Innovative R&D by NTT

sheepdog: software defined storage system for converged infrastructure

Hitoshi Mitake
NTT Software Innovation Center

Outline

1. The trend of SDS and sheepdog
2. Recent development status
3. Performance comparison with Ceph and GlusterFS
4. Introduction of use cases
5. Conclusion

Outline

1. **The trend of SDS and sheepdog**
   - What does “SDS” mean in the context of this presentation?
   - And how does sheepdog relate to it?
2. **Recent development status**
3. **Performance comparison with Ceph and GlusterFS**
4. **Introduction of use cases**
5. **Conclusion**

The trend of Software Defined Storage

- **Many startups and giants are saying "Software Defined Storage"**
  - what is this?
- **4 weeks before, Red Hat, Inc acquired Inktank for $175 million**
  - seems to be hot

Red Hat to Acquire Inktank, Provider of Ceph

Raleigh

April 30, 2014

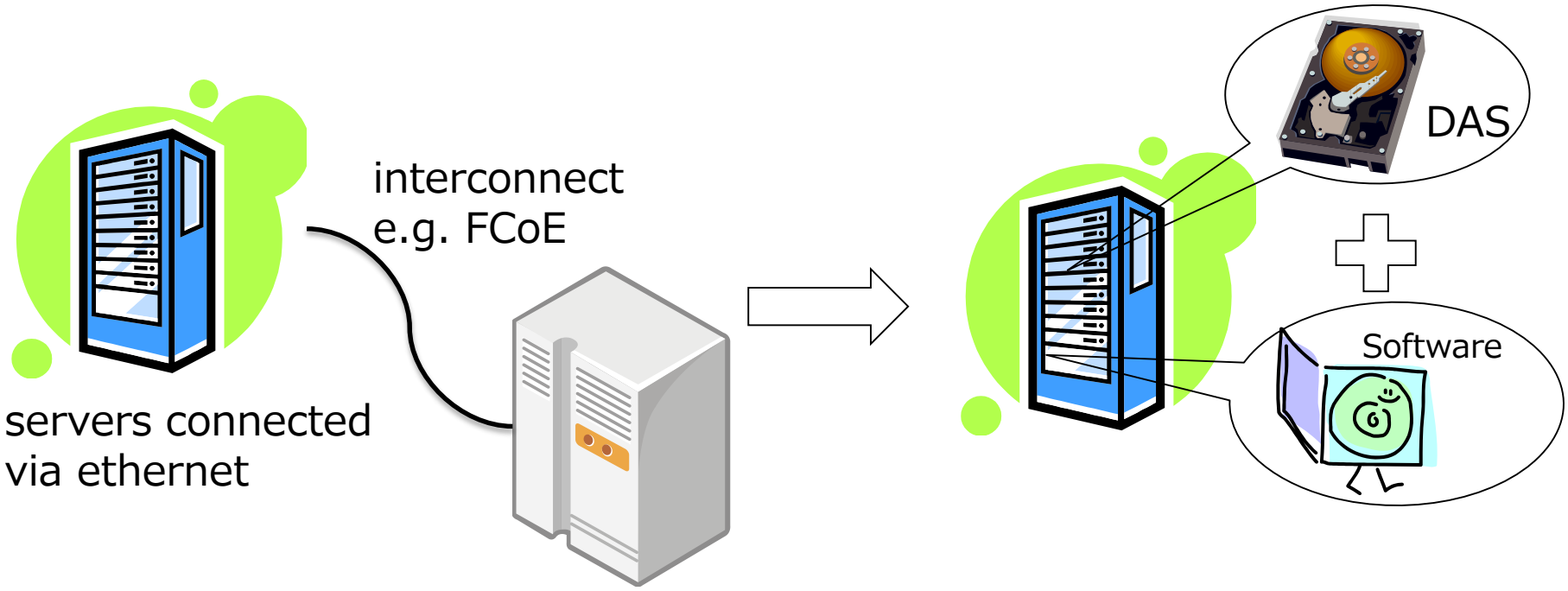
- Open source leaders unite to accelerate adoption of open software-defined storage for enterprises and service providers
- Addition of Inktank positions Red Hat as leading provider of open software-defined storage across object, block, and file system storage

software defined storage?

http://www.redhat.com/about/news/press-archive/2014/4/red-hat-to-acquire-inktank-provider-of-ceph

The trend of Software Defined Storage

- **The trend of throwing out traditional storage appliances from data centers**
  - Instead of appliances, software defined storages utilize DAS in commodity servers and implement functionalities of the appliances.



The trend of Software Defined Storage

- **Why should we through out the appliances?**
  - Because they need careful capacity planning
    - This limitation comes from their inflexibility
      - Storage is just storage even if it is idle (cannot used for other computation)
      - Expanding capacity of the appliances is virtually impossible
    - Capacity planning is difficult
      - Risk of low utilization: your investment would go to waste
      - Risk of resource exhaustion: your service wouldn't work well

How much storage will my service require during its whole lifetime?

The trend of Software Defined Storage

Innovative R&D by NTT

Two types of SDS products

1. Converged infrastructure
   - co-existing VMs and storage on single hardware
   - e.g. Nutanix, Simplivity OmniCube, Cohodata Difference, VMware virtual SAN, EMC ScaleIO, Maxta MxSP, etc…

2. Scale-out SAN
   - storage which is separated from hypervisor (or bare metal OS)
   - e.g. Nimble storage, HP 3Par, HP LeftHand, IBM XIV, Dell EqualLogic, PureStorage, EMC XtreamIO, etc…

NTT

The trend of Software Defined Storage

Innovative R&D by NTT

Two key properties of the SDS products

(Virtually) every provider says their products are:

1. Scale-out
   - You can increase performance and capacity when you add a new node to your cluster
2. Easy to manage
   - Procedure of adding a new node doesn't depend on size of your cluster
   - Difficulty isn't changed even your cluster has 10 nodes or 100 nodes

Definition of SDS in this talk: storage system which can satisfy the above two properties

NTT

Sheepdog: SDS in the world of OSS

Innovative R&D by NTT

- **Distributed storage software for providing block devices of QEMU**
  - Since 2009
- **Design policies**
  - Providing functionalities of typical storage appliances
    - live snapshot, writable snapshot (cloned disk), thin provisioning, etc
  - Symmetric, scale-out architecture
    - no special nodes (e.g. metadata server)
    - difficulty of adding a new node doesn't depend on a number of existing nodes of cluster

NTT

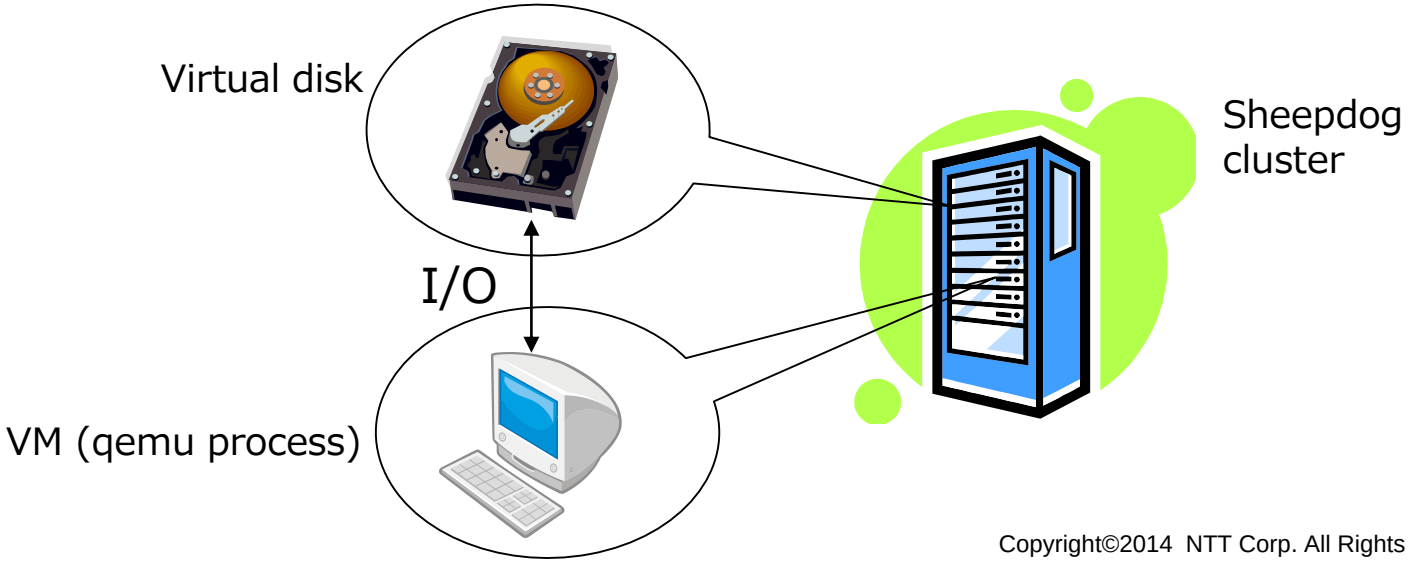
Sheepdog: SDS in the world of OSS

- **Using sheepdog from QEMU**

/* dog is a command for managing sheepdog cluster */
$ *dog vdi create my-new-disk 1TB*

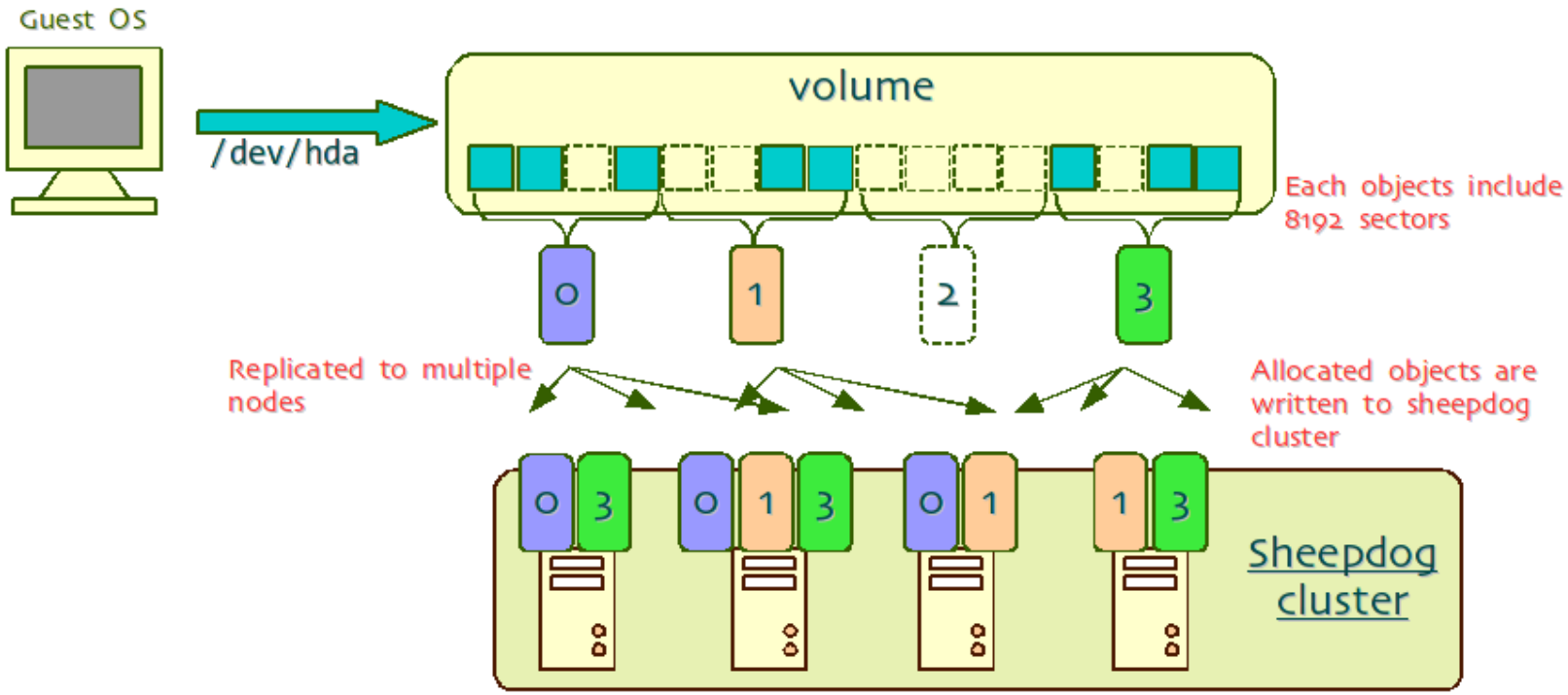
/* launch VM after installing OS. disks can be used like qcow2 */
$ *qemu-system-x86_64 –enable-kvm –hda sheepdog:my-new-disk*

/* you can create snapshot without killing qemu process */
$ *dog vdi snapshot my-new-disk*



How sheepdog store data: representation

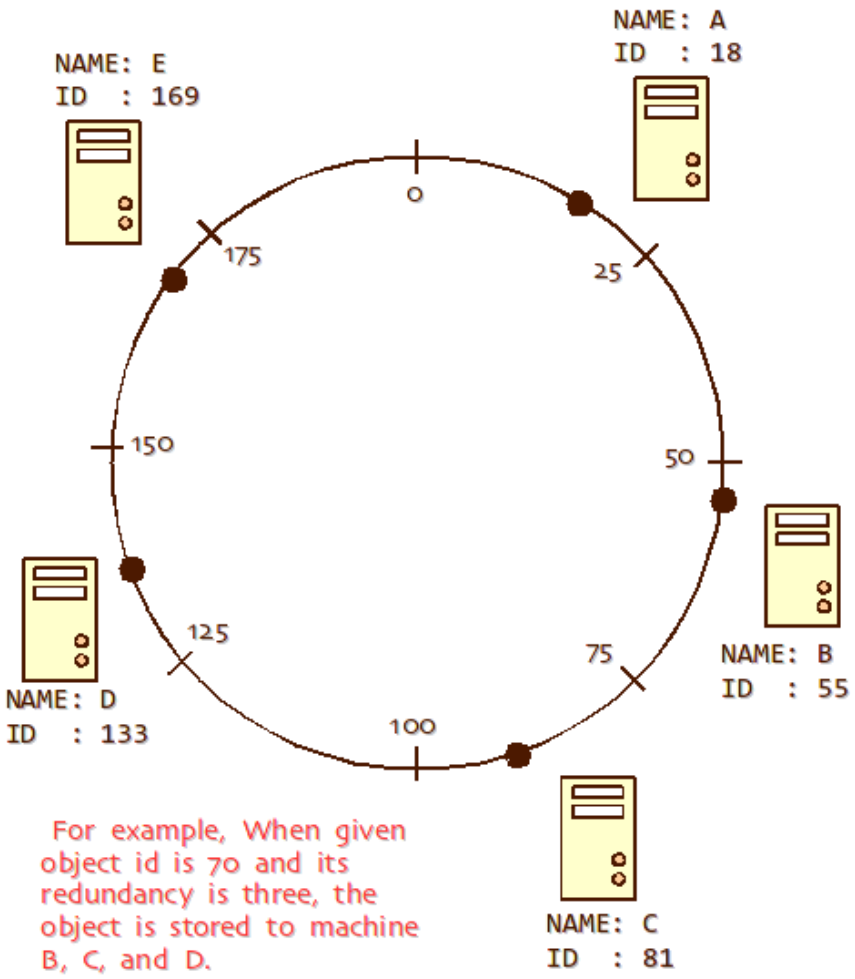
- **Virtual disks are stored as objects**
  - Disks are divided into fixed-size objects
  - Internal of sheepdog is a key-value store (e.g. Amazon's Dynamo)



How sheepdog store data: object placement

- **Object placement policy: consistent hashing**
  - A node for storing an object is determined by its ID
  - No metadata server: the policy contributes to the scale-out architecture

You can build the converged infrastructure and scale-out SAN based on sheepdog without any proprietary software

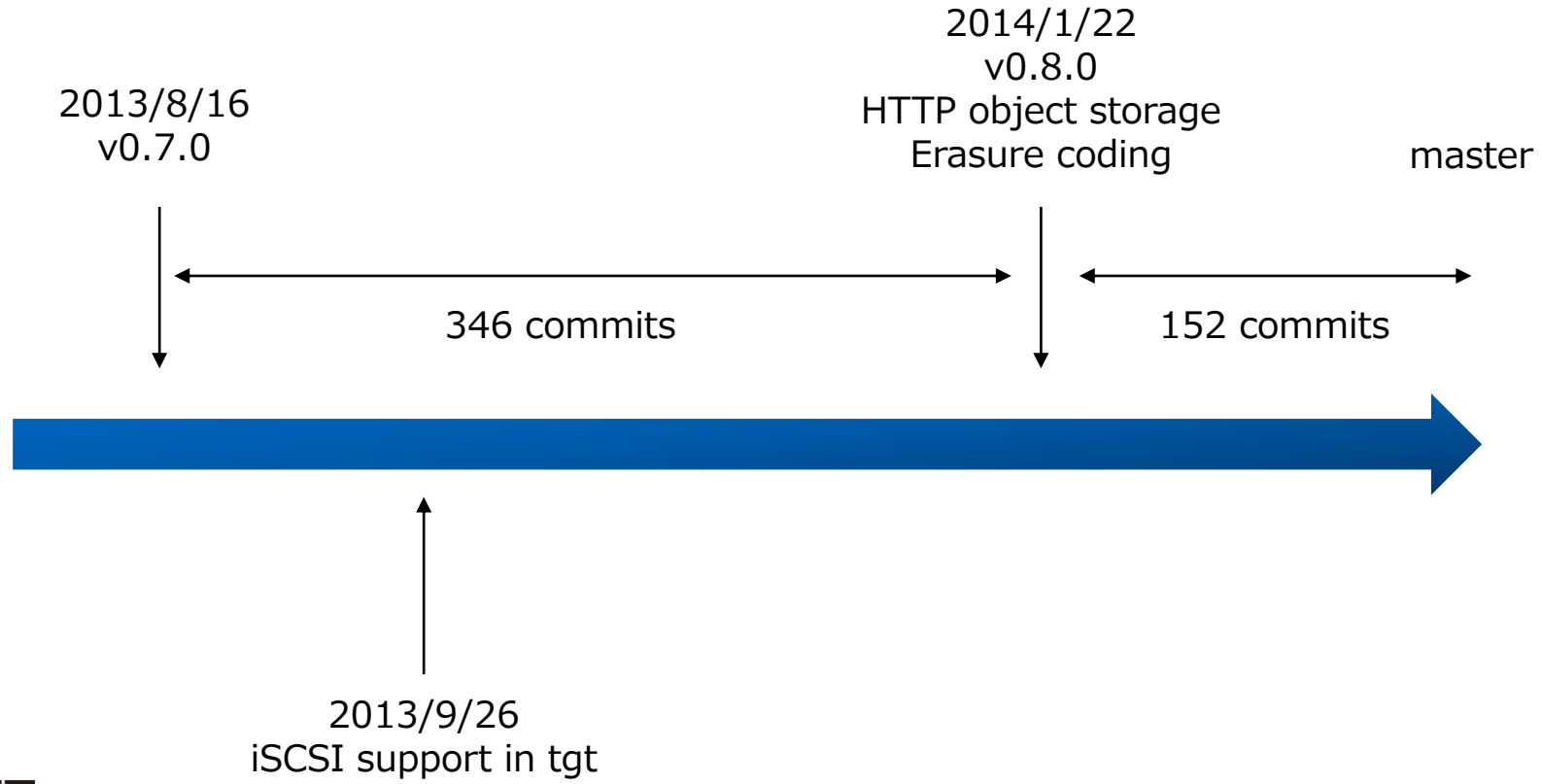


Outline

1. The trend of SDS and sheepdog
2. **Recent development status**
   - Erasure coded VDI
   - iSCSI interface support
   - Object storage support
3. Performance comparison with Ceph and GlusterFS
4. Introduction of use cases
5. Conclusion

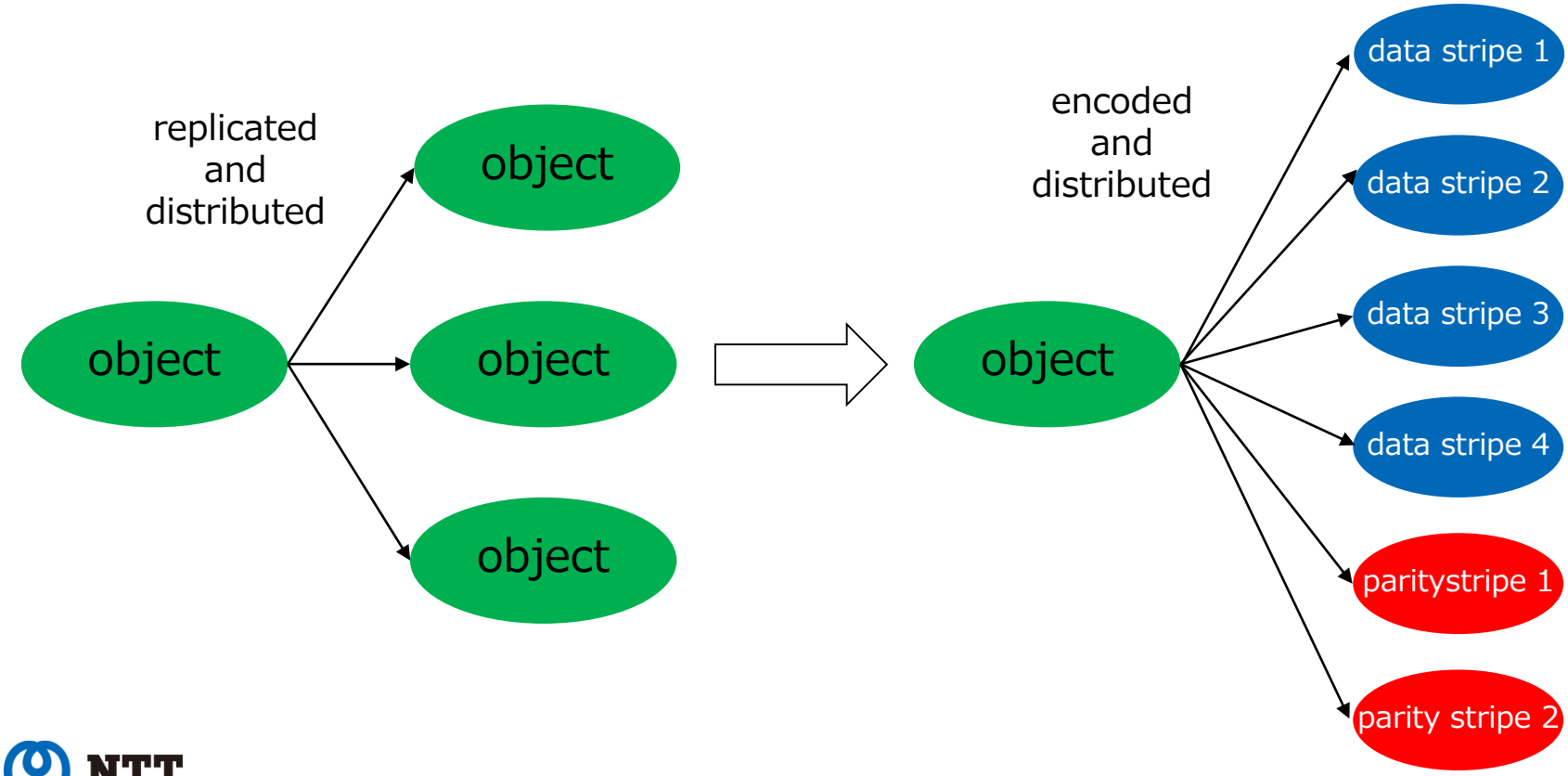
Recent development history

- **Many new features and bunch of bugfixes**
- **Stable releases for long-term support is provided**
  - 9 releases for v0.7.x
  - 2 releases for v0.8.x



Erasure coded VDI

- **New data storing policy**
  - An alternative of naïve replication
  - Objects are encoded to data stripes and parity stripes



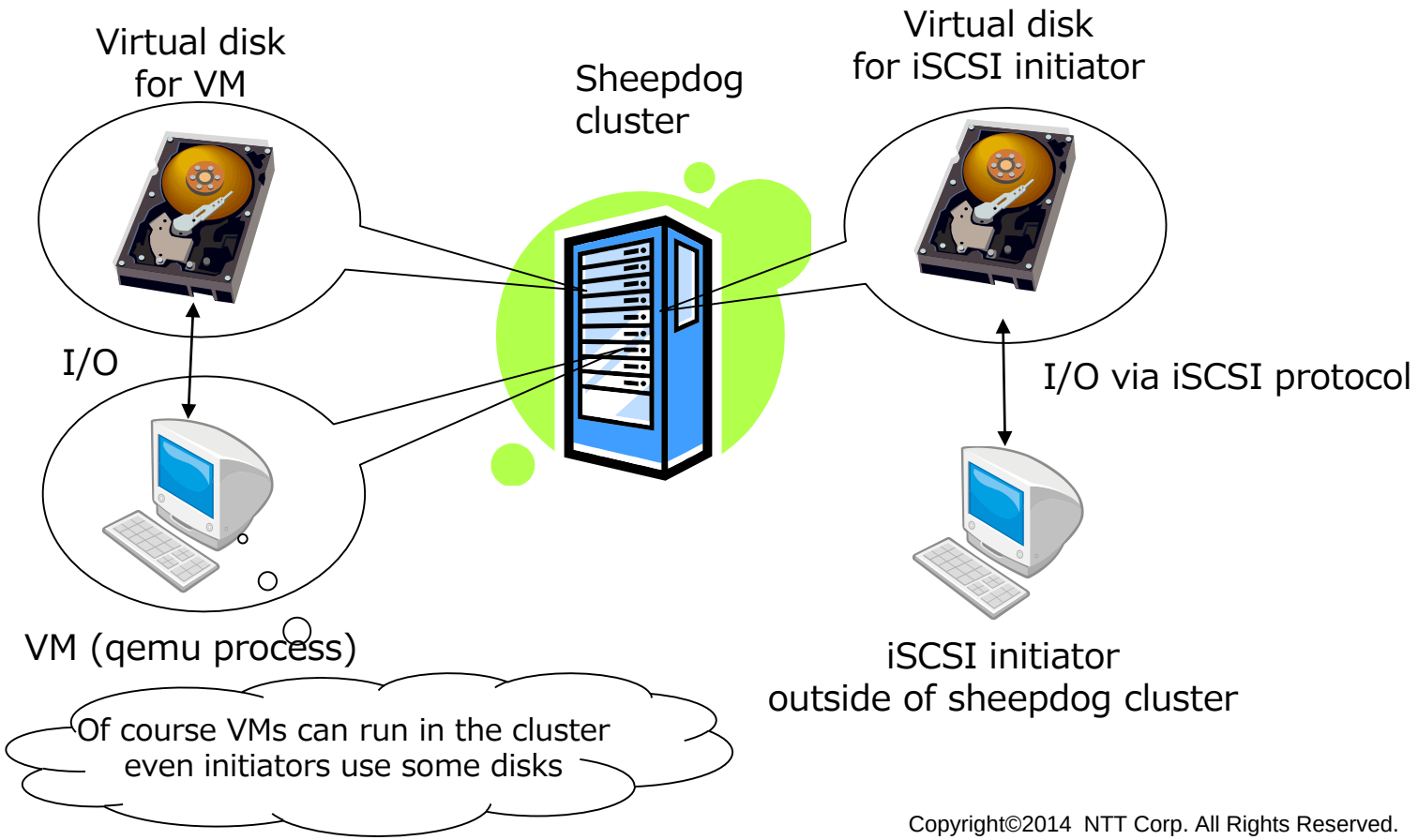
Erasure coded VDI

- **Pros of erasure coding**
  - Improving write performance and space efficiency without degrading durability
    - Also improve read performance in some cases
- **Cons of erasure coding**
  - Degrading read performance in some cases
    - But it can be covered by caching
  - Consume CPU resource for encoding and decoding

- **Coexistence of replication schemes are allowed**
  - Performance comparison of naïve replication and erasure coding is presented later

iSCSI interface support

- **Sheepdog is supporting iSCSI protocol**
  - Any iSCSI initiators (hypervisors other than KVM, bare metal OSes) can use virtual disks of sheepdog



iSCSI interface support

- **iSCSI support is implemented as a backing store driver of tgt: Linux scsi target framework**
  - http://stgt.sourceforge.net/

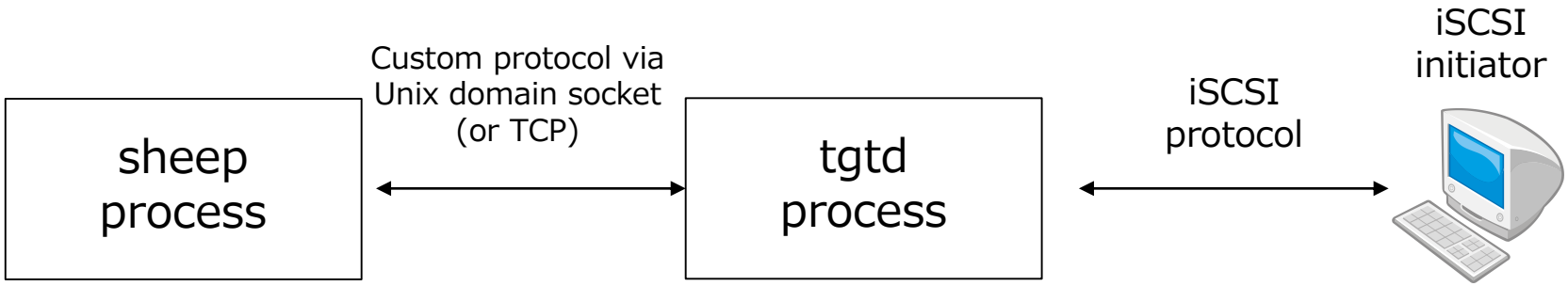
/* creating disks for iSCSI requires no special step */
$ dog vdi create my-new-disk 1TB

/* use the disk as a logical unit of tgtd in a similar way of other backing stores */
$ tgtd
$ tgtadm --op new --mode target --tid 1 --lld iscsi --targetname ¥
iqn.2013-10.org.sheepdog-project
$ tgtadm --op new --mode lu --tid 1 --lun 2 --bstype sheepdog ¥
--backing-store unix:/sheep_store/sock:my-new-disk

Instruction of setup iSCSI target:
https://github.com/sheepdog/sheepdog/wiki/General-protocol-support-(iSCSI-and-NBD)

iSCSI interface support

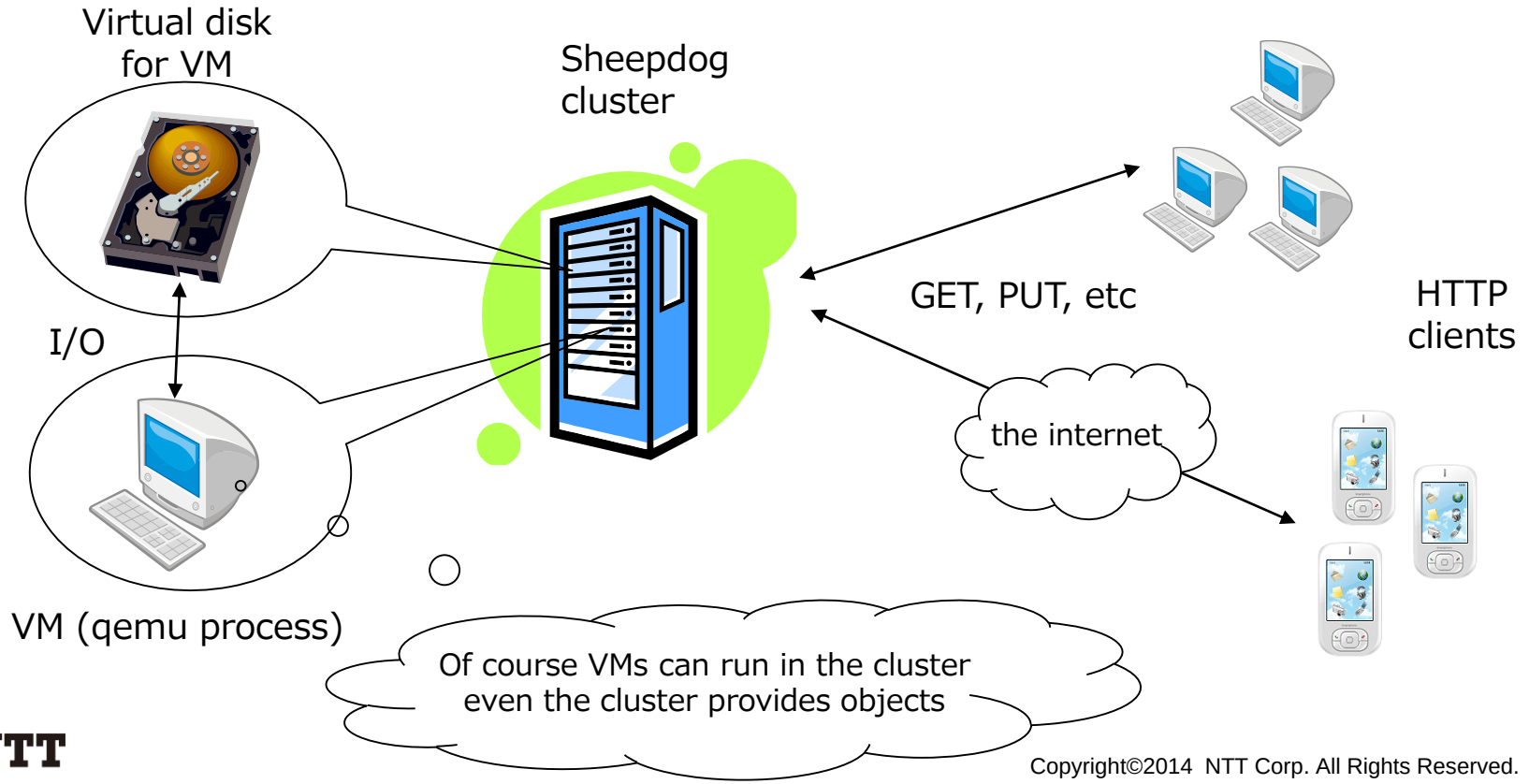
- **Relation of sheepdog, tgt and iSCSI initiator**



- **Who should care about this feature**
  - Non KVM users (e.g. VMware)
  - Owners of machines which have virtualization unfriendly facilities
    - GPUs, many-core based accelerators, etc
  - Owners of non green-field projects
    - e.g. you already have clusters for VMs and storage appliances, and need to migrate them gradually

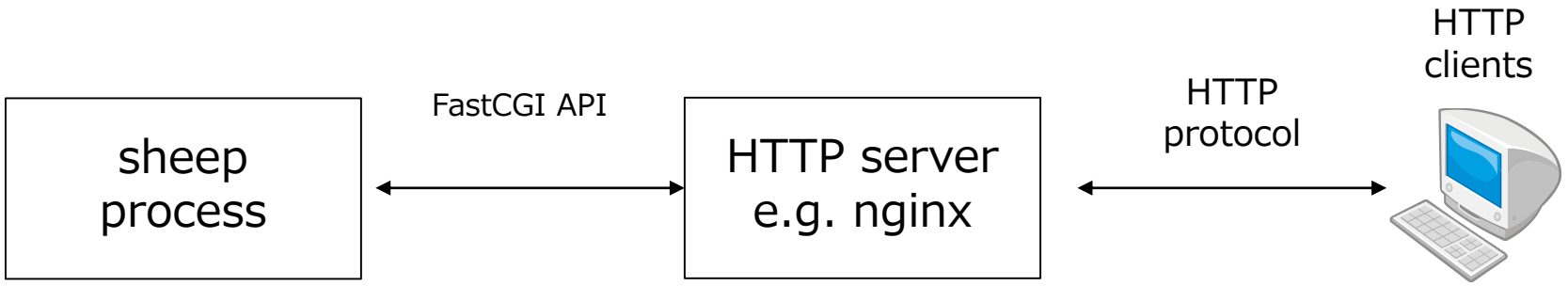
Object storage support

- **Object storage via HTTP interface is supported**
  - account/container/object style interface like AWS S3 and OpenStack Swift
  - suitable for storing static blob contents (e.g. photos, videos)



Object storage support

- **Benefit of providing both of disks and objects in a single cluster**
  - You don’t have to prepare two dedicated clusters for each purpose
  - Entire disk space of cluster can be used for both of disks and objects
    - Contribute to high utilization of resource



Instruction of setup object storage:
https://github.com/sheepdog/sheepdog/wiki/HTTP-Simple-Storage

Outline

1. The trend of SDS and sheepdog
2. Recent development status
3. **Performance comparison with Ceph and GlusterFS**
   - KVM: as a foundation of converged infrastructure
   - tgt: as a scale-out SAN
4. Introduction of use cases
5. Conclusion

Performance comparison

- **Environment**
  - 12 nodes cluster, connected via 10GbE
- **Workload**
  - fio with 16 pattern
    - sequential read, sequential write, random read, random write
    - block sizes: 4MB, 1MB, 512KB, 4KB
- **Assumed failure**
  - 2 node failure at the same time
- **Versions of storage software**
  - Sheepdog: 0.8.1
  - Ceph: 0.72.1
  - GlusterFS: 3.4.3-3

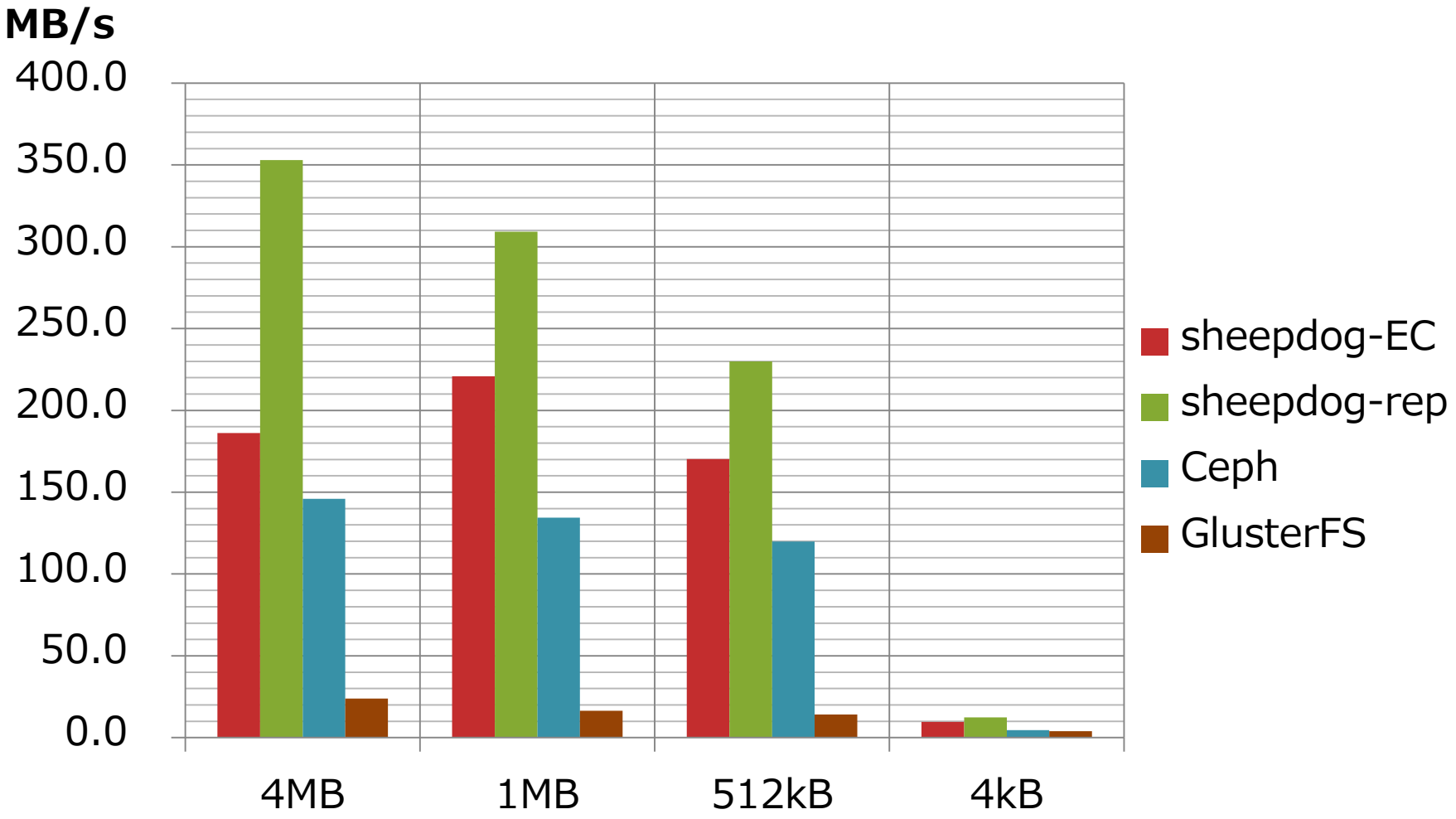
(more detailed information is described in appendix)

Performance comparison: KVM

- **Configurations for the case of KVM**
  - QEMU version: 1.7.1
  - 4 nodes of the cluster run 3 VM
    - 8 nodes are for storage only
    - Average score of the 12 VMs is presented (5 times run)
  - Sheepdog
    - 3 replicas and 4:2 erasure coding
  - Ceph
    - 3 replicas
  - GlusterFS
    - 3 replicas + 4 stripes

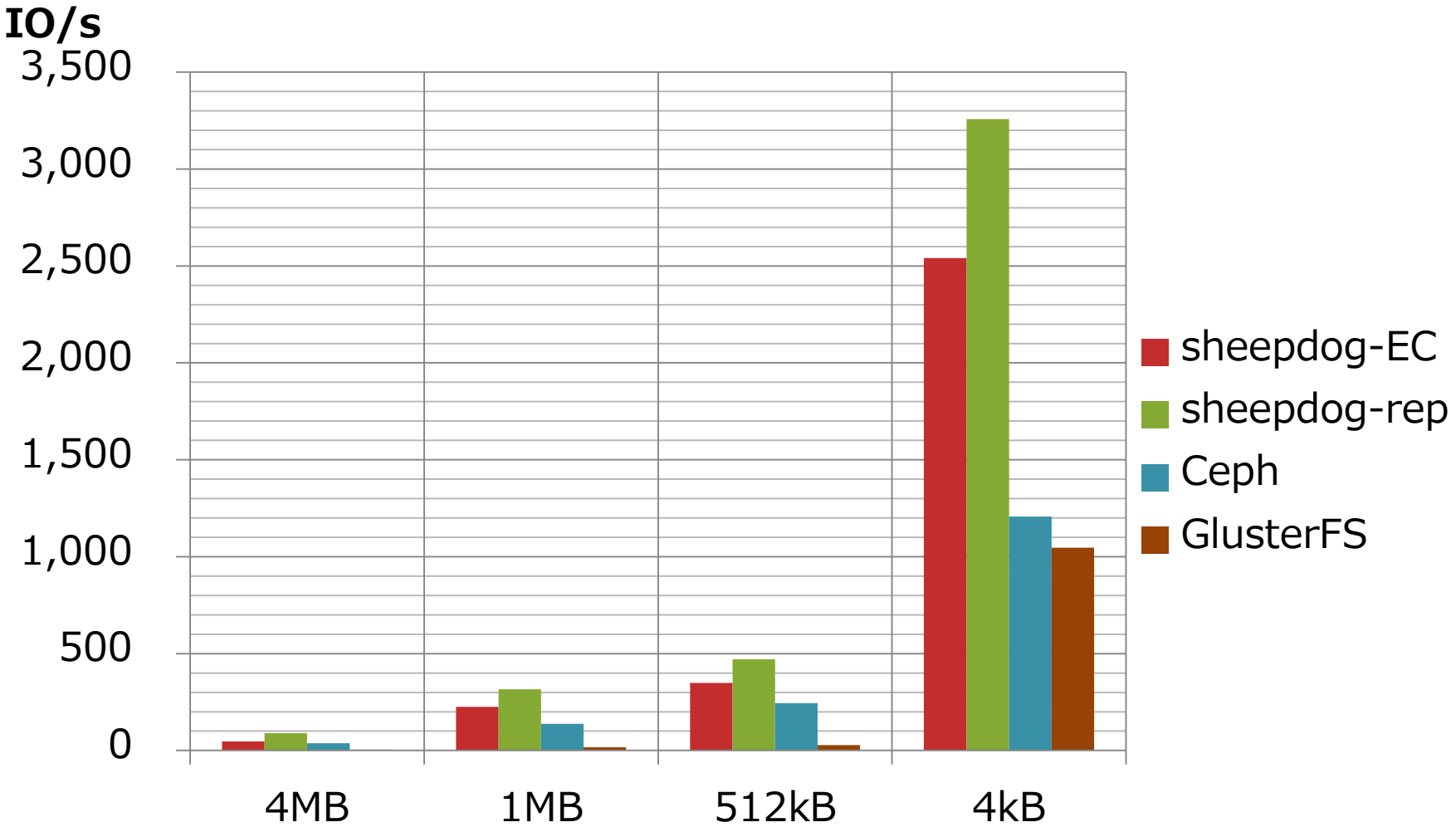
Performance comparison: KVM

- **Sequential Read: Bandwidth**



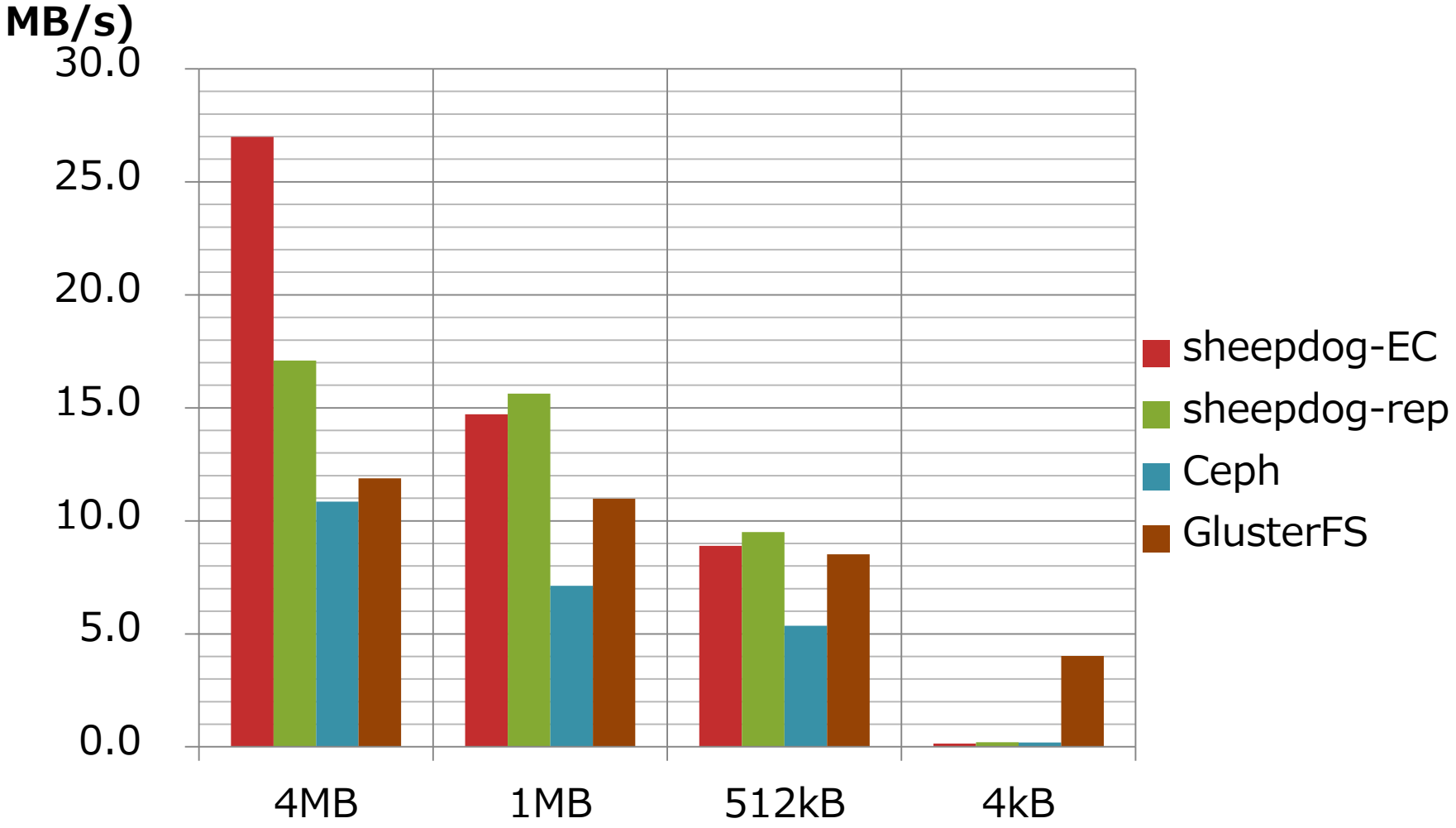
Performance comparison: KVM

- **Sequential Read: IOPS**



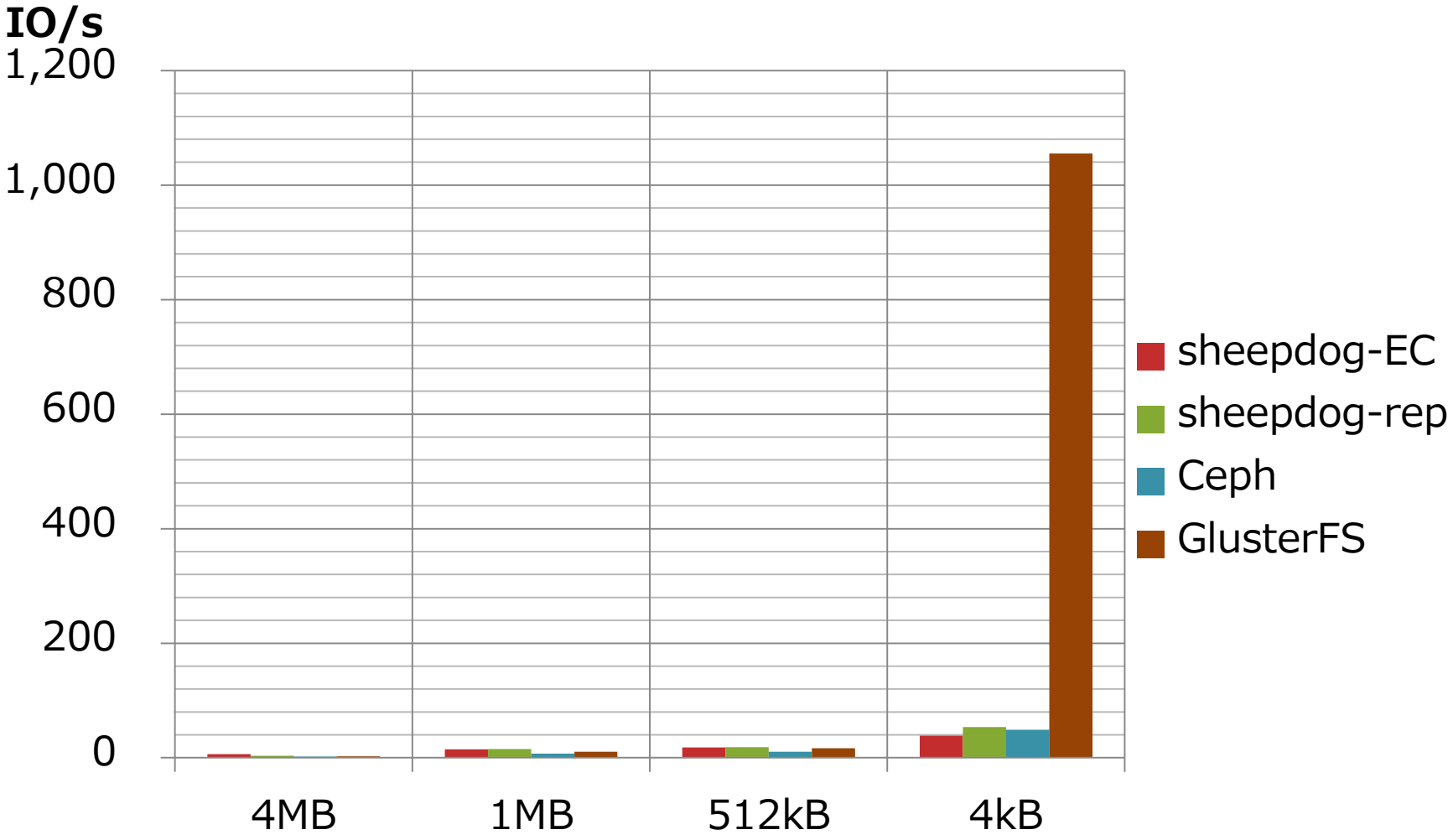
Performance comparison: KVM

- **Sequential Write: Bandwidth**



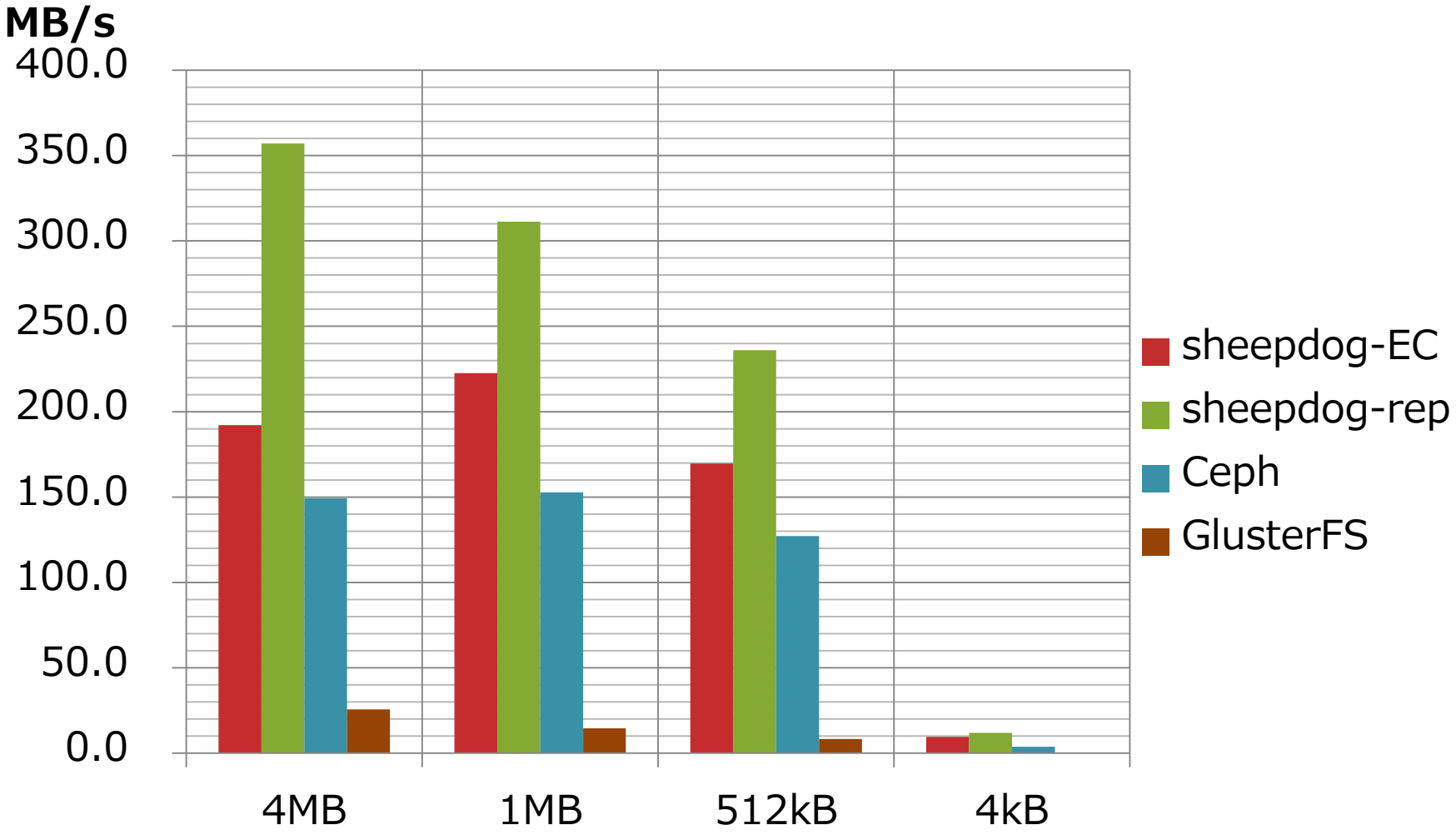
Performance comparison: KVM

- **Sequential Write: IOPS**



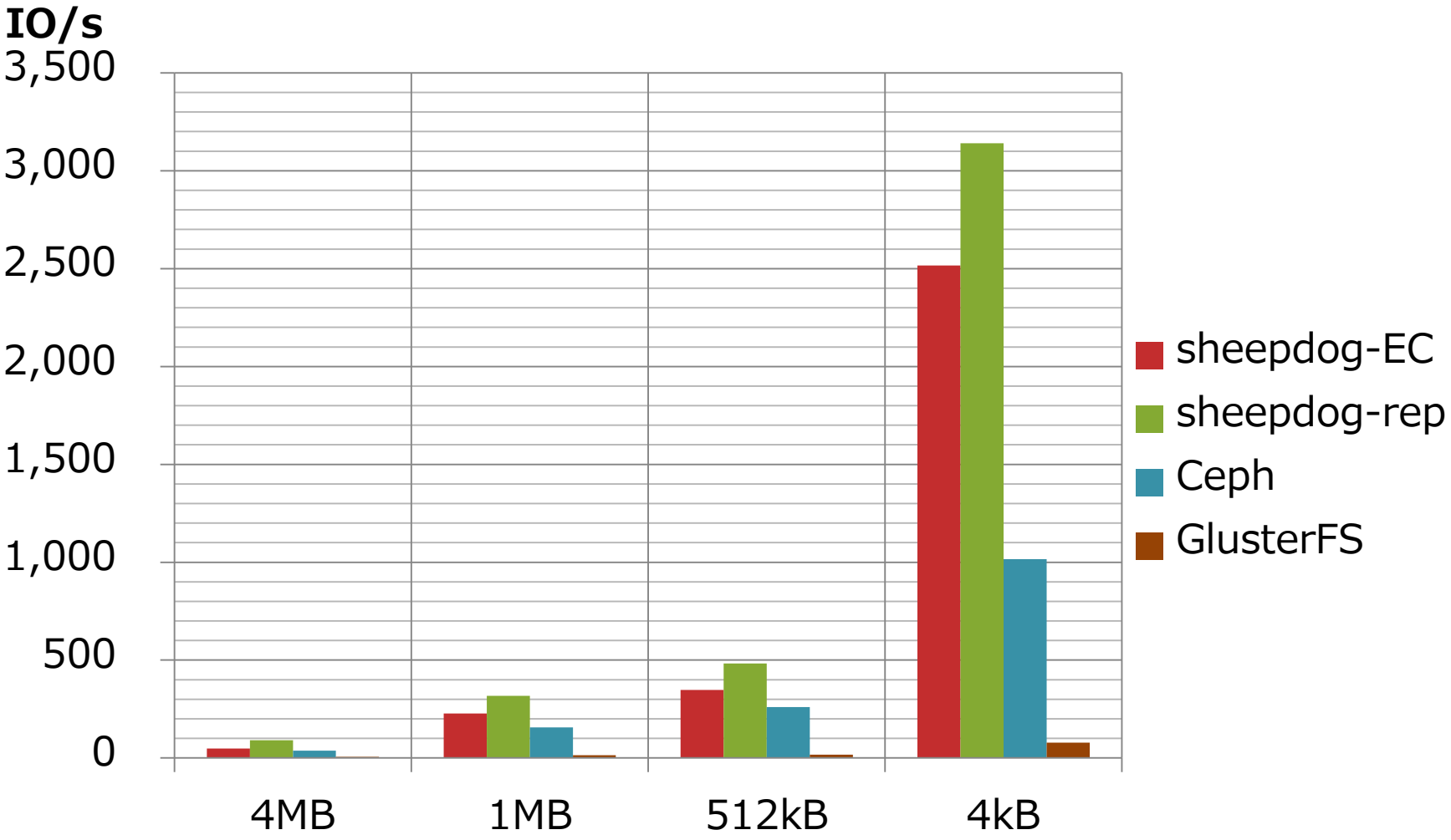
Performance comparison: KVM

- **Random Read: Bandwidth**



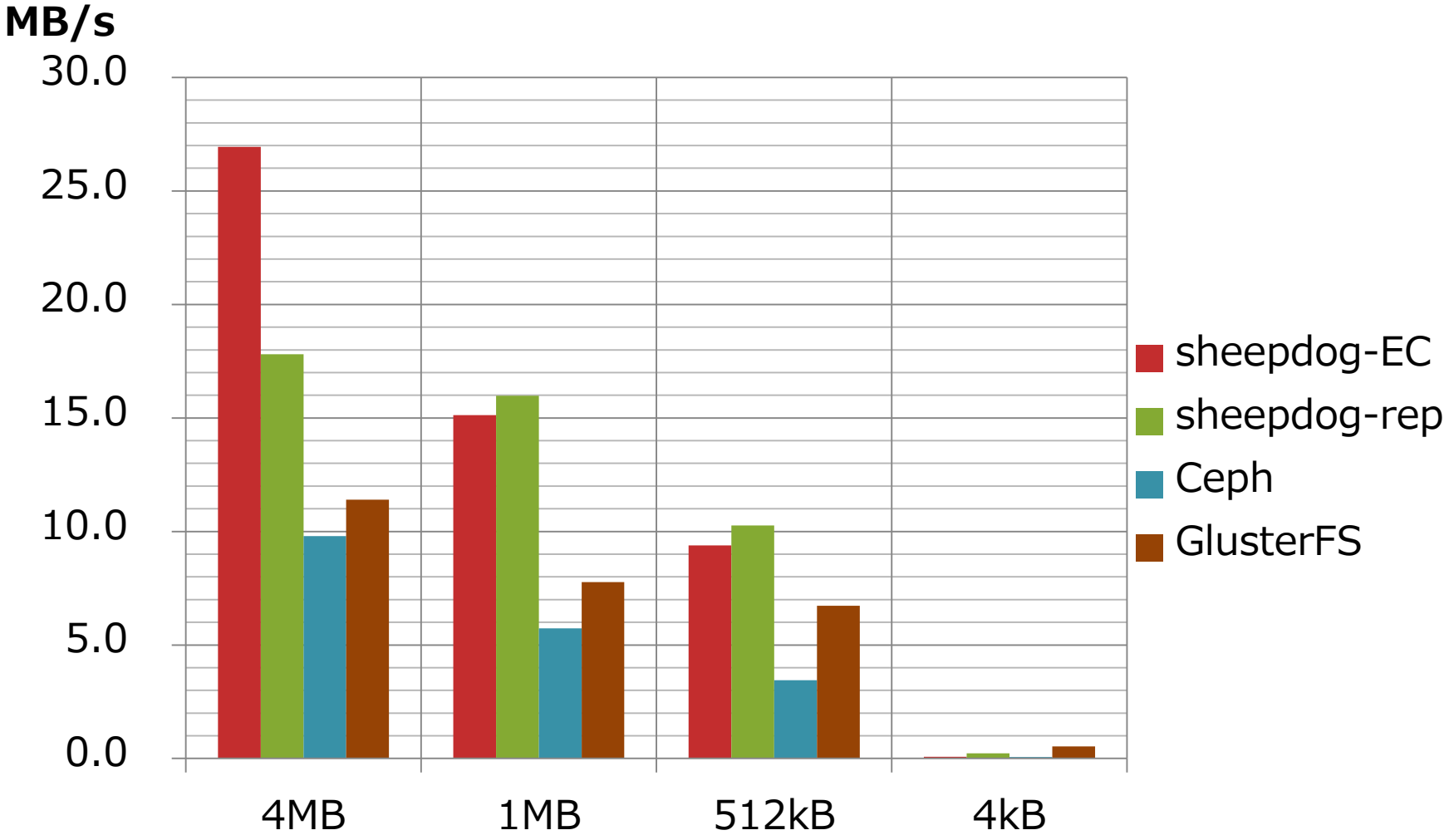
Performance comparison: KVM

- **Random Read: IOPS**



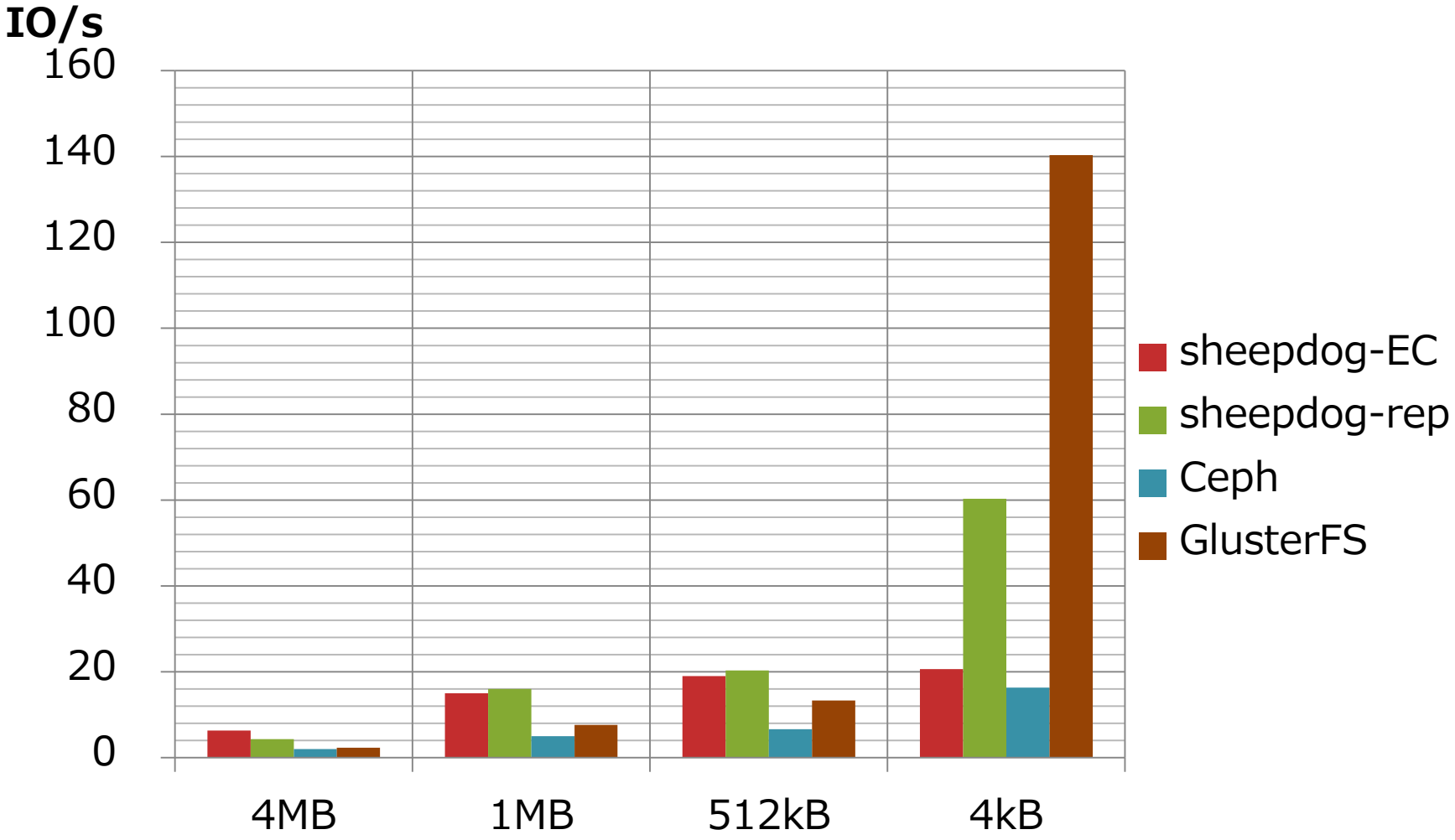
Performance comparison: KVM

- **Random Write: Bandwidth**



Performance comparison: KVM

- **Random Write: IOPS**



Performance comparison: KVM

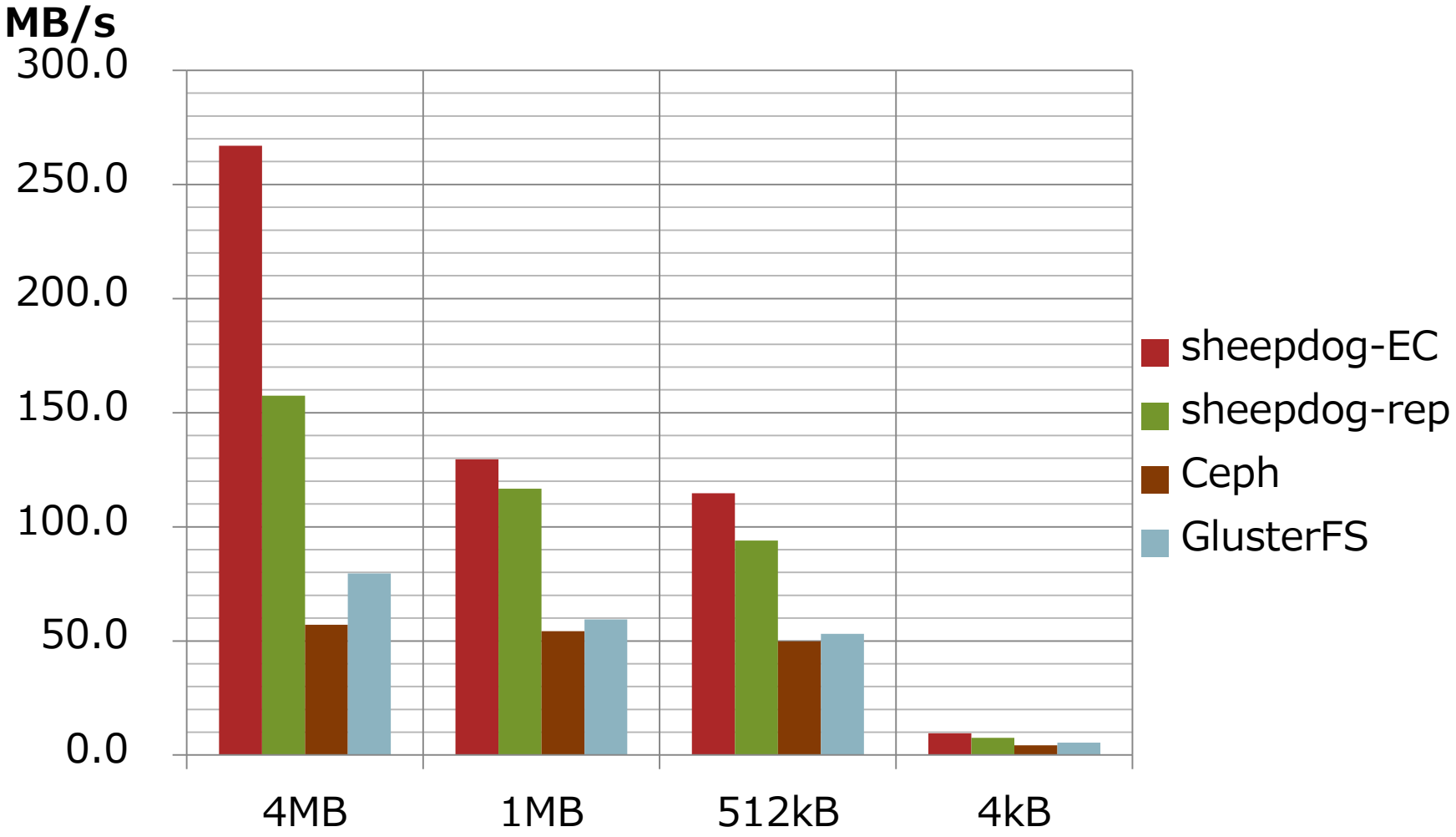
- **Summary**
  - Sheepdog is superior in larger (512KB <) block sizes
    - Under all workloads
    - Erasure coding is effective for improving large block writes
  - GlusterFS is superior in small block writes
    - Especially, tt can be seen in IOPS score

Performance comparison: iSCSI

- **Configurations for the case of iSCSI**
  - tgt version: 1.0.46
  - iSCSI initiator: Linux + open-iscsi 6.2.0.873-10
  - 9 nodes for storage, 3 nodes for client (iSCSI initiator)
    - 3 of 9 storage nodes run tgtd (iSCSI target)
  - Average score of 3 initiators is presented (5 times run)
  - Sheepdog
    - 3 replicas and 4:2 erasure coding
  - Ceph
    - 3 replicas
  - GlusterFS
    - 3 replicas + 3 stripes

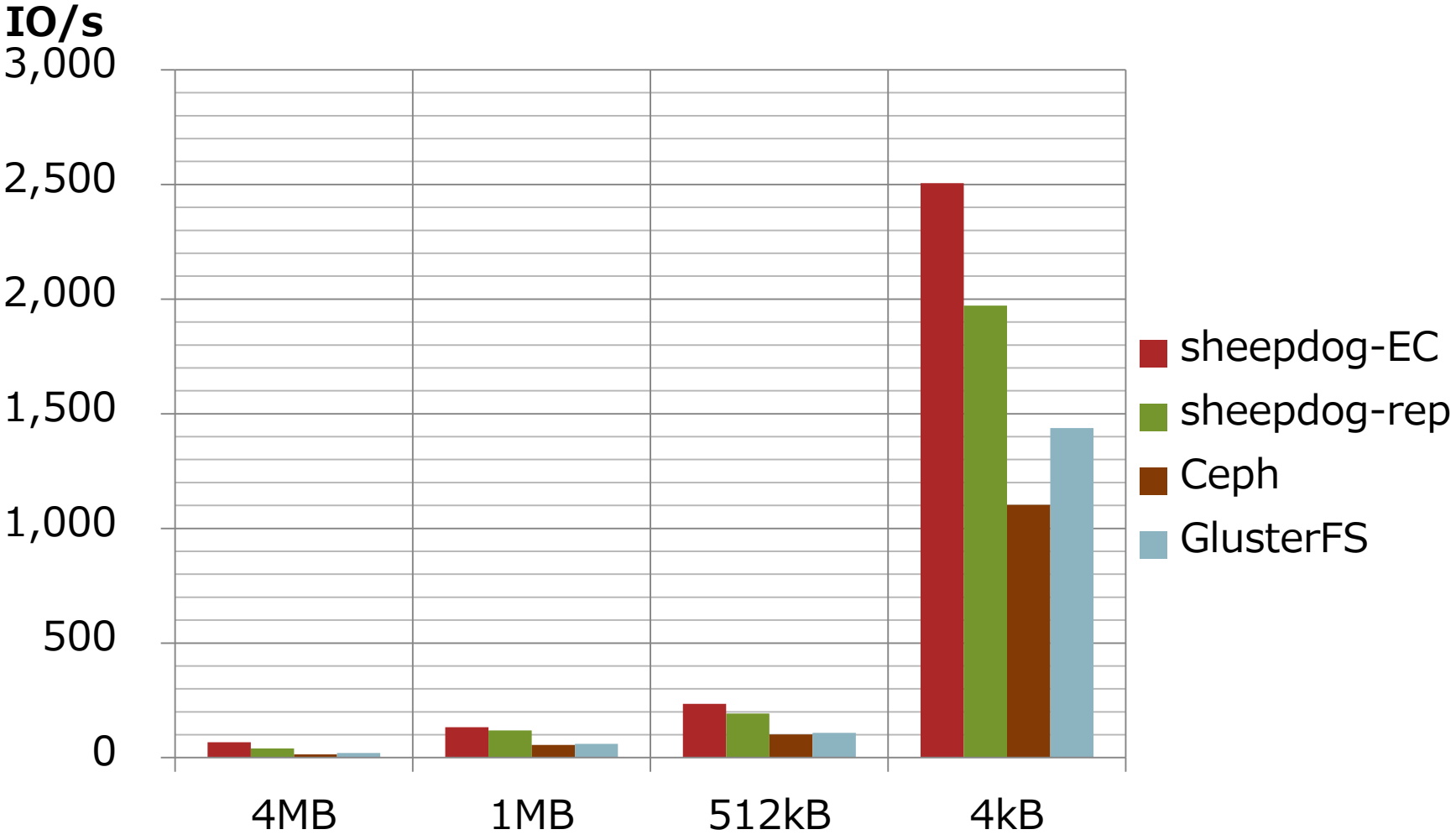
Performance comparison: iSCSI

- **Sequential Read: Bandwidth**



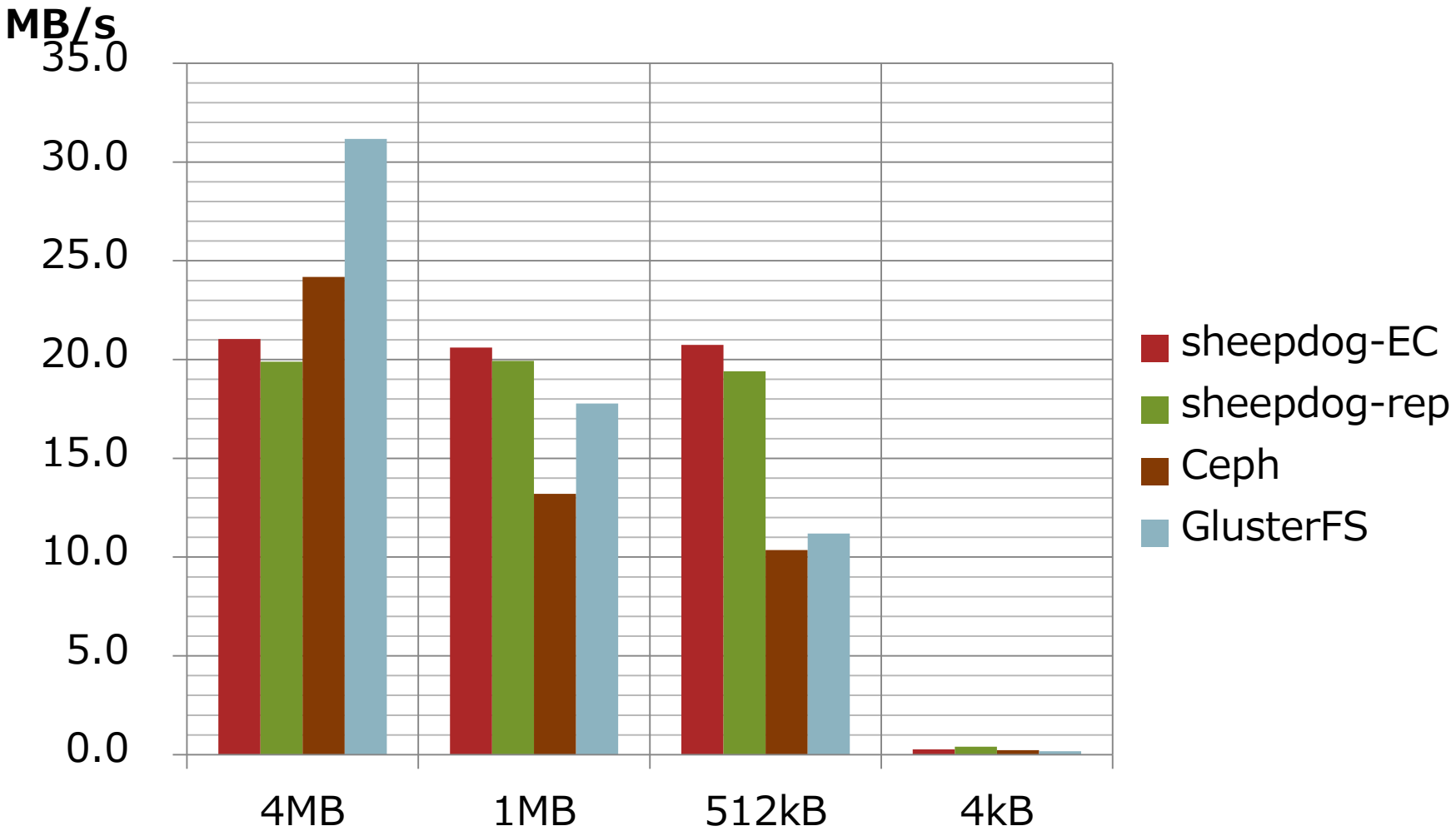
Performance comparison: iSCSI

- **Sequential Read: IOPS**



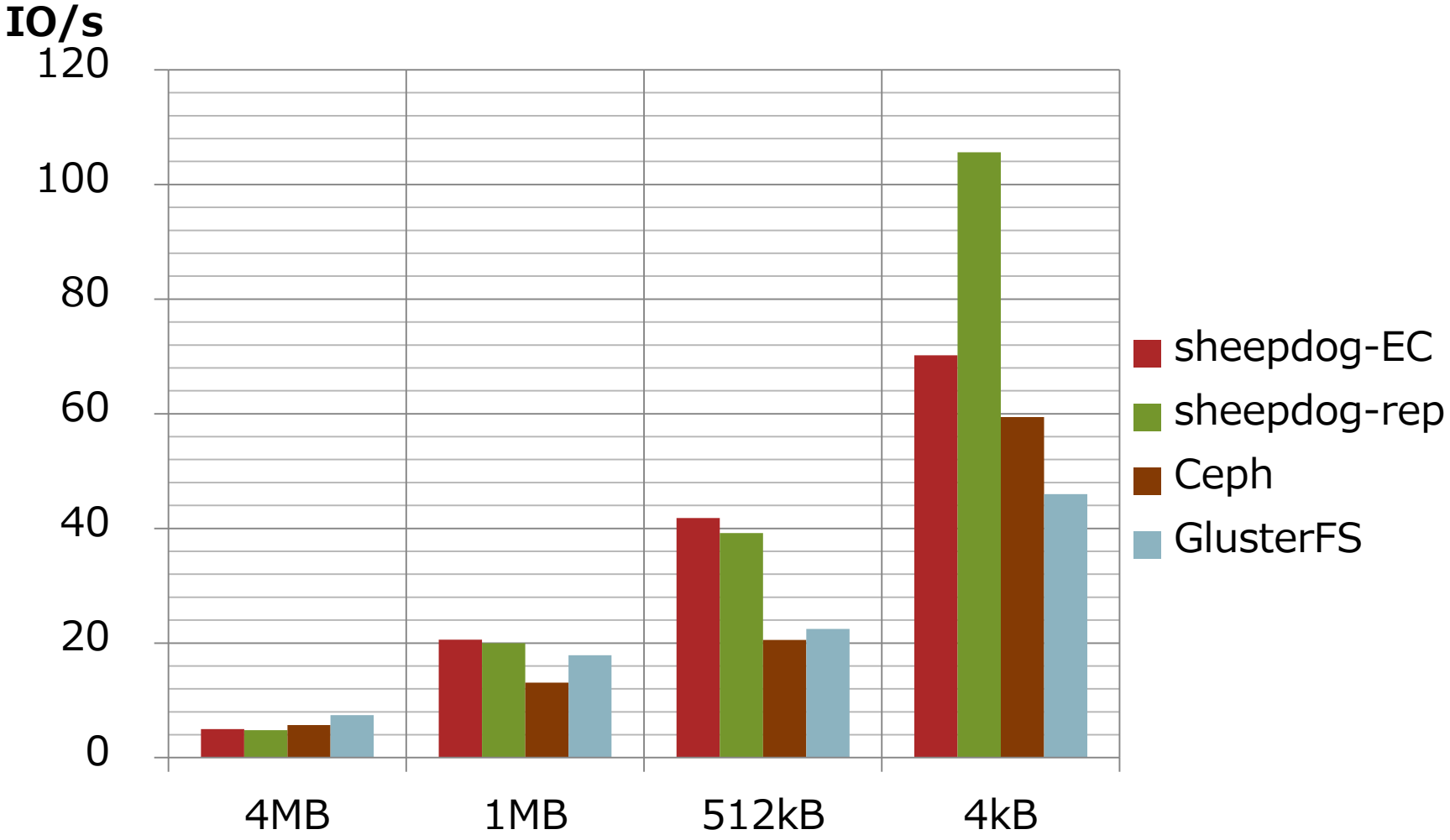
Performance comparison: iSCSI

- **Sequential Write: Bandwidth**



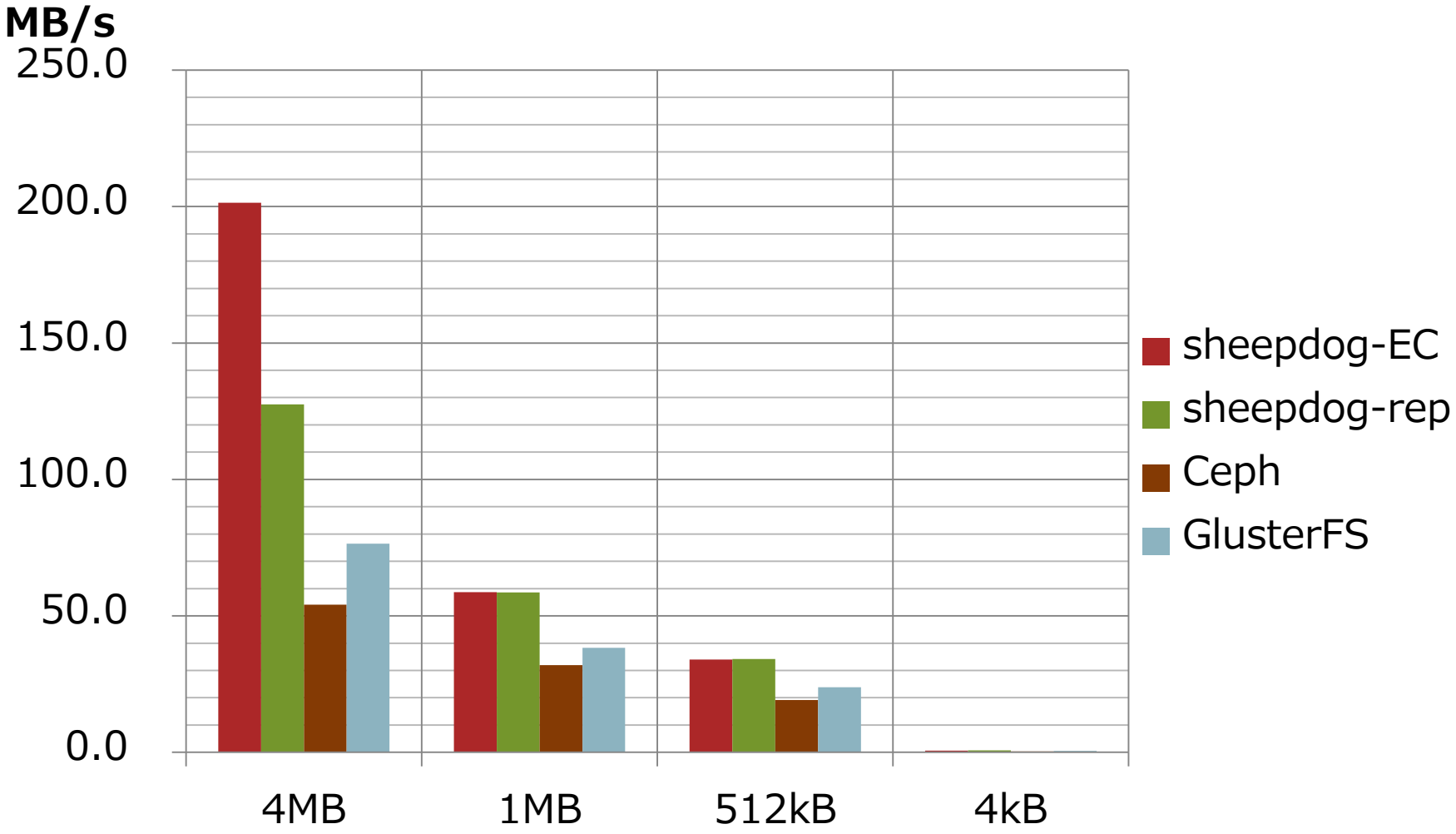
Performance comparison: iSCSI

- **Sequential Write: IOPS**



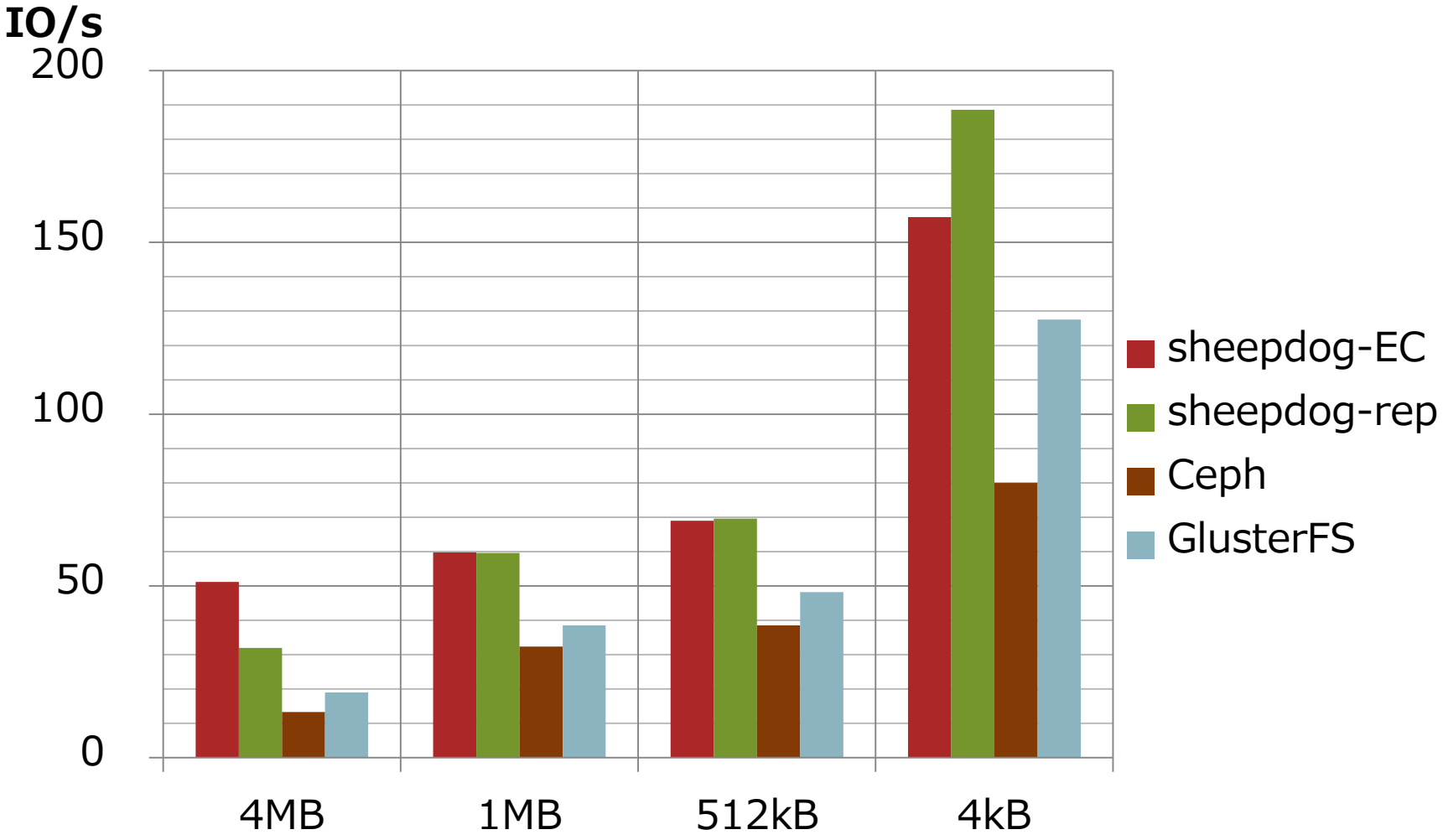
Performance comparison: iSCSI

- **Random Read: Bandwidth**



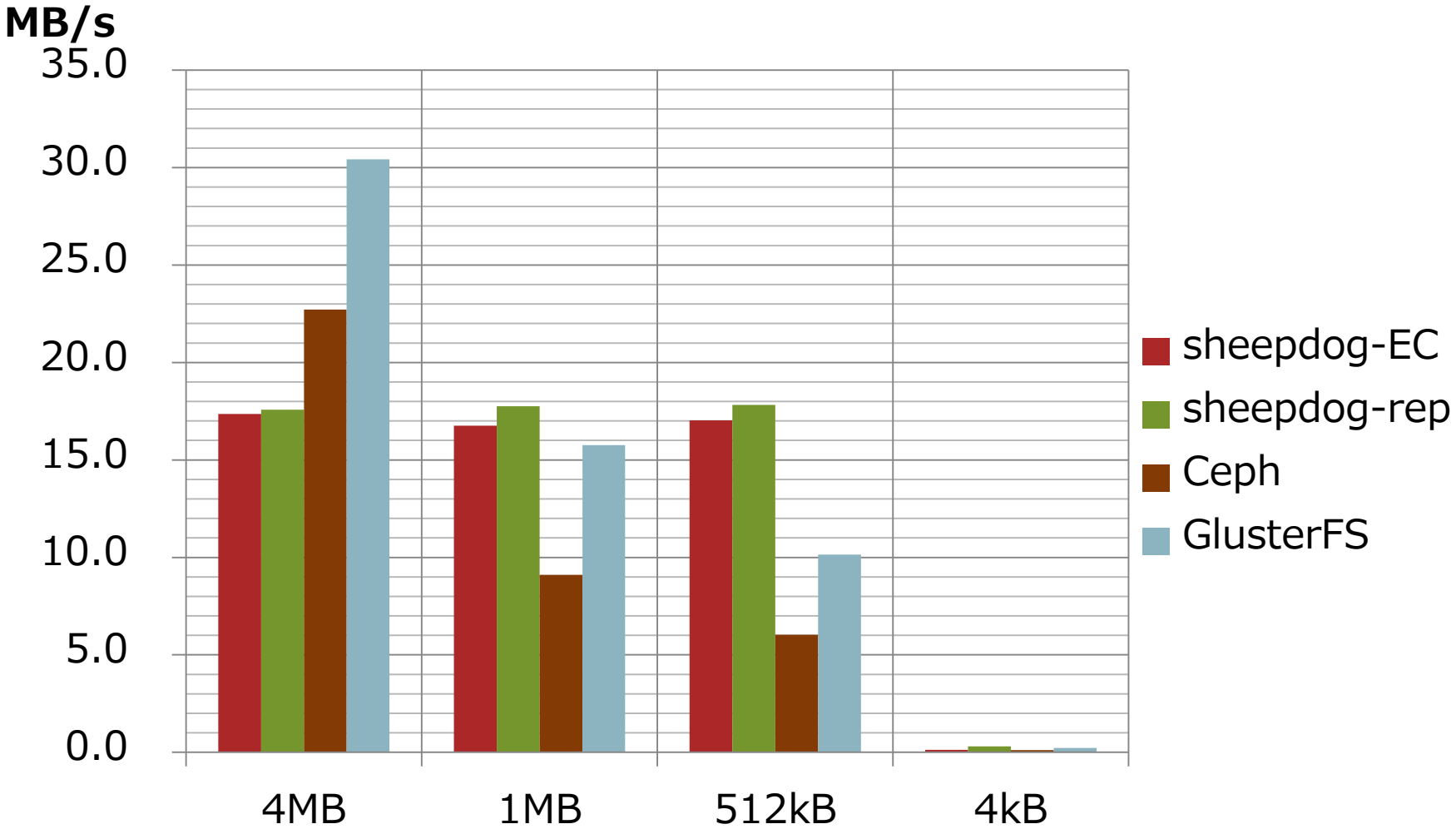
Performance comparison: iSCSI

- **Random Read: IOPS**



Performance comparison: iSCSI

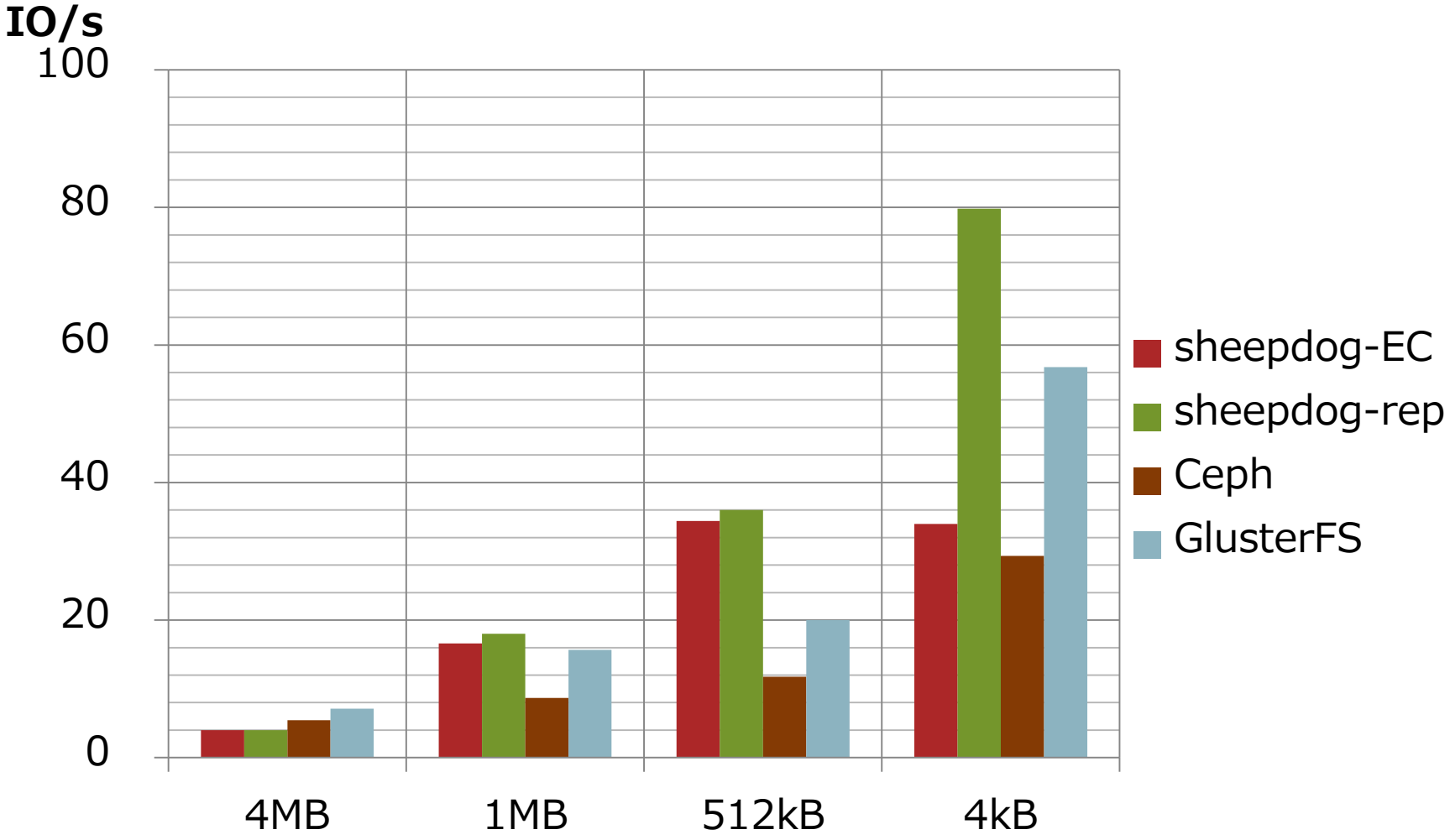
- **Random Write: Bandwidth**



Performance comparison: iSCSI

Innovative R&D by NTT

- **Random Write: IOPS**



NTT

Performance comparison: iSCSI

- **Summary**
  - Sheepdog is superior in many cases other than large block writes
  - Unlike the case of KVM, small block access is better than GlusterFS

Performance comparison: Summary

- **Sheepdog is superior in many cases**

- **Call for your evaluation**
  - Evaluating distributed storage is hard
    - very time consuming
    - innumerable workloads and configurations both in hardware and software
    - many perspectives
      - Scalability
      - Performance during recovery process caused by node failure
      - Worst case of latency
      - Fairness between clients
      - Etc…
  - We are not expert of Ceph and GlusterFS ☹

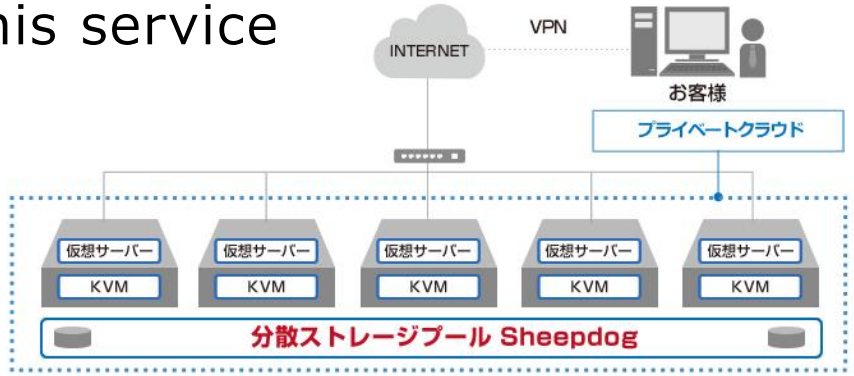
Outline

1. The trend of SDS and sheepdog
2. Recent development status
3. Performance comparison with Ceph and GlusterFS
4. **Introduction of use cases**
   - Examples of commercial users
   - Introducing names of people who can be reached in our mailing list
5. Conclusion

A.T. WORKS, Inc. (Japan)

- **A server vendor and service provider famous of “red server”**
  - http://atworks.co.jp/
  - Person: Masahiro Tsuji

- **Use case: Rental private cloud**
  - http://rental-private-cloud.jp/
  - Kind of private IaaS
    - dedicated hardware in DC of A.T. WORKS
    - no resource conflict between other users
  - sheepdog is a storage of this service



Extensys, Inc. (Italy)

- **A system integrator**
  - http://extensys.it/
  - People: Valerio Pachera and Alessandro Bolgia
- **Use case: build services based on sheepdog for their customers**
  - Cooperating with proxmox server solutions, GmbH
    - for official support of sheepdog in the proxmox VE package

UCWeb, Inc. (China)

- **A software and service company famous for UC Browser**
  - http://www.ucweb.com/
  - Person: Ruoyu
- **Use case: backend services of UC Browser**
  - They are also planning to migrate their online game service (http://www.9game.com/) in the near future.

Delaware State University (U.S.)

- **A university**
  - http://desu.edu/
  - Person: Andrew J. Hobbs
- **Use case: virtualization environment for hosting various services in the campus network**
  - websites based on LAMP, NFS, VPN server, Matlab licensing server, etc…
  - Their cluster started from “retired workstations”, and is expanded by adding new machines.

Argonne National Laboratory (U.S.)

- **A national laboratory**
  - http://www.anl.gov/
  - People: Scott Devoid and Ryan M. Aydelott
- **Use case: cluster for scientific computation**
  - Primary purposes: bioinformatics
    - genome assembly and metagenomics
    - a cosmology project is running on the cluster, too
  - 20 nodes, 14 disks per node, infiniband based cluster
  - They are also working on OpenStack integration
    - http://goo.gl/k9hxeu

Outline

1. The trend of SDS and sheepdog
2. Recent development status
3. Performance comparison with Ceph and GlusterFS
4. Introduction of use cases
5. **Conclusion**

Conclusion

- **Sheepdog can provide a storage functionality for OSS-based SDS products**
  - Both of converged infrastructure and scale-out SAN
- **Future directions and major ToDos**
  - Towards polyglot, all-in-one storage
    - completing NFS interface
  - Cooperation with other software and ecosystems
    - integration with OpenStack, CloudStack
  - Workload aware optimizations
    - smart cache replacement and prefetching policies
    - deduplication, especially for read only objects
  - Advanced hardware support
    - Infiniband
    - PCIe-connected flash drives

Conclusion

- **Getting started**
  - web site: http://sheepdog.github.io/sheepdog/
  - start your sheepdog cluster: https://github.com/sheepdog/sheepdog/wiki/Getting-Started
  - mailing lists
    - for developers: http://lists.wpkg.org/mailman/listinfo/sheepdog
    - for users: http://lists.wpkg.org/mailman/listinfo/sheepdog-users
  - repository: https://github.com/sheepdog/sheepdog
  - latest stable release: v0.8.1
    - https://github.com/sheepdog/sheepdog/tarball/v0.8.1

Thanks a lot for your attention!

Any questions?

APPENDIX

Hardware configuration

- **Machine: Fujitsu PRIMERGY RX200 S8 * 12**
  - Hostname
    - sds01 – 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 * 4
    - 1 for system area, 3 for data area with RAID 0
  - 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

fio parameter for QEMU

```
[global]
thread=1
time_based=1
runtime=60
ramp_time=10
invalidate=1
stonewall=1
direct=1
fsync=1
unlink=1
ioengine=sync
numjobs=1
directory=/var/tmp/
size=1G
```

```
; workload specific part, example of 512k sequential write
[512k-seqwrite-fs-1thread-sync]
bs=512k
rw=write
```

fio parameter for iSCSI

```
[global]
thread=1
time_based=1
runtime=60
ramp_time=10
invalidate=1
stonewall=1
direct=1
fsync=1
unlink=1
ioengine=sync
numjobs=1
directory=/dev/sdg
size=10G
```

```
; workload specific part, example of 512k sequential write
[512k-seqwrite-fs-1thread-sync]
bs=512k
rw=write
```

Sheepdog boot parameter

- run on every node with different IP address

```
# sheep -p 7000 -b 192.168.2.11 -i host=192.168.2.11,port=7001 -l
dir=/var/log/sheepdog,level=info /data/sheepdog/sheepdog_qemu
```

- Cluter initialization for erasure coding

```
# dog cluster format –c 4:2
```

- Cluster initialization for 3 replication

```
# dog cluster format –c 3
```

Configuration of ceph

```
[osd.9]
public_addr = 192.168.2.19
cluster_addr = 192.168.2.19

[osd.8]
public_addr = 192.168.2.18
cluster_addr = 192.168.2.18

[global]
auth_service_required = cephx
filestore_xattr_use_omap = true
auth_client_required = cephx
auth_cluster_required = cephx
mon_host = 172.16.2.22
mon_initial_members = sds12
cluster_network = 192.168.2.0/24
cluster_addr = 192.168.2.22
fsid = 1b15baf1-17fb-454a-a051-df32db6e9708

[osd.1]
public_addr = 192.168.2.11
cluster_addr = 192.168.2.11

[osd.3]
public_addr = 192.168.2.13
cluster_addr = 192.168.2.13

[osd.2]
public_addr = 192.168.2.12
cluster_addr = 192.168.2.12
```

```
[osd.5]
public_addr = 192.168.2.15
cluster_addr = 192.168.2.15

[osd.4]
public_addr = 192.168.2.14
cluster_addr = 192.168.2.14

[osd.7]
public_addr = 192.168.2.17
cluster_addr = 192.168.2.17

[osd.6]
public_addr = 192.168.2.16
cluster_addr = 192.168.2.16

[osd.11]
public_addr = 192.168.2.21
cluster_addr = 192.168.2.21

[osd.10]
public_addr = 192.168.2.20
cluster_addr = 192.168.2.20

[mon.sds01]
host = sds01
addr = 192.168.2.11:6789

[mon.sds03]
host = sds03
addr = 192.168.2.13:6789

[mon.sds02]
host = sds02
addr = 192.168.2.12:6789
```

Procedure of creating volumes of Ceph: for the case of QEMU/KVM

Innovative R&D by NTT

```
# ceph-deploy new sds12
•    modify config on sds12
# ceph-deploy install sds12 sds01 sds02 sds03 sds04 sds05 sds06 sds07 sds08 sds09 sds10 sds11
# ceph-deploy mon create-initial
# ceph-deploy gatherkeys sds12
# ceph-deploy osd prepare sds01:/data/ceph/osd sds02:/data/ceph/osd
# ceph-deploy osd prepare sds03:/data/ceph/osd sds04:/data/ceph/osd sds05:/data/ceph/osd
# ceph-deploy osd prepare sds06:/data/ceph/osd sds07:/data/ceph/osd sds08:/data/ceph/osd
# ceph-deploy osd activate sds01:/data/ceph/osd sds02:/data/ceph/osd
# ceph-deploy osd activate sds03:/data/ceph/osd sds04:/data/ceph/osd sds05:/data/ceph/osd
# ceph-deploy osd activate sds06:/data/ceph/osd sds07:/data/ceph/osd sds08:/data/ceph/osd

# ceph-deploy osd activate sds09:/data/ceph/osd sds10:/data/ceph/osd sds11:/data/ceph/osd
# ceph osd tree

# ceph osd pool create pool01 400 400
# ceph osd pool set pool01 size 3
# ceph osd dump | grep 'rep size'

•    Configuring authentication on sds12
# ceph auth get-or-create client.libvirt mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=pool01'

•    Add mon, on sds12
# ceph-deploy --overwrite-conf mon add sds01
# ceph-deploy --overwrite-conf mon add sds02
# ceph-deploy --overwrite-conf mon add sds03

•    Cluster configuration on sds12
# ceph-deploy admin sds12 sds01 sds02 sds03 sds04 sds05 sds06 sds07 sds08 sds09 sds10 sds11

•    Create volume from golden image, on sds12, repeat for 12 VMs with different names
# qemu-img convert -t directsync /root/sds/goldgen_image/vm_golden_image.raw rbd:rbd/sds12-vm01-ceph.img
```

NTT

Procedure of creating volumes of Ceph: for the case of iSCSI

```
# ceph-deploy new sds12
•    modify config on sds12
# ceph-deploy install sds12 sds01 sds02 sds03 sds04 sds05 sds06 sds07 sds08 sds09 sds10 sds11
# ceph-deploy mon create-initial
# ceph-deploy gatherkeys sds12
# ceph-deploy osd prepare sds01:/data/ceph/osd sds02:/data/ceph/osd
# ceph-deploy osd prepare sds03:/data/ceph/osd sds04:/data/ceph/osd sds05:/data/ceph/osd
# ceph-deploy osd prepare sds06:/data/ceph/osd sds07:/data/ceph/osd sds08:/data/ceph/osd
# ceph-deploy osd activate sds01:/data/ceph/osd sds02:/data/ceph/osd
# ceph-deploy osd activate sds03:/data/ceph/osd sds04:/data/ceph/osd sds05:/data/ceph/osd
# ceph-deploy osd activate sds06:/data/ceph/osd sds07:/data/ceph/osd sds08:/data/ceph/osd

# ceph osd tree

•    Pool configuration, on sds12
# ceph osd pool set rbd size 3
# ceph osd dump | grep 'rep size'

•    Cluster configuration, on sds12
# ceph-deploy admin sds12 sds01 sds02 sds03 sds04 sds05 sds06 sds07 sds08

•    Create volume, on sds12
# rbd create image1 --size 15360
# rbd create image2 --size 15360
# rbd create image3 --size 15360
```

Configuration of GlusterFS

Innovative R&D by NTT

- /etc/glusterfs/glusterd.vol

```
volume management
    type mgmt/glusterd
    option working-directory /var/lib/glusterd
    option transport-type socket,rdma
    option transport.socket.keepalive-time 10
    option transport.socket.keepalive-interval 2
    option transport.socket.read-fail-log off
end-volume
```

NTT

Procedure of creating volume of GlusterFS: the case of QEMU/KVM

・create a directory for blick

```
# mkdir /data/gluster/brick-qemu
(# dsh -g sds mkdir /data/gluster/brick-qemu/)
```

・create a volume

```
# gluster vol create vol-qemu replica 3 stripe 4
192.168.2.11:/data/gluster/brick-qemu 192.168.2.12:/data/gluster/brick-
qemu 192.168.2.13:/data/gluster/brick-qemu
192.168.2.14:/data/gluster/brick-qemu 192.168.2.15:/data/gluster/brick-
qemu 192.168.2.16:/data/gluster/brick-qemu
192.168.2.17:/data/gluster/brick-qemu 192.168.2.18:/data/gluster/brick-
qemu 192.168.2.19:/data/gluster/brick-qemu
192.168.2.20:/data/gluster/brick-qemu 192.168.2.21:/data/gluster/brick-
qemu 192.168.2.22:/data/gluster/brick-qemu force
```

Procedure of creating volume of GlusterFS: the case of iSCSI

Innovative R&D by NTT

•create directory for blick area
mkdir /data/gluster/brick-iscsi
(# dsh -w sds01,sds02,sds03,sds04,sds05,sds06,sds07,sds08,sds09 mkdir /data/gluster/brick-iscsi/)

•create volume
gluster vol create vol-iscsi replica 3 stripe 3 192.168.2.11:/data/gluster/brick-iscsi 192.168.2.12:/data/gluster/brick-iscsi 192.168.2.13:/data/gluster/brick-iscsi 192.168.2.14:/data/gluster/brick-iscsi 192.168.2.15:/data/gluster/brick-iscsi 192.168.2.16:/data/gluster/brick-iscsi 192.168.2.17:/data/gluster/brick-iscsi 192.168.2.18:/data/gluster/brick-iscsi 192.168.2.19:/data/gluster/brick-iscsi force

•activate volume
gluster vol start vol-iscsi

•mount
mkdir /mnt/glustervol/
mount -t glusterfs 192.168.2.11:vol-iscsi /mnt/glustervol/

•create file for logical unit
dd if=/dev/zero of=/mnt/glustervol/lun1 bs=1024M count=15

• Create target
/root/work/tgt/usr/tgtadm --lld iscsi --mode target --op new --tid 1 -T iqn.tgt01.local
/root/work/tgt/usr/tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address ALL
/root/work/tgt/usr/tgtadm --lld iscsi --mode logicalunit --op new --tid 1 --lun 1 --bstype glfs --backing-store="vol-iscsi@192.168.2.11:lun1"

NTT