WEBSCALE

FEATURE BRIEF

Caching for Ecommerce with Webscale

Enhance user experience with intelligent caching

What is Caching?

Caching is the process of storing data in a temporary location known as a 'cache.' When a web page is requested by a user, the assets that form the web page can be retrieved from a cache closest to the end user, instead of the origin server or the hosting server. Cache reduces resource-intensive computation on the backend, while saving on network transfers. This shortens data access times, improves application performance with faster page loads, and delivers dramatically improved user experience.

Caching also frees up resources on the origin server, so they can be used to prioritize and serve requests for unique, dynamic content, like processing checkouts.

There are four types of caching that you should be aware of, if you manage web applications. Webscale uses a combination of all four to successfully serve ecommerce websites hosted in the cloud.

Full Page Cache

A full page cache minimizes code execution at the origin server. For example, in Magento applications, this cache is typically implemented using Redis or Varnish to reduce PHP execution. Webscale prefers Redis as it delivers the same performance benefits as Varnish without introducing unnecessary complexity to the environment. However, if application needs vary, we provide Varnish as well.

With full page cache enabled, when the request arrives at the web server, the web server invokes PHP, which checks the full page cache to see if, for the blocks that need to be built, there is cached HTML already in existence. If there is, it quickly pulls from the full page cache, avoiding the use of expensive compute resources to re-generate this block, speeding up delivery to the end user.

Webscale Data Plane Cache

The next layer of cache is in the Webscale Data Plane itself. The Webscale Data Plane functions as a reverse proxy, sitting in front of the web application infrastructure, and handling all traffic directed at the application. The data plane has a few different sub-caches built into it to further offload traffic from the infrastructure.

The first type is Dynamic Site Cache, which caches or combines a variety of static assets like JavaScript files, images, CSS, etc. These assets don't change very often, but there are many that make up a site, and each opens its own connection to the origin. Webscale keeps the number of open connections to the web servers down to a minimum, to scale infrastructure by delivering requests directly from the Webscale Data Plane. This eliminates round trips to the backend servers, which in turn lowers costs and delivers faster response times.

Most ecommerce pages are dynamic and personalized based on a user's shopping behavior, so they cannot be cached and re-used for other visitors. However, anonymous users, as well as bots, can receive cached HTML pages. Dynamic Site Cache delivers lightning fast page loads for users when they visit a storefront for the first few times, until they create an account, by allowing caching of HTML pages and content for all anonymous sessions.

Dynamic Site Cache also offloads the application origin from processing bot traffic, which can represent up to 50% of a typical ecommerce website's daily visitors. This significantly increases the efficiency of the application infrastructure, improving performance and reducing operating costs. Inside the Webscale Data Plane, there is another sub-cache, which is our implementation of page speed cache – an optimization layer built into the data plane, which optimizes the HTML code. This is not just a cache, it also performs optimizations by re-writing static assets, like images or javascript, that it references from a content delivery network (CDN)

CDN Cache

A CDN is a globally distributed collection of servers placed in close proximity to Internet users all over the world. While Webscale is not a CDN in itself, we are deeply integrated with leading CDNs and this functionality comes included with all Webscale products and plans. The Webscale technology stack provides the intelligence needed for performance, availability, security and scale at the origin, and the attached CDN provides the global network to get closer to the end customer. No additional customer integration is required to manage and run the CDN.

Alternatively, the Webscale CloudEDGE CDN comes with advanced features that enable Core Web Vitals improvements of up to 30 points. Webscale CloudEDGE CDN is available to modern commerce businesses of all sizes, whether they are already deployed on Webscale's SaaS platform or not.

CDNs focus on accelerating website performance by storing and delivering static content from a cache located geographically closer to the end user requesting the content. If the requested content does not exist in the edge cache, the request comes back to the origin through the Webscale Data Plane, site cache, or origin server. The required assets are then sent back to the edge, and from there to the end user's browser.

Browser Cache

The final layer of caching is actually in the browser itself. Webscale defines a maximum age or lifetime for content that loads in the browser (called the time to live or TTL), negating the need to make additional requests (to the origin or any other caching layer) for content that is still current and relevant (for which the TTL has not expired). This translates into zero latency and great application performance.

Across all these caching types, the required content will eventually cache once it is retrieved from the application backend and passed to the browser, even if the cache layer does not initially have the assets. All caches, by definition, store data temporarily and that data may be purged through inbuilt eviction policies or the cache object expiry itself.

Webscale allows application administrators to create and manage caching policies using cache control headers within the application code. These are enforced by the origin and the Webscale Data Plane. They can also control, customize, and override the cache control headers for web responses. An example of this would be changing or extending cache times for static assets, such as images that do not change frequently.