FCatch: Automatically Detecting Time-of-fault Bugs in Cloud Systems

Haopeng Liu, Xu Wang*, Guangpu Li, Shan Lu, Feng Ye†, and Chen Tian†

http://fcatch.cs.uchicago.edu/

THE UNIVERSITY OF CHICAGO

*

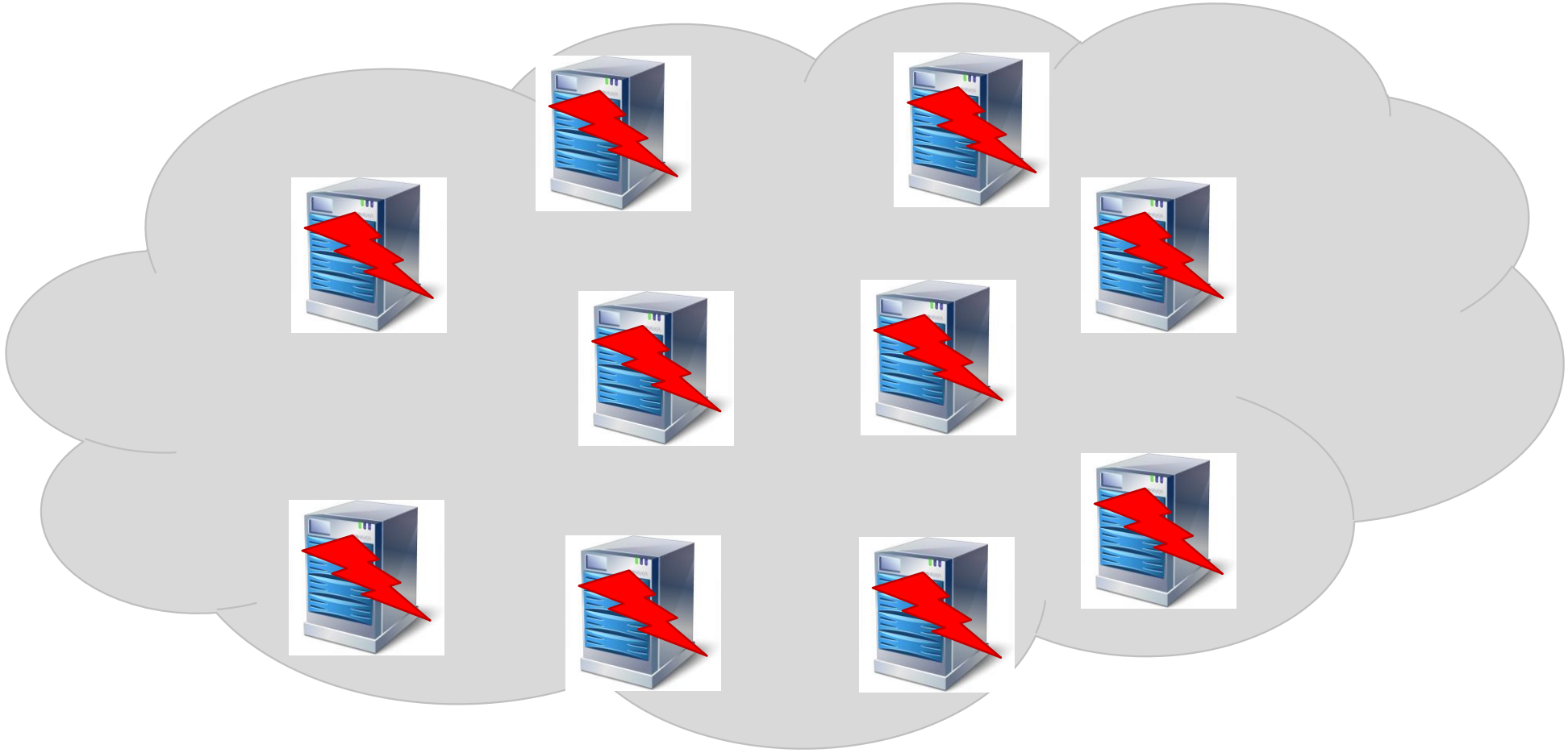
†

HUAWEI

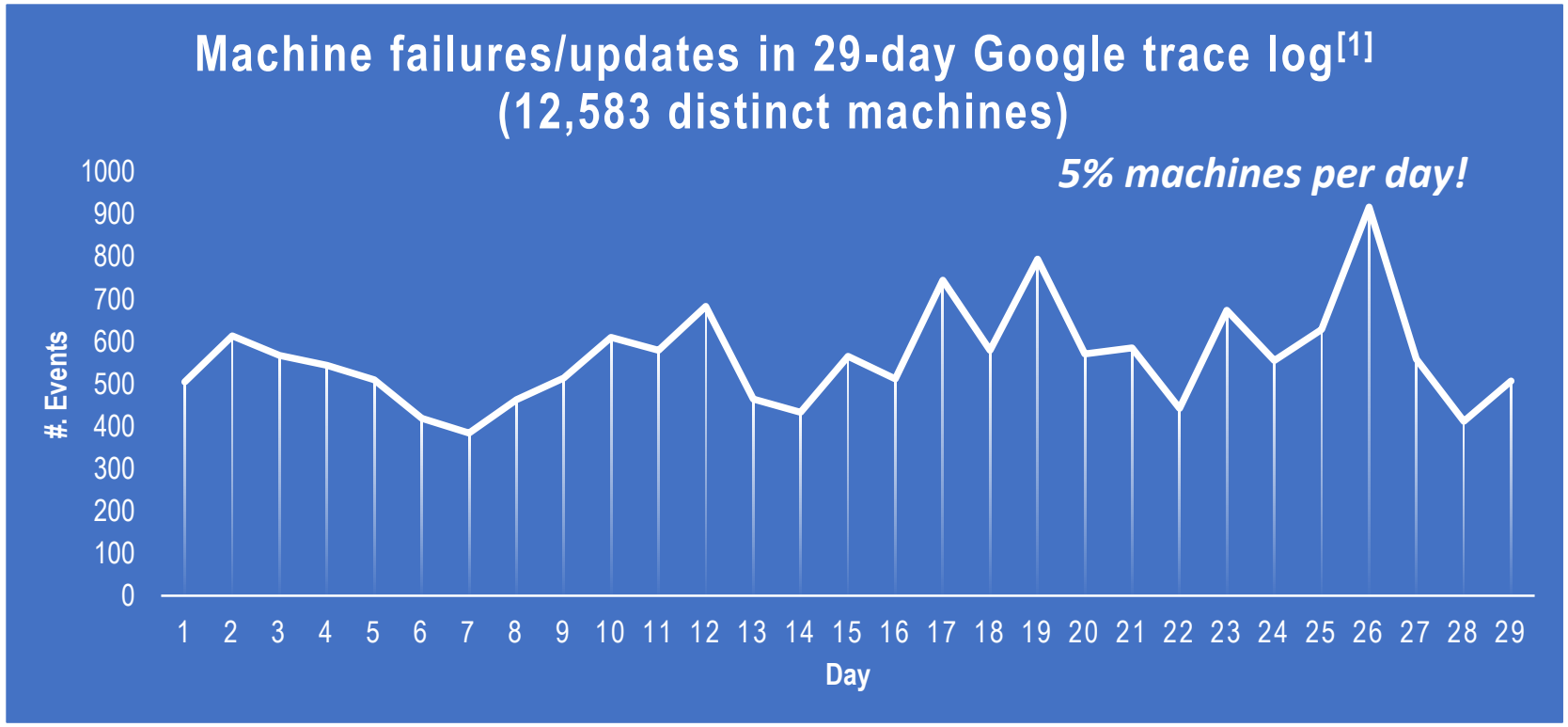
Cloud Systems

Fault

- Component failure: node crashes and message drops



Faults are common



[1] Mesbahi and etc. Cloud Dependability Analysis: Characterizing Google Cluster Infrastructure Reliability. In ICWR-2017

Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will timeout

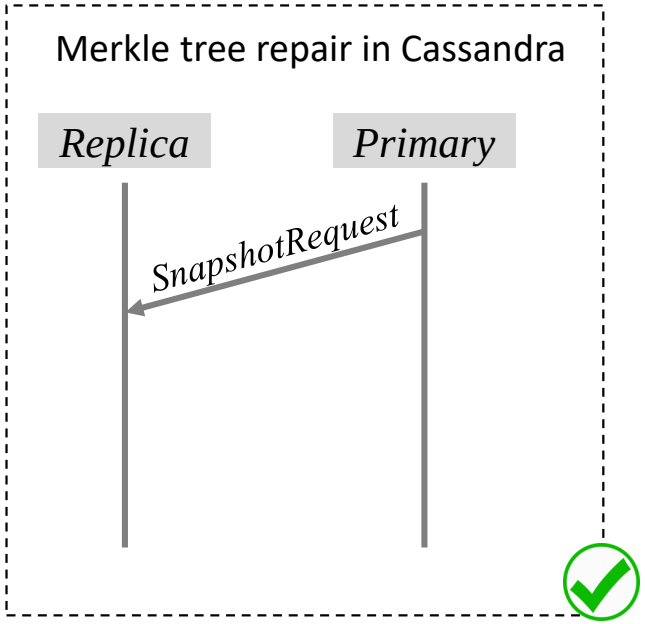
Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will timeout

Merkle tree repair in Cassandra

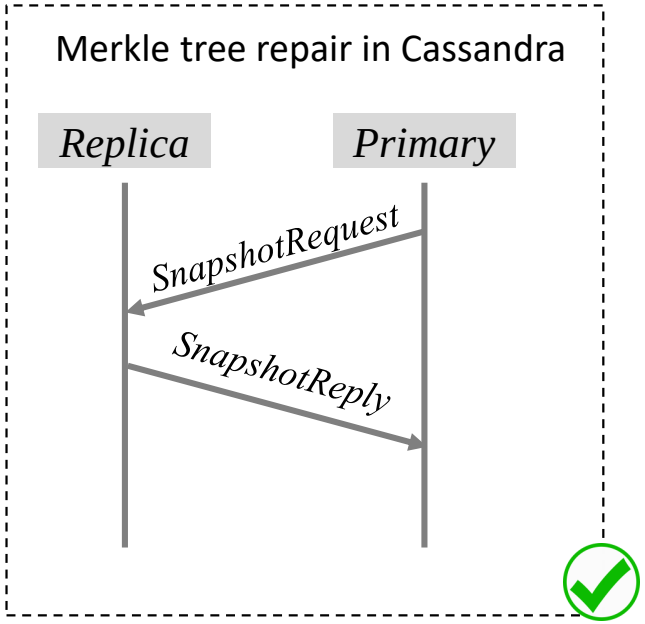
Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will timeout



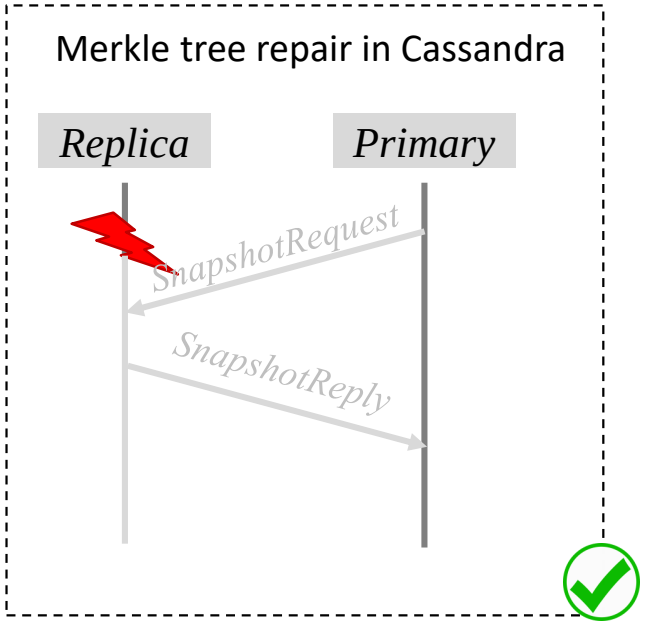
Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will timeout



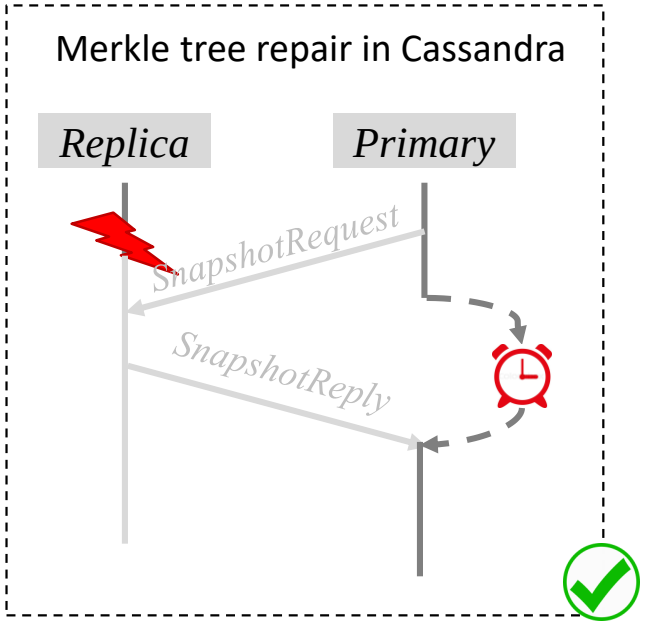
Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will timeout



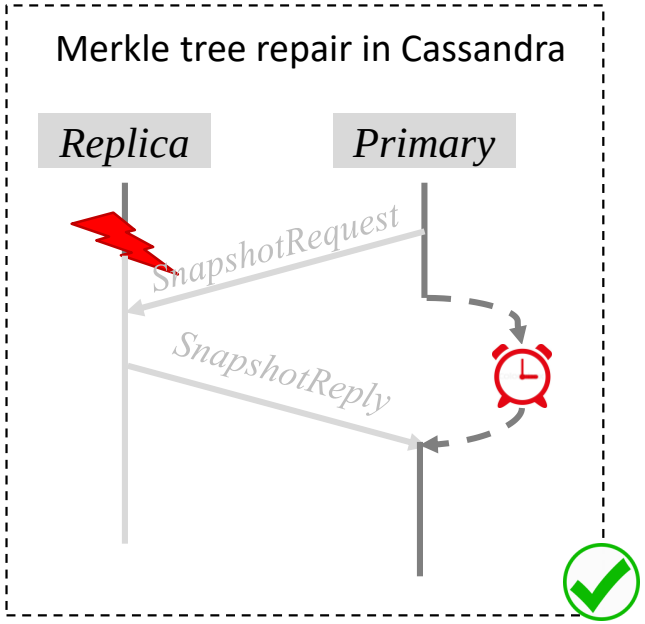
Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will timeout



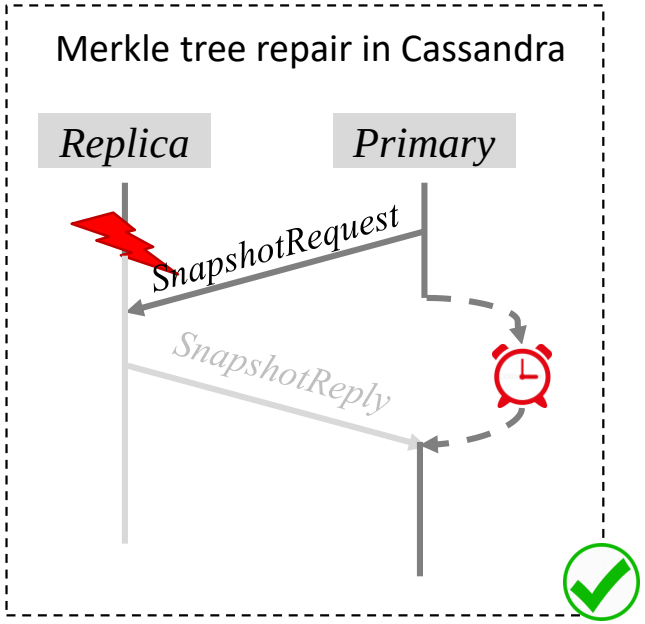
Cloud systems are fault tolerant?

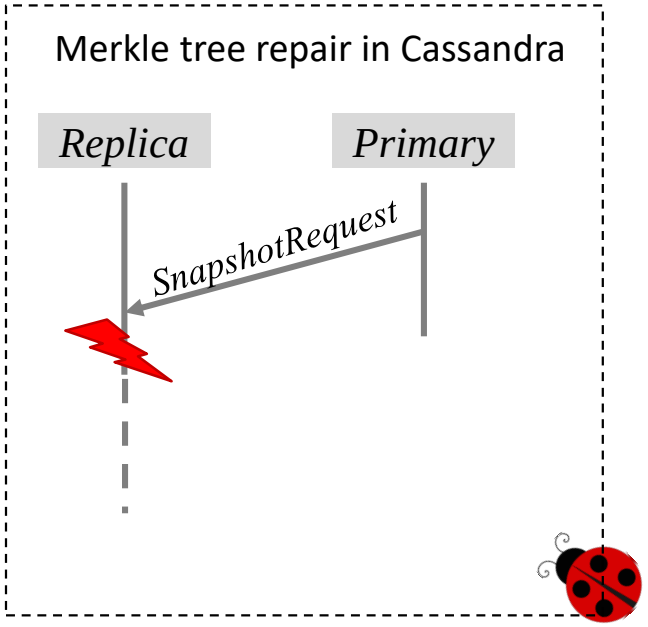
- When a fault happens at node *A* at a special moment
  - Communication with A will hang without timeout



Cloud systems are fault tolerant?

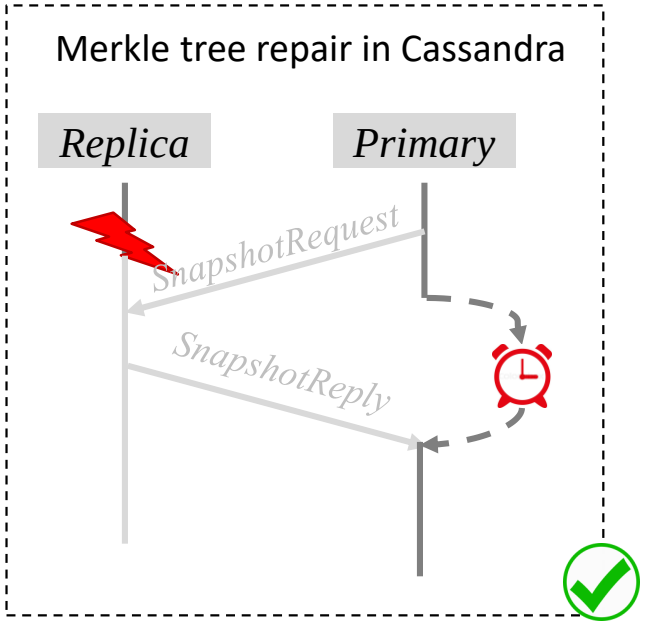
- When a fault happens at node *A* at a special moment
  - Communication with A will hang without timeout

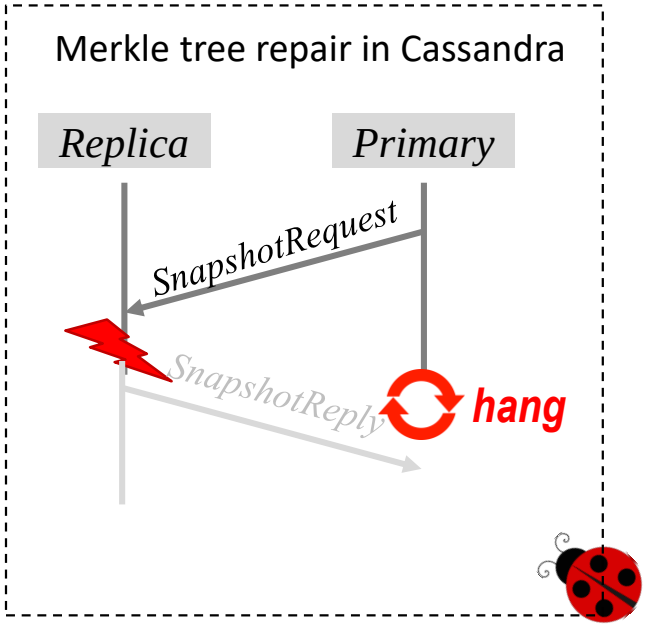




Cloud systems are fault tolerant?

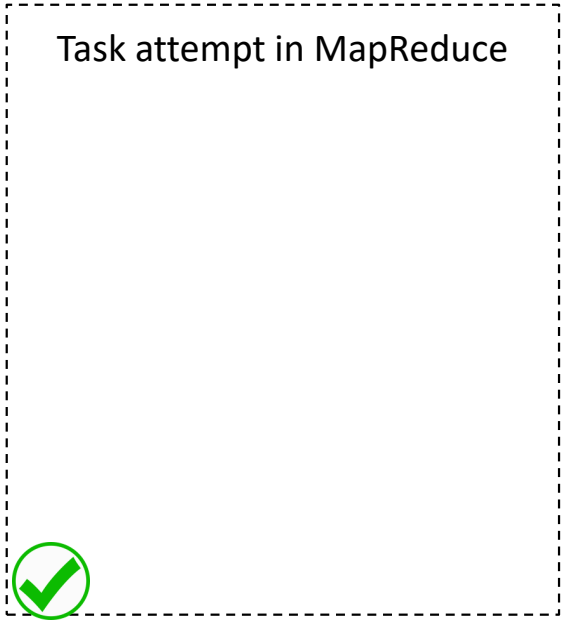
- When a fault happens at node *A* at a special moment
  - Communication with A will hang without timeout





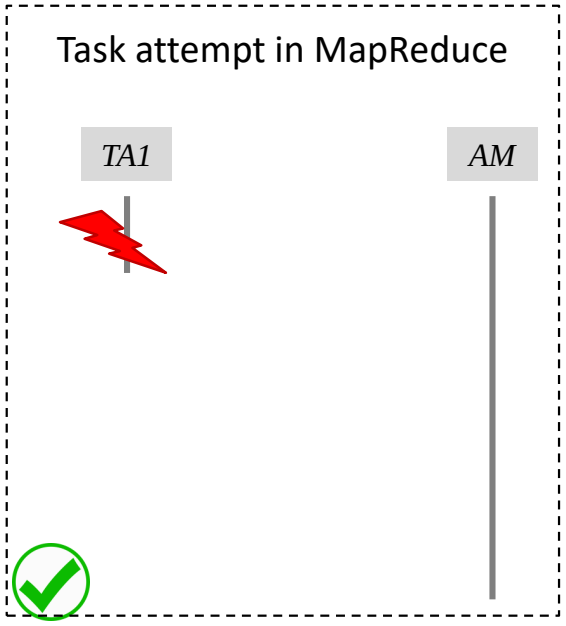
Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will hang without timeout
  - A restarted node will take over *A*'s task



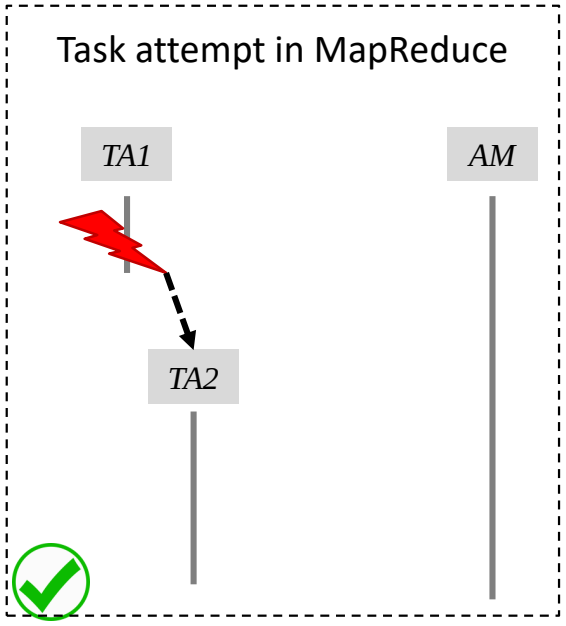
Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will hang without timeout
  - A restarted node will take over *A*'s task



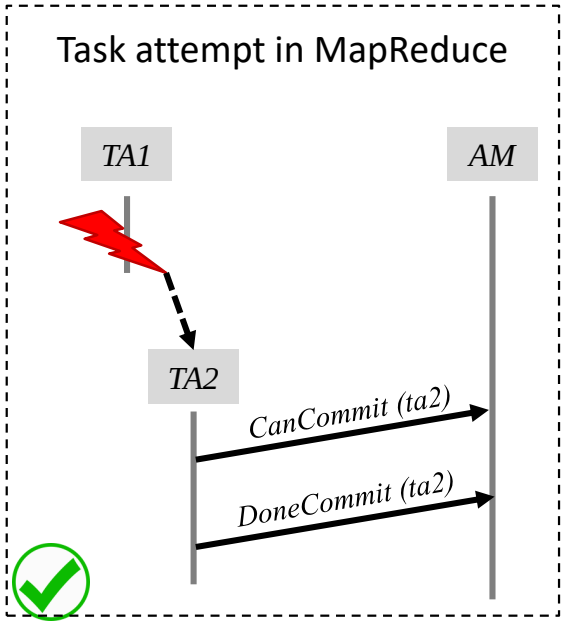
Cloud systems are fault tolerant?

- When a fault happens at node *A*
  - Communication with A will hang without timeout
  - A restarted node will take over *A*'s task



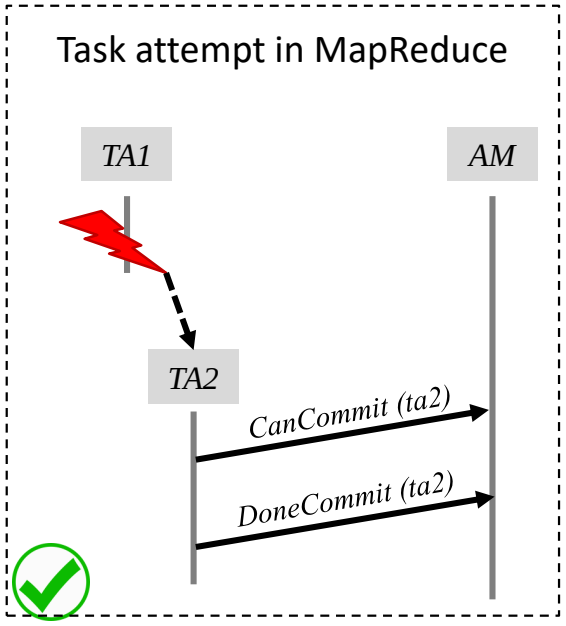
Cloud systems are fault tolerant?

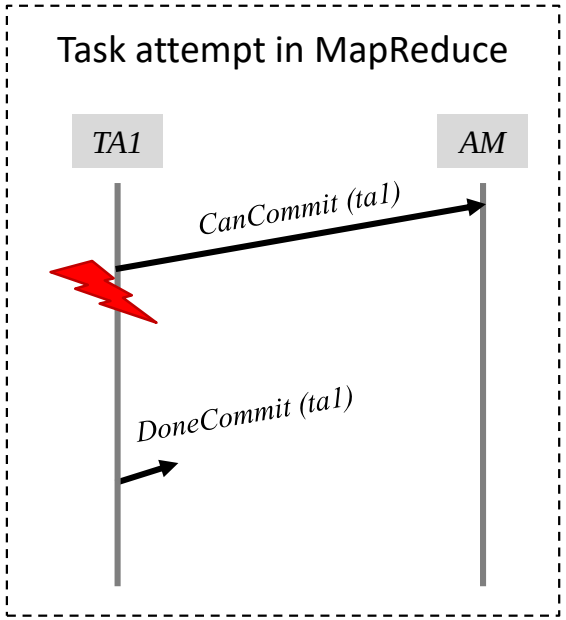
- When a fault happens at node *A*
  - Communication with A will hang without timeout
  - A restarted node will take over *A*'s task



Cloud systems are fault tolerant?

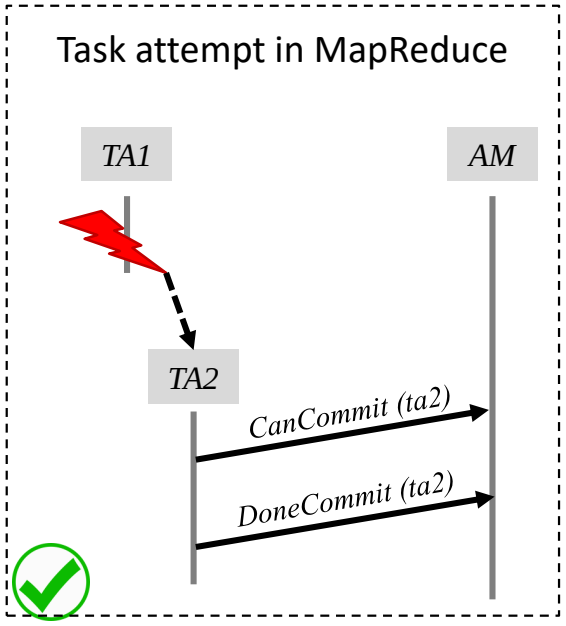
- When a fault happens at node *A* at a special moment
  - Communication with A will hang without timeout
  - A restarted node will fail to take over *A*'s task

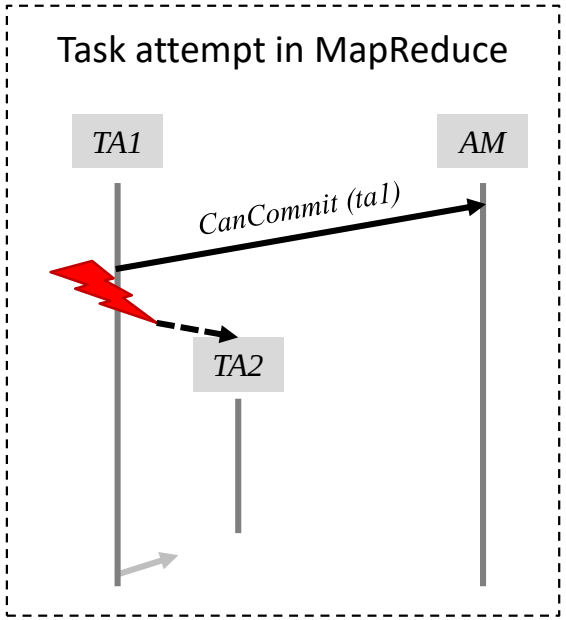




Cloud systems are fault tolerant?

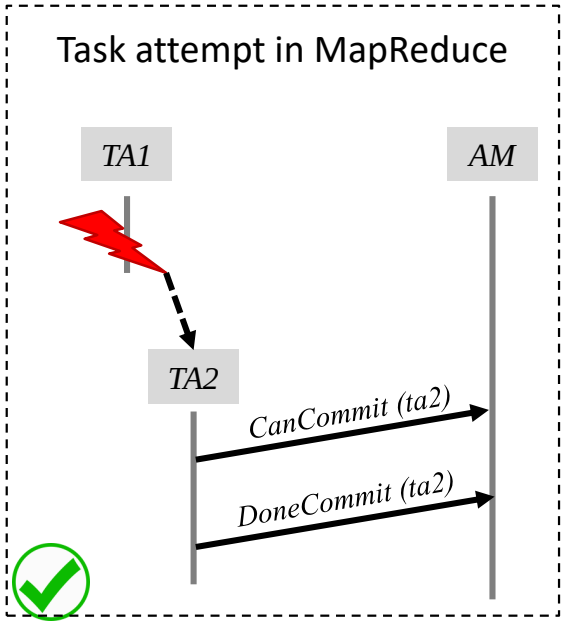
- When a fault happens at node *A* at a special moment
  - Communication with A will hang without timeout
  - A restarted node will fail to take over *A*'s task

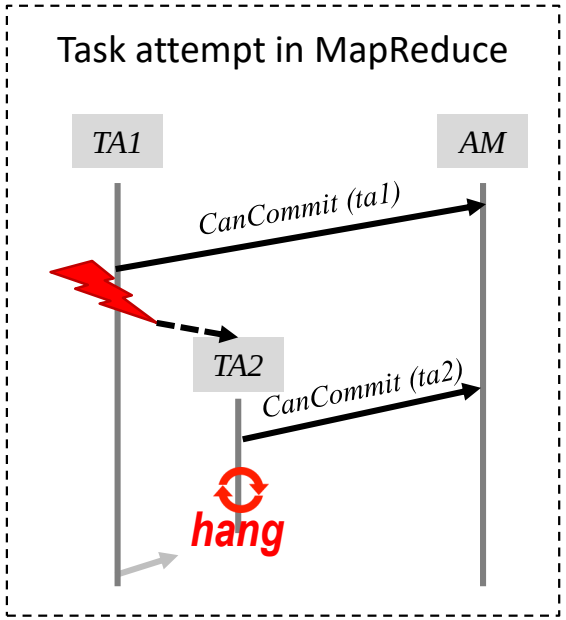




Cloud systems are fault tolerant?

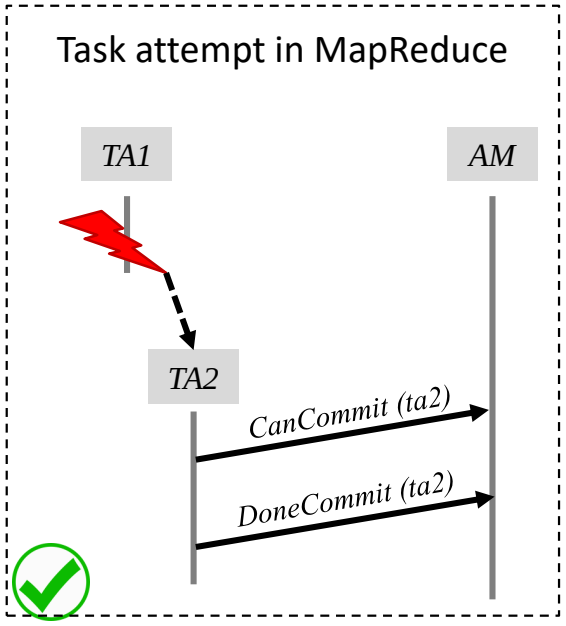
- When a fault happens at node *A* at a special moment
  - Communication with A will hang without timeout
  - A restarted node will fail to take over *A*'s task

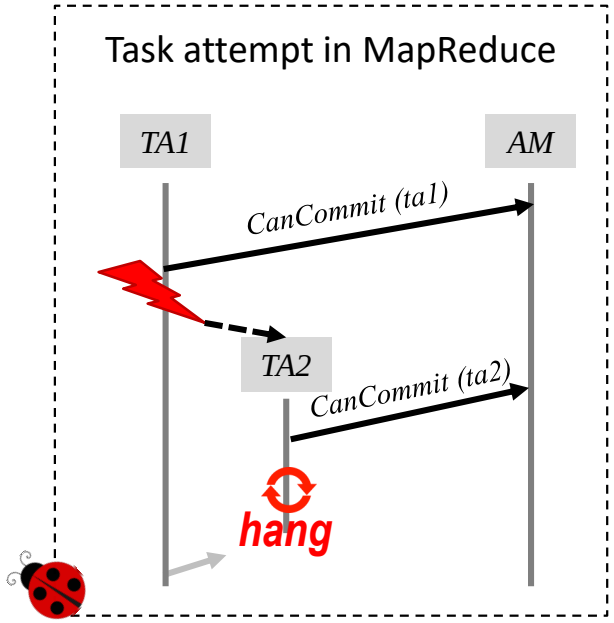




Cloud systems are fault tolerant?

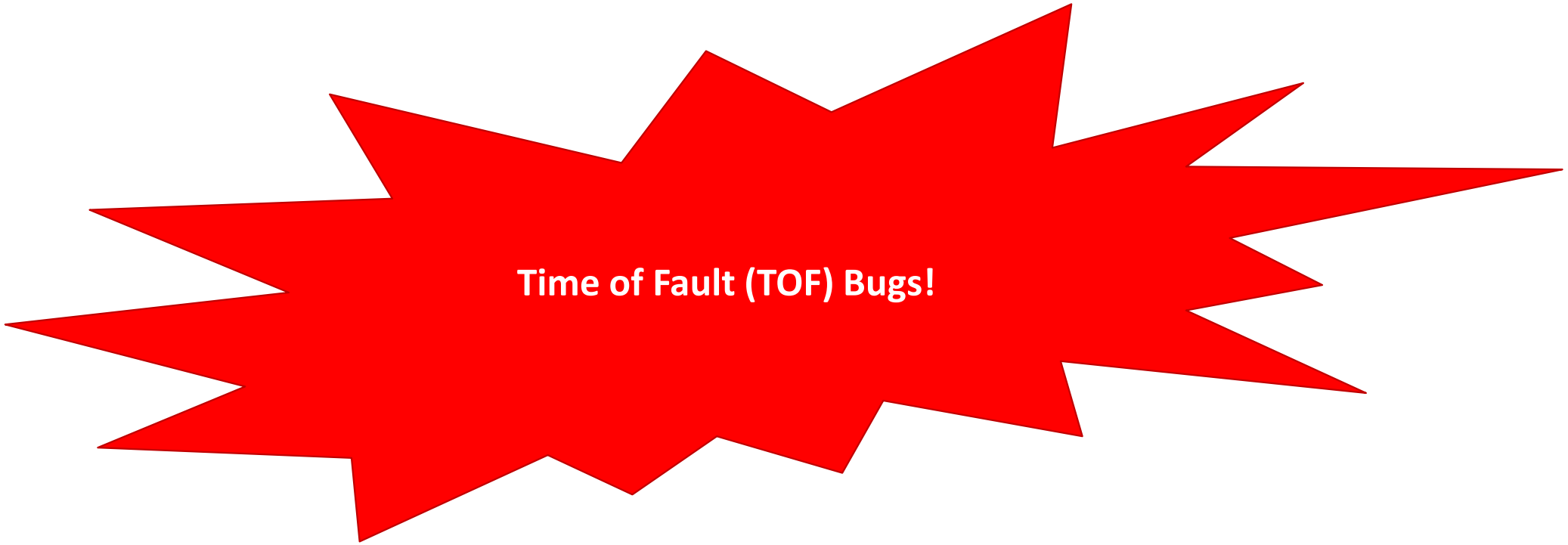
- When a fault happens at node *A* at a special moment
  - Communication with A will hang without timeout
  - A restarted node will fail to take over *A*'s task





Cloud systems are fault tolerant?

- When a fault happens at node *A* at a special moment
  - Communication with A will hang without timeout
  - A restarted node will fail to take over *A*'s task



Why fight TOF bug?

- Common in distributed system [1,2,3]
  - 32% distributed concurrency bugs [1]

[1] Leesatapornwongsa. TaxDC. In ASPLOS'16
[2] Ramnatthan. Correlated crash vulnerabilities. In OSDI'16
[3] Zhenyu. Failure recovery: When the cure is worse than the disease. In HotOS'13

Why fight TOF bug?

- Common in distributed system [1,2,3]
- Difficult to avoid, expose and diagnose
  - Fault rarely occurs during in-house testing
  - Only trigger under special timing

[1] Leesatapornwongsa. TaxDC. In ASPLOS'16
[2] Ramnatthan. Correlated crash vulnerabilities. In OSDI'16
[3] Zhenyu. Failure recovery: When the cure is worse than the disease. In HotOS'13

State of the art – Model checking

Complex manual specifications

State of the art – Model checking

Complex manual specifications

Q1: Can we judge what are TOF bugs without manual specifications?

State of the art – Random fault injection

Complex manual specifications

Many fault injection runs

Q1: Can we judge what are TOF bugs without manual specifications?

State of the art – Random fault injection

Complex manual specifications

Many fault injection runs

Q1: Can we judge what are TOF bugs without manual specifications?

Q2: Can we predict TOF bugs based on just one fault injection, instead of many?

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs

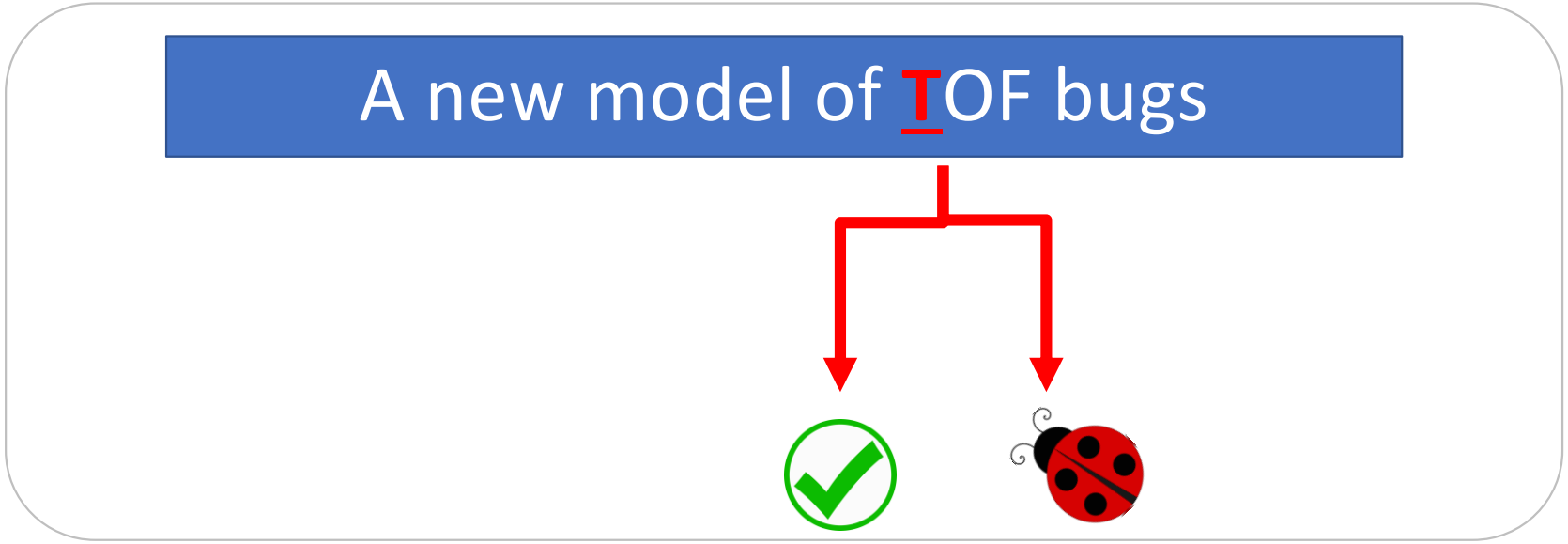
Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs

Key insights

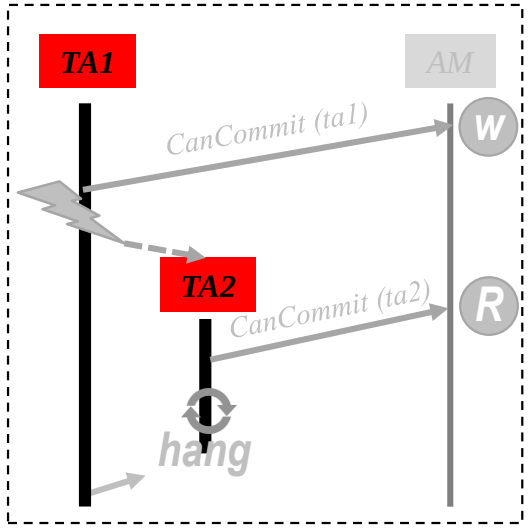
Q1: Can we judge what are TOF bugs without manual specifications?



Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs



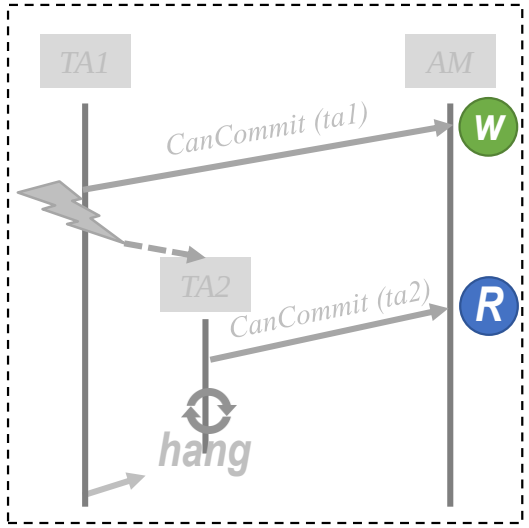

```
Boolean CanCommit(taID) {
  ...
  T.commitID = taID;
}
```

```
Boolean CanCommit(taID) {
  if (T.commitID) {
    return T.commitID == taID;
  }
}
```

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs



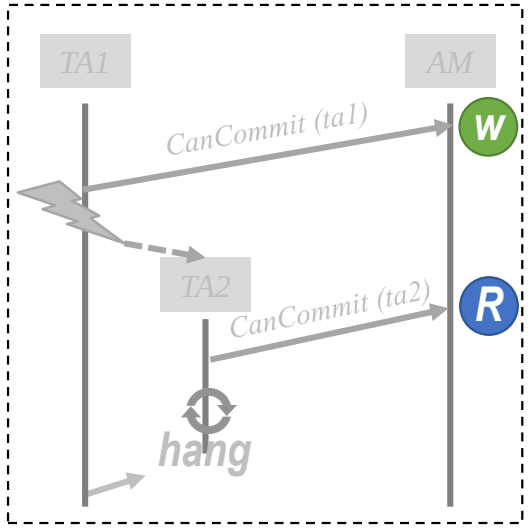

```
Boolean CanCommit(taID) {
  ...
  T.commitID = taID;
}
```

```
Boolean CanCommit(taID) {
  if (T.commitID) {
    return T.commitID == taID;
  }
}
```

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs

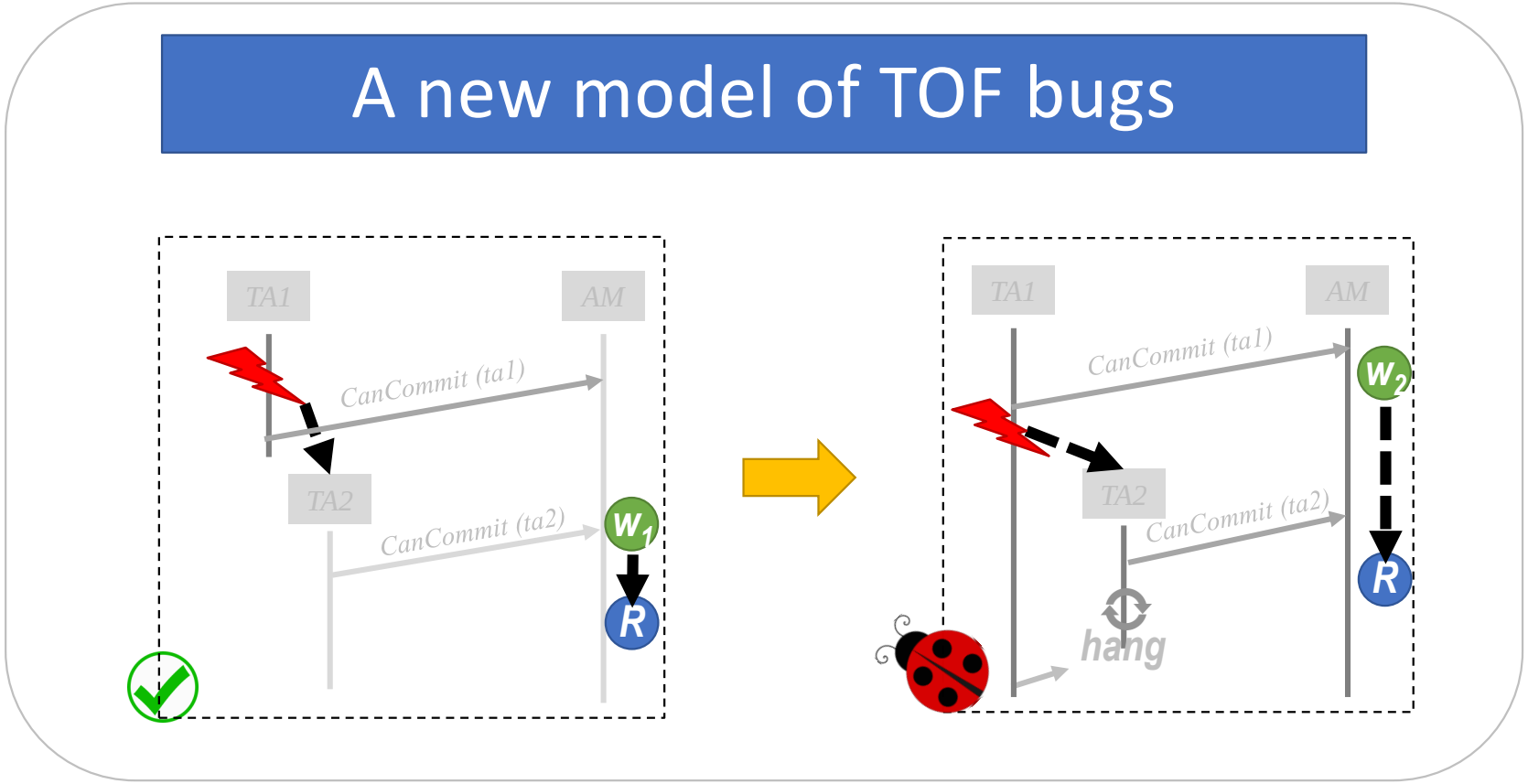



```
Boolean CanCommit(taID) {
  T.commitID = taID;
}
```

```
Boolean CanCommit(taID) {
  if (T.commitID) {
    return T.commitID == taID;
  }
}
```

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?



Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

Q2: Can we predict TOF bugs based on just one fault injection, instead of many?

Key insights

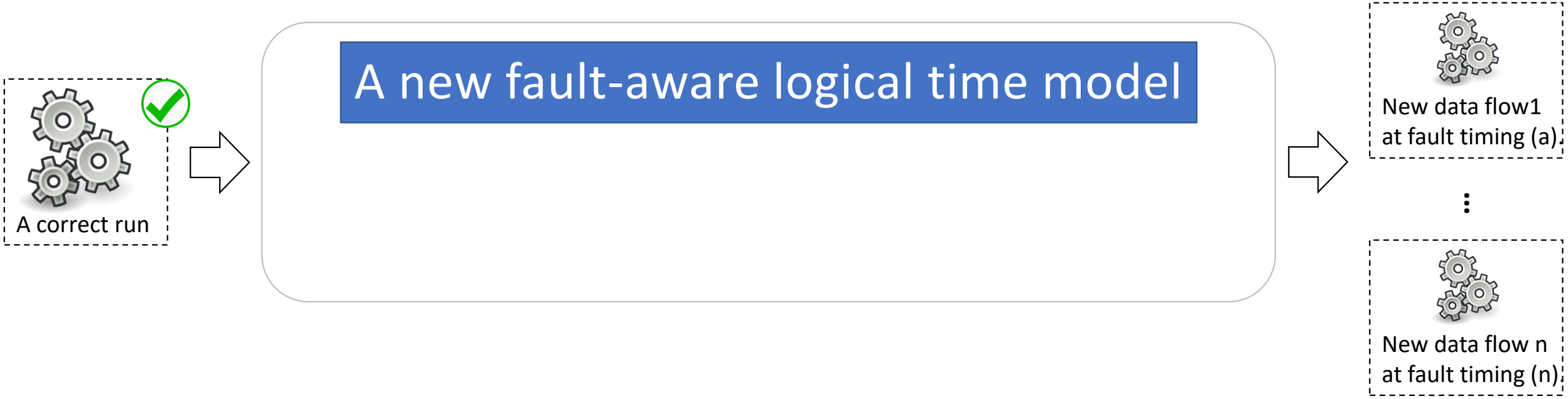
Q1: Can we judge what are TOF bugs without manual specifications?

Q2: Can we predict TOF bugs based on just one fault injection, instead of many?

Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

Q2: Can we predict TOF bugs based on just one fault injection, instead of many?



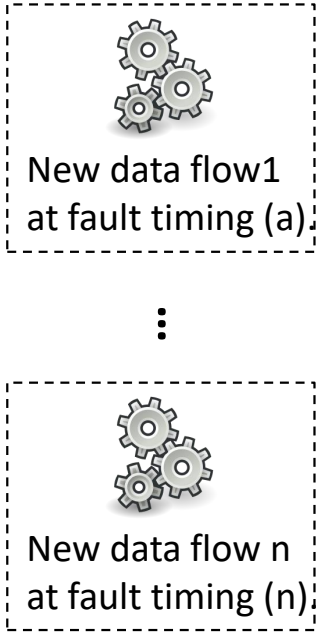
Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

Q2: Can we predict TOF bugs based on just one fault injection, instead of many?

A new fault-aware logical time model

- Predict data flow changes caused by time-of-fault changes



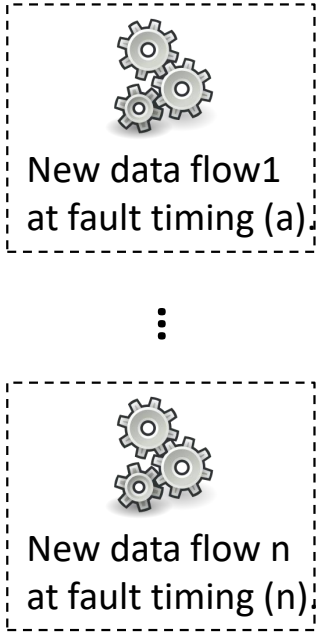
Key insights

Q1: Can we judge what are TOF bugs without manual specifications?

Q2: Can we predict TOF bugs based on just one fault injection, instead of many?

A new fault-aware logical time model

- Predict data flow changes caused by time-of-fault changes
- Consider both synchronization and fault-tolerance operations



Contribution

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

Contribution

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

A new fault-aware logical time

- Predict data flow changes caused by TOF changes
- Consider both synch. and fault-tolerance ops

Contribution

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

A new fault-aware logical time

- Predict data flow changes caused by TOF changes
- Consider both synch. and fault-tolerance ops

FCatch tool

Contribution

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

A new fault-aware logical time

- Predict data flow changes caused by TOF changes
- Consider both synch. and fault-tolerance ops

FCatch tool

Produce correct runs

Contribution

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

A new fault-aware logical time

- Predict data flow changes caused by TOF changes
- Consider both synch. and fault-tolerance ops

FCatch tool

Produce correct runs → Identify conflicting ops

Contribution

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

A new fault-aware logical time

- Predict data flow changes caused by TOF changes
- Consider both synch. and fault-tolerance ops

FCatch tool

Produce correct runs → Identify conflicting ops → Identify fault-tolerant ops

Contribution

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

A new fault-aware logical time

- Predict data flow changes caused by TOF changes
- Consider both synch. and fault-tolerance ops

FCatch tool

Produce correct runs → Identify conflicting ops → Identify fault-tolerant ops →

Contribution

A new model of TOF bugs

- Write and read to a shared state (heap, file, ...)
- From a crash node and a non-crash node
- Data flow changes with TOF change

A new fault-aware logical time

- Predict data flow changes caused by TOF changes
- Consider both synch. and fault-tolerance ops

FCatch tool

Produce correct runs → Identify conflicting ops → Identify fault-tolerant ops →

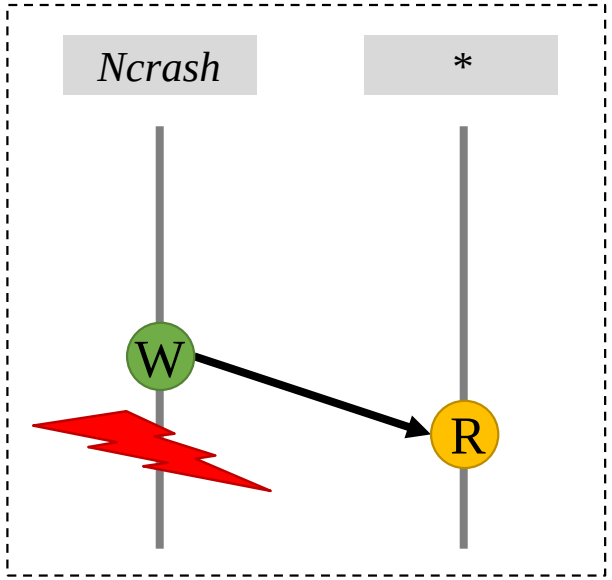
Evaluation

- Report 31 TOF bugs
  - 16 of them truly harmful

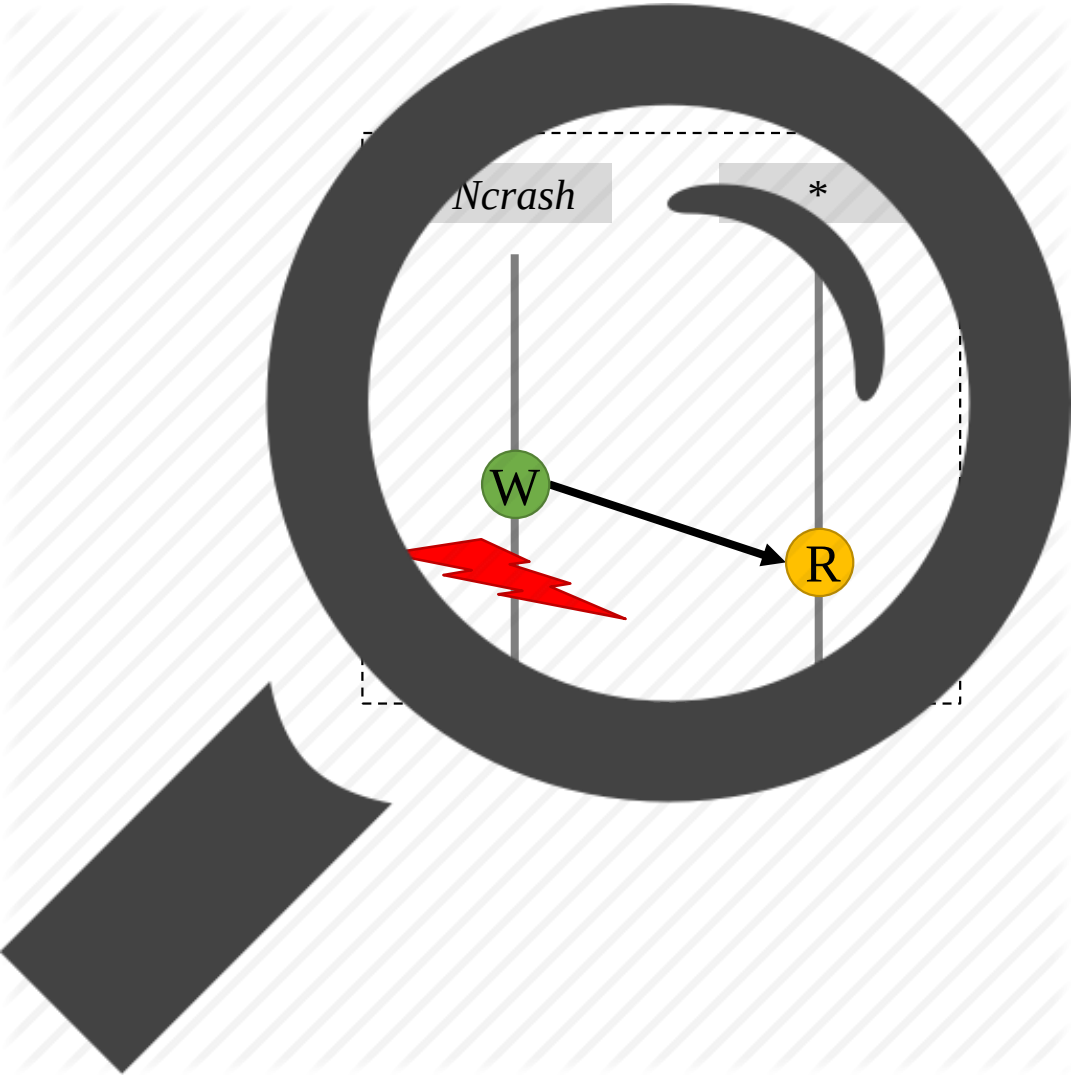
Outline

- Motivation
- Fault-aware logical time model
- FCatch tool
- Evaluation
- Conclusion

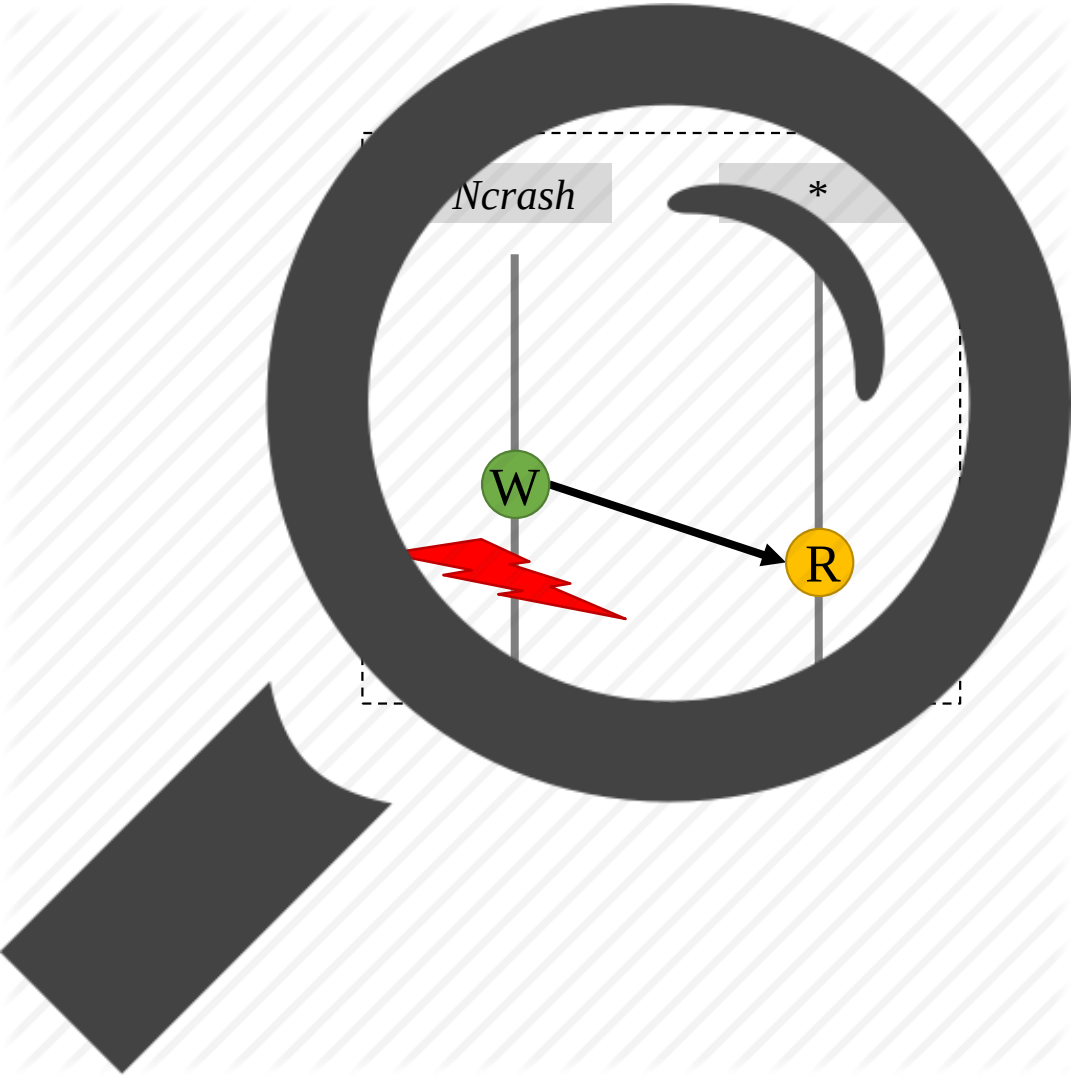
Overview



Overview



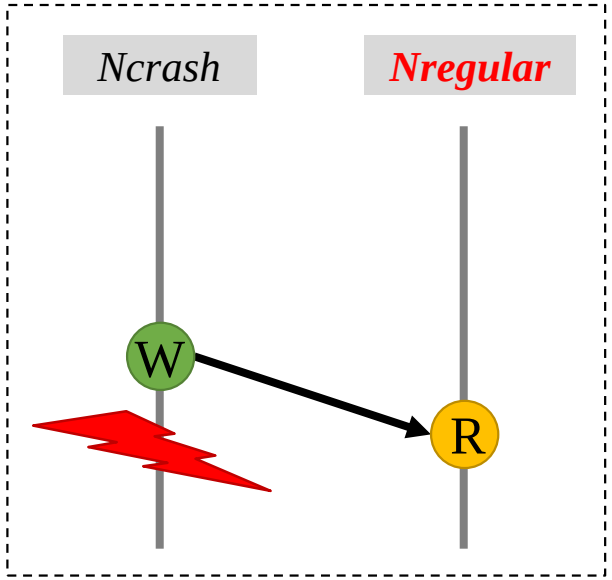
Overview



Fault-aware logical time model

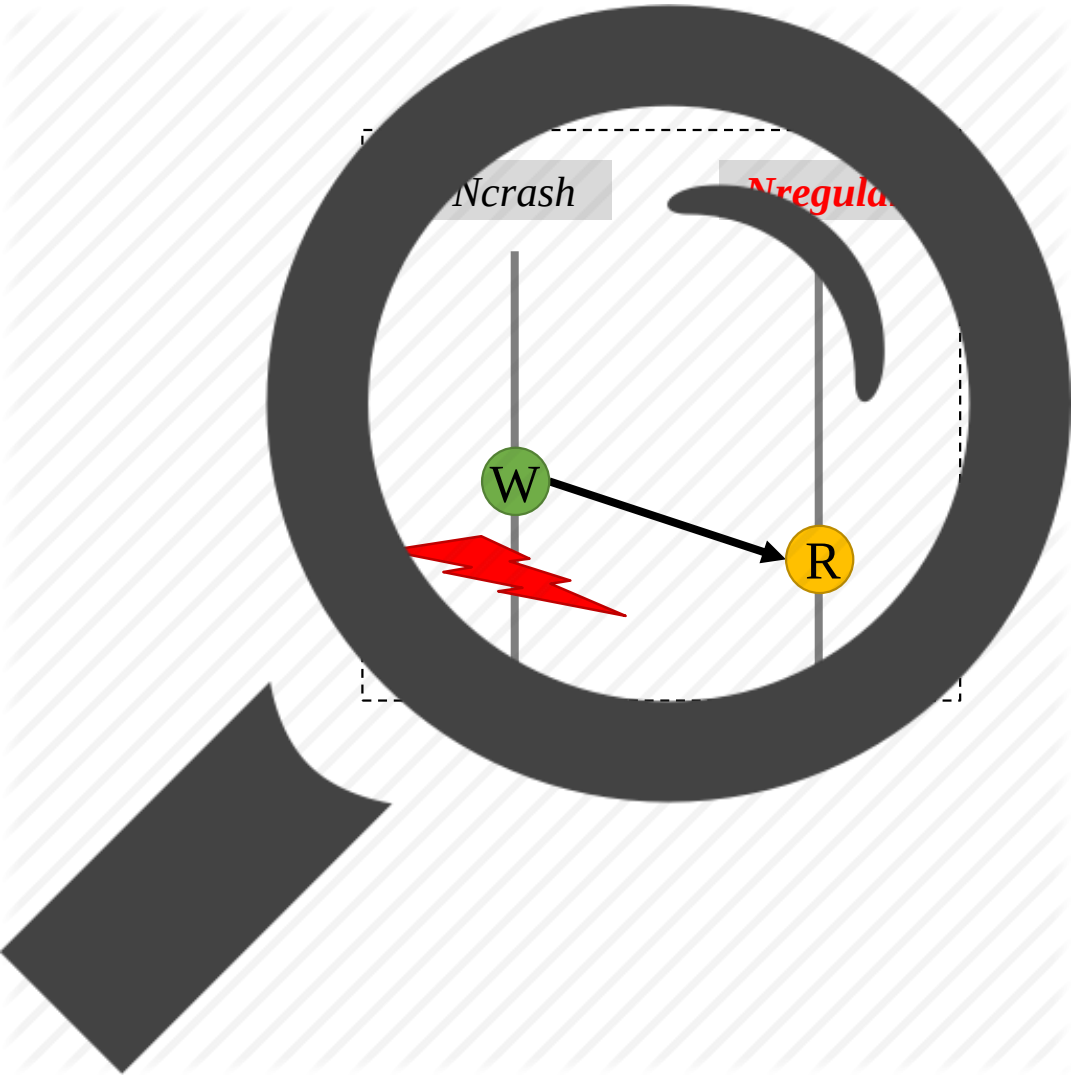
Fault-aware logical time model

-- Crash VS. regular



Fault-aware logical time model

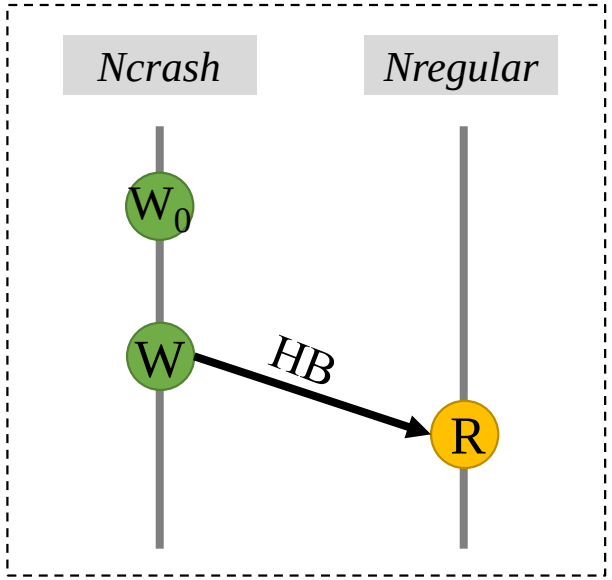
-- Crash VS. regular



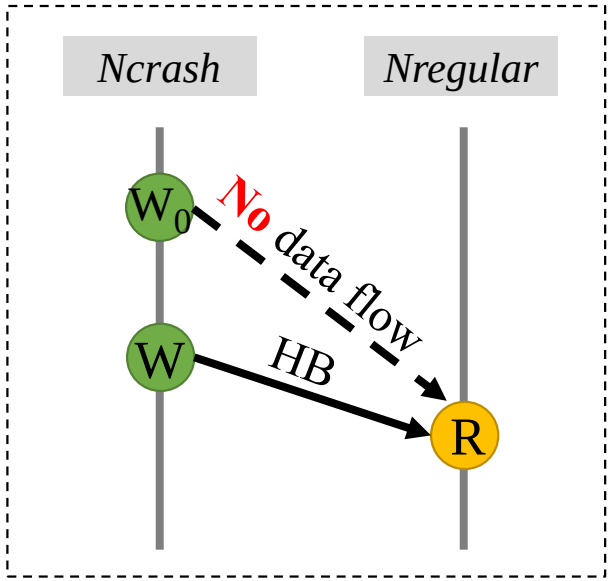
Fault-aware logical time model

Fault-aware logical time model

-- Crash VS. regular

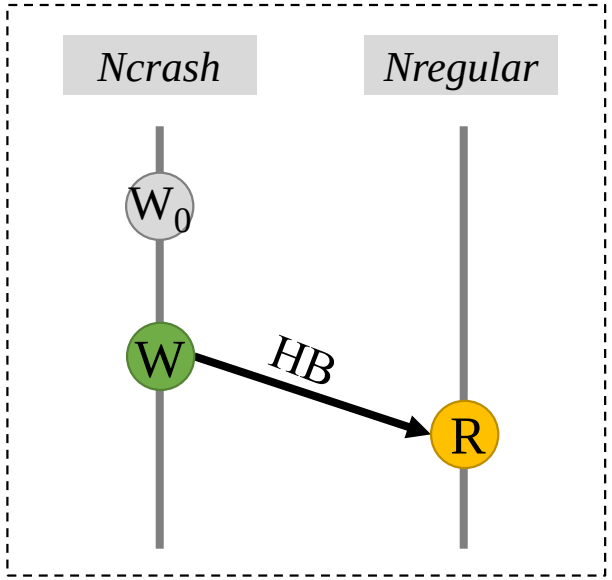


Traditional logical time model



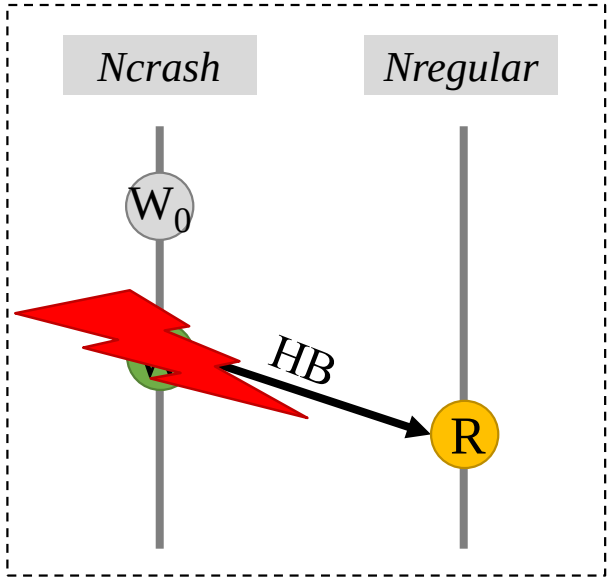
Fault-aware logical time model

-- Crash VS. regular



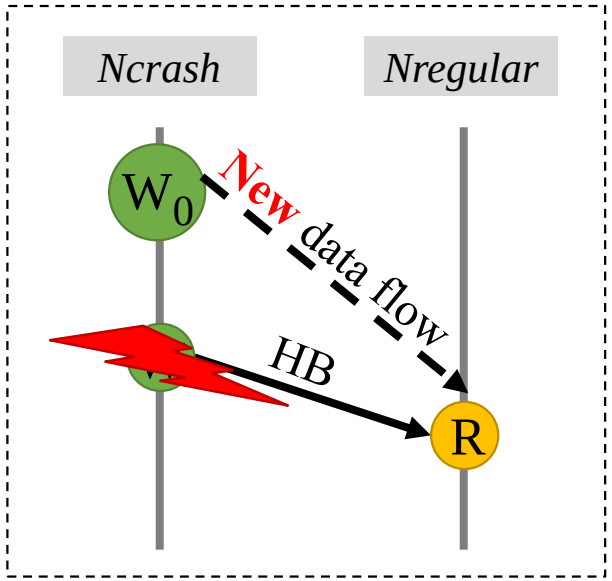
Fault-aware logical time model

-- Crash VS. regular



Fault-aware logical time model

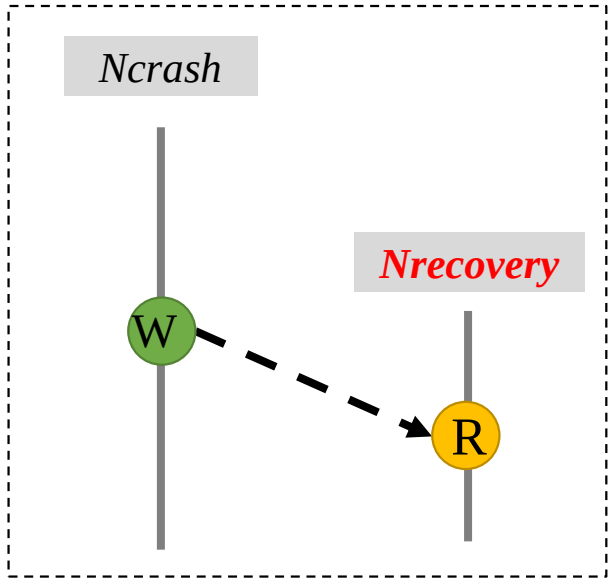
-- *Crash VS. regular*



New data flow between *Ncrash* and *Nregular* introduced by fault.

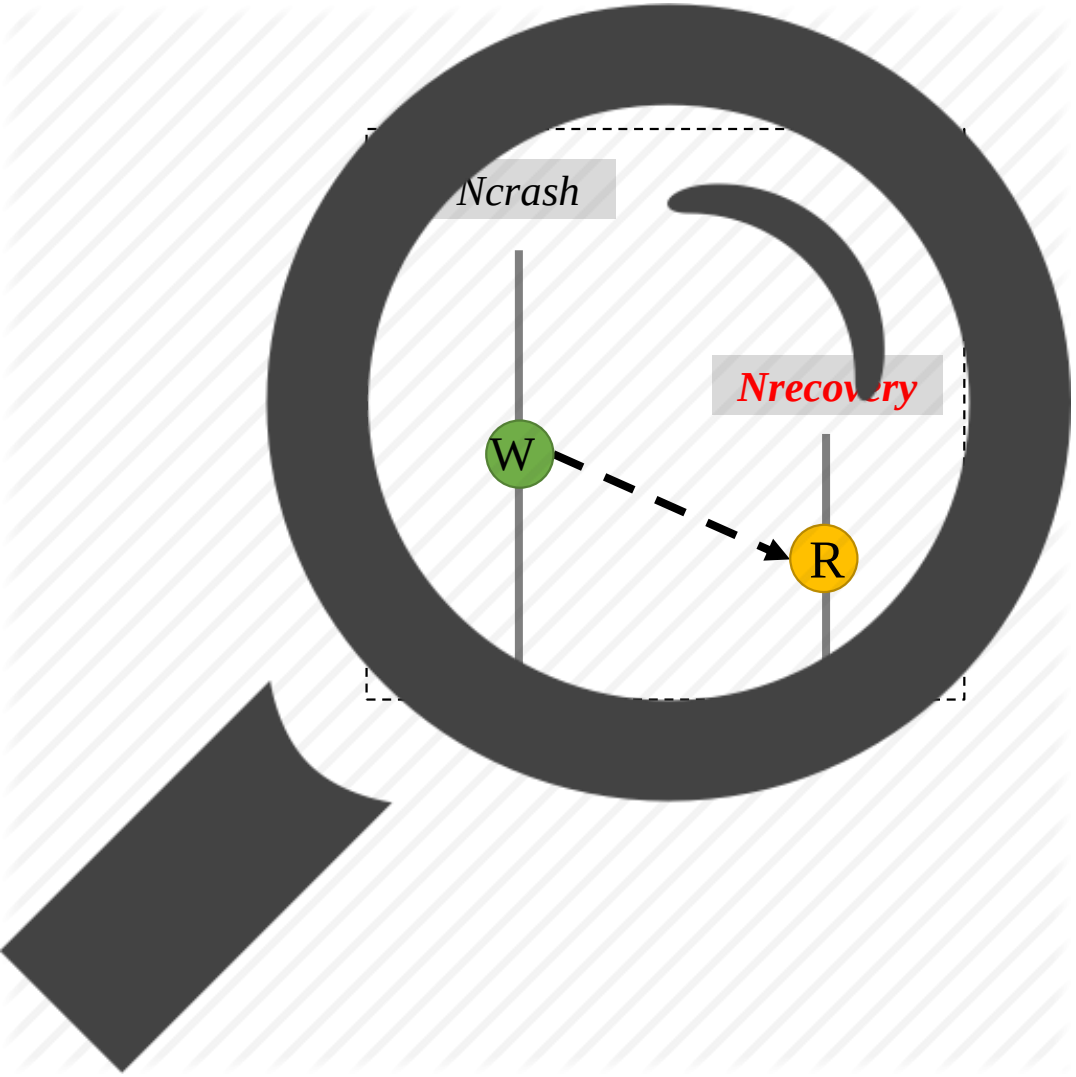
Fault-aware logical time model

-- *Crash VS. recovery*



Fault-aware logical time model

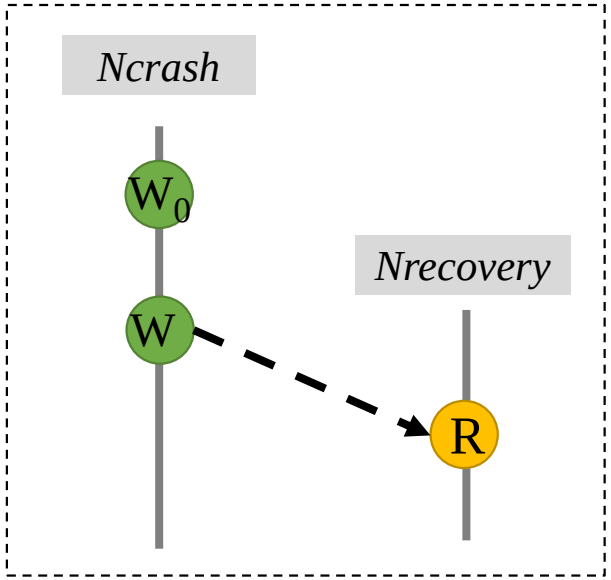
-- Crash VS. recovery



Fault-aware logical time model

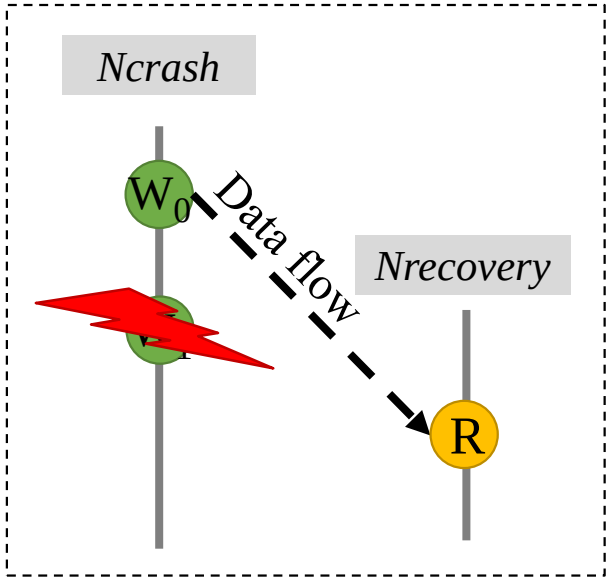
Fault-aware logical time model

-- Crash VS. recovery



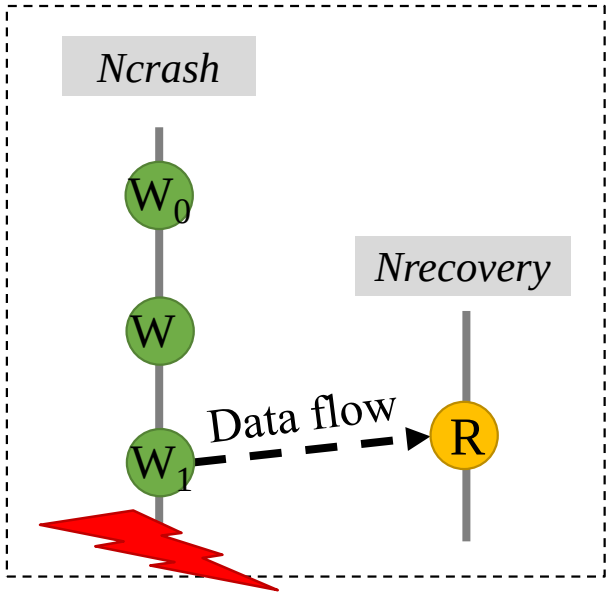
Fault-aware logical time model

-- Crash VS. recovery



Fault-aware logical time model

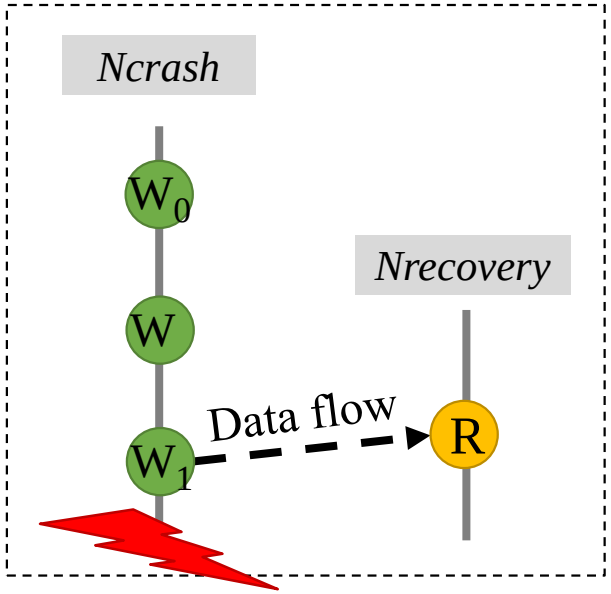
-- *Crash VS. recovery*



Data flow between *Ncrash* and *Nrecovery* is totally determined by TOF.

Fault-aware logical time model

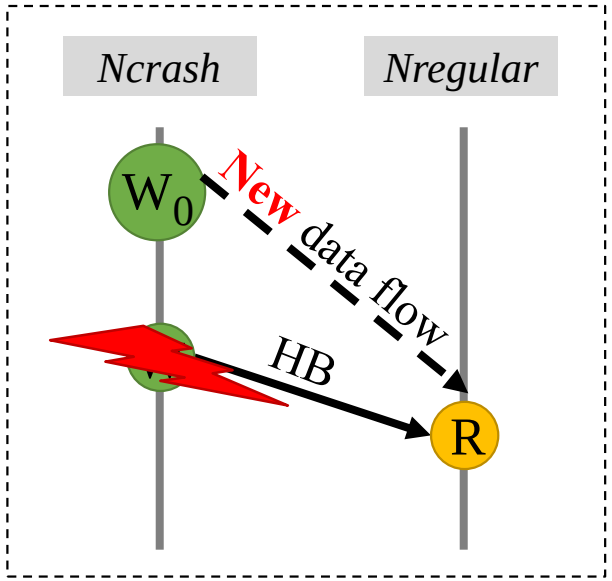
-- *Crash VS. recovery*

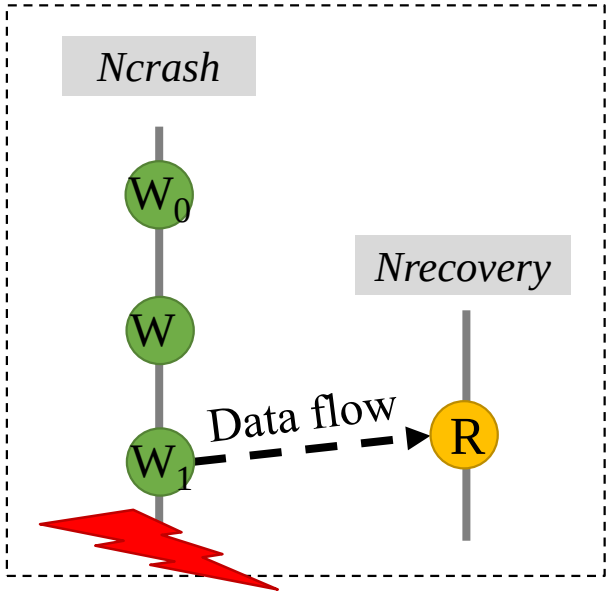


Fault-tolerance: sanity check

```
//Recovery node
if (f.valid()) { //sanity check
  dt = f.read(); //read
}
```

Fault-aware logical time model



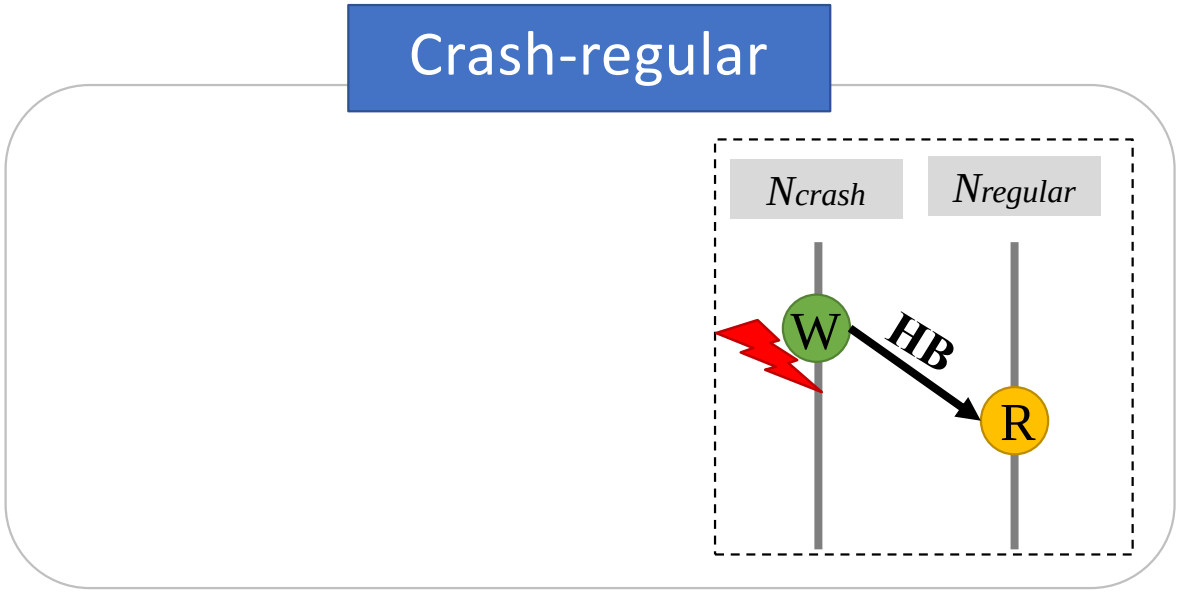


What are TOF bugs?

What are TOF bugs?

Crash-regular

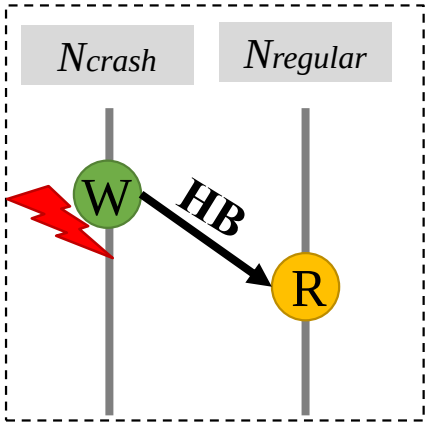
What are TOF bugs?



What are TOF bugs?

Crash-regular

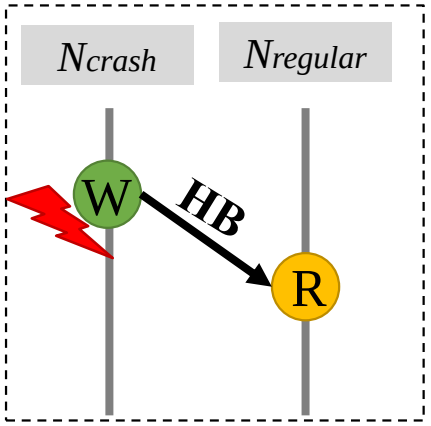
- *Fault timing:* before W



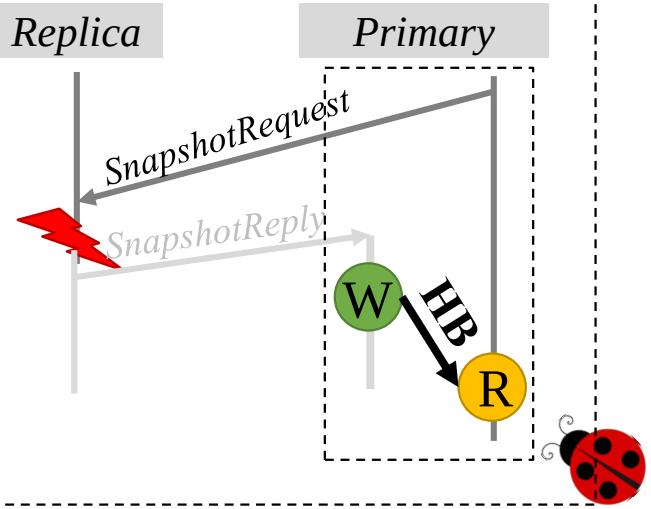
What are TOF bugs?

Crash-regular

- *Fault timing:* before W



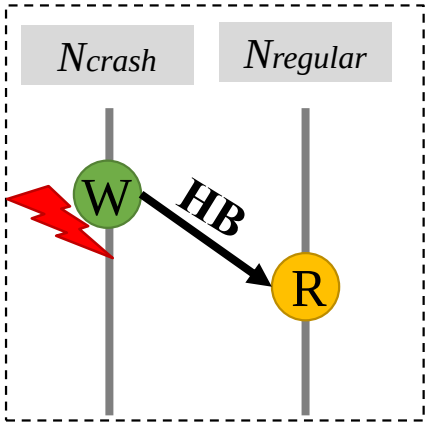
Merkle tree repair in Cassandra



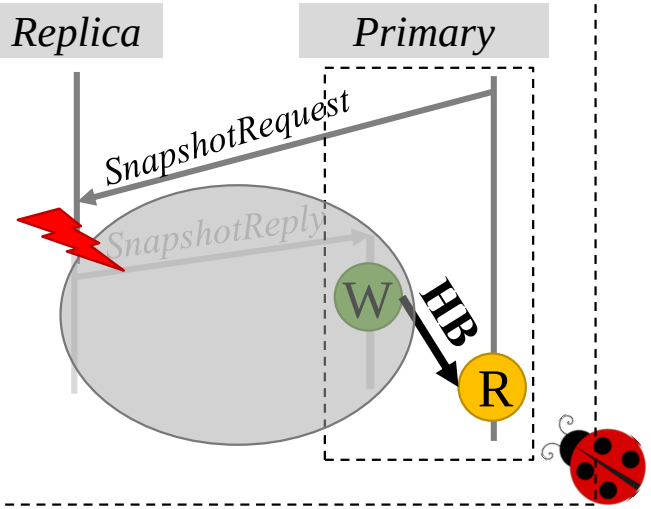
What are TOF bugs?

Crash-regular

- *Fault timing:* before W



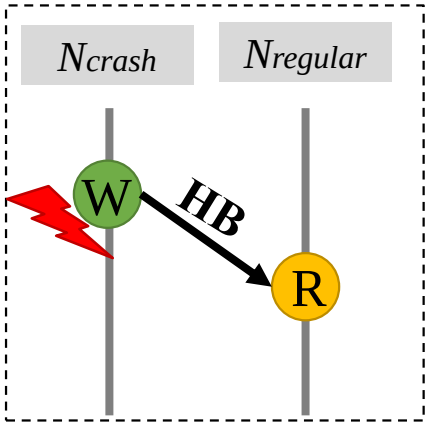
Merkle tree repair in Cassandra



What are TOF bugs?

Crash-regular

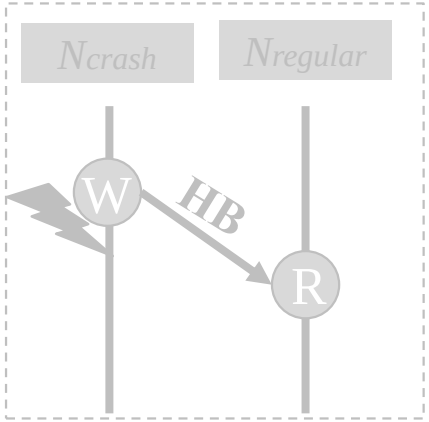
- *Fault timing:* before W
- *Fault-tolerance:* timeout



What are TOF bugs?

Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout

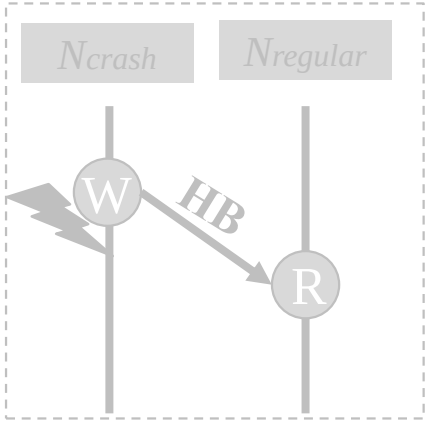


Crash-recovery

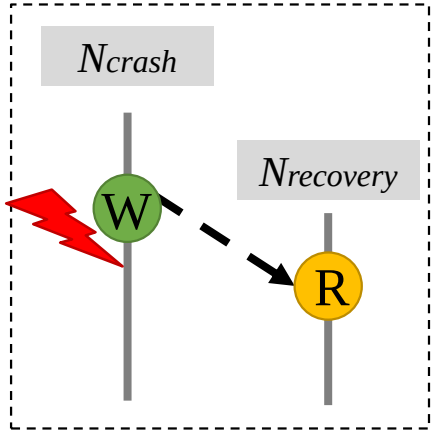
What are TOF bugs?

Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



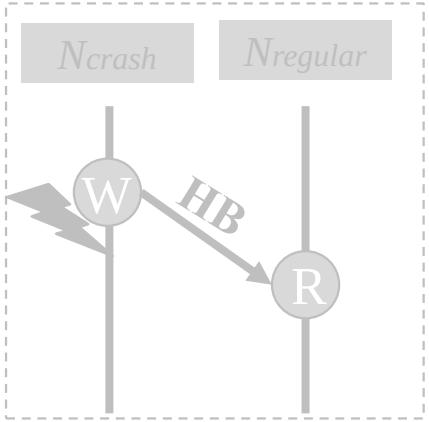
Crash-recovery



What are TOF bugs?

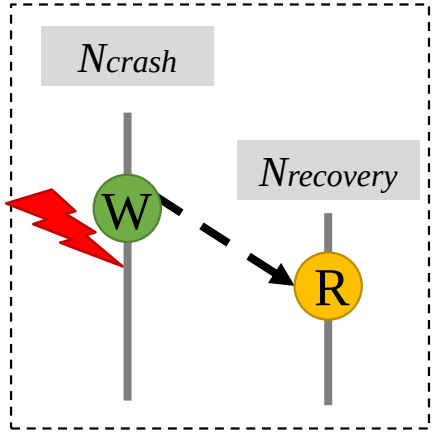
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout

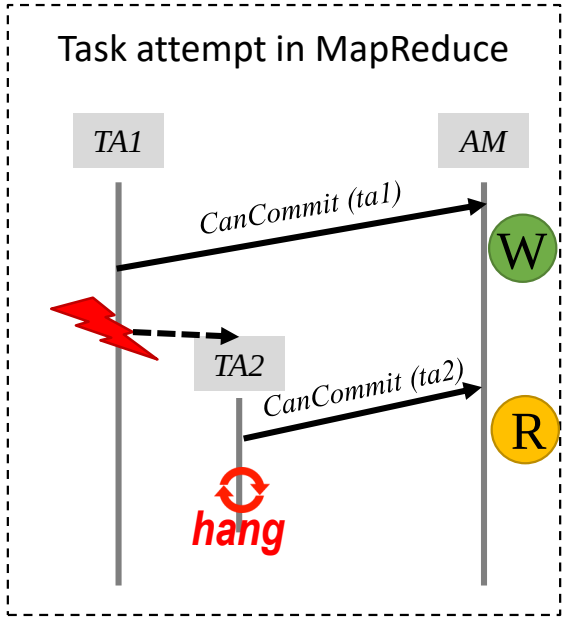


Crash-recovery

- *Fault timing:* after W

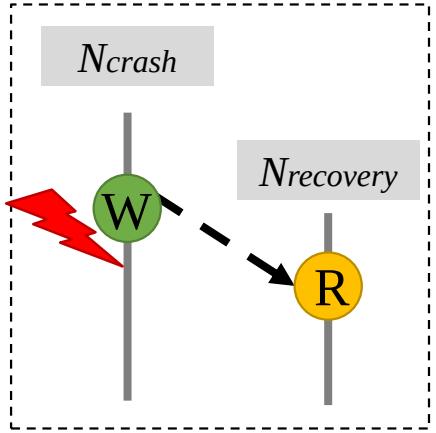


What are TOF bugs?

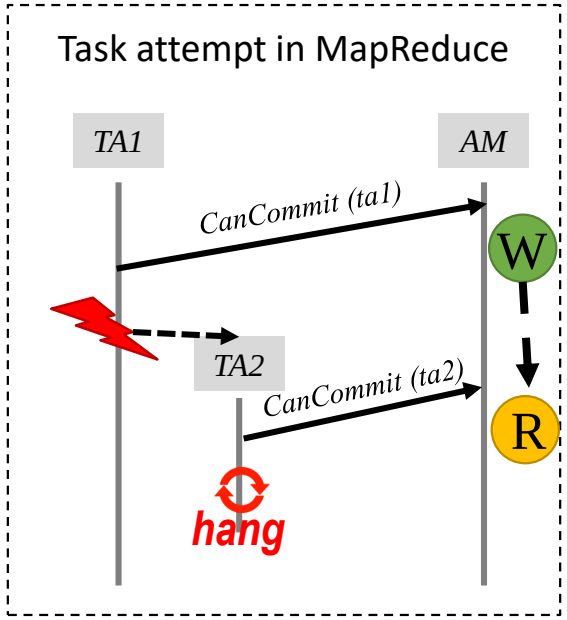


Crash-recovery

- *Fault timing:* after W

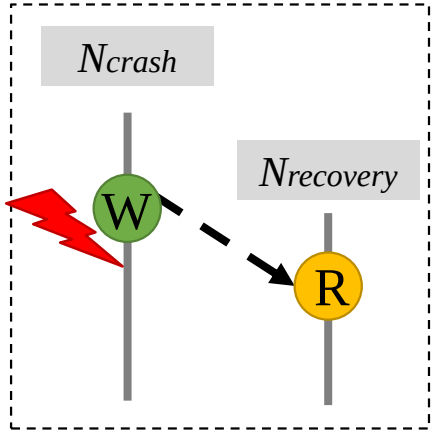


What are TOF bugs?



Crash-recovery

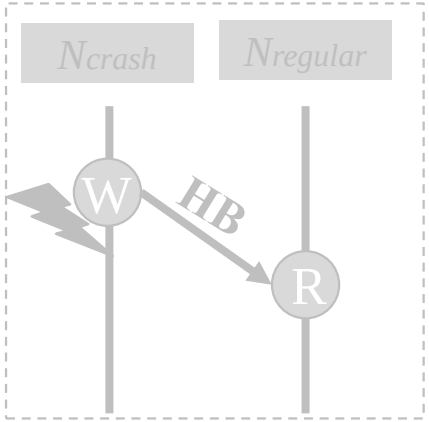
- *Fault timing:* after W



What are TOF bugs?

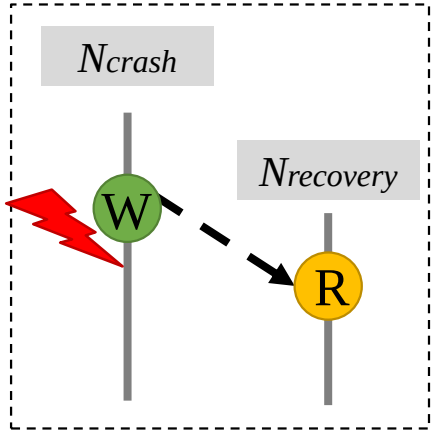
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



Crash-recovery

- *Fault timing:* after W
- *Fault-tolerance:* Sanity check



Outline

- Motivation
- Our TOF bug model
- FCatch tool
- Evaluation
- Conclusion

FCatch overview

Step1: produce correct runs

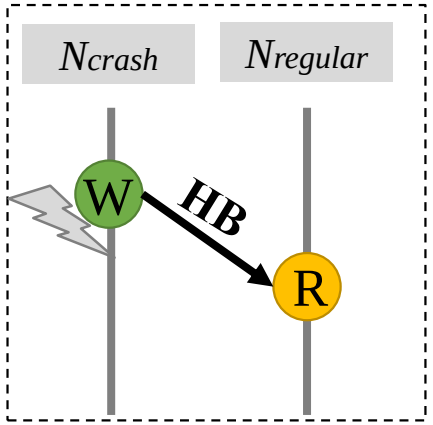
- What are correct runs to observe?

Step1: produce correct runs

- What are correct runs to observe?

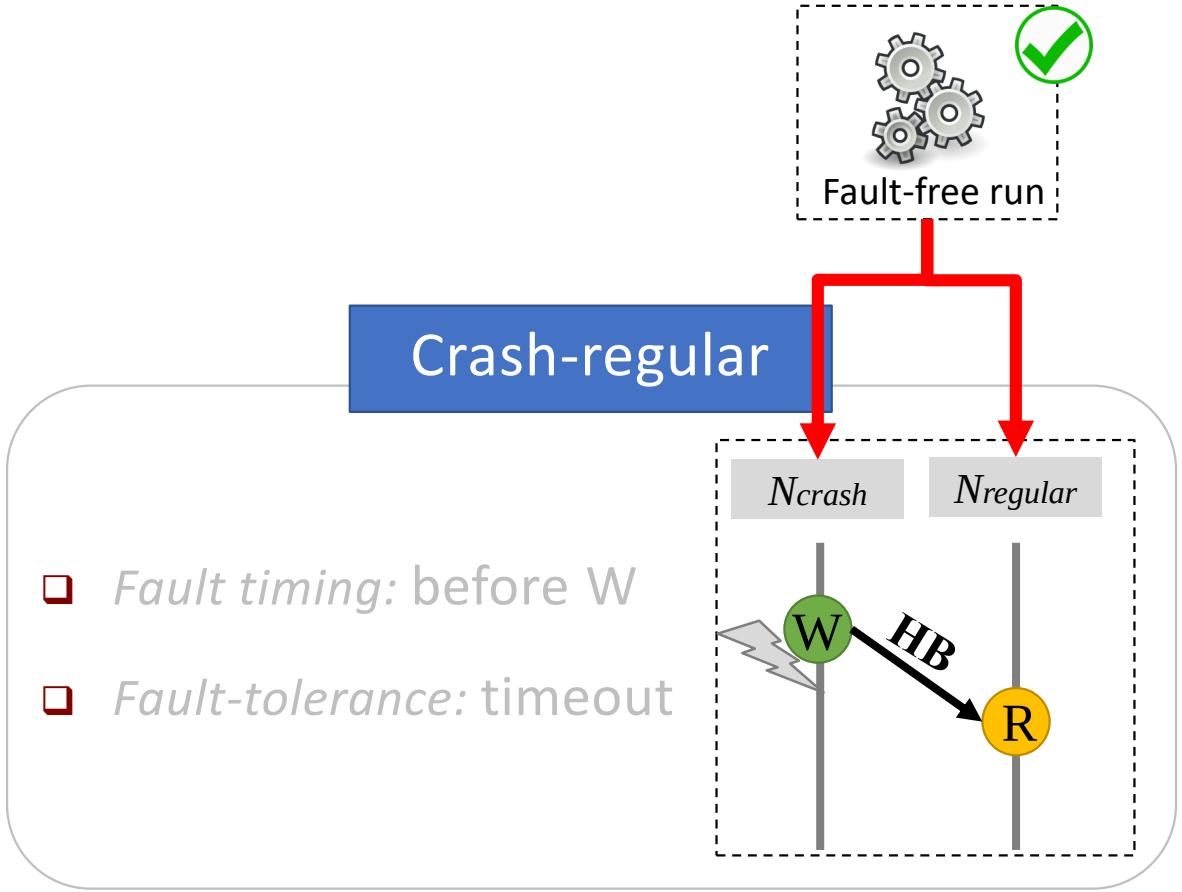
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



Step1: produce correct runs

- What are correct runs to observe?

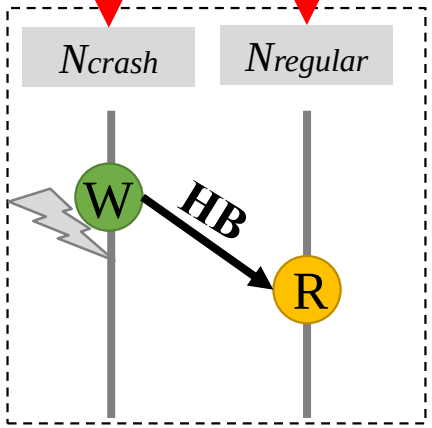


Step1: produce correct runs

- What are correct runs to observe?

Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



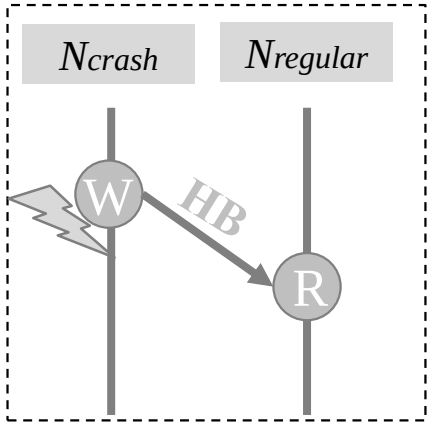
No faults need to be injected!

Step1: produce correct runs

- What are correct runs to observe?

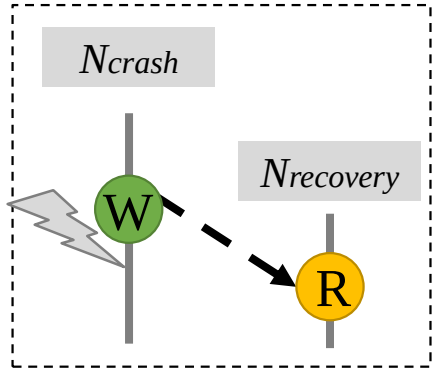
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



Crash-recovery

- *Fault timing:* after W
- *Fault-tolerance:* sanity check

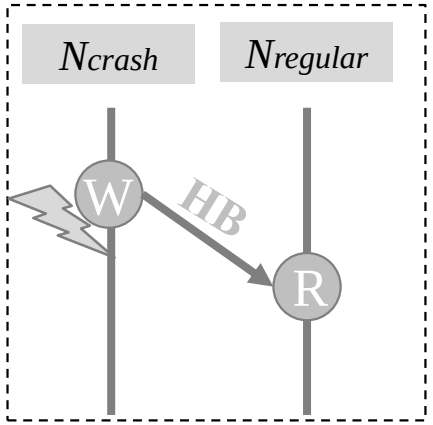


Step1: produce correct runs

- What are correct runs to observe?

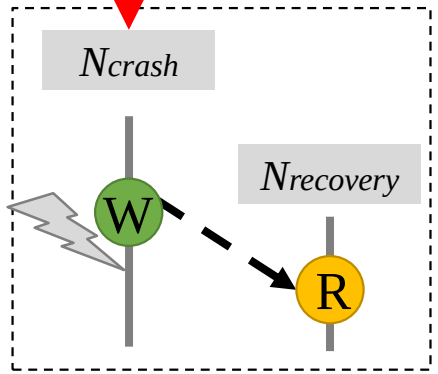
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



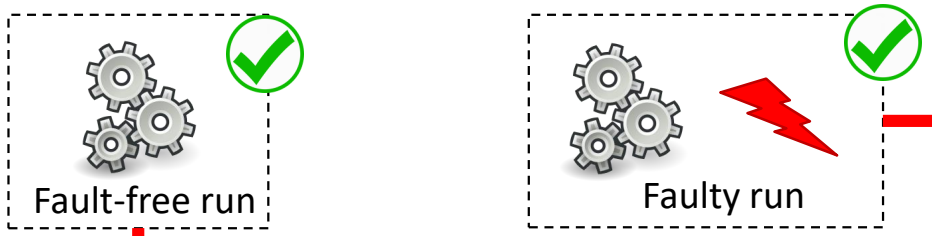
Crash-recovery

- *Fault timing:* after W
- *Fault-tolerance:* sanity check



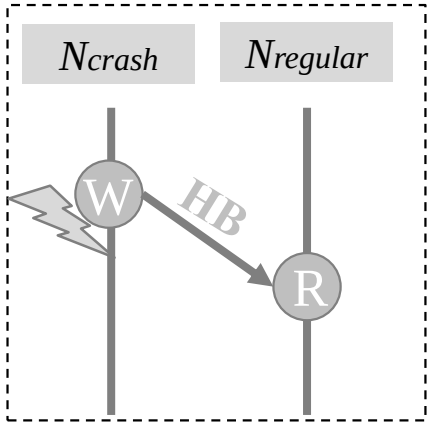
Step1: produce correct runs

- What are correct runs to observe?



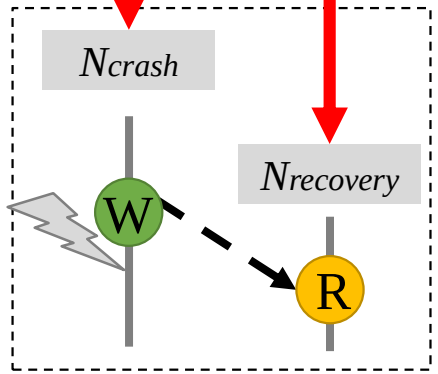
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



Crash-recovery

- *Fault timing:* after W
- *Fault-tolerance:* sanity check

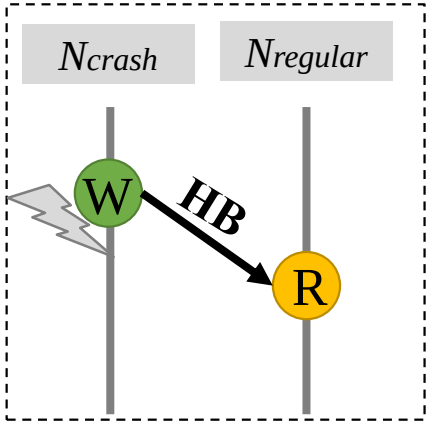


Step2: identify conflicting operations

- Crash-regular TOF bug

Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout

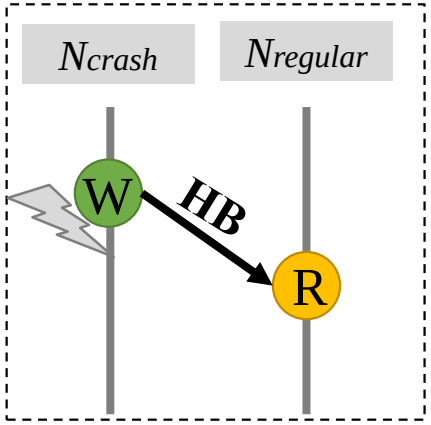


Step2: identify conflicting operations

- Crash-regular TOF bug
  - W & R are from different nodes (fault-free traces)
  - W & R have blocking happens-before relations

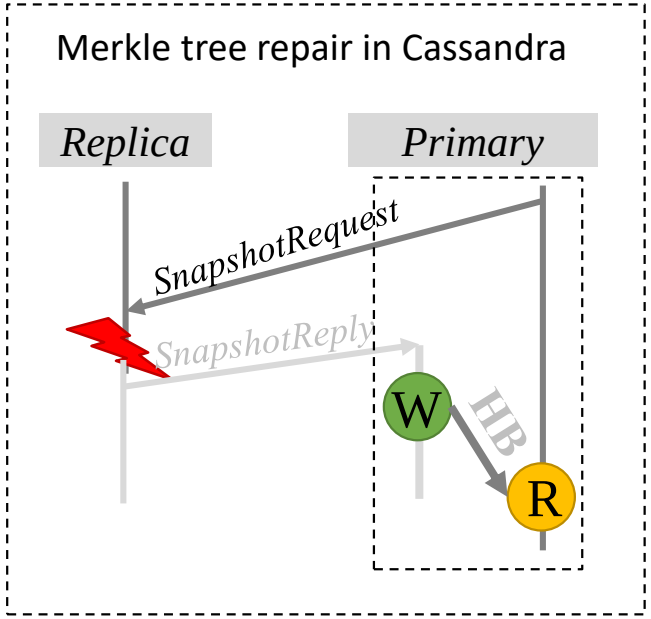
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



Step2: identify conflicting operations

- Crash-regular TOF bug
  - W & R are **from** different nodes (fault-free traces)
  - W & R have blocking happens-before relations



Step2: identify conflicting operations

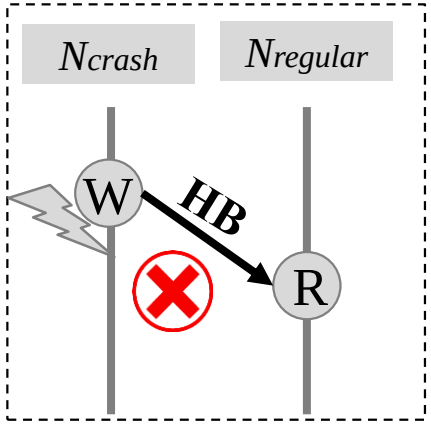
- Crash-regular TOF bug
- Crash-recovery TOF bug

Step3: identify fault-tolerant ops

- Crash-regular TOF bug

Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout

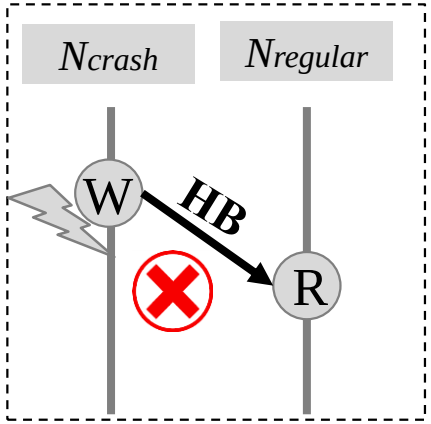


Step3: identify fault-tolerant ops

- Crash-regular TOF bug

Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout




```
//Regular node
obj.wait(long timeout); //R: obj
```

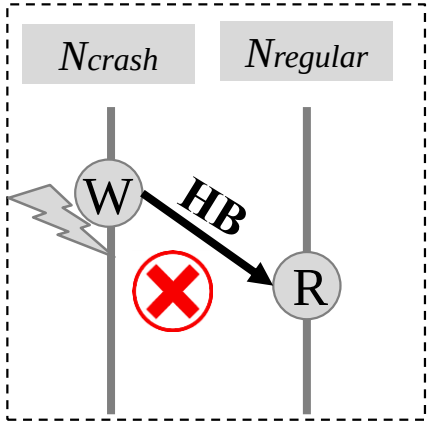
Fault-tolerance mechanism: timeout

Step3: identify fault-tolerant ops

- Crash-regular TOF bug

Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout




```
//Regular node
obj.wait(long timeout); //R: obj
```

Fault-tolerance mechanism: timeout

Sol: statically check *R*'s bytecode.

Step3: identify fault-tolerant ops

- Crash-regular TOF bug
- Crash-recovery TOF bug

Outline

- Motivation
- Our TOF bug model
- FCatch tool
- Evaluation
- Conclusion

Methodology

- Benchmarks
  - 7 real-world TOF bugs from TaxDC [1]
    - 3 crash-regular TOF bugs
    - 4 crash-recovery TOF bugs
  - 4 distributed systems

[1] Leesatapornwongsa. TaxDC. In ASPLOS'16

Overall results

	Crash-regular			Crash-recovery		
	#. Bench (harmful)	#. Unknown (harmful)	#. FP	#. Bench (harmful)	#. Unknown (harmful)	#. FP
CA1&2	2	1	0	-	0	2
HB-1	1	0	3	-	0	6
HB-2	-	2	2	1	2	0
MR-1	-	1	0	1	1	0
MR-2	-	1	0	2	1	0
ZK	-	0	0	1	0	2
Total	3	4	5	5	4	10

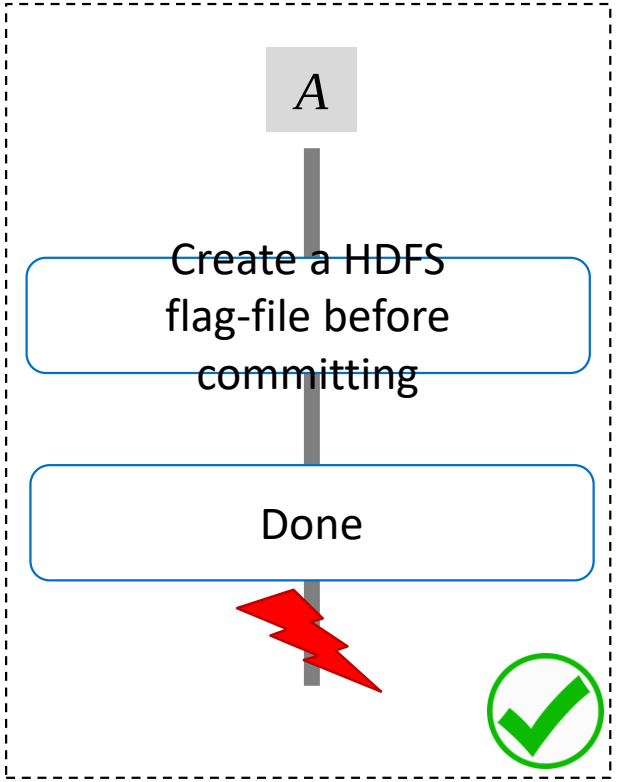
Overall results

	Crash-regular			Crash-recovery		
	#. Bench (harmful)	#. Unknown (harmful)	#. FP	#. Bench (harmful)	#. Unknown (harmful)	#. FP
CA1&2	2	1	0	-	0	2
HB-1	1	0	3	-	0	6
HB-2	-	2	2	1	2	0
MR-1	-	1	0	1	1	0
MR-2	-	1	0	2	1	0
ZK	-	0	0	1	0	2
Total	3	4	5	5	4	10

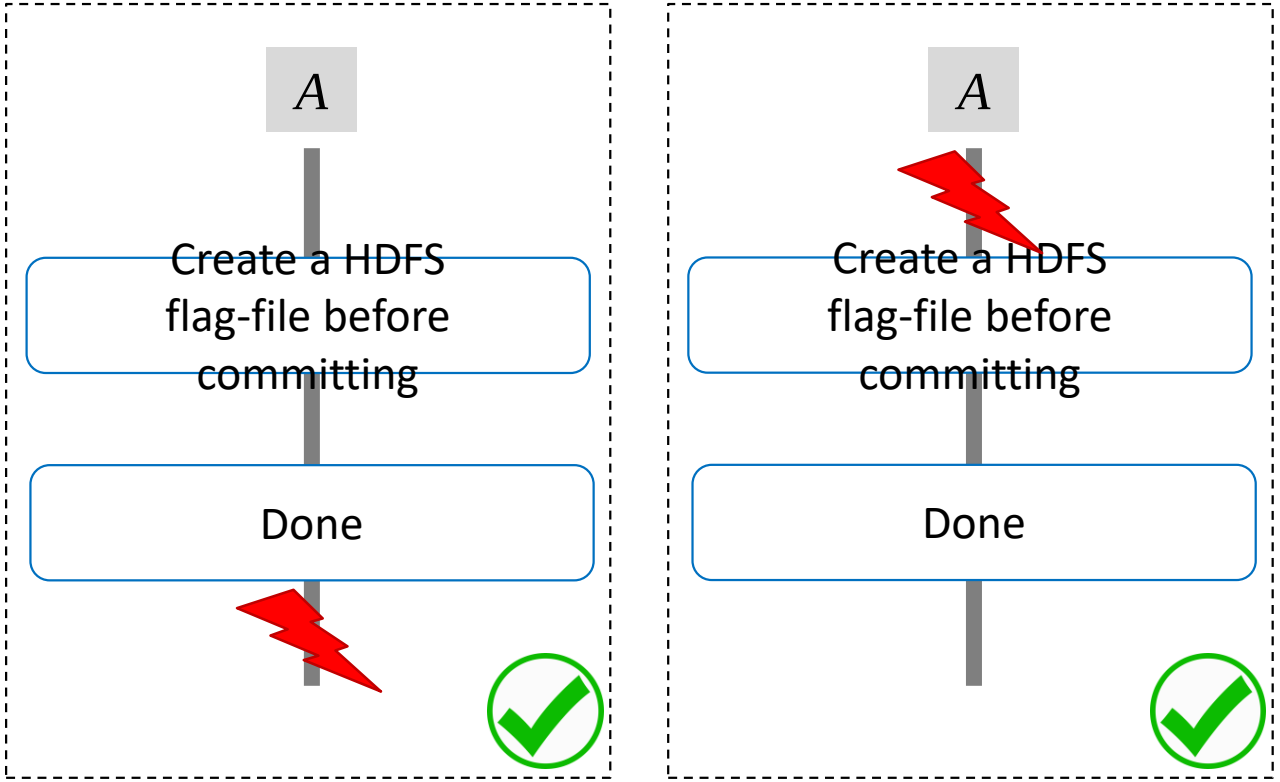
Overall results

	Crash-regular			Crash-recovery		
	#. Bench (harmful)	#. Unknown (harmful)	#. FP	#. Bench (harmful)	#. Unknown (harmful)	#. FP
CA1&2	2	1	0	-	0	2
HB-1	1	0	3	-	0	6
HB-2	-	2	2	1	2	0
MR-1	-	1	0	1	1	0
MR-2	-	1	0	2	1	0
ZK	-	0	0	1	0	2
Total	3	4	5	5	4	10

Case study: an unknown harmful report

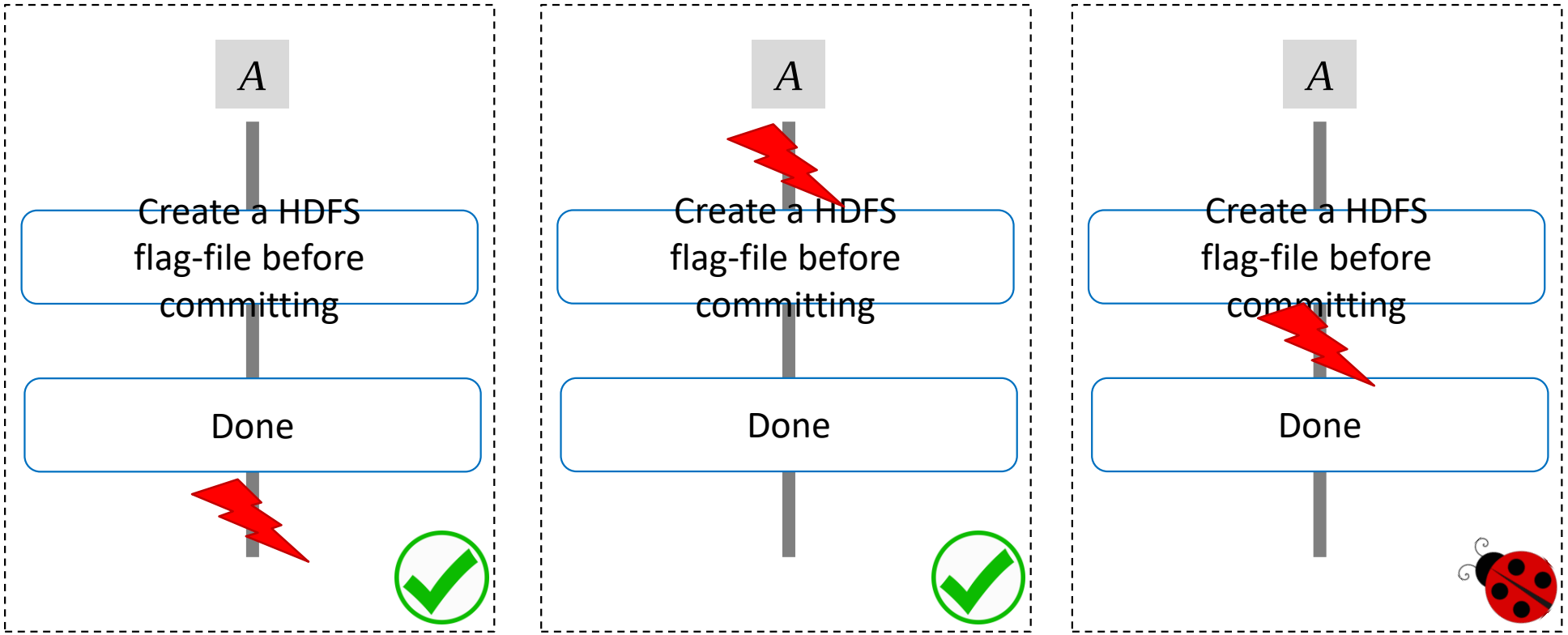


Case study: an unknown harmful report



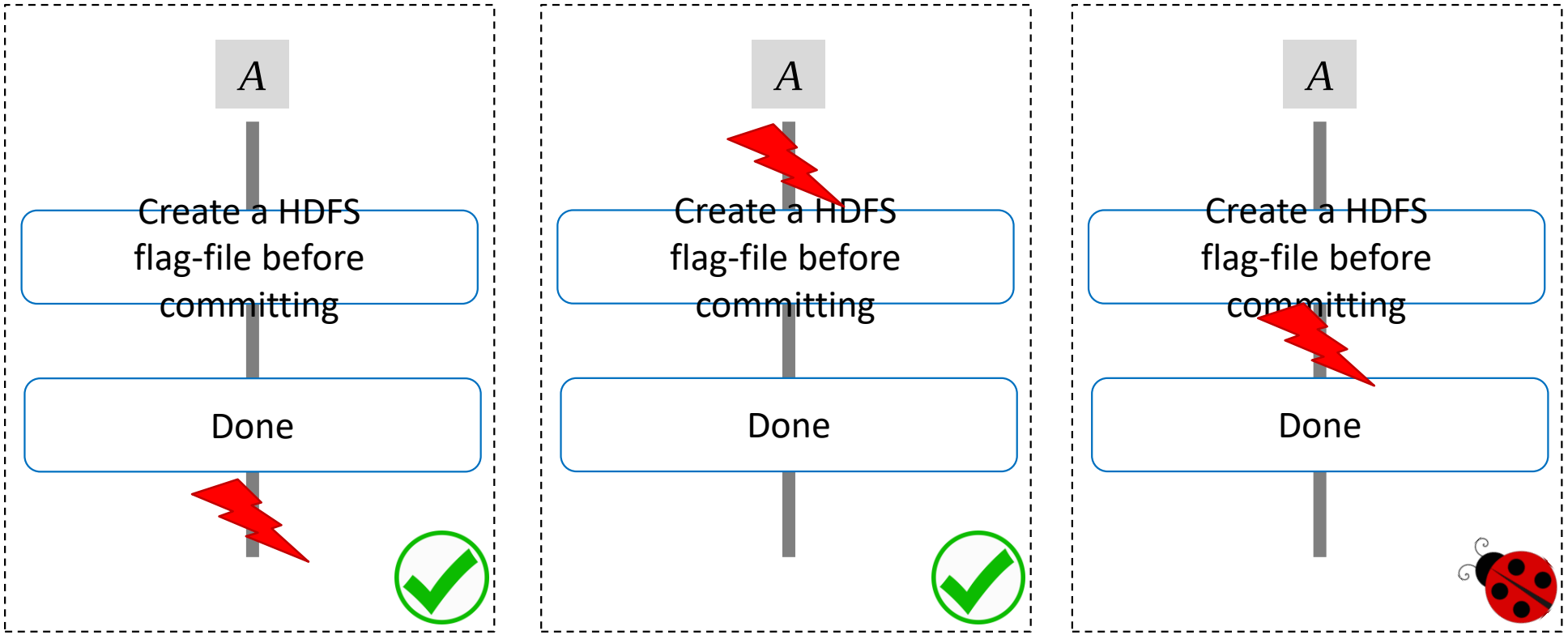
Case study: an unknown harmful report

- MR-5485



Case study: an unknown harmful report

- MR-5485



- Readme, VM and scripts to reproduce each harmful report:

http://fcatch.cs.uchicago.edu/

Other results in our paper

- Random fault injection
- Performance overhead
- Crash-point sensitivity
- …

Outline

- Motivation
- Our TOF bug model
- FCatch tool
- Evaluation
- Conclusion

Take away

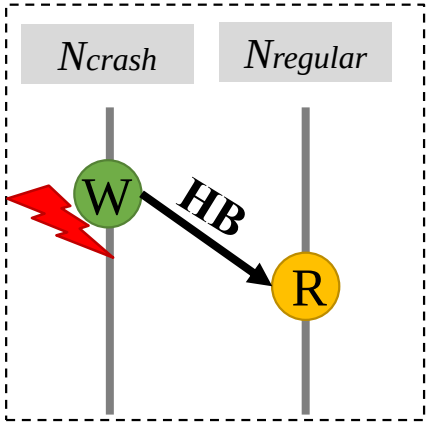
- TOF bugs are a timing problem.

Take away

- TOF bugs are a timing problem.
- Fault-aware logical time model.

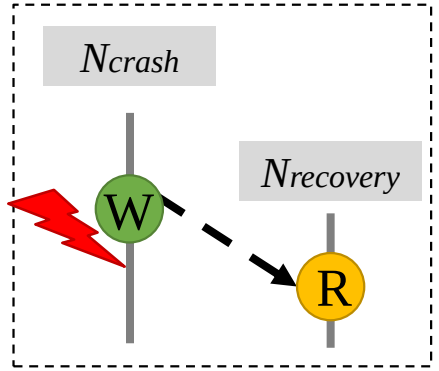
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



Crash-recovery

- *Fault timing:* after W
- *Fault-tolerance:* Sanity check

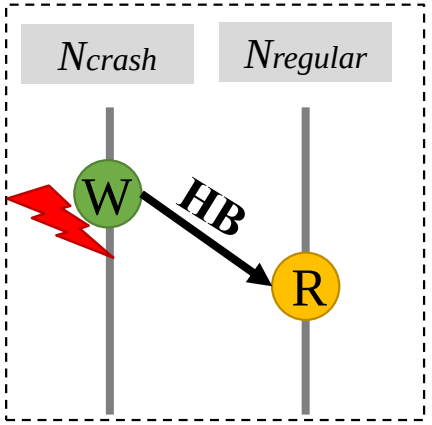


Take away

- TOF bugs are a timing problem.
- Fault-aware logical time model.
- FCatch detects TOF bugs from correct runs.

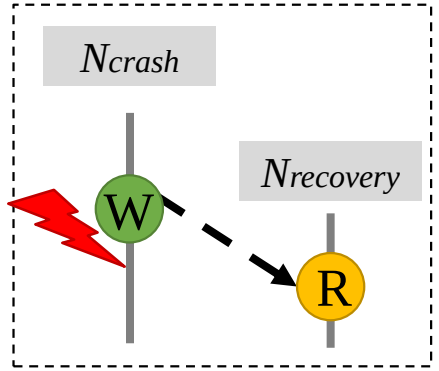
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



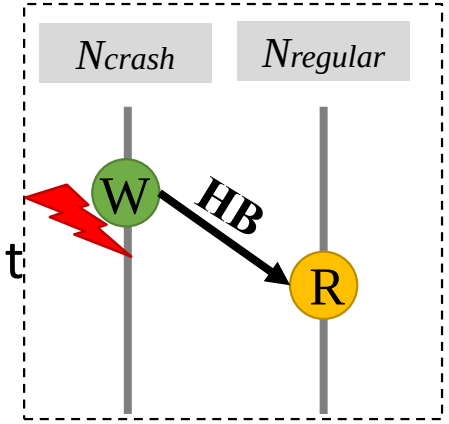
Crash-recovery

- *Fault timing:* after W
- *Fault-tolerance:* Sanity check



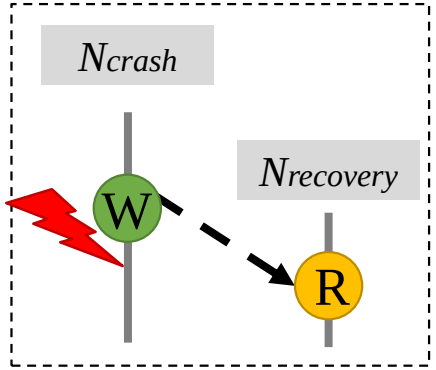
Crash-regular

- *Fault timing:* before W
- *Fault-tolerance:* timeout



Crash-recovery

- *Fault timing:* after W
- *Fault-tolerance:* Sanity check



Q&A

http://fcatch.cs.uchicago.edu/

DCatch: Automatically Detecting Distributed Concurrency Bugs in Cloud Systems

Thank you