

Package ‘taskPR’

February 15, 2012

Version 0.34

Date 2009-05-19

Title Task-Parallel R Package

Author Nagiza F. Samatova <samatovan@ornl.gov>, David Bauer
<bauerda@ieee.org>, Srikanth Yoginath <yoginathsb@ornl.gov>

Maintainer David Bauer <bauerda@ieee.org>

Depends R (>= 2.4.0)

Description The Task-Parallel R (“task-pR”) system, repackaged as an R package

License GPL-2

URL <http://users.ece.gatech.edu/~gte810u/Parallel-R/>

OS_type unix

Repository CRAN

Date/Publication 2009-05-20 06:55:50

R topics documented:

PE	2
POBJ	3
StartPE	4
StartWorker	5
Index	7

 PE

Parallel Execute

Description

Attempts to execute the given expression through the parallel engine.

Usage

```
PE(x, global=FALSE)
```

Arguments

x	expression to execute in background/parallel
global	should this command be executed globally?

Details

The parallel engine takes the given expression and tries to execute it on a worker process (which may be on a remote system). It will be executed in the background and possibly in parallel with other PE calls.

Only a limited number of jobs can be run through the parallel engine at once. If that number is exceeded, then the PE call will block until some of the jobs have finished.

*The given expression **must** be of the form "out <- f(...)".*

When global is true, the command is executed globally. This does several things. First, the parallel engine will wait until all workers are free before started to execute the command. Second, the command will be executed on all worker processes simultaneously. Third, the output of the command will NOT be deleted from the worker processes' workspaces (as is usually done). This "global execute" functionality is designed for loading libraries or defining functions, and not for the parallel execution of common commands.

See Also

[StartPE](#) For enabling the parallel engine. [POBJ](#) For returning background jobs to the main process.

Examples

```
## Not run:
# If you have MPI running
StartPE(2)

x = matrix(rnorm(128 * 128), 128, 128)

PE( a <- svd(x) )
PE( b <- solve(x) )
PE( y <- b %*% a$u )
POBJ( y )
```

```
str(y)
StopPE()

## End(Not run)
```

POBJ

Parallel Object (Return a Parallel Objection to R's workspace)

Description

Blocks and waits for the specified variable to be returned by the parallel engine.

Usage

```
POBJ(x)
```

Arguments

x variable to return.

Details

Because the [PE](#) function executes jobs in the background, there must be a way to return jobs to the foreground - to R's workspace. That method is the POBJ function. When passed a variable (which doesn't have to exist in R's workspace, yet), the POBJ function waits until that variable is available and returned from the parallel engine. If variable to return is given as NULL, then the POBJ function waits for all variables to be returned. The POBJ function returns the **symbol** of the variable, not the variable itself.

See Also

[StartPE](#) For enabling the parallel engine. [PE](#) For executing jobs in parallel.

Examples

```
## Not run:
# If you have MPI running
StartPE(2)

x = matrix(rnorm(128 * 128), 128, 128)

PE( a <- svd(x) )
PE( b <- solve(x) )
PE( y <- b %*% a$u )
POBJ( y )
str(y)
StopPE()

## End(Not run)
```

 StartPE

Start/Stop Parallel Execution

Description

StartPE starts the parallel engine. If spawn is true, then the worker processes are spawned (using MPI_COMM_Spawn from MPI-2). StopPE stops the parallel engine. This call blocks until all jobs are finished.

Usage

```
StartPE(num = 2, port = 32000, verbose=0, spawn=TRUE)
StopPE()
```

Arguments

num	number of worker processes to use
port	the TCP port to use for communicating with workers
verbose	the verbose level: 0, 1, or 2 at the moment
spawn	should the worker processes be spawned?

Details

The parallel engine must be enabled before instructions can be executed in parallel. The engine can be stopped and restarted with a different number of worker processes, if desired. The parallel engine consists of num + 1 threads and num worker processes. The worker processes can either be spawned (done through an MPI call) or connected manually. If StartPE is run with spawn = FALSE, then it will block until num worker processes have connected.

See Also

[PE](#) For executing jobs in the background/parallel. [POBJ](#) For returning background/parallel jobs to the main process. [StartWorker](#) For manually starting worker processes.

Examples

```
## Not run:
# If you have MPI running
StartPE(2)

x = matrix(rnorm(128 * 128), 128, 128)

PE( a <- svd(x) )
PE( b <- solve(x) )
PE( y <- b %*% a$u )
POBJ( y )
str(y)
StopPE()
```

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

StartWorker	<i>Start Parallel-R Worker Process</i>
-------------	--

Description

Attempts to connect to the given host and establish itself as a worker process. This function is called automatically when worker processes are spawned by the main process.

Usage

```
StartWorker(host = "localhost", port=32000, retries=2, sleeptime=1, quiet=TRUE)
```

Arguments

host	name of the machine that the main/controller process is on
port	the (TCP/IP) port number to connect to
retries	the number of times to retry making the connection
sleeptime	how long (in seconds) to sleep between connection tries
quiet	should the worker process suppress most logging messages?

Details

The only time a user should call this function is when they started the parallel engine on the main process using the `spawn=FALSE` option to `StartPE`. In that case, the main process will block waiting for the worker processes to connect. The user must run the appropriate number of worker processes and have them call this function.

See Also

[StartPE](#) For enabling the parallel engine. [PE](#) For running parallel jobs. [POBJ](#) For returning background jobs to the main process.

Examples

```
## Not run:
# If you have MPI running
StartPE(2)

x = matrix(rnorm(128 * 128), 128, 128)

PE( a <- svd(x) )
PE( b <- solve(x) )
PE( y <- b %*% a$u )
POBJ( y )
str(y)
```

6

StartWorker

StopPE()

End(Not run)

Index

*Topic **programming**

PE, [2](#)

POBJ, [3](#)

StartPE, [4](#)

StartWorker, [5](#)

ParallelR (StartPE), [4](#)

PE, [2](#), [3–5](#)

POBJ, [2](#), [3](#), [4](#), [5](#)

StartPE, [2](#), [3](#), [4](#), [5](#)

StartWorker, [4](#), [5](#)

StopPE (StartPE), [4](#)

taskPR (StartPE), [4](#)