Most people are well aware of and leverage the high-performance reverse proxy capabilities of NGINX+, however, many don't know that you can also leverage it as a high-performance caching layer. In this article, we'll quickly touch on the high points of these capabilities as well as some configuration considerations and options you may not be aware of.

Enabling Proxy Caching

In order to take advantage of NGINX as a caching layer for proxied traffic, there are a few components that need to be ironed out before enabling caching to ensure a good experience :

1.) **Caching Path** – There needs to be a location on the disk defined for NGINX caching. This should be a high-performance storage medium such as an NVME SSD or other flash-backed storage with low latency and high throughput.

2.) **Caching Key** – A unique identifier should be used to determine what makes up a cacheable item; by default, NGINX uses the following key: `$scheme$proxy_host$request_uri` . For authenticated requests, this should be something that isolates the cache to a per-user model. For other requests, it should be unique enough to ensure that you do not end up with a "contaminated" cache where one item is cached in place of another item, such as dynamic content that could change per request.

3.) **Caching Minimum Uses** – By default NGINX will cache after a singular request (use), however, this could lead to the cache filling with infrequently requested items. In most environments, this will need to be tuned to align with the nature of the proxied application.

4.) **Cached Methods** – By default, NGINX will cache for HEAD and GET requests; however, in some environments, it could be useful to cache for other methods such as POST.

5.) **Cache Validity** – In some environments, it can be useful to define how long a cached item is valid before NGINX flushes or re-validates its cache to help ensure that stale data is not returned to clients.

6.) **Cache Bypass** – There are often select endpoints that should never be cached, such as login pages, or locations that may handle sensitive or one-time use responses. These can be selectively bypassed where appropriate.

As you can see in our simple example below, this will do some basic caching with a key that includes the end user’s session cookie. This is an example of a simple application that may not need many caching considerations but is not indicative of all caching configurations.

Example Simple Caching Configuration

```
proxy_cache_path /var/cache/nginx/mycache keys_zone=mycache:50m inacti
ve=60m;

...

...

...

    location /<SOME PREFIX HERE> {

        proxy_pass https://<my upstream>;

        proxy_set_header Host               $host;

        proxy_set_header X-Forwarded-Proto  $scheme;

        proxy_set_header X-Forwarded-For    $proxy_add_x_forwarded_fo
r;

        proxy_http_version                  1.1;

        proxy_cache_key "$scheme$host$request_uri $cookie_mysession-co
okie";

        proxy_cache mycache;

        proxy_cache_valid 1h;

}
```

One “hack” to improve caching

There are some “hacks” you can leverage on a Linux system to improve the caching performance of NGINX in certain environments. One of the simplest, assuming you have enough RAM available in your servers, is to create a RAMDISK mount using the tmpfs mount type. This is rather simple to do with a basic example shown below, however, care should be taken when using ramdisks for caching as you could exhaust server memory which would fall back to swap or have issues with the cache filling to capacity:

Example FSTAB Line for a 4g Ram Disk

```
tmpfs   /var/cache/nginx/ramcache         tmpfs   rw,nodev,nosuid,size
=4G          0  0
```

Example NGINX Config using a 4G RAM cache

```
proxy_cache_path /var/cache/nginx/ramcache keys_zone=mycache:50m inactive=60m;

...

...

...

    location /<SOME PREFIX HERE> {

        proxy_pass https://<my upstream>;

        proxy_set_header Host                 $host;

        proxy_set_header X-Forwarded-Proto  $scheme;

        proxy_set_header X-Forwarded-For    $proxy_add_x_forwarded_for;

        proxy_http_version                  1.1;

        proxy_cache_key "$scheme$host$request_uri $cookie_mysession-cookie";

        proxy_cache mycache;

        proxy_cache_valid 1h;

}
```