NGINX Conf 2018

The official event for all things NGINX

Best Practices for Caching

Kevin Jones

Global Product Specialist | Engineering | NGINX, Inc.

Agenda

1. NGINX and Caching, Why and How?
2. Fine Tuning the Cache
3. Tips and Tricks
4. Architecting for High Availability
5. Wrap-up

NGINX and Caching, Why and How?

Google Changed the Rules…

“We want you to be able to get from one page to another as quickly as you turn the page on a book.“ – Urz Holzle, Google

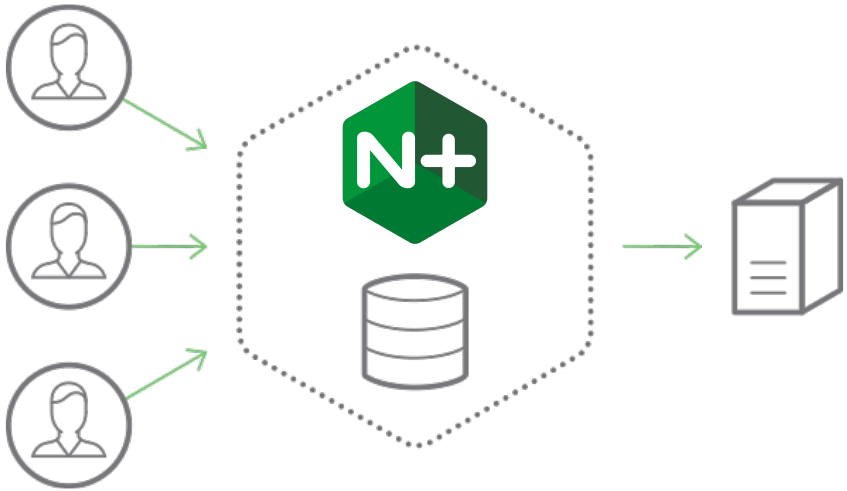
Why Cache?

- Performance
- Consolidate Infrastructure
- High Availability
- Lower Downtime

NGINX Can help…

It’s the cache that powers some of the largest CDNs:

- Cache static and dynamic content
- Improve dynamic content performance with microcaching
- Serve "stale" content while revalidating in background for better performance
- Override or set Cache-Control headers
- Manage the cache easily with the cache-purging API



NGINX perfect for both on-prem..

…And the Cloud

Netflix Cuts Costs and Reaches Over 50 million Subscribers in 40 Countries

NETFLIX

The Challenge

- *Big 3 CDN vendors (Akamai, Limelight, & Level 3) struggling to keep up with demand*
- *Rising costs*
- *Lack of control*

Solution

- *Create own CDN using NGINX, FreeBSD, and other OSS software*
- *14 TB 1RU appliance offered to all ISPs for free*

Results

- *Better experience for users*
- *Reduced costs with no CDN fees*
- *Reduced network traffic for partnering ISPs*

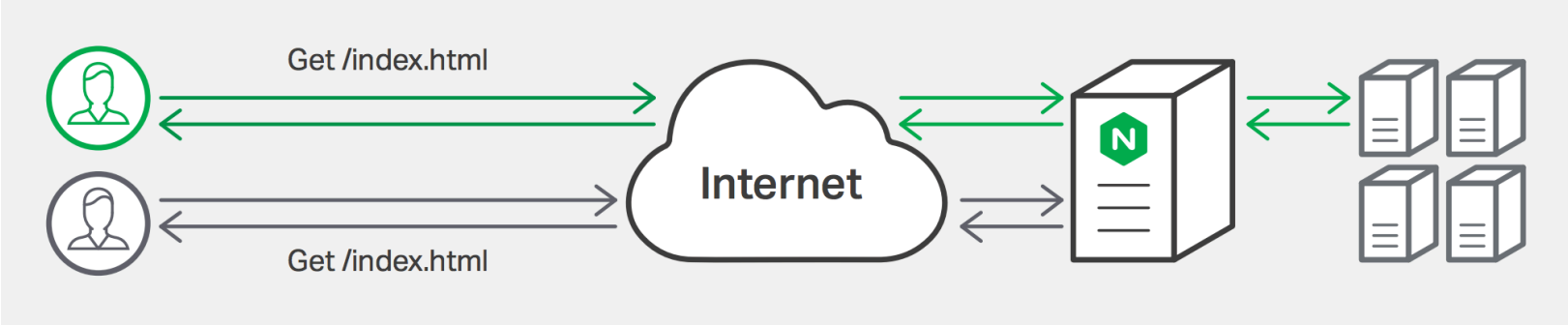
“The benefits we get from NGINX are speed and reliability. It’s our firewall, cache, and proxy. It fronts our APIs and our public facing Websites. Pretty much every request that goes to PBS.org or its subdomains runs through NGINX. Really the biggest reason we use NGINX is to mitigate thundering herd and improve our application performance.”

– Mike Howsden, Director of DevOps at PBS

The Basic Principle

1) HTTP Request:
GET /index.html

2) NGINX hashes the cache key, and checks if it already exists. If It does not or is expired, NGINX will send the request to the upstream server



3) Origin server responds

4) NGINX caches the response to disk, places the hash in memory and response is served to client

Content Caching with NGINX is Simple

proxy_cache_path

Definition: Sets the path and other parameters of a cache. Cache data are stored in files. The file name in a cache is a result of applying the MD5 function to the cache key.

```
http {

    proxy_cache_path /tmp/nginx/micro_cache/ keys_zone=large_cache:10m
    max_size=300g inactive=14d;
    ...
}
```

proxy_cache_key

Definition: Defines a key for caching. Used in the proxy_cache_path directive.

```
server {

    proxy_cache_key $scheme$proxy_host$request_uri;
    ...

}
```

proxy_cache

Definition: Defines a shared memory zone used for caching. The same zone can be used in several places.

```
location /video {
    ...
    proxy_cache large_cache;
}


location /images {
    ...
    proxy_cache small_cache;
}
```


proxy_cache_valid

Definition: Sets caching time for different response codes.

```
Location / {
    ...
    proxy_cache_valid 200 301 302 5m;
}

location ~* \.(jpg|png|gif|ico)$ {
    ...
    proxy_cache_valid any 1h;

}
```


```
http {

    proxy_cache_path /tmp/nginx/cache levels=1:2 keys_zone=cache:10m
                     max_size=100g inactive=7d use_temp_path=off;

    ...

    server {

        ...

        location / {
            ...
            proxy_pass http://backend.com;
        }

        location ^~ /images {
            ...
            proxy_cache cache;
            proxy_cache_valid 200 301 302 12h;
            proxy_pass http://images.origin.com;
        }

    }

}
```

Basic Caching

Additional Processes

```
# ps aux | grep nginx
root     14559  0.0  0.1  53308  3360 ?        Ss   Apr12   0:00 nginx: master process /usr/
sbin/nginx -c /etc/nginx/nginx.conf
nginx    27880  0.0  0.1  53692  2724 ?        S    00:06   0:00 nginx: worker process
nginx    27881  0.0  0.1  53692  2724 ?        S    00:06   0:00 nginx: worker process
nginx    27882  0.0  0.1  53472  2876 ?        S    00:06   0:00 nginx: cache manager process
nginx    27883  0.0  0.1  53472  2552 ?        S    00:06   0:00 nginx: cache loader process
```

- **Cache Manager** - activated periodically to check the state of the cache. If the cache size exceeds the limit set by the *max_size* parameter to the *proxy_cache_path* directive, the cache manager removes the data that was accessed least recently, as well as the cache considered inactive.

- **Cache Loader** - runs only once, right after NGINX starts. It loads metadata about previously cached data into the shared memory zone.

Cache Headers

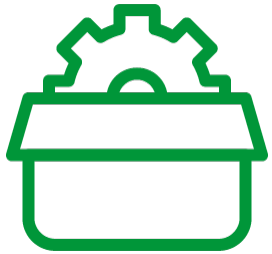
- **Cache-Control** - used to specify directives for caching mechanisms in both, requests and responses. (e.g. Cache-Control: no-cache or Cache-Control: private)
- **Expires** - contains the date/time after which the response is considered stale. If there is a Cache-Control header with the "max-age" or "s-max-age" directive in the response, the Expires header is ignored. (e.g. Expires: Wed, 21 Oct 2015 07:28:00 GMT)
- **Last-Modified** - contains the date and time at which the origin server believes the resource was last modified. HTTP dates are always expressed in GMT, never in local time. Less accurate than the ETag header (e.g. Last-Modified: Wed, 21 Oct 2015 07:28:00 GMT)
- **ETag** - is an identifier (or fingerprint) for a specific version of a resource. (e.g. ETag: “58efdcd0-268")

Override Cache-Control Headers

Tip: By default NGINX will honor all Cache-Control headers from the origin server, in turn not caching responses with Cache-Control set to Private, No-Cache, No-Store or with Set-Cookie in the response header.

Using proxy_ignore_headers you can disable processing of certain response header fields from the proxied server.

```
location ^~ /wordpress {
    ...
    proxy_cache cache;
    proxy_ignore_headers Cache-Control;
}
```

Fine Tuning the Cache

Types of Content

Static Content
- Images
- CSS
- Simple HTML

Dynamic Content
- Blog Posts
- Status
- API Data (Maybe?)

User Content
- Shopping Cart
- Unique Data
- Account Data

Easy to cache

Micro-cacheable!

Cannot Cache

```
http {

    upstream backend {
        keepalive 20;
        server 127.0.0.1:8080;
    }

    proxy_cache_path /var/nginx/micro_cache levels=1:2 keys_zone=micro_cache:10m
                     max_size=100m inactive=600s;

    ...

    server {

        listen 80;
        ...

        proxy_cache micro_cache;
        proxy_cache_valid any 1s;

        location / {
            proxy_http_version 1.1;
            proxy_set_header Connection "";
            proxy_set_header Accept-Encoding "";
            proxy_pass http://backend;
        }

    }

}
```

Enable keepalives on upstream

Set short inactive parameter

Set proxy_cache_valid to any status with a 1 second value

Set required HTTP version and pass HTTP headers for keepalives

Final Touches

proxy_cache_background_update

Definition: Allows starting a background subrequest to update an expired cache item, while a stale cached response is returned to the client. Note that it is necessary to allow the usage of a stale cached response when it is being updated.

```
location / {
    ...
    proxy_cache_background_update on;

}
```


Documentation

proxy_cache_lock

Definition: When enabled, only one request at a time will be allowed to populate a new cache element identified according to the proxy_cache_key directive by passing a request to a proxied server.

Other requests of the same cache element will either wait for a response to appear in the cache or the cache lock for this element to be released, up to the time set by the proxy_cache_lock_timeout directive.

Related: See the following for tuning…

- proxy_cache_lock_age,
- proxy_cache_lock_timeout

Documentation

proxy_cache_use_stale

Definition: Determines in which cases a stale cached response can be used during communication with the proxied server.

```
location /contact-us {
    ...
    proxy_cache_use_stale error timeout http_500 http_502 http_503 http_504;

}
```


Documentation

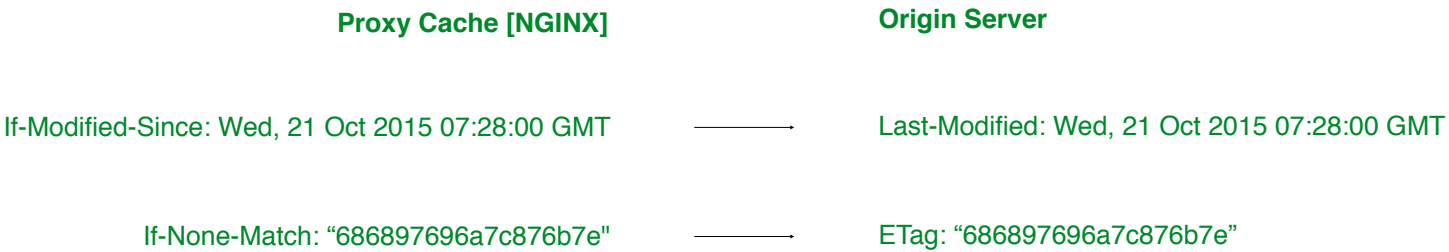
```
http {

    upstream backend {
        keepalive 20;
        server 127.0.0.1:8080;
    }

    proxy_cache_path /var/nginx/micro_cache levels=1:2 keys_zone=micro_cache:10m
                     max_size=100m inactive=600s;

    ...

    server {

        listen 80;
        ...

        proxy_cache micro_cache;
        proxy_cache_valid any 1s;
        proxy_cache_background_update on;
        proxy_cache_lock on;
        proxy_cache_use_stale updating;

        location / {
            ...
            proxy_http_version 1.1;
            proxy_set_header Connection "";
            proxy_set_header Accept-Encoding "";
            proxy_pass http://backend;
        }

    }

}
```

Final optimization

Further Tuning and Optimization

proxy_cache_revalidate

Definition: Enables revalidation of expired cache items using conditional GET requests with the “If-Modified-Since” and “If-None-Match” header fields.



Documentation

proxy_cache_min_uses

D**efinition:** Sets the number of requests after which the response will be cached. This will help with disk utilization and hit ratio of your cache.

```
location ~* /legacy {
   ...
    proxy_cache_min_uses 5;

}
```


Documentation

proxy_cache_methods

Definition: NGINX only caches GET and HEAD request methods by default. Using this directive you can add additional methods.

If you plan to add additional methods consider updating the cache key to include the $request_method variable if the response will be different depending on the request method.

```
location ~* /data {
    ...
    proxy_cache_methods GET HEAD POST;

}
```


Documentation

Tips and Tricks!

Logging is Your Friend

Tip: The more relevant information in your log the better. When troubleshooting you can easily add the proxy cache KEY to the log_format for debugging. For a list of all variables see the "Alphabetical index of variables" on nginx.org.

```
log_format main 'rid="$request_id" pck="$scheme://$proxy_host$request_uri" '
                'ucs="$upstream_cache_status" '
                'site="$server_name" server="$host" dest_port="$server_port" '
                'dest_ip="$server_addr" src="$remote_addr" src_ip="$realip_remote_addr" '
                'user="$remote_user" time_local="$time_local" protocol="$server_protocol" '
                'status="$status" bytes_out="$bytes_sent" '
                'bytes_in="$upstream_bytes_received" http_referer="$http_referer" '
                'http_user_agent="$http_user_agent" nginx_version="$nginx_version" '
                'http_x_forwarded_for="$http_x_forwarded_for" '
                'http_x_header="$http_x_header" uri_query="$query_string" uri_path="$uri" '
                'http_method="$request_method" response_time="$upstream_response_time" '
                'cookie="$http_cookie" request_time="$request_time" ';
```

Add Response Headers

Tip: Using the add_header directive you can add useful HTTP response headers allowing you to debug your NGINX deployment rather easily.

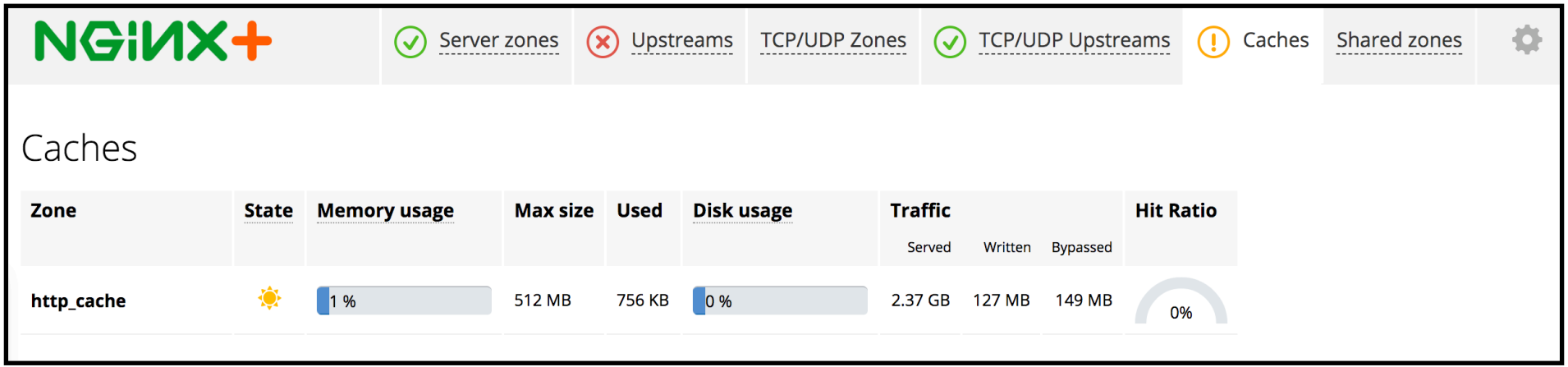
```
server {
    ...
    # add HTTP response headers
    add_header CC-X-Request-ID $request_id;
    add_header X-Cache-Status $upstream_cache_status;

}
```

Using cURL to Debug…

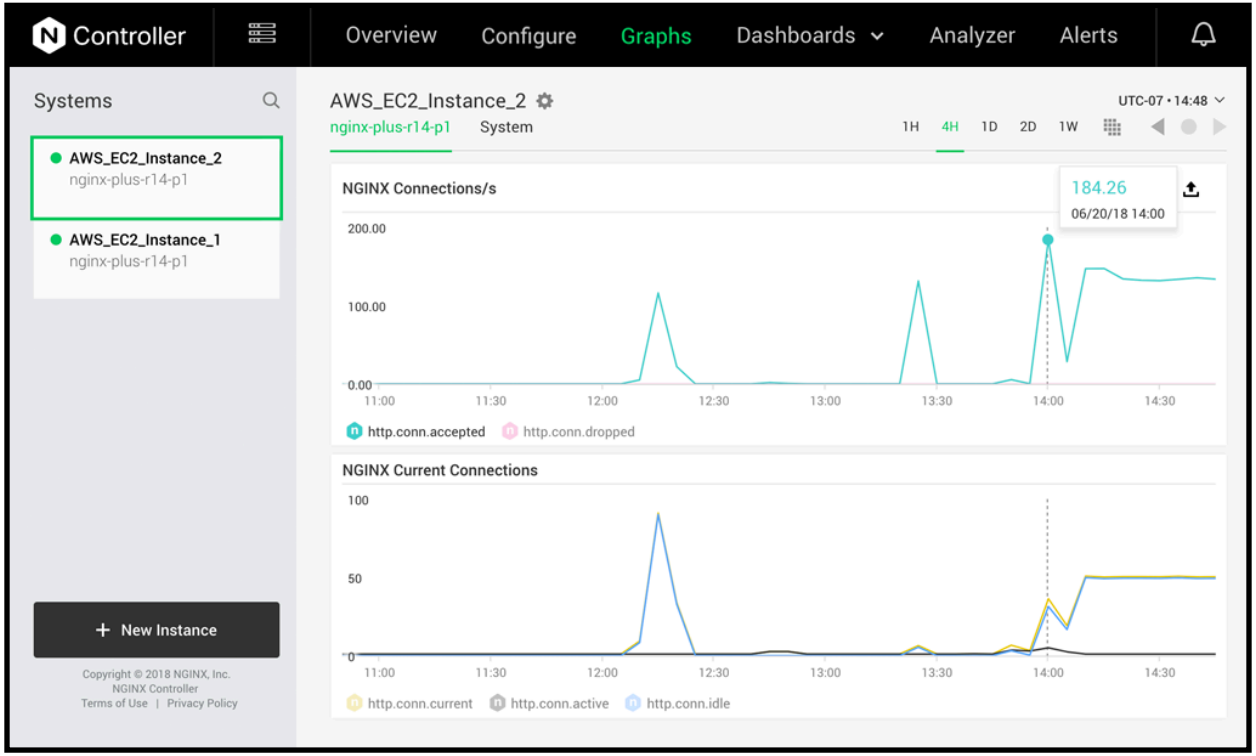
Tip: Use cURL or Chrome developer tools to grab the request ID or other various headers useful for debugging.

```
# curl -I 127.0.0.1/images/hawaii.jpg
HTTP/1.1 200 OK
Server: nginx/1.11.10
Date: Wed, 19 Apr 2017 22:20:53 GMT
Content-Type: image/jpeg
Content-Length: 21542868
Connection: keep-alive
Last-Modified: Thu, 13 Apr 2017 20:55:07 GMT
ETag: "58efe5ab-148b7d4"
OS-X-Request-ID: 1e7ae2cf83732e8859bc3e38df912ed1
CC-X-Request-ID: d4a5f7a8d25544b1409c351a22f42960
X-Cache-Status: HIT
Accept-Ranges: bytes
```

Use NGINX Plus for More Insight...



Consider NGINX Controller for Historical Data



NGINX Controller Metrics

Cache Metrics

- **nginx.cache.bypass**
- **nginx.cache.expired**
- **nginx.cache.hit**
- **nginx.cache.miss**
- **nginx.cache.revalidated**
- **nginx.cache.stale**
- **nginx.cache.updating**

```
Type:        counter, integer
Description: Various statistics about NGINX cache usage.
Source:      access.log (requires custom log format)
Variable:    $upstream_cache_status
```


NGINX Controller Metrics

Cache Zone Metrics

- **plus.cache.bypass**
- **plus.cache.bypass.bytes**
- **plus.cache.expired**
- **plus.cache.expired.bytes**
- **plus.cache.hit**
- **plus.cache.hit.bytes**
- **plus.cache.miss**
- **plus.cache.miss.bytes**
- **plus.cache.revalidated**
- **plus.cache.revalidated.bytes**
- **plus.cache.size**
- **plus.cache.stale**
- **plus.cache.stale.bytes**
- **plus.cache.updating**
- **plus.cache.updating.bytes**

```
Type:        counter, integer; counter, bytes
Description: Various statistics about NGINX Plus cache usage.
Source:      NGINX Plus status API
```


Can I Punch Through the Cache?

Tip: If you want to disregard the cache and go strait to the origin for a response, you can use the proxy_cache_bypass directive.

```
location / {
    ...
    proxy_cache cache;
    proxy_cache_bypass $cookie_nocache $arg_nocache $http_nocache;
}
```


proxy_cache_purge

Definition: Defines conditions under which the request will be considered a cache purge request. If at least one value of the string parameters is not empty and is not equal to “0” then the cache entry with a corresponding cache key is removed.

The result of successful operation is indicated by returning the 204 (No Content) response.

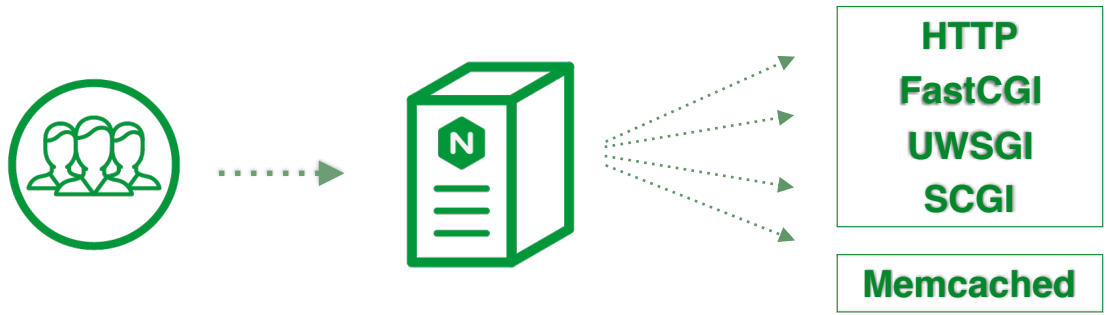
NGINX Plus only feature

Documentation

Caching is Not Just for HTTP

Tip: NGINX can also be used to cache other backends using their unique cache directives.
(e.g. fastcgi_cache, uwsgi_cache and scgi_cache)

Alternatively, NGINX can also be used to retrieve content directly from a memcached server.

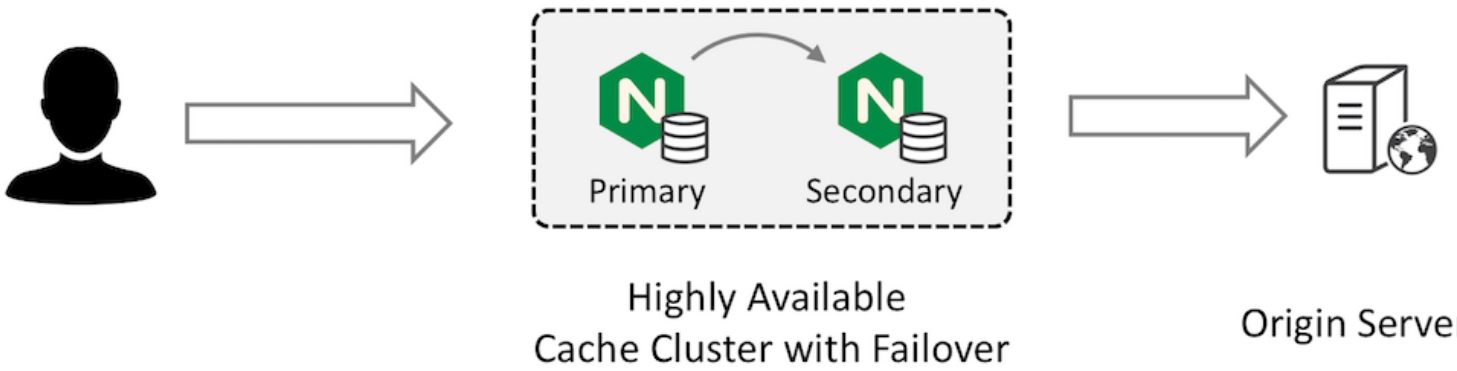


Architecting for High Availability

Two Approaches

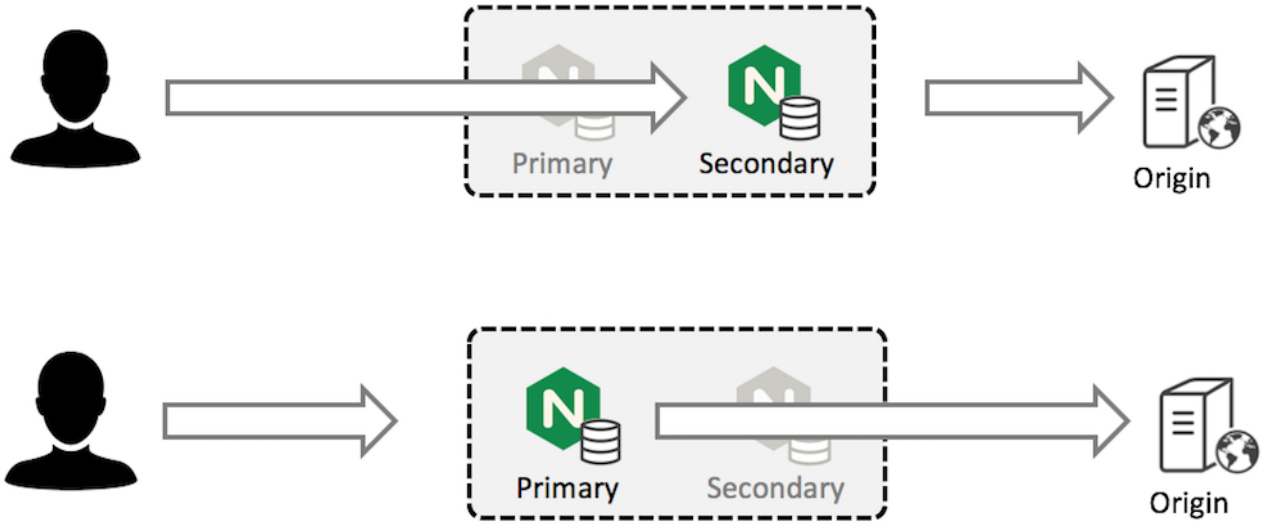
- Sharded (High Capacity)
- Shared (Replicated)

Shared Cache Clustering



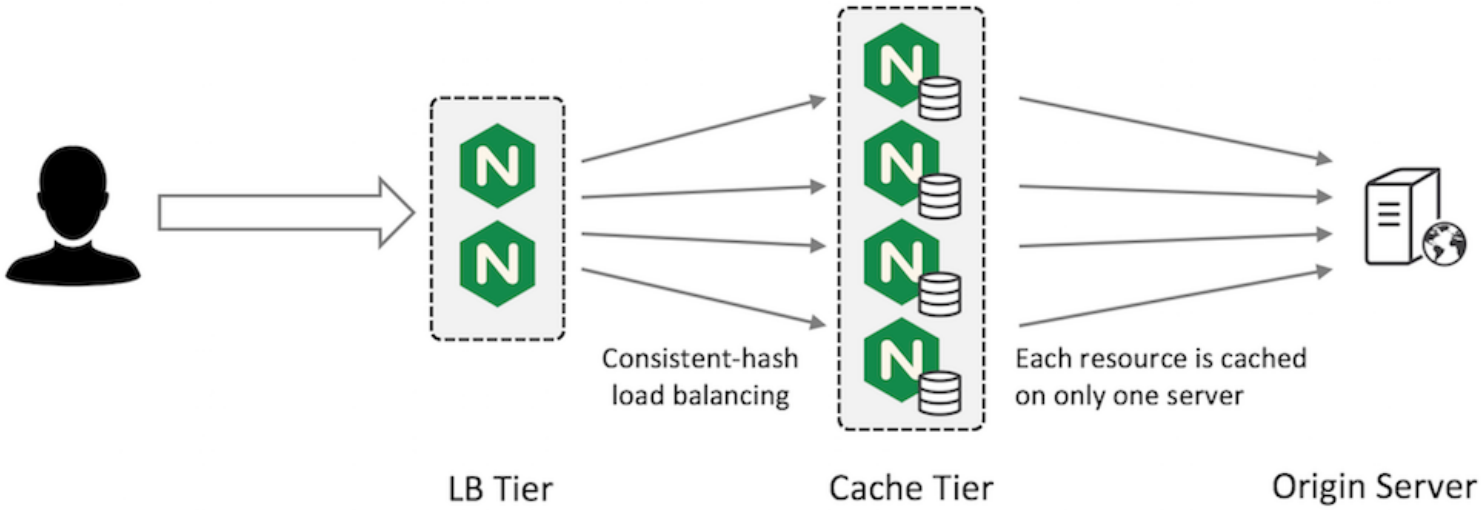
Tip: If your primary goal is to achieve high availability while minimizing load on the origin servers, this scenario provides a highly available shared cache. HA cluster should be Active/Passive configuration.

and Failover



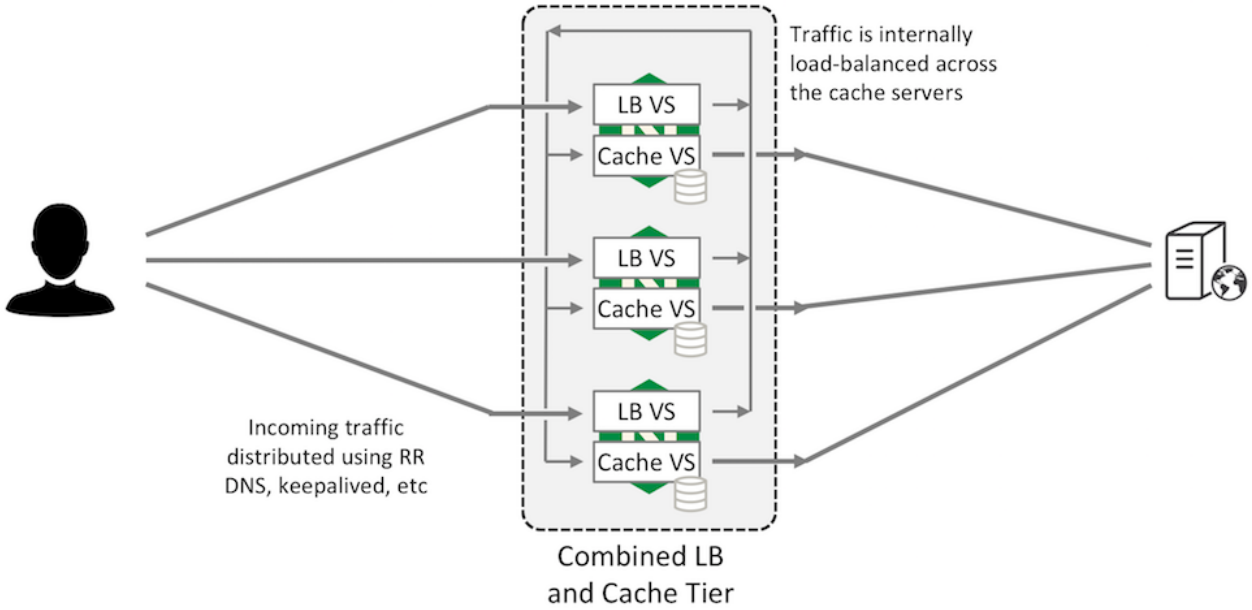
Tip: In the event of a failover there is no loss in cache and the origin does not suffer unneeded proxy requests.

Sharding Your Cache



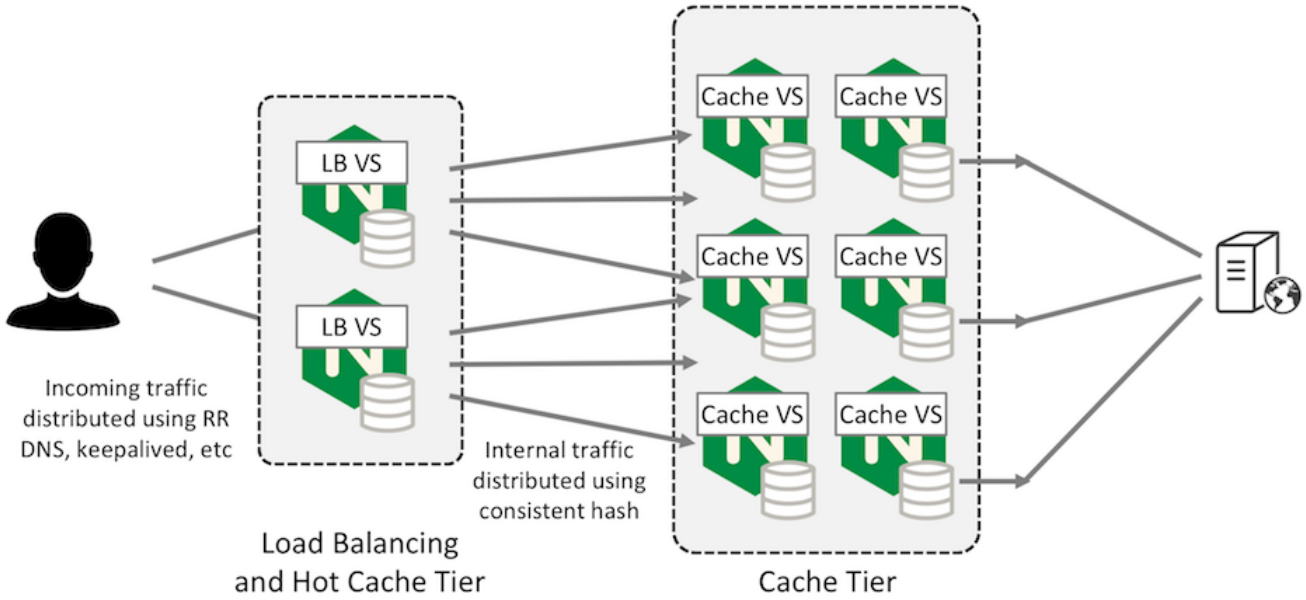
Tip: If your primary goal is to create a very high-capacity cache, shard (partition) your cache across multiple servers. This in turn maximizes the resources you have while minimizing impact on your origin servers depending on the amount of cache servers in your cache tier.

Combined Load Balancer and Cache



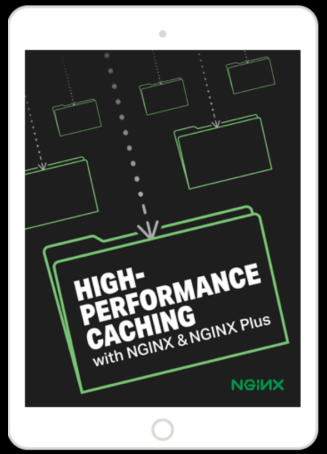
Tip: Alternatively, It is possible to consolidate the load balancer and cache tier into one with the use of a various NGINX directives and parameters.

Multi-Tier with “Hot Cache”



Tip: If needed, a “Hot Cache Tier” can be enabled on the load balancer layer which will give you the same high capacity cache and provide a high availability of specific cached resources.

EBOOK



High-Performance Caching with NGINX and NGINX Plus

Implement high-performance caching and cache clustering for your web applications with NGINX and NGINX Plus

DOWNLOAD FOR FREE

Thank you for coming!

Kevin Jones
kevin.jones@nginx.com

nginx.com | @nginxinc

Hash Load Balancing

Tip: Using the hash load balancing algorithm, we can specify the proxy cache key. This allows each resource to be cached on only one backend server.

```
upstream cache_servers {
    hash $scheme$proxy_host$request_uri consistent;

    server prod.cache1.host;
    server prod.cache2.host;
    server prod.cache3.host;
    server prod.cache4.host;
}
```

Example Cache Purge Configuration

Tip: Using NGINX Plus, you can issue unique request methods to invalidate the cache

```
proxy_cache_path /tmp/cache keys_zone=mycache:10m levels=1:2 inactive=60s;

map $request_method $purge_method {
    PURGE 1;
    default 0;
}

server {
    listen 80;
    server_name www.example.com;

    location / {
        proxy_pass http://localhost:8002;
        proxy_cache mycache;

        proxy_cache_purge $purge_method;
    }
}
```

dynamically set a variable

used later in the configuration

Primary Cache Server

```
proxy_cache_path /tmp/mycache keys_zone=mycache:10m;

server {

    listen 80;

    proxy_cache mycache;
    proxy_cache_valid 200 15s;

    location / {
        proxy_pass http://secondary;
    }

}

upstream secondary {
    server 192.168.56.11;          # secondary
    server 192.168.56.12 backup;   # origin
}
```

Secondary Cache Server

```
proxy_cache_path /tmp/mycache keys_zone=mycache:10m;

server {

    listen 80;

    proxy_cache mycache;
    proxy_cache_valid 200 15s;

    location / {
        proxy_pass http://origin;
    }
}

upstream origin {
    server 192.168.56.12;        # origin
}
```

Headlines go here. Two line maximum length.

- This is where bullet points about NGINX should go.
  - You can easily change this to a non-bulleted list by clicking on the "Bullets" icon in the "Home" ribbon.
    - Or you can apply a new, non-bulleted layout to this slide
      - One way to then make all the content on this page the same size is to use the "Indent Less" button on the subordinate bullet points.
- You can start a new series of bullets by clicking on the "Indent Less" button in the "Home" ribbon.

Headlines go here. Two line maximum length.

- This is where bullet points about NGINX go.
  - You can easily change this to a non-bulleted list by clicking on the "Bullets" icon in the "Home" ribbon.
    - Third level bullet
      - Fourth level bullet

- This is where bullet points about NGINX go.
  - You can easily change this to a non-bulleted list by clicking on the "Bullets" icon in the "Home" ribbon.
    - Third level bullet
      - Fourth level bullet

Headline goes here.

- This is where bullet points about NGINX go.
  - You can easily change this to a non-bulleted list by clicking on the "Bullets" icon in the "Home" ribbon.
    - Third level bullet
      - Fourth level bullet

Headline goes here.

```
Code code code code code
        Code code code code code
                Code code code code code

Code code code code code
        Code code code code code
                Code code code code code
```

Source | Notes

1. Use this page for source information
2. And/or notes
3. This text box is set-up to be two columns, and the text will wrap after each entry as shown here.
4. Four