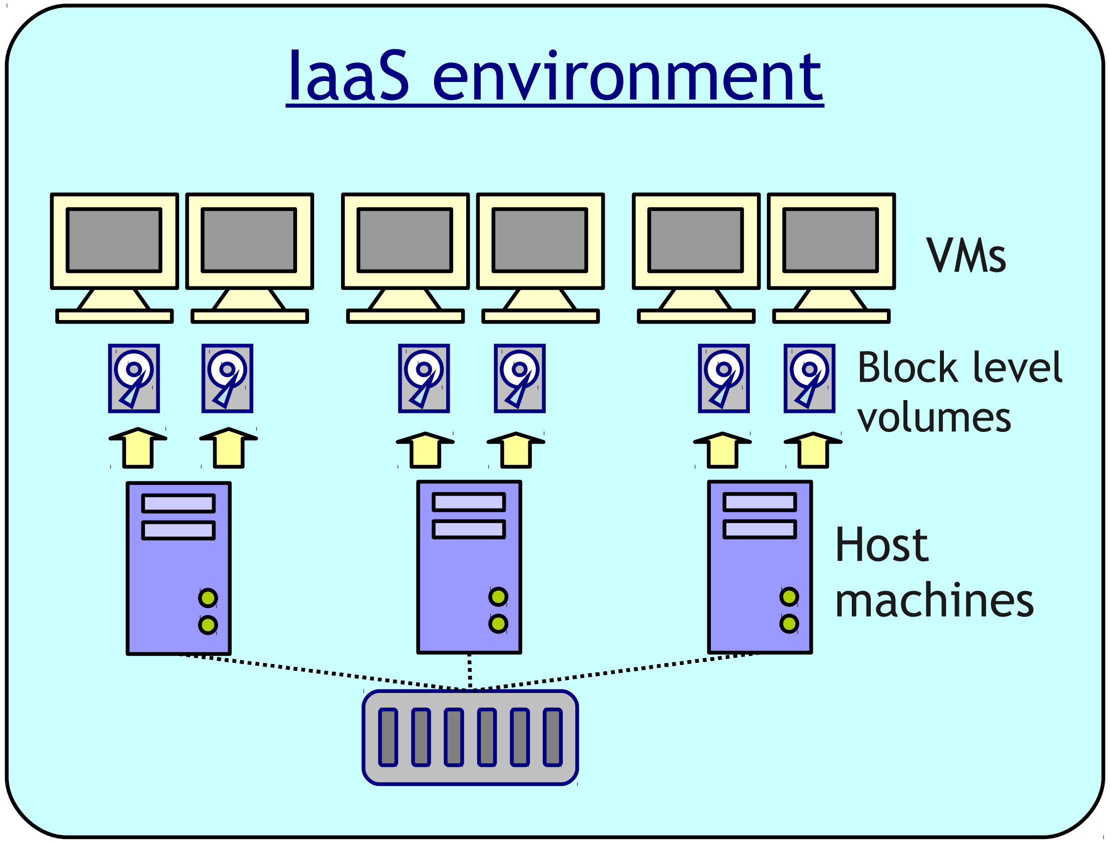
Sheepdog: distributed storage system for QEMU

Kazutaka Morita
NTT Cyber Space Labs.

9 August, 2010

Motivation

- There is no open source storage system which fits for IaaS environment like Amazon EBS

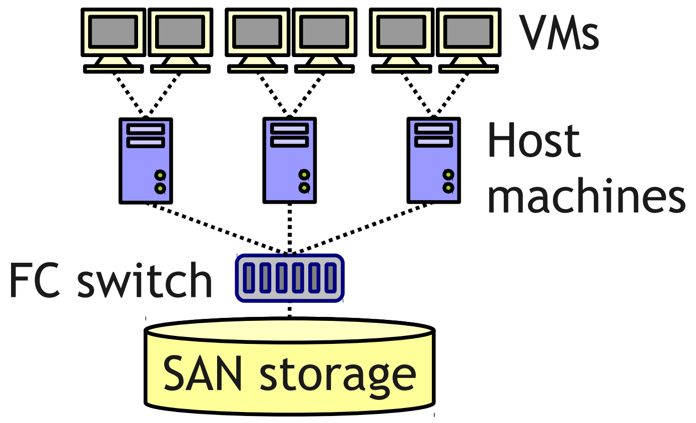


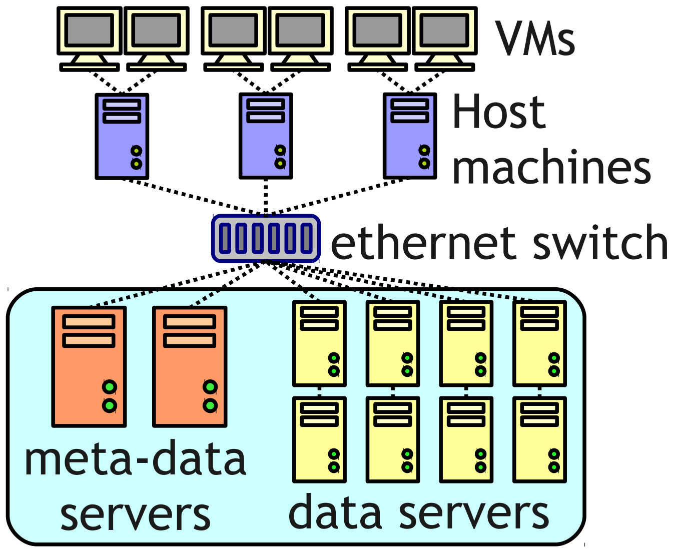
Requirements for storage system

- Scalability
- Reliability
- Manageability

Why another storage system?

- Why not SAN storage?
  - Large proprietary storage system is too expensive
  - Shared storage could be a single point of failure
- Why not distributed file systems? (e.g. Ceph, Luster)
  - Complex configuration about cluster membership



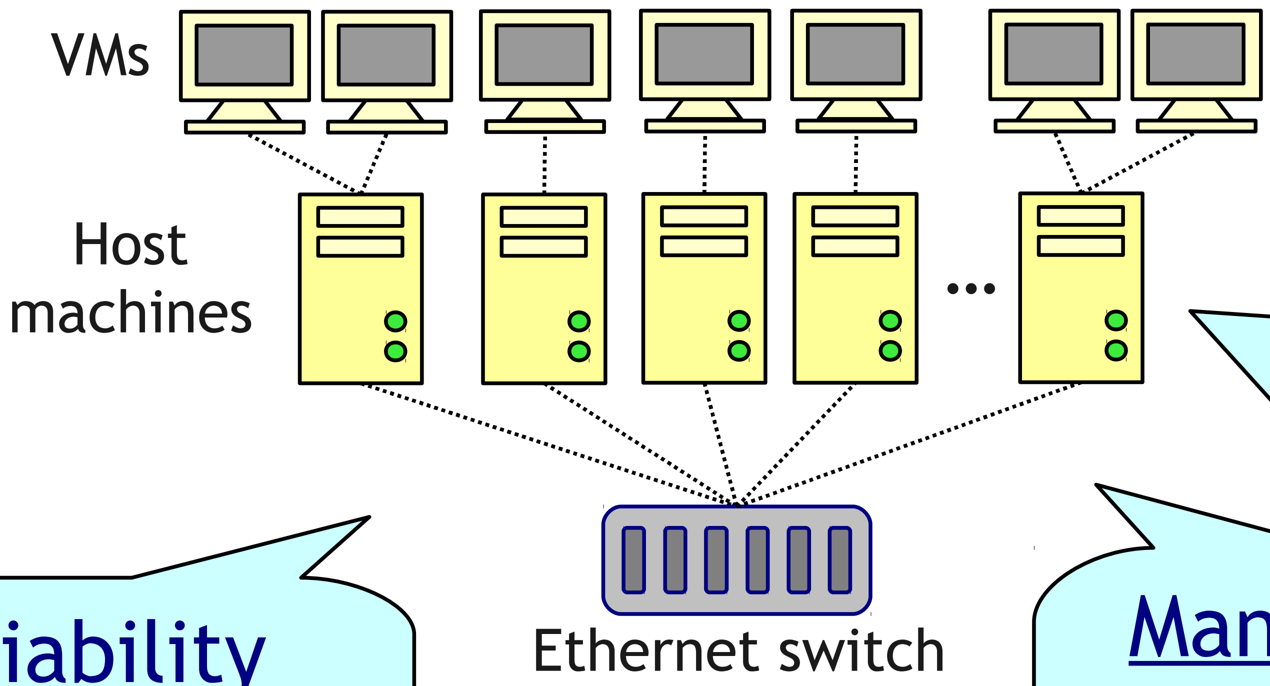


Distributed file system

Sheepdog

Fully symmetric architecture

there is no central node such as a meta-data server



Scalability

- Scales to 1000 nodes

Reliability

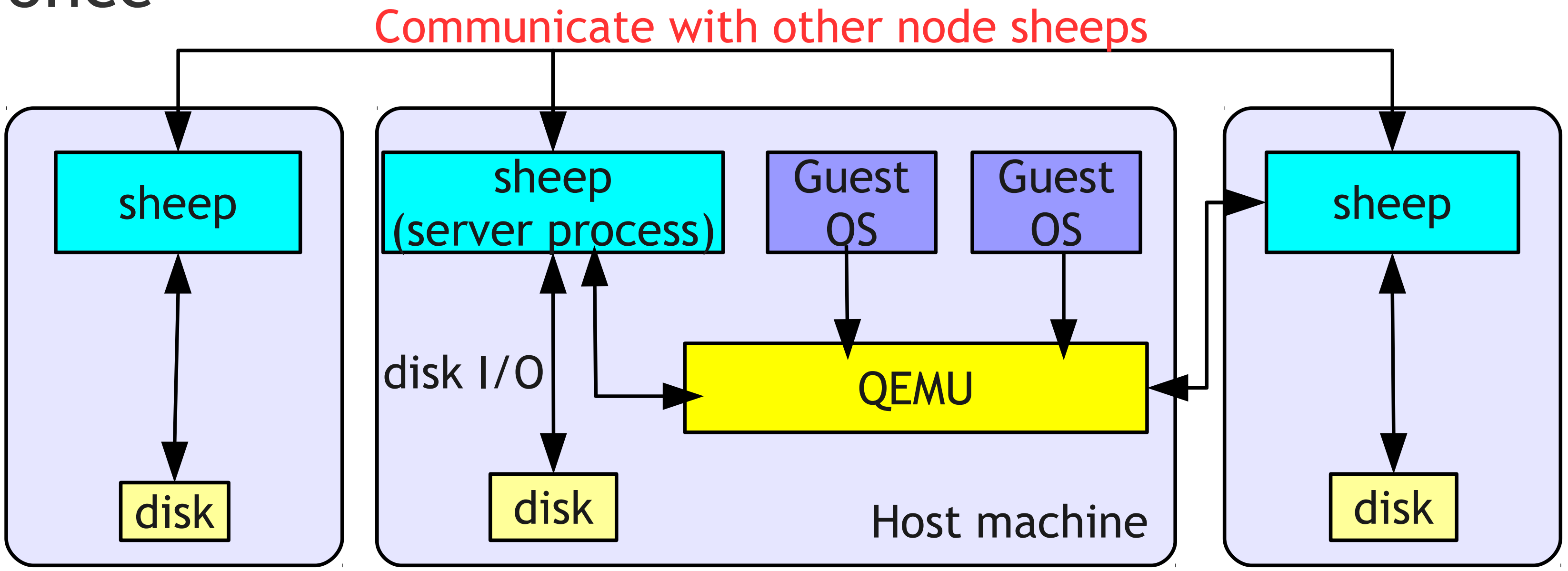
- Data replication
- No SPOF

Manageability

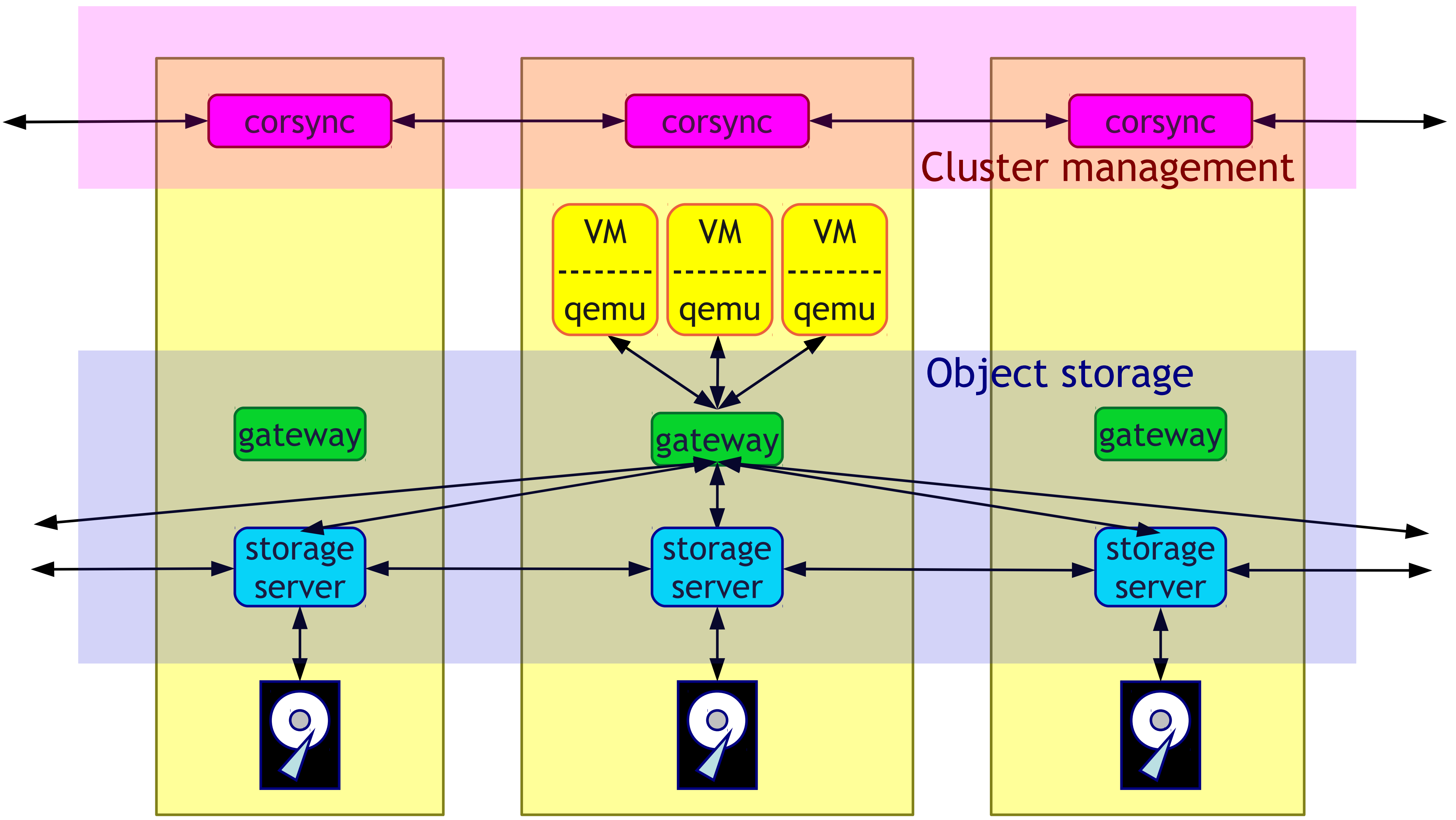
- Autonomous
- Dynamic membership
- Advanced volume manipulation

Design: not general file system

- We have simplified the design significantly
  - API is designed specific to QEMU
  - We cannot use sheepdog as a file system
  - One volume can be attached to only one VM at once

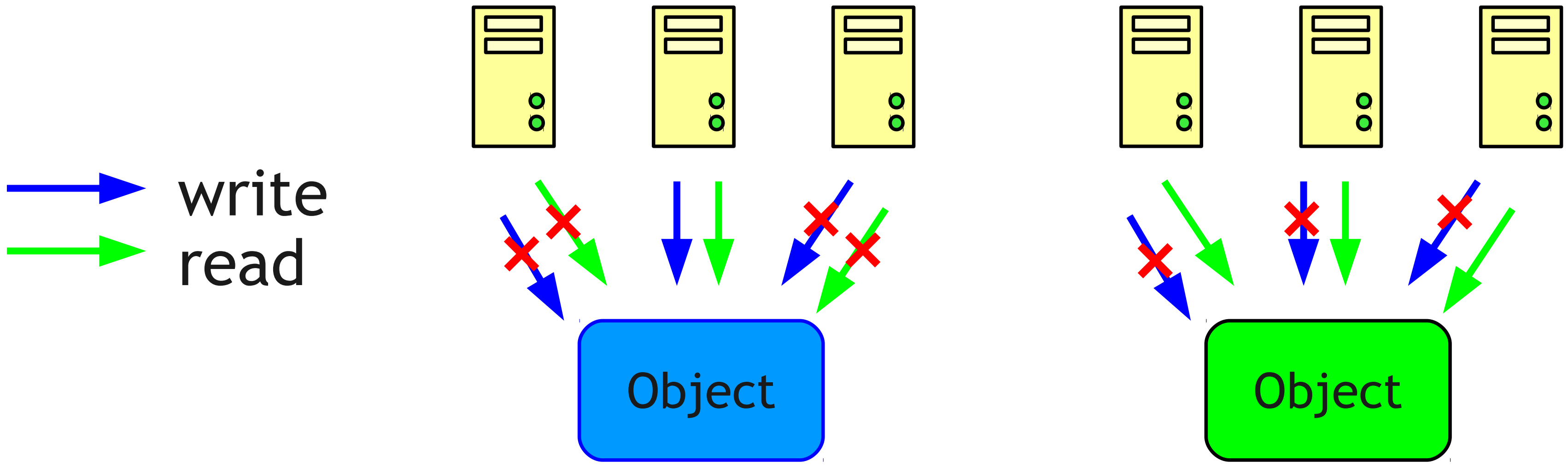


Sheepdog components



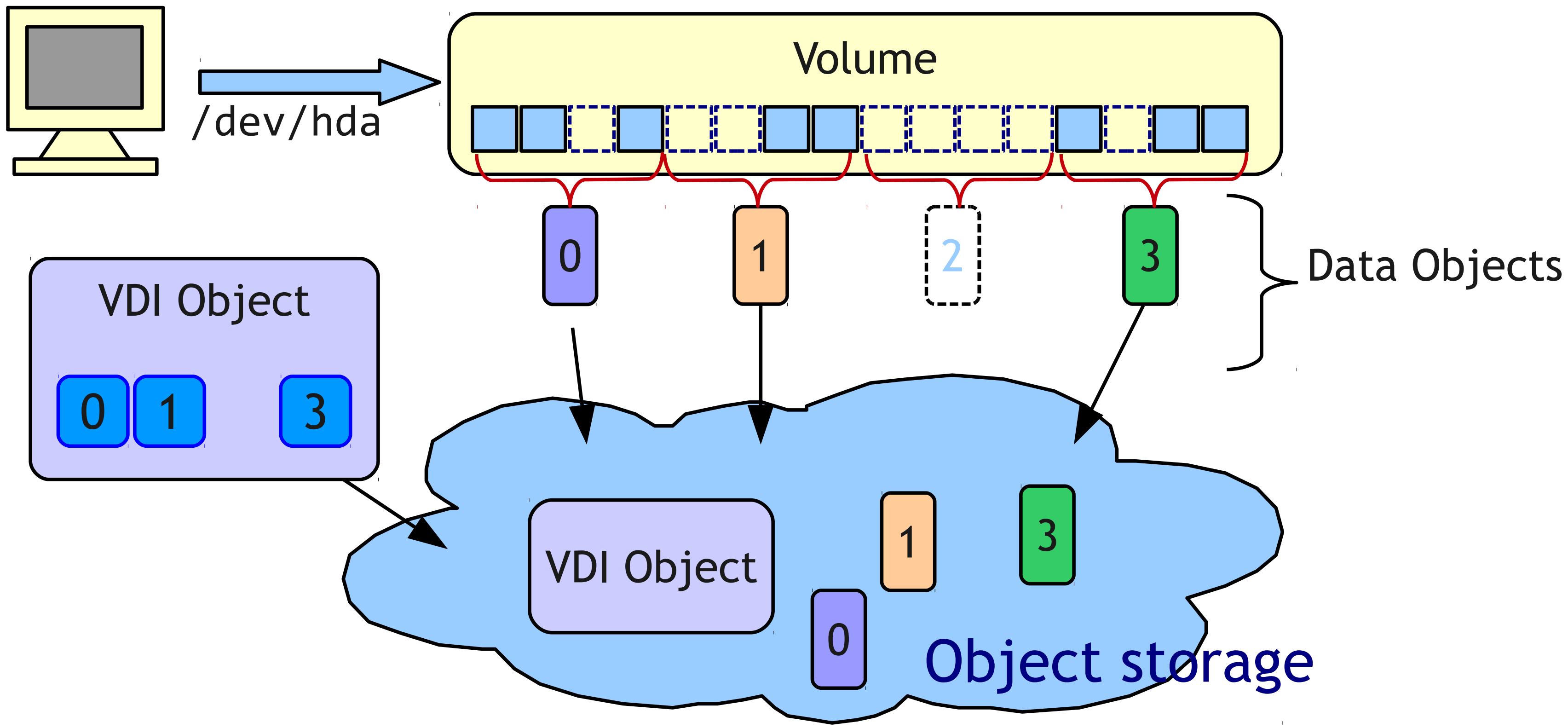
Object storage

- Stores flexible-sized data with a unique ID (objects)
- Clients don't care about where to store objects
- Two kinds of objects in Sheepdog
  - One writer, one reader
  - No writer, multiple readers



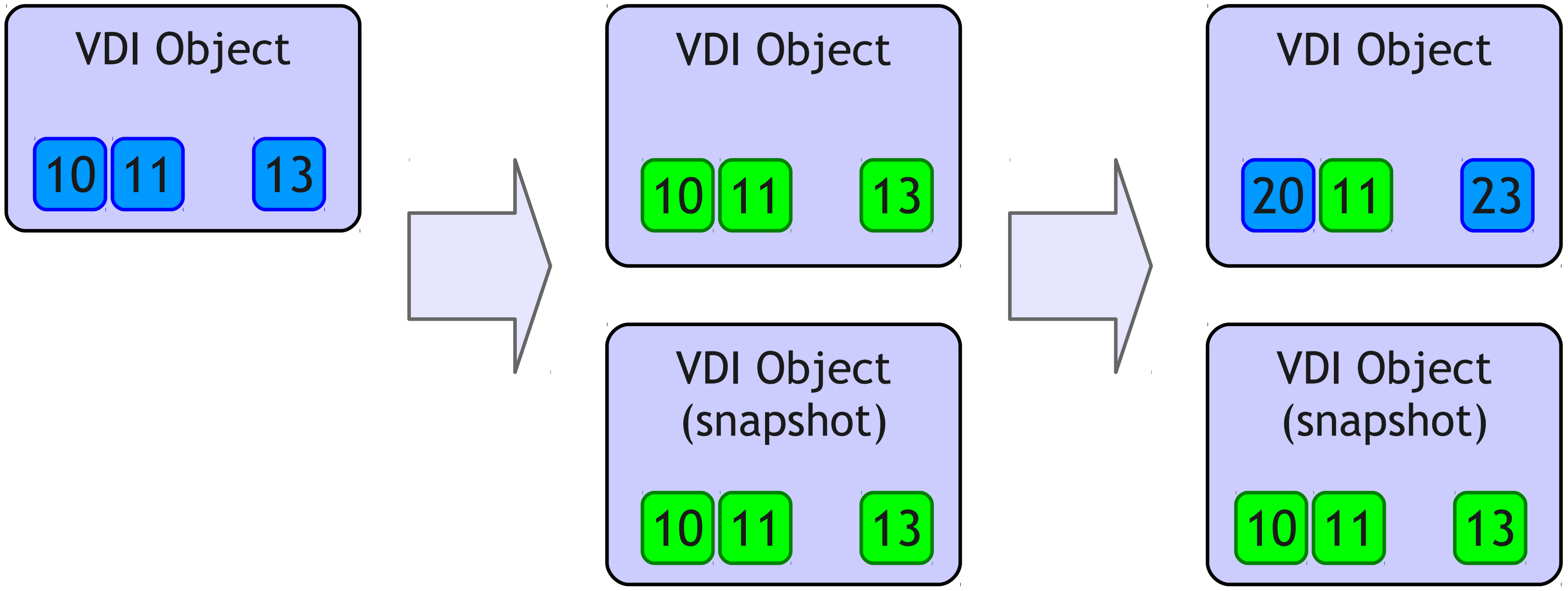
How to store volumes?

- Volumes are divided into 4 MB data objects
- Allocation table is stored to VDI object



Snapshot

- Copy VDI Object, and make allocated data objects read-only
- Updating read-only objects causes copy-on-write

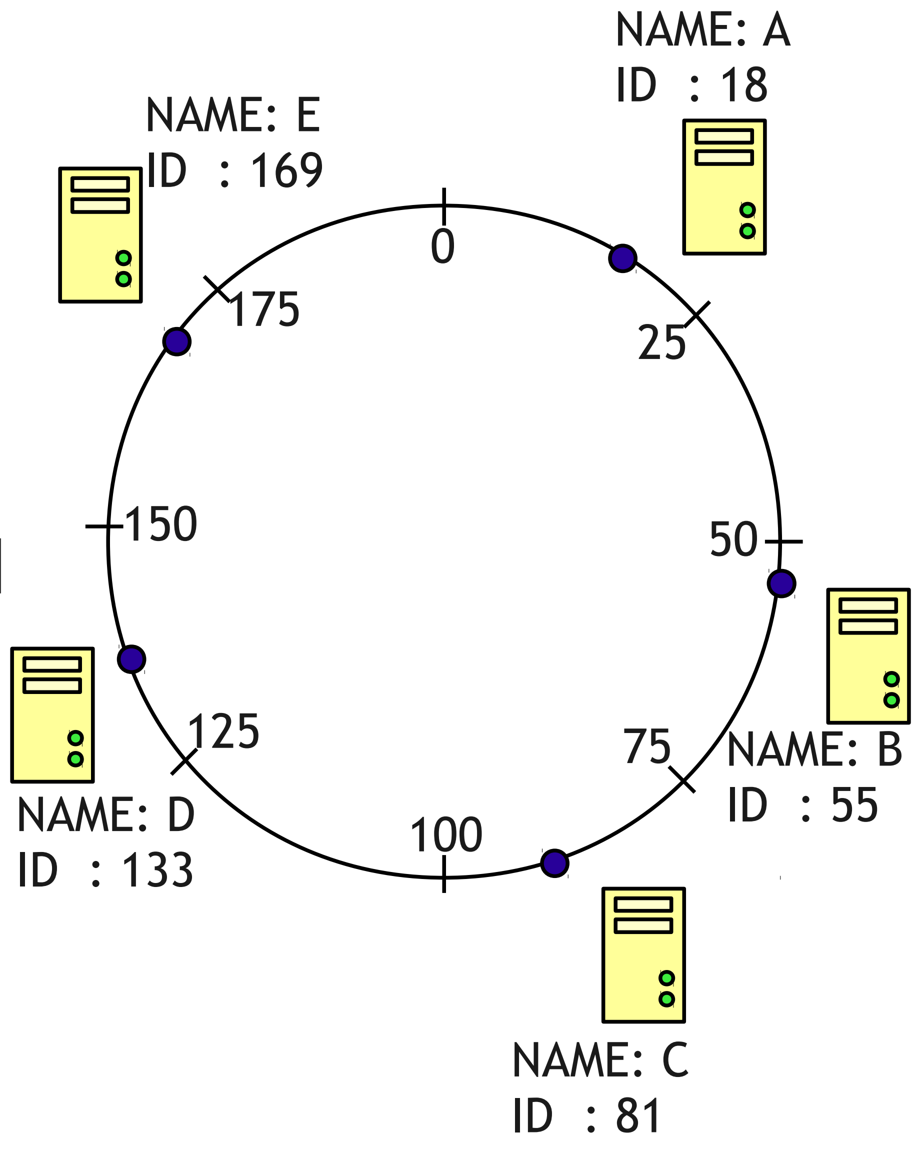


Create snapshot

Copy-on-write

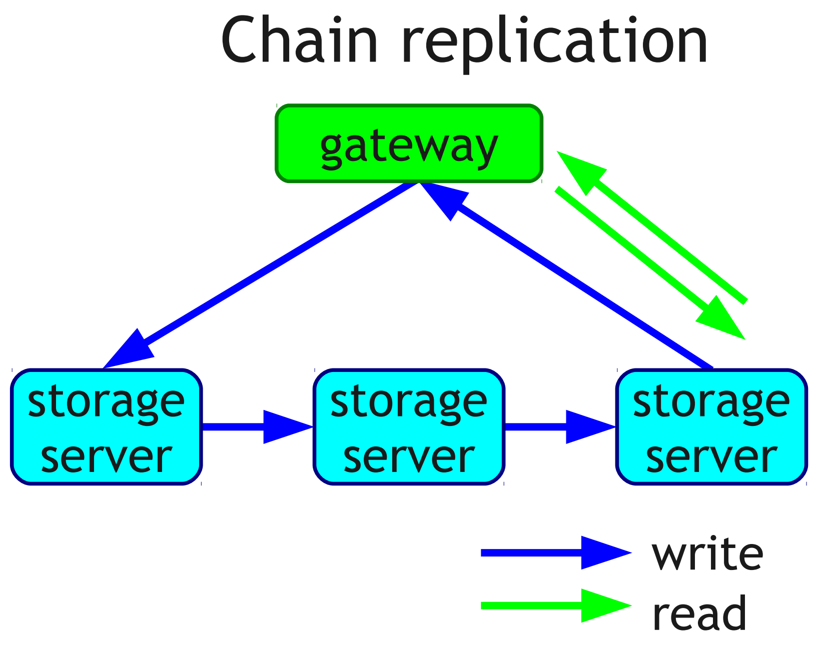
Where to store objects?

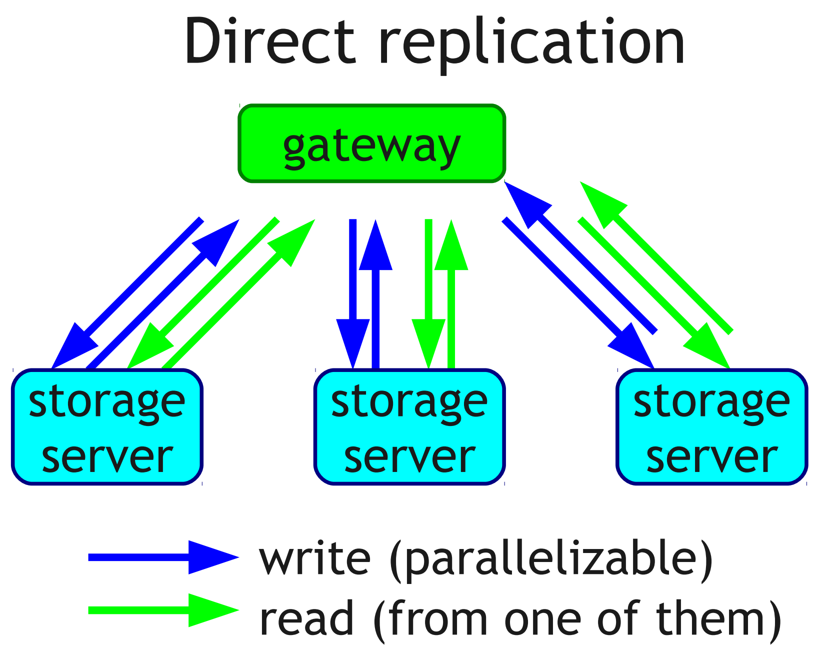
- We use consistent hashing to decide which node to store objects
  - Each node is also placed on the ring
  - addition or removal of nodes does not significantly change the mapping of objects



Replication

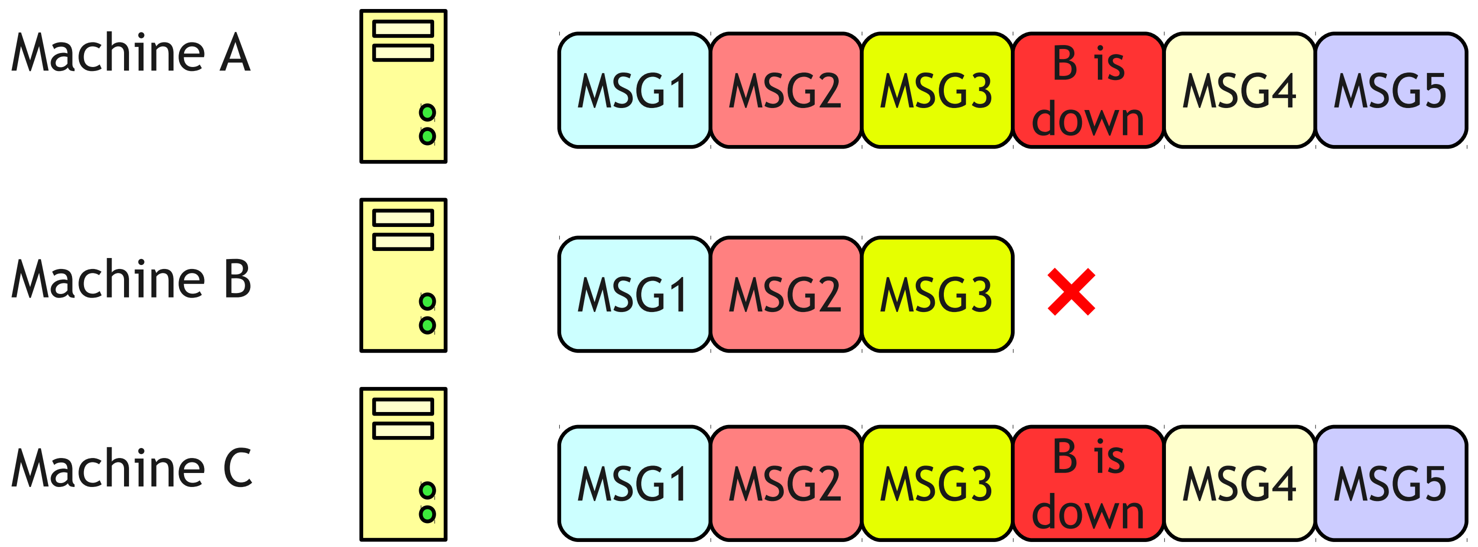
- Many distributed storage systems use chain replication to maintain I/O ordering
- Sheepdog can use direct replication because write collision cannot happen





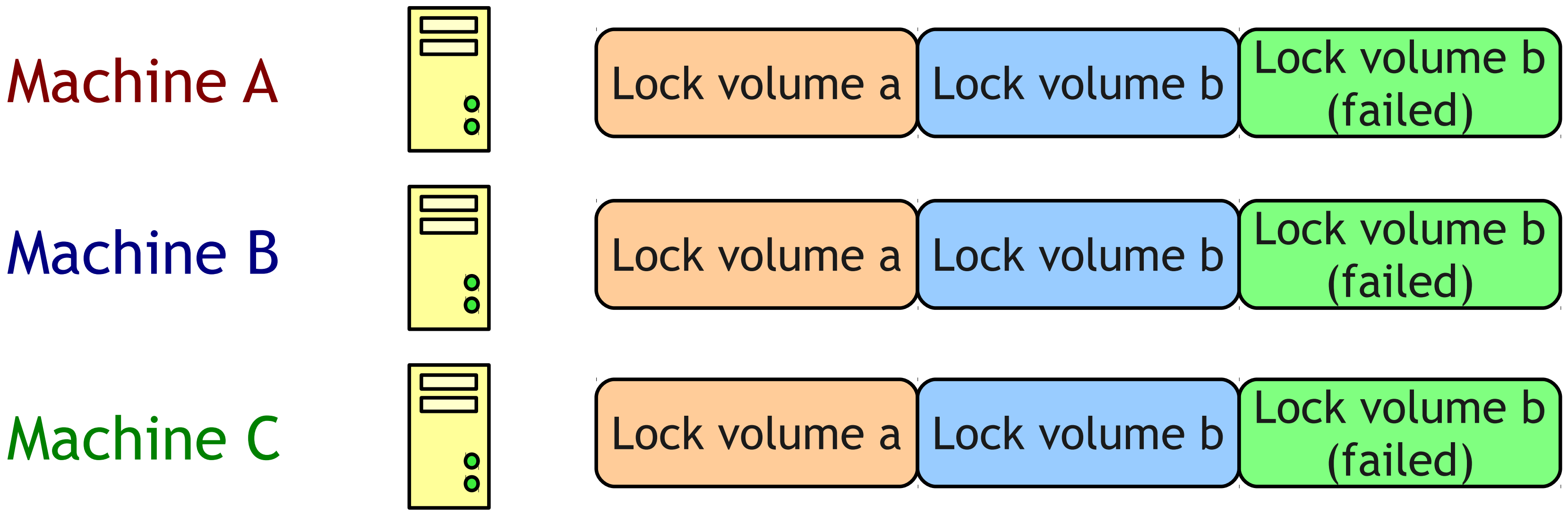
Cluster node management

- Totem ring protocol
  - Dynamic membership management
  - Total order and reliable multi-cast
  - Virtual synchrony

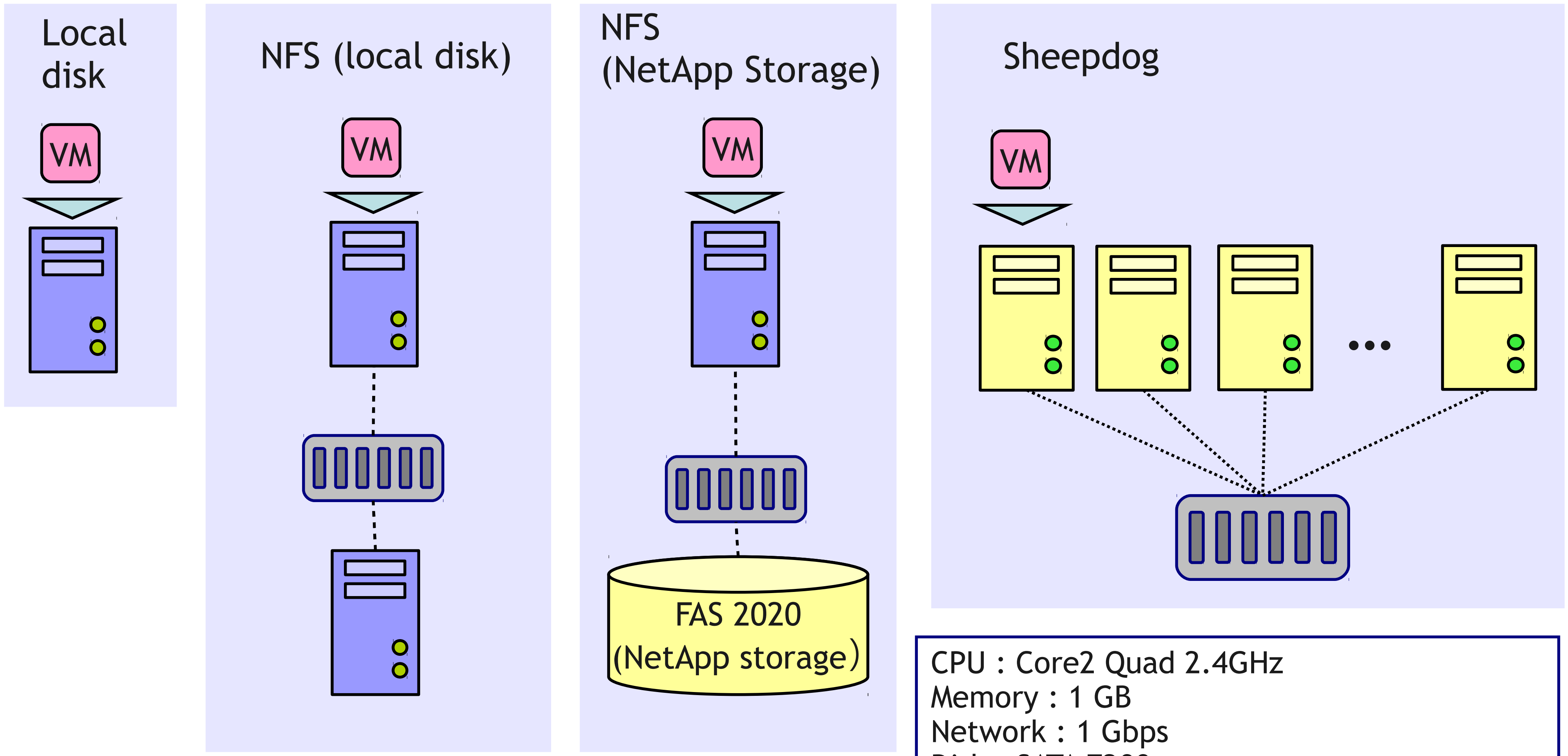


Cluster node management

- Corosync cluster engine
  - Implementation of totem-ring protocol
  - Is adopted by well-known open source projects (Pacemaker, GFS2, etc)
- Sheepdog uses corosync multi-cast to avoid metadata-server



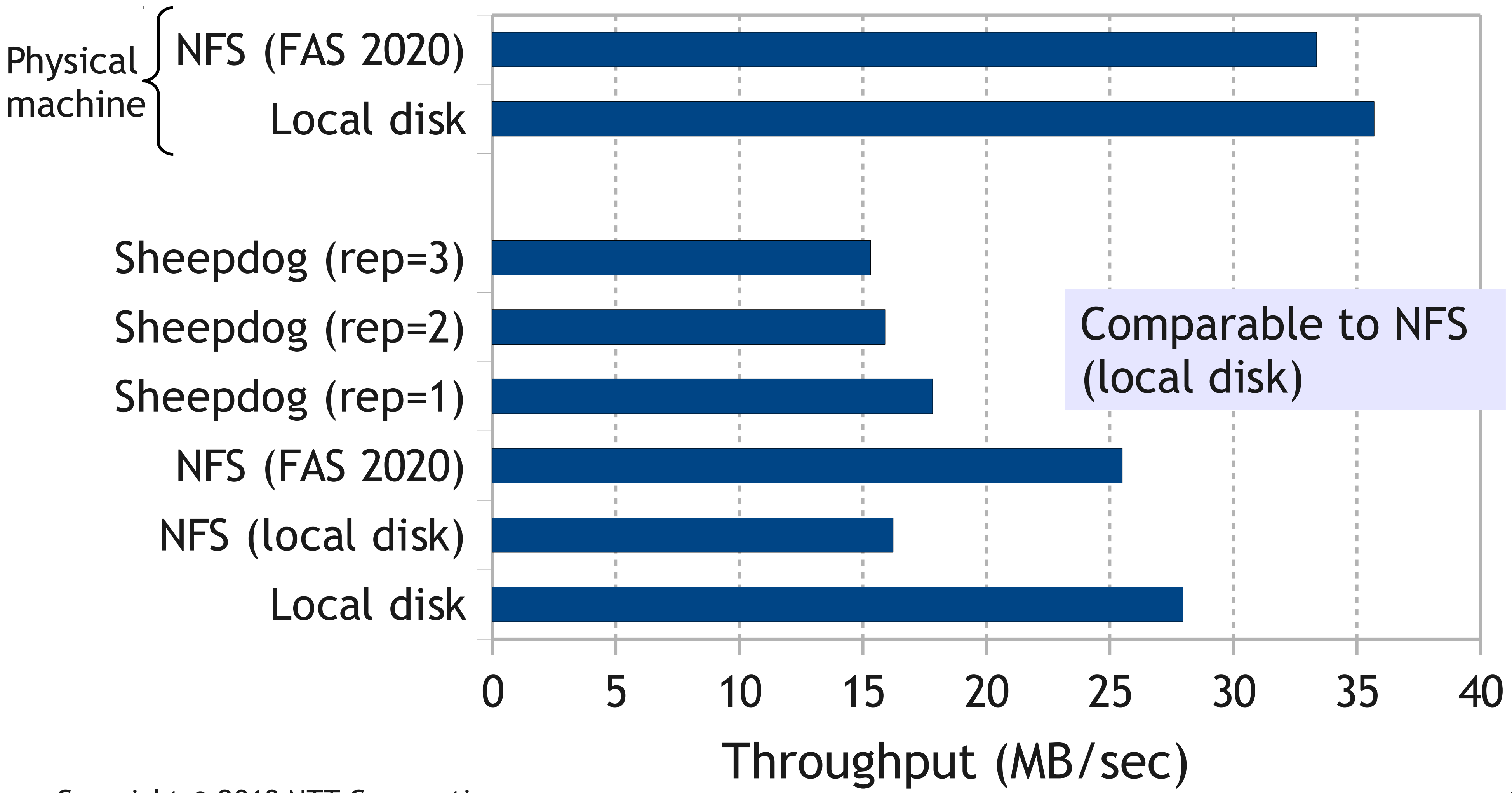
Performance (1 VM)



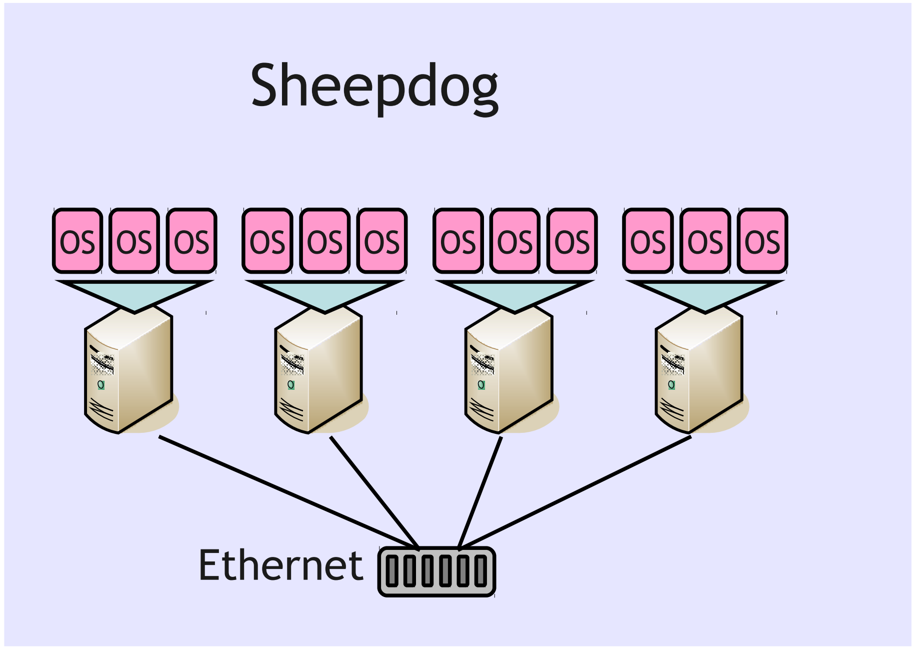
CPU : Core2 Quad 2.4GHz
Memory : 1 GB
Network : 1 Gbps
Disk : SATA 7200 rpm
Machines (Sheepog): 8
Data redundancy (Sheepdog): 1～3

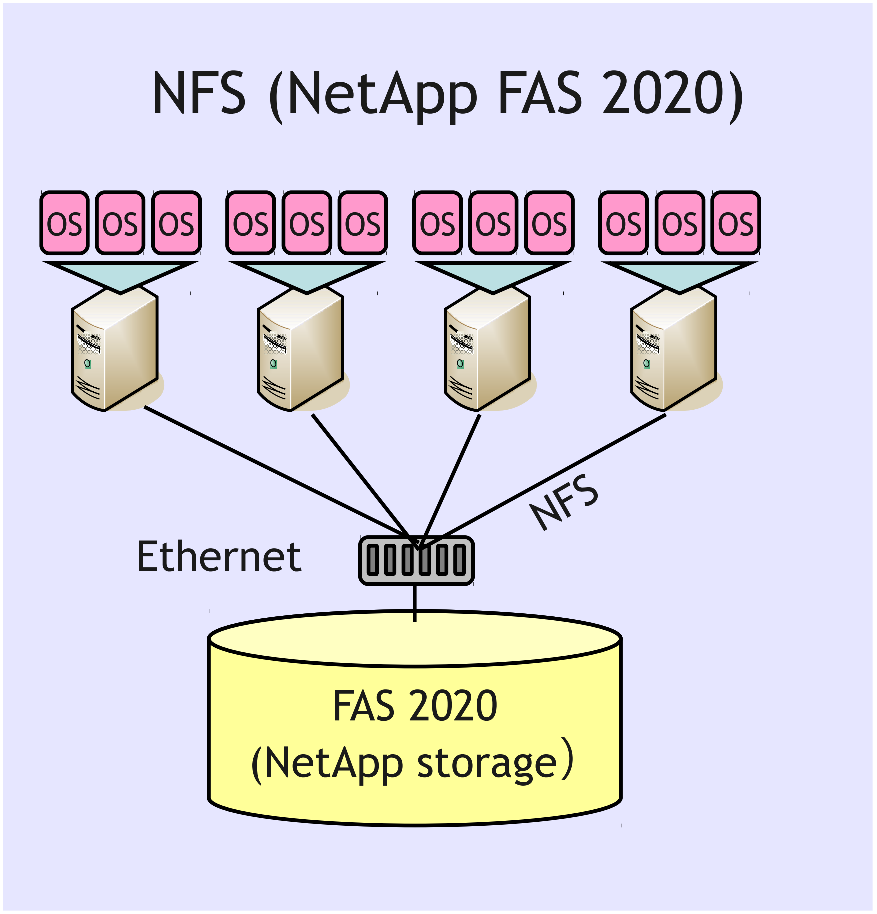
Performance (1 VM)

```
$ dbench -s -S
```



Performance (~ 256 VMs)

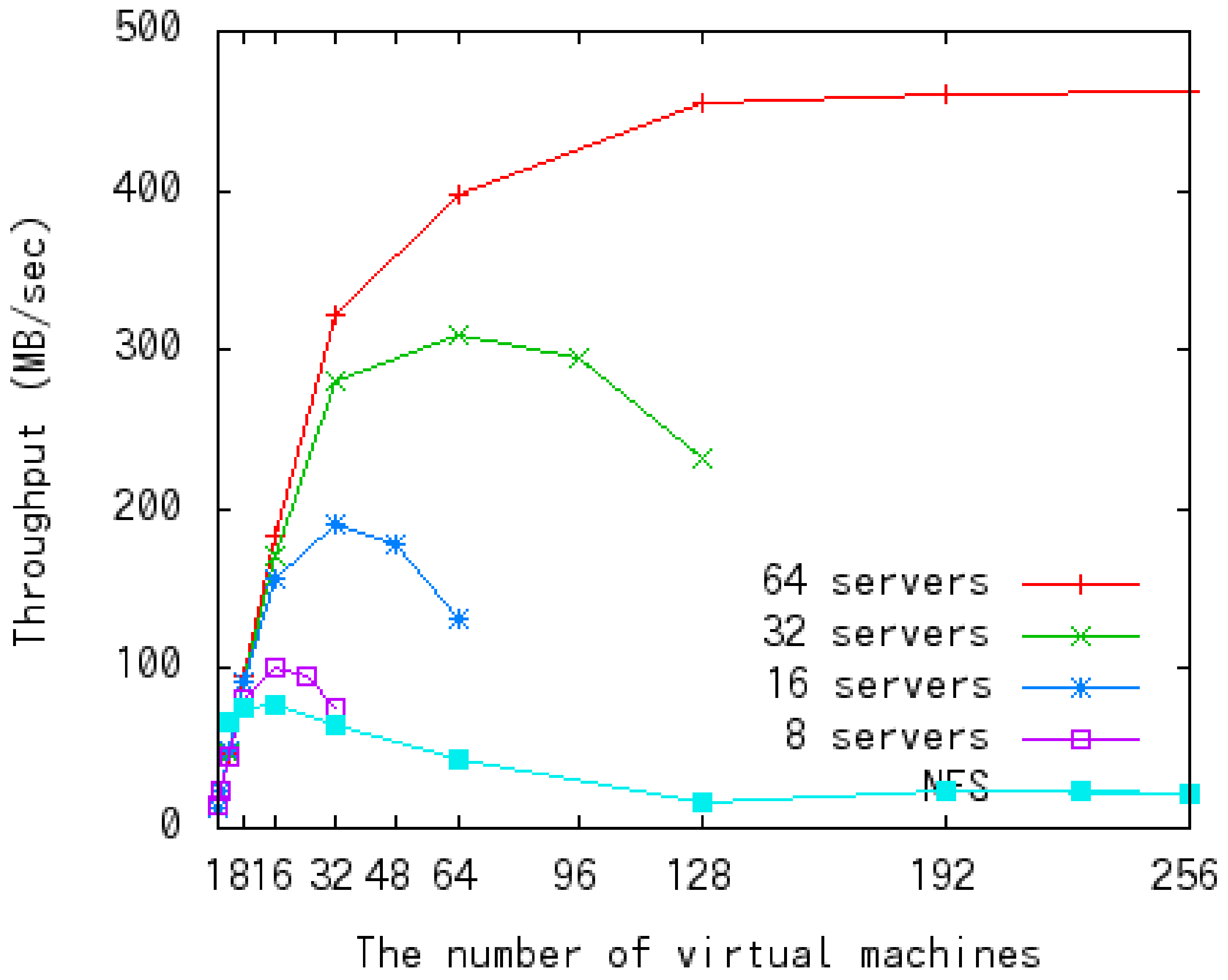




CPU : Core2 Quad 2.4GHz
Memory : 1 GB
Network : 1 Gbps
Disk : SATA 7200 rpm
Host machines : 8 ~ 64
Virtual machines: 1 ~ 256
Data redundancy : 3

Performance (~ 256 VMs)

```
$ dbench -s -S
```



Throughput scales according to the number of host machines

TODO items

- Short-term goals (in few month)
  - More scalability
  - Support libvirt, EBS API
  - Performance improvement
- Long-term goals (in one or two years)
  - guarantee reliability and availability under heavy load
  - tolerance against network partition (split-brain)
  - load balancing corresponding to I/O, CPU, memory load

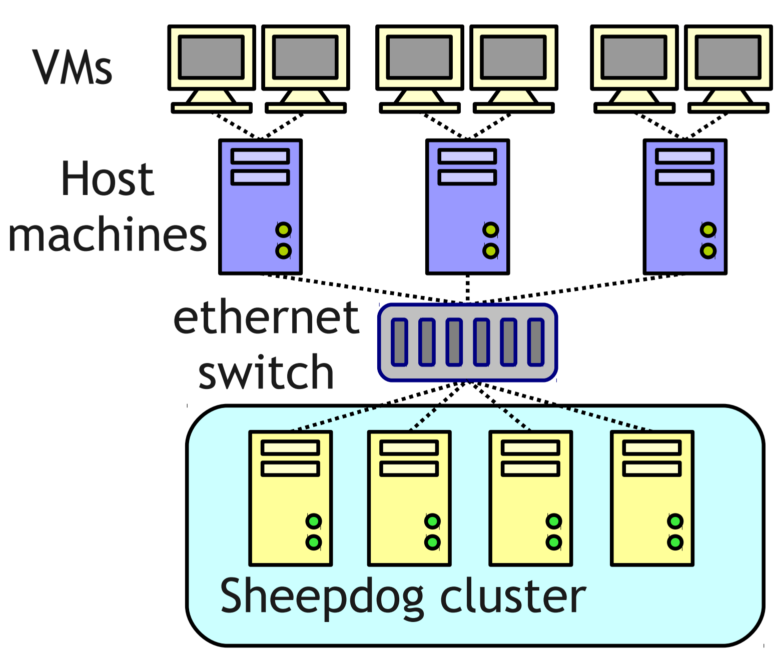
Conclusion

- Sheepdog is scalable, manageable, and reliable storage pool for IaaS environment
  - We hope Sheepdog will become the de facto standard of cloud storage system
- Further information
  - Project page
    - http://www.osrg.net/sheepdog/
  - Mailing list
    - sheepdog@lists.wpkg.org

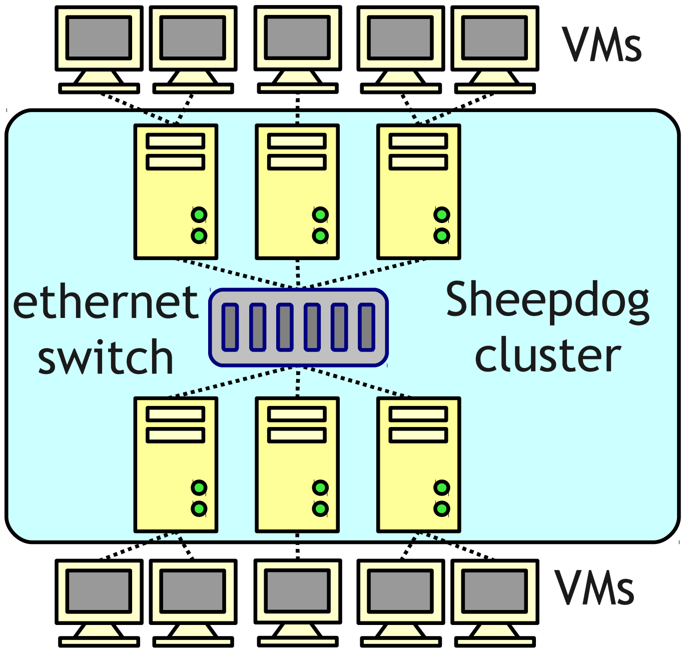
Appendix

Architecture: fully symmetric

- Zero configuration about cluster members
- Similar to Isilon architecture



Use sheepdog as a network storage

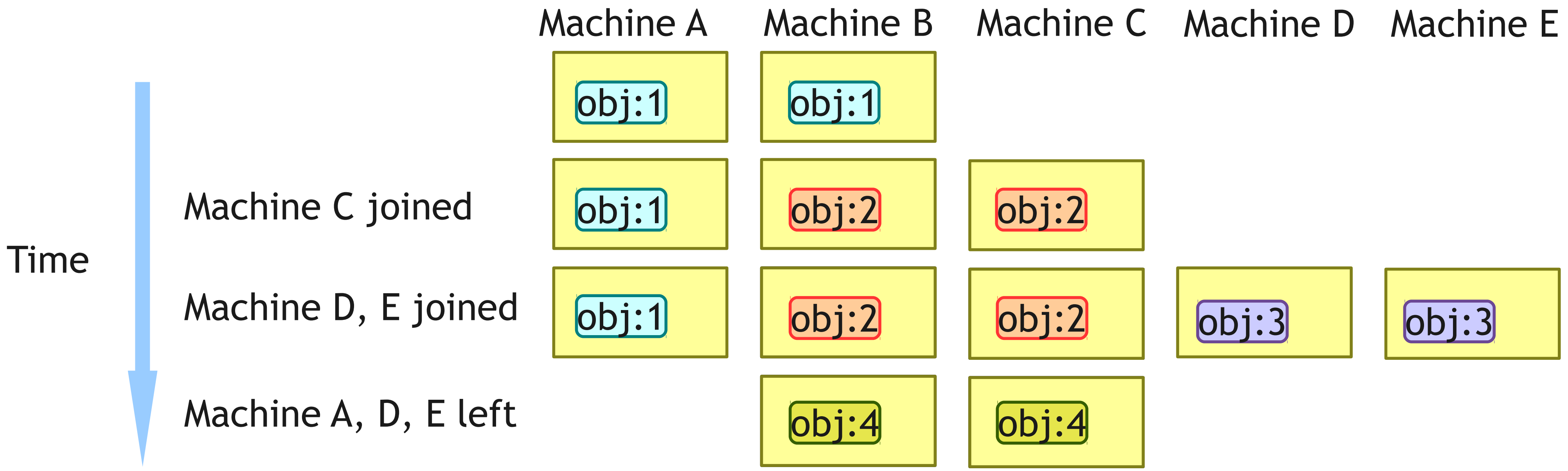


Use sheepdog as a virtual infrastructure

Node membership history

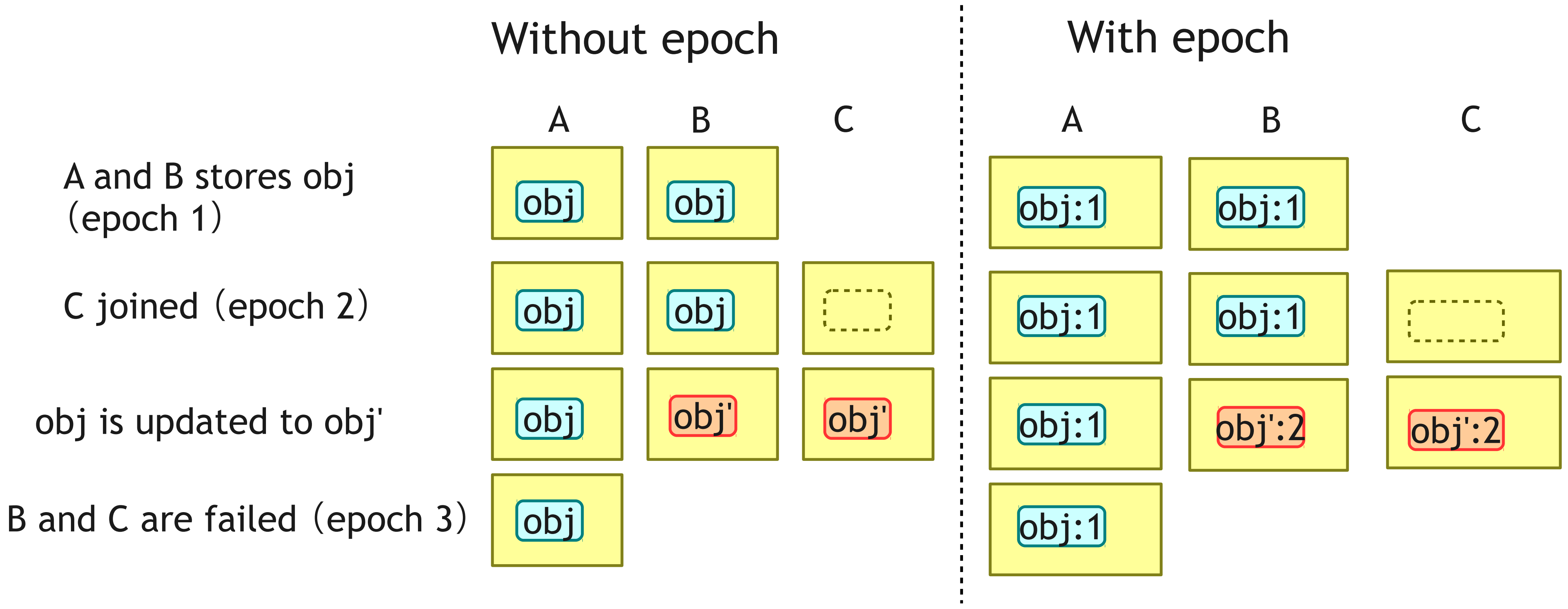
- All nodes stores the history of node membership
- Objects are stored with the version of node membership (epoch)

epoch	Node membership
1	A, B
2	A, B, C
3	A, B, C. D, E
4	B, C



Strong consistency

- Prevent from reading old objects


Without epoch
With epoch
A
B
C
A and B stores obj (epoch 1)
C joined (epoch 2)
obj is updated to obj'
B and C are failed (epoch 3)
obj
obj'
obj:1
obj':2