INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Snapshotting in HDFS for HOPS

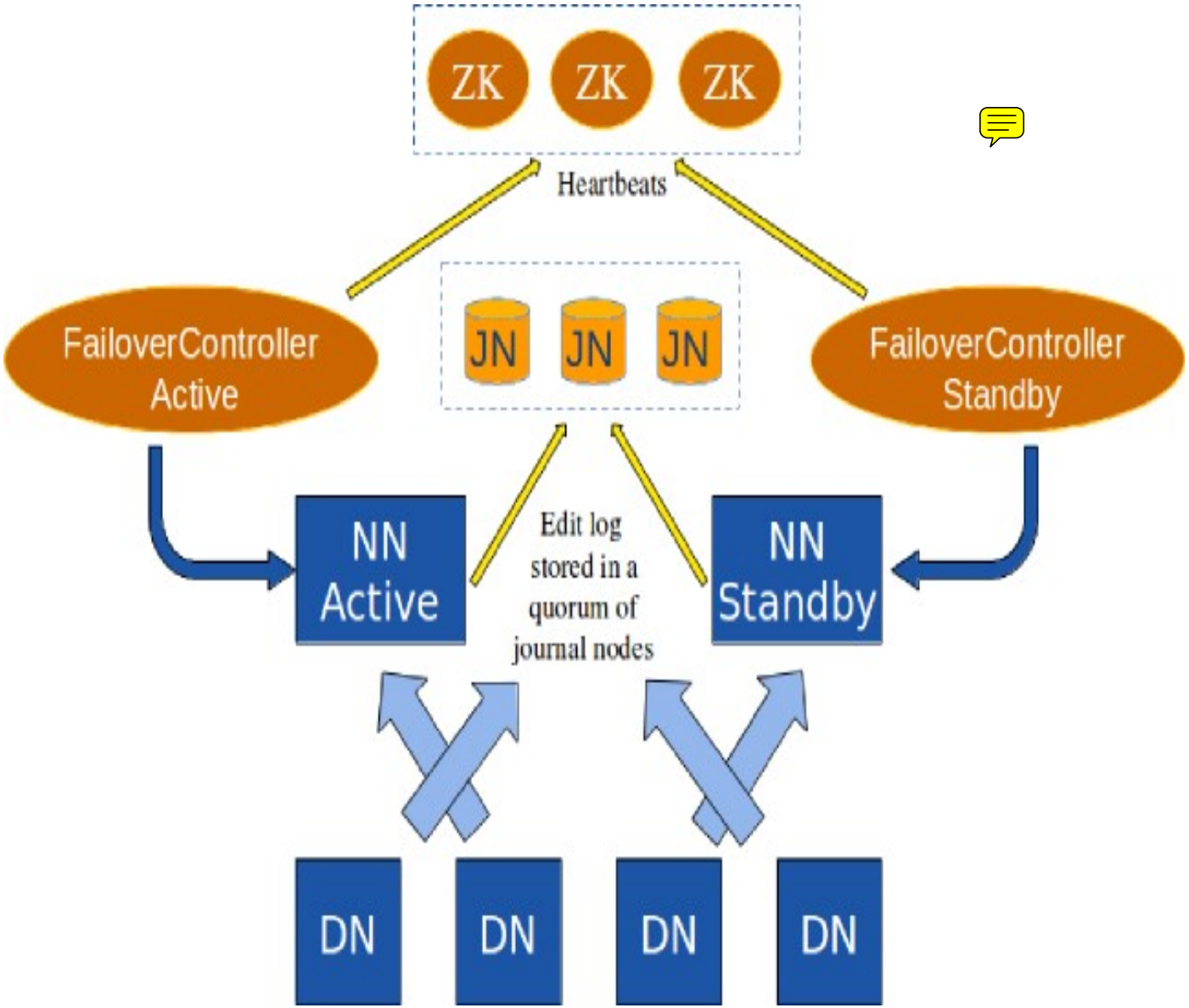
MSc Thesis – Pushparaj

Supervisor: Prof. Luís Manuel Antunes Veiga

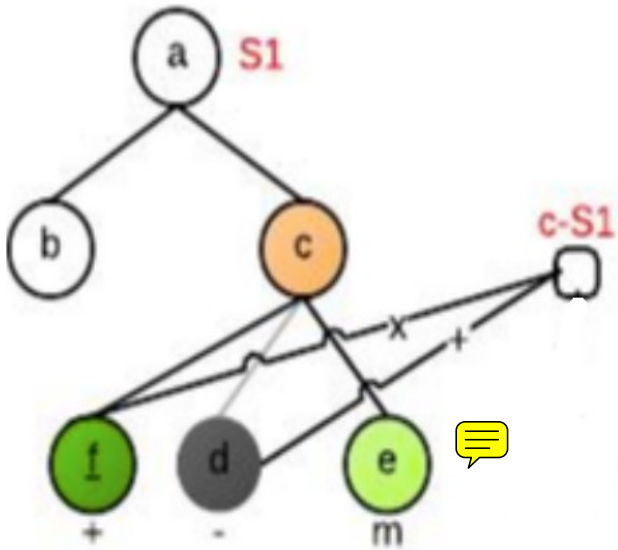
Motivation

- Software Upgrades
- Rollback from Errors

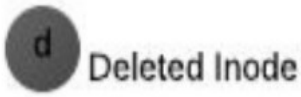
HDFS Architecture



Snapshots in HDFS



Snapshot with diff



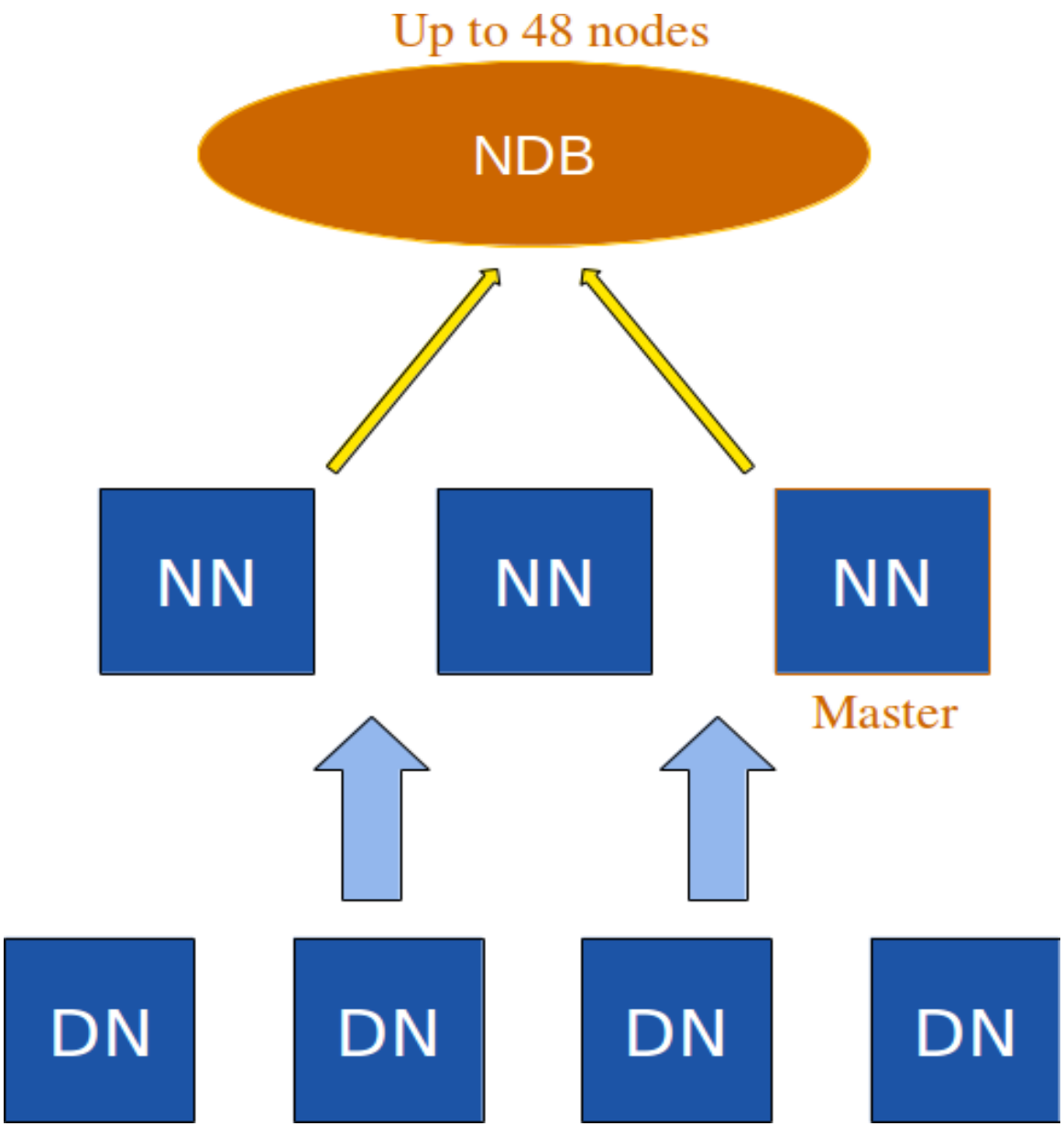
Deleted Inode

Added Inode

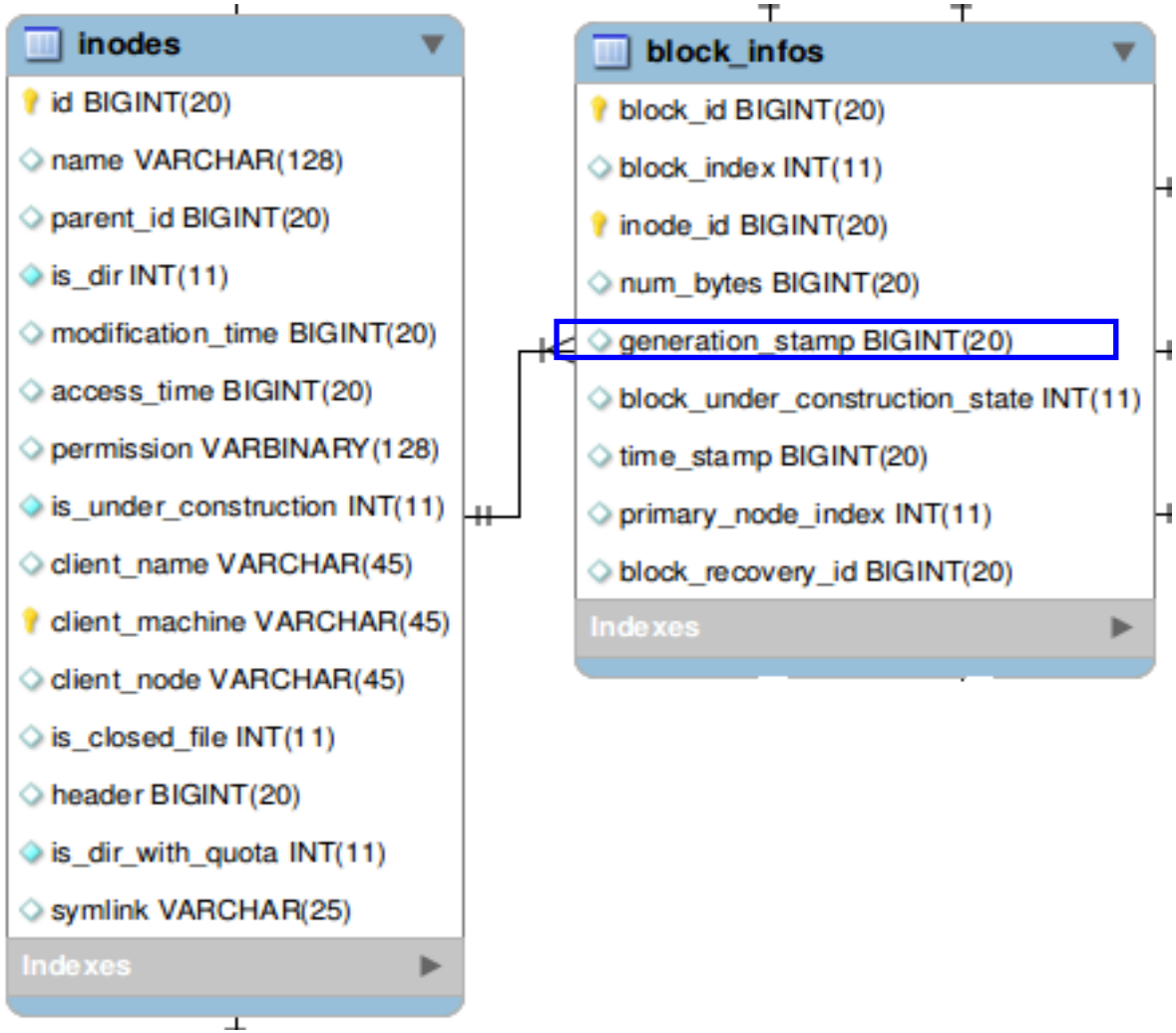
Modified Inode

InodeDirectoryWithSnapshot

HOP-HDFS



HOP-HDFS


inodes
id BIGINT(20)
name VARCHAR(128)
parent_id BIGINT(20)
is_dir INT(11)
modification_time BIGINT(20)
access_time BIGINT(20)
permission VARBINARY(128)
is_under_construction INT(11)
client_name VARCHAR(45)
client_machine VARCHAR(45)
client_node VARCHAR(45)
is_closed_file INT(11)
header BIGINT(20)
is_dir_with_quota INT(11)
symlink VARCHAR(25)
Indexes
block_infos
block_id BIGINT(20)
block_index INT(11)
inode_id BIGINT(20)
num_bytes BIGINT(20)
generation_stamp BIGINT(20)
block_under_construction_state INT(11)
time_stamp BIGINT(20)
primary_node_index INT(11)
block_recovery_id BIGINT(20)
Indexes

Design Goals

- Time to take Snapshot—O(1)
- Scale for Tera-bytes of metadata
- Efficient Rollback
- Low Space Overhead on Snapshot metadata

Solutions

- Read-Only(RO) Snapshots
- RO Root Level Single Snapshot
  - Specific for Rollback on Software Upgrades
- RO Nested Snapshots
  - General Purpose file/directory level multiple/nested Snapshots.

Read-Only Root Level Single Snapshot(ROSS)

- **isDeleted**

0 ==> inode is not deleted.

1 ==> inode deleted after Root Level snapshot was taken.

- **status**

0 ==> inode was created before taking Snapshot.

2 ==> inode created before taking snapshot but modified after that.

3 ==> inode was created after taking snapshot.

Deletion of Directories

- atomic{

  Set isDeleted=1 for this directory

  }
- Process the children in depth-first manner.
- If file status=3,
  - atomic{

    permanently delete blocks.

    delete the inode row.

    }
- Atomic construct to guarantee consistency of metadata in case of NameNode failure.

RollBack

For INodes:

- Delete from inodes where status=2 or status=3
- Update inodes set isDeleted=0 where id>0 and isDeleted=1
- Update inodes set id = -id, parent id = -parent id where id<0

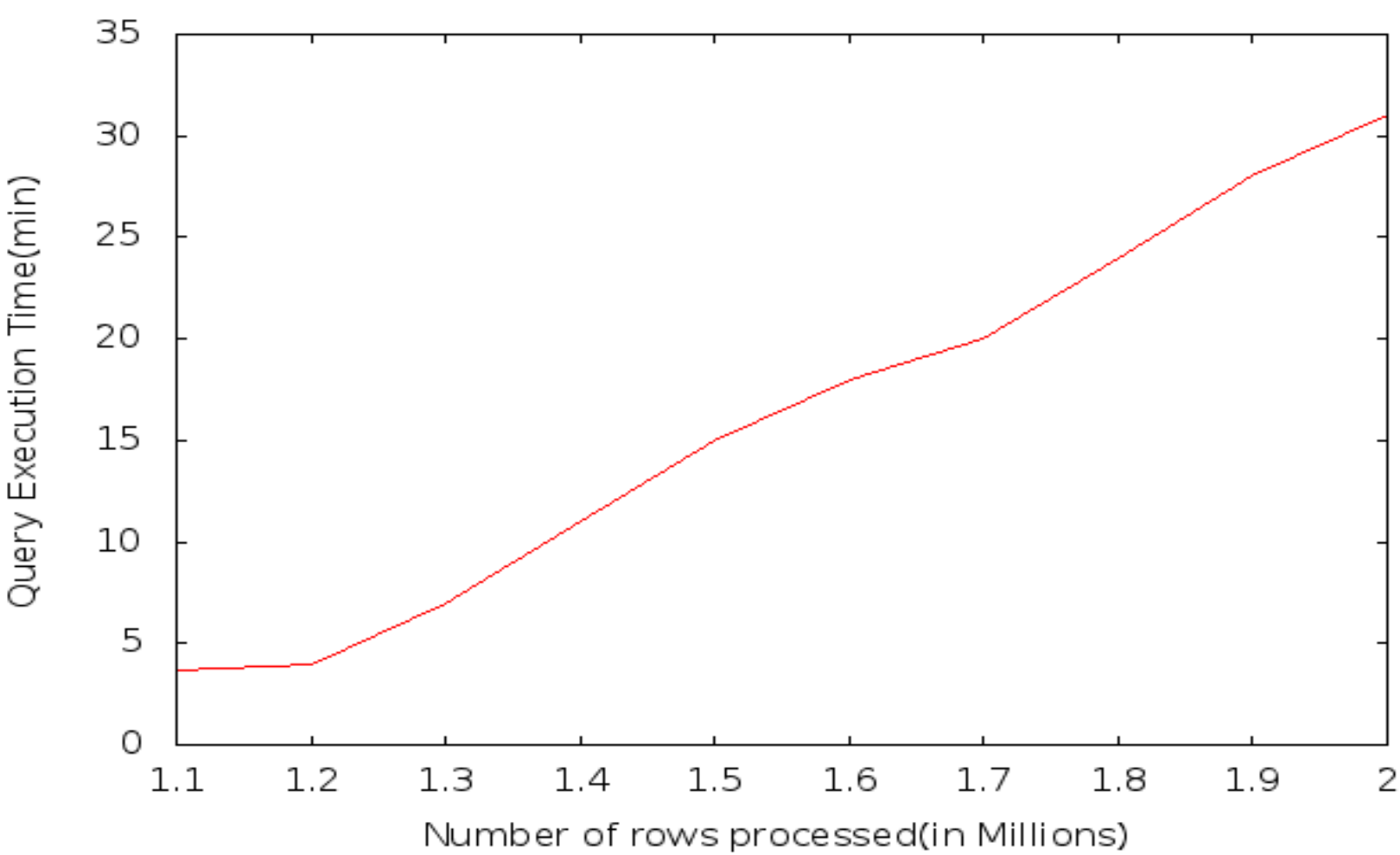
For Blocks:

- Delete from Block Info where status=2 or status=3
- Update Block Info set block id = -block id, inode id = -inode id where id<0
- Delete from Block Info where block id<0

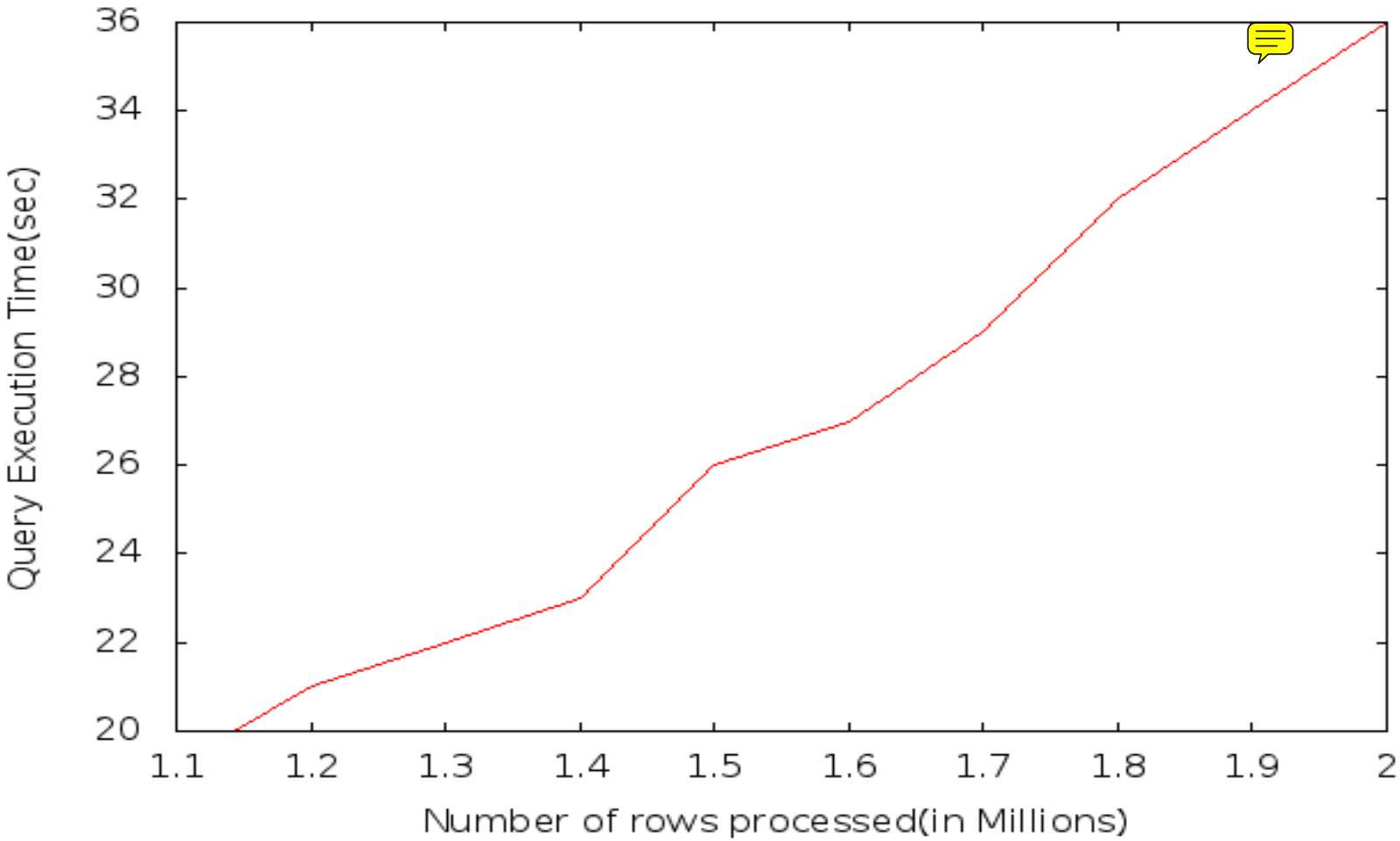
Rollback -Implementation

- Take subTreeOpLock on root
- Take read-lock on all rows
- Task1
  - Delete inodes with status=2 or Status=3
- Task2
  - Set isDeleted=0 for inodes isDeleted=1 and id>0
- Task3
  - Update inodes set id=-id, parent_id=-parent_id where id<0;
  - Deleted inodes where id<0;

Rollback on MySql Server



Rollback With ClusterJ



Read-Only Nested Snapshots

- SNAPS
  -

inode_Id	Snapshot_Id	User	Time

- C-List
  -

inode_Id	Time	Created_Inode_Id

- D-List
  -

inode_Id	Time	Deleted_Inode_Id

- M-List
  -

inode_Id	Time	Modified_Inode_Id	Original_Row

- MV-List
  -

inode_Id	Time	Modified_Inode_Id	Original_Row

- MV-In-List
  -

inode_Id	Time	Moved_In_Inode Id

RO-Nested Snapshots(RONS)

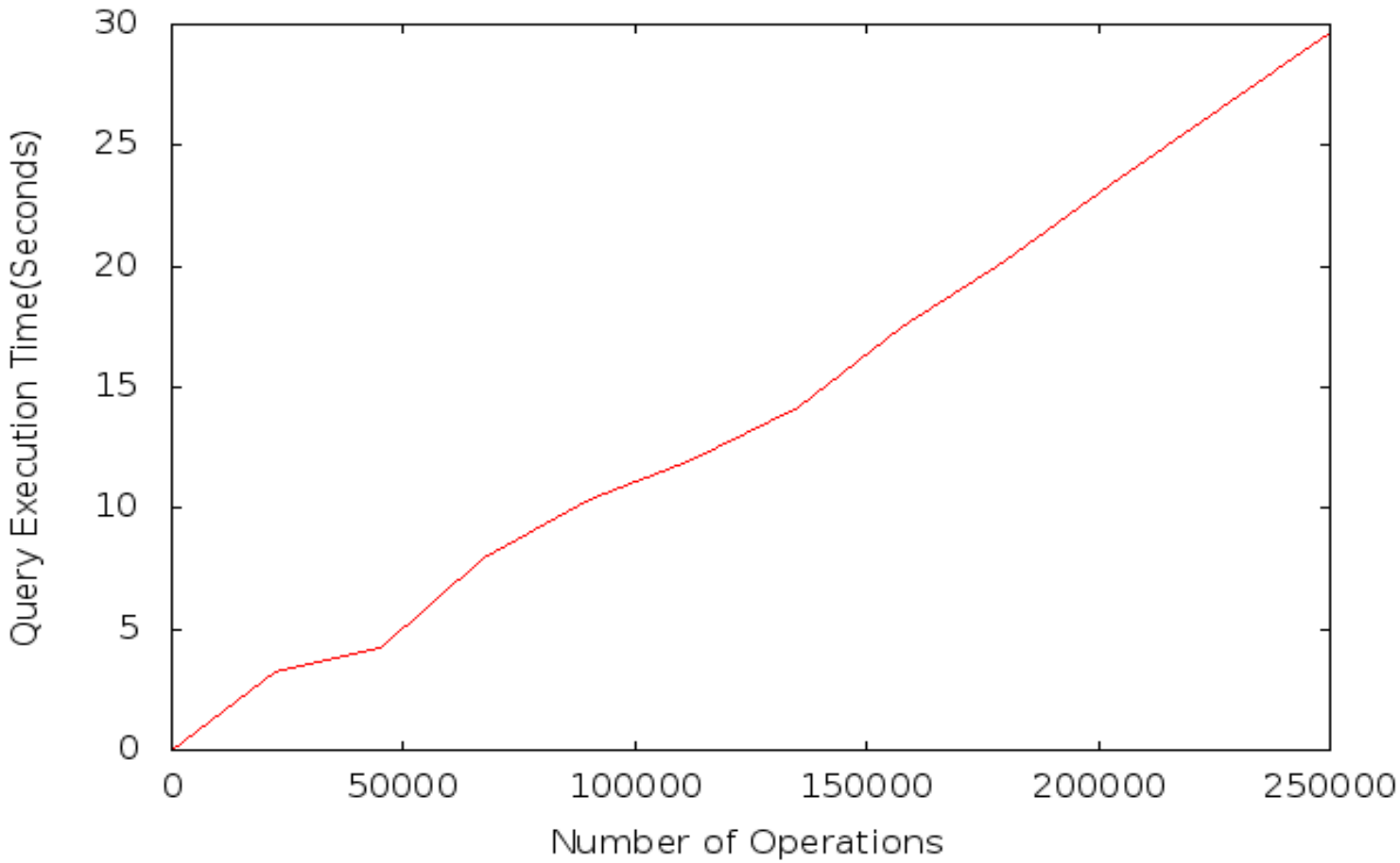
- Block-Info C-List
inode_Id	Block_Id	Time
- Block-Info M-List
inode_Id	Block_Id	Time	Original_Row

Listing files in a directory

Void ls(int stime, int id){

- children={Get children whose parentId=id};
- children = children - { children deleted before stime} -{ children created after stime} - {children moved_in after stime};
- children = children + {children moved-out after stime};
- modified-children = { children modified after stime};
- For-each children if it is modified first then moved then print former.
- }

Evaluation-Nested Snapshots



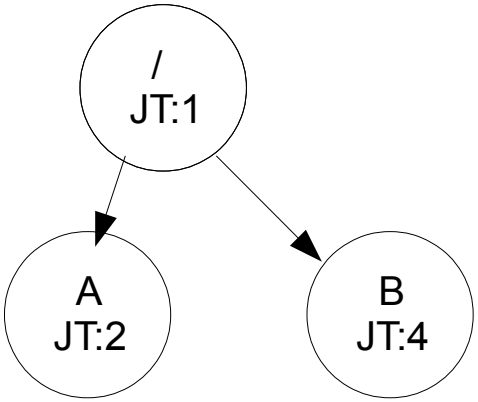
RO-Nested Snapshots

- How to determine whether an inode is in any snapshot ? Ex: /A/B/C/ Is C in any Snapshot?

RONS-Join Time

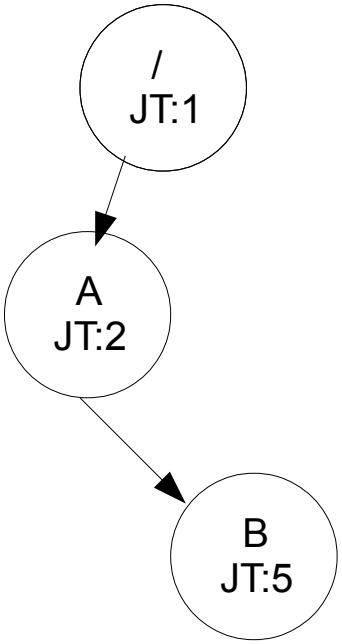
- JoinTime(JT): The time this inode joined its present path from root.

RONS-Join Time



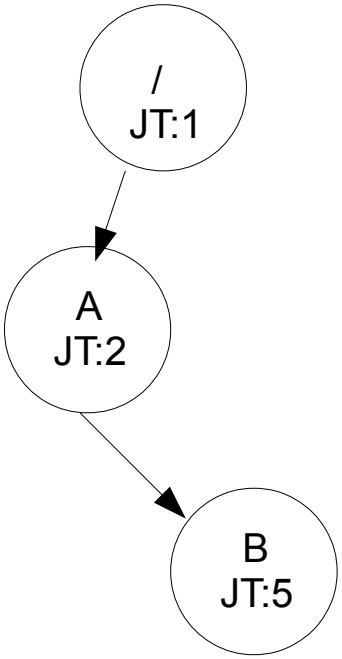
Id	Snap_Id	Time
A	SA1	3

RONS-Join Time



Id	Snap_Id	Time
A	SA1	3
A	SA2	6
B	SB1	7

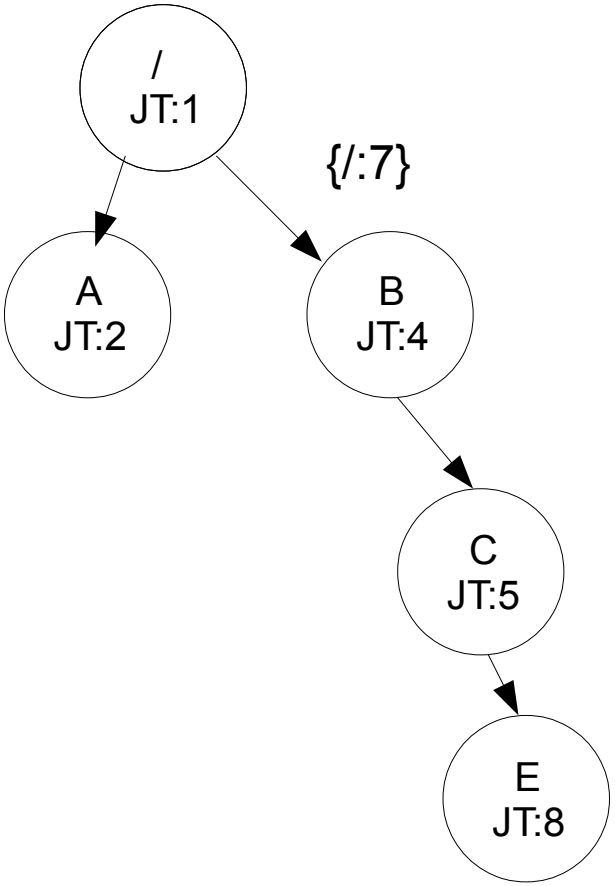
RO-Nested Snapshots



Id	Snap_Id	Time
A	SA1	3
A	SA2	6
B	SB1	7

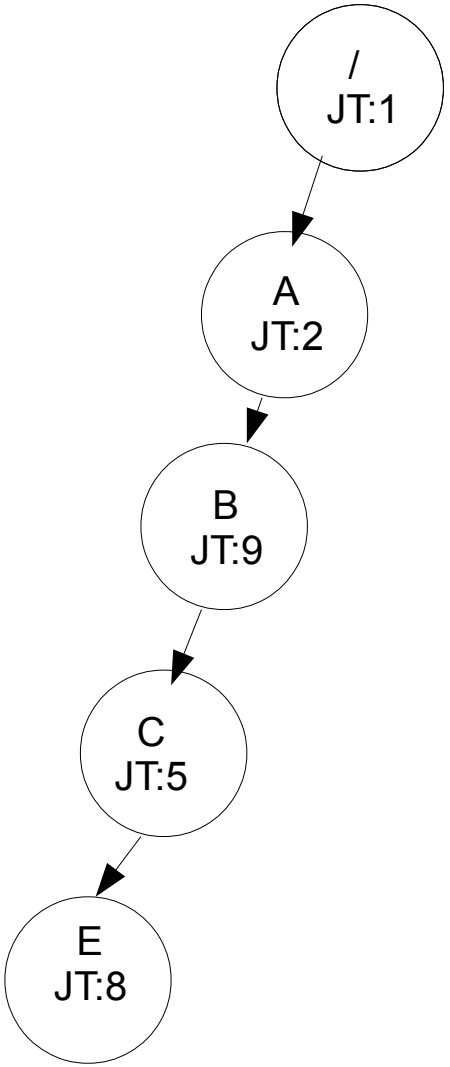
- Snapshots in which B is present
  - Snapshots taken on B ={SB1}
  - Snapshots taken on /[Root] after JT(B) &JT(A)={}
  - And Snapshots taken on A after JT(B)={SA2}

Move B to A



Id	Snap_Id	Time
A	SA1	3
B	SB1	6
/	S/1	7

Move B to A



Id	Snap_Id	Time
A	SA1	3
B	SB1	6
/	S/1	7

InodeSnapshotMap Table

Inode_Id	Belongs_to_ Inode_Id	StartTime	EndTime
B	/	7	7
C	/	7	7

Conclusion

- Efficient Root Level Snapshot.
- Efficient Nested Snapshot design.

Future Work

- Implementing Nested Snapshots
- Integrating RO Root Level Single Snapshot and RO Nested Snapshot solutions

Thank You