

# Package ‘RedisParam’

May 2, 2024

**Title** Provide a 'redis' back-end for BiocParallel

**Version** 1.7.0

**Description** This package provides a Redis-based back-end for BiocParallel, enabling an alternative mechanism for distributed computation. The 'manager' distributes tasks to a 'worker' pool through a central Redis server, rather than directly to workers as with other BiocParallel implementations. This means that the worker pool can change dynamically during job evaluation. All features of BiocParallel are supported, including reproducible random number streams, logging to the manager, and alternative 'load balancing' task distributions.

**Depends** R (>= 4.2.0), BiocParallel (>= 1.29.12)

**SystemRequirements** hiredis

**Imports** methods, redux, withr, futile.logger

**License** Artistic-2.0

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.0

**Suggests** rmarkdown, knitr, testthat, BiocStyle

**Collate** 'Redis.R' 'RedisBackend-class.R' 'RedisParam-class.R'  
'RedisParam-accessors.R' 'RedisParam-logger.R'  
'RedisParam-methods.R' 'RedisTaskManager.R' 'zzz.R'

**biocViews** Infrastructure

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/RedisParam>

**git\_branch** devel

**git\_last\_commit** cbd7681

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-05-01

**Author** Martin Morgan [aut, cre] (<<https://orcid.org/0000-0002-5874-8148>>),  
Jiefei Wang [aut]

**Maintainer** Martin Morgan <mtmorgan.bioc@gmail.com>

## Contents

bpstopall . . . . .	2
RedisBackend . . . . .	3
RedisParam . . . . .	4
<b>Index</b>	<b>8</b>

---

bpstopall	<i>Deprecated functions in the RedisParam package</i>
-----------	-------------------------------------------------------

---

## Description

bpstopall() is provided for compatibility with previous versions of RedisParam, and will be de-funct after the next release. Use rpstopall() instead.

## Usage

```
bpstopall(x)
```

## Arguments

x                    a RedisParam object.

## Value

See ?rpstopall for return value.

## Examples

```
if (FALSE) {
  ## bpstopall()
  ## deprecated -- use rpstopall() instead
}
```

**Description**

Creating the Redis backend

**Usage**

```
RedisBackend(  
  RedisParam = NULL,  
  jobname = "myjob",  
  host = rphost(),  
  port = rpport(),  
  password = rppassword(),  
  timeout = .Machine$integer.max,  
  type = c("manager", "worker"),  
  id = NULL,  
  log = FALSE,  
  redis.log = NULL,  
  flushInterval = 5L  
)  
  
## S4 method for signature 'RedisBackend'  
.recv(worker)  
  
## S4 method for signature 'RedisBackend'  
.send(worker, value)  
  
## S4 method for signature 'RedisBackend'  
.close(worker)  
  
## S4 method for signature 'RedisBackend'  
.send_to(backend, node, value)  
  
## S4 method for signature 'RedisBackend'  
.recv_any(backend)  
  
## S4 method for signature 'RedisBackend'  
.recv_all(backend)  
  
## S4 method for signature 'RedisBackend'  
bpjobname(x)  
  
## S4 method for signature 'RedisBackend'  
bpworkers(x)
```

**Arguments**

RedisParam	RedisParam, if this argument is not NULL, all the other arguments will be ignored except type.
jobname	character(1) The job name used by the manager and workers to connect.
host	character(1) The host of the Redis server.
port	integer(1) The port of the Redis server.
password	character(1) The password of the redis server.
timeout	integer(1) The waiting time in BLPOP.
type	character(1) The type of the backend (manager or worker?).
id	character(1) The manager/worker ID. If not given by the user and the environment REDISPARAM_ID is not defined, a random ID will be used
log	logical(1) Whether to enable the log
redis.log	logical(1) Whether to enable the redis server log
flushInterval	numeric(1) The waiting time between two flush operation.

**Value**

RedisBackend() returns an object of class RedisBackend. This object is not useful to the end user.

---

RedisParam	<i>Enable redis-based parallel evaluation in BiocParallel</i>
------------	---------------------------------------------------------------

---

**Description**

RedisParam() creates an object describing manager and worker configurations for parallel computation using a Redis server back-end.

rpalive() tests whether it is possible to connect to a redis server using the host, port, and password in the RedisParam object.

rpstopall() is used from the manager to stop redis workers launched independently, with is.worker=TRUE.

rpworkers() determines the number of workers using snowWorkers() if workers are created dynamically, or a fixed maximum (currently 1000) if workers are listening on a queue.

rphost() reads the host name of the Redis server from the system environment variable REDISPARAM\_HOST, if the variable is not defined, fallback to REDIS\_HOST. Otherwise default to "127.0.0.1". rphost(x) gives the host name used by x.

rpport() reads the port of the Redis server from a system environment variable REDISPARAM\_PORT, if the variable is not defined, fallback to REDIS\_PORT. Otherwise default to 6379. rpport(x) gives the port used by x.

rppassword() reads an (optional) password from the system environment variable REDISPARAM\_PASSWORD, if the variable is not defined, fallback to REDIS\_PASSWORD. Otherwise default to NA\_character\_ (no password). rppassword(x) gives the password used by x.

**Usage**

```
RedisParam(  
  workers = rpworkers(is.worker),  
  tasks = 0L,  
  jobname = ipcid(),  
  log = FALSE,  
  logdir = NA,  
  threshold = "INFO",  
  resultdir = NA_character_,  
  stop.on.error = TRUE,  
  timeout = NA_integer_,  
  exportglobals = TRUE,  
  progressbar = FALSE,  
  RNGseed = NULL,  
  queue.multiplier = 2L,  
  redis.hostname = rphost(),  
  redis.port = rpport(),  
  redis.password = rppassword(),  
  is.worker = NA  
)  
  
rpalive(x)  
  
rpstopall(x)  
  
rpworkers(is.worker)  
  
rphost(x)  
  
rpport(x)  
  
rppassword(x)  
  
rpisworker(x)  
  
## S4 method for signature 'RedisParam'  
bpisup(x)  
  
## S4 method for signature 'RedisParam'  
bpbackend(x)  
  
## S4 method for signature 'RedisParam'  
bpstart(x, ...)  
  
## S4 method for signature 'RedisParam'  
bpstop(x)  
  
## S4 method for signature 'RedisParam'
```

```
bpworkers(x)
```

```
## S4 replacement method for signature 'RedisParam,logical'
bplog(x) <- value
```

### Arguments

workers	integer(1) number of redis workers. For <code>is.worker=FALSE</code> , this parameter is the maximum number of workers expected to be available. For <code>is.worker=NA</code> , this is the number of workers opened by <code>bpstart()</code> .
tasks	See <code>"BiocParallelParam-class"</code> .
jobname	character(1) name (unique) used to associate manager & workers on a queue.
log	See <code>"BiocParallelParam-class"</code> .
logdir	See <code>"BiocParallelParam-class"</code> .
threshold	See <code>"BiocParallelParam-class"</code> .
resultdir	See <code>"BiocParallelParam-class"</code> .
stop.on.error	See <code>"BiocParallelParam-class"</code> .
timeout	See <code>"BiocParallelParam-class"</code> .
exportglobals	See <code>"BiocParallelParam-class"</code> .
progressbar	See <code>"BiocParallelParam-class"</code> .
RNGseed	See <code>"BiocParallelParam-class"</code> .
queue.multiplier	numeric(1), The multiplier of the queue depth. The depth of the queue is calculated by <code>queue.multiplier * bpworkers(p)</code> . A proper queue depth can provide more performance benefit in task dispatching, but the improvement is likely to be marginal for an excessively large <code>queue.multiplier</code> .
redis.hostname	character(1) host name of redis server, from system environment variable <code>REDISPARAM_HOST</code> or <code>REDIS_HOST</code> , if both are not defined, the default <code>"127.0.0.1"</code> is used.
redis.port	integer(1) port of redis server, from system environment variable <code>REDISPARAM_PORT</code> or <code>REDIS_PORT</code> , if both are not defined, the default 6379 is used.
redis.password	character(1) or <code>NULL</code> , host password of redis server from system environment variable <code>REDISPARAM_PASSWORD</code> or <code>REDIS_PASSWORD</code> , if both are not defined, the default <code>NA_character_</code> (no password) is used.
is.worker	logical(1) <code>bpstart()</code> creates worker-only ( <code>TRUE</code> ), manager-only ( <code>FALSE</code> ), or manager and worker ( <code>NA</code> , default) connections.
x	A <code>RedisParam</code> object.
...	ignored.
value	The value you want to replace with

## Details

Use an instance of `RedisParam()` for interactive parallel evaluation using `bplapply()` or `bpiterate()`. `RedisParam()` requires access to a redis server, running on `manager.hostname` (e.g., 127.0.0.1) at `manager.port` (e.g., 6379). The manager and workers communicate via the redis server, rather than the socket connections used by other `BiocParallel` back-ends.

When invoked with `is.worker = NA` (the default) `bpstart()`, `bplapply()` and `bpiterate()` start and stop redis workers on the local computer. It may be convenient to use `bpstart()` and `bpstop()` independently, to amortize the cost of worker start-up across multiple calls to `bplapply()` / `bpiterate()`.

Alternatively, a manager and one or more workers can each be started in different processes across a network. The manager is started, e.g., in an interactive session, by specifying `is.worker=FALSE`. Workers are started, typically as background processes, with `is.worker = TRUE`. Both manager and workers must specify the same value for `jobname =`, the redis key used for communication. In this scenario, workers can be added at any time, including during e.g., `bplapply()` evaluation on the manager. See the vignette for possible scenarios.

## Value

`RedisParam()` returns an object of class `RedisParam`, for use in controlling parallel evaluation with `BiocParallel::bplapply()` or `BiocParallel::bpiterate()`.

## Examples

```
param <- RedisParam()
if (rpalive(param)) {
  res <- bplapply(1:20, function(i) Sys.getpid(), BPPARAM = param)
  table(unlist(res))
}

## Not run:
## start workers in background process(es)
rscript <- R.home("bin/Rscript")
worker_script <- tempfile()
writeLines(c(
  'worker <- RedisParam::RedisParam(jobname = "demo", is.worker = TRUE)',
  'RedisParam::bpstart(worker)'
), worker_script)

for (i in seq_len(2))
  system2(rscript, worker_script, wait = FALSE)

## start manager
p <- RedisParam(jobname = "demo", is.worker = FALSE)
result <- bplapply(1:5, function(i) Sys.getpid(), BPPARAM = p)
table(unlist(result))

## stop all workers
rpstopall(p)

## End(Not run)
```

# Index

## \* **internal**

- RedisBackend, 3
- .close, RedisBackend-method  
(RedisBackend), 3
- .recv, RedisBackend-method  
(RedisBackend), 3
- .recv\_all, RedisBackend-method  
(RedisBackend), 3
- .recv\_any, RedisBackend-method  
(RedisBackend), 3
- .send, RedisBackend-method  
(RedisBackend), 3
- .send\_to, RedisBackend-method  
(RedisBackend), 3
  
- bpbackend, RedisParam-method  
(RedisParam), 4
- bpisup, RedisParam-method (RedisParam), 4
- bpjobname, RedisBackend-method  
(RedisBackend), 3
- bplog<-, RedisParam, logical-method  
(RedisParam), 4
- bpstart, RedisParam-method (RedisParam),  
4
- bpstop, RedisParam-method (RedisParam), 4
- bpstopall, 2
- bpworkers, RedisBackend-method  
(RedisBackend), 3
- bpworkers, RedisParam-method  
(RedisParam), 4
  
- RedisBackend, 3
- RedisParam, 4
- rpalive (RedisParam), 4
- rphost (RedisParam), 4
- rpisworker (RedisParam), 4
- rppassword (RedisParam), 4
- rpport (RedisParam), 4
- rpstopall (RedisParam), 4
- rpworkers (RedisParam), 4