SOLUTION BRIEF

GraphQL

GraphQL streamlines data access, but its dynamic, payload- based requests break traditional CDN caching, resulting in latency, increased costs, and backend strain. Harper solves this with a fused, edge-native runtime that combines database, cache, application logic, and messaging functions into a single process. By resolving queries closer to users with cached data, Harper reduces egress and boosts performance. And with an incremental adoption path—from full-query caching to field-level routing—teams can see immediate gains without major rewrites.

Challenges with Common GraphQL Deployments

1. **Caching Breaks:** GraphQL's POST-based, single - endpoint architecture bypasses traditional path and header-based CDN caching, often resulting in zero cache hits.
2. **Latency Stacks:** Each field in a resolver chain may trigger separate cross-network calls, thereby delaying time-to-first-byte (TTFB) and negatively impacting Core Web Vitals.
3. **Origin Strain:** Without a caching layer, every query hits backend databases or microservices directly, driving up compute costs, database load, and operational strain.
4. **Freshness Lags:** Most GraphQL caching solutions lack native Change Data Capture (CDC) or event streaming capabilities, making it challenging to keep data fresh without relying on expensive polling or brittle revalidation workarounds.

Why Harper Is Built for Modern GraphQL

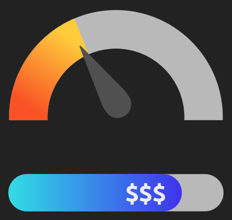
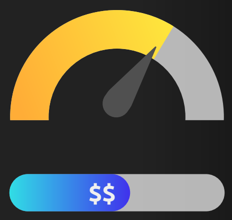
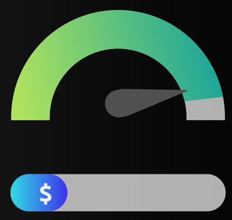
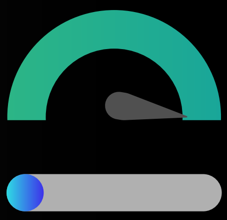
Most GraphQL platforms leave teams juggling too many moving parts - external databases, distributed caches, middleware, polling infrastructure—just to make a single query fast and fresh. Harper changes that by fusing the pieces together and bringing them to the edge.

Deployed at the edge near every user, Harper's fused runtime streamlines query resolution with one lightweight process. This architecture eliminates the traditional tradeoffs between speed, scale, and simplicity.

harper

harper

Incremental Adoption Path

Origin Offload Egress Costs	 $$$	 $$	 $	
	1 **Whole Query Caching**	2 **Partial Query Caching**	3 **Real-Time CDC**	4 **Full Edge-Native Delivery**
	Resolve and cache full query responses at the edge for instant latency gains and origin offload — no schema changes required.	Cache and serve reusable parts of a query, fetching only uncached fields to reduce origin load and improve performance.	Keep caches fresh automatically with change data capture —no polling or manual invalidation needed.	Run functions, transform APIs, and process data at the edge to simplify architecture and reduce backend load.
Use Cases	Product detail pages, search results, landing pages, or any high - read, low-churn query pattern.	Dynamic UIs with nested data (e.g. user dashboards, personalized content, carts), or federated schemas where different teams own different fields.	Inventory systems, pricing updates, user state or notification feeds, anything that needs fresh data.	Search interfaces, multi-format API gateways, auth checks, personalized content, mobile lookups, or A/B testing logic.

Getting Started Is Easy

Whether you're looking to reduce origin traffic, accelerate personalized UIs, or simplify GraphQL operations at scale, Harper meets you where you are. Start with whole-query caching for instant gains, or move straight into field-level control, real-time updates, and edge-native logic.

Harper can replace your existing GraphQL resolver, allowing you to maintain your current API contract while gaining performance and flexibility from the start. It's a lightweight switch with minimal impact on clients and a clear path to deeper optimization over time.

Start small. Scale fast. Modernize GraphQL without disrupting your users.

Contact Sales at hello@harperdb.io.