Package: snow (via r-universe)

September 17, 2024

Title Simple Network of Workstations

Version 0.4-4

Author Luke Tierney, A. J. Rossini, Na Li, H. Sevcikova

Description Support for simple parallel computing in R.

Maintainer Luke Tierney <luke-tierney@uiowa.edu>

Suggests rlecuyer

Enhances Rmpi

License GPL

Depends R (>= 2.13.1), utils

NeedsCompilation no

Date/Publication 2021-10-27 14:10:02 UTC

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

RemoteUrl https://github.com/cran/snow

RemoteRef HEAD

RemoteSha f518cddd4b9a94500b2ce6629ee7bd9d5ebab959

Contents

snow-cluster *Cluster-Level SNOW Functions*

Description

Functions for computing on a SNOW cluster.

Usage

```
clusterSplit(cl, seq)
clusterCall(cl, fun, ...)
clusterApply(cl, x, fun, ...)
clusterApplyLB(cl, x, fun, ...)
clusterEvalQ(cl, expr)
clusterExport(cl, list, envir = .GlobalEnv)
clusterMap(cl, fun, ..., MoreArgs = NULL, RECYCLE = TRUE)
```

Arguments

cl	cluster object
fun	function or character string naming a function
expr	expression to evaluate
seq	vector to split
list	character vector of variables to export
envir	environment from which t export variables
x	array
...	additional arguments to pass to standard function
MoreArgs	additional argument for fun
RECYCLE	logical; if true shorter arguments are recycled

Details

These are the basic functions for computing on a cluster. All evaluations on the worker nodes are done using tryCatch. Currently an error is signaled on the master if any one of the nodes produces an error. More sophisticated approaches will be considered in the future.

clusterCall calls a function fun with identical arguments ... on each node in the cluster cl and returns a list of the results.

clusterEvalQ evaluates a literal expression on each cluster node. It a cluster version of evalq, and is a convenience function defined in terms of clusterCall.

clusterApply calls fun on the first cluster node with arguments seq[[1]] and ..., on the second node with seq[[2]] and ..., and so on. If the length of seq is greater than the number of nodes in the cluster then cluster nodes are recycled. A list of the results is returned; the length of the result list will equal the length of seq.

clusterApplyLB is a load balancing version of clusterApply. if the length p of seq is greater than the number of cluster nodes n, then the first n jobs are placed in order on the n nodes. When the first job completes, the next job is placed on the available node; this continues until all jobs are complete. Using clusterApplyLB can result in better cluster utilization than using clusterApply. However, increased communication can reduce performance. Furthermore, the node that executes a particular job is nondeterministic, which can complicate ensuring reproducibility in simulations.

clusterMap is a multi-argument version of clusterApply, analogous to mapply. If RECYCLE is true shorter arguments are recycled; otherwise, the result length is the length of the shortest argument. Cluster nodes are recycled if the length of the result is greater than the number of nodes.

clusterExport assigns the values on the master of the variables named in list to variables of the same names in the global environments of each node. The environment on the master from which variables are exported defaults to the global environment.

clusterSplit splits seq into one consecutive piece for each cluster and returns the result as a list with length equal to the number of cluster nodes. Currently the pieces are chosen to be close to equal in length. Future releases may attempt to use relative performance information about nodes to choose split proportional to performance.

For more details see https://stat.uiowa.edu/~luke/R/cluster/cluster.html.

Examples

```
  ## Not run:
cl <- makeSOCKcluster(c("localhost","localhost"))

clusterApply(cl, 1:2, get("+"), 3)

clusterEvalQ(cl, library(boot))

x<-1
clusterExport(cl, "x")
clusterCall(cl, function(y) x + y, 2)


## End(Not run)
```

---

snow-parallel *Higher Level SNOW Functions*

Description

Parallel versions of apply and related functions.

Usage

```
parLapply(cl, x, fun, ...)
parSapply(cl, X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
parApply(cl, X, MARGIN, FUN, ...)
parRapply(cl, x, fun, ...)
parCapply(cl, x, fun, ...)
parMM(cl, A, B)
```

Arguments

cl	cluster object
fun	function or character string naming a function
X	array to be used
x	matrix to be used
FUN	function or character string naming a function
MARGIN	vector specifying the dimensions to use.
simplify	logical; see sapply
USE.NAMES	logical; see sapply
...	additional arguments to pass to standard function
A	matrix
B	matrix

Details

parLapply, parSapply, and parApply are parallel versions of lapply, sapply, and apply.

parRapply and parCapply are parallel row and column apply functions for a matrix x; they may be slightly more efficient than parApply.

parMM is a very simple(minded) parallel matrix multiply; it is intended as an illustration.

For more details see https://stat.uiowa.edu/~luke/R/cluster/cluster.html.

Examples

```
  ## Not run:
cl <- makeSOCKcluster(c("localhost","localhost"))
parSapply(cl, 1:20, get("+"), 3)

## End(Not run)
```

---

snow-rand *Uniform Random Number Generation in SNOW Clusters*

Description

Initialize independent uniform random number streams to be used in a SNOW cluster. It uses either the L'Ecuyer's random number generator (package rlecuyer required) or the SPRNG generator (package rsprng required).

Usage

```
clusterSetupRNG (cl, type = "RNGstream", ...)

clusterSetupRNGstream (cl, seed=rep(12345,6), ...)
clusterSetupSPRNG (cl, seed = round(2^32 * runif(1)),
                   prngkind = "default", para = 0, ...)
```

Arguments

cl	Cluster object.
type	type="RNGstream" (default) initializes the L'Ecuyer's RNG. type="SPRNG" initializes the SPRNG generator.
...	Arguments passed to the underlying function (see details bellow).
seed	Integer value (SPRNG) or a vector of six integer values (RNGstream) used as seed for the RNG.
prngkind	Character string naming generator type used with SPRNG.
para	Additional parameters for the generator.

Details

clusterSetupRNG calls (subject to its argument values) one of the other functions, passing arguments (cl, ...). If the "SPRNG" type is used, then the function clusterSetupSPRNG is called. If the "RNGstream" type is used, then the function clusterSetupRNGstream is called.

clusterSetupSPRNG loads the rsprng package and initializes separate streams on each node. For further details see the documentation of init.sprng. The generator on the master is not affected. NOTE: SPRNG is currently not supported.

clusterSetupRNGstream loads the rlecuyer package, creates one stream per node and distributes the stream states to the nodes.

For more details see https://stat.uiowa.edu/~luke/R/cluster/cluster.html.

Examples

```
  ## Not run:
clusterSetupSPRNG(cl)
clusterSetupSPRNG(cl, seed=1234)
clusterSetupRNG(cl, seed=rep(1,6))

## End(Not run)
```

---

snow-startstop *Starting and Stopping SNOW Clusters*

Description

Functions to start and stop a SNOW cluster and to set default cluster options.

Usage

```
makeCluster(spec, type = getClusterOption("type"), ...)
stopCluster(cl)

setDefaultClusterOptions(...)
```

```
makeSOCKcluster(names, ..., options = defaultClusterOptions)
makeMPIcluster(count, ..., options = defaultClusterOptions)
getMPIcluster()
```

Arguments

spec	cluster specification
count	number of nodes to create
names	character vector of node names
options	cluster options object
cl	cluster object
...	cluster option specifications
type	character; specifies cluster type.

Details

makeCluster starts a cluster of the specified or default type and returns a reference to the cluster. Supported cluster types are "SOCK", and "MPI". For "MPI" clusters the spec argument should be an integer specifying the number of worker nodes to create. For "SOCK" clusters spec should be a character vector naming the hosts on which worker nodes should be started; one node is started for each element in the vector. For "SOCK" clusters spec can also be an integer specifying the number of worker nodes to create on the local machine.

For SOCK clusters the spec can also be a list of machine specifications, each a list of named option values. Such a list must include a character value named host host specifying the name or address of the host to use. Any other option can be specified as well. For SOCK clusters this may be a more convenient alternative than inhomogeneous cluster startup procedure. The options rscript and snowlib are often useful; see the examples below.

stopCluster should be called to properly shut down the cluster before exiting R. If it is not called it may be necessary to use external means to ensure that all worker processes are shut down.

setDefaultClusterOptions can be used to specify alternate values for default cluster options. There are many options. The most useful ones are type and homogeneous. The default value of the type option is currently set to "MPI" if **Rmpi** is on the search path. Otherwise it is set to "MPI" if **Rmpi** is available, and to "SOCK" otherwise.

The homogeneous option should be set to FALSE to specify that the startup procedure for inhomogeneous clusters is to be used; this requires some additional configuration. The default setting is TRUE unless the environment variable R_SNOW_LIB is defined on the master host with a non-empty value.

The optionoutfile can be used to specify the file to which worker node output is to be directed. The default is /dev/null; during debugging of an installation it can be useful to set this to a proper file. On some systems setting outfile to "" or to /dev/tty will result in worker output being sent tothe terminal running the master process.

The functions makeSOCKcluster, and makeMPIcluster can be used to start a cluster of the corresponding type.

In MPI configurations where process spawning is not available and something like mpirun is used to start a master and a set of workers the corresponding cluster will have been pre-constructed and

can be obtained with `getMPIcluster`. It is also possible to obtain a reference to the running cluster using `makeCluster` or `makeMPIcluster`. In this case the `count` argument can be omitted; if it is supplied, it must equal the number of nodes in the cluster. This interface is still experimental and subject to change.

For SOCK clusters the option `manual = TRUE` forces a manual startup mode in which the master prints the command to be run manually to start a worker process. Together with setting the `outfile` option this can be useful for debugging cluster startup.

For more details see https://stat.uiowa.edu/~luke/R/cluster/cluster.html.

Examples

```
  ## Not run:
## Two workers run on the local machine as a SOCK cluster.
cl <- makeCluster(c("localhost","localhost"), type = "SOCK")
clusterApply(cl, 1:2, get("+"), 3)
stopCluster(cl)
## Another approach to running on the local machine as a SOCK cluster.
cl <- makeCluster(2, type = "SOCK")
clusterApply(cl, 1:2, get("+"), 3)
stopCluster(cl)
## A SOCK cluster with two workers on Mac OS X, two on Linux, and two
## on Windows:
macOptions <-
    list(host = "owasso",
         rscript = "/Library/Frameworks/R.framework/Resources/bin/Rscript",
         snowlib = "/Library/Frameworks/R.framework/Resources/library")
lnxOptions <-
    list(host = "itasca",
         rscript = "/usr/lib64/R/bin/Rscript",
  snowlib = "/home/luke/tmp/lib")
winOptions <-
    list(host="192.168.1.168",
         rscript="C:/Program Files/R/R-2.7.1/bin/Rscript.exe",
         snowlib="C:/Rlibs")
cl <- makeCluster(c(rep(list(macOptions), 2), rep(list(lnxOptions), 2),
                    rep(list(winOptions), 2)), type = "SOCK")
clusterApply(cl, 1:6, get("+"), 3)
stopCluster(cl)

## End(Not run)
```

`snow-timing` *Timing SNOW Clusters*

Description

Experimental functions to collect and display timing data for cluster computations.

Usage

```
snow.time(expr)
## S3 method for class 'snowTimingData'
print(x, ...)
## S3 method for class 'snowTimingData'
plot(x, xlab = "Elapsed Time", ylab = "Node",
                               title = "Cluster Usage", ...)
```

Arguments

expr	expression to evaluate
x	timing data object to plot or print
xlab	x axis label
ylab	y axis label
title	plot main title
...	additional arguments

Details

snow.time collects and returns and returns timing information for cluster usage in evaluating expr. The return value is an object of class snowTimingData; details of the return value are subject to change. The print method for snowTimingData objects shows the total elapsed time, the total communication time between master and worker nodes, and the compute time on each worker node. The plot, motivated by the display produced by xpvm, produces a Gantt chart of the computation, with green rectangles representing active computation, blue horizontal lines representing a worker waiting to return a result, and red lines representing master/worker communications.

Examples

```
  ## Not run:
cl <- makeCluster(2,type="SOCK")
x <- rnorm(1000000)
tm <- snow.time(clusterCall(cl, function(x) for (i in 1:100) sum(x), x))
print(tm)
plot(tm)
stopCluster(cl)

## End(Not run)
```

Index