Towards Automatically Checking Thousands of Failures with Micro-Specifications

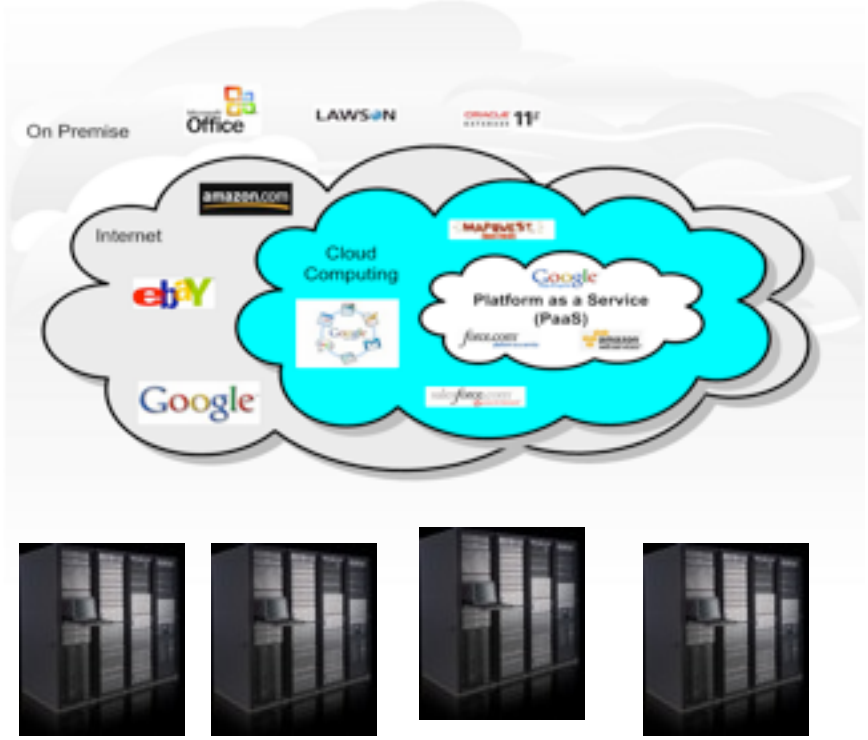
Haryadi S. Gunawi, **Thanh Do**†, Pallavi Joshi,
Joseph M. Hellerstein, Andrea C. Arpaci-Dusseau†,
Remzi H. Arpaci-Dusseau†, Koushik Sen

University of California, Berkeley
† University of Wisconsin, Madison

Cloud Era

Solve bigger human problems

Use cluster of thousands of machines



Failures in The Cloud

Failures in The Cloud

"The future is a world of **failures everywhere**" – **Garth Gibson**

3

Failures in The Cloud

"The future is a world of **failures everywhere**" - **Garth Gibson**

"Recovery must be a **first-class** operation" - **Raghu Ramakrishnan**

Failures in The Cloud

"The future is a world of **failures everywhere**" – **Garth Gibson**

"Recovery must be a **first-class** operation" – **Raghu Ramakrishnan**

"Reliability has to **come from the software**" – **Jeffrey Dean**

GIGAOM

PRO RESEARCH EVENTS

HOME APPLE CLEANTECH CLOUD COLLABORATION MOBILE VIDEO

Paisano: Ashton Kutcher Has Little Twitter Influence [STUDY] http://bit.ly/bx8uM4

Search

When the Cloud Fails: T-Mobile, Microsoft Lose Sidekick Customer Data

By Om Malik | Oct. 10, 2009, 3:57pm PDT | 32 Comments

Like Be the first of your friends to like this. Tweet 3

If you've ever been curious about what would happen when a cloud service fails, then you don't have to wonder any longer. Earlier today, customers of T-Mobile and Sidekick data services provider Danger, a subsidiary of Microsoft, lost access to all their data. Some believe that this data wipeout is because of a botched upgrade. Why it happened matters little to those who are unlikely to get their data back, according to a note posted on T-Mobile forums.

Sign up to get GigaOM news!

Your Email Address SUBSCRIBE

Like 6K

4

COMPUTERWORLD

Subscribe to a Newsletter | Solution Centers | Events | Magazine | RSS

Google™ Custom Search

Topics | News | In Depth | Reviews | Blogs | Opinion | Shark Tank | IT Jobs | White Papers | Webcasts | More

Security: App Security | Business Continuity | Cybercrime and Hacking | DRM and Legal Issues | Data Security | Malware and Vulnerabilities | Privacy | Security Hardware and Software

Home > Security > Business Continuity

News

Facebook temporarily loses more than 10% of photos in hard drive failure

Facebook is working to restore access to the photos

By Lucas Mearian
March 9, 2009 12:00 PM ET

Comments (5) Recommended (33) Share

Computerworld - Popular social networking site Facebook.com admitted in a blog post today that over the weekend, a hard drive failure led to the temporary loss of 10% to 15% of the photographs stored by its users.

According to the company, several drives failed at once during a routine upgrade Friday night.

"You may have noticed in the past day that some photos aren't appearing or are displaying a 'question mark' graphic when you go to view them. We're trying to fully understand what happened, since simultaneous hardware failures like this are rare," Evan Priestley, an engineer at Facebook

Top Stories

- Iran confirms massive Stuxnet infection of industrial systems
- Visual tour: Mozilla's Seabird concept phone
- Elgan: Why do tech CEOs have to be nice?
- Chinese Apple fans line up for iPhone 4 launch

Why Failure Recovery Hard?

- Testing is not advanced enough against complex failures
  - Diverse, frequent, and multiple failures
  - FaceBook photo loss
- Recovery is under specified
  - Need to specify failure recovery behaviors
  - Customized well-grounded protocols
- Example: Paxos made live – An engineering perspective [PODC' 07]

Our Solutions

7

Our Solutions

- **FTS ("FATE")** – Failure Testing Service
  - New abstraction for failure exploration
  - Systematically exercise **40,000** unique combinations of failures

7

Our Solutions

- **FTS ("FATE")** – Failure Testing Service
  - New abstraction for failure exploration
  - Systematically exercise **40,000** unique combinations of failures
- **DTS ("DESTINI")** – Declarative Testing Specification
  - Enable concise recovery specifications
  - We have written **74** checks (3 lines / check)

7

Our Solutions

- **FTS ("FATE")** – Failure Testing Service
  - New abstraction for failure exploration
  - Systematically exercise **40,000** unique combinations of failures
- **DTS ("DESTINI")** – Declarative Testing Specification
  - Enable concise recovery specifications
  - We have written **74** checks (3 lines / check)
- Note: Names have changed since the paper

7

Summary of Findings

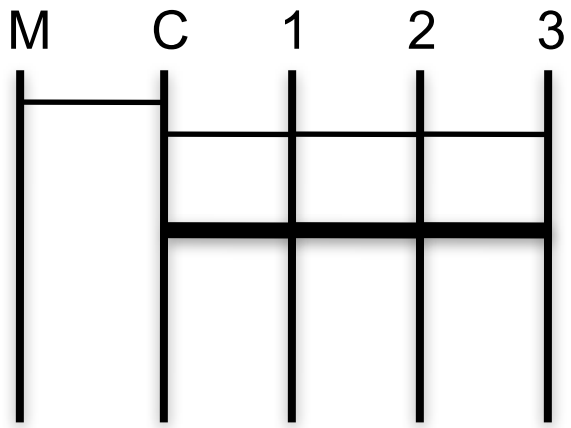
- Applied FATE and DESTINI to three cloud systems: HDFS, ZooKeeper, Cassandra
- Found 16 new bugs
- Reproduced 74 bugs
- Problems found
  - Inconsistency
  - Data loss
  - Rack awareness broken
  - Unavailability

8

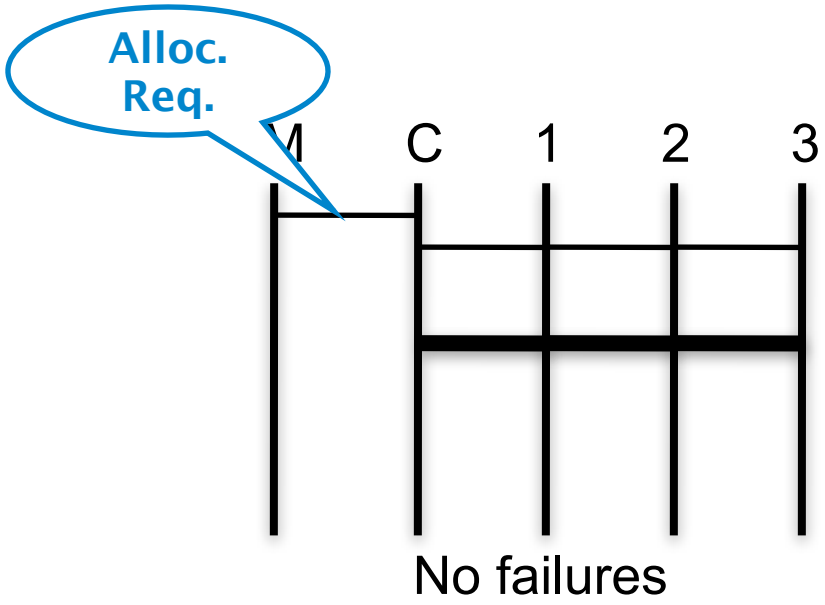
Outline

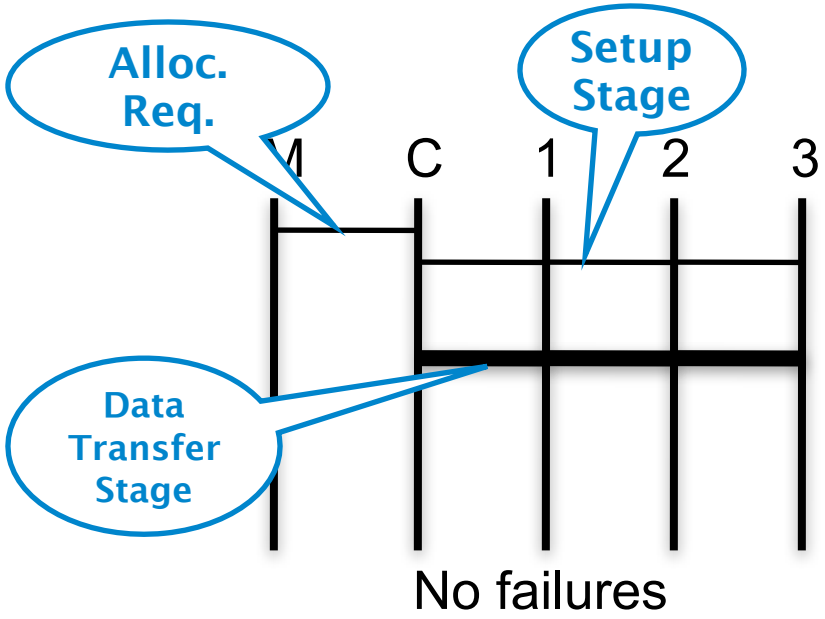
- ✓ Introduction
- FATE
- DESTINI
- Evaluation
- Summary

9

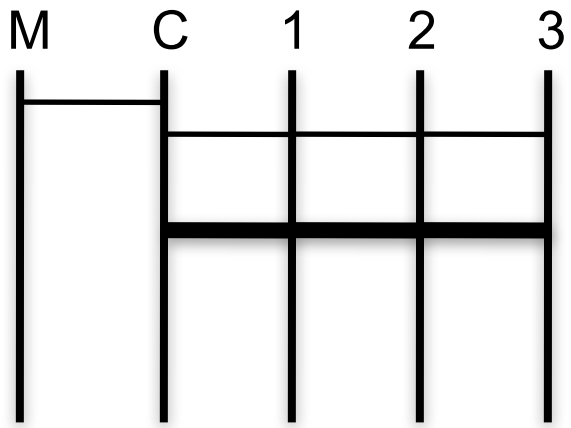


No failures

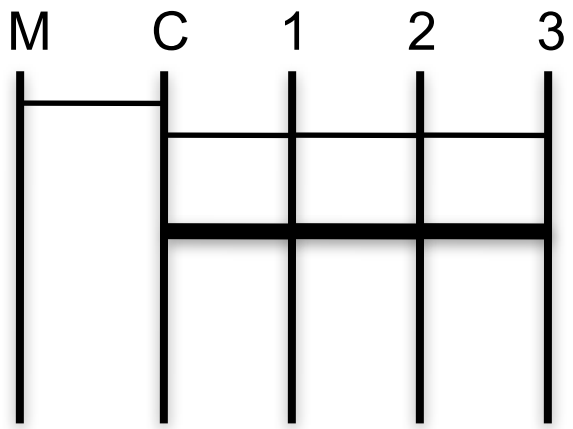

Alloc.
Req.
M
C
1
2
3
No failures


Alloc.
Req.
Setup
Stage
M
C
1
2
3
Data
Transfer
Stage
No failures

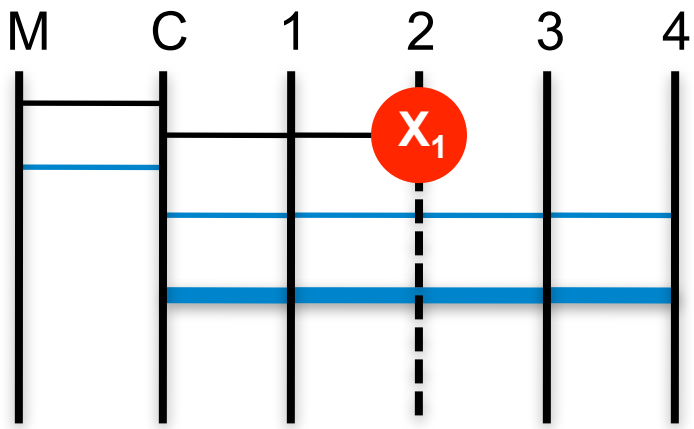
10



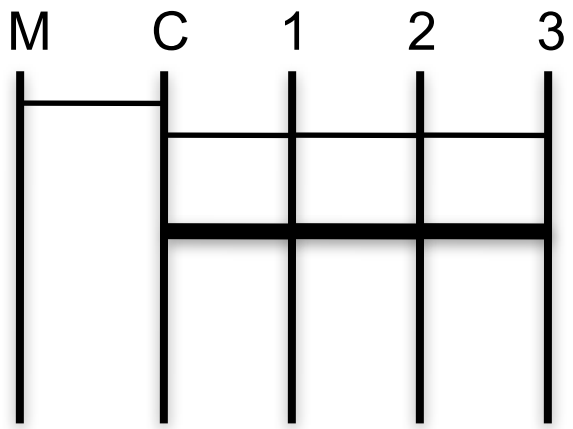
No failures



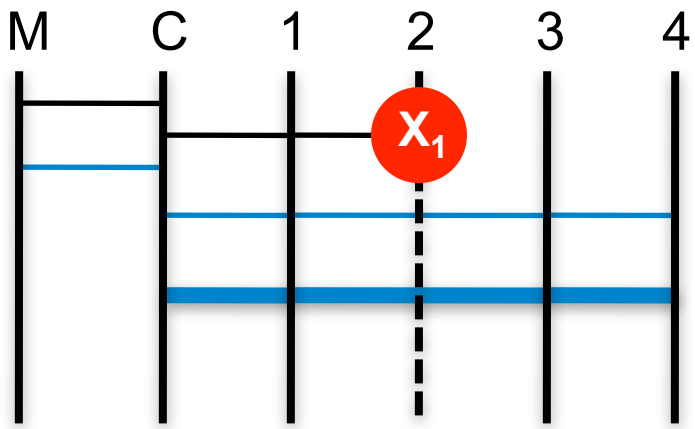
No failures



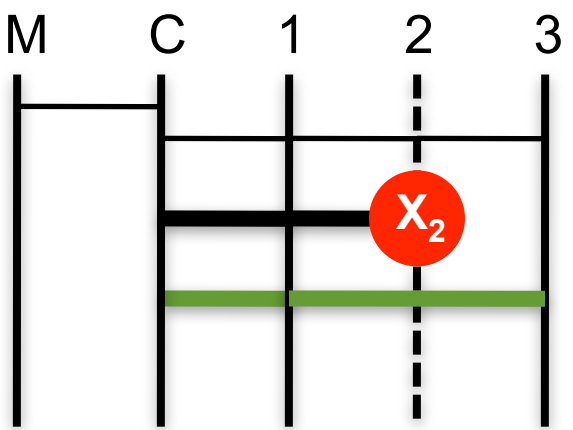
Setup Stage Recovery:
Recreate fresh pipeline



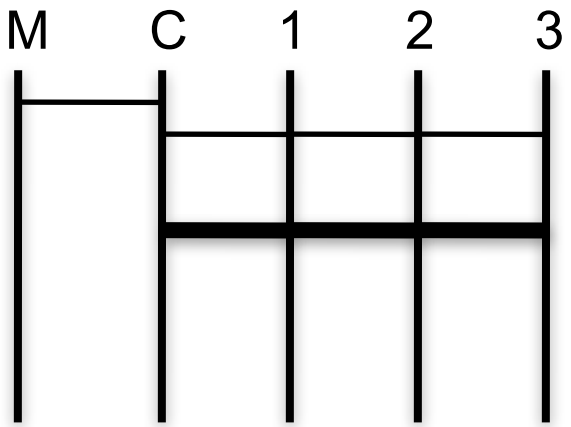
No failures



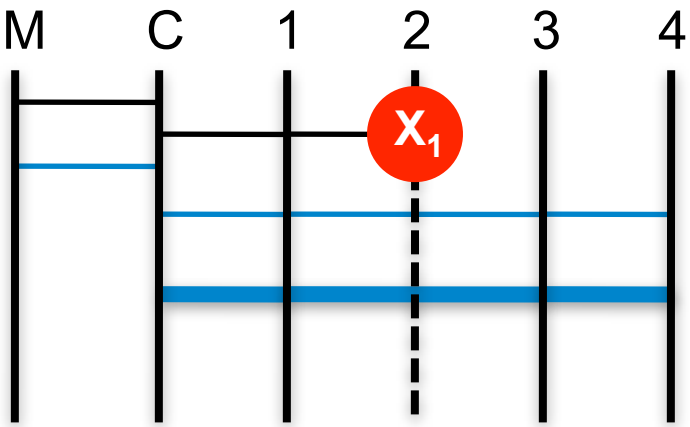
Setup Stage Recovery:
Recreate fresh pipeline



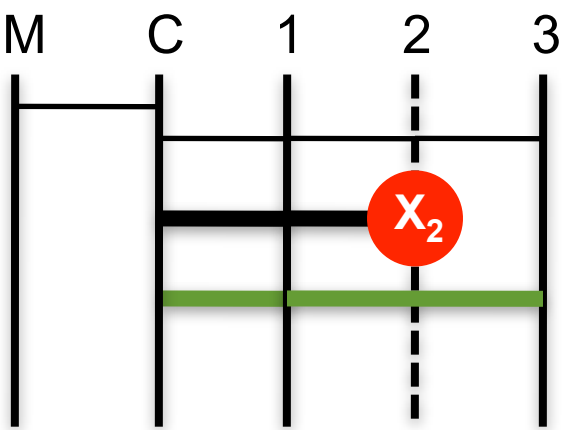
Data transfer Stage Recovery:
Continue on surviving nodes



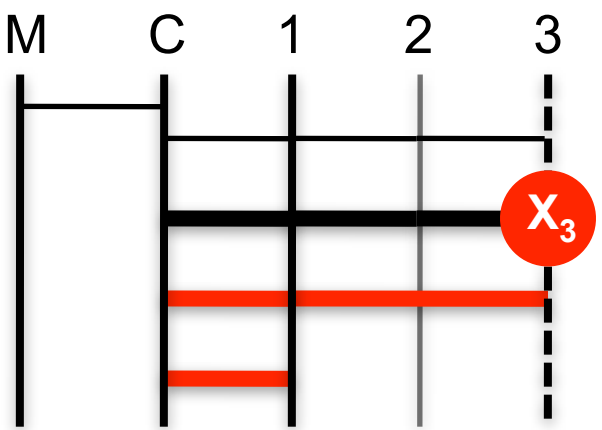
No failures



Setup Stage Recovery:
Recreate fresh pipeline

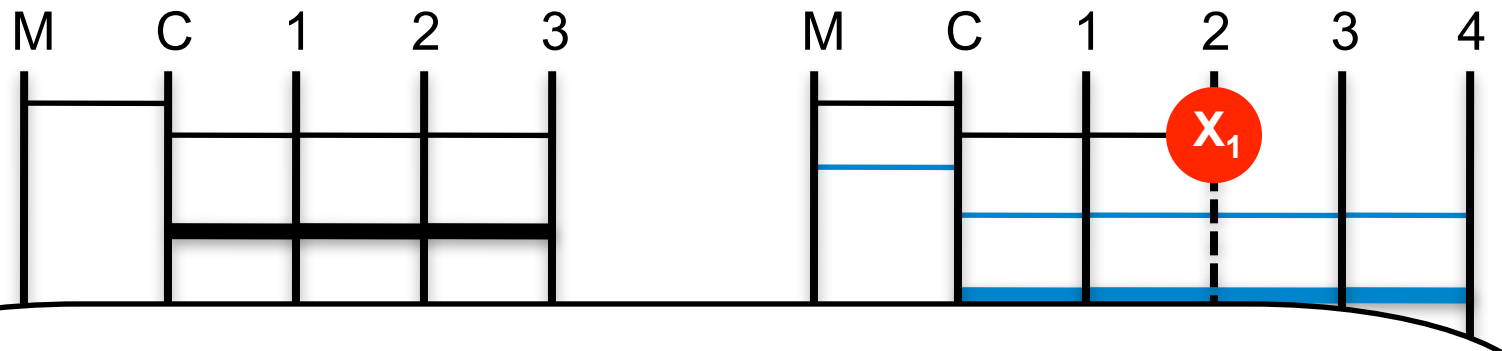


Data transfer Stage Recovery:
Continue on surviving nodes



Bug in Data Transfer Stage Recovery

10



Failures at
DIFFERENT STAGES
lead to
DIFFERENT FAILURE BEHAVIORS

Goal: Exercise different failure recovery path

Data transfer Stage Recovery: Continue on surviving nodes

Bug in Data Transfer Stage Recovery

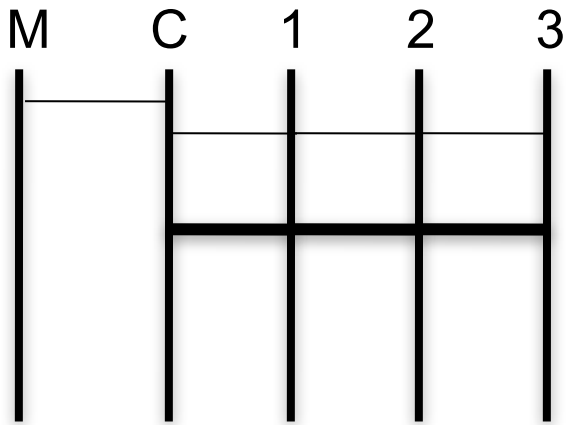
FATE

- A failure injection framework
  - target IO points
  - Systematically exploring failure
  - Multiple failures
- New abstraction of failure scenario
  - Remember injected failures
  - Increase failure coverage

11

FATE

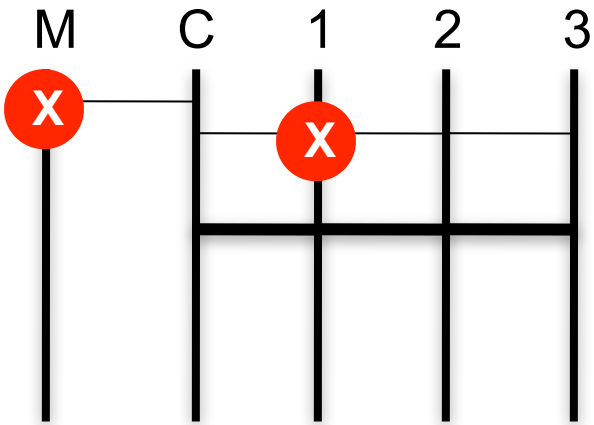
- A failure injection framework
  - target IO points
  - Systematically exploring failure
  - Multiple failures
- New abstraction of failure scenario
  - Remember injected failures
  - Increase failure coverage



11

FATE

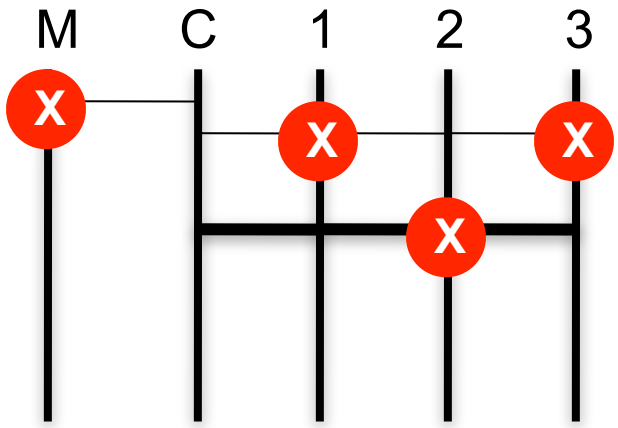
- A failure injection framework
  - target IO points
  - Systematically exploring failure
  - Multiple failures
- New abstraction of failure scenario
  - Remember injected failures
  - Increase failure coverage



11

FATE

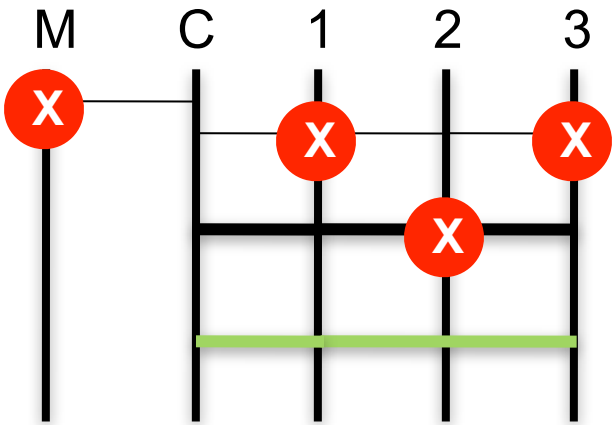
- A failure injection framework
  - target IO points
  - Systematically exploring failure
  - Multiple failures
- New abstraction of failure scenario
  - Remember injected failures
  - Increase failure coverage



11

FATE

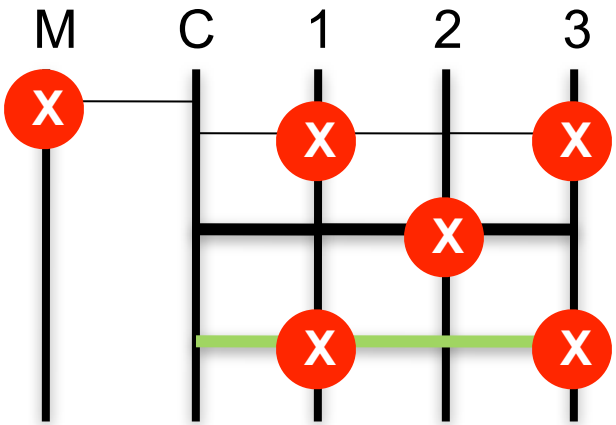
- A failure injection framework
  - target IO points
  - Systematically exploring failure
  - Multiple failures
- New abstraction of failure scenario
  - Remember injected failures
  - Increase failure coverage



11

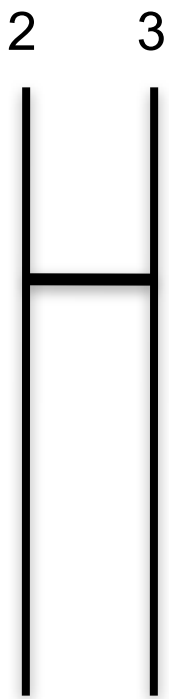
FATE

- A failure injection framework
  - target IO points
  - Systematically exploring failure
  - Multiple failures
- New abstraction of failure scenario
  - Remember injected failures
  - Increase failure coverage



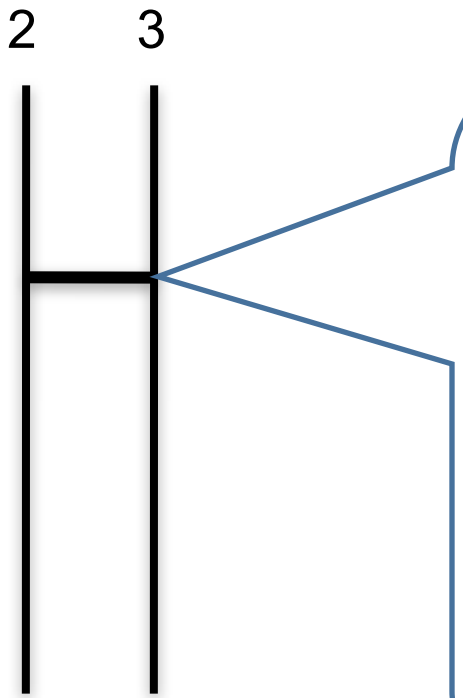
11

Failure ID



12

Failure ID



Fields		Values
Static	Func. Call	OutputStream.read()
	Source File	BlockReceiver.java
Dynamic	Stack Track	…
Domain specific	Source	Node 2
	Destination	Node 3
	Net. Message	Data Packet
Failure	Type	Crash After
Hash	**12348729**	

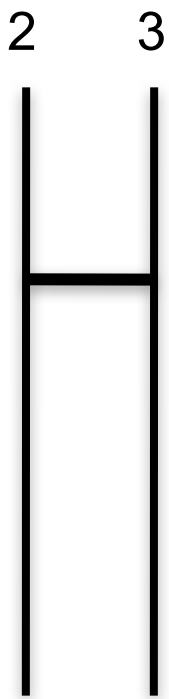
12

How Developers Build Failure ID?

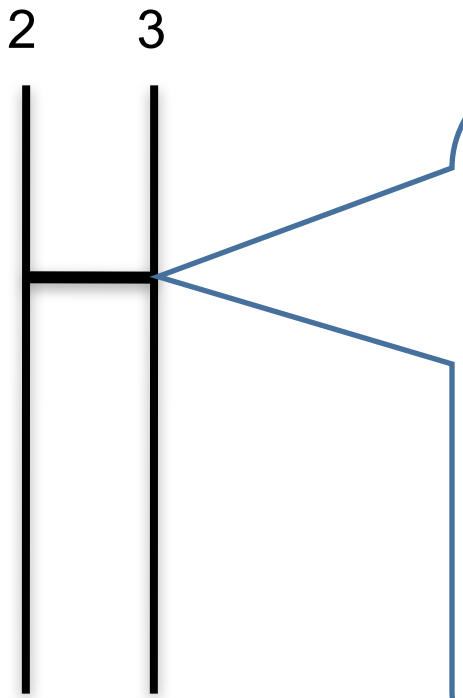
- FATE intercepts all I/Os
- Use aspectJ to collect information at every I/O point
  - I/O buffers (e.g file buffer, network buffer)
  - Target I/O (e.g. file name, IP address)
- Reverse engineer for domain specific information

13

Failure ID

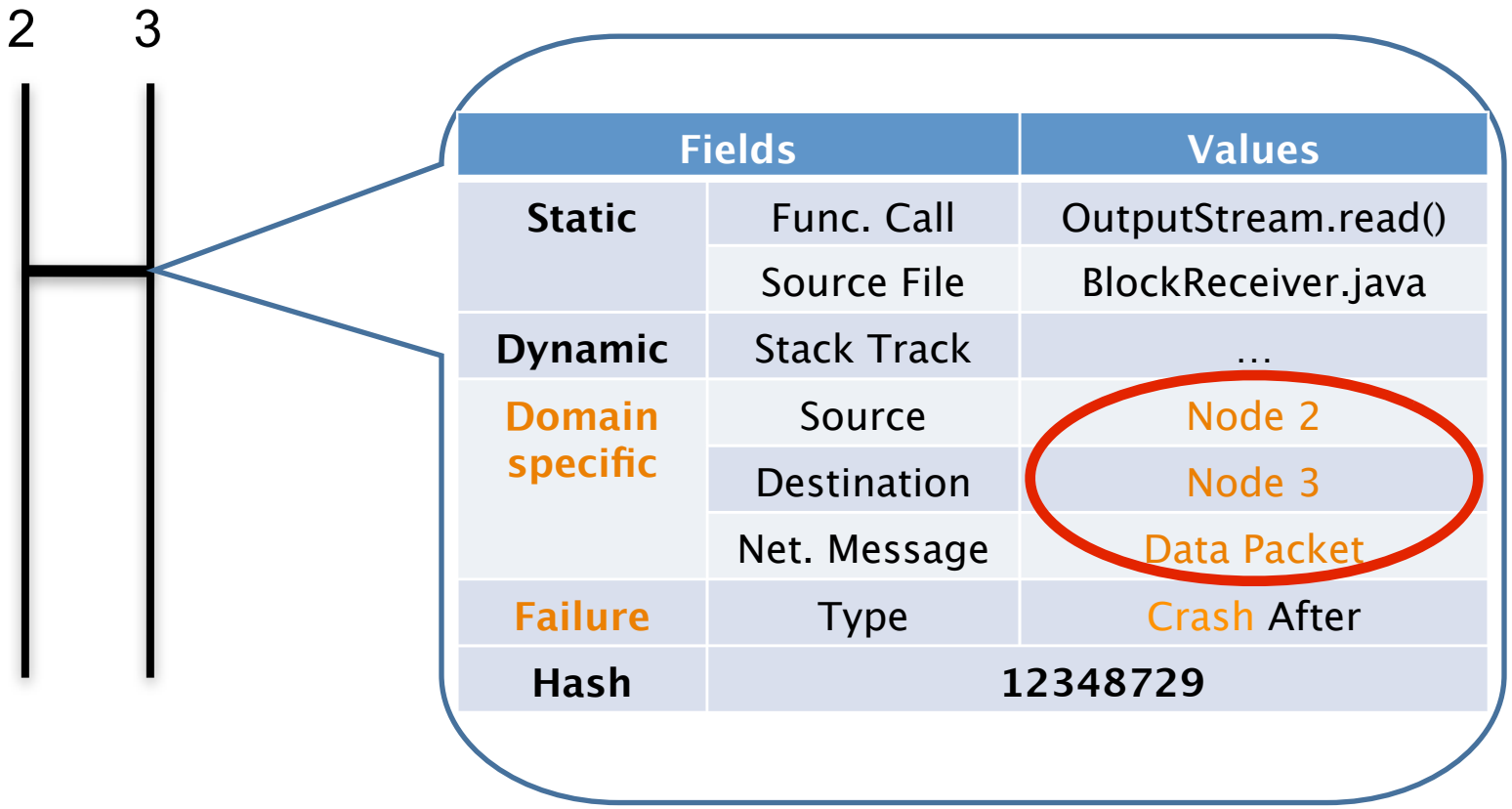


Failure ID



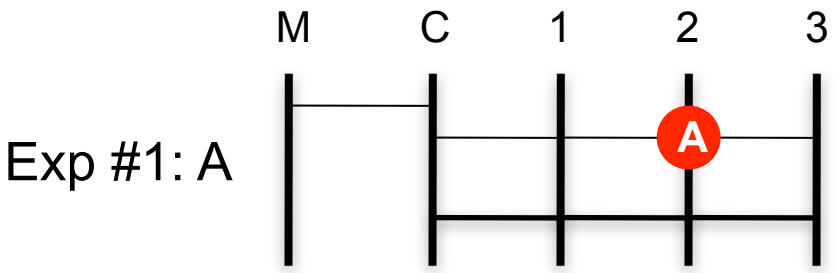
Fields		Values
Static	Func. Call	OutputStream.read()
	Source File	BlockReceiver.java
Dynamic	Stack Track	…
Domain specific	Source	Node 2
	Destination	Node 3
	Net. Message	Data Packet
Failure	Type	Crash After
Hash	**12348729**	

Failure ID

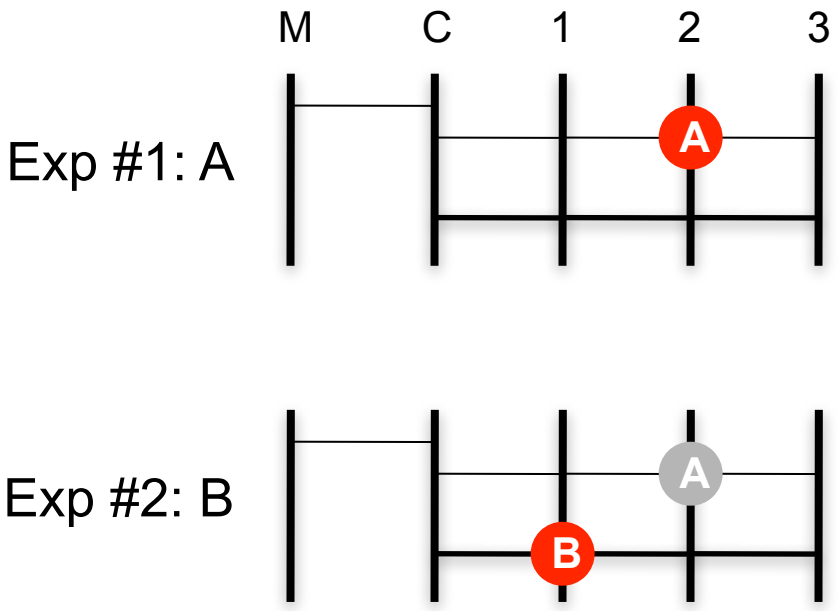

2
3
Fields
Values
Static
Func. Call
OutputStream.read()
Source File
BlockReceiver.java
Dynamic
Stack Track
…
Domain specific
Source
Node 2
Destination
Node 3
Net. Message
Data Packet
Failure
Type
Crash After
Hash
12348729

Exploring Failure Space

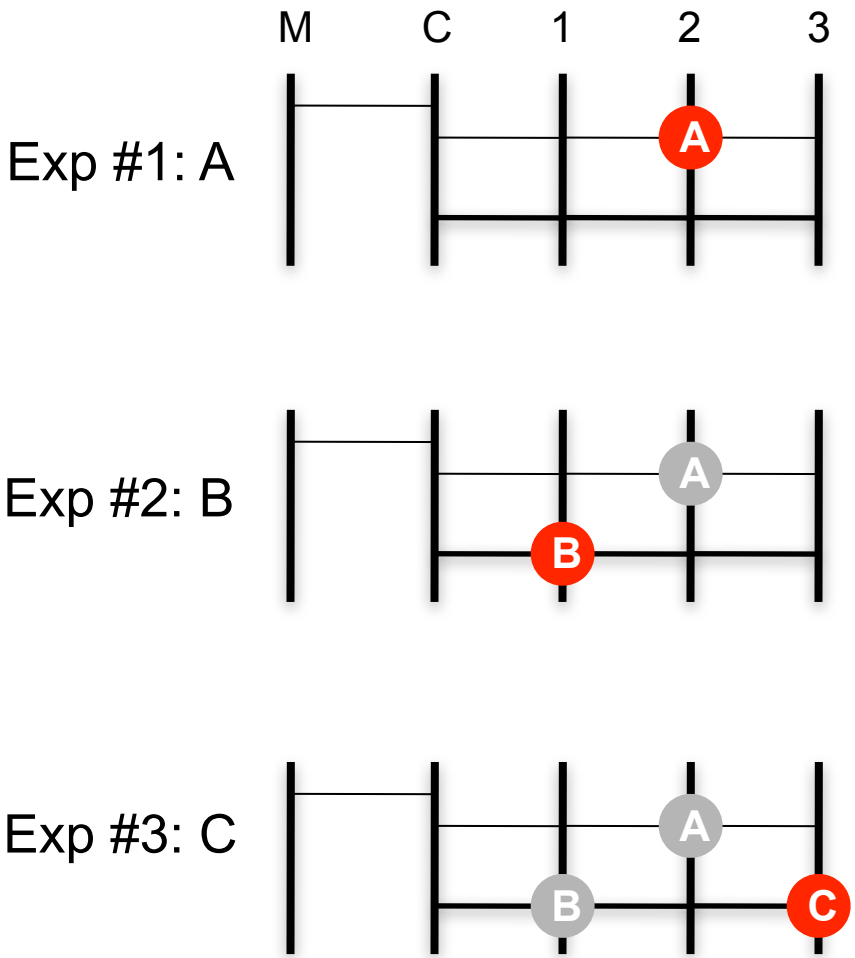
Exploring Failure Space



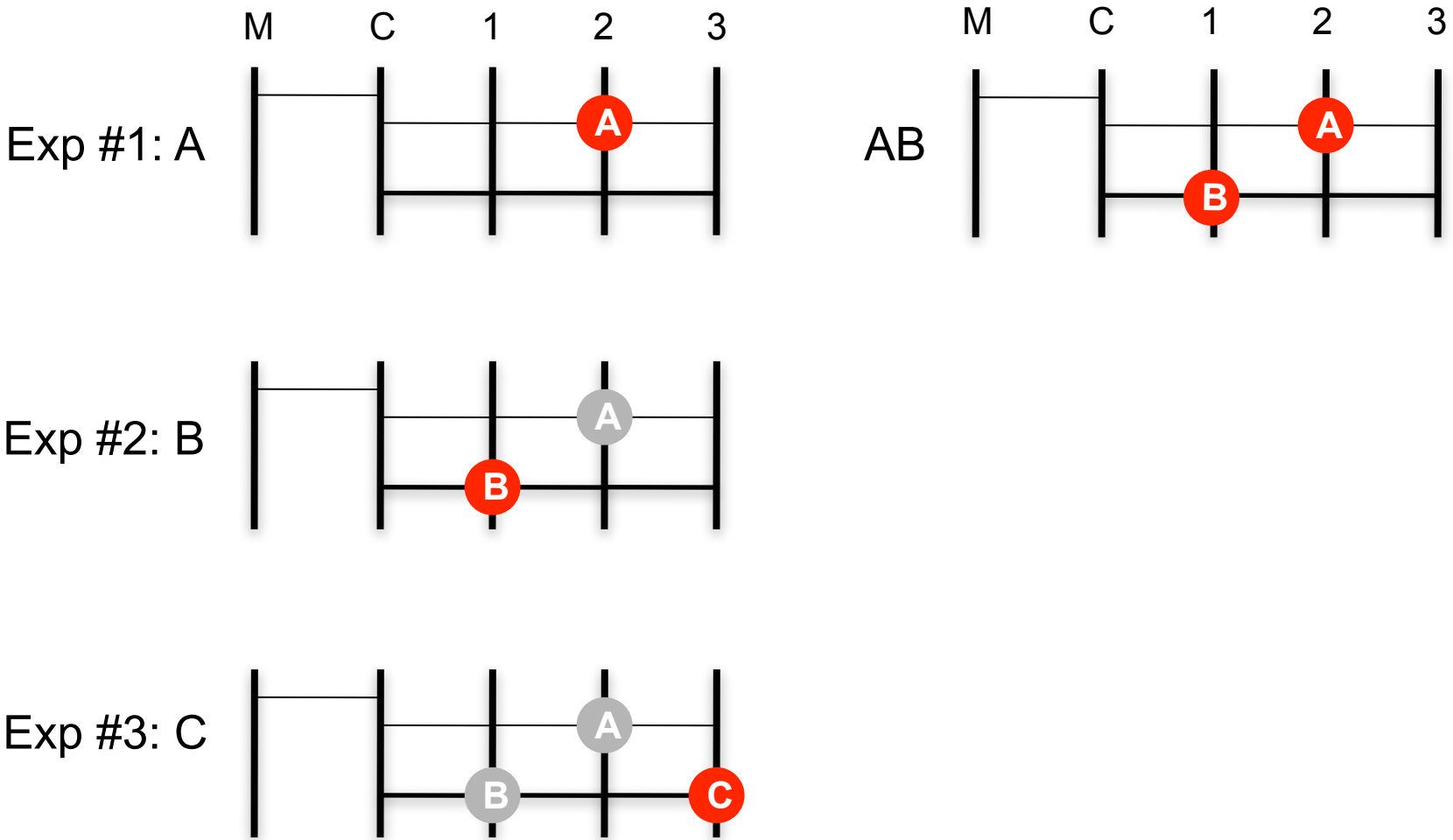
Exploring Failure Space



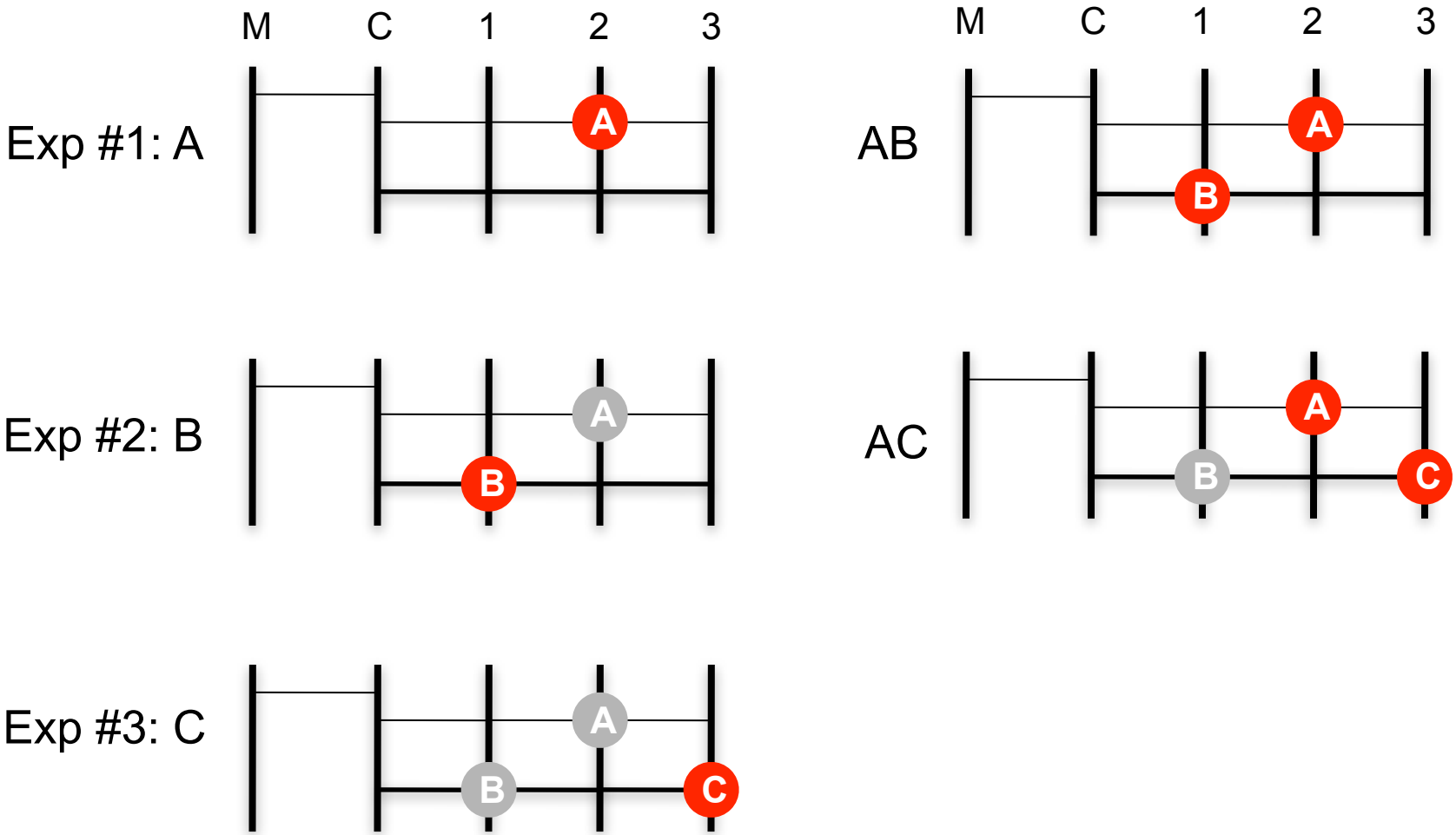
Exploring Failure Space



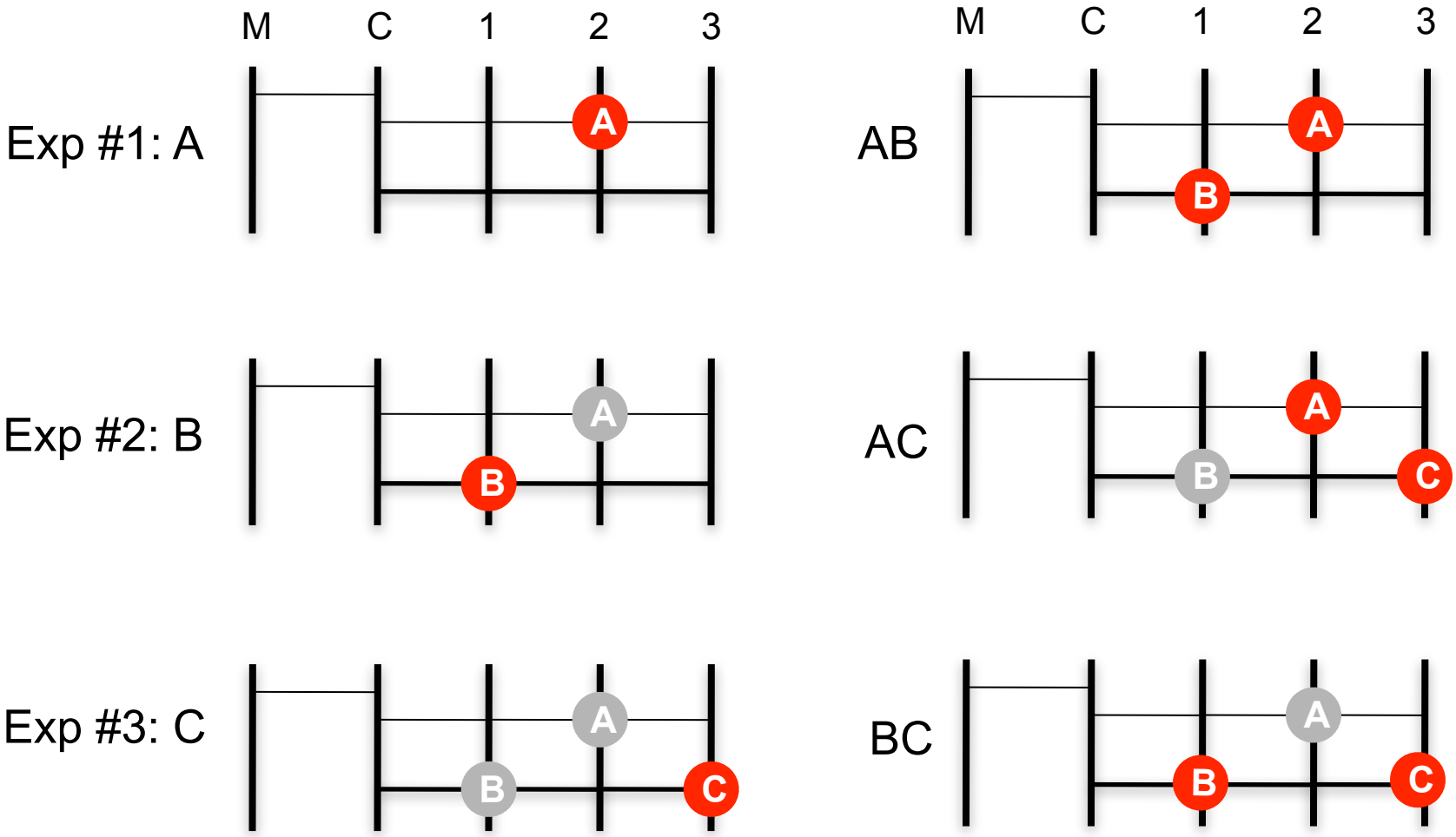
Exploring Failure Space



Exploring Failure Space



Exploring Failure Space



Outline

- ✓ Introduction
- ✓ FATE
- DESTINI
- Evaluation
- Summary

DESTINI

- Enable concise recovery specifications
- Check if expected behaviors match with **actual** behaviors
- Important elements:
  - Expectations
  - **Facts**
  - Failure Events
  - Check Timing
- Interpose network and disk protocols

Writing specifications

Writing specifications

"Violation if expectation is different from actual facts"

Writing specifications

"Violation if expectation is different from actual facts"

Writing specifications

"Violation if expectation is different from actual facts"

violationTable():- expectationTable(), **NOT-IN** actualTable()

Writing specifications

"Violation if expectation is different from actual facts"

violationTable():- expectationTable(), **NOT-IN** actualTable()

Writing specifications

"Violation if expectation is different from actual facts"

violationTable():- expectationTable(), **NOT-IN** actualTable()

DataLog syntax:

Writing specifications

"Violation if expectation is different from actual facts"

violationTable():- expectationTable(), **NOT-IN** actualTable()

DataLog syntax:

:- derivation

Writing specifications

"Violation if expectation is different from actual facts"

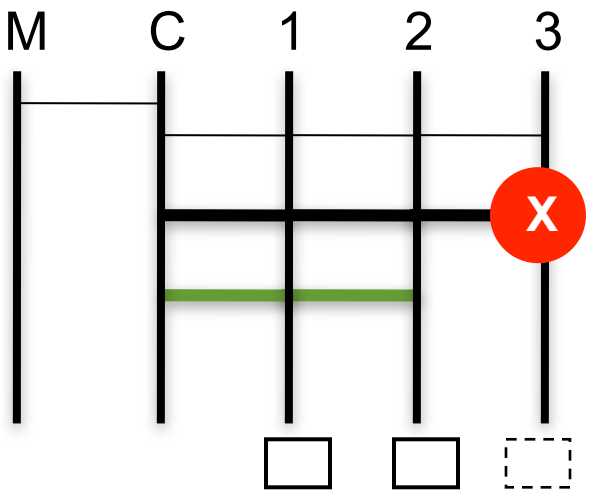
violationTable():- expectationTable(), **NOT-IN** actualTable()

DataLog syntax:

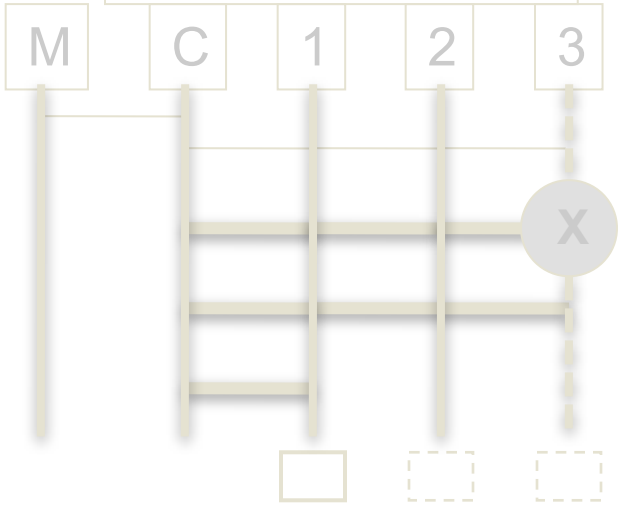
:- derivation

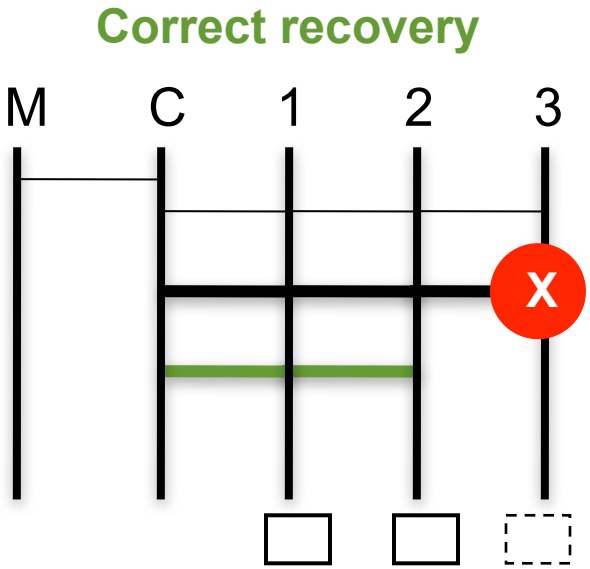
, AND

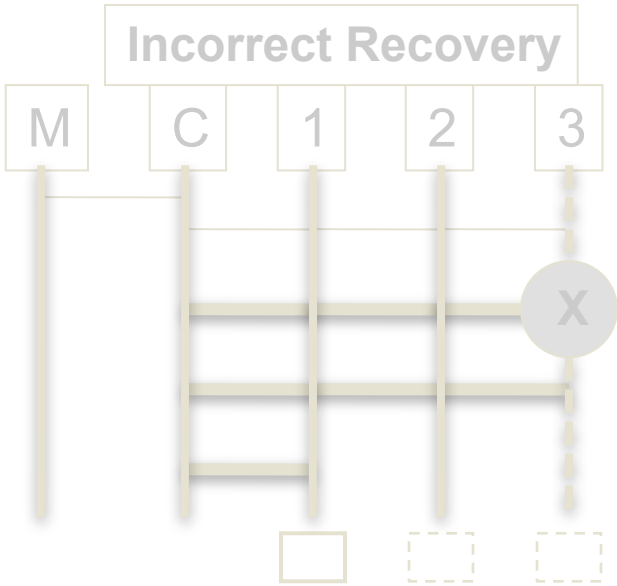
Correct recovery



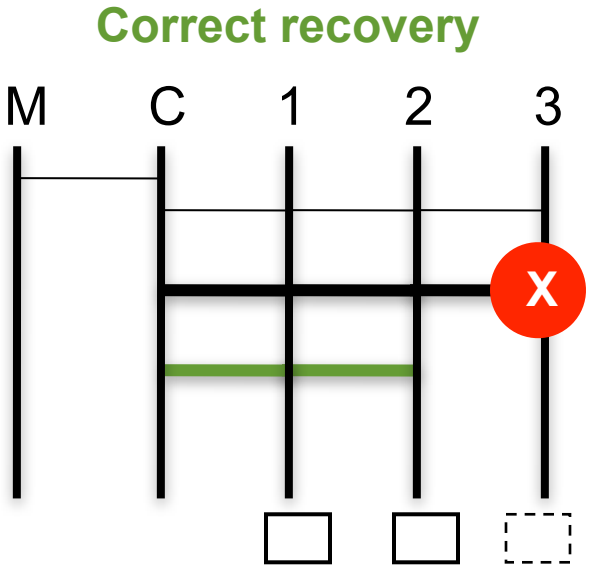
Incorrect Recovery

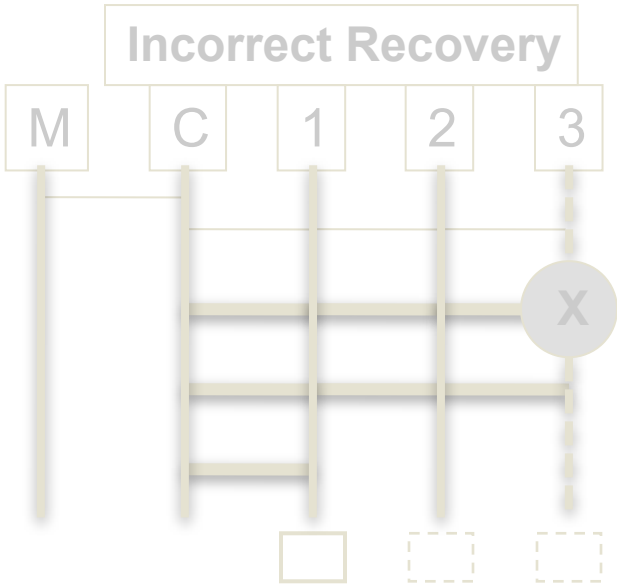






incorrectNodes(B, N) :- **expectedNodes**(B, N), NOT-IN **actualNodes**(B, N);

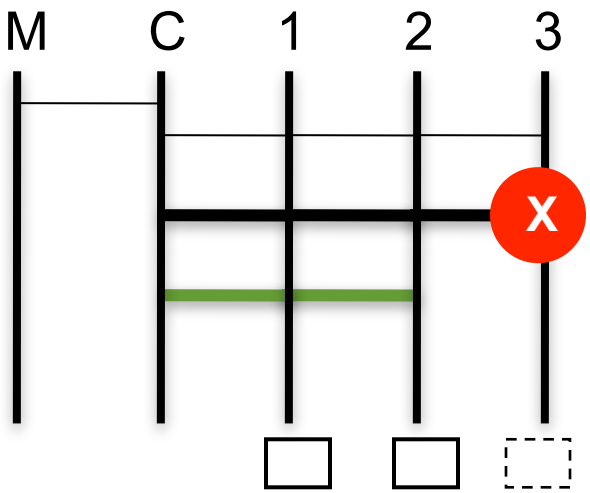




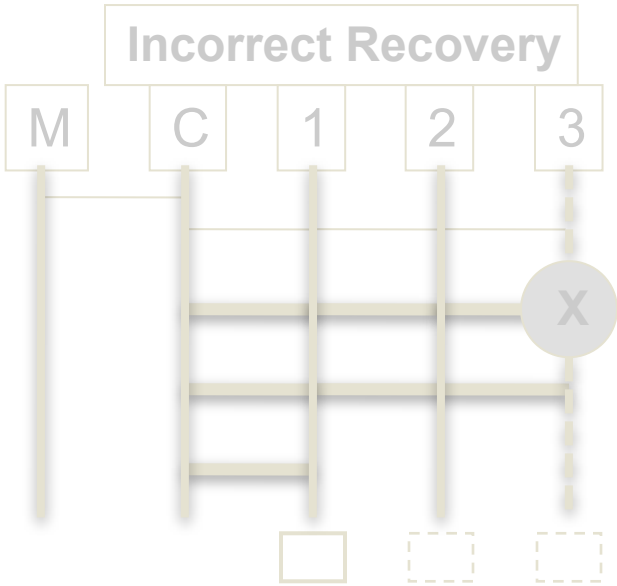
Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

incorrectNodes(B, N) :- **expectedNodes**(B, N), NOT-IN **actualNodes**(B, N);

Correct recovery



Incorrect Recovery

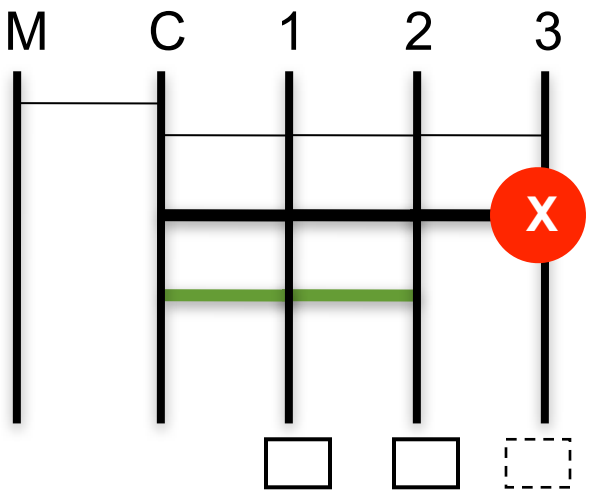


Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

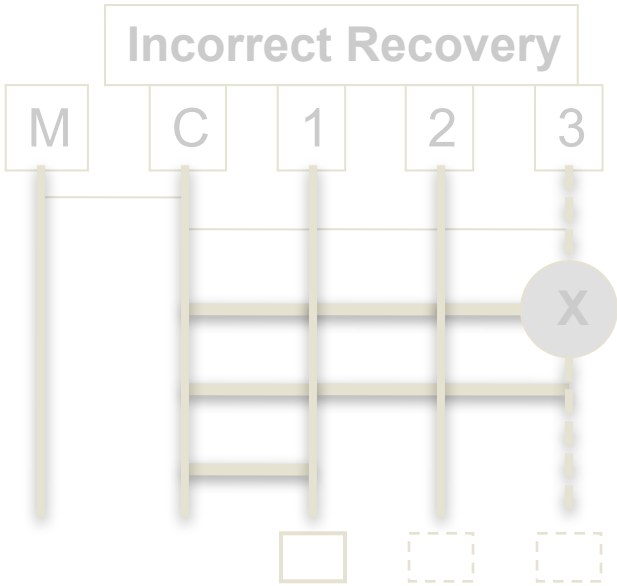
actualNodes(Block, Node)	
B	Node 1
B	Node 2

incorrectNodes(B, N) :- **expectedNodes**(B, N), NOT-IN **actualNodes**(B, N);

Correct recovery



Incorrect Recovery

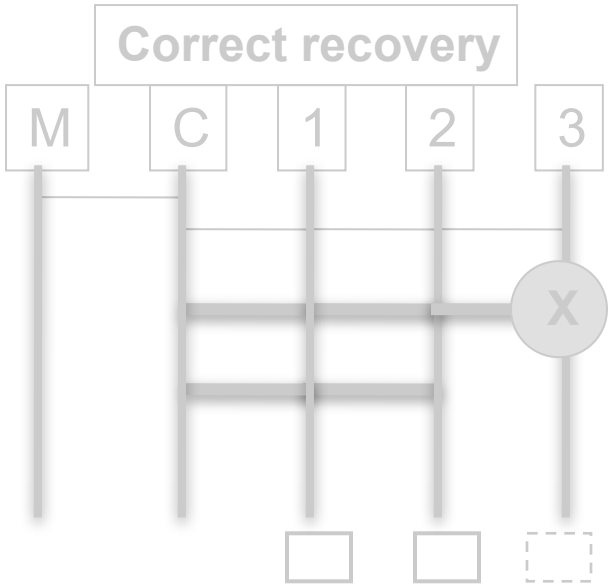


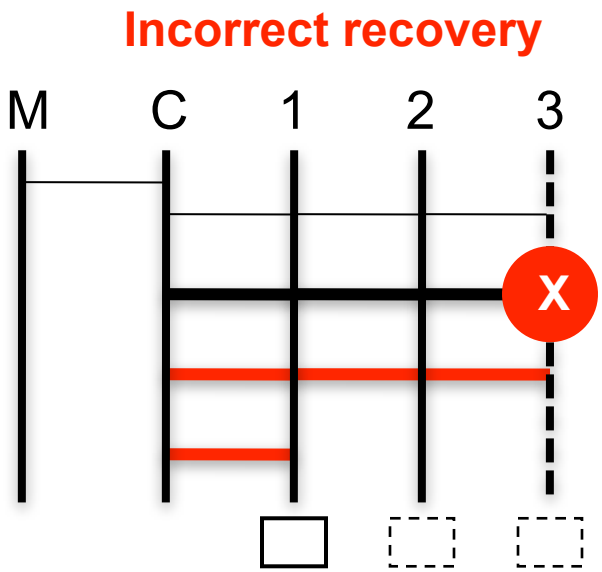
IncorrectNodes (Block, Node)	

Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

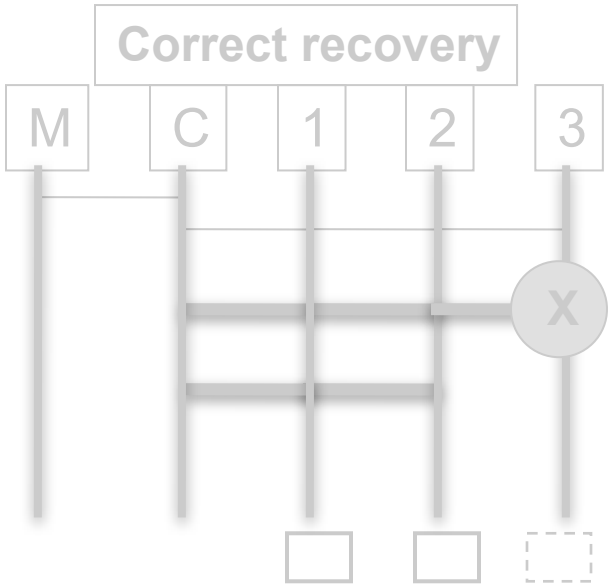
actualNodes(Block, Node)	
B	Node 1
B	Node 2

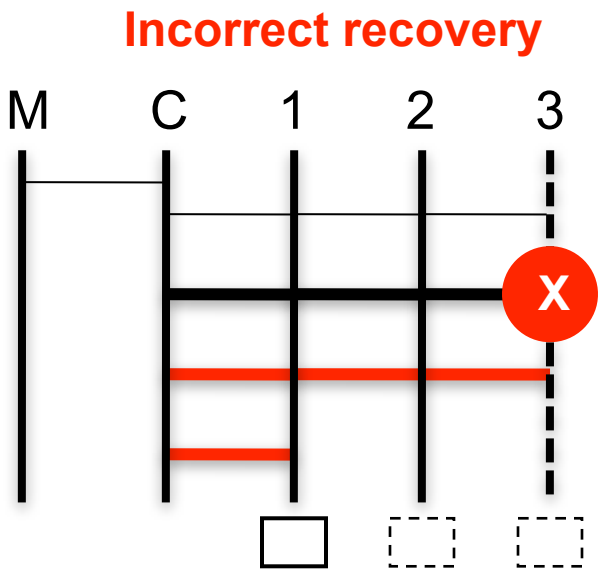
incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);





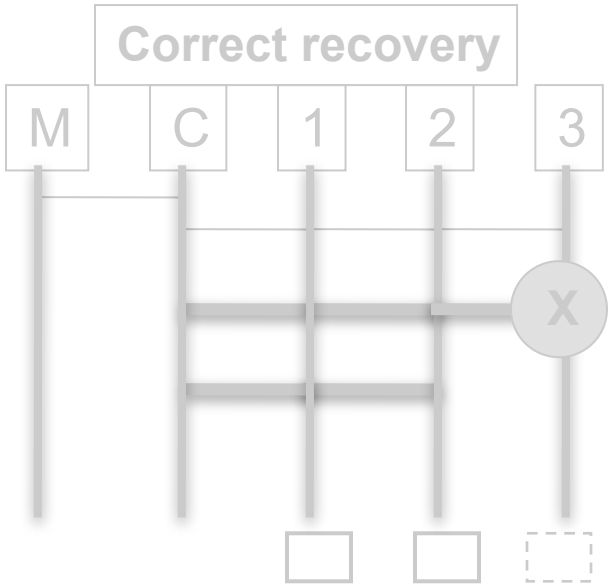
incorrectNodes(B, N) :- **expectedNodes**(B, N), NOT-IN **actualNodes**(B, N);

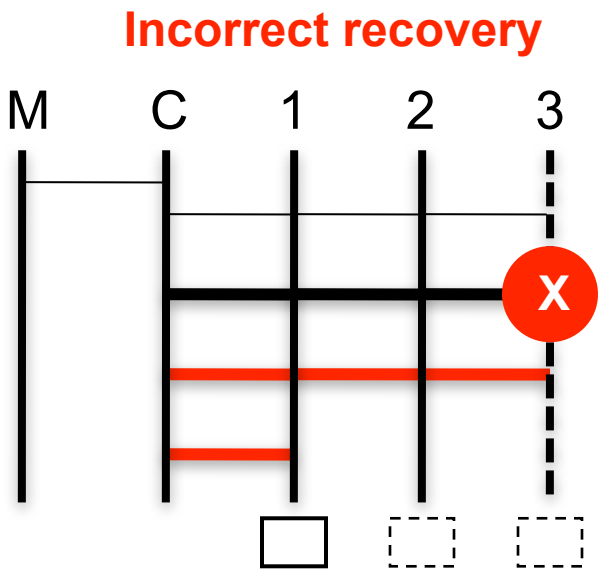




Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

incorrectNodes(B, N) :- **expectedNodes**(B, N), NOT-IN **actualNodes**(B, N);

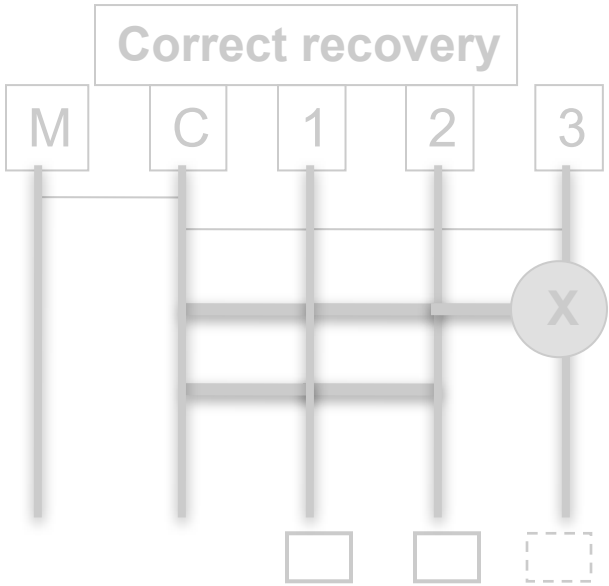


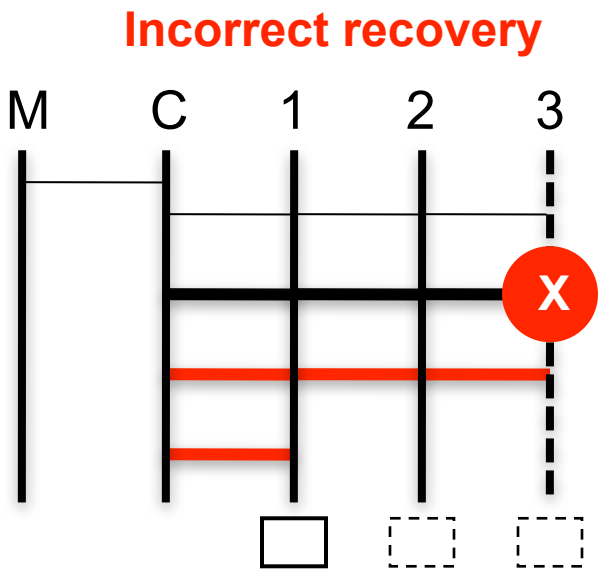


Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

actualNodes(Block, Node)	
B	Node 1

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);



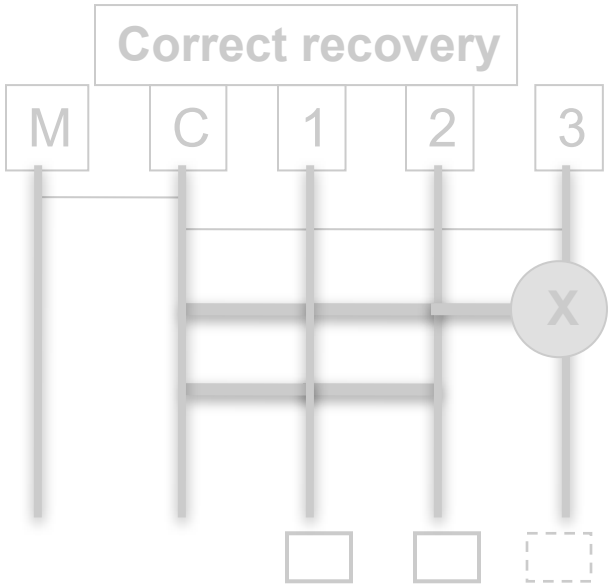


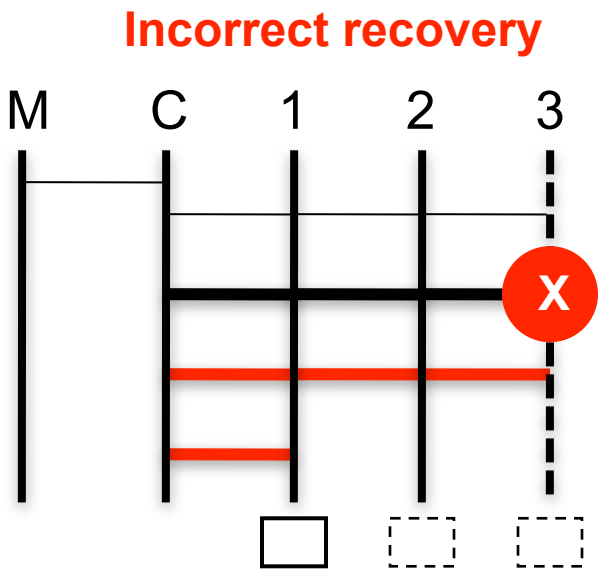
IncorrectNodes (Block, Node)	
B	Node 2

Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

actualNodes(Block, Node)	
B	Node 1

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);



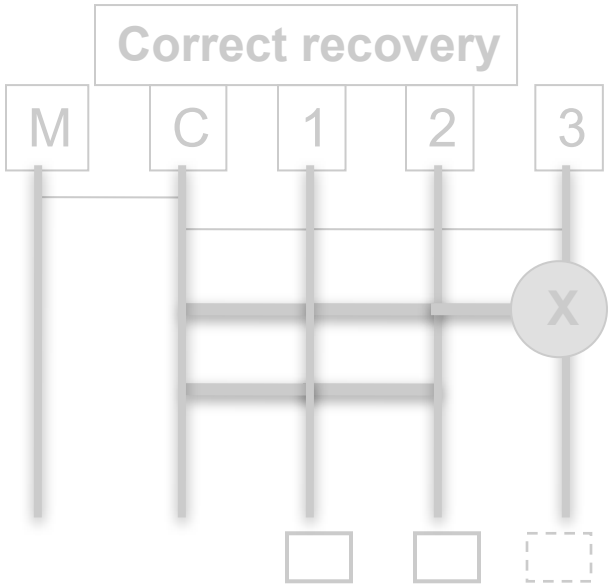


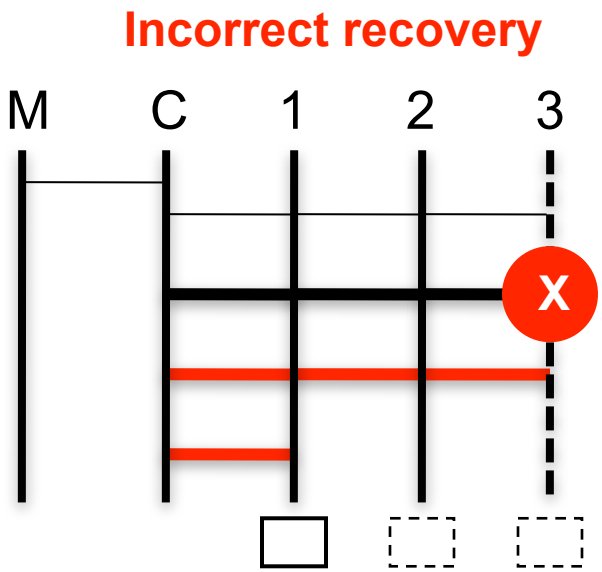
IncorrectNodes (Block, Node)	
B	Node 2

Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

actualNodes(Block, Node)	
B	Node 1

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);





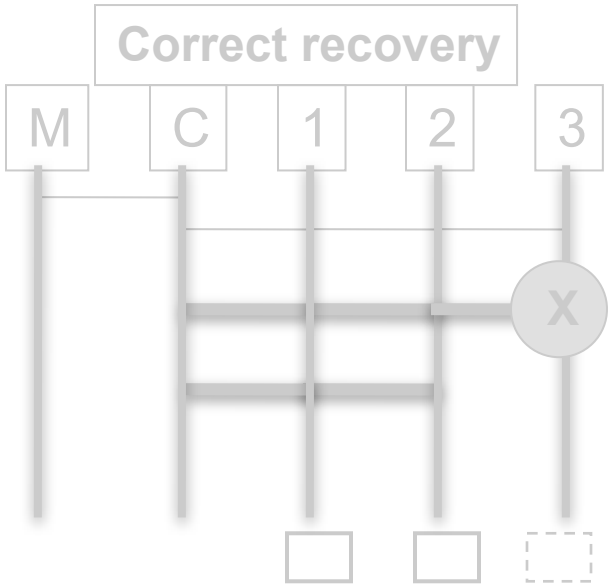
IncorrectNodes (Block, Node)	
B	Node 2

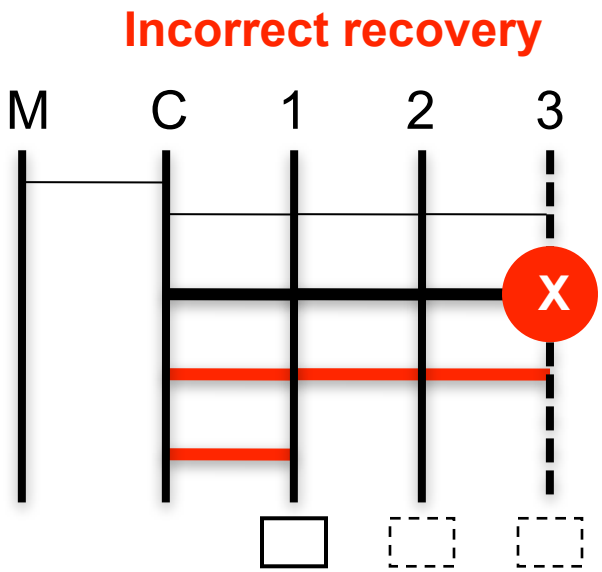
Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

actualNodes(Block, Node)	
B	Node 1

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);

BUILD EXPECTATIONS





IncorrectNodes (Block, Node)	
B	Node 2

Expected Nodes (Block, Node)	
B	Node 1
B	Node 2

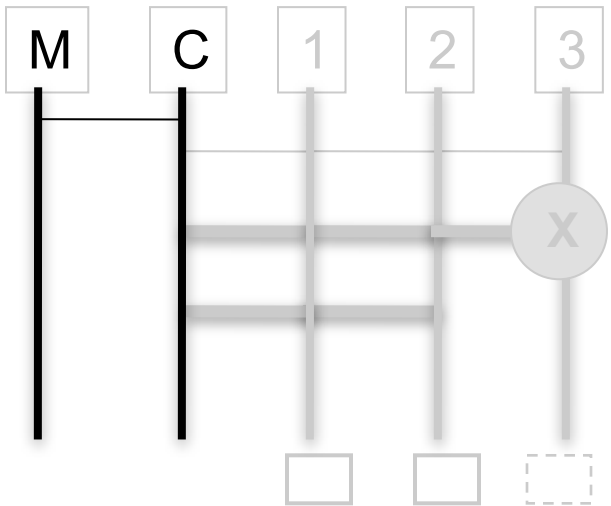
actualNodes(Block, Node)	
B	Node 1

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N);

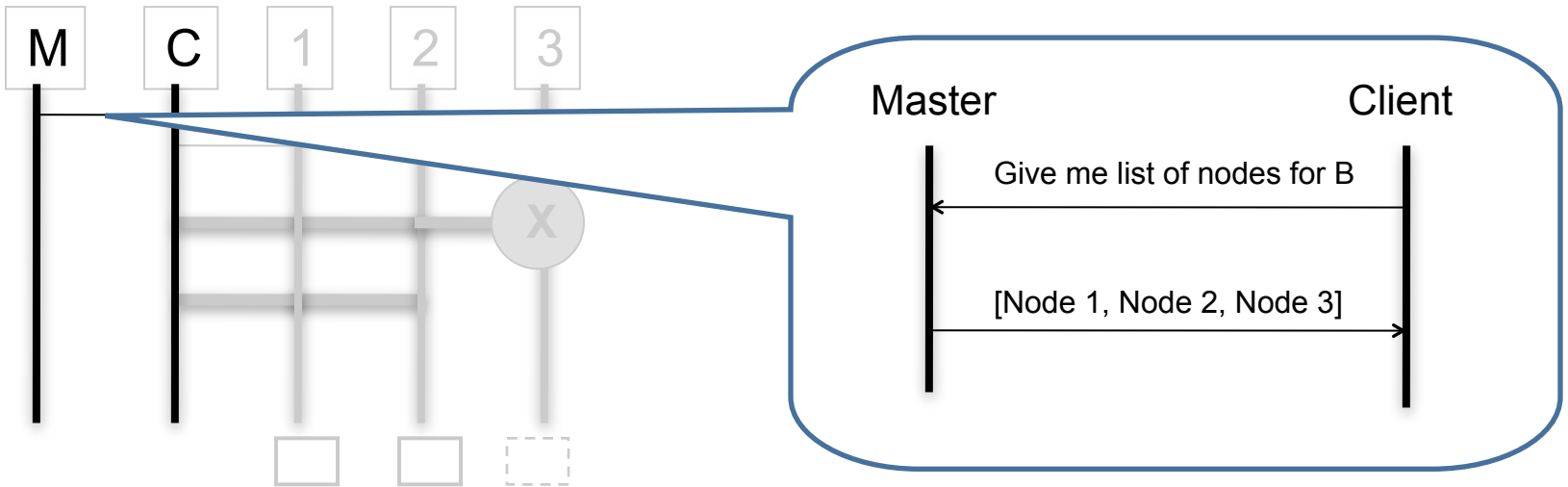
BUILD EXPECTATIONS

CAPTURE FACTS

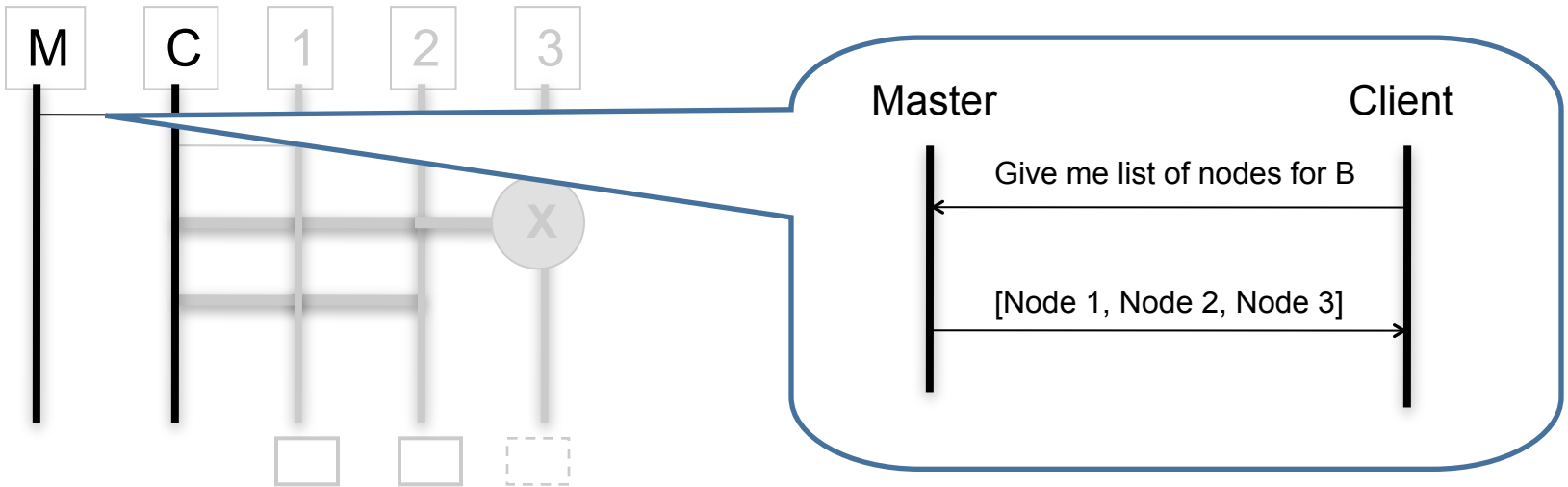
Building Expectations



Building Expectations

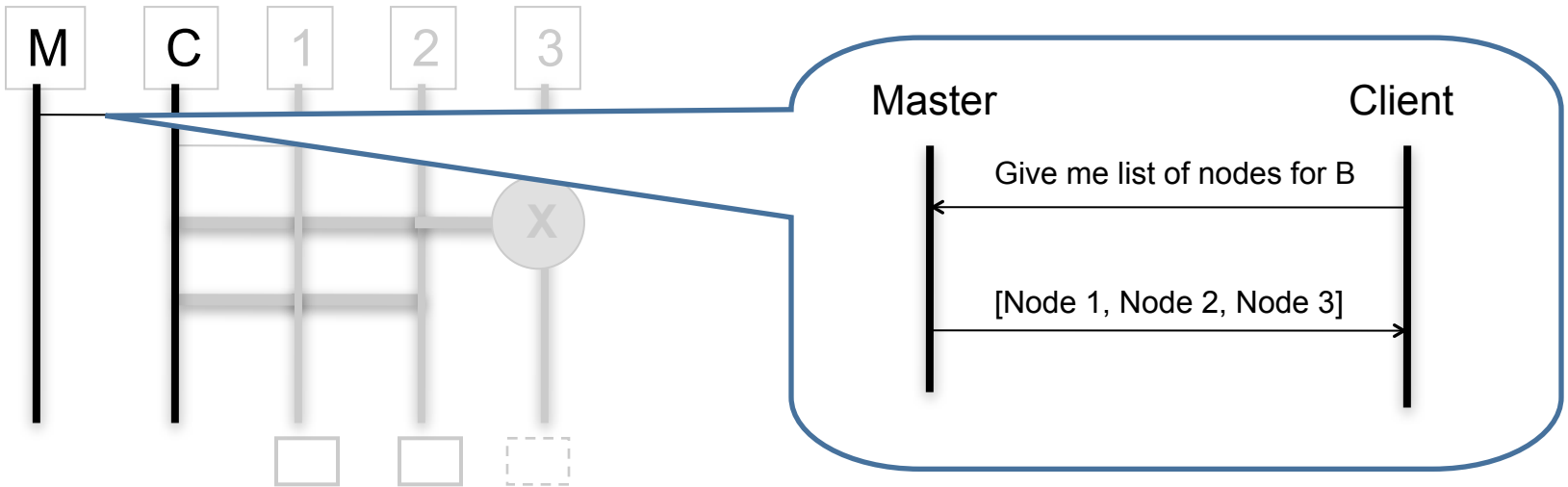


Building Expectations



expectedNodes(B, N) :- getBlockPipe(B, N);

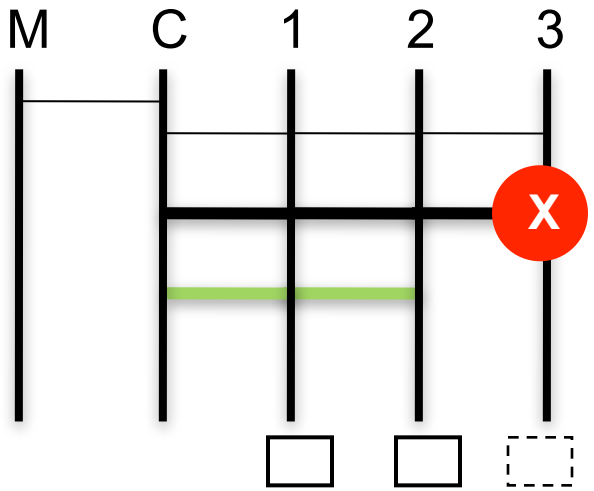
Building Expectations



expectedNodes(B, N) :- getBlockPipe(B, N);

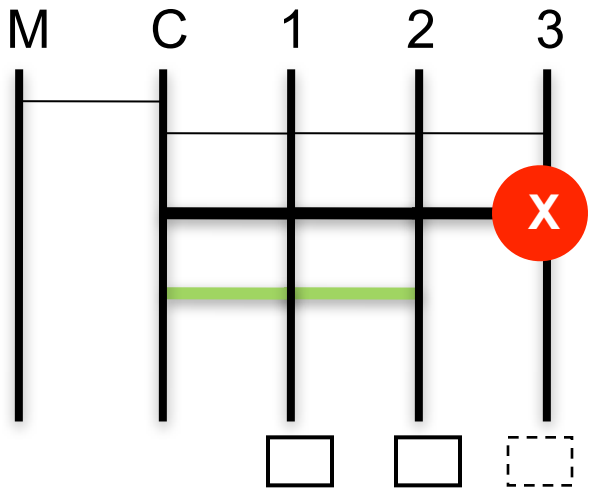
Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

Updating Expectation



Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

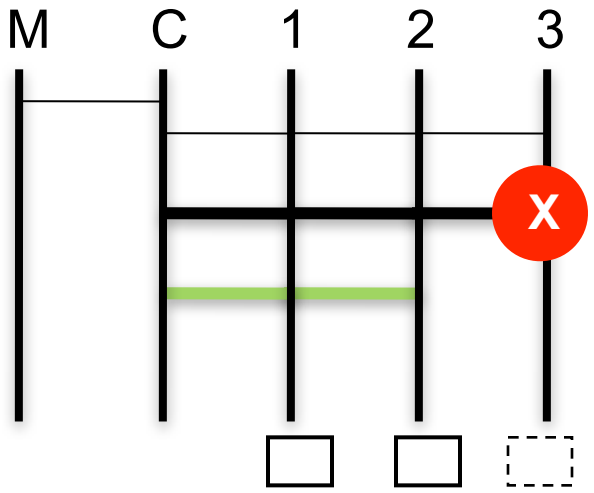
Updating Expectation



Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = “Data Transfer”, expectedNode(B, N)

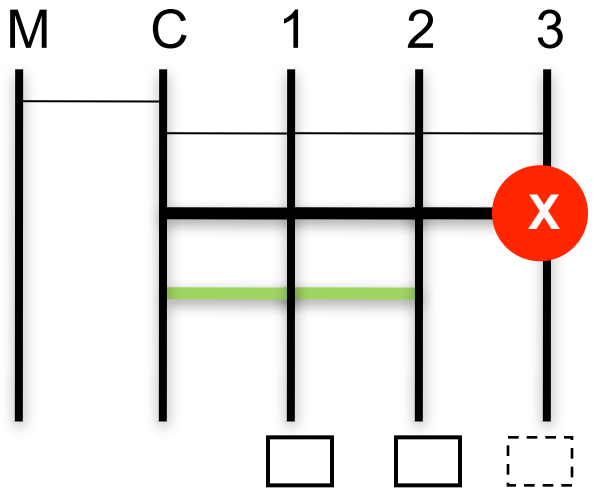
Updating Expectation



Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = “Data Transfer”, expectedNode(B, N)

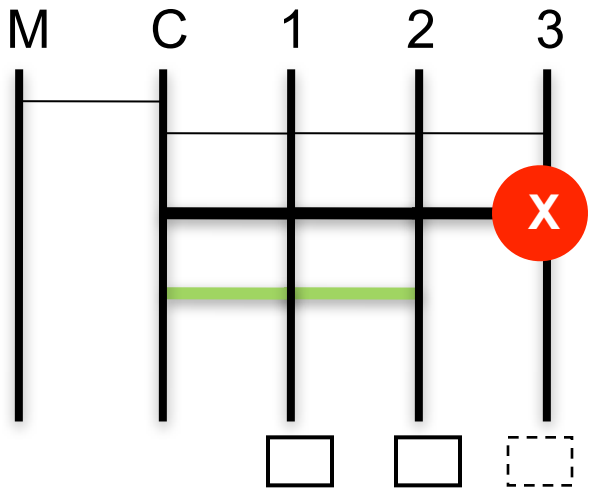
Updating Expectation



Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = “Data Transfer”, expectedNode(B, N)

Updating Expectation

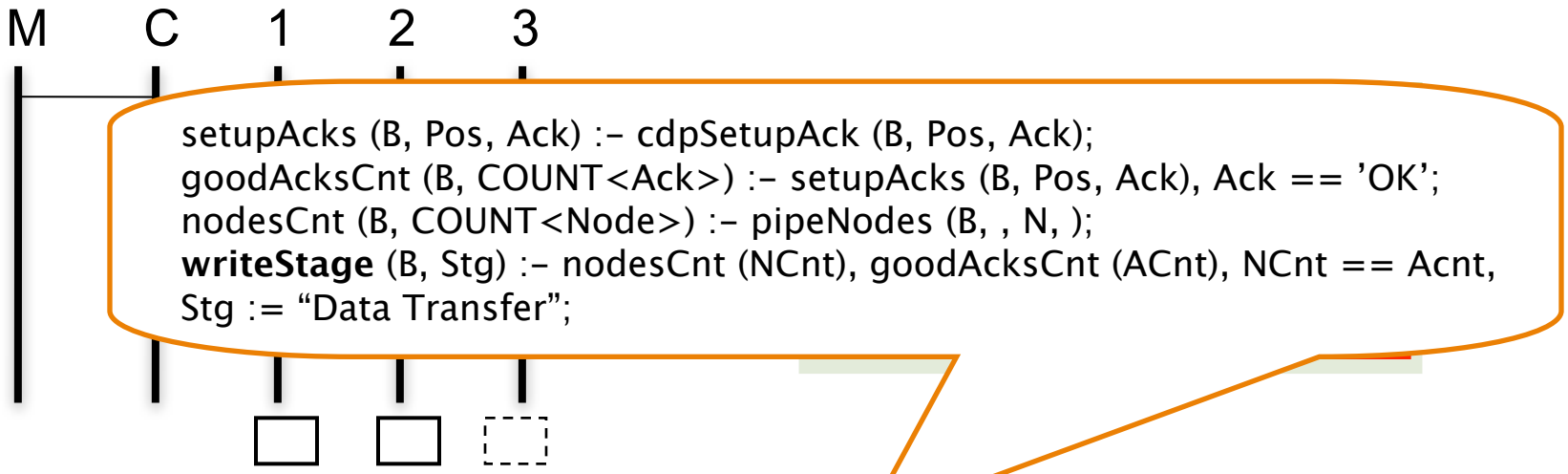


Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)

- "Client receives all acks from setup stage writeStage" → enter Data Transfer stage

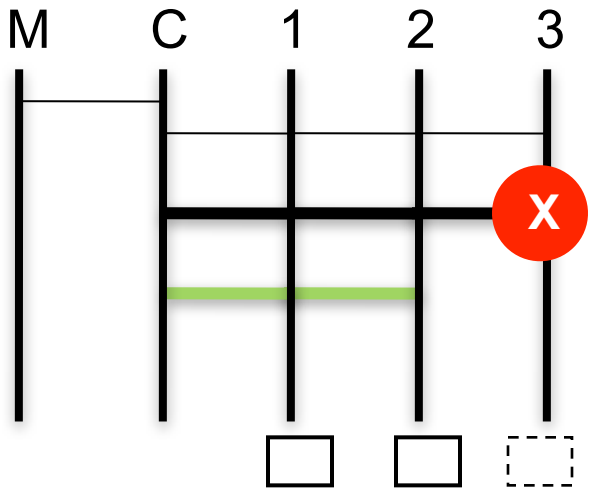
Updating Expectation



DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)

- "Client receives all acks from setup stage writeStage" → enter Data Transfer stage

Updating Expectation

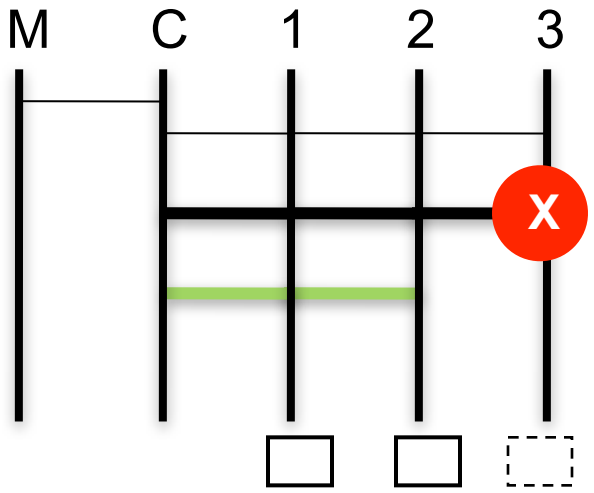


Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)

- "Client receives all acks from setup stage writeStage" → enter Data Transfer stage

Updating Expectation

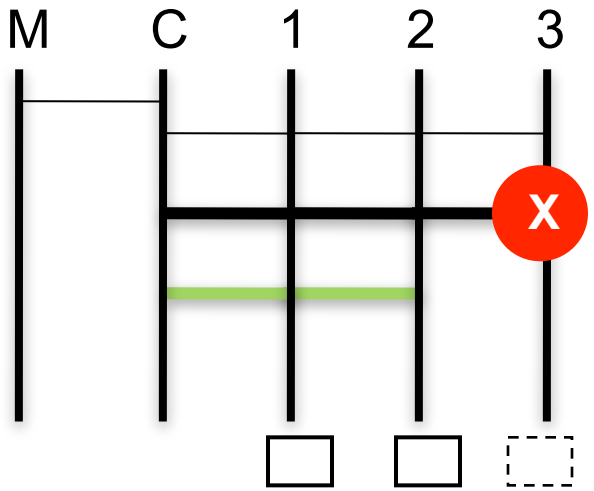


Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)

- "Client receives all acks from setup stage writeStage" → enter Data Transfer stage
- Precise failure events

Updating Expectation

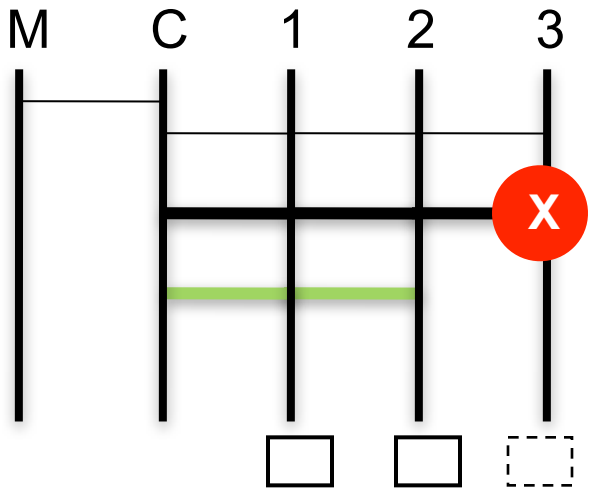


Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)

- "Client receives all acks from setup stage writeStage" → enter Data Transfer stage
- Precise failure events
  - Different stages → different recovery behaviors → different specifications

Updating Expectation

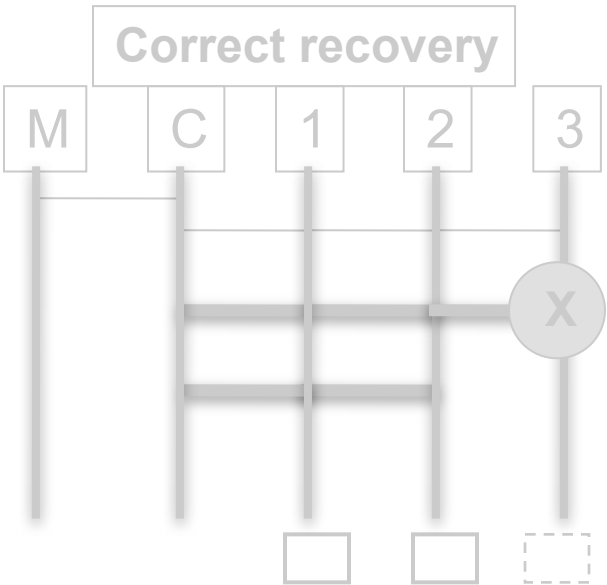


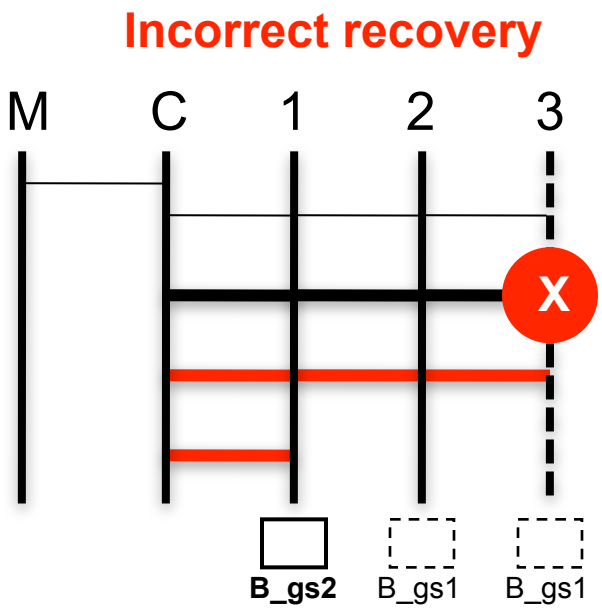
Expected Nodes(Block, Node)	
B	Node 1
B	Node 2
B	Node 3

DEL expectedNodes(B, N) :- fateCrashNode(N), writeStage(B, Stage),
Stage = "Data Transfer", expectedNode(B, N)

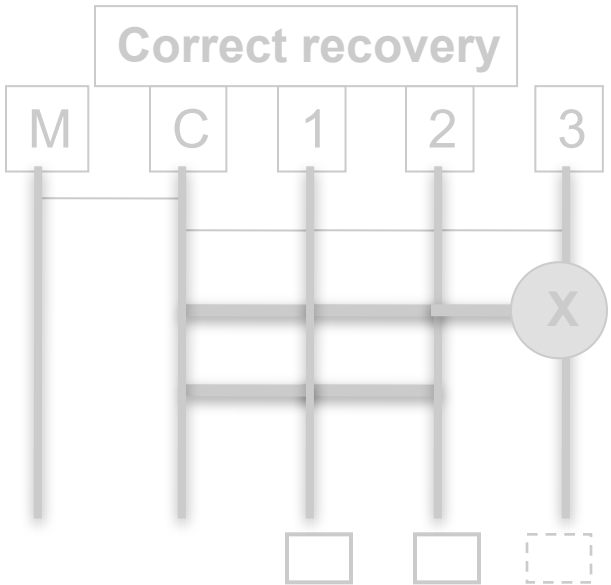
- "Client receives all acks from setup stage writeStage" → enter Data Transfer stage
- Precise failure events
  - Different stages → different recovery behaviors → different specifications
  - FATE and DESTINI must work hand in hand

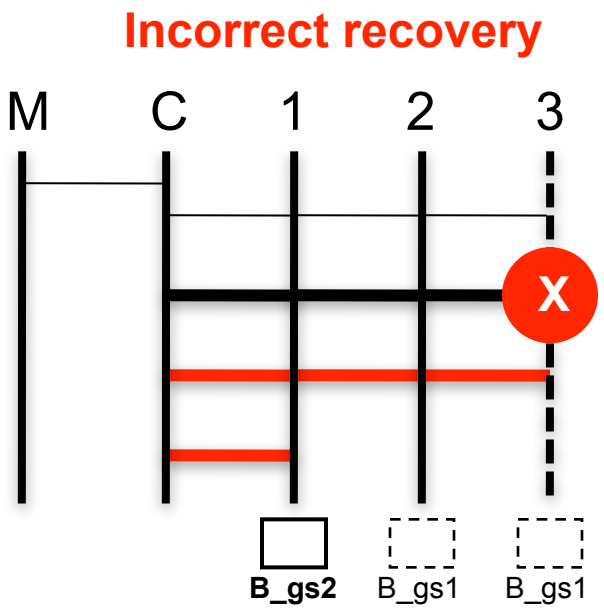
Capture Facts





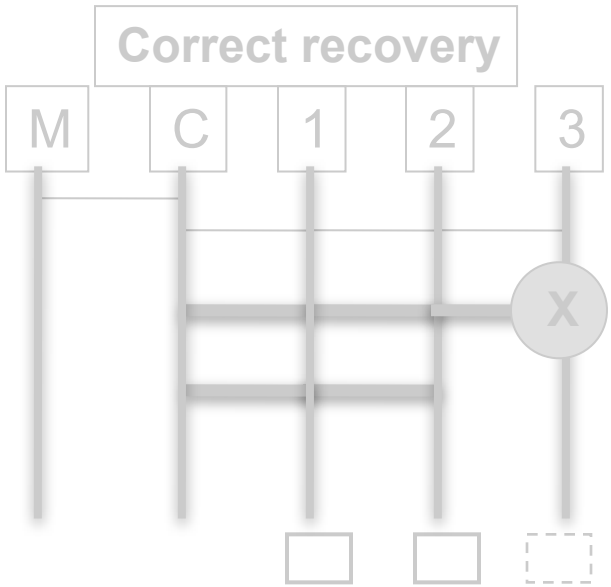
Capture Facts

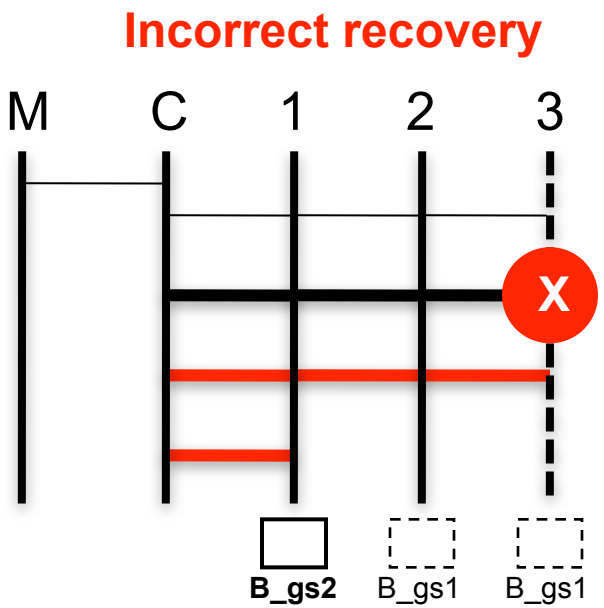




actualNodes(B, N) :- blocksLocation(B, N, Gs), latestGenStamp(B, Gs)

Capture Facts

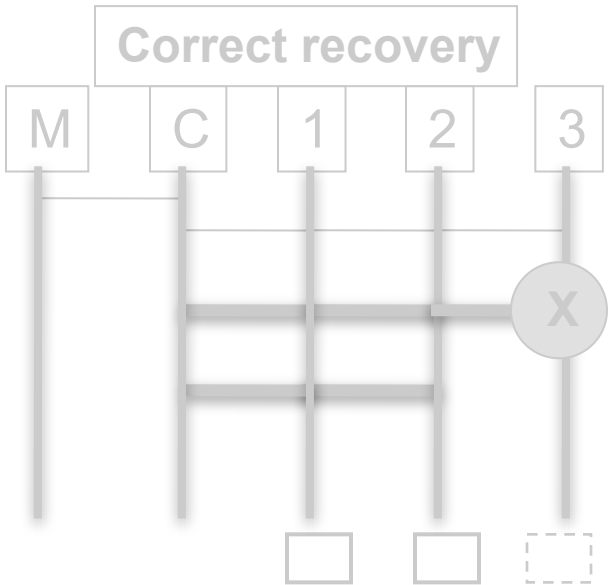


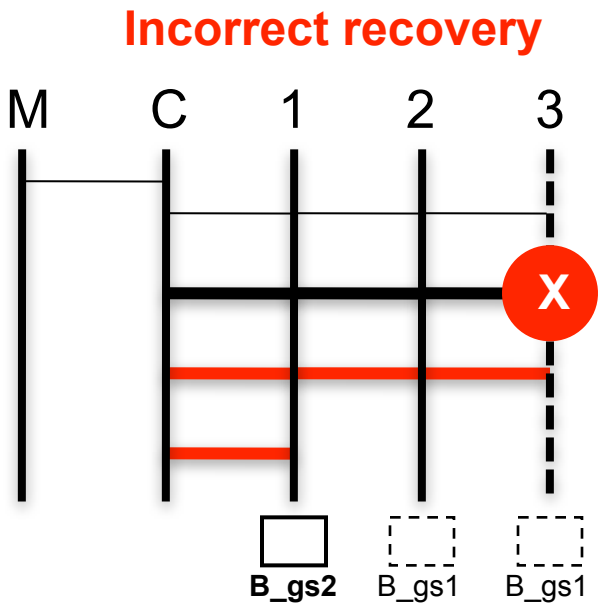


actualNodes(B, N) :- blocksLocation(B, N, Gs), latestGenStamp(B, Gs)

blocksLocations(B, N, Gs)		
B	Node 1	2
B	Node 2	1
B	Node 3	1

Capture Facts



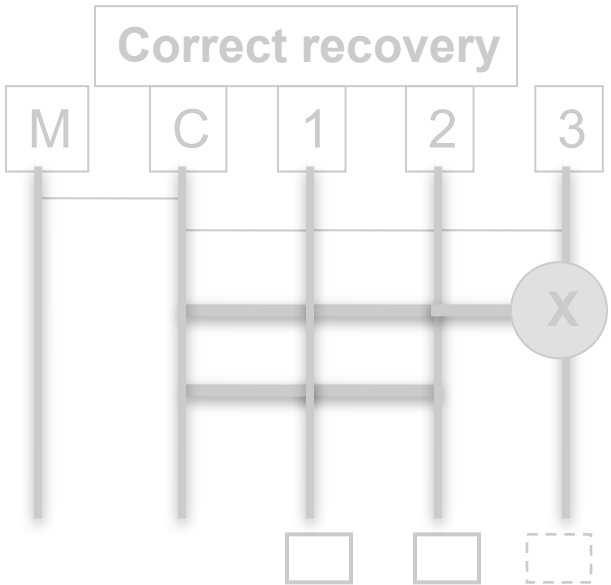


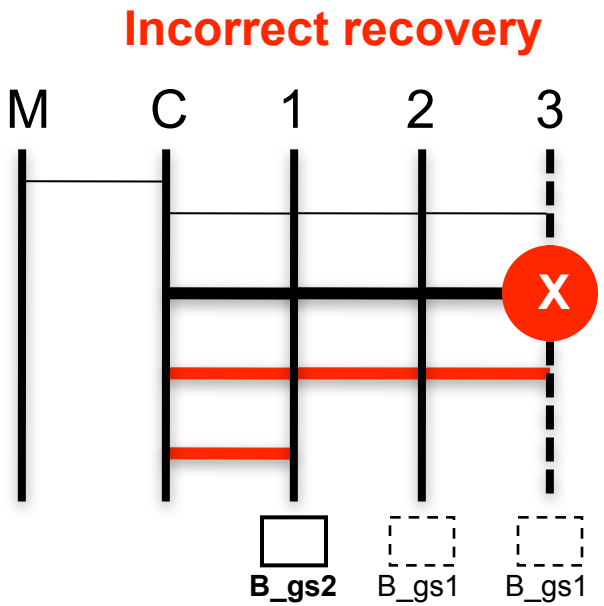
actualNodes(B, N) :- blocksLocation(B, N, Gs), latestGenStamp(B, Gs)

blocksLocations(B, N, Gs)		
B	Node 1	2
B	Node 2	1
B	Node 3	1

latestGenStamp(B, Gs)	
B	2

Capture Facts



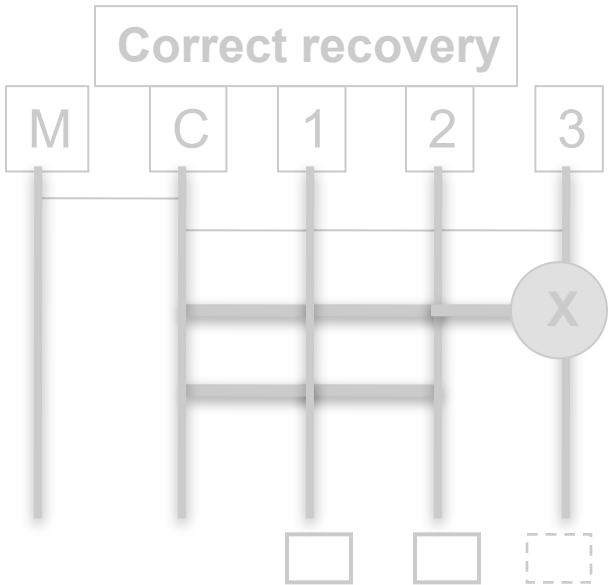


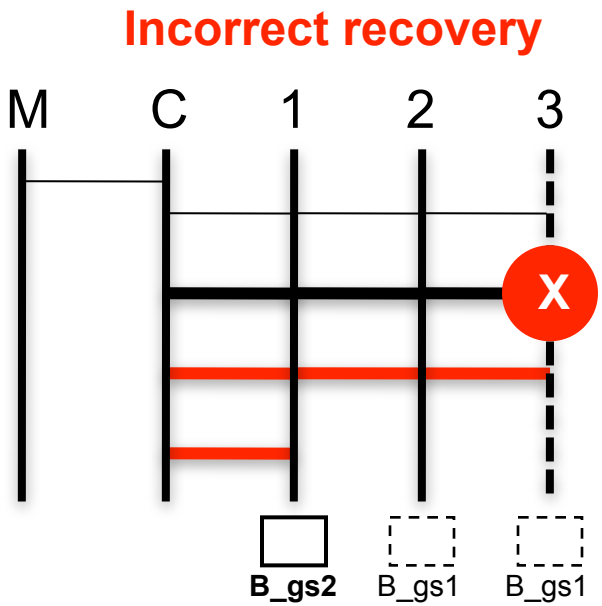
actualNodes(B, N) :- blocksLocation(B, N, Gs), latestGenStamp(B, Gs)

blocksLocations(B, N, Gs)		
B	Node 1	2
B	Node 2	1
B	Node 3	1

latestGenStamp(B, Gs)	
B	2

Capture Facts



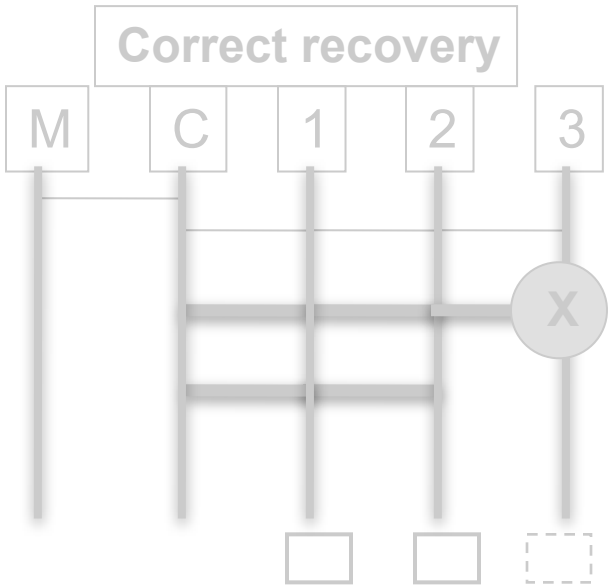


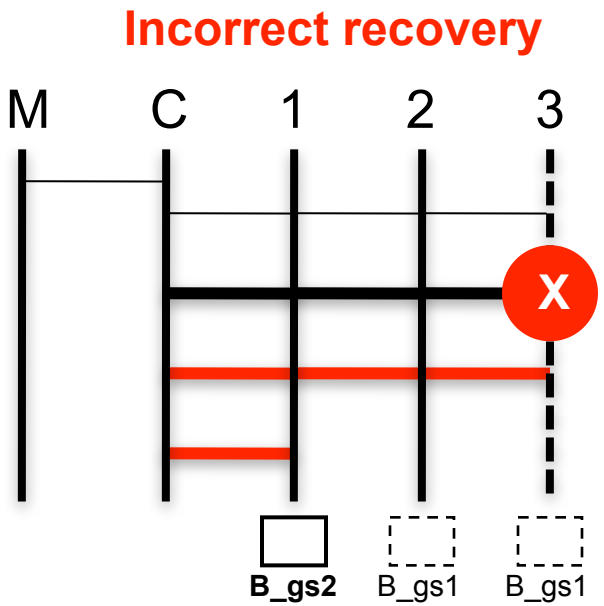
actualNodes(B, N) :- blocksLocation(B, N, Gs), latestGenStamp(B, Gs)

blocksLocations(B, N, Gs)		
B	Node 1	2
B	Node 2	1
B	Node 3	1

latestGenStamp(B, Gs)	
B	2

Capture Facts





actualNodes(B, N) :- blocksLocation(B, N, Gs), latestGenStamp(B, Gs)

actualNodes(Block, Node)	
B	Node 1

blocksLocations(B, N, Gs)		
B	Node 1	2
B	Node 2	1
B	Node 3	1

latestGenStamp(B, Gs)	
B	2

Violation and Check-Timing

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N),
cnpComplete(B) ;

IncorrectNodes (Block, Node)	
B	Node 2

ExpectedNodes (Block, Node)	
B	Node 1
B	Node 2

actualNodes(Block, Node)	
B	Node 1

Violation and Check-Timing

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N),
cnpComplete(B) ;

IncorrectNodes (Block, Node)	
B	Node 2

ExpectedNodes (Block, Node)	
B	Node 1
B	Node 2

actualNodes(Block, Node)	
B	Node 1

- There is a point in time where recovery is ongoing, thus specifications are violated

Violation and Check-Timing

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N),
cnpComplete(B) ;

IncorrectNodes (Block, Node)	
B	Node 2

ExpectedNodes (Block, Node)	
B	Node 1
B	Node 2

actualNodes(Block, Node)	
B	Node 1

- There is a point in time where recovery is ongoing, thus specifications are violated
- Need precise events to decide when the check should be done

Violation and Check-Timing

incorrectNodes(B, N) :- expectedNodes(B, N), NOT-IN actualNodes(B, N),
cnpComplete(B) ;

IncorrectNodes (Block, Node)	
B	Node 2

ExpectedNodes (Block, Node)	
B	Node 1
B	Node 2

actualNodes(Block, Node)	
B	Node 1

- There is a point in time where recovery is ongoing, thus specifications are violated
- Need precise events to decide when the check should be done
  - In this example, upon block completion

Rules

r1	**incorrectNodes** (B, N)	:–	cnpComplete (B), expectedNodes (B, N), NOT-IN actualNodes (B, N);
r2	pipeNodes (B, Pos, N)	:–	getBlkPipe (UFile, B, Gs, Pos, N);
r3	**expectedNodes** (B, N)	:–	getBlkPipe (UFile, B, Gs, Pos, N);
r4	DEL expectedNodes (B, N)	:–	fateCrashNode (N), pipeStage (B, Stg), Stg == 2, expectedNodes (B, N);
r5	setupAcks (B, Pos, Ack)	:–	cdpSetupAck (B, Pos, Ack);
r6	goodAcksCnt (B, CUUNT<Ack>)	:–	setupAcks (B, Pos, Ack), Ack == 'OK';
r7	nodesCnt (B, COUNT<Node>)	:–	pipeNodes (B, , N,);
r8	pipeStage (B, Stg)	:–	nodesCnt (NCnt), goodAcksCnt (ACnt), NCnt == Acnt, Stg := 2;
r9	blkGenStamp (B, Gs)	:–	dnpNextGenStamp (B, Gs);
r10	blkGenStamp (B, Gs)	:–	cnpGetBlkPipe (UFile, B, Gs, ,);
r11	diskFiles (N, File)	:–	fsCreate (N, File);
r12	diskFiles (N, Dst)	:–	fsRename (N, Src, Dst), diskFiles (N, Src, Type);
r13	DEL diskFiles (N, Src)	:–	fsRename (N, Src, Dst), diskFiles (N, Src, Type);
r14	fileTypes (N, File, Type)	:–	diskFiles(N, File), Type := Util.getType(File);
r15	blkMetas (N, B, Gs)	:–	fileTypes (N, File, Type), Type == metafile, Gs := Util.getGs(File);
r16	**actualNodes** (B, N)	:–	blkMetas (N, B, Gs), blkGenStamp (B, Gs);

Rules

r1	**incorrectNodes** (B, N)	:–	cnpComplete (B), expectedNodes (B, N), NOT-IN actualNodes (B, N);
r2	pipeNodes (B, Pos, N)	:–	getBlkPipe (UFile, B, Gs, Pos, N);
r3	**expectedNodes** (B, N)	:–	getBlkPipe (UFile, B, Gs, Pos, N);
r4	DEL expectedNodes (B, N)	:–	fateCrashNode (N), pipeStage (B, Stg), Stg == 2, expectedNodes (B, N);
r5	setupAcks (B, Pos, Ack)	:–	cdpSetupAck (B, Pos, Ack);
r6	goodAcksCnt (B, CUUNT<Ack>)	:–	setupAcks (B, Pos, Ack), Ack == 'OK';
r7	nodesCnt (B, COUNT<Node>)	:–	pipeNodes (B, , N,);
r8	pipeStage (B, Stg)	:–	nodesCnt (NCnt), goodAcksCnt (ACnt), NCnt == Acnt, Stg := 2;
r9	blkGenStamp (B, Gs)	:–	dnpNextGenStamp (B, Gs);
r10	blkGenStamp (B, Gs)	:–	cnpGetBlkPipe (UFile, B, Gs, ,);
r11	diskFiles (N, File)	:–	fsCreate (N, File);
r12	diskFiles (N, Dst)	:–	fsRename (N, Src, Dst), diskFiles (N, Src, Type);
r13	DEL diskFiles (N, Src)	:–	fsRename (N, Src, Dst), diskFiles (N, Src, Type);
r14	fileTypes (N, File, Type)	:–	diskFiles(N, File), Type := Util.getType(File);
r15	blkMetas (N, B, Gs)	:–	fileTypes (N, File, Type), Type == metafile, Gs := Util.getGs(File);
r16	**actualNodes** (B, N)	:–	blkMetas (N, B, Gs), blkGenStamp (B, Gs);

Rules

r1	**incorrectNodes** (B, N)	:-	cnpComplete (B), expectedNodes (B, N), NOT-IN actualNodes (B, N);
r2	pipeNodes (B, Pos, N)	:-	getBlkPipe (UFile, B, Gs, Pos, N);
r3	**expectedNodes** (B, N)	:-	getBlkPipe (UFile, B, Gs, Pos, N);
r4	DEL expectedNodes (B, N)	:-	fateCrashNode (N), pipeStage (B, Stg), Stg == 2, expectedNodes (B, N);
r5	setupAcks (B, Pos, Ack)	:-	cdpSetupAck (B, Pos, Ack);
r6			
r7			
r8			
r9			
r1			
r1			
r1			
r1			
r14	fileTypes (N, File, Type)	:-	diskFiles(N, File), Type := Util.getType(File);
r15	blkMetas (N, B, Gs)	:-	fileTypes (N, File, Type), Type == metafile, Gs := Util.getGs(File);
r16	**actualNodes** (B, N)	:-	blkMetas (N, B, Gs), blkGenStamp (B, Gs);

- Capture Facts, Build Expectation from IO events
  - No need to interpose internal functions
- Specification Reuse
  - For the first check, # rules : #check is 16:1
  - Overall, #rules: # check ratio is 3:1

Outline

- ✓ Introduction
- ✓ FATE
- ✓ DESTINI
- Evaluation
- Summary

Evaluation

- FATE: 3900 lines, DESTINI: 1200 lines
- Applied FATE and DESTINI to three cloud systems
  - HDFS, ZooKeeper, Cassandra
- 40,000 unique combination of failures
- Found 16 new bugs, reproduced 74 bugs
- 74 recovery specifications
  - 3 lines / check

Bugs found

- Reduced availability and performance
- Data loss due to multiple failures
- Data loss in log recovery protocol
- Data loss in append protocol
- Rack awareness property is broken

Conclusion

- FATE explores multiple failure systematically
- DESTINI enables concise recovery specifications
- FATE and DESTINI: a unified framework
  - Testing recovery specifications requires a failure service
  - Failure service needs recovery specifications to catch recovery bugs

Thank you!

QUESTIONS?

Berkeley Orders of Magnitude
http://boom.cs.berkeley.edu

The Advanced Systems Laboratory
http://www.cs.wisc.edu/adsl

Downloads our full TR paper from these websites