Package: zaro (via r-universe)

May 29, 2026

Title Zarr via Arrow

Version 0.0.1.9010

Description Low-level Zarr interface (V2 and V3) built on Arrow filesystem and codec infrastructure. Provides direct access to Zarr stores on local filesystems, S3, and GCS via Arrow, with fallback to GDAL virtual filesystems for HTTP and other protocols via gdalraster. Supports Kerchunk JSON and Parquet reference stores for virtual Zarr access to existing archives. Automatically detects Zarr version and handles S3 region redirects.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports arrow, jsonlite, S7

Suggests gdalraster, blosc, future.apply, future.mirai, testthat (>= 3.0.0)

Config/testthat/edition 3

Collate 'utils.R' 'zaro-package.R' 'store.R' 'meta.R' 'codec.R' 'chunk.R' 'reference.R' 'print.R' 'api.R' 'virtualizarr.R'

Config/pak/sysreqs cmake libssl-dev

Repository https://hypertidy.r-universe.dev

Date/Publication 2026-05-29 06:49:43 UTC

RemoteUrl https://github.com/hypertidy/zaro

RemoteRef HEAD

RemoteSha 9ad40491493011727d1f72e930b248b49c51db89

Contents

`zaro` *Open a Zarr store*

Description

Connects to a Zarr store at the given path or URI. Dispatches to the appropriate backend based on the URI scheme:

- Local paths: Arrow LocalFileSystem
- `s3://`: Arrow S3FileSystem (with automatic region detection)
- `gs://`: Arrow GcsFileSystem
- `http://`, `https://`: gdalraster VSI (`/vsicurl/`)
- `reference+json://`: Kerchunk JSON reference store
- `reference+parquet://`: Kerchunk Parquet reference store
- `virtualizarr://`: VirtualiZarr Parquet manifest store

Usage

```
zaro(source, verbose = TRUE, ..., validate = FALSE)
```

Arguments

`source`	character. Path, URI, or reference store locator.
`verbose`	logical. Emit progress and diagnostic messages (default TRUE). Suppress with `suppressMessages()`.
`...`	additional arguments passed to Arrow filesystem constructors (e.g. `anonymous = TRUE`, `endpoint_override`, `region`).
`validate`	should the store be checked for valid metadata, `FALSE` is default

Details

For S3 stores, if Arrow returns a 301 redirect (wrong region), zaro will fall through to gdalraster's `/vsis3/` which handles region detection automatically.

Known public buckets (e.g. cmip6, aodn-cloud-optimised) are accessed anonymously by default. For other public stores, pass `anonymous = TRUE`.

Validation of the store may be done by setting `validate = TRUE`, but this is otherwise avoided at open time by default (`source` is checked for being character, non- empty string).

Value

A store object (internal class) for use with other zaro functions.

Examples

```
## Not run:
# local store
store <- zaro("/data/my_dataset.zarr")

# S3 store (anonymous)
store <- zaro("s3://mur-sst/zarr-v1", anonymous = TRUE)

# GCS — known public bucket, anonymous auto-detected
store <- zaro("gs://cmip6/CMIP6/ScenarioMIP/NOAA-GFDL/GFDL-ESM4/ssp585/r1i1p1f1/Omon/zos/gn/v20180701")

# VirtualiZarr Parquet reference
store <- zaro("virtualizarr://https://example.com/dataset.parq")

# then explore
zaro_list(store)
zaro_meta(store, "analysed_sst")
data <- zaro_read(store, "analysed_sst",
                  start = c(0, 0, 0), count = c(1, 100, 100))

## End(Not run)
```

---

zaro_chunk *Read a single chunk by grid index*

Description

Fetches and decodes one chunk, identified by its position in the chunk grid. Returns the decoded values with metadata about where the chunk sits in the array.

Usage

```
zaro_chunk(store, path, chunk_idx, meta = NULL, shaped = TRUE, verbose = TRUE)
```

Arguments

store	a store object from zaro()
path	character. Path to the array.
chunk_idx	integer vector. 0-based chunk grid index (e.g. c(0, 2, 3) for the first time step, third lat chunk, fourth lon chunk).
meta	optional pre-fetched ZaroMeta (or root group meta).
shaped	logical. If TRUE (default), reshape the values to the chunk's actual dimensions (handling C-order and edge chunks). If FALSE, return a flat vector.
verbose	logical.

Value

A list with components:

cidx integer vector — the chunk grid index

values array or vector — decoded chunk data

start integer vector — 0-based array coordinates of chunk origin

shape integer vector — actual shape of this chunk (may be smaller than chunk_shape for edge chunks)

Returns NULL if the chunk doesn't exist (missing/sparse).

`zaro_chunk_apply` *Apply a function over chunks*

Description

Iterates over chunks and applies a function to each, accumulating results. Chunks are fetched and decoded one at a time (or in parallel) so memory usage is proportional to one chunk, not the full array.

Usage

```
zaro_chunk_apply(
  store,
  path,
  fun,
  ranges = NULL,
  meta = NULL,
  shaped = TRUE,
  parallel = FALSE,
  verbose = TRUE
)
```

Arguments

`store`	a store object from `zaro()`
`path`	character. Path to the array.
`fun`	function. Called with a single chunk result list (as from `zaro_chunk()`). NULL chunks (missing) are skipped.
`ranges`	optional range specification (as in `zaro_chunks()`).
`meta`	optional pre-fetched ZaroMeta (or root group meta).
`shaped`	logical. If TRUE, reshape each chunk's values.
`parallel`	logical. Use future.apply for parallel execution.
`verbose`	logical.

Value

A list of results from `fun`, one per non-NULL chunk.

Examples

```
## Not run:
# per-chunk mean, memory-efficient
means <- zaro_chunk_apply(store, "temperature", function(chunk) {
  mean(chunk$values, na.rm = TRUE)
})

# first time step only, all spatial chunks
means <- zaro_chunk_apply(store, "temperature", function(chunk) {
  c(mean = mean(chunk$values, na.rm = TRUE),
    n_valid = sum(is.finite(chunk$values)))
}, ranges = list(0, NULL, NULL))

## End(Not run)
```

`zaro_chunk_info` *Get chunk grid information for an array*

Description

Returns the chunk grid dimensions and related metadata needed for chunk-indexed operations.

Usage

```
zaro_chunk_info(store, path, meta = NULL, verbose = TRUE)
```

Arguments

`store`	a store object from `zaro()`
`path`	character. Path to the array.
`meta`	optional pre-fetched ZaroMeta (or root group meta).
`verbose`	logical.

Value

A list with components:

grid integer vector — number of chunks along each dimension

chunk_shape integer vector — shape of a full (non-edge) chunk

array_shape integer vector — shape of the full array

n_chunks integer — total number of chunks

dimension_names character vector or NULL

meta ZaroMeta for the array

`zaro_chunks` *Read multiple chunks by index range*

Description

Fetches and decodes a set of chunks identified by ranges along each dimension of the chunk grid. Supports parallel fetching.

Usage

```
zaro_chunks(
  store,
  path,
  ranges = NULL,
  meta = NULL,
  shaped = TRUE,
  parallel = FALSE,
  verbose = TRUE
)
```

Arguments

`store`	a store object from `zaro()`
`path`	character. Path to the array.
`ranges`	a list of integer vectors, one per dimension, giving the 0-based chunk indices to read. Use `NULL` for "all chunks along this dimension". E.g. `list(0, NULL, 2:4)` reads chunk 0 along dim 1, all chunks along dim 2, chunks 2-4 along dim 3.
`meta`	optional pre-fetched ZaroMeta (or root group meta).
`shaped`	logical. If TRUE, reshape each chunk's values.
`parallel`	logical. Use future.apply for parallel fetch+decode.
`verbose`	logical.

Value

A list of chunk results (as from `zaro_chunk()`). NULL entries indicate missing chunks.

zaro_list *List contents of a Zarr store*

Description

List contents of a Zarr store

Usage

```
zaro_list(store, path = "", ...)
```

Arguments

store	a store object from zaro()
path	character. Path prefix within the store (default root).
...	arguments for methods (specifically 'recursive = FALSE' is default for 'ArrowStore')

Value

character vector of keys (relative paths).

zaro_meta *Get metadata for a Zarr node*

Description

Reads and parses the metadata for an array or group at the given path. Automatically detects Zarr V2 (.zarray/.zattrs/.zmetadata) vs V3 (zarr.json) format. For cloud stores, consolidated metadata (.zmetadata for V2, consolidated zarr.json for V3) is preferred to minimise the number of requests.

Usage

```
zaro_meta(store, path = "", consolidated = TRUE, verbose = TRUE)
```

Arguments

store	a store object from zaro()
path	character. Path to the array or group within the store. Use "" or "/" for the root group.
consolidated	logical. If TRUE, attempt to read consolidated metadata first (default).
verbose	logical. Emit diagnostic messages (default TRUE).

Value

A ZaroMeta object. For root group requests with consolidated metadata, the individual array metadata are attached as `attr(, "consolidated")`.

`zaro_read` *Read data from a Zarr array*

Description

Reads a hyperslab from a Zarr array, fetching and decoding the necessary chunks and assembling them into an R array. Supports both Zarr V2 and V3.

Usage

```
zaro_read(
  store,
  path = "",
  start = NULL,
  count = NULL,
  meta = NULL,
  parallel = FALSE,
  verbose = TRUE,
  assemble = TRUE
)
```

Arguments

`store`	a store object from `zaro()`
`path`	character. Path to the array within the store.
`start`	integer vector. 0-based start indices for each dimension. Defaults to the origin.
`count`	integer vector. Number of elements to read along each dimension. Use `NA` for any axis to read the full extent from `start`. Defaults to the full extent.
`meta`	optional pre-fetched ZaroMeta object. If NULL (default), metadata is read from the store. Can also be the root group metadata from `zaro_meta(store)` — the array will be looked up by path in the consolidated entries.
`parallel`	logical. If `TRUE`, fetch and decode chunks in parallel using `future.apply::future_lapply()`. Requires the `future.apply` package and a `future::plan()` to be set. Recommended backend: `future::plan(future.mirai::mirai_multisession)`. Skipped automatically for reads of fewer than 4 chunks. Default FALSE.
`verbose`	logical. Emit diagnostic messages (default TRUE).
`assemble`	logical. If `TRUE`return a full array (the default), else the raw list of decoded chunks.

Value

An R array with dimensions matching `count`.

Index