Namenode High Availability

Uma Maheswara Rao G

www.huawei.com

maheswara@huawei.com

umamahesh@apache.org

HUAWEI TECHNOLOGIES CO., LTD.

Who am I

- R&D Tech Lead in Huawei
- Apache Hadoop Committer at ASF
- Active Contributor to Apache BookKeeper at ASF
- Main development focus on HDFS

Huawei Hadoop R&D – In a Glance

Hadoop Development

- Secondary Index in HBase
- HDFS NN HA (Hadoop-2)
- Bookkeeper as shared storage for NN HA (Hadoop-2)
- HDFS NN HA (Hadoop-1)
- MapReduce ResourceManager HA (Hadoop-2 / YARN)
- MapReduce JobTracker HA (Hadoop-1)
- Hive HA

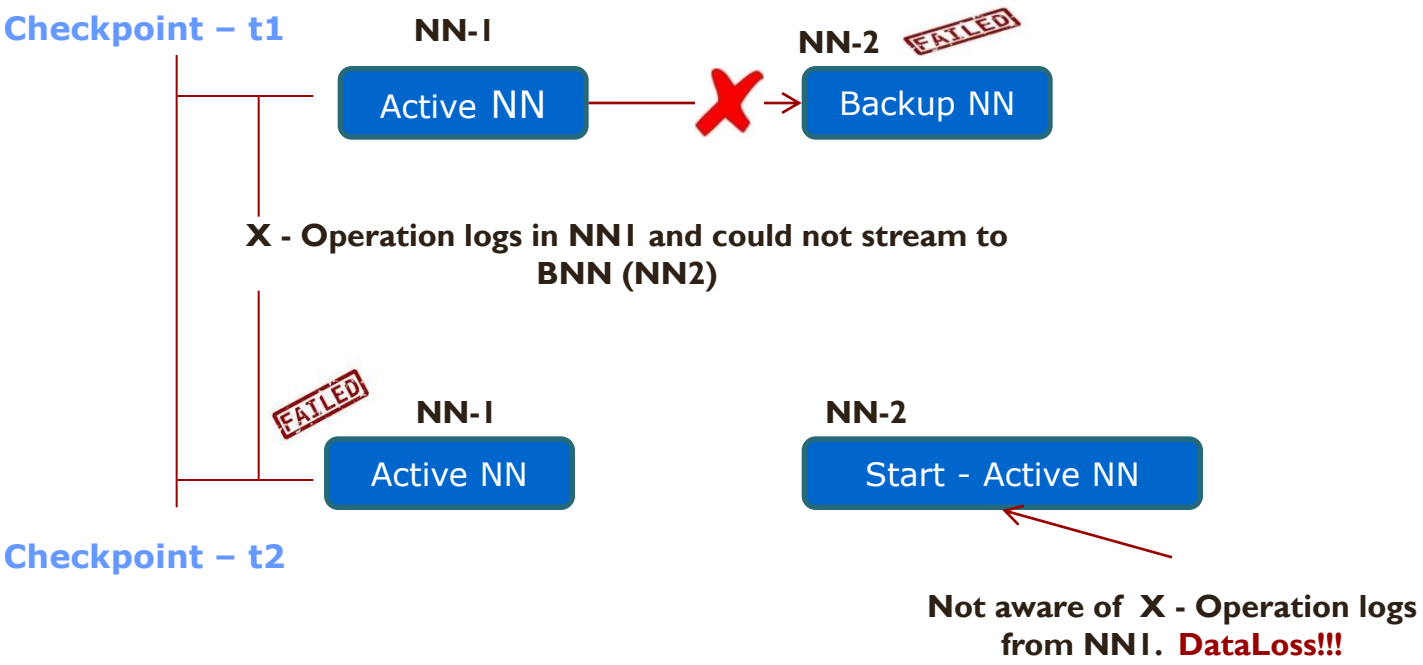
Stabilization

- Raised over 650 defects since Jan’11
- Fixed over 500 defects since Jan’11, and contributed back to community.

Namenode High Availability

- **In 2011, we implemented HA in Hadoop 0.20.1, based on Backup Namenode(BNN) and ZooKeeper at Huawei**
  - Intelligent clients - find active NN from configured NNs
  - Streaming edits to the BNN
  - Sending block reports to both active NN and BNN
  - BNN does periodic checkpoints
  - ZK based leader election
- **Achieved hot standby and Automatic Switch**
- **However, BNN based solution did not address double failure scenario**

What is Double Failure here?



- NN1 is active and NN2 is Backup node
- NN1 failed to stream edits to the Backup node and removed the stream
- NN1 received X- operation edit logs, where NN2 not aware of them
- Now NN1 crashed, NN2 becomes active
- NN2 continues to work but not aware of the edits after checkpoint:t1

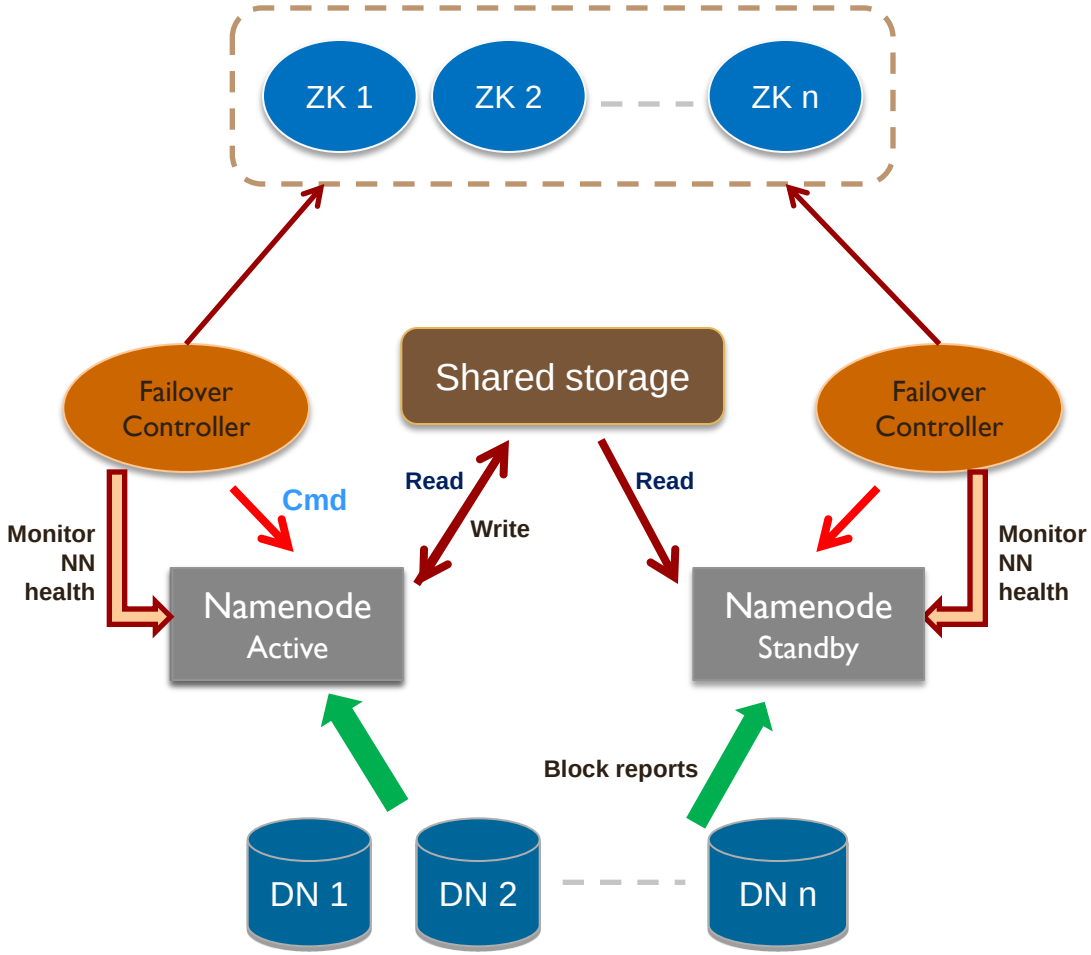
Community discussions on double failure

- **In 2011 - 2012, Community started discussion on Shared storage approach**
  - HDFS-1623 : Discussed double failure issue
  - Store the edit logs at a common place and share to both the Namenodes
- **Huawei collaborated with community in HDFS-1623 development**

Constraints with the HDFS-1623:

- The shared storage is an SPOF. A highly available shared storage is required.

NameNode HA Architecture



Both NameNodes start as standby and try loading the edits

New process **FailOver Controller (ZKFC)** responsible for monitoring and failover

On ZK based **leader election** ZKFC issue command to its local NN to become **Active** or **Standby**

Only **Active NN** will write to shared storage, **Standby NN** will read from it.

All DataNodes will send **block reports** to both Namenodes to achieve **hot Standby.**

Shared storage options

- NFS: May not be a better fit for many deployments as NFS is an external device, costly, less control on timeouts etc.
- QuorumJournalManager(QJM): Newly implemented in HDFS. It did not come out from any of the Apache releases yet.
- BoookKeeperJournalManager(BKJM): Integrated BookKeeper based shared storage to HDFS. Available in Hadoop-2 onwards.

Motivations to use BookKeeper as shared storage

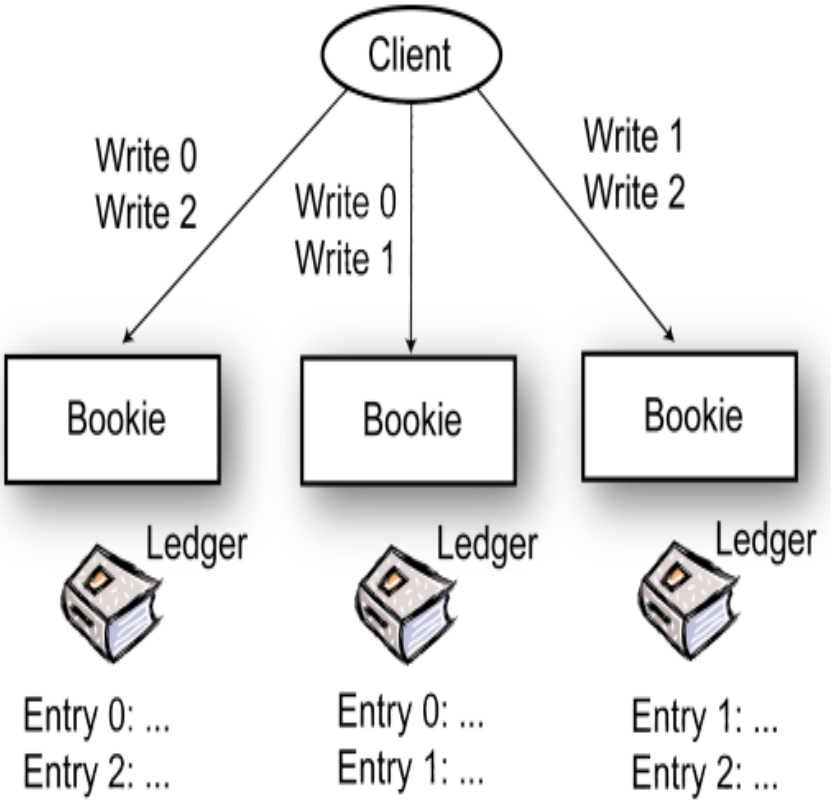
- BookKeeper is mainly inspired by Namenode. Can work as shared storage with HA Namenode
- Can Support Federated Namenodes
- Dynamic scaling up/down of storage nodes(BK)
  - Useful when HBase, Federated NN also uses it.
- Multiple disk support - Reduces the disk seeks
- Round Robin reads from quorum - distributes the load
- HDFS-234 - BookKeeper based Journal manager
  - Available in Hadoop-2 releases and trunk
- Quorum based commits in trunk
  - Parallel writes to write-quorum Bookies and wait for ack-quorum (see BOOKKEEPER-208)

Contd..

Motivations to use BookKeeper as shared storage

- BookKeeper is in production use at Yahoo! for guaranteed delivery of log messages to Hedwig Servers
- Code was there for more than a year in Apache and has active contributors and committers list

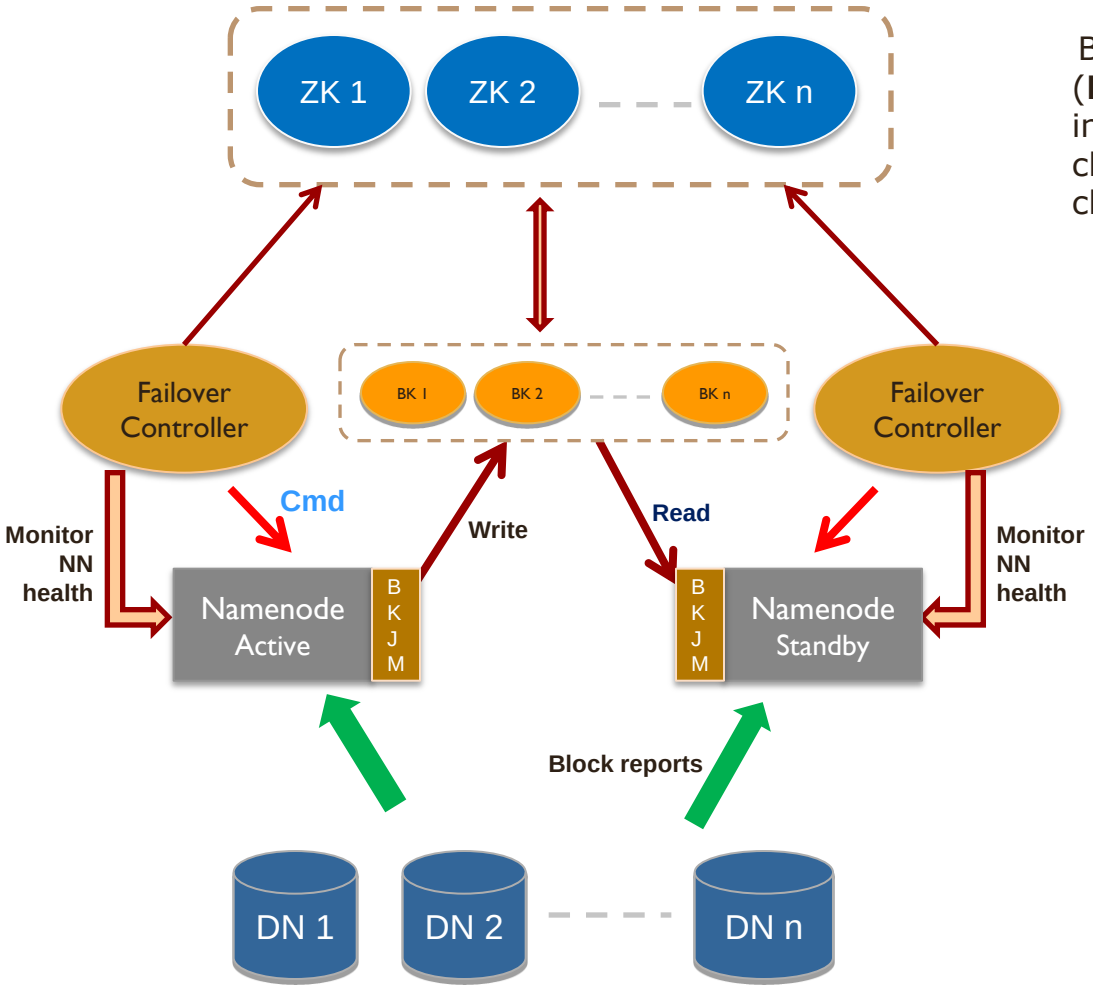
BookKeeper at glance



- Bookie: Storage node
- Ledger: log file
- Ensemble: group of bookies storing a ledger
- Writes to quorums of Bookies
- Parallel writes to quorums
- Wait for ack quorum (in BK 4.2)
- Reads from the same quorum

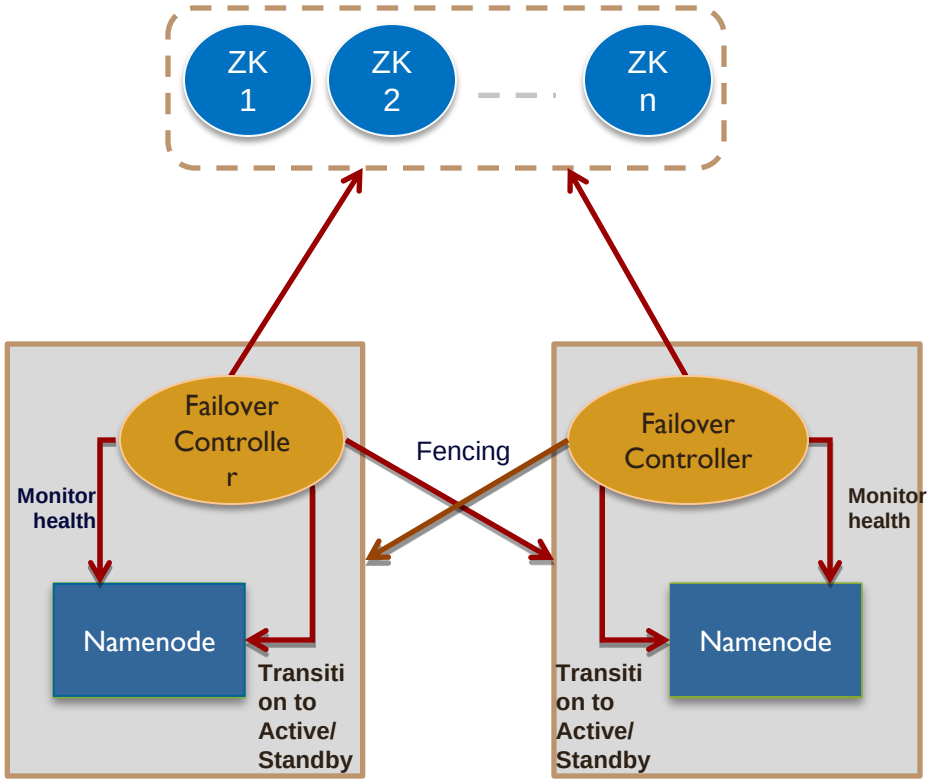
Throughput: max latency of ack quorum bookies response

NN HA Architecture with BK as shared storage



BookKeeper Journal Manager (**BKJM)** is NameNode plugin implementation, involves BK client to read/write to/from BK cluster

ZooKeeper Failover Controller (ZKFC)



- Depends on ZooKeeper for leader election
- Monitors the health of Namenodes
- Invoke the RPC calls to NN to switch the state when leader election happens
- Perform leader election when health monitor find NN is bad
- **Fencing**: ZKFC will kill the remote NN before invoking switch commands to ensure single Active NN at any time.

Special fencing scenarios:

What if remote node is not accessible to kill?

- Requires to write a fence method with shared storage.

  Note: Shared storage should support fencing for this purpose

How BK based shared storage handles fencing?

- BookKeeper allows single writer for a ledger

  "Eliminates the need of strict ZKFC level fencing"

- BKJM solves the issue of concurrent ledger creations by two Active Namenodes - see HDFS-3452

Configurations

- **Namenode side:**

```
<property>
    <name>dfs.namenode.shared.edits.dir</name>
    <value>bookkeeper://zk1:2181;zk2:2181;zk3:2181/haedit</value>
</property>

<property>
    <name>dfs.namenode.edits.journal-plugin.bookkeeper</name>
    <value>org.apache.hadoop.contrib.bkjournal.BookKeeperJournalManager</value>
</property>
```

- **BookKeeper side:**

  zkServers = **zk1:2181,zk2:2181,zk3:2181**

  Bookies can start with remaining default configurations

Other Enhancements/Work In Progress

- BOOKKEEPER-237 - Automatic recovery on Bookie failures.

  Importance – To avoid the data-loss when multiple Bookie crashes

- Entry read latency improvement with (round-robin + speculative) reads with less timeout - Avoid latencies in switch

- Already developed support for shared storage upgrades

- Further improvements and work progress is in HDFS-3399

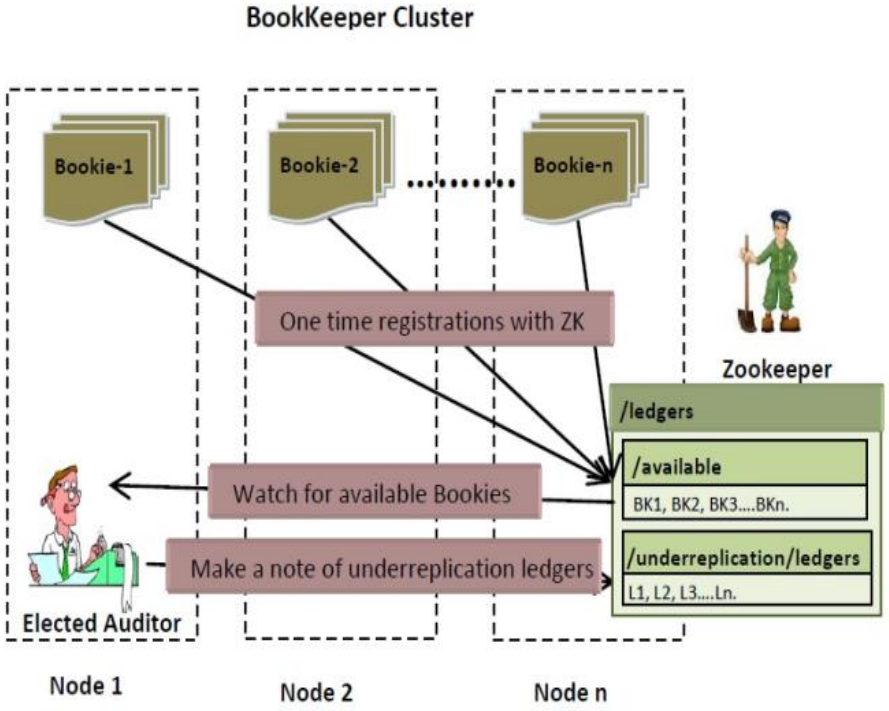
- Take a look at BookKeeper community work

  https://issues.apache.org/jira/browse/BOOKKEEPER

BookKeeper Auto Recovery (BOOKKEEPER-237)

- Autorecovery is a new module in BookKeeper
- Re-replicates the ledgers automatically when there are Bookie failures

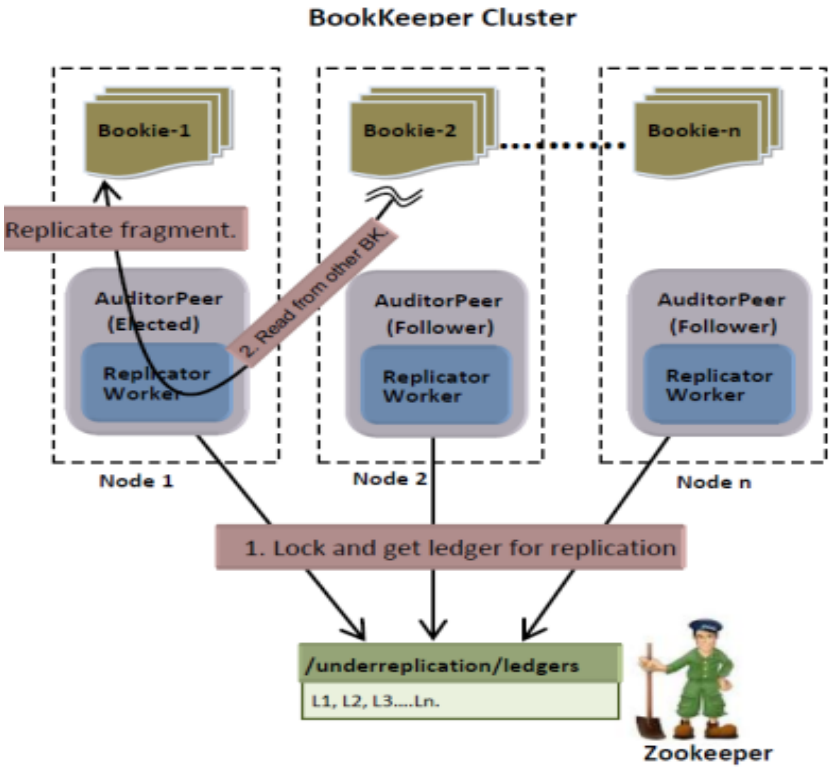
Auditor:



- Watch on Available Bookies
- Detects the failed Bookie ledgers
- Add that ledgers ids in ZK

BookKeeper Auto Recovery (BOOKKEEPER-237)

ReplicationWorker:



- Monitor for the ledger ids published by Auditor
- Scan and find the missed fragments from the ledger
- Initiates the re-replication
- Clean up the re-replicated ledger ids from ZK

Testing

- Tested using HA cluster with HBase, MR, Hive, Huawei OM
- 500+ automated integration HA tests running in daily CI
- Verified the fault tolerant tests with BK recovery feature – see the list in HDFS-3399
- Periodic kill node scenarios to verify the switch with BK – Did not see any data-loss
- In our test cluster, we did not see significant degradations and expected linear degradations when we increase quorum

Any queries?

Thank you

www.huawei.com