Package: rslurm (via r-universe)

September 17, 2024

Type Package

Title Submit R Calculations to a 'Slurm' Cluster

Description Functions that simplify submitting R scripts to a 'Slurm' workload manager, in part by automating the division of embarrassingly parallel calculations across cluster nodes.

Acknowledgements Development of this R package was supported by the National Socio-Environmental Synthesis Center (SESYNC) under funding received from the National Science Foundation grants DBI-1052875 and DBI-1639145.

Version 0.6.2

Date 2023-01-31

License GPL-3

URL https://www.earthdatascience.org/rslurm/

BugReports https://github.com/earthlab/rslurm/issues

Depends R (>= 3.5.0)

Imports whisker (>= 0.3)

RoxygenNote 7.2.3

Suggests parallel, testthat, knitr, rmarkdown, markdown

VignetteBuilder knitr

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

RemoteUrl https://github.com/earthlab/rslurm

RemoteRef HEAD

RemoteSha fa2b82824301316e3d212993973783f967d96e62

Contents

`rslurm-package` *Introduction to the* `rslurm` *Package*

Description

Send long-running or parallel jobs to a Slurm workload manager (i.e. cluster) using the `slurm_call`, `slurm_apply`, or `slurm_map` functions.

Job submission

This package includes three core functions used to send computations to a Slurm cluster: 1) `slurm_call` executes a function using a single set of parameters (passed as a list), 2) `slurm_apply` evaluates a function in parallel for each row of parameters in a given data frame, and 3) `slurm_map` evaluates a function in parallel for each element of a list. The functions `slurm_apply` and `slurm_map` automatically split the parameter rows or list elements into equal-size chunks, each chunk to be processed by a separate cluster node. They use functions from the `parallel-package` package to parallelize computations across processors on a given node.

The output of `slurm_apply`, `slurm_map`, or `slurm_call` is a `slurm_job` object that serves as an input to the other functions in the package: `print_job_status`, `cancel_slurm`, `get_slurm_out` and `cleanup_files`.

Function specification

To be compatible with `slurm_apply`, a function may accept any number of single value parameters. The names of these parameters must match the column names of the `params` data frame supplied. There are no restrictions on the types of parameters passed as a list to `slurm_call` or `slurm_map`

If the function passed to `slurm_call` or `slurm_apply` requires knowledge of any R objects (data, custom helper functions) besides `params`, a character vector corresponding to their names should be passed to the optional `global_objects` argument.

When parallelizing a function, since any error will interrupt all calculations for the current node, it may be useful to wrap expressions which may generate errors into a `try` or `tryCatch` function. This will ensure the computation continues with the next parameter set after reporting the error.

Output Format

The default output format for `get_slurm_out` (`outtype = "raw"`) is a list where each element is the return value of one function call. If the function passed to `slurm_apply` produces a vector output, you may use `outtype = "table"` to collect the output in a single data frame, with one row by function call.

Slurm Configuration

Advanced options for the Slurm workload manager may accompany job submission by `slurm_call`, `slurm_map`, and `slurm_apply` through the optional `slurm_options` argument. For example, passing `list(time = '1:30:00')` for this options limits the job to 1 hour and 30 minutes. Some advanced configuration must be set through environment variables. On a multi-cluster head node, for example, the `SLURM_CLUSTERS` environment variable must be set to direct jobs to a non-default cluster.

Examples

```
## Not run:
# Create a data frame of mean/sd values for normal distributions
pars <- data.frame(par_m = seq(-10, 10, length.out = 1000),
                   par_sd = seq(0.1, 10, length.out = 1000))

# Create a function to parallelize
ftest <- function(par_m, par_sd) {
 samp <- rnorm(10^7, par_m, par_sd)
 c(s_m = mean(samp), s_sd = sd(samp))
}

sjob1 <- slurm_apply(ftest, pars)
print_job_status(sjob1)
res <- get_slurm_out(sjob1, "table")
all.equal(pars, res) # Confirm correct output
cleanup_files(sjob1)

## End(Not run)
```

`cancel_slurm` *Cancels a scheduled Slurm job*

Description

This function cancels the specified Slurm job by invoking the Slurm `scancel` command. It does *not* delete the temporary files (e.g. scripts) created by `slurm_apply` or `slurm_call`. Use `cleanup_files` to remove those files.

Usage

```
cancel_slurm(slr_job)
```

Arguments

`slr_job`	A `slurm_job` object.

See Also

`cleanup_files`

cleanup_files *Deletes temporary files associated with a Slurm job*

Description

This function deletes all temporary files associated with the specified Slurm job, including files created by `slurm_apply` or `slurm_call`, as well as outputs from the cluster. These files should be located in the *_rslurm_[jobname]* folder of the current working directory.

Usage

```
cleanup_files(slr_job, wait = TRUE)
```

Arguments

`slr_job`	A `slurm_job` object.
`wait`	Specify whether to block until `slr_job` completes.

See Also

`slurm_apply`, `slurm_call`

Examples

```
## Not run:
sjob <- slurm_apply(func, pars)
print_job_status(sjob) # Prints console/error output once job is completed.
func_result <- get_slurm_out(sjob, "table") # Loads output data into R.
cleanup_files(sjob)

## End(Not run)
```

get_job_status *Get the status of a Slurm job*

Description

This function returns the completion status of a Slurm job, its queue status if any and log outputs.

Usage

```
get_job_status(slr_job)
```

Arguments

`slr_job`	A `slurm_job` object.

Details

The queue element of the output is a data frame matching the output of the Slurm squeue command for that job; it will only indicate portions of job that are running or in queue. The log element is a vector of the contents of console/error output files for each node where the job is running.

Value

A list with three elements: completed is a logical value indicating if all portions of the job have completed or stopped, queue contains the information on job elements still in queue, and log contains the console/error logs.

---

get_slurm_out *Reads the output of a function calculated on the Slurm cluster*

Description

This function reads all function output files (one by cluster node used) from the specified Slurm job and returns the result in a single data frame (if "table" format selected) or list (if "raw" format selected). It doesn't record any messages (including warnings or errors) output to the R console during the computation; these can be consulted by invoking print_job_status.

Usage

```
get_slurm_out(slr_job, outtype = "raw", wait = TRUE, ncores = NULL)
```

Arguments

slr_job	A slurm_job object.
outtype	Can be "table" or "raw", see "Value" below for details.
wait	Specify whether to block until slr_job completes.
ncores	(optional) If not null, the number of cores passed to mclapply

Details

The outtype option is only relevant for jobs submitted with slurm_apply. Jobs sent with slurm_call only return a single object, and setting outtype = "table" creates an error in that case.

Value

If outtype = "table": A data frame with one column by return value of the function passed to slurm_apply, where each row is the output of the corresponding row in the params data frame passed to slurm_apply.

If outtype = "raw": A list where each element is the output of the function passed to slurm_apply for the corresponding row in the params data frame passed to slurm_apply.

See Also

slurm_apply, slurm_call

local_slurm_array *Execute a Slurm job locally*

Description

Run a previously created `slurm_job` object locally instead of on a Slurm cluster

Usage

```
local_slurm_array(slr_job, rscript_path = NULL)
```

Arguments

`slr_job`	An object of class `slurm_job`.
`rscript_path`	The location of the Rscript command. If not specified, defaults to the location of Rscript within the R installation being run.

Details

This function is most useful for testing your function on a reduced dataset before submitting the full job to the Slurm cluster.

Call `local_slurm_array` on a `slurm_job` object created with `slurm_apply(..., submit = FALSE)` or `slurm_map(..., submit = FALSE)`. The job will run serially on the local system rather than being submitted to the Slurm cluster.

Examples

```
## Not run:
sjob <- slurm_apply(func, pars, submit = FALSE)
local_slurm_array(sjob)
func_result <- get_slurm_out(sjob, "table") # Loads output data into R.
cleanup_files(sjob)

## End(Not run)
```

slurm_apply *Parallel execution of a function on the Slurm cluster*

Description

Use `slurm_apply` to compute function over multiple sets of parameters in parallel, spread across multiple nodes of a Slurm cluster, with similar syntax to `mapply`.

Usage

```
slurm_apply(
  f,
  params,
  ...,
  jobname = NA,
  nodes = 2,
  cpus_per_node = 2,
  processes_per_node = cpus_per_node,
  preschedule_cores = TRUE,
  job_array_task_limit = NULL,
  global_objects = NULL,
  add_objects = NULL,
  pkgs = rev(.packages()),
  libPaths = NULL,
  rscript_path = NULL,
  r_template = NULL,
  sh_template = NULL,
  slurm_options = list(),
  submit = TRUE
)
```

Arguments

f	A function that accepts one or many single values as parameters and may return any type of R object.
params	A data frame of parameter values to apply f to. Each column corresponds to a parameter of f (*Note*: names must match) and each row corresponds to a separate function call.
...	Additional arguments to f. These arguments do not vary with each call to f.
jobname	The name of the Slurm job; if NA, it is assigned a random name of the form "slr####".
nodes	The (maximum) number of cluster nodes to spread the calculation over. slurm_apply automatically divides params in chunks of approximately equal size to send to each node. Less nodes are allocated if the parameter set is too small to use all CPUs on the requested nodes.
cpus_per_node	The number of CPUs requested per node. This argument is mapped to the Slurm parameter cpus-per-task.
processes_per_node	The number of logical CPUs to utilize per node, i.e. how many processes to run in parallel per node. This can exceed cpus_per_node for nodes which support hyperthreading. Defaults to processes_per_node = cpus_per_node.
preschedule_cores	Corresponds to the mc.preschedule argument of parallel::mcmapply. Defaults to TRUE. If TRUE, the rows of params are assigned to cores before computation. If FALSE, each row of params is executed by the next available core.

Setting FALSE may be faster if different values of params result in very variable completion time for jobs.

job_array_task_limit
: The maximum number of job array tasks to run at the same time. Defaults to NULL (no limit).

global_objects
: A character vector containing the name of R objects to be saved in a .RData file and loaded on each cluster node prior to calling f.

add_objects
: Older deprecated name of global_objects, retained for backwards compatibility.

pkgs
: A character vector containing the names of packages that must be loaded on each cluster node. By default, it includes all packages loaded by the user when slurm_apply is called.

libPaths
: A character vector describing the location of additional R library trees to search through, or NULL. The default value of NULL corresponds to libraries returned by .libPaths() on a cluster node. Non-existent library trees are silently ignored.

rscript_path
: The location of the Rscript command. If not specified, defaults to the location of Rscript within the R installation being run.

r_template
: The path to the template file for the R script run on each node. If NULL, uses the default template "rslurm/templates/slurm_run_R.txt".

sh_template
: The path to the template file for the sbatch submission script. If NULL, uses the default template "rslurm/templates/submit_sh.txt".

slurm_options
: A named list of options recognized by sbatch; see Details below for more information.

submit
: Whether or not to submit the job to the cluster with sbatch; see Details below for more information.

Details

This function creates a temporary folder ("_rslurm_[jobname]") in the current directory, holding .RData and .RDS data files, the R script to run and the Bash submission script generated for the Slurm job.

The set of input parameters is divided in equal chunks sent to each node, and f is evaluated in parallel within each node using functions from the parallel R package. The names of any other R objects (besides params) that f needs to access should be included in global_objects or passed as additional arguments through ....

Use slurm_options to set any option recognized by sbatch, e.g. slurm_options = list(time = "1:00:00", share = TRUE). See http://slurm.schedmd.com/sbatch.html for details on possible options. Note that full names must be used (e.g. "time" rather than "t") and that flags (such as "share") must be specified as TRUE. The "array", "job-name", "nodes", "cpus-per-task" and "output" options are already determined by slurm_apply and should not be manually set.

When processing the computation job, the Slurm cluster will output two types of files in the temporary folder: those containing the return values of the function for each subset of parameters ("results_[node_id].RDS") and those containing any console or error output produced by R on each node ("slurm_[node_id].out").

If submit = TRUE, the job is sent to the cluster and a confirmation message (or error) is output to the console. If submit = FALSE, a message indicates the location of the saved data and script files; the job can be submitted manually by running the shell command sbatch submit.sh from that directory.

After sending the job to the Slurm cluster, slurm_apply returns a slurm_job object which can be used to cancel the job, get the job status or output, and delete the temporary files associated with it. See the description of the related functions for more details.

Value

A slurm_job object containing the jobname and the number of nodes effectively used.

See Also

slurm_call to evaluate a single function call.

slurm_map to evaluate a function over a list.

cancel_slurm, cleanup_files, get_slurm_out and get_job_status which use the output of this function.

Examples

```
## Not run:
sjob <- slurm_apply(func, pars)
get_job_status(sjob) # Prints console/error output once job is completed.
func_result <- get_slurm_out(sjob, "table") # Loads output data into R.
cleanup_files(sjob)

## End(Not run)
```

---

slurm_call *Execution of a single function call on the Slurm cluster*

Description

Use slurm_call to perform a single function evaluation a the Slurm cluster.

Usage

```
slurm_call(
  f,
  params = list(),
  jobname = NA,
  global_objects = NULL,
  add_objects = NULL,
  pkgs = rev(.packages()),
  libPaths = NULL,
  rscript_path = NULL,
```

```
  r_template = NULL,
  sh_template = NULL,
  slurm_options = list(),
  submit = TRUE
)
```

Arguments

f	Any R function.
params	A named list of parameters to pass to f.
jobname	The name of the Slurm job; if NA, it is assigned a random name of the form "slr####".
global_objects	A character vector containing the name of R objects to be saved in a .RData file and loaded on each cluster node prior to calling f.
add_objects	Older deprecated name of global_objects, retained for backwards compatibility.
pkgs	A character vector containing the names of packages that must be loaded on each cluster node. By default, it includes all packages loaded by the user when slurm_call is called.
libPaths	A character vector describing the location of additional R library trees to search through, or NULL. The default value of NULL corresponds to libraries returned by .libPaths() on a cluster node. Non-existent library trees are silently ignored.
rscript_path	The location of the Rscript command. If not specified, defaults to the location of Rscript within the R installation being run.
r_template	The path to the template file for the R script run on each node. If NULL, uses the default template "rslurm/templates/slurm_run_single_R.txt".
sh_template	The path to the template file for the sbatch submission script. If NULL, uses the default template "rslurm/templates/submit_single_sh.txt".
slurm_options	A named list of options recognized by sbatch; see Details below for more information.
submit	Whether or not to submit the job to the cluster with sbatch; see Details below for more information.

Details

This function creates a temporary folder ("_rslurm_[jobname]") in the current directory, holding .RData and .RDS data files, the R script to run and the Bash submission script generated for the Slurm job.

The names of any other R objects (besides params) that f needs to access should be listed in the global_objects argument.

Use slurm_options to set any option recognized by sbatch, e.g. slurm_options = list(time = "1:00:00", share = TRUE). See http://slurm.schedmd.com/sbatch.html for details on possible options. Note that full names must be used (e.g. "time" rather than "t") and that flags (such

as "share") must be specified as TRUE. The "job-name", "ntasks" and "output" options are already determined by `slurm_call` and should not be manually set.

When processing the computation job, the Slurm cluster will output two files in the temporary folder: one with the return value of the function ("results_0.RDS") and one containing any console or error output produced by R ("slurm_[node_id].out").

If `submit = TRUE`, the job is sent to the cluster and a confirmation message (or error) is output to the console. If `submit = FALSE`, a message indicates the location of the saved data and script files; the job can be submitted manually by running the shell command `sbatch submit.sh` from that directory.

After sending the job to the Slurm cluster, `slurm_call` returns a `slurm_job` object which can be used to cancel the job, get the job status or output, and delete the temporary files associated with it. See the description of the related functions for more details.

Value

A `slurm_job` object containing the `jobname` and the number of `nodes` effectively used.

See Also

`slurm_apply` to parallelize a function over a parameter set.

`cancel_slurm`, `cleanup_files`, `get_slurm_out` and `get_job_status` which use the output of this function.

`slurm_job` *Create a slurm_job object*

Description

This function creates a `slurm_job` object which can be passed to other functions such as `cancel_slurm`, `cleanup_files`, `get_slurm_out` and `get_job_status`.

Usage

```
slurm_job(jobname = NULL, jobid = NULL, nodes = NULL)
```

Arguments

`jobname`	The name of the Slurm job. The rslurm-generated scripts and output files associated with a job should be found in the *_rslurm_[jobname]* folder.
`jobid`	The id of the Slurm job created by the sbatch command.
`nodes`	The number of cluster nodes used by that job.

Details

In general, `slurm_job` objects are created automatically as the output of `slurm_apply` or `slurm_call`, but it may be necessary to manually recreate one if the job was submitted in a different R session.

Value

A `slurm_job` object.

`slurm_map` *Parallel execution of a function over a list on the Slurm cluster*

Description

Use `slurm_map` to compute function over a list in parallel, spread across multiple nodes of a Slurm cluster, with similar syntax to `lapply`.

Usage

```
slurm_map(
  x,
  f,
  ...,
  jobname = NA,
  nodes = 2,
  cpus_per_node = 2,
  processes_per_node = cpus_per_node,
  preschedule_cores = TRUE,
  job_array_task_limit = NULL,
  global_objects = NULL,
  pkgs = rev(.packages()),
  libPaths = NULL,
  rscript_path = NULL,
  r_template = NULL,
  sh_template = NULL,
  slurm_options = list(),
  submit = TRUE
)
```

Arguments

`x`	A list to apply `f` to. Each element of `x` corresponds to a separate function call.
`f`	A function that accepts one element of `x` as its first argument, and may return any type of R object.
`...`	Additional arguments to `f`. These arguments do not vary with each call to `f`.
`jobname`	The name of the Slurm job; if `NA`, it is assigned a random name of the form "slr####".
`nodes`	The (maximum) number of cluster nodes to spread the calculation over. `slurm_map` automatically divides `x` in chunks of approximately equal size to send to each node. Less nodes are allocated if the parameter set is too small to use all CPUs on the requested nodes.

`cpus_per_node` The number of CPUs requested per node. This argument is mapped to the Slurm parameter `cpus-per-task`.

`processes_per_node`
The number of logical CPUs to utilize per node, i.e. how many processes to run in parallel per node. This can exceed `cpus_per_node` for nodes which support hyperthreading. Defaults to `processes_per_node = cpus_per_node`.

`preschedule_cores`
Corresponds to the `mc.preschedule` argument of `parallel::mclapply`. Defaults to `TRUE`. If `TRUE`, the elements of `x` are assigned to cores before computation. If `FALSE`, each element of `x` is executed by the next available core. Setting `FALSE` may be faster if different elements of `x` result in very variable completion time for jobs.

`job_array_task_limit`
The maximum number of job array tasks to run at the same time. Defaults to `NULL` (no limit).

`global_objects` A character vector containing the name of R objects to be saved in a .RData file and loaded on each cluster node prior to calling `f`.

`pkgs` A character vector containing the names of packages that must be loaded on each cluster node. By default, it includes all packages loaded by the user when `slurm_map` is called.

`libPaths` A character vector describing the location of additional R library trees to search through, or NULL. The default value of NULL corresponds to libraries returned by `.libPaths()` on a cluster node. Non-existent library trees are silently ignored.

`rscript_path` The location of the Rscript command. If not specified, defaults to the location of Rscript within the R installation being run.

`r_template` The path to the template file for the R script run on each node. If NULL, uses the default template "rslurm/templates/slurm_run_R.txt".

`sh_template` The path to the template file for the sbatch submission script. If NULL, uses the default template "rslurm/templates/submit_sh.txt".

`slurm_options` A named list of options recognized by `sbatch`; see Details below for more information.

`submit` Whether or not to submit the job to the cluster with `sbatch`; see Details below for more information.

Details

This function creates a temporary folder ("_rslurm_[jobname]") in the current directory, holding .RData and .RDS data files, the R script to run and the Bash submission script generated for the Slurm job.

The set of input parameters is divided in equal chunks sent to each node, and `f` is evaluated in parallel within each node using functions from the `parallel` R package. The names of any other R objects (besides `x`) that `f` needs to access should be included in `global_objects` or passed as additional arguments through `...`.

Use slurm_options to set any option recognized by sbatch, e.g. slurm_options = list(time = "1:00:00", share = TRUE). See http://slurm.schedmd.com/sbatch.html for details on possible options. Note that full names must be used (e.g. "time" rather than "t") and that flags (such as "share") must be specified as TRUE. The "array", "job-name", "nodes", "cpus-per-task" and "output" options are already determined by slurm_map and should not be manually set.

When processing the computation job, the Slurm cluster will output two types of files in the temporary folder: those containing the return values of the function for each subset of parameters ("results_[node_id].RDS") and those containing any console or error output produced by R on each node ("slurm_[node_id].out").

If submit = TRUE, the job is sent to the cluster and a confirmation message (or error) is output to the console. If submit = FALSE, a message indicates the location of the saved data and script files; the job can be submitted manually by running the shell command sbatch submit.sh from that directory.

After sending the job to the Slurm cluster, slurm_map returns a slurm_job object which can be used to cancel the job, get the job status or output, and delete the temporary files associated with it. See the description of the related functions for more details.

Value

A slurm_job object containing the jobname and the number of nodes effectively used.

See Also

slurm_call to evaluate a single function call.

slurm_apply to evaluate a function row-wise over a data frame of parameters.

cancel_slurm, cleanup_files, get_slurm_out and get_job_status which use the output of this function.

Examples

```
## Not run:
sjob <- slurm_map(func, list)
get_job_status(sjob) # Prints console/error output once job is completed.
func_result <- get_slurm_out(sjob, "table") # Loads output data into R.
cleanup_files(sjob)

## End(Not run)
```

Index