Improving Kafka Resilience - Gray Failures Mitigation

Michelle Valentinova, Senior Site Reliability Engineer
Kafka Platform Team
Apache Kafka - event streaming
97 clusters
2000 brokers
180M messages/s
Distributed systems

It works

It doesn’t work
Distributed systems

It kind of works
Kafka concepts
Topics are partitioned

- Parallel processing
- High availability
  - Failover between partitions
Topics, partitions, and brokers in Kafka
Kafka in the cloud architecture
Kafka broker’s network storage very slow
Kafka broker’s network storage very slow
Issues with a single broker affect all producers
Producers optimise for throughput (send buffer)
One broker is slow
Producers retry infinitely (configurable)

Don’t want to lose customer logs, metrics, miss alerts, etc.
Send buffer fills up
The issue is resolved
The issue is resolved continued
Issue with a single broker affects all producers
Impact for all producers:
25 minutes

Customer metrics, logs, alerts, etc. delayed for 25 minutes
Why do we even use network storage if it causes problems?
Our busiest Kafka clusters have 1.7PB of storage

We use a managed service and can’t choose local disk
Batching records using a partitioner
What happens when there is a failure
Adaptive random partitioning
Strictly uniform sticky partitioner

- Available in Apache Kafka 3.3.1 - 1 year ago
  - Our busiest clusters have 280 topics
  - Rolling out changes can take some time
- Partition probability to get records is inverse of queue size
- The partitioner is configured on the Kafka producers
Strictly uniform sticky partitioner

- Works well out of the box
- But can be configured to improve even further
Send buffer is shared
Partition availability timeout - disabled vs 5s

5s timeout

disabled
Partition availability configuration continued

- Value too low can result in too much flapping with usable brokers
  - Less throughput (Kafka specification is 10MB/s per partition)
- Value too high can result in still exhausting the send buffer
Recovery requires even more throughput
More throughput

• Analysed load
  ○ CPU had headroom - no need for more nodes
  ○ Disk write throughput could be improved
More throughput continued

- Originally we could only use EBS gp2 volumes
  - Limited to 250MB/s write throughput
- New EBS gp3 provisioned throughput costs 30-40x less compared to extra EC2 instances with EBS gp2 volumes
Does this solve all gray failures in Kafka?
High availability + durability with replication factor 3
Data path from producer to consumer
Fetch from closest replica / less cross-AZ data
Broker in AZ B has a problem
Broker in AZ B recovers but it’s still out of sync
Other AZs are NOT affected
Why not disable fetch from closest replica?
Data path from producer to consumer
Fetch from closest replica / less cross-AZ data
Peak daily ingest 85GB/s
Cross AZ traffic is only about $\frac{2}{3}$ of all.
Why we didn’t disable fetch from closest replica

- Finance would still be unhappy even with “just” 57GB/s
- The fix was already in the works
- Can be disabled either in client or server
  - But 95 clusters
  - And close to 300 topics in busiest clusters
What’s needed

- Kafka clients 3.3.2 / 3.4.0 - 8 months ago
- Kafka server 3.3.2 / 3.4.0
- Either helps, but full fix requires both
Alternative - if you can’t upgrade

● If you are OK with more cross-AZ traffic
  ○ disable fetch from closest replica
Does this solve all problems?
Producing with a partition key
Takeaways - producers with random partitioning

- Use Kafka clients 3.3.1 - 1 year ago
- Use the Strictly Uniform Sticky Partitioner
  - It’s the default - don’t override the partition class
  - Improves send buffer exhaustion on its own
  - Even better with `partitioner.timeout.availability.ms`
Takeaways continued

- Use Kafka clients 3.3.2 / 3.4.0 for consumers
  - 8 months ago
- Use Kafka server 3.3.2 / 3.4.0

Alternative:

- If you can’t upgrade but you are OK with more cross-AZ traffic - disable fetch from closest replica
Takeaways SRE

- Gray failures are hard to deal with
- Gamedays to reproduce gray failures
- Analyse system architecture
- Plan capacity according to your needs
Special thanks

- Alex Thengumpalli
- Anton Rodriguez
- Luke Kirby
- Christopher Wildman
- Alex Lindeman
- Kafka Platform Team
- Streaming SRE team
Thank you.

https://www.linkedin.com/in/michellevalentinova/