Scalable Object Storage with Apache CloudStack and Apache Hadoop

February 26 2013

Chiradeep Vittal
@chiradeep

Agenda

- What is CloudStack
- Object Storage for IAAS
- Current Architecture and Limitations
- Requirements for Object Storage
- Object Storage integrations in CloudStack
- HDFS for Object Storage
- Future directions

Apache CloudStack

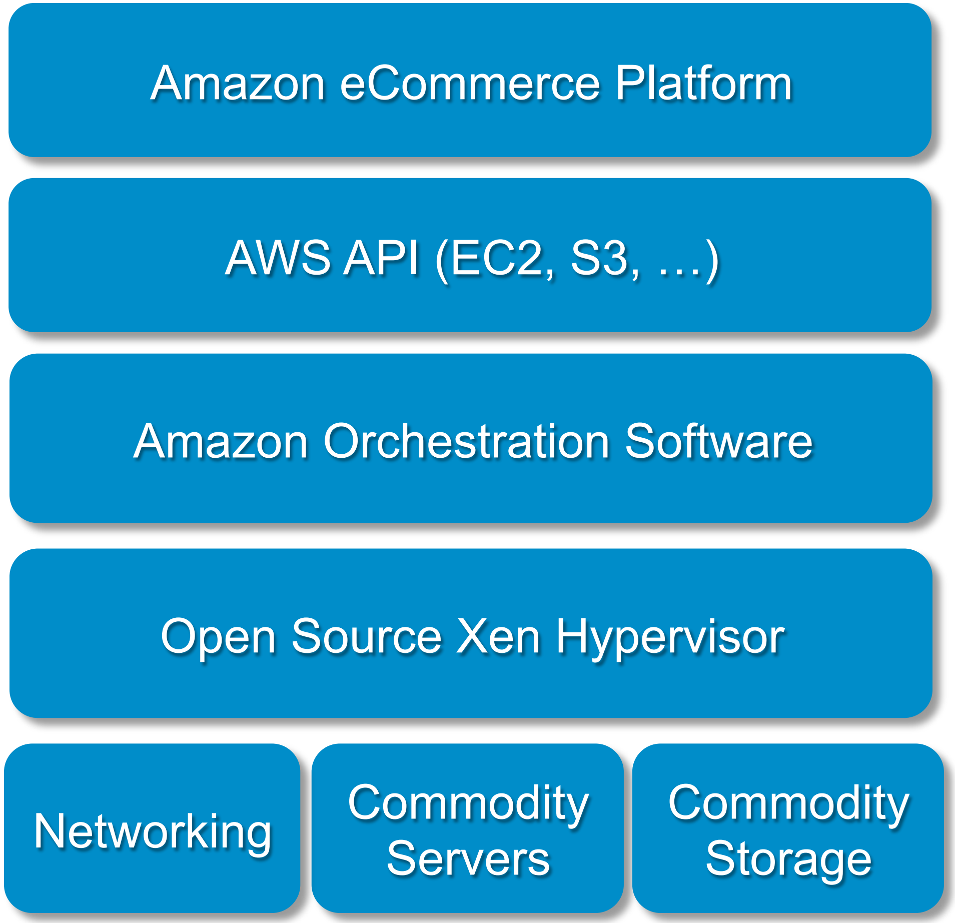
Apache incubator™

cloudstack

Build your cloud the way
the world's most successful
clouds are built

- History
  - Incubating in the Apache Software Foundation since April 2012
  - Open Source since May 2010
- In production since 2009
  - Turnkey platform for delivering IaaS clouds
  - Full featured GUI, end-user API and admin API

How did Amazon build its cloud?



Amazon eCommerce Platform

AWS API (EC2, S3, …)

Amazon Orchestration Software

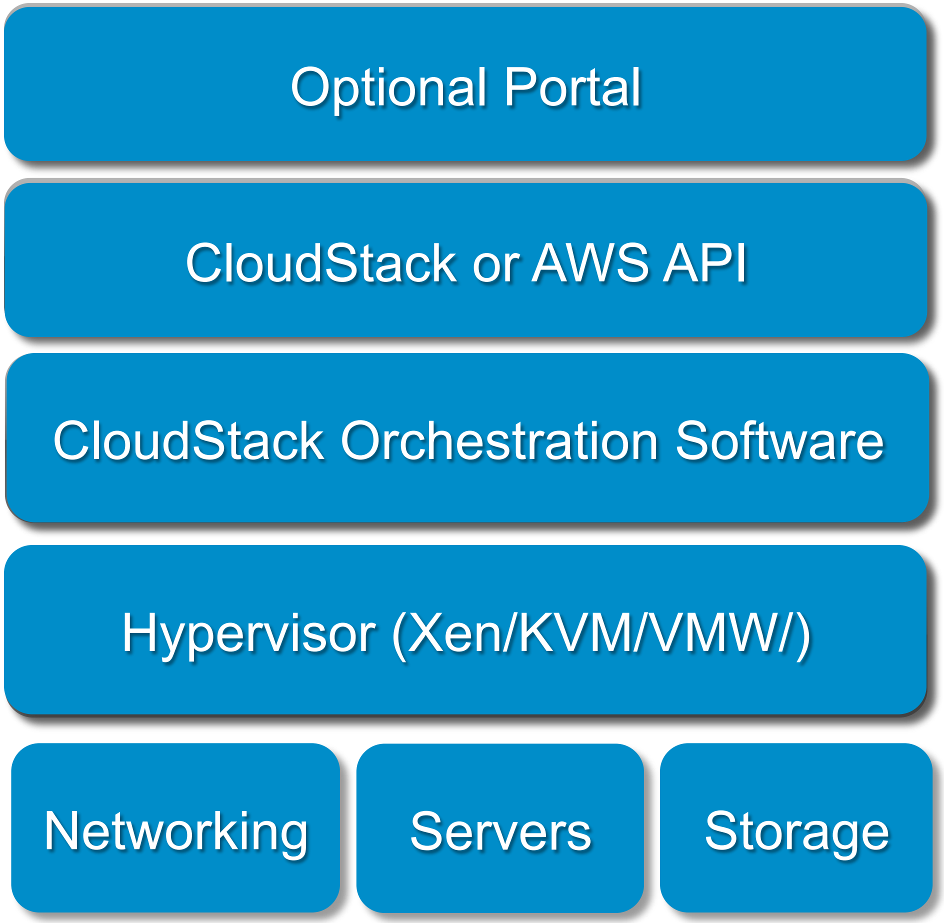
Open Source Xen Hypervisor

Networking

Commodity Servers

Commodity Storage

How can YOU build a cloud?



Optional Portal

CloudStack or AWS API

CloudStack Orchestration Software

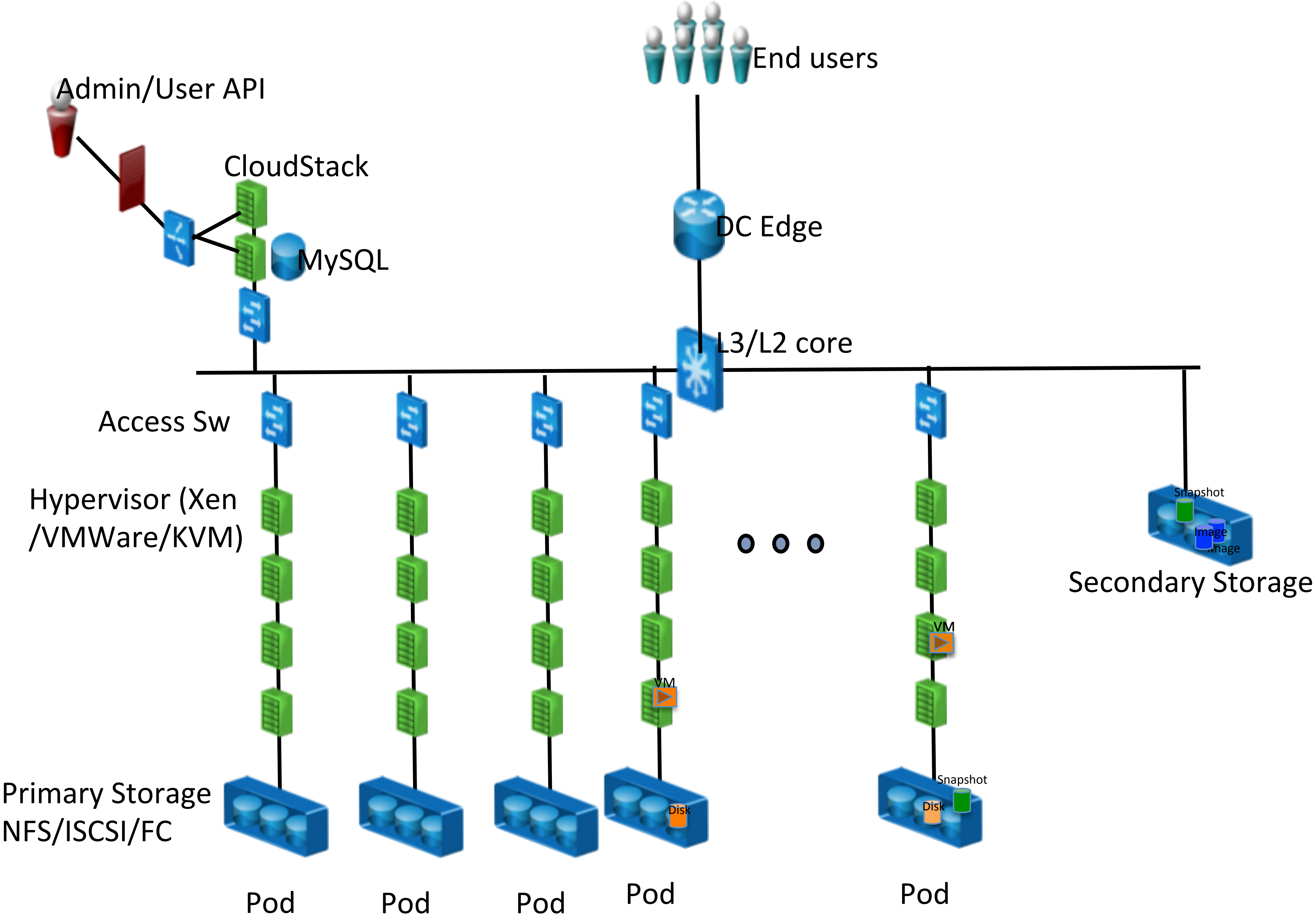
Hypervisor (Xen/KVM/VMW/)

Networking

Servers

Storage

Zone Architecture



Cloud-Style Workloads

- Low cost
  - Standardized, cookie cutter infrastructure
  - Highly automated and efficient
- Application owns availability
  - At scale everything breaks
  - Focus on MTTR instead of MTBF

Scale

"*At scale, everything breaks*"

- Urs Hölzle, Google

8%

Annual Failure Rate of servers

Kashi Venkatesh Vishwanath and Nachiappan Nagappan, **Characterizing Cloud Computing Hardware Reliability,** *SoCC'10*

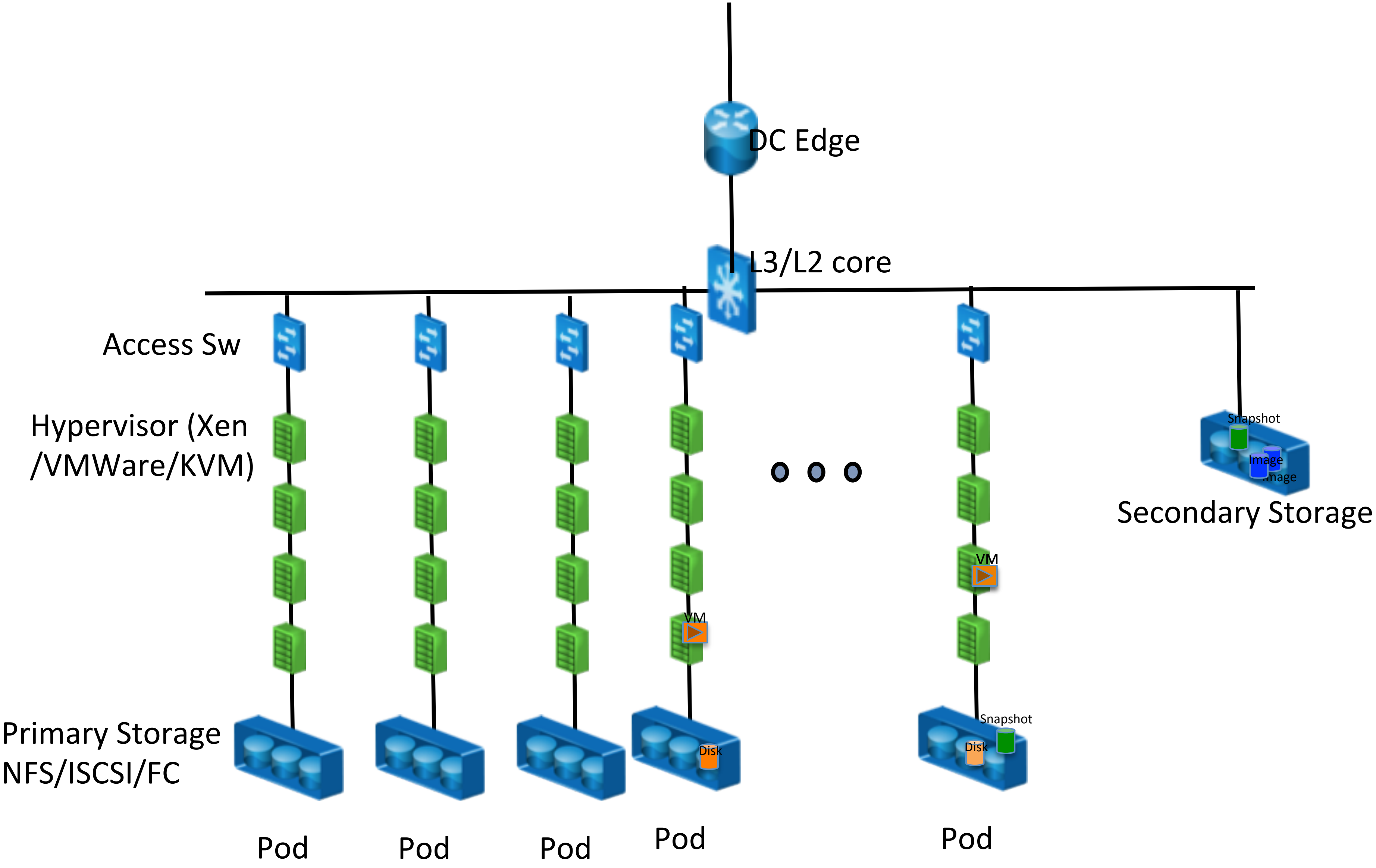
Server failure comes from:

- 70% - hard disk
- 6% - RAID controller
- 5% - memory
- 18% - other factors

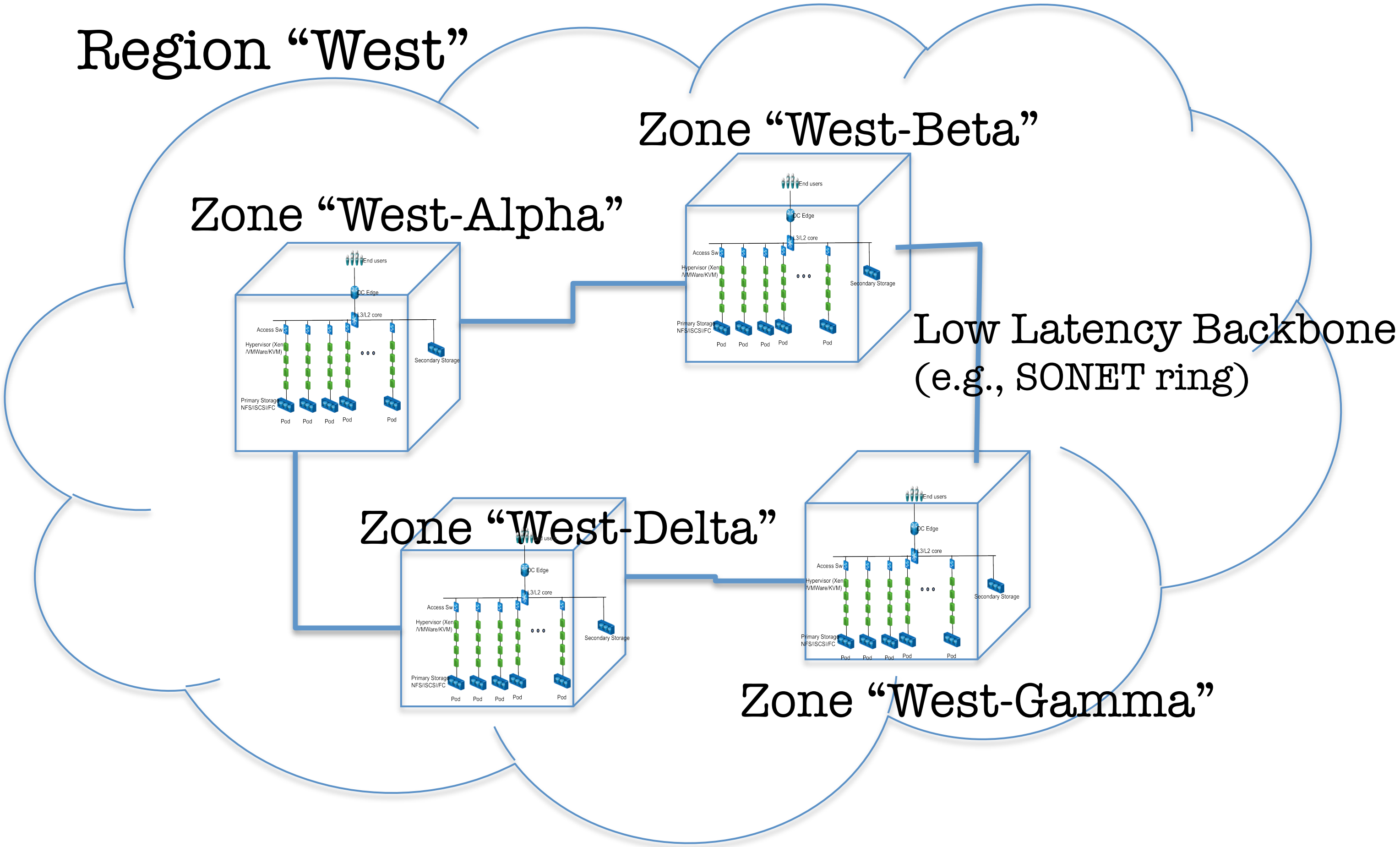
Application can still fail for other reasons:

- Network failure
- Software bugs
- Human admin error

At scale…everything breaks



Regions and zones



Region “West”

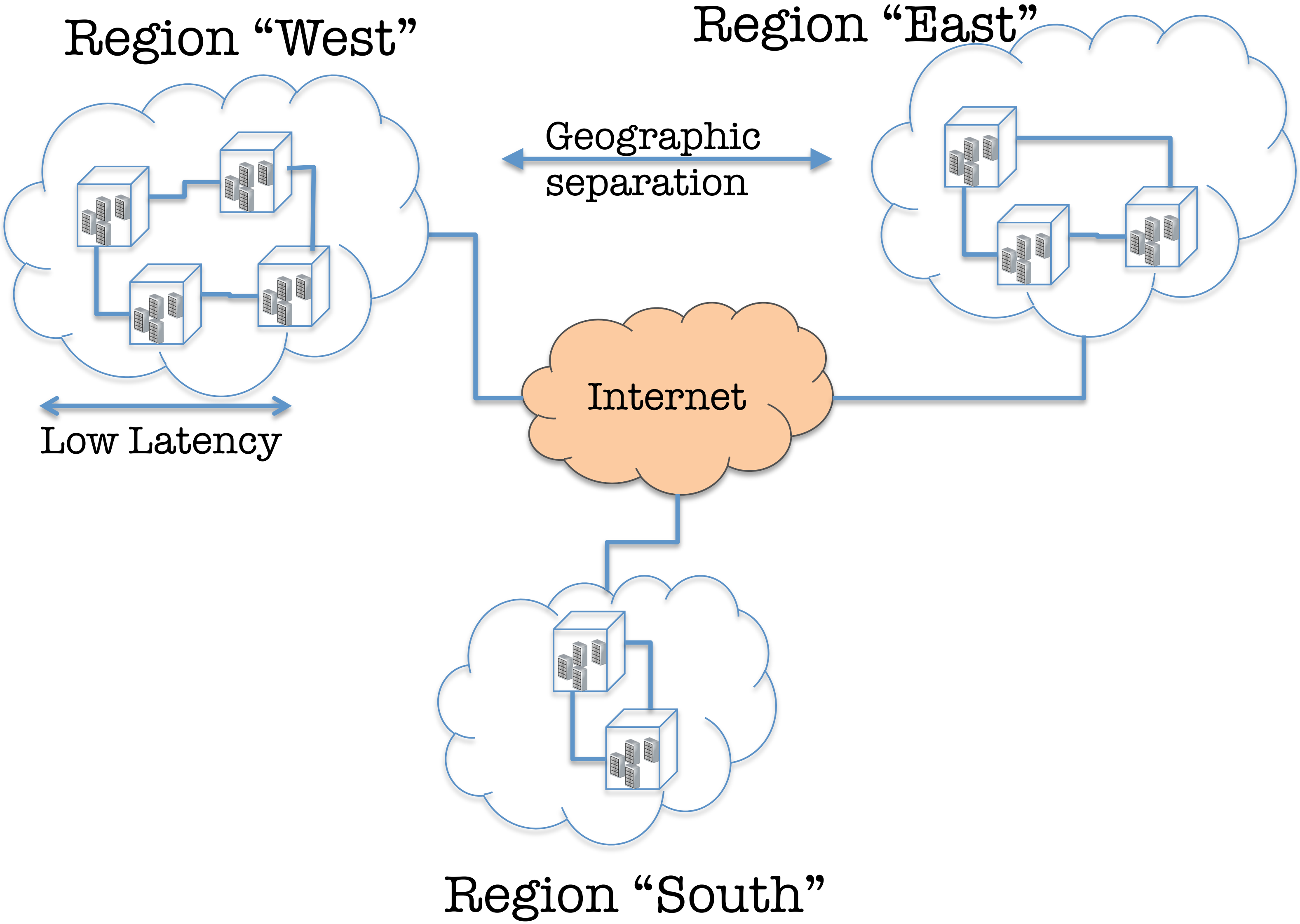
Zone “West-Alpha”

Zone “West-Beta”

Zone “West-Delta”

Zone “West-Gamma”

Low Latency Backbone
(e.g., SONET ring)



Region “West”

Region “East”

Geographic separation

Internet

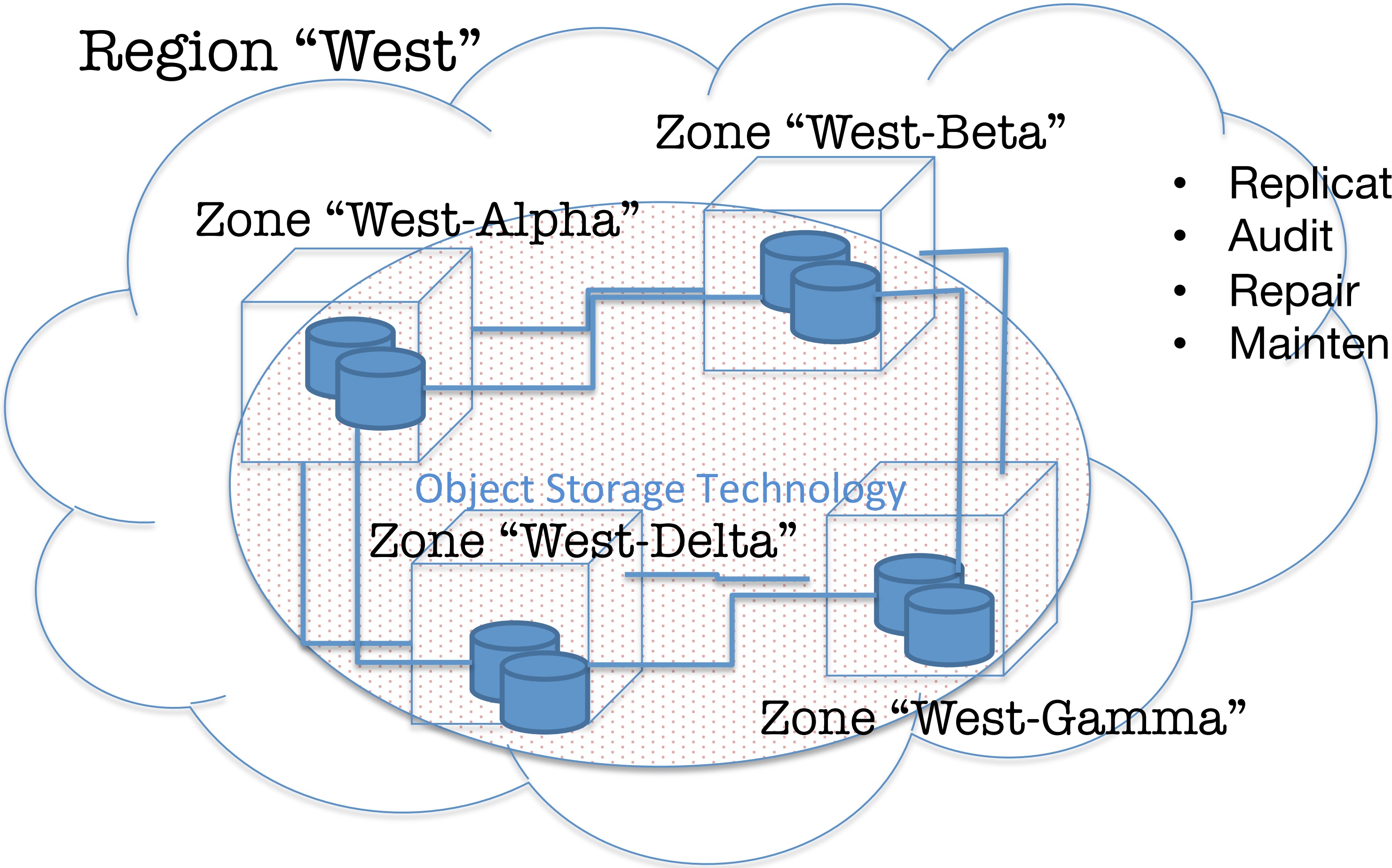
Low Latency

Region “South”

Secondary Storage in CloudStack 4.0

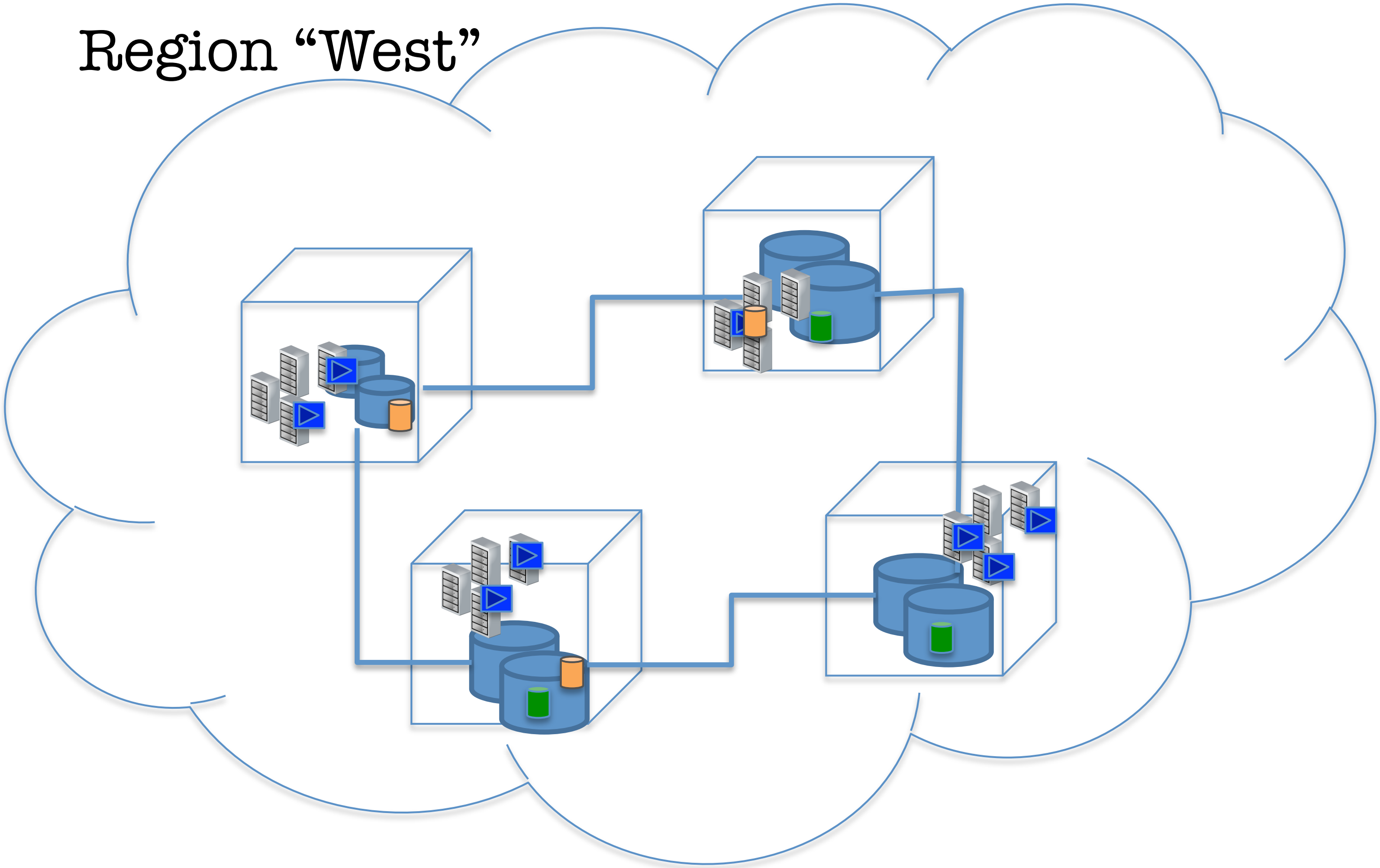
- NFS server default
  - can be mounted by hypervisor
  - Easy to obtain, set up and operate
- Problems with NFS:
  - Scale: max limits of file systems
    - Solution: CloudStack can manage multiple NFS stores (+ complexity)
  - Performance
    - N hypervisors : 1 storage CPU / 1 network link
  - Wide area suitability for cross-region storage
    - Chatty protocol
  - Lack of replication

Object Storage in a region



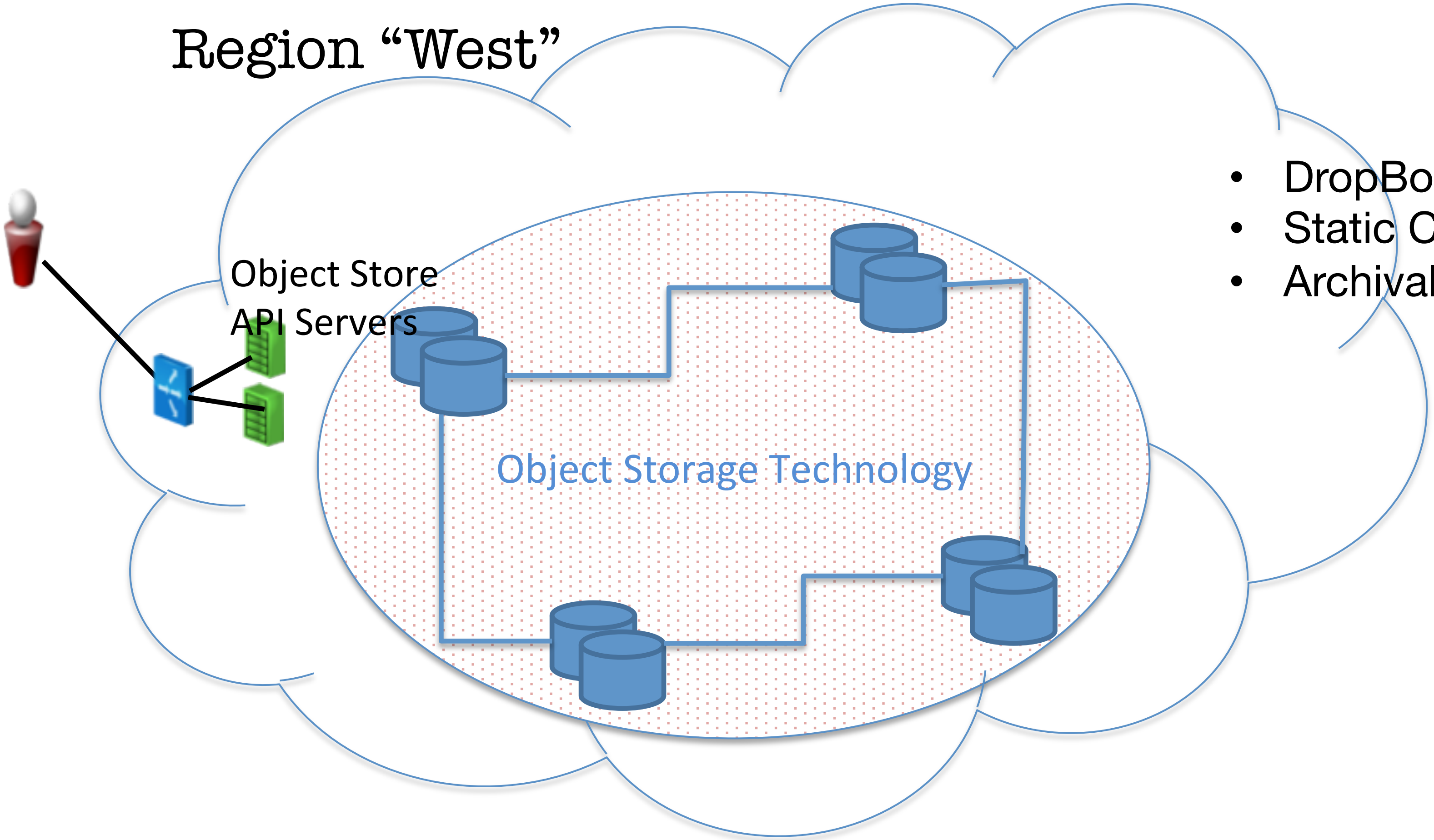
- Replication
- Audit
- Repair
- Maintenance

Object Storage enables reliability



Region “West”

Object Storage also enables other applications

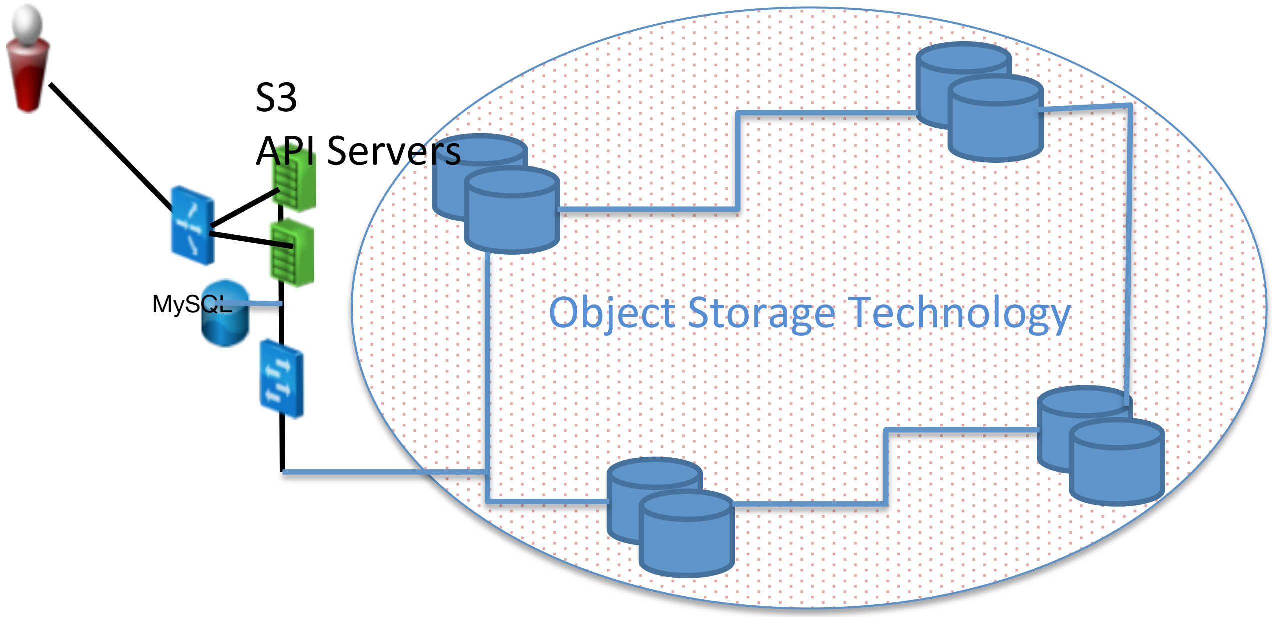


- DropBox
- Static Content
- Archival

Object Storage characteristics

- Highly reliable and durable
  - 99.9 % availability for AWS S3
  - 99.999999999 % durability
- Massive scale
  - 1.3 trillion objects stored across 7 AWS regions [Nov 2012 figures]
  - Throughput: 830,000 requests per second
- Immutable objects
  - Objects cannot be modified, only deleted
- Simple API
  - PUT/POST objects, GET objects, DELETE objects
  - No seek / no mutation / no POSIX API
- Flat namespace
  - Everything stored in buckets.
  - Bucket names are unique
  - Buckets can only contain objects, not other buckets
- Cheap and getting cheaper

CloudStack S3 API Server



CloudStack S3 API Server

- Understands AWS S3 REST-style and SOAP API
- Pluggable backend
  - Backend storage needs to map simple calls to their API
    - E.g., `createContainer`, `saveObject`, `loadObject`
  - Default backend is a POSIX filesystem
  - Backend with Caringo Object Store (commercial vendor) available
  - HDFS backend also available
- MySQL storage
  - Bucket -> object mapping
  - ACLs, bucket policies

Object Store Integration into CloudStack

- For images and snapshots
- Replacement for NFS secondary storage

Or

Augmentation for NFS secondary storage

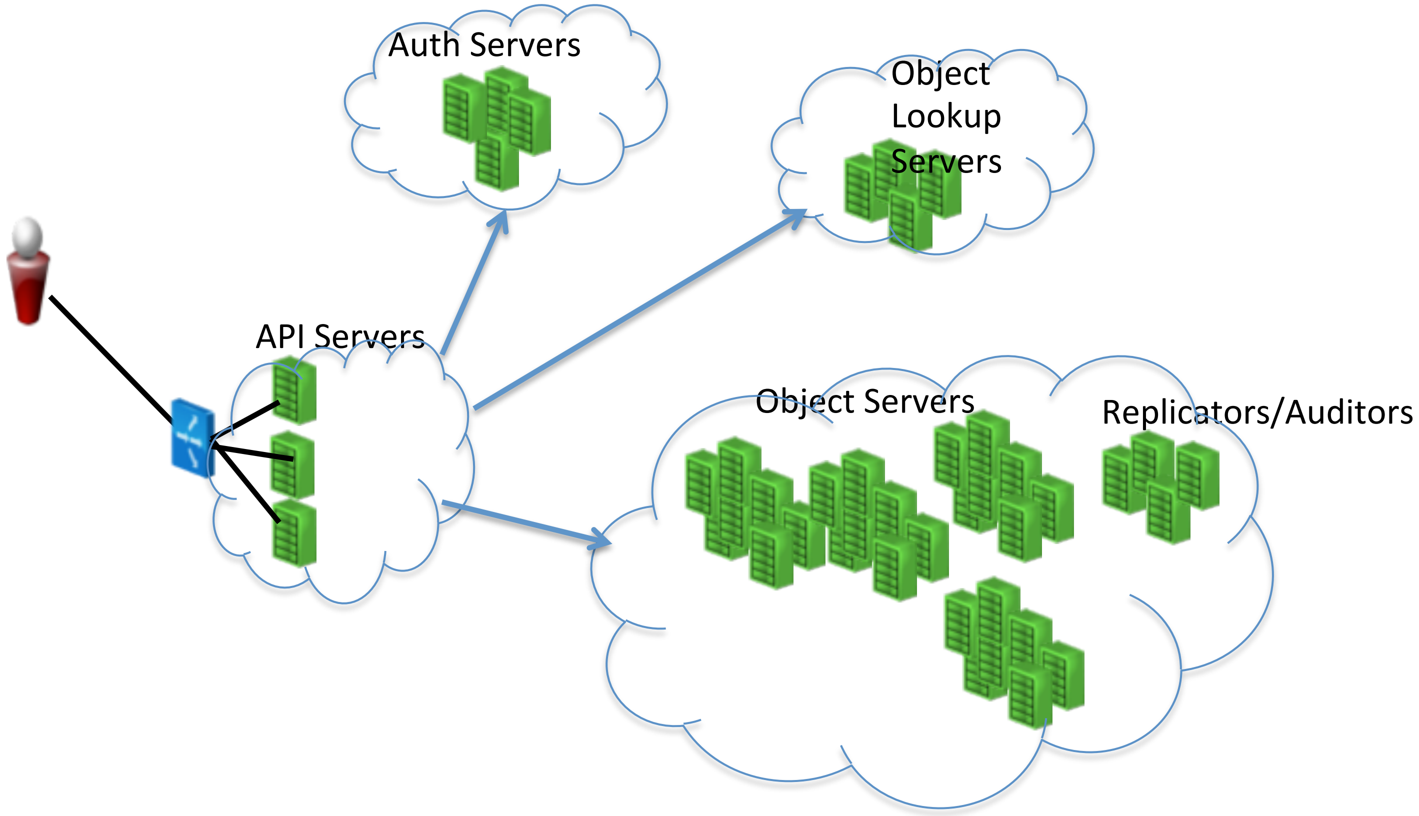
- Integrations available with
  - Riak CS
  - Openstack Swift
- New in 4.2 (upcoming):
  - Framework for integrating storage providers

What do we want to build ?

- Open source, ASL licensed object storage
- Scales to at least 1 billion objects
- Reliability and durability on par with S3
- S3 API (or similar, e.g., Google Storage)
- Tooling around maintenance and operation, specific to object storage

The following slides are a design discussion

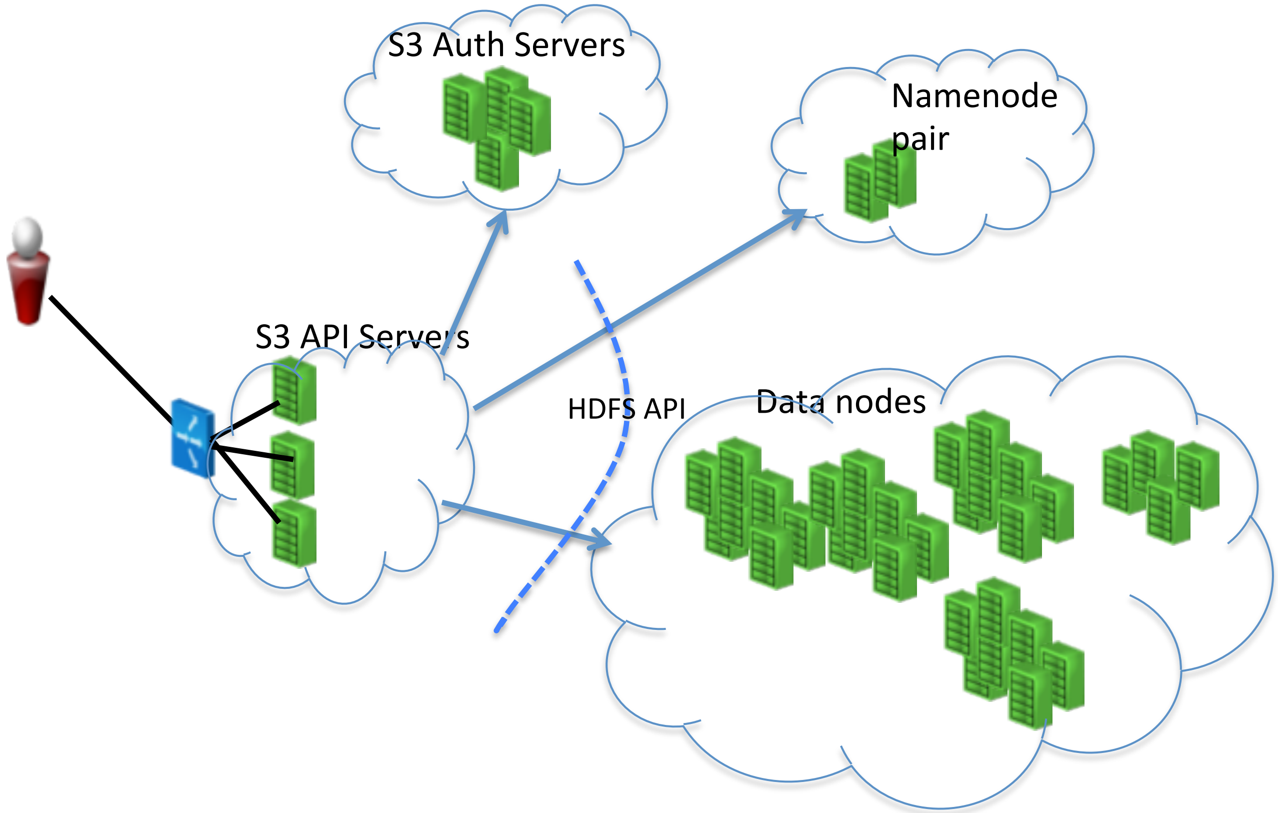
Architecture of Scalable Object Storage



Why HDFS

- ASF Project (Apache Hadoop)
- Immutable objects, replication
- Reliability, scale and performance
  - 200 million objects in 1 cluster [Facebook]
  - 100 PB in 1 cluster [Facebook]
- Simple operation
  - Just add data nodes

HDFS-based Object Storage



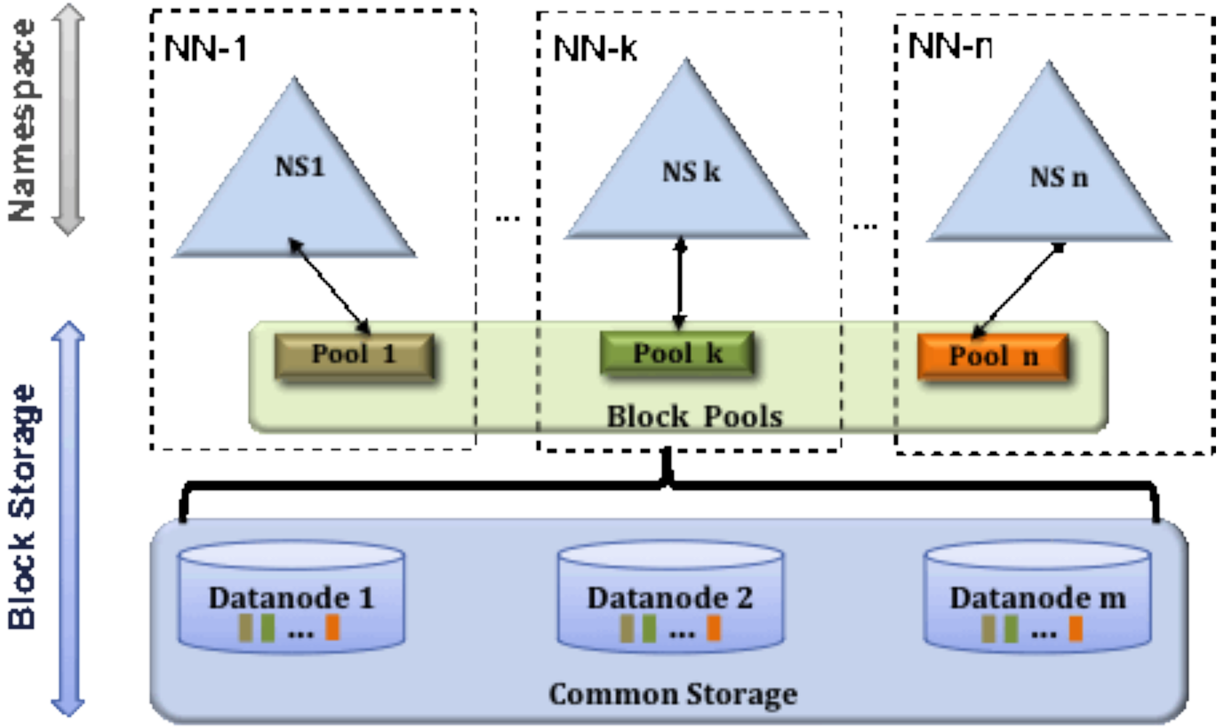
BUT

- Name Node Scalability
  - 150 bytes RAM / block
  - GC issues
- Name Node SPOF
  - Being addressed in the community✔
- Cross-zone replication
  - Rack-awareness placement ✔
  - What if the zones are spread a little further apart?
- Storage for object metadata
  - ACLs, policies, timers

Name Node scalability

- 1 billion objects = 3 billion blocks (chunks)
  - Average of 5 MB/object = 5 PB (actual), 15 PB (raw)
  - 450 GB of RAM per Name Node
    - 150b x 3 x 10^9
  - 16 TB / node => 1000 Data nodes
- Requires Name Node federation ?
- Or an approach like HAR files

Name Node Federation



Extension: Federated NameNodes are HA pairs

Federation issues

- HA for name nodes
- Namespace shards
  - Map object -> name node
    - Requires another scalable key-value store
      - HBase?
- Rebalancing between name nodes

Replication over lossy/slower links

A. Asynchronous replication
   - Use `distcp` to replicate between clusters
   - 6 copies *vs*. 3
   - Master/Slave relationship
     - Possibility of loss of data during failover
     - Need coordination logic outside of HDFS

B. Synchronous replication
   - API server writes to 2 clusters and acks only when both writes are successful
   - Availability compromised when one zone is down

CAP Theorem

Consistency *or* Availability during partition

Many nuances

Storage for object metadata

A. Store it in HDFS along with the object
   - Reads are expensive (e.g., to check ACL)
   - Mutable data, needs layer over HDFS

B. Use another storage system (e.g. HBase)
   - Name node federation also requires this.

C. Modify Name Node to store metadata
   - High performance
   - Not extensible

Object store on HDFS Future

- Viable for small-sized deployments
  - Up to 100-200 million objects
  - Datacenters close together
- Larger deployments needs development
  - No effort ongoing at this time

Conclusion

- CloudStack needs object storage for “cloud-style” workloads
- Object Storage is not easy
- HDFS comes close but not close enough
- Join the community!