Your projector is working...
screenshare
Your projector is working...
TWO AND A HALF DATACENTER (FOR KAFKA)
SPECIAL THANKS

@jakekorab

@gAmUssA | #devoops | @confluentinc
WHAT ARE WE TRYING TO DO HERE?
The high-level view

SystemX

orders

SystemY

SystemZ
PARTITIONS AND CONSUMERS

Partition 0

Partition 1

Partition 2

SystemX

SystemY:0

SystemY:1

SystemZ
RELIABLE SENDS

1. send
2. store
3. ack
KAFKA, MEET STORAGE

OS Buffer Cache
Drive Controller Buffer
Drive Buffer

Operating System

Hardware
RELIABILITY THROUGH REPLICATION

1. send
2. write
3.a. fetch
3.b. fetch
4. ack

Producer
acks=all
MECHANICS OF SENDING

Producer

Network threads

Request queue

Response queue

IO threads

Purgatory

Mechanics of Sending

@gAmUssA   |   #devoops   |   @confluentinc
PARTITION DISTRIBUTION

Partition 0
Partition 1
Partition 2
Kafka Node Failure

Partition 0
Partition 1
Partition 2
ZOOKEEPER NODE FAILURE
**RUNTIME VIEW**

Diagram showing interactions between SystemX, SystemY, SystemZ, and ZooKeeper.

- SystemX
- SystemY
- SystemZ
- ZooKeeper

Arrows indicate the flow of information or actions between the systems.

@gAmUssA   |   #devoops   |   @confluentinc
Replication
THREE DATA CENTERS
3 DATA CENTRES

- DC1: ZooKeeper
- DC2: ZooKeeper
- DC3: ZooKeeper

@gAmUssA   |   #devoops   |   @confluentinc
RACK AWARE REPLICA ASSIGNMENT

DC1
1
Partition 0
L
2
Partition 1

DC2
3

DC3
4
Partition 2
L

broker.rack=DC1
broker.rack=DC2
broker.rack=DC3

@gAmUssA | #devoops | @confluentinc
NETWORK CONSIDERATIONS

- What are the costs associated with data transfer?
- What is your latency? ~30ms OK
- Shared infrastructure that could cause contention?
- Single point of failure?
KIP-392: ALLOW CONSUMERS TO FETCH FROM CLOSEST REPLICA

Client Config

rack.id=<location>

Broker Config

replica.selector.class=
<ReplicaSelector impl>

Out of the box:
• LeaderSelector (default)
• RackAwareReplicaSelector
KIP-392: ALLOW CONSUMERS TO FETCH FROM CLOSEST REPLICA

```java
class ClientMetadata {
    final String rackId;
    final String clientId;
    final InetAddress address;
    final KafkaPrincipal principal;
}

interface ReplicaSelector extends Configurable, Closeable {
    /**
     * Select the preferred replica a client should use for fetching.
     * If no replica is available, this method should return null.
     */
    Node select(ClientMetadata metadata, PartitionInfo partitionInfo);
}
```
replica.selector.class=RackAwareReplicaSelector

@gAmUssA   |   #devoops   |   @confluentinc

rack.id=DC2
rack.id=DC3
TWO DATA CENTERS
2 DATA CENTRES

Problems:

1. ZooKeeper
2. Topic Replication
2DC - NAIVE ZOOKEEPER SETUP

DC Failure Scenarios:

1. **DCmin** goes down - all OK
2. **DCmaj** goes down - ZK3 in quorum minority, shuts down. Outage.

Do not do this
2DC - HIERARCHICAL QUORUMS IN ZOOKEEPER

Brokers configured to talk to local ZKs.

Tolerates outage of one ZooKeeper per local cluster.

Trades off Availability for Consistency.
Communication outage looks just like a DC outage.

- Clients lose visibility of partition leaders in other DC
- Production either partially continues or blocks, depending on replication settings

Manual intervention required to resume processing
## 2DC - Replication Settings

<table>
<thead>
<tr>
<th>replication-factor</th>
<th>min.insync.replicas</th>
<th>enable.unclean.leader.election</th>
<th>Behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Consistency over Availability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Guarantees that all data is replicated to both DCs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Topics need to be reconfigured during outage to resume flow.</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>false</td>
<td>Availability over Consistency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data not guaranteed to be replicated to both DCs under some conditions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No topics reconfiguration needed during outage to resume flow.</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>

**GitHub**

Dabz/kafka-boom-boom

@gAmUssA | #devoops | @confluentinc
TWO AND A HALF DATA CENTERS
2.5 Data Centers

ZooKeeper behaviour same as 3DC setup.
Replication tradeoffs same as 2DC setup.
Mirroring
MULTI-DC VIA MIRRORING

Drivers

• Don’t have 3 Data Centers or inter-DC latency >30ms
• Can’t accept data loss
• Can’t accept stop the world

Characteristics

• Uses multiple clusters
• Asynchronous
• Typical uses are uni-directional
• Typically used for inter-region traffic
MIRRroring - General Setup
LOCATION-PREFIXED TOPICS

DC1

P

DC1_orders

C

DC2

P

DC1_orders

C

DC2_orders

DC2_orders

MM

MM
DEALING WITH DC FAILURE

Constraints:
- Consumer reconfiguration required
- Duplicate processing likely
CONSUMER OFFSETS

1. consumer.poll()
2. invoke
3. consumer.commitSync() or consumer.commitAsync()
Replicator features needed:
1. Consumer offset translation
2. Message provenance
SINGLE TOPIC + MIRRORED OFFSETS

Do not do this

@gAmUssA   |   #devoops   |   @confluentinc
What does it mean for consumer group X to be up to offset 1?
THE LIMITS OF OFFSETS

1. consumer.poll()
2. invoke
3. consumer.commitSync() / consumer.commitAsync()
IDEMPOTENT CONSUMPTION

orders

orders_processed

__consumer_offsets

2. extract id from event
if (id in store)
skip
else
write id into store
invoke
"WHAT IF WE DO THIS…"

Do not do this

orders

orders_processed

Replicator

orders

orders_processed

Replicator

@gAmUssA | #devoops | @confluentinc
MIRRORING PROCESSED STATE

DC1

orders

orders_processed

Consumer

order:X

processed:X

DC2

Replicator

orders

orders_processed

Consumer

Duplicate Processed

@gAmUssA | #devoops | @confluentinc
ONE LAST THING...
Multi-region Replication - Stretch Cluster Done Right

Following Fetching aka KIP-392
allows consumers to read from a replica other than the leader

Observers
Aka async replicas which are not part of ISR and can't be elected a leader

Replica Placement
JSON-based specification allows you to specify replica assignment as a set of matching constraints. For example, allows to keep the regular replicas in a single region and putting an observer in a different region

https://gamov.dev/mrc-demo
WHAT YOU NEED TO KNOW

• Stretched clusters are awesome; assuming <~30ms latency

• 3DC > 2.5DC > 2DC

• KIP-392 will make them even better
  • Confluent Server takes this further

• Mirroring is an alternative
  • Asynchronicity means it acts differently than a single cluster
  • Think about the impacts on operations and design of your code
CONFLUENT