TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Systematic Testing of Distributed Systems

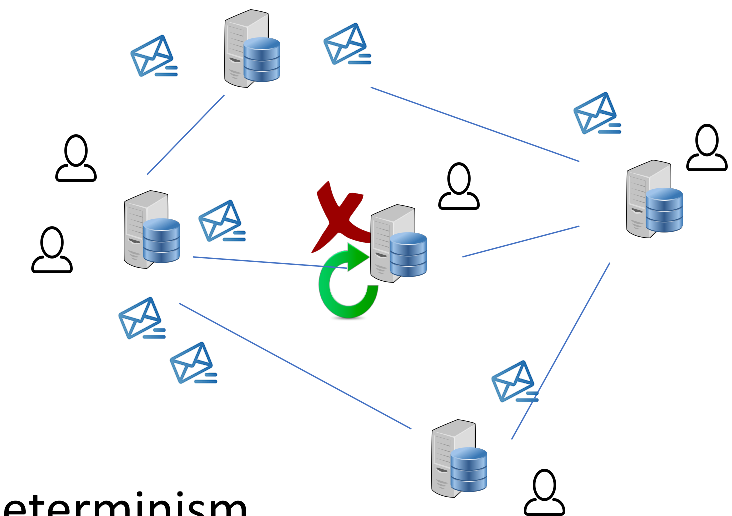
Burcu Kulahcioglu Ozkan

TU Kaiserslautern

Summer Term 2019

Distributed systems are prone to bugs!

- Distribution
- Asynchrony
- Replication
- ...



They are difficult to test!

- Many components, many sources of nondeterminism

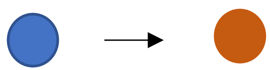
Cassandra / CASSANDRA-14702
Cassandra Write failed

HBase / HBASE-20368
Fix RIT stuck when a rsgroup has no online servers

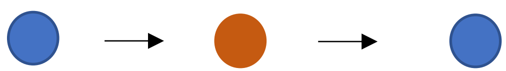
ZooKeeper / ZOOKEEPER-2930
Leader cannot be elected due to network timeout

Distributed systems bugs are deep!

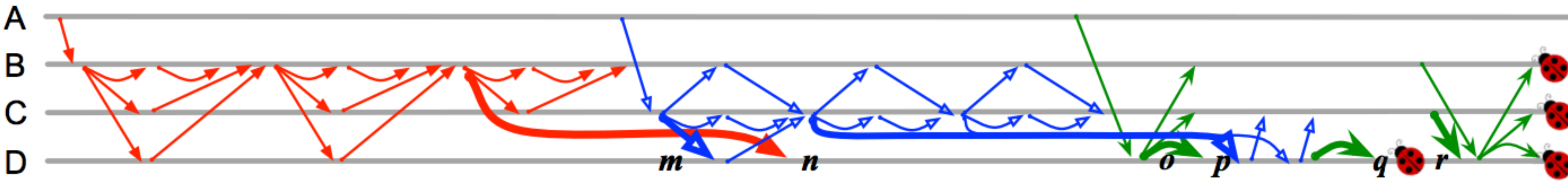
- $d = 2 \;\; \langle e_1, e_2 \rangle$ e.g. order violation



- $d = 3 \;\; \langle e_1, e_2, e_3 \rangle$ e.g. atomicity violation

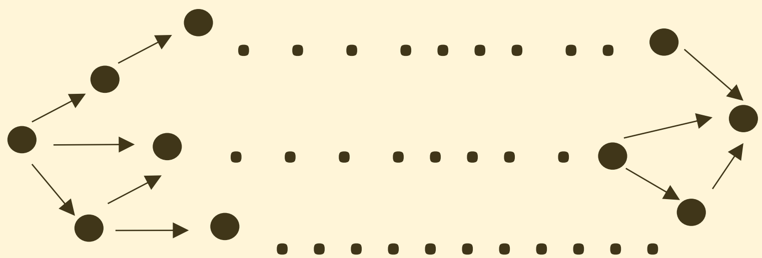


- $d = n \;\; \langle e_1, \ldots, e_n \rangle$ more complicated bugs



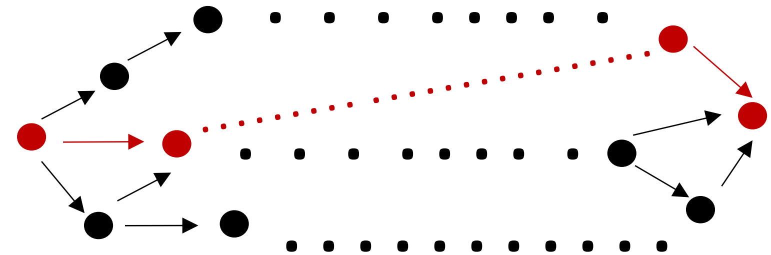
Bug in Cassandra 2.0.0 *(img. from Leesatapornwongsa et. al. ASPLOS'16)*

How to detect bugs?

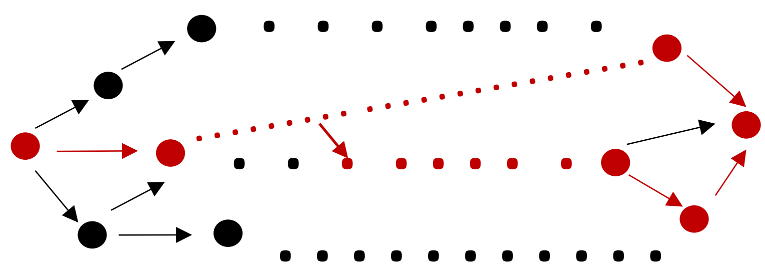


Systematic testing - infeasible

Needs reduction techniques (e.g. SAMC, FlyMC)

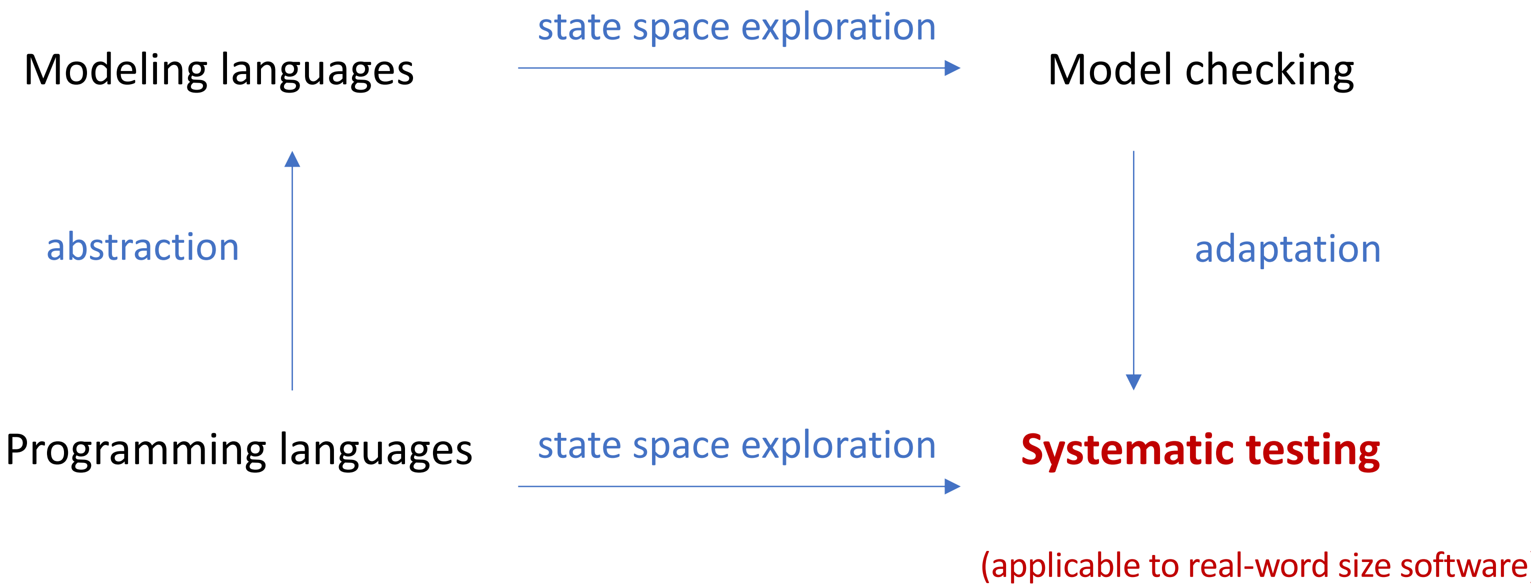


Random testing (e.g. PCTCP, Jepsen)



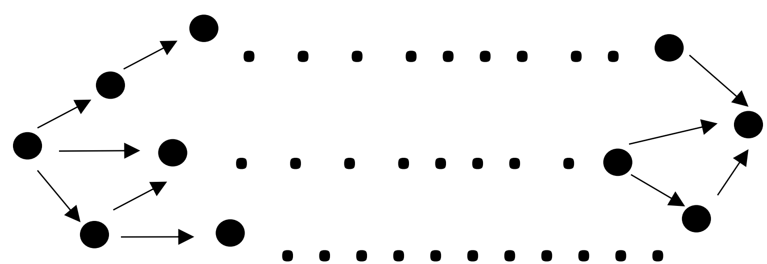
Guided testing (e.g. Molly)

Combining Model Checking and Testing



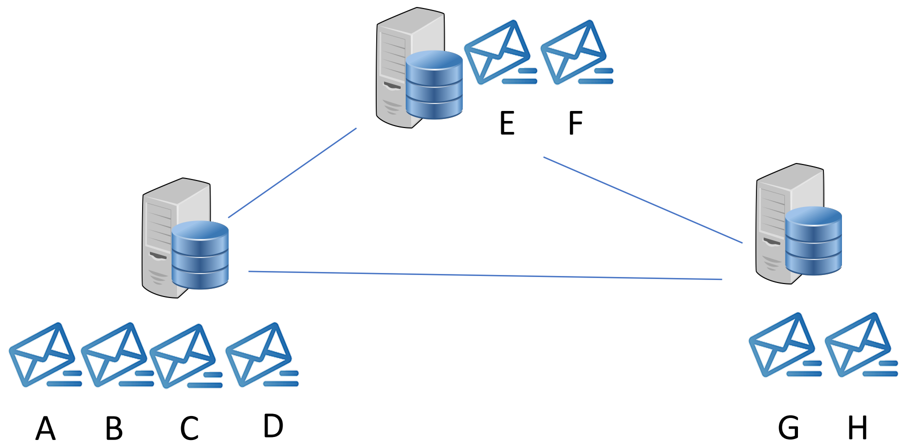
Systematic Testing of Distributed Systems

- Explore the state space systematically
  - Run time scheduler to exercise all possible sequences of events
  - Ability to inject crash/reboot events
- Infeasible to test all executions
  - State space explosion problem



A Simple Example

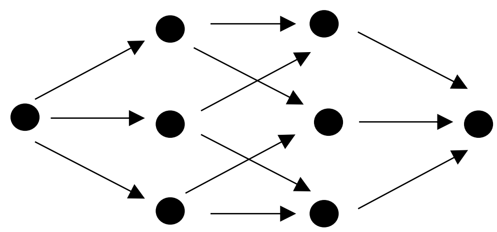
- How many different executions does the system have?

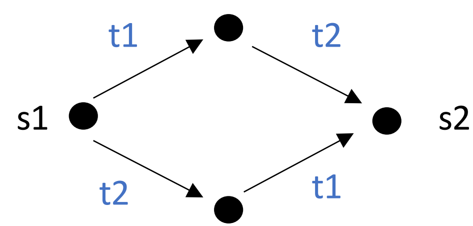


- Each node operates on its own local state
- The messages to different nodes are commutative

Partial Order Reduction

- Avoids redundantly exploring parts of the state space reachable by different executions
- Exploits the commutativity of concurrent transitions
- Based on the dependency relation between the transitions of a system





- Dynamic Partial Order Reduction (DPOR) dynamically tracks interactions between transactions

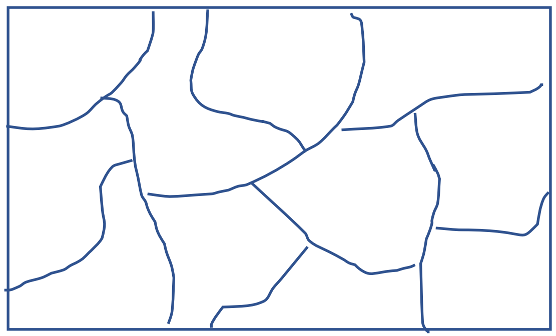
Partial Order Reduction for Distributed Systems

Based on the dependency relation between the events:

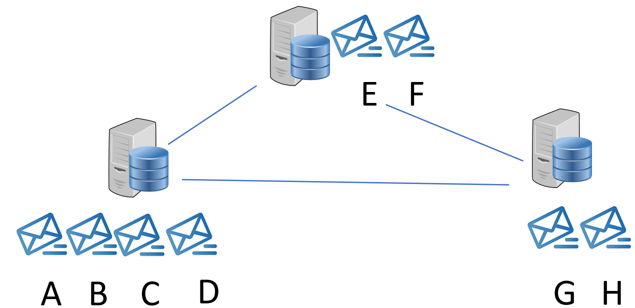
- A distributed system event: $e = \langle receiver, sender, message \rangle$
- An execution: $E = e_1, e_2, \ldots, e_n$
- Dependence relation: $(e_1, e_2) \in D$iff $e_1.\text{receiver} = e_2.receiver$

- Two executions E_1 and E_2 are equivalent iff:
  - $Set(E_1) = Set(E_2)$
  - For every $(e_1, e_2) \in D$: $e_1 \overset{E_1}{\rightarrow} e_2$ iff $e_1 \overset{E_2}{\rightarrow} e_2$

Partial Order Reduction for Distributed Systems



D partitions the state space
into equivalence classes w.r.t $\equiv_D$



A B C D E F G H $\equiv_D$ A B C E F G H D

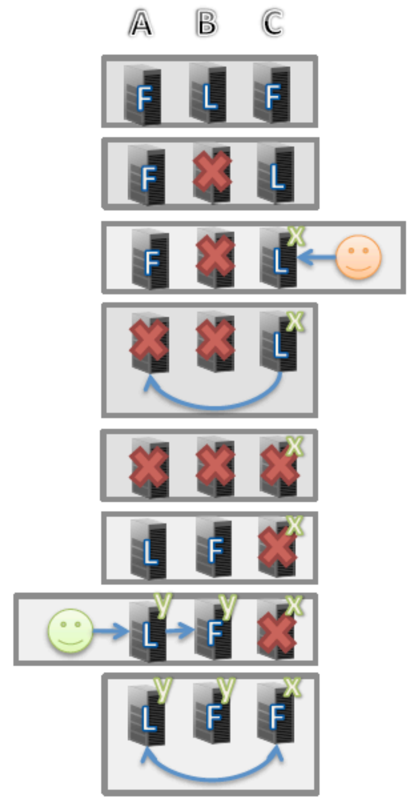
A B C D E F G H $\not\equiv_D$ B A C D E F G H

A Complex Example

ZooKeeper (synchronization service)
Issue #335.

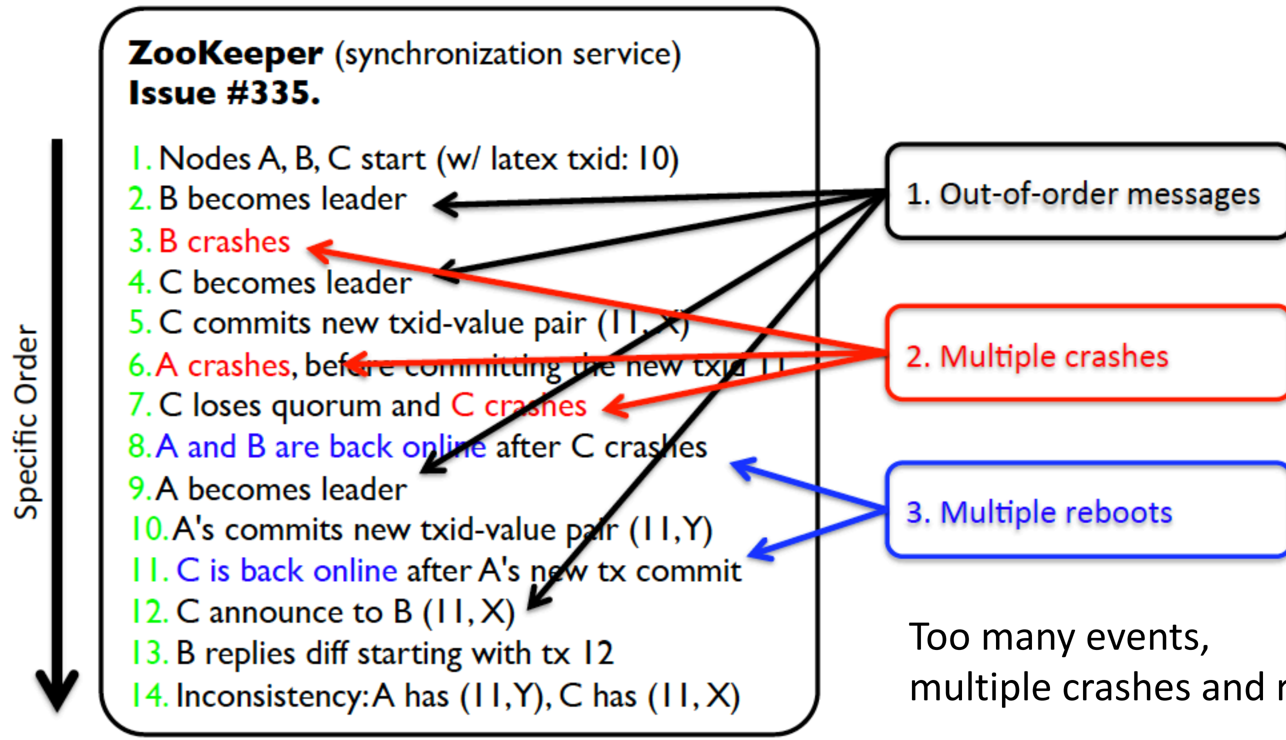
1. Nodes A, B, C start (w/ latex txid: 10)
2. B becomes leader
3. B crashes
4. C becomes leader
5. C commits new txid-value pair (11, X)
6. A crashes, before committing the new txid 11
7. C loses quorum and C crashes
8. A and B are back online after C crashes
9. A becomes leader
10. A's commits new txid-value pair (11, Y)
11. C is back online after A's new tx commit
12. C announce to B (11, X)
13. B replies diff starting with tx 12
14. Inconsistency: A has (11, Y), C has (11, X)

PERMANENT INCONSISTENT REPLICA



From “SAMC: Semantic-Aware Model Checking for Fast Discovery of Deep Bugs in Cloud Systems OSDI’14”

A Complex Example



Too many events,
multiple crashes and reboots!

From "SAMC: Semantic-Aware Model Checking for Fast Discovery of Deep Bugs in Cloud Systems OSDI'14"

SAMC-Semantic Aware Model Checking[1]

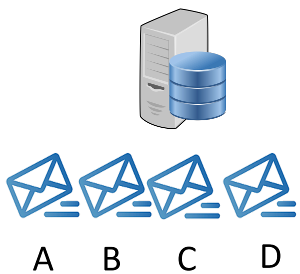
Existing approaches for reduction is not sufficient

- Classical DPOR
  - Black box, exploits general properties of distributed systems
- SAMC
  - White-box, exploits system specific semantic information
- Use system semantics for state space reduction
  - Local Message Independence
  - Crash Message Independence
  - Crash Recovery Symmetry
  - Reboot Synchronization Symmetry

[1] SAMC: semantic-aware model checking for fast discovery of deep bugs in cloud systems, OSDI’14

Local Message Independence

- Some messages sent to a node are concurrent



Black box DPOR

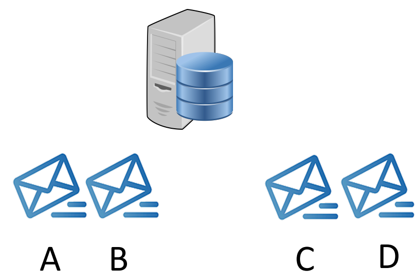
ABCD

ABDC

ACBD

...

4! reorderings



White box DPOR (with message processing semantics)

ABCD

ABDC

BACD

BADC

4 reorderings

Local Message Independence

Discard:

```
if(pd(m, ls))
    noop;
```

Increment:

```
if(pi(m, ls))
    ls ++;
```

Constant:

```
if(pc(m, ls))
    ls = Const;
```

Modify:

```
if(pm(m, ls))
    ls = modify(m, ls)
```

- `m1` is independent of `m2` if `pd` is true for any of `m1` and `m2`
- `m1` is independent of `m2` if `pi` (or `pc`) is true on both `m1` and `m2`
- `m1` and `m2` are dependent if `pm` is true on `m1` and `pd` is not true on `m2` (they modify the state in unique ways)

Crash Message Independence

- Some messages and node crashes are concurrent

Global impact:

```
if(pg(X, ls))
     modify(ls);
     sendMsg();
```

Local impact:

```
if(pg(X, ls))
     modify(ls);
```

- E.g. Crash of a node N is concurrent with messages A, B, C, D

Black box DPOR	White box DPOR
ABCDX	ABCDX
ABCXD	
ABXCD	
AXBCD	
XABCD	

Crash Recovery Symmetry

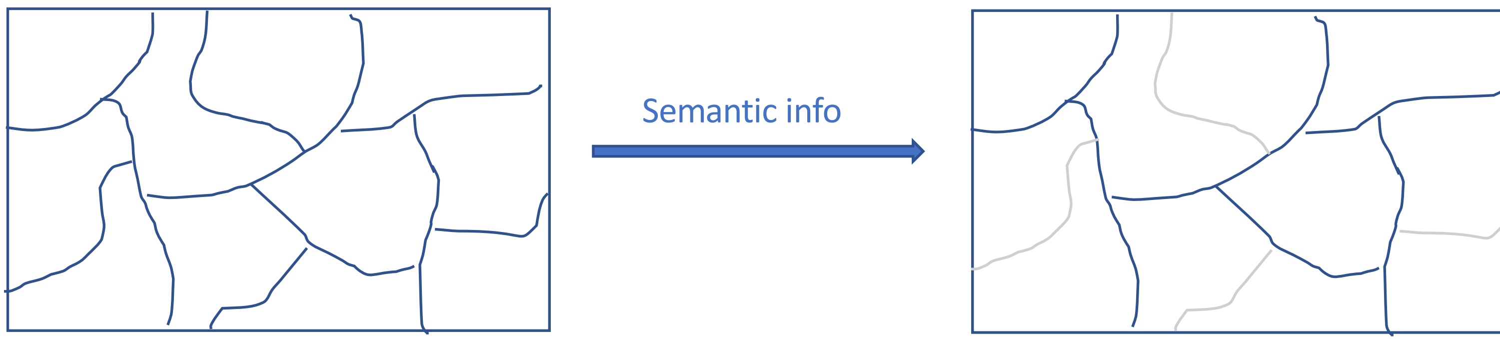
- Guide the model checker with the crash decisions
- Some crashes lead to symmetrical recovery behaviors
  - In a 4-node system with FFFL, crashing the first and the second node may lead to the same behavior
  - Two recovery actions are symmetrical if they produce the same message and update the local state in the same way
- Needs to extract recovery logic

Reboot Synchronization Symmetry

- Guide the model checker with the reboot decisions
- A reboot will not lead to a new scenario if the current state of the system is similar to the state it crashed
- Needs to extract reboot synchronization predicates and corresponding actions

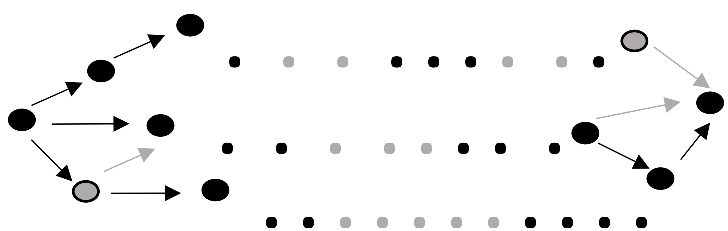
Partial Order Reduction for Distributed Systems

Semantic information provides coarser equivalence of executions:



Equivalence w.r.t black box D

Equivalence w.r.t white box D



Systematic testing with pruning

Summary

- Systematic testing suffers from state space explosion problem
- Partial order reduction techniques reduce the state space
  - Generic notion of dependency – black box
  - Semantic knowledge for fine grained dependency – white box
  - Used for testing on Cassandra, Zookeeper, Hadoop
    - Reduction ratio between 37x to 166x in model checking Zookeeper

- Research Questions:
  - What other semantic knowledge can scale MC distributed systems?
  - How to extract the system specific white-box information?
  - What other techniques can be used for an efficient systematic testing?