Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 1: Data Modeling (Q1–Q10)

Q1. How would you design partitions for high write throughput in time-series data?

Answer: Cassandra partitions rows based on the partition key. For time-series, avoid using raw timestamps as partition keys (causes hotspots). Instead, use composite keys with time bucketing.

CQL: CREATE TABLE sensor_data (device_id TEXT, day DATE, ts TIMESTAMP, reading DOUBLE, PRIMARY KEY ((device_id, day), ts)) WITH CLUSTERING ORDER BY (ts DESC);

Best practices: Keep partitions 10–200MB, use descending clustering order, monitor via system.size_estimates.

Q2. How do you handle wide partitions in Cassandra?

Answer: Wide partitions cause high latency and GC pressure. Split large entities with bucketing (e.g., user_id + month).

CQL: CREATE TABLE user_activity (user_id TEXT, bucket_month TEXT, ts TIMESTAMP, activity TEXT, PRIMARY KEY ((user_id, bucket_month), ts));

Best practices: Keep partitions <200MB, use nodetool tablestats to detect.

Q3. How do you model user activity feeds efficiently?

Answer: Partition by user_id, cluster by timestamp. Precompute timelines at write time (fan-out).

CQL: CREATE TABLE user_feed (user_id TEXT, ts TIMESTAMP, post_id UUID, content TEXT, PRIMARY KEY ((user_id), ts)) WITH CLUSTERING ORDER BY (ts DESC);

Best practices: Denormalize for each query pattern, avoid cross-partition queries.

Q4. When should you use clustering columns vs static columns?

Answer: - Clustering columns → order rows inside partition. - Static columns → shared value for all rows in a partition.

CQL: CREATE TABLE orders (customer_id TEXT, order_id UUID, order_date TIMESTAMP, customer_name TEXT STATIC, PRIMARY KEY ((customer_id), order_id));

Best practices: Use static columns for common attributes, clustering for ordering.

Q5. What are the trade-offs of using collections (list, set, map)?

Answer: Collections are rewritten fully on updates → expensive for large sets.

CQL: CREATE TABLE user_profile (user_id TEXT PRIMARY KEY, emails SET, preferences MAP);

Best practices: Use only for small metadata. Model unbounded collections as tables.

Q6. How do you model hierarchical data (categories, subcategories)?

Answer: Flatten hierarchies using composite keys.

CQL: CREATE TABLE catalog (category TEXT, subcategory TEXT, item_id UUID, item_name TEXT, PRIMARY KEY ((category, subcategory), item_id));

Best practices: Denormalize aggressively, avoid recursive lookups.

Q7. How do you design schema for multi-tenant applications?

Answer: Include tenant_id in partition key. Add bucketing for large tenants.

CQL: CREATE TABLE tenant_orders (tenant_id TEXT, order_id UUID, customer_id TEXT, order_date TIMESTAMP, PRIMARY KEY ((tenant_id, order_id)));

Best practices: Monitor tenant skew, consider quotas/sharding.

Q8. How do you choose partition keys to avoid hotspots?

Answer: Avoid sequential IDs/timestamps. Use UUIDs + time bucketing.

CQL: CREATE TABLE metrics (metric_id UUID, bucket_day DATE, ts TIMESTAMP, value DOUBLE, PRIMARY KEY ((metric_id, bucket_day), ts));

Best practices: Monitor with nodetool status, ensure token-aware drivers.

Q9. What are denormalization strategies in Cassandra?

Answer: Design multiple tables per query. Explicit denormalization > materialized views.

CQL: CREATE TABLE orders_by_customer (...); CREATE TABLE orders_by_date (...);

Best practices: Use idempotent dual writes, avoid over-normalization.

Q10. How do you handle evolving schemas in Cassandra?

Answer: Prefer creating new tables + dual writes instead of frequent ALTER TABLE.

CQL: ALTER TABLE orders ADD payment_type TEXT;

Best practices: Use ALTER only for small additive changes, otherwise migrate.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 1: Data Modeling (Q11–Q20)

Q11. How do you avoid unbounded partition growth in Cassandra?

Answer: Unbounded partitions hurt performance. Use bucketing strategies based on time, geography, or logical grouping.

CQL: CREATE TABLE logs (app_id TEXT, bucket_day DATE, ts TIMESTAMP, message TEXT, PRIMARY KEY ((app_id, bucket_day), ts));

Best practices: Monitor partition size (system.size_estimates), use time windows.

Q12. What is the role of clustering order in query performance?

Answer: Clustering order defines row ordering within a partition. Correct ordering avoids in-memory sorting during queries.

CQL: PRIMARY KEY ((user_id), ts) WITH CLUSTERING ORDER BY (ts DESC);

Best practices: Use DESC for recent data queries, ASC for range scans.

Q13. How do you design for pagination in Cassandra?

Answer: Cassandra provides server-side paging. Use FETCH SIZE in drivers.

CQL: SELECT * FROM user_feed WHERE user_id='u1';

Best practices: Avoid LIMIT loops; use driver paging.

Q14. How do you enforce uniqueness in Cassandra?

Answer: Use Lightweight Transactions (LWT).

CQL: INSERT INTO users (id, email) VALUES ('u1', 'x@test.com') IF NOT EXISTS;

Best practices: Use LWT sparingly (4× latency), only for critical uniqueness constraints.

Q15. How do you handle large objects (BLOBs) in Cassandra?

Answer: Cassandra supports BLOB, but not suited for very large files. Store metadata in Cassandra, file in object storage.

Best practices: Keep <10MB per value, offload larger files.

Q16. How do you model many-to-many relationships?

Answer: Use two denormalized tables.

CQL: CREATE TABLE student_courses (student_id TEXT, course_id TEXT, PRIMARY KEY ((student_id), course_id)); CREATE TABLE course_students (course_id TEXT, student_id TEXT, PRIMARY KEY ((course_id), student_id));

Best practices: Duplicate data, keep writes idempotent.

Q17. How do you handle counters in Cassandra?

Answer: Counters support atomic increments but are tricky.

CQL: CREATE TABLE page_views (page_id TEXT PRIMARY KEY, views COUNTER); UPDATE page_views SET views = views + 1 WHERE page_id='p1';

Best practices: Use counters only when necessary, avoid in multi-DC with low latency SLA.

Q18. How do you design for query flexibility in Cassandra?

Answer: Model for known queries only. Cassandra doesn’t support ad-hoc queries.

Best practices: Create multiple tables per query, avoid ALLOW FILTERING.

Q19. What is the impact of TTL in schema design?

Answer: TTL creates tombstones. Large TTL use can overload compaction and reads.

CQL: INSERT INTO sessions (id, data) VALUES ('s1','abc') USING TTL 3600;

Best practices: Use TTL only for expiring data (caches, sessions). Use TWCS for TTL-heavy tables.

Q20. How do you design Cassandra schema for IoT workloads?

Answer: Partition by device ID + time bucket, cluster by timestamp.

CQL: CREATE TABLE iot_data (device_id TEXT, day DATE, ts TIMESTAMP, reading DOUBLE, PRIMARY KEY ((device_id, day), ts));

Best practices: Bucketing for partition limits, clustering order DESC for recent data queries.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 2: Replication & Consistency (Q21–Q30)

Q21. What is replication factor and its impact on durability?

Answer: Replication factor (RF) defines how many copies of each partition are stored.

Example: RF=3 → three replicas across nodes.

Impact: - Higher RF improves durability & availability. - More replicas increase write latency & storage cost.

Best practices: Use RF=3 for production; use NetworkTopologyStrategy in multi-DC.

Q22. Difference between SimpleStrategy and NetworkTopologyStrategy?

Answer: - SimpleStrategy: Places replicas in a single DC, sequentially on ring. - NetworkTopologyStrategy: Places replicas per data center, rack-aware.

Config: CREATE KEYSPACE ks WITH replication = {'class':'NetworkTopologyStrategy','dc1':3,'dc2':3};

Best practices: Use NetworkTopologyStrategy for production.

Q23. Trade-offs between consistency levels (ONE, QUORUM, ALL)?

Answer: - ONE: Fast, but risk of stale reads. - QUORUM: Balance between consistency & availability (RF/2+1). - ALL: Strong consistency, high latency.

Best practices: Use LOCAL_QUORUM for multi-DC apps.

Q24. How does read repair work?

Answer: Read repair ensures replicas converge during reads.

Types: - Blocking (coordinated with query). - Async background.

Config (cassandra.yaml): read_repair_chance: 0.1

Best practices: Keep chance low (0.1–0.2).

Q25. What is hinted handoff and when to disable it?

Answer: Hinted handoff temporarily stores missed writes when node is down.

Config (cassandra.yaml): hinted_handoff_enabled: true

Disable when: - Node is down long term. - To avoid overload after recovery.

Best practices: Enable for short outages (<3 hours).

Q26. What is speculative retry?

Answer: Speculative retry sends extra read requests if replicas are slow.

Config: speculative_retry: 99PERCENTILE

Best practices: Monitor latency before enabling; prevents tail latency issues.

Q27. How does Cassandra handle consistency in multi-DC?

Answer: Use LOCAL_QUORUM for low-latency DC-local consistency. EACH_QUORUM ensures consistency across DCs but higher latency.

Best practices: Prefer LOCAL_QUORUM in geo-distributed apps.

Q28. Difference between strong consistency and eventual consistency in Cassandra?

Answer: - Strong: Read after write always reflects latest value (requires ALL). - Eventual: Reads may return stale until replicas converge.

Cassandra defaults to eventual consistency; can tune per query with CL.

Q29. When should you use EACH_QUORUM?

Answer: EACH_QUORUM waits for quorum of replicas in each DC.

Use case: Financial applications requiring strict global consistency.

Trade-off: High latency, less availability.

Q30. How does consistency work with Lightweight Transactions (LWT)?

Answer: LWTs use Paxos to achieve linearizable consistency.

Example: INSERT INTO users (id,email) VALUES('u1','x@test.com') IF NOT EXISTS;

Best practices: Use only for critical uniqueness constraints due to 4x latency overhead.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 2: Replication & Consistency (Q31–Q40)

Q31. How does Cassandra ensure replica synchronization?

Answer: - Hinted handoff: temporary writes to be replayed. - Read repair: synchronizes during reads. - Anti-entropy repair: compares Merkle trees.

Best practices: Run nodetool repair regularly to ensure consistency.

Q32. What is anti-entropy repair?

Answer: Anti-entropy repair uses Merkle trees to compare SSTables between replicas.

Command: $ nodetool repair

Best practices: - Run full repairs weekly or incremental repairs daily. - Use subrange repairs for large clusters.

Q33. Difference between incremental and full repair?

Answer: - Full repair: Compares all data. - Incremental repair: Repairs only SSTables not already repaired.

Best practices: Use incremental for efficiency; full repairs occasionally to clean up.

Q34. How does Cassandra handle write consistency with RF=3 and CL=QUORUM?

Answer: QUORUM = RF/2 + 1 $\rightarrow$ 2 replicas must acknowledge.

Impact: - Ensures strong consistency with RF=3. - Tolerates 1 replica failure.

Best practices: Commonly used balance of consistency vs availability.

Q35. What is the difference between LOCAL_QUORUM and QUORUM?

Answer: - QUORUM: Across all DCs. - LOCAL_QUORUM: Only replicas in local DC.

Use LOCAL_QUORUM for low-latency regional apps.

Best practices: Use LOCAL_QUORUM in multi-DC deployments.

Q36. What happens if a read is sent with CL=ALL and one replica is down?

Answer: Read fails since ALL replicas must respond.

Best practices: Avoid ALL in production due to fragility.

Q37. How does consistency level affect latency?

Answer: - ONE: Low latency, low consistency. - QUORUM: Moderate latency, balanced consistency. - ALL: High latency, strong consistency.

Best practices: Choose based on SLA (e.g., LOCAL_QUORUM for user-facing apps).

Q38. How does Cassandra handle conflicting writes?

Answer: Last-write-wins (based on timestamp).

Limitations: - Clock skew can cause anomalies. - Use client clock sync (NTP).

Best practices: Keep timestamps synchronized across clients.

Q39. How does speculative retry impact consistency guarantees?

Answer: Speculative retry does not weaken consistency. It only sends additional requests if replica is slow.

Best practices: Configure properly to avoid overloading nodes.

Q40. When should you use consistency level SERIAL or LOCAL_SERIAL?

Answer: - SERIAL: Ensures linearizability in LWT across DCs. - LOCAL_SERIAL: Restricts to local DC.

Best practices: Use LOCAL_SERIAL to reduce latency in multi-DC LWTs.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 3: Compaction & Repairs (Q41–Q50)

Q41. What are the main compaction strategies in Cassandra?

Answer: - SizeTieredCompactionStrategy (STCS): Groups similar-sized SSTables, default for write-heavy workloads. - LeveledCompactionStrategy (LCS): Organizes into levels, better for read-heavy workloads. - TimeWindowCompactionStrategy (TWCS): Optimized for TTL/time-series.

Best practices: Use STCS for write-heavy, LCS for read-heavy, TWCS for TTL-based time-series data.

Q42. How does SizeTieredCompactionStrategy (STCS) work?

Answer: STCS merges SSTables of similar sizes into larger ones.

Pros: Good for bulk writes. Cons: High read amplification.

Best practices: Default strategy unless workload favors LCS or TWCS.

Q43. How does LeveledCompactionStrategy (LCS) work?

Answer: LCS maintains SSTables in levels of fixed size (10x per level). Ensures only 1 SSTable checked per level during reads.

Pros: Low read amplification. Cons: Higher write amplification.

Best practices: Use for read-heavy workloads.

Q44. How does TimeWindowCompactionStrategy (TWCS) work?

Answer: TWCS groups SSTables by fixed time windows, ideal for TTL data.

Config: compaction = {'class': 'TimeWindowCompactionStrategy','compaction_window_unit':'DAYS','compaction_window_size':'1'}

Best practices: Use for IoT/time-series with TTLs.

Q45. How do tombstones affect read performance?

Answer: Tombstones are markers for deleted/expired data. Reads must scan through them.

Impact: High tombstones → slow reads, timeouts.

Best practices: Minimize TTLs, avoid frequent deletes, monitor tombstone warnings in logs.

Q46. How do you monitor compaction activity?

Answer: Command: $ nodetool compactionstats

Shows pending compactions, completed tasks.

Best practices: Alert if pending compactions remain high.

Q47. What is incremental repair?

Answer: Repairs only SSTables not already repaired.

Command: $ nodetool repair --inc

Best practices: Use incremental repair frequently, full repairs periodically.

Q48. How does Cassandra handle anti-entropy repair?

Answer: Uses Merkle trees to compare data across replicas.

Command: $ nodetool repair

Best practices: Run repairs regularly to prevent zombie rows.

Q49. What are zombie rows in Cassandra?

Answer: Rows reappearing after deletion due to missed repair or tombstone GC.

Causes: Incomplete repairs, low gc_grace_seconds.

Best practices: Ensure consistent repairs, keep gc_grace_seconds >= repair interval.

Q50. How does gc_grace_seconds impact repairs and tombstones?

Answer: Defines how long tombstones are kept before garbage collection.

Default: 10 days.

Impact: Too low → risk of resurrected data; too high → more tombstones.

Best practices: Set gc_grace_seconds >= repair interval.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 3: Compaction & Repairs (Q51–Q60)

Q51. How does nodetool cleanup work?

Answer: Cleanup removes SSTables that contain data no longer belonging to the node after topology changes.

Command: $ nodetool cleanup keyspace_name

Best practices: Run after decommission or topology changes.

Q52. What is major compaction and when should you use it?

Answer: Major compaction merges all SSTables in a table into one.

Command: $ nodetool compact

Impact: High I/O, space usage.

Best practices: Avoid in production except for specific maintenance (e.g., TTL cleanup).

Q53. How do you detect tombstone issues in a table?

Answer: Command: $ nodetool tablestats keyspace.table

Look for: - Tombstone Scanned Histogram (high values indicate issues).

Best practices: Alert on high tombstone counts, optimize TTL usage.

Q54. How does Cassandra handle expired data with TTL?

Answer: TTL creates tombstones; expired data is purged after gc_grace_seconds.

Best practices: Use TWCS for TTL-heavy workloads, monitor compaction for tombstone cleanup.

Q55. How do incremental repairs improve efficiency?

Answer: They repair only unrepaired SSTables.

Command: $ nodetool repair --inc

Best practices: Run incremental repairs daily, schedule full repair occasionally.

Q56. What is the impact of disabling auto-compaction?

Answer: Auto-compaction merges SSTables automatically. Disabling leads to unbounded SSTable growth and poor performance.

Best practices: Do not disable auto-compaction except for maintenance.

Q57. How can you monitor repair progress?

Answer: Command: $ nodetool repair -pr

Logs show progress, metrics available via JMX.

Best practices: Use Cassandra Reaper for automated repair management.

Q58. What is the effect of very high gc_grace_seconds?

Answer: Prolongs tombstone lifetime, increasing read overhead.

Best practices: Balance value with repair schedule; default 10 days is safe.

Q59. What is the difference between nodetool scrub and compact?

Answer: - scrub: Rebuilds SSTables, fixes corruption. - compact: Merges SSTables.

Best practices: Use scrub only for corruption issues.

Q60. How do repairs prevent zombie rows?

Answer: Zombie rows = resurrected deleted data due to missed tombstones.

Repairs ensure all replicas see tombstones before gc_grace_seconds expires.

Best practices: Run regular repairs aligned with gc_grace_seconds.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 4: Cluster Management (Q61–Q70)

Q61. What happens during Cassandra node bootstrap?

Answer: During bootstrap, a new node joins the cluster and streams data for its token ranges from existing nodes.

Steps: 1. Node contacts seeds and joins gossip. 2. Token assignment is chosen (vnodes/manual). 3. Data streaming begins from replicas. 4. Node transitions to UN (Up/Normal).

Command: $ nodetool status

Best practices: Enable auto_bootstrap, monitor with nodetool netstats.

Q62. How does the gossip protocol work in Cassandra?

Answer: Gossip is a peer-to-peer protocol for membership and state dissemination.

- Exchanges state every second with 1–3 peers. - State includes heartbeat, schema, tokens, load.

Best practices: Consistent snitch config, monitor gossip logs for issues.

Q63. How do you safely decommission a node?

Answer: Decommission streams data to remaining replicas and removes the node.

Command: $ nodetool decommission

Best practices: Run repair before decommission, avoid simultaneous DC node removals.

Q64. What is the process to replace a dead node?

Answer: Steps: 1. Start new node with -Dcassandra.replace_address= 2. Node streams data from replicas. 3. New node assumes ownership.

Best practices: Ensure old node is fully down, avoid multiple replacements at once.

Q65. What is the role of seed nodes?

Answer: Seeds help new nodes discover cluster on bootstrap. Gossip maintains membership afterward.

Config (cassandra.yaml): seed_provider: - class_name: org.apache.cassandra.locator.SimpleSeedProvider parameters: - seeds: "10.0.0.1,10.0.0.2"

Best practices: Configure multiple seeds per DC.

Q66. How do you handle scaling out a Cassandra cluster?

Answer: Add nodes → bootstrap → data rebalances.

Steps: 1. Install Cassandra. 2. Configure cassandra.yaml. 3. Start node, it bootstraps.

Best practices: Add in pairs, rebalance before disks exceed 70%.

Q67. What does nodetool cleanup do?

Answer: Removes data not owned by the node after topology change.

Command: $ nodetool cleanup keyspace

Best practices: Run after decommission, not on new nodes.

Q68. How do you drain a node before maintenance?

Answer: Flushes memtables to SSTables and stops accepting writes.

Command: $ nodetool drain

Best practices: Always drain before shutdown for maintenance.

Q69. How can you monitor node health?

Answer: Use nodetool and JMX metrics.

Command: $ nodetool status

Best practices: Integrate with Prometheus/Grafana, alert on state changes.

Q70. How do vnodes simplify cluster management?

Answer: Vnodes assign multiple token ranges per node (default 256).

Benefits: Easier scaling, balanced data distribution.

Config: num_tokens: 256

Best practices: Reduce num_tokens (16–32) for large clusters to improve repair speed.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 4: Cluster Management (Q71–Q80)

Q71. How do you handle node failures in Cassandra?

Answer: Cassandra tolerates node failures using replication and hinted handoff.

Best practices: Replace dead nodes quickly, monitor hinted handoff load, run repairs after recovery.

Q72. What is the difference between replace_address and removenode?

Answer: - replace_address: Used to replace a dead node with a new one. - removenode: Used to remove a dead node from the cluster metadata.

Commands: $ nodetool removenode $ cassandra -Dcassandra.replace_address=

Best practices: Use replace_address when you want new node to take ownership.

Q73. How do you configure snitches in Cassandra?

Answer: Snitches determine data placement across racks/DCs.

Common: - GossipingPropertyFileSnitch (default). - EC2Snitch, RackInferringSnitch.

Config (cassandra-rackdc.properties): dc=DC1 rack=RAC1

Best practices: Use GossipingPropertyFileSnitch in modern deployments.

Q74. How does Cassandra handle adding a new data center?

Answer: Steps: 1. Configure snitches with new DC info. 2. Add nodes to new DC with proper settings. 3. Run nodetool rebuild to stream data.

Best practices: Add DC gradually, verify replication settings.

Q75. How do you verify cluster topology?

Answer: Command: $ nodetool status

Shows nodes, DCs, racks, token distribution.

Best practices: Verify after topology changes, ensure balance.

Q76. How do you handle rolling upgrades in Cassandra?

Answer: Steps: 1. Drain node → stop Cassandra → upgrade binaries. 2. Restart node, verify with nodetool status. 3. Repeat node by node.

Best practices: Do not upgrade all nodes simultaneously, check logs for schema version agreement.

Q77. How does Cassandra handle network partitions?

Answer: Partition tolerance is prioritized (AP system).

- Writes may succeed partially depending on CL. - Hinted handoff helps when partition heals.

Best practices: Use LOCAL_QUORUM in multi-DC for resilience.

Q78. What is the role of system.peers and system.local tables?

Answer: - system.local: Stores info about current node. - system.peers: Stores info about other nodes.

Best practices: Query for cluster metadata verification.

Q79. How do you scale Cassandra horizontally?

Answer: Scale out by adding nodes → automatic data rebalance.

Best practices: Monitor load, rebalance before hotspots occur, use vnodes for easy scaling.

Q80. How do you back pressure or throttle streaming traffic?

Answer: Streaming throttle controls bootstrap/repair streaming bandwidth.

Config (cassandra.yaml): stream_throughput_outbound_megabits_per_sec: 200

Best practices: Tune based on network capacity to avoid congestion.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 5: Performance Tuning (Q81–Q90)

Q81. How do you tune JVM heap size in Cassandra?

Answer: Heap size directly affects GC performance. Recommended max heap: 8GB.

Config (jvm.options): -Xms8G -Xmx8G

Best practices: Keep heap <=50% RAM, avoid >16GB, monitor GC logs.

Q82. What is the impact of row cache and key cache?

Answer: - Key cache: caches partition key locations in SSTables (low memory usage, effective). - Row cache: caches full rows, rarely used.

Config (cassandra.yaml): key_cache_size_in_mb: 100 row_cache_size_in_mb: 0

Best practices: Enable key cache, avoid row cache except for static data.

Q83. How do you tune memtables?

Answer: Memtables buffer writes before flush.

Config: memtable_flush_writers: auto

Best practices: Monitor flush queues via nodetool tpstats, adjust flush writers per disk count.

Q84. How do you analyze read latency issues?

Answer: Check for large partitions, tombstones, compaction backlog.

Commands: $ nodetool tablestats $ nodetool compactionstats

Best practices: Optimize schema, avoid ALLOW FILTERING, monitor GC and I/O.

Q85. How do you analyze write latency issues?

Answer: Causes: slow disk flush, overloaded commit log, memtable stalls.

Commands: $ nodetool tpstats $ iostat

Best practices: Use SSDs, separate commit log/data disks, adjust flush writers.

Q86. How do you tune compaction throughput?

Answer: Config (cassandra.yaml): compaction_throughput_mb_per_sec: 64

Best practices: Increase if backlog grows, balance compaction vs query latency.

Q87. How do concurrent_reads and concurrent_writes affect performance?

Answer: Define thread pools for read/write requests.

Config: concurrent_reads: 32 concurrent_writes: 32

Best practices: Set ~8x number of cores, monitor tpstats queues.

Q88. How do you optimize batch operations?

Answer: Batches ensure atomicity, not performance. Large batches increase load.

Best practices: Keep <50 statements per batch, use unlogged batches for performance.

Q89. How do you optimize hinted handoff?

Answer: Hinted handoff replays missed writes when node recovers.

Config: hinted_handoff_enabled: true max_hint_window_in_ms: 10800000

Best practices: Enable for short outages, disable for long outages.

Q90. How do you reduce tombstone impact on reads?

Answer: - Avoid excessive TTLs and deletes. - Monitor logs for tombstone warnings.

Best practices: Use TWCS for TTL-heavy workloads, tune gc_grace_seconds, avoid queries scanning many tombstones.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 5: Performance Tuning (Q91–Q100)

Q91. What are signs of GC pressure in Cassandra?

Answer: - Long GC pauses in gc.log. - High heap usage. - Increased read/write latency.

Best practices: Use G1GC, keep heap <=8GB, monitor logs continuously.

Q92. How do you tune commit log settings?

Answer: Commit log durability impacts writes.

Config (cassandra.yaml): commitlog_segment_size_in_mb: 32 commitlog_sync: periodic commitlog_sync_period_in_ms: 10000

Best practices: Place commit log on dedicated SSD/NVMe, monitor disk latency.

Q93. How do you optimize read path performance?

Answer: - Use token-aware drivers. - Limit partition size. - Tune key_cache.

Best practices: Avoid ALLOW FILTERING, use driver paging, optimize schema for queries.

Q94. How do you optimize write path performance?

Answer: - Avoid hotspots (design partition keys well). - Use fast disks for commit log. - Tune memtables and flush writers.

Best practices: Use LOCAL_QUORUM for balanced latency and safety.

Q95. How do you tune compaction strategy for mixed workloads?

Answer: - STCS: good for write-heavy. - LCS: good for read-heavy. - TWCS: good for TTL-based.

Best practices: Choose per table; avoid global defaults.

Q96. How do you detect I/O bottlenecks?

Answer: Tools: $ iostat $ nodetool tpstats

Indicators: high disk await, long write latency.

Best practices: Use SSDs, separate commit log/data disks, monitor file descriptors.

Q97. How do you reduce latency caused by compaction backlog?

Answer: - Increase compaction_throughput_mb_per_sec. - Add nodes if capacity low. - Run repairs to reduce backlog.

Best practices: Ensure disk has 50% free space for compaction.

Q98. How do you optimize Cassandra for SSDs?

Answer: - Reduce commitlog_segment_size_in_mb for low-latency devices. - Monitor SSD wear leveling.

Best practices: Keep high file descriptor limits, monitor latency via iostat.

Q99. How do you monitor performance metrics?

Answer: - nodetool tablestats, tpstats, compactionstats. - JMX/Prometheus + Grafana dashboards.

Best practices: Alert on latency, dropped mutations, compaction backlog.

Q100. What are best practices for benchmarking Cassandra?

Answer: - Use cassandra-stress for load testing. - Simulate production replication/consistency levels. - Run long enough to trigger compactions and GC.

Best practices: Capture system + Cassandra metrics, avoid microbenchmarks.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 6: Security & Backup (Q101–Q110)

Q101. How do you enable authentication in Cassandra?

Answer: Enable PasswordAuthenticator in cassandra.yaml.

Config: authenticator: PasswordAuthenticator

Command: $ cqlsh -u cassandra -p cassandra

Best practices: Change default password, use role-based access.

Q102. How do you configure authorization in Cassandra?

Answer: Enable CassandraAuthorizer for role-based access control.

Config (cassandra.yaml): authorizer: CassandraAuthorizer

CQL: GRANT SELECT ON keyspace.table TO role1;

Best practices: Grant least privilege required.

Q103. How do you enable client-to-node SSL encryption?

Answer: Enable client_encryption_options in cassandra.yaml.

Config: client_encryption_options: enabled: true keystore: conf/.keystore truststore: conf/.truststore

Best practices: Use TLS 1.2+, rotate certificates regularly.

Q104. How do you enable node-to-node encryption?

Answer: Enable internode_encryption in cassandra.yaml.

Config: server_encryption_options: internode_encryption: all

Best practices: Encrypt both internode and client connections for compliance.

Q105. How are passwords stored in Cassandra?

Answer: Passwords are hashed using salted SHA-256.

Table: system_auth.roles

Best practices: Use external auth (LDAP/Kerberos) for enterprises.

Q106. How do you audit queries in Cassandra?

Answer: Enable audit logging via AuditLog.

Config (cassandra.yaml): audit_logging_options: enabled: true logger: BinAuditLogger

Best practices: Send logs to centralized SIEM.

Q107. How do you perform a full backup in Cassandra?

Answer: Take a snapshot using nodetool.

Command: $ nodetool snapshot keyspace_name

Creates hard links to SSTables under snapshots/ directory.

Best practices: Copy snapshots to external storage.

Q108. How do incremental backups work?

Answer: When enabled, Cassandra copies flushed SSTables to backups directory.

Config (cassandra.yaml): incremental_backups: true

Best practices: Combine incremental with periodic full snapshots.

Q109. How do you restore from a snapshot?

Answer: 1. Stop Cassandra. 2. Clear data directory (except system keyspaces). 3. Copy snapshot SSTables back to data dir. 4. Start Cassandra.

Best practices: Run nodetool repair after restore.

Q110. What are best practices for Cassandra backups?

Answer: - Use snapshots + incremental backups. - Store offsite or in cloud. - Monitor disk usage of snapshots. - Automate cleanup with sstableloader when restoring.

Best practices: Test restores periodically.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 6: Security & Backup (Q111–Q120)

Q111. How do you plan a disaster recovery strategy in Cassandra?

Answer: - Use multi-DC deployment with NetworkTopologyStrategy. - Enable incremental + full backups. - Document restore runbooks.

Best practices: Test DR plan quarterly.

Q112. How do you restore a single table from backup?

Answer: 1. Stop Cassandra. 2. Restore table SSTables from snapshot. 3. Use sstableloader to stream into cluster.

Command: $ sstableloader -d /path/to/sstables

Best practices: Run nodetool repair after restore.

Q113. How do you validate backup integrity?

Answer: - Verify file checksums after backup copy. - Perform test restores in staging cluster.

Best practices: Automate checksum verification.

Q114. What is the role of sstableloader in backup restore?

Answer: sstableloader streams SSTables into a live cluster.

Command: $ sstableloader -d snapshot_dir

Best practices: Use for restoring into new cluster or table.

Q115. How do you back up system keyspaces?

Answer: System keyspaces (system_auth, system_schema) must also be snapshotted.

Command: $ nodetool snapshot -t sysbackup system_auth

Best practices: Always back up auth tables to preserve roles and permissions.

Q116. How do you handle backups in a multi-DC cluster?

Answer: - Take local snapshots per DC. - Sync to centralized storage. - Ensure consistent timestamp across DCs.

Best practices: Use LOCAL_QUORUM for backup consistency.

Q117. What are best practices for incremental backups?

Answer: - Enable in cassandra.yaml: incremental_backups: true - Clean up periodically to avoid disk bloat.

Best practices: Combine with full snapshots weekly.

Q118. How do you schedule automated backups?

Answer: - Use cron jobs or orchestration tools (Ansible, Kubernetes CronJobs). - Automate nodetool snapshot + file copy to cloud storage.

Best practices: Centralize logs and alerts for failures.

Q119. How do you perform point-in-time recovery?

Answer: - Restore snapshot + apply incremental backups. - Requires restoring SSTables up to desired timestamp.

Best practices: Maintain clear backup catalog and timestamps.

Q120. What is the difference between nodetool snapshot and incremental backups?

Answer: - Snapshot: Full copy of SSTables at a point in time (hard links). - Incremental: Copies only new SSTables flushed since last snapshot.

Best practices: Use both — snapshots weekly, incremental daily/hourly.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 7: Monitoring & Troubleshooting (Q121–Q130)

Q121. How do you check cluster health in Cassandra?

Answer: Use nodetool status.

Command: $ nodetool status

Shows node states (UN, DN), load, tokens, and DC distribution.

Best practices: Automate checks via scripts/monitoring dashboards.

Q122. How do you use nodetool tpstats?

Answer: tpstats shows thread pool statistics.

Command: $ nodetool tpstats

Monitor: - Active, Pending, Blocked tasks. - Dropped message counts.

Best practices: Alert on high pending tasks or dropped messages.

Q123. How do you monitor compaction activity?

Answer: Use nodetool compactionstats.

Command: $ nodetool compactionstats

Shows pending compactions, completed tasks.

Best practices: Alert if pending compactions stay high.

Q124. How do you analyze Cassandra logs?

Answer: Logs located at /var/log/cassandra.

- system.log → startup, errors, warnings. - debug.log → detailed tracing (if enabled).

Best practices: Centralize logs with ELK/Fluentd for analysis.

Q125. How do you trace slow queries?

Answer: Enable tracing per query.

CQL: CONSISTENCY QUORUM; TRACING ON; SELECT * FROM table WHERE ...;

Best practices: Disable tracing after use; high overhead in production.

Q126. How do you detect GC issues?

Answer: Check gc.log.

Symptoms: - Long pauses (>1s). - Frequent Full GC.

Best practices: Tune heap size, use G1GC, monitor via JMX.

Q127. How do you capture a heap dump?

Answer: Use jmap or jcmd.

Command: $ jmap -dump:live,format=b,file=heap.bin

Best practices: Capture during off-peak, analyze with Eclipse MAT.

Q128. How do you monitor OS-level performance?

Answer: Tools: - top, iostat, vmstat, sar. - iostat → disk I/O latency. - vmstat → CPU, memory pressure.

Best practices: Collect via Prometheus Node Exporter.

Q129. How do you use JMX for Cassandra monitoring?

Answer: Cassandra exposes metrics via JMX (default port 7199).

Tools: JConsole, VisualVM, Prometheus JMX exporter.

Best practices: Secure JMX with authentication & SSL.

Q130. How do you identify hot partitions?

Answer: Hot partitions cause uneven load.

Detection: - nodetool tablestats (large partitions). - Metrics: partition_size, read_latency.

Best practices: Use bucketing, monitor partition sizes regularly.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 7: Monitoring & Troubleshooting (Q131–Q140)

Q131. How do you troubleshoot high read latency?

Answer: Check for: - Large partitions or wide rows. - Tombstones from TTL/deletes. - Compaction backlog.

Tools: $ nodetool tablestats $ nodetool compactionstats

Best practices: Redesign schema, tune compaction, avoid ALLOW FILTERING.

Q132. How do you troubleshoot high write latency?

Answer: Causes: - Slow commit log disk. - Memtable flush stalls. - Overloaded compaction.

Tools: $ nodetool tpstats $ iostat

Best practices: Use SSDs, separate commit log, increase memtable_flush_writers.

Q133. How do you detect dropped mutations?

Answer: Dropped mutations = writes that could not be delivered.

Command: $ nodetool tpstats

Check "Dropped Mutations" metric.

Best practices: Investigate network, increase concurrent_writes, monitor GC.

Q134. How do you troubleshoot gossip issues?

Answer: Check system.log for gossip errors.

Command: $ nodetool gossipinfo

Common issues: snitch misconfig, firewall blocks.

Best practices: Ensure consistent snitch config, open ports 7000/7001.

Q135. How do you debug repair failures?

Answer: Logs: system.log, debug.log.

Commands: $ nodetool repair $ nodetool netstats

Best practices: Run incremental repairs, check network, adjust streaming throttle.

Q136. How do you troubleshoot compaction stalls?

Answer: Stalls occur when compaction backlog grows.

Tools: $ nodetool compactionstats

Fix: - Increase compaction_throughput_mb_per_sec. - Add disk capacity.

Best practices: Monitor compaction metrics, avoid under-provisioning storage.

Q137. How do you troubleshoot schema disagreement?

Answer: Check via nodetool describecluster.

Command: $ nodetool describecluster

Cause: concurrent schema changes, network issues.

Best practices: Apply schema changes with migrations, not cqlsh ad-hoc.

Q138. How do you detect network latency issues?

Answer: Monitor inter-node latency.

Tools: $ nodetool statusbinary $ ping, traceroute

Best practices: Place nodes in low-latency network, enable internode compression if bandwidth limited.

Q139. How do you handle JVM thread pool exhaustion?

Answer: Symptom: high pending tasks in tpstats.

Fix: - Increase concurrent_reads/writes. - Optimize schema to reduce workload.

Best practices: Scale horizontally if saturation persists.

Q140. How do you identify and resolve disk space issues?

Answer: Monitor via df -h, nodetool info.

Fix: - Run nodetool cleanup/compact. - Add more nodes/disks.

Best practices: Keep <70% disk utilization to leave headroom for compaction.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 8: Advanced Topics & Multi-DC Operations (Q141–Q150)

Q141. How do you configure Cassandra for multi-DC replication?

Answer: Use NetworkTopologyStrategy for keyspaces.

Example: CREATE KEYSPACE ks WITH replication = { 'class': 'NetworkTopologyStrategy', 'dc1': 3, 'dc2': 3 };

Best practices: Define RF per DC, avoid SimpleStrategy in production.

Q142. How do you manage consistency across multiple DCs?

Answer: - LOCAL_QUORUM: Low-latency, consistency within DC. - EACH_QUORUM: Ensures quorum in each DC, stronger consistency. - QUORUM: Quorum across all replicas in all DCs.

Best practices: Use LOCAL_QUORUM for most workloads, EACH_QUORUM for strict global consistency.

Q143. How does hinted handoff work in multi-DC clusters?

Answer: Hints are stored locally and replayed when remote DC node recovers.

Impact: Large inter-DC replay after outages can stress network.

Best practices: Tune hinted handoff window, consider disabling in multi-DC with long outages.

Q144. How do you run repairs in geo-distributed clusters?

Answer: Use incremental repairs, preferably per DC.

Command: $ nodetool repair --dc dc1

Best practices: Schedule repairs per DC, align with gc_grace_seconds.

Q145. What are best practices for inter-DC network topology?

Answer: - Use dedicated private links (VPN/MPLS). - Minimize cross-DC latency (<100ms recommended). - Enable internode compression.

Best practices: Monitor network throughput, use LOCAL_QUORUM for reads/writes.

Q146. How do you handle cross-DC latency issues?

Answer: - Use LOCAL_QUORUM for app queries. - Increase request_timeout_in_ms for long-latency links. - Optimize speculative retry.

Best practices: Avoid EACH_QUORUM unless mandatory.

Q147. How do you troubleshoot schema disagreement in multi-DC?

Answer: Check via nodetool describecluster.

Command: $ nodetool describecluster

Best practices: Apply schema changes with migrations, avoid concurrent changes, verify schema agreement after update.

Q148. Why is clock synchronization critical in Cassandra?

Answer: Timestamps determine last-write-wins.

Tools: NTP, Chrony.

Best practices: Ensure NTP sync across all nodes, monitor clock drift.

Q149. How do you handle a full DC outage?

Answer: - Use LOCAL_QUORUM for surviving DCs. - Set RF in each DC >=2 to maintain availability. - Rebuild failed DC when restored.

Best practices: Plan RF per DC for failover resilience.

Q150. How do you expand to a new region/DC safely?

Answer: Steps: 1. Add new DC with proper snitch configuration. 2. Set replication factor for new DC. 3. Run nodetool rebuild to stream data.

Best practices: Add nodes gradually, verify replication health after rebuild.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 8: Advanced Topics & Multi-DC Operations (Q151–Q160)

Q151. What are the limitations of materialized views in Cassandra?

Answer: - Experimental feature, not recommended for production. - Potential for data inconsistency. - Higher write amplification.

Best practices: Use manually denormalized tables instead of MVs.

Q152. What are pitfalls of secondary indexes?

Answer: - Work poorly with high-cardinality columns. - Performance degrades with large datasets. - Not distributed efficiently.

Best practices: Use only on low-cardinality fields, prefer denormalization.

Q153. What are SASI indexes and when to use them?

Answer: SASI = SSTable-Attached Secondary Index.

Supports LIKE queries, range queries.

Best practices: Use for small datasets or search-like queries, not large-scale workloads.

Q154. How does Change Data Capture (CDC) work in Cassandra?

Answer: CDC logs mutations to commitlog CDC directory.

Config (cassandra.yaml): cdc_enabled: true

Best practices: Stream CDC logs to external systems (Kafka, Spark).

Q155. How do you handle large partitions in Cassandra?

Answer: Large partitions cause high latency and GC pressure.

Best practices: - Use bucketing strategies (time-based, logical). - Keep partitions <200MB. - Monitor via nodetool tablestats.

Q156. What is tiered storage in Cassandra?

Answer: Tiered storage = placing hot data on SSDs, cold data on HDDs/cloud.

Implementation: via custom compaction + storage-attached indexes.

Best practices: Keep active data on SSDs, historical data in cheaper storage.

Q157. How do you isolate workloads in Cassandra?

Answer: Use separate keyspaces/tables, or dedicate clusters.

Config: workload: transactional / analytical

Best practices: Avoid mixing OLTP + OLAP in same cluster; use Spark for analytics.

Q158. How do you handle multi-tenant workloads?

Answer: Include tenant_id in partition key.

Best practices: Monitor tenant skew, apply quotas, isolate noisy tenants.

Q159. What tools exist for managing repairs at scale?

Answer: - Cassandra Reaper: automates repairs. - Nodetool repair: manual.

Best practices: Use Cassandra Reaper in large clusters for automated incremental repair.

Q160. How do you monitor and optimize cross-DC repairs?

Answer: - Run repairs per DC separately. - Use incremental repairs. - Tune streaming throughput.

Best practices: Schedule during low traffic, monitor bandwidth usage.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 9: Operations at Scale & Automation (Q161–Q170)

Q161. How do you automate node provisioning in Cassandra?

Answer: - Use configuration management tools (Ansible, Puppet, Chef). - Automate installation, cassandra.yaml configuration, and service start.

Best practices: Bake Cassandra AMIs/Docker images with pre-configured settings.

Q162. How do you deploy Cassandra on Kubernetes?

Answer: Use StatefulSets with persistent volumes.

Tools: cass-operator, K8ssandra.

Best practices: One pod = one Cassandra node, use anti-affinity rules for resilience.

Q163. How do you handle rolling restarts in large clusters?

Answer: - Restart nodes one at a time. - Use nodetool drain before stop. - Verify node rejoined with nodetool status.

Best practices: Automate rolling restarts with scripts/Ansible.

Q164. How do you automate repairs in Cassandra?

Answer: - Use Cassandra Reaper for scheduled, incremental repairs. - Integrate with monitoring/alerting.

Best practices: Automate per keyspace/DC repairs, align with gc_grace_seconds.

Q165. How do you scale Cassandra clusters automatically?

Answer: - Use monitoring to detect resource saturation. - Provision new nodes via automation (Terraform/Ansible). - Bootstrap and join cluster automatically.

Best practices: Use predictive scaling for workloads with known patterns.

Q166. How do you manage schema migrations at scale?

Answer: - Use migration tools (Liquibase, Flyway). - Apply schema changes via controlled rollout.

Best practices: Avoid ad-hoc cqlsh changes; test migrations in staging.

Q167. How do you handle version upgrades at scale?

Answer: - Automate rolling upgrades with orchestration tools. - Upgrade one DC at a time in multi-DC.

Best practices: Always read release notes, verify schema agreement after upgrade.

Q168. How do you monitor large clusters effectively?

Answer: - Use Prometheus + Grafana dashboards. - Monitor key metrics: latency, compaction, dropped mutations, heap.

Best practices: Centralize logs and alerts, use service discovery.

Q169. How do you automate backups in large clusters?

Answer: - Use scripts or orchestration to run nodetool snapshot across nodes. - Push snapshots to centralized/cloud storage.

Best practices: Schedule incremental + full backups, automate cleanup.

Q170. How do you perform capacity planning in Cassandra?

Answer: - Monitor disk, CPU, memory, network usage per node. - Estimate growth based on writes/day and compaction overhead.

Best practices: Keep disk <70%, plan 6–12 months ahead, benchmark with cassandra-stress.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 9: Operations at Scale & Automation (Q171–Q180)

Q171. How do you manage Cassandra clusters with Terraform?

Answer: - Use Terraform for infrastructure provisioning (instances, networking, storage). - Combine with Ansible for Cassandra installation.

Best practices: Version control infra configs, integrate with CI/CD.

Q172. How do you containerize Cassandra effectively?

Answer: - Use official Cassandra Docker images. - Persist data with volumes. - Configure JVM and cassandra.yaml via env vars.

Best practices: Use Kubernetes for orchestration, StatefulSets for stable identities.

Q173. How do you manage multi-region clusters at scale?

Answer: - Use NetworkTopologyStrategy for RF per region. - Automate node provisioning and repairs.

Best practices: Monitor inter-region latency, use LOCAL_QUORUM.

Q174. How do you automate schema synchronization across environments?

Answer: - Use migration frameworks (Flyway, Liquibase). - Apply versioned schema scripts.

Best practices: Enforce schema changes via CI/CD pipelines.

Q175. How do you manage secrets in automated Cassandra deployments?

Answer: - Store secrets in Vault/Kubernetes secrets. - Avoid plaintext passwords in configs.

Best practices: Rotate DB credentials regularly, enforce RBAC.

Q176. How do you automate rolling upgrades across DCs?

Answer: - Upgrade one node at a time per DC. - Automate drain → stop → upgrade → restart.

Best practices: Upgrade one DC at a time, verify schema agreement.

Q177. How do you automate scaling Cassandra in cloud environments?

Answer: - Use monitoring to trigger scaling events. - Provision nodes with Terraform/CloudFormation.

Best practices: Test scaling scripts in staging, use predictive scaling for spikes.

Q178. How do you enforce configuration consistency across nodes?

Answer: - Use Ansible/Puppet for cassandra.yaml distribution. - Automate checksum verification.

Best practices: Keep configs in version control, use templates.

Q179. How do you automate Cassandra deployments with CI/CD?

Answer: - Build Docker images with configs baked in. - Deploy via Jenkins/GitLab pipelines.

Best practices: Automate integration tests against staging clusters.

Q180. How do you ensure operational readiness of Cassandra clusters?

Answer: - Automated monitoring/alerting in place. - Regular DR drills and backup tests. - Automated repair and capacity planning.

Best practices: Document runbooks, rehearse incident response.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 10: Case Studies & Real-World Scenarios (Q181–Q190)

Q181. How would you design Cassandra for an e-commerce order system?

Answer: Schema: - orders_by_customer - orders_by_date

Best practices: Denormalize for query patterns (customer history, daily reports). Use QUORUM for order writes.

Q182. How do you handle IoT sensor data ingestion at scale?

Answer: - Partition by device_id + day. - Cluster by timestamp.

Best practices: Use TWCS for TTL data, monitor hot partitions, scale writes with LOCAL_QUORUM.

Q183. How do you support a recommendation engine with Cassandra?

Answer: - Store precomputed recommendations per user. - Partition by user_id.

Best practices: Use Spark for computation, Cassandra for storage.

Q184. How would you design a time-series analytics platform?

Answer: - Partition by metric_id + time bucket. - Cluster by timestamp DESC.

Best practices: Use TWCS, avoid unbounded partitions, offload cold data to cheaper storage.

Q185. How do you handle a workload with heavy deletes?

Answer: - Avoid frequent deletes; use TTL instead. - Monitor tombstones via tablestats.

Best practices: Use TWCS for TTL-heavy workloads.

Q186. How would you design Cassandra for chat/messaging app?

Answer: - messages_by_user (partition by user_id, cluster by ts). - messages_by_conversation (partition by conversation_id).

Best practices: Use DESC clustering for recent messages, denormalize per query.

Q187. How do you handle global user base with low-latency requirements?

Answer: - Deploy multi-DC cluster. - Use LOCAL_QUORUM for reads/writes.

Best practices: Tune request timeouts, monitor inter-DC latency.

Q188. How do you support GDPR-compliant data deletion?

Answer: - Use TTL for auto-expiry. - Ensure regular repairs to propagate tombstones.

Best practices: Align gc_grace_seconds with retention, document compliance.

Q189. How do you handle financial transactions with strict consistency?

Answer: - Use EACH_QUORUM or SERIAL LWT for linearizability. - Ensure RF=3+ per DC.

Best practices: Accept higher latency for strict correctness.

Q190. How would you design for multi-tenant SaaS workloads?

Answer: - Include tenant_id in partition key. - Isolate large tenants into separate clusters if needed.

Best practices: Monitor tenant skew, apply quotas, enforce RBAC.

Cassandra Architecture & Operations Interview Handbook

Prepared by Debasis Maity

Chapter 10: Case Studies & Real-World Scenarios (Q191–Q200)

Q191. How do you design Cassandra for a ride-hailing application?

Answer: - trips_by_driver (partition by driver_id, cluster by trip_ts). - trips_by_rider (partition by rider_id).

Best practices: Use TTL for location updates, QUORUM for trip writes.

Q192. How would you support streaming analytics with Cassandra?

Answer: - Ingest into Cassandra, process with Spark/Flink. - Partition by stream_id + time bucket.

Best practices: Keep raw data TTL-based, store aggregates in Cassandra.

Q193. How do you design a real-time leaderboard system?

Answer: - Partition by game_id. - Cluster by score DESC.

Best practices: Use bucketing for high cardinality, store top-N in memory cache.

Q194. How would you design logging/audit storage in Cassandra?

Answer: - Partition by date/hour + service. - Cluster by timestamp.

Best practices: Use TWCS, compress logs, offload old logs to S3/HDFS.

Q195. How do you handle multi-cloud Cassandra deployments?

Answer: - Use NetworkTopologyStrategy with DC per cloud. - Ensure VPN/peering between clouds.

Best practices: Avoid cross-cloud replication unless latency acceptable.

Q196. How do you support hybrid transactional/analytical processing (HTAP)?

Answer: - Use Cassandra for OLTP, Spark/SparkSQL for OLAP. - Keep OLAP queries off production cluster.

Best practices: Use dual clusters or DSE Analytics.

Q197. How would you migrate data from RDBMS to Cassandra?

Answer: Steps: 1. Model queries → schema design. 2. Use DSBulk or custom ETL for migration. 3. Validate data consistency.

Best practices: Avoid 1:1 schema mapping, design for access patterns.

Q198. How do you design for high availability in financial services?

Answer: - RF=3+ per DC, LOCAL_QUORUM for reads/writes. - Multi-DC deployment.

Best practices: Strict monitoring, regular DR drills.

Q199. How would you use Cassandra for content management?

Answer: - Partition by content_id. - Store metadata in Cassandra, binary in object store.

Best practices: Keep large blobs outside Cassandra, use references.

Q200. How do you summarize Cassandra best practices from real-world ops?

Answer: - Model for queries, not relations. - Always run repairs aligned with gc_grace_seconds. - Monitor latency, compaction, dropped mutations. - Keep RF=3+ for HA, use LOCAL_QUORUM. - Automate backups, test restores, rehearse DR.