Networked Storage

Cloud and Virtualization Workshop

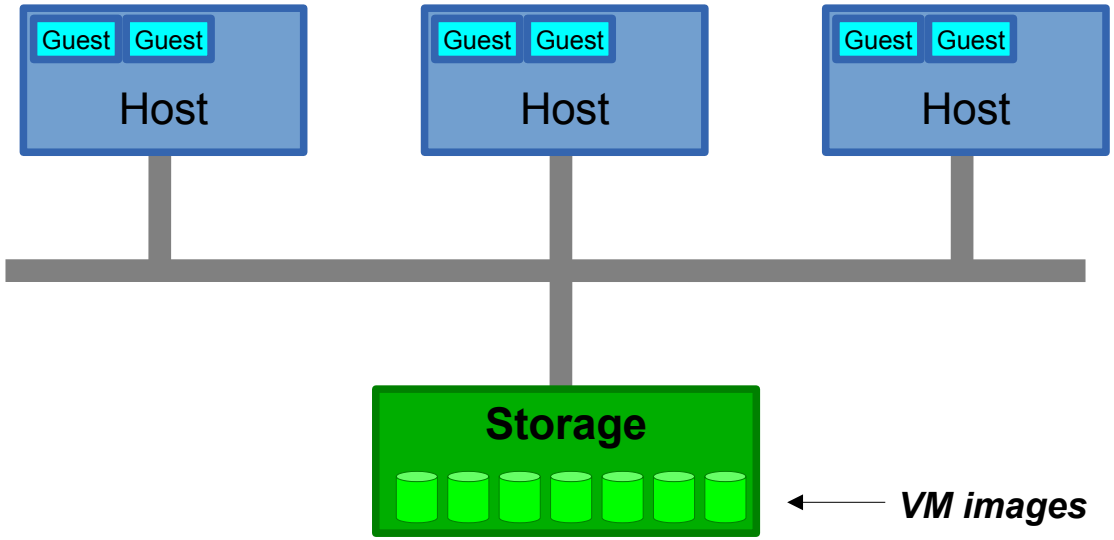
UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Why networked storage?

- Locally-attached storage is fast, but has limitations:
  - Migrating a VM means copying all its storage (slow)
    - increased risks if VM remains in "migrating" state for a long time
  - While a host is down, you lose access to all the data on that host
  - Size of data limited to what fits in one host
  - Lots of separate hosts all need to be backed up
- What if we separate the compute from the storage?

Traditional solution: shared storage



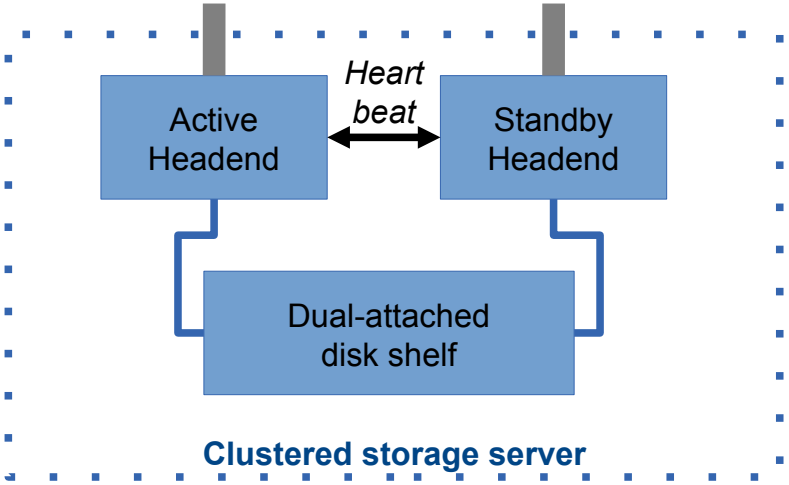
Types of shared storage

- Networked block storage, a.k.a. "SAN"
  - Examples: FC (old), iSCSI, nbd (Linux), NVMe fabrics (new)
  - iSCSI and nbd run over regular ethernet
- Networked file storage, a.k.a. "NAS"
  - Examples: NFS, SMB/CIFS
  - Can also store VM image files, and use them as if they were block devices
- Object storage
  - S3 frontend to local filesystem, e.g. Minio

New Problems!

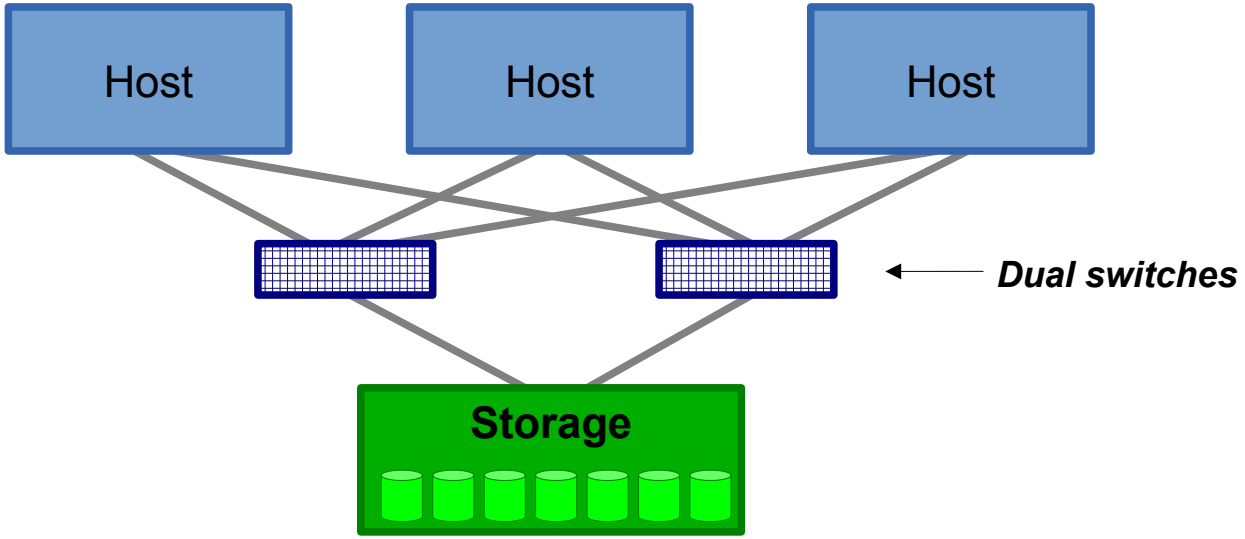
- The storage server is a SPOF (and a bottleneck)
- The storage network is a SPOF (and a bottleneck)
- Total storage capacity and throughput limited by the server
- Some of these can be worked around

Avoiding storage server SPOF



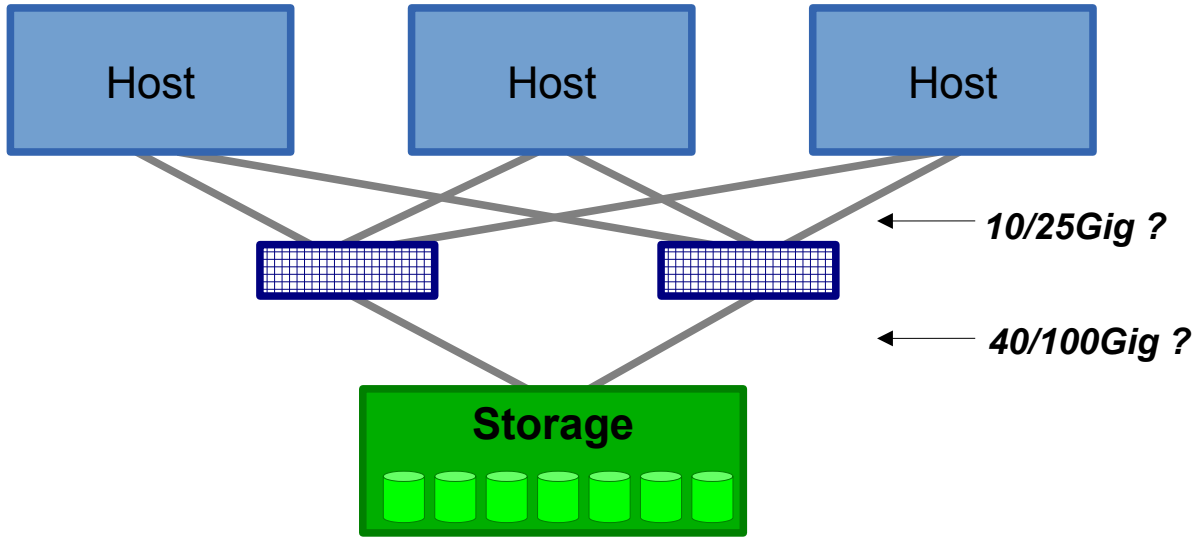
- **This is very hard to build correctly**
- **Vendors will sell this to you for $$$**

Avoiding network SPOF



Or you can buy a *really expensive* chassis switch with multiple power supplies, line cards, switching fabrics and management cards (and still could fail)

Network bandwidth



Note: 1Gbps ≈ 100MB/sec ≈ sequential throughput of a single hard drive

Alternative 1:
Periodic Replication

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Periodic Replication

- Use local storage
- From time to time, copy the VM image to another server
  - e.g. once every hour? once every minute?
- Gives you an "almost" current backup of the VM which you can restart in emergency
- ZFS is particularly good for this
  - Create periodic snapshots (which are useful anyway)
  - Replication copies just the *differences* between snapshots
  - Proxmox has built-in support for this

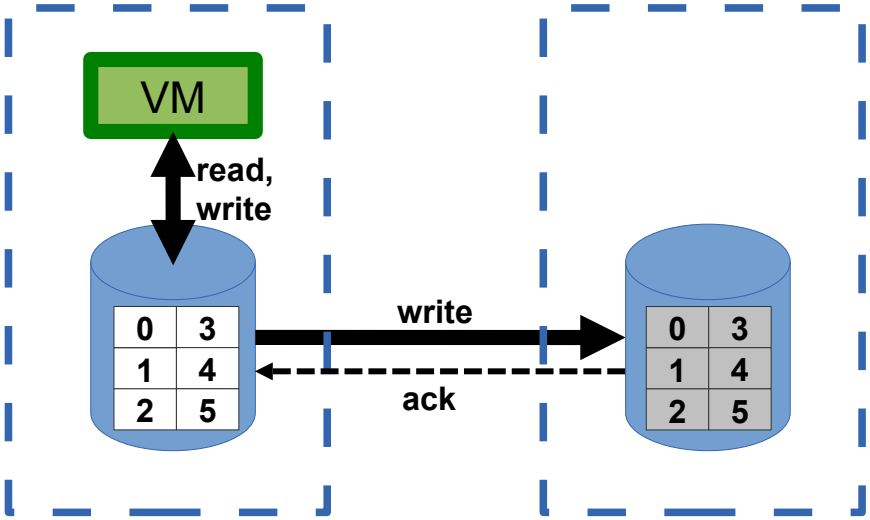
Periodic Replication

- Pros
  - Very easy to use and understand
  - A break in network doesn't stop the VM from running
- Cons
  - Doesn't provide live mobility
    - To move: shutdown VM, refresh the copy, start VM
  - Limited mobility
    - Only to where a copy exists
  - Copy is behind, does not include changes since most recent replication

Alternative 2:
Synchronous Replication

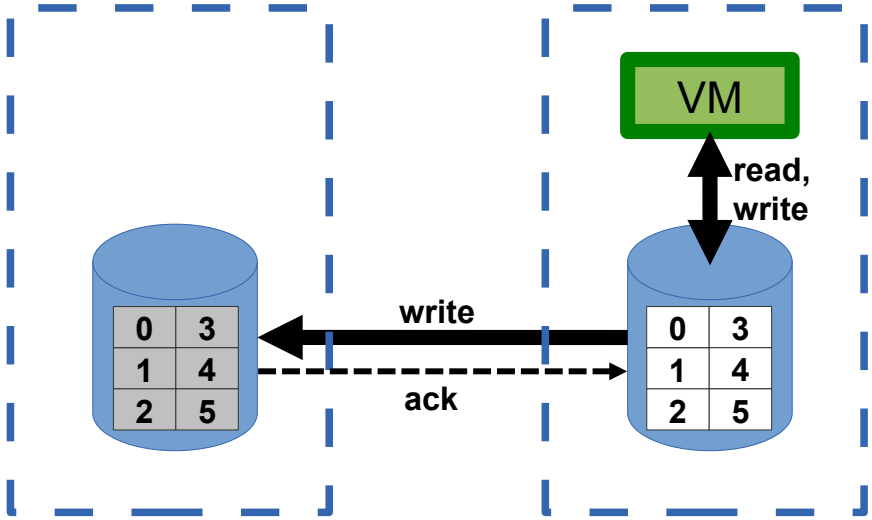
Synchronous Replicated Storage

- **Like "RAID1" over network**
- **Read/write block locally, replicate over network**



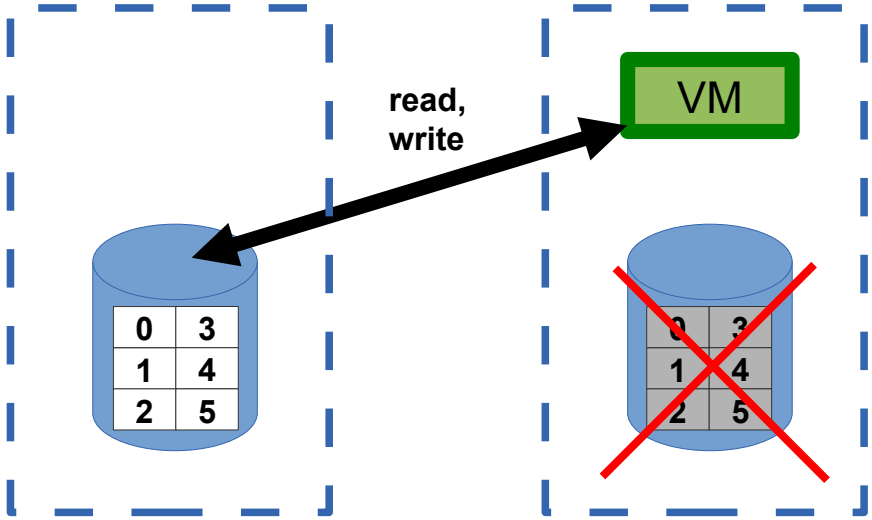
Synchronous Replicated Storage

- **Reverse roles when VM migrates**



Disk failure

- **Transparently redirect all I/O over network**



(Catch up later when disk replaced)

Examples

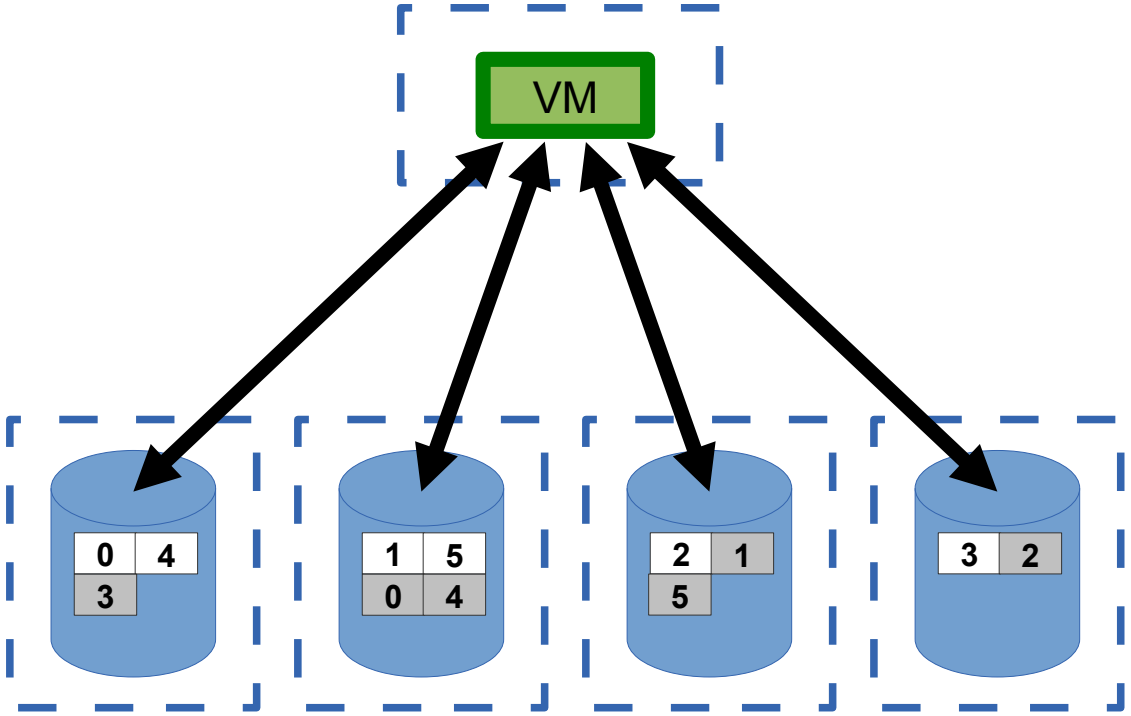
- DRBD (Linux kernel module)
- Out-of-box can configure to replicate entire drive or partition
  - Then you divide up that space: "LVM over DRBD"
- Better if you can replicate individual LVM volumes
  - "DRBD over LVM": a separate DRBD instance replicates each volume
- Ganeti did this
- Linstor can do this
  - From the company that wrote DRBD (and sells support)
  - Proxmox plugin is available

Pros and Cons

- Pros
  - Read access as fast as local disk
  - RAID1-like protection from disk failure
  - Relatively simple to understand
  - Your data is still just blocks in LVM, and in the worst case is retrievable
- Cons
  - VM mobility might be limited to where the replicas are
  - You might only have 2 copies
    - Although Linstor uses DRBD9 which allows up to 32 copies and diskless access
  - Not that widely used

Alternative 3:
Fully Distributed Storage

Distributed Storage



How does it work?

- Client needs to know which server(s) to talk to, for each piece
- Option 1: Central database
  - keeps a map of each data item to its location
  - allows new items to be placed in arbitrary location
  - database grows indefinitely, is a SPOF & performance bottleneck
- Option 2: Hash rings
  - take a hash of the object name or ID, modulo the number of servers, to find the server ID deterministically
  - when you add a new server these results change, and some existing objects need to be moved ("rebalancing")

Distributed Storage

- Your data is split into pieces over *multiple servers*, e.g.
  - Different blocks from your block volume
  - Different files from your file share
  - Different objects in your object bucket
- The pieces can be stored in multiple places for redundancy
  - Or use Erasure Coding: like RAID5/6 across servers
  - Uses less storage space, but much slower
- If a server is down, you can carrying on working
- When server comes back, "healing" catches up missing replicas

Examples

- Block stores
  - Ceph (rbd), Longhorn for kubernetes
- File stores
  - Ceph (cephfs), Lustre, ~~Glusterfs~~ *end of life!*
- Object stores
  - Ceph (radosgw), Swift/Swiftstack, SeaweedFS
- These are free; there are commercial options too
- Beware that many projects have come and gone
  - e.g. sheepdog block storage for KVM

Pros and Cons

- Pros
  - Storage can grow indefinitely: keep adding new servers
- Cons
  - It's a complex system, which itself can fail
    - Simple example: if it runs out of space, all your VMs freeze
    - Cascading failures are possible
    - Performance issues are hard to diagnose
  - You need good engineers and/or dependable third-party support
  - You need well-exercised maintenance procedures
    - e.g. replacing of failed drives and failed servers
  - You need good monitoring and reporting