HDFS What is New and Futures

Sanjay Radia, Founder, Architect
Suresh Srinivas, Founder, Architect
Hortonworks Inc.

About me

- **Founder, Architect, Hortonworks**

- **Part of the Hadoop team at Yahoo! since 2007**
  - ***Chief Architect*** of ***Hadoop Core*** at Yahoo!
  - Apache Hadoop ***PMC*** and ***Committer***

- **Prior**
  - Data center automation, virtualization, Java, HA, OSs, File Systems (Startup, Sun Microsystems, …)
  - Ph.D., University of Waterloo

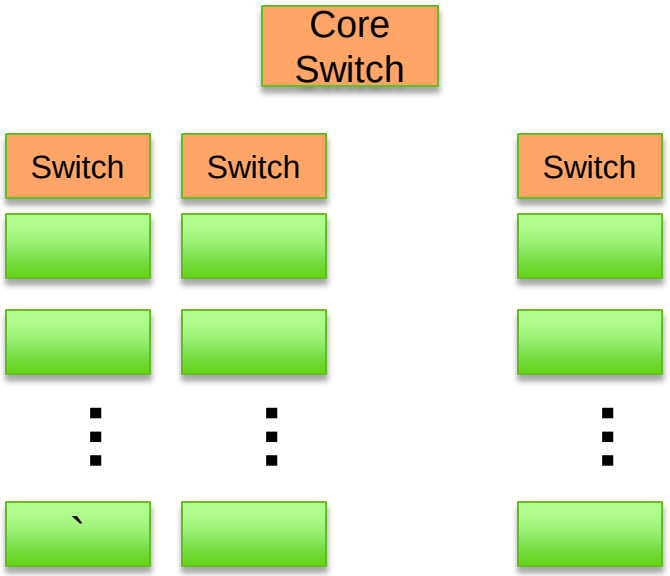
Agenda

- **Hadoop 2.0 – What's new**
  - Federation
  - HA
  - Snapshots
  - Other features
- **Future**
  - Major Architectural Directions
  - Short term and long term features

HDFS: Scalable, Reliable, Manageable

Scale IO, Storage, CPU

- Add ***commodity*** servers & ***JBODs***
- 6K nodes in cluster, 120PB



- Fault Tolerant & Easy management
  - Built in redundancy
  - Tolerate disk and node failures
  - Automatically manage addition/removal of nodes
  - One operator per 3K node!!
- Storage server used for computation
  - Move computation to data
- Not a SAN
  - But high-bandwidth network access to data via Ethernet
- Scalable file system
  - Read, Write, rename, append

    No random writes

Simplicity of design

why a small team could build such a large system in the first place

We have been hard at work…

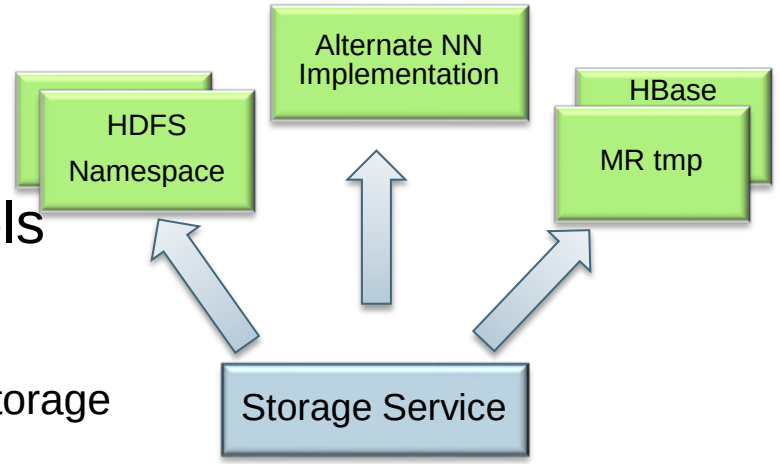
- **Progress is being made in many areas**
  - Write-pipeline, Append
  - Scalability
  - Performance
  - Enterprise features
  - Ongoing operability improvements
  - Enhancements for other projects in the ecosystem
  - Expand Hadoop ecosystem to more platforms and use cases
- **2192 commits in Hadoop in the last year**
  - Almost a million lines of changes
  - ~150 contributors
  - Lot of new contributors - ~80 with < 3 patches
- **350K lines of changes in HDFS and common**

Building on Rock-solid Foundation

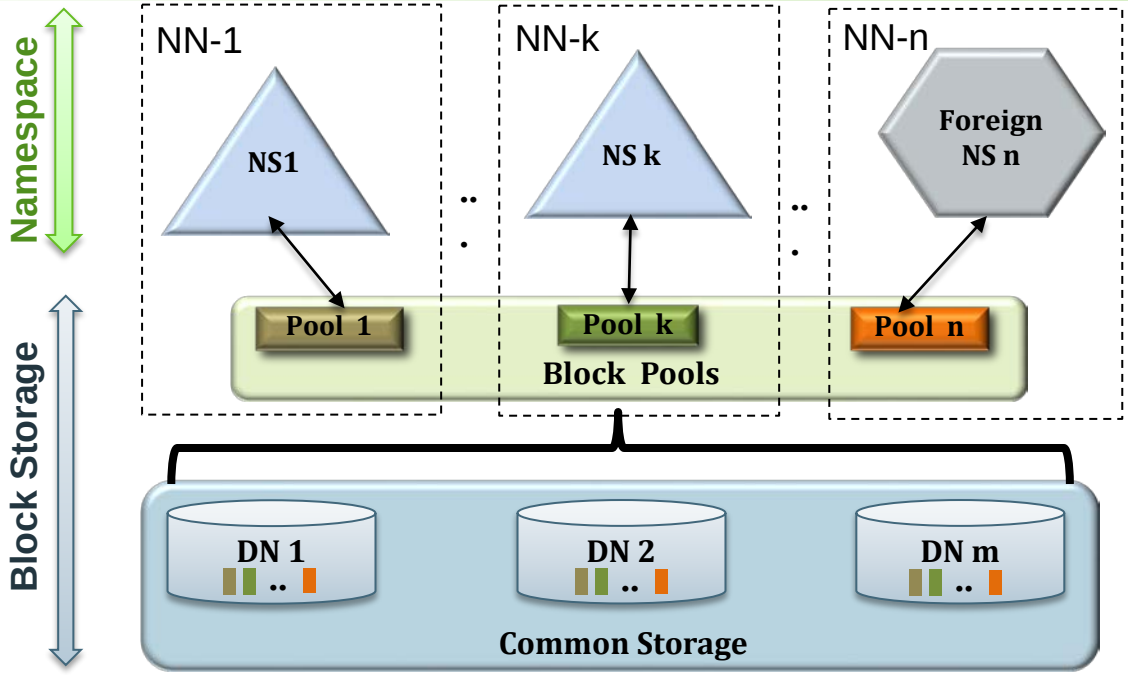
- Original design choices - simple and robust
  - Storage: Rely in OS's file system rather than use raw disk
  - Storage Fault Tolerance: multiple replicas, active monitoring
  - Namenode Master
- Reliability
  - Over 7 9's of data reliability, less than 0.58 failures across 25 clusters
- Operability
  - Small teams can manage large clusters
    - An operator per 3K node cluster
  - Fast Time to repair on node or disk failure
    - Minutes to an hour Vs. RAID array repairs taking many long hours
- Scalable - proven by large scale deployments not bits
  - > 100 PB storage, > 500 million files, > 4500 nodes in a single cluster
  - > 60 K nodes of HDFS in deployment and use

HDFS' Generic Storage Service Opportunities for Innovation

- Federation - Distributed (Partitioned) Namespace
  - Simple and Robust due to independent masters
  - Scalability, Isolation, Availability
- New Services – Independent Block Pools
  - New FS - Partial namespace in memory
  - MR Tmp storage, Object store directly on block storage
  - Shadow file system – caches HDFS, NFS, S3
- Future: move Block Management in DataNodes
  - Simplifies namespace/application implementation
  - Distributed namenode becomes significantly simple



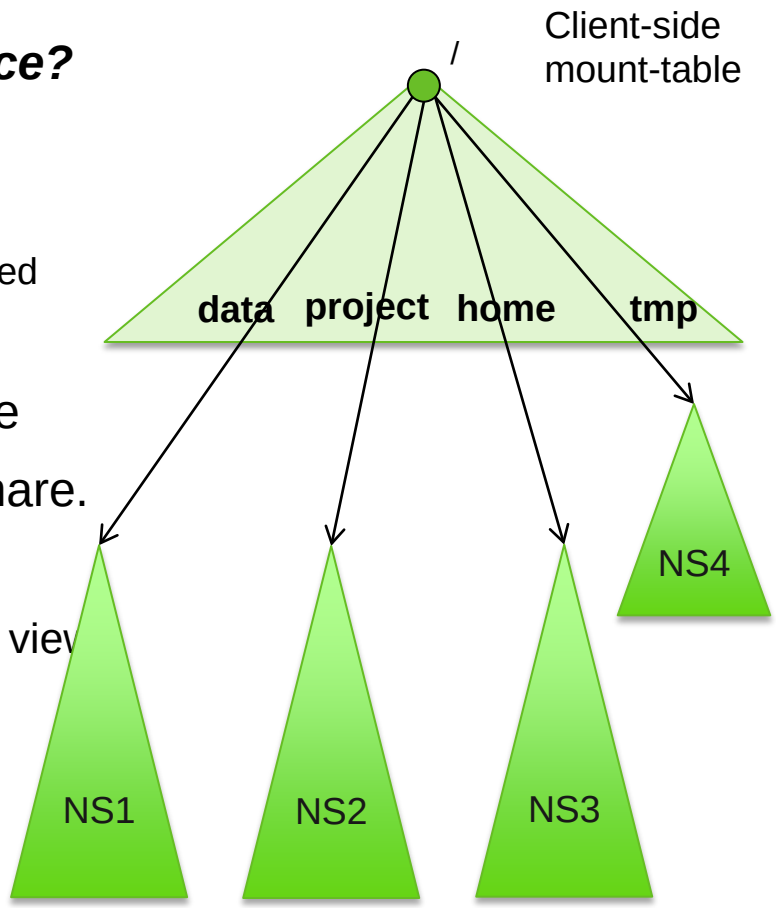
Federation



- Block Storage as generic storage service
  - DNs store blocks in ***Block Pools*** for all the Namespace Volumes
- Multiple independent Namenodes and *Namespace Volumes* in a cluster
  - Scalability by adding more namenodes/namespaces
  - Isolation – separating applications to their own namespaces
  - Client side mount tables/ViewFS for integrated views

Managing Namespaces

- Federation has multiple namespaces
- ***Don't you need a single global namespace?***
  - Some tenants want private namespace
    - Hadoop as service – each tenant its own namespace
  - Global? Key is to share the data and the names used to access the data
- A single global namespace is one way share
- Client-side mount table is another way to share.
  - Shared mount-table => "global" shared view
  - Personalized mount-table => per-application view
    - Share the data that matter by mounting it
- Client-side implementation of mount tables
  - No single point of failure
  - No hotspot for root and top level directories

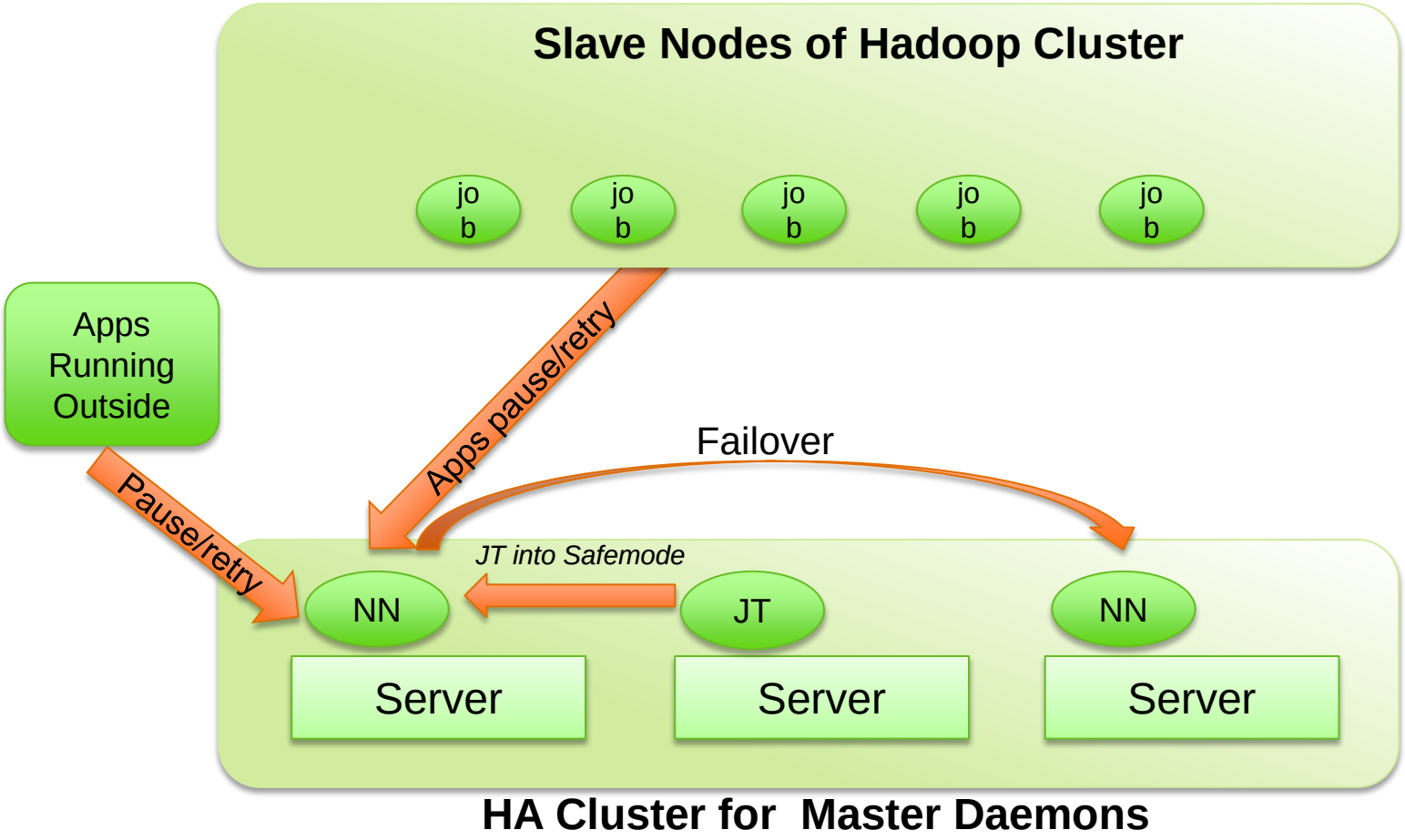


High Availability

HA for HDFS

- **Hadoop 1.x (HDP 1.x)**
  - Failover using industry standard solution (Linux HA, VSphere)
  - Shared storage
  - Failover times 1 minutes to 3-4 minutes for 100 to 300 node cluster
  - Full-stack HA
    - Clients, JT, HBase, HCat automatically pause and retry during failover
    - NN, JT, Hcat all have automatic failover
- **Hadoop 2.x (HDP 2.x)**
  - Failover over using Failover Controller
  - Quorum Journal Manager (No shared storage)
    - Failover times are 30 to 120 seconds less (since Standby NN is hot)
  - Full-stack HA

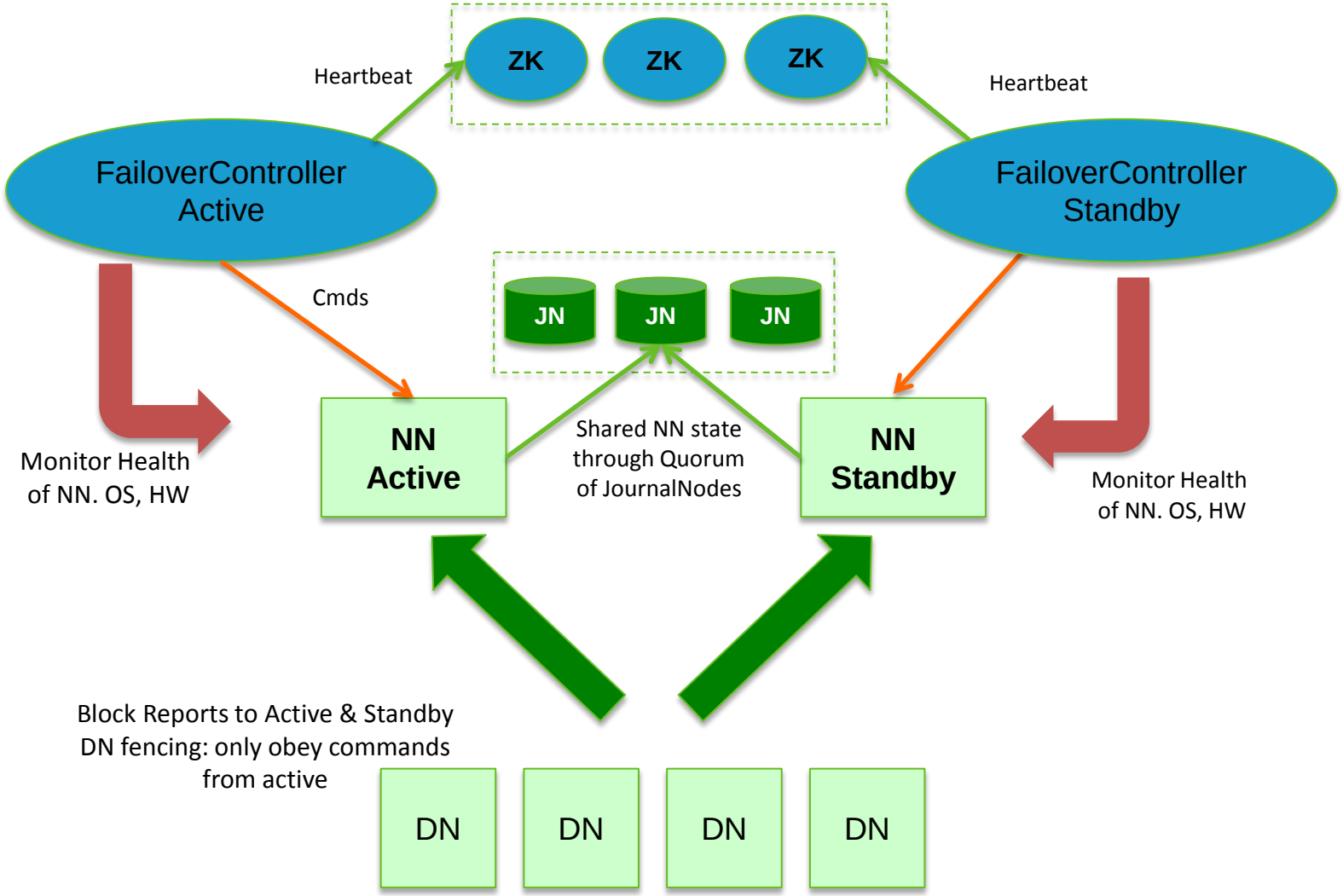
Hadoop Full Stack HA



High Availability – Release 2.0

- **Support for Hot Standby**
- **Supports manual and automatic failover**
- **Automatic failover with Failover Controller**
  - Active NN election and failure detection using ZooKeeper
  - Periodic NN health check
  - Failover on NN failure
- **Removed shared storage dependency**
  - Quorum Journal Manager
    - 3 to 5 Journal Nodes for storing editlog
    - Edit must be written to quorum number of Journal Nodes
- **Replay cache for correctness & transparent failovers**
  - In progress – will be in 2.1.0-beta

Namenode HA in Hadoop 2



Namenode HA has no external dependency

Snapshots (HDFS-2802)

- **Snapshot entire namespace or sub directories**
  - Nested snapshots allowed
  - Managed by Admin
    - Users can take snapshots of directories they own
- **Support for read-only COW snapshots**
  - Design allows read-write snapshots
- **Namenode only operation – no data copy made**
  - Metadata in namenode - no complicated distributed mechanism
  - Datanodes have no knowledge
- **Efficient**
  - Instantaneous creation
  - Memory used is highly optimized
    - State proportional to the changes between the snapshots
  - Does not affect regular HDFS operations

Snapshot – APIs and CLIs

- **All regular commands & APIs can be used with snapshot path**
  - `/<path>/.snapshot/snapshot_name/file.txt`
  - `Copy /<path>/.snapshot/snap1/ImportantFile  /<path>/`

- **CLIs**
  - Allow snapshots
    - `dfsadmin –allowSnapshots <dir>`
    - `dfsadmin –disAllowSnapshots <dir>`
  - Create/delete/rename snapshots
    - `fs –createSnapshot<dir> [snapshot_name]`
    - `fs –deleteSnapshot<dir> <snapshot_name>`
    - `fs –renameSnapshot<dir> <old_name> <new_name>`
  - Tool to print diff between snapshots
  - Admin tool to print all snapshottable directories and snapshots

Performance Improvements

- **Many Improvements**
  - SSE4.2 CRC32C – ~3x less CPU on read path
  - Read path improvements for fewer memory copies
  - Short-circuit read for 2-3x faster random reads
    - Unix domain socket based local reads
      - All applications, not just for special services like HBase
  - I/O improvements using *posix_fadvise()*
  - *libhdfs* improvements for zero copy reads
- **Significant improvements - IO 2.5x to 5x faster**
  - Lot of improvements back ported to release 1.x

NFS Support (HDFS-4750)

- **NFS Gateway provides NFS access to HDFS**
  - File browsing, Data download/upload, Data streaming
  - No client-side library
  - Better alternative to Hadoop + Fuse based solution
    - Better consistency guarantees
- **Supports NFSv3**
- **Stateless Gateway**
  - Simpler design, easy to handle failures
- **Future work**
  - High Availability for NFS Gateway
  - NFSv4 support?

Wire Protocol Improvements

- **Hadoop RPC on the wire encryption uses protobuf**
- **Post 2.1.0 beta stronger compatibility**
  - Java API
  - Wire protocol
  - Both forward and backward compatibility
  - Simplifies migrating to newer releases
- **Rolling upgrades**
  - Enabled by strong wire protocol compatibility
- **Security improvements**
  - Negotiation of authentication
  - Multiplex secure sessions in a connection

Other Features

- **New append pipeline**
- **Improvements for other projects**
  - ***Stale Node*** to improve HBase MTTR
- **Block placement enhancements**
  - Better support for other topologies such as VMs and Cloud
- **Expanding ecosystem, platforms and applicability**
  - Native support for Windows
- **File and Block IDs are unique**
  - Major architectural step
  - Allows caches, layered file systems
  - A key requirement for archives and disaster recovery

Enterprise Readiness

- **Storage fault-tolerance – built into HDFS ☑**
  - Over 7’9s of data reliability
- **High Availability ☑**
- **Standard Interfaces ☑**
  - WebHdfs(REST), Fuse, NFS, HTTPFS, libwebhdfs and libhdfs
- **Wire protocol compatibility ☑**
  - Protocol buffers
- **Rolling upgrades ☑**
- **Snapshots ☑**
- **Disaster Recovery ☑**
  - Distcp for parallel and incremental copies across cluster
  - Apache Ambari and HDP for automated management

Hortonworks

HDFS Futures

From Batch to Real-time

- **New latency sensitive real-time use cases**
  - Interactive queries - Stinger/Tez
  - HBase
- **Heterogeneous (HDFS-2832)**
  - Storage properties were hidden
    - Datanode abstraction from **single storage to collection of storages**
  - Memory, SSDs, Disks as storage types
    - Hierarchical storage tiers
    - Enables memory cache
  - Work in progress
- **Datanode Cache (HDFS-4949)**
  - Enable fast zero copy access to data cached/pinned in datanode memory
  - Co-ordinated cache management
- **Future – more sophisticated caching policies**
  - Cache partial blocks based on access pattern
  - LRU, LRU-2

Storage Abstraction

- **Fundamental storage abstraction improvements**
- **Short Term (Post 2.x GA)**
  - Heterogeneous storage
  - Block level APIs for direct access to fault tolerant storage
    - Apps & Services can by-pass file system interface
  - Granular block placement policies
    - Co-locate related data/blocks in specific nodes
    - Policy/tools to migrate related data together
- **Long Term**
  - Support for objects/key-value store and APIs
  - Serving from Datanodes optimized based on file structure

Higher Scalability

- **Even higher scalability of namespace**
  - Only working set in Namenode memory
  - Namenode as container of namespaces
    - Support large number of namespaces
  - Explore new types of namespaces

- **Further scale the block storage**
  - Block management to Datanodes
  - Block collection/Mega block group abstraction

Next Steps… first class support for volumes

NameServers as
Containers of Namespaces

Storage Layer

- **NameServer - Container for namespaces**
  - Management policies (quota, …)
  - Mount tables for unified namespace
    - Can be managed by a central volume server
  - › Move namespace for balancing
- **WorkingSet of namespace in memory**
  - › Many namespaces in a server
- **Number of NameServers =**
  - › Sum of (Namespace working set)
  - › Sum of (Namespace throughput)

High Availability

- **Further enhancements to HA**
  - Expand Full stack HA to include other dependent services
  - Support multiple standby nodes, including N+K
  - Use standby for reads
  - Simplify management – eliminate special daemons for journals
    - Move Namenode metadata to HDFS

Q & A

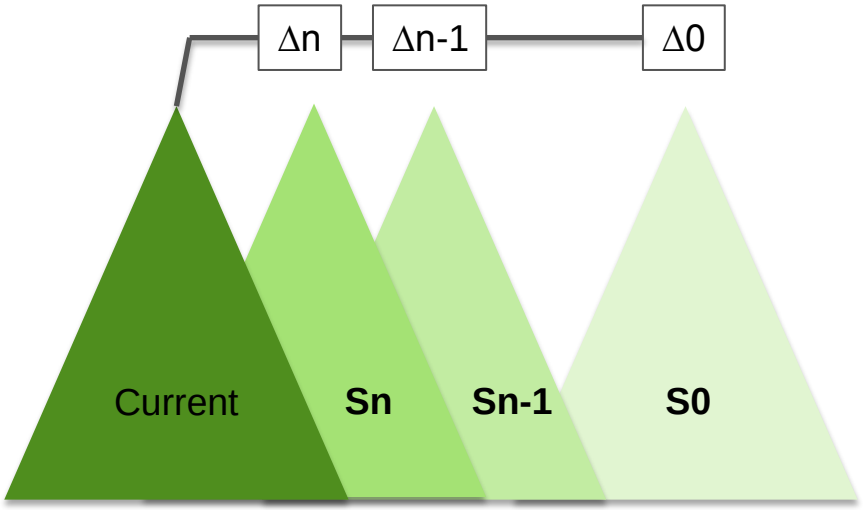
- **Myths and misinformation**
  - ~~Not reliable (was never true)~~
  - ~~Namenode dies, all state is lost (was never true)~~
  - ~~Does not support disaster recovery (distcp in Hadcop0.15)~~
  - ~~Hard to operate for new comers~~
  - Performance improvements (always ongoing)
    - Major improvements in 1.2 and 2.x
  - ~~Namenode is a single point of failure~~
  - ~~Needs shared NFS storage for HA~~
  - ~~Does not have point in time recovery~~

Thank You!

Hortonworks

Backup slides

Snapshot Design



- **Based on** *Persistent Data Structures*
  - Maintains changes in the diff list at the Inodes
    - Tracks creation, deletion, and modification
  - Snapshot state Sn = current - Δn
- **A large number of snapshots supported**
  - State proportional to the changes between the snapshots
  - Supports millions of snapshots