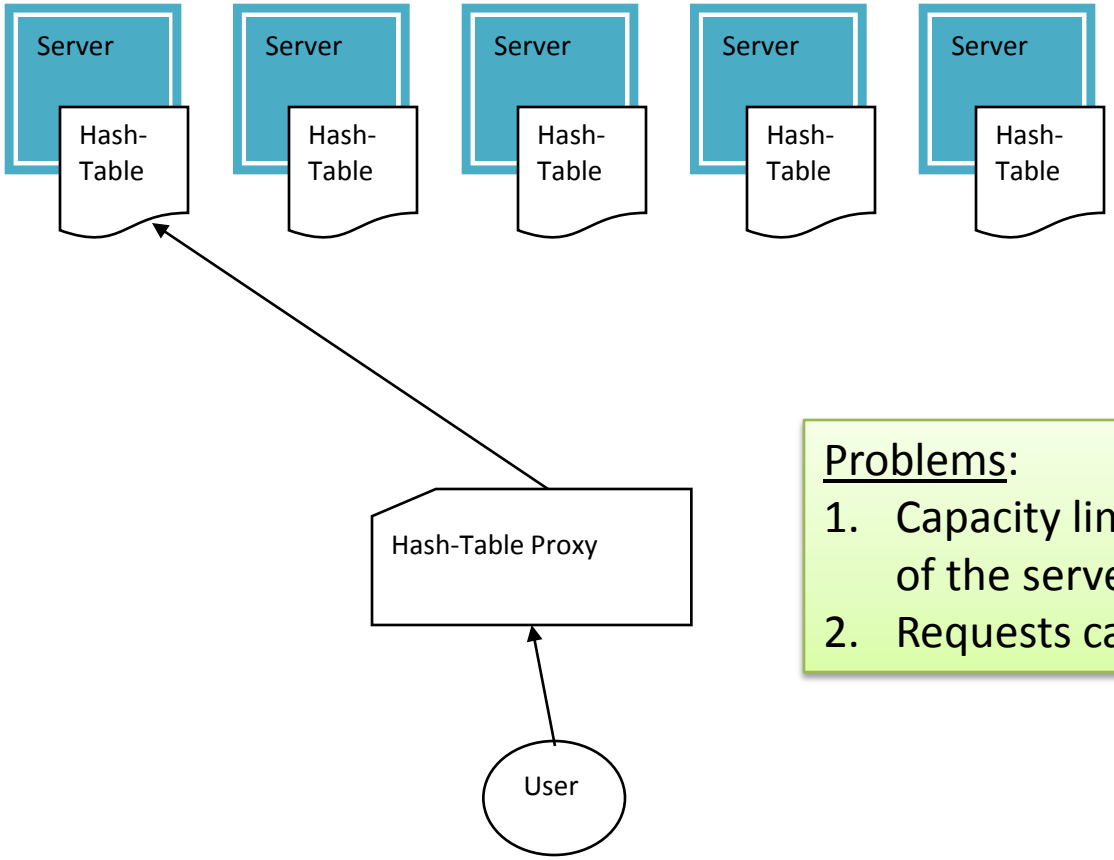
f-Fault Tolerant Distributed Hash-Table Implementation Using OpenReplica

Anastasios Souris

A Simple f-fault tolerant Hash-Table

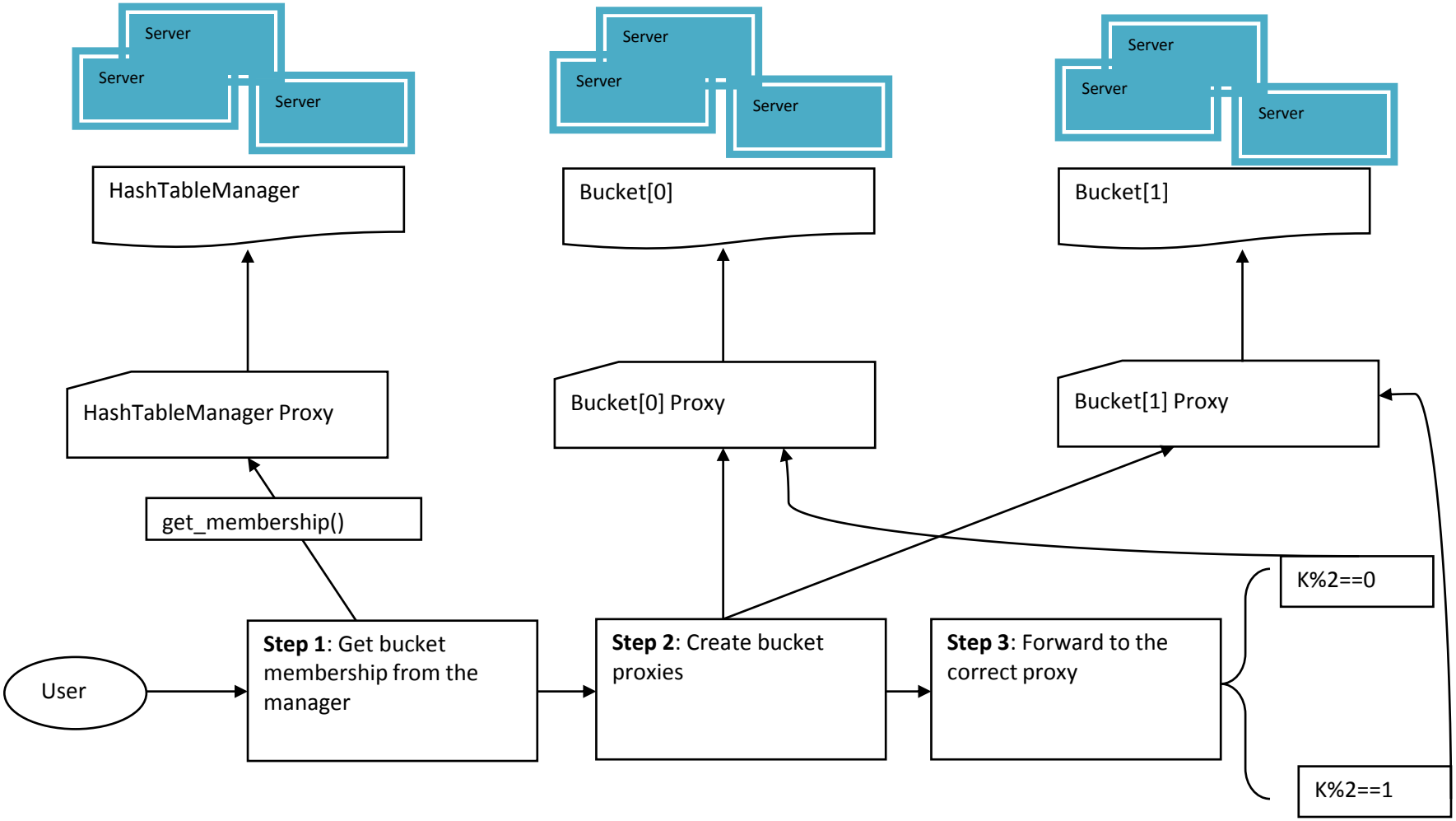
class sequential_hash_table → concoord object –o … → proxy



Problems:

1. Capacity limited to the minimum capacity of the server machines
2. Requests cannot be handled in parallel

Now Distributed (Static Case Without Resizing)



Pseudo-Latency Results

(n,m)	Time (in seconds)
(16,1)	~ 21.5
(16,2)	~ 15.9
(16,4)	~12.3
(16,8)	~10.3
(16,16)	~9.5

Adding more buckets

- Use Case:
  - The provider of the hash-table gets access to more servers which he/she wants to add to the hash-table
- Restrictions:
  - Only the service provider can add buckets *one-at-a-time*
    - Maybe not so…
  - No shrinking
    - Worst-case: *migrate* the replicas to servers holding other buckets

How to Split A Bucket (1/3)

1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1001
12	1100
13	1101
14	1110
15	1111

Bucket[0]	0010 0100 0110 1000 1010 1100 1110
Bucket[1]	0001 0011 0101 0111 1001 1001 1101 1111

Bucket[0]	0100 1000 1100
Bucket[1]	0001 0011 0101 0111 1001 1001 1101 1111
Bucket[2]	0010 0110 1010 1110

- Only data movement from Bucket[0] to Bucket[2]!!!
- When Bucket[2] is added only requests to Bucket[0] are affected

Bucket[0]	0100 1000 1100
Bucket[1]	0001 0101 1001 1001 1101
Bucket[2]	0010 0110 1010 1110
Bucket[3]	0011 0111 1111

How to Split A Bucket (2/3)

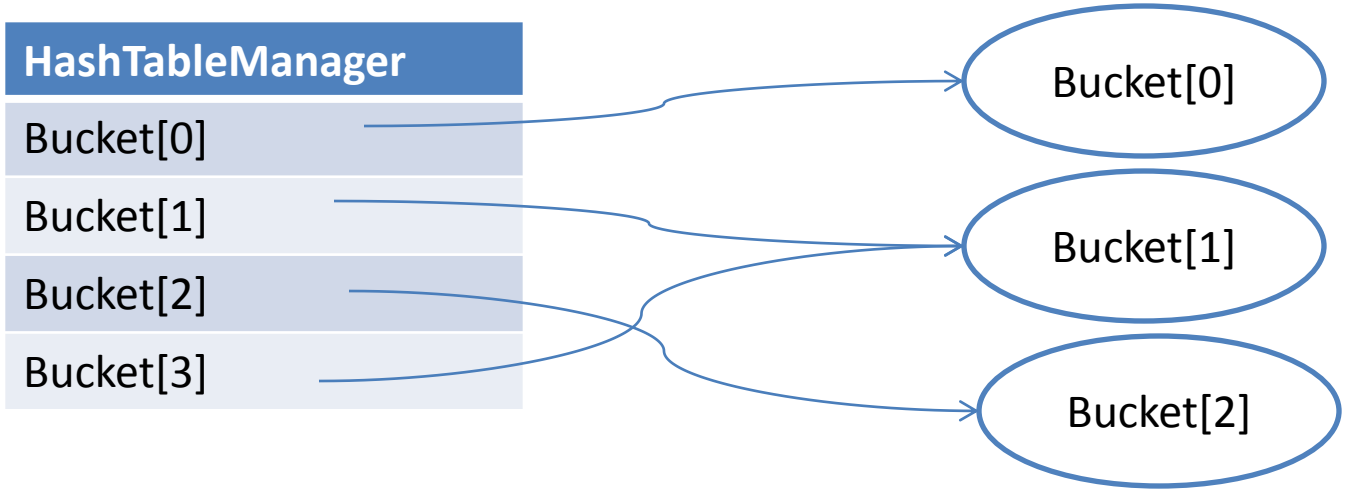
Bucket[0]	1**000**
Bucket[1]	00**01** 01**01** 10**01** 10**01** 11**01**
Bucket[2]	00**10** 01**10** 10**10** 11**10**
Bucket[3]	00**11** 01**11** 11**11**
Bucket[4]	0**100** 1**100**

- ✓Bucket[5] will split from Bucket[1]
- ✓Bucket [6] will split from Bucket[2]
- ✓Bucket[7] will split from Bucket [3]

How to Split A Bucket (3/3)

- This method works only for tables with size a power of two

Bucket[0]	0100 1000 1100
Bucket[1]	0001 0011 0101 0111 1001 1001 1101 1111
Bucket[2]	0010 0110 1010 1110
Bucket[3]	???



Buckets form a List

- The buckets form a list such that the contents of some bucket if they are split over to other buckets, those buckets are on the right
- 0 → 1
- 0 → 2 → 1
- 0 →2 → 1 → 3
- 0 → 4 → 2 → 6 → 1 → 5 → 3 → 7
- Each bucket (replica) holds a link to the next bucket in that list
  - i.e dns name, ip-port pairs

Bucket Addition Protocol

- Protocol Executed by the Service Provider:
  1. Get from the HashTableManager the next Bucket to be split: the *victim bucket*
  2. The service provider gets from the *victim* bucket a *snapshot* of the key-space that will be transferred to the new bucket
  3. It transfers that snapshot to the new bucket
  4. It tells the victim bucket that it is no longer responsible for that key-space and it provides the link to the next bucket
  5. Tells the HashTableManager of the new bucket

A Small Clarification

- Why steps (4) and (5) must be done in that order
- Because *otherwise* this bad thing could happen:
  1. I split from the *victim* bucket to a new bucket
  2. I tell the HashTableManager of the new bucket
  3. *Before* I inform the victim bucket:
     1. A new client arrives and sees the new bucket.
     2. The old clients still use the victim bucket for the key-space that has moved to the new bucket

Concurrent Client Requests and Splits (1/2)

- Bad Scenario:
  - A split occurs to some victim bucket
  - The service provider gets a *snapshot* from the victim bucket
  - While the split is in progress:
    - Some clients may alter the key-space being transferred with put/remove operations sent to the victim bucket
  - So the snapshot went to the new bucket is stale.
  - Note: <u>The victim bucket can respond to *get()* operations in this case</u>

- Bad Solution:
  - While a split is in progress have the victim bucket know it
  - If a request arrives that affects the snapshot remember it
  - When the provider comes and informs the victim bucket that the split is over respond with a new snapshot
    - Getting more snapshots is bad
- My Solution:
  - Stop the clients

Concurrent Client Requests and Splits (2/2)

- When a put/get operation arrives to a victim bucket that affects a split in progress
  - Non-Blocking Solution:
    - Raise an exception
    - The client will keep querying the bucket (i.e with backoff)
    - When the split is over the bucket will have the link to the next bucket so when the client asks it will return the link to the next bucket and the client will follow that link(s)
      - Due to the list I know that the key is somewhere on the right
    - At some point it will re-load the buckets from the HashTableManager
  - Blocking Solution:
    - OpenReplica Rendezvous objects
    - Block the client
    - When the split is over and the service provider informs the victim bucket with the next link
    - Unblock all the blocked clients with the next link
    - **This doesn't work**

One Last Note

- Probably:
  - Hash table of size M
  - To add M more buckets (new size 2M) → in parallel
    - Intuition: each bucket splits from a different bucket from the initial M