Package 'pbdRPC'

May 5, 2018

Version 0.2-1

Date 2018-04-30

Title Programming with Big Data -- Remote Procedure Call

Depends R (>= 3.2.0), tools

Enhances pbdZMQ, remoter, pbdMPI

LazyLoad yes

LazyData yes

Copyright See pbdRPC/inst/putty_LICENCE for files in src/putty/.

Description A very light implementation yet secure for remote procedure calls with unified interface via ssh (OpenSSH) or plink/plink.exe (PuTTY).

SystemRequirements ssh (OpenSSH) or plink (PuTTY) on Solaris, Linux, and Mac.

License Mozilla Public License 2.0

URL http://r-pbd.org/

BugReports https://github.com/snoweye/pbdRPC/issues

MailingList Please send questions and comments regarding pbdR to RBigData@gmail.com

RoxygenNote 6.0.1

NeedsCompilation yes

Maintainer Wei-Chen Chen <wccsnow@gmail.com>

Author Wei-Chen Chen [aut, cre], Drew Schmidt [aut]

Repository CRAN

Date/Publication 2018-05-05 17:53:30 UTC

R topics documented:

`pbdRPC-package` *pbdRPC*

Description

A very light yet secure implementation of remote procedure call.

Author(s)

Wei-Chen Chen and Drew Schmidt

References

Project URL: https://github.com/RBigData/pbdRPC

`find_plink` *Find plink*

Description

Find the plink internal compiled with pbdRPC package.

Usage

```
find_plink()
```

Details

The function returns a full path to the `plink` or `plink.exe` command if found.

Value

A full path a full path to the `plink` or `plink.exe` command if found. Otherwise, "plink" is returned.

Examples

```
## Not run:
library(pbdRPC, quietly = TRUE)

find_plink()

## End(Not run)
```

`Initial Control Functions`

Initial controls in pbdRPC

Description

Initial control functions

Usage

```
.rpcopt_init(envir = .GlobalEnv)
```

Arguments

`envir`	an environment where RPC controls locate

Details

`.rpcopt_init()` initials default RPC controls.

Value

`.rpcopt_init()` initial the RPC control at `envir`.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>.

References

ZeroMQ/4.1.0 API Reference: http://api.zeromq.org/4-1:_start

Programming with Big Data in R Website: http://r-pbd.org/

See Also

.pbd_env.

Examples

```
## Not run:
library(pbdRPC, quietly = TRUE)
rpcopt_set(user = "snoweye", hostname = "192.168.56.101")

ls(.pbd_env)
rm(.pbd_env)
.rpcopt_init()
ls(.pbd_env)

.pbd_env$RPC.LI

## End(Not run)
```

---

machine *machine*

Description

Remote machine configuration settings.

Usage

```
machine(hostname, user, exec.type = .pbd_env$RPC.LI$exec.type,
  args = .pbd_env$RPC.LI$args, pport = .pbd_env$RPC.LI$pport,
  priv.key = .pbd_env$RPC.LI$priv.key,
  priv.key.ppk = .pbd_env$RPC.LI$priv.key.ppk)
```

Arguments

hostname	the server ip or host name.
user	user id for logging to the server. If none is supplied, then the system user name will be used instead.
exec.type	either "ssh" or "plink" in character. Windows will force to use "plink".
args	further arguments to "ssh" or "plink" for connecting to the server in addition to port, user id, and host name.
pport	ssh port opened on the server.
priv.key, priv.key.ppk	location of the private key for user authentication, the file will be checked first then `-i priv.key` will be added to `args` when the file exists. `priv.key.ppk` is only used when `plink` is called.

Value

An object of class machine.

Examples

```
## Not run:
# note: not my actual aws url
myaws <- machine("ec2-1-2-3-4.compute-1.amazonaws.com", user="my_aws_username")

# if you don't specify 'user', we use your host machine's user name
myvm <- machine("192.168.1.10")

## End(Not run)
```

---

print-machine *print-machine*

Description

Printing for machine class objects.

Usage

```
## S3 method for class 'machine'
print(x, ...)
```

Arguments

x	machine class object
...	ignored

---

rpc *Remote Procedure Call*

Description

Launch a command via ssh or plink on a (remote) server.

Usage

```
rpc(machine, cmd = "whoami", intern = .pbd_env$RPC.CT$intern,
  wait = .pbd_env$RPC.CT$wait)
```

Arguments

machine	A machine configuration. See ?machine.
cmd	the command to be executed on the server.
intern, wait	arguments passed to system() or shell() whereever they are applicable.

Details

Using either ssh or plink to launch a command on a (remote) server. Authentication is working currently for ssh.

NO further input from stdin is expected. Outputs or errors may not be captured by R.

Value

Mainly the message received from the command line of server may be returned but may not be captured by R.

For example, Windows with plink will not capture the return because currently the authentication is not working. A windows bat file is launched by shell.exec() in an additional cmd.exe window to avoid saving password inside R.

See Also

machine(), start_rr(), check_rr(), kill_rr(), srpc(), ssh(), and plink().

RPC Control Environment

Sets of controls in pbdRPC.

Description

These sets of controls are used to provide default values in this package.

Format

Objects contain several parameters for communicators and methods.

Details

The elements of .pbd_env$RPC.CT are default values for looking RPC control files.

The elements of .pbd_env$RPC.LI are default values for RPC defult login information.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>.

Programming with Big Data in R Website: http://r-pbd.org/

See Also

.rpcopt_init().

RPC Control Functions *Sets of controls in pbdRPC.*

Description

These sets of controls are used to provide default values in pbdRPC package.

Usage

```
RPC.CT(verbose = FALSE, intern = FALSE, wait = TRUE, check.exec = FALSE,
  use.shell.exec = TRUE, pause = TRUE)

RPC.LI(exec.type = "ssh", args = "", pport = 22, user = "snoweye",
  hostname = "192.168.56.101", priv.key = "~/.ssh/id_rsa",
  priv.key.ppk = "./id_rsa.ppk")

RPC.RR(check = "ps axww|grep '[r]emoter::server'",
  kill = "kill -9 $(ps axww|grep '[r]emoter::server'|awk '{print $1}')",
  start = "nohup Rscript -e 'remoter::server()' > .rrlog 2>&1 < /dev/null &",
  preload = "source ~/work-my/00_set_devel_R; ",
  checkx = "ps axww|grep '[r]emoter::server\\|[x]vfb-run'",
  killx = "kill -9 $(ps axww|grep '[r]emoter::server\\|[x]vfb-run'|awk '{print $1}')",
  startx = "nohup xvfb-run Rscript -e 'remoter::server()' > .rrlog 2>&1 < /dev/null &")
```

Arguments

verbose, intern, wait, check.exec, use.shell.exec, pause
: RPC control for system, shell.exec, and so on. wait = FALSE allows asynchronous commands which need more test. e.g. continuously port forwarding without sending commands to background.

exec.type, args, pport, user, hostname, priv.key, priv.key.ppk
: RPC login information used by srpc(), ssh(), or plink().

check, kill, start, preload, checkx, killx, startx
: RPC remoter commands used by check_rr(), kill_rr(), or start_rr() and virtual X11 related versions checkx_rr(), killx_rr(), or startx_rr().. RPC pbdCS commands used by check_cs(), kill_cs(), or start_cs().

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>.

Programming with Big Data in R Website: http://r-pbd.org/

See Also

.pbd_env.

rpc_cs_example *Example Functions of RPC Related to pbdCS*

Description

Example functions of RPC related to pbdCS

Usage

```
check_cs(machine, cmd = .pbd_env$RPC.CS$check)

kill_cs(machine, cmd = .pbd_env$RPC.CS$kill)

start_cs(machine, cmd = .pbd_env$RPC.CS$start,
  preload = .pbd_env$RPC.CS$preload)
```

Arguments

machine	A machine configuration. See ?machine.
cmd	the main command to be executed on the server.
preload	further commands preloaded before the main command is executed.

Details

Using either ssh or plink to launch a pbdCS R cluster.

NO further input from stdin is expected. Outputs or errors may not be captured by R.

start_cs() starts a pbdCS R cluster on remote servers. Typical commands to launch a pbdCS R cluster is given in the example. The linux/unix commands are as the following:

- 1)nohup is for the non-stop call after disconnection.
- 2)mpiexec -np 4 is for launching 4 MPI instances.
- 3)Rscript -e 'pbdCS::pbdserver()' is for R to start the pbdCS R cluster within MPI in batch.
- 4)> .rrlog is to direct the stdout to a file .rrlog
- 5)2>&1 is to turn stderr to stdout which is the file .cslog.
- 6)< /dev/null is to get stdin from /dev/null which is nothing happen.
- 7)& is to put the batch command for the pbdCS R cluster in background.

check_cs() checks if there is a pbdCS R cluster on a remote server. Lunix/unix commands ps and grep are used.

kill_cs() kills the pbdCS R cluster if found. Lunix/unix commands ps, grep, awk, and kill are used.

Value

Mainly the message received at the command line will be printed, but may not be captured by R.

Examples

```
## Not run:
library(pbdRPC, quietly = TRUE)
# rpcopt_set(user = "snoweye", hostname = "192.168.56.101")
m <- machine(user = "snoweye", hostname = "192.168.56.101")

check_cs(m)    # pid 2857 (may differ)
kill_cs(m)     # all pbdCS pids are killed
check_cs(m)    # make sure no pbdCS R cluster is running

### use "; " to bypass multiple commands
preload <- "source ~/work-my/00_set_devel_R; "

### start a new pbdCS R cluster
start_cs(m, preload = preload)
check_cs(m)
kill_cs(m)

### Example: for module load on supercomputers
preload <- "module load r; "    # e.g. via module load r
start_cs(m, preload = preload)
check_cs(m)
kill_cs(m)

## End(Not run)
```

---

rpc_options *Set and Get Options for Remote Procedure Call*

Description

Set and get default options of login information for `srpc()`.

Usage

```
rpcopt_set(user = NULL, hostname = NULL, pport = NULL, exec.type = NULL,
  args = NULL, priv.key = NULL, priv.key.ppk = NULL)

rpcopt_get()
```

Arguments

user	user id for logging to the server.
hostname	the server ip or host name.
pport	ssh port opened on the server.
exec.type	either "ssh" or "plink".
args	arguments to the ssh or plink.exe.
priv.key	file name/path to the private key in OpenSSH format.
priv.key.ppk	file name/path to the private key in PuTTY format.

Details

rpcopt_set() is to alter default options of login information. The user defined options are set to .pbd_env$RPC.LI. rpcopt_get() is to get default options of login information from .pbd_env$RPC.LI.

Value

A list is returned.

See Also

rpc(), srpc().

Examples

```
## Not run:
library(pbdRPC, quietly = TRUE)
rpcopt_set(user = "snoweye", hostname = "192.168.56.101")

rpcopt_get()

## End(Not run)
```

---

rpc_pid *Search or kill possible pid's of Remote Procedure Call*

Description

Search or kill possible pid's that running remote procedure calls in background.

Usage

```
rpc_ps()

rpc_kill(pid)
```

Arguments

pid	a vector containing proccess id's in interger.

Details

rpc_ps() prints possible pid's that running remote procedure calls in background via one of ssh, plink, plink.exe, or cmd.exe.

rpc_kill() kill pid's.

Value

List or kill all possible pid's.

---

rpc_rr_example *Example Functions of RPC Related to Remote R Server*

Description

Example functions of RPC related to remote R server

Usage

```
check_rr(machine, cmd = .pbd_env$RPC.RR$check)

checkx_rr(machine, cmd = .pbd_env$RPC.RR$checkx)

kill_rr(machine, cmd = .pbd_env$RPC.RR$kill)

killx_rr(machine, cmd = .pbd_env$RPC.RR$killx)

start_rr(machine, cmd = .pbd_env$RPC.RR$start,
  preload = .pbd_env$RPC.RR$preload)

startx_rr(machine, cmd = .pbd_env$RPC.RR$startx,
  preload = .pbd_env$RPC.RR$preload)
```

Arguments

machine	A machine configuration. See ?machine.
cmd	the main command to be executed on the server.
preload	further commands preloaded before the main command is executed.

Details

Using either ssh or plink to launch a remote R server.

NO further input from stdin is expected. Outputs or errors may not be captured by R.

start_rr() starts a remote R server on a remote server. Typical commands to launch a remoter R server is given in the example. The linux/unix commands are as the following:

- 1)nohup is for the non-stop call after disconnection.
- 2)Rscript -e 'remoter::server()' is for R to start the remote R server in batch.
- 3)> .rrlog is to direct the stdout to a file .rrlog
- 4)2>&1 is to turn stderr to stdout which is the file .rrlog.
- 5)< /dev/null is to get stdin from /dev/null which is nothing happen.
- 6)& is to put the batch command for the remoter R server in background.

check_rr() checks if there is a remote R server on a remote server. Lunix/unix commands ps and grep are used.

kill_rr() kills remote R servers if found. Lunix/unix commands ps, grep, awk, and kill are used.

checkx_rr(), killx_rr(), and startx_rr() are functions with virtual X11 windows.

Value

Mainly the message received at the command line will be printed, but may not be captured by R.

Examples

```
## Not run:
library(pbdRPC, quietly = TRUE)
# rpcopt_set(user = "snoweye", hostname = "192.168.56.101")
m <- machine(user = "snoweye", hostname = "192.168.56.101")

check_rr(m)    # pid 2857 (may differ)
kill_rr(m)     # all remoter pids are killed
check_rr(m)    # make sure no remoter servers are running

### use "; " to bypass multiple commands
preload <- "source ~/work-my/00_set_devel_R; "

### start a new remoter server
start_rr(m, preload = preload)
check_rr(m)
kill_rr(m)

### Example: for module load on supercomputers
preload <- "module load r; "    # e.g. via module load r
start_rr(m, preload = preload)
check_rr(m)
kill_rr(m)
```

```
## End(Not run)
```

---

srpc *Secure Remote Procedure Call*

Description

Launch a command via ssh or plink on a (remote) server.

Usage

```
srpc(cmd = "whoami", exec.type = .pbd_env$RPC.LI$exec.type,
  args = .pbd_env$RPC.LI$args, pport = .pbd_env$RPC.LI$pport,
  user = .pbd_env$RPC.LI$user, hostname = .pbd_env$RPC.LI$hostname,
  priv.key = .pbd_env$RPC.LI$priv.key,
  priv.key.ppk = .pbd_env$RPC.LI$priv.key.ppk,
  intern = .pbd_env$RPC.CT$intern, wait = .pbd_env$RPC.CT$wait)
```

Arguments

`cmd`	the command to be executed on the server.
`exec.type`	either "ssh" or "plink" in character. Windows will force to use "plink".
`args`	further arguments to "ssh" or "plink" for connecting to the server in addition to port, user id, and host name.
`pport`	ssh port opened on the server.
`user`	user id for logging to the server.
`hostname`	the server ip or host name.
`priv.key, priv.key.ppk`	location of the private key for user authentication, the file will be checked first then `-i priv.key` will be added to `args` when the file exists. `priv.key.ppk` is only used when `plink` is called.
`intern, wait`	arguments passed to `system()` or `shell()` whereever they are applicable.

Details

Using either `ssh` or `plink` to launch a command on a (remote) server. Authentication is working currently for `ssh`.

NO further input from `stdin` is expected. Outputs or errors may not be captured by R.

Value

Mainly the message received from the command line of server may be returned but may not be captured by R.

For example, Windows with `plink` will not capture the return because currently the authentication is not working. A windows `bat` file is launched by `shell.exec()` in an additional `cmd.exe` window to avoid saving password inside R.

See Also

`start_rr()`, `check_rr()`, `kill_rr()`, `ssh()`, and `plink()`.

Examples

```
## Not run:
library(pbdRPC, quietly = TRUE)
rpcopt_set(user = "snoweye", hostname = "192.168.56.101")

### see start_rr(), check_rr(), and kill_rr() for more examples.
srpc()
srpc("ls")
srpc("ls ~/work-my")
srpc("cat ~/work-my/00_set_devel_R")

### see ssh(), plink(), and run_args() for lower level examples.

### Local port forwarding
srpc(args = "-N -T -L 55555:localhost:55555")
start_rr()

library(remoter)
client()    # equivalent to client(addr = "192.168.56.101")

## End(Not run)
```

---

ssh_plink *ssh and plink*

Description

Commend line tools including `ssh` for Linux, Mac OSX, Solaris, or `plink` for Windows.

Usage

```
ssh(args = "snoweye@192.168.56.101 whoami", intern = .pbd_env$RPC.CT$intern,
  wait = .pbd_env$RPC.CT$wait)

plink(args = "snoweye@192.168.56.101 whoami",
```

```
  use.shell.exec = .pbd_env$RPC.CT$use.shell.exec,
  pause = .pbd_env$RPC.CT$pause, intern = .pbd_env$RPC.CT$intern,
  wait = .pbd_env$RPC.CT$wait)

check_exec(exec)

run_args(exec = "ssh", args = "",
  use.shell.exec = .pbd_env$RPC.CT$use.shell.exec,
  pause = .pbd_env$RPC.CT$pause, intern = .pbd_env$RPC.CT$intern,
  wait = .pbd_env$RPC.CT$wait)
```

Arguments

args	All in text/characters that are passed to the command line.
intern, wait	arguments passed to system() or shell() whereever they are applicable.
use.shell.exec	if shell.exec() is used to execute the plink command in windows. No returns can be captured by R when this is TRUE as the default, because the authentication may not be generally available in most windows system. No easy yet secure way to by passing the password from R to external calls in shell.
pause	if pause when shell.exec is used in Windows.
exec	either ssh (i.e. /usr/bin/ssh) or a "full path" to plink.

Details

These functions only execute option/command on remote servers via secure client commands.

NO further input from stdin is expected. Outputs or errors may not be captured by R.

ssh() starts a ssh command.

plink() starts a plink command used by default for Windows.

check_exec() runs a quick check on the exec (either ssh or plink) for rpc.

run_args() runs a rpc via either ssh or plink.

Value

Mainly the message received at the command line will be returned.

Examples

```
## Not run:
library(pbdRPC, quietly = TRUE)
rpcopt_set(user = "snoweye", hostname = "192.168.56.101")

### Check an R session.
cmd <- "Rscript -e 'sessionInfo()'"

### For Linux, Mac OSX, Solaris.
rpc(cmd = cmd, exec.type = "ssh")
```

```
### For Windows.
rpc(cmd = cmd, exec.type = "plink")

### Manually
args <- "snoweye@192.168.56.101 Rscript -e 'sessionInfo()'"
ssh(args)      # Note ssh uses "-p" for server port.
plink(args)    # Note plink uses "-P" for server port.

### Manually launch a remoter server at background.
user.hostname <- "snoweye@192.168.56.101"
preload <- "source ./work-my/00_devel_R"
rr <- "nohup Rscript -e 'remoter::server()' > .rrlog 2>&1 < /dev/null &"
args <- paste(user.hostname, " \"", preload, "; ", rr, "\"", sep = "")
plink(args)

## End(Not run)
```

---

tunnel *tunnel*

Description

tunnel

Usage

```
tunnel(pport = .pbd_env$RPC.LI$pport, user = .pbd_env$RPC.LI$user,
  hostname = .pbd_env$RPC.LI$hostname, priv.key = .pbd_env$RPC.LI$priv.key,
  priv.key.ppk = .pbd_env$RPC.LI$priv.key.ppk, lport = 55555,
  rport = 55555, rhostname = "127.0.0.1")
```

Arguments

pport, user, priv.key, priv.key.ppk	See ?pbdRPC::srpc.
hostname	the server ip or host name.
lport, rport	local and remote ports.
rhostname	the local ip or host name.

Index