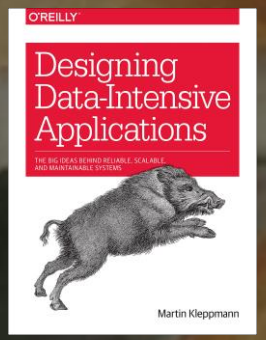
MSFT Sys Meetup



http://

Freely accessible resources

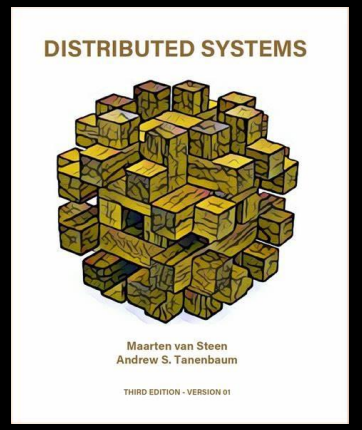
[Code]

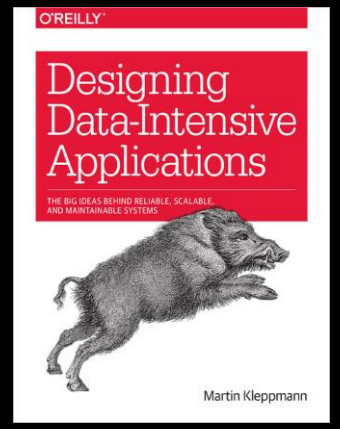
[Zoom]

[Course]

[DDIA (O'Reilly)]

[Distributed System 3rd edition]





Calendar: https://docs.google.com/spreadsheets/d/1RsbGpq1cwNSmYn5hcmT8Hv5O4qssl2HXsTcG82RHVQk/edit?usp=sharing

(Internal) Teams: g078pwd

(Public) Discord

(Public) WeChat: add mossaka or Lin1991Wen

Notion: https://www.notion.so/invite/cd6df70a94e7f67f6d21f4c509783d3c9cfd0e69

YouTube: https://www.youtube.com/playlist?list=PL1voNxn5MODMJxAZVvgFHZ0jZ-fuSut68

Company Privacy

Microsoft

A little about me

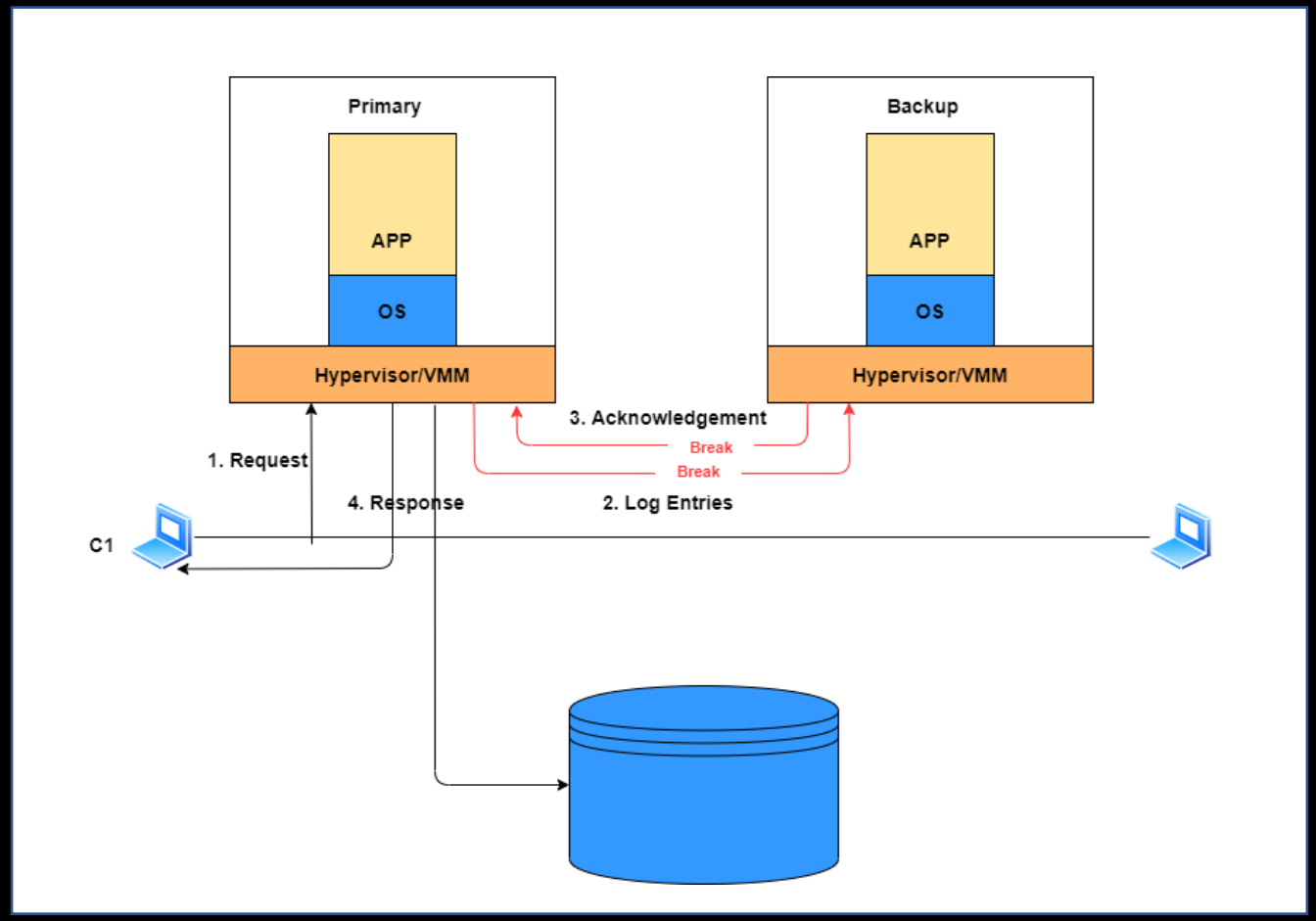
- Wantong Jiang (姜宛彤)
- Microsoft Azure Networking, SDE
  - SDN
  - Network Virtualization
- B.S. in CS at Peking University
- M.S. in CS at Stanford University

What is Zookeeper?

A stand-alone and general-purpose service providing wait-free coordination for distributed application processes.

- Stand-alone
  - Not a library, but a cluster of servers running, processing client requests, and maintaining data
  - Clients use the service via Zookeeper client API
- General-purpose
  - No specified primitives (e.g. locks) implemented, clients build their own
- Wait-free APIs
  - No blocking primitives
  - Event-driven (watches)

Zookeeper Applications



- Vmware Fault-Tolerance Test-and-Set Service
  - Avoid split-brain
- Master election
- Group membership
- Meta-data maintenance

(More on Zookeeper wiki)

Zookeeper Client API

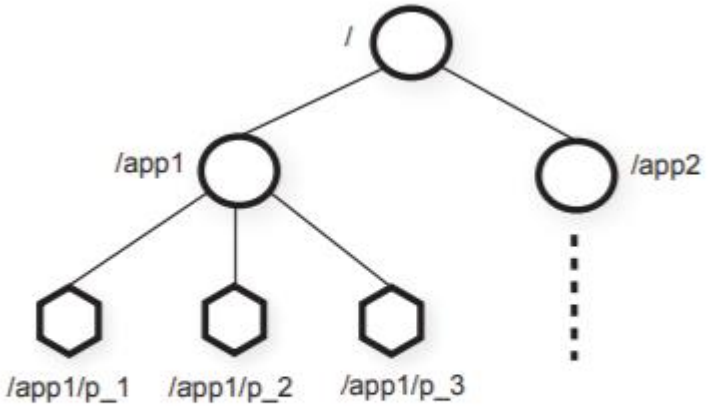


Figure 1: Illustration of ZooKeeper hierarchical name space.

```
create(path, data, flags); // regular, ephemeral, sequential

delete(path, version); // conditional delete

exists(path, watch);

getData(path, watch);

setData(path, data, version);

getChildren(path, watch);

sync(path);
```

Zookeeper API Usage Example – Mini-transaction

```
while true:
    x, v := getData("/f")
    if setData(x + 1, version=v):
        break
```

Atomic read-modify-write

Zookeeper API Usage Example – Locks

Simple Lock

```
Lock():
    while true:
        if create("/f", ephemeral=true)
            return # success
        if exists("/f", watch=true)
            # wait for notification

Unlock():
    delete("/f")
```

Simple Lock without Herd Effect

Lock

```
1 n = create(l + "/lock-", EPHEMERAL|SEQUENTIAL)
2 C = getChildren(l, false)
3 if n is lowest znode in C, exit
4 p = znode in C ordered just before n
5 if exists(p, true) wait for watch event
6 goto 2
```

Unlock

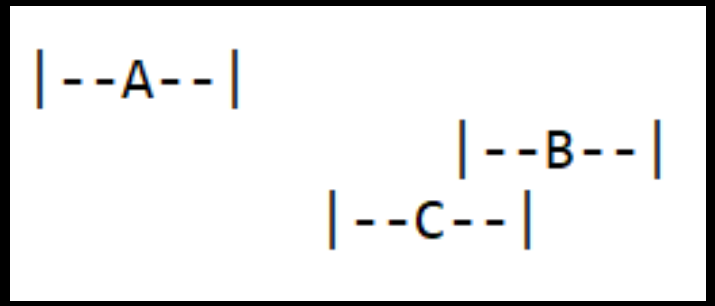
```
1 delete(n)
```

Zookeeper Guarantees

- Globally Linearizable writes (But not reads)
  - Clients send writes to the leader
  - Leader chooses an order and executes the writes in the order (like Raft)
- Per-client FIFO order
  - Clients specify an order for its operations (reads AND writes), *zxid*
  - Writes are executed in zxid order
  - Reads
    - Each read executes at a particular point in the write order
    - A client's successive reads execute at non-decreasing points in the order
    - A client's read executes after all previous writes by that client
    - A left-over server may need to block and catch up before replying to a client!

Linearizability

- Observed effects equivalent to all operations happening in some sequential order in which non-overlapping operations are temporally ordered.

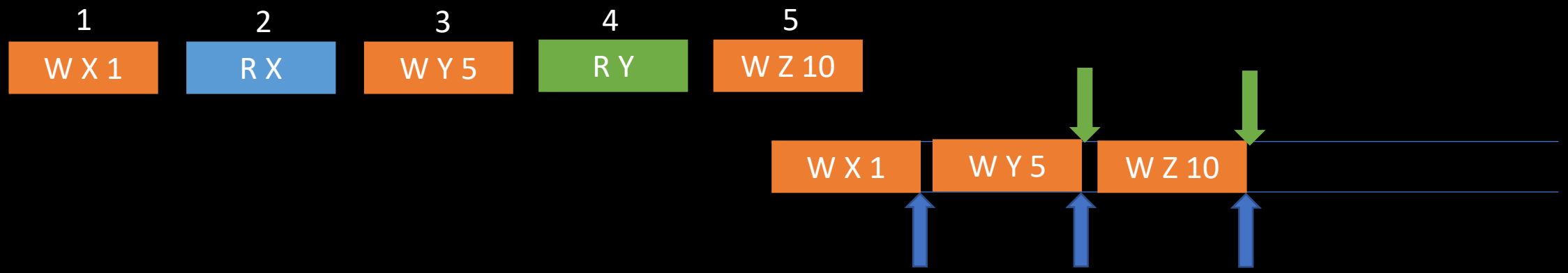


A must happen before B and C.
B and C can happen in either order, but all clients must see the same.

- Raft: linearizable, all operations go to the leader
- Zookeeper: writes linearizable; reads handled on local server

Per-client FIFO Order

- Clients specify an order for its operations (reads AND writes), *zxid*
- Writes are executed in zxid order
- Reads
  - Each read executes at a particular point in the write order
  - A client's successive reads execute at non-decreasing points in the order
  - A client's read executes after all previous writes by that client
  - A left-over server may need to block and catch up before replying to a client!



Tradeoff

- Good read performance
  - Each server deal with reads locally, good scalability
- Reads may not be consistent
  - Reads may return stale data (written by other clients)

- Why feasible?
  - Zookeeper workload is mainly read only
  - Sufficient to implement coordination primitives
  - Sync() for reading latest data

Intuition

```
Write order:                Read order:
delete("ready")
write f1
write f2
create("ready")
                            exists("ready")
                            read f1
                            read f2

=========================================================

                            exists("ready", watch=true)
                            read f1
delete("ready")
write f1
write f2
                            read f2
```

Other Implementation Details

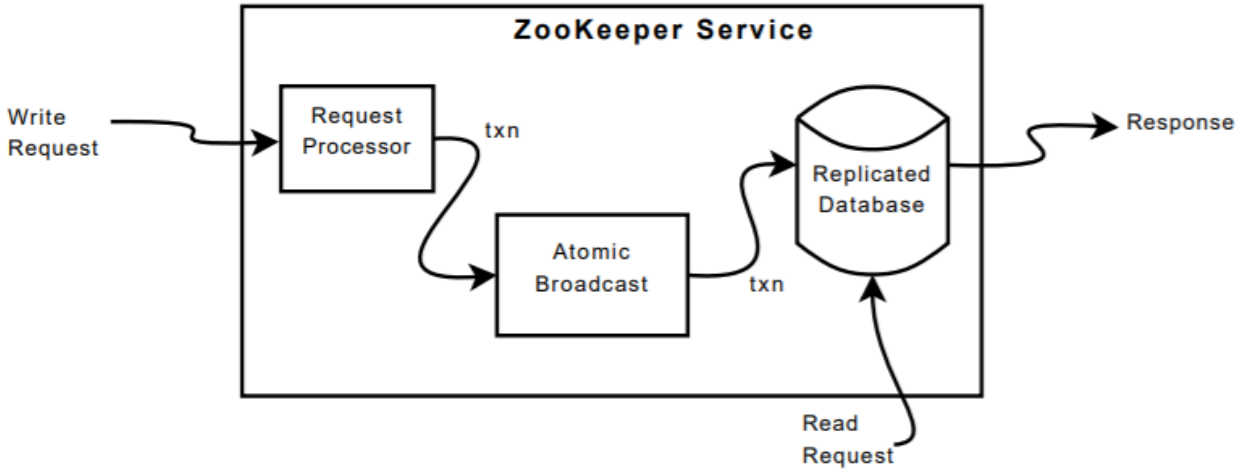


Figure 4: The components of the ZooKeeper service.

- Pipelining
  - Improve processing throughput
  - APIs can be asynchronous, allow multiple requests on the flight
  - Requests can further be batched

Other Implementation Details

- Request Processor
  - Translate non-idempotent APIs into idempotent transactions
  - <transaction_type, path, value, new_version>
  - May need to compute future states (e.g. seq number)

- Zab
  - Zookeeper atomic broadcast
  - Raft like consensus protocol
  - May have to resend txns

Other Implementation Details

- Replicated Database
  - All data stored in memory
  - Write-ahead logs
    - Maximum 1MB, memory capable
    - Reads do not need disk access
    - Leverage Zab's log as write-ahead log
    - Sequential disk writes
  - Fuzzy snapshots (idempotent)

Performance

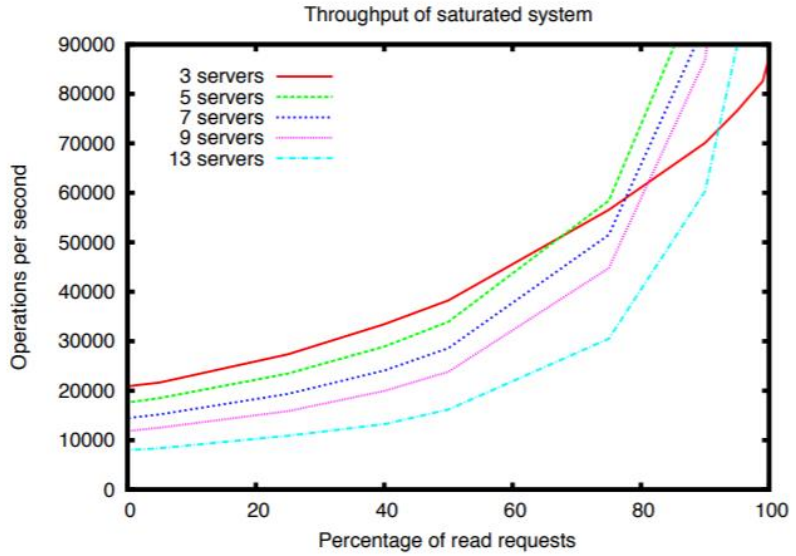


Figure 5: The throughput performance of a saturated system as the ratio of reads to writes vary.

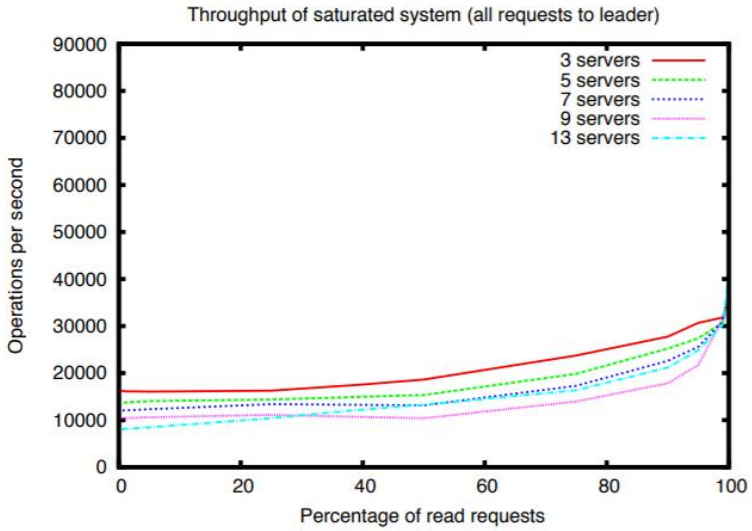


Figure 6: Throughput of a saturated system, varying the ratio of reads to writes when all clients connect to the leader.

Questions?

Freely accessible resources

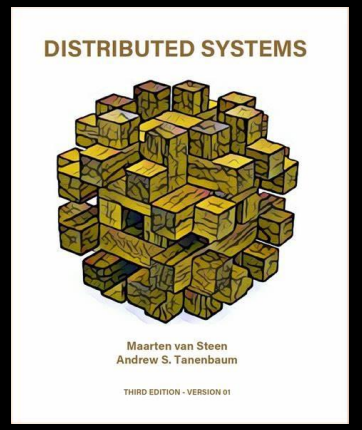
[Code]

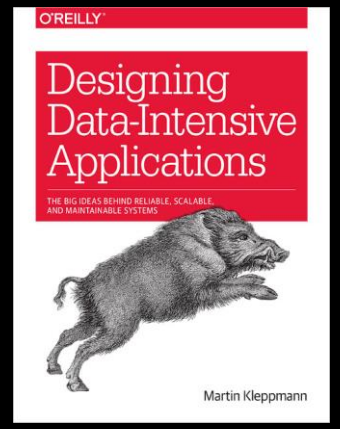
[Zoom]

[Course]

[DDIA (O'Reilly)]

[Distributed System 3rd edition]





Calendar: https://docs.google.com/spreadsheets/d/1RsbGpq1cwNSmYn5hcmT8Hv5O4qssl2HXsTcG82RHVQk/edit?usp=sharing

(Internal) Teams: g078pwd

(Public) Discord

(Public) WeChat: add mossaka or Lin1991Wen

Notion: https://www.notion.so/invite/cd6df70a94e7f67f6d21f4c509783d3c9cfd0e69

YouTube: https://www.youtube.com/playlist?list=PL1voNxn5MODMJxAZVvgFHZ0jZ-fuSut68