Bilko Demo Event-Loop Freeze — Root Cause, Fix & Hardening (2026-06-08, MC #103134)

Bilko Demo Event-Loop Freeze — Root Cause, Fix & Hardening (2026-06-08, MC #103134)

1. Incident Summary

Date: 2026-06-08
Service: bilko-api-demo (Cloud Run, europe-north1, tribal-sign-487920-k0)
Live revision at incident: bilko-api-demo-00139-b26 (git sha c8dfb6c)

MC: #103134 (child of #103132)
Detected by: CEO (Alem Basic) — no automated alerts existed

Symptom: POST /api/v1/auth/login and GET /api/v1/health intermittently returned HTTP 504 with server-side latency of exactly 59.999 s (Cloud Run hard timeout). Observed failure rate: 17–25% of requests. Both endpoints froze simultaneously on both Cloud Run instances. A manual container restart did not fix the issue.

Why restart did not help: The blocking code paths are baked into the deployed image (c8dfb6c). The freeze re-triggers the moment any user navigates to Reports or Billing/Usage, which re-executes the same bare JDBC calls on the Netty event-loop thread.

---

2. Root Cause — Causal Chain

Source: 5-agent synthesis (petter-synthesis.md) + source-verified app-layer audit (dev-applayer.md) + data/storage audit (devops2-data.md).

2.1 Ktor Netty Thread Pool Exhaustion (Primary)

Ktor runs on Netty. On a 1-vCPU Cloud Run container with no `callGroupSize` set (main branch pre-fix), Netty defaults to approximately 2 call-handling threads. With `containerConcurrency=8`, Cloud Run routes up to 8 simultaneous requests into the container, all competing for those 2 threads.

Any route that executes a bare `orgTransaction()` call — not wrapped in `dbQuery{}` — runs its JDBC work directly on the Netty call thread, blocking it for the duration of the DB round-trip (10–100 ms typical, up to 30 s if HikariCP must wait for a pool slot). With only 2 call threads and 8 concurrent slots, two simultaneous requests to a bare-orgTransaction route exhaust both call threads. The event loop is frozen.

Confirmed bare orgTransaction sites on main at incident time:

File	Line(s)	Route
ReportRoutes.kt	26	GET /reports (country lookup)
ReportRoutes.kt	240	GET /reports/kpo (country pre-check)
ReportRoutes.kt	264	GET /reports/kpo/export/pdf (country pre-check)
ReportRoutes.kt	285	GET /reports/kpo/export/xlsx (country pre-check)
BillingRoutes.kt	214	GET /billing/usage (plan tier + invoice count)
MarketRoutes.kt	2 sites	resolveMarketPlugin call sites
ResourceAccessFilter.kt	multiple	Per-request security filter
Authentication.kt	impersonation path	Per impersonation request

2.2 PermissionService.loadFromDb — New RBAC Blocking Call (Critical, Entra cutover)

The Entra External ID cutover (MC #103140) introduced RBAC via `requirePermission()`, which is called on every authenticated request across all 17 route files. On first call per role (4 roles: owner/admin/accountant/viewer), `resolve()` calls `loadFromDb()`:

```
// BEFORE fix — blocking on Netty event-loop thread
private fun loadFromDb(role: String): Set<String> {
    return transaction {  // bare JDBC on Netty thread
```

```
        RolePermissionsTable.selectAll()...
    }
}
```

This fires on every cold-start / scale-out event for every authenticated route. Without the fix, the Entra cutover would have immediately re-introduced a freeze worse than the original: 4 role cache-misses per instance restart, each running blocking JDBC on the event loop. Source: /tmp/evidence-103134/rebase-on-entra.md.

2.3 HikariCP connectionTimeout=30,000 ms (Secondary Amplifier)

When the `Dispatchers.IO` pool workers (correctly dispatched DB calls) hit HikariCP pool pressure — 10 connections shared across 8 concurrent slots — a pool-borrow wait can last up to 30 s per request. Two stacked pool-borrow waits = 60 s = Cloud Run hard kill. This amplifier remains even after the bare-orgTransaction calls are fixed; it is addressed separately by reducing `connectionTimeout` to 5 s (P0.3).

2.4 Why /health Freezes (Not a DB Problem)

`HealthRoutes.kt` lines 10–21 are confirmed DB-independent (zero blocking calls, pure in-memory response). The health freeze is not because health touches the DB — it is because both Netty call threads are occupied by blocked orgTransaction callers. Health cannot be scheduled. The TCP startup probe continues to pass because the JVM still has port 4001 bound; only HTTP scheduling is stalled.

2.5 Why Login Freezes (Login Is the Victim, Not the Cause)

`AuthRoutes.kt` line 150 correctly wraps login in `dbQuery{}`. Login itself is not a blocking caller. Login freezes because it cannot obtain a call thread to execute — both threads are consumed by the bare-orgTransaction routes.

2.6 Tertiary — Absent Server-Side DB Timeouts

No `statement_timeout`, `idle_in_transaction_session_timeout`, or `lock_timeout` on Cloud SQL (confirmed: databaseFlags null). A single slow query (vacuum contention on db-f1-micro, missing index on a growing table) holds a pool connection indefinitely, compounding pool pressure.

2.7 gcsfuse Is Not a Cause of Auth/Health Freezes

gcsfuse GC timestamps do not align with freeze onsets (GC at 06:35:04 UTC, freeze at 06:34:52 and 06:35:43). Login does no file I/O. gcsfuse FUSE blocking I/O affects receipt endpoints only; it is a separate hygiene issue and was also offloaded to `Dispatchers.IO` in the fix.

---

3. Fix Applied (MC #103134)

Branch: `fix/demo-freeze-on-entra-103134`, commit 5252182, rebased on `origin/main` @ b21b366 (Entra cutover). Merged as PR #279 → main 839ee7f. Source: /tmp/evidence-103134/rebase-on-entra.md.

Code Fixes

File	Change
routes/ReportRoutes.kt (4 sites)	Bare orgTransaction → wrapped in dbQuery{}
routes/BillingRoutes.kt (line 214)	Bare orgTransaction in /billing/usage → dbQuery{}
routes/MarketRoutes.kt (2 sites)	resolveMarketPlugin call sites → dbQuery{}
plugins/ResourceAccessFilter.kt	fetchResourceOwner + logSecurityViolation → suspend + withContext(IO); Thread.sleep → delay
plugins/Authentication.kt	Impersonation transaction → withContext(Dispatchers.IO)
services/PermissionService.kt (new finding)	loadFromDb() → runBlocking { withContext(Dispatchers.IO) { ... } } — RBAC per-request DB call offloaded
services/ReceiptService.kt	readLocalDocument + persistLocalIfEnabled → suspend + withContext(IO)
routes/ExpenseRoutes.kt	readLocalDocument moved out of dbQuery{} (now suspend-safe)
plugins/Database.kt	connectionTimeout 30,000 → 5,000 ms
routes/HealthRoutes.kt	Added GET /api/v1/health/deep — DB-touching endpoint with withTimeout 3 s + Dispatchers.IO + SELECT 1

Infrastructure Fixes (cloudbuild-stage.yaml + cloudbuild.yaml)

Change	Value
Memory	512 Mi → 1 Gi
Liveness probe	httpGet /api/v1/health/deep, port 4001, initialDelaySeconds 30, periodSeconds 30, failureThreshold 2, timeoutSeconds 8
Startup probe	httpGet /api/v1/health (was TCP), port 4001, periodSeconds 10, failureThreshold 3, timeoutSeconds 5
containerConcurrency	80 → 8 (in cloudbuild-demo-api.yaml; applied to stage + demo templates)

Build and Test Results

```
./gradlew build -x detekt → BUILD SUCCESSFUL in 12s
Tests: 2561 passed, 1 skipped (EntraLiveCiamTokenWp5Test — stale token, assumeTrue guard), 0 failures
```

PermissionService Cache Note

The 4-role cache (one entry per role after first load) bounds future DB hits. Pre-warming the cache at JVM startup is recommended as a follow-on polish task (task #7), but is not a blocker for the freeze fix: the IO offload alone prevents the event-loop block on cold misses.

---

4. Industry-Standard Scorecard

Condensed from petter-synthesis.md §2. Full details in audit reports.

Application Layer

Component	Assessment	Finding
Ktor/Netty callGroupSize	DEVIATES (pre-fix)	No callGroupSize set on main. Defaults to ~2 call threads on 1 vCPU. Fix branch sets callGroupSize=32; must be confirmed active in Cloud Run (env var KTOR_CALL_GROUP_SIZE=32 or production application.yaml profile).

Component	Assessment	Finding
dbQuery{} discipline	DEVIATES → FIXED	5 confirmed bare orgTransaction sites + PermissionService.loadFromDb all wrapped in this fix. 97 raw transaction{} calls in repo; remaining ones are safely nested inside outer dbQuery{} or on non-hot paths.
Login path (AuthRoutes.kt:150)	STANDARD	Correctly wraps in dbQuery{}. Login was the victim of thread starvation, not a cause.
BCrypt dispatcher	DEVIATES (minor)	BCrypt (CPU-bound, 250–400 ms) runs on Dispatchers.IO (I/O-bound). Not the freeze cause but degrades throughput. Move to Dispatchers.Default is a follow-on (P1.3).
HikariCP connectionTimeout	DEVIATES → FIXED	30,000 ms → 5,000 ms. Pool-borrow wait now fails fast with 503 in 5 s.
Login 3-transaction pool pressure	DEVIATES	AuthService.login() makes 3 separate orgTransaction{} calls (3 pool borrows per login). Follow-on: consolidate to 1 (P1.1).
Server-side RequestTimeout	DEVIATES	No Ktor RequestTimeout plugin. Stuck handlers are killed by Cloud Run at 60 s, not the app at 10 s. Follow-on: P1.5.
/health endpoint	STANDARD	DB-independent, correct. Preserved as-is for startup probe.
/health/deep endpoint	ADDED	New DB-touching endpoint (SELECT 1 via HikariCP, 3 s internal timeout). Returns 200 {db:up} or 503 {db:down}.
gcsfuse (ReceiptService)	DEVIATES → FIXED (hygiene)	Blocking POSIX I/O on FUSE mount offloaded to withContext(IO). Not the auth/health freeze cause, but blocks safe concurrency > 1–8 without this fix.

Infrastructure Layer

Component	Assessment	Finding

containerConcurrency	DEVIATES	8 with blocking event-loop paths present. Correct value is 1 until all blocking paths are confirmed on Dispatchers.IO. Prior safe state (v0.2.33) was concurrency=1. Fix branch applies concurrency=8 in cloudbuild templates (improvement from 80 default; full concurrency=1 revert deferred pending gcsfuse→SDK swap per P1.2).
minScale/maxScale	DEVIATES	min=2, max=2. Pinned — no relief valve. With maxScale=2 and both instances holding frozen slots, no new instance can absorb traffic. Follow-on: maxScale=5 paired with pool=2 (P2.2).
HikariCP pool math	FRAGILE	pool=10, maxScale=2 = 20 connections vs ~22 usable on db-f1-micro. Within bounds by 2 connections only. Any scale-up to 3 instances without reducing pool causes exhaustion.
Cloud SQL databaseFlags	DEVIATES	Zero flags. No statement_timeout, no idle_in_transaction_session_timeout, no lock_timeout. Follow-on: MC #103133 (blocked on GCP-write channel MC #103149).
Cloud SQL tier	DEVIATES	db-f1-micro: max_connections=25, shared CPU, VACUUM contention risk. Follow-on recommendation: db-g1-small minimum for customer-facing demo.
Liveness probe	ABSENT → ADDED	HTTP liveness on /health/deep added. Without it, frozen containers ran for the full incident duration with no auto-recovery.
Startup probe	DEVIATES → FIXED	Was TCP period=240s failureThreshold=1 timeout=240s. Now HTTP GET /health period=10s failureThreshold=3 timeout=5s.
Cloud Monitoring alerts	DEVIATES (critical)	0 policies (confirmed: gcloud monitoring policies list = 0 items). Incident was CEO-detected. Follow-on: P2.1 (5 alert policies).
Memory	DEVIATES → FIXED	512 Mi → 1 Gi. JVM + HikariCP + gcsfuse daemon on 512 Mi was tight; GC pressure is a secondary freeze vector.

Cloud SQL public IP	DEVIATES	ipv4Enabled=true. No unauthorized networks found (mitigated), but public attack surface exists. Follow-on: private IP + VPC (P2 hardening).
CPU throttling / startup-cpu-boost / gen2	STANDARD	All correct. No change needed.

---

5. Deploy Path and Cross-Session Coordination

Source: /tmp/alai/p2p-pairing-evidence/john-7fedd67f-freeze-fix-coordination-20260608.md

This fix was developed in session 7fedd67f and rebased on top of the Entra External ID cutover branch (b21b366). The coordination agreement between the freeze-fix session and the Entra/RBAC workstream (MC #103080, Phase 4) is:

1. **Step 1 (complete):** PR #279 merged fix/demo-freeze-on-entra-103134 → main (839ee7f). Stage auto-deploy triggered (bilko-stage-auto-deploy: push to ^main$ → cloudbuild-stage.yaml). Stage revision bilko-api-stage-00531-cub deployed.
2. **Step 2 (pending):** Demo and production semver tag is owned by the Entra/RBAC workstream (MC #103080). That single semver tag vX.Y.Z ships Entra + RBAC + freeze fix together, via bilko-main-deploy trigger (^v.*$ → cloudbuild.yaml, 8 gates + approval). Neither session tags unilaterally.

Deploy pipeline mechanics (verified):

- `bilko-stage-auto-deploy`: triggered by push to branch matching ^main$, runs cloudbuild-stage.yaml
- `bilko-main-deploy`: triggered by semver tag matching ^v.*$, runs cloudbuild.yaml (demo, 8 gates + approval required)
- `cloudbuild-demo-api.yaml`: DEAD CODE — orphaned trigger deleted 2026-05-21. Do not reference or invoke.

---

6. Proveo Stage Validation — PASS

Source: /tmp/evidence-103134/proveo-stage-verdict.json + /tmp/evidence-103134/proveo-stage-validation.md
Timestamp: 2026-06-08T09:54:10Z
Revision: bilko-api-stage-00531-cub (git sha 839ee7f)

Check	Result	Detail
#1 /health x30 sequential	PASS	0/30 failures, max latency 170 ms
#1b /health x20 concurrent	PASS	0/20 failures, max latency 279 ms
#2 /health/deep x15 (DB-touching)	PASS	Endpoint present, db:up, all under 200 ms
#3 Concurrency starvation test (6 waves x20 = 120 requests)	PASS	0 non-200, 0x 504@60s. Wave 1 had 2/20 requests at ~10.8 s (HTTP 200, pool warm-up artifact, not freeze). Waves 2–6: zero over 2 s. Original 504@59.999s pattern NOT observed.
#4 Auth path	PASS	POST /auth/login → 410 (legacy retired); POST /auth/entra/session (invalid) → 400; GET /invoices (no auth) → 401; GET /contacts (no auth) → 401. Entra + RBAC enforced.
#5 Probe config	PASS	livenessProbe httpGet /health/deep present; startupProbe httpGet /health present; memory 1Gi; git sha 839ee7f confirmed.

Overall verdict: PASS. The event-loop freeze fix is confirmed on stage. The original 504@59.999s symptom is not reproduced under sustained concurrent load.

Caveat: Wave 1 had 2 requests at ~10.8 s (HTTP 200, not 504) — confirmed as connection pool warm-up artifact on first concurrent burst. Not a freeze regression. Waves 2–6 (100 requests) were all under 2 s.

---

7. Follow-Up Tasks

MC	Item	Status
#103133	Cloud SQL statement_timeout + idle_in_transaction_session_timeout + lock_timeout (gcloud sql instances patch)	Blocked on GCP-write channel MC #103149
#103148	Gate wc bug (claim-gate stale-transcript count issue)	Open
#103173	Claim-gate stale-transcript loop	Open
#103138	This documentation page (WS-D)	Complete (this page)
Task #7	PermissionService pre-warm cache at JVM startup (recommended polish)	Deferred

MC	Item	Status
P1.1	Consolidate AuthService.login() 3 transactions into 1 (reduce pool borrows per login from 3 to 1)	Deferred — daytime review
P1.2	Replace gcsfuse with GCS Storage SDK in ReceiptService (unlock safe concurrency > 8)	Deferred
P1.5	Install Ktor RequestTimeout plugin (force-kill stuck handlers at 10 s, return 503)	Deferred
P2.1	Create 5 Cloud Monitoring alert policies (5xx rate, p99 latency, instance ceiling, Cloud SQL connections, /health/deep uptime)	Deferred — no blocking dependency
P2.2	Scaling fix: containerConcurrency=1, maxScale=5, pool=2 (requires P1.2 gcsfuse→SDK first)	Deferred
P2.6	Add k6 event-loop-freeze regression gate to CI (tests/k6/event-loop-freeze.js)	Deferred — spec in test-observability.md §4 B5

8. Evidence References

- /tmp/alai/bilko-infra-team/petter-synthesis.md — 5-agent root-cause synthesis
- /tmp/alai/bilko-infra-team/devops1-cloudrun.md — Cloud Run compute/scaling audit
- /tmp/alai/bilko-infra-team/devops2-data.md — Cloud SQL / HikariCP / gcsfuse / networking audit
- /tmp/alai/bilko-infra-team/dev-applayer.md — Application layer audit (CodeCraft)
- /tmp/alai/bilko-infra-team/test-observability.md — Observability gaps + Proveo validation plan
- /tmp/evidence-103134/rebase-on-entra.md — Fix branch rebase on Entra main, PermissionService finding, build/test results
- /tmp/evidence-103134/proveo-stage-verdict.json — Stage validation machine-readable verdict
- /tmp/evidence-103134/proveo-stage-validation.md — Stage validation human-readable report
- /tmp/alai/p2p-pairing-evidence/john-7fedd67f-freeze-fix-coordination-20260608.md — Cross-session coordination with Entra workstream

Revision #1
Created 2026-06-08 11:00:33 UTC by John
Updated 2026-06-08 11:00:33 UTC by John