Package: heartbeatr (via r-universe)

September 26, 2024

Title Heartbeat Support using 'Redis'

Version 0.6.0

Description Simple heartbeat support for R using 'Redis'. A heartbeat is a background thread that acts as a dead-man's switch. It will create a key on Redis that will automatically expire after a number of seconds and then periodically refresh that key, even when the R process is busy. If the process dies for some reason, then the key will disappear. A heartbeat can be used to detect loss of worker processes on shared systems.

License MIT + file LICENSE

URL `https://github.com/mrc-ide/heartbeatr`

BugReports `https://github.com/mrc-ide/heartbeatr/issues`

SystemRequirements C++11, libhiredis, redis-server

Imports R6, redux (>= 1.0.0)

Suggests processx, testthat

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

Language en-GB

Repository https://mrc-ide.r-universe.dev

RemoteUrl https://github.com/mrc-ide/heartbeatr

RemoteRef master

RemoteSha f44cee1ef8d82e829930ae04b8964b4c2cfede10

Contents

heartbeat *Create a heartbeat instance*

Description

Create a heartbeat instance. This can be used by running `obj$start()` which will reset the TTL (Time To Live) on `key` every `period` seconds (don't set this too high). If the R process dies, then the key will expire after `3 * period` seconds (or set `expire`) and another application can tell that this R instance has died.

Usage

```
heartbeat(
  key,
  period,
  expire = 3 * period,
  value = expire,
  config = NULL,
  start = TRUE,
  timeout = 10
)
```

Arguments

`key`	Key to use
`period`	Timeout period (in seconds)
`expire`	Key expiry time (in seconds)
`value`	Value to store in the key. By default it stores the expiry time, so the time since last heartbeat can be computed.
`config`	Configuration parameters passed through to `redux::redis_config`. Provide as either a named list or a `redis_config` object. This allows host, port, password, db, etc all to be set. Socket connections (i.e., using `path` to access Redis over a socket) are not currently supported.
`start`	Should the heartbeat be started immediately?
`timeout`	Time, in seconds, to wait for the heartbeat to appear. It should generally appear very quickly (within a second unless your connection is very slow) so this can be generally left alone.

Details

The heartbeat object has three methods:

- `is_running()` which returns `TRUE` or `FALSE` if the heartbeat is/is not running.
- `start()` which starts a heartbeat
- `stop()` which requests a stop for the heartbeat

Heavily inspired by the `doRedis` package.

Examples

```
if (redux::redis_available()) {
  rand_str <- function() {
    paste(sample(letters, 20, TRUE), collapse = "")
  }
  key <- sprintf("heartbeatr:test:%s", rand_str())
  h <- heartbeatr::heartbeat(key, 1, expire = 2)
  con <- redux::hiredis()

  # The heartbeat key exists
  con$EXISTS(key)

  # And has an expiry of less than 2000ms
  con$PTTL(key)

  # We can manually stop the heartbeat, and 2s later the key will
  # stop existing
  h$stop()

  # Sys.sleep(2)
  # con$EXISTS(key) # 0
}
```

`heartbeat_send_signal` *Send a signal*

Description

Sends a signal to a heartbeat process that is using key key

Usage

```
heartbeat_send_signal(con, key, signal)
```

Arguments

`con`	A hiredis object
`key`	The heartbeat key
`signal`	A signal to send (e.g. `tools::SIGINT` or `tools::SIGKILL`)

Examples

```
if (redux::redis_available()) {
  rand_str <- function() {
    paste(sample(letters, 20, TRUE), collapse = "")
  }
  # Suppose we have a process that exposes a heartbeat running on
  # this key:
```

```
  key <- sprintf("heartbeatr:test:%s", rand_str())

  # We can send it an interrupt over redis using:
  con <- redux::hiredis()
  heartbeatr::heartbeat_send_signal(con, key, tools::SIGINT)
}
```

Index