Cache for owncloud

Making ownCloud Faster Through Caching

July 15, 2015

Some operations an ownCloud server executes take a lot of time to complete, such as expensive calculations or communication with a remote storage server. Some other operations don’t take particularly long – but are needed many, many times per second. To improve performance and reduce the load on the system caused by expensive or frequently needed work, ownCloud can cache the result of these operations. In ownCloud 8.1, caching has become an explicit setting. Read on to learn more about caching in ownCloud and how to configure it so you get the best possible ownCloud experience.

Caching in PHP

Caching is used in PHP to store compiled versions of the scripts so they don’t need to be recompiled on every request. This is called opcaching and has been included and enabled by default in PHP since the 5.5 release. On PHP 5.4, the oldest currently supported PHP release which works with ownCloud, it is highly recommended to install an opcache. The most used implementation is called APC (see the Caching section in our Performance Tuning documentation). It is a webserver setting, meaning it won’t require you to configure anything in ownCloud.

Memory caching on the other hand is used directly by web applications like ownCloud. It helps ownCloud avoid slow database queries or file system checks by retrieving a result from a memory cache, either on the local machine or distributed on a cluster of servers. The result is a trade-off of memory usage for improved performance. As the memory usage of these caches is typically small, it is generally worth the effort to set them up.

In PHP, a handful of memory caches, or ‘memcaches,’ exist. A typical one available on nearly all distributions is APCu, or the Alternative PHP User Cache. It runs locally in the PHP process and can save information across requests. It is only accessible by the local machine however, so for a cluster a distributed memcache is needed.

Memcached is a popular distributed memcache which runs as a separate service and communicates with PHP via TCP port 11211. Redis is also a distributed memcache, very similar to Memcached and is available on TCP port 6379. Redis is much more advanced than other memcaches, with support for atomic operations, which makes it more suitable for ownCloud and is the recommended option.

The principle of caching – animated

For a distributed memcache, the communication port can be exposed to the network, allowing multiple servers access, further improving performance as cached results are available to all servers in the cluster.

How does memcaching work?

A memcache is a key-value store. An application can store a value under a key and can then retrieve that value by searching the memcache with the key. Values may also have a Time To Live (TTL) set so that values eventually expire from the memcache and need to be re-inserted by the application. This is useful as a sanity check – if we get something wrong with cache expunging, the TTL acts as a backup to recalculate the values at some point.

ownCloud Memcaches

Memcaches can be configured in config.php, under the ‘memcache.local’ and ‘memcache.distributed’ parameters. ‘memcache.distributed’ will default to the value of

‘memcache.local’ if unset, so a single-server installation can just set a local memcache which will be used for everything. Setting an explicit distributed memcache is only useful for multiple servers in a cluster where the data that can be shared between servers is stored in a cache that is available to all the servers.

- ownCloud supports the following memcaches:
- **APC** (for PHP 5.4 and under)
- **APCu** (for PHP 5.5 and above)
- **Redis** (for distributed systems)
- **Memcached** (not recommended with locking)
- **XCache** (local like APCu; not recommended)

Highly recommended are: APCu for all installations and Redis for multi-server installations in the distributed memcache configuration field. See the memchache configuration options in the sample config.php documentation to understand the parameters needed as well as for additional configuration requirements for certain memcache backends. memcached is not recommended as it does not guarantee the availability of stored values which doesn’t play well with the ownCloud locking mechanism. If you use the locking mechanism you can configure a separate cache for the locking (“memcache.locking”). Outside of the locking issue, memcached should work without problems.

For home users, the typical setup would be to add this line to config.php:

'memcache.local' => '\OC\Memcache\APCu',

But make sure that APCu is installed. Usually this package is named ‘php-apcu’ or ‘php5-apcu’. If you get a white screen upon visiting your ownCloud that means your cache configuration is broken! Make sure the memcache tool is installed and enabled correctly.

There is currently a **caveat with APCu** and the command line interface. You need to set the php.ini option apc.enable_cli = 1. See this issue for more details and the error you might get. We’re looking into this and will most likely provide a solution for 8.1.1, (coming out in about 3 weeks) which will make this no longer needed.

Set up your cache and enjoy a faster ownCloud!

Thanks to Robin ‘Xenopathic’ McCorkell for much of the write-up

3 Responses to “Making ownCloud Faster Through Caching”

1. ThFree

   July 16, 2015

   Thanks. We verify the performance with Caching .) Good luck in the works.

   Reply

2. **tony1**

   July 15, 2015

   If you are using php 5.4 in config.php you should use ‘memcache.local’ =>

Sonstiges

‘\OC\Memcache\APC’,
apc.enable_cli = 1 and apc.enabled=1 can be set in apc.ini or apcu.ini as well as other settings.
if you move /usr/share/doc/php5-apcu/apc.php to your web root you can see apc in action!!

take care

Quelle: https://owncloud.org/blog/making-owncloud-faster-through-caching/

Eindeutige ID: #1045
Verfasser: n/a
Letzte Änderung: 2015-07-16 23:15