

Package: dasklyr (via r-universe)

October 16, 2026

Title R Interface to 'Dask'

Version 0.1.2

Description Run 'dplyr' verbs on 'Dask' data frames from R. Connect to a local 'Dask' client, a remote scheduler, or an optional 'Coiled' cluster, copy R data frames onto the cluster, read and write 'Parquet' and CSV, and collect results as 'tibble' objects. Computation stays in 'Python' workers through 'reticulate'; R only builds the graph. A 'Python' installation with the 'dask' package is required.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Depends R (>= 4.1)

Imports dplyr, rlang, tibble, tidyselect, reticulate (>= 1.38)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

SystemRequirements 'Python' with the 'dask' (dataframe, distributed), 'pandas', and 'pyarrow' packages

URL <https://github.com/hebian1994/dasklyr>

BugReports <https://github.com/hebian1994/dasklyr/issues>

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libpng-dev python3

Repository <https://hebian1994.r-universe.dev>

Date/Publication 2026-10-16 14:18:28 UTC

RemoteUrl <https://github.com/hebian1994/dasklyr>

RemoteRef HEAD

RemoteSha 8d94a30da086701223aad51c7419bd601753ca84

Contents

copy_to.dask_connection	2
dask_available	3
dask_coiled_available	4
dask_coiled_clusters	4
dask_config	5
dask_connect	6
dask_connection_find	7
dask_connection_is_open	8
dask_dataframe	8
dask_disconnect	9
dask_disconnect_all	10
dask_invoke	10
dask_read_parquet	11
dask_web	12
dask_write_parquet	13
ddf_schema	14
tbl.dask_connection	15
tbl_dask	16
Index	18

copy_to.dask_connection
<i>Copy an R data frame to Dask</i>

Description

dplyr `dplyr::copy_to()` method for dask_connection. Counterpart to sparklyr's `copy_to.spark_connection`. There is no `dask_copy_to()`; use the dplyr generic, same as sparklyr.

Usage

```
## S3 method for class 'dask_connection'
copy_to(
  dest,
  df,
  name = deparse(substitute(df)),
  overwrite = FALSE,
  memory = FALSE,
  repartition = 1L,
  ...
)
```

Arguments

dest	A dask_connection.
df	A data.frame.
name	Table name in the registry.
overwrite	If TRUE, replace an existing name.
memory	If TRUE, persist() the frame after upload.
repartition	Number of Dask partitions. Same argument name as sparklyr (not npartitions).
...	Unused.

Details

Named tables live in a client-side registry (sc\$tables), not a Hive catalog. Retrieve them with `dplyr::tbl()`.

Value

A tbl_dask.

Examples

```
## Not run:
sc <- dask_connect()
iris_tbl <- copy_to(sc, iris, "iris", overwrite = TRUE)

## End(Not run)
```

dask_available	<i>Is Dask available to this R session?</i>
----------------	---

Description

Checks that reticulate can start Python and import dask.dataframe. Honors DASKLYR_PYTHON if set.

Usage

```
dask_available()
```

Value

TRUE or FALSE.

Examples

```
## Not run:
dask_available()

## End(Not run)
```

dask_coiled_available *Is the Python coiled package available?*

Description

Checks that reticulate can import coiled. Honors DASKLYR_PYTHON if set. Does not start a cluster.

Usage

```
dask_coiled_available()
```

Value

TRUE or FALSE.

Examples

```
## Not run:
dask_coiled_available()
# pip install coiled && coiled login (same Python as DASKLYR_PYTHON)

## End(Not run)
```

dask_coiled_clusters *List Coiled clusters for this account*

Description

Thin wrapper around `coiled.list_clusters()`. Requires the Python coiled package and a token from `coiled login` or `DASK_COILED__TOKEN`.

Usage

```
dask_coiled_clusters()
```

Value

A tibble with id, name, and state.

Examples

```
## Not run:
dask_coiled_clusters()
sc <- dask_connect(
  master = "coiled",
  cluster_name = Sys.getenv("DASKLYR_COILED_NAME"),
  n_workers = 1L,
  shutdown_on_close = FALSE
)
dask_disconnect(sc)

## End(Not run)
```

dask_config

Runtime configuration for dasklyr

Description

Counterpart to sparklyr's `spark_config()`. Values are stored on the connection object returned by [dask_connect\(\)](#).

Usage

```
dask_config(...)
```

Arguments

... Named options. Known keys:

- `timeout`: seconds to wait when connecting to a remote scheduler.
- `processes`: local Client processes flag (default FALSE).
- `n_workers`: Coiled worker count when creating a cluster (default 1).
- `shutdown_on_close`: Coiled cluster teardown on [dask_disconnect\(\)](#).

Value

A named list.

Examples

```
dask_config(timeout = 30)
```

dask_connect

Connect to Dask

Description

Counterpart to sparklyr's `spark_connect(master = ...)`. With no arguments, starts a **local** client. `DASK_SCHEDULER` is not read here; pass `master` (or `dask_connect(Sys.getenv("DASK_SCHEDULER"))`) for a cluster. `master = "coiled"` uses Python `coiled.Cluster` (token + TLS); do not pass a Coiled `tcp://` or `tls://` address here.

Usage

```
dask_connect(
  master = "local",
  app_name = "dasklyr",
  config = dask_config(),
  timeout = NULL,
  processes = NULL,
  scheduler = NULL,
  cluster_name = NULL,
  n_workers = 1L,
  shutdown_on_close = FALSE,
  ...
)
```

Arguments

<code>master</code>	"local" starts a local client. A scheduler address such as "tcp://hostname:8786" connects to a self-managed cluster. "coiled" opens or attaches a Coiled cluster.
<code>app_name</code>	Application name used when reusing local / tcp:// connections. Not the Coiled cluster name.
<code>config</code>	A list from <code>dask_config()</code> .
<code>timeout</code>	Seconds to wait when connecting to a remote scheduler.
<code>processes</code>	Passed to <code>dask.distributed.Client</code> for local masters. Default FALSE avoids Windows process-spawn issues.
<code>scheduler</code>	Alias for <code>master</code> .
<code>cluster_name</code>	Coiled cluster name (<code>coiled.Cluster(name=)</code>). Required when <code>master = "coiled"</code> .
<code>n_workers</code>	Worker count passed to Coiled only when creating a cluster (not when attaching to a ready one). Default 1 so tests and demos stay on 1 scheduler + 1 worker.
<code>shutdown_on_close</code>	If TRUE, <code>dask_disconnect()</code> shuts down the Coiled cluster. Default FALSE leaves the VMs so the same <code>cluster_name</code> can be reused.
<code>...</code>	Unused; present for sparklyr-like call compatibility.

Details

Local / tcp:// reuse the same master + app_name pair. Coiled reuses the same cluster_name in this R session, and attaches to a live cluster of that name on Coiled.

Value

A dask_connection.

Examples

```
## Not run:
sc <- dask_connect(master = "local")
sc <- dask_connect(master = Sys.getenv("DASK_SCHEDULER"))
sc <- dask_connect(
  master = "coiled",
  cluster_name = "demo-keep",
  n_workers = 1L,
  shutdown_on_close = FALSE
)

## End(Not run)
```

dask_connection_find *Find an open Dask connection*

Description

Counterpart to sparklyr's spark_connection_find().

Usage

```
dask_connection_find(master = NULL, app_name = NULL, cluster_name = NULL)
```

Arguments

master	Scheduler address, "local", or "coiled". NULL matches any.
app_name	Application name used for local / tcp:// reuse. NULL matches any.
cluster_name	Coiled cluster name. NULL matches any.

Value

A list of matching dask_connection objects (possibly empty).

Examples

```
dask_connection_find()
```

`dask_connection_is_open`*Is a Dask connection still usable?*

Description

Is a Dask connection still usable?

Usage

```
dask_connection_is_open(sc)
```

Arguments

`sc` A `dask_connection`.

Value

TRUE or FALSE.

Examples

```
dask_connection_is_open(NULL)
```

`dask_dataframe`*Underlying Dask DataFrame*

Description

Counterpart to sparklyr's `spark_dataframe()`. Returns the Python object so you can call APIs that dasklyr has not wrapped.

Usage

```
dask_dataframe(x)
```

Arguments

`x` A `tbl_dask`.

Value

A reticulate Python Dask DataFrame.

Examples

```
## Not run:
sc <- dask_connect()
st <- copy_to(sc, mtcars, overwrite = TRUE)
dask_dataframe(st)

## End(Not run)
```

dask_disconnect	<i>Disconnect from Dask</i>
-----------------	-----------------------------

Description

Counterpart to sparklyr's `spark_disconnect()`.

Usage

```
dask_disconnect(sc, ...)
```

Arguments

sc	A dask_connection.
...	Unused.

Value

sc, invisibly.

Examples

```
## Not run:
sc <- dask_connect()
dask_disconnect(sc)

## End(Not run)
```

dask_disconnect_all	<i>Disconnect every open Dask connection</i>
---------------------	--

Description

Counterpart to sparklyr's `spark_disconnect_all()`.

Usage

```
dask_disconnect_all(...)
```

Arguments

... Passed to `dask_disconnect()`.

Value

The number of connections closed.

Examples

```
dask_disconnect_all()
```

dask_invoke	<i>Invoke a Python method</i>
-------------	-------------------------------

Description

Counterpart to sparklyr's `invoke()`: call a method on a Dask DataFrame or on the distributed Client. If the result looks like a Dask DataFrame it is wrapped as a `tbl_dask`.

Usage

```
dask_invoke(x, method, ...)
```

```
## S3 method for class 'tbl_dask'
dask_invoke(x, method, ...)
```

```
## S3 method for class 'dask_connection'
dask_invoke(x, method, ...)
```

Arguments

x	A <code>tbl_dask</code> or <code>dask_connection</code> .
method	Python method name.
...	Arguments forwarded to the Python method.

Value

A `tbl_dask` if the Python result looks like a Dask DataFrame; otherwise the value returned by `reticulate`.

Examples

```
## Not run:
sc <- dask_connect()
st <- copy_to(sc, mtcars, overwrite = TRUE)
dask_invoke(st, "npartitions")
dask_invoke(st, "repartition", npartitions = 2L)
dask_invoke(sc, "scheduler_info")

## End(Not run)
```

dask_read_parquet	<i>Read CSV or Parquet into a Dask table</i>
-------------------	--

Description

Counterpart to sparklyr's `spark_read_csv()` / `spark_read_parquet()`. Signature is `(sc, name, path)`: `name` may be omitted and is then derived from the `path`.

Usage

```
dask_read_parquet(
  sc,
  name = NULL,
  path = name,
  memory = FALSE,
  overwrite = TRUE,
  ...
)

dask_read_csv(
  sc,
  name = NULL,
  path = name,
  memory = FALSE,
  overwrite = TRUE,
  ...
)
```

Arguments

<code>sc</code>	A <code>dask_connection</code> .
<code>name</code>	Name to register on <code>sc</code> . If <code>NULL</code> , derived from <code>path</code> .

path	File or glob. Defaults to name when you pass a single path-like argument as sparklyr does.
memory	If TRUE, persist() after read.
overwrite	If TRUE, replace an existing registered name.
...	Extra arguments forwarded to dask.dataframe.read_csv or read_parquet.

Details

path must be visible to **workers**. A Windows tempfile() is fine for master = "local"; it is not usable from a Docker cluster unless the path is a shared volume or object store.

Value

A tbl_dask.

Examples

```
## Not run:
sc <- dask_connect()
st <- copy_to(sc, mtcars, overwrite = TRUE)
tmp <- tempfile("dasklyr")
dir.create(tmp)
dask_write_parquet(st, file.path(tmp, "mtcars.parquet"), overwrite = TRUE)
dask_read_parquet(sc, "mtcars_pq", file.path(tmp, "mtcars.parquet"))
dask_write_csv(st, file.path(tmp, "mtcars_csv"))
dask_read_csv(sc, "mtcars_csv", file.path(tmp, "mtcars_csv", "*"))

## End(Not run)
```

dask_web

Dask dashboard URL

Description

Counterpart to sparklyr's spark_web().

Usage

```
dask_web(sc, browse = interactive())
```

Arguments

sc	A dask_connection.
browse	If TRUE, open the URL in a browser.

Value

The dashboard URL, invisibly.

Examples

```
## Not run:
sc <- dask_connect()
dask_web(sc, browse = FALSE)

## End(Not run)
```

dask_write_parquet	<i>Write a Dask table to Parquet or CSV</i>
--------------------	---

Description

Counterpart to sparklyr's `spark_write_parquet()` / `spark_write_csv()`. The destination must be writable from workers.

Usage

```
dask_write_parquet(x, path, mode = NULL, overwrite = FALSE, ...)

dask_write_csv(x, path, mode = NULL, overwrite = FALSE, ...)
```

Arguments

x	A <code>tbl_dask</code> .
path	Destination path.
mode	"overwrite" is accepted as an alias for <code>overwrite = TRUE</code> .
overwrite	If <code>TRUE</code> , replace existing Parquet files.
...	Extra arguments forwarded to <code>to_parquet</code> / <code>to_csv</code> .

Value

x, invisibly.

Examples

```
## Not run:
sc <- dask_connect()
st <- copy_to(sc, mtcars, overwrite = TRUE)
tmp <- tempfile()
dask_write_parquet(st, tmp, overwrite = TRUE)
dask_write_csv(st, tempfile())

## End(Not run)
```

ddf_schema

*Table utilities for Dask DataFrames***Description**

Counterpart to sparklyr's sdf_* helpers. Names use ddf_* so they are not mistaken for Spark.

Usage

```
ddf_schema(x)
ddf_columns(x)
ddf_npartitions(x)
ddf_ncol(x)
ddf_nrow(x)
ddf_dim(x)
ddf_repartition(x, npartitions = NULL, partitions = npartitions)
ddf_persist(x)
```

Arguments

x A tbl_dask.
npartitions, partitions Number of partitions. partitions is an alias for npartitions.

Details

ddf_nrow() computes (len(ddf)). dim() on a tbl_dask reports NA rows on purpose; use this when you need the real count.

Value

ddf_columns() a character vector; ddf_npartitions() / ddf_ncol() / ddf_nrow() integers; ddf_dim() c(nrow, ncol); ddf_schema() a tibble of name and type; ddf_repartition() / ddf_persist() the same tbl_dask with an updated Python frame.

Examples

```
## Not run:
sc <- dask_connect()
st <- copy_to(sc, mtcars, overwrite = TRUE, repartition = 2L)
ddf_schema(st)
```

```
ddf_npartitions(st)
ddf_nrow(st)
ddf_repartition(st, npartitions = 4L)

## End(Not run)
```

tbl.dask_connection	<i>Retrieve a registered Dask table</i>
---------------------	---

Description

dplyr [dplyr::tbl\(\)](#) method for dask_connection.

Usage

```
## S3 method for class 'dask_connection'
tbl(src, from, ...)
```

Arguments

src	A dask_connection.
from	Table name previously passed to copy_to() or a read helper.
...	Unused.

Value

A tbl_dask.

Examples

```
## Not run:
sc <- dask_connect()
copy_to(sc, iris, "iris", overwrite = TRUE)
tbl(sc, "iris")

## End(Not run)
```

tbl_dask

Distributed Dask tables

Description

A `tbl_dask` is a lazy Dask DataFrame plus optional grouping columns. `print()` and `head()` preview **the first partition** (Dask's `DataFrame.head()`). Use `dplyr::slice_head()` when you need a global first-n (that path collects, then wraps the result).

Usage

```
## S3 method for class 'tbl_dask'
print(x, n = 6L, ...)
```

```
## S3 method for class 'tbl_dask'
head(x, n = 6L, ...)
```

```
## S3 method for class 'tbl_dask'
dim(x)
```

```
## S3 method for class 'tbl_dask'
colnames(x, ...)
```

```
## S3 method for class 'tbl_dask'
names(x)
```

```
## S3 method for class 'tbl_dask'
collect(x, ...)
```

Arguments

<code>x</code>	A <code>tbl_dask</code> .
<code>n</code>	Number of rows to preview.
<code>...</code>	Unused.

Details

`dplyr::collect()` computes the full frame, resets the pandas index, and returns a tibble.

Value

`print()` and `head()` return a tibble preview; `collect()` a tibble of the full compute; `dim()` is `c(NA, ncol)`; `names()` / `colnames()` a character vector of column names.

Examples

```
## Not run:  
sc <- dask_connect()  
st <- copy_to(sc, mtcars, overwrite = TRUE)  
names(st)  
head(st, 3)  
dplyr::collect(st)  
  
## End(Not run)
```

Index

`collect.tbl_dask (tbl_dask)`, [16](#)
`colnames.tbl_dask (tbl_dask)`, [16](#)
`copy_to()`, [15](#)
`copy_to.dask_connection`, [2](#)

`dask_available`, [3](#)
`dask_coiled_available`, [4](#)
`dask_coiled_clusters`, [4](#)
`dask_config`, [5](#)
`dask_config()`, [6](#)
`dask_connect`, [6](#)
`dask_connect()`, [5](#)
`dask_connection_find`, [7](#)
`dask_connection_is_open`, [8](#)
`dask_dataframe`, [8](#)
`dask_disconnect`, [9](#)
`dask_disconnect()`, [5](#), [6](#), [10](#)
`dask_disconnect_all`, [10](#)
`dask_invoke`, [10](#)
`dask_read_csv (dask_read_parquet)`, [11](#)
`dask_read_parquet`, [11](#)
`dask_web`, [12](#)
`dask_write_csv (dask_write_parquet)`, [13](#)
`dask_write_parquet`, [13](#)
`ddf_columns (ddf_schema)`, [14](#)
`ddf_dim (ddf_schema)`, [14](#)
`ddf_ncol (ddf_schema)`, [14](#)
`ddf_npartitions (ddf_schema)`, [14](#)
`ddf_nrow (ddf_schema)`, [14](#)
`ddf_nrow()`, [14](#)
`ddf_persist (ddf_schema)`, [14](#)
`ddf_repartition (ddf_schema)`, [14](#)
`ddf_schema`, [14](#)
`dim()`, [14](#)
`dim.tbl_dask (tbl_dask)`, [16](#)
`dplyr::collect()`, [16](#)
`dplyr::copy_to()`, [2](#)
`dplyr::slice_head()`, [16](#)
`dplyr::tbl()`, [3](#), [15](#)

`head()`, [16](#)
`head.tbl_dask (tbl_dask)`, [16](#)

`names.tbl_dask (tbl_dask)`, [16](#)

`print.tbl_dask (tbl_dask)`, [16](#)

`tbl.dask_connection`, [15](#)
`tbl_dask`, [16](#)