

Package ‘Rsge’

May 15, 2009

Title Interface to the SGE Queuing System

Date 2009-5-14

Version 0.6.3

Author Dan Bode <bodepd@yahoo.com>

Description This package provides functions for using R with the SGE cluster/grid queuing system.

Depends snow

SystemRequirements Requires SGE (qsub, qstat, optionally qacct)

Maintainer Dan Bode <bodepd@yahoo.com>

License LGPL

Repository CRAN

Date/Publication 2009-05-15 11:25:59

R topics documented:

sge.checkNotNow	2
sge.get.jobid	2
sge.get.result	3
sge.job.status	4
sge.options	4
sge.parApply	6
sge.parPrep	9
sge.split	9
sge.submit	10
Index	12

`sge.checkNotNow` *Determines if the -now y flag was run and if the job can be scheduled*

Description

Internal function. Determines if the job can be scheduled immediately

Usage

```
sge.checkNotNow(result)
```

Arguments

`result` result of qsub that is parsed.

Author(s)

Dan Bode

`sge.get.jobid` *Returns the jobid from a qsub call.*

Description

This function parses the output from a qsub call and returns the job id.

Usage

```
sge.get.jobid(result)
```

Arguments

`result` Resulting output from qsub call

Details

This function parses the output from a qsub call and returns the job id.

Value

integer id of job.

Author(s)

Dan Bode

<code>sge.get.result</code>	<i>Returns the results for a job that has been executed in the SGE environment</i>
-----------------------------	--

Description

Used to retrieve stored results that were written by the worker processes.

Usage

```
sge.get.result(sge.fname, jobid)
sge.list.get.result(list, ...)
```

Arguments

<code>sge.fname</code>	File location where the worker process results are stored
<code>jobid</code>	Job id of the job whose results we are returning
<code>list</code>	List that contains the jobid and filename
<code>...</code>	also passes <code>sge.fname</code> , <code>jobid</code> , and <code>remove</code> arguments to <code>sge.get.result</code>

Details

If the list version is used, then the list must have the named elements filename and jobid.

Rsgc integrates with SGE by using the R functions load/store to write files to a shared file system accessible by both the worker processes and the submission process.

Every node that runs will store its results in a unique file specified by the options `sge.file.prefix` and `sge.ret.ext`. This function expects the results filename to be passed as an option.

```
sge.file.prefixuniquestring[SGE_TASK_ARRAY_ID].sge.ret.ext
```

This function is used internally by `sge.parApply`, and should be called by the user to gather results from calls to `sge.submit`.

Value

Returns results from the worker processes.

Author(s)

Dan Bode (dbode@univaud.com)

See Also

[sge.submit](#)

sge.job.status *Determines the state of jobs*

Description

Used to determine if a job is running.

Usage

```
sge.job.status(jobid, qacct=getOption("sge.use.qacct"))
```

Arguments

jobid	ID of the job to be checked
qacct	Specifies if qacct should be used to determine the status of completed jobs

Details

This method checks the qstat command to see if the specified job is still running. It returns 0 if the specified job has completed. If qacct is set to "TRUE", then the program will return the return code of qacct j JOBID (0 for success, non zero for error) Otherwise, the return will be the return code for qstat j JOBID.

Value

0 specifies job/completion Other returns are forwarded from qacct j JOBID or qstat j JOBID.

Author(s)

Dan Bode <dbode@univaud.com>

See Also

[sge.submit](#) [sge.list.get.result](#)

sge.options *Allows a user to examine and reset values of global parameters*

Description

Allows a user to examine and reset values of global parameters

Usage

```
sge.options(...)  
sge.getOption(...)  
sge.setDefaultOptions()
```

Arguments

. . . Either empty, or a succession of parameter names in quotes, or a succession of name=value pairs. See below for the parameter names.

Details

This function allows the user to examine and reset the values of global parameters. It is analogous to the system function options.

set.setDefaultOptions is used to set all of the default options for sge, it is mainly used by the loader to init options.

The global parameters are:

sge.save.global Should the global environment be saved by default? Default is TRUE (this may change in future versions) sge.use.cluster Specifies if the run should be performed using an SGE cluster. Default is TRUE. sge.block.size Specifies the size of the blocks of X used if njobs is not specified. Default is 100. sge.file.prefix Prefix to be used for data and return files. Default is Rsg_data. sge.qsub qsub to call. Default assumes that qsub is in the path. Change this if it is not. sge.qstat qstat to call. Default assumes that qstat is in the path. Change this if it is not. sge.qacct qacct to call. Default assumes that qacct is in the path. Change this if it is not. sge.user.options user options to be passed to qsub, change this if you want to. Default is -S /bin/bash. If this value is changed, remember that the scheduler must still run bash. sge.ret.ext extension used for the return files. Default sge.ret sge.remove.files Specifies if files should be removed after execution. This applies to both data files as well as qsub output files. Default is TRUE. sge.use.qacct Determines if the qacct command is configured for usage. Default is FALSE. sge.debug If set to TRUE, then the program will run on debug mode. Default is FALSE. sge.trace If set to TRUE, some status information will be showed. Default is TRUE.

The following were included as parameters for the developers, but not currently supported. It is recommended that they are not changed unless the user had read the code and understands the implications of the changes.

sge.qsub.options options to be passed to qsub. Default is -cwd. There are some operations that require this to be -cwd. sge.qsub.blocking options passed to qsub to indicate a blocking call. CAN-NOT BE CHANGED. Default -sync y -t 1-. sge.script location of the sge.script file. Default Run-SgeJob sge.monitor.script location of the monitor file. Default MonitorJob.sh.

Value

no return.

Author(s)

Dan Bode (dbode@univaud.com)

sge.parApply *Method used to integrate apply, lapply, sapply functionality with qsub.*

Description

applies each row/column of a matrix or each element of a list to a function.

Usage

```
sge.apply (X, MARGIN, FUN, ...,
          join.method=cbind,
          njobs, batch.size=getOption('sge.block.size'),
          packages=NULL, global.savelist=NULL, function.savelist=NULL,
          cluster =getOption("sge.use.cluster"),
          trace=getOption("sge.trace"), debug=getOption("sge.debug"),
          file.prefix=getOption("sge.file.prefix"))

sge.parRapply(X, FUN, ..., join.method=cbind,
             njobs, batch.size=getOption('sge.block.size'),
             packages=NULL, global.savelist=NULL, function.savelist=NULL,
             cluster =getOption("sge.use.cluster"),
             trace=getOption("sge.trace"), debug=getOption("sge.debug"),
             file.prefix=getOption("sge.file.prefix"))

sge.parCapply(X, FUN, ..., join.method=cbind,
             njobs, batch.size=getOption('sge.block.size'),
             packages=NULL, global.savelist=NULL, function.savelist=NULL,
             cluster =getOption("sge.use.cluster"),
             trace=getOption("sge.trace"), debug=getOption("sge.debug"),
             file.prefix=getOption("sge.file.prefix"))

sge.parLapply(X, FUN, ..., join.method=c,
             njobs, batch.size=getOption('sge.block.size'),
             packages=NULL, global.savelist=NULL, function.savelist=NULL,
             cluster =getOption("sge.use.cluster"),
             trace=getOption("sge.trace"), debug=getOption("sge.debug"),
             file.prefix=getOption("sge.file.prefix"))

sge.parSapply(X, FUN, ...,
             USE.NAMES=TRUE, simplify=TRUE,
             join.method=c,
             njobs, batch.size=getOption('sge.block.size'),
             packages=NULL, global.savelist=NULL, function.savelist=NULL,
             cluster=getOption("sge.use.cluster"),
             trace=getOption("sge.trace"), debug=getOption("sge.debug"),
             file.prefix=getOption("sge.file.prefix"))

sge.parParApply(X, FUN, ...,
              join.method=cbind,
```

```

    njobs,
    batch.size=getOption('sge.block.size'),
    trace=getOption("sge.trace"),
    packages=NULL,
    global.savelist=NULL, function.savelist=NULL,
    debug=getOption("sge.debug"),
    file.prefix=getOption('sge.file.prefix'),
    apply.method
  )

```

Arguments

<code>x</code>	Object to be applied to function (matrix, data.frame, list, array, or vector supported)
<code>MARGIN</code>	Used by <code>sge.par</code> to determine if (1) <code>sge.Rapply</code> or (2) <code>sge.Capply</code> should be called.
<code>FUN</code>	Function to be applied to object
<code>...</code>	Additional arguments to be applied to function
<code>njobs</code>	Number of parallel jobs to use
<code>join.method</code>	Function used to merge results from each job.
<code>batch.size</code>	Number of rows to include in parallel job if <code>njobs</code> is excluded.
<code>global.savelist</code>	Character vector giving the names of variables from the global environment that should be imported. If <code>sge.save.global</code> is set to <code>TRUE</code> , then this will clear the global environment. To set the global environment to be empty, use <code>vector()</code>
<code>function.savelist</code>	Character vector giving the variables to save from the local environment. <Passing a <code>vector()</code> will cause the local function to be empty, any value here will remove any non-listed values
<code>packages</code>	List of library packages to be loaded by each worker process before computation is started.
<code>cluster</code>	determines if the job should be submitted to the cluster or run locally (default:submit to cluster, <code>TRUE</code>)
<code>trace</code>	Prints some information about job submission. (<code>TRUE FALSE</code>)
<code>file.prefix</code>	Prefix for the data files used to pass information between nodes
<code>debug</code>	Prints debug level info about jobs. (<code>TRUE FALSE</code>)
<code>USE.NAMES</code>	Determines if the <code>sapply</code> use names functionality should be used
<code>simplify</code>	Determines if the <code>sapply</code> simplify functionality should be used
<code>apply.method</code>	tells <code>apply</code> if it should apply as a list or matrix

Details

`sge.parApply` applies the function argument to either the rows, or elements of `x`. (depending on the value of `apply.method`)

The arguments `packages` and `savelist` can be used to properly initialize the worker processes. `sge.parCapply` computes the transpose of `X` and uses it for the arguments of `parRapply`.

`sge.parLapply` and `sge.parSapply` use `lapply`, `sge.parCapply`, and `sge.parRapply` use `apply` (`Margin=1`)

Value

Returns an object whose type is determined by the `join.method`. This object should be equivalent to the object that would be returned by an equivalent call to `apply`, `sapply`, or `lapply`.

Author(s)

Dan Bode (dbode@univaud.com)

Examples

```
## Not run:
# there are tons of examples (actually my crude unit tests)
# in the tests directory of this package. These contain tons of
# examples of how to dop everything
# I also recommend taking a look at the ENVIRONMENTS document, since
# the environment is being handled specially.

# basic lapply call
sge.parLapply(c(1:10), function(x) x + 1)
# run locally
options(sge.use.cluster=FALSE)
sge.parLapply(c(1:10), function(x) x + 1)
options(sge.use.cluster=TRUE)
#pass extra arguments
sge.parLapply(c(1:10), function(x,y) x + y, 3)

# work with matrices
m2 <- array(1:20, dim=c(4,5))
# apply rows, split into 3 jobs
sge.parRapply(m2, function(x,y) x + y, 3, njobs=3)
# bind results with c, since we are changing dimensions
sge.parRapply(m2, mean, njobs=3, join.method=c)

# working with GLOBAL variables
GLOBAL1 = 1
# rather this variable is saved by default depends on the value of sge.save.global
sge.options(sge.save.global=TRUE)
sge.parLapply(c(1:10), function(x) x + GLOBAL1)
sge.options(sge.save.global=FALSE)
sge.parLapply(c(1:10), function(x) x + GLOBAL1, global.savelist=c("GLOBAL1"))
## End(Not run)
```

`sge.parPrep`*Prepares environment for parallel run*

Description

This is an internal function that is used to prepare data files that will be processed by the worker threads.

Author(s)

Dan Bode

`sge.split`*Splits an object and returns a list containing those segments*

Description

This functions splits an object into a number of pieces as specified by ncl.

Usage

```
sge.split(x, ncl)
```

Arguments

<code>x</code>	Object to be split (only supports matrix, array, data.frame, list, and vector)
<code>ncl</code>	Number of segments that the object should be split into.

Details

This method is an internal helper method that is used by parApply to split the objects into segments that are distributed to nodes of an SGE cluster. This function relies on the splitIndices method from the snow package to determine how many rows each segment should have.

Value

List containing all split data segments.

Author(s)

Dan Bode

<code>sge.submit</code>	<i>Performs an asynchronous submission of a function to the remote node</i>
-------------------------	---

Description

Used to asynchronously submit R tasks to a SGE cluster. `sge.run`, makes and synchronious call to `sge.submit`.

Usage

```
sge.submit(func, ...,
           global.savelist=NULL,
           function.savelist=NULL,
           packages=NULL,
           debug=getOption("sge.debug"), file.prefix=getOption('sge.file.prefix')
           )
sge.run(...)
```

Arguments

<code>func</code>	Function to be executed remotely.
<code>...</code>	Arguments to be passed to the function. For <code>sge.run</code> , arguments to be passed to <code>sge.submit</code>
<code>global.savelist</code>	Character vector giving the names of variables from the global environment that should be imported. If <code>sge.save.global</code> is set to <code>TRUE</code> , then this will clear the global environment. To set the global environment to be empty, use <code>vector()</code>
<code>function.savelist</code>	Character vector giving the variables to save from the local environment. Passing any arguments here will cause the function environment to be cleared. Passing a <code>vector()</code> will cause the local function to be empty.
<code>packages</code>	List of library packages to be loaded by each worker process before computation is started.
<code>debug</code>	Runs at debug level.
<code>file.prefix</code>	Prefix used to create data file

Details

Submits work to SGE with an asynchronous `qsub` call. The user needs to use `sge.job.status` and `sge.list.get.result` to monitor the progress of jobs and retrieve results.

Value

Returns a list that has the named element `filename` for the name of the input file created to send commands to the SGE cluster and a named element `jobid` with the id of the job submitted. `list(jobid=JOBID, filename=FILENAME)`

Author(s)

Dan Bode (dbode@univaud.com)

See Also

[sge.list.get.result](#)

Examples

```
## Not run:
#execute this easy function on the cluster
  info <- sge.submit(function(x) {
    Sys.sleep(x)
    x
  }, 10)
#check the status by passing the job id
  status <- sge.job.status(info$jobid)
  while(status != 0) {
#continue to check the status, until job is completed
  Sys.sleep(4)
  status <- sge.job.status(info$jobid)
  }
#once we sure sure that the job is finished, retrieve the results
  result <- sge.list.get.result(info)

# this is a more complicated example that shows how lists can be used to store
# multiple results

v1 <- c(1:10)
funcl <- function(x) {
  Sys.sleep(x)
  x
}

l1 <- list(length=length(v1))
for(i in 1:length(v1)) {
  l1[[i]] <- sge.submit(funcl, v1[[i]])
}

r1 <- lapply(l1, sge.job.status)
while(! all(r1 == 0)) {
  Sys.sleep(4)
  r1 <- lapply(l1, sge.job.status)
}
# its ok to just pass the list of jobid and filename
x1Par <- lapply(l1, sge.list.get.result)
## End(Not run)
```

Index

*Topic **interface**

- `sge.checkNotNow`, 1
- `sge.get.jobid`, 2
- `sge.get.result`, 2
- `sge.job.status`, 3
- `sge.options`, 4
- `sge.parApply`, 5
- `sge.parPrep`, 8
- `sge.split`, 8
- `sge.submit`, 9

- `sge.apply(sge.parApply)`, 5
- `sge.checkNotNow`, 1
- `sge.get.jobid`, 2
- `sge.get.result`, 2
- `sge.getOption(sge.options)`, 4
- `sge.globalPrep(sge.parPrep)`, 8
- `sge.job.status`, 3
- `sge.list.get.result`, 4, 10
- `sge.list.get.result`
 - `(sge.get.result)`, 2
- `sge.options`, 4
- `sge.parApply`, 5
- `sge.parCapply(sge.parApply)`, 5
- `sge.parLapply(sge.parApply)`, 5
- `sge.parParApply(sge.parApply)`, 5
- `sge.parPrep`, 8
- `sge.parRapply(sge.parApply)`, 5
- `sge.parSapply(sge.parApply)`, 5
- `sge.run(sge.submit)`, 9
- `sge.setDefaultOptions`
 - `(sge.options)`, 4
- `sge.split`, 8
- `sge.submit`, 3, 4, 9
- `sge.taskPrep(sge.parPrep)`, 8