Package 'chunknet'

November 26, 2023

Type Package

Title Create a fast, scalable network of peer-to-peer R processes to distribute computation with larger than memory data.

Version 1.1

Date 2022-10-14

Author Jason Cairns

Maintainer <jason.cairns@auckland.ac.nz>

Description Create a fast, scalable network of peer-to-peer R processes to distribute computation with larger than memory data.

Imports orcv, uuid

License MIT

NeedsCompilation no

R topics documented:

`async_pull` *Asynchronous Data Pull*

Description

Send a request for data as referenced by a list of `href`'s, with the response to be returned asynchronously.

Usage

```
async_pull(hrefs, ...)
```

Arguments

`hrefs`	Character vector of href references for data objects
`...`	Kept for future methods

Details

Locations for the data referenced by the `href` vector are attained, with the containing nodes requested to send the data when available. Data can be obtained from the underlying message queue after it is posted.

Value

Closed file descriptors of the locations sought.

See Also

`pull`, `push`

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 (hrefs, ...)
{
    if (!length(hrefs))
        return()
    stopifnot(is.character(hrefs))
    locations <- get_locations(hrefs)
    hrefs_at_locs <- split(hrefs, as.factor(locations))
    mapply(function(location, hrefs) orcv::send(location, paste0("GET /async/data/",
        paste(hrefs, collapse = ","))), unique(locations), hrefs_at_locs)
  }
```

`do.ccall` *Chunk Call*

Description

A chunk function applicator

Usage

```
do.ccall(procedures, argument_lists, target, post_locs = TRUE, balance = FALSE)
```

Arguments

procedures	list of functions, or character vector naming functions
argument_lists	list containing argument lists corresponding to each procedure
target	Optional target ChunkReference
post_locs	Logical, send location of created chunk references to locator server or not.
balance	Logical of whether to balance results along the cluster, or a Balance function to apply the balancing.

Details

The principal means of performing remote operations on chunks. Returns immediately, without waiting for results of chunk operations.

Value

List of ChunkReferences.

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 (procedures, argument_lists, target, post_locs = TRUE,
    balance = FALSE)
{
    locations <- determine_locations(argument_lists, target,
        balance)
    arguments_by_loc <- disperse_arguments(argument_lists, locations)
    comps_by_loc <- send_computations(procedures, arguments_by_loc,
        locations)
    comprefs <- unsplit(comps_by_loc, as.factor(locations))
    output_hrefs <- sapply(comprefs, output_href)
    if (post_locs)
        post_locations(output_hrefs, locations)
    mapply(ChunkReference, output_hrefs, locations, comprefs,
        SIMPLIFY = FALSE)
  }
```

---

pull *Pull data to current node*

Description

Synchronous pull of data to current node.

Usage

```
pull(x, ...)
pull.ChunkReference(x, ...)
pull.character(x, ...)
pull.list(x, ...)
```

Arguments

x	Object to dispatch on.
...	Further arguments sent to methods.

Details

Pull data from external source locally. A vector of character hrefs yield a list of the referenced data that is then unsplit, with whatever method defined for unsplit on the data then determining the resultant return value. A list of ChunkReferences returns their resultant values, as does a singular ChunkReference.

Value

Value of the unsplit data sources.

See Also

async_pull, push

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, ...)
UseMethod("pull", x)
```

---

push *Push data to other node*

Description

Push data to another node's message queue.

Usage

```
push(x, locations, ...)
push.default(x, locations, post_locs=TRUE,...)
push.list(x, locations, ...)
push.Chunk(x, locations, ...)
```

Arguments

x	Argument to dispatch on. What item or contained item to send.
locations	Optional set of locations to send x to.
post_locs	Logical, to send locations of objects to the locator node or not.
...	Further arguments passed on to methods

Value

List of ChunkReferences referring to the sent object(s).

See Also

pull

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, locations, ...)
UseMethod("push", x)
```

---

worker_node *Cluster node initialisers*

Description

Initiate one of either worker or locator node.

Usage

```
worker_node(address = NULL, port = 0L, ..., verbose = FALSE)
locator_node(address = NULL, port = 0L, ..., verbose = FALSE)
```

Arguments

address	character address for the communication node to be reachable by. Leave NULL for localhost.
port	Integer port to bind to.
...	Arguments to pass on to methods
verbose	Logical, start verbose or not.

Value

None; loops.

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 (address = NULL, port = 0L, ..., verbose = FALSE)
{
    options(chunknetVerbose = verbose)
    orcv::start(address, port, threads = 1L)
    init_function(...)
    repeat {
        event <- orcv::receive(keep_conn = TRUE)
        handle(event)
        log("...DONE")
    }
  }
```

Index