Package: future.mizu (via r-universe)

November 23, 2026

Version 0.1.0

Depends R (>= 4.3.0),

Imports future (>= 1.82.0), mizu (>= 0.1.1), parallelly

Suggests future.tests, future.apply, listenv

Title Parallel Future Backend using 'mizu'

Description Implementation of the Future API <doi:10.32614/RJ-2021-048> on top of the 'mizu' package <doi:10.32614/CRAN.package.mizu>. The backend is fully compliant with the Future API, as validated by the 'future.tests' conformance test suite, meaning that code using futures gives identical results and behaves the same regardless of which future backend is used. Futures are resolved in parallel on the local machine by R worker processes that exchange data via shared memory. A common use is to parallelize existing map-reduce calls via the 'futurize' package <doi:10.48550/arXiv.2601.17578>, e.g. 'plan(future.mizu::mizu_multisession)' followed by 'lapply(xs, fcn) |> futurize()' or 'purrr::map(xs, fcn) |> futurize()'.

License Apache License (>= 2)

Language en-US

Encoding UTF-8

URL https://future.mizu.futureverse.org/

BugReports https://github.com/futureverse/future.mizu/issues

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Repository https://futureverse.r-universe.dev

Date/Publication 2026-11-23 15:12:36 UTC

RemoteUrl https://github.com/futureverse/future.mizu

RemoteRef main

RemoteSha a515c09dc8b7c113a7d4e01e5f72ce1ece5762b5

Contents

future.mizu 2
mizu_multisession 3
MizuFutureBackend 4

Index **5**

`future.mizu` *future.mizu: Parallel Future Backend using 'mizu'*

Description

The **future.mizu** package implements the Future API using the **mizu** package, which runs parallel R workers on the local machine communicating via shared memory. The Future API is defined by the **future** package.

Details

To use mizu futures, attach **future**, and select the type of future you wish to use, e.g. `plan(future.mizu::mizu_multisessi`

Example

```
library(futurize)                     # also attaches 'future'
plan(future.mizu::mizu_multisession)  # parallelize on the current machine

# A slow function
slow_fcn <- function(x) {
  Sys.sleep(1.0)
  x^2
}

# Example data
xs <- 1:10

# Sequential and parallel base R apply
ys <- lapply(xs, slow_fcn)
ys <- lapply(xs, slow_fcn) |> futurize()

# Sequential and parallel purrr map
library(purrr)
ys <- xs |> map(slow_fcn)
ys <- xs |> map(slow_fcn) |> futurize()

# Sequential and parallel foreach calls
library(foreach)
ys <- foreach(x = xs) %do% slow_fcn(x)
ys <- foreach(x = xs) %do% slow_fcn(x) |> futurize()
```

Author(s)

Maintainer: Henrik Bengtsson <henrikb@braju.com> (ORCID) [copyright holder]

Authors:

- Henrik Bengtsson <henrikb@braju.com> (ORCID) [copyright holder]

See Also

Useful links:

- https://future.mizu.futureverse.org/
- Report bugs at https://github.com/futureverse/future.mizu/issues

---

`mizu_multisession` *Mizu-based localhost multisession futures*

Description

WARNING: This function must never be called. It may only be used with `future::plan()`

Usage

```
mizu_multisession(..., workers = availableCores(), envir = parent.frame())
```

Arguments

`workers`	The number of parallel processes to use. If a function, it is called without arguments *when the future is created* and its value is used to configure the workers. If `workers == 1`, then all processing using done in the current/main R session and we therefore fall back to using a sequential future. To override this fallback, use `workers = I(1)`.
`envir`	The environment from where global objects should be identified.
`...`	Not used.

Value

Nothing.

Examples

```
library(future)
plan(future.mizu::mizu_multisession)

# A function that returns a future
# (note that N is a global variable)
f <- function() future({
  4 * sum((runif(N) ^ 2 + runif(N) ^ 2) < 1) / N
```

```
}, seed = TRUE)

# Run a simple sampling approximation of pi in parallel using  M * N points:
N <- 1e6  # samples per worker
M <- 10   # iterations
pi_est <- Reduce(sum, Map(value, replicate(M, f()))) / M
print(pi_est)

## Switch back to sequential processing, which also
## shuts down the automatically launched mizu workers
plan(sequential)
```

`MizuFutureBackend` *A future backend based on the 'mizu' framework*

Description

Set up the future parameters.

Usage

```
MizuFutureBackend(workers = availableCores(), ...)
```

Arguments

`workers`	The number of parallel workers (R processes) to launch, or a function returning such a number. A single worker, unless protected by `base::I()`, falls back to sequential processing in the current R process. At most 64 workers are supported by **mizu**.
`...`	Additional arguments passed to `future::FutureBackend()`.

Value

An object of class `MizuFutureBackend`.

Limitations

mizu workers always run on the local machine, and they communicate via shared memory. Cancelling a future is advisory: a future that is already running on a worker cannot be interrupted, and runs to completion with its result being discarded.

Index

base::I(), *4*

environment, *3*

future.mizu, 2
future.mizu-package *(*future.mizu*)*, 2
future::FutureBackend(), *4*
future::plan(), *3*

mizu_multisession, 3
MizuFutureBackend, 4