Package: Rhpc (via r-universe)

October 27, 2026

Version 0.26.5

Date 2026-07-28

Title Apply-Style Dispatch for High-Performance Computing

Depends R (>= 4.6.0)

Imports parallel

SystemRequirements R built as a shared or static library, 'MPI' library.

Description Provides apply-style functions using the Message Passing Interface ('MPI') to improve the High-Performance Computing ('HPC') environment in R. The package supports long vectors and efficient handling of large datasets for MPI-based parallel computations.

Encoding UTF-8

License AGPL-3

URL https://github.com/e-nakama/Rhpc

BugReports https://github.com/e-nakama/Rhpc/issues

ByteCompile true

NeedsCompilation yes

VignetteBuilder utils

BuildVignettes true

Author Ei-ji Nakama [aut, cre], Junji NAKANO [aut]

Maintainer Ei-ji Nakama <nakama@ki.rim.or.jp>

Config/pak/sysreqs libopenmpi-dev

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

Date/Publication 2026-08-03 23:20:08 UTC

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

RemoteRef HEAD

RemoteSha 043612cc59ed88023098e5c2e767969e92cc27e2

Contents

Rhpc-package . . . 2
Rhpc_apply . . . 3
Rhpc_enquote . . . 4
Rhpc_EvalQ . . . 4
Rhpc_Export . . . 5
Rhpc_finalize . . . 6
Rhpc_getHandle . . . 7
Rhpc_initialize . . . 7
Rhpc_lapply . . . 8
Rhpc_lapplyLB . . . 9
Rhpc_numberOfWorker . . . 10
Rhpc_sapply . . . 10
Rhpc_sapplyLB . . . 11
Rhpc_serialize . . . 12
Rhpc_setupRNG . . . 13
Rhpc_splitList . . . 13
Rhpc_worker_call . . . 14
Rhpc_worker_noback . . . 15

Index **16**

`Rhpc-package` *Rhpc: MPI-based apply-style parallel computing for R*

Description

Provides MPI-aware apply-style functions and worker utilities for high-performance computing in R.

Details

The Rhpc package offers MPI-aware versions of common apply-style functions and helper utilities for distributed computation. It supports both static worker configuration and dynamic process generation.

Main functions:

`Rhpc_initialize` Initialize MPI environment

`Rhpc_getHandle` Get communicator handle to workers

`Rhpc_lapply`, `Rhpc_lapplyLB` Parallel lapply variants

`Rhpc_sapply`, `Rhpc_sapplyLB` Parallel sapply variants

`Rhpc_apply` Parallel apply for arrays/matrices

`Rhpc_worker_call` Call functions on workers

`Rhpc_worker_noback` Call functions without waiting for results on workers

`Rhpc_Export` Export variables to workers

`Rhpc_setupRNG` Setup random number generators

`Rhpc_finalize` Clean up MPI environment

MPI-related options are exposed through `getOption()`: `Rhpc.mpi.c.comm`, `Rhpc.mpi.f.comm`, `Rhpc.mpi.rank`, and `Rhpc.mpi.procs`.

See `https://github.com/e-nakama/Rhpc` for more information and examples.

`Rhpc_apply` *Parallel apply using MPI*

Description

MPI-based version of `apply` for arrays and matrices.

Usage

```
Rhpc_apply(cl = NULL, X, MARGIN, FUN, ...,
           usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                             getOption("Rhpc.usequote"), TRUE))
```

Arguments

`cl`	external pointer to MPI communicator
`X`	array or matrix
`MARGIN`	vector giving the subscripts which the function will be applied over. `1` indicates rows, `2` indicates columns, `c(1, 2)` indicates rows and columns
`FUN`	function to apply
`...`	additional arguments passed to `FUN`
`usequote`	logical; if `FALSE`, quote handling is skipped for performance improvement

Details

This function applies `FUN` to array margins specified by `MARGIN`, distributing the work across MPI workers.

If `X` has dimension subscripts not in `MARGIN`, those are used as the margin dimensions for applying the function.

Value

Array or matrix of results from applying `FUN` to the specified margins.

See Also

`Rhpc_lapply`, `Rhpc_sapply`

Rhpc_enquote *Quote arguments for remote evaluation*

Description

Quotes arguments for use in remote function calls on worker processes.

Usage

```
Rhpc_enquote(...)
```

Arguments

...	arguments to be quoted

Details

This function quotes its arguments to prevent premature evaluation in the master process. The quoted expressions are then evaluated on the worker processes where they have access to the worker's data and environment.

This is primarily an internal utility; users typically do not need to call it directly.

Value

A quoted representation of the arguments suitable for remote evaluation.

See Also

Rhpc_worker_call, Rhpc_EvalQ

Rhpc_EvalQ *Evaluate an expression on worker processes*

Description

Evaluates an R expression on all worker processes.

Usage

```
Rhpc_EvalQ(cl, expr,
           usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                             getOption("Rhpc.usequote"), TRUE),
           envir = .GlobalEnv)
```

Arguments

cl	external pointer to MPI communicator
expr	expression to evaluate on workers
usequote	logical; if FALSE, quote handling is skipped for performance improvement
envir	environment in which to evaluate the expression on workers (default: global environment)

Details

This function evaluates the given expression on all worker processes and returns the results. It is useful for initialization tasks or querying worker state.

Value

Results of evaluating the expression on all workers.

See Also

Rhpc_worker_call, Rhpc_Export

Rhpc_Export *Export variables to worker processes*

Description

Sends R objects from the master process to all worker processes.

Usage

```
Rhpc_Export(cl, variableNames,
            usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                              getOption("Rhpc.usequote"), TRUE),
            pos = 1,
            envir = as.environment(pos))
```

Arguments

cl	external pointer to MPI communicator
variableNames	character vector of object names to export
usequote	logical; if FALSE, quote handling is skipped for performance improvement
pos	environment position from which to retrieve objects (default: global environment)
envir	environment from which to retrieve objects (default: environment at position pos)

Details

This function retrieves objects from the master's environment and sends them to all worker processes, making them available in the workers' global environment.

Objects are sent via collective communication.

Value

Returns invisibly. Objects are assigned in worker environments.

See Also

Rhpc_worker_call, Rhpc_EvalQ

Rhpc_finalize *Finalize MPI environment*

Description

Finalizes the MPI environment and terminates all worker processes.

Usage

```
Rhpc_finalize()
```

Details

This function should be called at the end of an Rhpc session to properly shut down the MPI environment and clean up all resources.

Value

Returns invisibly. Resources are cleaned up.

See Also

Rhpc_initialize, Rhpc_getHandle

Rhpc_getHandle *Get MPI communicator handle*

Description

Obtains a handle to the MPI communicator for worker processes.

Usage

```
Rhpc_getHandle(procs = NA)
```

Arguments

procs	number of processes to create via dynamic process generation. If NA (default), uses static configuration.

Details

This function returns an external pointer to the MPI communicator, which is used in all subsequent Rhpc function calls. It also initializes the random number generation stream for workers.

If procs is specified, dynamic process generation is used (MPI_Comm_spawn). If procs is NA, static worker configuration is assumed.

Value

An external pointer to the MPI communicator handle, which can be passed to other Rhpc functions.

See Also

Rhpc_initialize, Rhpc_numberOfWorker, Rhpc_finalize

Rhpc_initialize *Initialize MPI environment*

Description

Initializes the MPI environment for Rhpc.

Usage

```
Rhpc_initialize()
```

Details

This function must be called before any other Rhpc functions to initialize the MPI communicator. It sets up the MPI environment and prepares the system for distributed computation.

Value

Returns invisibly. MPI initialization results are stored internally.

See Also

Rhpc_finalize, Rhpc_getHandle

Rhpc_lapply *Parallel lapply using MPI*

Description

MPI-based version of lapply that distributes work across worker processes sequentially.

Usage

```
Rhpc_lapply(cl, X, FUN, ...,
            usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                              getOption("Rhpc.usequote"), TRUE))
```

Arguments

cl	external pointer to MPI communicator
X	vector or list to be processed
FUN	function to apply to each element of X
...	additional arguments passed to FUN
usequote	logical; if FALSE, quote handling is skipped for performance improvement

Details

This function distributes elements of X sequentially to worker processes. The input X is divided among workers and each worker processes its portion in order.

Results are collected and returned in a list maintaining the original order.

Value

A list with results from applying FUN to each element of X.

See Also

Rhpc_lapplyLB, Rhpc_sapply, Rhpc_apply

`Rhpc_lapplyLB` *Parallel lapply with load balancing*

Description

MPI-based version of `lapply` with load balancing for better performance on heterogeneous systems.

Usage

```
Rhpc_lapplyLB(cl, X, FUN, ...,
              usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                                getOption("Rhpc.usequote"), TRUE))
```

Arguments

`cl`	external pointer to MPI communicator
`X`	vector or list to be processed
`FUN`	function to apply to each element of `X`
`...`	additional arguments passed to `FUN`
`usequote`	logical; if `FALSE`, quote handling is skipped for performance improvement

Details

This function distributes work to workers using a load-balancing approach. Workers receive tasks dynamically as they become available, which can provide better performance when task execution times vary significantly.

Results are collected and returned in a list maintaining the original order.

Value

A list with results from applying `FUN` to each element of `X`.

See Also

`Rhpc_lapply`, `Rhpc_sapplyLB`, `Rhpc_apply`

Rhpc_numberOfWorker *Get number of worker processes*

Description

Returns the number of active worker processes.

Usage

```
Rhpc_numberOfWorker(cl)
```

Arguments

cl	external pointer to MPI communicator (obtained from Rhpc_getHandle)

Value

An integer giving the number of worker processes.

See Also

Rhpc_getHandle

Rhpc_sapply *Parallel sapply using MPI*

Description

MPI-based version of sapply that distributes work across worker processes.

Usage

```
Rhpc_sapply(cl = NULL, X, FUN, ...,
            usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                              getOption("Rhpc.usequote"), TRUE),
            simplify = TRUE,
            USE.NAMES = TRUE)
```

Arguments

cl	external pointer to MPI communicator
X	vector or list to be processed
FUN	function to apply to each element of X
...	additional arguments passed to FUN
usequote	logical; if FALSE, quote handling is skipped for performance improvement
simplify	logical or character string; if possible, simplify the result to a vector or matrix
USE.NAMES	logical; if TRUE and X is character, use X as names for the result

Details

This function combines the functionality of Rhpc_lapply with automatic simplification of results. When possible, results are simplified to a vector or matrix form rather than remaining as a list.

Value

A simplified version of the result (vector, matrix, or array) or list depending on simplify parameter.

See Also

Rhpc_sapplyLB, Rhpc_lapply, Rhpc_apply

Rhpc_sapplyLB *Parallel sapply with load balancing*

Description

MPI-based version of sapply with load balancing for better performance.

Usage

```
Rhpc_sapplyLB(cl = NULL, X, FUN, ...,
              usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                                getOption("Rhpc.usequote"), TRUE),
              simplify = TRUE,
              USE.NAMES = TRUE)
```

Arguments

cl	external pointer to MPI communicator
X	vector or list to be processed
FUN	function to apply to each element of X
...	additional arguments passed to FUN
usequote	logical; if FALSE, quote handling is skipped for performance improvement
simplify	logical or character string; if possible, simplify the result
USE.NAMES	logical; if TRUE and X is character, use X as names

Details

This function combines load-balanced distribution with result simplification. Workers receive tasks dynamically as they become available, and results are simplified when possible.

Value

A simplified version of the result (vector, matrix, array, or list) depending on simplify.

See Also

Rhpc_sapply, Rhpc_lapplyLB, Rhpc_apply

Rhpc_serialize *Serialize and unserialize R objects*

Description

Serialize R objects for transmission or storage, and unserialize them back.

Usage

```
Rhpc_serialize(obj)
Rhpc_serialize_onlysize(obj)
Rhpc_serialize_norealloc(obj)
Rhpc_unserialize(obj)
```

Arguments

obj	R object to serialize, or serialized object to unserialize

Details

These functions provide low-level serialization utilities for handling R objects:

Rhpc_serialize Fully serializes an object to raw bytes, allocating necessary memory

Rhpc_serialize_onlysize Returns only the size needed for serialization without actual serialization

Rhpc_serialize_norealloc Serializes without reallocating memory (for efficiency)

Rhpc_unserialize Reconstructs an R object from its serialized form

These functions are primarily intended for internal Rhpc use but may be useful for debugging or advanced users.

Value

For Rhpc_serialize: raw vector containing serialized data

For Rhpc_serialize_onlysize: integer giving the size in bytes

For Rhpc_serialize_norealloc: raw vector containing serialized data

For Rhpc_unserialize: original R object

Examples

```
## Serialization example (executable without MPI)
data_to_serialize <- list(a = 1:5, b = c("x", "y", "z"))
serialized <- Rhpc_serialize(data_to_serialize)
deserialized <- Rhpc_unserialize(serialized)
stopifnot(identical(data_to_serialize, deserialized))
```

`Rhpc_setupRNG` *Setup random number generators for worker processes*

Description

Initializes independent random number streams for each worker process using L'Ecuyer-CMRG algorithm.

Usage

```
Rhpc_setupRNG(cl, iseed = NULL)
```

Arguments

`cl`	external pointer to MPI communicator
`iseed`	optional integer seed for reproducibility. If `NULL`, uses current .Random.seed

Details

This function sets up independent random number streams for each worker to ensure they produce different but reproducible random sequences. It uses the L'Ecuyer-CMRG (Combined Multiple Recursive Generator) algorithm.

The master process's original .Random.seed is saved to `options("Rhpc.oldseed")` but is NOT automatically restored due to CRAN policy. Users should manually restore if needed: `assign(".Random.seed", getOption("Rhpc.oldseed"), envir = globalenv())`

Value

Returns invisibly the original .Random.seed that was saved.

See Also

`Rhpc_getHandle`, `Rhpc_lapply`

`Rhpc_splitList` *Split a list into chunks*

Description

Divides a list or vector into approximately equal-sized chunks for distribution to workers.

Usage

```
Rhpc_splitList(var, num)
```

Arguments

var	vector or list to be split
num	number of chunks to create (typically equal to the number of workers)

Details

This function divides var into num roughly equal-sized chunks. It is useful for manually distributing work to workers when fine-grained control is needed.

This is primarily an internal utility; the apply-style functions handle splitting automatically.

Value

A list of length num, each element containing a portion of var.

See Also

Rhpc_lapply, Rhpc_numberOfWorker

Rhpc_worker_call *Call a function on worker processes*

Description

Executes a function on all worker processes and collects the results.

Usage

```
Rhpc_worker_call(cl, FUN, ...,
                 usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                                   getOption("Rhpc.usequote"), TRUE))
```

Arguments

cl	external pointer to MPI communicator
FUN	function or string naming the function to be called on workers
...	arguments to pass to FUN
usequote	logical; if FALSE, quote handling is skipped for performance improvement

Details

This function broadcasts a function call to all worker processes and waits for results. Results from all workers are collected and returned.

The function or its name is sent to workers via collective communication only on the first call; subsequent calls with the same function reuse the cached version.

Value

Results collected from all worker processes.

See Also

Rhpc_worker_noback, Rhpc_lapply, Rhpc_EvalQ

Rhpc_worker_noback *Call a function on worker processes without collecting results*

Description

Executes a function on all worker processes without waiting for or collecting results.

Usage

```
Rhpc_worker_noback(cl, FUN, ...,
                   usequote = ifelse(is.logical(getOption("Rhpc.usequote")),
                                     getOption("Rhpc.usequote"), TRUE))
```

Arguments

cl	external pointer to MPI communicator
FUN	function or string naming the function to be called on workers
...	arguments to pass to FUN
usequote	logical; if FALSE, quote handling is skipped for performance improvement

Details

This function sends a function call to all worker processes but does not wait for results. It is useful for operations like setting options or initializing worker state where return values are not needed.

Value

Returns invisibly. No results are collected from workers.

See Also

Rhpc_worker_call, Rhpc_Export

Index

∗ **package**
Rhpc-package, 2
∗ **utilities**
Rhpc_apply, 3
Rhpc_enquote, 4
Rhpc_EvalQ, 4
Rhpc_Export, 5
Rhpc_finalize, 6
Rhpc_getHandle, 7
Rhpc_initialize, 7
Rhpc_lapply, 8
Rhpc_lapplyLB, 9
Rhpc_numberOfWorker, 10
Rhpc_sapply, 10
Rhpc_sapplyLB, 11
Rhpc_serialize, 12
Rhpc_setupRNG, 13
Rhpc_splitList, 13
Rhpc_worker_call, 14
Rhpc_worker_noback, 15

Rhpc-package, 2
Rhpc_apply, *2*, 3, *8, 9*, *11, 12*
Rhpc_enquote, 4
Rhpc_EvalQ, 4, *4*, *6*, *15*
Rhpc_Export, *2*, 5, *5*, *15*
Rhpc_finalize, *3*, 6, *7, 8*
Rhpc_getHandle, *2*, *6*, 7, *8*, *10*, *13*
Rhpc_gethandle *(*Rhpc-package*)*, 2
Rhpc_initialize, *2*, *6*, 7, *7*
Rhpc_lapply, *2, 3*, 8, *9*, *11*, *13–15*
Rhpc_lapplyLB, *2*, *8*, 9, *12*
Rhpc_mpi_finalize *(*Rhpc-package*)*, 2
Rhpc_mpi_initialize *(*Rhpc-package*)*, 2
Rhpc_mpi_lapply_LB *(*Rhpc-package*)*, 2
Rhpc_mpi_lapply_seq *(*Rhpc-package*)*, 2
Rhpc_mpi_worker_call *(*Rhpc-package*)*, 2
Rhpc_number_of_worker *(*Rhpc-package*)*, 2
Rhpc_numberOfWorker, *7*, 10, *14*
Rhpc_sapply, *2, 3*, *8*, 10, *12*
Rhpc_sapplyLB, *2*, *9*, 11, *11*
Rhpc_serialize, 12
Rhpc_serialize_norealloc *(*Rhpc_serialize*)*, 12
Rhpc_serialize_onlysize *(*Rhpc_serialize*)*, 12
Rhpc_setupRNG, *3*, 13
Rhpc_splitList, 13
Rhpc_unserialize *(*Rhpc_serialize*)*, 12
Rhpc_worker_call, *2*, *4–6*, 14, *15*
Rhpc_worker_noback, *2*, 15, *15*