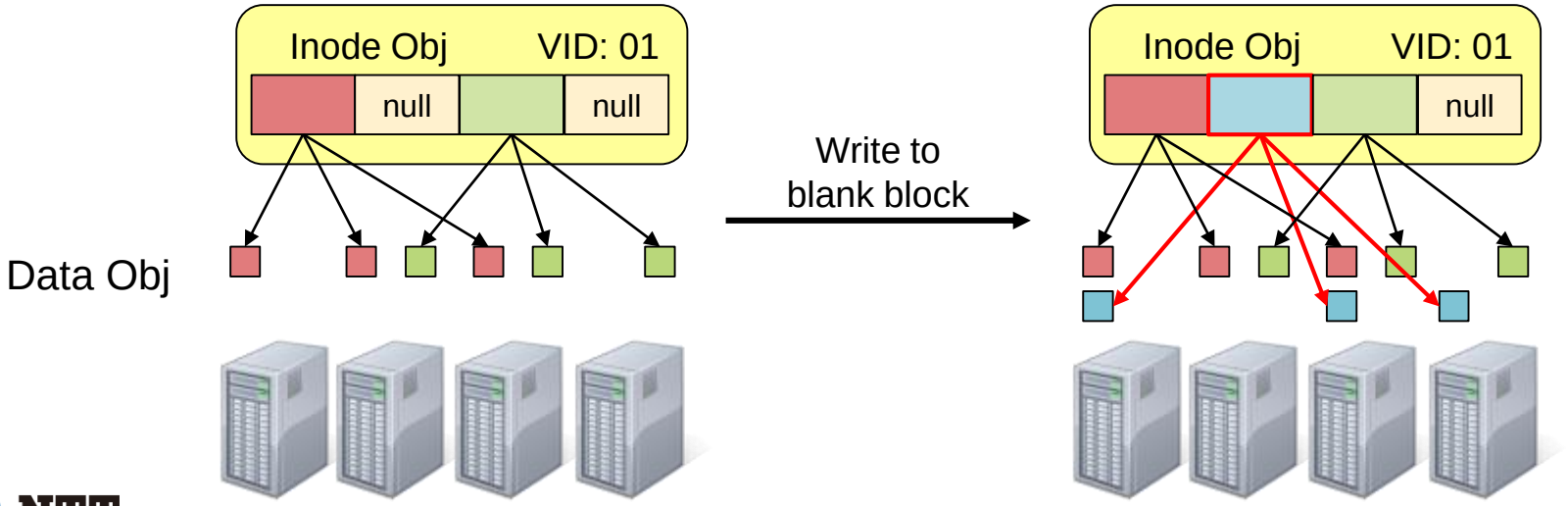
Developments for commercial use

We have developed for solving problems to satisfy ‘Availability’

- **iSCSI multipath support (v0.9.1~)**
- **Improved GC implementation of Data Obj (v0.9~)**

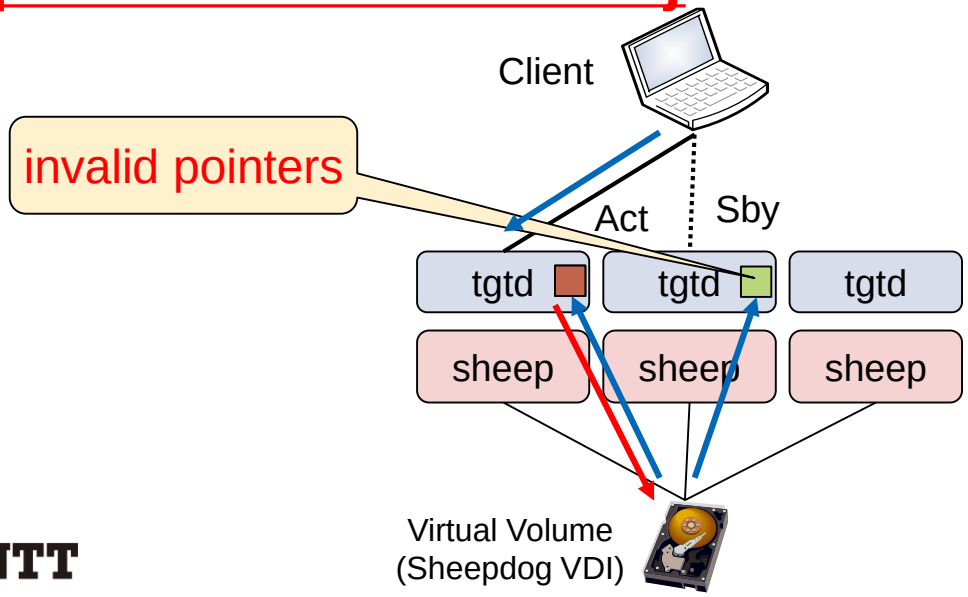
Internal data of Sheepdog: Inode

- **Inode Obj includes pointers to Data Objs**
- **Inode Obj is saved on Disks such as Data Obj**
- **pointers can be changed by write operation**
  - Write to blank block (No-Preallocation Volume)
  - Write after obtaining snapshot/clone/rollback (COW)
- **Storage Head has pointer info on memory**



iSCSI multipath Problem

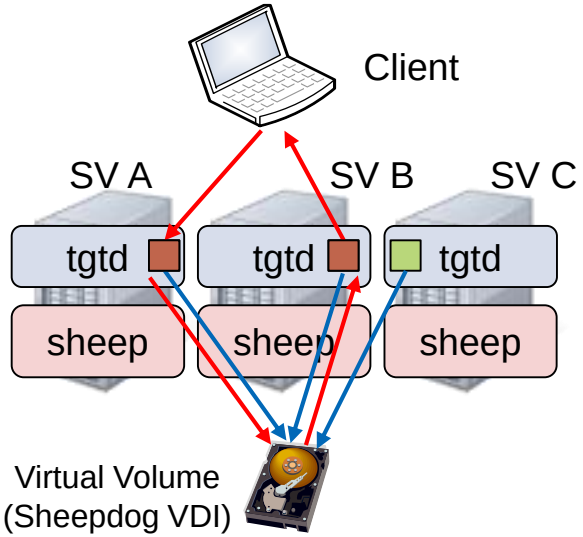
- **Act tgtd and Sby tgtd have pointer info of Inode Obj on memory**
- **Tgtds reads Inode Obj from Sheepdog when connecting Sheepdog Volume for creating iSCSI LUN**
- **Sby tgtd cannot detect if Act tgtd has changed pointers in Inode Obj**



1. read inode obj from disk when connecting to the VDI
2. tgtds have inode info on mem
3. Act write to Sheepdog VDI with changing inode info
4. inode info of Act is changed
5. inode info is written back to disk
6. Sby's pointers info is invalid

Keeping tgtds Coherent

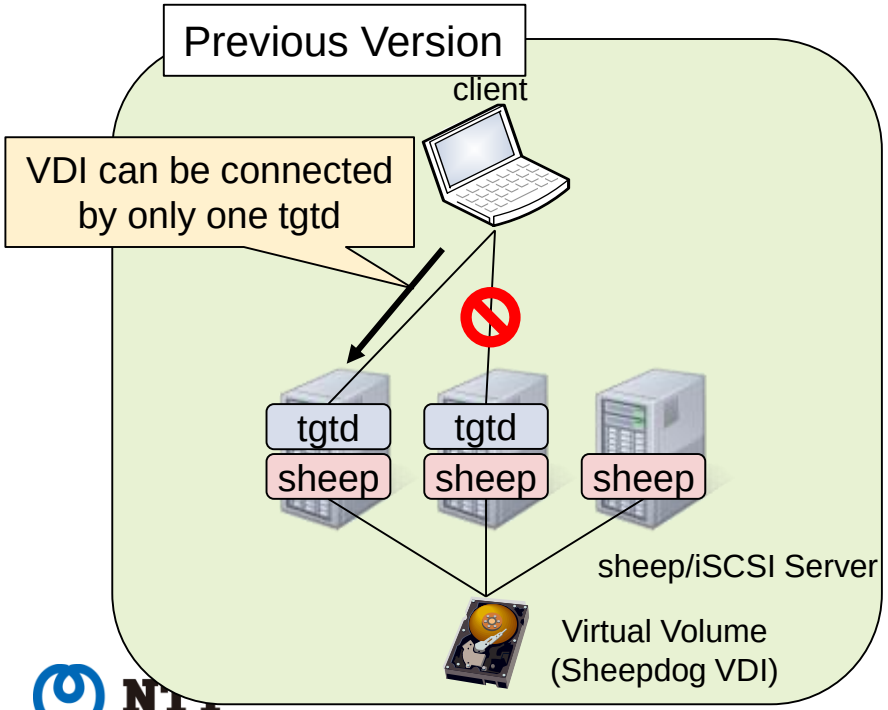
- **Sheepdog involves MSI Protocol for Inode Obj cache coherency between tgtds**
  - Modified
  - Shared
  - Invalidated
- **tgtd reloads Inode if node status of running tgtd is Invalidated**

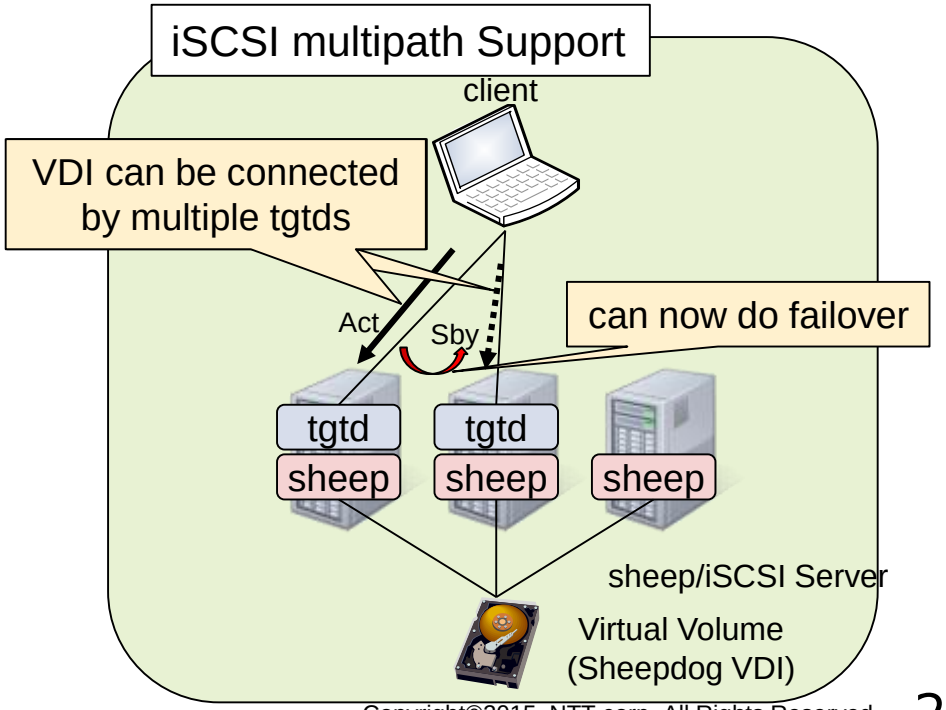


step/status	Server A	Server B	Server C
initial	(none)	(none)	(none)
Act connect (load inode)	modified	(none)	(none)
Sby connect (load inode)	shared	shared	shared
write to Act (write inode)	modified	invalidated	invalidated
read from SbyB (reload inode)	shared	shared	invalidated

No SPOF for using Sheepdog as iSCSI

- **Now we can use iSCSI multipath feature**
  - Support Act-Sby only
  - Client must use multipath-tool of Windows or Linux
  - Sheepdog v0.9.1 or later
  - tgtd v1.0.51 or later
  - option “-l” is needed for cluster format





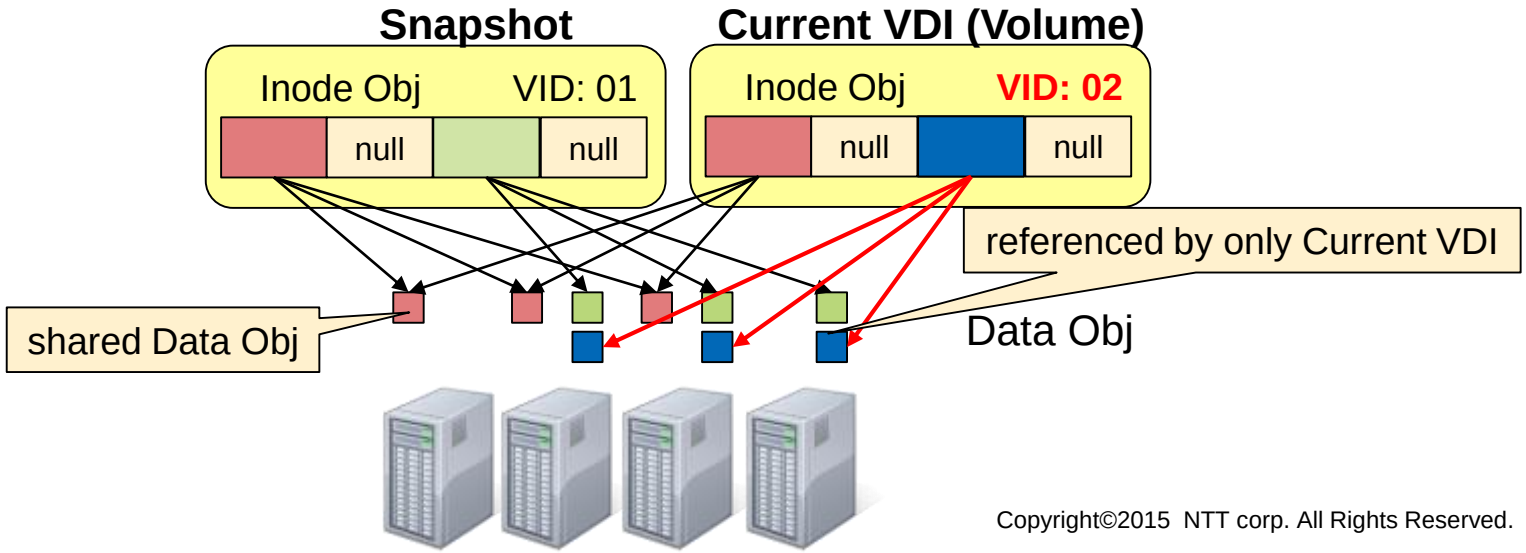
Developments for commercial use

We have developed for solving problems to satisfy ‘Availability’

- iSCSI multipath support (v0.9.1~)
- **Improved GC implementation of Data Obj (v0.9~)**

Difficulty of Data Obj's lifecycle

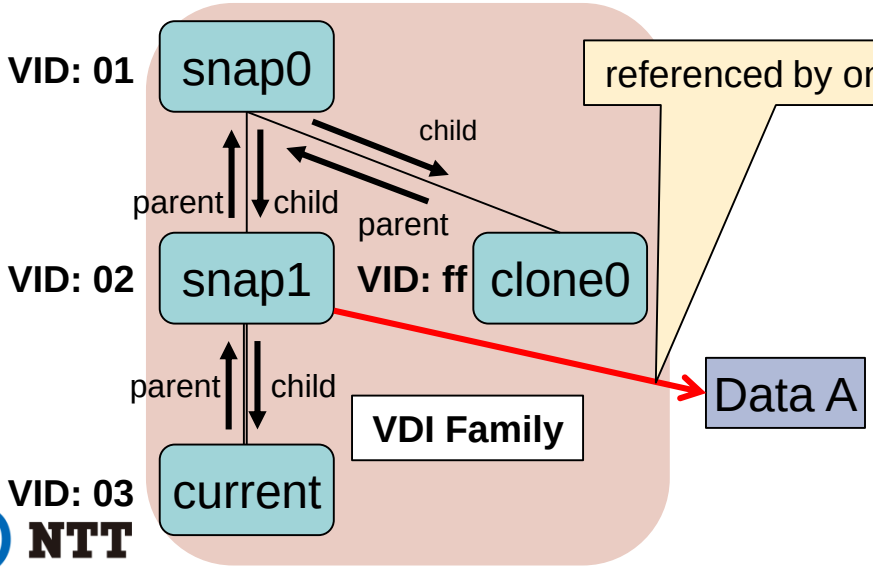
- **We must know whether Data Obj is referenced or not for deleting it**
  - Data Obj may be referenced by different VDIs of same family (snap/clone)
- **If each Data Obj has reference count simply, Snapshot/Clone operation will be very slow**
  - Need to increase reference count of all Data Objs pointed by base VDI and write all Data Objs to Disk



Sheepdog involves Generation Reference Count algorithm

- **Previous problem(Sheepdog v0.8)**
  - Data Objs could not be deleted until all members of VDI family were deleted

    → It wastes disk space
- **Sheepdog v0.9.0 or later**
  - Data Obj is deleted after all VDIs refering to it have deleted
  - Low cost for obtaining Snapshot/Clone
  - https://github.com/sheepdog/sheepdog/blob/master/doc/object-reclaim.txt



Sheepdog v0.8 or before
- Data A is deleted after All member of VDI family have deleted

Sheepdog v0.9 or later
- Data A is deleted after snap1 has deleted

Now Sheepdog is ready

Sheepdog satisfies all features for block storage and is ready for commercial use from v0.9.1 or later

- **Durability**
  - Replica/Erasure Code
- **Scalability**
  - Load Balance
  - I/O Node Scalability
- **Manageability**
  - Auto Operation
  - Volume Management
- **Availability**
  - No SPOF
  - Appropriate Volume's Data Lifecycle

Outline

1. What is Sheepdog?
2. What are required for distributed storage?
3. Developments for commercial use
4. **Performance test with SSD**
   1. FIO test for Qemu Block
   2. pgbench for Qemu Block
5. Use Cases
6. Conclusions

Test Environment

Innovative R&D by NTT

All test cases were done on the same Environment

- **Storage Software**
  - Sheepdog-0.9.1 (2 cases: 3 Replica, 4:2 Erasure Code)
  - Ceph-0.94.0 (1 case: 3 Replica)
- **Server Environment**
  - 9 Storage node cluster, connected by10GbE
  - 3 nodes of cluster are zookeeper (Sheepdog) or monitor(Ceph) servers
  - 1 Test Client VM is working in 1 node of cluster
  - each node has 1 HDD (OS/VM OS) and 1 SSD (Storage)
- **Test Client Environment**
  - KVM VM (qemu-2.2.1 + kernel-3.1.0-229.1.2.el7)
  - Client OS is CentOS7.1
  - Connect to Storage with own Qemu block driver

NTT

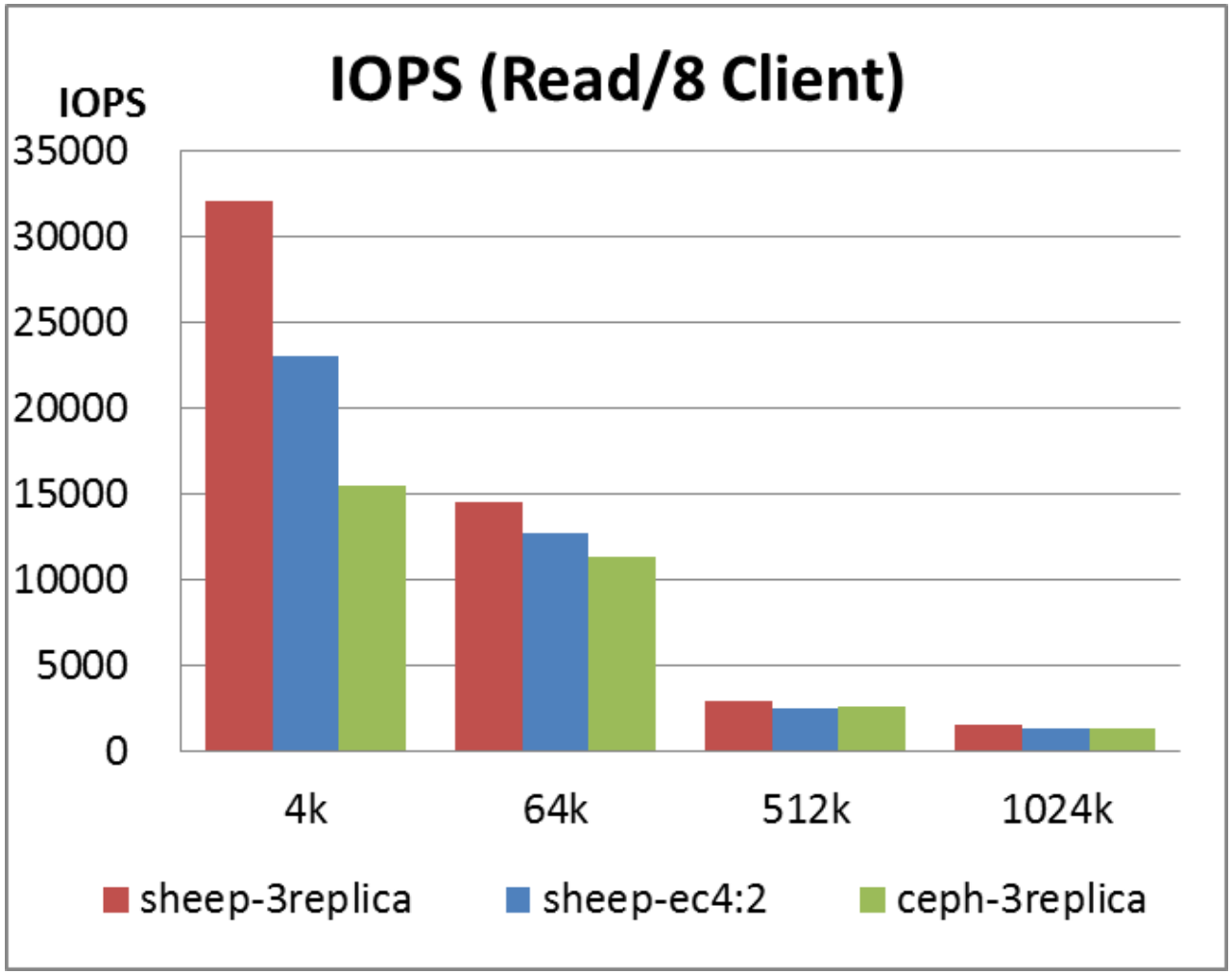
Performance comparison: FIO

- **FIO Base Setting: fio-2.2.6**
  - runtime: 60 sec
  - ramp_time: 10 sec
  - ioengine: libaio (asyncronous)
  - iodepth: 32
  - fsync: 0
  - direct: 1
  - invalidate: 1
  - numjobs: 8

- **Test Pattern**
  - sequential read/random read/sequential write/random write
  - block size: 4 KB, 64 KB, 512 KB, 1024 KB

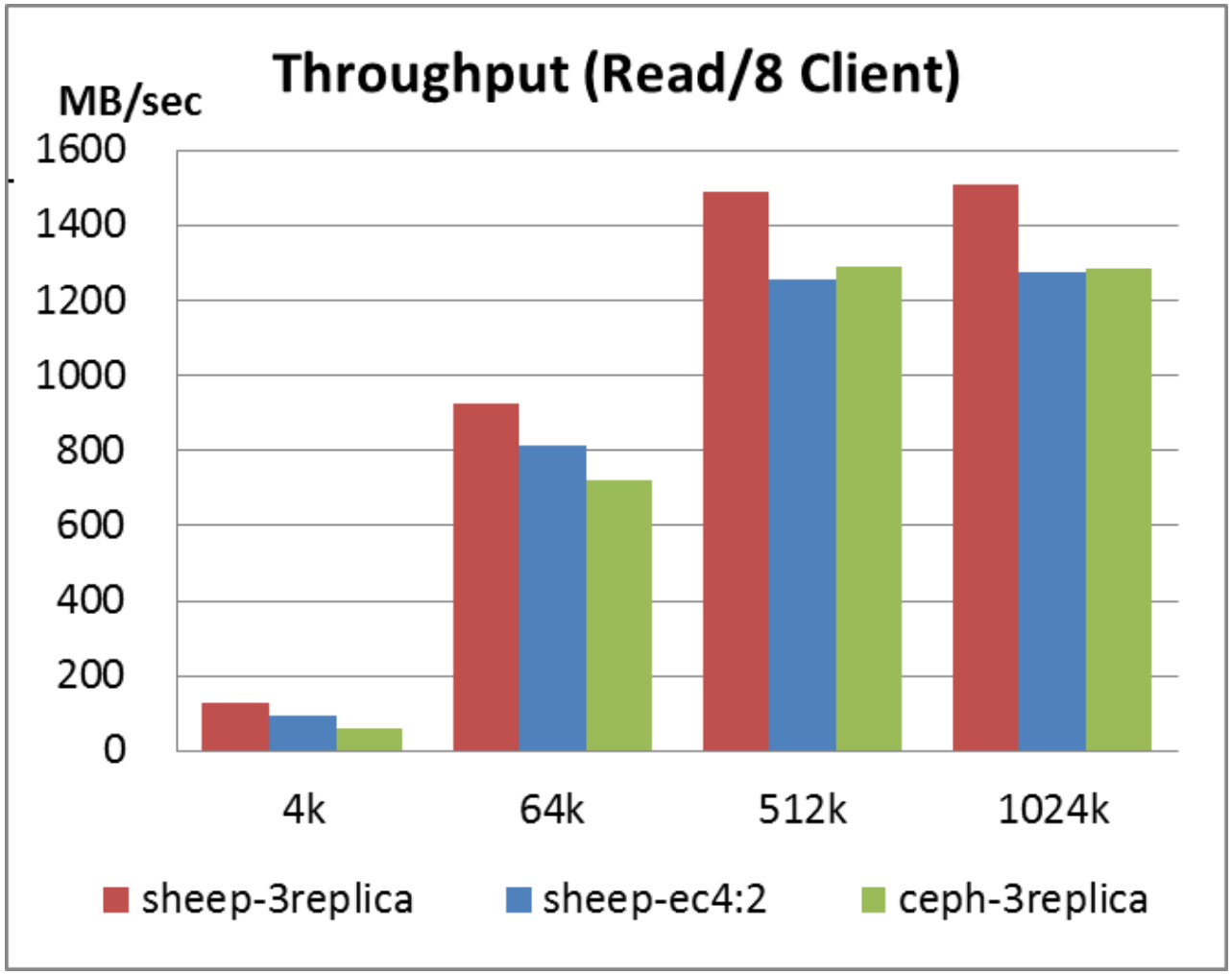
FIO test: Qemu Read IOPS

- change block size: 4 KB, 64 KB, 512 KB, 1024 KB



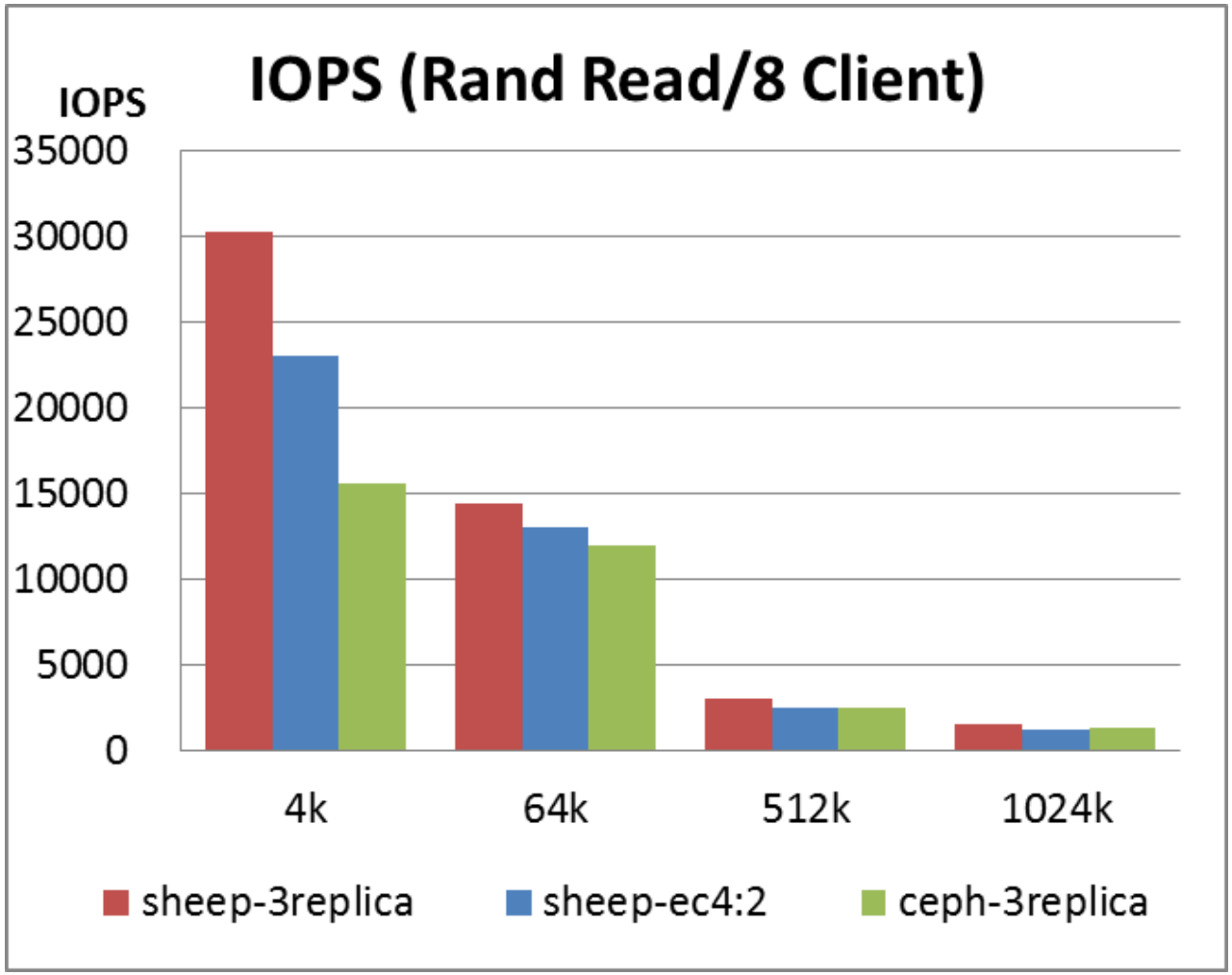
FIO test: Qemu Read Throughput

- change block size: 4 KB, 64 KB, 512 KB, 1024 KB



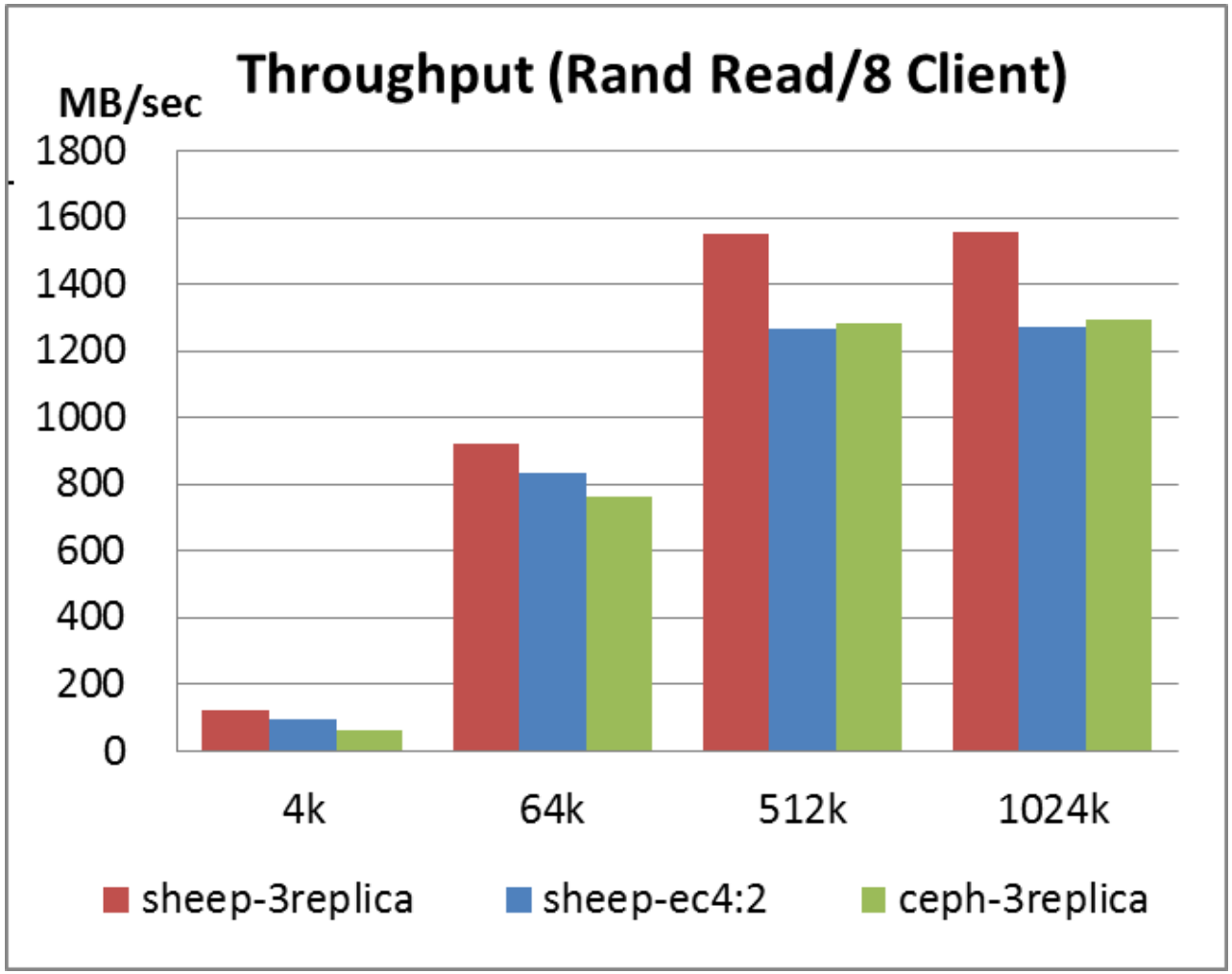
FIO test: Qemu Rand Read IOPS

- change block size: 4 KB, 64 KB, 512 KB, 1024 KB



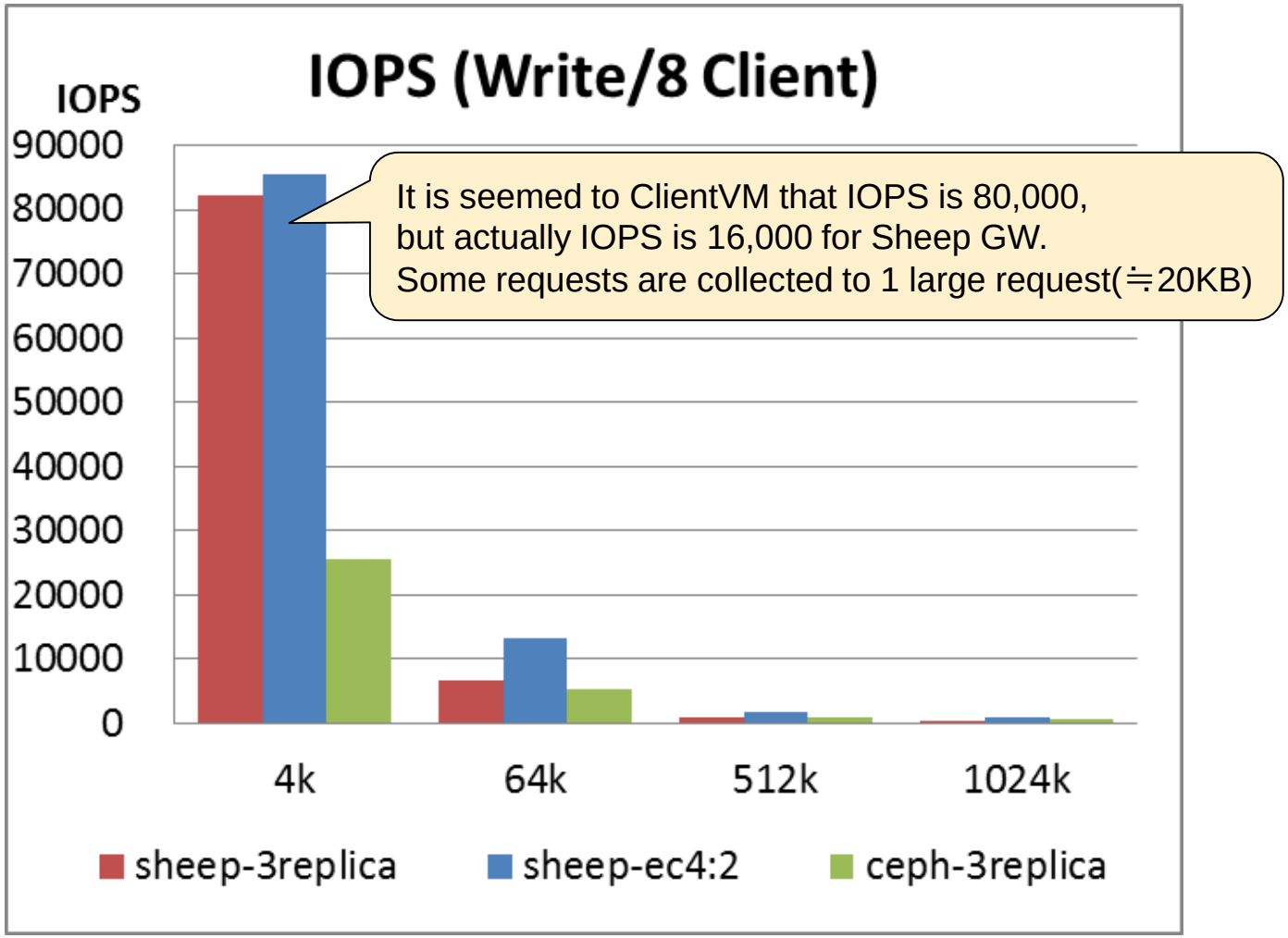
FIO test : Qemu Rand Read Throughput

- change block size: 4 KB, 64 KB, 512 KB, 1024 KB



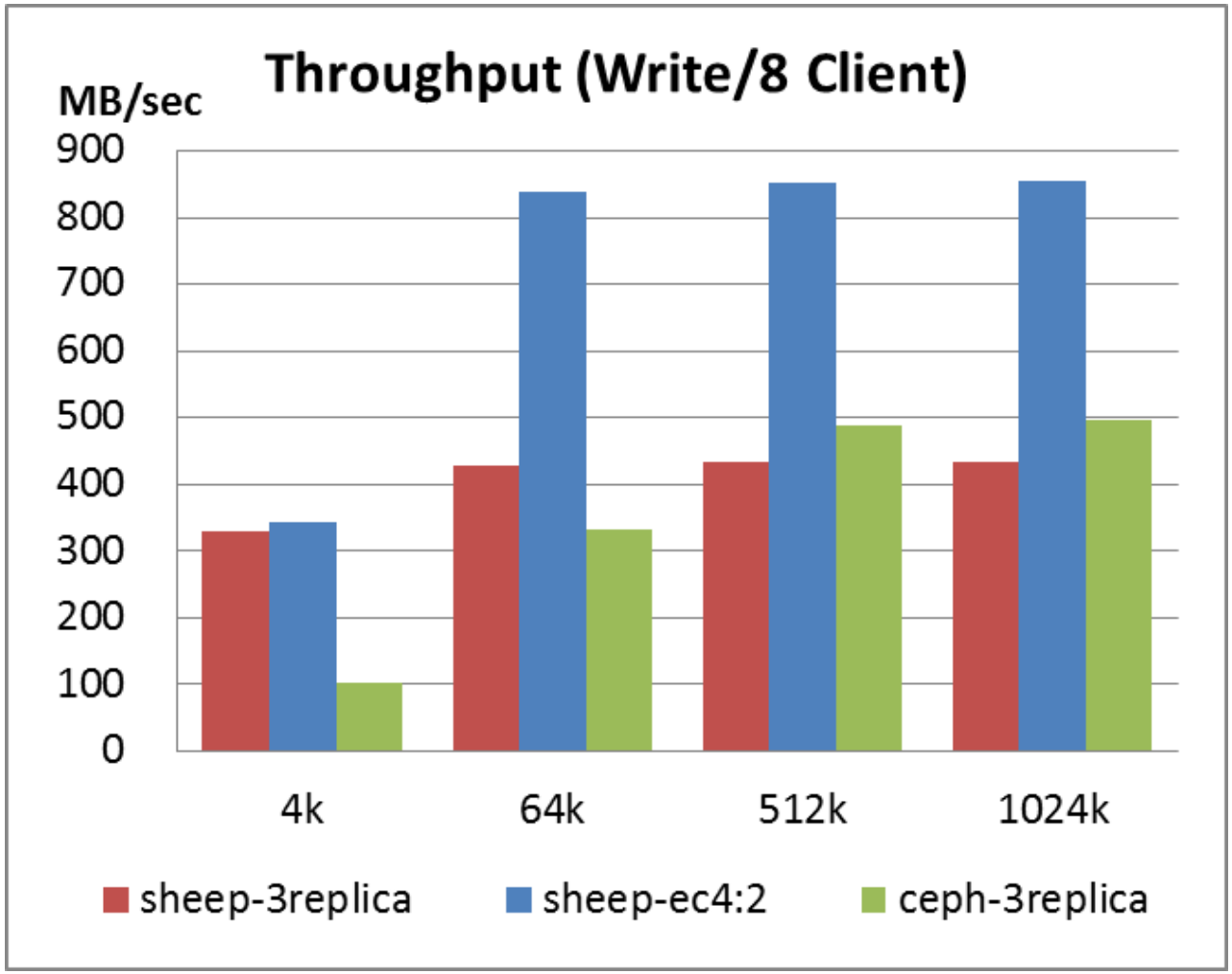
FIO test : Qemu Write IOPS

- change block size: 4 KB, 64 KB, 512 KB, 1024 KB



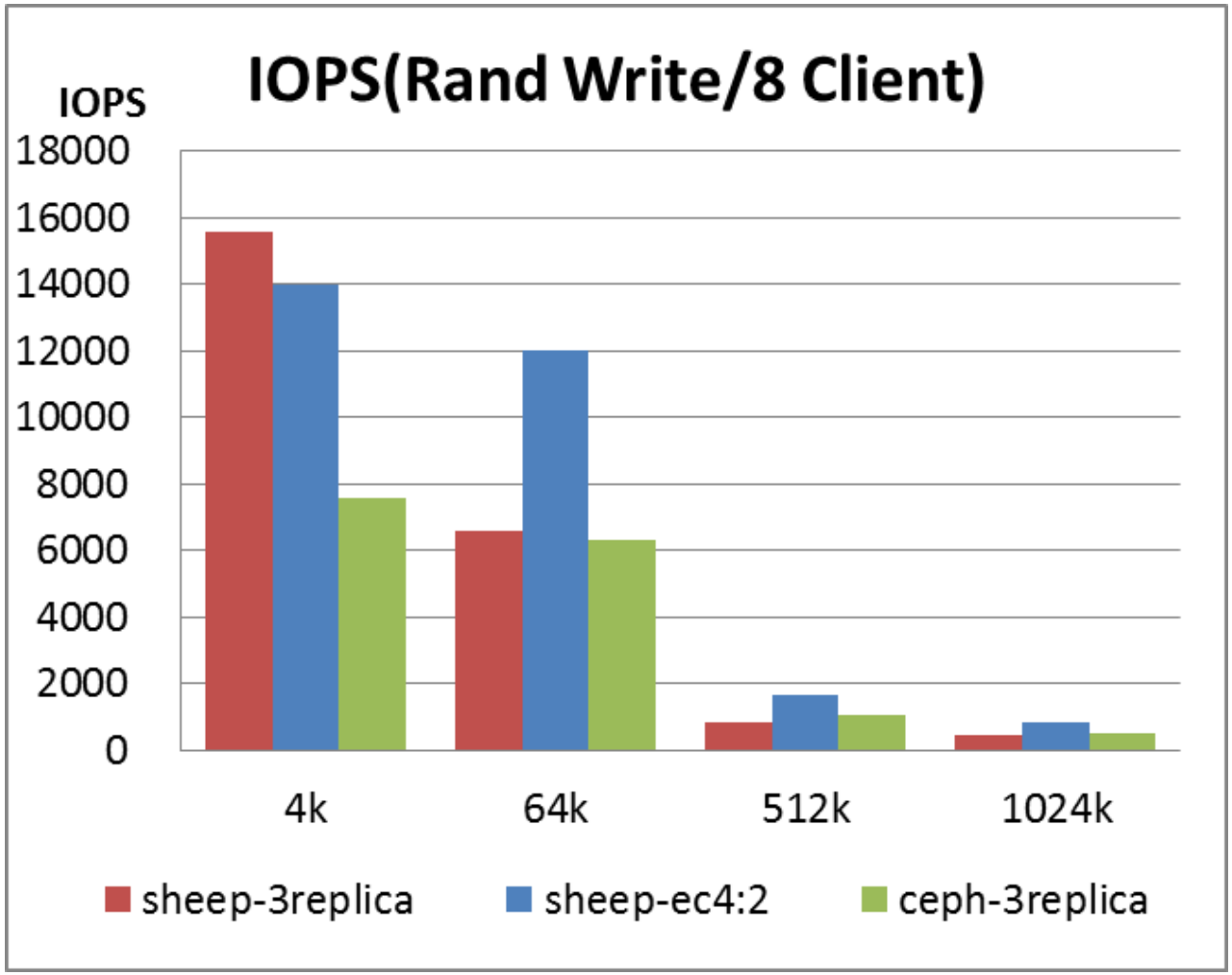
FIO test: Qemu Write Throughput

- change block size: 4 KB, 64 KB, 512 KB, 1024 KB



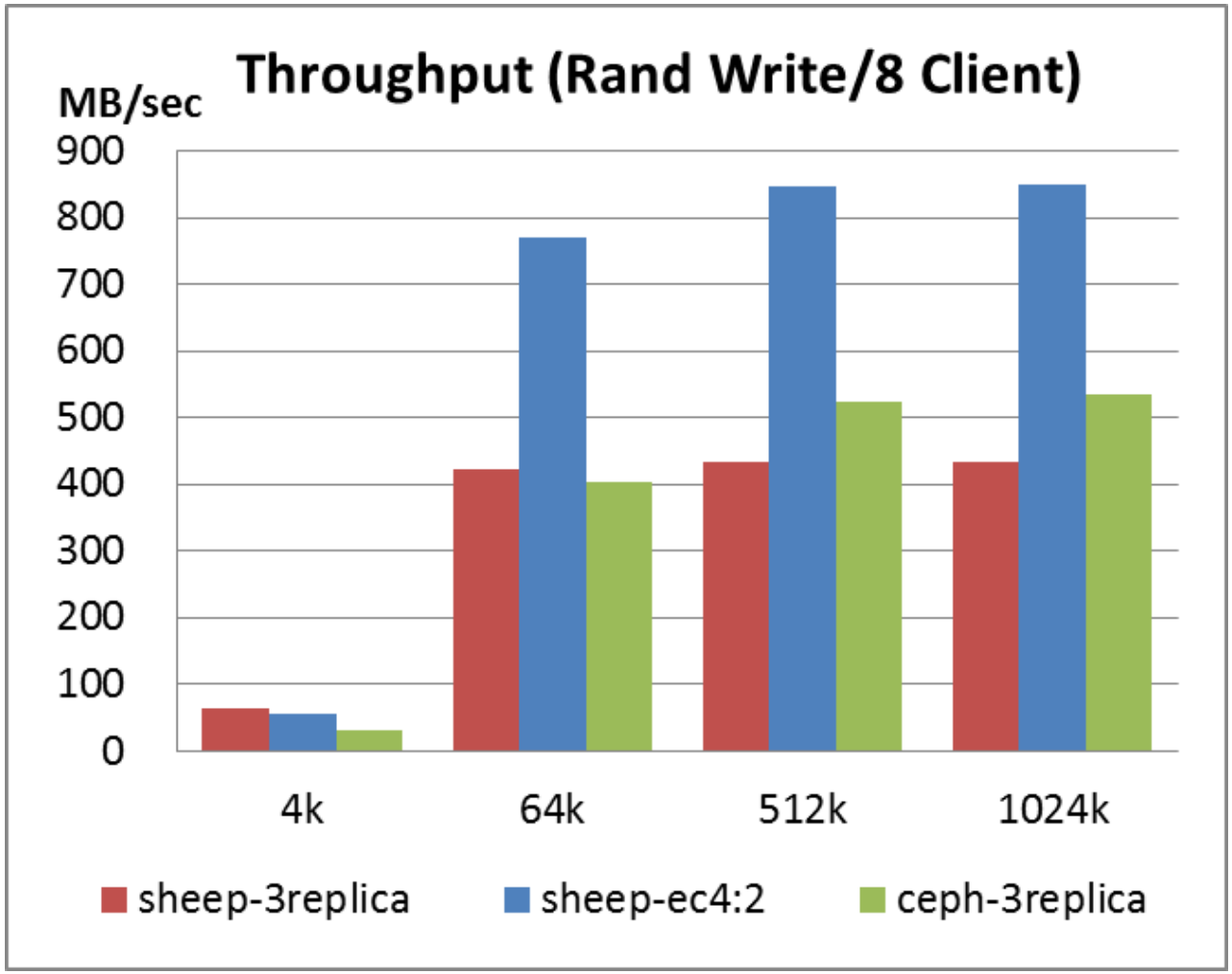
FIO test: Qemu Rand Write IOPS

- change block size: 4 KB, 64 KB, 512 KB, 1024 KB



FIO test: Qemu Rand Write Throughput

- change block size: 4 KB, 64 KB, 512 KB, 1024 KB



Performance Test Summary: FIO

The I/O pattern of all tests are asyncronous I/O

- **Sheepdog is superior in all cases of 4-KB blocks**
  ⇒ Sheepdog is better for R/W IOPS

- **Sheepdog-Replica is superior in all Read cases of large blocks**
  ⇒ Better read throughput
- **Sheepdog-EC is superior in all Write cases of large blocks**
  ⇒ Better write throughput

- **Ceph is better for Read/Write performance balance of large blocks**

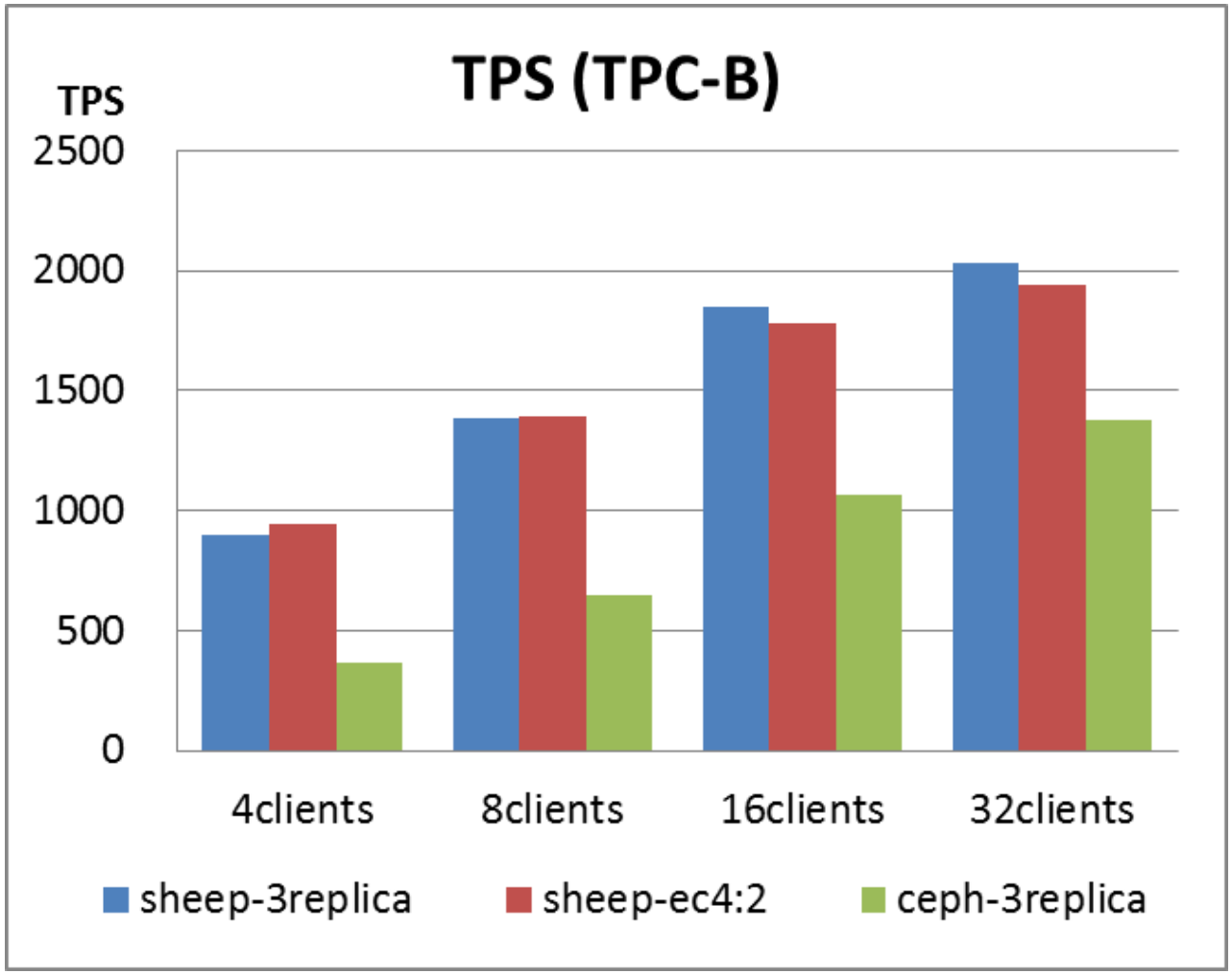
Performance comparison: pgbench

- **pgbench Base Setting: PostgreSQL-9.4.1**
  - max_connections: 100 (default)
  - shared_buffer: 256 MB
  - autovacuume: off

- **Test Pattern**
  - Preprocessing(initialization)
    # createdb pgbench
    # pgbench –I –s 2000 pgbench –q (scale factor: 2000)
  - Test Scenario
    - TPC-B
      - UPDATE -> SELECT -> UPDATE -> UPDATE -> INSERT
    - SELECT only
  - Client num: 4, 8, 16, 32
  - runtime: 300 seconds

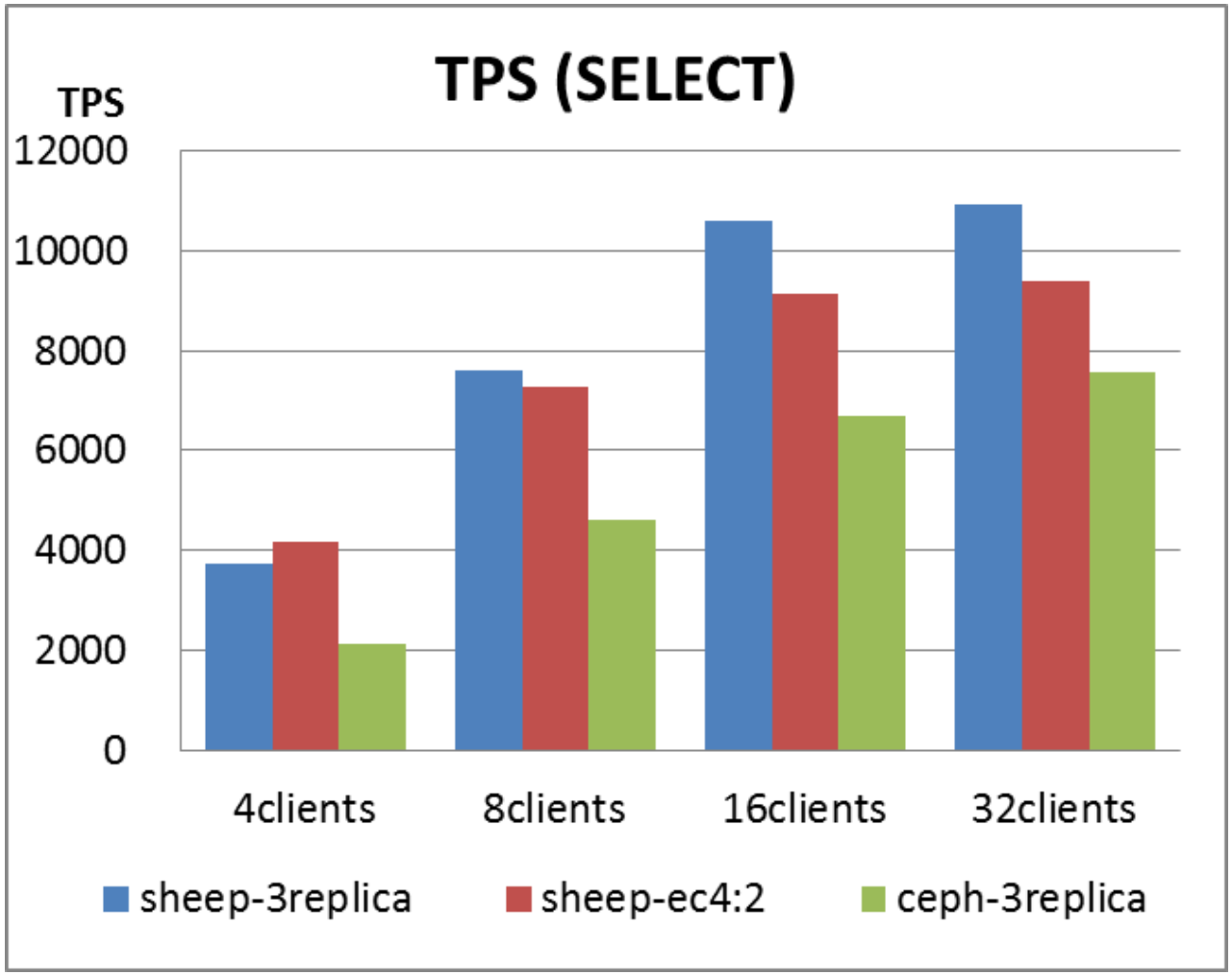
pgbench: Qemu TPC-B

- change client number: 4, 8, 16, 32



pgbench: Qemu SELECT

- change client number: 4, 8, 16, 32



Performance Test Summary: pgbench

- **Sheepdog VM volume works better than Ceph for TPC-B and SELECT work loads**

- **Pgbench Performance results are dependent on the environment, so, please verify the results on your environment**

Outline

1. What is Sheepdog?
2. What are required for distributed storage?
3. Developments for commercial use
4. Performance test with SSD
5. **Use Cases**
   - NTT Data Corporation
   - Alibaba Group
6. Conclusions

NTT Data Corporation

- ***NTT Data Open Solutions – Block Storage***
- **Use case: Block Storage Solution**
  - iSCSI Storage
  - OpenStack Block Storage – Qemu Block for Private Cloud
  - Product Summary (Japanese Page)
    - http://www.nttdata.com/jp/ja/news/release/2014/101401.html
- **First Commercial Product of Sheepdog in NTT Group**

Alibaba Group

- ***Lambert***
- **Use case: Cold Data Storage**
  - Sheepdog is used as Object Storage
  - Lambert consists of many Sheepdog clusters
  - External Tool handles Sheepdog clusters and manages cold data placement
  - Details of Lambert
    - http://events.linuxfoundation.org/sites/events/files/slides/LFVault2015_Alibaba.pdf
- **First Use Case for very large Object Storage**

Outline

1. What is Sheepdog?
2. What are required for distributed storage?
   Developments for commercial use
3. Performance test with SSD
4. Use Cases
5. **Conclusions**

Conclusions

- **Sheepdog is ready for commercial use**
  - Durability
  - Scalability
  - Manageability
  - Availability
- **It can be used for larger or more critical systems**
- **Sheepdog v1.0 is coming soon**
- **More information on Sheepdog**
  - web site: http://sheepdog.github.io/sheepdog
  - repository: https://github.com/sheepdog/sheepdog

Appendix

Hardware configuration

- **Machine: Fujitsu PRIMERGY RX200 S8 * 9**
  - Hostname
    - sds04 – sds12
  - CPU
    - Intel Xeon CPU E5-2620 v2 @ 2.10GHz
    - 2CPU*6core*HT=24 logical cores
  - Memory
    - DDR3 4GB 1600MHz * 8, 32GB total
  - HDD
    - SATA 250GB 7.2krpm * 3 (system area with RAID0)
    - Samsung 850pro SSD 250GB * 1 (data area)
  - NICs
    - Intel 82599ES SFP+ (10Gbps) for traffic of storage
    - Intel I350 (1Gbps) for management
  - RAID card
    - LSI MegaRAID SAS 2208
      - 1GB cache, 6Gbps bandwidth
      - Write through, no read ahead, Enable Drive Cache

FIO command parameter

- Pre-Create files for seq-read/rand-read test

```
# fio -rw=read -numjobs=8 –name=test -bs=1M -size=1G -fsync=0
      -direct=1 -invalidate=1 -ioengine=sync -iodepth=32
      -iodepth_batch=32 -group_reporting
      --output-format=json -directory=/sheepdog
      -create_only=1 -unlink=0
```

- Drop Page Cache for each Client VM and all Storage Nodes before tests

```
# echo 3 > /proc/sys/vm/drop_caches
```

- FIO test command

```
# fio -rw=${_RW} -bs=${_BLOCK_SIZE}
      -numjobs=8 -name=test -size=1G -fsync=0 -runtime=60
      -direct=1 -invalidate=1 -ioengine=libaio -iodepth=32
      -iodepth_batch=32 -group_reporting --output-format=json
      -directory=${MOUNT_PATH}
```

pgbench command parameter

- Create DB for benchmark

```
# ./createdb pgbench
# ./pgbench -i -s 2000 pgbench -q
```

- TPC-B senario test (change client number: 4, 8, 16, 32)

```
# ./pgbench -c ${_NUM_JOBS} -T 300 -r pgbench
```

- SELECT senario test (change client number: 4, 8, 16, 32)

```
# ./pgbench -S -c ${_NUM_JOBS} -T 300 -r pgbench
```

Sheepdog boot parameter

- Run on every node with different IP address

  # sheep -p 7000 -l dir=/var/log/sheep level=info
  -c zookeeper:192.168.2.17:2181,192.168.2.18:2181,192.168.2.19:2181
  -b <server_ip> -y <server_ip> /data/sheepdog

- Cluster initialization for erasure coding

  # dog cluster format –a 192.168.2.19 –c 4:2

- Cluster initialization for 3 replication

  # dog cluster format –a 192.168.2.19 -c 3

Configuration of ceph (ceph.conf)

```
[global]
fsid = 6744ad0a-061d-4822-be71-467e525b4643
mon_initial_members = sds12, sds11, sds10
mon_host = 192.168.2.22,192.168.2.21,192.168.2.20
auth_cluster_required = cephx
auth_service_required = cephx
auth_client_required = cephx
filestore_xattr_use_omap = true

osd_pool_default_size = 3
osd_pool_default_pg_num = 512
osd_pool_default_pgp_num = 512

public_network = 192.168.2.0/24
cluster_network = 192.168.2.0/24

[osd]
osd_journal_size = 1000

[osd.sds04]
public_addr = 192.168.2.14
cluster_addr = 192.168.2.14

[osd.sds05]
public_addr = 192.168.2.15
cluster_addr = 192.168.2.15

[osd.sds06]
public_addr = 192.168.2.16
cluster_addr = 192.168.2.16
```

```
[osd.sds07]
public_addr = 192.168.2.17
cluster_addr = 192.168.2.17
[osd.sds08]
public_addr = 192.168.2.18
cluster_addr = 192.168.2.18
[osd.sds09]
public_addr = 192.168.2.19
cluster_addr = 192.168.2.19
[osd.sds10]
public_addr = 192.168.2.20
cluster_addr = 192.168.2.20
[osd.sds11]
public_addr = 192.168.2.21
cluster_addr = 192.168.2.21
[osd.sds12]
public_addr = 192.168.2.22
cluster_addr = 192.168.2.22

[mon.sds12]
host = sds12
addr = 192.168.2.22:6789
[mon.sds11]
host = sds11
addr = 192.168.2.21:6789
[mon.sds10]
host = sds10
addr = 192.168.2.20:6789
```

Procedure of creating volume of Ceph (1/2)

```
# ceph-deploy new sds12 sds11 sds10
  modify ceph.conf
# ceph-deploy install sds04 sds05 sds06 sds07 sds08 sds09 sds10 sds11 sds12
# ceph-deploy mon create-initial
# ceph-deploy mon create sds12 sds11 sds10
# ceph-deploy gatherkeys sds12 sds11 sds10

# ceph-deploy osd prepare sds12:/data/ceph/osd sds04:/data/ceph/osd sds05:/data/ceph/osd
                          sds06:/data/ceph/osd sds07:/data/ceph/osd sds08:/data/ceph/osd
                          sds09:/data/ceph/osd sds10:/data/ceph/osd sds11:/data/ceph/osd
# ceph-deploy osd activate sds04:/data/ceph/osd sds05:/data/ceph/osd sds06:/data/ceph/osd
                           sds07:/data/ceph/osd sds08:/data/ceph/osd sds09:/data/ceph/osd
                           sds10:/data/ceph/osd sds11:/data/ceph/osd sds12:/data/ceph/osd
# ceph-deploy admin sds12 sds04 sds05 sds06 sds07 sds08 sds09 sds10 sds11

 (for each server)
# ssh sds04 chmod +r /etc/ceph/ceph.client.admin.keyring

# ceph osd pool create ceph-qemu 512 512 replicated
# ceph osd pool set ceph-qemu size 3

# ceph osd tree

# ceph auth get-or-create client.libvirt mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx
pool=ceph-qemu‘

(cont…)
```

Procedure of creating volume of Ceph (2/2)

```
# cat > secret.xml <<EOF
<secret ephemeral='no' private='no'>
      <usage type='ceph'>
            <name>client.libvirt secret</name>
      </usage>
</secret>
EOF

# virsh secret-define --file secret.xml
⇒ UUID for secret info is out (437e43e1-7b0b-429c-bb34-78f07d859d75)

# ceph auth get-key client.libvirt | tee client.libvirt.key
# virsh secret-set-value --secret 437e43e1-7b0b-429c-bb34-78f07d859d75 --base64 $(cat client.libvirt.key)

# cat > ceph-volume.xml <<EOF
<disk type='network' device='disk'>
  <driver name='qemu' cache='none'/>
  <auth username='libvirt'>
    <secret type='ceph' uuid='437e43e1-7b0b-429c-bb34-78f07d859d75'/>
  </auth>
  <source protocol='rbd' name='ceph-qemu/sds12-vm01'>
    <host name='sds12' port='6789'/>
  </source>
  <target dev='vdb' bus='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
</disk>
EOF
# virsh attach-device testVM ceph-volume.xml
```

Libvirt XML for Test Volume

Innovative R&D by NTT

- Sheepdog Volume

```
<disk type='network' device='disk'>
  <driver name='qemu' cache='none'/>
  <source protocol='sheepdog' name='sds12-vm01'><host name='192.168.2.22' port='7000'/></source>
  <target dev='vdb' bus='virtio'/><address type='pci' domain='0x0000' bus='0x00' slot='0x07'
function='0x0'/>
</disk>
```

- Ceph Volume

```
<disk type='network' device='disk'>
  <driver name='qemu' cache='none'/>
  <auth username='libvirt'>
    <secret type='ceph' uuid='437e43e1-7b0b-429c-bb34-78f07d859d75'/>
  </auth>
  <source protocol='rbd' name='ceph-qemu/sds12-vm01'>
    <host name='sds12' port='6789'/>
  </source>
  <target dev='vdb' bus='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
</disk>
```

NTT