Package 'future.batchtools'

January 4, 2021

Version 0.10.0

Depends R (>= 3.2.0), future (>= 1.17.0)

Imports batchtools (>= 0.9.13), utils

Suggests future.apply, listenv, markdown, R.rsp

VignetteBuilder R.rsp

Title A Future API for Parallel and Distributed Processing using 'batchtools'

Description Implementation of the Future API on top of the 'batchtools' package. This allows you to process futures, as defined by the 'future' package, in parallel out of the box, not only on your local machine or ad-hoc cluster of machines, but also via high-performance compute ('HPC') job schedulers such as 'LSF', 'OpenLava', 'Slurm', 'SGE', and 'TORQUE' / 'PBS', e.g. 'y <- future.apply::future_lapply(files, FUN = process)'.

License LGPL (>= 2.1)

LazyLoad TRUE

URL `https://github.com/HenrikBengtsson/future.batchtools`

BugReports `https://github.com/HenrikBengtsson/future.batchtools/issues`

RoxygenNote 7.1.1

NeedsCompilation no

Author Henrik Bengtsson [aut, cre, cph]

Maintainer Henrik Bengtsson <`henrikb@braju.com`>

Repository CRAN

Date/Publication 2021-01-04 21:20:13 UTC

R topics documented:

batchtools_custom 2
batchtools_local 4
batchtools_template 5
future.batchtools 8

Index **10**

batchtools_custom *Batchtools futures for custom batchtools configuration*

Description

Batchtools futures for custom batchtools configuration

Usage

```
batchtools_custom(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  resources = list(),
  workers = NULL,
  conf.file = findConfFile(),
  cluster.functions = NULL,
  registry = list(),
  ...
)
```

Arguments

`expr`	The R expression to be evaluated
`envir`	The environment in which global environment should be located.
`substitute`	Controls whether `expr` should be `substitute()`:d or not.
`globals`	(optional) a logical, a character vector, a named list, or a Globals object. If TRUE, globals are identified by code inspection based on `expr` and `tweak` searching from environment `envir`. If FALSE, no globals are used. If a character vector, then globals are identified by lookup based their names `globals` searching from environment `envir`. If a named list or a Globals object, the globals are used as is.
`label`	(optional) Label of the future (where applicable, becomes the job name for most job schedulers).
`resources`	(optional) A named list passed to the **batchtools** template (available as variable `resources`). See Section 'Resources' in `batchtools::submitJobs()` more details.
`workers`	(optional) The maximum number of workers the batchtools backend may use at any time. Interactive and "local" backends can only process one future at the time (`workers = 1L`), whereas HPC backends, where futures are resolved via separate jobs on a scheduler, can have multiple workers. In the latter, the default is `workers = NULL`, which will resolve to `getOption("future.batchtools.workers")`. If that is not specified, the value of environment variable `R_FUTURE_BATCHTOOLS_WORKERS` will be used. If neither are specified, then the default is `100`.

`conf.file`	(character) A batchtools configuration file as for instance returned by `batchtools::findConfFile()`.
`cluster.functions`	A ClusterFunctions object.
`registry`	(optional) A named list of settings to control the setup of the batchtools registry.
`...`	Additional arguments passed to `BatchtoolsFuture()`.

Value

An object of class `BatchtoolsFuture`.

Examples

```
options(error = function(...) {
  print(traceback())
})

cf <- batchtools::makeClusterFunctionsInteractive(external = TRUE)
print(cf)
str(cf)
plan(batchtools_custom, cluster.functions = cf)
print(plan())
print(nbrOfWorkers())

## Create explicit future
f <- future({
  cat("PID:", Sys.getpid(), "\n")
  42L
})
print(f)
v <- value(f)
print(v)

options(error = NULL)


## Create explicit future
f <- future({
  cat("PID:", Sys.getpid(), "\n")
  42L
})
print(f)
v <- value(f)
print(v)



## Create explicit future
f <- future({
  cat("PID:", Sys.getpid(), "\n")
  42L
})
```

```
v <- value(f)
print(v)
```

`batchtools_local` *batchtools local and interactive futures*

Description

A batchtools local future is an synchronous uniprocess future that will be evaluated in a background R session. A batchtools interactive future is an synchronous uniprocess future that will be evaluated in the current R session (and variables will be assigned to the calling environment rather than to a local one). Both types of futures will block until the futures are resolved.

Usage

```
batchtools_local(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  workers = 1L,
  registry = list(),
  ...
)
```

Arguments

`expr`	The R expression to be evaluated
`envir`	The environment in which global environment should be located.
`substitute`	Controls whether `expr` should be `substitute()`:d or not.
`globals`	(optional) a logical, a character vector, a named list, or a Globals object. If TRUE, globals are identified by code inspection based on `expr` and `tweak` searching from environment `envir`. If FALSE, no globals are used. If a character vector, then globals are identified by lookup based their names `globals` searching from environment `envir`. If a named list or a Globals object, the globals are used as is.
`label`	(optional) Label of the future (where applicable, becomes the job name for most job schedulers).
`workers`	(optional) The maximum number of workers the batchtools backend may use at any time. Interactive and "local" backends can only process one future at the time (`workers = 1L`), whereas HPC backends, where futures are resolved via separate jobs on a scheduler, can have multiple workers. In the latter, the default is `workers = NULL`, which will resolve to `getOption("future.batchtools.workers")`. If that is not specified, the value of environment variable `R_FUTURE_BATCHTOOLS_WORKERS` will be used. If neither are specified, then the default is `100`.

`registry`	(optional) A named list of settings to control the setup of the batchtools registry.
`...`	Additional arguments passed to `BatchtoolsFuture()`.

Details

batchtools local futures rely on the batchtools backend set up by `batchtools::makeClusterFunctionsInteractive(exter` `= TRUE)` and batchtools interactive futures on the one set up by `batchtools::makeClusterFunctionsInteractive()`. These are supported by all operating systems.

An alternative to batchtools local futures is to use cluster futures of the **future** package with a single local background session, i.e. `plan(cluster,workers = "localhost")`.

An alternative to batchtools interactive futures is to use transparent futures of the **future** package.

Value

An object of class `BatchtoolsFuture`.

Examples

```
## Use local batchtools futures
plan(batchtools_local)

## A global variable
a <- 1

## Create explicit future
f <- future({
  b <- 3
  c <- 2
  a * b * c
})
v <- value(f)
print(v)


## Create implicit future
v %<-% {
  b <- 3
  c <- 2
  a * b * c
}
print(v)
```

`batchtools_template` *Batchtools futures for LSF, OpenLava, SGE, Slurm, TORQUE etc.*

Description

Batchtools futures for LSF, OpenLava, SGE, Slurm, TORQUE etc. are asynchronous multiprocess futures that will be evaluated on a compute cluster via a job scheduler.

Usage

```
batchtools_lsf(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)

batchtools_openlava(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)

batchtools_sge(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)

batchtools_slurm(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)

batchtools_torque(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
  globals = TRUE,
  label = NULL,
  template = NULL,
  resources = list(),
  workers = NULL,
  registry = list(),
  ...
)
```

Arguments

`expr`	The R expression to be evaluated
`envir`	The environment in which global environment should be located.
`substitute`	Controls whether `expr` should be `substitute()`:d or not.
`globals`	(optional) a logical, a character vector, a named list, or a Globals object. If TRUE, globals are identified by code inspection based on `expr` and `tweak` searching from environment `envir`. If FALSE, no globals are used. If a character vector, then globals are identified by lookup based their names `globals` searching from environment `envir`. If a named list or a Globals object, the globals are used as is.
`label`	(optional) Label of the future (where applicable, becomes the job name for most job schedulers).
`template`	(optional) A batchtools template file or a template string (in **brew** format). If not specified, it is left to the **batchtools** package to locate such file using its search rules.
`resources`	(optional) A named list passed to the **batchtools** template (available as variable `resources`). See Section 'Resources' in `batchtools::submitJobs()` more details.
`workers`	(optional) The maximum number of workers the batchtools backend may use at any time. Interactive and "local" backends can only process one future at the time (`workers = 1L`), whereas HPC backends, where futures are resolved via separate jobs on a scheduler, can have multiple workers. In the latter, the default is `workers = NULL`, which will resolve to `getOption("future.batchtools.workers")`. If that is not specified, the value of environment variable `R_FUTURE_BATCHTOOLS_WORKERS` will be used. If neither are specified, then the default is `100`.
`registry`	(optional) A named list of settings to control the setup of the batchtools registry.
`...`	Additional arguments passed to `BatchtoolsFuture()`.

Details

These type of batchtools futures rely on batchtools backends set up using the following **batchtools** functions:

- `batchtools::makeClusterFunctionsLSF()` for Load Sharing Facility (LSF)
- `batchtools::makeClusterFunctionsOpenLava()` for OpenLava
- `batchtools::makeClusterFunctionsSGE()` for Sun/Oracle Grid Engine (SGE)
- `batchtools::makeClusterFunctionsSlurm()` for Slurm
- `batchtools::makeClusterFunctionsTORQUE()` for TORQUE / PBS

Value

An object of class `BatchtoolsFuture`.

`future.batchtools` *future.batchtools: A Future for batchtools*

Description

The **future.batchtools** package implements the Future API on top of **batchtools** such that futures can be resolved on for instance high-performance compute (HPC) clusters via job schedulers. The Future API is defined by the **future** package.

Details

To use batchtools futures, load **future.batchtools**, and select the type of future you wish to use via `future::plan()`.

Examples

```
library(future.batchtools)

## Use local batchtools futures
plan(batchtools_local)

## A global variable
a <- 1

v %<-% {
  b <- 3
  c <- 2
  a * b * c
}

print(v)

plan(batchtools_local)
```

```
demo("mandelbrot", package = "future", ask = FALSE)
```

Index

batchtools::findConfFile(), *3*
batchtools::makeClusterFunctionsInteractive(), *5*
batchtools::makeClusterFunctionsInteractive(external = TRUE), *5*
batchtools::makeClusterFunctionsLSF(), *8*
batchtools::makeClusterFunctionsOpenLava(), *8*
batchtools::makeClusterFunctionsSGE(), *8*
batchtools::makeClusterFunctionsSlurm(), *8*
batchtools::makeClusterFunctionsTORQUE(), *8*
batchtools::submitJobs(), *2*, *7*
batchtools_custom, 2
batchtools_interactive (batchtools_local), 4
batchtools_local, 4
batchtools_lsf (batchtools_template), 5
batchtools_openlava (batchtools_template), 5
batchtools_sge (batchtools_template), 5
batchtools_slurm (batchtools_template), 5
batchtools_template, 5
batchtools_torque (batchtools_template), 5
BatchtoolsFuture(), *3*, *5*, *7*

cluster, *5*
ClusterFunctions, *3*

future.batchtools, 8
future.batchtools-package (future.batchtools), 8
future::plan(), *8*

Globals, *2*, *4*, *7*

transparent, *5*