Cache (BETA)

The `cache` remote wraps another existing remote and stores file structure and its data for long running tasks like `rclone mount`.

Status

The cache backend code is working but it currently doesn't have a maintainer so there are outstanding bugs which aren't getting fixed.

The cache backend is due to be phased out in favour of the VFS caching layer eventually which is more tightly integrated into rclone.

Until this happens we recommend only using the cache backend if you find you can't work without it. There are many docs online describing the use of the cache backend to minimize API hits and by-and-large these are out of date and the cache backend isn't needed in those scenarios any more.

Setup

To get started you just need to have an existing remote which can be configured with `cache`.

Here is an example of how to make a remote called `test-cache`. First run:

```
rclone config
```

This will guide you through an interactive setup process:

```
No remotes found - make a new one
n) New remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
n/r/c/s/q> n
name> test-cache
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Cache a remote
   \ "cache"
[snip]
Storage> cache
Remote to cache.
Normally should contain a ':' and a path, e.g. "myremote:path/to/dir",
"myremote:bucket" or maybe "myremote:" (not recommended).
remote> local:/test
Optional: The URL of the Plex server
plex_url> http://127.0.0.1:32400
Optional: The username of the Plex user
plex_username> dummyusername
Optional: The password of the Plex user
y) Yes type in my own password
g) Generate random password
n) No leave this optional password blank
y/g/n> y
Enter the password:
password:
Confirm the password:
password:
The size of a chunk. Lower value good for slow connections but can affect seamless reading.
Default: 5M
Choose a number from below, or type in your own value
 1 / 1MB
   \ "1m"
 2 / 5 MB
   \ "5M"
 3 / 10 MB
   \ "10M"
chunk_size> 2
How much time should object info (file size, file hashes, etc.) be stored in cache. Use a ver
Accepted units are: "s", "m", "h".
Default: 5m
Choose a number from below, or type in your own value
 1 / 1 hour
   \ "1h"
 2 / 24 hours
   \ "24h"
 3 / 24 hours
```

```
   \ "48h"
info_age> 2
The maximum size of stored chunks. When the storage grows beyond this size, the oldest chunks
Default: 10G
Choose a number from below, or type in your own value
 1 / 500 MB
   \ "500M"
 2 / 1 GB
   \ "1G"
 3 / 10 GB
   \ "10G"
chunk_total_size> 3
Remote config
--------------------
[test-cache]
remote = local:/test
plex_url = http://127.0.0.1:32400
plex_username = dummyusername
plex_password = *** ENCRYPTED ***
chunk_size = 5M
info_age = 48h
chunk_total_size = 10G
```

You can then use it like this,

List directories in top level of your drive

```
rclone lsd test-cache:
```

List all the files in your drive

```
rclone ls test-cache:
```

To start a cached mount

```
rclone mount --allow-other test-cache: /var/tmp/test-cache
```

Write Features

Offline uploading

In an effort to make writing through cache more reliable, the backend now supports this feature which can be activated by specifying a`cache-tmp-upload-path`.

A files goes through these states when using this feature:

1. An upload is started (usually by copying a file on the cache remote)

2. When the copy to the temporary location is complete the file is part of the cached remote and looks and behaves like any other file (reading included)
3. After `cache-tmp-wait-time` passes and the file is next in line, `rclone move` is used to move the file to the cloud provider
4. Reading the file still works during the upload but most modifications on it will be prohibited
5. Once the move is complete the file is unlocked for modifications as it becomes as any other regular file
6. If the file is being read through `cache` when it's actually deleted from the temporary path then `cache` will simply swap the source to the cloud provider without interrupting the reading (small blip can happen though)

Files are uploaded in sequence and only one file is uploaded at a time. Uploads will be stored in a queue and be processed based on the order they were added. The queue and the temporary storage is persistent across restarts but can be cleared on startup with the `--cache-db-purge` flag.

Write Support

Writes are supported through `cache`. One caveat is that a mounted cache remote does not add any retry or fallback mechanism to the upload operation. This will depend on the implementation of the wrapped remote. Consider using `Offline uploading` for reliable writes.

One special case is covered with `cache-writes` which will cache the file data at the same time as the upload when it is enabled making it available from the cache store immediately once the upload is finished.

Read Features

Multiple connections

To counter the high latency between a local PC where rclone is running and cloud providers, the cache remote can split multiple requests to the cloud provider for smaller file chunks and combines them together locally where they can be available almost immediately before the reader usually needs them.

This is similar to buffering when media files are played online. Rclone will stay around the current marker but always try its best to stay ahead and prepare the data before.

Plex Integration

There is a direct integration with Plex which allows cache to detect during reading if the file is in playback or not. This helps cache to adapt how it queries the cloud provider depending on what is needed for.

Scans will have a minimum amount of workers (1) while in a confirmed playback cache will deploy the configured number of workers.

This integration opens the doorway to additional performance improvements which will be explored in the near future.

Note: If Plex options are not configured, `cache` will function with its configured options without adapting any of its settings.

How to enable? Run `rclone config` and add all the Plex options (endpoint, username and password) in your remote and it will be automatically enabled.

Affected settings:

- `cache-workers`: *Configured value* during confirmed playback or *1* all the other times

Certificate Validation

When the Plex server is configured to only accept secure connections, it is possible to use `.plex.direct` URLs to ensure certificate validation succeeds. These URLs are used by Plex internally to connect to the Plex server securely.

The format for these URLs is the following:

https://ip-with-dots-replaced.server-hash.plex.direct:32400/

The `ip-with-dots-replaced` part can be any IPv4 address, where the dots have been replaced with dashes, e.g. `127.0.0.1` becomes `127-0-0-1`.

To get the `server-hash` part, the easiest way is to visit

https://plex.tv/api/resources?includeHttps=1&X-Plex-Token=your-plex-token

This page will list all the available Plex servers for your account with at least one `.plex.direct` link for each. Copy one URL and replace the IP address with the desired address. This can be used as the `plex_url` value.

Known issues

Mount and --dir-cache-time

--dir-cache-time controls the first layer of directory caching which works at the mount layer. Being an independent caching mechanism from the `cache` backend, it will manage its own entries based on the configured time.

To avoid getting in a scenario where dir cache has obsolete data and cache would have the correct one, try to set `--dir-cache-time` to a lower time than `--cache-info-age`. Default values are already configured in this way.

Windows support - Experimental

There are a couple of issues with Windows `mount` functionality that still require some investigations. It should be considered as experimental thus far as fixes come in for this OS.

Most of the issues seem to be related to the difference between filesystems on Linux flavors and Windows as cache is heavily dependent on them.

Any reports or feedback on how cache behaves on this OS is greatly appreciated.

- https://github.com/rclone/rclone/issues/1935
- https://github.com/rclone/rclone/issues/1907
- https://github.com/rclone/rclone/issues/1834

Risk of throttling

Future iterations of the cache backend will make use of the pooling functionality of the cloud provider to synchronize and at the same time make writing through it more tolerant to failures.

There are a couple of enhancements in track to add these but in the meantime there is a valid concern that the expiring cache listings can lead to cloud provider throttles or bans due to repeated queries on it for very large mounts.

Some recommendations:

- don't use a very small interval for entry information (`--cache-info-age`)
- while writes aren't yet optimised, you can still write through `cache` which gives you the advantage of adding the file in the cache at the same time if configured to do so.

Future enhancements:

- https://github.com/rclone/rclone/issues/1937
- https://github.com/rclone/rclone/issues/1936

cache and crypt

One common scenario is to keep your data encrypted in the cloud provider using the `crypt` remote. `crypt` uses a similar technique to wrap around an existing remote and handles this translation in a seamless way.

There is an issue with wrapping the remotes in this order: **cloud remote** -> **crypt** -> **cache**

During testing, I experienced a lot of bans with the remotes in this order. I suspect it might be related to how crypt opens files on the cloud provider which makes it think we're downloading the full file instead of small chunks. Organizing the remotes in this order yields better results: **cloud remote** -> **cache** -> **crypt**

absolute remote paths

`cache` can not differentiate between relative and absolute paths for the wrapped remote. Any path given in the `remote` config setting and on the command line will be passed to the wrapped remote as is, but for storing the chunks on disk the path will be made relative by removing any leading `/` character.

This behavior is irrelevant for most backend types, but there are backends where a leading `/` changes the effective directory, e.g. in the `sftp` backend paths starting with a `/` are relative to the root of the SSH server and paths without are relative to the user home directory. As a result `sftp:bin` and `sftp:/bin` will share the same cache folder, even if they represent a different directory on the SSH server.

Cache and Remote Control (--rc)

Cache supports the new `--rc` mode in rclone and can be remote controlled through the following end points: By default, the listener is disabled if you do not add the flag.

rc cache/expire

Purge a remote from the cache backend. Supports either a directory or a file. It supports both encrypted and unencrypted file names if cache is wrapped by crypt.

Params:

- **remote** = path to remote **(required)**
- **withData** = true/false to delete cached data (chunks) as well *(optional, false by default)*

Standard Options

Here are the standard options specific to cache (Cache a remote).

--cache-remote

Remote to cache. Normally should contain a ':' and a path, e.g. "myremote:path/to/dir", "myremote:bucket" or maybe "myremote:" (not recommended).

- Config: remote
- Env Var: RCLONE_CACHE_REMOTE
- Type: string
- Default: ""

--cache-plex-url

The URL of the Plex server

- Config: plex_url
- Env Var: RCLONE_CACHE_PLEX_URL
- Type: string
- Default: ""

--cache-plex-username

The username of the Plex user

- Config: plex_username
- Env Var: RCLONE_CACHE_PLEX_USERNAME
- Type: string
- Default: ""

--cache-plex-password

The password of the Plex user

NB Input to this must be obscured - see rclone obscure.

- Config: plex_password
- Env Var: RCLONE_CACHE_PLEX_PASSWORD
- Type: string
- Default: ""

--cache-chunk-size

The size of a chunk (partial file data).

Use lower numbers for slower connections. If the chunk size is changed, any downloaded chunks will be invalid and cache-chunk-path will need to be cleared or unexpected EOF errors will occur.

- Config: chunk_size
- Env Var: RCLONE_CACHE_CHUNK_SIZE
- Type: SizeSuffix
- Default: 5M
- Examples:
  - "1m"
    - 1MB
  - "5M"
    - 5 MB
  - "10M"
    - 10 MB

--cache-info-age

How long to cache file structure information (directory listings, file size, times, etc.). If all write operations are done through the cache then you can safely make this value very large as the cache store will also be updated in real time.

- Config: info_age
- Env Var: RCLONE_CACHE_INFO_AGE
- Type: Duration
- Default: 6h0m0s
- Examples:
  - "1h"
    - 1 hour
  - "24h"
    - 24 hours
  - "48h"
    - 48 hours

--cache-chunk-total-size

The total size that the chunks can take up on the local disk.

If the cache exceeds this value then it will start to delete the oldest chunks until it goes under this value.

- Config: chunk_total_size
- Env Var: RCLONE_CACHE_CHUNK_TOTAL_SIZE
- Type: SizeSuffix
- Default: 10G
- Examples:
  - "500M"
    - 500 MB
  - "1G"
    - 1 GB
  - "10G"
    - 10 GB

Advanced Options

Here are the advanced options specific to cache (Cache a remote).

--cache-plex-token

The plex token for authentication - auto set normally

- Config: plex_token
- Env Var: RCLONE_CACHE_PLEX_TOKEN
- Type: string
- Default: ""

--cache-plex-insecure

Skip all certificate verification when connecting to the Plex server

- Config: plex_insecure
- Env Var: RCLONE_CACHE_PLEX_INSECURE
- Type: string
- Default: ""

--cache-db-path

Directory to store file structure metadata DB. The remote name is used as the DB file name.

- Config: db_path
- Env Var: RCLONE_CACHE_DB_PATH
- Type: string
- Default: "$HOME/.cache/rclone/cache-backend"

--cache-chunk-path

Directory to cache chunk files.

Path to where partial file data (chunks) are stored locally. The remote name is appended to the final path.

This config follows the "--cache-db-path". If you specify a custom location for "--cache-db-path" and don't specify one for "--cache-chunk-path" then "--cache-chunk-path" will use the same path as "--cache-db-path".

- Config: chunk_path
- Env Var: RCLONE_CACHE_CHUNK_PATH
- Type: string
- Default: "$HOME/.cache/rclone/cache-backend"

--cache-db-purge

Clear all the cached data for this remote on start.

- Config: db_purge
- Env Var: RCLONE_CACHE_DB_PURGE
- Type: bool
- Default: false

--cache-chunk-clean-interval

How often should the cache perform cleanups of the chunk storage. The default value should be ok for most people. If you find that the cache goes over "cache-chunk-total-size" too often then try to lower this value to force it to perform cleanups more often.

- Config: chunk_clean_interval
- Env Var: RCLONE_CACHE_CHUNK_CLEAN_INTERVAL
- Type: Duration
- Default: 1m0s

--cache-read-retries

How many times to retry a read from a cache storage.

Since reading from a cache stream is independent from downloading file data, readers can get to a point where there's no more data in the cache. Most of the times this can indicate a connectivity issue if cache isn't able to provide file data anymore.

For really slow connections, increase this to a point where the stream is able to provide data but your experience will be very stuttering.

- Config: read_retries
- Env Var: RCLONE_CACHE_READ_RETRIES
- Type: int
- Default: 10

--cache-workers

How many workers should run in parallel to download chunks.

Higher values will mean more parallel processing (better CPU needed) and more concurrent requests on the cloud provider. This impacts several aspects like the cloud provider API limits, more stress on the hardware that rclone runs on but it also means that streams will be more fluid and data will be available much more faster to readers.

Note: If the optional Plex integration is enabled then this setting will adapt to the type of reading performed and the value specified here will be used as a maximum number of workers to use.

- Config: workers
- Env Var: RCLONE_CACHE_WORKERS
- Type: int
- Default: 4

--cache-chunk-no-memory

Disable the in-memory cache for storing chunks during streaming.

By default, cache will keep file data during streaming in RAM as well to provide it to readers as fast as possible.

This transient data is evicted as soon as it is read and the number of chunks stored doesn't exceed the number of workers. However, depending on other settings like "cache-chunk-size" and "cache-workers" this footprint can increase if there are parallel streams too (multiple files being read at the same time).

If the hardware permits it, use this feature to provide an overall better performance during streaming but it can also be disabled if RAM is not available on the local machine.

- Config: chunk_no_memory
- Env Var: RCLONE_CACHE_CHUNK_NO_MEMORY
- Type: bool
- Default: false

--cache-rps

Limits the number of requests per second to the source FS (-1 to disable)

This setting places a hard limit on the number of requests per second that cache will be doing to the cloud provider remote and try to respect that value by setting waits between reads.

If you find that you're getting banned or limited on the cloud provider through cache and know that a smaller number of requests per second will allow you to work with it then you can use this setting for that.

A good balance of all the other settings should make this setting useless but it is available to set for more special cases.

NOTE: This will limit the number of requests during streams but other API calls to the cloud provider like directory listings will still pass.

- Config: rps
- Env Var: RCLONE_CACHE_RPS
- Type: int
- Default: -1

--cache-writes

Cache file data on writes through the FS

If you need to read files immediately after you upload them through cache you can enable this flag to have their data stored in the cache store at the same time during upload.

- Config: writes
- Env Var: RCLONE_CACHE_WRITES
- Type: bool
- Default: false

--cache-tmp-upload-path

Directory to keep temporary files until they are uploaded.

This is the path where cache will use as a temporary storage for new files that need to be uploaded to the cloud provider.

Specifying a value will enable this feature. Without it, it is completely disabled and files will be uploaded directly to the cloud provider

- Config: tmp_upload_path
- Env Var: RCLONE_CACHE_TMP_UPLOAD_PATH
- Type: string
- Default: ""

--cache-tmp-wait-time

How long should files be stored in local cache before being uploaded

This is the duration that a file must wait in the temporary location *cache-tmp-upload-path* before it is selected for upload.

Note that only one file is uploaded at a time and it can take longer to start the upload if a queue formed for this purpose.

- Config: tmp_wait_time
- Env Var: RCLONE_CACHE_TMP_WAIT_TIME
- Type: Duration
- Default: 15s

--cache-db-wait-time

How long to wait for the DB to be available - 0 is unlimited

Only one process can have the DB open at any one time, so rclone waits for this duration for the DB to become available before it gives an error.

If you set it to 0 then it will wait forever.

- Config: db_wait_time
- Env Var: RCLONE_CACHE_DB_WAIT_TIME
- Type: Duration
- Default: 1s

Backend commands

Here are the commands specific to the cache backend.

Run them with

```
rclone backend COMMAND remote:
```

The help below will explain what arguments each command takes.

See the "rclone backend" command for more info on how to pass options and arguments.

These can be run on a running backend using the rc command backend/command.

stats

Print stats on the cache backend in JSON format.

```
rclone backend stats remote: [options] [<arguments>+]
```