Fault Tree Analysis Applied to Apache Kafka®
Agenda

The Challenge: Quantify Kafka Reliability

Introduction to Fault Tree Analysis

Kafka Fault Trees
  Availability
  Data Durability

Conclusion
The Challenge: Quantify Kafka Reliability
What are we trying to do?

Kafka is a “reliability tool”

Move data without lossiness

High stakes usage
Observability Data
Event Streaming
Change Data Capture
Why Quantify?

Determine probability of success

Find opportunities to trim cost
Defining SLOs

Need to define Service Level Objectives

- Availability
- Durability
- Latency
Quantifying SLOs

Availability

What is the probability that writes or reads fail?

How long do we tolerate downtime?
Quantifying SLOs

Durability

What is the probability that we’ll lose data?

How much will we lose?
Quantifying SLOs

Latency

How long are transactions allowed to take?
Introduction to Fault Tree Analysis
What is Fault Tree Analysis?

Deductive Failure Analysis

Invented in 1962 for Minuteman I ICBM Launch Control System

Industry wide adoption

Aerospace

Military

Petrochemical

Et al.
Fault Tree Analysis: Event Symbols

- Basic
- Intermediate
- Transfer
Fault Tree Analysis: Gate Symbols

OR

AND
Fault Tree Analysis: OR Example

- Write failure
- RAID0
- Disk failure
- Disk1
- Disk failure
- Disk2
- Disk failure
- Disk3
Fault Tree Analysis: OR Example

4% probability of failure
Fault Tree Analysis: OR Example

4% probability of failure annualized
Fault Tree Analysis: OR Example

\[ p(A \text{ “or” } B) = p(A) + p(B) - p(A) \times p(B) \]

Almost always small
Fault Tree Analysis: OR Example

12% chance of failure annualized
Fault Tree Analysis: AND Example

Write failure

RAID1

Disk failure

Disk1

Disk2

Disk3

Disk failure

Disk failure
Fault Tree Analysis: AND Example

\[ p(A \text{ and } B) = p(A) \times p(B) \]
Fault Tree Analysis: AND Example

99.9936% chance of success annualized
Fault Tree Analysis: AND Example

99.9936% chance of success annualized if we don’t remediate
Fault Tree Analysis: AND Example

99.999999996% chance of success if we remediate within 3 days
Fault Tree Analysis: AND Example

99.99999% chance of success if we remediate within 3 days
Fault Tree Analysis: AND Example

\[ p(A \text{ and } B) = \]
\[ p(A) \times p(B) = \]
\[ (1 - e^{-p(A)^*t}) \times (1 - e^{-p(B)^*t}) \]

Where \( t = \) time to remediate

If \( p(A) \) and \( p(B) < .1 \), approximate to \( p(A)^*p(B)^*t^2 \)
Kafka Fault Trees
Availability

Can we write or read to a Kafka cluster?

Service Level Objective (SLO):

99.99% success rate per year
Availability

Write or read to single broker

Write-Read

Standalone Zookeeper Failure

SingleZookeeper

Broker failure

Broker

Disk Fails

Network partition

Other HW failure

OS Faults

Disk

Network

System

OS

4%

2%

1%

1%
Availability

Write or read to single broker

12.8% failure chance

Write-Read

4.8% failure chance

Standalone Zookeeper Failure

Zookeeper Cluster

8% failure chance

Broker failure

Solid State Disk failure

Network partition

OS Faults

Other HW failure

Disk Fails

Network partition

Other HW failure

OS Faults

SSD

Network

OS

System

Disk

Network

System

OS

.8%

2%

1%

1%

4%

2%

1%

1%
Availability - Two brokers, single ZK
Availability - Collapse Host Faults

98.4% success chance
Availability - Multiple ZKs

99.4% success chance
Availability - Three Brokers

99.95% success chance
## Availability - Summary

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Success Probability</th>
<th>Cost Per Nine*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone</td>
<td>87.2%</td>
<td>n/a</td>
</tr>
<tr>
<td>Two brokers, ISR=1, One ZK</td>
<td>98.36%</td>
<td>2</td>
</tr>
<tr>
<td>Two brokers, ISR=1, Three ZKs</td>
<td>99.36%</td>
<td>2</td>
</tr>
<tr>
<td>Two brokers, ISR=1, Five ZKs</td>
<td>99.36%</td>
<td>3</td>
</tr>
<tr>
<td>Three brokers, ISR=1, Three ZKs</td>
<td>99.95%</td>
<td>1.5</td>
</tr>
<tr>
<td>Three brokers, ISR=2, Three ZKs</td>
<td>99.36%</td>
<td>2.25</td>
</tr>
</tbody>
</table>

* Cost is computed in “disk units” / “number of nines”:
  - Kafka Broker Rotational Disk = .5
  - Zookeeper SSD Disk = 1
  Lower is better
## Availability - Four Brokers

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Success Probability</th>
<th>Cost Per Nine*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Three brokers, ISR=1, Three ZKs</td>
<td>99.95%</td>
<td>1.5</td>
</tr>
<tr>
<td>Three brokers, ISR=2, Three ZKs</td>
<td>99.36%</td>
<td>2.25</td>
</tr>
<tr>
<td>Four brokers, ISR=1, Three ZKs</td>
<td>99.995%</td>
<td>1.25</td>
</tr>
<tr>
<td>Four brokers, ISR=2, Three ZKs</td>
<td>99.95%</td>
<td>1.67</td>
</tr>
</tbody>
</table>

* Cost is computed in “disk units” / “number of nines”:
  - Kafka Broker Rotational Disk = .5
  - Zookeeper SSD Disk = 1
  - Lower is better
## Availability - Broker SSD

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Success Probability</th>
<th>Cost Per Nine*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone</td>
<td>90.4%</td>
<td>1</td>
</tr>
<tr>
<td>Two brokers, ISR=1, One ZK</td>
<td>99.08%</td>
<td>1.5</td>
</tr>
<tr>
<td>Two brokers, ISR=1, Three ZKs</td>
<td>99.77%</td>
<td>2.5</td>
</tr>
<tr>
<td>Two brokers, ISR=1, Five ZKs</td>
<td>99.77%</td>
<td>3.5</td>
</tr>
<tr>
<td>Three brokers, ISR=1, Three ZKs</td>
<td>99.99%</td>
<td>1.5</td>
</tr>
<tr>
<td>Three brokers, ISR=2, Three ZKs</td>
<td>99.77%</td>
<td>3</td>
</tr>
</tbody>
</table>

* SSD Disk = 1
## Availability - Broker EBS

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Success Probability</th>
<th>Cost Per Nine*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone</td>
<td>91.6%</td>
<td>1.5</td>
</tr>
<tr>
<td>Two brokers, ISR=1, One ZK</td>
<td>99.29%</td>
<td>2</td>
</tr>
<tr>
<td>Two brokers, ISR=1, Three ZKs</td>
<td>99.82%</td>
<td>3.0</td>
</tr>
<tr>
<td>Two brokers, ISR=1, Five ZKs</td>
<td>99.82%</td>
<td>3.5</td>
</tr>
<tr>
<td>Three brokers, ISR=1, Three ZKs</td>
<td>99.99%</td>
<td>1.875</td>
</tr>
<tr>
<td>Three brokers, ISR=2, Three ZKs</td>
<td>99.82%</td>
<td>3.75</td>
</tr>
</tbody>
</table>

* EBS disk units:
EBS SSD Disk = 1.5

Assumption that EBS fails at .2%
Durability

What are the chances of losing data?

Service Level Objective (SLO):

99.999999% durability per year
Durability

We lose data when all hosts with replicas go down

Assumptions:

- 6TB per broker (2TB per disk w/ RAID)
- 70MB/s replication rate
- ~24 hours to replicate full broker
- We replace bad hosts almost immediately
Durability - Two brokers - One 6TB Disk

\[ 1 - e^{-p(A) \cdot t} \times 1 - e^{-p(B) \cdot t} \]

99.999999% durability
Durability - Add Raid0

99.9999% durability
Durability - Three brokers

99.99999999% durability
Durability

Assumption:

48TB per broker

70MB/s replication rate

~8 days to replicate full broker

We replace bad hosts almost immediately
Durability - Three brokers 24 disks

99.999% completeness
# Durability - Summary

<table>
<thead>
<tr>
<th></th>
<th>Data completeness</th>
<th>Cost Per Nine*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone</td>
<td>99.99%</td>
<td>.125</td>
</tr>
<tr>
<td>Two brokers</td>
<td>99.999999%</td>
<td>.5</td>
</tr>
<tr>
<td>Two brokers RAID0</td>
<td>99.9999%</td>
<td>.22</td>
</tr>
<tr>
<td>Three brokers RAID0</td>
<td>99.999999999%</td>
<td>.15</td>
</tr>
<tr>
<td>Three brokers RAID0 - 48TB</td>
<td>99.999%</td>
<td>1</td>
</tr>
</tbody>
</table>

* Cost is computed in “disk units” / “number of nines”:
  - Single non-raid disk = .5
  - Raid0 = .167
  - Zookeeper SSD Disk = 1
  - Lower is better
Latency

FTA focused on failures
Latency is not an inherent failure
Experiment with worst-case scenarios
Conclusion
Tools and References

Fault Tree Models: [github.com/afalko/fta-kafka](https://github.com/afalko/fta-kafka)

OSS tool to draw and compute models: [github.com/rakhimov/scram](https://github.com/rakhimov/scram)

*How Not to Go Boom: Lessons for SREs from Oil Refineries* by Emil Stolarsky

*Fault Tree Analysis - A History* by Clifton A. Ericson II

*Fault Tree Handbook with Aerospace Applications* by Dr. Michael Stamatelatos and Mr. José Caraballo

*Failure Trends in a Large Disk Drive Population* by Eduardo Pinheiro, Wolf-Dietrich Weber and Luiz Andre Barroso

*Solving Data Loss in Massive Storage Systems* by Jason Resch

*Failures at Scale and How to Ride Through Them* by James Hamilton
Takeaways

FTA can be applied to:
- Kafka Availability and Durability SLOs
- Find cost savings
- Uncover decisions that reduce reliability
Future Work

Kafka on Kubernetes analysis

**KIP-500**: Kafka Removing ZK Dependency

Improve `scram-pra`

Better FTA inputs via Distributed Tracing
Thank you!

github.com/afalko/fta-kafka

Andrey Falko <afalko@lyft.com>

linkedin.com/in/in/andrey-falko/

Thank You