HDFS Router-based Federation

Íñigo Goiri
Microsoft

Chao Sun
Uber

High-level HDFS setup

Microsoft

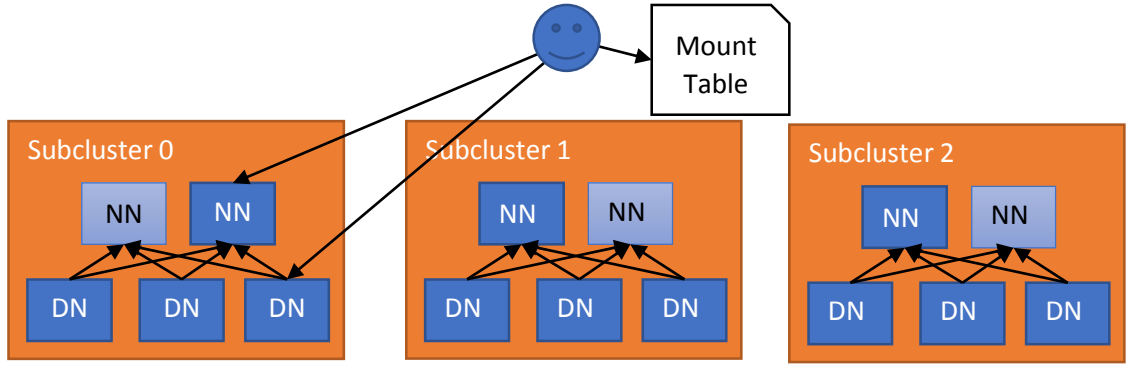
- 3 data centers
  - 2 more provisioning
- Harvested capacity [OSDI'16]
- Hadoop 2.7.1 (migrating to 2.9)
  - Many internal extensions
- Internal batch workloads

Uber

- 3 data centers
  - 4th is coming soon
- Thousands of servers
- 100+ PB of data
- Hadoop 2.8.2
  - Many custom and backported patches
- Serves applications
  - Fraud detection, ML/DL, ETA calculation…
- **Foundation for Uber's data lake**

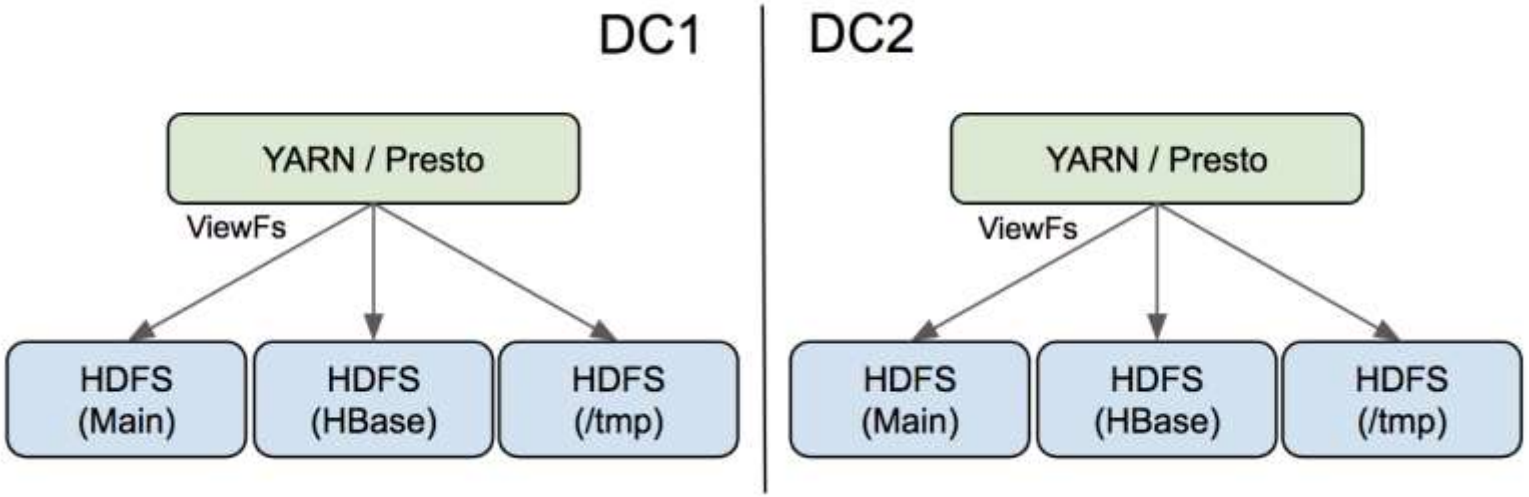
Current architecture in HDFS

- NameNode scalability/performance issues
- Split into independent namespaces (subclusters)
  - Fragmentation
  - Users in charge of choosing subcluster
- ViewFS to unify subclusters



Setup at Uber

- Use client-side ViewFS config
- Split into 3 clusters (each DC)
  - Main production HDFS cluster
  - Hbase cluster
  - Tmp cluster (Hive scratch directory, Yarn application logs, etc).



Problems with ViewFS

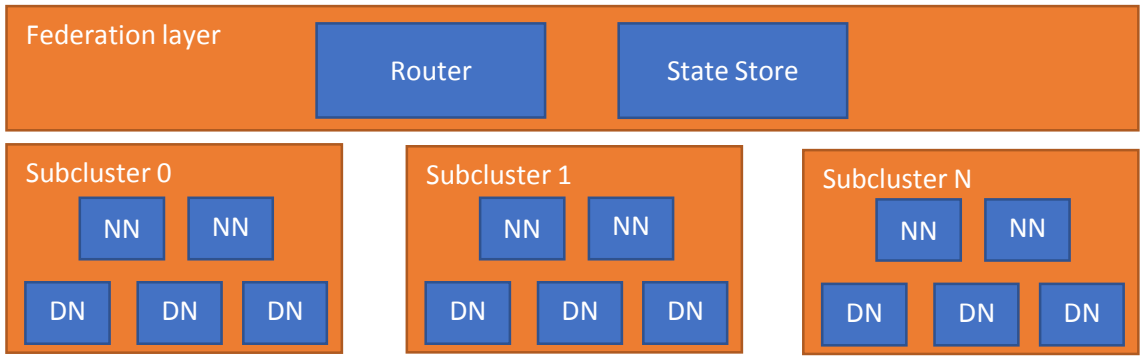
- Hard to maintain per-client mount table
  - Many HDFS clients, libraries, configurations,…
  - Each client may have its own mount table
  - Requires infrastructure to distribute mount table updates
  - Adding more datacenters
- Manual balancing of subclusters

- Solution
  - “Centralize” the mount table
  - Introduce a routing layer: Router-Based Federation (RBF)

Our solution: RBF

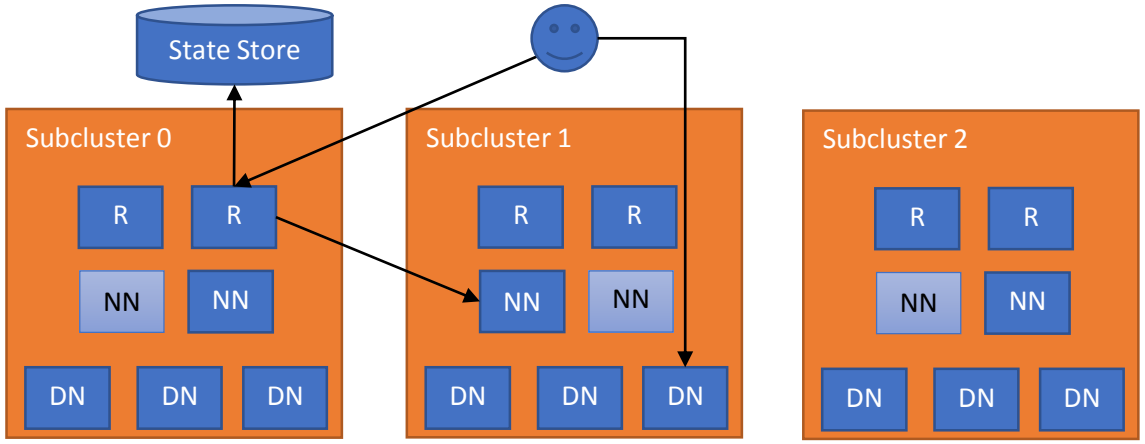
- “Centralize” the mount table

- Introduce a routing layer: Router-Based Federation (RBF)
  - Router: Proxy requests from users to the right NameNode
  - State Store: Shared mount table and other state



Router

- Unified view of the federation
- Performance improvements
  - Caching State Store data
  - Includes modified client to contact NameNodes



State Store

- Shared information
  - Mount table: /DC0-2 → DC0-2, /
  - Federation membership: DC0, DC0-1, DC0-2, DC0-3
  - Router tracking: R1, R2, R11, R22,…
- Not critical for performance (cached in each Router)
- Implementations
  - ZooKeeper, HDFS, SQL server,…

HDFS RBF deployments

Microsoft

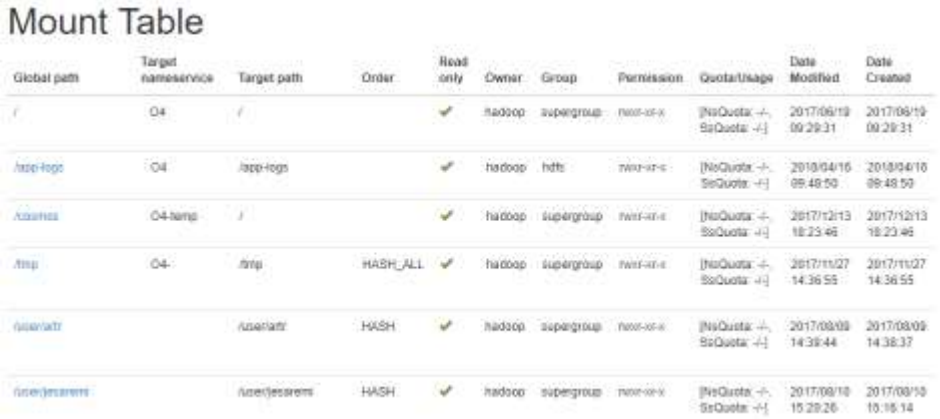
- 1 Router per Namenode
- All data centers
- ZooKeeper as the State Store
- Main workloads
  - Large deep learning
  - Application logs

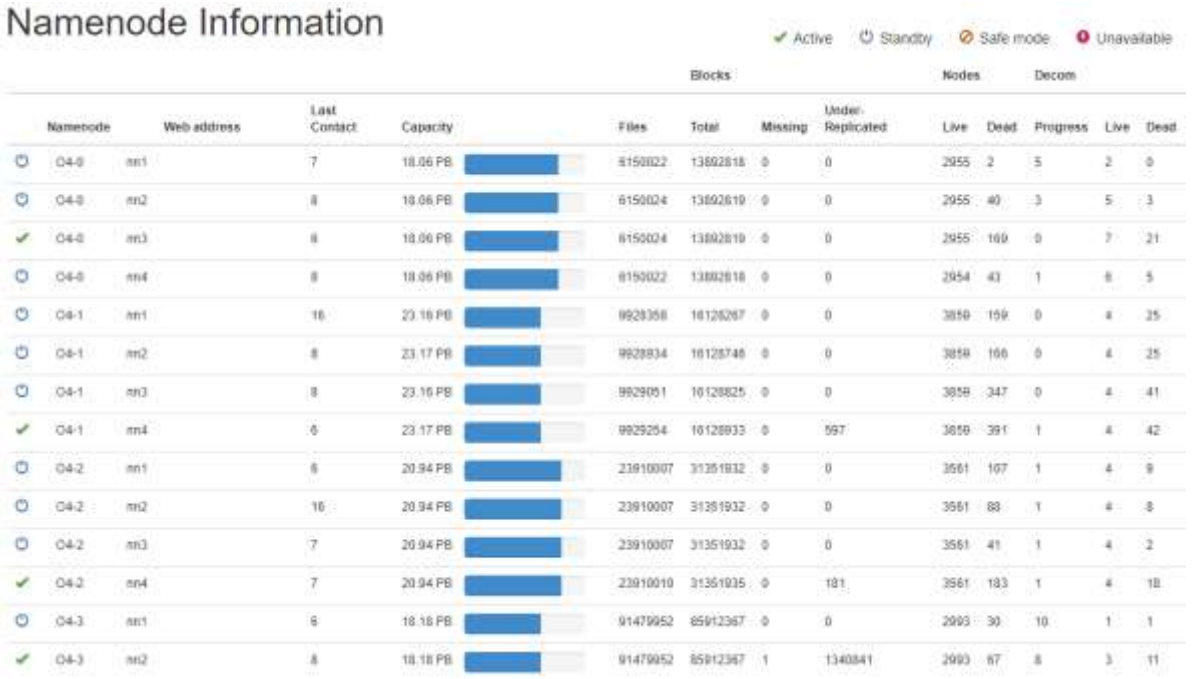
Uber

- 2 routers
- 1 data center
- ZooKeeper as the State Store
- Workload
  - Hive traffic accessing scratch dir

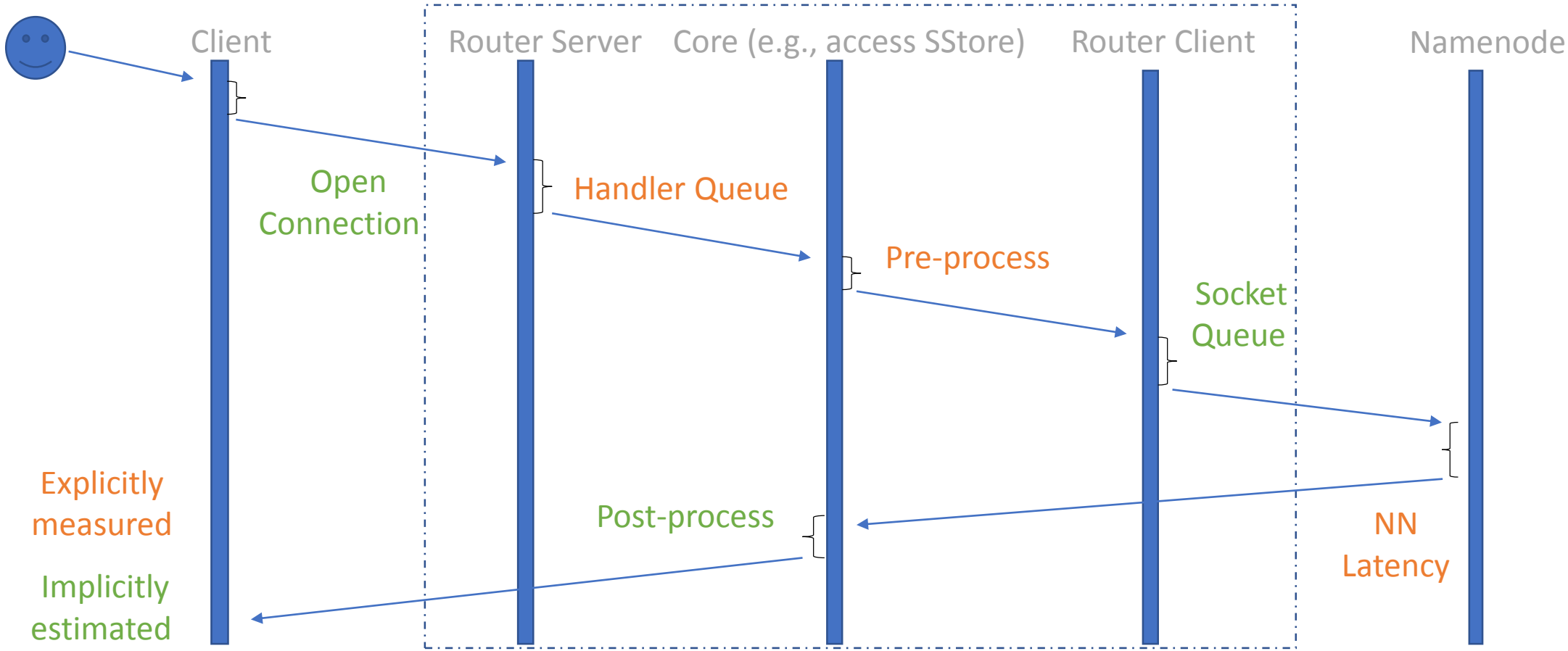
Microsoft RBF deployment

- 23k servers
  - 8 subclusters
  - 28 NameNodes (4 per subcluster)
  - 28 Routers: load balancing
- Transparent access
  - RPC
  - WebHDFS
  - Web UI



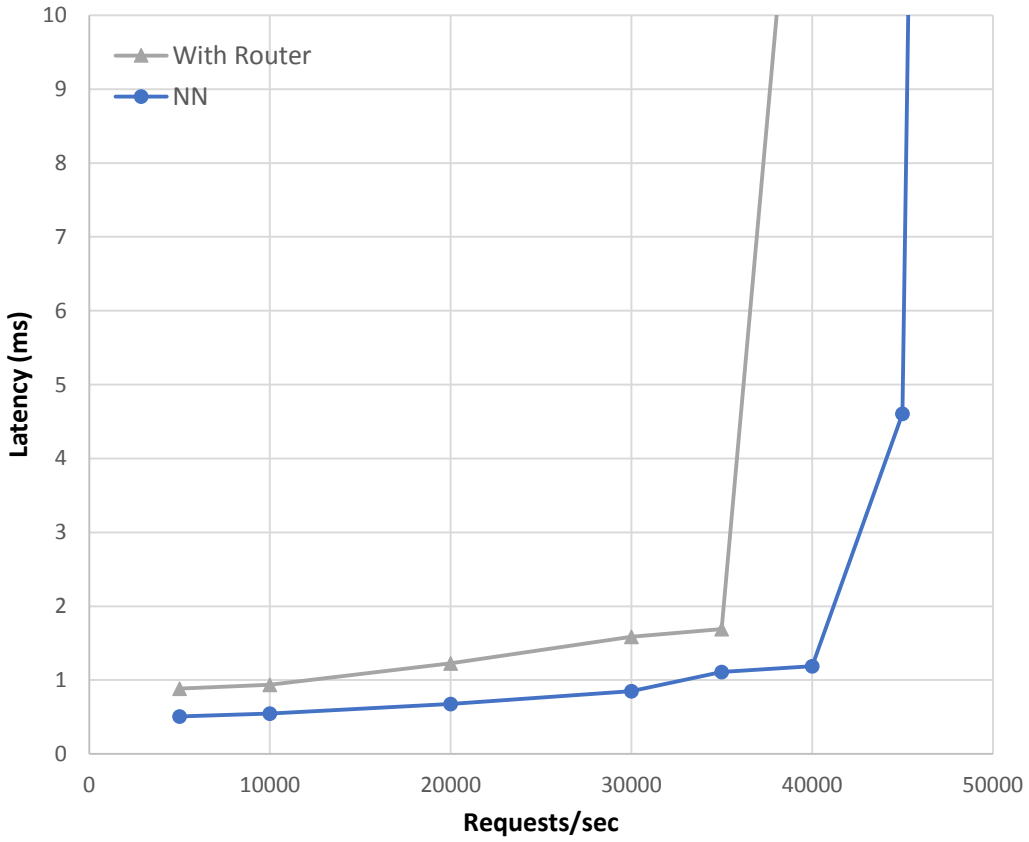


Router adds latency

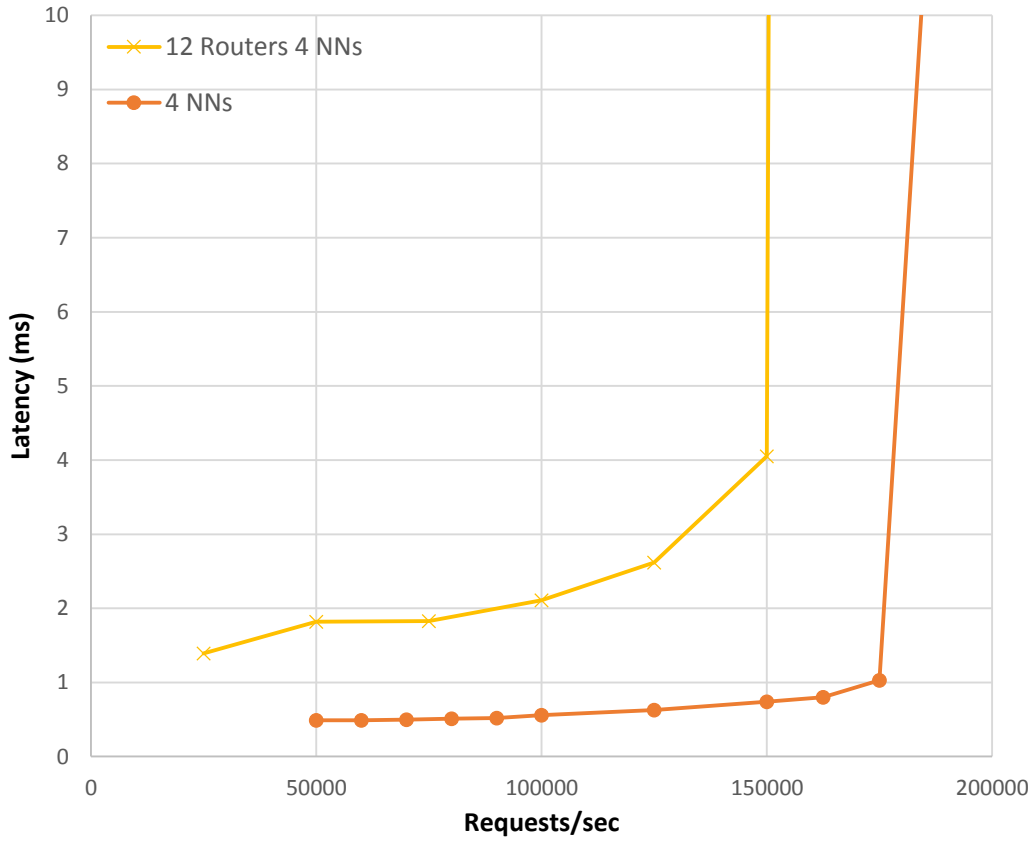


Performance and scalability

NN vs R→NN



4NN vs 12R → 4NN



Read metadata (cheapest access → worst case for Router)

Tuning Router RPC client

- Router uses a connection pool for each <user, namenode> pair
  - Connection creation/cleanup is done asynchronously
  - Hard to configure: Connection pool size, creator queue size,…
- Default IPC client is a bottleneck
  - Router forwarding many requests to NameNodes
  - HADOOP-13144 allow multiple concurrent client/connections
  - HDFS-13274 leverage concurrent connections

Active development

- Many contributors
  - VipShop, Uber, Huawei,…
- Bug fixes
- New features:
  - WebHDFS
  - Better subcluster isolation
  - Federated quotas
  - Tracking the state of the Routers
  - Spreading mount points across subclusters

Spreading mount points across subclusters

- Large jobs processing data spanning multiple subclusters
- Similar to merge mount points (HADOOP-8298)
- Policies to decide the subcluster to write a file
  - Consistent hashing, available space, local subcluster,…
- Limitations
  - Needs to find files in multiple subclusters
  - Renaming has some inefficiencies
  - Subcluster unavailability is hard to manage

Open issues

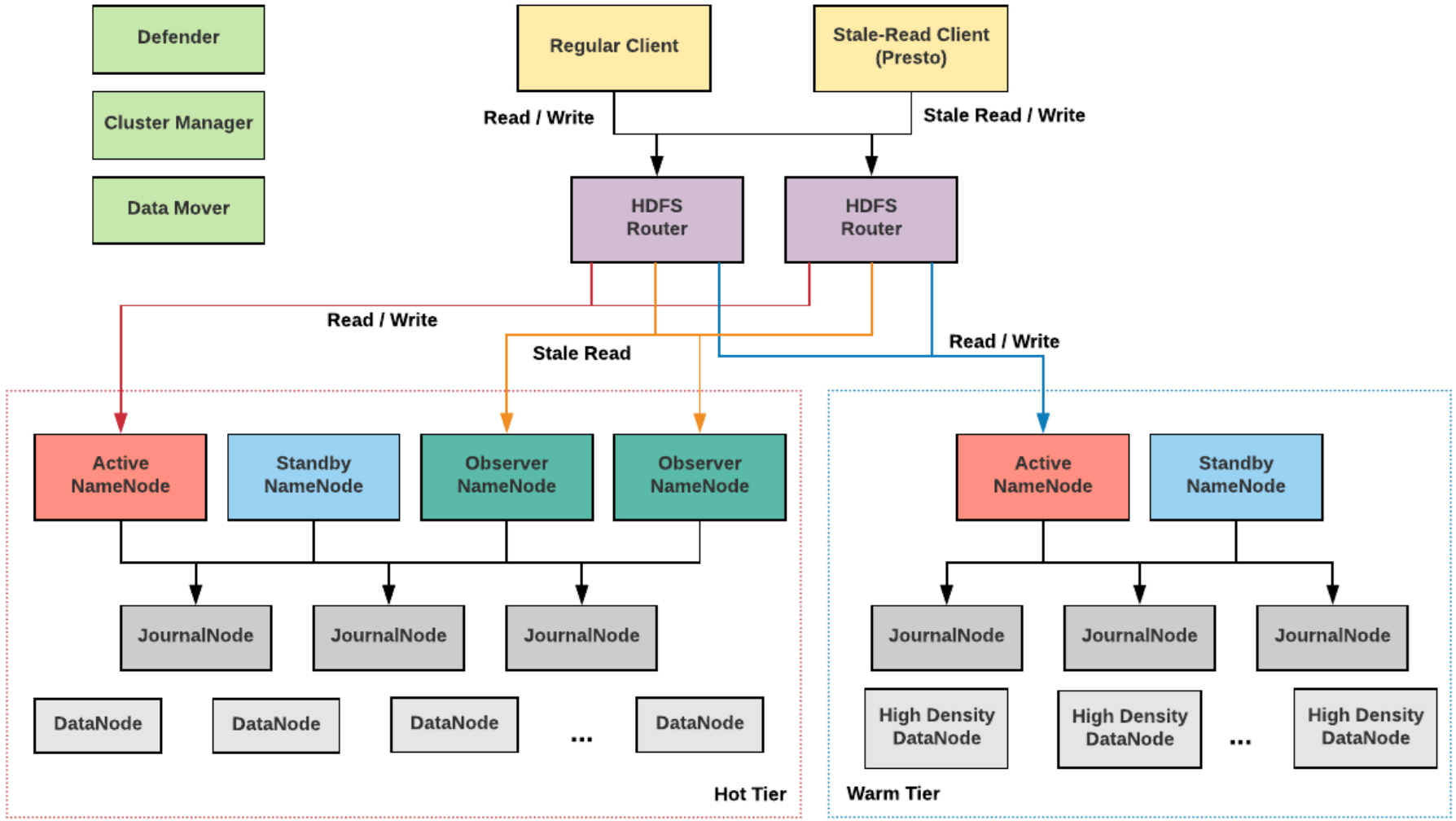
- Router → Namenode connections (HDFS-13274)
- Security (HDFS-13532)
  - Manage delegation tokens
  - LinkedIn driving this effort
- Rebalancer (HDFS-13123)
- inodes (HDFS-13469)
  - We need to identify the nameservice
- Other issues tracked in HDFS-12615

Rebalancer

- Subclusters can get unbalanced
  - Load, space,…
- Manually move files/folders between subclusters
  - Hard to provide strong consistency

- Transparent data movement between subclusters
  - Service to monitor if subclusters balanced
  - Policies to decide what to move and trigger rebalancing [ATC'17]
- Rebalancer
  - Move data: DistCp, Tiered storage, DN block linking,…
  - Change namespace (mount table) consistently

Uber future plan: overview



Uber future plan: detailed

- Routers on top of all HDFS clusters
- Service to initiate data movement between hot and warm clusters
  - Similar to Rebalancer, implemented in Golang
- Service to track dataset temperature and initiate data movement
- Observer (ReadOnly) NameNodes to serve stale read requests
  - HDFS-12943
  - > 90% traffic are read RPC requests
  - Mostly for Presto queries

New ideas/brainstorming

- Federating federations
  1. Creating a hierarchy: putting Routers in front of other Routers
  2. Large federation: sharing State Store across deployments
- Connecting DataNodes to RBF
  - DNs need to be configured to join subclusters
  - Can DNs use the Routers to join subclusters?
    1. Heartbeat through the Router
    2. Find the Datanodes through the Router