OHT: Hierarchical Distributed Hash Tables

Kun Feng, Tianyang Che

Outline

- Introduction
- Contribution
- Motivation
- Hierarchy Design
- Fault Tolerance Design
- Evaluation
- Summary
- Future Work

Introduction

- ZHT
  - Zero-Hop Distributed Hash Table
  - Light-weight, high performance, fault tolerant

Contribution

- Implement a hierarchical ZHT
- Server failure handling: verified
- Proxy failure handling: verified
- Dedicated listening thread for client
- Strong consistency in proxy replica group
- Demo Benchmark
- 1800+ lines of C++ code

Motivation

- Scalability of ZHT
    - n-to-n connection between clients and servers
    - Currently around 8000
- Hierarchical design
    - Add proxy to manage server groups

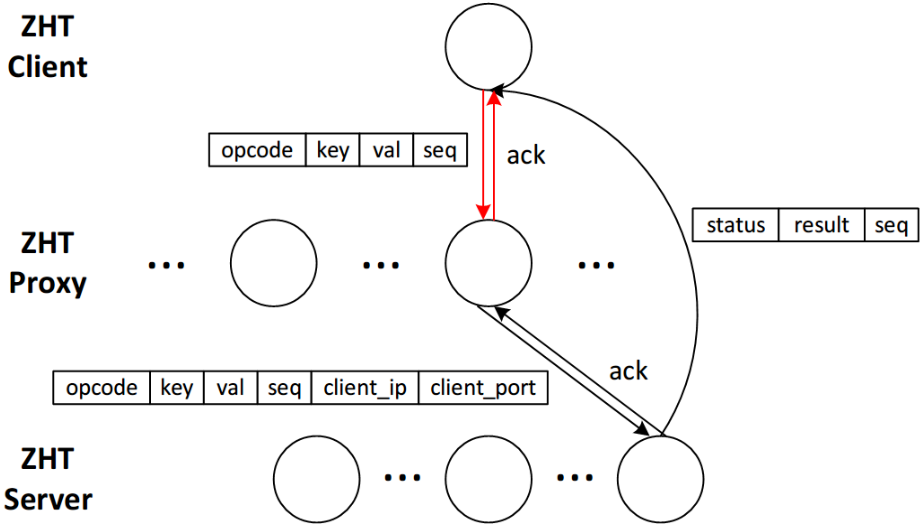
Hierarchy Design

- Add proxy layer between servers and clients
- Number of proxies is much smaller
- Each proxy manages several servers
- n-to-n connection among proxies
- 1-to-n connection between proxy and servers

Design

Client:

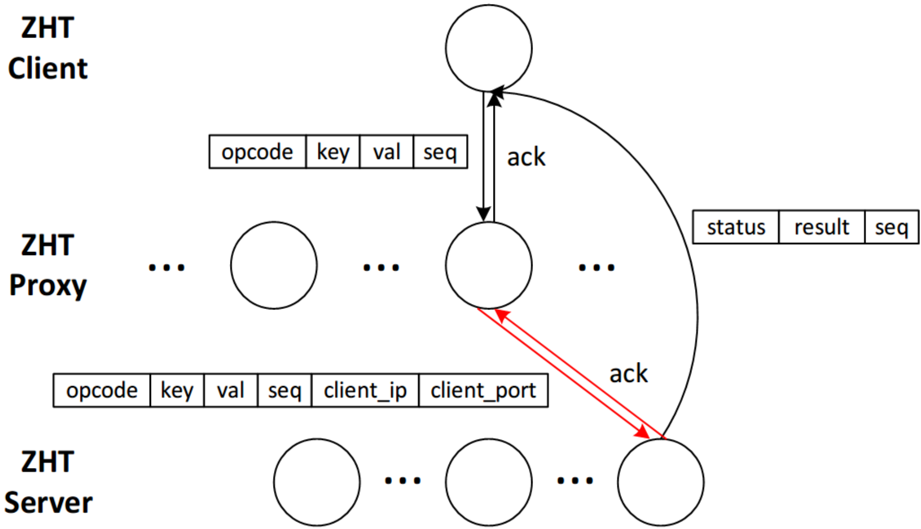
- Send requests to corresponding proxy
- Wait for ack from proxy (main thread)
- Dedicated listening thread to receive result from servers



Design

Proxy:

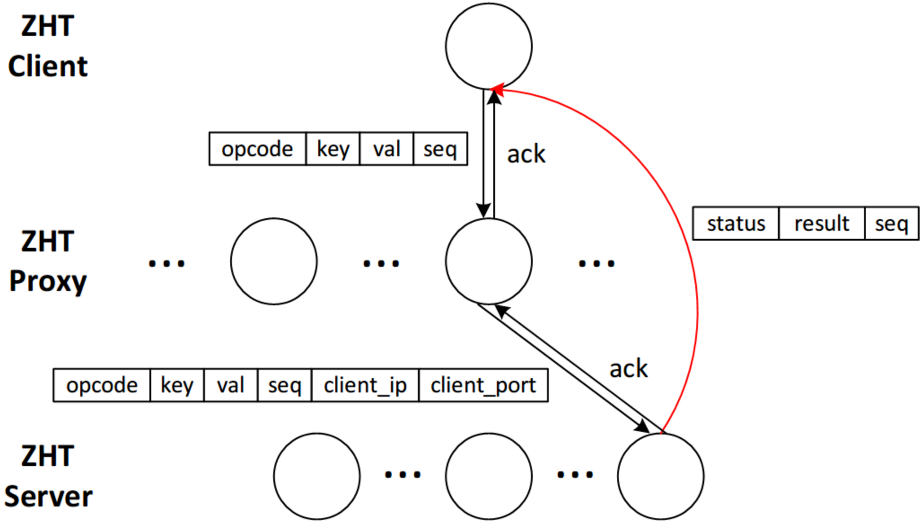
- Receive request from client
- Send client an ack
- Add client ip and port to request
- Forward the request to corresponding server
- Wait for ack from server



Design

Server:

- Wait for requests forwarded from proxy
- Process operation (lookup, insert ...)
- Send back the result directly to client



Fault Tolerance Design

Failure

- Server failure
- Proxy failure

Fault Tolerance Design

Server failure handling

- Detected by proxy
- Faulty server marked to be down (proxy)
- Randomly pick replica instead (proxy)
- Standby server (replicas, do nothing)

Fault Tolerance Design

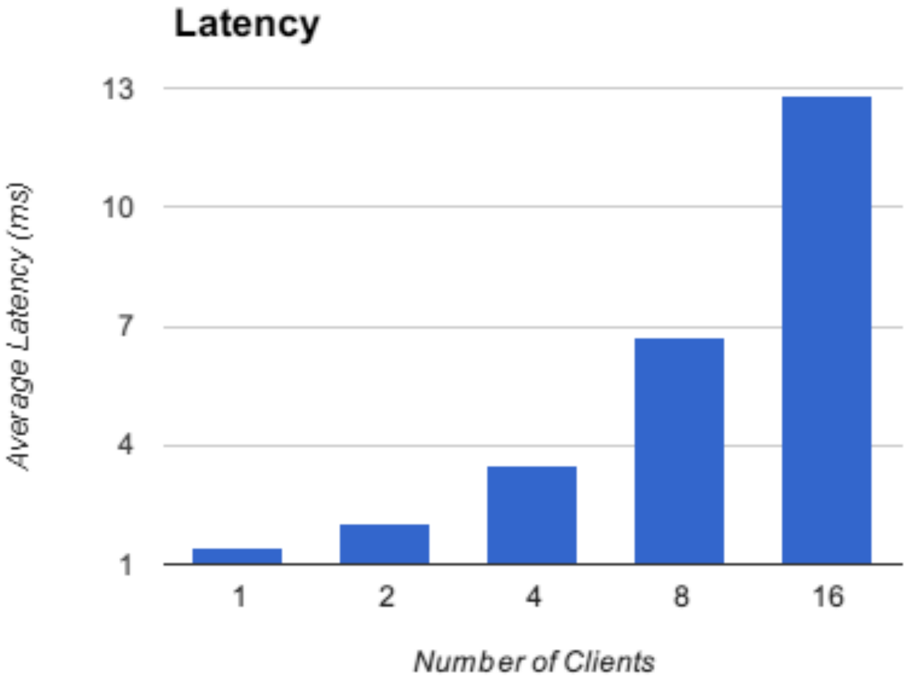
Proxy failure handling

- Detected by client
- Faulty proxy marked to be down (client)
- **Proxy broadcast this change to other proxies (strong consistent)**
- Randomly pick replica instead (client)
- Standby proxy (replicas, do nothing)

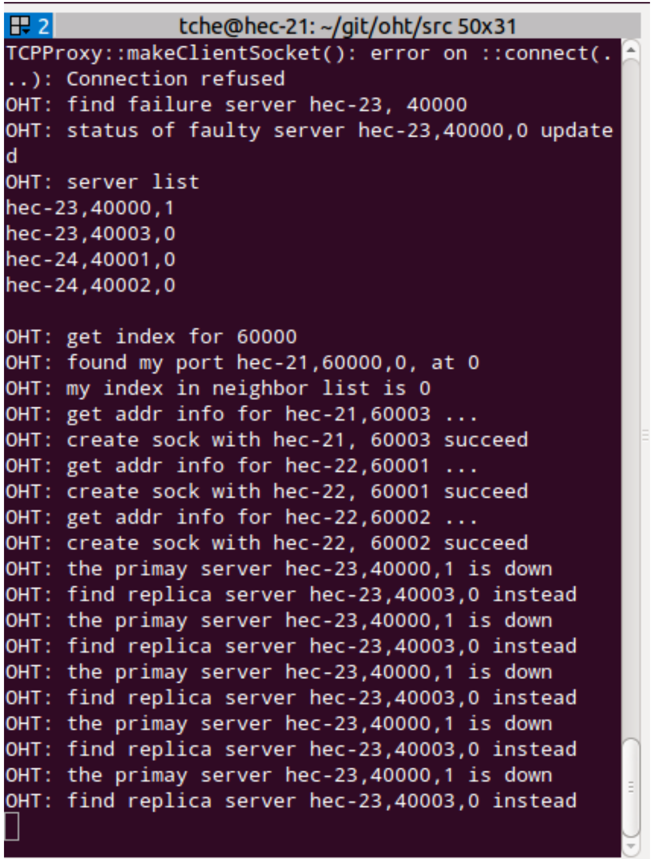
Evaluation

- Setup
  - HEC cluster in SCS lab
  - 2 proxies, 4 servers, 1 to 16 clients
  - Replicas: 2 for proxies, 2 for servers
  - Use zht_ben as benchmark

Evaluation

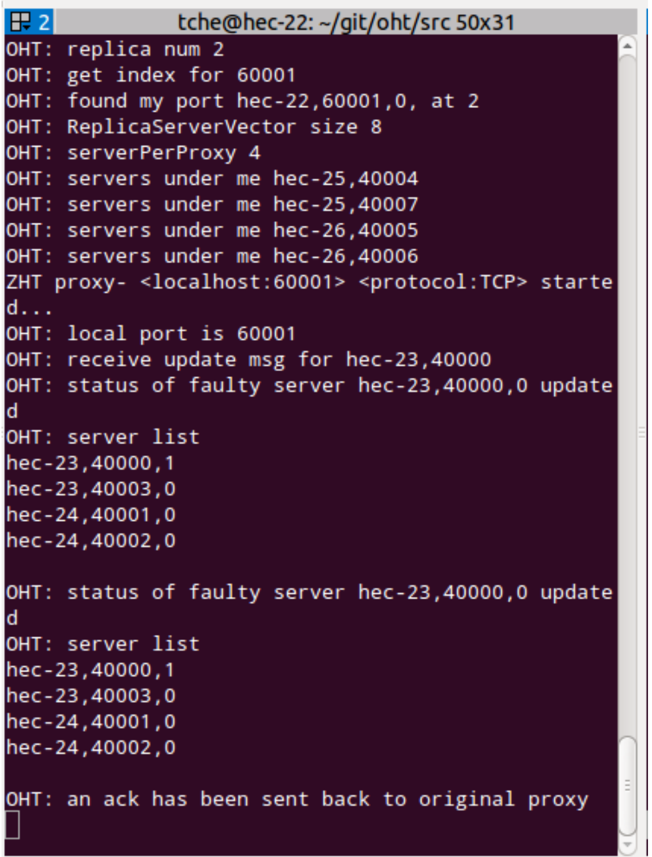

Latency
13
10
7
4
1
Average Latency (ms)
1
2
4
8
16
Number of Clients

Verifying Server Failure Handling



```
TCPProxy::makeClientSocket(): error on ::connect(.
..): Connection refused
OHT: find failure server hec-23, 40000
OHT: status of faulty server hec-23,40000,0 update
d
OHT: server list
hec-23,40000,1
hec-23,40003,0
hec-24,40001,0
hec-24,40002,0

OHT: get index for 60000
OHT: found my port hec-21,60000,0, at 0
OHT: my index in neighbor list is 0
OHT: get addr info for hec-21,60003 ...
OHT: create sock with hec-21, 60003 succeed
OHT: get addr info for hec-22,60001 ...
OHT: create sock with hec-22, 60001 succeed
OHT: get addr info for hec-22,60002 ...
OHT: create sock with hec-22, 60002 succeed
OHT: the primay server hec-23,40000,1 is down
OHT: find replica server hec-23,40003,0 instead
OHT: the primay server hec-23,40000,1 is down
OHT: find replica server hec-23,40003,0 instead
OHT: the primay server hec-23,40000,1 is down
OHT: find replica server hec-23,40003,0 instead
OHT: the primay server hec-23,40000,1 is down
OHT: find replica server hec-23,40003,0 instead
OHT: the primay server hec-23,40000,1 is down
OHT: find replica server hec-23,40003,0 instead
```

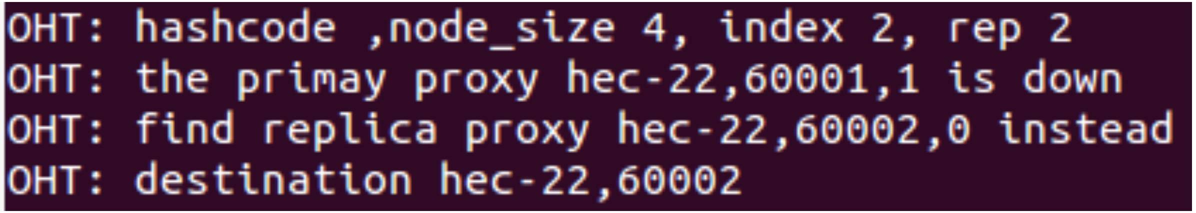


```
OHT: replica num 2
OHT: get index for 60001
OHT: found my port hec-22,60001,0, at 2
OHT: ReplicaServerVector size 8
OHT: serverPerProxy 4
OHT: servers under me hec-25,40004
OHT: servers under me hec-25,40007
OHT: servers under me hec-26,40005
OHT: servers under me hec-26,40006
ZHT proxy- <localhost:60001> <protocol:TCP> starte
d...
OHT: local port is 60001
OHT: receive update msg for hec-23,40000
OHT: status of faulty server hec-23,40000,0 update
d
OHT: server list
hec-23,40000,1
hec-23,40003,0
hec-24,40001,0
hec-24,40002,0

OHT: status of faulty server hec-23,40000,0 update
d
OHT: server list
hec-23,40000,1
hec-23,40003,0
hec-24,40001,0
hec-24,40002,0

OHT: an ack has been sent back to original proxy
```

Verifying Proxy Failure Handling



```
OHT: hashcode ,node_size 4, index 2, rep 2
OHT: the primay proxy hec-22,60001,1 is down
OHT: find replica proxy hec-22,60002,0 instead
OHT: destination hec-22,60002
```

Summary

- Implement a hierarchical ZHT
- Server failure handling
- Proxy failure handling
- Strong consistency in proxy replica group

Future Work

- Large scale test
- Merge eventual consistency code to server layer

Q & A