

**whamcloud**

The logo for Whamcloud features the word "whamcloud" in a bold, lowercase, sans-serif font. A thick blue horizontal line underlines the text. To the right of the text, a blue graphic element consists of a thick line that curves upwards and then downwards, resembling a stylized '3' or a bracket that encloses the end of the text.

# Imperative Recovery

- Jinshan Xiong /jinsan shiung/
- Jinshan.xiong@whamcloud.com

# Agenda

- Recovery 101
- Why is recovery slow
- How does imperative recovery help

## Recovery 101 1/2

- Lustre is a distributed file system built on multiple nodes
- Nodes are connected by networks
- Everything can fail:
  - Node power outage
  - Network partition
  - Software bugs
- Servers will rejoin the cluster after restarting
  - Recovery restores system consistent

## Recovery 101 2/2

- Service can't be interrupted during failures
- POSIX semantics have to be maintained always
- Recovery stages
  - Connection recovery
  - Transaction recovery
  - Llog is used to keep multiple nodes transactions in sync

## What makes recovery slow?

- Server must wait for all clients to reconnect
  - Recovery replays uncommitted client transactions
    - Must be executed in original order – transno
  - No new transactions until recovery completes
    - Could invalidate recovery transactions
- Clients slow to detect server death
  - Only fault detection is in-band RPC timeout
    - Includes both network and service latency
    - Server under heavy load hard to distinguish from dead server
  - Ping not scalable
    - Ping overhead  $O(\#servers * \#clients / ping\_interval)$
    - Ping interval must increase with system size
  - A client may know the server failure after ping interval + RPC timeout

# Introduction of Imperative Recovery

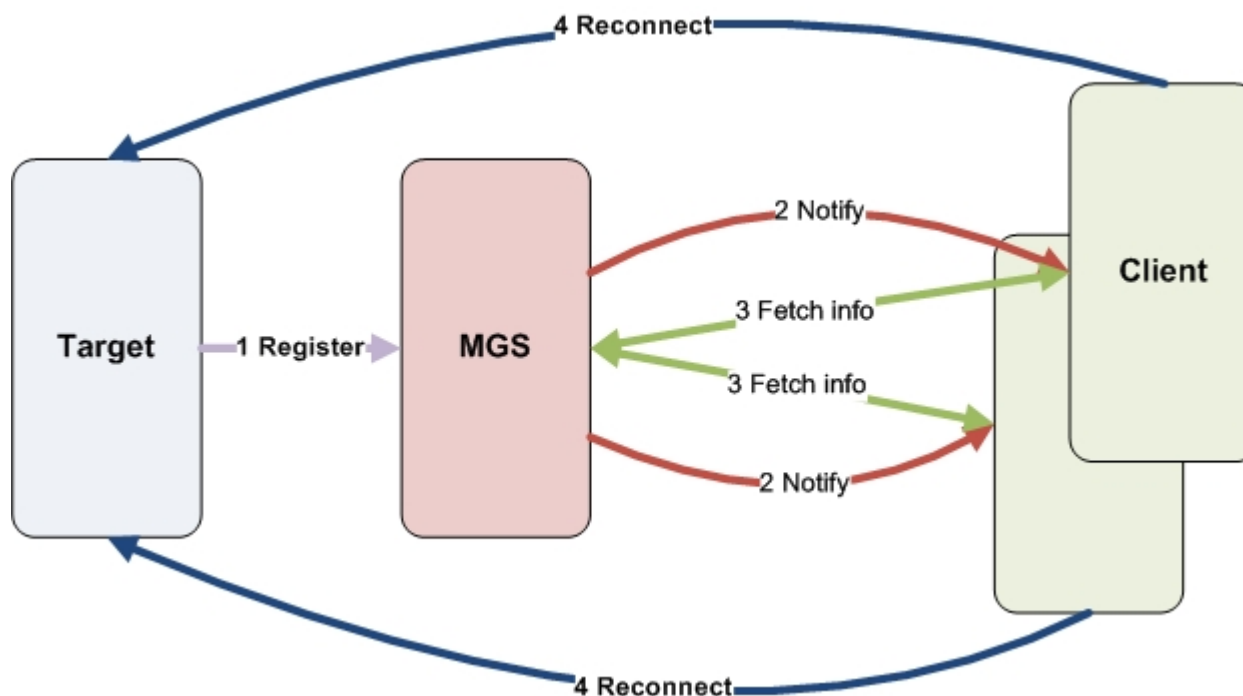
- Accelerate reconnection by notifying clients of server restarts, no longer use timeout
- MGS is used to reflect server failure event to clients
  - Notify clients when a restarted target registers itself to MGS
  - Clients will do reconnection
- Imperative recovery depends on MGS, it's a best-effort service
  - Not impede normal recovery from happening
  - It's important to identify which instance of targets the clients are connecting
- Failover server support

## Implementation - overall

- The MGS maintains a target NIDs table
  - This table records what NID the targets live on
  - Upon receiving a target register message, MGS changes the table
- There is an Idlm resource for each running file system on MGS
  - Recovlock on the resource
  - Clients hold shared mode recovlocks on this resource to cache the NID table
  - Whenever MGS updates the table, it enqueues an EXCL recovlock so the clients will be notified
- After being notified, clients will query MGS and then update its cache copy
  - Reconnecting to the target by NEW nid



# Implementation - overall



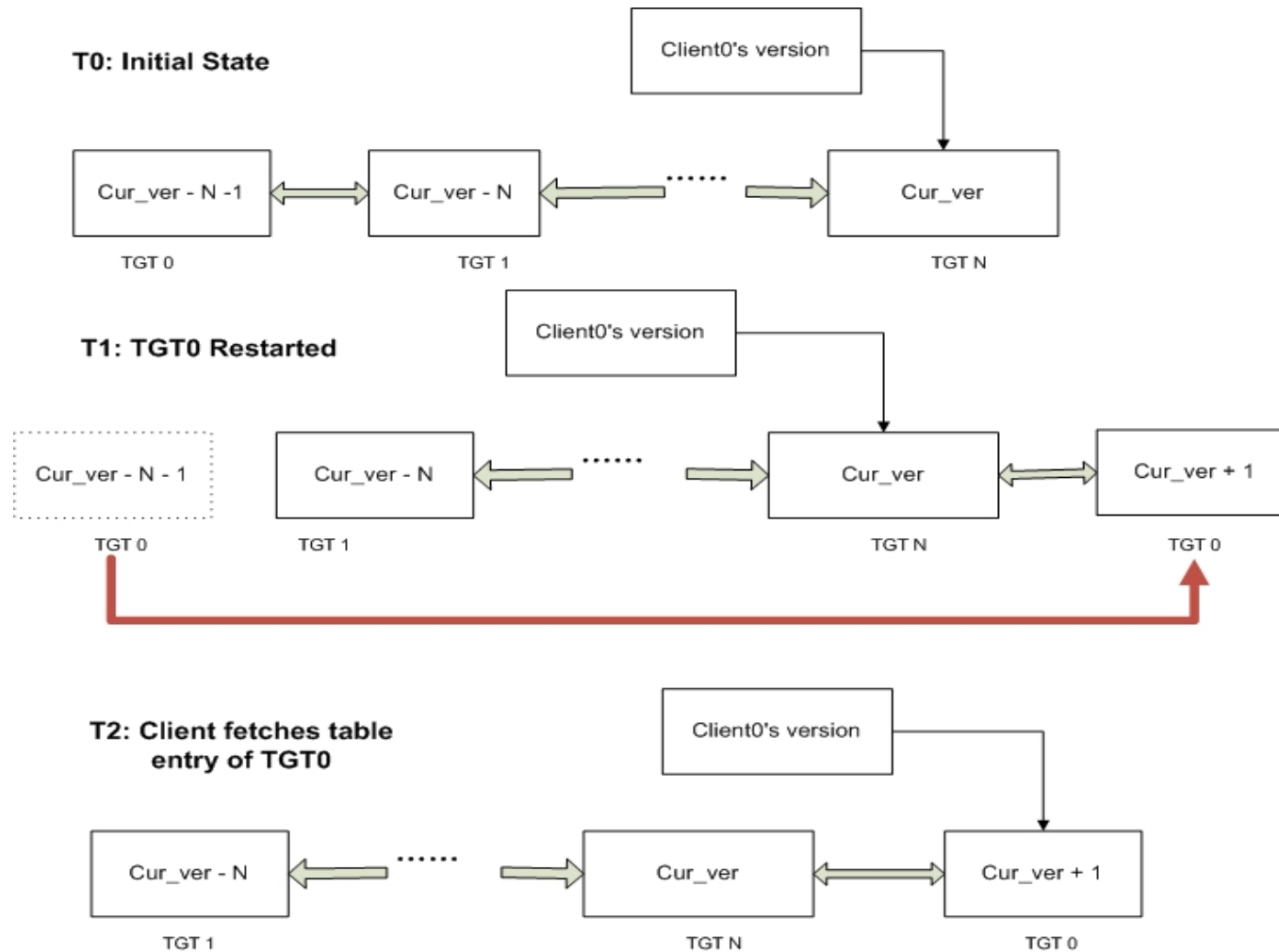
## Implementation – Target NID Table

- Version of Target NID Table
  - Version is increased by 1 whenever the Table is changed
  - Clients cache the target NID table locally with a version
  - Latest version has to be written into persistent storage
- Each living target has a entry in the table
- What info should be included in the NID table entry
  - Target name
  - Target server index – ost index or mdt index
  - # of target NIDs and NID list
    - 128 bytes per NID for `Inet_nid6_t`
  - Table version when the entry was last updated
  - Target instance number – uniquely identify a running target

## Implementation – Target NID Table

- The target NID table may be large
  - In a large cluster with 1K targets, it's impractical to transfer the whole table in one RPC
  - Only updated entries will be transferred
- MGS maintains table in a linked list with the increasing version # of entries
- Sync the table between the MGS and clients
  - When being notified, clients will provide their versions to the MGS, the MGS only returns the entries whose version # is greater than their version
  - Only a few entries will be fetched

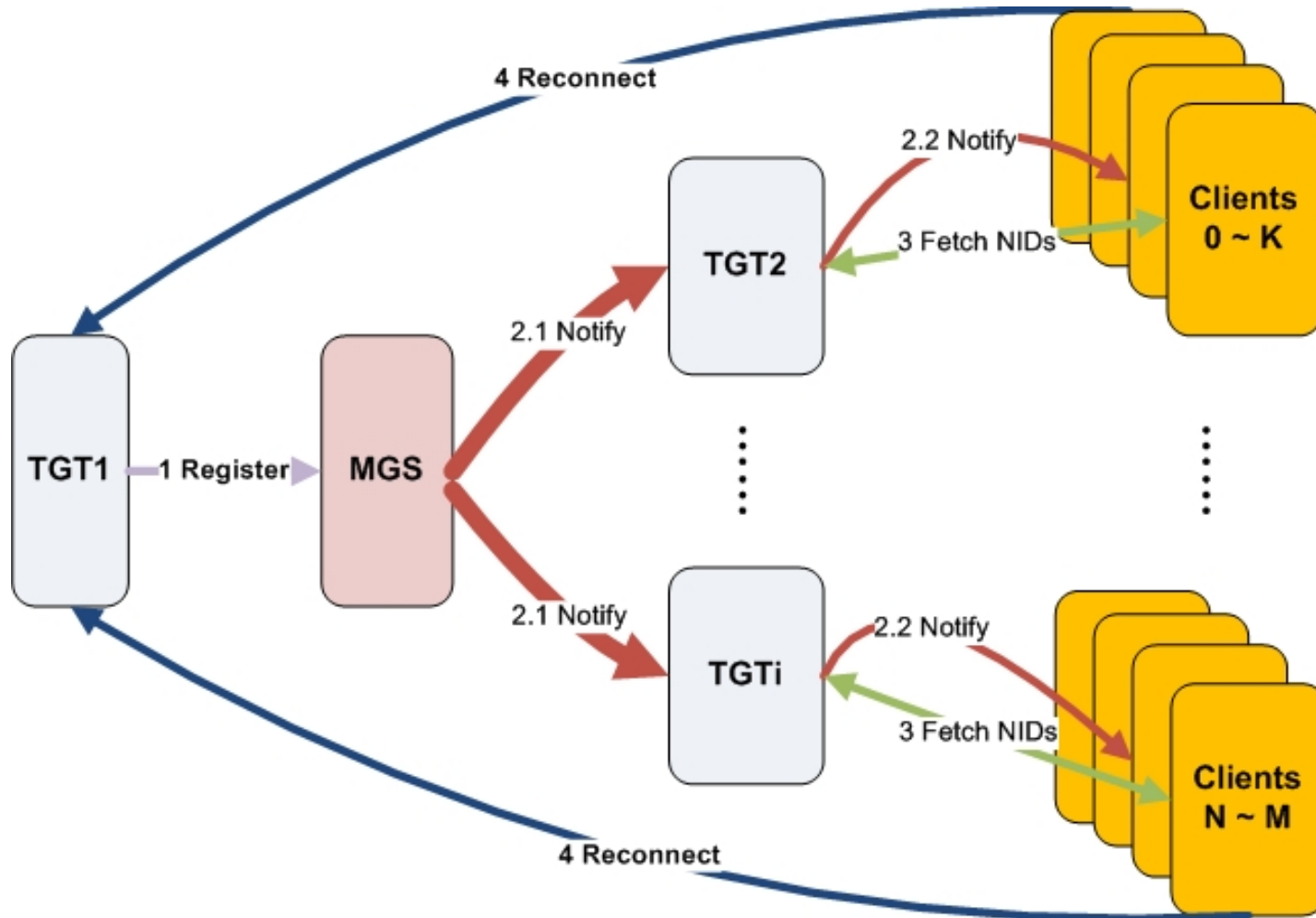
# Implementation – Target NID Table



## Implementation – Future work

- Problems of current implementation
  - MGS is the single point of failure
    - If MGS is down, imperative recovery won't work, but normal recovery is not affected
  - If the cluster is really big(100K clients?), it would be slow for the MGS to notify clients
    - MGS needs to send 100K RPCs to notify clients
    - Clients fetch update of entries with 100K RPC in total
    - Recovlock requeue overhead
- Solutions
  - Health network
  - Map/Reduce similar algorithm on MGS

# Implementation – Future work



## Work in Progress

- Planned to release in Q2 of 2012
- Sponsored by ORNL



**Thank You**

- Jinshan Xiong @ Whamcloud
- Jay@whamcloud.com