Package 'largescaleobjects'

November 26, 2023

Type Package

Title Provides a Distributed Framework for Statistical Modelling

Version 1.0

Date 2022-02-15

Author Jason Cairns [aut, cre]

Maintainer Jason Cairns <jcai849@aucklanduni.ac.nz>

Description Allows for transparent interaction with distributed objects, including the import of very large distributed data files, as well as the rapid development of parallel algorithms over the distributed data for fast run time.

Imports chunknet, utils

Depends iotools

Enhances dplyr

License MIT

URL https://github.com/jcai849/largeScaleR

BugReports https://github.com/jcai849/largeScaleR/issues

OS_type unix

NeedsCompilation no

R topics documented:

`do.dcall` *Execute a call over a distributed object.*

Description

Abstract function application for a distributed object.

Usage

```
do.dcall(what, args, balance = FALSE)
d(what)
```

Arguments

`what`	Function to apply.
`args`	List of arguments, including distributed objects.
`balance`	Logical, to balance results over cluster.

Details

`do.dcall` asynchronously requests the function application over a distributed object, with the remote nodes taking care of argument location and transfers. `d` wraps `do.dcall`, returning a distributed function.

Value

`d`	Returns a distributed version of the input function. Function application returns a DistributedObject.
`do.dcall`	DistributedObject.

See Also

`emerge`

Examples

```
d.model.matrix <- d(model.matrix)
d.model.matrix(object=~ a + b, dd)

##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (what, args, balance = FALSE)
{
    if (inherits(args, "DistributedObject"))
        stop("Requires list for argument, not distributed object")
    prealigned_args <- lapply(args, prealign)
    aligned_args <- do.call(mapply, c(list, prealigned_args,
        SIMPLIFY = FALSE, USE.NAMES = FALSE))
    chunks <- chunknet::do.ccall(rep(list(what), length(aligned_args)),
```

```
        aligned_args, balance = balance)
  DistributedObject(chunks)
}
```

`emerge` *Pull and reassemble a distributed object locally*

Description

Pull and reassemble a distributed object locally

Usage

```
emerge(x, combiner = TRUE, ...)
emerge.default(x, combiner = TRUE, ...)
emerge.DistributedObject(x, combiner = TRUE, ...)
```

Arguments

`x`	Object to emerge; S3 method dispatches on.
`combiner`	Logical; run a combiner on the underlying chunks for the distributed object? Optionally pass a combiner function to run on the chunks.
`...`	Further arguments passed to methods.

Details

Synchronously pull and recreate a distributed object. If no combiner function provided, and `combiner=TRUE`, combine based on the `combine` method defined for the chunk classes.

Value

The value of the emerged object, or a list of chunks if `combine=FALSE`.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x, combiner = TRUE, ...)
UseMethod("emerge", x)
```

`init_locator` *Remote initialisation of node types*

Description

Remotely initialise a worker or locator node.

Usage

```
init_locator(host, port)
```

Arguments

`host`	Scalar character naming the host to initialise the node on.
`port`	Scalar integer declaring the port to bind to. Can be 0L if unknown.

Details

Initialises a worker node, or a locator node.

Value

None

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (host, port)
{
    chunknet::LOCATOR(host, port)
    remote_sys(host, "chunknet::locator_node", list(host, port))
  }
```

Index