Package 'slurmR'

April 23, 2020

Title A Lightweight Wrapper for 'Slurm'

Version 0.4-1

Description 'Slurm', Simple Linux Utility for Resource Management <https://slurm.schedmd.com/>, is a popular 'Linux' based software used to schedule jobs in 'HPC' (High Performance Computing) clusters. This R package provides a specialized lightweight wrapper of 'Slurm' with a syntax similar to that found in the 'parallel' R package. The package also includes a method for creating socket cluster objects spanning multiple nodes that can be used with the 'parallel' package.

Depends R (>= 3.3.0), parallel

License MIT + file LICENSE

BugReports https://github.com/USCbiostats/slurmR/issues

URL https://github.com/USCbiostats/slurmR, https://slurm.schedmd.com/

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Suggests rslurm, knitr, rmarkdown, covr, tinytest

Imports utils

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author George Vega Yon [aut, cre] (<https://orcid.org/0000-0002-3171-0844>),
Paul Marjoram [ctb, ths] (<https://orcid.org/0000-0003-0824-7449>),
National Cancer Institute (NCI) [fnd] (Grant Number 5P01CA196569-02),
Michael Schubert [rev] (JOSS reviewer,
<https://orcid.org/0000-0002-6862-5221>),
Michel Lang [rev] (JOSS reviewer,
<https://orcid.org/0000-0001-9754-0393>)

Maintainer George Vega Yon <g.vegayon@gmail.com>

Repository CRAN

Date/Publication 2020-04-23 00:10:02 UTC

R topics documented:

`expand_array_indexes` *Expand Array Indexes*

Description

When submitting array jobs using `sbatch`, users can specify indices in several ways. These could be specified as, for example, ranges, "`1-9`", lists, "1,2,5", or intervals as "`1-7:3`", which translates into "1, 4, 7". This function expands those cases.

Usage

```
expand_array_indexes(x)
```

Arguments

`x`	A character vector. Array indexes (see details).

Details

x is assumed to be in the form of [jobid](_[array expression]), where the expression after the underscore is optional. The first The function will return an expanded version of this, e.g. if x = "8123_[1,3-6]" the resulting expression will be the vector "8123_1", "8123_3", "8123_4", "8123_5", and "8123_6".

This function was developed mainly to be used internally.

Value

A character vector with the expanded indices.

Examples

```
expand_array_indexes(c("512", "123_1", "55_[1-5]", "122_[1, 5-6]", "44_[1-3:2]"))
# [1] "512"   "123_1" "55_1"  "55_2"  "55_3"  "55_4"  "55_5"
# "122_1" "122_5" "122_6" "44_1"  "44_3"
```

JOB_STATE_CODES *Slurm Job state codes*

Description

This data frame contains information regarding the job state codes that Slurm returns when querying the status of a given job. The last column, type, shows a description of how that corresponding state is considered in the package's various operations. This is used in the function status.

Usage

```
JOB_STATE_CODES
```

Format

A data frame with 24 rows and 4 columns.

References

Slurm's website https://slurm.schedmd.com/squeue.html

`makeSlurmCluster` *Create a Parallel Socket Cluster using Slurm*

Description

This function is essentially a wrapper of the function parallel::makePSOCKcluster. `makeSlurmCluster` main feature is adding node addresses.

Usage

```
makeSlurmCluster(
  n,
  job_name = random_job_name(),
  tmp_path = opts_slurmR$get_tmp_path(),
  cluster_opt = list(),
  max_wait = 300L,
  verb = TRUE,
  ...
)

## S3 method for class 'slurm_cluster'
stopCluster(cl)
```

Arguments

`n`	Integer scalar. Size of the cluster object (see details).
`job_name`	Character. Name of the job to be passed to `Slurm`.
`tmp_path`	Character. Path to the directory where all the data (including scripts) will be stored. Notice that this path must be accessible by all the nodes in the network (See opts_slurmR).
`cluster_opt`	A list of arguments passed to parallel::makePSOCKcluster.
`max_wait`	Integer scalar. Wait time before exiting with error while trying to read the nodes information.
`verb`	Logical scalar. If `TRUE`, the function will print messages on screen reporting on the status of the job submission.
`...`	Further arguments passed to Slurm_EvalQ via `sbatch_opt`.
`cl`	An object of class `slurm_cluster`.

Details

By default, if the `time` option is not specified via `...`, then it is set to the value `01:00:00`, this is, 1 hour.

Once a job is submitted via Slurm, the user gets access to the nodes associated with it, which allows users to star new processes within those. By means of this, we can create Socket, also known as

"PSOCK", clusters across nodes in a Slurm environment. The name of the hosts are retrieved and passed later on to parallel::makePSOCKcluster.

The method `stopCluster` for `slurm_cluster` stops the cluster doing the following:

1. Closes the connection by calling the `stopCluster` method for `PSOCK` objects.
2. Cancel the Slurm job using `scancel`.

Value

A object of class `c("slurm_cluster","SOCKcluster","cluster")`. It is the same as what is returned by parallel::makePSOCKcluster with the main difference that it has two extra attributes:

- `SLURM_JOBID` Which is the id of the Job that initialized that cluster.

Maximum number of connections

By default, R limits the number of simultaneous connections (see this thread in R-sig-hpc https://stat.ethz.ch/pipermail/r-sig-hpc/2012-May/001373.html) Current maximum is 128 (R version 3.6.1). To modify that limit, you would need to reinstall R updating the macro `NCONNECTIONS` in the file `src/main/connections.c`.

For now, if the user sets `n` above 128 it will get an immediate warning pointing to this issue, in particular, specifying that the cluster object may not be able to be created.

Examples

```
## Not run:

# Creating a cluster with 100 workers/offpring/child R sessions
cl <- makeSlurmCluster(100)

# Computing the mean of a 100 random uniforms within each worker
# for this we can use any of the function available in the parallel package.
ans <- parSapply(1:200, function(x) mean(runif(100)))

# We simply call stopCluster as we would do with any other cluster
# object
stopCluster(ans)

# We can also specify SBATCH options directly (...)
cl <- makeSlurmCluster(200, partition = "thomas", time = "02:00:00")
stopCluster(cl)


## End(Not run)
```

new_rscript *General purpose function to write R scripts*

Description

This function will create an object of class slurmR_rscript that can be used to write the R component in a batch job.

Usage

```
new_rscript(
  njobs,
  tmp_path,
  job_name,
  pkgs = list_loaded_pkgs(),
  libPaths = .libPaths()
)
```

Arguments

njobs	Integer. Number of jobs to use in the job-array. This specifies the number of R sessions to initialize. This does not specify the number of cores to be used.
tmp_path	Character. Path to the directory where all the data (including scripts) will be stored. Notice that this path must be accessible by all the nodes in the network (See opts_slurmR).
job_name	Character. Name of the job to be passed to Slurm.
pkgs	A named list with packages to be included. Each element of the list must be a path to the R library, while the names of the list are the names of the R packages to be loaded.
libPaths	A character vector. See .libPaths.

Value

An environment of class slurmR_rscript. This has the following accessible components:

- add_rds Add rds files to be loaded in each job.", x is a named list with the objects that should be loaded in the jobs. If index = TRUE the function assumes that the user will be accessing a particular subset of x during the job, which is accessed according to INDICES[[ARRAY_ID]]. The option compress is passed to saveRDS.

  One important side effect is that when this function is called, the object will be saved in the current job directory, this is opts_slurmR$get_tmp_path().
- append Adds a line to the R script. Its only argument, x is a character vector that will be added to the R script.
- rscript A character vector. This is the actual R script that will be written at the end.

- `finalize` Adds the final line of the R script. This function receives a character scalar `x` which is used as the name of the object to be saved. If missing, the function will save a NULL object. The `compress` argument is passed to saveRDS.
- `set_seed` Adds a vector of seeds to be used across the jobs. This vector of seeds should be of length `njobs`. The other two parameters of the function are passed to set.seed. By default the seed is picked as follows:

  ```
  seeds <- sample.int(.Machine$integer.max, njobs, replace = FALSE)
  ```

- `write` Finalizes the process by writing the R script in the corresponding folder to be used with Slurm.

`opts_slurmR` *Get and set default options for* `sbatch` *and* `slurmR` *internals*

Description

Most of the functions in the `slurmR` package use `tmp_path` and `job-name` options to write and submit jobs to **Slurm**. These options have global defaults that are set and retrieved using `opts_slurmR`.

Usage

```
opts_slurmR
```

Format

An object of class `opts_slurmR` of length 14.

Details

Whatever the path specified on `tmp_path`, all nodes should have access to it. Moreover, it is recommended to use a path located in a high-performing drive. See for example disk staging.

The `tmp_path` directory is only created at the time that one of the functions needs to I/O files. Job creation calls like Slurm_EvalQ and Slurm_lapply do such.

Current supported options are:

Debugging mode

- debug_on : function () Activates the debugging mode. When active, jobs will be submitted using sh and not sbatch. Also, only a single chunk of the data will be processed.
- debug_off : function () Deactivates the debugging mode.
- get_debug : function () Returns TRUE of debug mode is on

Verbose mode

- verbose_on : function () Deactivates the verbose mode. When ON, sbatch prints the Rscript and batch files on screen so that the user knows what will be submitted to Slurm.
- verbose_off : function () Deactivates the verbose mode.
- get_verbose : function () Returns TRUE if verbose mode is on.

Slurm options

- set_tmp_path : function (path, recursive = TRUE) Sets the tempfile path for I/O
- get_tmp_path : function () Retrieves tempfile path for I/O
- set_job_name : function (path, check = TRUE, overwrite = TRUE) Changes the job-name. When changing the name of the job the function will check whether the folder chdir/job-name is empty or not. If empty/not created it will create it, otherwise it will delete its contents (if 'overwrite = TRUE", else it will return with an Error)..
- get_job_name : function (check = TRUE) Returns the current value of 'job-name'.

Other options

- get_cmd : function () If debug mode is active, then it returns 'sh', otherwise 'sbatch'

For general set/retrieve options

- set_opts : function (...) A generic function to set options.
- get_opts_job : function (...) A generic function to retrieve options for the job (Slurm).
- get_opts_r : function (...) A generic function to retrieve options in R.

Examples

```
# Common setup
## Not run:
opts_slurmR$set_tmp_path("/staging/pdt/vegayon")
opts_slurmR$set_job_name("simulations-1")
opts_slurm$set_opts(partition="thomas", account="lc_pdt")

## End(Not run)
```

`parse_flags` *Utility function*

Description

Utility function

Usage

```
parse_flags(...)

## Default S3 method:
parse_flags(...)

## S3 method for class 'list'
parse_flags(x, ...)
```

Arguments

...	Options to be parsed as bash flags.
x	A named list.

Value

A character vector with the processed flags.

See Also

Other utilities: `Slurm_clean()`, `Slurm_env()`, `Slurm_log()`, `WhoAmI()`, `snames()`, `status()`

Examples

```
cat(parse_flags(a=1, b=TRUE, hola=2, y="I have spaces", ms=2, `cpus-per-task`=4))
# -a 1 -b --hola=2 -y "I have spaces" --ms=2 --cpus-per-task=4
```

---

random_job_name *Generate a random job name*

Description

Generate a random job name

Usage

```
random_job_name()
```

Value

A character scalar that can be used as job. All names will start with the prefix slurmr-job- and then some random string. This is a wrapper of the function `tempfile()` and uses as `tmpdir` argument `opts_slurmR$get_tmp_path()`.

Examples

```
random_job_name()
```

read_sbatch *Read a slurm batch file and capture the SBATCH options*

Description

Read a slurm batch file and capture the SBATCH options

Usage

```
read_sbatch(x)
```

Arguments

x	Character scalar. Either the path to the batch file to process, or a character vector.

Value

A named vector of the options starting with #SBATCH in the file. If no option is found, then returns a character vector length 0.

Examples

```
# Reading in an example script
x <- system.file("example.slurm", package="slurmR")
read_sbatch(x)
```

slurmR *A Lightweight Wrapper for 'Slurm'*

Description

A Lightweight Wrapper for 'Slurm'

Details

To cite slurmR in publications use:

Vega Yon et al., (2019). slurmR: A lightweight wrapper for HPC with Slurm. Journal of Open Source Software, 4(39), 1493, https://doi.org/10.21105/joss.01493

A BibTeX entry for LaTeX users is

```
 @Article{,
  title = {slurmR: A lightweight wrapper for HPC with Slurm},
  author = {George {Vega Yon} and Paul Marjoram},
  journal = {The Journal of Open Source Software},
  year = {2019},
  month = {jul},
  volume = {4},
  number = {39},
  doi = {10.21105/joss.01493},
  url = {https://doi.org/10.21105/joss.01493},
}
```

`slurm_available` *R wrappers for* Slurm *commands*

Description

The functions `sbatch`, `scancel`, `squeue`, `sacct`, and `slurm.conf` are wrappers of calls to Slurm functions via system2.

Usage

```
slurm_available()

squeue(x = NULL, ...)

## Default S3 method:
squeue(x = NULL, ...)

## S3 method for class 'slurm_job'
squeue(x, ...)

scancel(x = NULL, ...)

## Default S3 method:
scancel(x = NULL, ...)

## S3 method for class 'slurm_job'
scancel(x = NULL, ...)

sacct(x, ...)

## Default S3 method:
sacct(x = NULL, brief = TRUE, parsable = TRUE, allocations = TRUE, ...)

## S3 method for class 'slurm_job'
sacct(x, ...)
```

```
slurm.conf()

SchedulerParameters()

sbatch(x, wait = FALSE, submit = TRUE, ...)

## S3 method for class 'slurm_job'
sbatch(x, wait = FALSE, submit = TRUE, ...)

## S3 method for class 'character'
sbatch(x, wait = FALSE, submit = TRUE, ...)
```

Arguments

x	Either an object of class slurm_job, or, in some cases, an integer as a Slurm jobid. Note that some functions allow passing no arguments.
...	Further flags passed to the command line function.
brief, parsable, allocations	Logical. When TRUE, these are passed as flags directly to the command line function sacct.
wait	Logical scalar. When TRUE the function will pass the --wait flag to Slurm and set wait = TRUE in the system2 call.
submit	Logical, when TRUE calls sbatch to submit the job to slurm.

Details

The function slurm_available checks whether Slurm is available in the system or not. It is usually called before calling any bash wrapper. If available, the function will return TRUE, otherwise FALSE.

The wrapper of squeue includes the flag -o%all which returns all available fields separated by a vertical bar. This cannot be changed since it is the easiest way of processing the information in R.

The function slurm.conf is a wrapper of the function scontrol that returns configuration info about Slurm, in particular, the underlying command that is called is scontrol show conf. This returns a named character vector with configuration info about the cluster. The name of this function matches the name of the file that holds this information.

The function SchedulerParameters is just a wrapper of slurm.conf. It processes the field "SchedulerParameters" included in the configuration file and has information relevant for the scheduler.

In the case of sbatch, function takes an object of class slurm_job and submits it to the queue. In debug mode the job will be submitted via sh instead.

The method for character scalars is used to submit jobs using a slurm script.

Value

In the case of sbatch, depends on what x is:

- If x is of class slurm_job, then it returns the same object including the Slurm job ID (if the job was submitted to the queue).

- If x is a file path (e.g. a bash script), an integer with the jobid number (again, if the job was submitted to Slurm).

The functions squeue and sacct return a data frame with the information returned by the command line utilities. The function scancel returns NULL.

slurm_available() returns a logical scalar equal to TRUE if Slurm is available.

slurm.conf() and SchedulerParameters() return information about the Slurm cluster, if available.

Examples

```
# Are we under a Slurm Cluster?
slurm_available()


## Not run:
# What is the maximum number of jobs (array size) that the system
# allows?
sconfig <- slurm.conf() # We first retrieve the info.
sconfig["MaxArraySize"]

## End(Not run)

## Not run:
# Submitting a simple job
job <- Slurm_EvalQ(slurmR::WhoAmI(), njobs = 4L, plan = "submit")

# Checking the status of the job (we can simply print)
job
status(job) # or use the state function
sacct(job) # or get more info with the sactt wrapper.

# Suppose one of the jobs is taking too long to complete (say #4)
# we can stop it and resubmit the job as follows:
scancel(job)

# Resubmitting only 4
sbatch(job, array = 4) # A new jobid will be assigned


## End(Not run)
```

Slurm_clean *Clean a session.*

Description

The functions of the family Slurm_*apply generate a set of temporary files that are used for the job design, submission and collection. This function will remove all the contents of directory created by calling those functions.

Usage

```
Slurm_clean(x)
```

Arguments

x	An object of class slurm_job.

Details

If the job is finalized, it returns 0 if able to clean the directory otherwise return whatever unlink returns after trying to remove the job path.

See Also

Other post submission: Slurm_collect(), Slurm_log(), status()

Other utilities: Slurm_env(), Slurm_log(), WhoAmI(), parse_flags(), snames(), status()

Examples

```
## Not run:

job <- Slurm_EvalQ(1 + 1, 2, plan = "collect")

# This will remove all the files generated by Slurm_EvalQ
Slurm_clean(job)


## End(Not run)
```

Slurm_collect *Collect the results of a slurm job*

Description

This function takes an object of class slurm_job and retrieves the results, this is, combines the R objects generated by each job. Object of class slurm_job.

Usage

```
Slurm_collect(...)

## S3 method for class 'slurm_job'
Slurm_collect(x, any. = FALSE, wait = 10L, ...)
```

Arguments

...	Further arguments passed to the method.
x	An object of class slurm_job.
any.	Logical. When TRUE, will collect any output available regardless of whether the job is completed or not.
wait	Integer scalar. Number of seconds to wait before checking the state of a job if the first try returned -1 (no job found).

Details

If the given job has hooks, which is a list of functions, these will be applied sequentially to the set of retrieved results before returning.

Value

By default, it returns a concatenated list of the output files generated by each job. If the job object has a hook, it will apply each hook to the full list before returning. See new_slurm_job.

See Also

Other post submission: Slurm_clean(), Slurm_log(), status()

Examples

```
## Not run:
# Collecting a job after calling it
job <- Slurm_EvalQ(slurmR::WhoAmI(), njobs = 4, plan = "wait")
Slurm_collect(job)

# Collecting a job from a previous R session
job <- read_slurm_job("/path/to/a/job/tmp_dir")
Slurm_collect(job)

## End(Not run)
```

Slurm_env *A wrapper of Sys.getenv*

Description

This function is used within the R script written by slurmR to get the current value of SLURM_ARRAY_TASK_ID, an environment variable that Slurm creates when running an array. In the case that opts_slurmR$get_debug() == TRUE, the function will return a 1 (see opts_slurmR).

Usage

```
Slurm_env(x = "SLURM_ARRAY_TASK_ID")
```

Arguments

x	Character scalar. Environment variable to get.

Value

If slurm is available and the R session is running under a job array, meaning that SLURM_ARRAY_TASK_ID is defined, then it returns that value, otherwise it will return 1.

See Also

Other utilities: Slurm_clean(), Slurm_log(), WhoAmI(), parse_flags(), snames(), status()

Slurm_EvalQ *Submit an expression to be evaluated to multiple jobs.*

Description

Submit an expression to be evaluated to multiple jobs.

Usage

```
Slurm_EvalQ(
  expr,
  njobs = 2L,
  job_name = opts_slurmR$get_job_name(),
  tmp_path = opts_slurmR$get_tmp_path(),
  plan = "collect",
  sbatch_opt = list(),
  rscript_opt = list(),
  seeds = NULL,
  compress = TRUE,
  export = NULL,
  export_env = NULL,
  libPaths = .libPaths(),
  hooks = NULL,
  overwrite = TRUE
)
```

Arguments

expr	An expression to be passed to Slurm.
njobs	Integer. Number of jobs to use in the job-array. This specifies the number of R sessions to initialize. This does not specify the number of cores to be used.
job_name	Character. Name of the job to be passed to Slurm.
tmp_path	Character. Path to the directory where all the data (including scripts) will be stored. Notice that this path must be accessible by all the nodes in the network (See opts_slurmR).

plan	A character scalar. (See the_plan).
sbatch_opt	List of options to be passed to sbatch. This is usually done by adding the flags #SBATCH to the bash file.
rscript_opt	List. Options to be passed to Rscript.
seeds	Integer vector of length njobs. Seeds to be passed to each job. When NULL (default), seeds will be picked automatically (see new_rscript).
compress	Logical scalar (default TRUE). Passed to saveRDS. Setting this value to FALSE can be useful when the user requires faster read/write of R objects on disk.
export	A named list with objects to be included in the Spawned sessions.
export_env	An environment. Environment where the objects listed in export are located (default parent.frame()).
libPaths	A character vector. See .libPaths.
hooks	A list of functions (passed to new_slurm_job).
overwrite	Logical scalar. When TRUE, if the path specified by tmp_path/job_name already exists, it will overwrite it, otherwise the function returns with an error.

Value

A list of length njobs.

---

slurm_job *Creating Slurm jobs*

Description

Utilities to deal with objects of class slurm_job. The function new_slurm_job, which is mostly intended to be for internal used, creates an object of class slurm_job. The function last_submitted_job returns the last submitted job in the current R session, and the functions read/write_slurm_job are utility functions to read and write R jobs respectively.

Usage

```
new_slurm_job(
  call,
  rscript,
  bashfile,
  robjects,
  njobs,
  opts_job,
  opts_r,
  hooks = NULL
)

## S3 method for class 'slurm_job'
```

```
print(x, ...)

read_slurm_job(path)

write_slurm_job(x, path = NULL)

last_submitted_job()

last_job()
```

Arguments

call	The original call
rscript, bashfile	The R script and bash file path.
robjects	A character vector of R objects that will be imported in the job.
njobs	Integer. Number of jobs to start (array).
opts_job, opts_r	List. In the case of opts_job, a list of parameters passed to sbatch. opts_r is a list of parameters used within R. Both can be retrieved by opts_slurmR$get_opts_job() and opts_slurmR$get_opts_r() respectively.
hooks	List of functions. To be called on the collected results after it finalizes.
x	An object of class slurm_job.
...	Further arguments passed to the method.
path	Character scalar. Path to either a directory with a job.rds file, or directly to a job.rds file.

Details

In the case of the function new_slurm_job, besides of creating the object of class slurm_job, the function calls write_slurm_job and stores the job object in an rds class file. The name and location of the saved rds file is generated using the function snames("job").

The read_slurm_job can help the user recovering a previously saved slurm_job object. If path is a directory, then the function will assume that the file that is looking for lives within that directory and is named job.rds. Otherwise, if a file, then it will read it directly. In any case, it will check that the read object is an object of class slurm_job.

The write_slurm_job function simply takes a slurm_job object and saves it in, if path is not specified, whatever the job$options$chdir folder is under the name job.rds. If a path is specified, the it is directly passed to saveRDS().

The las_submitted_job function will return the latest slurm_job object that was submitted via sbatch in the current session. The last_job function is just an alias of the later. If no job has been submitted, then the resulting value will be NULL.

Value

An environment of class `slurm_job`. This has the following items:

- `call` The original call (Slurm_lapply, Slurm_Map, etc.)
- `rscript` The full path to the R script to be executed by bash file.
- `bashfile` The full path to the bash file to be executed by sbatch.
- `robjects` Ignored.
- `njobs` The number of jobs to be submitted (job array).
- `opts_job`,`opts_r` Two lists of options as returned by opts_slurmR$get_opts_job() and opts_slurmR$get_r_opts() at the moment of the creation of the `slurm_job`.
- `hooks` A list of functions to be called on the collected objects by Slurm_collect.

In the case of the function `write_slurm_job`, it returns the full path to the file.

Examples

```
## Not run:
# The last_job function can be handy when `plan = "collect"` in a called,
# for example
job <- Slurm_lapply(1:1000, function(i) runif(100), njobs = 2, plan = "collect")

# Post collection analysis
status(last_job())

## End(Not run)
```

Slurm_log *Check the R logfile of a job.*

Description

After submission, the functions of type Slurm_*apply generate log files, one per each job in the job array. The `Slurm_log` function can be used to check the log files of jobs in the array that failed.

Usage

```
Slurm_log(x, which. = NULL, cmd = NULL)
```

Arguments

`x`	An object of class slurm_job.
`which.`	An integer scalar. The number of the array job to check. This should range between 1 and `x$njobs`.
`cmd`	Character scalar. The name of the command to use to call view the log file. Default to `less` when interactive mode, otherwise `cat` (see details).

Details

If other than `less` is used, then the function will try to check by calling `cmd --version`. If returns with error, it assumes the function is not available. Using the `cmd` argument only works in interactive mode.

Value

Whatever the command-line call returns.

See Also

Other post submission: `Slurm_clean()`, `Slurm_collect()`, `status()`

Other utilities: `Slurm_clean()`, `Slurm_env()`, `WhoAmI()`, `parse_flags()`, `snames()`, `status()`

Examples

```
## Not run:
x <- Slurm_EvalQ(slurmR::whoami(), plan = "wait")
Slurm_log(x) # Checking the R log

## End(Not run)
```

Slurm_Map *The Slurm version of the `*apply` family of functions.*

Description

The Slurm version of the `*apply` family of functions.

Usage

```
Slurm_Map(
  f,
  ...,
  njobs = 2L,
  mc.cores = 1L,
  job_name = opts_slurmR$get_job_name(),
  tmp_path = opts_slurmR$get_tmp_path(),
  plan = "collect",
  sbatch_opt = list(),
  rscript_opt = list(),
  seeds = NULL,
  compress = TRUE,
  export = NULL,
  export_env = NULL,
  libPaths = .libPaths(),
  hooks = NULL,
```

```
  overwrite = TRUE
)

Slurm_lapply(
  X,
  FUN,
  ...,
  njobs = 2L,
  mc.cores = 1L,
  job_name = opts_slurmR$get_job_name(),
  tmp_path = opts_slurmR$get_tmp_path(),
  plan = "collect",
  sbatch_opt = list(),
  rscript_opt = list(),
  seeds = NULL,
  compress = TRUE,
  export = NULL,
  export_env = NULL,
  libPaths = .libPaths(),
  hooks = NULL,
  overwrite = TRUE
)

Slurm_sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
```

Arguments

njobs	Integer. Number of jobs to use in the job-array. This specifies the number of R sessions to initialize. This does not specify the number of cores to be used.
job_name	Character. Name of the job to be passed to Slurm.
tmp_path	Character. Path to the directory where all the data (including scripts) will be stored. Notice that this path must be accessible by all the nodes in the network (See opts_slurmR).
plan	A character scalar. (See the_plan).
sbatch_opt	List of options to be passed to sbatch. This is usually done by adding the flags #SBATCH to the bash file.
rscript_opt	List. Options to be passed to Rscript.
seeds	Integer vector of length njobs. Seeds to be passed to each job. When NULL (default), seeds will be picked automatically (see new_rscript).
compress	Logical scalar (default TRUE). Passed to saveRDS. Setting this value to FALSE can be useful when the user requires faster read/write of R objects on disk.
export	A named list with objects to be included in the Spawned sessions.
export_env	An environment. Environment where the objects listed in export are located (default parent.frame()).
libPaths	A character vector. See .libPaths.
hooks	A list of functions (passed to new_slurm_job).

`overwrite` Logical scalar. When `TRUE`, if the path specified by `tmp_path/job_name` already exists, it will overwrite it, otherwise the function returns with an error.

`X, FUN, f, mc.cores, ...`
Arguments passed to either parallel::mclapply or parallel::mcMap.

`simplify, USE.NAMES`
Logical scalar. See sapply.

Details

The function `Slurm_lapply` will submit `njobs` to the queue and distribute `X` according to parallel::splitIndices. For example, if `X` is list with 1,000 elements, and `njobs = 2`, then `Slurm_lapply` will submit 2 jobs with 500 elements of `X` each (2 chunks of data). The same principle applies to `Slurm_sapply` and `Slurm_Map`, this is, the data is split by chunks so all the information is sent at once when the job is submitted.

Just like sapply is to lapply, `Slurm_sapply` is just a wrapper of `Slurm_lapply` with an extra argument, `simplify`. When `TRUE`, once the job is collected, the function simplify2array is called.

Value

If `plan == "collect"`, then whatever the analogous function returns, otherwise, an object of class slurm_job.

References

Job Array Support https://slurm.schedmd.com/job_array.html

See Also

For resubmitting a job, see the example in sbatch.

Examples

```
## Not run:
  # A job drawing 1e6 uniforms on 10 jobs (array)
  # The option plan = "wait" makes it return only once the job is completed.
  job1 <- Slurm_lapply(1:20, function(i) runif(1e6), njobs=10, plan = "wait")

  # To collect
  ans <- Slurm_collect(job1)

  # As before, but this time not waiting, and now we are passing more
  # arguments to the function
  # plan = "none" only creates the job object (and the files), we submit
  # later
  job1 <- Slurm_lapply(1:20, function(i, a) runif(1e6, a), a = -1, njobs=10,
      plan = "none")

  # We submit
  job1 <- sbatch(job1)
```

```
  # In order to cancel a job
  scancel(job1)

  # How to clean up
  Slurm_clean(job1)

## End(Not run)
```

`snames` *Full path names for Slurm jobs*

Description

Using opts_slurmR$get_tmp_path and opts_slurmR$get_job_name creates file names with full path to the objects. This function is intended for internal use only.

Usage

```
snames(type, array_id = NULL, tmp_path = NULL, job_name = NULL)
```

Arguments

`type`	can be any of r, sh, out, or rds.
`array_id`	Integer. ID of the array to create the name.
`tmp_path`	Character scalar. Path to the temp directory used by the job to write files.
`job_name`	Character scalar. Name of the job.

Details

By default, the parameters `tmp_path` and `job_name` are retrieved from the current options specified in opts_slurmR.

Value

A character scalar. The normalized path to the corresponding file.

See Also

Other utilities: `Slurm_clean()`, `Slurm_env()`, `Slurm_log()`, `WhoAmI()`, `parse_flags()`, `status()`

`sourceSlurm` *Source an R script as a Slurm job*

Description

This function sources R scripts using Slurm by creating a batch script file and submitting it via sbatch.

Usage

```
sourceSlurm(
  file,
  job_name = NULL,
  tmp_path = tempdir(),
  rscript_opt = list(vanilla = TRUE),
  plan = "submit",
  ...
)

slurmr_cmd(
  cmd_path,
  cmd_name = "slurmr",
  add_alias = TRUE,
  bashrc_path = "~/.bashrc"
)
```

Arguments

`file`	Character. Path to the R script to source using Slurm.
`job_name`	Character. Name of the job to be passed to `Slurm`.
`tmp_path`	Character. Path to the directory where all the data (including scripts) will be stored. Notice that this path must be accessible by all the nodes in the network (See opts_slurmR).
`rscript_opt`	List. Options to be passed to `Rscript`.
`plan`	A character scalar. (See the_plan).
`...`	Further options passed to sbatch.
`cmd_path`	Character scalar. Path (directory) where to put the command function. This is usually your home directory.
`cmd_name`	Character scalar. Name of the command (of the file).
`add_alias, bashrc_path`	Logical scalar and character scalar. When `add_alias=TRUE` it will modify (or create, if non-existent) the `.bashrc` file to add an alias of the same name of `cmd_name`. The path to `.bashrc` can be specified via the `bashrc_path` option.

Details

sourceSlurm checks for flags that may be included in the Slurm job file. If the R script starts with #!/bin/ or similar, then #SBATCH flags will be read from the R script and added to the Slurm job file.

The function slurmr_cmd writes a simple command that works as a wrapper of sourceSlurm. In particular, from command line, if the user wants to source an R script using sourceSlurm, we can either:

```
$ Rscript -e "slurmR::sourceSlurm('path/to/the/script.R', plan = 'submit')"
```

Or, after calling slurmr_cmd from within R, do the following instead

```
$ ./slurmr path/to/the/script.R
```

And, if you used the option add_alias = TRUE, then, after restarting bash, you can run R scripts with Slurm as follows:

```
$ slurmr path/to/the/script.R
```

The main side effect of this function is that it creates a file named cmd_name in the directory specified by cmd_path, and, if add_alias = TRUE. it will create (if not found) or modify (if found) the .bashrc file adding a line with an alias. For more information on .bashrc see here.

Value

In the case of sourceSlurm, Whatever sbatch returns.

The function slurmr_cmd returns invisible().

Examples

```
# In this example we will be sourcing an R script that also has #SBATCH
# flags. Here are the contents
file <- system.file("example.R", package="slurmR")

cat(readLines(file), sep="\n")
# #!/bin/sh
# #SBATCH --account=lc_ggv
# #SBATCH --time=01:00:00
# #SBATCH --mem-per-cpu=4G
# #SBATCH --job-name=Waiting
# Sys.sleep(10)
# message("done.")

# We can directly submit this R script as a job by calling `sourceSlurm`.
# (of course you need Slurm to do this!)
## Not run:
sourceSlurm(file)

## End(Not run)
```

```
# The function will create a bash script that is used later to be submitted to
# the queue using `sbatch`. The resulting file looks something like this
# #!/bin/sh
# #SBATCH --job-name=Waiting
# #SBATCH --output=/home/vegayon/Documents/slurmR/Waiting.out
# #SBATCH --account=lc_ggv
# #SBATCH --time=01:00:00
# #SBATCH --mem-per-cpu=4G
# /usr/lib/R/bin/Rscript --vanilla /usr/local/lib/R/site-library/slurmR/example.R
```

`status` *Check the status of a Slurm JOB*

Description

Using the sacct function, it checks the status of a particular job and returns information about its current state, with details regarding the jobs (if an array) that are done, running, pending, or failed.

Usage

```
status(x)

## S3 method for class 'slurm_job'
status(x)

## Default S3 method:
status(x)

## S3 method for class 'slurm_status'
x$name
```

Arguments

`x`	Either a Job id, an object of class `slurm_job`, or an object of class `slurm_status`.
`name`	Character scalar. List of status to retrieve. This can be any of `"done"`, `"failed"`, `"running"`, or `"pending"`.

Value

An integer with attributes of class `slurm_status`. The attributes are integer vectors indicating which jobs fail in the categories of done, `failed`, `pending`, and `running` (see JOB_STATE_CODES). Possible return values are:

- `-1`: No job found. This may be a false negative as the job may still be on it's way to be submitted.
- `0`: Job completed.
- `1`: All jobs are pending resource allocation or are on it's way to start.

- 2: All jobs are currently running.
- 3: One or more jobs are still running.
- 99: One or more jobs failed.

If the job is not an array, then function will return the corresponding code but the attributes will only have a single number, 1, according to the state of the job (completed, failed, pending).

See Also

Other utilities: Slurm_clean(), Slurm_env(), Slurm_log(), WhoAmI(), parse_flags(), snames()

Other post submission: Slurm_clean(), Slurm_collect(), Slurm_log()

Examples

```
## Not run:

x <- Slurm_EvalQ(Sys.sleep(100), njobs = 2)

status(x) # A possible result: An integer with attributes
# Status: All jobs are pending resource allocation or are on it's way to start. (Code 1)
# This is a job array. The status of each job, by array id, is the following:
# done      :  -
# failed    :  -
# pending   : 1, 2.
# running   :  -


## End(Not run)
```

---

the_plan *Check for possible actions for a* slurm_job *wrapper*

Description

Users can choose whether to submit the job or not, to wait for it, and whether they want to collect the results right away after the job has finished. This function will help developers to figure out what set of actions need to be taken depending on the plan.

Usage

```
the_plan(plan)
```

Arguments

plan	A character scalar with either of the following values: "collect", "wait", "submit", or "none".

Details

This is a helper function that returns a list with three logical values: `wait`, `collect`, and `submit`. There are four possible cases:

- `plan == "collect"`, then all three are `TRUE`.
- `plan == "wait"`, then all but `collect` are `TRUE`.
- `plan == "submit"` then only `submit` equals `TRUE`.
- `plan == "none"` then all three are `FALSE`.

In general, bot `wait` and `submit` will be passed to sbatch.

When `collect == TRUE`, then it usually means that the function will be calling Slurm_collect right after submitting the job via sbatch.

Value

A list with three logical scalars.

See Also

This is used in apply functions and in Slurm_EvalQ.

Examples

```
the_plan("none")
# $collect
# [1] FALSE
#
# $wait
# [1] FALSE
#
# $submit
# [1] FALSE

the_plan("wait")
# $collect
# [1] FALSE
#
# $wait
# [1] TRUE
#
# $submit
# [1] TRUE
```

`wait_slurm` *Wait for a Slurm job to be completed*

Description

Wait for a Slurm job to be completed

Usage

```
wait_slurm(x, ...)

## S3 method for class 'slurm_job'
wait_slurm(x, ...)

## S3 method for class 'integer'
wait_slurm(x, timeout = -1, freq = 0.1, force = TRUE, ...)
```

Arguments

`x`	Either a job id number, or an object of class slurm_job.
`...`	Further arguments passed to the method
`timeout`	Integer. Maximum wait time in seconds. If `timeout < 0` then the command will only return when the job finishes.
`freq`	Frequency in seconds to query for the state of the job.
`force`	Logical scalar. When `TRUE`, if the job is not found after checking for its status, the function will continue to wait still.

Value

Invisible `NULL`.

Examples

```
# Waiting is only available if there are Slurm clusters
if (slurm_available()) {
  job <- Slurm_EvalQ(Sys.sleep(1000), plan = "submit", njobs = 2)
  wait_slurm(job, timeout = 1) # This will return a warning
  scancel(job)
  Slurm_clean(job)
}
```

WhoAmI *Information about where jobs are submitted*

Description

This returns a named vector with the following variables: SLURM_LOCALID, SLURMD_NODENAME, SLURM_ARRAY_TASK_ID, SLURM_CLUSTER_NAME, SLURM_JOB_PARTITION, SLURM_TASK_PID

Usage

```
WhoAmI()

whoami()
```

Details

`whoami` is just an alias of `WhoAmI`.

Value

A character vector with the corresponding system environment variables' values.

See Also

Other utilities: `Slurm_clean()`, `Slurm_env()`, `Slurm_log()`, `parse_flags()`, `snames()`, `status()`

Index