Installing APCu Memcache for NextCloud 27.0 on Ubuntu 22.04

Reference:
https://docs.nextcloud.com/server/latest/admin_manual/configuration_server/caching_configuration.html

APCu is a data cache, and it is available in most Linux distributions. On Red Hat/CentOS/Fedora systems install php-pecl-apcu. On Debian/Ubuntu/Mint systems install php-apcu.

```
$ sudo apt install php-apcu
[sudo] password for user:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  php8.1-apcu
The following NEW packages will be installed:
  php-apcu php8.1-apcu
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 48.9 kB of archives.
After this operation, 197 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

```
sudo apt install php-apcu
[sudo] password for user:

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  php8.1-apcu
The following NEW packages will be installed:
  php-apcu php8.1-apcu
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 48.9 kB of archives.

Selecting previously unselected package php8.1-apcu.e will be used.
(Reading database ... 83934 files and directories currently installed.)
Preparing to unpack .../php8.1-apcu_5.1.21+4.0.11-7ubuntu1_amd64.deb ...
Unpacking php8.1-apcu (5.1.21+4.0.11-7ubuntu1) ...
Selecting previously unselected package php-apcu.
Preparing to unpack .../php-apcu_5.1.21+4.0.11-7ubuntu1_amd64.deb ...
Unpacking php-apcu (5.1.21+4.0.11-7ubuntu1) ...
Setting up php8.1-apcu (5.1.21+4.0.11-7ubuntu1) ...
Setting up php-apcu (5.1.21+4.0.11-7ubuntu1) ...
Processing triggers for libapache2-mod-php8.1 (8.1.2-1ubuntu2.11) ...
Processing triggers for php8.1-cli (8.1.2-1ubuntu2.11) ...
```

```
Scanning processes...
Scanning processor microcode...
Scanning linux images...
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 php8.1-apcu
amd64 5.1.21+4.0.11-7ubuntu1 [45.7 kB]
Running kernel seems to be up-to-date.ntu jammy/universe amd64 php-apcu
amd64 5.1.21+4.0.11-7ubuntu1 [3,132 B]
Fetched 48.9 kB in 0s (105 kB/s)
The processor microcode seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

Now, restart apache2 webserver

```
sudo systemctl restart apache2
```

After restarting your Web server, add this line to your NextCloud's config.php file:

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

Where to find your config.php file:

```
cd /var/www/[path to nextcloud installation]/nextcloud/config/

ls -l
```

Which lists the config.php file

```
sudo vim config.php
```

Edit the file with vim editor to insert the following line and save the change and exit Vim:

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

Refresh your Nextcloud admin page, and the cache warning should disappear. In other words, while logged into NextCloud as an 'admin' group user, access or refresh “Overview” in the “Administration” menu section and the warning message, that a memcache has not been installed, is gone. Well, it's gone if everything was done properly. There is no need at this point to restart the apache2 service in order to obtain this result.

Warning The NextCloud documentation states that –

APCu is disabled by default on CLI which could cause issues with nextcloud’s cron jobs. **Please make**

sure you set the apc.enable_cli to 1 on your php.ini config file or append –define apc.enable_cli=1 to the cron job call.

```
cd /etc/php/8.1/apache2/

ls


[which should list the 'php.ini' configuration file]
```

Use vim to edit the php.ini file, and insert the following content, and save the php.ini file, and then restart the apache2 service.

```
sudo vim php.ini
```

There is no section of the php.ini file named APCu. There is no line containing the setting 'apc.enable-cli=1. So, within vim, page down to the end of the php.ini file content and click the INS key in Vim to enter the Insert mode. Prese Enter key twice after that last line setting in the file in order to add a new line. On the new line, enter the following setting.

```
apc.enable-cli=1
```

Save the php.ini file with this edited content. Then, restart apache2 service.

```
sudo systemctl restart apache2
```

Temporarily place a phpinfo.php file in any website on the server, and then access it from a web browser and locate the APCu section setttings to ensure that “apc.enable-cli” is “On.” You should be done.

Check the administrative panel of NextCloud and see if any warning disappears.

There should still be a couple/few warnings in Administration / Overview, including the following:

“The database is used for transactional file locking. To enhance performance, please configure memcache, if available. See the documentation for more information.”

Although APCu is now configured as a memcache for data, we may still need to install and configure 'Redis' for file locking and distributed, but not local. Continue to use APCu for local memcache. So, configure both APCu and Redis to handle different things

Redis

Redis is an excellent modern memcache to use for distributed caching, and as a key-value store for Transactional File Locking because it guarantees that cached objects are available for as long as they are needed.

The Redis PHP module must be version 2.2.6+. If you are running a Linux distribution that does not package the supported versions of this module, or does not package Redis at all, see install_redis_label.

On Debian/Ubuntu/Mint install redis-server and php-redis. The installer will automatically launch redis-server and configure it to launch at startup.

On CentOS and Fedora install redis and php-pecl-redis. It will not start automatically, so you must use your service manager to start redis, and to launch it at boot as a daemon.

You can verify that the Redis daemon is running with ps ax:

```
ps ax | grep redis
22203 ? Ssl    0:00 /usr/bin/redis-server 127.0.0.1:6379
```

Restart your Web server, add the appropriate entries to your config.php, and refresh your Nextcloud admin page.

Redis configuration in Nextcloud (config.php)

For best performance, use Redis for file locking by adding this:

```
'memcache.locking' => '\OC\Memcache\Redis',
```

Additionally, you should use Redis for the distributed server cache:

```
'memcache.distributed' => '\OC\Memcache\Redis',
```

Further more, you could use Redis for the local cache like so, but it’s not recommended (see warning below).

Also, recall that we already installed and configure config.php for APCu as memcache.local. So, the following setting should probably be omitted or skipped. Memcache.local is already addressed by APCu setting.

```
'memcache.local' => '\OC\Memcache\Redis',
```

Warning

Using Redis for local cache on a multi-server setup can cause issues. Also, even on a single-server setup, APCu (see section above) should be faster.

Contents of config.sample.php with respect to MemCache Configuration

```
**
 * Memory caching backend configuration
 *
 * Available cache backends:
 *
 * * ``\OC\Memcache\APCu``       APC user backend
 * * ``\OC\Memcache\ArrayCache`` In-memory array-based backend (not
recommended)
 * * ``\OC\Memcache\Memcached``  Memcached backend
 * * ``\OC\Memcache\Redis``      Redis backend
 *
 * Advice on choosing between the various backends:
 *
 * * APCu should be easiest to install. Almost all distributions have
packages.
 *   Use this for single user environment for all caches.
 * * Use Redis or Memcached for distributed environments.
 *   For the local cache (you can configure two) take APCu.
 */

/**
 * Memory caching backend for locally stored data
 *
 * * Used for host-specific data, e.g. file paths
 *
 * Defaults to ``none``
 */
'memcache.local' => '\OC\Memcache\APCu',

/**
 * Memory caching backend for distributed data
 *
 * * Used for installation-specific data, e.g. database caching
 * * If unset, defaults to the value of memcache.local
 *
 * Defaults to ``none``
 */
'memcache.distributed' => '\OC\Memcache\Memcached',

/**
 * Connection details for redis to use for memory caching in a single server
configuration.
 *
 * For enhanced security it is recommended to configure Redis
 * to require a password. See http://redis.io/topics/security
 * for more information.
 *
 * We also support redis SSL/TLS encryption as of version 6.
 * See https://redis.io/topics/encryption for more information.
 */
'redis' => [
        'host' => 'localhost', // can also be a unix domain socket:
'/tmp/redis.sock'
        'port' => 6379,
        'timeout' => 0.0,
        'read_timeout' => 0.0,
        'user' =>  '', // Optional: if not defined, no password will be
used.
        'password' => '', // Optional: if not defined, no password will be
used.
        'dbindex' => 0, // Optional: if undefined SELECT will not run and
will use Redis Server's default DB Index.
        // If redis in-transit encryption is enabled, provide certificates
        // SSL context https://www.php.net/manual/en/context.ssl.php
        'ssl_context' => [
                'local_cert' => '/certs/redis.crt',
                'local_pk' => '/certs/redis.key',
                'cafile' => '/certs/ca.crt'
        ]
],

/**
 * Connection details for a Redis Cluster.
 *
 * Redis Cluster support requires the php module phpredis in version 3.0.0
or
 * higher.
 *
 * Available failover modes:
 *  - \RedisCluster::FAILOVER_NONE - only send commands to master nodes
(default)
 *  - \RedisCluster::FAILOVER_ERROR - failover to slaves for read commands
if master is unavailable (recommended)
 *  - \RedisCluster::FAILOVER_DISTRIBUTE - randomly distribute read commands
across master and slaves
 *
 * WARNING: FAILOVER_DISTRIBUTE is a not recommended setting, and we
strongly
 * suggest to not use it if you use Redis for file locking. Due to the way
Redis
 * is synchronized it could happen, that the read for an existing lock is
 * scheduled to a slave that is not fully synchronized with the connected
master
 * which then causes a FileLocked exception.
 *
 * See https://redis.io/topics/cluster-spec for details about the Redis
cluster
 *
 * Authentication works with phpredis version 4.2.1+. See
 *
https://github.com/phpredis/phpredis/commit/c5994f2a42b8a348af92d3acb4edff13
28ad8ce1
 */
'redis.cluster' => [
        'seeds' => [ // provide some or all of the cluster servers to
bootstrap discovery, port required
```

```
                'localhost:7000',
                'localhost:7001',
        ],
        'timeout' => 0.0,
        'read_timeout' => 0.0,
        'failover_mode' => \RedisCluster::FAILOVER_ERROR,
        'user' =>  '', // Optional: if not defined, no password will be
used.
        'password' => '', // Optional: if not defined, no password will be
used.
        // If redis in-transit encryption is enabled, provide certificates
        // SSL context https://www.php.net/manual/en/context.ssl.php
        'ssl_context' => [
                'local_cert' => '/certs/redis.crt',
                'local_pk' => '/certs/redis.key',
                'cafile' => '/certs/ca.crt'
        ]
],


/**
 * Server details for one or more memcached servers to use for memory
caching.
 */
'memcached_servers' => [
        // hostname, port and optional weight
        // or path and port 0 for unix socket. Also see:
        // https://www.php.net/manual/en/memcached.addservers.php
        // https://www.php.net/manual/en/memcached.addserver.php
        ['localhost', 11211],
        //array('other.host.local', 11211),
],

/**
 * Connection options for memcached
 */
'memcached_options' => [
        // Set timeouts to 50ms
        \Memcached::OPT_CONNECT_TIMEOUT => 50,
        \Memcached::OPT_RETRY_TIMEOUT =>   50,
        \Memcached::OPT_SEND_TIMEOUT =>    50,
        \Memcached::OPT_RECV_TIMEOUT =>    50,
        \Memcached::OPT_POLL_TIMEOUT =>    50,

        // Enable compression
        \Memcached::OPT_COMPRESSION =>          true,

        // Turn on consistent hashing
        \Memcached::OPT_LIBKETAMA_COMPATIBLE => true,

        // Enable Binary Protocol
        \Memcached::OPT_BINARY_PROTOCOL =>      true,

        // Binary serializer vill be enabled if the igbinary PECL module is
available
        //\Memcached::OPT_SERIALIZER => \Memcached::SERIALIZER_IGBINARY,
],



/**
 * Location of the cache folder, defaults to ``data/$user/cache`` where
 * ``$user`` is the current user. When specified, the format will change to
 * ``$cache_path/$user`` where ``$cache_path`` is the configured cache
directory
 * and ``$user`` is the user.
 *
 * Defaults to ``''`` (empty string)
 */
'cache_path' => '',

/**
 * TTL of chunks located in the cache folder before they're removed by
 * garbage collection (in seconds). Increase this value if users have
 * issues uploading very large files via the Nextcloud Client as upload
isn't
 * completed within one day.
 *
 * Defaults to ``60*60*24`` (1 day)
 */
'cache_chunk_gc_ttl' => 60*60*24,
```