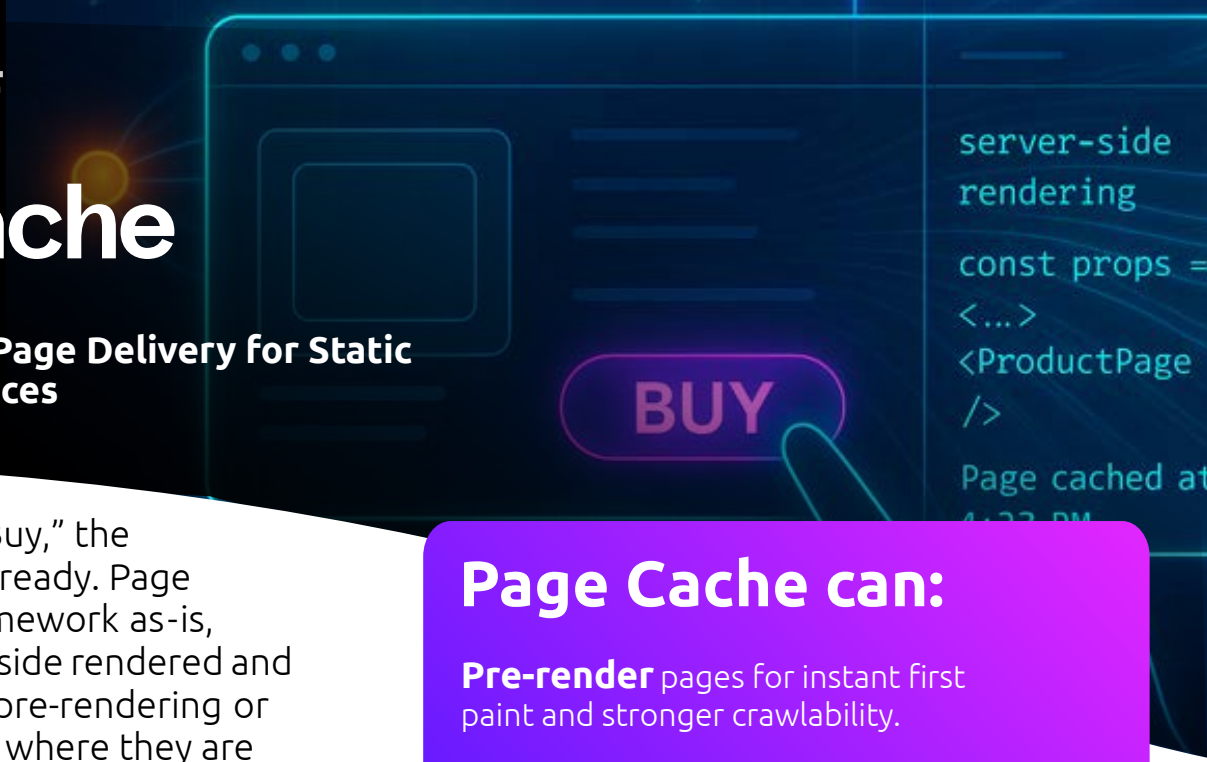
SOLUTION BRIEF

Page Cache

Low Latency Product Page Delivery for Static and Dynamic Experiences



When a shopper taps “Buy,” the page should already be ready. Page Cache accepts your framework as-is, supporting both server-side rendered and static origins, and adds pre-rendering or dynamic attributes only where they are beneficial. Either way, you get instant, current pages with minimal code change and a deployment plan that fits your roadmap.

Problem

Every product page starts its life as a template, then begs data systems for prices, inventory, promos, recommendations, and content fragments. Traditional “HTML + CDN” helps with assets, but whenever those dynamic bits are missing, the page slows. Teams layer on microservices, sprinkle in edge logic, and still chase cold starts, cache misses, and origin bottlenecks. You feel it as SEO decay, conversion drag, and operational sprawl. The customer feels it as wait time.

Page Cache can:

Pre-render pages for instant first paint and stronger crawlability.

Inject dynamic attributes without a round trip to origin.

Deliver globally from dedicated, predictable infrastructure.

Work push or pull: push rendered pages into Page Cache from your pipeline, or let Page Cache pull from your origin and materialize the final, cacheable result.

Scale simply as traffic, personalization, or catalog depth grows.

(Advanced) Host frameworks directly — unify modern app frameworks (e.g., React/Next) with Page Cache to minimize networking hops, simplify infrastructure, and cut costs.

Page Cache Solution

Page Cache enables full page load in as little as **600ms or less for 95% of users**. To accomplish this, we generate or accept a complete page (framework-rendered or prerendered), enrich it with dynamic attributes at the last responsible moment (price, availability, personalization, promotions), and cache the finished experience as a first-class artifact that can be served from the closest point to the user. Because Page Cache runs on Harper’s platform, with database, cache, application, and messaging systems in a single runtime, it avoids multi-server slowdowns and costly operational sprawl, common with alternative solutions.

harper

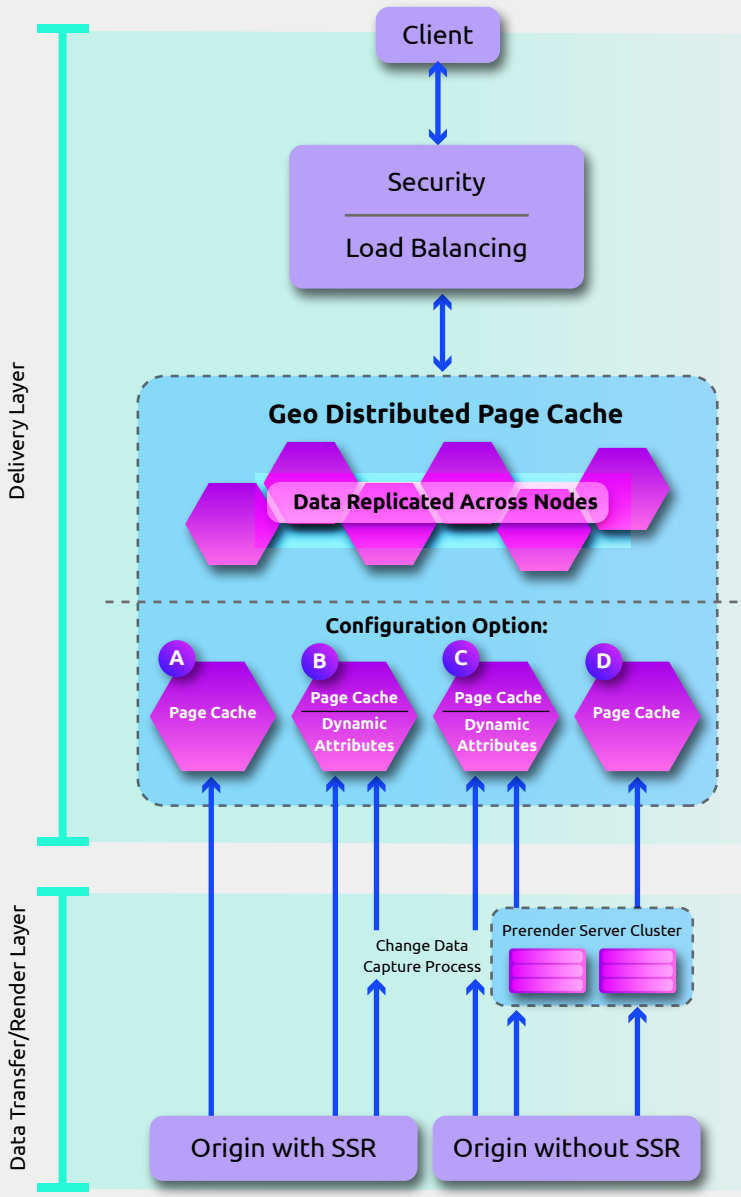
Why Page Cache Works

Page Cache works because it delivers the final, complete experience—not fragments. Pre-rendering eliminates assembly, attribute injection brings personalization into the delivery path, and Harper's global footprint removes the network drag that slows conversions. Fewer moving parts mean simpler operations, faster iteration, and customers who never wait for your architecture to catch up.

Adopt Page Cache all at once for maximum ROI on day one or plan it out over a few sprints. Either way, the outcome is the same: pages that are fully formed, always current, and instantly delivered.

Contact the Harper sales team at hello@harperdb.io to get started.

Page Cache Architecture with Deployment Options



Contact Sales at hello@harperdb.io.

harper