



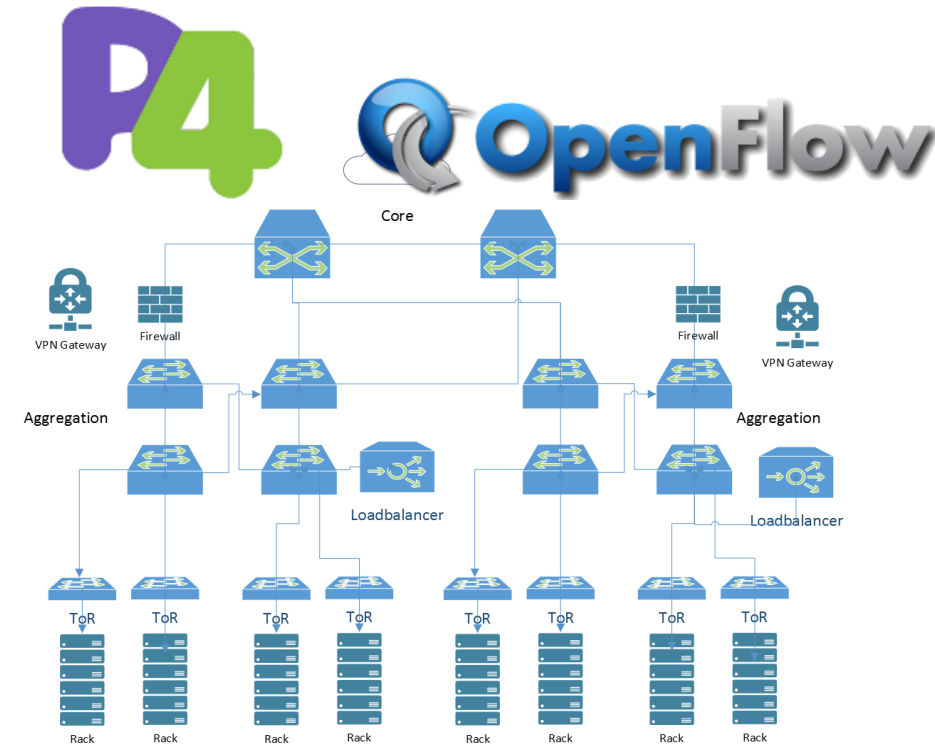
UNIVERSITY OF  
**WATERLOO**

# Toward a Generic Fault Tolerance Technique for Partial Network Partitioning

**Mohammed Alfatafta**, Basil Alkhatib, Ahmed Alquraan, Samer Al-Kiswany

# Modern Networks are Complex

- Multiple data centers
- Large scale
- Variety of middle boxes
- Heterogenous hardware and software
- Softwarization



Catastrophic network failures are common [1, 2, 3, 4]

[1] Daniel Turner et. al. On failure in managed enterprise networks. HP Labs HPL-2012-101, 2012.

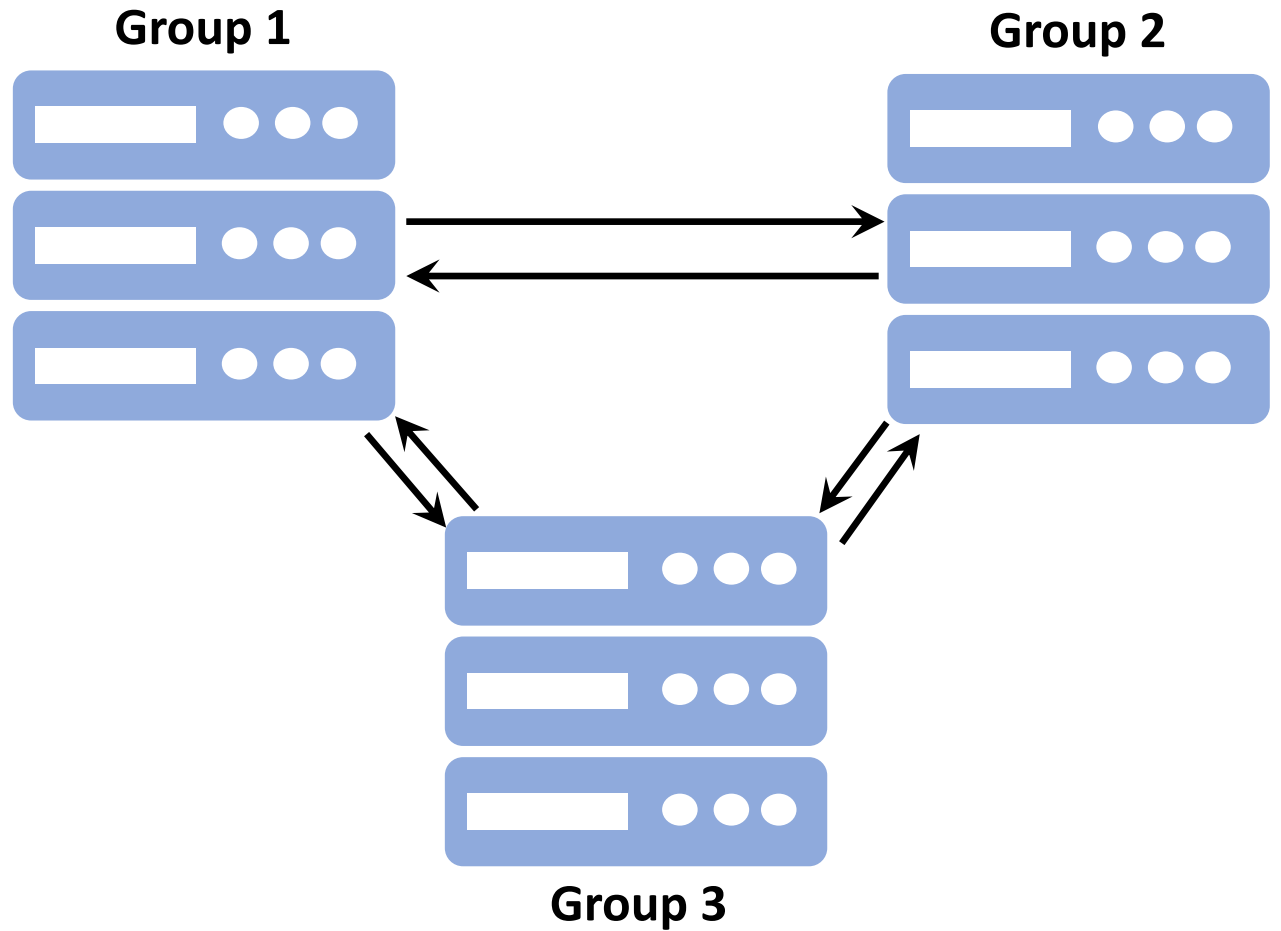
[2] Ramesh Govindan et. al. Evolve or die: High-availability design principles drawn from googles net-work infrastructure. 2016 ACM SIGCOMM

[3] Phillipa Gill et. al. Understanding network failures in data centers: measurement, analysis, and implications. 2011 SIGCOMM

[4] Daniel Turner et. al. California fault lines: understanding the causes and impact of network failures. 2011 SIGCOMM

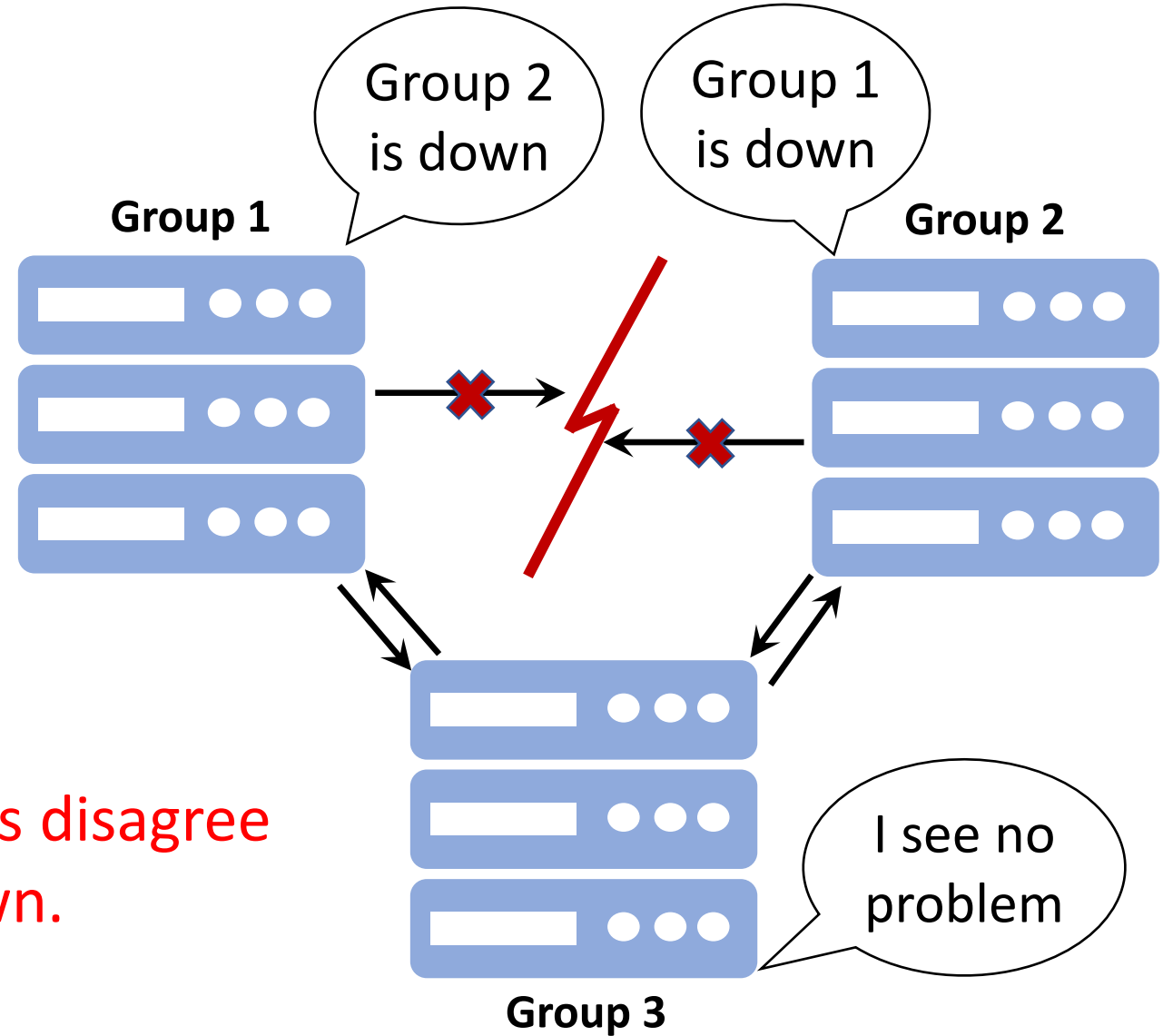
# Partial partitions

Isolate a set of nodes from some, but not all, nodes in the cluster.



# Partial partitions

Isolate a set of nodes from some, but not all, nodes in the cluster.



Impact: Confuses systems as nodes disagree whether a given node is up or down.

# Outline

- What causes partial network partitioning?
- How do they impact systems?
- Are there any fault tolerance techniques?
- NIFTY: a generic fault tolerance technique
- Evaluation

# Outline

- What causes partial network partitioning?
- How do they impact systems?
- Are there any fault tolerance techniques?
- NIFTY: a generic fault tolerance technique
- Evaluation

# Causes of partial partitions

- Failure of additional links between racks [1,2]
- Network and Firewall misconfigurations [3]
- Network upgrades [4]
- Flaky links between switches [5]

[1] Elasticsearch ticket: <https://github.com/elastic/elasticsearch/issues/6105>

[2] Blog post: <https://rachelbythebay.com/w/2012/02/16/partition/>

[3] Blog post: <https://www.robustperception.io/healthchecking-is-not-transitive>

[4] Elasticsearch ticket: <https://github.com/elastic/elasticsearch/issues/9495>

[5] MapReduce ticket: <https://issues.apache.org/jira/browse/MAPREDUCE-1800>

# Outline

- What causes partial network partitioning?
- How do they impact systems?
- Are there any fault tolerance techniques?
- NIFTY: a generic fault tolerance technique
- Evaluation

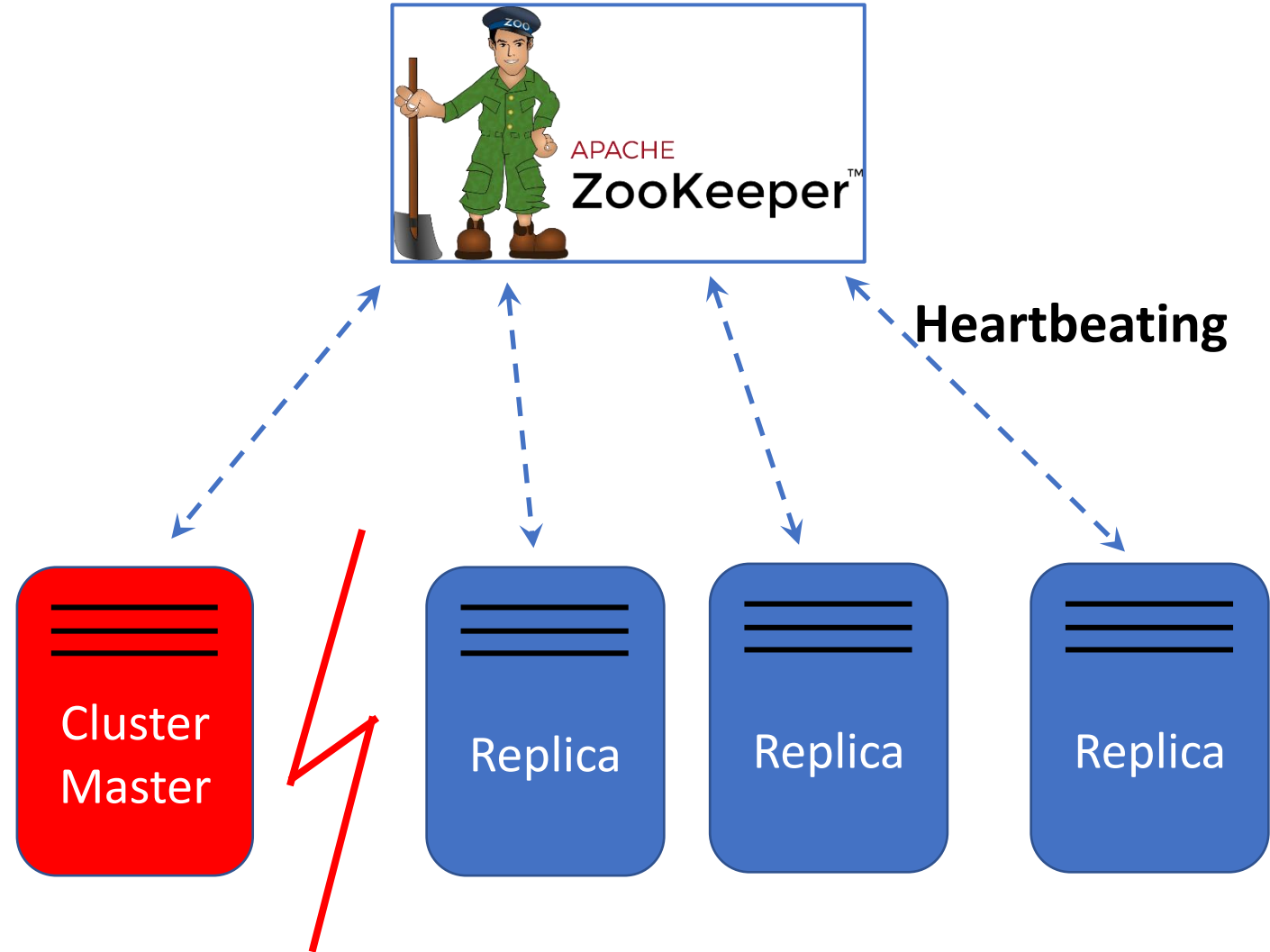


# Methodology

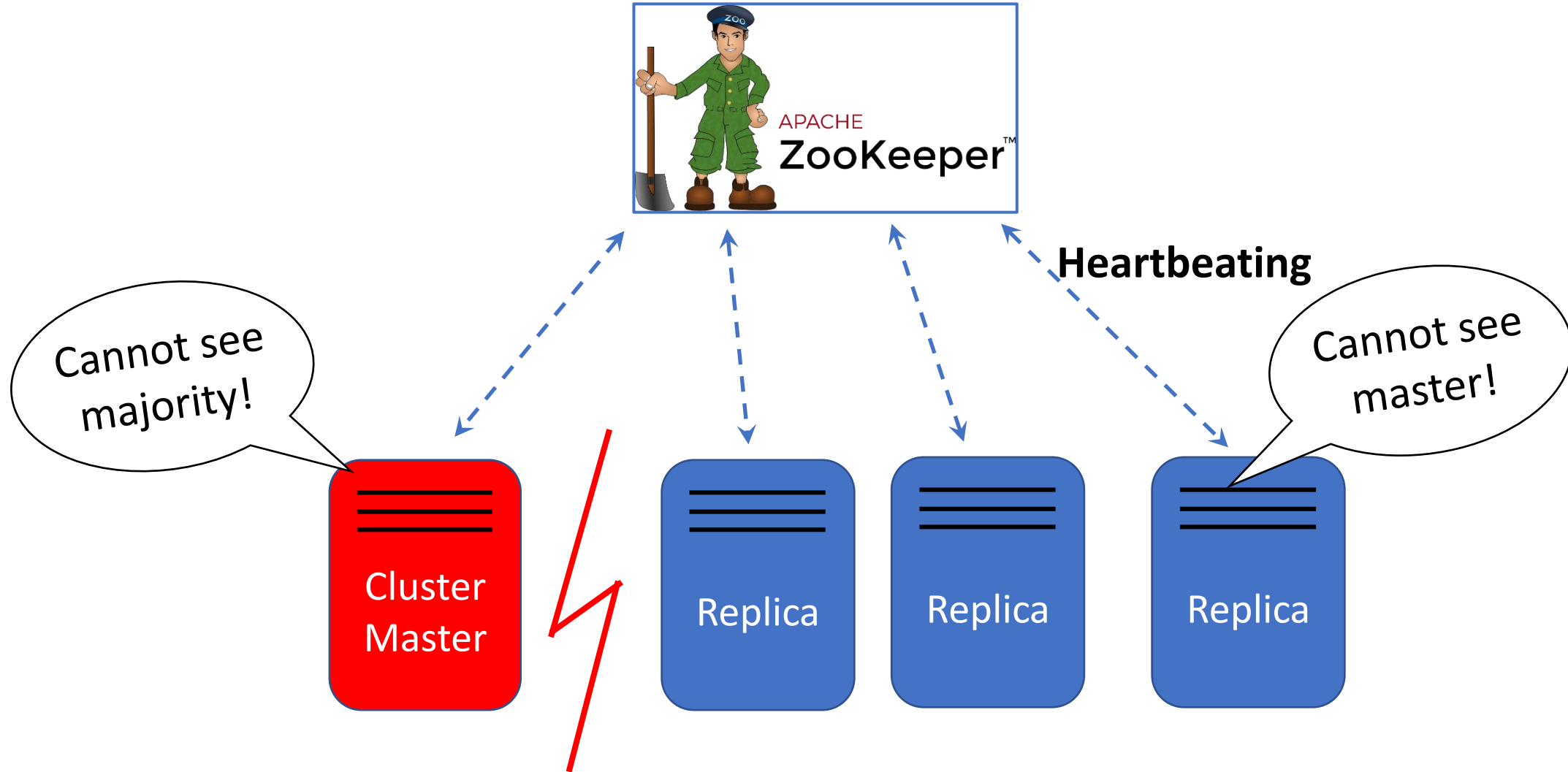
- Study 51 high-impact partial partitioning failures from 12 systems.
- Study failure report, discussion, logs, code, and tests.
- Reproduce some of the failures.



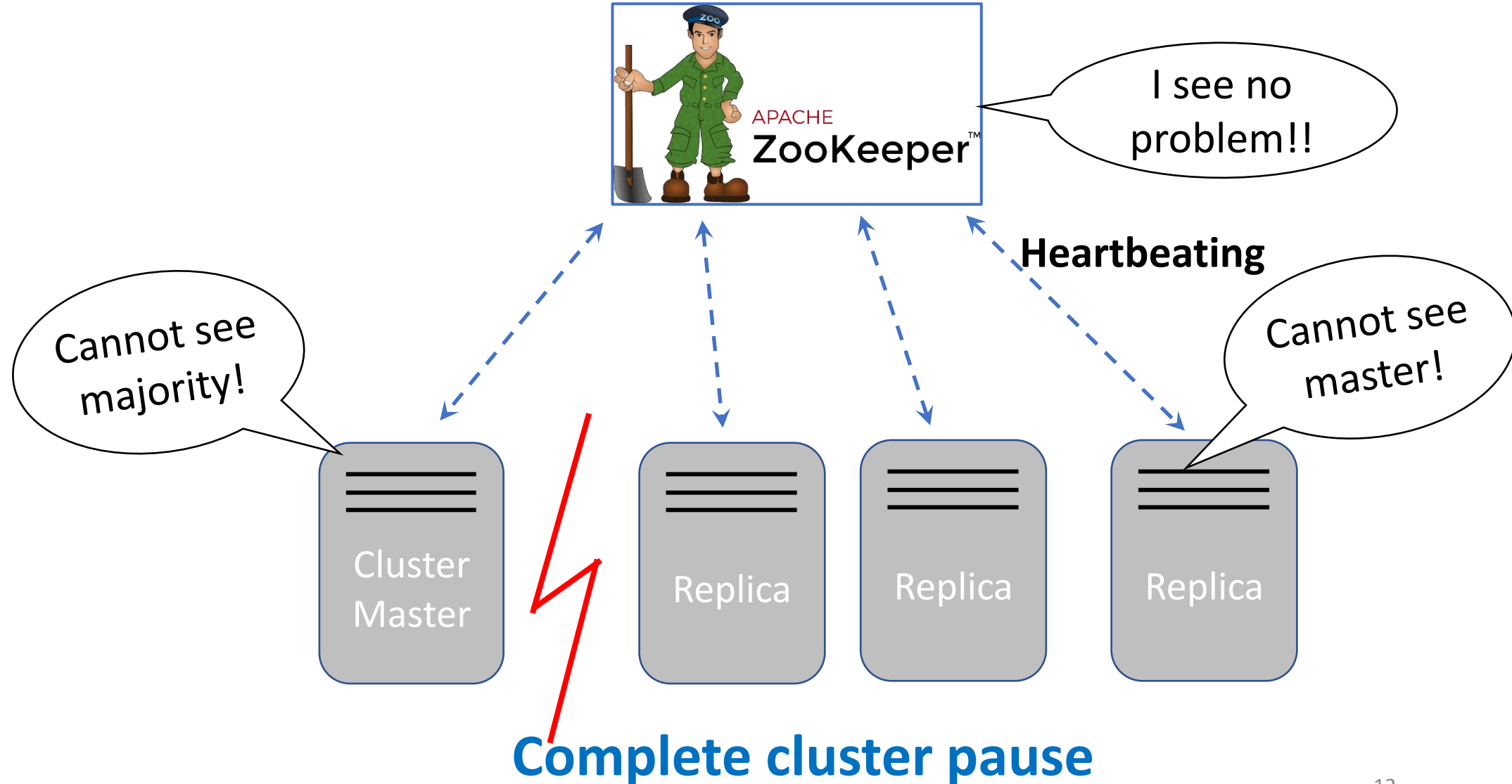
# Partial network partition – ActiveMQ



# Partial network partition – ActiveMQ

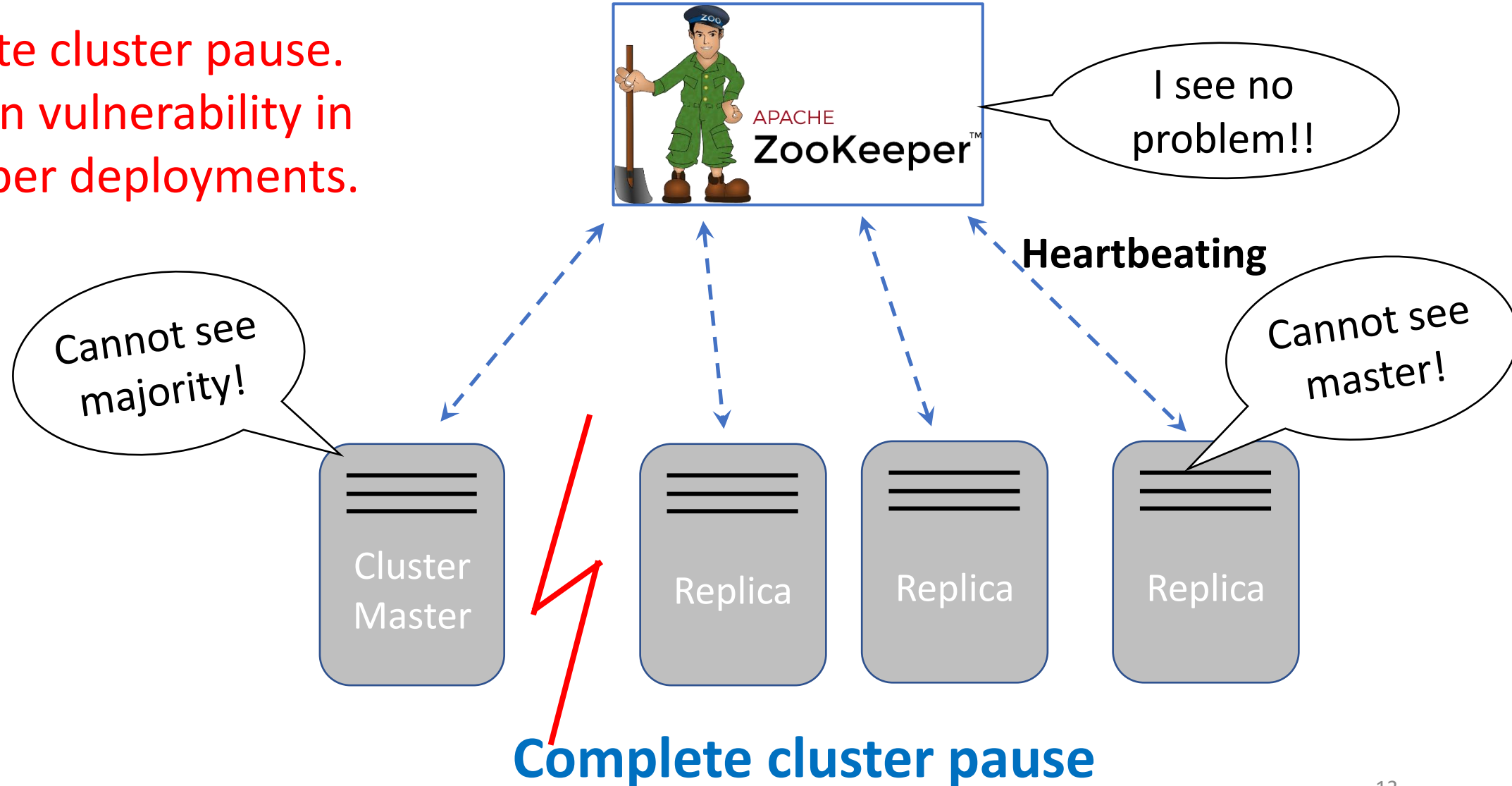


# Partial network partition – ActiveMQ

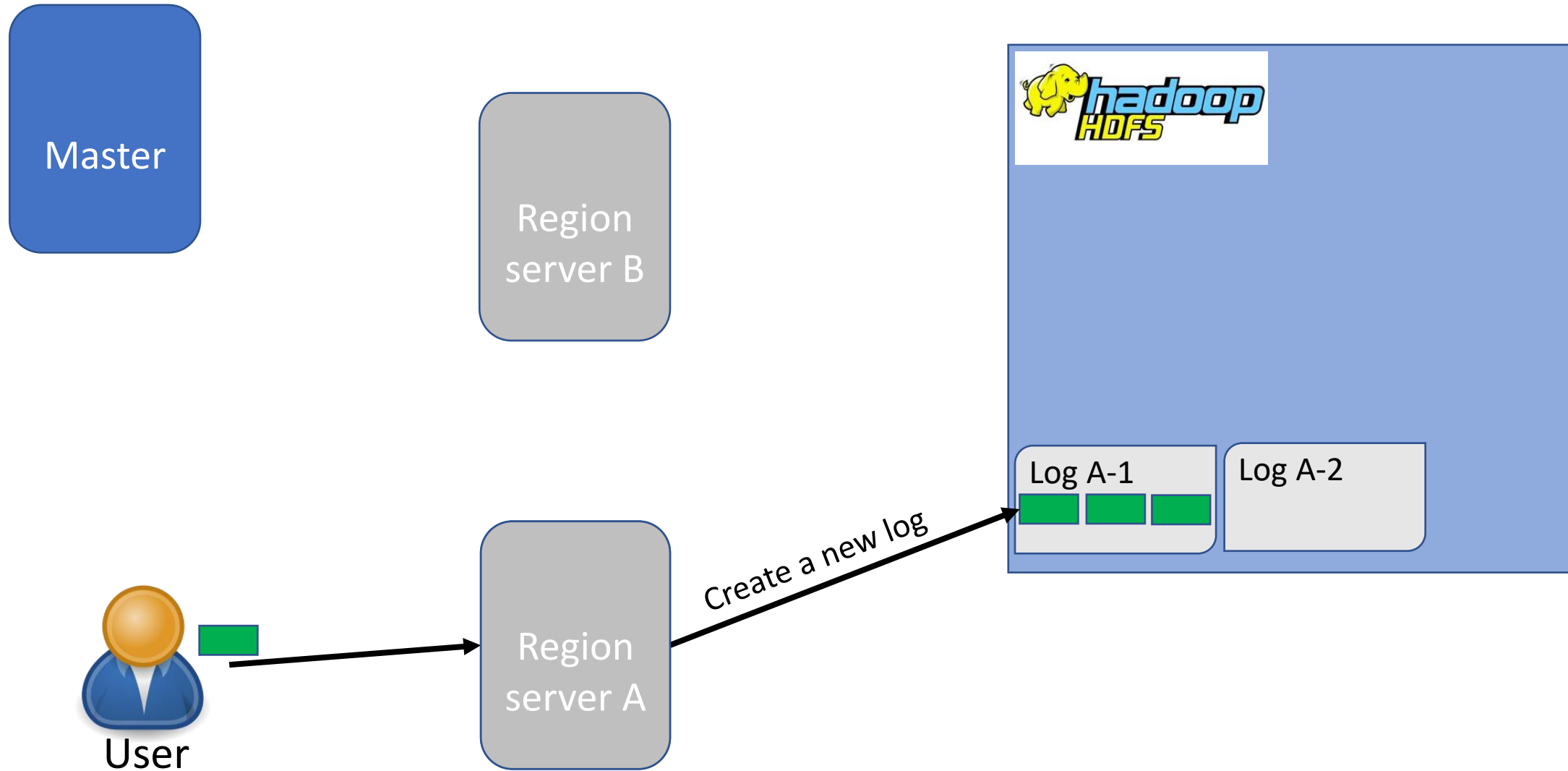


# Partial network partition – ActiveMQ

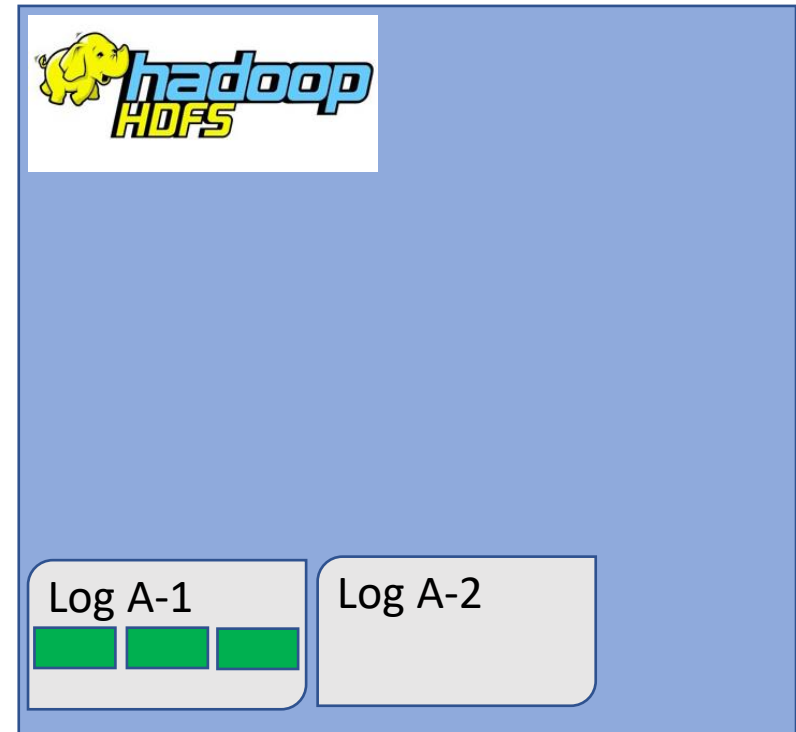
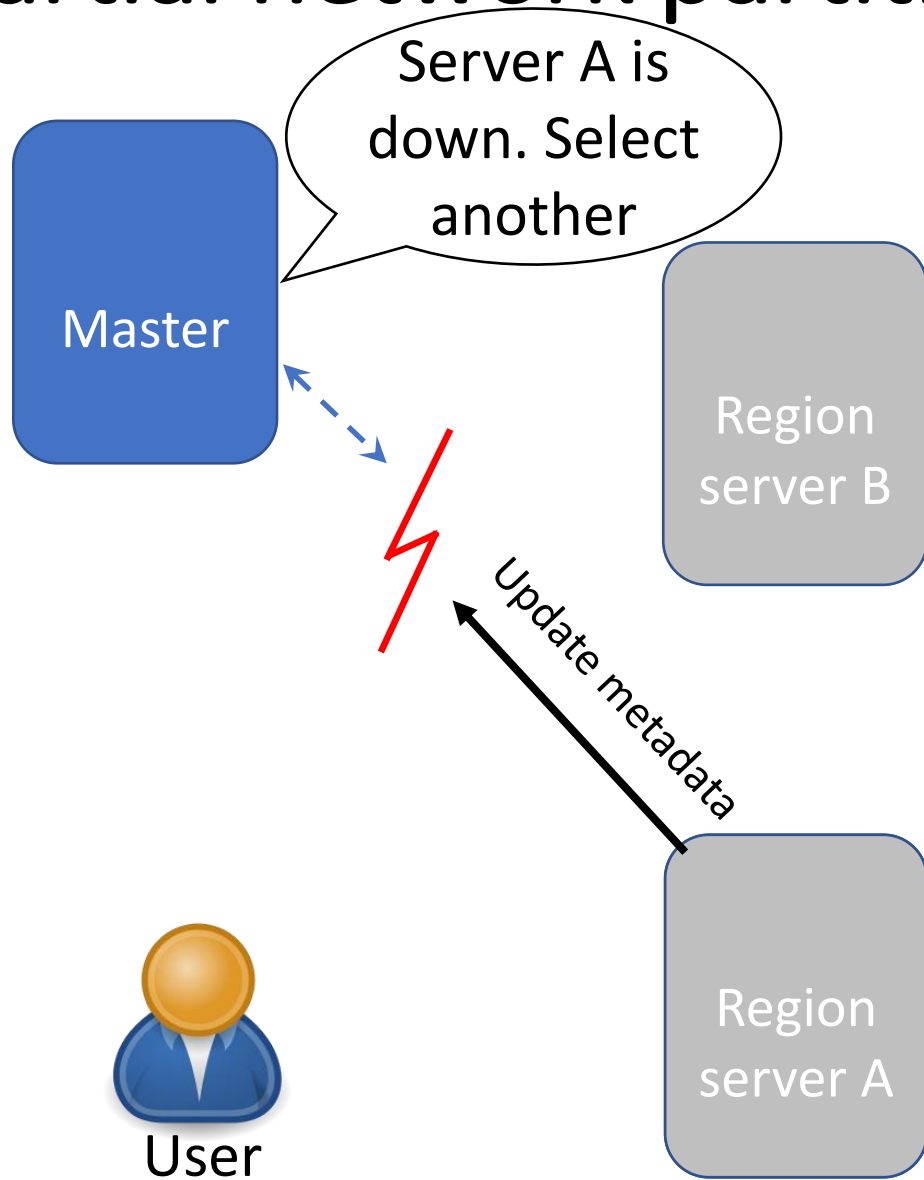
- Complete cluster pause.
- Common vulnerability in Zookeeper deployments.



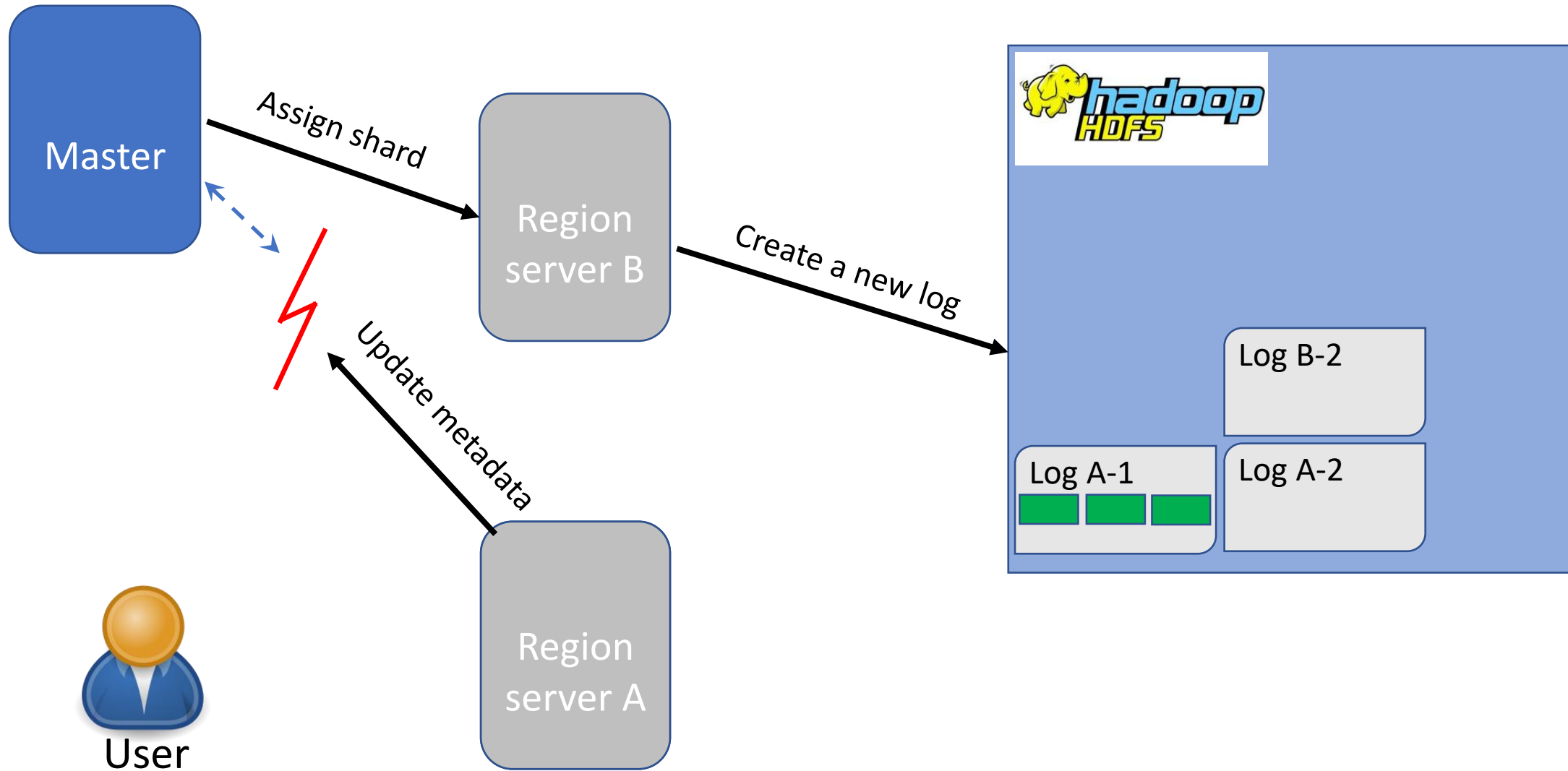
# Partial network partition – HBase



# Partial network partition – HBase

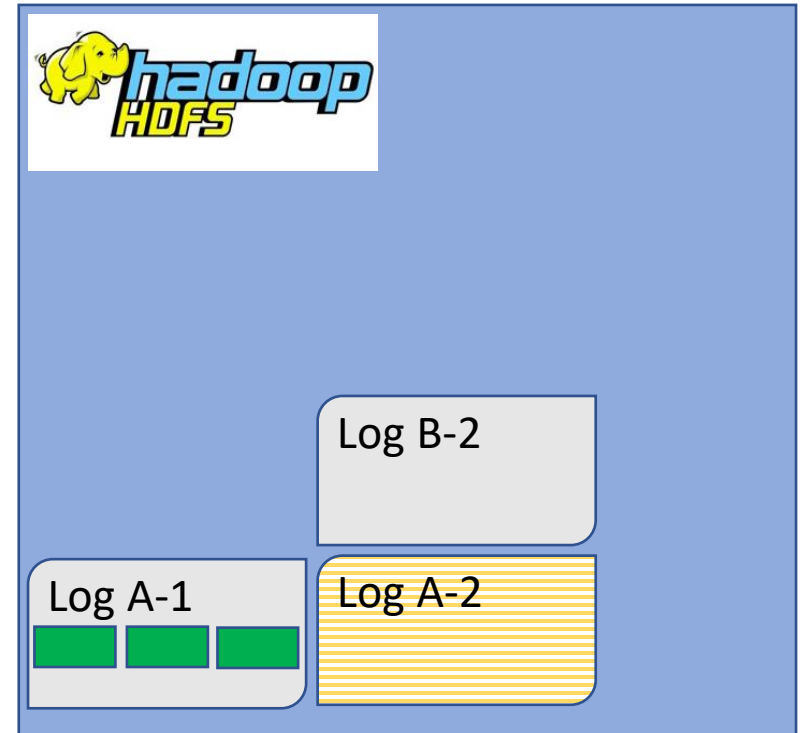
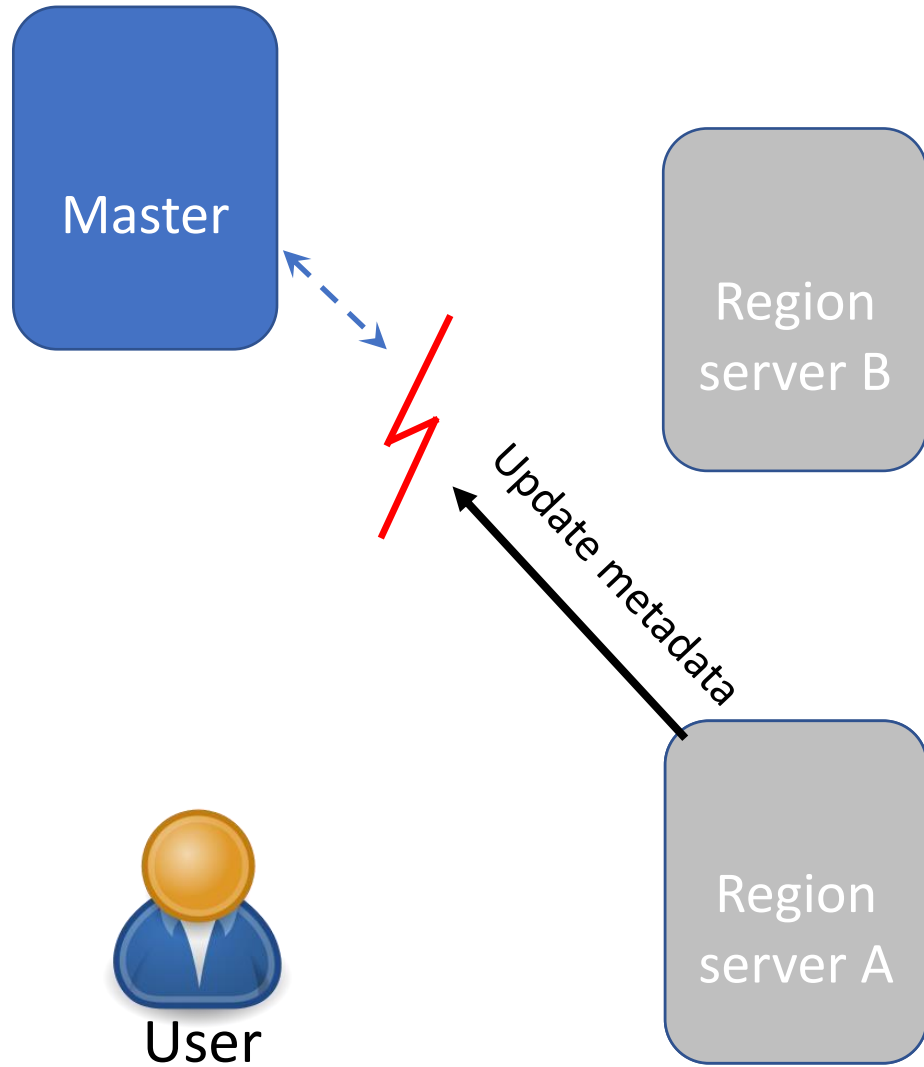


# Partial network partition – HBase

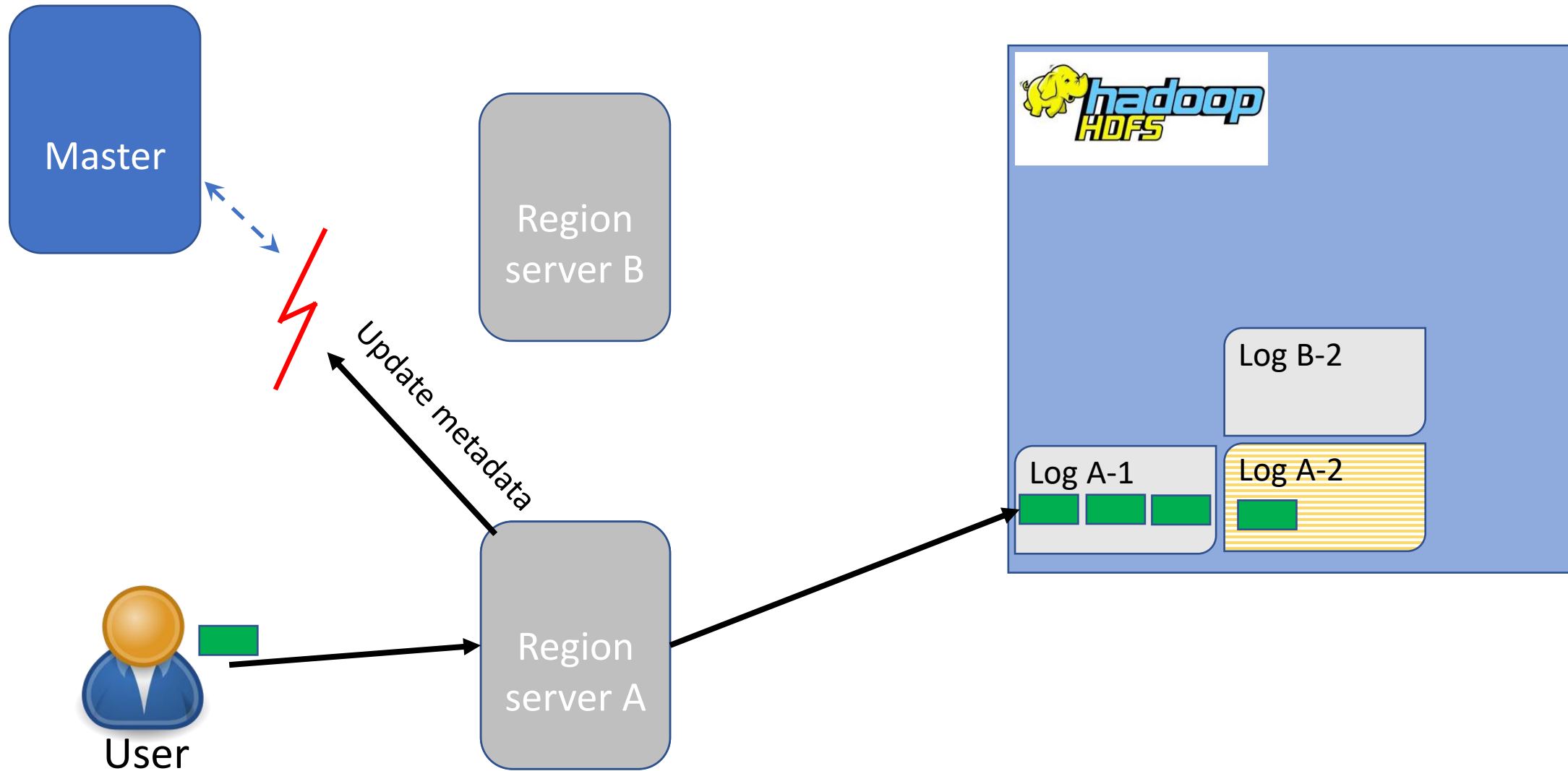




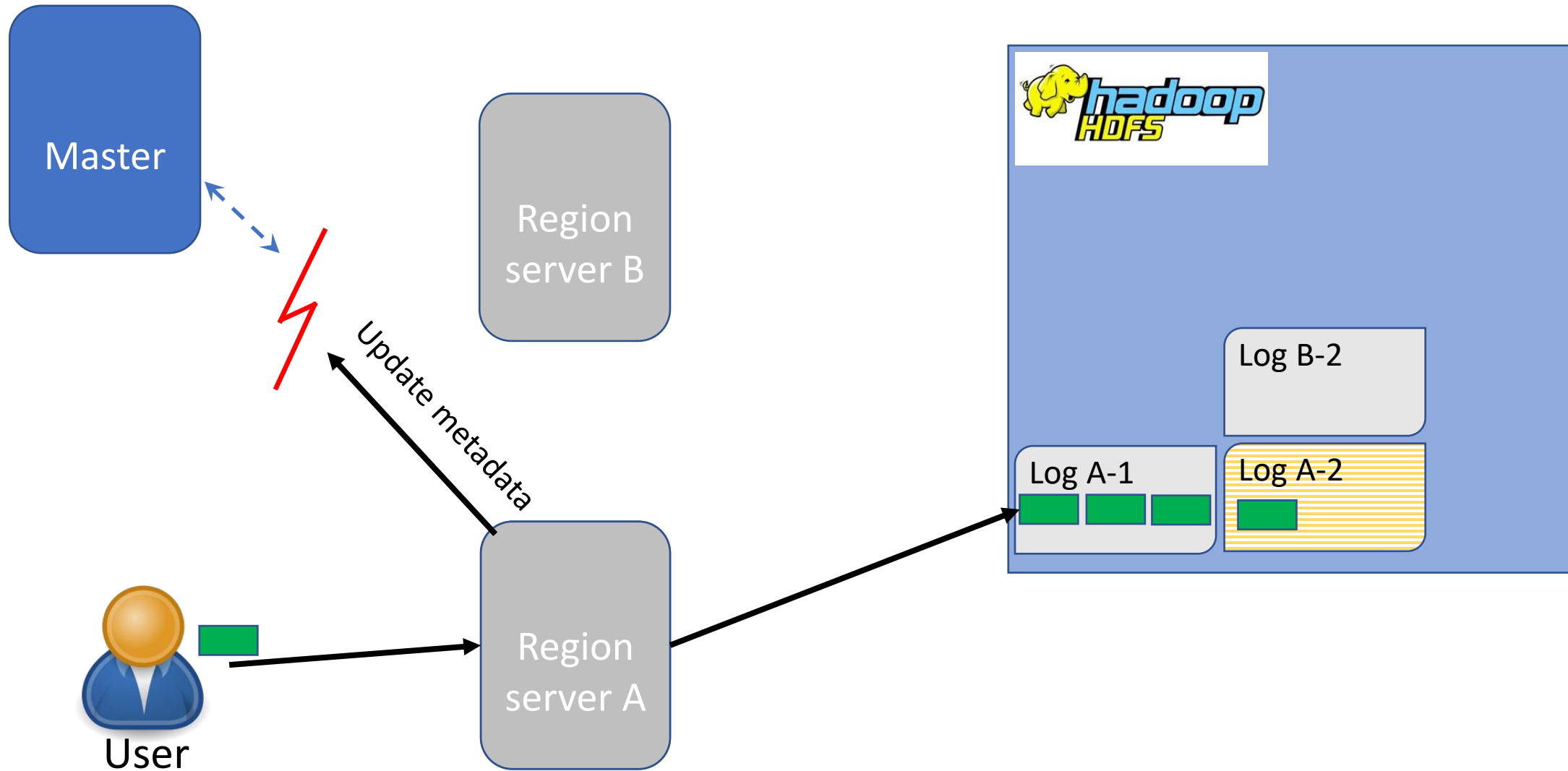
# Partial network partition – HBase



# Partial network partition – HBase

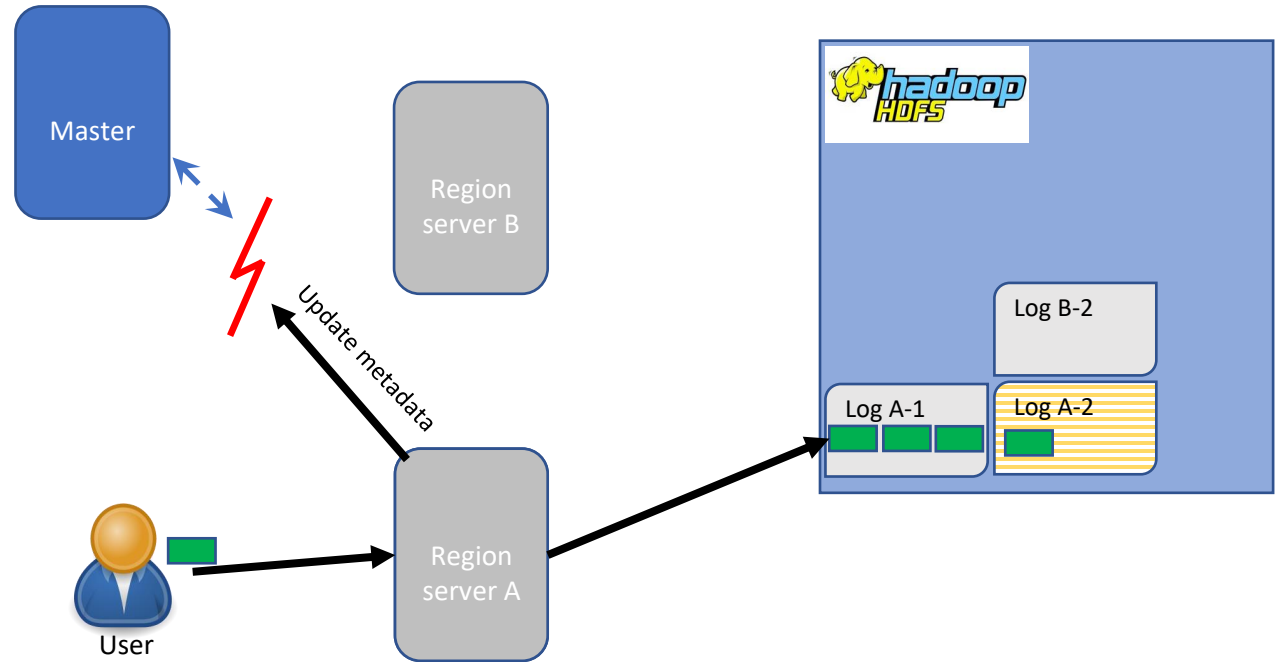


# Partial network partition – HBase



# Partial network partition – HBase

- Catastrophic: data loss.
- Easy to manifest: deterministic and requires a few events.



# What is the impact of partial partitioning?

- **Catastrophic**: 75% (e.g., data loss or corruption).
- **Silent**: 84%.
- **Permanent**: 24% have lasting impact.

# How easy are they to manifest?

- Partition only **one node**.
- **No client access** or a client access to one side: 60%
- **Three or less events**: 69%
- **Deterministic**

**Surprisingly, easy to manifest failures cause catastrophic effects.**

# Other findings

- **Vulnerable mechanisms:** leader election, config. change, and replication
- **Testability:** reproducible on 5 nodes
- **Design flaws:** majority are due to design flaws

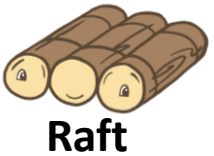
# Outline

- What causes partial network partitioning?
- How do they impact systems?
- Are there any fault tolerance techniques?
- NIFTY: a generic fault tolerance technique
- Evaluation



# Study of fault tolerance techniques

- Study the fault tolerance techniques of 8 popular systems.
- Study code patches of all studied failures.



# Current Fault Tolerance Techniques

1. Graph-based connectivity monitoring (VoltDB)
2. Checking with neighbours (Elasticsearch, RabbitMQ)
3. Failure verification (MongoDB, Raft, Elasticsearch)
4. Neutralizing partitioned nodes (Mesos, MapReduce, HBase)

# Current Fault Tolerance Techniques

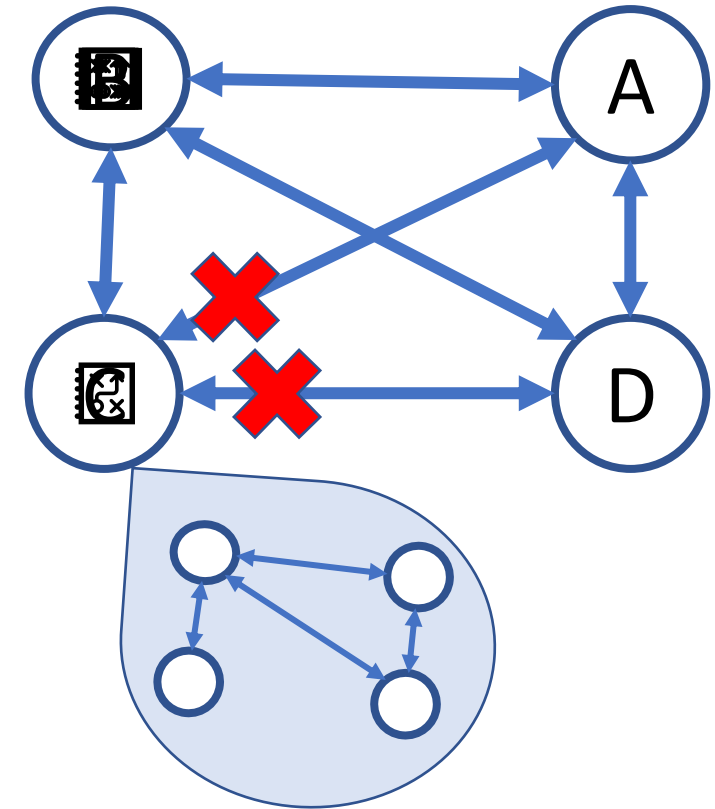
1. Graph-based connectivity monitoring (VoltDB)
2. Checking with neighbours (Elasticsearch, RabbitMQ)
3. Failure verification (MongoDB, Raft, Elasticsearch)
4. Neutralizing partitioned nodes (Mesos, MapReduce, HBase)

# Graph-based connectivity monitoring

**Idea:** Build and analyze a connectivity graph.

**How it works:**

- All-to-all heart beating
- On a partition: nodes exchange connectivity information
- Each node finds the largest fully-connected sub-graph

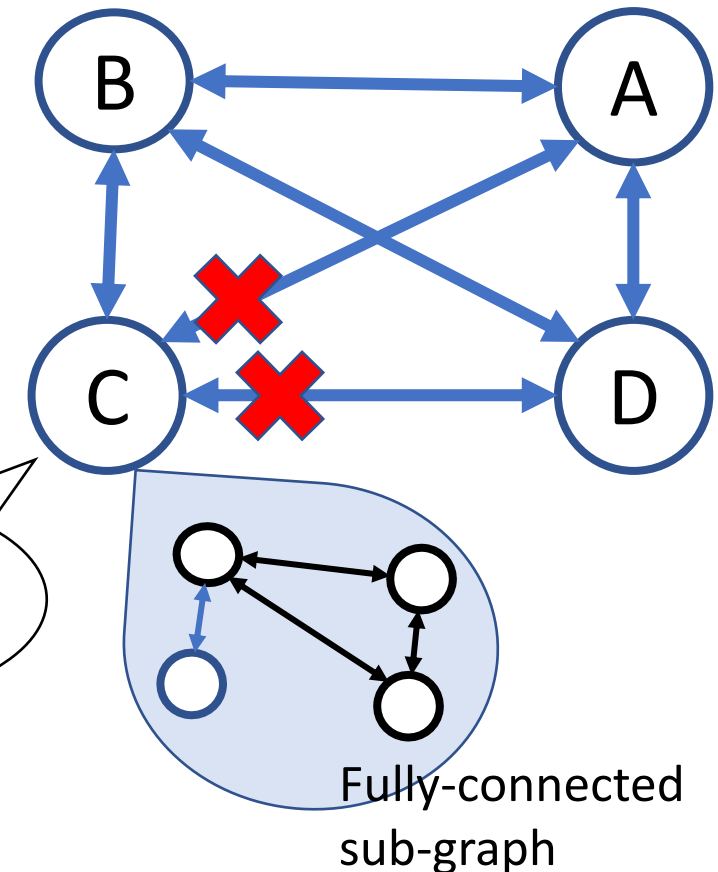
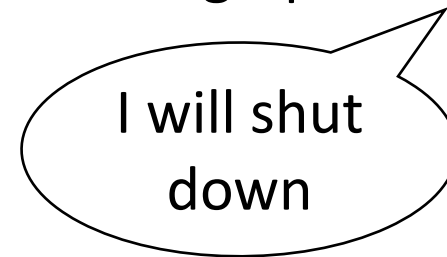


# Graph-based connectivity monitoring

**Idea:** Build and analyze a connectivity graph.

**How it works:**

- All-to-all heart beating
- On a partition: nodes exchange connectivity information
- Each node finds the largest fully-connected sub-graph
- Nodes out of the sub-graph shut down
- If any data is lost, shut down the cluster



# Graph-based Technique Shortcomings

- Unnecessarily shut down nodes.
- High chance of a complete cluster shutdown.
  - Partitioning 20% of nodes often leads to complete cluster shutdown.

# Shortcomings

	<b>Surviving Clique</b>
	VoltDB
Reduced Availability	X
Complete Unavailability	X
Complete Partition	
Double Execution	
Data Unavailability	
Scope (System/Mechanism)	S

# Shortcomings

	Surviving Clique	Checking w/ Neighbors	Failure Verification	Neutralizing Nodes
	VoltDB	Elasticsearch/RabbitMQ	MongoDB/LogCabin	MapReduce/Hbase/Mesos
Reduced Availability	X	X	X	X
Complete Unavailability	X	X		
Complete Partition		X		
Double Execution				X
Data Unavailability		X		
Scope (System/Mechanism)	S	M/S	M	M

**All current fault tolerance techniques have severe shortcomings.**



# Outline

- What causes partial network partitioning?
- How do they impact systems?
- Are there any fault tolerance techniques?
- **NIFTY: a generic fault tolerance technique**
- Evaluation

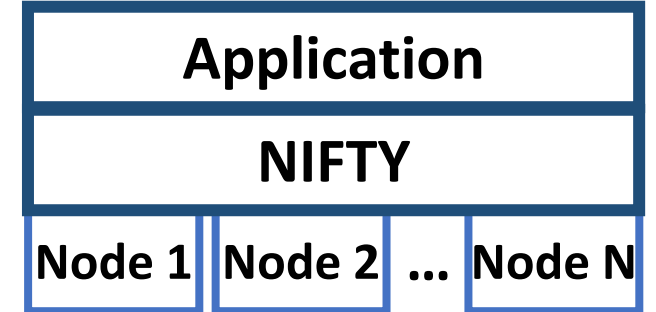
# NIFTY

A **N**etwork partitioning **f**ault **t**olerance layer (NIFTY)

Goals:

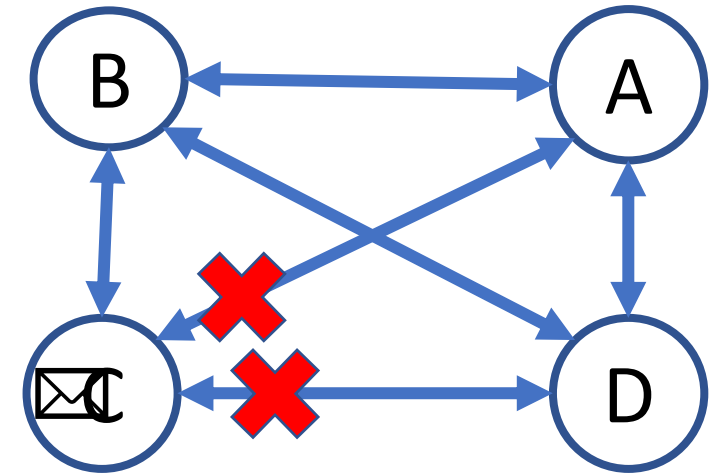
- System agnostic
- No changes to existing systems
- Negligible overhead

Insight: leverage existing monitoring techniques to detour traffic around partial partitions.



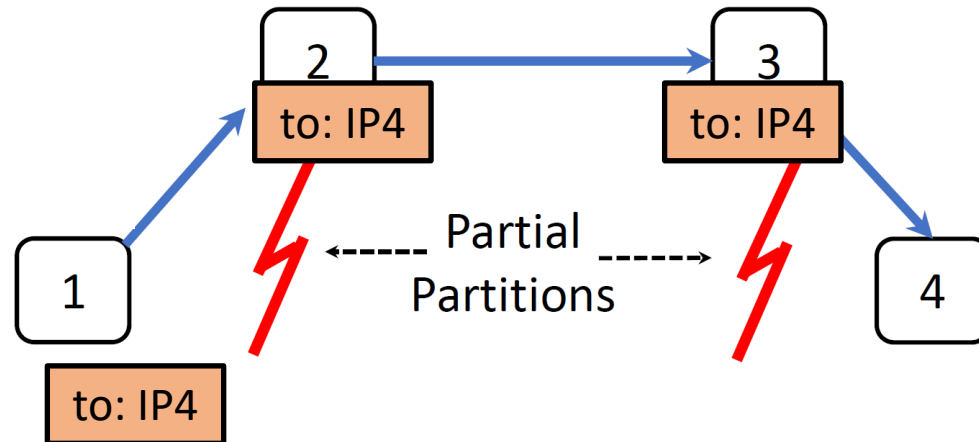
# How NIFTY works

- Use heartbeats to detect partial partitions
- On a partial partition: detour packets through intermediate nodes
- Use distance vector routing
- Use OpenVSwitch to deploy routes on end nodes



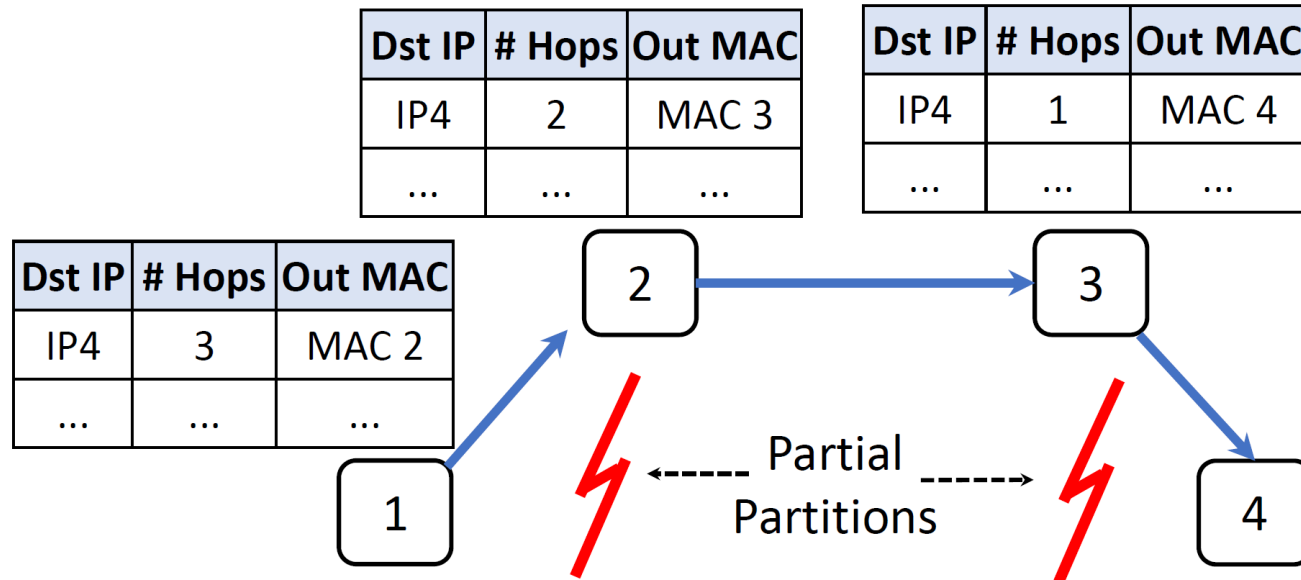
# How NIFTY works

Rerouting done through MAC address manipulation



# How NIFTY works

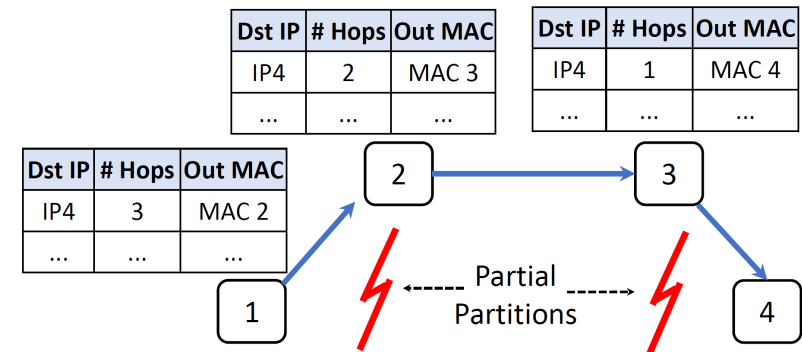
Rerouting done through MAC address manipulation



# How NIFTY works

Rerouting done through MAC address manipulation

- Simple
- Agnostic to system running atop of it
- Transparently masks partial partitions



# Outline

- What causes partial network partitioning?
- How do they impact systems?
- Are there any fault tolerance techniques?
- NIFTY: a generic fault tolerance technique
- Evaluation

# Evaluation

- What is Nifty's overhead?
- How systems perform under a partial partition?
- How does nifty scale for large clusters?
- What is the utility of Nifty's classification API?



# Evaluation

- What is Nifty's overhead?
- How systems perform under a partial partition?
- How does nifty scale for large clusters?
- What is the utility of Nifty's classification API?

# Evaluation setup

- Measure the impact of Nifty on 6 systems.
- 40 nodes in Cloudfab Utah cluster.



mongoDB



elastic



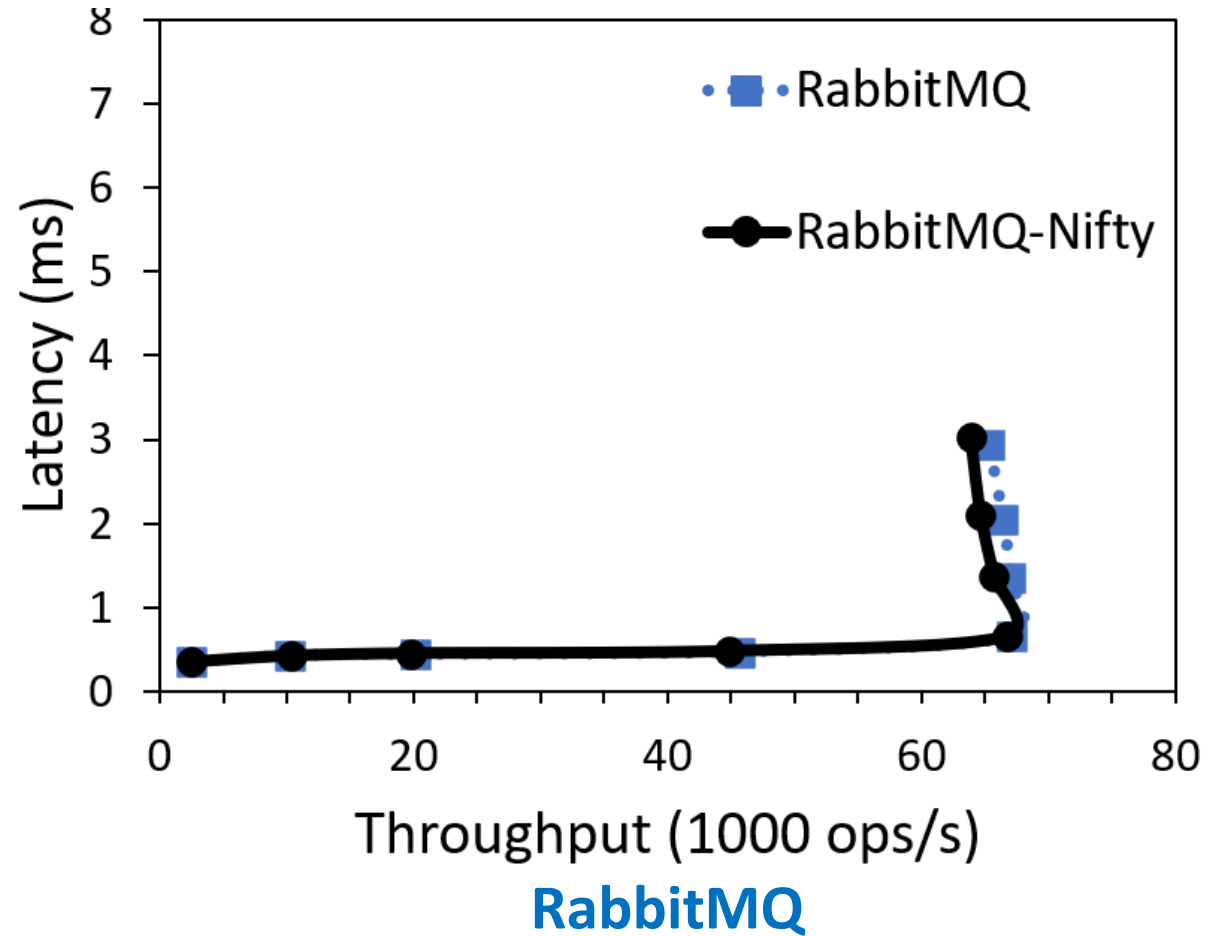
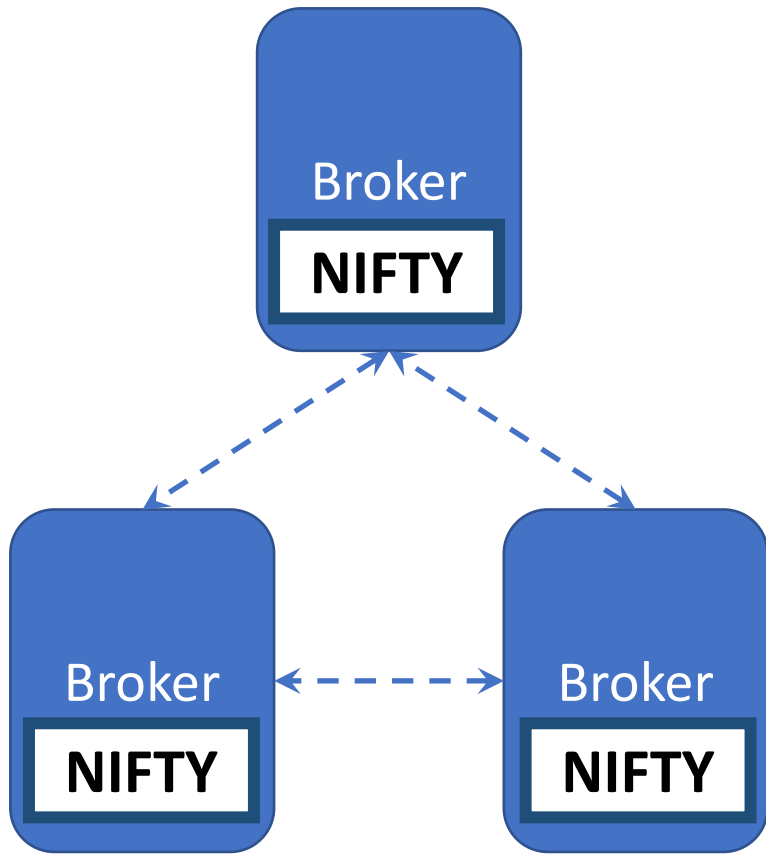
kafka

**VOLT**DB

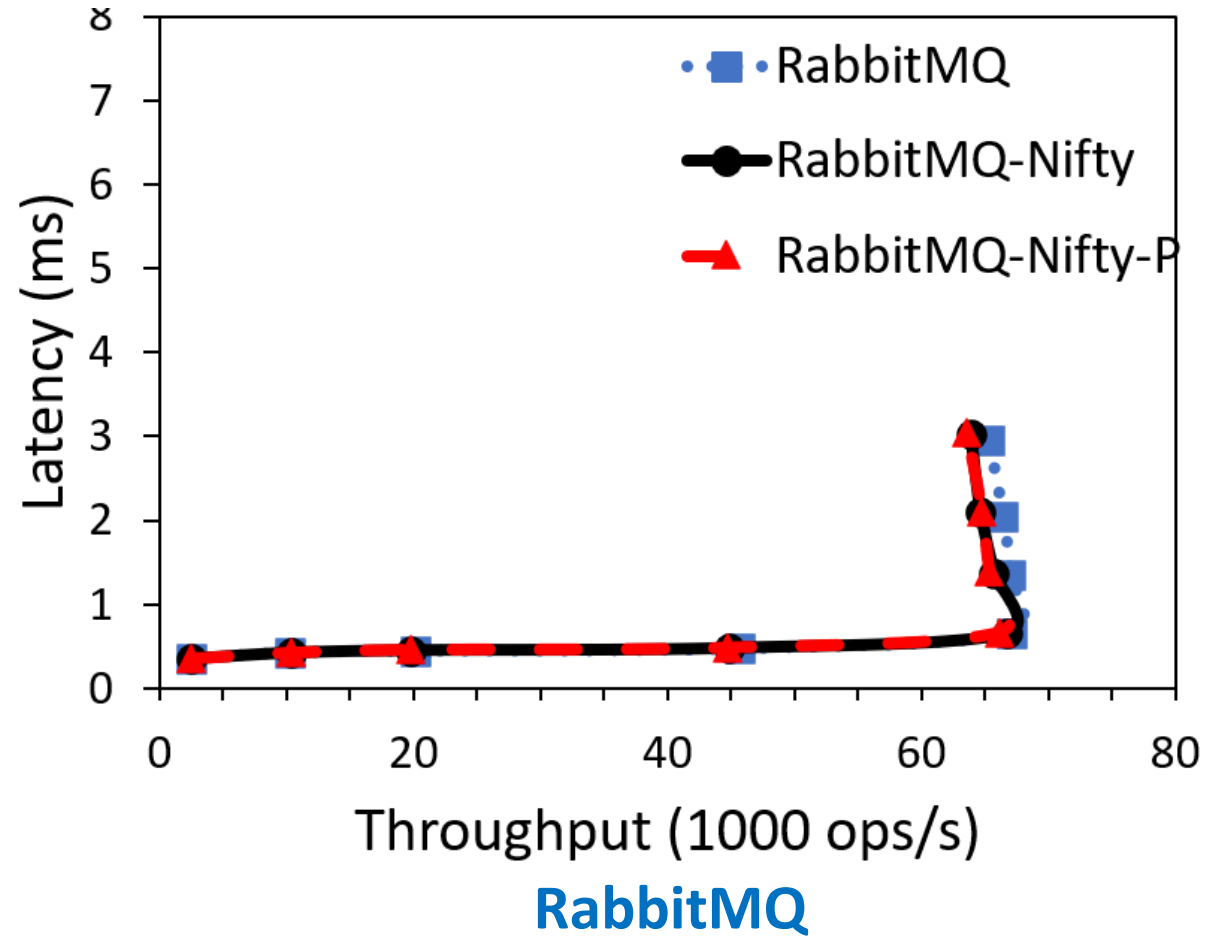
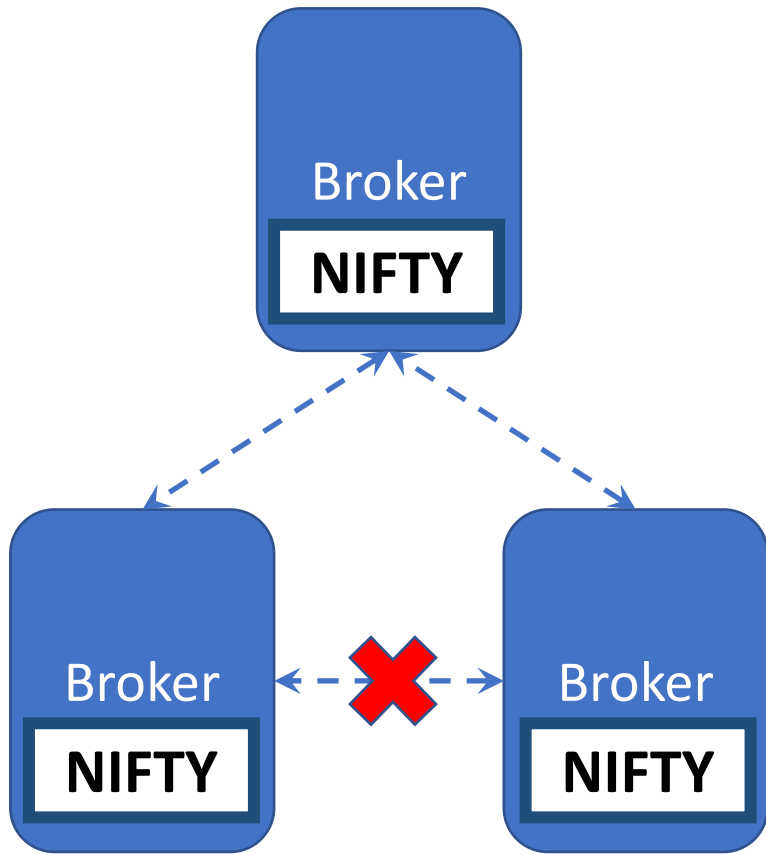


RabbitMQ

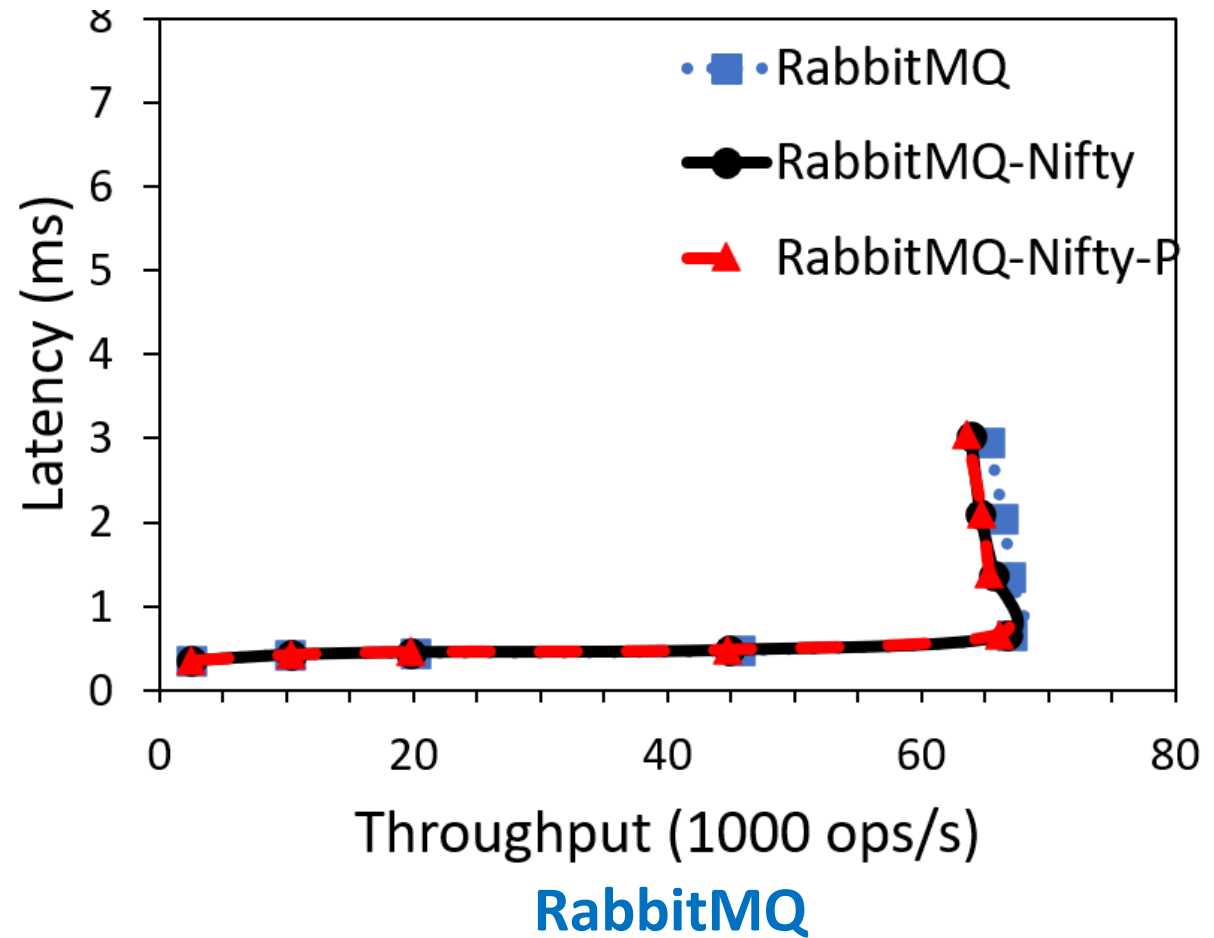
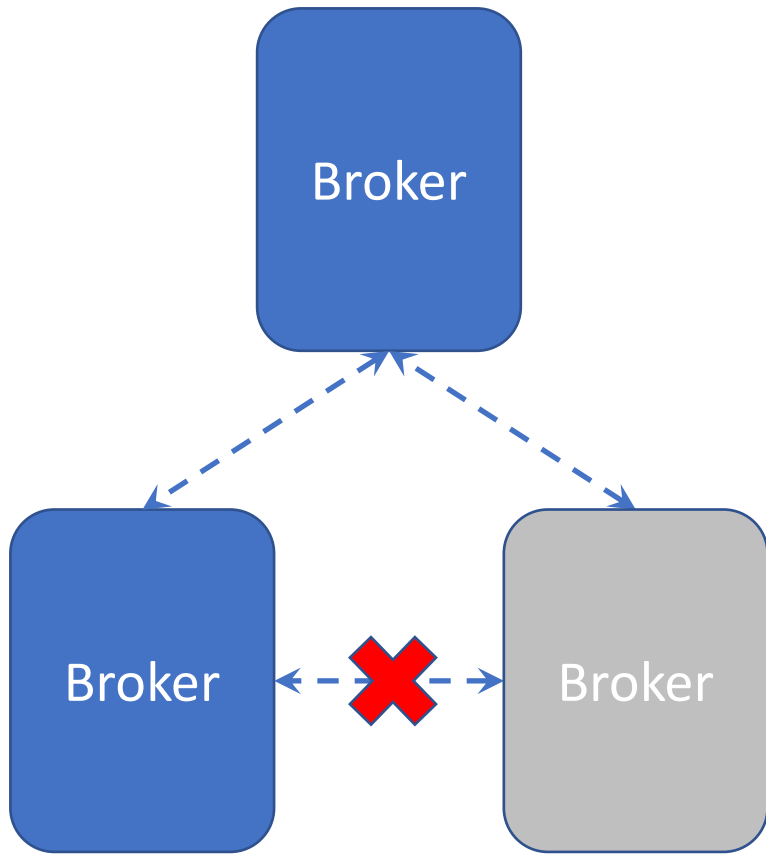
# Evaluation: Overhead - RabbitMQ



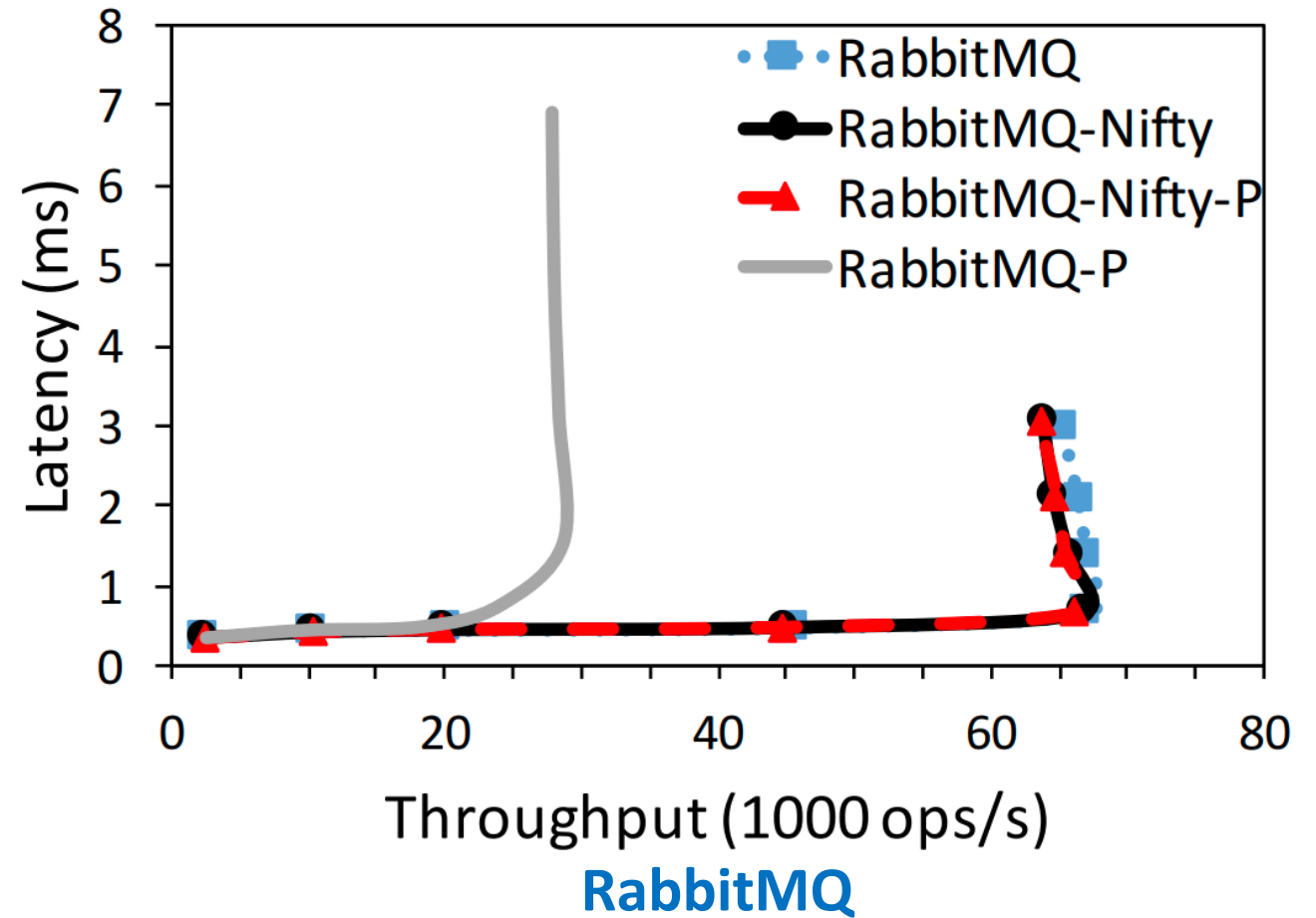
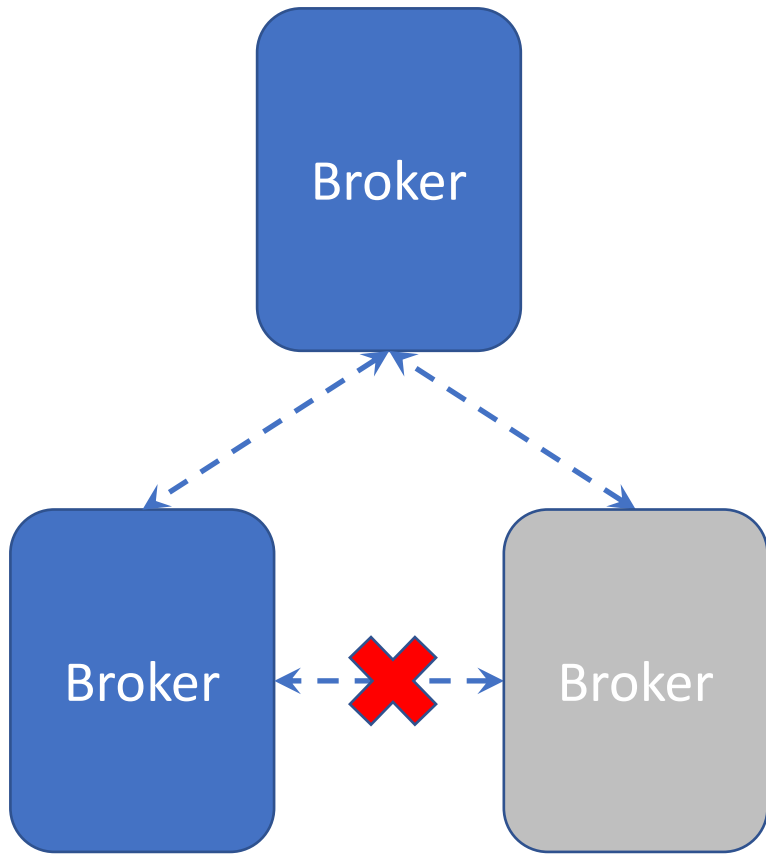
# Evaluation: Performance with a Partition



# Evaluation: Performance with a Partition



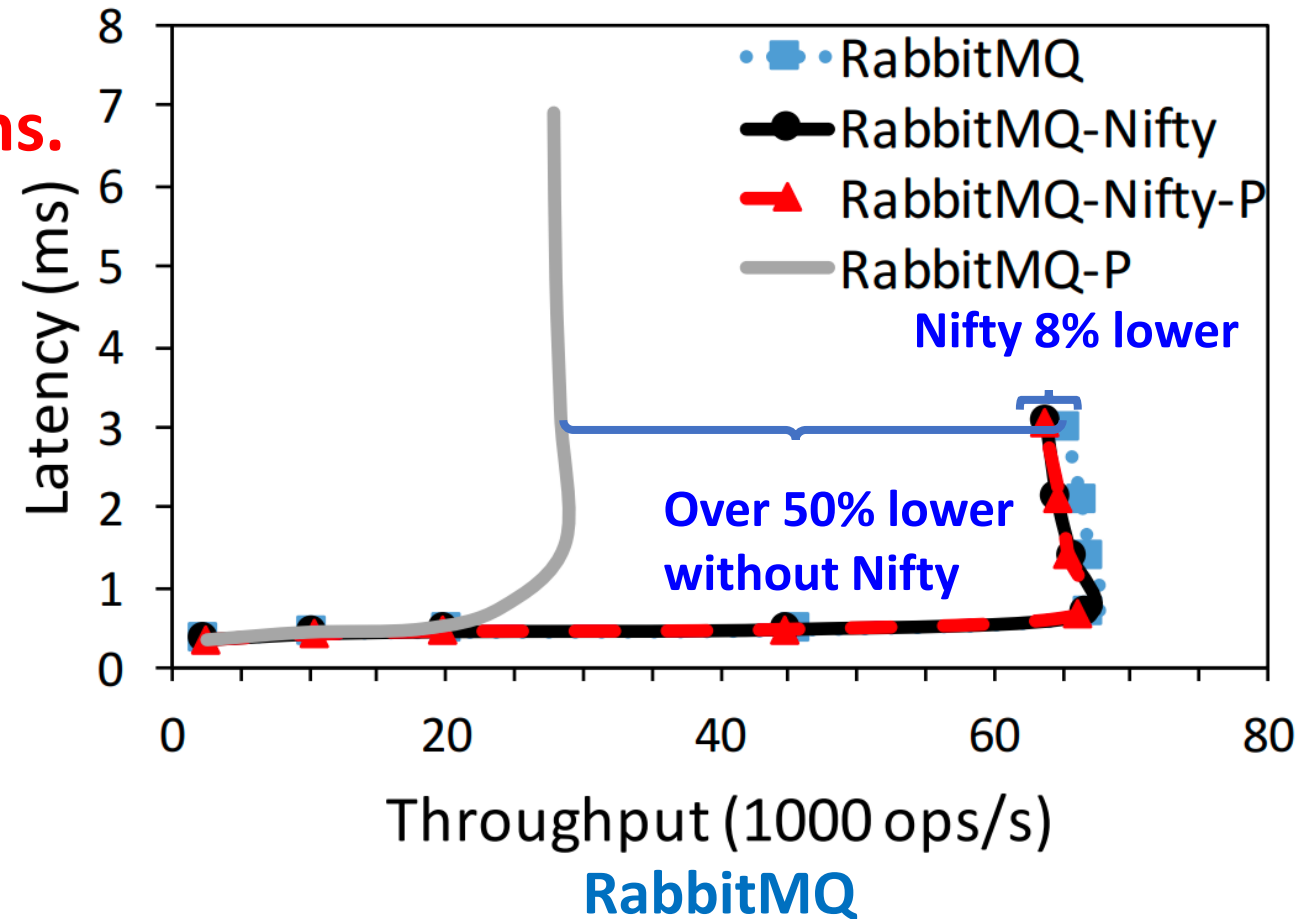
# Evaluation: Performance with a Partition



# Evaluation: Performance with a Partition

**NIFTY has a negligible overhead.**

**Effectively mask partial partitions.**



# Conclusion

- First comprehensive study of partial partitioning failures:
  - Failures are catastrophic
  - Failures are easy to manifest
- First study of current fault tolerance techniques:
  - All current techniques have severe shortcomings
- Built Nifty
  - Simple
  - Transparent
  - Low overhead



# Thank you!

Source code available at: <https://wasl.uwaterloo.ca/projects/nifty/>