Z5008: Spark, Lakehouse & Streaming Engineering Workbook

Big Data Lab (MTech Core) · Labs 1–15 · innocent@iitmz.ac.in

This workbook walks through four production big data engineering calculations covering Delta/Iceberg table compaction, Spark join and skew tuning, Structured Streaming watermark state sizing, and MLflow/Prometheus observability alerting.

Problem 1: Delta Lake Small-Files Compaction and Iceberg Field IDs

An IoT ingestion pipeline writes 120 micro-batches per hour (each writing 8 tiny 1.5 MB Parquet files) into a Delta Lake table partitioned by `event_date`. Diagnose the storage overhead over 30 days and prescribe maintenance jobs plus an Iceberg schema-evolution check.

Worked Step-by-Step Solution

1. **Small-File Explosion:** $120 \times 8 \times 24 \times 30 = 691{,}200$ Parquet files totaling ≈ 1.04 TB, c latency and thousands of tiny Spark driver tasks.
2. **Compaction Schedule:** Run daily `OPTIMIZE events WHERE event_date >= cu (device_id)` to coalesce into ≈ 256 MB files ($\sim 4{,}100$ files total, a 168× file reduction `events RETAIN 168 HOURS` to purge superseded files safely past active reader snapshots.
3. **Iceberg Schema Safety:** When renaming `temp_c` to `temperature_c` in Apache Icebe `field_id` mappings avoid rewriting historical Parquet files while preventing positional co and Trino.

Problem 2: Spark Broadcast Thresholds, Executor Sizing, and Skew Salting

A cluster has 8 worker nodes (16 vCPUs, 64 GB RAM each). You join a 600 GB `fact_txns` table with a 60 MB `dim_merchants` table, followed by a `groupBy("country_code")` where `'TZ'` accounts for 65% of all rows.

Worked Step-by-Step Solution

1. **Executor Allocation:** Reserve 1 core and 4 GB per node for OS/daemon overhead. Allocate 3 executors per node × 5 cores × 18 GB heap (leaving ~ 2 GB off-heap overhead per executor) = 24 executors, 120 total executor cores.
2. **Eliminating Fact-Side Shuffle:** Default `autoBroadcastJoinThreshold` is 10 MB, forcing a massive 600 GB sort-merge shuffle. Raising `spark.sql.autoBroadcastJoinThreshold = 100MB` (or calling `F.broadcast(dim_merchants)`) eliminates the fact-table shuffle cleanly within 18 GB executor RAM.
3. **Two-Stage Salted Aggregation for `'TZ'`:** Map `salt_key = concat(country_code, '_', floor(rand() * 20))` to spread `'TZ'` across 20 parallel partitions for partial `SUM`/`COUNT`, then drop the salt suffix in stage 2 to compute final totals per country without straggler tasks.

Problem 3: Spark Structured Streaming Watermarks and Output Modes

Events carry `event_time`. Trigger runs every 1 minute with `.withWatermark("event_time", "3 minutes")` and 5-minute tumbling windows. Determine which late events are accepted or dropped and compare `Append` vs. `Update` modes.

Worked Step-by-Step Solution

1. **Watermark Calculation:** Suppose batch k observes maximum `event_time = 10:14:00`. Watermark updates to $W = 10{:}14{:}00 - 3 \text{ min} = \mathbf{10:11:00}$.
2. **Late-Event Fate:** An arriving packet stamped `10:11:30` exceeds W, so Spark accepts it into window `[10:10, 10:15)`. A packet stamped `10:09:45` falls strictly below W, and its window `[10:05, 10:10)` has closed at 10:10:00 ($< W$), so Spark drops it safely and evicts `[10:05, 10:10)` state from RocksDB.
3. **Output Mode Trade-Off:** `Append` emits finalized rows only after window end $< W$ (required for immutable Delta/Parquet file sinks without updates), whereas `Update` emits revised counts every micro-batch to stateful stores such as Redis or Delta `MERGE`.

Problem 4: MLflow Registry Promotion Gates and Prometheus SLA Alerting

Design automated CI promotion rules for a BentoML fraud-scoring service backed by MLflow and monitored via

Prometheus + Grafana under a 50 ms p99 latency SLA.

Worked Step-by-Step Solution

1. **Registry Lifecycle Gate:** Log run parameters, Delta table snapshot ver
   Enforce a CI script checking `f1 >= 0.88` and `calib_ece <= 0.03` befo
   archiving the outgoing champion).
2. **Prometheus Telemetry:** Expose `inference_requests_total{status}`
   `inference_latency_ms_bucket` (Histogram computing `histogram_quant`
3. **Canary Rollback Trigger:** Route 5% live traffic to the new BentoML co
   rollback to the archived Production tag if p99 latency exceeds 50 ms for 3