

Package ‘GridR’

January 2, 2012

Type Package

Title Executes functions on remote hosts, clusters or grids.

Version 0.9.1

Date 2009-01-30

Author Dennis Wegener, Malte Lohmeyer, Stefan Rueping

Maintainer Dennis Wegener <dennis.wegener@iais.fraunhofer.de>

Description GridR is an R-Package that can be used to submit R functions for execution on remote computers, clusters or grids. In addition, users are provided with an interface to share variables and functions with other users.

License GPL-2

Depends codetools

Repository CRAN

Date/Publication 2009-02-03 20:02:40

R topics documented:

GridR-package	2
grid.apply	3
grid.check	5
grid.cogMyproxy	5
grid.compileScript	6
grid.consistency	6
grid.disableSharing	7
grid.enableSharing	7
grid.exit	8
grid.FTPDownload	8
grid.FTPUpload	9
grid.globusMyproxy	9
grid.GridFTPDownload	10

grid.GridFTPUpload	10
grid.init	11
grid.isLocked	14
grid.lock	15
grid.printJobs	15
grid.proxyInit	16
grid.restartJob	16
grid.restartScheduler	17
grid.share	17
grid.stopJob	18
grid.stopScheduler	18
grid.unlock	19
grid.unlockAll	19
grid.waitForResult	20

Index	21
--------------	-----------

GridR-package	<i>GridR executes a function on remote Hosts or Cluster.</i>
---------------	--

Description

GridR is an R-Package that submits R functions to execute them on another computer or cluster and it provides an interface to share functions and variables with other users.

Submission modes are using a web service, ssh or local, execution modes are condor, globus or using a single server.

All needed functions and variables that are necessary to execute that function will be copied to the execution machine.

Details

Package:	GridR
Type:	Package
Version:	0.9.1
Date:	2009-01-30
License:	GPL-2

With `grid.init(...)` GridR is initialized.

The specification of the configuration parameters can be performed with the help of a config file (that is read when the command `grid.init()` is performed) or at runtime by passing the parameters to the functions `grid.init()` and `grid.apply()`.

The name of the config file is ".gridr.conf" or "gridr.conf" and it should be placed inside the user home directory or the directory from which R is executed.(see [grid.init](#))

Using `grid.apply` you can submit a function to be executed in the grid.(see [grid.apply](#))

After execution the result will be copied back.

With `grid.share` you can share functions and variables with other users on your computer. If you have installed an nfs you can share them with remote users.(see [grid.share](#))

Author(s)

Malte Lohmeyer Dennis Wegener Stefan Rueping

See Also

[grid.init](#) [grid.apply](#) [grid.share](#) [grid.enableSharing](#) [grid.disableSharing](#) [grid.exit](#)
[grid.printJobs](#) [grid.isLocked](#) [grid.waitForResult](#) [grid.consistency](#) [grid.callback](#) [grid.check](#)
[grid.cogMyproxy](#) [grid.globusMyproxy](#) [grid.proxyInit](#) [grid.lock](#) [grid.unlock](#)

Examples

```
a<-function(s){return(2*s)}
#define a function that will be executed remotely
library("GridR")
#load the gridR-Code
grid.init(service="local",debug=FALSE, localTmpDir="GridRTmp/")
#initializes gridR
grid.apply("x",a, 3, wait=TRUE)
#applies function 'a' with parameter 3 and writes the result to variable x. until the function is executed, x has a 1
```

grid.apply

grid.apply

Description

`grid.apply` submits a function to another Host or Cluster, to execute it there.

Usage

```
grid.apply(grid.input.Parameters.y=NULL,grid.input.Parameters.f=NULL,... ,wait=FALSE ,varlist=c(),pl
```

Arguments

<code>grid.input.Parameters.y</code>	The variable in " " where to write the output if the job is finished
<code>grid.input.Parameters.f</code>	which function should be executed remotely
<code>...</code>	the parameters of the function
<code>wait</code>	if <code>wait=TRUE</code> , R blocks until the Job is executed, otherwise Job starts in background
<code>varlist</code>	if you know which variables or functions are needed to execute that function remotely, add them here if you do not want to use <code>check=TRUE</code>

plots	is there a plot output of the function? Will be implemented in future
run	please do not change, for internal use
check	if check=TRUE, before submission it is checked if the function needs other variables or other functions to executed it remotely. If so, they are copied. If an internal function like sum is applied, please use check=FALSE, otherwise codetools doesnt work
batch	If you want the output of different parameter sets, you can add here which parameters are vectors and should be swept ie. c(1,3) then all combinations of parameter 1 and 3 are executed, only works with condor.ssh, remote.ssh and local modes
javaSsh	if TRUE, ssh transmission will work on Linux Systems over a JavaSsh Library and not directly over ssh

Details

Until the result is back, there is a lock to variable y

Be careful: The variable `"grid.input.Parameters"` is used internally, so please do not use it in your code!

To delete old local tmp files call [grid.consistency](#)

Author(s)

Malte Lohmeyer

See Also

[grid.init](#)

Examples

```
#example to execute function a on a single remote host via ssh and save the result to x
library("GridR")
a<-function(s){Sys.sleep(s); return(s)}
grid.init(service="local", localTmpDir="GridRTmp/")
grid.apply("x",a, 3, wait=TRUE, check=TRUE)
x
```

```
#example for batch mode
library("GridR")
a<-function(s, p, q){return(s+p+q)}
grid.init(service="local", localTmpDir="GridRTmp/")
grid.apply("y",a, c(0,1,2),1, c(100,200,300), wait=TRUE, check=FALSE, batch=c(1,3))
#Here all combinations of (0,1,2), 1 and (100,200,300) are executed as parameters of function a. A List with all comb
y
```

`grid.check`*grid.check*

Description

checks if varlist contains all variables and functions to execute function f. If not the missing variables are returned

Usage

```
grid.check(grid.input.Parameters.f,x="",varlist=c(), fName="", intern=FALSE)
```

Arguments

<code>grid.input.Parameters.f</code>	which functions should be checked
<code>x</code>	for internal use
<code>varlist</code>	a vector which contains needed variables and functions to execute f
<code>fName</code>	for internal use
<code>intern</code>	with <code>intern=FALSE</code> a human output is printed, otherwise null or the missing variables are returned

Author(s)

Malte Lohmeyer

`grid.cogMyproxy`*grid.cogMyproxy*

Description

renews the myproxy certificate, if COG is installed

Usage

```
grid.cogMyproxy()
```

Author(s)

Malte Lohmeyer

grid.compileScript *grid.compileScript*

Description

compiles an R script from the R history. All entered lines in the actual R Session are saved to path. If there is a parsing error, an error message is shown

Usage

```
grid.compileScript(path)
```

Arguments

path path of the output file

Author(s)

Malte Lohmeyer

Examples

```
#library("GridR")
#grid.init(service="local")
#a=1
#1
#grid.compileScript("test.R")
```

grid.consistency *grid.consistency*

Description

checks if there are errors in the .grid variable, local files without jobs or jobs without local files. Deletes old local tmp files.

Usage

```
grid.consistency()
```

Author(s)

Malte Lohmeyer

`grid.disableSharing` *grid.disableSharing*

Description

disables variable sharing and removes the Task Callback

Usage

`grid.disableSharing()`

Author(s)

Malte Lohmeyer

See Also

[grid.share](#)

`grid.enableSharing` *grid.enableSharing*

Description

enables variable sharing and adds a Task Callback to update local variables

Usage

`grid.enableSharing()`

Author(s)

Malte Lohmeyer

See Also

[grid.share](#)

`grid.exit`*grid.exit*

Description

removes all locks, deletes .grid and removes task callback (that grid.callback is executed with every ENTER hit)

Usage`grid.exit()`**Author(s)**

Stefan Rueping

`grid.FTPDownload`*grid.FTPDownload*

Description

Downloads a file from url and saves it it path

Usage`grid.FTPDownload(url,path)`**Arguments**

<code>url</code>	The url of the file to download
<code>path</code>	The path where to download the file

Author(s)

Malte Lohmeyer

`grid.FTPUpload` *grid.FTPUpload*

Description

Uploads the file path to url

Usage

`grid.FTPUpload(url,path)`

Arguments

<code>url</code>	The url where to upload the file
<code>path</code>	The path of the file to upload

Author(s)

Malte Lohmeyer

`grid.globusMyproxy` *grid.globusMyproxy*

Description

renews the myproxy certificate, if Globus is installed

Usage

`grid.globusMyproxy()`

Author(s)

Malte Lohmeyer

`grid.GridFTPDownload` *grid.GridFTPDownload*

Description

Downloads a file from url and saves it it path with the use of GridFTP. Only works if <ACGTLIB-PATH> in the config file points to the globus toolkit libraries

Usage

```
grid.GridFTPDownload(url,path)
```

Arguments

url	The url of the file to download
path	The path where to download the file

Author(s)

Malte Lohmeyer

`grid.GridFTPUpload` *grid.GridFTPUpload*

Description

Uploads the file path to url with the use of GridFTP. Only works if <ACGTLIBPATH> in the config file points to the globus toolkit libraries

Usage

```
grid.GridFTPUpload(url,path)
```

Arguments

url	The url where to upload the file
path	The path of the file to upload

Author(s)

Malte Lohmeyer

grid.init	<i>grid.init</i>
-----------	------------------

Description

grid.init initializes the GridR Package. If you use a config file, it is not necessary to add one of the parameters explained below.

Usage

```
grid.init(confFile=NULL, localTmpDir=NULL, verbose=TRUE, sshRemoteIp=NULL, sshUsername=NULL, sshRemoteDir=NULL, sshKey=NULL, myProxyHost=NULL, myProxyUsername=NULL, credentialName=NULL, myProxyPwd=NULL, myProxyPort=NULL, service=NULL, debug=NULL, sharedDir=NULL, remoteRPath=NULL, schedulerIp=NULL, schedulerPort=NULL)
```

Arguments

confFile	Path to the config file
localTmpDir	Path to a directory where to store temporal data
verbose	if verbose=TRUE a message is printed if a result is available
sshRemoteIp	If ssh mode is used, the IP of the remote host is specified here
sshUsername	If an ssh mode is used, the remote username is specified here
sshRemoteDir	If an ssh mode is used, the remote temp dir is specified here
sshKey	If an ssh mode with windows is used, the path to the public RSA key is specified here
myProxyHost	the IP of the Host where a myproxy server is running is specified here
myProxyUsername	the username of the myproxy certificate is specified here
credentialName	the credentialname of the myproxy certificate is specified here, if needed
myProxyPwd	the password of the myproxy certificate is specified here
myProxyPort	the port of the myproxy certificate is specified here, if not the default port is used
service	Here you can add the default service mode, see ?grid.apply
debug	If TRUE, all files will not be deleted locally and on serverside
sharedDir	Path of the directory where shared variables will be loaded
remoteRPath	Path to R, if needed
schedulerIp	If a scheduler should be used, add his IP here
schedulerPort	Port of the scheduler, if used

Details

The easiest way to use GridR is to use a config file.

The name of the config file is "gridr.conf" and it should be placed inside the user home directory or the directory from where R is executed. if another path is used, please specify it with the value "\"configFile\""

The content depends on which modes you want to use:

There are different ways to submit a function. The services are only available for Linux servers and it is necessary to setup the system in a way that the user is able to login via ssh on the remote computer without entering a password. Please generate RSA Keys. (see ie. http://www.csua.berkeley.edu/~ranga/notes/ssh_nopass.html for Linux)

To use the services with Windows, a version which uses Trilead Java SSH is implemented. Please download trilead-ssh2-build212.jar or similar at http://www.trilead.com/Download/Trilead_SSH_for_Java/ and place it to <GridRInstallationDir>/GridR/GridR/ Please generate RSA Keys with Puttygen(ie. <http://the.earth.li/~sgtatham/putty/latest/x86/puttygen.exe>) Klick on Generate. This will generate a private/public RSA keypair. Change the Comment to your local username \@ yourLocalComputer Click on Conversions/Export OpenSSH Key and save it to the path added to your config file. Now add the Text in the "Public Key for pasting into OpenSSH authorized_keys file" to a new line to ~/.ssh/authorized_keys on the server where to execute GridR

On each Server, R must be added to the PATH environment variable, R_HOME must be set or remoteRPath must be declared. If condor modes are used, a link to R must be added to /usr/bin/ or add a line to the config file: <REMOTERPATH>/path/to/R</REMOTERPATH>

If the condor batch modes are used, the package GridR must be installed on each Cluster-Host.

Available Modes:

variableSharing

only variable and function sharing is initialized.

You must enter at least the following lines to the config file or command line:

```
<GRIDR> \#start tag, necessary
<SHAREDDIR>/home/user/mlohmeyer/share</SHAREDDIR> \#local dir where to put tmp files
<SERVICE>variableSharing</SERVICE> \#which default service to use by default
</GRIDR> \# end tag
```

local

the function is executed locally.

You must enter at least the following lines to the config file or command line:

```
<GRIDR> \#start tag, necessary
<LOCALTMPDIR>/home/user</LOCALTMPDIR> \#local dir where to put tmp files
<SERVICE>local</SERVICE> \#which default service to use by default
</GRIDR> \# end tag
```

remote.ssh

the function is copied with ssh to a single computer and is executed there directly in R. You must enter at least the following lines to the config file or command line:

```
<GRIDR> \#start tag, necessary
<LOCALTMPDIR>/home/user</LOCALTMPDIR> \#local dir where to put tmp files
<SSHREMOTEDIR>grid/</SSHREMOTEDIR> \# remote dir where to put tmp files
<SSHREMOTEIP>ip</SSHREMOTEIP> \# ip of the remote host
<SSHUSERNAME>user</SSHUSERNAME> \#ssh username to login on remote host
<SERVICE>remote.ssh</SERVICE> \#which default service to use by default
<SSHKEY>/home/user/.ssh/id_rsa</SSHKEY> \#on windows systems or with javaSsh=TRUE
you have to specify the path to your public RSA key
</GRIDR> \# end tag
```

condor.ssh

the function is copied with ssh to a computer which is connected to a condor pool and where submission of jobs is possible.

You must enter at least the following lines to the config file or command line:

```
<GRIDR> \#start tag, necessary
<LOCALTMPDIR>/home/user</LOCALTMPDIR> \#local dir where to put tmp files
<SSHREMOTEDIR>grid/</SSHREMOTEDIR> \# remote dir where to put tmp files
<SSHREMOTEIP>ip</SSHREMOTEIP> \# ip of the remote host
<SSHUSERNAME>username</SSHUSERNAME> \#ssh username to login on remote host
<SERVICE>condor.ssh</SERVICE> \#which default service to use by default
<SSHKEY>/home/user/.ssh/id_rsa</SSHKEY> \#on windows systems or with javaSsh=TRUE
you have to specify the path to your public RSA key
<REMOTERPATH>pathToRemoteR</REMOTERPATH>\# if R is not linked to /usr/bin/R on server-
side, please add this Tag here </GRIDR> \# end tag
```

scheduler if the variable schedulerIp is set, all jobs are started with the use of a scheduler. At grid.init() it is checked if there are jobs which are started earlier from another R Session. If so they are imported to your active R Session. You can start the scheduler by copying all files from R.home("library/GridR/GridR/scheduler") to the server where the scheduler should run, set up a passwordless ssh-logon from your client, to this scheduler-server and from the scheduler-server to all execution machines. Download trilead-ssh2-build212.jar or similar at http://www.trilead.com/Download/Trilead_SSH_for_Java/ and save it to the lib Directory. Now you can start the scheduler with ./start.sh <port>. The scheduler can be restarted with `grid.restartScheduler` and stopped with `grid.stopScheduler`. In scheduler Mode, all submitted jobs can be stopped by `grid.stopJob` or restarted by `grid.restartJob`

Author(s)

Malte Lohmeyer

See Also

[grid.apply](#) [GridR](#) [grid.share](#)

Examples

```

a<-function(s){return(2*s)}
#define a function that will be executed remotely
library("GridR")
#load the gridR-Code
grid.init(service="local", localTmpDir="GridRTmp/")
#initializes gridR with the parameters entered in the config file
grid.apply("x",a, 3, wait=TRUE)
#applies function 'a' with parameter 3 and writes the result to variable x. until the function is executed, x has a 1
x
grid.apply("y", sum,1:5, wait=TRUE, check=FALSE) # if internal functions are used, its important to set check=FALSE
y

```

grid.isLocked

grid.isLocked

Description

returns TRUE if varName is Locked, FALSE otherwise

Usage

```
grid.isLocked(varName)
```

Arguments

varName this variable will be checked

Author(s)

Malte Lohmeyer

Examples

```

library("GridR")
grid.init(service="variableSharing")
x=1
y=2
grid.lock("x")
grid.isLocked("x")
grid.isLocked("y")

```

`grid.lock`*grid.lock*

Description

Adds a lock to variable varName

Usage

```
grid.lock(varName)
```

Arguments

varName variable which should be locked

Author(s)

Stefan Rueping

Examples

```
library("GridR")
grid.init(service="local")
x=1
y=2
grid.lock("x")
## Not run: x
```

`grid.printJobs`*grid.printJobs*

Description

prints all running GridR jobs

Usage

```
grid.printJobs()
```

Author(s)

Stefan Rueping

`grid.proxyInit` *grid.proxyInit*

Description

Initializes a new local proxy certificate for globus.cog mode

Usage

```
grid.proxyInit()
```

Author(s)

Malte Lohmeyer

See Also

[grid.init](#)

`grid.restartJob` *grid.restartJob*

Description

`grid.restartJob` restarts a Job which is submitted to the scheduler

Usage

```
grid.restartJob(job)
```

Arguments

<code>job</code>	The local Variable as String where the output of the Job is saved or the Jobnumber seen by <code>grid.printJobs</code>
------------------	--

Author(s)

Malte Lohmeyer

`grid.restartScheduler` *grid.restartScheduler*

Description

`grid.restartScheduler` restarts the scheduler

Usage

`grid.restartScheduler()`

Author(s)

Malte Lohmeyer

`grid.share` *grid.share*

Description

`grid.share` shares functions and variables with other users on your computer. If an nfs is installed, they can be shared them with remote users.

Usage

`grid.share(varName)`

Arguments

`varName` The variable or functionname in " " which should be shared

Details

If this function is executed, `varName` is stored to `sharedDir` which must be specified in the config file or the `grid.init` function.

If `grid.init` is called, a `taskCallback` is added which checks each time a command is entered to R if a new function or variable is stored to the `sharedDir`. If so it is loaded into R.

Be Careful: it overrides the old value automatically!

Author(s)

Malte Lohmeyer

Examples

```
library("GridR")
grid.init(service="variableSharing", sharedDir="GridRTmp/share")
x=2
grid.share("x")
```

`grid.stopJob` *grid.stopJob*

Description

`grid.stopJob` stops a Job which is submitted to the scheduler

Usage

```
grid.stopJob(job)
```

Arguments

`job` The local Variable as String where the output of the Job is saved or the Jobnumber seen by `grid.printJobs`

Author(s)

Malte Lohmeyer

`grid.stopScheduler` *grid.stopScheduler*

Description

`grid.stopScheduler` stops the scheduler

Usage

```
grid.stopScheduler()
```

Author(s)

Malte Lohmeyer

`grid.unlock` *grid.unlock*

Description

unlocks the specified variable

Usage

`grid.unlock(varName)`

Arguments

`varName` this variable will be unlocked

Author(s)

Stefan Rueping

Examples

```
library("GridR")
grid.init(service="local")
x=1
grid.lock("x")
grid.unlock("x")
```

`grid.unlockAll` *grid.unlockAll*

Description

unlocks all locked variables

Usage

`grid.unlockAll()`

Author(s)

Stefan Rueping

Examples

```
library("GridR")
grid.init(service="local")
x=1
y=1
grid.lock("x")
grid.lock("y")
grid.unlockAll()
```

`grid.waitForResult` *grid.waitForResult*

Description

waits until all variables are unlocked

Usage

```
grid.waitForResult(vars)
```

Arguments

`vars` a vector with locked variables

Author(s)

Malte Lohmeyer

Index

*Topic **file**

- grid.FTPDownload, 8
- grid.FTPUpload, 9
- grid.GridFTPDownload, 10
- grid.GridFTPUpload, 10

*Topic **methods**

- grid.init, 11

*Topic **package**

- GridR-package, 2

*Topic **print**

- grid.printJobs, 15

*Topic **programming**

- grid.apply, 3
- grid.check, 5
- grid.cogMyproxy, 5
- grid.compileScript, 6
- grid.consistency, 6
- grid.disableSharing, 7
- grid.enableSharing, 7
- grid.exit, 8
- grid.globusMyproxy, 9
- grid.isLocked, 14
- grid.lock, 15
- grid.proxyInit, 16
- grid.restartJob, 16
- grid.restartScheduler, 17
- grid.share, 17
- grid.stopJob, 18
- grid.stopScheduler, 18
- grid.unlock, 19
- grid.unlockAll, 19
- grid.waitForResult, 20

- grid.apply, 2, 3, 3, 13
- grid.callback, 3
- grid.check, 3, 5
- grid.cogMyproxy, 3, 5
- grid.compileScript, 6
- grid.consistency, 3, 4, 6
- grid.disableSharing, 3, 7

- grid.enableSharing, 3, 7
- grid.exit, 3, 8
- grid.FTPDownload, 8
- grid.FTPUpload, 9
- grid.globusMyproxy, 3, 9
- grid.GridFTPDownload, 10
- grid.GridFTPUpload, 10
- grid.init, 2-4, 11, 16
- grid.isLocked, 3, 14
- grid.lock, 3, 15
- grid.printJobs, 3, 15
- grid.proxyInit, 3, 16
- grid.restartJob, 13, 16
- grid.restartScheduler, 13, 17
- grid.share, 3, 7, 13, 17
- grid.stopJob, 13, 18
- grid.stopScheduler, 13, 18
- grid.unlock, 3, 19
- grid.unlockAll, 19
- grid.waitForResult, 3, 20
- GridR, 13
- GridR (GridR-package), 2
- Gridr (GridR-package), 2
- gridR (GridR-package), 2
- gridr (GridR-package), 2
- GridR-package, 2