Testing Raft-replicated Database Systems

Guohao Ding*, Weining Qian*, Peng Cai*, Tianze Pang+, and Qiong Zhao+

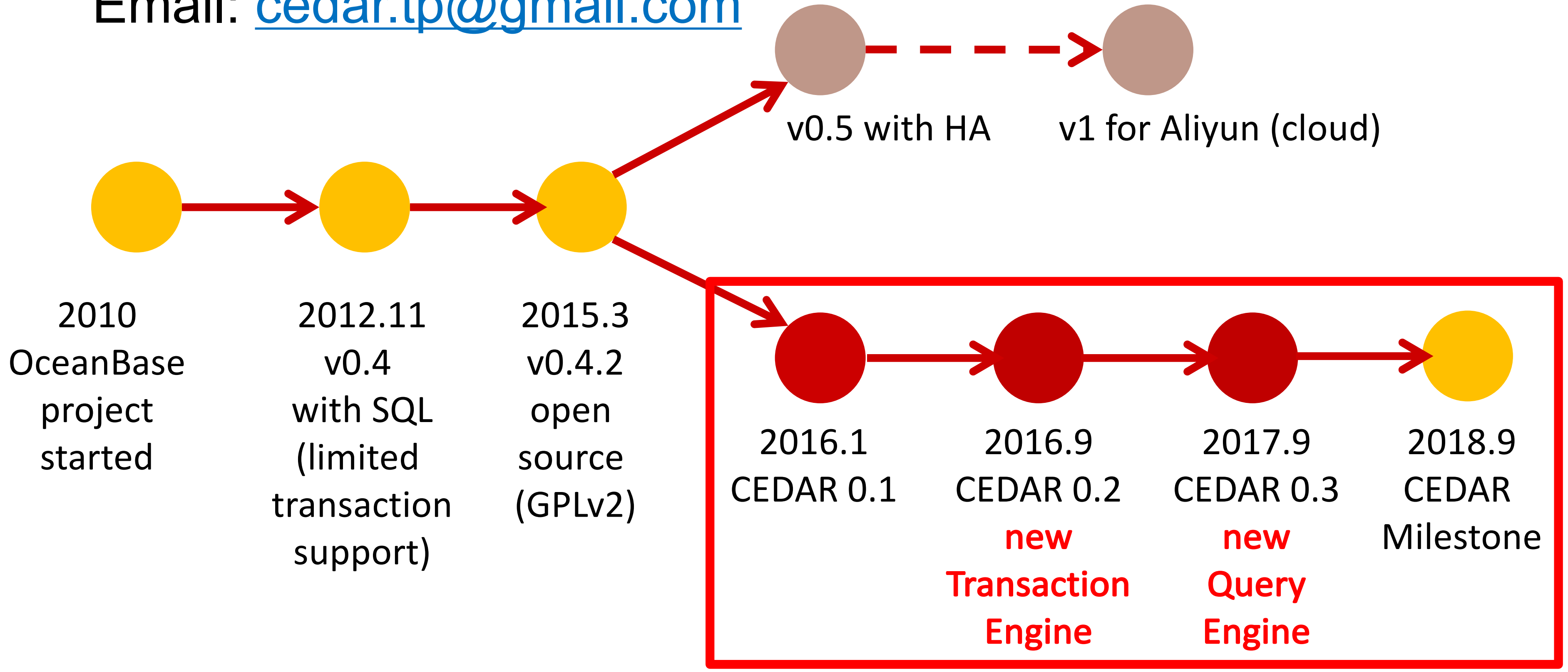
* School of Data Science and Engineering, East China Normal University

+ Bank of Communications

Outline

Motivation

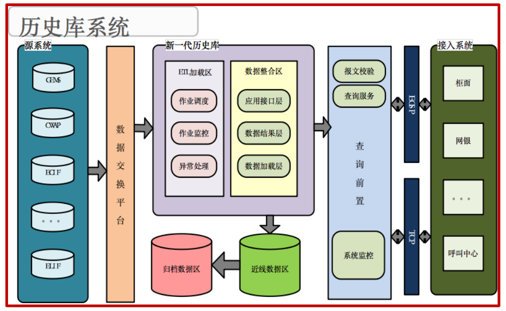
Homepage: https://github.com/daseECNU/CEDAR
Email: cedar.tp@gmail.com



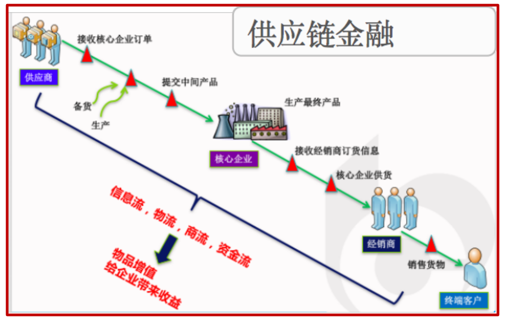
v0.5 with HA
v1 for Aliyun (cloud)
2010 OceanBase project started
2012.11 v0.4 with SQL (limited transaction support)
2015.3 v0.4.2 open source (GPLv2)
2016.1 CEDAR 0.1
2016.9 CEDAR 0.2 new Transaction Engine
2017.9 CEDAR 0.3 new Query Engine
2018.9 CEDAR Milestone

Applications

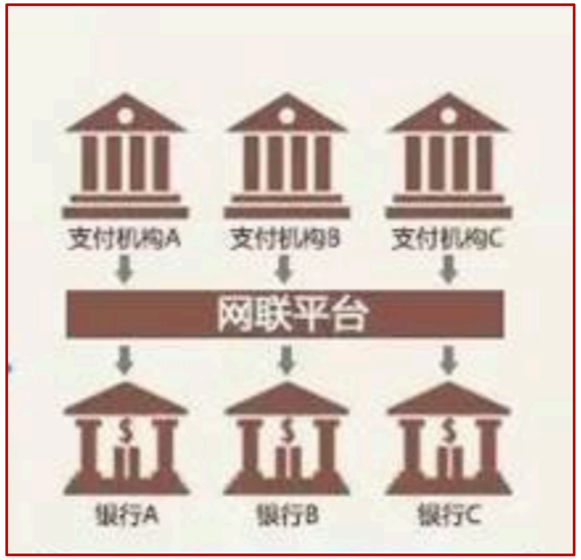
Bank notes recording
Massive datasets



History repository
Class-A app.



Supply-chain finance
Replacing IBM DB2



Netpay
Debt-Credit
Internet-scale
HA

Motivation

The core of implementing a distributed and highly-available database system is consensus protocols

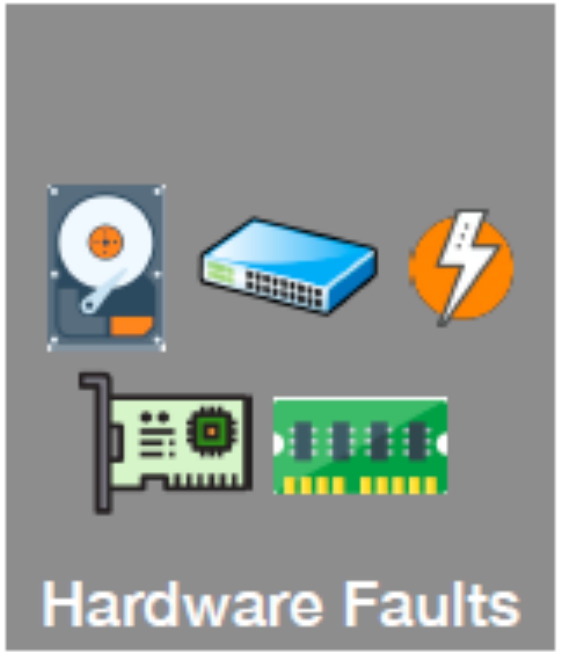
- Paxos
- Raft
- …

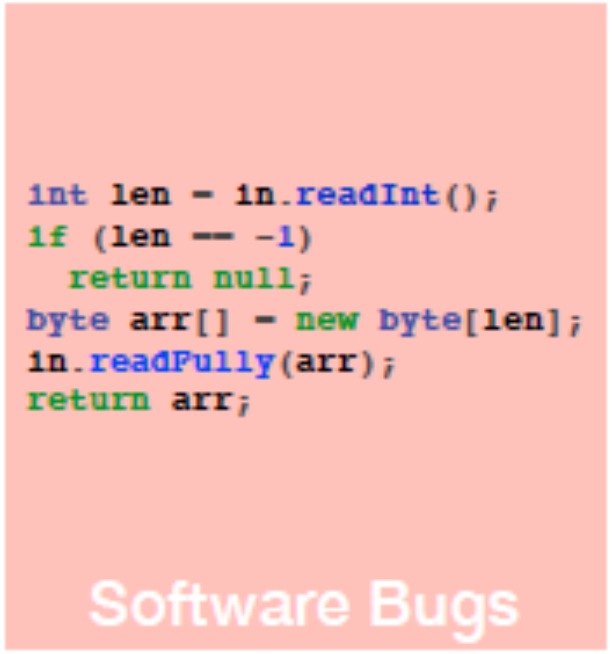
There are currently over 100 different implementations of Raft listed on their website. However, how to test these implementations?

Testing distributed systems is so HARD

Challenges

Faults are common in large systems and can happen anywhere at anytime!!!





```
int len = in.readInt();
if (len == -1)
  return null;
byte arr[] = new byte[len];
in.readFully(arr);
return arr;
```

Software Bugs

Challenges

Conventional testing techniques are not enough

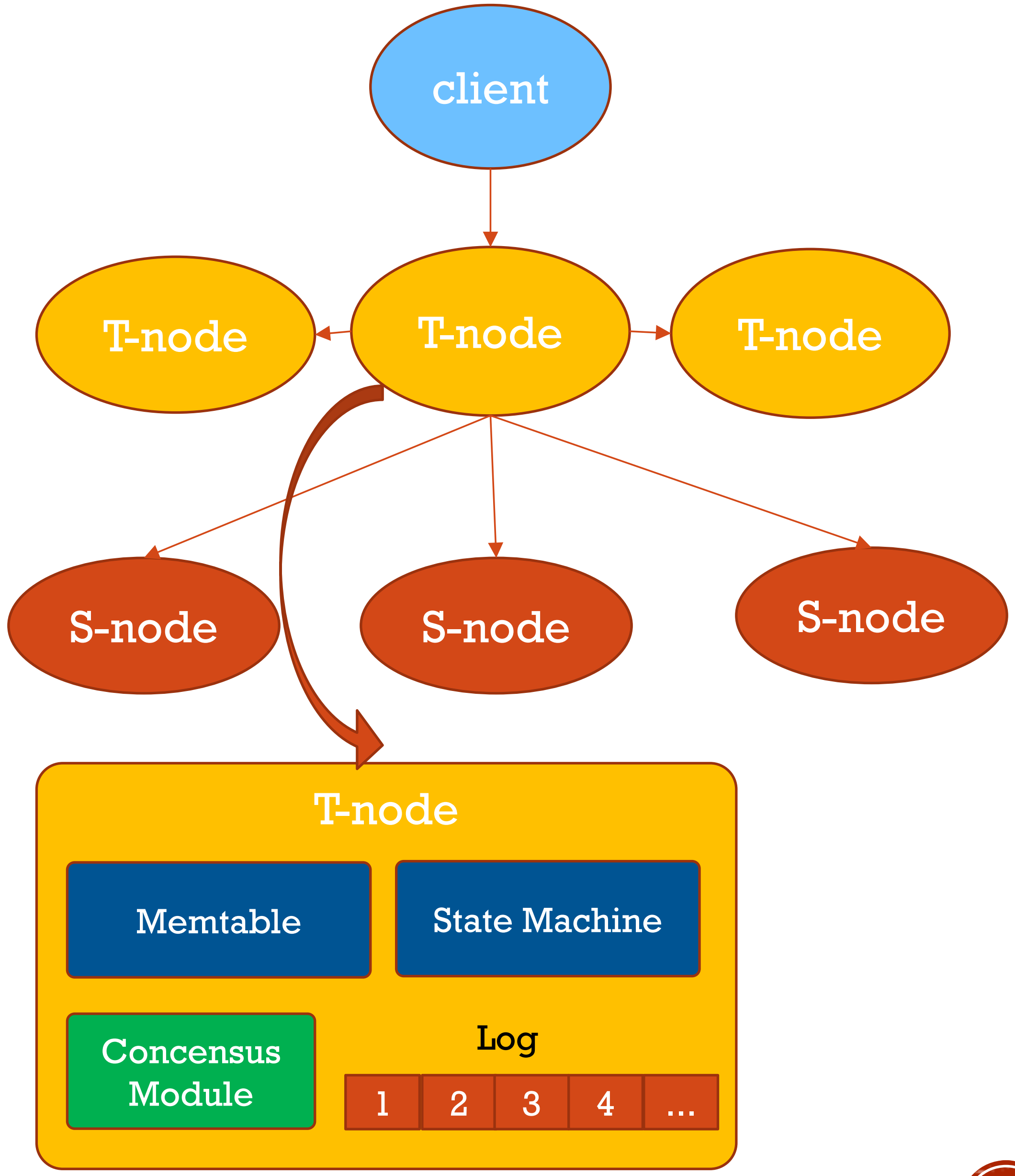
Unit test is not enough

Integration test is not enough

Performance test is not enough

Abstraction of the System Model

- T-node
  - In-memory transaction engine
  - Receive read/write request
  - Consist of four parts(state machine, consensus module, memory table (Memtable), log
- S-node
  - Distributed storage engine



Evaluation Metrics?

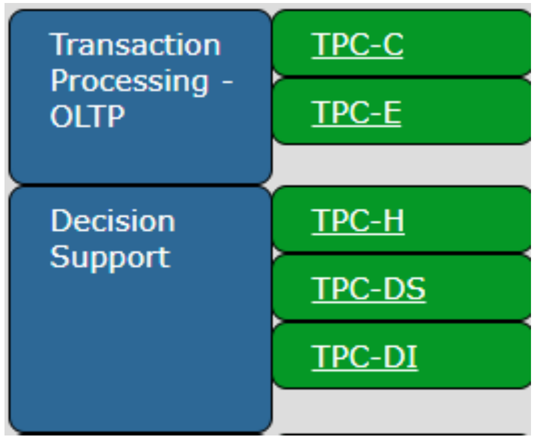
Test Dimension ?

How to test Raft-replicated database systems

Test Case?

Why are metrics important?

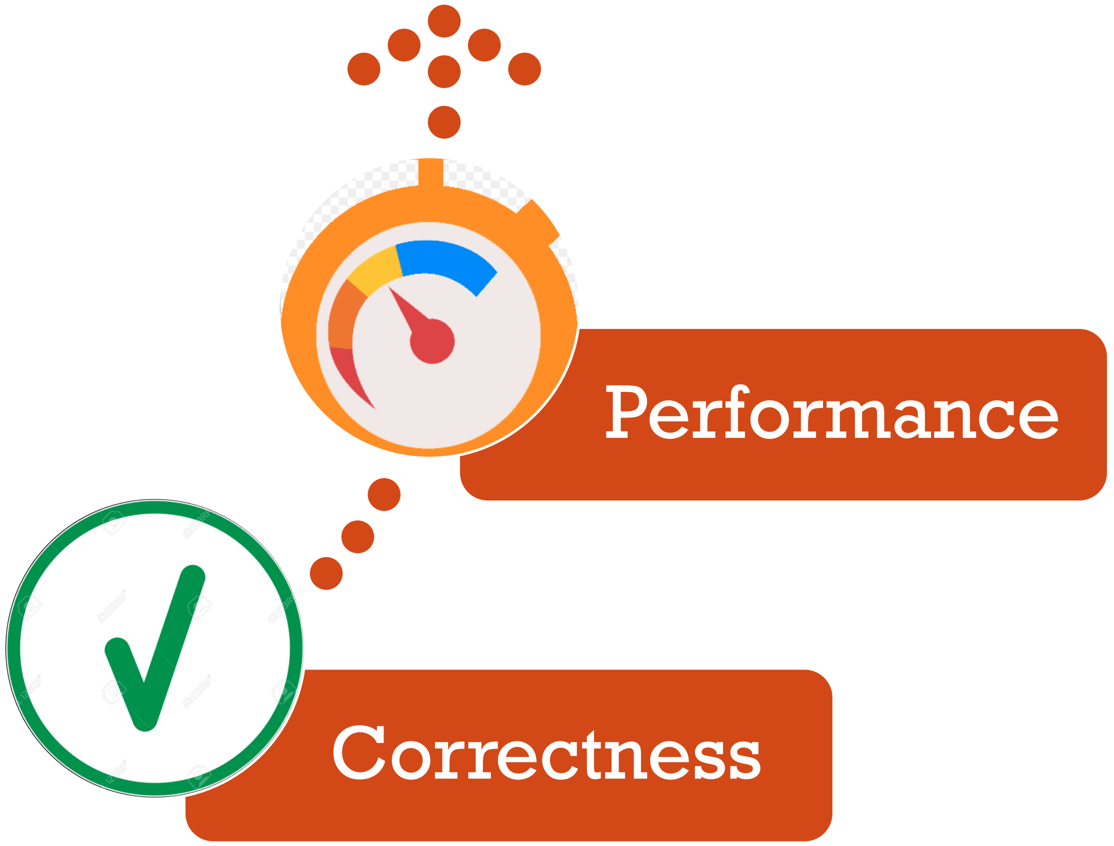
It is essential part of any test benchmark definition



May be the most controversial when trying to reach agreements between different vendors

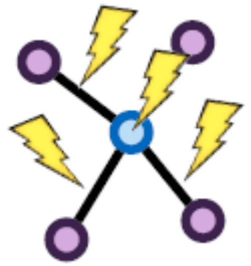
Desirable metrics ➔ Vendors and Users embrace it

What metrics do we care about?



Correctness

- Behave as expected(both under normal and fault conditions), consistent



- The most basic test for both centralized and distributed software systems, but it is often overlooked

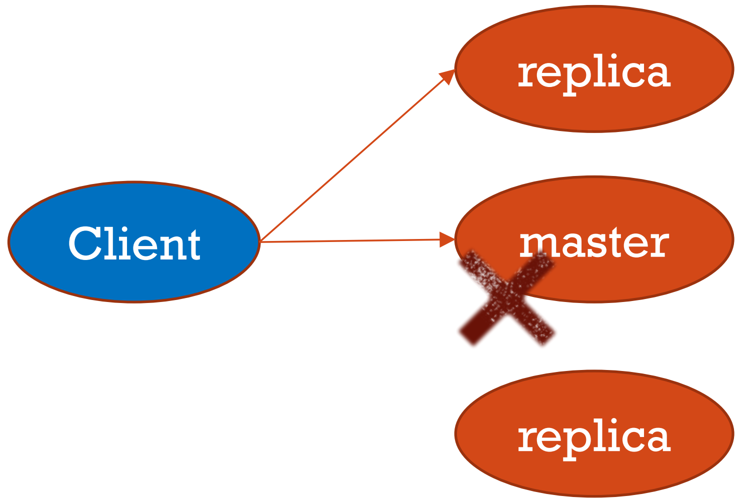
Availability

- Common failure : node downtime, network partition
- Detecting failure is crucial

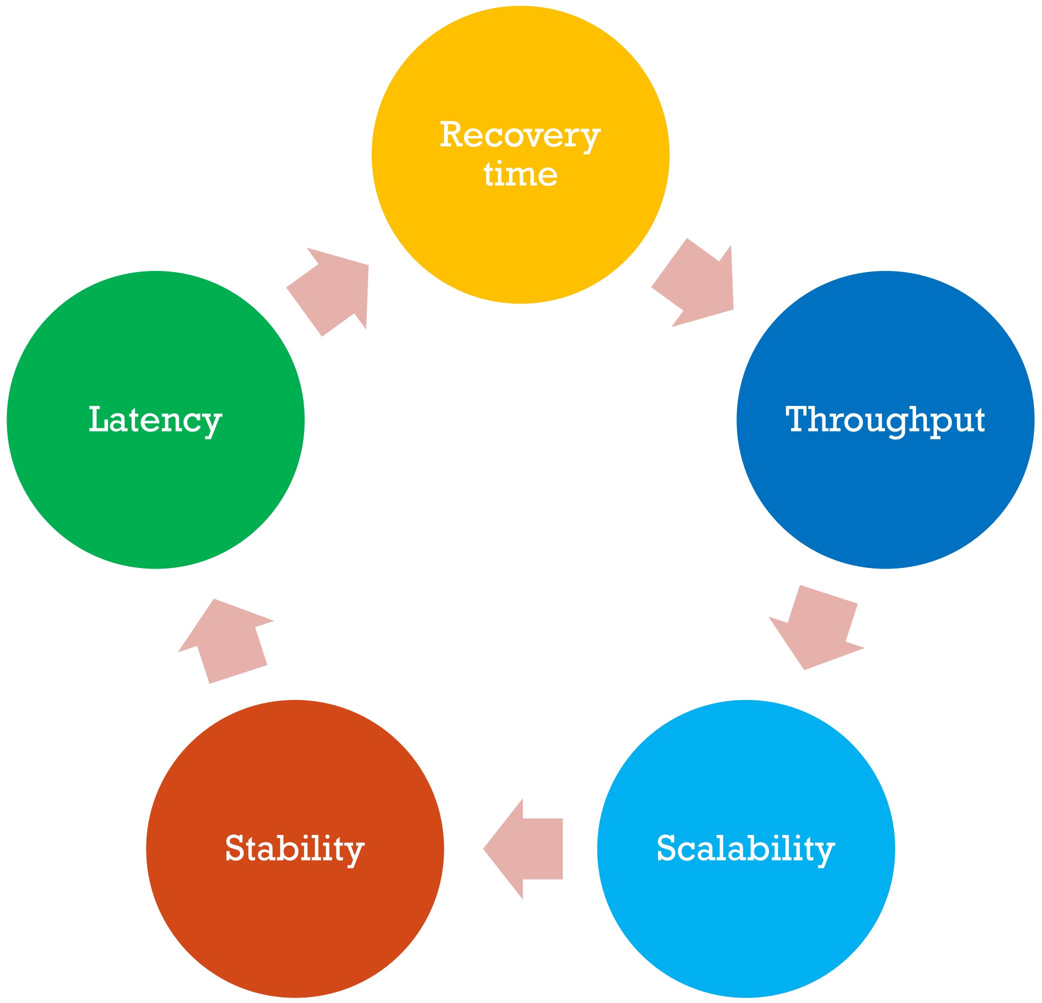
$$Availability\uparrow \approx \frac{\boxed{MTBF}\uparrow \text{ Improve reliability}}{MTBF + \boxed{MTTR}\downarrow \text{ Speed up recovery[1]}}$$

Data consistency

- The essence of Raft protocol is to guarantee the consistency between different data replicas



Performance



Stability

- Reliable: the ability of a system or component to perform its required functions under stated conditions for a specified period of time
- a stability metric model based on TPS fluctuations

$$\theta(TPS) = \sigma(TPS)/\overline{TPS} * 100\%$$

$\overline{TPS}$	: average number of transactions processed per second
$\sigma(TPS)$	: the standard deviation of TPS
$\theta(TPS)$	: the fluctuation range of TPS, acceptable value is 5% +/- 3%

Recovery time & Scalability

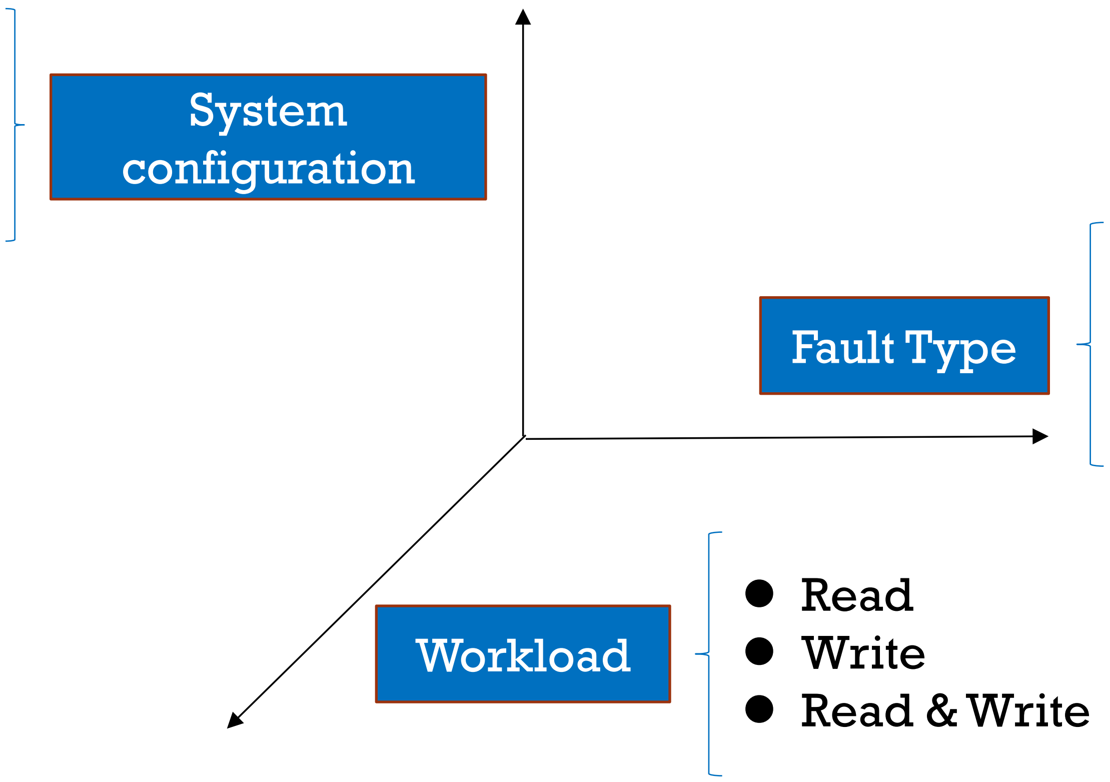
- Recovery Time
  - the time interval from the system can not provide external service to normal service when the system encounters a failure
- Scalability
  - the overall performance of the system is linear with the number of servers

Distributed Database System Dream

- Correct
  - consistent, behaves as expected

- Performant
  - scalable, low latency, high throughput, fault-tolerant, dependable, highly available

Test dimension

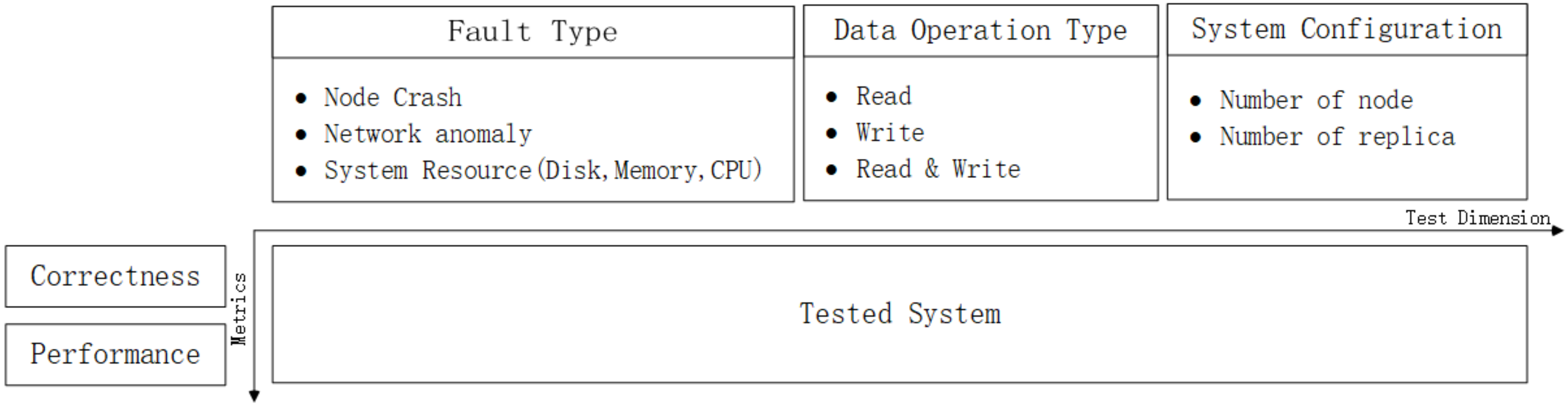


- Number of nodes
- Number of replica

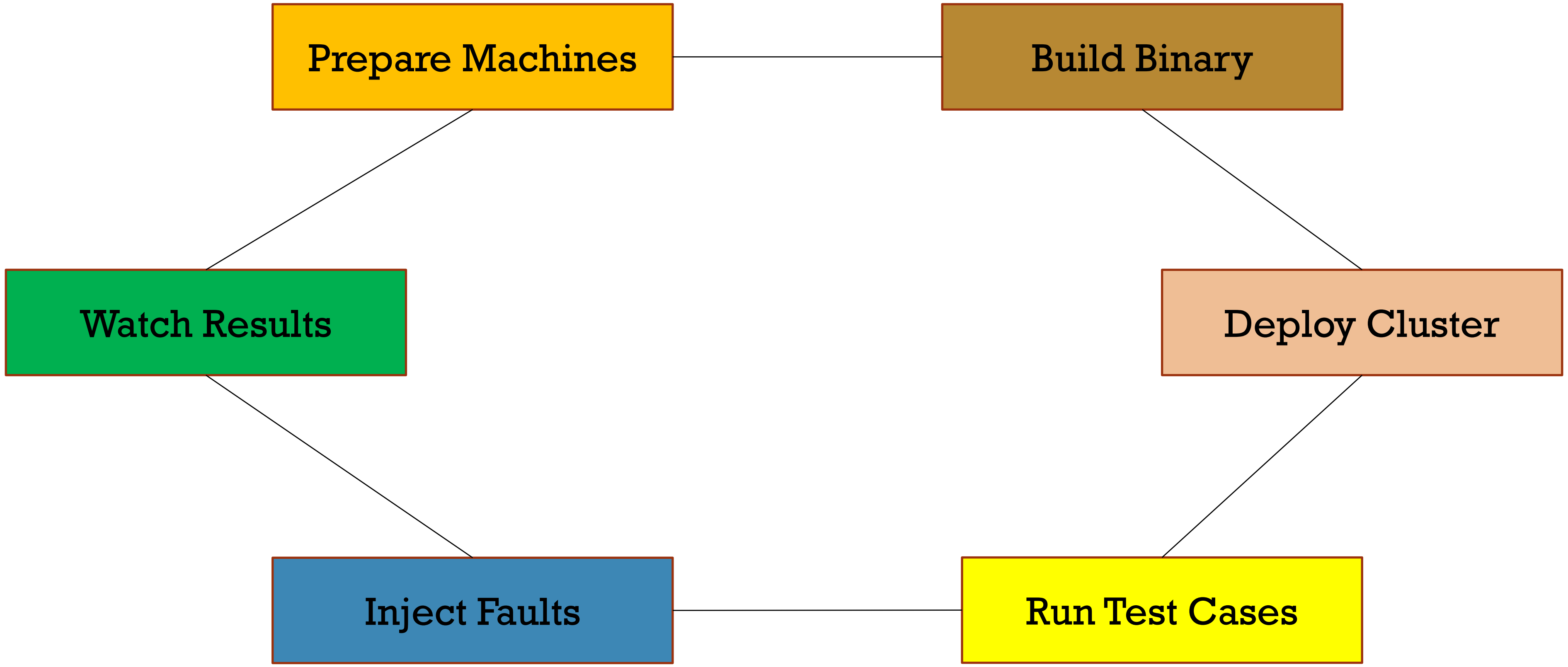
- Node Crash
- Network normality
- System Resources (Disk,Memory,CPU)

- Read
- Write
- Read & Write

Test case design

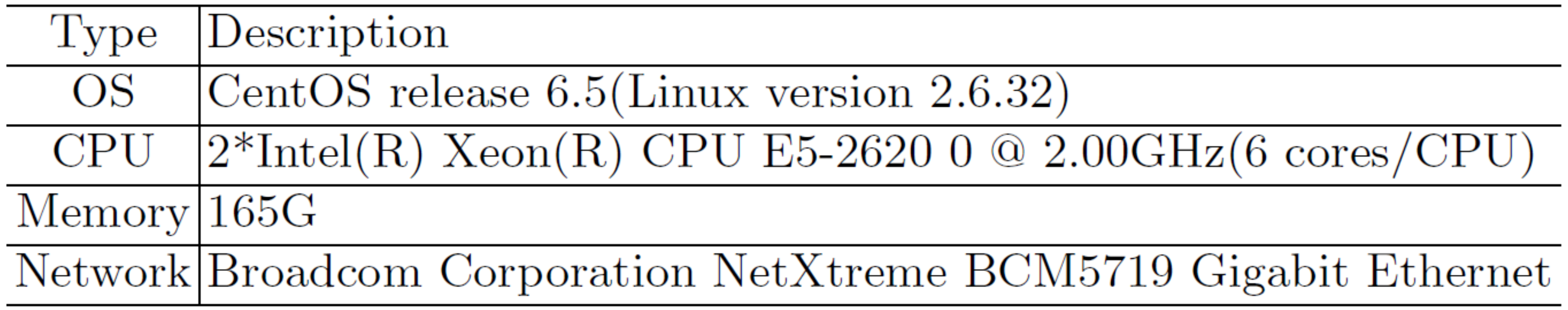


Run the test


Prepare Machines
Build Binary
Deploy Cluster
Run Test Cases
Inject Faults
Watch Results

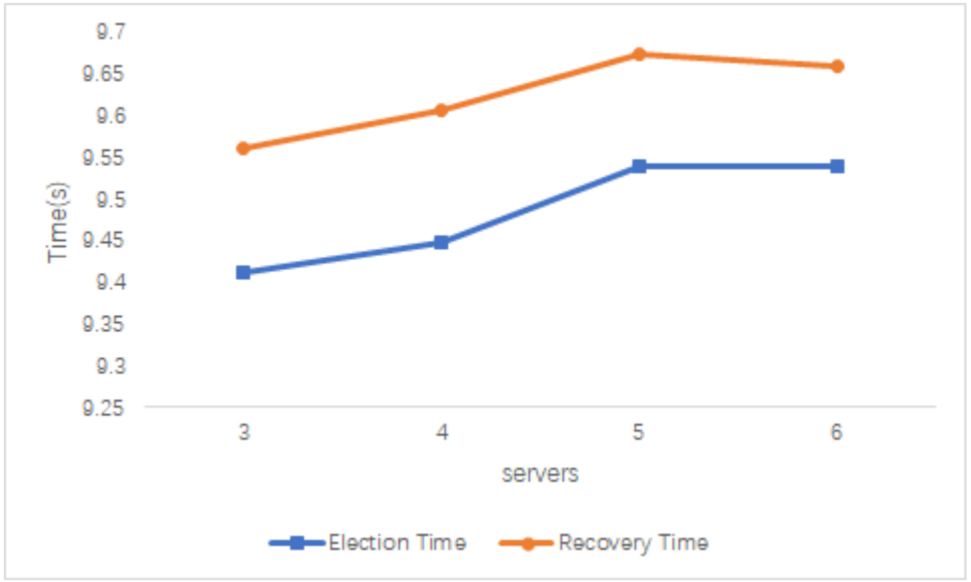
Testing results

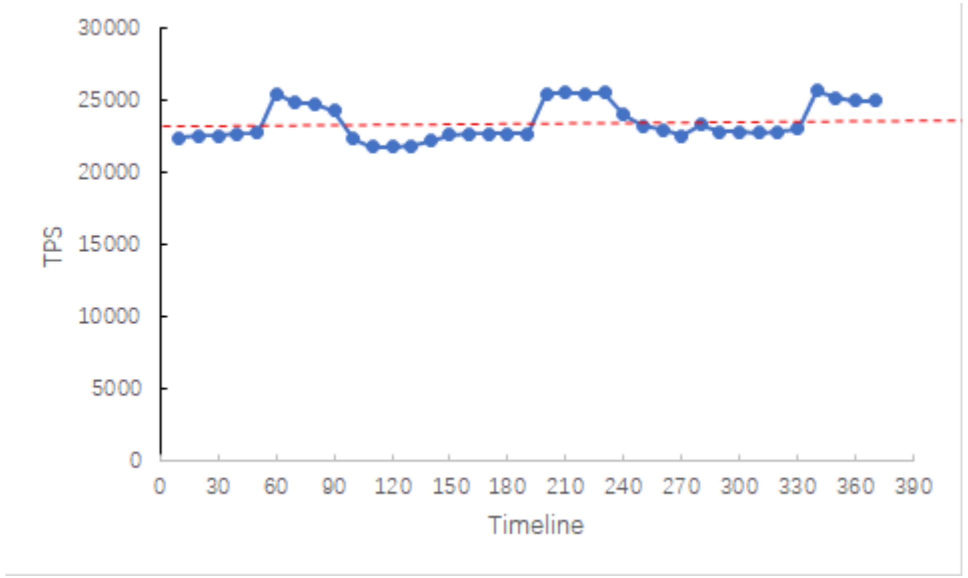
- Implement a variant of the Raft protocol on the distributed database CBase
- Experimental setups



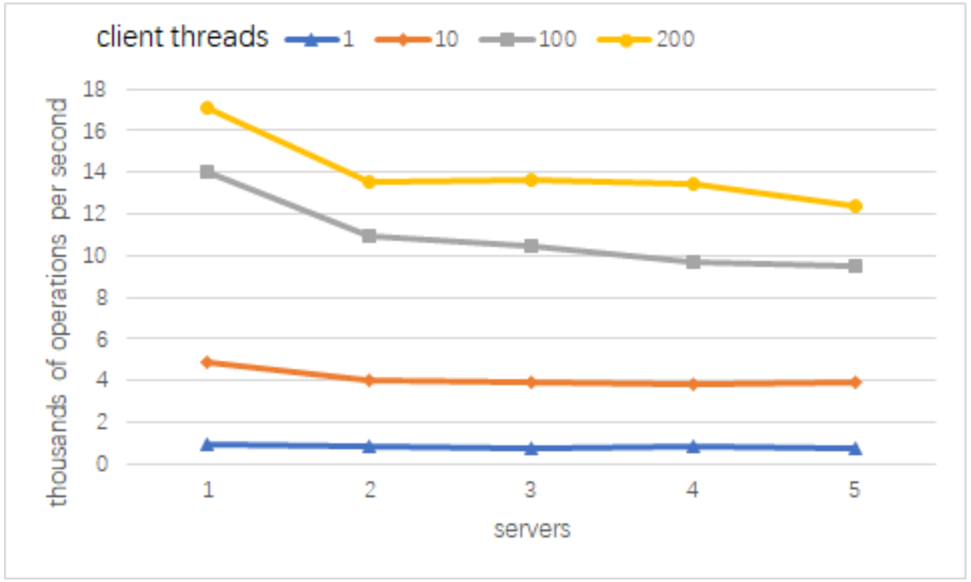
Type	Description
OS	CentOS release 6.5(Linux version 2.6.32)
CPU	2*Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz(6 cores/CPU)
Memory	165G
Network	Broadcom Corporation NetXtreme BCM5719 Gigabit Ethernet

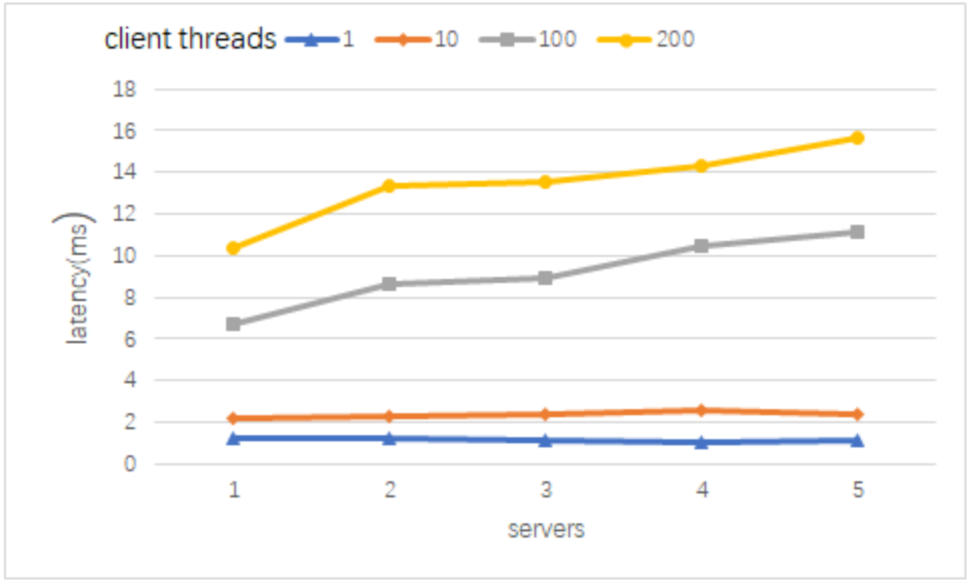
Recovery time & Stability





Throughput and latency

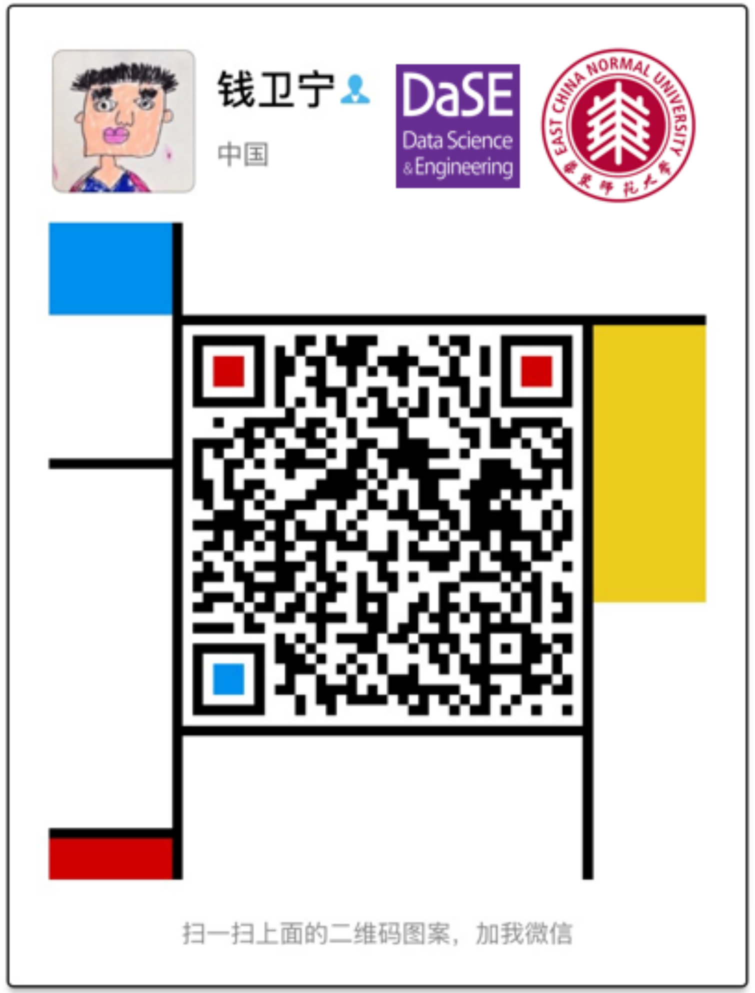




Conclusions and Future Work

- Abstraction of a system model
- Definition of test metrics and dimensions
- A set of (over 2000) testing cases and tools
- Give the test result on an open source distributed database system
- Future work
  - Build an automated testing framework, which can automatic system deployment, generation of test cases, and comparison of test results

Thanks !



钱卫宁

中国

扫一扫上面的二维码图案，加我微信