Package ‘mirai’

March 15, 2022

Type Package

Title Minimalist Async Evaluation Framework for R

Version 0.1.1

Description Extremely simple and lightweight method for concurrent /
parallel code execution, built on ‘nanonext’ and ‘NNG’ (Nanomsg Next Gen)
technology.

License GPL (>= 3)

BugReports https://github.com/shikokuchuo/mirai/issues

URL https://shikokuchuo.net/mirai/,
https://github.com/shikokuchuo/mirai/

Encoding UTF-8

Depends R (>= 2.4)

Imports nanonext (>= 0.3.0)

RoxygenNote 7.1.2

NeedsCompilation no

Author Charlie Gao [aut, cre] (<https://orcid.org/0000-0002-0750-061X>)

Maintainer Charlie Gao <charlie.gao@shikokuchuo.net>

Repository CRAN

Date/Publication 2022-03-15 10:00:14 UTC

R topics documented:

  mirai-package .................................................. 2
call_mirai ......................................................... 2
daemons ........................................................... 3
eval_mirai .......................................................... 5
stop_mirai ......................................................... 6

Index 8
**Description**

Extremely simple and lightweight method for concurrent / parallel code execution, built on ‘nanonext’ and ‘NNG’ (Nanomsg Next Gen) technology. mirai is Japanese for ‘future’.

**Links**

- mirai website: [https://shikokuchuo.net/mirai/](https://shikokuchuo.net/mirai/)
- mirai on CRAN: [https://cran.r-project.org/package=mirai](https://cran.r-project.org/package=mirai)
- nanonext website: [https://shikokuchuo.net/nanonext/](https://shikokuchuo.net/nanonext/)
- nanonext on CRAN: [https://cran.r-project.org/package=nanonext](https://cran.r-project.org/package=nanonext)
- NNG website: [https://nng.nanomsg.org/](https://nng.nanomsg.org/)

**Author(s)**

Charlie Gao <charlie.gao@shikokuchuo.net> (ORCID)

---

**call_mirai**

**mirai (Call Value)**

**Description**

Call the value of a ‘mirai’, waiting for the the asynchronous operation to resolve if it is still in progress.

**Usage**

```
call_mirai(mirai)
```

**Arguments**

- **mirai**
  - a ‘mirai’ object.

**Details**

This function will wait for the async operation to complete if still in progress (blocking).

If an error occurred in evaluation, a nul byte $\emptyset$ (or serialized nul byte) will be returned. `is_nul_byte` can be used to test for a nul byte.

The ‘mirai’ updates itself in place, so to access the value of a ‘mirai’ x directly, use `call_mirai(x)$value`. 
Value

The passed 'mirai' (invisibly). The retrieved value is stored in $data.

Alternatively

The value of a 'mirai' may be accessed at any time at $value, and if yet to resolve, an 'unresolved' logical NA will be returned instead.

`unresolved` may also be used on a 'mirai', and returns TRUE only if a 'mirai' has yet to resolve and FALSE otherwise. This is suitable for use in control flow statements such as `while` or `if`.

Examples

```r
if (interactive()) {
  # Only run examples in interactive R sessions
  m <- mirai(x + y + 1, x = 2, y = 3)
  m
  m$value
  Sys.sleep(0.2)
  m$value

  m <- mirai(as.matrix(df), df = data.frame())
  call_mirai(m)$value

  m <- mirai({
    res <- rnorm(n)
    res / rev(res)
  }, n = 1e6)
  while (unresolved(m)) {
    cat("unresolved\n")
    Sys.sleep(0.1)
  }
  m$value
}
```

---

**daemons (Background Processes)**

**Description**

Set or view the number of daemons (background processes). Create persistent background processes to send `mirai` requests. Setting a positive number of daemons provides a potentially more efficient solution for async operations as new processes no longer need to be created on an ad hoc basis.
Usage

daemons(...)  

Arguments

...  
an integer to set the number of daemons. 'view' to view the currently set number of daemons.

Details

{mirai} will revert to the default behaviour of creating a new background process for each request if the number of daemons is set to 0.

It is highly recommended to shut down daemons by setting daemons(0) or explicitly unloading the package before exiting your R session. This will ensure that all processes exit cleanly and resources are freed.

The current implementation is low-level and ensures tasks are evenly-distributed amongst daemons without actively managing a task queue. This approach provides a robust and resource-light solution, particularly well-suited to working with similar-length tasks, or where the number of concurrent tasks typically does not exceed the number of available daemons.

Value

Depending on the specified ... parameter:

- integer: integer change in number of daemons (created or destroyed).
- 'view': integer number of currently set daemons.
- missing: the 'nanoSocket' for connecting to the daemons, or NULL if it is yet to be created.

Examples

if (interactive()) {
  # Only run examples in interactive R sessions

  # To create 4 background processes
daemons(4)
  # To view the number of background processes
daemons("view")
  # To destroy them all
daemons(0)
}
Description

Evaluate an expression asynchronously in a new background R process. This function will return immediately with a 'mirai', which will resolve to the evaluated result once complete.

Usage

eval_mirai(.expr, ...)
mirai(.expr, ...)

Arguments

.exprr an expression to evaluate in a new R process. This may be of arbitrary length, wrapped in {} if necessary.

... named arguments specifying the variables contained in '.expr'.

Details

This function will return a 'mirai' object immediately.

The value of a 'mirai' may be accessed at any time at $value, and if yet to resolve, an 'unresolved' logical NA will be returned instead.

unresolved may also be used on a 'mirai', which returns TRUE only if a 'mirai' has yet to resolve and FALSE otherwise. This is suitable for use in control flow statements such as while or if.

Alternatively, to call (and wait for) the result, use call_mirai on the returned 'mirai' object. This will block until the result is returned.

The expression '.expr' will be evaluated in a new R process in a clean environment consisting of the named objects passed as '...' only.

mirai is an alias for eval_mirai.

Value

A 'mirai' object.

Examples

if (interactive()) {
  # Only run examples in interactive R sessions

  m <- mirai(x + y + 1, x = 2, y = 3)

  m
  m$value
  Sys.sleep(0.2)
m$value

m <- mirai(as.matrix(df), df = data.frame())
call_mirai(m)$value

m <- mirai({
  res <- rnorm(n)
  res / rev(res)
}, n = 1e6)
while (unresolved(m)) {
  cat("unresolved
"
  Sys.sleep(0.1)
}
m$value

stop_mirai

Description
Stop evaluation of a mirai that is in progress.

Usage
stop_mirai(mirai)

Arguments
mirai a 'mirai' object.

Details
Stops the asynchronous operation associated with 'mirai' by aborting, and then waits for it to complete or to be completely aborted. The 'mirai' is then deallocated and attempting to access the value at $value will result in an error.

Value
Invisible NULL.

Examples
if (interactive()) {
  # Only run examples in interactive R sessions
  m <- mirai(Sys.sleep(n), n = 5)
  stop_mirai(m)
stop_mirai

}
Index

call_mirai, 2, 5

daemons, 3

eval_mirai, 5, 5

is_nul_byte, 2

mirai, 3, 5
mirai(eval_mirai), 5
mirai-package, 2

stop_mirai, 6

unresolved, 3, 5