APQ Case Study

Source: Publicaciones · **Status:** Shipped · **Period:** 2025-11 (active)

Academic publication on persisted-query behaviour, payload reduction, and operational API discipline at 90.9% cache hit rate over a 135k-line schema.

The challenge

Large GraphQL schemas pay full request-body cost on every operation even when the operation is a repeat. The cost shows up as bandwidth, CPU on the server, and p95 latency on the wire.

My role

I published a case study on Automatic Persisted Queries at scale: 90.9% cache hit rate over a 135,504-line schema, 2,089 query types, 707 Prisma models, with a cgroup v2-isolated compile runner and a five-phase auto-warmup.

What changed

Cache hit rate in production

Baseline: 0%

90.9%

p95 latency reduction

Baseline: 25 ms without APQ

12 ms with APQ

Payload reduction on cached queries

Baseline: 100% body

25% body (75% reduction)

External SaaS spend for the APQ stack

Baseline: $0 — not yet implemented

$0/month — self-hosted Redis + Lua EVAL atomic counters

Stack

GraphQL | Apollo Server v4 | Redis 7 | IndexedDB