Package 'doRedis'

January 28, 2020

Type Package

Title 'Foreach' Parallel Adapter Using the 'Redis' Database

Version 2.0.0

Date 2020-01-26

Author B. W. Lewis <blewis@illposed.net>

Maintainer B. W. Lewis <blewis@illposed.net>

Description A parallel back end for the 'foreach' package using the 'Redis' database.

BugReports https://github.com/bwlewis/doRedis/issues

Depends R (>= 3.0), foreach(>= 1.3.0), iterators(>= 1.0.0), utils

Imports parallel, redux, stats

Suggests knitr

VignetteBuilder knitr

License GPL-2

LazyLoad yes

RoxygenNote 7.0.2

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-01-28 17:30:03 UTC

R topics documented:

`doRedis-package` *A Redis parallel back end for foreach.*

Description

The doRedis package imlpements an elastic parallel back end for foreach using the Redis key/value database.

See Also

`registerDoRedis`, `startLocalWorkers`

`.doRedis` *internal function called by foreach*

Description

internal function called by foreach

Usage

```
.doRedis(obj, expr, envir, data)
```

Arguments

`obj`	a foreach object
`expr`	the expression to evaluate
`envir`	the expression environment
`data`	a list of parameters from registerDoRedis

Value

the foreach result

jobs *List doRedis jobs*

Description

List doRedis jobs

Usage

```
jobs(queue = "*")
```

Arguments

queue	List jobs for the specified queue, or set to "*" to list jobs for all queues

Value

a data frame listing jobs by row with variables queue, id, user, host and time (submitted).

logger *Print a timestamped message to the standard error stream.*

Description

Use to help debug remote doRedis workers.

Usage

```
logger(msg)
```

Arguments

msg	a character message to print to the standard error stream

Value

The character string that was printed, decorated with time and system info.

redisConnect *Explicitly connect to a Redis server.*

Description

This function is normally not needed, use the redux package functions instead, or simply registerDoRedis.

Usage

```
redisConnect(host = "localhost", port = 6379L, password, ...)
```

Arguments

host	character Redis host name
port	integer Redis port number
password	optional character Redis password
...	optional additional arguments for compatability with old rredis, ignored

See Also

registerDoRedis, redisWorker, startLocalWorkers

redisDelete *A convenience function to delete a Redis key*

Description

A convenience function to delete a Redis key

Usage

```
redisDelete(key)
```

Arguments

key	(character or raw) Redis key name to delete

Value

Redis status message

See Also

hiredis

redisGet — *A convenience function to return an R value from a Redis key.*

Description

This function assumes the value associated with the Redis key is a serialized (binary) R value and unserializes it on return.

Usage

```
redisGet(key)
```

Arguments

key	(character or raw) Redis key name

Value

Unserialized R value.

See Also

hiredis

redisSet — *A convenience function to set an R value in a Redis key*

Description

This function serializes the val argument.

Usage

```
redisSet(key, val)
```

Arguments

key	(character or raw) Redis key name
val	R value to set

Value

Redis status message

See Also

hiredis

`redisWorker` *Initialize a doRedis worker process.*

Description

The redisWorker function enrolls the current R session in one or more doRedis worker pools specified by the work queue names. The worker loop takes over the R session until the work queue(s) are deleted, after which the worker loop exits after the `linger` period, or until the worker has processed `iter` tasks. Running workers also terminate after network activity with Redis remains inactive for longer than the `timeout` period set in the `redisConnect` function. That value defaults internally to 30 seconds in `redisWorker`. You can increase it by including a timeout=n argument value.

Usage

```
redisWorker(
  queue,
  host = "localhost",
  port = 6379,
  iter = Inf,
  linger = 30,
  log = stderr(),
  connected = FALSE,
  password = NULL,
  loglevel = 0,
  timelimit = 0,
  ...
)
```

Arguments

`queue`	work queue name or a vector of queue names
`host`	Redis database host name or IP address
`port`	Redis database port number
`iter`	maximum number of tasks to process before exiting the worker loop
`linger`	timeout in seconds after which the work queue is deleted that the worker terminates
`log`	print messages to the specified file connection
`connected`	set to `TRUE` to reuse an existing open connection to Redis, otherwise establish a new one
`password`	optional Redis database password
`loglevel`	set to > 0 to increase verbosity in the log
`timelimit`	set to > 0 to specify a task time limit in seconds, after which worker processes are killed; beware that setting this value > 0 will terminate any R worker process if their task takes too long.
`...`	Optional additional parameters passed to `redisConnect`

Value

NULL is invisibly returned.

Note

The worker connection to Redis uses a TCP timeout value of 30 seconds by default. That means that the worker will exit after about 30 seconds of inactivity. If you want the worker to remain active for longer periods, set the `timeout` option to a larger value.

Use the `linger` option to instruct the worker to linger for up to the indicated number of seconds after the listening work queue has been removed. After at most that interval, the worker will exit after removing the queue.

See Also

`registerDoRedis`, `startLocalWorkers`

`registerDoRedis` *Register the Redis back end for foreach.*

Description

The doRedis package imlpements a simple but flexible parallel back end for foreach that uses Redis for inter-process communication. The work queue name specifies the base name of a small set of Redis keys that the master and worker processes use to exchange data.

Usage

```
registerDoRedis(
  queue,
  host = "localhost",
  port = 6379,
  password,
  ftinterval = 30,
  chunkSize = 1,
  progress = FALSE,
  ...
)
```

Arguments

`queue`	A work queue name
`host`	The Redis server host name or IP address
`port`	The Redis server port number
`password`	An optional Redis database password
`ftinterval`	Default fault tolerance interval in seconds

chunkSize	Default iteration granularity, see setChunkSize
progress	(logical) Show progress bar for computations?
...	Optional arguments passed to redisConnect

Details

Back-end worker R processes advertise their availablility for work with the redisWorker function.

The doRedis parallel back end tolerates faults among the worker processes and automatically resubmits failed tasks. It is also portable and supports heterogeneous sets of workers, even across operative systems. The back end supports dynamic pools of worker processes. New workers may be added to work queues at any time and can be used by running foreach computations.

Value

NULL is invisibly returned.

Note

All doRedis functions require access to a Redis database server (not included with this package).

The doRedis package sets RNG streams across the worker processes using the L'Ecuyer-CMRG method from R's parallel package for reproducible pseudorandom numbers independent of the number of workers or task distribution. See the package vignette for more details and additional options.

Avoid using fork-based parallel functions within doRedis expressions. Use of `mclapply` and similar functions in the body of a doRedis foreach loop can result in worker faults.

See Also

foreach, doRedis-package, setChunkSize, removeQueue

Examples

```
## Not run:
## The example assumes that a Redis server is running on the local host
## and standard port.

## 1. Open one or more 'worker' R sessions and run:
require('doRedis')
redisWorker('jobs')

## 2. Open another R session acting as a 'master' and run this simple
##    sampling approximation of pi:
require('doRedis')
registerDoRedis('jobs')
foreach(j=1:10, .combine=sum, .multicombine=TRUE) \
        4 * sum((runif(1000000) ^ 2 + runif(1000000) ^ 2) < 1) / 10000000
removeQueue('jobs')

## End(Not run)
```

removeJob *Remove Redis keys associated with one or more doRedis jobs*

Description

Remove Redis keys associated with one or more doRedis jobs

Usage

```
removeJob(job)
```

Arguments

job	Either a named character vector with "queue" and "id" entries corresponding to a doRedis job queue and job id, or a list with equal-length "queue" and "id" entries, or a data frame with "queue" and "id" entries, for example as returned by jobs.

Value

NULL is invisibly returned

removeQueue *Remove a doRedis queue and delete all associated keys from Redis.*

Description

Removing a doRedis queue cleans up associated keys in the Redis database and signals to workers listening on the queue to terminate. Workers terminate after their timeout period after their work queue is deleted.

Usage

```
removeQueue(queue)
```

Arguments

queue	the doRedis queue name

Value

NULL is invisibly returned.

Note

Workers listening for work on more than one queue will only terminate after all their queues have been deleted.

setChunkSize *Set the default granularity of distributed tasks.*

Description

A job is the collection of all tasks in a foreach loop. A task is a collection of loop iterations of at most size chunkSize. R workers are assigned work by task in blocks of at most chunkSize loop iterations per task. The default value is one iteration per task. Setting the default chunk size larger for shorter-running jobs can substantially improve performance. Setting this value too high can negatively impact load-balancing across workers, however.

Usage

```
setChunkSize(value = 1)
```

Arguments

value	positive integer chunk size setting

Value

value is invisibly returned.

Examples

```
## Not run:
setChunkSize(5)
foreach(j=1:10) %dopar% j

## End(Not run)
```

setExport *Manually add symbol names to the worker environment export list.*

Description

The setExport function lets users manually declare symbol names of corresponding objects that should be exported to workers.

Usage

```
setExport(names = c())
```

Arguments

names	A character vector of symbol names to export.

Details

The `foreach` function includes a similar `.export` parameter.

We provide this supplemental export option for users without direct access to the `foreach` function, for example, when `foreach` is used inside another package.

Value

`names` is invisibly returned.

Examples

```
## Not run:
require("doRedis")
registerDoRedis("work queue")
startLocalWorkers(n=1, queue="work queue")

f <- function() pi

foreach(1) %dopar% eval(call("f"))
# Returns the error:
# Error in eval(call("f")) : task 1 failed - could not find function "f"

# Manuall export the symbol f:
setExport("f")
foreach(1) %dopar% eval(call("f"))
# Ok then.
#[[1]]
#[1] 3.141593
removeQueue("work queue")

## End(Not run)
```

---

setFtinterval *Set the fault tolerance check interval in seconds.*

Description

Set the fault tolerance check interval in seconds.

Usage

```
setFtinterval(value = 30)
```

Arguments

`value`	positive integer number of seconds

Value

value is invisibly returned.

Examples

```
## Not run:
setFtinterval(5)
foreach(j=1:10) %dopar% j

## End(Not run)
```

---

setPackages *Manually set package names in the worker environment package list.*

Description

The setPackages function lets users manually declare packages that R worker processes need to load before running their tasks.

Usage

```
setPackages(packages = c())
```

Arguments

packages	A character vector of package names.

Details

The foreach function includes a similar .packages parameter.

Defines a way to set the foreach .packages option for users without direct access to the foreach function, for example, when foreach is used inside another package.

Value

The value of packages is invisibly returned.

setProgress *Progress bar*

Description

Progress bar

Usage

```
setProgress(value = FALSE)
```

Arguments

value	if TRUE, display a text progress bar indicating status of the computation

Value

value is invisibly returned

setReduce *Set two-level distributed reduction*

Description

Instruct doRedis to perform either the .combine reduction function or another specified function per task on each worker before returning results, cf. foreach. Combined results are then processed through the specified function fun for two levels of reduction functions. This option only applies when the chunkSize option is greater than one, and automatically sets .multicombine=FALSE.

Usage

```
setReduce(fun = NULL)
```

Arguments

fun	a function of two arguments, set to NULL to disable combining, or leave missing to implicitly set the gather function formally identical to the .combine function but with an empty environment.

Details

This approach can improve performance when the .combine function is expensive to compute, and when function emits significantly less data than it consumes. The same effect is usually achievable by simply adding the reduction function to the end of the foreach loop expression (but must be decided prior to run time).

Value

fun is invisibly returned, or TRUE is returned when fun is missing (in which case the .combine function is used).

Note

Do not use this function, use the 'compile-time' version instead directly in the foreach loop: foreach(...,.options.redis=
Setting a reduction function at run-time will generally alter the result.

See Also

foreach, setChunkSize

Examples

```
## Not run:
setChunkSize(3)
foreach(j=1:10, .combine=c, .options.redis=list(reduce=list)) %dopar% j

## End(Not run)
```

---

startLocalWorkers *Start one or more background R worker processes on the local system.*

Description

Use startLocalWorkers to start one or more doRedis R worker processes in the background. The worker processes are started on the local system using the redisWorker function.

Usage

```
startLocalWorkers(
  n,
  queue,
  host = "localhost",
  port = 6379,
  iter = Inf,
  linger = 30,
  log = stdout(),
  Rbin = paste(R.home(component = "bin"), "R", sep = "/"),
  password,
  ...
)
```

Arguments

n	number of workers to start
queue	work queue name
host	Redis database host name or IP address
port	Redis database port number
iter	maximum number of tasks to process before exiting the worker loop
linger	timeout in seconds after which the work queue is deleted that the worker terminates
log	print messages to the specified file connection
Rbin	full path to the command-line R program
password	optional Redis database password
...	optional additional parameters passed to the redisWorker function

Details

Running workers self-terminate after a linger period if their work queues are deleted with the removeQueue function, or when network activity with Redis remains inactive for longer than the timeout period set in the redisConnect function. That value defaults internally to 3600 (one hour) in startLocalWorkers. You can increase it by including a timeout=n argument value.

Value

NULL is invisibly returned.

See Also

registerDoRedis, redisWorker

Examples

```
## Not run:
require('doRedis')
registerDoRedis('jobs')
startLocalWorkers(n=2, queue='jobs', linger=5)
print(getDoParWorkers())
foreach(j=1:10,.combine=sum,.multicombine=TRUE) \
          4*sum((runif(1000000)^2 + runif(1000000)^2)<1)/10000000
removeQueue('jobs')

## End(Not run)
```

`tasks` *List running doRedis tasks*

Description

List running doRedis tasks

Usage

```
tasks(queue = "*", id = "*")
```

Arguments

`queue`	List jobs for the specified queue, or set to "*" to list jobs for all queues
`id`	List tasks for the specified job id, or set to "*" to list tasks for all job ids

Value

a data frame listing jobs by row with variables queue, id, user, master, time, iter, host, pid

Note

The returned values indicate

1. `queue` the doRedis queue name
2. `id` the doRedis job id
3. `user` the user running the job
4. `master` the host name or I.P. address where the job was submitted (and the master R process runs)
5. `time` system time on the worker node when the task was started
6. `iter` the loop iterations being run by the task
7. `host` the host name or I.P. address where the task is running
8. `pid` the process ID of the R worker running the task on `host`

Tasks are listed until a key associated with them expires in Redis. Thus running tasks are not explicitly removed from the task list immediately when they terminate, but may linger on the list for a short while after (a few seconds).

Index