Kafka Interview - Day 3 (April 30) - Professional Notes & Feedback

---

Topics Covered

Main Focus: Kafka Partition Assignment Strategies, Rebalancing Impact, and Configuration Tuning

---

Questions and Professional Interview-Style Answers

Q1. How does RangeAssignor work and when can it cause imbalance?

- It assigns partitions in contiguous ranges.
- If partition count is not a multiple of consumer count or topic subscription varies, imbalance occurs.
- Example: C1 gets 0-2, C2 gets 3-5, but if there are 7 partitions, C1 might get 4 and C2 only 3.

Q2. What is RoundRobinAssignor and when is it preferred?

- It assigns partitions in round-robin manner, rotating across consumers.
- Works best when all consumers subscribe to same topics.
- Example: C1 gets P0, P2; C2 gets P1, P3.

Q3. What is StickyAssignor?

- Attempts to retain previous assignments across rebalances.
- Tries to balance partitions evenly while minimizing movement.

- Good for reducing disruption in production.

Q4. What is CooperativeStickyAssignor and how is it different?

- Introduced incremental rebalancing (Kafka 2.4+).
- Consumers revoke only required partitions, not full stop-the-world rebalance.
- Great for large-scale systems to reduce lag spikes.

Q5. How do you monitor and detect partition skew?

- Use lag per partition metrics.
- Monitor throughput difference across consumers.
- Look for uneven data volumes in topic (e.g., skewed keys).

Q6. What are signs of assignment strategy failure?

- Lag on specific partitions only.
- One consumer heavily loaded, others idle.
- Frequent rebalances and instability.

Q7. How to tune configs to reduce rebalancing impact?

- Increase `session.timeout.ms` to avoid false rebalances.
- Tune `max.poll.interval.ms` to align with processing time.
- Use `CooperativeStickyAssignor` to reduce reassignment volume.

Q8. What production issues arise from bad partition assignment?

- Lag buildup, slow processing, delayed SLAs.
- CPU/memory spikes on overloaded consumers.
- Rebalance loops (rebalance → slow consumer → rebalance again).

Q9. What are In-Sync Replicas (ISR) and why do they matter?

- Set of replicas that are fully caught up with leader.
- If ISR falls below `min.insync.replicas`, Kafka will reject produce requests.
- Key for maintaining durability.

Q10. How to achieve exactly-once semantics in Kafka?

- Use idempotent producer.
- Use transactional APIs to produce and commit offsets atomically.
- Consumer must write atomically to target (e.g., DB) and commit only after success.
