An Analysis of Network-Partitioning Failures in Cloud Systems

Ahmed Alquraan, Hatem Takruri, Mohammed Alfatafta, Samer Al-Kiswany

Highlights

- Network-partitioning failures are catastrophic, silent, and deterministic
- Surprisingly, partial partitions cause large number of failures
- Debunk two common presumptions
  1. Admins believe that systems can tolerate network partitions
  2. Designers believe isolating one side of the partition is enough
- NEAT: a network partitioning testing framework
  - Tested 7 systems → 32 failures

Motivation

- High availability: systems should tolerate infrastructure failures
  (Devices, nodes, network, data centers)

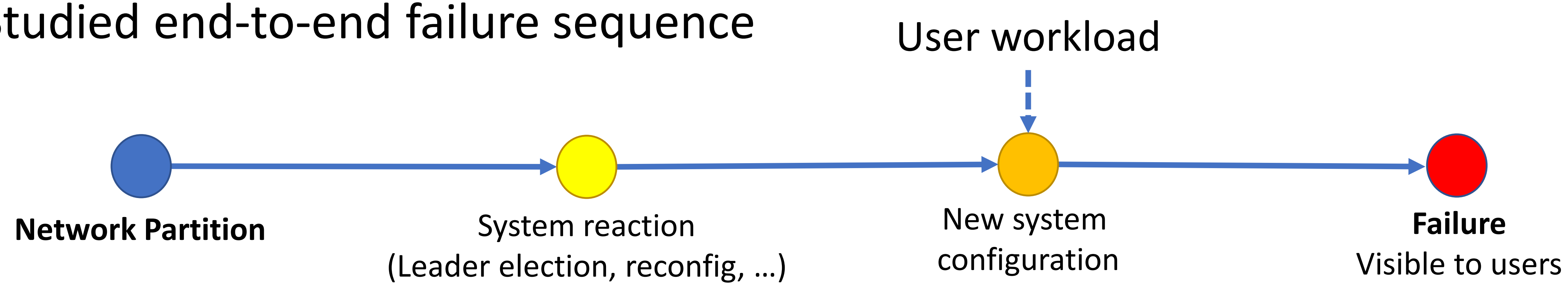
- We focus on network partitioning
  - Partitioning faults are common
    (once every two weeks at Google[1], 70% of downtime at Microsoft[2], once every 4 days at CENIC[3])
  - Complex to handle

What is the impact of network partitions on modern systems?

[1] Govindan et al, "Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure", ACM SIGCOMM 2016
[2] Gill et al, "Understanding network failures in data centers: measurement, analysis, and implications", ACM SIGCOMM 2011
[3] Turner et al, "California fault lines: understanding the causes and impact of network failures", ACM SIGCOMM 2010

In-depth analysis of production failures

Studied end-to-end failure sequence



- Study the impact of failures
- Characterize conditions and sequence of events
- Identify opportunities to improve fault tolerance

Methodology

- Studied 136 high-impact network-partitioning failures from 25 systems
    - 104 failures are user-reported failures
    - 32 failures are discovered by NEAT
- Studied failure report, discussion, logs, code, and tests
- Reproduced 24 failures to understand intricate details

Highlights

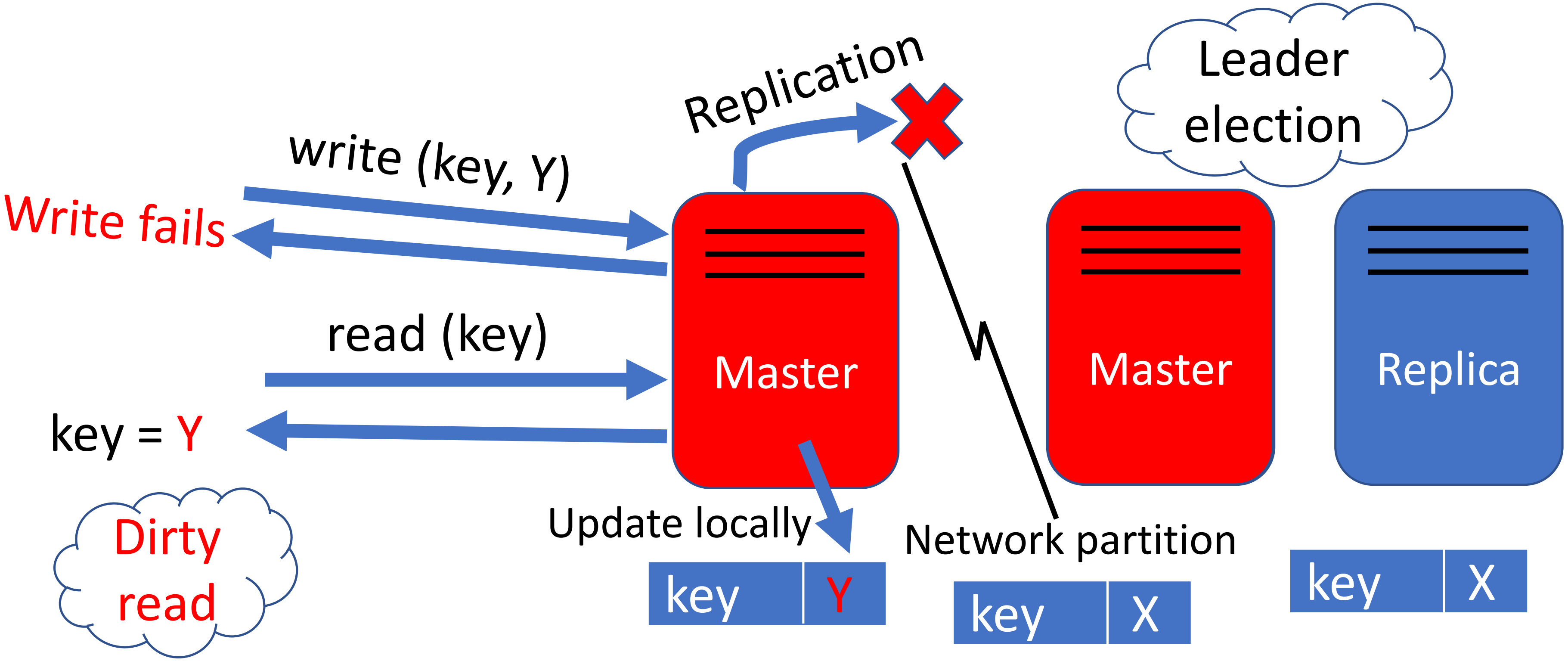
- Network partitioning failures are catastrophic, silent, and easy to manifest
- Surprisingly, partial partitions cause large number of failures
- Debunk two common presumptions
    1. Admins believe that systems can tolerate network partitions
    2. Designers believe isolating one side of the partition is enough
- NEAT: a network partitioning testing framework
    - Tested 7 systems → 32 failures

Example – Dirty read in VoltDB

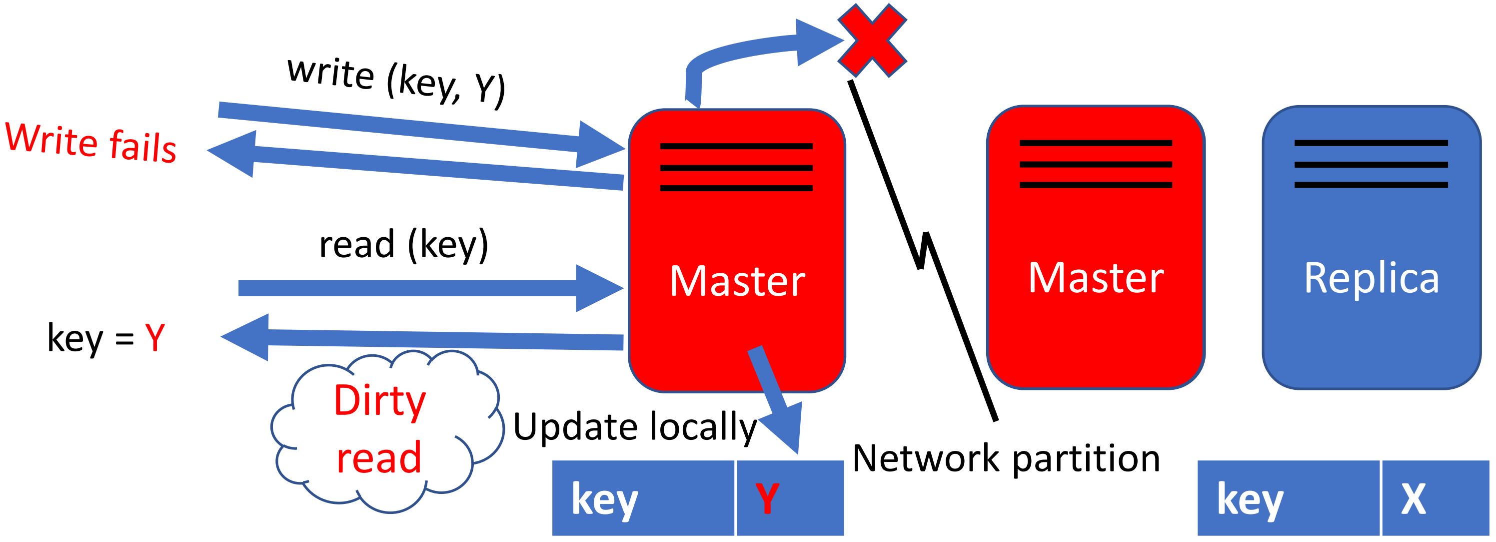
Event1: Network partition

Event2: Write to minority

Event3: Read from minority



Failure impact



- Catastrophic failure
  - Data loss, dirty read, broken locks, double dequeue, corruption

Majority (80%) of the failures are catastrophic

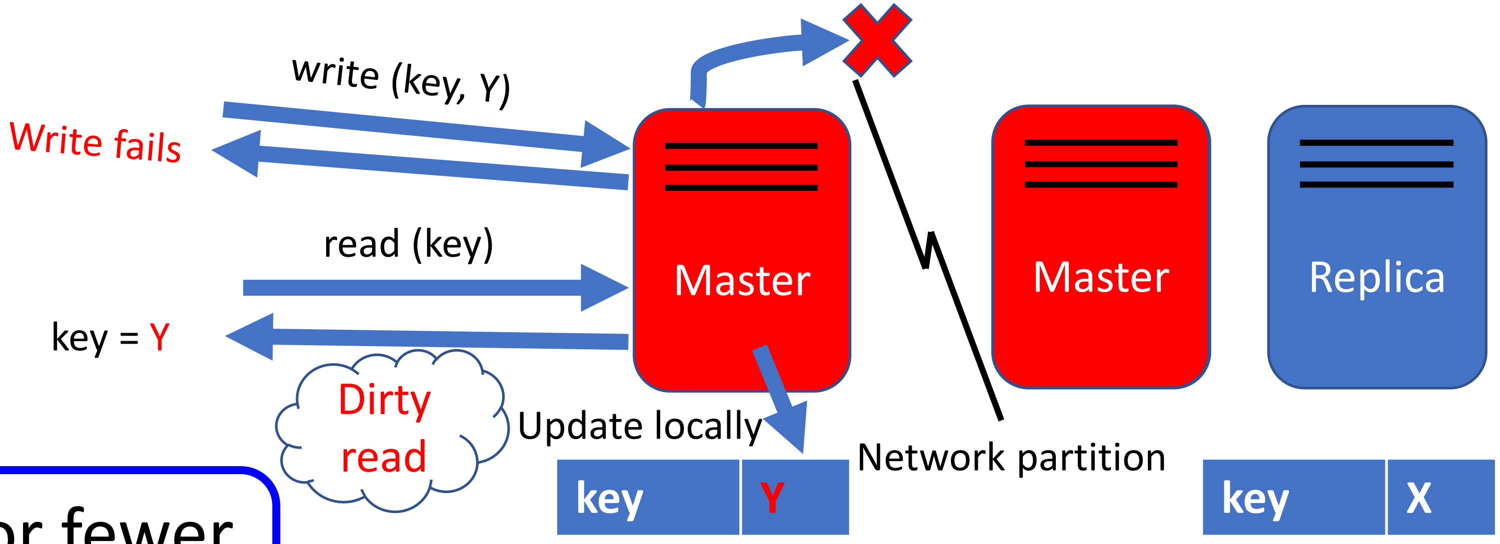
Majority (90%) of the failures are silent

Event 1: Network partition

Event 2: Write to minority

Event 3: Read from minority

Timing and ordering



- Require 3 events

70% of the failures require 3 or fewer events

Multiple events should happen in a specific order

Majority (80%) are deterministic or have known timing constraints

timeout

Event 1: Network partition

Event 2: Write to minority

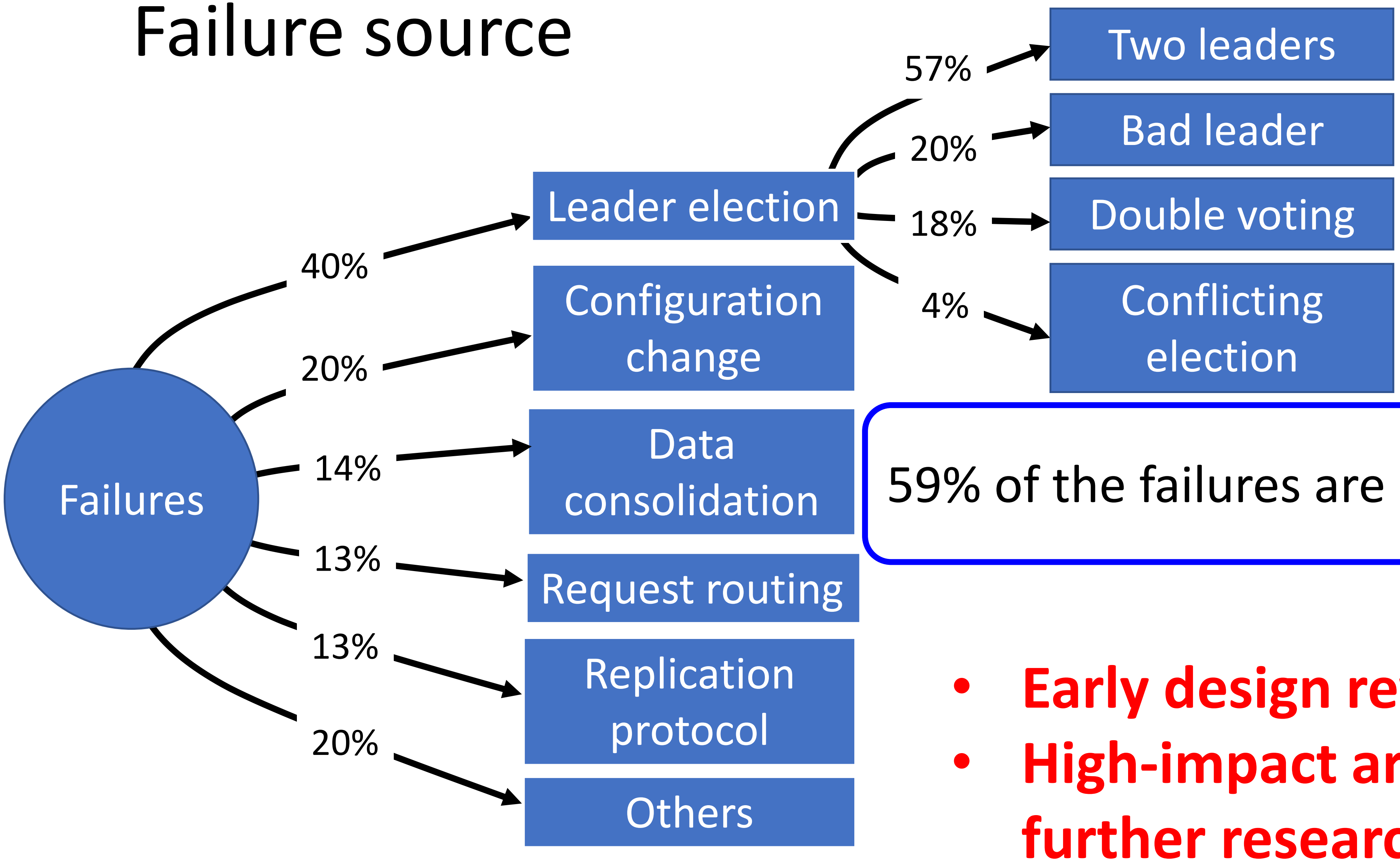
Event 3: Read from minority

Old master shuts down

Timing: should occur before the old master shuts down

Surprisingly, partition failures are deterministic, silent, and catastrophic

Failure source



59% of the failures are due to design flaws

- **Early design reviews can help**
- **High-impact area that needs further research**

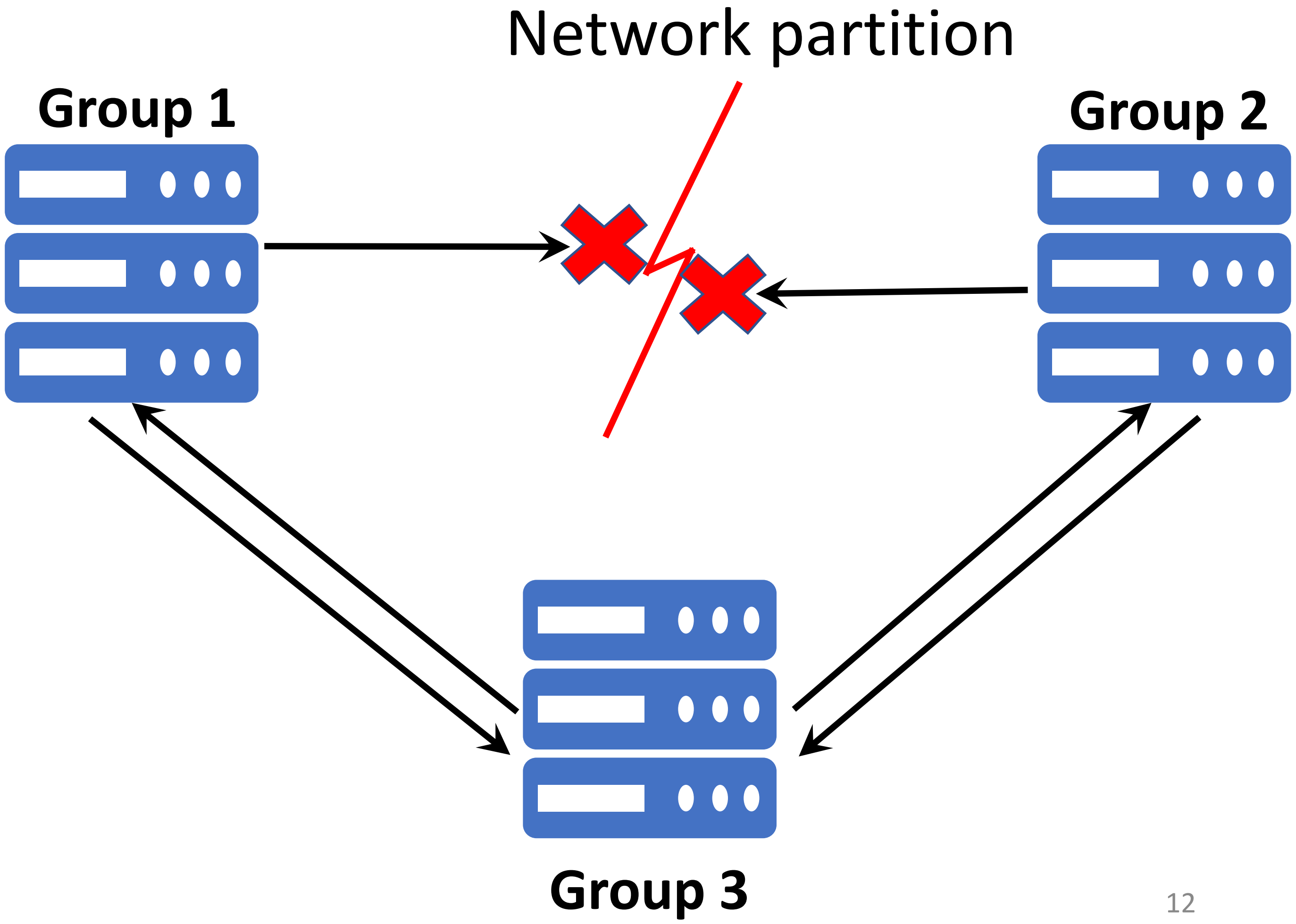
Highlights

- Network partitioning failures are catastrophic, silent, and easy to manifest
- Surprisingly, partial partitions cause large number of failures
- Debunk two common presumptions
    1. Admins believe that systems can tolerate network partitions
    2. Designers believe isolating one side of the partition is enough
- NEAT: a network partitioning testing framework
    - Tested 7 systems → 32 failures

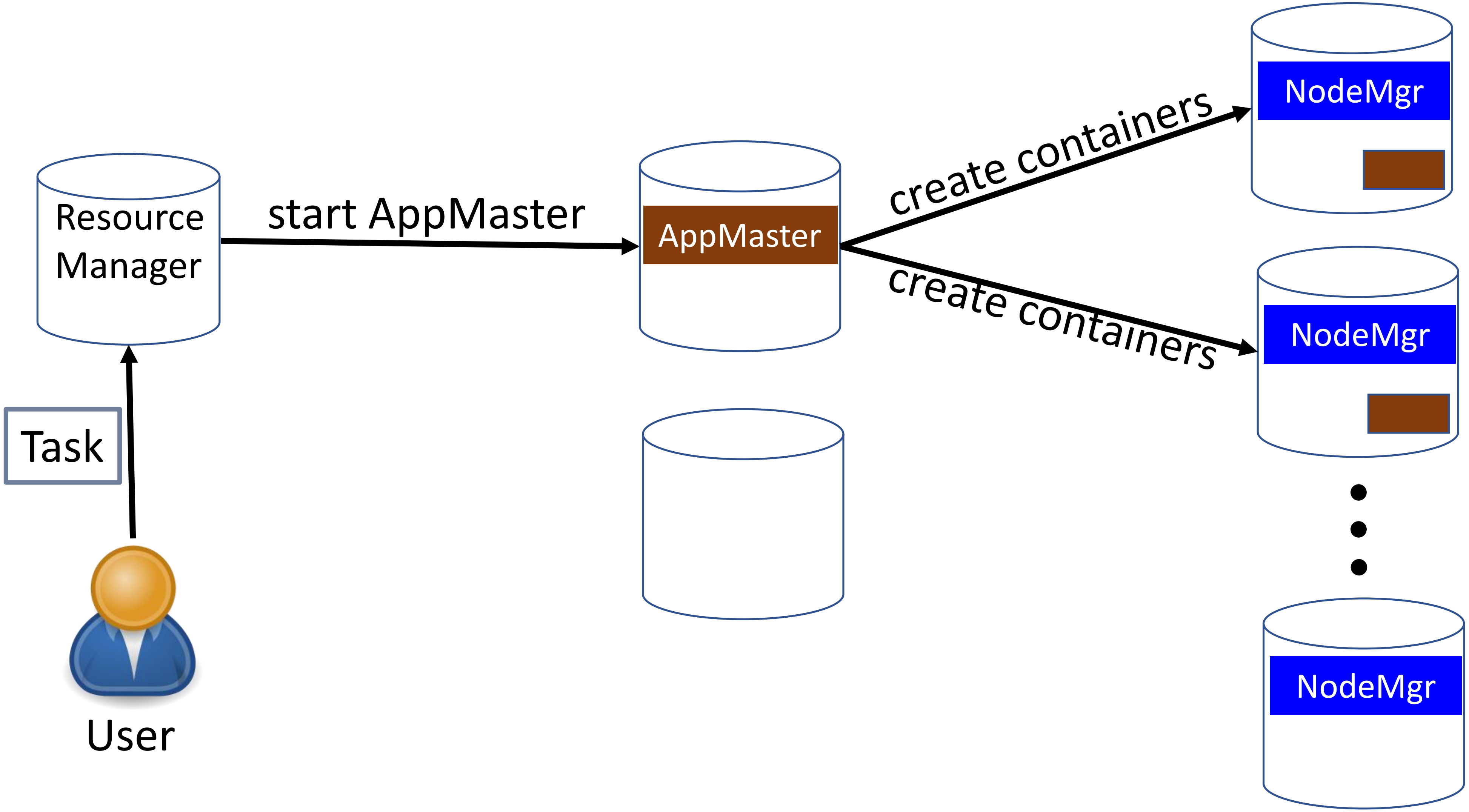
Partial network partitioning

Network partition types

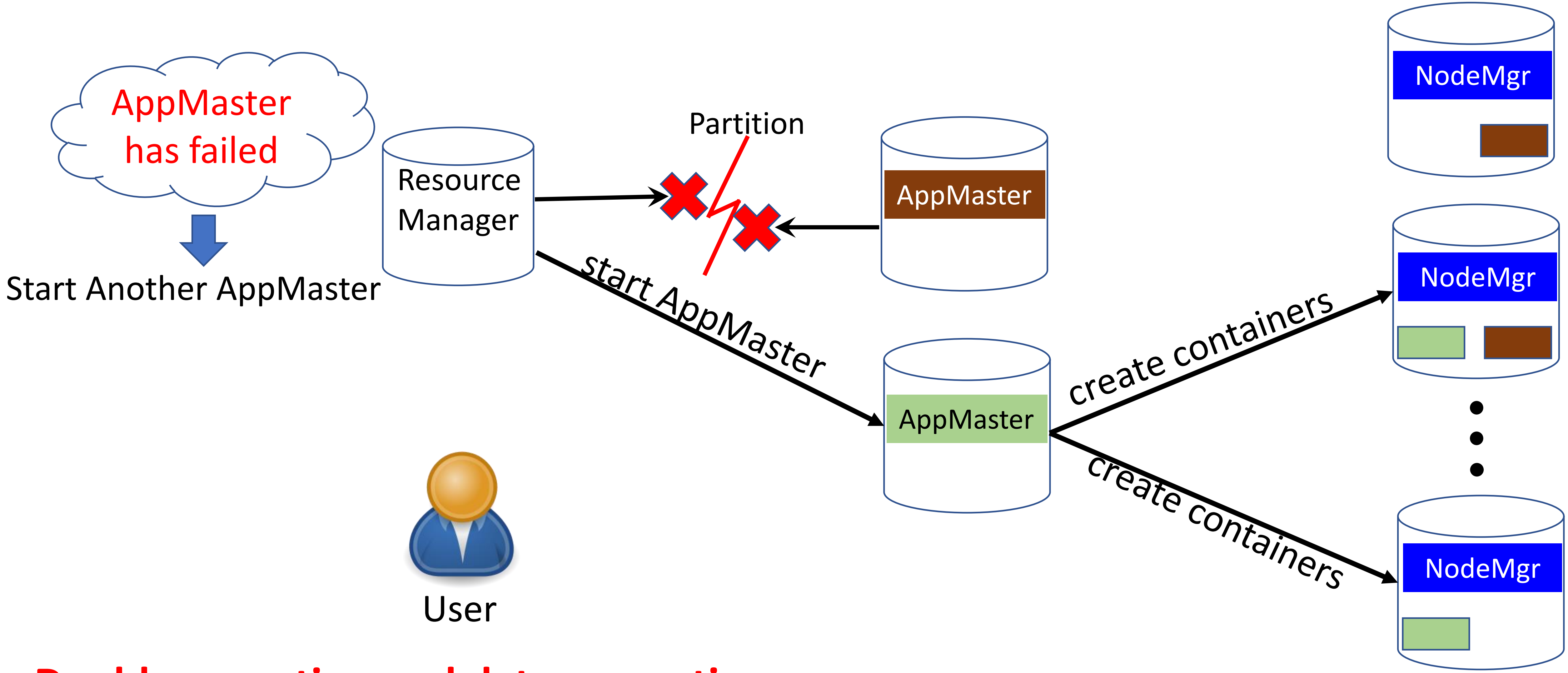
- Complete
- Partial
- Simplex



Partial network partition - double execution in MapReduce

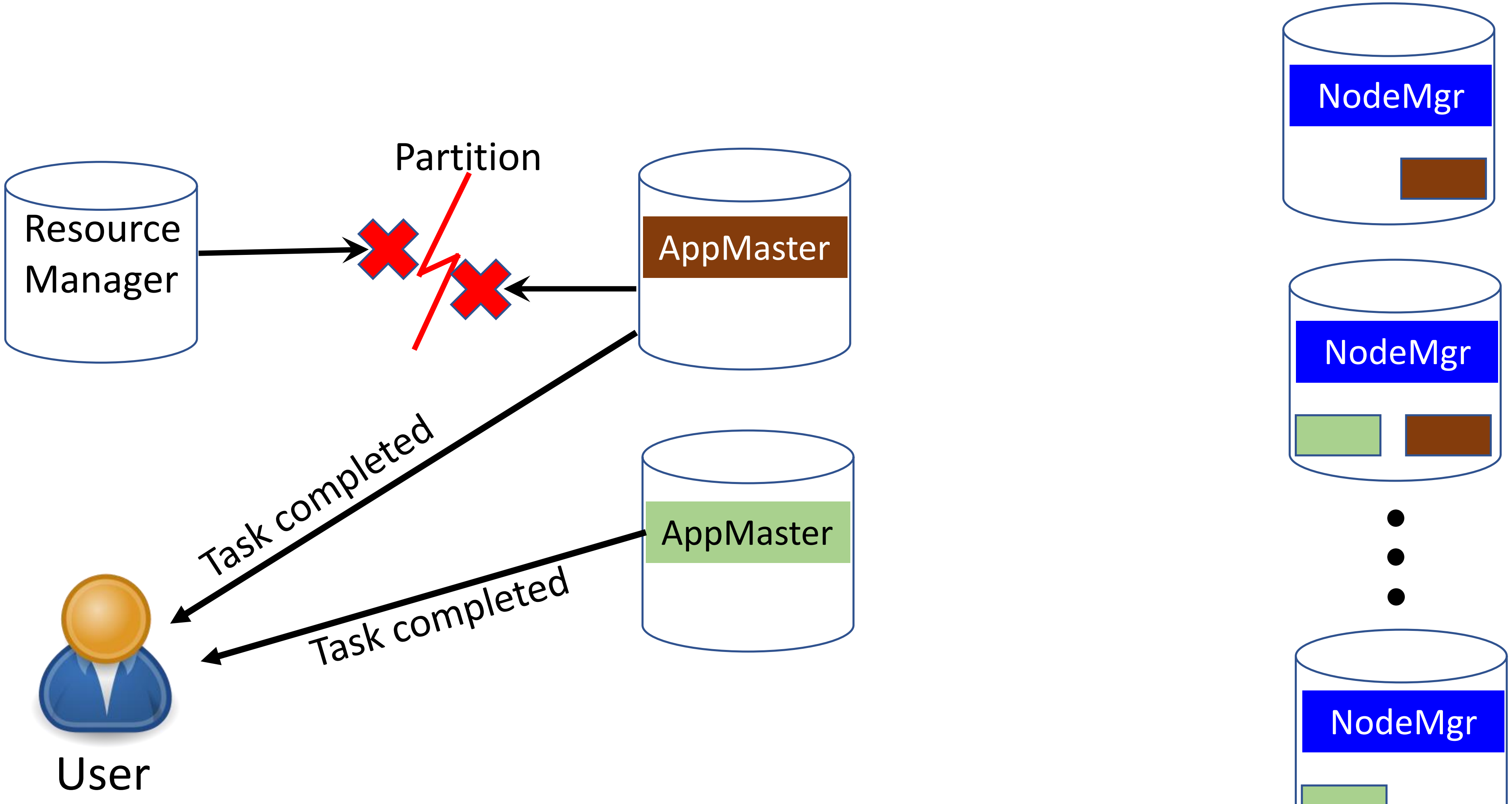


Partial network partition - double execution in MapReduce



- **Double execution and data corruption**

Partial network partition - double execution in MapReduce

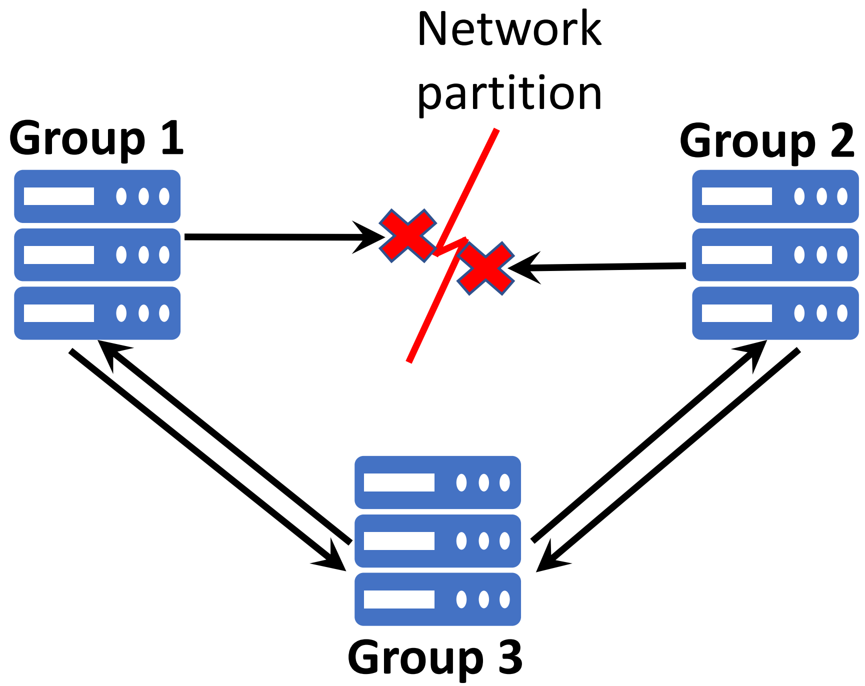


- **Double execution and data corruption**
- **Confuses the user**

Partial network partitioning

Partial partitioning leads to 28% of the failures

- Affects leader election, scheduling, data placement, and configuration change



- **Leads to inconsistent view of system state**
- **Partial partitions are poorly understood and tested**

Highlights

- Network partitioning failures are catastrophic, silent, and easy to manifest
- Surprisingly, partial partitions cause large number of failures
- Debunk two common presumptions
    1. Admins believe that systems can tolerate network partitions
    2. Designers believe isolating one side of the partition is enough
- NEAT: a network partitioning testing framework
    - Tested 7 systems → 32 failures

Debunks two presumptions

- Admins believe systems with data redundancy can tolerate partitioning
  - ➢Action: low priority for repairing ToR switches[1]

Reality: 83% of the failures occur by isolating a single node

- Systems restrict client access to one side to eliminate failures

Reality: 64% of the failures require no client access or access to one side only

[1] Phillipa et al, "Understanding network failures in data centers: measurement, analysis, and implications" in OSDI'14

Other findings

- Failures in proven protocols are due to optimizations
- Majority (83%) of the failures can be reproduced with 3 nodes
- Majority (93%) of the failures can be reproduced through tests

Highlights

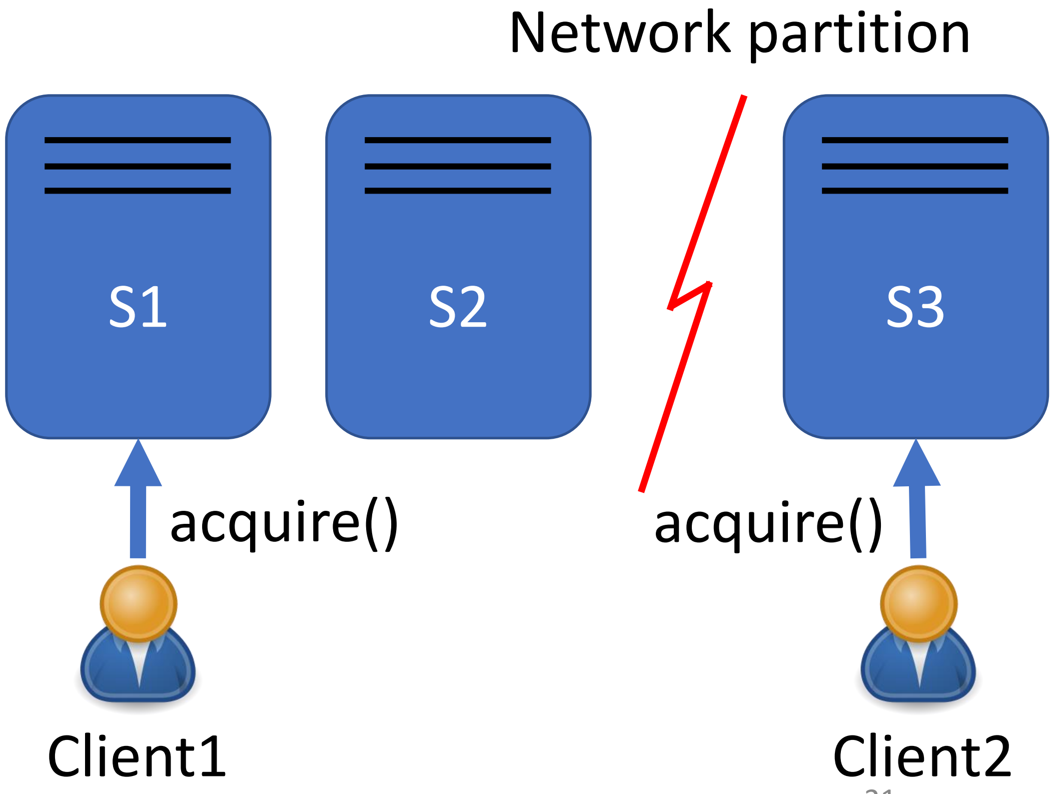
- Network partitioning failures are catastrophic, silent, and easy to manifest
- Surprisingly, partial partitions cause large number of failures
- Debunk two common presumptions
  1. Admins believe that systems can tolerate network partitions
  2. Designers believe isolating one side of the partition is enough
- NEAT: a network partitioning testing framework
  - Tested 7 systems → 32 failures

NEtwork pArtitioning Testing framework (NEAT)

- Supports all types of network partitions
- Simple API

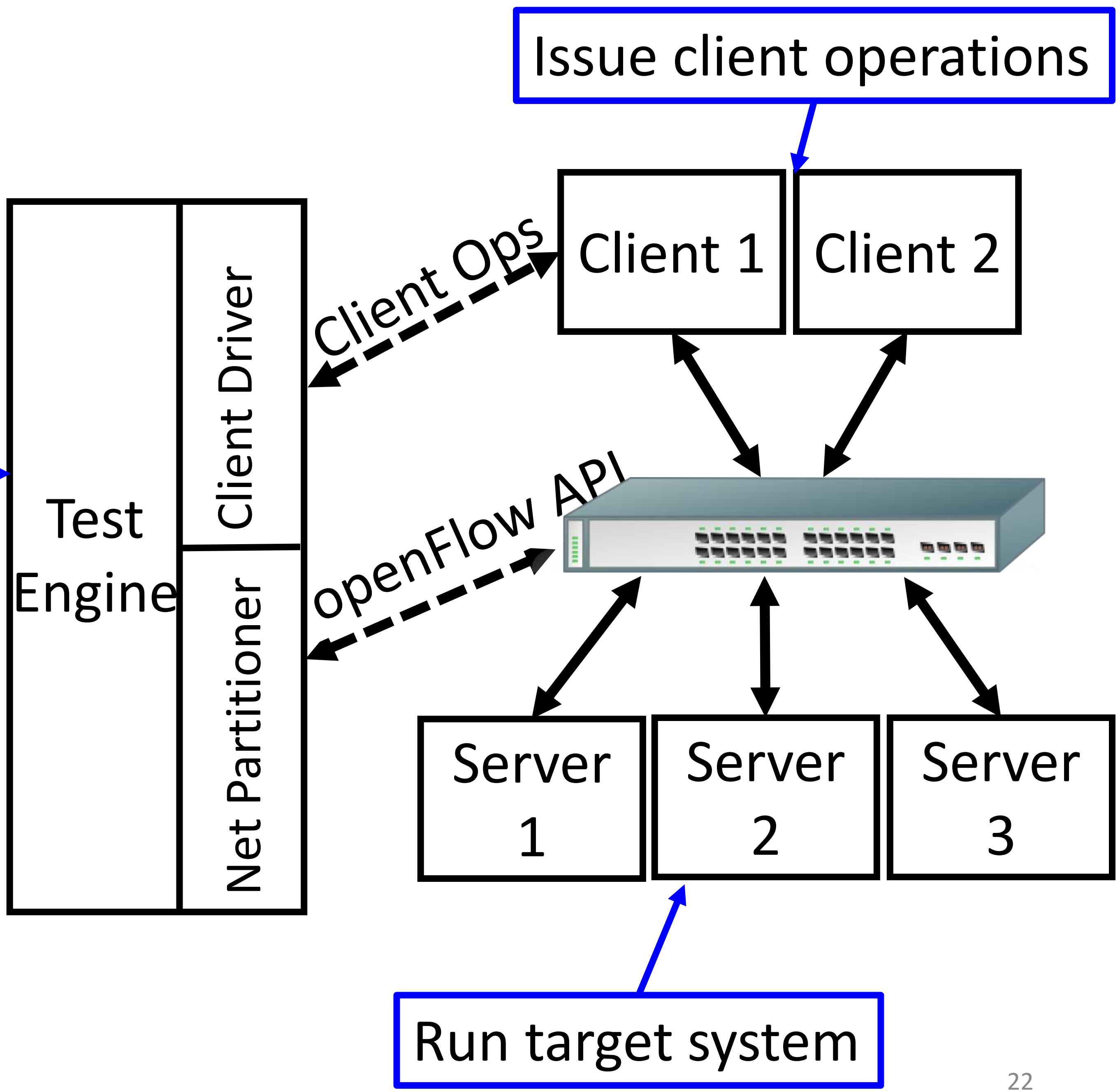
```
client1.createSemaphore(1)
side1 = asList(S1, S2, client1);
side2 = asList(S3, client2);
netPart = Partitioner.complete(side1, side2);
assertTrue(client1.sem_trywait());
assertFalse(client2.sem_trywait());
Partitioner.heal(netPart);
```

Apache Ignite double locking failure



NEAT design

- Orders client operations
- Injects and heals partitions
  - OpenFlow
  - iptables



Testing with NEAT

- We tested 7 systems using NEAT
- Discovered 32 failures → 30 catastrophic
  - Confirmed: 12

System	# failures found
ActiveMQ	2
Ceph	2
Ignite	15
Infinispan	1
Terracotta	9
MooseFS	2
DKron	1

Concluding remarks

- Further research is needed for network partition fault tolerance
  - Specially partial partitions
- Highlight the danger of using unreachability as an indicator of node crash
- Identify ordering, timing, network characteristics to simplify testing
- Identify common pitfalls for developers and admins
- NEAT: network partitioning testing framework

https://dsl.uwaterloo.ca/projects/neat/