

Производительность Webприложений. Кеширование

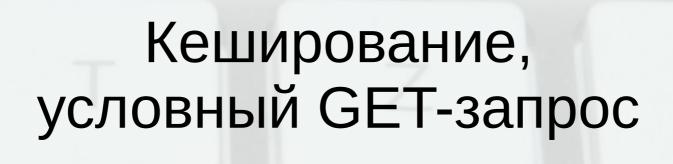
Сергей Геннадьевич Синица

КубГУ, 2020

sin@kubsu.ru

Пять уровней кеша

- 1. На клиенте. Условный GET-запрос.
- 2. На реверс-прокси. SSI/ESI.
- 3. В памяти на бекэнде. Локальные переменные. Memcached/Redis.
- 4. В базе данных или Solr/Elastic Search.
- 5. В операционной системе на сервере.



Кеширование – это сохрание промежуточных результатов вычисления где-либо.

Кеширование НТТР

По умолчанию клиент и прокси может:

- кешировать,
- использовать кеш если не устарел,
- но должен проверять актуальность кеша перед использованием, если он устарел.

Изменить это поведение можно заголовком ответа Cache-Control в HTTP/1.1 (Pragma и Expires в 1.0).

Cache-Control при запросе

no-store для прокси: кэшам не разрешено сохранять сообщение

no-transform для прокси: нельзя изменять тип данных ресурса

no-cache для прокси: отправлять запрос дальше исходному серверу

only-if-cached для прокси: требование ресурса только из кэша

max-age для прокси: возраст ответа должен быть не больше указанного значения (max-age=0 прокси ревалидируют кеш у сервера источника)

max-stale для прокси: возможен устаревший ответ, но не старше указанного значения

min-fresh для прокси: ответ должен быть актуальным по меньшей мере указанное время

Cache-Control при ответе

no-store для клиентов и прокси: кэшам не разрешено сохранять сообщение

no-transform для прокси: нельзя изменять тип данных ресурса

no-cache для клиентов и прокси: не обслуживать из кэша без проверки

актуальности

public для клиентов и прокси: разрешение кэшировать ответ, где угодно

private для прокси: не кэшировать в совместно используемых кэшах

must-revalidate для клиентов и прокси: возвращать только актуальные ответы

proxy-revalidate для прокси: возвращать только актуальные ответы

max-age для клиентов и прокси: срок актуальности ответа в секундах

s-maxage для прокси: срок актуальности ответа в совместно

используемых кэшах

Пример Cache-Control

Cache-Control: no-cache, must-revalidate

Expires: Sat, 26 Jul 1997 05:00:00 GMT

Cache-Control: max-age=3600, must-revalidate

Expires: Fri, 30 Mar 2014 14:19:41 GMT

Условный GET-запрос

Сервер может ответить 304 Not Modified и не передавать сущность. Клиент использует кеш.

Заголовки ответа:

Etag (entity tag)

Date (дата ответа сервера)

Last-Modified (дата изменения документа, используют поисковики)

Заголовки запроса:

If-modified-Since (отправлять строку Last-Modified из кеша, используют поисковики и отправляют "дату индексации", на сервере сравнивается на <=, но некоторые серверы сравнивают ==)
If-None-Match (etag)

Пример запроса

```
GET / HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0
 (X11; Linux x86_64; rv:14.0)
 Gecko/20100101 Firefox/14.0.1
Accept: text/html,
 application/xhtml+xml,
 application/xml;q=0.9,
 */*;q=0.8
Accept-Language: ru, en-us; q=0.7, en; q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
If-Modified-Since: Fri, 18 Feb 2011 20:37:42 GMT
If-None-Match: "43356-b1-49c947c5a8482"
Cache-Control: max-age=0
```

Пример ответа

HTTP/1.1 304 Not Modified

Date: Mon, 17 Sep 2012 10:50:41 GMT

Server: Apache/2.2.16 (Debian)

Connection: Keep-Alive

Keep-Alive: timeout=15, max=100
Etag: "43356-b1-49c947c5a8482"

Vary: Accept-Encoding

Валидаторы

Два типа валидаторов – дата изменения (Last-Modified) и теги (Etag, только в HTTP/1.1). Каждый может быть сильным или слабым.

Strong – валидатор меняется при любом изменении сущности. Weak – валидатор может не меняться при незначительных изменениях.

В НТТР/1.1 две функции сравнения валидаторов. Сильная – равны если оба сильные и идентичны полностью. Слабая – равны если идентичны полностью но один или оба могут быть слабыми.

Слабые валидаторы не могут быть использованы в Rangeзапросах. К

Last-Modified

Bpeмя Last-Modified когда используется как валидатор в запросе, не явно слаб, но становится сильным если применимы следующие правила:

- Валидатор сравнивается сервером источником непосредственно с валидатором сущности и,
- Сервер знает что сущность не могла измениться дважды в секунду.

или

- Валидатор будет использован клиентом в заголовке If-Modified-Since или If-Unmodified-Since, так как у клиента есть копия сушности в кеше, и
- В кеше есть Date со временем оригинального ответа сервера, и
- Полученное время Last-Modified как минимум 60 секунд до значения Date.

или

- Валидатор сравнивается промежуточным кешем для хранимой в нем сущности, и
- В кеше есть Date со временем оригинального ответа сервера, и
- Полученное время Last-Modified как минимум 60 секунд до значения Date.

Основано на том, что если сервер ответил два раза за секунду, то в последнем ответе Date и Last-Modified равны.

60 секунд защищают от разницы между Date и Last-Modified если они генерируются по разным часам или с отставанием.

Entity tag

Etag по умолчанию сильный, но сервер моджет сделать слабым через префикс:

Etag: \W"132324523453"

Слабый Etag может не меняться при незначительном изменении сущности без смены семантики ответа.

Например это делает Nginx при сжатии gzip.

В Etag нельзя писать ID сущности, нужно писать данные, которые поменяются при смене байтов сущности, например Unixtime изменения или хеш от сущности.

Vary

Заголовок ответа Vary показывает набор заголовков запроса, которые полностью определяют, пока ответ "свежий", может ли кеш использовать ответ при обработке последующих запросов без ревалидации.

Когда кеш получает последующий запрос и его URI указывает на запись в кеше, содержащие поля Vary, кеш НЕ ДОЛЖЕН (MUST NOT) использовать запись в ответе за исключением случая, когда все указанные в Vary оригинального ответа поля содержатся в кеше, присутствуют в запросе и полностью совпадают с передаваемым в запросе значением.

Важно для корректной работы кеширующих прокси. Vary: cookies позволяет корректно кешировать ответы с сессией на сервере.

Серверы источники НТТР 1/1

Предпочительно высылать сильный Etag и Last-Modified.

Получив условный запрос, включающий одновременно и дату Last-Modified (в заголовке If-Modified-Since или If-Unmodified-Since) и один или более тег сущности (заголовок If-Match, If-None-Match или If-Range) как валидаторы кеша, НЕ ДОЛЖНЫ возвращать (Not Modified) если нарушается целостность хотябы по одному условию в запросе.

Аналогично для кеширующих прокси.

Нужно учитывать, что HTTP/1.0 клиенты, боты Яндекса и Google, не поддерживают теги, но поддердивают Last-Modified!

Клиенты НТТР 1/1

- Если Etag выдан сервером ДОЛЖНЫ ИСПОЛЬЗОВАТЬ (MUST) его в условном запросе (If-Match или If-None-Match).
- Если только Last-Modified предоставлен сервером, ЖЕЛАТЕЛЬНО (SHOULD) использовать в условном запросе (If-Modified-Since).
- Если только Last-Modified дан сервером HTTP/1.0, могут использовать его в условных Range-запросах. Но должны дать возможность отключить эту возможность в случае сложностей.
- Если и тег и Last-Modified предоставлены сервером, то ЖЕЛАТЕЛЬНО (SHOULD) использовать оба в условных запросах. Это позволяет отвечать кешам и HTTP/1.0 и HTTP/1.1. Яндекс бот так не делает ("conditionally compilant" так как не выполяет все SHOULD).

Пример

```
function drupal_serve_page_from_cache(stdClass $cache) {
    // Negotiate whether to use compression.
    $page_compression = variable_get('page_compression', TRUE)
&& extension_loaded('zlib');
    $return_compressed = $page_compression &&
isset($_SERVER['HTTP_ACCEPT_ENCODING']) &&
strpos($_SERVER['HTTP_ACCEPT_ENCODING'], 'gzip') !== FALSE;
```

```
// Get headers set in hook_boot(). Keys are lower-case.
 $hook_boot_headers = drupal_get_http_header();
 // Headers generated in this function, that may be
replaced or unset using
 // drupal_add_http_headers(). Keys are mixed-case.
 $default_headers = array();
 foreach ($cache->data['headers'] as $name => $value) {
   // In the case of a 304 response, certain headers must
be sent, and the
   // remaining may not (see RFC 2616, section 10.3.5). Do
not override
   // headers set in hook_boot().
    $name_lower = strtolower($name);
    if (in_array($name_lower, array('content-location',
'expires', 'cache-control', 'vary')) &&!
isset($hook_boot_headers[$name_lower])) {
      drupal_add_http_header($name, $value);
      unset($cache->data['headers'][$name]);
```

```
// If the client sent a session cookie, a cached copy will
only be served
 // to that one particular client due to Vary: Cookie.
Thus, do not set
  // max-age > 0, allowing the page to be cached by external
proxies, when a
 // session cookie is present unless the Vary header has
been replaced or
 // unset in hook boot().
 $max_age = !isset($_COOKIE[session_name()]) ||
isset($hook_boot_headers['vary']) ?
variable_get('page_cache_maximum_age', 0) : 0;
 $default_headers['Cache-Control'] = 'public, max-age=' .
$max age;
 // Entity tag should change if the output changes.
 setag = '"' . scache->created . '-' .
intval($return_compressed) . '"';
 header('Etag: ' . $etag);
```

```
// See if the client has provided the required HTTP headers.
 $if_modified_since =
isset($_SERVER['HTTP_IF_MODIFIED_SINCE']) ?
strtotime($_SERVER['HTTP_IF_MODIFIED_SINCE']) : FALSE;
 $if_none_match = isset($_SERVER['HTTP_IF_NONE_MATCH']) ?
stripslashes($_SERVER['HTTP_IF_NONE_MATCH']) : FALSE;
  if ($if_modified_since && $if_none_match
     && $if_none_match == $etag // etag must match
     && $if_modified_since == $cache->created) { // if-
modified-since must match
    header($_SERVER['SERVER_PROTOCOL'] . ' 304 Not
Modified');
    drupal_send_headers($default_headers);
    return;
```

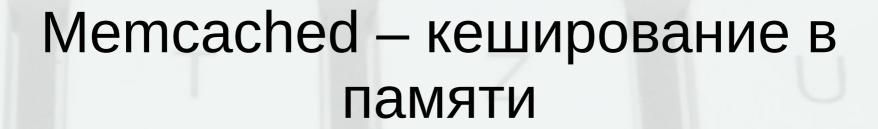
```
// Send the remaining headers.
  foreach ($cache->data['headers'] as $name => $value) {
    drupal_add_http_header($name, $value);
 $default_headers['Last-Modified'] = gmdate(DATE_RFC1123,
$cache->created);
 // HTTP/1.0 proxies does not support the Vary header, so
prevent any caching
  // by sending an Expires date in the past. HTTP/1.1
clients ignores the
 // Expires header if a Cache-Control: max-age= directive
is specified (see RFC
 // 2616, section 14.9.3).
 $default_headers['Expires'] = 'Sun, 19 Nov 1978 05:00:00
GMT';
 drupal_send_headers($default_headers);
```

```
// Allow HTTP proxies to cache pages for anonymous users
without a session
 // cookie. The Vary header is used to indicates the set of
request-header
  // fields that fully determines whether a cache is
permitted to use the
  // response to reply to a subsequent request for a given
URL without
 // revalidation. If a Vary header has been set in
hook_boot(), it is assumed
  // that the module knows how to cache the page.
  if (!isset($hook boot headers['vary']) && !
variable_get('omit_vary_cookie')) {
    header('Vary: Cookie');
```

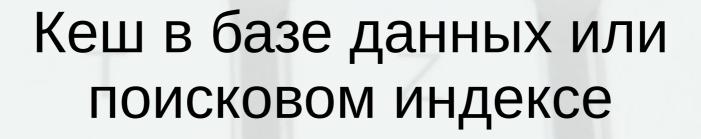
```
if ($page_compression) {
    header('Vary: Accept-Encoding', FALSE);
    // If page_compression is enabled, the cache contains
gzipped data.
    if ($return_compressed) {
      // $cache->data['body'] is already gzip'ed, so make
sure
     // zlib.output_compression does not compress it once
more.
      ini_set('zlib.output_compression', '0');
      header('Content-Encoding: gzip');
    else {
     // The client does not support compression, so unzip
the data in the
      // cache. Strip the gzip header and run uncompress.
      $cache->data['body'] = gzinflate(substr(substr($cache-
>data['body'], 10), 0, -8));
 // Print the page.
 print $cache->data['body'];
```



- 1. Общая схема проксирования.
- 2. Кеширование на примере NGINX.
- 2. Кеширование на примере Varnish.



- 1. Общая схема кеширования в памяти на бекенде
- 2. Кеширование на примере Drupal.



1. Настройка на примере MySQL. MySQLTuner, Releem.

2. Поисковые машины Solr и Elastic Search.

Литература

- 1. Интернет-программирование: учебное пособие / С.Г. Синица. Краснодар: КубГУ, 2013.
- 2. Спецификация НТТР/1.1.
- 3. Пример ускорения сайта в 30 раз с помощью организации кеширования и настройки нового сервера: https://drupal-coder.ru/blog/nastroyka-keshirovaniya-dannykh-uskorenie-raboty-sayta-v-30-raz