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?

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

Network Partition → System reaction (Leader election, reconfig, …) → New system configuration → Failure Visible to users

- New system configuration
- Network Partition
- Failed to users
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

- Write fails
  - write (key, Y)
  - read (key)

- Key = Y

- Dirty read

- Leader election

- Event 1: Network partition
  - Master
  - Replica

- Event 2: Write to minority
  - Master
  - Replica

- Event 3: Read from minority
  - Master
  - Replica

- Update locally
  - Key = Y
  - Key = X

- Network partition
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
Surprisingly, partition failures are deterministic, silent, and catastrophic. Dirty reads during network partition update locally. Timing: should occur before the old master shuts down. Majority (80%) are deterministic or have known timing constraints. Multiple events should happen in a specific order. 70% of the failures require 3 or fewer events. Requires 3 events.
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

User → Task → Resource Manager → start AppMaster → AppMaster → create containers → NodeMgr → create containers → NodeMgr → NodeMgr → NodeMgr
Partial network partition - double execution in MapReduce

- Double execution and data corruption

- AppMaster has failed

- Start Another AppMaster
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 \( \rightarrow \) 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

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 \( \rightarrow \) 32 failures
NEtwork pArtitioning Testing framework (NEAT)

- Supports all types of network partitions
- Simple API

```java
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);
```
NEAT design

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

Issue client operations

Run target system

Test Engine

Client Driver

Net Partitioner

Client 1

Client 2

Server 1

Server 2

Server 3
Testing with NEAT

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

<table>
<thead>
<tr>
<th>System</th>
<th># failures found</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveMQ</td>
<td>2</td>
</tr>
<tr>
<td>Ceph</td>
<td>2</td>
</tr>
<tr>
<td>Ignite</td>
<td>15</td>
</tr>
<tr>
<td>Infinispan</td>
<td>1</td>
</tr>
<tr>
<td>Terracotta</td>
<td>9</td>
</tr>
<tr>
<td>MooseFS</td>
<td>2</td>
</tr>
<tr>
<td>DKron</td>
<td>1</td>
</tr>
</tbody>
</table>
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/