Advanced Hybrid & Distributed Troubleshooting Case Studies for CloudOps Engineers

10 Complex Production Scenarios across Hybrid, Multi-region, and Distributed Systems

Case 1: Zookeeper Leader Election Flap Causing Service Instability

Problem: Distributed service experienced frequent master failovers causing write rejects.
Investigation:

```
$ zkServer.sh status
Mode: follower
$ tail -n 50 /var/log/zookeeper/zookeeper.log
WARN Server 2: Connection to 1 lost, starting new election
```

Root Cause: Network packet loss between quorum members caused repeated leader elections.
Resolution: Repaired network link, increased tickTime and election timeout, and promoted stable leader.
Prevention: Monitor inter-node network latency and tune election timeouts for unstable networks.

Case 2: Cassandra Read Anomalies due to Consistency Level Misuse

Problem: Clients observed stale reads intermittently across datacenters.
Investigation:

```
$ nodetool status
Datacenter: dc1 status=Up/Normal
$ grep consistency /app/config
consistency=ONE
```

Root Cause: Using CONSISTENCY=ONE in multi-DC writes led to eventual consistency showing stale data on some reads.
Resolution: Changed client consistency to QUORUM for reads/writes for critical paths and ran repair across nodes.
Prevention: Document and enforce appropriate consistency levels in multi-datacenter deployments.

Case 3: Kafka Consumer Lag after Broker Rebalance

Problem: Consumers lagging massively after broker maintenance.
Investigation:

```
$ kafka-consumer-groups --describe --group api-consumers --bootstrap-server
broker:9092
CURRENT-OFFSET LAG CONSUMER-ID
$ tail -n 100 /var/log/kafka/server.log
Rebalance in progress, group rebalance took 120s
```

Root Cause: Large rebalance caused consumers to be removed and reassign partitions leading to lag.
Resolution: Enabled cooperative rebalancing, increased session.timeout.ms, and rolled brokers with fewer partitions moved at once.
Prevention: Use incremental cooperative rebalancing and monitor consumer lag during maintenance.

Case 4: Split-brain in HA Cluster after Network Partition

Problem: Two masters active simultaneously after transient network partition, causing conflicting writes.
Investigation:

```
$ crm status
NodeA: Master, NodeB: Master
```

```
$ dmesg | grep cluster
Network heartbeat lost for 12s
```

Root Cause: Cluster quorum lost due to inadequate quorum policy allowing both sides to think they were authoritative.
Resolution: Manually resolved split-brain by fencing one partition and reconciling data; adjusted quorum policy to require majority.
Prevention: Implement proper fencing (STONITH), quorum policies, and monitor heartbeats across sites.

Case 5: Global DNS Propagation Delay Causing Failover Failures

Problem: Failover to secondary region succeeded but clients cached old DNS and kept hitting failed region.
Investigation:

```
$ dig +short app.example.com
Primary IP (cached)
$ grep TTL /etc/dns/config
ttl 86400
```

Root Cause: Very high DNS TTL caused clients to continue using old IP addresses after failover.
Resolution: Lowered DNS TTLs for failover endpoints and issued DNS flush; documented failover procedure.
Prevention: Use low TTLs for failover records and plan for cache behavior in runbooks.

Case 6: iSCSI Multipath Path Flapping in Hybrid Environment

Problem: Storage I/O intermittently failed on on-premise LUN presented to cloud gateways.
Investigation:

```
$ multipath -ll
mpatha (3600a0b8...): fail_if_no_path no
$ dmesg | grep multipath
path to 10.0.0.5 lost
```

Root Cause: Asymmetric routing and ACL changes caused certain iSCSI paths to flap.
Resolution: Fixed network routing, updated multipath.conf to prefer stable paths, and restarted multipathd.
Prevention: Validate network paths end-to-end and test multipath failover scenarios regularly.

Case 7: Consul Gossip Partition Preventing Service Discovery

Problem: Services could not discover each other after a WAN link saturated intermittently.
Investigation:

```
$ consul members
left=3 joined=2
$ tail -n 50 /var/log/consul.log
gossip: dropped messages, slow transport
```

Root Cause: Gossip protocol messages dropped due to network saturation causing member isolation.
Resolution: Increased WAN bandwidth, tuned gossip interval and enabled WAN federation with ACLs.
Prevention: Monitor gossip health, use dedicated control plane connectivity for critical services.

Case 8: Multi-region Time Skew Causing TLS and Token Failures

Problem: Authentication tokens and TLS cert validations failing in multi-region setup.
Investigation:

```
$ date -u; ssh -v user@host
Wed Oct 01 12:05:00 UTC 2025
certificate verify failed: certificate has expired
$ chronyc tracking
System clock offset: 300s
```

Root Cause: Regional NTP servers were unreachable, causing clocks to drift beyond acceptable skew.
Resolution: Restored NTP connectivity, forced time sync, and reissued short-lived tokens.
Prevention: Use redundant time sources and monitor clock skew across regions; use chrony with multiple peers.

Case 9: Load Balancer Sticky Sessions Causing Unequal Load Distribution

Problem: Certain app instances overloaded while others idle due to sticky session configuration.
Investigation:

```
$ aws elb describe-load-balancers --load-balancer-names app-lb
StickinessPolicy: app-sticky
$ top on overloaded node
High load and GC pauses on some app pods
```

Root Cause: Sticky session policy caused clients to stay on unhealthy or overloaded instances.
Resolution: Disabled session stickiness for stateless services and enabled session affinity via centralized store when needed.
Prevention: Prefer stateless services or use distributed session stores instead of LB stickiness for scale.

Case 10: Message Queue Poison Messages Blocking Processing

Problem: Message processing halted due to a poison message repeatedly failing and blocking the queue.
Investigation:

```
$ rabbitmqctl list_queues name messages_ready messages_unacknowledged
api-queue 0 1
$ tail -n 50 /var/log/app/worker.log
Error: JSON decode error at byte 0
```

Root Cause: Malformed message caused consumer to repeatedly fail and not ack, blocking the queue.
Resolution: Moved the poison message to a dead-letter queue, patched producer validation, and restarted consumers.
Prevention: Implement DLQ patterns, message validation at producer, and consumer retries with backoff.